Diversity-promoting and Large-scale Machine Learning for Healthcare

Pengtao Xie

CMU-ML-18-106
Aug 2018

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Eric P. Xing, Chair
Ruslan Salakhutdinov
Pradeep Ravikumar
Ryan Adams (Princeton & Google Brain)
David Sontag (MIT)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Copyright © 2018 Pengtao Xie

This research was sponsored by: Office of Naval Research award number N000140910758; Air Force Office of Scientific Research award FA95501010247; National Science Foundation awards IIS1115313, IIS1111142, IIS1218282 and IIS1447676; Defense Advanced Research Projects Agency award FA872105C0003; Department of the Air Force award FA870215D0002; National Institutes of Health award P30DA035778; Pennsylvania Department of Health’s Big Data for Better Health (BD4BH) award; and a grant from the University of Pittsburgh Medical Center.
Keywords: Diversity-promoting Learning, Large-scale Distributed Learning, Machine Learning for Healthcare, Regularization, Bayesian Priors, Generalization Error Analysis, System and Algorithm Co-design
Dedicated to my parents
for their endless love, support, and encouragement
Abstract

In healthcare, a tsunami of medical data has emerged, including electronic health records, images, literature, etc. These data are heterogeneous and noisy, which renders clinical decision-makings time-consuming, error-prone, and suboptimal. In this thesis, we develop machine learning (ML) models and systems for distilling high-value patterns from unstructured clinical data and making informed and real-time medical predictions and recommendations, to aid physicians in improving the efficiency of workflow and the quality of patient care. When developing these models, we encounter several challenges: (1) How to better capture infrequent clinical patterns, such as rare subtypes of diseases; (2) How to make the models generalize well on unseen patients? (3) How to promote the interpretability of the decisions? (4) How to improve the timeliness of decision-making without sacrificing its quality? (5) How to efficiently discover massive clinical patterns from large-scale data?

To address challenges (1-4), we systematically study diversity-promoting learning, which encourages the components in ML models (1) to diversely spread out to give infrequent patterns a broader coverage, (2) to be imposed with structured constraints for better generalization performance, (3) to be mutually complementary for more compact representation of information, and (4) to be less redundant for better interpretability. The study is performed in both frequentist statistics and Bayesian statistics. In the former, we develop diversity-promoting regularizers that are empirically effective, theoretically analyzable, and computationally efficient, and propose a rich set of optimization algorithms to solve the regularized problems. In the latter, we propose Bayesian priors that can effectively entail an inductive bias of “diversity” among a finite or infinite number of components and develop efficient posterior inference algorithms. We provide theoretical analysis on why promoting diversity can better capture infrequent betters and improve generalization. The developed regularizers and priors are demonstrated to be effective in a wide range of ML models.

To address challenge (5), we study large-scale learning. Specifically, we design efficient distributed ML systems by exploiting a system-algorithm co-design approach. Inspired by a sufficient factor property of many ML models, we design a peer-to-peer system – Orpheus – that significantly reduces communication and fault tolerance costs. We also provide theoretical analysis showing that algorithms executed on Orpheus are guaranteed to converge. The efficiency of our system is demonstrated in several large-scale applications.

We apply the proposed diversity-promoting learning (DPL) techniques and the distributed ML system to solve healthcare problems. In a similar-patient retrieval application, DPL shows great effectiveness in improving retrieval performance on infrequent diseases, enabling fast and accurate retrieval, and reducing overfitting. In a medical-topic discovery task, our Orpheus system is able to extract tens of thousands of topics from millions of documents in a few hours. Besides these two applications, we also design effective ML models for hierarchical multi-label tagging of medical images and automated ICD coding.
Acknowledgments

First and foremost, I would like to thank my PhD advisor Professor Eric Xing for his great advice, encouragement, support, and help throughout my journey at CMU. Eric’s exceptional insights and broad vision have played an instrumental role in helping me to pick impactful problems. He has consistently put in his time and energy in guiding me through every challenge in research, from designing models to diagnosing errors in experiments, from writing papers to giving presentations. I am also grateful to him for allowing me the freedom to do research in a diverse range of fields that I am passionate about. His enthusiasm for research, his passion for life, and his work ethic have been a great source of inspiration and will continue to shape me in the future.

I would like to express my deepest gratitude to my other thesis committee members: Professors Ruslan Salakhutdinov, Pradeep Ravikumar, Ryan Adams, and David Sontag, for their invaluable feedback that has helped to improve many parts of this thesis. The insightful discussions with Russ and Pradeep inspired several ideas in this thesis. Ryan’s early work on priors for diversity in generative latent variable models is the motivation for systematically studying diversity-promoting learning in this thesis. David’s sharp insights and invaluable comments steered me in the right direction in the study of ML for healthcare.

I have been very fortunate to collaborate with many brilliant friends: Qirong Ho, Ruslan Salakhutdinov, Wei Wu, Barnabas Poczos, Aarti Singh, James Zou, Yao-liang Yu, Jun Zhu, Jianxin Li, Jin Kyu Kim, Hongbao Zhang, Baoyu Jing, Devendra Sachan, Yuntian Deng, Yi Zhou, Haoran Shi, Yichen Zhu, Hao Zhang, Shizhen Xu, Haoyi Zhou, Shuai Zhang, Yuan Xie, Yulong Pei, among many others. It has always been a great pleasure and a valuable learning experience to brainstorm ideas, design models and algorithms, debug code, and write papers with these talented people and I am proud of the work we have done together over the last 5 years.

I am also very thankful to the Sailing Lab and Machine Learning Department for fostering a friendly and wonderful environment for research and life. I would like to thank Diane Stidle, Michelle Martin, Mallory Deptola and the entire MLD staff for their warm help and kind support. I deeply appreciate many of CMU faculties, for their wonderful courses and talks.

Lastly and most importantly, I am extremely grateful to my family. I could not have done this without the constant encouragement, unwavering support, and endless love from my parents.
## Contents

1 Introduction .................................................. 1
  1.1 Thesis Introduction and Scope .............................. 1
  1.2 Contributions .............................................. 5
    1.2.1 Diversity-promoting Learning ......................... 6
    1.2.2 Large-scale Distributed Learning ..................... 8
    1.2.3 ML for Healthcare ..................................... 9
  1.3 Related Works ............................................... 10
    1.3.1 Diversity-promoting Learning ......................... 10
    1.3.2 Distributed Learning ................................ 11
    1.3.3 ML for Healthcare ..................................... 12

2 Diversity-promoting Learning I – Regularization .......... 14
  2.1 Uncorrelation and Evenness: A Diversity-promoting Regularizer ............. 14
    2.1.1 Uniform Eigenvalue Regularizer ....................... 16
    2.1.2 Case Studies .......................................... 19
    2.1.3 A Projected Gradient Decent Algorithm ............... 20
    2.1.4 Evaluation ............................................ 21
  2.2 Convex Diversity-promoting Regularizers .................. 26
    2.2.1 Nonconvex Bregman Matrix Divergence Regularizers .......... 27
    2.2.2 Convex Bregman Matrix Divergence Regularizers ........... 28
    2.2.3 A Proximal Gradient Descent Algorithm ................. 31
    2.2.4 Evaluation ............................................ 32
  2.3 Angular Constraints for Improving Generalization Performance .......... 36
    2.3.1 Angular Constraints ................................... 37
    2.3.2 An ADMM-based Algorithm ............................. 38
    2.3.3 Evaluation ............................................ 41
    2.3.4 Appendix: Proofs and Details of Algorithms ............. 44
  2.4 Diversity in the RKHS: Orthogonality-promoting Regularization of Kernel Methods .................................................. 52
    2.4.1 Bregman Matrix Divergence Regularized Kernel Methods ........ 53
    2.4.2 A Functional Gradient Descent Algorithm ............... 54
    2.4.3 Evaluation ............................................ 57
    2.4.4 Appendix: Details of Algorithms ....................... 60
  2.5 Diversity and Sparsity: Nonoverlapness-promoting Regularization .......... 62
2.5.1 Nonoverlap-promoting Regularization ............................................................................ 63
2.5.2 A Coordinate Descent Algorithm ................................................................................ 65
2.5.3 Evaluation .................................................................................................................. 66
2.5.4 Appendix: Details of Algorithms .................................................................................. 70

3 Diversity-promoting Learning II – Bayesian Inference ...................................................... 84
3.1 Diversity-promoting Learning of Bayesian Parametric Models ..................................... 84
3.1.1 Mutual Angular Process .............................................................................................. 85
3.1.2 Approximate Inference Algorithms ........................................................................... 87
3.1.3 Diversity-promoting Posterior Regularization ............................................................ 89
3.1.4 Case Study: Bayesian Mixture of Experts Model ....................................................... 90
3.1.5 Evaluation .................................................................................................................. 91
3.1.6 Appendix: Details of Algorithms .................................................................................. 94
3.2 Diversity-promoting Learning of Bayesian Nonparametric Models .............................. 100
3.2.1 Infinite Mutual Angular Process .................................................................................. 101
3.2.2 Case Study: Infinite Latent Feature Model ................................................................. 102
3.2.3 A Sampling-based Inference Algorithm ..................................................................... 102
3.2.4 Evaluation .................................................................................................................. 105

4 Diversity-promoting Learning III – Analysis .................................................................. 110
4.1 Analysis of Better Capturing of Infrequent Patterns ....................................................... 110
4.2 Analysis of Generalization Errors .................................................................................. 112
4.2.1 Generalization Error Analysis for the Angular Constraints ................................... 112
4.2.2 Estimation Error Analysis for the Nonconvex Bregman Matrix Divergence Regularizers ........................................................................................................ 114
4.2.3 Estimation Error Analysis for the Convex Bregman Matrix Divergence Regularizers ........................................................................................................ 115
4.3 Appendix: Proofs .......................................................................................................... 116
4.3.1 Proof of Theorem 1 ................................................................................................. 116
4.3.2 Proof of Theorem 2 ................................................................................................. 123
4.3.3 Proof of Theorem 3 ................................................................................................. 125
4.3.4 Proof of Theorem 4 ................................................................................................. 131
4.3.5 Proof of Theorem 5 ................................................................................................. 135

5 Large-scale Learning via System and Algorithm Co-design .............................................. 142
5.1 Sufficient Factor Property ............................................................................................... 144
5.2 Orpheus: A Light-weight Peer-to-peer System ............................................................... 146
5.2.1 Communication: Sufficient Factor Broadcasting (SFB) .......................................... 147
5.2.2 Fault Tolerance ....................................................................................................... 151
5.2.3 Programming Model ............................................................................................... 152
5.2.4 Implementation ....................................................................................................... 155
5.2.5 Evaluation ............................................................................................................... 158
5.3 Poseidon: A Hybrid Architecture of SFB and Parameter Server .................................. 165
5.3.1 Structure-aware Message Passing (SACP) Protocol ................................................ 165
List of Figures

1.1 Different types of clinical data .................................................. 2
1.2 Thesis goal: develop machine learning algorithms and systems to transform the raw clinical data into actionable insights ........................................... 2
1.3 (Left) Frequencies of 8 antihypertensive medications. (Right) F1 scores on individual medications in a discharge medication prediction task ........................................ 3
1.4 In this illustration, circles denote patterns and triangles denote the components. The size of a circle is proportional to the frequency of the corresponding pattern. (Left) Without diversification, ML models allocate a lot of components to cover the frequent patterns as best as possible. For the infrequent patterns, since they are weak signals, ML models tend to ignore them. (Right) With diversification, components that are originally cluttered around the frequent patterns are pushed apart. They diversely spread out. Some components that are originally cluttered around the frequent patterns are spared to cover the infrequent patterns ........................................ 4
1.5 Thesis scope. To address the first four challenges, we study diversity-promoting learning, which encourages the components in ML models to be diverse. To address the fifth challenge, we study large-scale distributed learning by exploiting a system and algorithm co-design approach. Then we apply the methods developed in these two learning paradigms to solve problems in healthcare ........................................ 5
1.6 Study scope of diversity-promoting learning ........................................ 6
2.1 An overview of the studies of diversity-promoting regularization ................. 15
2.2 Two views of the component matrix ........................................... 16
2.3 When the principal directions (u_1 and u_2) are not aligned with the coordinate axis, the level of disparity between the eigenvalues (λ_1 and λ_2) indicates the correlation between random variables (components) ........................................ 16
2.4 When the principal directions (u_1 and u_2) are aligned with the coordinate axis, the magnitude of eigenvalues represents the importance of components ................. 17
2.5 Phone error rate on TIMIT with varying τ ........................................ 43
2.6 Precision@10 versus the regularization parameter λ on MIMIC-III ............... 59
2.7 (a) Under L1 regularization, the vectors are sparse, but their supports are overlapped; (b) Under LDD regularization, the vectors are orthogonal, but their supports are overlapped; (c) Under LDD-L1 regularization, the vectors are sparse and mutually orthogonal and their supports are not overlapped ................. 63
2.8 Visualization of basis vectors ........................................... 68
2.9 Overlap score versus the regularization parameter ........................................ 70
3.1 A Bayesian network representation of the mutual angular process. ................. 86
3.2 Bayesian mixture of experts with mutual angular process. ......................... 90
3.3 L2 error on the Yale test set versus the concentration parameter $\kappa$. ........... 109

4.1 Large and small circles denote frequent and infrequent classes respectively. The
objective function of DML tends to make the inter-class distance to be large.
(Left) The frequent classes are dominant in the training objective of DML, and
hence are given higher priority for being pushed far away from each other. The
infrequent classes are paid less attention to. Their distance ends up being small
since DML is constrained by inner-class data pairs which need to have small
distances. (Right) By promoting diversity, the infrequent classes are paid more
attention to and their distance $d_2$ is enlarged. As a result, the ratio between $d_1$
and $d_2$ decreases. .......................................................... 111

5.1 Sufficient factor broadcasting. .................................................. 147
5.2 Random multicast. ............................................................... 148
5.3 Cost of using SFB versus PS. $K$ is the mini-batch size, $J,D$ are the dimensions
of $W$, and $P$ is the number of workers. ........................................ 151
5.4 Expression graph. ................................................................. 154
5.5 Software stack. ................................................................... 155
5.6 Convergence curves in the MLR experiments. ......................................... 159
5.7 Scalability with more machines: (left) Orpheus-MLR; (right) Orpheus-LSTM. ... 161
5.8 Breakdown of network waiting time (hours) and computation time (hours). ....... 162
5.9 How the system parameter $Q$ in random multicast (left) and $C$ in SF selection
(right) affect the running time of Orpheus for MLR. ............................. 163
5.10 (Left) Convergence time in Orpheus under deterministic, round-robin, and ran-
dom broadcast for MLR and LSTM. (Right) Relative increase of convergence
time when network connection fails. ................................................. 164
5.11 The illustration of three types of communications: (a) Full matrices communica-
tions via centralized parameter server (PS); (b) Sufficient factor broadcasting via
decentralized P2P scheme; (c) SF-based PS: from workers to the server, SFs are
transferred; from the server to workers, full matrices are transmitted. ............ 166
5.12 The comparison of training different CNNs to convergence between Poseidon on
8 GPU nodes and Caffe on a single node: (a)-(b) cifar10_quick_train_test; (c)-(d)
bvlc_alexnet; (e)-(f) bvlc_googlenet. Test errors are shown with respect to (left)
training time, and (right) training iterations. ........................................ 168
5.13 The speedups on throughpout with different values of staleness, when training
using Poseidon on 8 nodes, compared to Caffe on a single node. (a) AlexNet
with batch size 256, and (b) GoogLeNet with batch size 32. ..................... 170
5.14 Speedups against the single-machine Caffe in terms of throughpout, under differ-
ten number of GPU nodes: (a) AlexNet with a batch size of 256; (b) GoogLeNet
with a batch size of 32. .............................................................. 171
6.1 (a) Retrieval performance on frequent and infrequent diseases. (b) Retrieval
AUCs on the training and test sets. .................................................. 178

6.2 (a) A medical image has multiple tags which are correlated clinically or biolog-
ically. (b) The medical concepts are organized into a hierarchical tree. (c) The
abnormal regions (marked by red contours) are small. .......................... 184

6.3 Our model consists of three major modules. Given the medical image, a contex-
tual attention encoder is leveraged to learn a representation vector for this image.
The tree-of-sequences LSTM encoder is used to learn a representation for each
concept in the ontology. The image representation and concept representations
are fed into the adversarial multi-label tagger to generate the tags for this image. 186

6.4 Adversarial learning for capturing correlations among concepts. The discrimina-
tive network (DN) aims at distinguishing the labels produced by the predictive
network (PN) from the groundtruth labels provided by physicians. The PN tries
to make the two sets of labels indistinguishable. ................................. 187

6.5 Tree-of-sequences LSTM. For each concept name, we build a sequential LSTM
(SLSTM) to capture the semantics of words and the sequential structure among
words. Over the concept hierarchy, we build a tree LSTM (TLSTM) to capture
the hierarchical relationship among concepts. The encodings produced by the
SLSTM are the inputs of the TLSTM. ................................................. 187

6.6 Contextual attention module. For each patch, an attention score is calculated by
comparing the representation of this patch and that of the entire image. These
patches are re-weighted using the corresponding attention scores to generate an
attentional representation for this image. ........................................... 190

6.7 Attentional convolution. Each pixel is weighted by the attention score of the
patch containing this pixel. Then the weighted pixels are convoluted with the
filter weights. ................................................................. 190

6.8 Architecture of the ICD coding model. ........................................... 199
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Dataset statistics</td>
<td>22</td>
</tr>
<tr>
<td>2.2</td>
<td>Precision@10 (%) on three datasets. The Cars dataset has a single</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>train/test split, hence the standard error is 0. UE-DML is our method.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>On the second panel (EUC, etc.) are well established or state of the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>art distance metric learning baselines. On the the third panel (L2-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DML, etc.) and the fourth panel (DCM-DML, etc.) are DML methods</td>
<td></td>
</tr>
<tr>
<td></td>
<td>regularized by non-diversity regularizers and previously proposed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>diversity-promoting regularizers.</td>
<td></td>
</tr>
<tr>
<td>2.3</td>
<td>Optimal number of components.</td>
<td>24</td>
</tr>
<tr>
<td>2.4</td>
<td>Precision@10 (%) on frequent and infrequent diseases of the MIMIC-III</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>dataset.</td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td>Precision@10 (%) on three frequent and five infrequent diseases. The</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>number next to a disease ID is its frequency.</td>
<td></td>
</tr>
<tr>
<td>2.6</td>
<td>Average runtime (hours) of DML methods regularized by different</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>diversity-promoting regularizers.</td>
<td></td>
</tr>
<tr>
<td>2.7</td>
<td>Accuracy (%) on the two QA datasets. On the second panel (BIDAF, etc.</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>) we compare different diversity-promoting regularizers. On the first</td>
<td></td>
</tr>
<tr>
<td></td>
<td>panel (Kadlec et al., etc.) and the third panel (Dhingra et al., etc)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>are other state-of-the-art baselines.</td>
<td></td>
</tr>
<tr>
<td>2.8</td>
<td>Dataset statistics</td>
<td>32</td>
</tr>
<tr>
<td>2.9</td>
<td>Training time (hours) on seven datasets. On the second panel (DCM-</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>NCDML, etc.) are NCDML methods regularized by previously proposed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>diversity-promoting regularizers.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>On the third panel (VND-NCDML, etc.) are NCDML methods regularized</td>
<td></td>
</tr>
<tr>
<td></td>
<td>by our proposed nonconvex BMD regularizers. On the fourth panel (CSFN-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CDML, etc.) are CDML methods regularized by our proposed convex</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BMD regularizers.</td>
<td></td>
</tr>
<tr>
<td>2.10</td>
<td>On three imbalanced datasets – MIMIC, EICU, Reuters, we show the</td>
<td>73</td>
</tr>
<tr>
<td></td>
<td>mean AUC (averaged on 5 random train/test splits) on all classes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(AUC-All), frequent classes (AUC-F), and infrequent classes (AUC-IF).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>On the second panel (EUC, etc.) are well established or state of the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>art baselines. On the third panel (ℓ2-CDML, etc.) are CDML methods</td>
<td></td>
</tr>
<tr>
<td></td>
<td>regularized by non-diversity regularizers. On the fourth panel (DCM-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NCDML, etc.) are NCDML methods regularized by previously proposed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>diversity-promoting regularizers.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>On the fifth panel (VND-NCDML, etc.) are NCDML methods regularized</td>
<td></td>
</tr>
<tr>
<td></td>
<td>by our proposed nonconvex BMD regularizers. On the sixth panel (CSFN-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CDML, etc.) are CDML methods regularized by our proposed convex BMD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>regularizers.</td>
<td></td>
</tr>
<tr>
<td>2.11</td>
<td>AUC-All on 4 balanced datasets.</td>
<td>74</td>
</tr>
</tbody>
</table>
2.12 The numbers of components that achieve the AUCs in Table 6.1 and 2.11 . . . . 74
2.13 The gap of training AUC and testing AUC . . . . . . . . . . . . . . . . . . . . . 75
2.14 Classification accuracy (%) on three datasets. We compare the proposed ACs
with existing diversity-promoting regularizers . . . . . . . . . . . . . . . . . . . . . . 75
2.15 Phone error rate (%) on the TIMIT test set. On the second panel (Kaldi, etc.), we
compare AC with previously proposed diversity-promoting regularizers. On the
first panel (Segmental NN, etc.) and the third panel (CTC, etc.) are other state of
the art baselines . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 76
2.16 Classification error (%) on the CIFAR-10 test set . . . . . . . . . . . . . . . . . 76
2.17 Accuracy (%) on two QA datasets . . . . . . . . . . . . . . . . . . . . . . . . . . . 78
2.18 Average runtime (hours). We compare AC with existing diversity-promoting
regularizers . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 78
2.19 Dataset statistics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 78
2.20 Retrieval precision@10 (%) on three datasets. On the first panel (EUC, etc.)
are non-kernel DML methods. On the second panel (Tsang, etc.) are well es-
established or state of the art kernel DML methods. On the third panel (KDML,
etc.) are KDML methods regularized by existing regularizers. On the fourth
panel (KDML-SFN-RTR, etc.) are KDML methods regularized by our proposed
near-orthogonality regularizers . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 79
2.21 The number of RKHS functions achieving the precision@10 in Table 2.20 . . . . 79
2.22 Retrieval precision@10 (%) on three frequent and five infrequent diseases in the
MIMIC-III dataset. The number next to a disease ID is its frequency. Note that
diseases D1–D3 are frequent diseases, while that D4–D8 are infrequent ones . . . 80
2.23 Convergence time (hours) on three datasets . . . . . . . . . . . . . . . . . . . . . 80
2.24 Classification accuracy (%) on three datasets . . . . . . . . . . . . . . . . . . . . 80
2.25 Sensitivity and specificity for support recovery and error rate for prediction . . . 80
2.26 Classification accuracy on the test sets of 20-News and RCV1, and the gap be-
tween training and test accuracy . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 81
2.27 Selected words of 9 exemplar basis vectors . . . . . . . . . . . . . . . . . . . . . . 81
2.28 Word-level perplexities on the PTB test set. On the second panel (PytorchLM,
etc., we compare LDD-L1 with the L1 regularizer and orthogonality-promoting
regularizers. On the first panel (RNN, etc.) and the third panel (Pointer Sentinel
LSTM, etc.) are other state of the art baselines . . . . . . . . . . . . . . . . . . . . . . 82
2.29 Classification error (%) on the CIFAR-10 test set . . . . . . . . . . . . . . . . . 83

3.1 Classification accuracy (%) on the Adult-9 dataset . . . . . . . . . . . . . . . . . 92
3.2 Classification accuracy (%) on the SUN-Building dataset . . . . . . . . . . . . . . 92
3.3 Training time (hours) of different methods with K = 30 . . . . . . . . . . . . . . . 93
3.4 Accuracy on 9 subcategories of the CCAT category in the RCV1.Binary dataset . 93
3.5 Classification accuracy (%) on two datasets . . . . . . . . . . . . . . . . . . . . . 94
3.6 L2 test error . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 104
3.7 Test log-likelihood . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 104
3.8 Clustering accuracy (%) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 105
3.9 Normalized mutual information (%) . . . . . . . . . . . . . . . . . . . . . . . . . . 105
6.8 Weighted sensitivity and specificity on the test set. On the first panel are baseline
lines for holistic comparison. On the second panel are baselines compared in the
ablation study of tree-of-sequences LSTM for capturing hierarchical relation-
ship. On the third panel are baselines compared in the ablation study of adver-
arial learning for writing-style reconciliation, isotonic constraints for ranking,
and attentional matching.

6.9 Comparison of NDCG scores in the ablation study of isotonic constraints.
Chapter 1

Introduction

1.1 Thesis Introduction and Scope

With the widespread adoption of electronic health records (EHR) systems, and the rapid development of new technologies such as high-throughput medical imaging devices, low-cost genome profiling systems, networked and even wearable sensors, mobile applications, and rich accumulation of medical knowledge/discoveries in databases, a tsunami of medical and healthcare data has emerged. It was estimated that 153 exabytes (one exabyte equals one billion gigabytes) of healthcare data were produced in 2013 [431]. In 2020, an estimated 2314 exabytes will be produced. From 2013 to 2020, an overall rate of increase is at least 48 percent annually [431]. In addition to the sheer volume, the complexity of healthcare data is also overwhelming. As shown in Figure 1.1, it includes clinical notes, medical images, lab values, vital signs, etc., coming from multiple heterogeneous modalities including texts, images, tabular data, time series, graph and so on. The rich clinical data is becoming an increasingly important source of holistic and detailed information for both healthcare providers and receivers. Collectively analyzing and digesting these rich information generated from multiple sources; uncovering the health implications, risk factors, and mechanisms underlying the heterogeneous and noisy data records at both individual patient and whole population levels; making clinical decisions including diagnosis, triage, and treatment thereupon, are now routine activities expected to be conducted by medical professionals including physicians, nurses, pharmacists and so on. As the amount and complexity of medical data are rapidly growing, these activities are becoming increasingly more difficult for human experts. The information overload makes medical analytics and decisions-making time-consuming, error-prone, suboptimal, and less-transparent. As a result, physicians, patients, and hospitals suffer a number of pain points, quality-wise and efficiency-wise. For example, in terms of quality, 250,000 Americans die each year from medical errors, which has become the third leading cause of death in the US [90]. 12 million Americans are misdiagnosed each year [305]. Preventable medication errors impact more than 7 million patients and cost almost $21 billion annually [87]. 15 to 25 percent of patients are readmitted within 30 days and readmissions are costly (e.g., $41.3 billion in 2011) [158]. In terms of inefficiency, patients wait on average 6 hours in emergency rooms [12]. Nearly 400,000 patients wait 24 hours or more. Physicians spend only 27 percent of their office day on direct clinical face time with patients [306]. The
U.S. healthcare system wastes $750 billion annually due to unnecessary services, inefficient care delivery, excess administrative costs, etc [308].

The advancement of machine learning (ML) technology opens up opportunities for next generation computer-aided medical data analysis and data-driven clinical decision making, where machine learning algorithms and systems can be developed to automatically and collectively digest massive medical data such as electronic health records, images, behavioral data, and the genome, to make data-driven and intelligent diagnostic predictions. An ML system can automatically analyze multiple sources of information with rich structure; uncover the medically-meaningful hidden concepts from low-level records to aid medical professionals to easily and concisely understand the medical data; and create a compact set of informative diagnostic procedures and treatment courses and make healthcare recommendations thereupon.

In this thesis, we aim at leveraging the power of machine learning in automatically distilling insights from large-scale heterogeneous data for automatic smart data-driven medical predictions, recommendations, and decision-making, to assist physicians and hospitals in improving the quality and efficiency of healthcare (Figure 1.2). We develop machine learning algorithms...

Figure 1.1: Different types of clinical data.

Figure 1.2: Thesis goal: develop machine learning algorithms and systems to transform the raw clinical data into actionable insights.
and systems that turn the raw clinical data into actionable insights. Specifically, we focus on the following clinical applications: retrieving similar patients [376], discovering medical topics from large-scale texts, hierarchical tagging of medical images, and automatically assigning ICD codes [378].

During the development of these algorithms, we identify several fundamental issues.

- **How to better capture infrequent patterns?** At the core of ML-based healthcare is to discover the latent patterns (e.g., topics in clinical notes, disease subtypes, phenotypes) underlying the observed clinical data. Under many circumstances, the frequency of patterns is highly imbalanced [376]. Some patterns have very high frequency while others occur less frequently. For instance, Figure 1.3 (Left) shows the frequencies that 8 antihypertensive medications are prescribed. Metoprolol and furosemide are used very frequently while the rest less often. Existing ML models lack the capability of capturing infrequent patterns. We applied a convolutional neural network model to classify the aforementioned 8 antihypertensive medications [393]. Figure 1.3 (Right) shows the F1 scores (the higher, the better) on individual medications. As can be seen, while achieving high F1 scores on frequent medications, CNN performs less well on the infrequent ones. Such a deficiency of existing models possibly results from the design of their objective function used for training [371]. For example, a maximum likelihood estimator would reward itself by modeling the frequent patterns well as they are the major contributors to the likelihood function. On the other hand, infrequent patterns contribute much less to the likelihood, thereby it is not very rewarding to model them well and they tend to be ignored. Figure 1.4 (Left) presents an illustration. Since dominant patterns denoted by these two large circles are the major contributors of the likelihood function, ML models would allocate a number of triangles to cover the large circles as best as possible. On the other hand, the infrequent patterns denoted by the small circles contribute less to the likelihood function, thereby it is not very rewarding to model them well and ML models tend to ignore them. Infrequent patterns are of crucial importance in clinical settings. For example, many infrequent diseases are life-threatening [363]. It is critical to capture them.

- **How to alleviate overfitting?** In certain clinical applications, the number of medical records available for training is limited. For example, when training a diagnostic model for an infrequent disease, we typically have no access to a sufficiently large number of patient cases due to the rareness of this disease. Under such circumstances, overfitting easily happens: the trained model works well on the training data but generalizes poorly on unseen patients. To alleviate overfitting, we need to incorporate prior beliefs of the model structure.
Figure 1.4: In this illustration, circles denote patterns and triangles denote the components. The size of a circle is proportional to the frequency of the corresponding pattern. (Left) Without diversification, ML models allocate a lot of components to cover the frequent patterns as best as possible. For the infrequent patterns, since they are weak signals, ML models tend to ignore them. (Right) With diversification, components that are originally cluttered around the frequent patterns are pushed apart. They diversely spread out. Some components that are originally cluttered around the frequent patterns are spared to cover the infrequent patterns.

- **How to improve interpretability?** Being interpretable and transparent is a must for an ML model to be willingly used by human physicians. Oftentimes, the patterns extracted by existing ML methods have a lot of redundancy and overlap [350], which are ambiguous and difficult to interpret. For example, in computational phenotyping from EHRs, it is observed that the learned phenotypes by the standard matrix and tensor factorization [350] algorithms have much overlap, causing confusion such as two similar treatment plans are learned for the same type of disease [350]. It is necessary to make the learned patterns distinct and interpretable.

- **How to compress model size without sacrificing modeling power?** In clinical practice, making a timely decision is crucial for improving patient outcome. To achieve time efficiency, the size (specifically, the number of weight parameters) of ML models needs to be kept small. However, reducing the model size, which accordingly reduces the capacity and expressivity of this model, typically sacrifice modeling power and performance. It is technically appealing but challenging to compress model size without losing performance.

- **How to efficiently learn large-scale models?** In certain healthcare applications, both the model size and data size are large, incurring substantial computation overhead that exceeds the capacity of a single machine. It is necessary to design and build distributed systems to efficiently train such models.

To solve the first four problems, we study *diversity-promoting learning* (DPL) [223, 367, 369, 371, 372, 373, 376, 379, 380, 381]. Many ML models are equipped with *components*, each aiming at capturing a latent pattern and is parameterized by a weight vector. For instance, in a topic model [49], the components are referred to as *topics*, aiming at discovering the semantics underlying documents. Each topic is associated with a multinomial vector. In neural networks, the components are called *hidden units*. Each unit, parameterized by a weight vector, is desired to capture a latent feature. DPL aims at encouraging the component vectors to be “diverse”. First,
Figure 1.5: Thesis scope. To address the first four challenges, we study diversity-promoting learning, which encourages the components in ML models to be diverse. To address the fifth challenge, we study large-scale distributed learning by exploiting a system and algorithm co-design approach. Then we apply the methods developed in these two learning paradigms to solve problems in healthcare.

Regarding better capturing infrequent patterns, if the model components are biased to be far apart from each other, then one would expect that such components will tend to be less overlapping and less aggregated over frequent patterns. They diversely spread out to different regions in the pattern space and give the infrequent patterns a better chance to be captured [369], as illustrated in Figure 1.4 (Right). Second, regarding alleviating overfitting, promoting diversity imposes a structural constraint on model parameters, which reduces the model capacity and therefore improves generalization performance on unseen data [372]. Third, regarding interpretability, if components are encouraged to be distinct from each other and non-overlapping, then it would be cognitively easy for a human to associate each component to an object or concept in the physical world [350]. Fourth, regarding performance-lossless model-compression, “diversified” components bear less redundancy and are mutually complementary, making it possible to capture information sufficiently well with a small set of components [367]. To address the fifth problem, we design efficient distributed ML systems [370, 377, 384, 411], by exploiting a system-algorithm co-design approach: system design is tailored to the unique mathematical properties of ML algorithms, and algorithms can be re-designed to better exploit the system architecture. We apply the developed diversity-promoting learning techniques and distributed ML systems to healthcare applications. Figure 1.5 summarizes the scope of this thesis.

1.2 Contributions

Overall, the contributions of this thesis are made in three areas: diversity-promoting learning, large-scale distributed learning, and ML for healthcare.
1.2.1 Diversity-promoting Learning

This thesis work represents the first one that systematically studies this new learning paradigm: diversity-promoting learning. In statistics, there are two paradigms: frequentist-style point estimation and Bayesian inference. Our study of diversity-promoting learning is tailored to these two paradigms. In frequentist-style point estimation, we develop empirically effective, theoretically analyzable, and computationally efficient regularization approaches to encourage model components to be diverse. In Bayesian inference, we design Bayesian priors that effectively entail an inductive bias of “diversity” among a finite or infinite number of components and develop efficient posterior inference algorithms. Finally, we provide theoretical analysis regarding the benefits of promoting diversity. Figure 1.6 summarizes the study scope of diversity-promoting learning.

Diversity-promoting Regularization

In diversity-promoting regularization, we made the following contributions.

- We propose to characterize “diversity” from two perspectives: uncorrelation and evenness, based on which we define a uniform eigenvalue regularizer (UER) [376]. Compared with previous diversity-promoting regularizers, the UER is able to measure “diversity” in a global way, is insensitive to vector scaling, and is amenable for computation. We apply UER to distance metric learning [383] and long short-term memory networks [163] and develop an efficient projected gradient descent algorithm. In various experiments, we demonstrate the effectiveness of UER in better capturing infrequent patterns, reducing model size without sacrificing modeling power, and improving generalization performance.

- Considering UER is nonconvex which presents great challenges for optimization, we develop a family of convex diversity-promoting regularizers [379] based on the Bregman matrix divergence (BMD) [100], where the global optimal is guaranteed to be achievable. We apply these regularizers to distance metric learning and develop an efficient proximal gradient algorithm [270]. In experiments, we demonstrate the advantages of the convex BMD (CBMD) regularizers over the nonconvex counterparts. First, because the global optimal solution is achievable, CBMD obtains better modeling performance. Second, unlike nonconvex regular-
izers that need multiple (random) restarts for a better local optimal, CBMD runs only once, hence is computationally more efficient.

- While UER and convex BMD regularizers are empirically effective in alleviating overfitting, a theoretical analysis of their effectiveness is difficult to establish. In light of this, we propose a new regularization approach: angular constraints (ACs) [372], that is both empirically effective and theoretically analyzable. The analysis reveals that properly manipulating the ACs can achieve the lowest generalization errors. We develop an efficient algorithm based on ADMM [52] and demonstrate the empirical effectiveness of ACs in alleviating overfitting of deep neural networks and sparse coding [266]. A connection is also made between the ACs and the log-determinant divergence regularizer [379].

- We extend the study of diversity-promoting learning from finite-dimensional vectors to infinite-dimensional functions in the reproducing kernel Hilbert space (RKHS) [374]. We define Bregman matrix divergence regularizers on RKHS functions and solve the regularized problems using functional gradient descent [89]. On two case studies – kernel sparse coding [120] and kernel distance metric learning [335] – we demonstrate the merits of promoting diversity in RKHS.

- We combine diversity-promoting regularization with sparsity-promoting regularization, which jointly bring in an effect of nonoverlapness [380], for the sake of selecting less-overlapped variables in ML problems that have multiple responses. We propose an LDD-L1 regularizer and derive efficient coordinate descent algorithms. Experiments on four ML models demonstrate the effectiveness of LDD-L1 in selecting less-overlapped variables and improving generalization performance.

**Diversity-promoting Bayesian Learning**

In diversity-promoting Bayesian learning, we made the following contributions.

- We define a mutual angular process (MAP) [381], which is a Bayesian prior biased towards components that have large mutual angles. This prior facilitates the development of posterior inference algorithms based on variational inference [340], which is usually more efficient than sampling-based [127] algorithms. We apply this prior to a Bayesian mixture of experts model [354] and demonstrate its effectiveness and efficiency in experiments.

- To promote diversity in Bayesian nonparametric models [111], we extend the MAP to infinite MAP (IMAP) which encourages infinitely many components to have large mutual angles. We apply the IMAP to an infinite latent feature model [143] and develop a posterior inference algorithm based on slicing sampling [328] and Riemann manifold Hamiltonian Monte Carlo [129]. Experiments demonstrate the effectiveness of IMAP.

**Theoretical Analysis**

We performed various analysis to formally understand the effectiveness of promoting diversity and made the following contributions.

- We analyze why the nonconvex Bregman matrix divergence (BMD) regularizers can better capture infrequent patterns [379]. In the context of distance metric learning, we define an
imbalance factor (the lower, the better) to characterize the performance on infrequent patterns. The analysis shows that decreasing the BMD regularizers can reduce the upper bound of the imbalance factor and hence achieve better performance on infrequent patterns.

- We analyze how the angular constraints (ACs) affect the generalization error (which is the sum of estimation error and approximation error) of neural networks [372]. The analysis reveals that a stronger regularization reduces estimation errors and increases approximation errors. Properly tuning the regularization strength can achieve the best tradeoff among these two types of errors and accordingly the optimal generalization performance on unseen data.
- We analyze how the nonconvex [374, 380] and convex [379] BMD regularizers affect the estimation error of distance metric learning. The analysis shows that decreasing these regularizers can effectively reduce the estimation error bound.

1.2.2 Large-scale Distributed Learning

The second part of this thesis studies large-scale learning. We design efficient distributed ML systems by exploiting a system-algorithm co-design approach. Specifically, inspired by a mathematical property – the sufficient factor (SF) property [370] – that is shared by many ML models parameterized by matrices, we design a peer-to-peer system [370, 377], that significantly reduces communication and fault tolerance costs. We made the following contributions.

- For efficient communication, we propose (1) sufficient factor broadcasting (SFB) [370] which transfers small-sized vectors among machines for the synchronization of matrix-form parameters, (2) random multicast [377] where each machine randomly selects a subset of machines to communicate within each clock, (3) SF selection [377] that selects a subset of most representative SFs to communicate. These techniques greatly reduce the number of network messages and the size of each message.
- For efficient fault tolerance, we propose to represent the parameter matrix using SFs and propose an incremental SF checkpoint (ISFC) scheme [377]. ISFC continuously saves new SFs computed at each clock to stable storage and greatly reduces disk IO, avoids compute-cycle waste, and provides fine-grained (per-clock) rollbacks.
- We provide an easy-to-use programming interface [377] which can automatically identify the computation of SFs and the transformation from SFs to update matrices.
- We conduct convergence analysis of the SFB computation model [370]. The analysis shows that though synchronized in a decentralized manner, the parameter replicas on different machines converge to the same optimal.
- We evaluate our system on three representative ML models and show that it achieves high efficiency and scales well with more machines.
- For ML models having both small-sized and large-sized update matrices, we propose a structure-aware message passing (SAMP) protocol [411], which is a hybrid communication approach between the centralized parameter server (PS) [226, 355] and decentralized SFB. PS and SFB are utilized to transfer small and large matrices respectively. The protocol leverages the best of these two communication schemes and significantly minimizes the communication cost. We evaluate this protocol on convolutional neural networks. Un-
der 8 GPU machines, SAMP improves the speedup from 3.9 to 5.7 on AlexNet \cite{203} and from 3.8 to 4.8 on GoogLeNet.

### 1.2.3 ML for Healthcare

In the third part of this thesis, we design ML models and apply the diversity-promoting and large-scale learning techniques developed in the first two parts to address critical problems in healthcare. We made the following contributions.

- We apply our diversity-promoting distance metric learning model \cite{379} for similar-patient retrieval and demonstrate its effectiveness on two electronic health records datasets. Thanks to the ability of our method in better capturing infrequent patterns, better retrieval performance is achieved on infrequent diseases. By promoting diversity, the dimension of latent representations (accordingly, the time complexity of retrieval) can be reduced, without sacrificing the retrieval accuracy. This enables fast and accurate retrieval, which facilitates timely and sound clinical decision-making. Besides, our diversity-promoting regularizer effectively reduces overfitting.

- We implement a distributed topic model (TM) on our Orpheus system \cite{377} and apply it to large-scale medical-topic discovery. Leveraging the sufficient factor (SF) property of the TM, Orpheus performs SF broadcasting and incremental SF checkpoint to significantly reduce communication and fault tolerance costs. Using 34 CPU machines, the Orpheus-TM is able to learn 50K topics from 8.2M PubMed documents (vocabulary size is 141K) in 5.4 hours, which is much faster than FlexiFaCT \cite{207} (33.9 hours) – a Hadoop-based model-parallel system, and Bosen \cite{355} (23.5) – a parameter server based data-parallel system.

- To solve the problem that medical images are difficult to index and search due to the lack of textual tags, we study multi-label hierarchical tagging of images. We propose an adversarial learning \cite{134} strategy to capture the correlations among medical concepts, a tree-of-sequences LSTM model \cite{378} to explore the hierarchical structure of the concept ontology, and a contextual attention model to localize abnormal regions. Experiments on a pathology dataset and a radiology dataset demonstrate the effectiveness of our methods.

- We study the automatic assignment of ICD codes based on physicians’ free-form diagnosis descriptions, to reduce coding errors and costs \cite{378}. A neural architecture is proposed, which consists of four ingredients: (1) tree-of-sequences LSTM encoding for simultaneously capturing the semantics and hierarchical relationship of codes, (2) adversarial learning for reconciling the different writing styles of diagnosis descriptions (DDs) and code descriptions (CDs), (3) isotonic constraints for incorporating the importance order among the assigned codes, and (4) attentional matching for performing many-to-one and one-to-many mappings from DDs to CDs. We demonstrate the effectiveness of the proposed methods on a clinical datasets with 59K patient visits.
1.3 Related Works

In this section, we present a general review of related works in diversity-promoting learning, large-scale distributed learning, and machine learning for healthcare. Additional related works are reviewed in individual sections.

1.3.1 Diversity-promoting Learning

Diversity promoting regularization has been studied in ensemble learning [402], latent space modeling [365, 369, 429], and classification [242]. In ensemble learning, several studies [27, 209, 271, 402] explore how to select a diverse subset of base classifiers or regressors, with the aim to improve generalization performance. In latent space modeling, several works [29, 82, 429] encourage components to be mutually different for the sake of reducing redundancy and alleviating overfitting. In multi-way and hierarchical classification, two works [177, 242, 420] promote “diversity” among the coefficient vectors of classifiers, for the sake of ensuring model continuity and exploiting the semantic relationship embedded in the class hierarchy. In these works, various diversity-promoting regularizers have been proposed, based on determinantal point process [204, 429], cosine similarity [29, 369, 402] and matrix covariance [82]. In the sequel, we present a brief review of them.

**Determinantal point process (DPP)**

DPP [204] was used by [246, 429] to encourage the topic vectors in latent Dirichlet allocation [49], Gaussian mean vectors in Gaussian mixture model and hidden units in neural network to be “diverse”. DPP is defined over \( K \) vectors: 

\[
p(\{a_i\}_{i=1}^{K}) \propto \det(L),
\]

where \( L \) is a \( K \times K \) kernel matrix with \( L_{ij} = k(a_i, a_j) \) and \( k(\cdot, \cdot) \) is a kernel function. \( \det(\cdot) \) denotes the determinant of a matrix. A configuration of \( \{a_i\}_{i=1}^{K} \) with larger probability is deemed to be more “diverse”. The underlying intuition is that: \( \det(L) \) represents the volume of the parallelepiped formed by vectors in the feature space associated with the kernel \( k \). If these vectors are more mutually different, the volume is larger, which results in a larger \( p(\{a_i\}_{i=1}^{K}) \).

The shortcoming of DPP is that it is sensitive to vector scaling. Enlarging the magnitudes of vectors results in larger volume, but does not essentially change the “diversity” of vectors.

**Pairwise cosine similarity**

Several works [29, 369, 402] define diversity-promoting regularizers based on the pairwise cosine similarity among component vectors: smaller cosine similarity scores imply that the components are more different from each other, hence are more “diverse”. Cosine similarity is preferred over other distances or similarity measures because of its insensitivity to geometry transformations of vectors such as scaling, translation, and rotation. Given \( K \) component vectors, the cosine similarity \( s_{ij} \) between each pair of components \( a_i \) and \( a_j \) is computed as 

\[
s_{ij} = a_i \cdot a_j / (\|a_i\|_2 \|a_j\|_2).
\]

Then these scores are aggregated to measure the overall “diversity” of all components. In [402], these scores are aggregated as 

\[
\sum_{1 \leq i < j \leq K} (1 - s_{ij}).
\]

In [29], the aggregation is performed as 

\[
- \log(\frac{1}{K(K-1)} \sum_{1 \leq i < j \leq K} \beta |s_{ij}|) \beta > 0.
\]

In [369], the aggregated score is defined as the mean of \( \arccos(|s_{ij}|) \) minus the variance of \( \arccos(|s_{ij}|) \). The variance term is utilized to encourage the vectors to evenly spread out to different directions. Xie et al. [365] define the regularizer as 

\[
\sum_{1 \leq i < j \leq m} k(a_i, a_j) \text{ where } k(\cdot, \cdot) \text{ is a kernel function.}
\]
These regularizers are applied to classifiers ensemble, neural networks, and restricted Boltzmann machine [369]. These approaches can only capture second-order (pairwise) relations among vectors. Representing higher-order relations such as how to measure the “diversity” of three vectors as a whole (without reducing to pairwise dissimilarity) is beyond the capability of these methods.

**Covariance**  
Two works [82, 242] defined diversity-promoting regularizers based on matrix covariance. Malkin and Bilmes [242] compute the covariance matrix \( C \) of the component vectors:
\[
C = \frac{1}{K} \sum_{i=1}^{K} (a_i - \mu) (a_i - \mu)^\top
\]
where \( \mu = \frac{1}{K} \sum_{i=1}^{K} a_i \), then encourage diversity by maximizing the determinant of \( C \). Similar to the DPP, this regularizer is sensitive to vector scaling. Cogswell et al. [82] designed a regularizer to reduce the correlation among hidden units in neural networks. Given \( N \) activation vectors \( \{h_i\}_{i=1}^{N} \) computed over \( N \) data samples where different dimensions of each vector correspond to different hidden units, they compute the sample covariance matrix of hidden units \( C = \frac{1}{N} \sum_{i=1}^{N} (h_i - \mu)(h_i - \mu)^\top \) where \( \mu = \frac{1}{N} \sum_{i=1}^{K} h_i \) and define the regularizer as \( \|C\|_2^2 - \|\text{diag}(C)\|_2^2 \), which encourages the off-diagonal entries of \( C \), i.e., the covariance between different hidden units, to be small. This regularizer is defined over intermediate variables (which are hidden activations in the neural network case) and influences the weight parameters indirectly. The number of intermediate variables could be much larger than that of weight parameters (which is the case in convolutional neural networks). This renders this regularizer computationally inefficient.

### 1.3.2 Distributed Learning

Many distributed ML systems have been built recently, including (1) dataflow systems such as Hadoop-based Mahout [93] and Spark-based MLlib [407]; (2) graph computation frameworks such as Pregel [241] and GraphLab [132]; (3) parameter server (PS) architectures such as DistBelief [93], Project Adam [78], ParameterServer [226], Bosen PS [355], and GeePS [85]; (4) hybrid systems such as TensorFlow [13] and MXNet [73]. These systems explore the trade-offs among correctness of computation, ease of programmability, and efficiency of execution. Hadoop-Mahout and Spark-MLlib provide an easy-to-use MapReduce-style programming interface and strictly guarantee computational exactness using BSP. However, their speed of execution is typically slower than ML-specialized systems, partially because (1) BSP is sensitive to stragglers [162]; (2) parameter state is immutable, which does not support fast in-place update. Frameworks based on graph abstractions [132, 241] support fine-grained scheduling of computations and flexible consistency models, which contribute to high efficiency. However, to run on these frameworks, an ML model needs to be abstracted as a graph, which is very difficult to achieve for MPMs. Parameter server architectures [78, 85, 94, 226, 355] and hybrid systems [13, 73] (partially adopting PS) offer a flexible and easy-to-use *distributed shared memory* programming abstraction and possess high efficiency by allowing mutable parameter states and in-place updates, fine-grained partitioning of computation, and flexible consistency models. However, when used to train MPMs, these systems communicate large matrices, which incur high communication overhead.

Peer-to-peer (P2P) architectures have been investigated in distributed ML [222, 353, 370, 411]. Li et al. [222] propose to synchronize parameter replicas by exchanging parameter updates
in a P2P manner to simplify fault-tolerance. Watcharapichat et al. [353] design a decentralized architecture to exchange partial gradient of deep neural networks among machines, for the sake of saturating cluster resources. These two works transmit matrices in the network and do not leverage the SVs to reduce communication cost.

1.3.3 ML for Healthcare

With the prosperity of healthcare data such as electronic health record (EHR), genomic data, patient behavior data and the growing need of extracting knowledge and insights from these data, ML-based data-driven healthcare analytics have received much attention recently.

Predictive modeling  Predictive modeling in data-driven healthcare is concerned about building machine learning models to predict diagnosis, prognosis, patient risk factors, readmission, disease onset and so on. Wang et al. [344] studied disease prognosis by leveraging the information of clinically similar patient cohorts which are retrieved using a local spline regression based similarity measure. Zhou et al. [422] proposed a top-k stability selection method to select the most informative features for patient risk prediction. Chen et al. [72] developed a cloud-based predictive modeling system for pediatric asthma readmission prediction. Choi et al. [79] developed a temporal model using recurrent neural networks to predict the diagnosis and medication categories for a subsequent visit. Razavian et al. [287] developed predictive models to predict the onset and assess the risk factors of type 2 diabetes. Razavian et al. [288] used LSTM [163] networks and convolutional networks for multi-task prediction of disease onset.

Natural language processing and understanding of clinical notes  Clinical notes contain rich medical information. Many studies have developed natural language processing and machine learning methods to extract useful information from free-form clinical texts. Chen et al. [75] studied word sense disambiguation using support vector machine and active learning. Tang et al. [324] developed a temporal information extraction system that can identify events, temporal expressions, and their temporal relations in clinical texts. Tang et al. [323] studied clinical entities recognition in hospital discharge summaries using structural support vector machines. Gobbel et al. [130] designed a tool to assist in the annotation of medical texts, which leverages interactive training to reduce the annotation time without introducing bias. Tang et al. [325] performed a comparison study of three types of word representations, including clustering-based representation, distributional representation, and word embeddings, for biomedical named entity recognition. Halpern et al. [150] developed a bipartite probabilistic graphical models for joint prediction of clinical conditions from the electronic medical records.

Computational phenotyping  Computational phenotyping, which extracts high-level clinical concepts and patterns from EHRs, have been widely investigated. Chen et al. [75] integrated an uncertainty sampling active learning approach with support vector machine for high-throughput phenotyping. Doan et al. [103] developed an information retrieval system that consists of text processing tools to standardize phenotype variables and information retrieval tools that support
queries from users and return ranked results. Ho et al. [161] developed a non-negative tensor factorization method to derive phenotype candidates with minimal human supervision. Wang et al. [350] proposed a constrained non-negative tensor factorization and completion method for phenotyping which incorporates guidance constraints to align with existing medical knowledge, and pairwise constraints for obtaining less-overlapped phenotypes. Halpern et al. [149] developed a phenotype library that uses both structured and unstructured data from the EHRs to represent patients for real-time clinical decision support. Joshi et al. [187] proposed a non-negative matrix factorization method augmented with domain specific constraints for automated phenotyping of multiple co-occurring medical conditions.

**Patient similarity measure** Effectively measuring the similarity between patients can help with a number of downstream applications such as diagnosis, prognosis, and treatment. Ebadollahi et al. [106] leveraged inter-patient similarity for retrieving patients who display similar trends in their physiological time-series data. Wang et al. [343] proposed a composite distance integration approach to combine the individual distance metrics learned from different physicians into a globally consistent metric. Sun et al. [315] proposed a locally supervised metric learning approach to learn a generalized Mahalanobis distance that is tailored toward physician feedback and introduce an interactive metric learning method that can incrementally update an existing metric based on streaming feedback. Ng et al. [261] applied distance metric learning to find clinically similar patients and trained personalized predictive models on the retrieved patients.

**Disease progression modeling** Disease progression modeling is instrumental in early diagnosis and personalized care. Jackson et al. [173] developed a hidden Markov model for simultaneously estimating the transition rates and the probabilities of stage misclassification. Zhou et al. [421] proposed a fused group lasso approach for disease progression modeling with known biomarkers. Exarchos et al. [108] developed a dynamic Bayesian network to model the progression of coronary atherosclerosis. Wang et al. [348] proposed a probabilistic disease progression model that learns from discrete-time observations with non-equal intervals and discovers the full progression trajectory from a set of incomplete records.
Chapter 2

Diversity-promoting Learning I – Regularization

In this chapter, we study diversity-promoting learning in the context of frequentist statistics, by developing regularization techniques.

Figure 2.1 shows an overview of the studies of diversity-promoting regularization. We begin with defining a uniform eigenvalue regularizer that simultaneously encourages uncorrelation and evenness among components [376]. This regularizer is nonconvex, presenting great challenges for optimization. In light of this, we develop convex Bregman matrix divergence regularizers where the global optimal is achievable [379]. While the uniform eigenvalue regularizer and the convex Bregman matrix divergence regularizers are empirically effective, theoretically they are not amenable for the analysis of generalization errors, especially the approximation errors. Therefore, we propose a new regularization approach called angular constraints [372] which are not only empirically effective, but also theoretically analyzable. These three regularizers promote diversity among finite-dimensional vectors. In the next work, we extend the study to infinite-dimensional functions in the reproducing kernel Hilbert space (RKHS) [373]. Finally, we combine diversity-promoting regularization with sparsity-promoting regularization [380], which jointly brings in an effect of reducing overlap among the supports of vectors.

2.1 Uncorrelation and Evenness: A Diversity-promoting Regularizer

We start with formally defining “diversity” [376]. Several diversity-promoting regularizers have been proposed, based upon determinantal point process [204, 429], cosine similarity [29, 369, 402], and covariance [82, 242]. While these regularizers demonstrate notable efficacy, they have certain limitations, such as sensitivity to vector scaling [242, 429], inability to measure diversity in a global manner [29, 369, 402], and computational inefficiency [82]. To address these limitations, we propose a new diversity-promoting regularizer [376] gaining inspiration from principal component analysis [185], biological diversity [240], and information theory [83]. We characterize “diversity” by considering two factors: uncorrelation and evenness. Uncorrelation is a measure of how uncorrelated the components are. Literally, less correlation is equivalent to more
diversity. By encouraging the components to be uncorrelated, each of them can independently capture a unique pattern. Evenness is borrowed from biological diversity [240], which measures how equally important different species are in maintaining the ecological balance within an ecosystem. If no species dominates another, the ecosystem is deemed as being more diverse. Likewise, in ML, we desire the components to play equally important roles and no one dominates another, such that each component contributes significantly to the modeling of data. Specifically, we assign each component an “importance” score and encourage these scores to be even.

We study uncorrelation and evenness from a statistical perspective. The components are considered as random variables and the eigenvalues of their covariance matrix can be leveraged to characterize these two factors. First, according to the principle component analysis [185], the disparity of eigenvalues reflects the correlation among components: the more uniform the eigenvalues, the less correlated the components. Second, eigenvalues represent the variance along principal directions and can be used to measure the “importance” of components. Promoting uniform importance amounts to encouraging evenness among eigenvalues.

To promote uniformity among the eigenvalues, we encourage the discrete distribution parameterized by the normalized eigenvalues to have small Kullback-Leibler divergence with the uniform distribution, based on which, we define a uniform eigenvalue regularizer (UER) [376]. We apply UER to two ML models – distance metric learning (DML) [383] and long short-term memory (LSTM) network [163] – to encourage their components to be diverse and develop an efficient projected gradient descent algorithm. Experiments on healthcare, image, and text data demonstrate that UER (1) greatly improves generalization performance; (2) better captures infrequent patterns; (3) reduces model size without sacrificing modeling power; (4) outperforms other diversity-promoting regularizers.
2.1.1 Uniform Eigenvalue Regularizer

We start with characterizing the uncorrelation among components: treating the components as random variables and measuring their covariance which is proportional to their correlation. Let $A \in \mathbb{R}^{d \times m}$ denote the component matrix where in the $k$-th column is the parameter vector $a_k$ of component $k$. Alternatively, we can take a row view (Figure 2.2b) of $A$: each component is treated as a random variable and each row vector $\tilde{a}_i^\top$ can be seen as a sample drawn from the random vector formed by the $m$ components. Let $\mu = \frac{1}{d} \sum_{i=1}^{d} \tilde{a}_i = \frac{1}{d} A^\top \mathbf{1}$ be the sample mean, where the elements of $\mathbf{1} \in \mathbb{R}^d$ are all 1. We compute the empirical covariance matrix of the components as

$$G = \frac{1}{d} \sum_{i=1}^{d} (\tilde{a}_i - \mu)(\tilde{a}_i - \mu)^\top = \frac{1}{d} A^\top A - \left(\frac{1}{d} A^\top \mathbf{1}\right)\left(\frac{1}{d} A^\top \mathbf{1}\right)^\top. \tag{2.1}$$

Imposing the constraint $A^\top \mathbf{1} = \mathbf{0}$, we have $G = \frac{1}{d} A^\top A$. Suppose $A$ is a full rank matrix and $m < d$, then $G$ is a full-rank matrix with rank $m$.

For the next step, we show that the eigenvalues of $G$ play important roles in characterizing the uncorrelation and evenness of components. We start with uncorrelation. Let $G = \sum_{k=1}^{m} \lambda_k u_k u_k^\top$ be the eigen-decomposition where $\lambda_k$ is an eigenvalue and $u_k$ is the associated eigenvector. As is well known in principle component analysis [185], an eigenvector $u_k$ of the covariance matrix $G$ represents a principal direction of the data points and the associated eigenvalue $\lambda_k$ tells the variability of points along that direction. As shown in Figure 2.3a, the larger $\lambda_k$ is, the more spread out the points along the direction $u_k$. When the eigenvectors (principal directions) are not
When the principal directions (\(u_1\) and \(u_2\)) are aligned with the coordinate axis, the magnitude of eigenvalues represents the importance of components.

aligned with the coordinate axis (as shown in Figure 2.3), the level of disparity among eigenvalues indicates the level of correlation among the \(m\) components (random variables). The more different the eigenvalues are, the higher the correlation is. As shown in Figure 2.3a, \(\lambda_1\) is about three times larger than \(\lambda_2\) and there is a high correlation along the direction \(u_1\). On the other hand, in Figure 2.3b, the two eigenvalues are close to each other and the points evenly spread out in both directions with negligible correlation. In light of this, we would utilize the uniformity among eigenvalues of \(G\) to measure how uncorrelated the components are.

Secondly, we relate the eigenvalues with the other factor of diversity: evenness. When the eigenvectors are aligned with the coordinate axis (as shown in Figure 2.4), the components are uncorrelated. In this case, we bring in evenness to measure diversity. As stated earlier, we first need to assign each component an importance score. Since the eigenvectors are in parallel to the coordinate axis, the eigenvalues reflect the variance of components. Analogous to PCA which posits that random variables with larger variance are more important, we use variance to measure importance. As shown in Figure 2.4a, component 1 has a larger eigenvalue \(\lambda_1\) and accordingly larger variability, hence is more important than component 2. According to the evenness criteria, the components are more diverse if their importance scores match, which motivates us to encourage the eigenvalues to be uniform. As shown in Figure 2.4b, the two eigenvalues are close and the two components have roughly the same variability, hence are similarly important.

To sum up, we desire to encourage the eigenvalues to be even in both cases: (1) when the eigenvectors are not aligned with the coordinate axis, they are preferred to be even to reduce the correlation of components; (2) when the eigenvectors are aligned with the coordinate axis, they are encouraged to be even such that different components contribute equally in modeling data.

Next, we discuss how to promote uniformity among eigenvalues. The basic idea is: we normalize the eigenvalues into a probability simplex and encourage the discrete distribution parameterized by the normalized eigenvalues to have small Kullback-Leibler (KL) divergence with the uniform distribution. Given the eigenvalues \(\{\lambda_k\}_{k=1}^m\), we first normalize them into a probability simplex \(\tilde{\lambda}_k = \frac{\lambda_k}{\sum_{j=1}^m \lambda_j}\) based on which we define a distribution on a discrete random variable \(X = 1, \cdots, m\) where \(p(X = k) = \tilde{\lambda}_k\). In addition, to guarantee the eigenvalues are strictly positive, we require \(A^\top A\) to be positive definite. To encourage \(\{\tilde{\lambda}_k\}_{k=1}^m\) to be uniform, we encourage the distribution \(p(X)\) to be “close” to a uniform distribution \(q(X = k) = \frac{1}{m}\).
where the “closeness” is measured using KL divergence $KL(p||q)$:

$$\sum_{k=1}^{m} \hat{\lambda}_k \log \frac{\hat{\lambda}_k}{1/m} = \frac{\sum_{k=1}^{m} \lambda_k \log \lambda_k}{\sum_{j=1}^{m} \lambda_j} - \log \sum_{j=1}^{m} \lambda_j + \log m. \quad (2.2)$$

In this equation, $\sum_{k=1}^{m} \lambda_k \log \lambda_k$ is equivalent to $\text{tr}((\frac{1}{d} A^\top A) \log((\frac{1}{d} A^\top A)))$, where $\log(\cdot)$ denotes matrix logarithm. To show this, note that $\log((\frac{1}{d} A^\top A)) = \sum_{k=1}^{m} \log(\lambda_k) u_k u_k^\top$, according to the property of matrix logarithm. Then we have $\text{tr}((\frac{1}{d} A^\top A) \log((\frac{1}{d} A^\top A)))$ equals $\text{tr}((\sum_{k=1}^{m} \lambda_k u_k u_k^\top) (\sum_{k=1}^{m} \log(\lambda_k) u_k u_k^\top))$ which equals $\sum_{k=1}^{m} \lambda_k \log \lambda_k$. According to the property of matrix trace, we have $\text{tr}(\frac{1}{d} A^\top A) = \sum_{k=1}^{m} \lambda_k$. Then the KL divergence can be turned into a diversity-promoting uniform eigenvalue regularizer (UER):

$$\frac{\text{tr}((\frac{1}{d} A^\top A) \log((\frac{1}{d} A^\top A)))}{\text{tr}(\frac{1}{d} A^\top A)} - \log \text{tr}(\frac{1}{d} A^\top A), \quad (2.3)$$

subject to $A^\top A \succ 0$ and $A^\top 1 = 0$. In principle, the “closeness” between $p$ and $q$ can be measured by other distances such as the total variation distance, Hellinger distance, etc. However, the resultant formula defined on the eigenvalues (like the one in Eq.(2.2)) is very difficult (if possible) to be transformed into a formula defined on $A$ (like the one in Eq.(2.3)). Consequently, it is very challenging to perform estimation of $A$. In light of this, we choose to use the KL divergence.

Compared with previous diversity-promoting regularizers, UER has the following benefits: (1) It measures the diversity of all components in a holistic way, rather than reducing to pairwise dissimilarities as other regularizers [29, 369, 402] do. This enables UER to capture global relations among components. (2) Unlike determinant-based regularizers [242, 429] that are sensitive to vector scaling, UER is derived from normalized eigenvalues where the normalization effectively removes scaling. (3) UER is amenable for computation. First, unlike the decorrelation regularizer [82] that is defined over data-dependent intermediate variables and thus incurs computational inefficiency, UER is directly defined on model parameters and is independent of data. Second, unlike the regularizers proposed in [29, 369] that are non-smooth, UER is a smooth function. In general, smooth functions are more amenable for deriving optimization algorithms than non-smooth functions. The dominating computation in UER is matrix logarithm. It does not substantially increase computational overhead as long as the number of components is not too large (e.g., less than 1000).

Compared with commonly-used regularizers that are not designed for promoting diversity, such as L2, L1, and the trace norm [62], UER has the following limitations. First, it is computationally heavier. It involves eigen-decomposition that incurs $O(m^3)$ time complexity where $m$ is the number of components. Second, in modern deep learning, most computation is conducted on GPUs. UER is not amenable for GPU implementation, which hinders its applicability. Third, it is a non-convex function, which renders the optimization of the regularized ML problems to be NP hard. Fourth, it is less amenable for theoretical analysis than L2, L1, and the trace norm.

We apply UER to promote diversity in ML models. Let $\mathcal{L}(A)$ denote the objective function of a model, then a UE-regularized problem can be defined as

$$\begin{align*}
\min_A & \quad \mathcal{L}(A) + \lambda \left( \frac{\text{tr}((\frac{1}{d} A^\top A) \log((\frac{1}{d} A^\top A)))}{\text{tr}(\frac{1}{d} A^\top A)} - \log \text{tr}(\frac{1}{d} A^\top A) \right) \\
\text{s.t.} & \quad A^\top 1 = 0, \ A^\top A \succ 0.
\end{align*} \quad (2.4)$$
where $\lambda$ is the regularization parameter. Similar to other diversity-promoting regularizers, UER is non-convex. Since $\mathcal{L}(A)$ in most models is non-convex, adding UER does not substantially increase difficulty for optimization.

**Connection with the von Neumann entropy** We make a connection between UER and the von Neumann entropy [39]. A matrix $M$ is referred to as a density matrix [39] if its eigenvalues are strictly positive and sum to one, equivalently, $M \succ 0$ and $\text{tr}(M) = 1$. The von Neumann entropy of $M$ is defined as $S(M) = -\text{tr}(M \log M)$, which is essentially the Shannon entropy [83] of its eigenvalues. If the covariance matrix $G$ of components is a density matrix, then we can use its von Neumann entropy to define a UER. To encourage the eigenvalues $\{\lambda_k\}_{k=1}^m$ of $G$ to be even, we directly encourage the KL divergence $\sum_{k=1}^m \lambda_k \log \frac{\lambda_k}{1/m} = \sum_{k=1}^m \lambda_k \log \lambda_k + \log m$ between the distribution parameterized by the eigenvalues (without normalization) and the uniform distribution to be small, which is equivalent to encouraging the Shannon entropy of the eigenvalues $-\sum_{k=1}^m \lambda_k \log \lambda_k$, i.e., the von Neumann entropy of $G$ to be large. Then a new UER can be defined as the negative von Neumann entropy of $G$: \[ \text{tr}((\frac{1}{d} A^\top A) \log(\frac{1}{d} A^\top A)) \] subject to the constraints: (1) $A^\top A \succ 0$; (2) $\text{tr}(\frac{1}{d} A^\top A) = 1$; (3) $A^\top 1 = 0$. This new UER is a special case of the previous one defined in Eq.(2.3) by adding the new constraint $\text{tr}(\frac{1}{d} A^\top A) = 1$.

**Connection with the von Neumann divergence** Next we make a connection between UER and the von Neumann divergence [206]. Given two positive definite matrices $X$ and $Y$, their von Neumann divergence is defined as $\text{tr}(X \log X - X \log Y - X + Y)$, which measures the closeness between the two matrices. Given two vectors $x, y \in \mathbb{R}^m$, their generalized KL divergence can be defined as $\sum_{k=1}^m x_k \log(\frac{x_k}{y_k}) - (x_k - y_k)$, which measures the closeness between two vectors. To encourage uniformity among the eigenvalues of the covariance matrix $G$, we can decrease the generalized KL divergence between these eigenvalues and an all-1 vector:

$$\sum_{k=1}^m \lambda_k \log(\frac{\lambda_k}{1}) - (\lambda_k - 1) = \text{tr}((\frac{1}{d} A^\top A) \log(\frac{1}{d} A^\top A)) - \text{tr}(\frac{1}{d} A^\top A) + m,$$

which is the von Neumann divergence between $G$ and an identity matrix. Hence, encouraging uniformity among eigenvalues can be achieved by making $G$ close to an identity matrix based on the von Neumann divergence.

**2.1.2 Case Studies**

In this section, we apply the uniform eigenvalue regularizer to promote diversity in two ML models: distance metric learning (DML) [383] and long short-term memory (LSTM) [163] networks.

**Distance metric learning** Given data pairs either labeled as “similar” or “dissimilar”, DML [92, 146, 383] aims at learning a distance metric under which similar pairs would be placed close to each other and dissimilar pairs are separated apart. The learned distance can benefit a wide range of tasks, including retrieval, clustering, and classification. Following [357], we define the distance metric between $x, y \in \mathbb{R}^d$ as $\|A^\top x - A^\top y\|_2^2$ where $A \in \mathbb{R}^{d \times m}$ is a parameter matrix.
whose column vectors are called components. Built upon the DML formulation in [367], an uniform-eigenvalue regularized DML (UE-DML) problem can be formulated as

$$
\min_A \sum_{(x,y) \in S} \|A^T x - A^T y\|_2^2 + \sum_{(x,y) \in D} \max(0, 1 - \|A^T x - A^T y\|_2^2)
+ \lambda \left( \frac{u((\frac{1}{2}A^T A) \log((\frac{1}{2}A^T A)))}{u(\frac{1}{2}A^T A)} - \log \text{tr}(\frac{1}{2}A^T A) \right)
\quad \text{s.t.} \quad A^T 1 = 0, \quad A^T A > 0,
$$

where $S$ and $D$ are the set of similar and dissimilar pairs respectively. The first and second term in the objective function encourage similar pairs to have small distances and dissimilar pairs to have large distances respectively. The learned metrics are applied for information retrieval.

Our UE-DML method is related to the matching networks (MNs) Vinyals et al. [339] used for one shot learning. First, in matching networks, distance metrics are utilized to calculate the similarity between data examples. Second, MNs are designed to achieve accurate classification performance on infrequent classes; likewise, UE-DML aims at better capturing infrequent patterns (e.g., classes). The difference is: MNs requires the data examples to have class labels while the classes in UE-DML are latent. Though in the experiments we use the ground-truth class labels to evaluate UE-DML’s performance on infrequent classes, in practice these labels are oftentimes not accessible: UE-DML can only access data pairs labeled as being similar or dissimilar, rather the class labels for individual examples.

**Long short-term memory network** The LSTM [163] network is a type of recurrent neural network, that is better at capturing long-term dependency in sequential modeling. At each time step $t$ where the input is $x_t$, there is an input gate $i_t$, a forget gate $f_t$, an output gate $o_t$, a memory cell $c_t$, and a hidden state $h_t$. The transition equations among them are

$$
i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)})
f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)})
o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)})
c_t = i_t \odot \tanh(W^{(c)}x_t + U^{(c)}h_{t-1} + b^{(c)}) + f_t \odot c_{t-1}
h_t = o_t \odot \tanh(c_t),
$$

where $W = \{W^{(s)}|s \in S = \{i, f, o, c\}\}$ and $U = \{U^{(s)}|s \in S\}$ are gate-specific weight matrices and $B = \{b^{(s)}|s \in S\}$ are bias vectors. The row vectors in $W$ and $U$ are treated as components. Let $L(W, U, B)$ denote the loss function of an LSTM network and $\mathcal{R}(\cdot)$ denote the UER (including constraints), then a UE-regularized LSTM problem can be defined as

$$
\min_{W, U, B} \quad L(W, U, B) + \lambda \sum_{s \in S} (\mathcal{R}(W^{(s)}) + \mathcal{R}(U^{(s)})).
$$

The LSTM network is applied for cloze-style reading comprehension (CSRC) [154].

### 2.1.3 A Projected Gradient Decent Algorithm

We develop a projected gradient descent (PGD) [53] algorithm to solve the UE-regularized problem in Eq.(2.4). The constraint $A^T A \succ 0$ ensures the eigenvalues of $A^T A$ are positive, such
that \(\log(A^\top A)\) is well-defined. However, it makes optimization very nasty. To address this issue, we add a small perturbation \(\epsilon I\) over \(A^\top A\) where \(\epsilon\) is a close-to-zero positive scalar and \(I\) is an identity matrix, to ensure \(\log(A^\top A + \epsilon I)\) is always well-defined. Accordingly, the constraint \(A^\top A \succ 0\) can be eliminated. The PGD algorithm iteratively performs three steps: (1) compute (sub)gradient \(\triangle A\) of the objective function; (2) update \(A\) using gradient descent: 
\[
\tilde{A} \leftarrow A - \eta \triangle A;
\]
(3) project \(\tilde{A}\) to the constraint set \(\{A | A^\top 1 = 0\}\).

In step (1), the derivative of \(\text{tr}((\frac{1}{d} A^\top A + \epsilon I) \log(\frac{1}{d} A^\top A + \epsilon I))\) is \(\frac{d}{d} A (\log(\frac{1}{d} A^\top A + \epsilon I) + I)\). To compute the logarithm of \(\frac{1}{d} A^\top A + \epsilon I\), we perform an eigen-decomposition of this matrix into \(U\Lambda U^\top\), transform \(\Lambda\) into another diagonal matrix \(\tilde{\Lambda}\) where \(\tilde{\Lambda}_{jj} = \log(\Lambda_{jj})\) and then compute \(\log(\frac{1}{d} A^\top A + \epsilon I)\) as \(U\tilde{\Lambda}U^\top\). The complexity of eigen-decomposing this \(m\)-by-\(m\) matrix is \(O(m^3)\). In our applications, \(m\) is no more than 500, so \(O(m^3)\) is not a big bottleneck. In addition, this matrix is symmetric and the symmetry can be leveraged for fast eigen-decomposition. In implementation, we use the MAGMA [7] library that supports efficient eigen-decomposition of symmetric matrices on both CPUs and GPUs. In step (3), the projection operation amounts to solving the following problem:
\[
\begin{align*}
\min_{A} & \quad \frac{1}{2} \|A - \tilde{A}\|_F^2 \\
\text{s.t.} & \quad A^\top 1 = 0.
\end{align*}
\]
According to the KKT conditions [53], we have \(A - \tilde{A} + 1\lambda^\top = 0\) and \(A^\top 1 = 0\). Solving this system of equations, we get
\[
A = (I - \frac{1}{d} 11^\top)\tilde{A},
\]
which centers the row vectors in \(\tilde{A}\) so that they have zero mean.

### 2.1.4 Evaluation

In this section, we present experimental results.

**Datasets** We used five datasets in the experiments: an electronic health record dataset MIMIC-III [184]; two image datasets Stanford-Cars [202] and Caltech-UCSD-Birds [358]; two question answering (QA) datasets CNN and DailyMail [154]. The first three were used for DML and the last two for LSTM. Their statistics are summarized in Table 2.1.

MIMIC-III contains 58K hospital admissions of patients who stayed within the intensive care units at the Beth Israel Deaconess Medical Center between 2001 and 2012. Each admission has a primary diagnosis (a disease), which acts as the class label of this admission. There are 2833 unique diseases. We extracted 7207-dimensional features: (1) 2 dimensions from demographics, including age and gender; (2) 5300 dimensions from clinical notes, including 5000-dimensional bag-of-words (weighted using \(tf-idf\)) and 300-dimensional Word2Vec [256]; (3) 1905-dimensions from lab tests where the zero-order, first-order, and second-order temporal features were extracted for each of the 635 lab items. In the extraction of bag-of-words features from clinical notes, we removed stop words, then counted the document frequency (DF) of the remaining words. Then we selected 5000 words with the largest DFs to form the dictionary. Based on this dictionary, we extracted \(tf-idf\) features. In the extraction of Word2Vec features,
we trained a 300-dimensional embedding vector for each word using an open source tool\footnote{https://code.google.com/archive/p/word2vec/}. To represent a clinical note, we averaged the embeddings of all words in this note. In lab tests, there are 635 test items in total. Each item is tested at different time points for each admission. For an item, we extracted three types of temporal features: (1) zero-order: averaging the values of this item measured at different time points; (2) first-order: taking the difference of values at every two consecutive time points \( t \) and \( t - 1 \), and averaging these differences; (3) second-order: for the sequence of first-order differences generated in (2), taking the difference (called second-order difference) of values at every two consecutive time points \( t \) and \( t - 1 \), and averaging these second-order differences. If an item is missing in an admission, we set the zero-order, first-order, and second-order feature values to 0. For the two image datasets, we used the VGG16\footnote{https://code.google.com/archive/p/word2vec/} convolutional neural network trained on the ImageNet\footnote{https://code.google.com/archive/p/word2vec/} dataset to extract features, which are the 4096-dimensional outputs of the second fully-connected layer. For MIMIC-III, Stanford-Cars, and Caltech-UCSD-Birds, the features were normalized using min-max normalization along each dimension. We used PCA to reduce the feature dimension to 1000. In the two QA datasets, each data example consists of a passage, a question, and an answer. The question is a cloze-style\footnote{https://code.google.com/archive/p/word2vec/} task where an entity is replaced by a placeholder and the goal is to infer this missing entity (answer) from all the possible entities appearing in the passage. For Stanford-Cars, CNN, and DailyMail, we used a single train/test split specified by the data providers; for the other two, five random splits were performed and the results were averaged over the five runs.

**Experimental setup** In the DML experiments, two data examples were labeled as similar if belonging to the same class and dissimilar if otherwise. The learned distance metrics were applied for retrieval whose performance was evaluated using precision@K: among the top \( K \) retrieved examples, what is the percentage of samples that share the same class label with the query example? We compared with two sets of regularizers: (1) diversity-promoting regularizers based on determinant of covariance matrix (DCM)\footnote{https://code.google.com/archive/p/word2vec/}, cosine similarity (CS)\footnote{https://code.google.com/archive/p/word2vec/}, determinantal point process (DPP)\footnote{https://code.google.com/archive/p/word2vec/}, InCoherence (IC)\footnote{https://code.google.com/archive/p/word2vec/}, mutual angles (MA)\footnote{https://code.google.com/archive/p/word2vec/}, and decorrelation (DC)\footnote{https://code.google.com/archive/p/word2vec/}; (2) regularizers that are designed for other purposes, including L2 norm for small norm, L1 norm for sparsity, low-rankness\footnote{https://code.google.com/archive/p/word2vec/}, and dropout\footnote{https://code.google.com/archive/p/word2vec/}. All these regularizers were applied to the same DML formulation (Eq.\ref{eq:dml} without the UER regularizer). In addition, we compared with the vanilla Euclidean distance (EUC) and other distance learning methods including information theoretic metric learning (ITML)\footnote{https://code.google.com/archive/p/word2vec/}, logistic discriminant metric learning (LDML)\footnote{https://code.google.com/archive/p/word2vec/}, and geometric mean metric learning (GMML)\footnote{https://code.google.com/archive/p/word2vec/}. We used 5-fold cross validation to perform the experiments.
Table 2.2: Precision@10 (%) on three datasets. The Cars dataset has a single train/test split, hence the standard error is 0. UE-DML is our method. On the second panel (EUC, etc.) are well established or state of the art distance metric learning baselines. On the the third panel (L2-DML, etc.) and the fourth panel (DCM-DML, etc.) are DML methods regularized by non-diversity regularizers and previously proposed diversity-promoting regularizers.

validation to tune the regularization parameter in \( \{10^{-5}, 10^{-4}, \cdots, 10^5\} \) and the number of components in \( \{50, 100, 200, \cdots, 500\} \). The best tuned regularization parameters of UER are: 0.001 for MIMIC, 0.01 for Cars and Birds. The best tuned component numbers are: 200 for MIMIC, 100 for Cars, and 200 for Birds. The learning rate of the PGD algorithm was set to 0.001.

In the LSTM experiments, the network architecture and experimental settings followed that in the BIDirectional Attention Flow (BIDAF) model, which consists of the following layers: character embedding, word embedding, contextual embedding, attention flow, modeling, and output. The contextual and modeling layers are based on LSTM. In the character embedding based on convolutional neural networks, 100 1D filters were used, each with a width of 5. The hidden state size was set to 100. AdaDelta was used for optimization with a minibatch size of 48. Dropout with probability 0.2 was used for all LSTM layers. The model was trained for 8 epochs with an early stop when the validation accuracy started to drop. We compared UER with other diversity-promoting regularizers including DCM, CS, DPP, IC, MA, and DC.

**Results** Table 2.2 shows the retrieval precision \( (K = 10) \) on three datasets, where we observe:

1. UE-DML achieves much better precision than DML, proving that UER is an effective regularizer in improving generalization performance;
2. UER outperforms other diversity-promoting regularizers possibly due to its capability of capturing global relations among all components and insensitivity to vector scaling;
3. diversity-promoting regularizers perform better than other
types of regularizers such as L2, L1, low rank, and dropout, demonstrating the efficacy of inducing diversity; (4) UE-DML outperforms other popular distance learning methods such as ITML, LDML, and GMML.

Table 2.3 shows the number of components that achieves the precision in Table 2.2. Compared with DML, UE-DML uses much fewer components to achieve better precision. For example, on the Cars dataset, UE-DML achieves a 58.2% precision with 100 components. In contrast, with more components (300), DML achieves a much lower precision (53.1%). This demonstrates that by encouraging the components to be diverse, UER is able to reduce model size without sacrificing modeling power. UER encourages equal “importance” among components such that each component plays a significant role in modeling data. As a result, it suffices to use a small number of components to achieve larger modeling power. Compared with other diversity-promoting regularizers, UER achieves better precision with fewer components, demonstrating its ability to better promote diversity.

Next, we verify whether “diversifying” the components in DML can better capture infrequent patterns. In the MIMIC-III dataset, we consider diseases as patterns and consider a disease as “frequent” if more than 1000 hospital admissions are diagnosed with this disease and “infrequent” if otherwise. Table 2.4 shows the retrieval precision on frequent diseases and infrequent diseases. As can be seen, compared with the baselines, UE-DML achieves more improvement on infrequent diseases than on frequent diseases. This indicates that by encouraging the components to diversely spread out, UER is able to better capture infrequent patterns (diseases in this case) without compromising the performance on frequent patterns. On infrequent diseases, UE-DML outperforms other diversity-promoting methods, showing the advantage of UER over other diversity-promoting regularizers. To further verify this, we select 3 most frequent diseases (hypertension, AFib, CAD) and randomly select 5 infrequent ones (helicobacter pylori, acute cholecystitis, joint pain-shoulder, dysarthria, pressure ulcer), and show the precision@10 on each individual disease in Table 2.5. As can be seen, on the five infrequent diseases, UE-DML achieves higher precision than baselines while on the three frequent diseases UE-DML achieves
<table>
<thead>
<tr>
<th>Method</th>
<th>Frequent</th>
<th>Infrequent</th>
</tr>
</thead>
<tbody>
<tr>
<td>DML</td>
<td>77.6 ± 0.2</td>
<td>64.2 ± 0.3</td>
</tr>
<tr>
<td>EUC</td>
<td>58.7 ± 0.1</td>
<td>57.6 ± 0.2</td>
</tr>
<tr>
<td>ITML [92]</td>
<td>74.2 ± 0.6</td>
<td>61.3 ± 0.3</td>
</tr>
<tr>
<td>LDML [146]</td>
<td>76.1 ± 0.8</td>
<td>62.3 ± 0.9</td>
</tr>
<tr>
<td>GMML [405]</td>
<td>75.9 ± 0.1</td>
<td>63.5 ± 0.4</td>
</tr>
<tr>
<td>L2-DML</td>
<td>77.5 ± 0.3</td>
<td>65.4 ± 0.1</td>
</tr>
<tr>
<td>L1-DML</td>
<td>77.4 ± 0.5</td>
<td>64.8 ± 0.8</td>
</tr>
<tr>
<td>LowRank-DML [290]</td>
<td>77.7 ± 0.5</td>
<td>64.0 ± 0.8</td>
</tr>
<tr>
<td>Dropout-DML [282]</td>
<td>78.1 ± 0.2</td>
<td>64.9 ± 0.4</td>
</tr>
<tr>
<td>DCM-DML [242]</td>
<td>77.9 ± 0.4</td>
<td>66.8 ± 0.2</td>
</tr>
<tr>
<td>CS-DML [402]</td>
<td>78.0 ± 0.5</td>
<td>66.2 ± 0.7</td>
</tr>
<tr>
<td>DPP-DML [429]</td>
<td>77.3 ± 0.2</td>
<td>69.1 ± 0.5</td>
</tr>
<tr>
<td>IC-DML [29]</td>
<td>78.5 ± 0.3</td>
<td>67.4 ± 0.2</td>
</tr>
<tr>
<td>MA-DML [367]</td>
<td>76.8 ± 0.2</td>
<td>68.4 ± 0.4</td>
</tr>
<tr>
<td>DC-DML [82]</td>
<td>77.1 ± 0.1</td>
<td>65.3 ± 0.1</td>
</tr>
<tr>
<td>UE-DML</td>
<td>78.3 ± 0.3</td>
<td>70.7 ± 0.4</td>
</tr>
</tbody>
</table>

Table 2.4: Precision@10 (%) on frequent and infrequent diseases of the MIMIC-III dataset.

We empirically verified whether UER can promote uncorrelation and evenness. Given \( m \) component vectors, we computed the empirical correlation (cosine similarity) of every two vectors, then averaged these pairwise correlation scores to measure the overall correlation of the \( m \) vectors. We performed the study by learning distance metrics that have 200 components, on the MIMIC-III dataset. The average correlation under unregularized DML and UE-DML is 0.73 and 0.57 respectively. This shows that UER can reduce correlation. To measure evenness, we first measured the “importance” of components. For each component with a parameter vector \( a \), we projected the training examples \( \{x_i\}_{i=1}^N \) onto \( a \): \( \{x_i^T a\}_{i=1}^N \), then used the variance of \( \{x_i^T a\}_{i=1}^N \) to measure the importance of this component. After that, we mapped these importance scores into a probabilistic simplex using softmax. Finally, the evenness was measured by the KL divergence between the discrete distribution parameterized by these probabilities and a uniform distribution. A smaller KL divergence indicates larger evenness. On MIMIC-III with 200 components, the KL divergence under unregularized DML and UE-DML is 3.54 and 2.92 respectively. This suggests that our regularizer is able to encourage evenness.

Table 2.6 shows the runtime taken by DML methods to reach convergence. Compared with unregularized DML, UE-DML does not increase the training time substantially. The relative increase is 11.2\% on MIMIC, 15.4\% on Cars, and 13.9\% on Birds. The runtime of UE-DML is close to DML regularized by other diversity-promoting regularizers.

In the LSTM experiments, Table 2.7 shows state of the art accuracy on the two QA datasets. Compared with the original BIDAF [296], our method UE-BIDAF achieves better accuracy, further demonstrating UER’s ability to improve generalization performance. Besides, UER outperforms other regularizers.
result, a family of nonconvex BMD regularizers are defined. Then we perform convex relaxations easy [379]. Similar to [74, 122, 342, 347, 367], we use near-orthogonality to represent far from the global optimal.

ML problems are nonconvex where achieving the global optimal is NP-hard. To achieve a better local optima, the algorithm needs to rerun multiple times, each with a different random initialization of the model, and selects the best solution among them, which incurs substantial computational overhead. Even so, the obtained optimal is a local (therefore inferior) one that might be far from the global optimal.

In this section, we design new diversity-promoting regularizers that make convex relaxation easy [379]. Similar to [74, 122, 342, 347, 367], we use near-orthogonality to represent “diversity”: projection vectors are more “diverse” if they are closer to orthogonality. Near-orthogonality is promoted by encouraging the Gram matrix of vectors to be close to an identity matrix and the “closeness” is measured using the Bregman matrix divergence (BMD) [206]. As a result, a family of nonconvex BMD regularizers are defined. Then we perform convex relaxations of them by exploring the properties of eigenvalues. We apply the convex BMD regularizers to distance metric learning [353] and develop efficient proximal gradient descent [270] algorithms.

Table 2.5: Precision@10 (%) on three frequent and five infrequent diseases. The number next to a disease ID is its frequency.

<table>
<thead>
<tr>
<th>Disease</th>
<th>MIMIC</th>
<th>Cars</th>
<th>Birds</th>
</tr>
</thead>
<tbody>
<tr>
<td>DML</td>
<td>20.5</td>
<td>9.1</td>
<td>10.1</td>
</tr>
<tr>
<td>DCM-DML [242]</td>
<td>22.3</td>
<td>10.9</td>
<td>11.7</td>
</tr>
<tr>
<td>CS-DML [402]</td>
<td>20.9</td>
<td>9.7</td>
<td>10.5</td>
</tr>
<tr>
<td>DPP-DML [429]</td>
<td>22.6</td>
<td>10.6</td>
<td>11.2</td>
</tr>
<tr>
<td>IC-DML [29]</td>
<td>21.1</td>
<td>9.7</td>
<td>10.5</td>
</tr>
<tr>
<td>MA-DML [367]</td>
<td>21.3</td>
<td>9.4</td>
<td>10.6</td>
</tr>
<tr>
<td>DC-DML [82]</td>
<td>21.7</td>
<td>10.1</td>
<td>10.8</td>
</tr>
<tr>
<td>UE-DML</td>
<td>22.8</td>
<td>10.5</td>
<td>11.5</td>
</tr>
</tbody>
</table>

Table 2.6: Average runtime (hours) of DML methods regularized by different diversity-promoting regularizers.

<table>
<thead>
<tr>
<th>Method</th>
<th>Average Runtime ((hours))</th>
</tr>
</thead>
<tbody>
<tr>
<td>DML</td>
<td>80.2 ± 0.5 79.4 ± 0.8 80.9 ± 0.7</td>
</tr>
<tr>
<td>EUC</td>
<td>66.4 ± 1.2 69.6 ± 1.2 61.8 ± 0.4</td>
</tr>
<tr>
<td>ITML</td>
<td>75.4 ± 1.0 76.3 ± 0.9 79.3 ± 0.9</td>
</tr>
<tr>
<td>LDML [146]</td>
<td>77.0 ± 0.8 75.7 ± 1.0 78.1 ± 0.6</td>
</tr>
<tr>
<td>GMML [405]</td>
<td>76.3 ± 0.4 78.7 ± 0.8 79.8 ± 0.8</td>
</tr>
<tr>
<td>L2-DML</td>
<td>81.0 ± 0.5 79.3 ± 0.8 77.6 ± 0.4</td>
</tr>
<tr>
<td>L1-DML</td>
<td>78.2 ± 1.0 79.9 ± 1.1 80.8 ± 1.2</td>
</tr>
<tr>
<td>LowRank-DML [250]</td>
<td>79.6 ± 0.4 79.7 ± 0.5 75.4 ± 0.5</td>
</tr>
<tr>
<td>Dropout-DML [222]</td>
<td>81.6 ± 0.5 78.7 ± 0.7 80.7 ± 0.5</td>
</tr>
<tr>
<td>DCM-DML [242]</td>
<td>77.9 ± 1.0 77.3 ± 0.9 80.3 ± 1.2</td>
</tr>
<tr>
<td>CS-DML [402]</td>
<td>80.0 ± 0.5 80.3 ± 0.7 80.8 ± 0.6</td>
</tr>
<tr>
<td>DPP-DML [429]</td>
<td>79.8 ± 0.8 77.6 ± 0.2 77.4 ± 0.7</td>
</tr>
<tr>
<td>IC-DML [29]</td>
<td>78.8 ± 1.3 79.2 ± 1.1 77.0 ± 0.8</td>
</tr>
<tr>
<td>MA-DML [367]</td>
<td>77.3 ± 1.1 80.1 ± 1.0 81.0 ± 0.7</td>
</tr>
<tr>
<td>DC-DML [82]</td>
<td>80.7 ± 0.5 78.8 ± 0.7 80.5 ± 1.1</td>
</tr>
<tr>
<td>UE-DML</td>
<td>81.4 ± 0.9 82.4 ± 0.8 80.5 ± 0.4</td>
</tr>
</tbody>
</table>

2.2 Convex Diversity-promoting Regularizers

The UE regularizer is nonconvex and is difficult to be convexified. As a result, the UE-regularized ML problems are nonconvex where achieving the global optimal is NP-hard. To achieve a better local optima, the algorithm needs to rerun multiple times, each with a different random initialization of the model, and selects the best solution among them, which incurs substantial computational overhead. Even so, the obtained optimal is a local (therefore inferior) one that might be far from the global optimal.
Table 2.7: Accuracy (%) on the two QA datasets. On the second panel (BIDAF, etc.), we compare different diversity-promoting regularizers. On the first panel (Kadlec et al., etc.) and the third panel (Dhingra et al., etc) are other state-of-the-art baselines.

The algorithms only run once with a single initialization, hence are more efficient than existing nonconvex methods. Since the global optimal can be achieved, our methods can potentially yield more effective diversity-biased distance metrics. We apply the learned distance metrics for information retrieval on healthcare, texts, images, and sensory data. Compared with nonconvex baselines, our methods achieve: (1) higher computational efficiency; (2) better retrieval performance with fewer projection vectors; (3) better performance on infrequent classes.

2.2.1 Nonconvex Bregman Matrix Divergence Regularizers

We begin with defining nonconvex regularizers based on the Bregman matrix divergence, then discuss how to convexify them. We first introduce another measure of “diversity” — near-orthogonality [379]: the component vectors are deemed to be more diverse if they are closer to being orthogonal. To encourage near-orthogonality between two vectors $a_i$ and $a_j$, one way is to make their inner product $a_i^\top a_j$ close to zero and their $\ell_2$ norm $\|a_i\|_2$, $\|a_j\|_2$ close to one. For $m$ vectors $A \in \mathbb{R}^{m \times d}$, their near-orthogonality can be achieved in the following manner: computing the Gram matrix $G$ where $G_{ij} = a_i^\top a_j$, then encouraging $G$ to be close to an identity matrix. Off the diagonal of $G$ and $I$ are $a_i^\top a_j$ and zero respectively. On the diagonal of $G$ and $I$ are $\|a_i\|_2^2$ and one respectively. Making $G$ close to $I$ effectively encourages $a_i^\top a_j$ to be close to zero and $\|a_i\|_2$ close to one, which therefore encourages $a_i$ and $a_j$ to get close to being orthogonal.
One straightforward distance between $G$ and $I$ is based on the squared Frobenius norm (SFN): $\|G - I\|^2_2$. The SFN can be factorized into pairwise inner products: $\sum_{i=1}^{m} \sum_{j \neq i} (a_i, a_j)^2 + \sum_{i=1}^{m} (\|a_i\|_2 - 1)^2$. As a result, it measures orthogonality of functions in a pairwise manner. We conjecture that measuring orthogonality in a global manner is more desirable (the conjecture is validated in experiments). To achieve this goal, we resort to another measure: Bregman matrix divergence (BMD) [100]. Let $S^n$ denote real symmetric $n \times n$ matrices. Given a strictly convex, differentiable function $\phi : S^n \to \mathbb{R}$, the BMD is defined as $D_{\phi}(X, Y) = \phi(X) - \phi(Y) - \langle \nabla \phi(Y)^\top (X - Y) \rangle$, where $\langle A \rangle$ denotes the trace of matrix $A$. Different choices of $\phi(X)$ lead to different divergences. If $\phi(X) = \text{tr}(X \log X - X)$, where $\log X$ denotes the matrix logarithm of $X$, the divergence becomes $D_{\text{vnd}}(X, Y) = \text{tr}(X \log X - X \log Y - X + Y)$, which is referred to as von Neumann divergence (VND) [336]. If $\phi(X) = -\log \det X$ where $\det(X)$ denotes the determinant of $X$, we get the log-determinant divergence (LDD) [206]: $D_{\text{ldd}}(X, Y) = \text{tr}(XY^{-1}) - \log \det(XY^{-1}) - n$. Interestingly, SFN is a special case of BMD when $\phi(X) = \|X\|^2_2$. To encourage near-orthogonality among components, we encourage the BMD between their Gram matrix $AA^\top$ and an identity matrix $I$ to be small, which results in a family of BMD regularizers: $\Omega_{\phi}(A) = D_{\phi}(AA^\top, I)$. $\Omega_{\phi}(A)$ can be specialized to different instances, according to the choices of $D_{\phi}(\cdot, \cdot)$. Under VND, $\Omega_{\phi}(A)$ becomes

$$\Omega_{\text{vnd}}(A) = \text{tr}(AA^\top \log(AA^\top) - AA^\top) + m. \quad (2.11)$$

Under LDD, $\Omega_{\phi}(A)$ becomes

$$\Omega_{\text{ldd}}(A) = \text{tr}(AA^\top) - \log \det(AA^\top) - m. \quad (2.12)$$

Under SFN, $\Omega_{\phi}(A)$ becomes

$$\Omega_{\text{sf}}(A) = \|AA^\top - I\|^2_F. \quad (2.13)$$

Different from the SFN regularizer, VND and LDD do not admit a pairwise factorization, and hence allow one to measure orthogonality globally. Unlike DPP [204], LDD utilizes an additional term $\text{tr}(G) = \sum_{i=1}^{m} \|a_i\|_2^2$ to control the magnitude of vectors, and thus avoiding DPP’s sensitivity to scaling. Similarly, VND and SFN are also insensitive to scaling since they encourage $|a_i|$ to be close to one.

Applying these regularizers to ML models, e.g., distance metric learning (DML, Section 2.1.2), we define the following BMD-regularized DML (BMD-DML) problem:

$$\min_A \frac{1}{|S|} \sum_{(x,y) \in S} \|Ax - Ay\|^2_2 + \frac{1}{|D|} \sum_{(x,y) \in D} \max(0, 1 - \|Ax - Ay\|^2_2) + \lambda \Omega_{\phi}(A), \quad (2.14)$$

which is nonconvex.

### 2.2.2 Convex Bregman Matrix Divergence Regularizers

Next, we discuss how to relax the nonconvex BMD regularizers into convex functions. The relaxations are based on the properties of eigenvalues. Given a full-rank matrix $A \in \mathbb{R}^{m \times d}$ ($m < d$), we know that $AA^\top \in \mathbb{R}^{m \times m}$ is a full-rank matrix with $m$ positive eigenvalues (denoted by $\lambda_1, \ldots, \lambda_m$) and $A^\top A \in \mathbb{R}^{d \times d}$ is a rank-deficient matrix with $d - m$ zero eigenvalues and $m$ positive eigenvalues that equal $\lambda_1, \ldots, \lambda_m$. For a general positive definite matrix
$Z \in \mathbb{R}^{m \times m}$ whose eigenvalues are $\gamma_1, \ldots, \gamma_m$, we have $\|Z\|_F^2 = \sum_{j=1}^m \gamma_j^2$, $\text{tr}(Z) = \sum_{j=1}^m \gamma_j$, and $\log \det Z = \sum_{j=1}^m \log \gamma_j$. Next, we leverage these facts to seek convex approximations of the BMD regularizers.

**Convex SFN regularizer** The eigenvalues of $AA^T - I_m$ are $\lambda_1 - 1, \ldots, \lambda_m - 1$ and those of $A^TA - I_d$ are $\lambda_1 - 1, \ldots, \lambda_m - 1, -1, \ldots, -1$. According to the fact $\|Z\|_F^2 = \sum_{j=1}^m \gamma_j^2$, we have $\|A^TA - I_d\|_F^2 = \sum_{j=1}^m (\lambda_j - 1)^2 + \sum_{j=m+1}^d (-1)^2 = \|AA^T - I_m\|_F^2 + d - m$. Let $M$ denote $A^TA$, then the SFN regularizer $\|AA^T - I_m\|_F^2$ equals to $\|A^TA - I_d\|_F^2 - d + m = \|M - I_d\|_F^2 - d + m$, where $m = \text{rank}(A^TA) = \text{rank}(M)$. It is well-known that the trace norm of a matrix is a convex envelope of its rank [311]. We use $\text{tr}(M)$ to approximate $\text{rank}(M)$ and get $\|AA^T - I_m\|_F^2 \approx \|M - I_d\|_F^2 + \text{tr}(M) - d$, where the right hand side is a convex function. Dropping the constant, we get the convex SFN (CSFN) regularizer defined over $M$

$$\hat{\Omega}_{sfn}(M) = \|M - I_d\|_F^2 + \text{tr}(M). \quad (2.15)$$

**Convex VND regularizer** Given $AA^T = U\Lambda U^T$ where $\Lambda_{jj} = \lambda_j$, according to the property of matrix logarithm, we have $\log(\Lambda UU^T) = U\log(\Lambda)U^T$ where $\log(\Lambda)_{jj} = \log \Lambda_{jj}$. Then $(\Lambda_{jj} \log(\Lambda)_{jj})_{jj} = \log(\Lambda_{jj}) + \log(\Lambda_{jj})_{jj}$, we have $\Omega_{vnd}(A) = \sum_{j=1}^m (\lambda_j \log \Lambda_{jj} - \Lambda_{jj}) + m$. For $A^TA + \epsilon I_d$, where $\epsilon > 0$ is a small scalar and the matrix’s eigenvalues are $\lambda_1 + \epsilon, \ldots, \lambda_m + \epsilon, \epsilon, \ldots, \epsilon$, we have

$$D_{vnd}(A^TA + \epsilon I_d, I_d) = \text{tr}((A^TA + \epsilon I_d) \log(A^TA + \epsilon I_d) - (A^TA + \epsilon I_d)) + d$$
$$= \sum_{j=1}^m (\lambda_j + \epsilon) \log(\lambda_j + \epsilon) - (\lambda_j + \epsilon) + \sum_{j=m+1}^d \epsilon \log(\epsilon - \epsilon) + d$$
$$= \sum_{j=1}^m (\lambda_j + \epsilon) \log(\lambda_j + \epsilon) - (\lambda_j + \epsilon) + (d - m) \epsilon \log(\epsilon - \epsilon) + d$$
$$= \sum_{j=1}^m \lambda_j \log(\lambda_j + \epsilon) - \lambda_j + \epsilon \log(\lambda_j + \epsilon) - (\lambda_j + \epsilon) + (d - m) \epsilon \log(\epsilon - \epsilon) + d$$
$$= \Omega_{vnd}(A) - m + \sum_{j=1}^m \lambda_j \log(\lambda_j + \epsilon) - (\lambda_j + \epsilon) + (d - m) \epsilon \log(\epsilon - \epsilon) + d. \quad (2.16)$$

Since $\epsilon$ is small, we have $\log(1 + \frac{\epsilon}{\lambda_j}) \approx \frac{\epsilon}{\lambda_j}$. Then $\lambda_j \log(1 + \frac{\epsilon}{\lambda_j}) \approx \epsilon$ and the last line in the above equation can be approximated with $\Omega_{vnd}(A) - m + d + O(\epsilon)$, and therefore

$$\Omega_{vnd}(A) \approx D_{vnd}(A^TA + \epsilon I_d, I_d) + m - d, \quad (2.17)$$

where $O(\epsilon)$ is close to zero since $\epsilon$ is small, and can be dropped. Replacing $A^TA$ with $M$, approximating $m$ with $\text{tr}(M)$ and dropping the constant $d$, we get the convex VND (CVND) regularizer:

$$\hat{\Omega}_{vnd}(M) \quad (M + \epsilon I_d, I_d) \Rightarrow \text{tr}((M + \epsilon I_d) \log(M + \epsilon I_d)), \quad (2.18)$$

whose convexity is shown in [264].

**Convex LDD regularizer** Since $\text{tr}(M) = \sum_{j=1}^m \lambda_j$ and $\log \det M = \sum_{j=1}^m \log \lambda_j$, we have $\Omega_{ldd}(A) = \sum_{j=1}^m \lambda_j - m$ and $D_{ldd}(A^TA + \epsilon I_d, I_d) = \sum_{j=1}^m \lambda_j + d\epsilon - (d - m)$.
\[ m \log \epsilon - \sum_{j=1}^{m} \log(\lambda_j + \epsilon). \] Further,

\[
D_{\text{ldd}}(A^\top A + \epsilon I_d, I_d)
= \sum_{j=1}^{m} \lambda_j + d \epsilon - (d - m) \log \epsilon - \sum_{j=1}^{m} \log(\lambda_j + \epsilon)
= \sum_{j=1}^{m} \lambda_j + d \epsilon - (d - m) \log \epsilon - \sum_{j=1}^{m} (\log \lambda_j + \log(1 + \frac{\epsilon}{\lambda_j}))
\approx \sum_{j=1}^{m} (\lambda_j - \log \lambda_j) + m \log \epsilon - \epsilon \sum_{j=1}^{m} \frac{1}{\lambda_j} + d \epsilon - d \log \epsilon
= \Omega_{\text{ldd}}(A) + m + m \log \epsilon + O(\epsilon) - d \log \epsilon.
\] (2.19)

Dropping \( O(\epsilon) \), we obtain

\[
\Omega_{\text{ldd}}(A) = D_{\text{ldd}}(A^\top A + \epsilon I_d, I_d) - (\log \epsilon + 1)m + d \log \epsilon.
\] (2.20)

Replacing \( A^\top A \) with \( M \), approximating \( m \) with \( \text{tr}(M) \) and discarding constants, we obtain the convex LDD (CLDD) regularizer:

\[
\tilde{\Omega}_{\text{ldd}}(M) = D_{\text{ldd}}(M + \epsilon I_d, I_d) - (1 + \log \epsilon)\text{tr}(M) \propto -\log\det(M + \epsilon I_d) + (\log \frac{1}{\epsilon})\text{tr}(M),
\] (2.21)

where the convexity of \( \log\det(M + \epsilon I_d) \) is proved in [53]. In [92, 280], an information theoretic regularizer based on the log-determinant divergence \( D_{\text{ldd}}(M, I) = -\log\det(M) + \text{tr}(M) \) is applied to encourage the Mahalanobis matrix to be close to an identity matrix. This regularizer requires \( M \) to be full rank while our convex LDD regularizer encourages \( M \) to be low-rank by associating a large weight \( \log \frac{1}{\epsilon} \) to the trace norm \( \text{tr}(M) \). Since \( M = A^\top A \), reducing the rank of \( M \) effectively reduces the number of components in \( A \).

**Approximation errors** We discuss the errors of convex approximation, which are generated from two sources: one is the approximation of \( \Omega_\phi(A) \) using \( D_\phi(A^\top A + \epsilon I_d, I_d) \) where the error is controlled by \( \epsilon \) and can be arbitrarily small (by setting \( \epsilon \) to be very small); the other is the approximation of the matrix rank using the trace norm. Though the error of the second approximation can be large, it has been both empirically and theoretically [62] demonstrated that decreasing the trace norm can effectively reduce rank. As validated in our experiments, though bearing approximation errors, these convex regularizers yield better performance than the original non-convex ones.

**DML with convex BMD regularization** Given these convex BMD (CBMD) regularizers (denoted by \( \hat{\Omega}_\phi(M) \) uniformly), we can relax the nonconvex BMD-DML problems into convex CBMD-DML formulations by replacing \( \|Ax - Ay\|_2^2 \) with \( (x - y)^\top M(x - y) \) and replacing the nonconvex BMD regularizers \( \Omega_\phi(A) \) with \( \hat{\Omega}_\phi(M) \):

\[
\min_M \frac{1}{|S|} \sum_{(x,y) \in S} (x - y)^\top M(x - y) + \frac{1}{|D|} \sum_{(x,y) \in D} \max(0, 1 - (x - y)^\top M(x - y)) + \lambda \hat{\Omega}_\phi(M)
\text{s.t.} \quad M \succeq 0.
\] (2.22)

This convex problem facilitates optimization: the global optimal is guaranteed to be achievable.
2.2.3 A Proximal Gradient Descent Algorithm

We use the stochastic proximal subgradient descent algorithm \([270]\) to solve the CBMD-DML problems. The algorithm iteratively performs the following steps until convergence: (1) randomly sampling a mini-batch of data pairs, computing the subgradient \(\Delta M\) of the data-dependent loss (the first and second term in the objective function) defined on the mini-batch, then performing a subgradient descent update: \(\tilde{M} = M - \eta \Delta M\), where \(\eta\) is a small steps size; (2) applying proximal operators associated with the regularizers \(\tilde{\Omega}_\phi(M)\) to \(\tilde{M}\). In step (1), the gradient of CBMD is \(\log(M + \epsilon I_d) + I_d\). To compute \(\log(M + \epsilon I_d)\), we first perform an eigen-decomposition: \(M + \epsilon I_d = U \Lambda U^T\); then take the log of every eigenvalue in \(\Lambda\), resulting in a new diagonal matrix \(\tilde{\Lambda}\); finally compute \(\log(M + \epsilon I_d)\) as \(U \tilde{\Lambda} U^T\). In the CLDD regularizer, the gradient of \(\log(M + \epsilon I_d)\) is \((M + \epsilon I_d)^{-1}\), which can be computed by eigen-decomposition as well.

In step (2), the proximal operators associated with the regularizer \(\tilde{\Omega}_\phi(M)\) are derived by minimizing \(\frac{1}{2\eta} \|M - \tilde{M}\|_F^2 + \lambda \tilde{\Omega}_\phi(M)\) subject to \(M \succeq 0\). Let \(\lambda_j\) and \(x_j\) be the eigenvalues of \(M\) and \(M\), respectively. For any \(\lambda \in \{\lambda_j \}_{j=1}^d\) and \(x \in \{x_j \}_{j=1}^d\), this problem can be equivalently written as:

\[
\min_{x_j} \frac{1}{2\eta} \sum_{j=1}^d (x_j - \lambda_j)^2 + \sum_{j=1}^d h(x_j) \quad \text{s.t.} \quad \forall j = 1, \ldots, d, \quad x_j \geq 0,
\]

(2.23)

where \(h(x_j)\) is a regularizer-specific scalar function. Further, this problem can be decomposed into \(d\) independent problems: (a) \(\min_{x_j} f(x_j) = \frac{1}{2\eta} (x_j - \lambda_j)^2 + \lambda h(x_j)\) subject to \(x_j \geq 0\), for \(j = 1, \ldots, d\), which can be solved individually.

**SFN** For SFN where \(\tilde{\Omega}_\phi(M) = \|M - I_d\|_F^2 + \text{tr}(M)\) and \(h(x_j) = (x_j - 1)^2 + x_j\), problem (a) is simply a quadratic programming problem. The optimal solution is \(x^*_j = \max(0, \frac{\lambda_j + \eta \lambda}{1 + 2\eta \lambda})\).

**VND** For VND where \(\tilde{\Omega}_\phi(M) = \text{tr}((M + \epsilon I_d) \log(M + \epsilon I_d))\) and \(h(x_j) = (x_j + \epsilon) \log(x_j + \epsilon)\), by taking the derivative of the objective function \(f(x_j)\) in problem (a) w.r.t. \(x_j\) and setting the derivative to zero, we get \(\eta \lambda \log(x_j + \epsilon) + x_j + \eta \lambda - \lambda_j = 0\). The root of this equation is: \(\eta \lambda \omega\left(\frac{\epsilon - \eta \lambda + \lambda_j}{\eta \lambda}\right) - \epsilon\), where \(\omega(\cdot)\) is the Wright omega function \([137]\). If this root is negative, then the optimal \(x_j = 0\); if this root is positive, then the optimal \(x_j\) could be either this root or 0. We pick the one that yields the lowest \(f(x_j)\). Formally, \(x^*_j = \arg\min_{x_j} f(x_j)\), where \(x \in \{\max(\eta \lambda \omega\left(\frac{\epsilon - \eta \lambda + \lambda_j}{\eta \lambda}\right) - \epsilon, 0\), 0\}\).

**LDD** For LDD where \(\tilde{\Omega}_\phi(M) = -\log \det(M + \epsilon I_d) + (\log \frac{1}{\epsilon}) \text{tr}(M)\) and \(h(x_j) = -\log(x_j + \epsilon) + x_j \log \frac{1}{\epsilon}\), by taking the derivative of \(f(x_j)\) w.r.t. \(x_j\) and setting the derivative to zero, we get a quadratic equation: \(x_j^2 + ax_j + b = 0\), where \(a = \epsilon - \lambda_j - \eta \lambda \log \epsilon\) and \(b = \eta \lambda (1 - \epsilon \log \epsilon)\). The optimal solution is achieved either at the positive roots (if any) of this equation or 0. We pick the one that yields the lowest \(f(x_j)\). Formally, \(x^*_j = \arg\min_{x_j} f(x_j)\), where \(x \in \{\max(-b + \sqrt{b^2 - 4ac}/2a, 0), \max(-b - \sqrt{b^2 - 4ac}/2a, 0)\}\)
<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Train</th>
<th>#Test</th>
<th>Dimension</th>
<th>#Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIMIC</td>
<td>40K</td>
<td>18K</td>
<td>1000</td>
<td>2833</td>
</tr>
<tr>
<td>EICU</td>
<td>53K</td>
<td>39K</td>
<td>1000</td>
<td>2175</td>
</tr>
<tr>
<td>Reuters</td>
<td>4152</td>
<td>1779</td>
<td>1000</td>
<td>49</td>
</tr>
<tr>
<td>News</td>
<td>11307</td>
<td>7538</td>
<td>1000</td>
<td>20</td>
</tr>
<tr>
<td>Cars</td>
<td>8144</td>
<td>8041</td>
<td>1000</td>
<td>196</td>
</tr>
<tr>
<td>Birds</td>
<td>9000</td>
<td>2788</td>
<td>1000</td>
<td>200</td>
</tr>
<tr>
<td>Act</td>
<td>7352</td>
<td>2947</td>
<td>561</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 2.8: Dataset statistics.

Computational complexity In this algorithm, the major computation workload is eigen-decomposition of \( m \times m \) matrices, with a complexity of \( O(m^3) \). In our experiments, since \( m \) is no more than 1000, \( O(m^3) \) is not a big bottleneck. Besides, these matrices are symmetric, the structures of which can thus be leveraged to speed up eigen-decomposition. In implementation, we use the MAGMA [7] library that supports efficient eigen-decomposition of symmetric matrices on GPUs. Note that the unregularized convex DML problem [383] also requires eigen-decomposition (of \( M \)), hence adding these CBMD regularizes does not substantially increase additional computation cost.

2.2.4 Evaluation

In this section, we present experimental results on regularized distance metric learning, which demonstrate that compared with nonconvex BMD regularizers, the proposed convex regularizers are computationally more efficient and are more capable of capturing infrequent patterns and reducing model size without sacrificing modeling power.

Datasets We used seven datasets in the experiments: two electronic health record datasets MIMIC-III [184] and EICU (version 1.1) [131]; two text datasets Reuters [10] and 20-Newsgroups (News) [1]; two image datasets Stanford-Cars (Cars) [202] and Caltech-UCSD-Birds (Birds) [358]; and one sensory dataset 6-Activities (Act) [21]. The details of MIMIC-III, Cars, and Birds have been introduced in Section 2.1.4. The class labels in MIMIC are the primary diagnoses of patients. The EICU dataset contains hospital admissions of patients who were treated as part of the Philips eICU program across intensive care units in the United States between 2014 and 2015. Each admission has a primary diagnosis (a disease), which acts as the class label of this admission. There are 2175 unique diseases. There are 474 lab test items and 48 vital sign items. Each admission has a past medical history, which is a collection of diseases. There are 2644 unique past diseases. We extracted the following features: (1) age and gender; (2) zero, first, and second order temporal features of lab tests and vital signs; (3) past medical history: we used a binary vector to encode them; if an element in the vector is 1, then the patient had the corresponding disease in the past. The original Reuters-21578 dataset contains 21578 documents in 135 classes. We removed documents that have more than one labels, and removed classes that have less than 3 documents, which left us 5931 documents and 48 classes. Documents in Reuters and News are represented with \( tf-idf \) vectors where the vocabulary size is 5000. The 6-Activities
dataset contains sensory recordings of 30 subjects performing 6 activities (which are the class labels). The features are 561-dimensional sensory signals. For all the datasets except Act, the features are normalized using min-max normalization along each dimension. We used PCA to reduce the feature dimension to 1000 to reduce the computational complexity (performing eigen-decomposition of the Mahalanobis matrix bears cubic complexity in term of feature dimension). Since there is no standard split of the train/test set, we performed five random splits and averaged the results of the five runs. Dataset statistics are summarized in Table 2.8.

**Experimental settings** Two examples are considered as being similar if they belong to the same class and dissimilar if otherwise. The learned distance metrics were applied for retrieval (using each test example to query the rest of the test examples). For each test example, we used it to query the rest of test examples based on the learned distance metric. If the distance between $x$ and $y$ is smaller than a threshold $s$ and they have the same class label, then this is a true positive. By choosing different values of $s$, we obtained a receiver operating characteristic (ROC) curve. The retrieval performance was evaluated using the area under (ROC) curve (AUC) [243] which is the higher, the better. For AUC on infrequent classes, we used examples belonging to infrequent classes to query the entire test set (excluding the query). AUC on frequent classes was measured in a similar way. Note that the learned distance metrics can also be applied to other tasks such as clustering and classification. In this work, we focus on retrieval.

We compared the proposed convex diversity-promoting regularizers CSFN, CVND, CLDD with two sets of baseline regularizers. The first set aims at promoting near-orthogonality (“diversity”), which are based on determinant of covariance matrix (DCM) [242], cosine similarity (CS) [402], determinantal point process (DPP) [204, 429], InCoherence (IC) [29], variational Gram function (VGF) [177, 420], decorrelation (DC) [82], mutual angles (MA) [369], squared Frobenius norm (SFN) [74, 114, 122, 347], von Neumann divergence (VND) (Section 2.2.1), log-determinant divergence (LDD) (Section 2.2.1), and orthogonal constraint (OC) $AA^\top = I$ [233, 342]. These regularizers were applied to the nonconvex DML (NCDML) formulation in Eq.(2.14). Though VGF is convex, when it is used to regularize NCDML, the overall problem is non-convex and it is unclear how to seek a convex relaxation. The other set of regularizers are not designed particularly for promoting diversity but are commonly used, including $\ell_2$ norm, $\ell_1$ norm [280], $\ell_{2,1}$ norm [399], trace norm (Tr) [234], information theoretic (IT) regularizer $-\log\det(M) + \text{tr}(M)$ [92], and dropout [313]. All these regularizers were applied to the convex DML (CDML) formulation in Eq.(2.22). One common way of dealing with class-imbalance is over-sampling (OS) [116], which repetitively draws samples from the empirical distributions of infrequent classes until all classes have the same number of samples. In addition, we compared with the vanilla Euclidean distance (EUC) and other distance learning methods including large margin nearest neighbor (LMNN) metric learning [357], information theoretic metric learning (ITML) [92], logistic discriminant metric learning (LDML) [146], metric learning from equivalence constraints (MLEC) [200], geometric mean metric learning (GMML) [405], and independent Laplacian hashing with diversity (ILHD) [63]. In LMNN [357], there is a non-convex formulation and a convex formulation. We used the convex one. In GMML [405], the prior matrix was set to an identity matrix. ILHD [63] has several variants, among which we used ILTITF.
<table>
<thead>
<tr>
<th>Method</th>
<th>MIMIC</th>
<th>EICU</th>
<th>Reuters</th>
<th>News</th>
<th>Cars</th>
<th>Birds</th>
<th>Act</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCDML</td>
<td>62.1</td>
<td>66.6</td>
<td>5.2</td>
<td>11.0</td>
<td>8.4</td>
<td>10.1</td>
<td>3.4</td>
</tr>
<tr>
<td>CDML</td>
<td>3.4</td>
<td>3.7</td>
<td>0.3</td>
<td>0.6</td>
<td>0.5</td>
<td>0.6</td>
<td>0.2</td>
</tr>
<tr>
<td>DCM-NCDML [242]</td>
<td>424.7</td>
<td>499.2</td>
<td>35.2</td>
<td>65.6</td>
<td>61.8</td>
<td>66.2</td>
<td>17.2</td>
</tr>
<tr>
<td>CS-NCDML [402]</td>
<td>263.2</td>
<td>284.8</td>
<td>22.6</td>
<td>47.2</td>
<td>34.5</td>
<td>42.8</td>
<td>14.4</td>
</tr>
<tr>
<td>DPP-NCDML [429]</td>
<td>411.8</td>
<td>479.1</td>
<td>36.9</td>
<td>61.9</td>
<td>64.2</td>
<td>70.5</td>
<td>16.5</td>
</tr>
<tr>
<td>IC-NCDML [29]</td>
<td>265.9</td>
<td>281.2</td>
<td>23.4</td>
<td>46.1</td>
<td>37.5</td>
<td>45.2</td>
<td>15.3</td>
</tr>
<tr>
<td>DC-NCDML [82]</td>
<td>458.5</td>
<td>497.5</td>
<td>41.8</td>
<td>78.2</td>
<td>78.9</td>
<td>80.7</td>
<td>19.9</td>
</tr>
<tr>
<td>VGF-NCDML [177]</td>
<td>267.3</td>
<td>284.1</td>
<td>22.3</td>
<td>48.9</td>
<td>35.8</td>
<td>38.7</td>
<td>15.4</td>
</tr>
<tr>
<td>MA-NCDML [367]</td>
<td>271.4</td>
<td>282.9</td>
<td>23.6</td>
<td>50.2</td>
<td>30.9</td>
<td>39.6</td>
<td>17.5</td>
</tr>
<tr>
<td>OC-NCDML [233]</td>
<td>104.9</td>
<td>118.2</td>
<td>9.6</td>
<td>14.3</td>
<td>14.8</td>
<td>17.0</td>
<td>3.9</td>
</tr>
<tr>
<td>SFN-NCDML [74]</td>
<td>261.7</td>
<td>277.6</td>
<td>22.9</td>
<td>46.3</td>
<td>36.2</td>
<td>38.2</td>
<td>15.9</td>
</tr>
<tr>
<td>VND-NCDML</td>
<td>401.8</td>
<td>488.3</td>
<td>33.8</td>
<td>62.5</td>
<td>67.5</td>
<td>73.4</td>
<td>17.1</td>
</tr>
<tr>
<td>LDD-NCDML</td>
<td>407.5</td>
<td>483.5</td>
<td>34.3</td>
<td>60.1</td>
<td>61.8</td>
<td>72.6</td>
<td>17.9</td>
</tr>
<tr>
<td>CSFN-CDML</td>
<td>41.1</td>
<td>43.9</td>
<td>3.3</td>
<td>7.3</td>
<td>6.5</td>
<td>6.9</td>
<td>1.8</td>
</tr>
<tr>
<td>CVND-CDML</td>
<td>43.8</td>
<td>46.2</td>
<td>3.6</td>
<td>8.1</td>
<td>6.9</td>
<td>7.8</td>
<td>2.0</td>
</tr>
<tr>
<td>CLDD-CDML</td>
<td>41.7</td>
<td>44.5</td>
<td>3.4</td>
<td>7.5</td>
<td>6.6</td>
<td>7.2</td>
<td>1.8</td>
</tr>
</tbody>
</table>

Table 2.9: Training time (hours) on seven datasets. On the second panel (DCM-NCDML, etc.) are NCDML methods regularized by previously proposed diversity-promoting regularizers. On the third panel (VND-NCDML, etc.) are NCDML methods regularized by our proposed nonconvex BMD regularizers. On the fourth panel (CSFN-CDML, etc.) are CDML methods regularized by our proposed convex BMD regularizers.

The NCDML-based methods except NCDML-OC were solved with stochastic subgradient descent (SSD). NCDML-OC was solved using the algorithm proposed in [359]. The CDML-based methods were solved with proximal SSD. The learning rate was set to 0.001. The mini-batch size was set to 100 (50 similar pairs and 50 dissimilar pairs). We used 5-fold cross validation to tune the regularization parameter among \( \{10^{-3}, \ldots, 10^0\} \) and the number of components of the NCDML methods among \( \{50, 100, 200, \ldots, 500\} \). In CVND and CLDD, \( \epsilon \) was set to 1e-5. The margin \( t \) was set to 1. In LMNN, the weighting parameter \( \mu \) was set to 0.5. In GMML [405], the regularization parameter \( \lambda \) was set to 0.1. The step length \( t \) of geodesic was set to 0.3. In ILHD [63], the hash function was set to the kernel SVM [294] with a radial basis function kernel whose scale parameter was chosen to be 0.1. Each method was implemented on top of GPUs using the MAGMA [7] library. The experiments were conducted on a GPU-cluster with 40 machines.

For computational efficiency, in CDML-based methods, we do not use \((x - y)^\top M(x - y)\) to compute distance directly. Given the learned matrix \( M \) (which is of rank \( m \)), we can decompose it into \( L^\top L \) where \( L \in \mathbb{R}^{m \times d} \). Let \( U \Sigma U^\top \) be the eigen-decomposition of \( M \). Let \( \lambda_1, \ldots, \lambda_m \) denote the \( m \) nonzero eigenvalues and \( u_1, \ldots, u_m \) denote the corresponding eigenvectors. Then \( L \) is the transpose of \( [\sqrt{\sigma_1} u_1, \ldots, \sqrt{\sigma_m} u_m] \). Given \( L \), we can use it to transform each input \( d \)-dimensional feature vector \( x \) into a new \( m \)-dimensional vector \( Lx \), then perform retrieval on the new vectors based on the Euclidean distance. Note that only when computing Euclidean distance...
between \(L_x\) and \(L_y\), we have that \(\|L_x - L_y\|_2^2\) is equivalent to \((x - y)^\top M(x - y)\). For other distances or similarity measures between \(L_x\) and \(L_y\), such as \(L_1\) distance and cosine similarity, this does not hold. Performing retrieval based on \(\|L_x - L_y\|_2^2\) is more efficient than that based on \((x - y)^\top M(x - y)\) when \(m\) is smaller than \(d\). Given \(n\) test examples, the computation complexity of \(\|L_x - L_y\|_2^2\)-based retrieval is \(O(nmd + n^2m)\), while that of \((x - y)^\top M(x - y)\)-based retrieval is \(O(n^2d^2)\).

### Results

The training time taken by different methods to reach convergence is shown in Table 2.9. For NCDML-based methods, we report the total time taken by the following computation: tuning the regularization parameter (4 choices) and the number of components (NCs, 6 choices) on a two-dimensional grid via 3-fold cross validation (\(4 \times 6 \times 3 = 72\) experiments in total); for each of the 72 experiments, the algorithm restarted 5 times, each with a different initialization, and we picked the one yielding the lowest objective value. In total, the number of runs is \(72 \times 5 = 360\). For the CDML-based methods, there is no need to restart multiple times or tune the NCs. The total number of runs is \(4 \times 3 = 12\). As can be seen from the table, the proposed convex methods are much faster than the non-convex ones, due to the greatly reduced number of experimental runs, although for each single run the convex methods are less efficient than the non-convex methods due to the overhead of eigen-decomposition. The unregularized CDML takes the least time to train since it has no parameters to tune and runs only once. On average, the time of each single run in (CSFN,CVND,CLDD)-CDML is close to that in the unregularized CDML, since an eigen-decomposition is required anyway regardless of the presence of the regularizers. Unregularized NCDML runs faster that regularized NCDML methods because it has no need to tune the regularization parameter, which reduces the number of experimental runs by 4 times. Unregularized CDML runs faster than regularized CDML methods because it has no need to tune the regularization parameter or the number of projection vectors, which reduces the number of experimental runs by 12 times. (DCM,DPP,VND,LDD)-NCDML methods take longer time than other regularized NCDML methods since they need eigen-decomposition to compute the gradients. OC-NCDML has no regularization parameter to tune, hence its number of experimental runs is 4 times fewer than other regularized NCDML methods.

Next, we verify whether CSFN, CVND, and CLDD are able to better capture infrequent patterns. On three datasets MIMIC, EICU, and Reuters where the classes are imbalanced, we consider a class as being “frequent” if it contains more than 1000 examples, and “infrequent” if otherwise. We measure AUCs on all classes (AUC-All), infrequent classes (AUC-IF), and frequent classes (AUC-F). As can be seen, in most DML methods, the AUCs on infrequent classes are worse than those on frequent classes, showing that DML is sensitive to the imbalance of class-frequency and tends to be biased towards frequent classes and is less capable of capturing infrequent classes. This is in accordance with the previous findings [369]. Adding our proposed CSFN, CVND, CLDD regularizers to CDML, the AUCs on infrequent classes are greatly improved. This demonstrates that these convex regularizers can effectively capture infrequent patterns. By encouraging the components to be close to being orthogonal, our methods can reduce the redundancy among vectors. Mutually complementary vectors can achieve a broader coverage of latent features, including those associated with infrequent classes. As a result, these vectors improve the performance on infrequent classes.
Thanks to their convexity nature, our methods can achieve the global optimal solution and outperform the non-convex ones that can only achieve a local optimal and hence a sub-optimal solution. (C)VND and (C)LDD outperform (C)SFN, possibly because they measure near-orthogonality in a global way while (C)SFN conducts that in a pairwise fashion. Comparing OS-(NCDML,CDML) with the unregularized NCDML/CDML, we can see that over-sampling indeed improves performance on infrequent classes. However, this improvement is less significant than that achieved by our methods. In general, the diversity-promoting (DP) regularizers outperform the non-DP regularizers, suggesting the effectiveness of promoting diversity. The orthogonal constraint (OC) \[233, 342\] imposes strict orthogonality, which may be too restrictive that hurts performance. ILHD \[63\] learns binary hash codes, which makes retrieval more efficient. However, it achieves lower AUCs due to the quantization errors. (CSFN,CVND,CLDD)-CDML outperform popular DML approaches including LMNN, LDML, MLEC, and GMML, demonstrating their competitive standing in the DML literature.

Next we verify whether the proposed CDML methods are able to reduce model size without sacrificing modeling power. Table 2.12 shows the numbers of components that achieve the AUCs in Table 6.1 and 2.11. For CDML-based methods, the number of components (NC) is the rank of the Mahalanobis matrix since \(M = A^\top A\). We define a compactness score (CS) which is the ratio between AUC-All and NC. A higher CS indicates the better ability of using fewer components to achieve higher AUC. From Table 2.12, we see that the convex methods achieve larger CS than the baseline methods, demonstrating its better ability of conducting performance-lossless reduction of model size. (C)VND and (C)LDD perform better than (C)SFN, demonstrating the necessity of promoting diversity in a global manner. The proposed convex diversity-promoting (DP) regularizers outperform nonconvex DP regularizers, demonstrating their better ability in promoting diversity thanks to their convex nature. Note that the reduced component number improves the efficiency of retrieval where the computational complexity grows linearly with this number.

As can be seen from Table 6.1 and 2.11, our methods (CSFN,CVND,CLDD)-CDML achieve the best AUC-All on the test set. Table 6.3 shows the difference between training AUC and testing AUC. Our methods have the smallest gap between training and testing AUCs. This indicates that our methods are better capable of reducing overfitting and improving generalization performance.

### 2.3 Angular Constraints for Improving Generalization Performance

In previous two sections, we have empirically demonstrated that diversity-promoting regularization can improve generalization performance. One intuitive explanation could be: promoting diversity imposes a structural constraint on model parameters, which reduces model capacity and therefore alleviates overfitting. However, in theory why larger “diversity” results in lower generalization error (which is the sum of estimation and approximation errors) is still missing. The uniform eigenvalue regularizer and the Bregman matrix divergence (BMD) regularizers studied previously are not amenable for such analysis, especially for the approximation errors. In this
section, we aim at bridging this gap, by proposing a diversity-promoting approach that is both empirically effective and theoretically analyzable. Similar to Section 2.2, we continue to use near-orthogonality to represent “diversity”, but via a different regularization approach – angular constraints (ACs) \[372\] where the angle between components is constrained to be close to \(\frac{\pi}{2}\), which hence encourages the components to be close to being orthogonal. Using neural network as a study case, we analyze how ACs affect its generalization error (Section 4.2.1). The analysis shows that the more close to \(\frac{\pi}{2}\) the angles are, the smaller the estimation error is and the larger the approximation error is. The best tradeoffs of these two errors can be explored by properly tuning the angles. We develop an algorithm based on the alternating direction method of multipliers (ADMM) \[52\] to solve the angle-constrained problems. In various experiments, we demonstrate that ACs improve the generalization performance and outperform other diversity-promoting regularization approaches.

2.3.1 Angular Constraints

Similar to the BMD regularizers (Section 2.2), angular constraints (ACs) use near-orthogonality to characterize “diversity” and encourage the angles between component vectors to be close to \(\frac{\pi}{2}\). The ACs are defined as requiring the absolute value of the cosine similarity between each pair of components to be less equal to a small value \(\tau\), which leads to the following angle-constrained problem

\[
\min_{\mathcal{W}} \mathcal{L}(\mathcal{W}) \\
s.t. \quad 1 \leq i < j \leq m, \quad \frac{|w_i \cdot w_j|}{\|w_i\|_2 \|w_j\|_2} \leq \tau,
\]

(2.24)

where \(\mathcal{W} = \{w_i\}_{i=1}^m\) denotes the component vectors and \(\mathcal{L}(\mathcal{W})\) is the objective function of this problem. The parameter \(\tau\) controls the level of near-orthogonality (or diversity). A smaller \(\tau\) indicates that the vectors are more close to being orthogonal, and hence are more diverse. As will be shown later, representing diversity using the angular constraints facilitates theoretical analysis and is empirically effective as well.

Case study I: sparse coding  Given a set of data samples \(\{x_i\}_{i=1}^n\), where \(x \in \mathbb{R}^d\), sparse coding (SC) \[266\] aims at using a set of “basis” vectors (referred to as dictionary) \(\mathcal{W} = \{w_j\}_{j=1}^m\) to reconstruct the data samples. Each data sample \(x\) is reconstructed by taking a sparse linear combination of the basis vectors \(x \approx \sum_{j=1}^m \alpha_j w_j\) where \(\{\alpha_j\}_{j=1}^m\) are the linear coefficients (referred to as sparse codes) and most of them are zero. The reconstruction error is measured using the squared \(\ell_2\) norm \(\|x - \sum_{j=1}^m \alpha_j w_j\|_2^2\). To achieve sparsity among the coefficients, \(\ell_1\)-regularization is utilized: \(\sum_{j=1}^m |\alpha_j|\). To avoid the degenerated case where most coefficients are zero and the basis vectors are of large magnitude, \(\ell_2\)-regularization is applied to the basis vectors: \(\|w_j\|_2^2\). Putting these pieces together, we learn the basis vectors and sparse codes (denoted by \(\mathcal{A}\)) by minimizing the following objective function: \(\mathcal{L}(\mathcal{W}, \mathcal{A}) = \frac{1}{2} \sum_{i=1}^n (\|x_i - \sum_{j=1}^m \alpha_{ij} w_j\|_2^2 + \lambda_1 \sum_{j=1}^m |\alpha_{ij}| + \lambda_2 \sum_{j=1}^m \|w_j\|_2^2\). We can use the ACs to encourage the basis vectors to be “diverse”.

37
Case study II: neural networks In a neural network (NN) with $L$ hidden layers, each hidden layer $l$ is equipped with $m^{(l)}$ units and each unit $i$ is connected with all units in layer $l-1$. Hidden unit $i$ at layer $l$ is parameterized by a weight vector $w_i^{(l)}$. These hidden units aim at capturing latent features underlying the data. We can apply ACs to the weight vectors of hidden units (in the same layer) to encourage diversity.

Connection with the log-determinant divergence regularizer

In this section, we make a connection between the angular constraints and the log-determinant divergence (LDD) regularizer (Section 2.2.1). Let $s_{ij} = \frac{|\langle w_i, w_j \rangle|}{\|w_i\|_2 \|w_j\|_2}$ be the absolute value of the cosine similarity between $w_i$ and $w_j$. The angular constraints can be equivalently written as

$$s(W) = (\max_{1 \leq i < j \leq m} s_{ij}) \leq \tau. \quad (2.25)$$

It can be proved that the negative gradient of the LDD regularizer $\Omega_{ldd}(W)$ is an descent direction of $s(W)$, which is formally given in the following lemma.

**Lemma 1.** Let $\tilde{W} = \{\tilde{w}_i\}_{i=1}^m$ be a set of vectors where $\tilde{w}_i = w_i - \eta g_i$, and $g_i$ is the gradient of $\Omega_{ldd}(W)$ w.r.t. $w_i$. Then $\exists \delta > 0$ such that $\forall \eta \in (0, \delta)$, $s(\tilde{W}) \leq s(W)$.

This implies that $\Omega_{ldd}(W)$ and $s(W)$ are closely aligned. Decreasing $\Omega_{ldd}(W)$ effectively decreases $s(W)$ and drives the angles among components to approach $\pi/2$.

In addition, it can be showed that the variance of angles can be reduced by minimizing $\Omega_{ldd}(W)$. Consider a set of angles $\{\theta_{ij}\}$ where $\theta_{ij} = \arccos(s_{ij})$, for all $i \neq j$. Let $\text{var}(\{\theta_{ij}\})$ denote the variance of the angles. Then we have the following lemma.

**Lemma 2.** Let $\tilde{W} = \{\tilde{w}_i\}_{i=1}^m$ be a set of vectors where $\tilde{w}_i = w_i - \eta g_i$, and $g_i$ is the gradient of $\Omega_{ldd}(W)$ w.r.t. $w_i$. Let $\tilde{s}_{ij} = \frac{|\langle \tilde{w}_i, \tilde{w}_j \rangle|}{\|\tilde{w}_i\|_2 \|\tilde{w}_j\|_2}$ and $\tilde{\theta}_{ij} = \arccos(\tilde{s}_{ij})$. Then $\exists \delta > 0$ such that $\forall \eta \in (0, \delta)$, $\text{var}(\{\tilde{\theta}_{ij}\}) \leq \text{var}(\{\theta_{ij}\})$.

2.3.2 An ADMM-based Algorithm

In this section, we develop an ADMM-based algorithm to solve the AC regularized problems. To make it amenable for optimization, we first factorize each weight vector $w$ into its $\ell_2$ norm $g = \|w\|_2$ and direction $\tilde{w} = \frac{w}{\|w\|_2}$. Under such a factorization, $w$ can be reparameterized as $w = g \tilde{w}$, where $g > 0$ and $\|\tilde{w}\|_2 = 1$. Then the problem defined in Eq.(2.24) can be transformed into

$$\min_{W,G} \mathcal{L}(\tilde{W}, G) \quad \text{s.t.} \quad \forall j, g_j \geq 0, \|\tilde{w}_j\|_2 = 1 \quad \forall i \neq j, |\tilde{w}_i \cdot \tilde{w}_j| \leq \tau, \quad (2.26)$$

where $\tilde{W} = \{\tilde{w}_j\}_{j=1}^m$ and $G = \{g_j\}_{j=1}^m$. We solve this new problem by alternating between $\tilde{W}$ and $G$. With $\tilde{W}$ fixed, the problem defined over $G$ is: $\min_G \mathcal{L}(G)$ subject to $\forall j, g_j \geq 0$, which can be solved using projected gradient descent. With $G$ fixed, the sub-problem defined over $\tilde{W}$
which we solve using an ADMM-based algorithm. There are \( R = m(m - 1) \) pairwise constraints \( |\tilde{w}_i \cdot \tilde{w}_j| \leq \tau \). For the \( r \)-th constraint, let \( p(r) \) and \( q(r) \) be the index of the first and second vector respectively, i.e., the \( r \)-th constraint is \( |\tilde{w}_{p(r)} \cdot \tilde{w}_{q(r)}| \leq \tau \). First, we introduce auxiliary variables \( \{v_1^{(r)}\}_{r=1}^R \) and \( \{v_2^{(r)}\}_{r=1}^R \), to rewrite the problem in Eq. (2.27) into an equivalent form. For each pairwise constraint: \( |\tilde{w}_{p(r)} \cdot \tilde{w}_{q(r)}| \leq \tau \), we introduce two auxiliary vectors \( v_1^{(r)} \) and \( v_2^{(r)} \), and let \( \tilde{w}_{p(r)} = v_1^{(r)} \), \( \tilde{w}_{q(r)} = v_2^{(r)} \), \( \|v_1^{(r)}\|_2 = 1 \), \( \|v_2^{(r)}\|_2 = 1 \), \( |v_1^{(r)} \cdot v_2^{(r)}| \leq \tau \). To this end, we obtain the following problem

\[
\min_{\tilde{W}, \tilde{V}} \mathcal{L}(\tilde{W}) \\
\text{s.t.} \quad \forall j, \|\tilde{w}_j\|_2 = 1 \\
\forall r, \tilde{w}_{p(r)} = v_1^{(r)}, \tilde{w}_{q(r)} = v_2^{(r)} \\
\forall r, \|v_1^{(r)}\|_2 = 1, \|v_2^{(r)}\|_2 = 1, |v_1^{(r)} \cdot v_2^{(r)}| \leq \tau,
\]

where \( \mathcal{V} = \{(v_1^{(r)}, v_2^{(r)})\}_{r=1}^R \). Then we define the augmented Lagrangian, with Lagrange multipliers \( \mathcal{Y} = \{(y_1^{(r)}, y_2^{(r)})\}_{r=1}^R \) and parameter \( \rho \)

\[
\min_{\tilde{W}, \tilde{V}, \mathcal{Y}} \mathcal{L}(\tilde{W}) + \sum_{r=1}^R \left( y_1^{(r)} \cdot (\tilde{w}_{p(r)} - v_1^{(r)}) + y_2^{(r)} \cdot (\tilde{w}_{q(r)} - v_2^{(r)}) + \frac{\rho}{2} \|\tilde{w}_{p(r)} - v_1^{(r)}\|_2^2 \\
+ \frac{\rho}{2} \|\tilde{w}_{q(r)} - v_2^{(r)}\|_2^2 \right) \\
\text{s.t.} \quad \forall j, \|\tilde{w}_j\|_2 = 1 \\
\forall r, \|v_1^{(r)}\|_2 = 1, \|v_2^{(r)}\|_2 = 1, |v_1^{(r)} \cdot v_2^{(r)}| \leq \tau,
\]

which can be solved by alternating between \( \tilde{W}, \mathcal{V}, \mathcal{Y} \).

**Solve \( \tilde{W} \) The sub-problem defined over \( \tilde{W} \) is**

\[
\min_{\tilde{W}} \mathcal{L}(\tilde{W}) + \sum_{r=1}^R \left( y_1^{(r)} \cdot \tilde{w}_{p(r)} + y_2^{(r)} \cdot \tilde{w}_{q(r)} + \frac{\rho}{2} \|\tilde{w}_{p(r)} - v_1^{(r)}\|_2^2 + \frac{\rho}{2} \|\tilde{w}_{q(r)} - v_2^{(r)}\|_2^2 \right) \\
\text{s.t.} \quad \forall j, \|\tilde{w}_j\|_2 = 1.
\]

For sparse coding, we solve this sub-problem using coordinate descent. At each iteration, we update \( \tilde{w}_j \) by fixing the other variables. For neural network, this sub-problem can be solved using projected gradient descent which iteratively performs the following three steps: (1) compute the gradient of \( \tilde{w}_j \) using backpropagation; (2) perform a gradient descent update of \( \tilde{w}_j \); (3) project each vector onto the unit sphere: \( \tilde{w}_j \leftarrow \tilde{w}_j / \|\tilde{w}_j\|_2 \).
Solve $v_1^{(r)}, v_2^{(r)}$ The corresponding sub-problem is

$$\min_{v_1^{(r)}, v_2^{(r)}} -y_1^{(r)} \cdot v_1^{(r)} - y_2^{(r)} \cdot v_2^{(r)} + \frac{\rho}{2} \|\tilde{w}_{p(r)} - v_1^{(r)}\|^2 + \frac{\rho}{2} \|\tilde{w}_{q(r)} - v_2^{(r)}\|^2$$

$$s.t. \quad \|v_1^{(r)}\|_2 = 1, \|v_2^{(r)}\|_2 = 1, v_1^{(r)} \cdot v_2^{(r)} \leq \tau, -v_1^{(r)} \cdot v_2^{(r)} \leq \tau.$$

(2.31)

Let $\gamma_1, \gamma_2, \lambda_1 \geq 0, \lambda_2 \geq 0$ be the KKT multipliers associated with the four constraints in this sub-problem. According to the KKT conditions [53], we have

$$-y_1^{(r)} + \rho(v_1^{(r)} - \tilde{w}_{p(r)}) + 2\gamma_1 v_1^{(r)} + (\lambda_1 - \lambda_2)v_2^{(r)} = 0$$

$$-y_2^{(r)} + \rho(v_2^{(r)} - \tilde{w}_{q(r)}) + 2\gamma_2 v_2^{(r)} + (\lambda_1 - \lambda_2)v_1^{(r)} = 0.$$

(2.32) \hspace{1cm} (2.33)

We solve these two equations by examining four cases.

**Case 1** First, we assume $\lambda_1 = 0, \lambda_2 = 0$, then $(\rho + 2\gamma_1)v_1^{(r)} = y_1^{(r)} + \rho\tilde{w}_{p(r)}$ and $(\rho + 2\gamma_2)v_2^{(r)} = y_2^{(r)} + \rho\tilde{w}_{q(r)}$. According to the primal feasibility [53] $\|v_1^{(r)}\|_2 = 1$ and $\|v_2^{(r)}\|_2 = 1$, we know

$$v_1^{(r)} = \frac{y_1^{(r)} + \rho\tilde{w}_{p(r)}}{\|y_1^{(r)} + \rho\tilde{w}_{p(r)}\|_2}, \quad v_2^{(r)} = \frac{y_2^{(r)} + \rho\tilde{w}_{q(r)}}{\|y_2^{(r)} + \rho\tilde{w}_{q(r)}\|_2}.$$

(2.34)

Then we check whether the constraint $|v_1^{(r)} \cdot v_2^{(r)}| \leq \tau$ is satisfied. If so, then $v_1^{(r)}$ and $v_2^{(r)}$ are the optimal solution.

**Case 2** We assume $\lambda_1 > 0$ and $\lambda_2 = 0$, then

$$(\rho + 2\gamma_1)v_1^{(r)} + \lambda_1 v_2^{(r)} = y_1^{(r)} + \rho\tilde{w}_{p(r)}$$

$$\quad (\rho + 2\gamma_2)v_2^{(r)} + \lambda_1 v_1^{(r)} = y_2^{(r)} + \rho\tilde{w}_{q(r)}.$$  \hspace{1cm} (2.35) \hspace{1cm} (2.36)

According to the complementary slackness condition [53], we know $v_1^{(r)} \cdot v_2^{(r)} = \tau$. For the vectors on both sides of Eq.(2.35), taking the square of their $\ell_2$ norm, we get

$$(\rho + 2\gamma_1)^2 + \lambda_1^2 + 2(\rho + 2\gamma_1)\lambda_1 \tau = \|y_1^{(r)} + \rho\tilde{w}_{p(r)}\|^2.$$  \hspace{1cm} (2.37)

Similarly, from Eq.(2.36), we get

$$(\rho + 2\gamma_2)^2 + \lambda_1^2 + 2(\rho + 2\gamma_2)\lambda_1 \tau = \|y_2^{(r)} + \rho\tilde{w}_{q(r)}\|^2.$$  \hspace{1cm} (2.38)

Taking the inner product of the two vectors on the left hand sides of Eq.(2.35) and Eq.(2.36), and those on the right hand sides, we get

$$(2\rho + 2\gamma_1 + 2\gamma_2)\lambda_1 + ((\rho + 2\gamma_1)(\rho + 2\gamma_2))\lambda_1^2 \tau = (y_1^{(r)} + \rho\tilde{w}_{p(r)})^T (y_2^{(r)} + \rho\tilde{w}_{q(r)}).$$  \hspace{1cm} (2.39)

Solving the system of equations consisting of Eq.(2.37) and Eq.(2.39), we obtain the optimal values of $\gamma_1, \gamma_2,$ and $\lambda_1$. Plugging them into Eq.(2.35) and Eq.(2.36), we obtain a solution of $v_1^{(r)}$ and $v_2^{(r)}$. Then we check whether this solution satisfies $-v_1^{(r)} \cdot v_2^{(r)} \leq \tau.$ If so, this is an optimal solution.

In Case 3, we discuss $\lambda_1 = 0, \lambda_2 > 0$. In Case 4, we discuss $\lambda_1 > 0, \lambda_2 > 0$. The corresponding problems can be solved in a similar way as Case 2.
Solve $y_1^{(r)}, y_2^{(r)}$ We simply perform the following updates:

$$y_1^{(r)} = y_1^{(r)} + \rho(\tilde{w}_{p(r)} - v_1^{(r)})$$

(2.40)

$$y_2^{(r)} = y_2^{(r)} + \rho(\tilde{w}_{q(r)} - v_2^{(r)}).$$

(2.41)

Compared with a vanilla backpropagation algorithm, the major extra cost in this ADMM-based algorithm comes from solving the $R = m(m-1)$ pairs of vectors $\{v_1^{(r)}, v_2^{(r)}\}_{r=1}^R$. Solving each pair incurs $O(m)$ cost. The $R$ pairs bring in a total cost of $O(m^3)$. Such a cubic cost is also incurred in other diversity-promoting methods such as [29, 213]. In practice, $m$ is typically less than 1000. This $O(m^3)$ cost does not substantially bottleneck computation, as we will validate in experiments.

2.3.3 Evaluation

In this section, we empirically show the effectiveness of the angular constraints. The theoretical analysis is deferred to Section 4.2.1.

Sparse Coding

Following [392], we applied sparse coding for image feature learning. We used three datasets in the experiments: Scenes-15 [212], Caltech-256 [141], and UIUC-Sport [224]. For each dataset, five random train/test splits were performed and the results were averaged over the five runs. We extracted pixel-level dense SIFT [237] features with step size 8 and patch size 16. On top of the SIFT features, we used sparse coding methods to learn a dictionary and represented each SIFT feature into a sparse code. To obtain image-level features, we applied max-pooling [392] and spatial pyramid matching [212, 392] over the pixel-level sparse codes. Then a linear SVM was applied to classify the images. We compared with other diversity-promoting regularizers including determinant of covariance matrix (DCM) [242], cosine similarity (CS) [402], determinant point process (DPP) [204, 429], InCoherence (IC) [29], and mutual angles (MA) [369]. We used 5-fold cross validation to tune $\tau$ in $\{0.3, 0.4, \cdots, 1\}$ and the number of basis vectors in $\{50, 100, 200, \cdots, 500\}$. The parameter $\rho$ in ADMM was set to 1.

Table 2.14 shows the classification accuracy on three datasets, from which we can see that compared with unregularized SC, AC-SC greatly improves performance. For example, on the Sports dataset, AC improves the accuracy from 87.4% to 90.9%. This suggests that AC is effective in reducing overfitting and improving generalization performance. Compared with other diversity-promoting regularizers, AC achieves better performance, demonstrating its better efficacy in promoting diversity.

Neural Networks

We evaluated AC on three types of neural networks: fully-connected NN (FNN) for phone recognition [159], CNN for image classification [203], and RNN for question answering [296]. In the main paper, we report results on four datasets: TIMIT [11], CIFAR-10 [5], CNN [154], and DailyMail [154].
FNN for phone recognition  The TIMIT dataset contains a total of 6300 sentences (5.4 hours), divided into a training set (462 speakers), a validation set (50 speakers), and a core test set (24 speakers). We used the Kaldi [277] toolkit to train the monophone system which was utilized to perform forced alignment and to get labels for speech frames. The toolkit was also utilized to preprocess the data into log-filter banks. Among FNN-based methods, Karel’s recipe in Kaldi achieves state of the art performance. We applied AC to the FNN in this recipe. The inputs of the FNN are the FMLLR [117] features of 21 neighboring frames, which are mean-centered and normalized to have unit variance. The number of hidden layers was 4. Each layer had 1024 hidden units. Stochastic gradient descent (SGD) was used to train the network. The learning rate was set to 0.008. We compared with four diversity-promoting regularizers: CS, IC, MA, and DeCorrelation (DC) [82]. The regularization parameter in these methods were tuned in \(\{10^{-6}, 10^{-5}, \ldots, 10^5\}\). The \(\beta\) parameter in IC was set to 1.

Table 2.15 shows state of the art phone error rate (PER) on the TIMIT core test set. Methods in the first panel are mostly based on FNN, which perform less well than Kaldi. Methods in the third panel are all based on RNNs which in general perform better than FNN since they are able to capture the temporal structure in speech data. In the second panel, we compare AC with other diversity-promoting regularizers. Without regularization, the error is 18.53%. With AC, the error is reduced to 18.41%, which is very close to a strong RNN-based baseline – connectionist temporal classification (CTC) [139]. Besides, AC outperforms other regularizers.

CNN for image classification  The CIFAR-10 dataset contains 32x32 color images from 10 categories, with 50,000 images for training and 10,000 for testing. We used 5000 training images as the validation set to tune hyperparameters. The data was augmented by first zero-padding the images with 4 pixels on each side, then randomly cropping the padded images to reproduce 32x32 images. We applied AC to the wide residual network (WideResNet) [406] where the depth was set to 28 and the width was set to 10. SGD was used for training, with epoch number 200, initial learning rate 0.1, minibatch size 128, Nesterov momentum 0.9, dropout probability 0.3, and weight decay 0.0005. The learning rate was dropped by 0.2 at 60, 120, and 160 epochs. The performance was the median of 5 runs. We compared with CS, IC, MA, DC, and locally constrained decorrelation (LCD) [291].

Table 2.16 shows state of the art classification error on the test set. Compared with the unregularized WideResNet which achieves an error of 3.89%, applying AC reduces the error to 3.63%. AC achieves lower error than other regularizers.

LSTM for question answering  We applied AC to the long short-term memory (LSTM) [163] network (Section 2.1.2). On the row vectors of each gate-specific weight matrix, the AC was applied. The LSTM was used for a question answering (QA) task on two datasets: CNN and DailyMail [154], each containing a training, development, and test set with 300k/4k/3k and 879k/65k/53k examples respectively. Each example consists of a passage, a question, and an answer. The question is a cloze-style task where an entity is replaced by a placeholder and the goal is to infer this missing entity (answer) from all the possible entities appearing in the passage. The LSTM architecture and experimental settings followed the BIDirectional Attention Flow (BIDAF) [296] model, which consists of the following layers: character embedding, word em-
bedding, contextual embedding, attention flow, modeling and output. LSTM was applied in the contextual embedding and modeling layer. Character embedding was based on one-dimensional convolutional neural network, where the number of filters was set to 100 and the width of receptive field was set to 5. In LSTM, the size of hidden state was set to 100. Optimization was based on AdaDelta [409], where the minibatch size and initial learning rate were set to 48 and 0.5. The model was trained for 8 epochs. Dropout [313] with probability 0.2 was applied. We compared with four diversity promoting regularizers: CS, IC, MA and DC.

Table 6.8 shows state of the art accuracy on the two datasets. As can be seen, after applying AC to BIDAF, the accuracy is improved from 76.94% to 77.23% on the CNN test set and from 79.63% to 79.88% on the DailyMail test set. Among the diversity-promoting regularizers, AC achieves the highest accuracy.

**Sensitivity to the parameter \( \tau \)**

Figure 2.5 shows how the phone error rates vary on the TIMIT core test set. As can be seen, the lowest test error is achieved under a moderate \( \tau (\approx 0.75) \). Either a smaller or a larger \( \tau \) degrades the performance. This suggests that \( \tau \) effectively incurs a tradeoff between estimation and approximation errors. When \( \tau \) is close to 0, the hidden units are close to being orthogonal, which yields much poorer performance. This confirms that the strict-orthogonality constraint proposed by [213] is too restrictive and is less favorable than a “soft” regularization approach.

**Computational time**

We compare the computational time of neural networks under different regularizers. Table 2.18 shows the total runtime time of FNNs on TIMIT and CNNs on CIFAR-10 with a single GTX TITAN X GPU, and the runtime of LSTM networks on the CNN dataset with 2 TITAN X GPUs. Compared with no regularization, AC incurs a 18.2% extra time on TIMIT, 12.7% on CIFAR-10, and 14.8% on CNN. The runtime of AC is comparable to that of other diversity-promoting regularizers.
2.3.4 Appendix: Proofs and Details of Algorithms

Proof of Lemma 1

To prove Lemma 1, we need the following Lemma.

**Lemma 3.** Let $G$ be the Gram matrix defined on $\mathcal{W} = \{w_i\}_{i=1}^m$. Let $g'_i$ be the gradient of $\det(G)$ w.r.t $w_i$, then $\langle g'_i, w_j \rangle = 0$ for all $j \neq i$, and $\langle g'_i, w_i \rangle > 0$.

**Proof.** We decompose $w_i$ into $w_i = w_i^0 + w_i^\perp$. Let $\langle w_i^0, w_j \rangle > 0$ for all $j$, $w_i^0$ is in the span of $\mathcal{W}/\{w_i\}$: $w_i^0 = \sum_{j \neq i} \alpha_j w_j$, where $\{\alpha_j\}_{j \neq i}$ are the linear coefficients. $w_i^\perp$ is orthogonal to $\mathcal{W}/\{w_i\}$: $\langle w_i^\perp, w_j \rangle = 0$ for all $j \neq i$. Let $c_j$ denote the $j$-th column of $G$. Subtracting $\sum_{j \neq i} \alpha_j c_j$ from the $i$-th column, we get

$$
\det(G) = \det(G_{-i}) \langle w_i^\perp, w_i \rangle, \tag{2.43}
$$

where $G_{-i}$ is the Gram matrix defined on $\mathcal{W}/\{w_i\}$. Then the functional gradient $g'_i$ of $\det(G)$ w.r.t $w_i$ is $\det(G_{-i}) w_i^\perp$, which is orthogonal to $w_j$ for all $j \neq i$. Since $G$ is full rank, we know $\det(G) > 0$. From Eq. (2.43) we know $\langle w_i^\perp, w_i \rangle$ is non-negative. Hence $\langle g'_i, w_i \rangle = \det(G_{-i}) \langle w_i^\perp, w_i \rangle > 0$.

Now we are ready to prove Lemma 1. We first compute $\hat{s}_{ij}$:

$$
\hat{s}_{ij} = \frac{\|\hat{w}_i - \eta g_i \| \|\hat{w}_j - \eta g_j \|}{\|w_i\| \|w_j\|} = \frac{\|w_i - \eta g_i - \eta g_j\|}{\|w_i - \eta g_i\| \|w_j - \eta g_j\|}, \tag{2.44}
$$

The functional gradient of $\log \det(G)$ w.r.t $w_i$ is computed as $g''_i = \frac{1}{\det(G)} g'_i$. According to Lemma 3 and the fact that $\det(G) > 0$, we know $\langle g''_i, w_j \rangle = 0$ for all $j \neq i$, and $\langle g''_i, w_i \rangle > 0$. Then we have

$$
\langle w_i - \eta g_i, w_j - \eta g_j \rangle = \langle w_i - \eta(2w_i - 2g''_i), w_j - \eta(2w_j - 2g''_j) \rangle = \langle (1 - 2\eta)w_i + 2\eta g''_i, (1 - 2\eta)w_j + 2\eta g''_j \rangle \tag{2.45}
$$

and

$$
\frac{1}{\sqrt{\|w_i - \eta g_i\|^2}} = \frac{1}{\sqrt{\|w_i\|^2 - 2\eta \langle w_i, g_i \rangle + \eta^2 \|g_i\|^2}} = \frac{1}{\sqrt{\|w_i\|^2 (1 - 2\eta \langle w_i, g_i \rangle) / \|w_i\|^2 + \eta^2 \|g_i\|^2 / \|w_i\|^2}} = \frac{1}{\sqrt{1 - 2\eta \langle w_i, g_i \rangle / \|w_i\|^2 + \eta^2 \|g_i\|^2 / \|w_i\|^2}}, \tag{2.46}
$$

44
According to the Taylor expansion, we have
\[
\frac{1}{\sqrt{1 + x}} = 1 - \frac{1}{2}x + o(x). \tag{2.47}
\]
Then
\[
\frac{1}{\sqrt{1 - 2\eta(w_i, g_i) + \eta^2 \| g_i \|^2}} = 1 - \frac{1}{2} \left( -\frac{2\eta(w_i, g_i)}{\| w_i \|^2} + \frac{\eta^2 \| g_i \|^2}{\| w_i \|^2} \right) + o \left( -\frac{2\eta(w_i, g_i)}{\| w_i \|^2} + \frac{\eta^2 \| g_i \|^2}{\| w_i \|^2} \right), \tag{2.48}
\]
where
\[
\frac{2\eta(w_i, g_i)}{\| w_i \|^2} = \frac{2\eta(w_i, 2w_i - 2g_i')}{\| w_i \|^2} = 4\eta - 4\eta \frac{(w_i, g_i')}{\| w_i \|^2}. \tag{2.49}
\]
Hence
\[
\frac{1}{\sqrt{1 + 2\eta(w_i, g_i) + \eta^2 \| g_i \|^2}} = 1 + 2\eta - 2\eta \frac{(w_i, g_i')}{\| w_i \|^2} + o(\eta), \tag{2.50}
\]
and
\[
\frac{1}{\sqrt{1 + 2\eta(w_j, g_j) + \eta^2 \| g_j \|^2}} \frac{1}{\sqrt{1 + 2\eta(w_j, g_j) + \eta^2 \| g_j \|^2}} = (1 + 2\eta)^2 - 2\eta \left( \frac{(w_i, g_i')}{\| w_i \|^2} + \frac{(w_j, g_j')}{\| w_j \|^2} \right) + o(\eta). \tag{2.51}
\]
Then
\[
\hat{s}_{ij} = \frac{|\langle w_i, w_j \rangle|}{\| w_i \| \| w_j \|} \left( 1 - 2\eta \right)^2 + \frac{4\eta^2 (g_i', g_j')}{\langle w_i, w_j \rangle} \left( 1 + 2\eta \right)^2 - 2\eta \left( \frac{(w_i, g_i')}{\| w_i \|^2} + \frac{(w_j, g_j')}{\| w_j \|^2} \right) + o(\eta) \tag{2.52}
\]
where in the second step, the absolute value can be removed because there always exists a sufficiently small \( \eta \) such that \( (1 - 2\eta)^2 + \frac{4\eta^2 (g_i', g_j')}{\langle w_i, w_j \rangle} > 0 \). Eq.\( (2.52) \) holds for all \( i, j \), hence \( \hat{s}(\hat{W}) < s(W) \). The proof completes.

**Proof of Lemma 2**

To prove this lemma, we need the following lemma.

**Lemma 4.** Given a nondecreasing sequence \( b = (b_i)_{i=1}^n \) and a strictly decreasing function \( g(x) \) which satisfies \( 0 \leq g(b_i) \leq \min \{ b_{i+1} - b_i \mid i = 1, 2, \ldots, n-1, b_{i+1} \neq b_i \} \), we define a sequence \( c = (c_i)_{i=1}^n \) where \( c_i = b_i + g(b_i) \). If \( b_1 < b_o \), then \( \text{var}(c) < \text{var}(b) \), where \( \text{var}(\cdot) \) denotes the variance of a sequence. Furthermore, let \( n' = \max \{ j \mid b_j \neq b_n \} \), we define a sequence \( b' = (b'_i)_{i=1}^n \) where \( b'_i = b_i + g(b_o) + (g(b_{n'}) - g(b_o)) \| i \leq n' \) and \( \| \cdot \) is the indicator function, then \( \text{var}(c) \leq \text{var}(b') < \text{var}(b) \).

The intuition behind the proof of Lemma 2 is: when the steps is sufficiently small, we can make sure the changes of smaller angles (between consecutive iterations) are larger than the changes of larger angles, then Lemma 4 can be used to prove that the variance decreases. Let \( \theta_{ij} \)
denote $\arccos(s_{ij})$. We sort $\theta_{ij}$ in nondecreasing order and denote the resultant sequence as $\theta = (\theta_k)_{k=1}^n$, then $\text{var}((\theta_{ij})) = \text{var}(\theta)$. We use the same order to index $\hat{\theta}_{ij} = \arccos(s_{ij})$ and denote the resultant sequence as $\hat{\theta} = (\hat{\theta}_k)_{k=1}^n$, then $\text{var}((\hat{\theta}_{ij})) = \text{var}(\hat{\theta})$. Let $g(\theta_{ij}) = \frac{2\cos(\theta_{ij})}{\sqrt{1-\cos(\theta_{ij})^2}}\eta$ if $\theta_{ij} < \frac{\pi}{2}$ and 0 if $\theta_{ij} = \frac{\pi}{2}$, then $g(\theta_{ij})$ is a strictly decreasing function. Let $\tilde{\theta}_k = \theta_k + g(\theta_k)$. It is easy to see when $\eta$ is sufficiently small, $0 \leq g(\theta_k) \leq \min\{\theta_{k+1} - \theta_k : k = 1, 2, \cdots, n-1, \theta_{k+1} \neq \theta_k\}$. According to Lemma 4 we have

$$\text{var}(\tilde{\theta}) < \text{var}(\theta),$$

where $\tilde{\theta} = (\tilde{\theta}_k)_{k=1}^n$. Furthermore, let $n' = \max\{j : \theta_j \neq \theta_n\}$, $\theta'_k = \theta_k + g(\theta_n) + (g(\theta_{n'}) - g(\theta_{n}))\|k \leq n'\)$, then

$$\text{var}(\tilde{\theta}) \leq \text{var}(\theta') < \text{var}(\theta),$$

where $\theta' = (\theta'_k)_{k=1}^n$. $\text{var}(\theta')$ can be written as:

$$= \frac{1}{n} \sum_{i=1}^n \theta'_i - \frac{1}{n} \sum_{j=1}^n \theta'_j^2$$

$$= \frac{1}{n} \sum_{i=1}^n (\theta_i + (g(\theta_{n'}) - g(\theta_n))\|i \leq n'\) - \frac{1}{n} \sum_{j=1}^n \theta_j - \frac{n'\theta'(\theta_{n'} - g(\theta_{n}))^2}{n}\|$$

$$+ \frac{1}{n} \sum_{i=1}^n (g(\theta_{n'}) - g(\theta_n))^2\|i \leq n'\) - \frac{n'\theta'(\theta_{n'} - g(\theta_{n}))^2}{n}\|$$

Let $\lambda = \frac{2}{n} \sum_{i=1}^n (\theta_i - \frac{1}{n} \sum_{j=1}^n \theta_j)\|i \leq n'\) - \frac{n'\theta'(\theta_{n'} - g(\theta_{n}))^2}{n}\|$, which can be further written as

$$= \frac{2}{n} \sum_{i=1}^{n'} (\theta_i - \frac{1}{n} \sum_{j=1}^n \theta_j)(1 - \frac{n'}{n}) + \frac{2}{n} \sum_{i=n'+1}^n \theta_i - \frac{1}{n} \sum_{j=1}^n \theta_j)\|(-\frac{n'}{n})$$

$$= \frac{2}{n} (\sum_{i=1}^{n'} \theta_i - \frac{n'}{n} \sum_{j=1}^n \theta_j)(1 - \frac{n'}{n}) + \frac{2}{n} \sum_{i=n'+1}^n \theta_i - \frac{n-n'}{n} \sum_{j=1}^n \theta_j)\|(-\frac{n'}{n})$$

As $(\theta_k)$ is nondecreasing and $\theta_n \neq \theta_{n'}$, we have $\lambda < 0$. Let $\mu = \frac{2\cos(\theta_{n'})}{\sqrt{1-\cos(\theta_{n'})^2}} - \frac{2\cos(\theta_n)}{\sqrt{1-\cos(\theta_n)^2}}$ when $\theta_n < \frac{\pi}{2}$ and $\mu = \frac{2\cos(\theta_{n'})}{\sqrt{1-\cos(\theta_{n'})^2}}$ when $\theta_n = \frac{\pi}{2}$, then $g(\theta_{n'}) - g(\theta_n) = \mu \eta$ and $\mu > 0$. Substituting $\lambda$ and $\mu$ into $\text{var}(\theta')$, we can obtain:

$$\text{var}(\theta') = \text{var}(\theta) + \lambda \mu \eta + \frac{1}{n} \sum_{i=1}^n \|i \leq n'\) - \frac{n'\theta'(\theta_{n'} - g(\theta_{n}))^2}{n}\|^2 \mu^2 \eta^2$$

$$= \text{var}(\theta) + \lambda \mu \eta + o(\eta)$$

Note that $\lambda < 0$ and $\mu > 0$, so $\exists \delta_1$, such that

$$\eta < \delta_1 \Rightarrow \text{var}(\theta') < \text{var}(\theta) + \frac{\lambda \mu \eta}{2}$$

As $\text{var}(\tilde{\theta}) < \text{var}(\theta')$, we can draw the conclusion that

$$\text{var}(\tilde{\theta}) < \text{var}(\theta) + \frac{\lambda \mu \eta}{2}.$$

Further, 

$$\text{var}(\tilde{\theta}) = \frac{1}{n} \sum_{i=1}^n (\tilde{\theta}_i - \frac{1}{n} \sum_{j=1}^n \tilde{\theta}_j)^2$$

$$= \frac{1}{n} \sum_{i=1}^n (\tilde{\theta}_i + o(\eta) - \frac{1}{n} \sum_{j=1}^n \tilde{\theta}_j + o(\eta))^2$$

$$= \frac{1}{n} \sum_{i=1}^n (\tilde{\theta}_i - \frac{1}{n} \sum_{j=1}^n \tilde{\theta}_j)^2 + o(\eta)$$

$$= \text{var}(\tilde{\theta}) + o(\eta).$$

46
So $\exists \delta_2 > 0$ such that $\eta < \delta_2 \Rightarrow \text{var}(\hat{\theta}) < \text{var}(\hat{\theta}) - \frac{\lambda \eta}{4}$. Let $\delta = \min\{\delta_1, \delta_2\}$, then

$$\eta < \delta \Rightarrow \text{var}(\hat{\theta}) < \text{var}(\theta) + \frac{\lambda \eta}{4} < \text{var}(\theta) \Rightarrow \text{var}(\hat{\theta}) < \text{var}(\theta).$$

The proof completes.

**Proof of Lemma 4**

The intuition behind the proof is that we can view the difference between corresponding elements of the sequence $b$ and sequence $c$ as “updates”, and we can find that the updates lead to smaller elements “catch up” larger elements. Alternatively, we can obtain the new sequence $c$ through a set of updates. First, we update the whole sequence $b$ by the update value of the largest elements, then the largest elements have found their correct values. Then we pick up the elements that are smaller than the largest elements, and update those by the update value of the second largest elements minus the previous update, then the second largest elements have found their correct values. In this manner, we can obtain a sequence of sequences, where the first sequence is $b$, the second sequence is $b'$, the last sequence is $c$, and the adjacent sequences only differ by a simpler update: to the left of some element, each element is updated by the same value; and to the right of the element, each value remains unchanged. We can prove that such simpler update can guarantee to decrease the variance under certain conditions, and we can use that to prove $\text{var}(c) \leq \text{var}(b') < \text{var}(b)$.

The formal proof starts here. First, following the intuition stated above, we construct a sequence of sequences with decreasing variance, in which the variance of the first sequence is $\text{var}(b)$ and the variance of the last sequence is $\text{var}(c)$. We sort the unique values in $b$ in ascending order and denote the resultant sequence as $d = (d_j)_{j=1}^m$. Let $l(j) = \max\{i : b_i = d_j\}$, $u(i) = \{j : d_j = b_i\}$, we construct a sequence of sequences $h^{(j)} = (h_i^{(j)})_{i=1}^n$ where $j = 1, 2, \cdots, m + 1$, in the following way:

- $h_i^{(1)} = b_i$, where $i = 1, 2, \cdots, n$;
- $h_i^{(j+1)} = h_i^{(j)}$, where $j = 1, 2, \cdots, m$ and $l(m - j + 1) < i \leq n$;
- $h_i^{(2)} = h_i^{(1)} + g(d_m)$, where $1 \leq i \leq l(m)$;
- $h_i^{(j+1)} = h_i^{(j)} + g(d_{m-j+1}) - g(d_{m-j+2})$, where $j = 2, 3, \cdots, m$ and $1 \leq i \leq l(m - j + 1)$.

From the definition of $h^{(j)}$, we know $\text{var}(h^{(1)}) = \text{var}(b)$. As $b_1 < b_n$, we have $m \geq 2$. Now we prove that $\text{var}(h^{(m+1)}) = \text{var}(c)$ and $\forall j = 1, 2, \cdots, m$, $\text{var}(h^{(j+1)}) < \text{var}(h^{(j)})$. First, we prove $\text{var}(h^{(m+1)}) = \text{var}(c)$. Actually, we can prove $h^{(m+1)} = c$:

$$h_i^{(m+1)} = \sum_{j=1}^m (h_i^{(j+1)} - h_i^{(j)}) + h_i^{(1)}$$

$$= \sum_{j=1}^{m+1-u(i)} (h_i^{(j+1)} - h_i^{(j)}) + \sum_{j=m+2-u(i)}^m (h_i^{(j+1)} - h_i^{(j)}) + b_i.$$
As \( j \geq m + 2 - u(i) \iff u(i) \leq m + 2 - j \iff d_{m+2-j} \geq d_{u(i)} = b_i \iff l(m + 1 - j) < i \), we know that

\[
\begin{cases}
    h_i^{(j)}, \text{ when } j \geq m + 2 - u(i) \\
    h_i^{(j)} + g(d_{m-j+1}) - g(d_{m-j+2}), \text{ when } 2 \leq j \leq m + 1 - u(i) \\
    h_i^{(j)} + g(d_{m}), \text{ when } j = 1.
\end{cases}
\]

So we have

\[
h_i^{(m+1)} = \sum_{j=1}^{m+1-u(i)} (h_i^{(j+1)} - h_i^{(j)}) + b_i
\]

\[
= g(d_m) + \sum_{j=2}^{m+1-u(i)} (g(d_{m-j+1}) - g(d_{m-j+2})) + b_i
\]

\[
= g(d_m) + g(d_{u(i)}) - g(d_m) + b_i
\]

\[
= g(b_i) + b_i
\]

\[
= c_i.
\]

So \( \text{var}(h^{(m+1)}) = \text{var}(c) \). Then we prove that \( \forall j = 1, 2, \cdots, m, \text{var}(h^{(j+1)}) < \text{var}(h^{(j)}) \). First, we need to prove that for any \( j \), \( h_i^{(j)} \) is a non-decreasing sequence in terms of \( i \). In order to prove that, we only need to prove \( \forall j = 2, \cdots, m, h_i^{(j+1)} - h_i^{(j)} < h_{l(m-j+1)+1}^{(j)} - h_{l(m-j+1)}^{(j)} \). Then from \( h^{(j+1)} \) to \( h^{(j)} \), since elements before \( l(m - j + 1) \) are updated by the same value, and elements after \( l(m - j + 1) + 1 \) are unchanged, if the \( l(m - j + 1)^{th} \) element does not exceed the \( l(m - j + 1) + 1^{th} \) element, then the order of the whole sequence remains the same during the update. The proof is as follows: \( \forall j \geq 2 \),

\[
h_{l(m-j+1)+1}^{(j)} - h_{l(m-j+1)}^{(j)} = \sum_{k=1}^{j-1} (h_{l(m-j+1)+1}^{(k+1)} - h_{l(m-j+1)+1}^{(k)}) + h_{l(m-j+1)}^{(1)}.
\]

As \( k \leq j - 1 \Rightarrow l(m - k + 1) \geq l(m - j + 2) = l(m - j + 1) \geq l(m - j + 1) + 1 \),

from the definition of \( h \) we know that

\[
h_{l(m-j+1)+1}^{(k+1)} - h_{l(m-j+1)+1}^{(k)} = \begin{cases}
    g(d_{m-k+1}) - g(d_{m-k+2}), \text{ when } k \geq 2 \\
    g(d_m), \text{ when } k = 1.
\end{cases}
\]

So we have

\[
h_{l(m-j+1)+1}^{(j)} = \sum_{k=1}^{j-1} (h_{l(m-j+1)+1}^{(k+1)} - h_{l(m-j+1)+1}^{(k)}) + h_{l(m-j+1)}^{(1)}
\]

\[
= g(d_m) + \sum_{k=2}^{j-1} (g(d_{m-k+1}) - g(d_{m-k+2})) + b_{l(m-j+1)+1}
\]

\[
= g(d_{m-j+2}) + b_{l(m-j+1)+1}.
\]

From the definition of \( l(\cdot) \), we have that \( b_{l(m-j+1)+1} = d_{m-j+2} \), so \( h_{l(m-j+1)+1}^{(j)} = g(b_{l(m-j+1)}) + b_{l(m-j+1)+1} \). Similarly, \( h_{l(m-j+1)}^{(j)} = b_{l(m-j+1)} + g(b_{l(m-j+2)}) = b_{l(m-j+1)} + g(b_{l(m-j+1)}) -
\((g(d_{m-j+1}) - g(d_{m-j+2})).\) So \(h_{l(m-j+1)+1}^{(j)} - h_{l(m-j+1)}^{(j)} = b_{l(m-j+1)+1} - b_{l(m-j+1)}\) 
\(+ (g(b_{l(m-j+1)+1}) - g(b_{l(m-j+1)}) + (g(d_{m-j+1}) - g(d_{m-j+2})).\) As the function \(g(x)\) is bounded 
between 0 and \(b_{l+1} - b_l,\) we have \(g(b_{l(m-j+1)+1}) - g(b_{l(m-j+1)}) > -(b_{l(m-j+1)+1} - b_{l(m-j+1)}).\) 
So \(h_{l(m-j+1)+1}^{(j)} - h_{l(m-j+1)}^{(j)} = b_{l(m-j+1)+1} - b_{l(m-j+1)} + (g(b_{l(m-j+1)+1}) - g(b_{l(m-j+1)}))\) 
\+ (g(d_{m-j+1}) - g(d_{m-j+2})) > 0 + (g(d_{m-j+1}) - g(d_{m-j+2})) = g(d_{m-j+1}) - g(d_{m-j+2}).\) 
\(\text{(2.55)}\)

As \(h_{i+1}^{(j+1)} - h_i^{(j)}\) is either 0 or \(g(d_{m-j+1}) - g(d_{m-j+2})\) which is positive, we have proved \(\forall j \geq 2, h_{i+1}^{(j+1)} - h_i^{(j)} < h_{i+1}^{(j)} - h_i^{(j)}\) \(\forall j \geq 2, \) according to the former discussion, for a fixed \(j, h_i^{(j)}\) is a non-decreasing sequence.

We can prove \(\text{var}(h_{i+1}^{(j)}) < \text{var}(h_i^{(j)})\) now. If \(j = 1, l(m) = n, \forall i = 1, 2, \ldots, n, h_i^{(2)} = h_i^{(1)} = g(d_m),\) so \(\text{var}(h_i^{(2)}) = \text{var}(h_i^{(1)}).\) For \(j \geq 2,\) let \(\Delta_i^{(j)} = g(d_{m-j+1}) - g(d_{m-j+2}),\) let 
\(l = l(m-j+1),\) we first use the recursive definition of \(h\) to represent \(h_{i+1}^{(j+1)}\) by \(h_i^{(j)}:\)

\[
\text{var}(h_{i+1}^{(j+1)}) = \frac{1}{n} \sum_{i=1}^{n} (h_{i+1}^{(j+1)} - \frac{1}{n} \sum_{i=1}^{n} h_{i}^{(j+1)})^2 \\
= \frac{1}{n} \sum_{i=1}^{n} (h_{i}^{(j+1)} + \mathbb{I}(i \leq l)\Delta_i^{(j)}) - \frac{1}{n} \sum_{i=1}^{n} h_{i}^{(j)} - \frac{l}{n} \Delta_i^{(j)}.
\]

Then following simple algebra to expand the above equation, we have

\[
\text{var}(h_{i+1}^{(j+1)}) = \text{var}(h_i^{(j)}) + \frac{l \Delta_i^{(j)}}{n} [2(\frac{1}{l} \sum_{i=1}^{l} h_{i}^{(j)} - \frac{1}{n} \sum_{i=1}^{n} h_{i}^{(j)}) + \frac{n-l}{n} \Delta_i^{(j)}] \\
= \text{var}(h_i^{(j)}) + \frac{l \Delta_i^{(j)}}{n} [2(\frac{1}{l} \sum_{i=1}^{l} h_{i}^{(j)} - \frac{1}{n} \sum_{i=1}^{n} h_{i}^{(j)}) - \frac{l}{n} \sum_{i=1}^{n} h_{i}^{(j)}] + \frac{n-l}{n} \Delta_i^{(j)}.
\]

Noting that \(h_{i+1}^{(j)} - h_i^{(j)} = \Delta_i^{(j)},\) we can further obtain

\[
\text{var}(h_{i+1}^{(j+1)}) = \text{var}(h_i^{(j)}) + \frac{l \Delta_i^{(j)}}{n} [2(\frac{1}{l} \sum_{i=1}^{l} h_{i}^{(j)} - \frac{1}{n} \sum_{i=1}^{n} h_{i}^{(j)}) - \frac{l}{n} \sum_{i=1}^{n} (h_{i}^{(j)} - h_{i+1}^{(j)} + h_{i}^{(j)})) - \frac{n-l}{n} \Delta_i^{(j)}].
\]

Since for a fixed \(j, h_i^{(j)}\) is a non-decreasing sequence, we have \(\forall i \geq l + 1, h_i^{(j)} - h_{i+1}^{(j)} + h_i^{(j)} \geq h_{i+1}^{(j)} - h_{i+1}^{(j)} = 0,\) and \(\frac{1}{n} \sum_{i=1}^{n} h_{i}^{(j)} \leq h_{i+1}^{(j)}.\) So \(\frac{1}{n} \sum_{i=1}^{l} h_{i}^{(j)} - \frac{l}{n} \sum_{i=1}^{n} h_{i}^{(j)} - \frac{l}{n} \sum_{i=1}^{n} (h_{i}^{(j)} - h_{i+1}^{(j)} + h_{i}^{(j)})) \leq 0\) and \(\text{var}(h_{i+1}^{(j+1)}) \leq \text{var}(h_i^{(j)}) - \frac{l \Delta_i^{(j)}}{n} \frac{n-l}{n} \Delta_i^{(j)} < \text{var}(h_i^{(j)}).\)

Putting the above results together, since \(\text{var}(h_{i+1}^{(j+1)}) \leq \text{var}(h_i^{(j)})\) and \(\text{var}(h_{i+1}^{(j+1)}) = \text{var}(c),\) we know that \(\text{var}(c) < \text{var}(h_i^{(j)}) = \text{var}(b).\) Furthermore, let \(n' = \max\{j : b_j \neq b_n\},\) then 
\(\forall i, h_{i}^{(2)} = h_i^{(1)} + g(d_m) = b_i + g(b_n), h_i^{(3)} = h_i^{(2)} + (g(d_{m-1}) - g(d_m)) \mathbb{I}(i \leq l(m-1))) = b_i + g(b_n) + (g(b_n) - g(b_n)) \mathbb{I}(i \leq n') = b_i',\) so \(\text{var}(c) \leq \text{var}(b') < \text{var}(b).\) The proof completes.
Solve $\widehat{W}$

The sub-problem defined over $\widehat{W}$ is

$$
\min_{\widehat{W}} \mathcal{L}(\widehat{W}) + \sum_{r=1}^{R} (y^{(r)}_1 \cdot \widehat{w}_{p(r)} + y^{(r)}_2 \cdot \widehat{w}_{q(r)}) + \rho \|\widehat{w}_{p(r)} - v_1^{(r)}\|^2_2 + \rho \|\widehat{w}_{q(r)} - v_2^{(r)}\|^2_2
$$
(2.56)

s.t. \quad \forall j, \|\widehat{w}_j\|_2 = 1.

In sparse coding, this problem becomes

$$
\min_{\widehat{W}} \frac{1}{2} \sum_{i=1}^{n} \|x_i - \sum_{j=1}^{m} \alpha_{ij} g_j \widehat{w}_j\|^2_2 + \lambda_2 \sum_{j=1}^{m} \|g_j \widehat{w}_j\|^2_2 + \sum_{r=1}^{R} (y^{(r)}_1 \cdot \widehat{w}_{p(r)} + y^{(r)}_2 \cdot \widehat{w}_{q(r)} + \rho \|\widehat{w}_{p(r)} - v_1^{(r)}\|^2_2)
$$
$$
+ \|\widehat{w}_{q(r)} - v_2^{(r)}\|^2_2)
$$
(2.57)

s.t. \quad \forall j, \|\widehat{w}_j\|_2 = 1.

We solve this sub-problem using coordinate descent. At each iteration, we update $\widehat{w}_j$ by fixing the other variables. The sub-problem defined on $\widehat{w}_j$ can be written as

$$
\min_{\widehat{w}_j} \frac{1}{2} \sum_{i=1}^{n} \|x_i - \sum_{k \neq j}^{m} \alpha_{ik} g_k \widehat{w}_j\|^2_2 + \lambda_2 \sum_{j=1}^{m} \|g_j \widehat{w}_j\|^2_2 + \sum_{r=1}^{R} (\|p(r) = j\|y^{(r)}_1 \cdot \widehat{w}_{p(r)} + \|q(r) = j\|y^{(r)}_2 \cdot \widehat{w}_{q(r)} + \rho \|\widehat{w}_{p(r)} - v_1^{(r)}\|^2_2)
$$
$$
+ \|\widehat{w}_{q(r)} - v_2^{(r)}\|^2_2)
$$
(2.58)

s.t. \quad \forall j, \|\widehat{w}_j\|_2 = 1,

where $b_i = x_i - \sum_{k \neq j}^{m} \alpha_{ik} g_k \widehat{w}_j$, Let $\gamma$ be the KKT multiplier. According to the KKT conditions, we have

$$
\sum_{i=1}^{n} \alpha_{ij} g_j (\alpha_{ij} g_j \widehat{w}_j - b_i) + 2\lambda_2 g_j^2 \widehat{w}_j + y_j + 2\rho N \widehat{w}_j - 2\rho v_j + 2\gamma \widehat{w}_j = 0,
$$
(2.59)

and

$$
\|\widehat{w}_j\|_2 = 1,
$$
(2.60)

where $y_j = \sum_{r=1}^{R} (\|p(r) = j\|y_1^{(r)} + \|q(r) = j\|y_2^{(r)})$, $v_j = \sum_{r=1}^{R} (\|p(r) = j\|v_1^{(r)} + \|q(r) = j\|v_2^{(r)})$, and $N = \sum_{r=1}^{R} (\|p(r) = j\| + \|q(r) = j\|)$. From Eq.(2.59), we get

$$
\widehat{w}_j = \frac{\sum_{i=1}^{n} \alpha_{ij} g_j b_i - y_j + 2\rho v_j}{\sum_{i=1}^{n} \alpha_{ij}^2 g_j^2 + 2\lambda_2 g_j^2 + 2\rho N + 2\gamma}.
$$
(2.61)

Plugging Eq.(2.61) into Eq.(2.60), we get the solution of $\gamma$. Then plugging $\gamma$ back into Eq.(2.61), we get the solution of $\widehat{w}_j$. 

50
Solve $v_1^{(r)}$, $v_2^{(r)}$ in Case 3 and 4

**Case 3** We assume $\lambda_1 = 0$ and $\lambda_2 > 0$, then

$$
(\rho + 2\gamma_1)v_1^{(r)} - \lambda_2 v_2^{(r)} = y_1^{(r)} + \rho \tilde{w}_{p(r)} \tag{2.62}
$$

$$
(\rho + 2\gamma_2)v_2^{(r)} - \lambda_2 v_1^{(r)} = y_2^{(r)} + \rho \tilde{w}_{q(r)}. \tag{2.63}
$$

According to the complementary slackness condition, we know $v_1^{(r)} \cdot v_2^{(r)} = -\tau$. For the vectors on both sides of Eq. (2.62), taking the square of their $\ell_2$ norm, we get

$$
(\rho + 2\gamma_1)^2 + \lambda_2^2 + 2(\rho + 2\gamma_1)\lambda_2 = \|y_1^{(r)} + \rho \tilde{w}_{p(r)}\|_2^2. \tag{2.64}
$$

Similarly, from Eq. (2.63), we get

$$
(\rho + 2\gamma_2)^2 + \lambda_2^2 + 2(\rho + 2\gamma_2)\lambda_2 = \|y_2^{(r)} + \rho \tilde{w}_{q(r)}\|_2^2. \tag{2.65}
$$

Taking the inner product of the two vectors on the left hand sides of Eq. (2.62, 2.63), and those on the right hand sides, we get

$$
-(2\rho + 2\gamma_1 + 2\gamma_2)\lambda_2 - ((\rho + 2\gamma_1)(\rho + 2\gamma_2) + \lambda_2^2)\tau = (y_1^{(r)} + \rho \tilde{w}_{p(r)})^\top (y_2^{(r)} + \rho \tilde{w}_{q(r)}). \tag{2.66}
$$

Solving the system of equations consisting of Eq. (2.64, 2.66), we obtain the optimal values of $\gamma_1$, $\gamma_2$, and $\lambda_2$. Plugging them into Eq. (2.62) and Eq. (2.63), we obtain a solution of $v_1^{(r)}$ and $v_2^{(r)}$. Then we check whether this solution satisfies $v_1^{(r)} \cdot v_2^{(r)} = -\tau$. If so, this is an optimal solution.

**Case 4** We assume $\lambda_1 = 0$ and $\lambda_2 = 0$, then according to the complementary slackness condition, we know $v_1^{(r)} \cdot v_2^{(r)} = \tau$ and $v_1^{(r)} \cdot v_2^{(r)} = -\tau$. This only holds when $\tau = 0$. Then the sub-problem defined on $v_1^{(r)}$, $v_2^{(r)}$ becomes

$$
\min_{v_1^{(r)}, v_2^{(r)}} -y_1^{(r)} \cdot v_1^{(r)} - y_2^{(r)} \cdot v_2^{(r)} + \frac{\rho}{2} \|\tilde{w}_{p(r)} - v_1^{(r)}\|_2^2 + \frac{\rho}{2} \|\tilde{w}_{q(r)} - v_2^{(r)}\|_2^2
$$

s.t. $\|v_1^{(r)}\|_2 = 1, \|v_2^{(r)}\|_2 = 1, v_1^{(r)} \cdot v_2^{(r)} = 0$.

Let $\gamma_1$, $\gamma_2$, $\gamma_3$ be the KKT multipliers associated with the three constraints in this sub-problem. According to the KKT conditions, we have

$$
(2\gamma_1 + \rho)v_1^{(r)} + \gamma_3 v_2^{(r)} = y_1^{(r)} + \rho \tilde{w}_{p(r)} \tag{2.67}
$$

$$
(2\gamma_2 + \rho)v_2^{(r)} + \gamma_3 v_1^{(r)} = y_2^{(r)} + \rho \tilde{w}_{q(r)}. \tag{2.68}
$$

For the vectors on both sides of Eq. (2.67), taking the square of their $\ell_2$ norm, we get

$$
(2\gamma_1 + \rho)^2 + \gamma_3^2 = \|y_1^{(r)} + \rho \tilde{w}_{p(r)}\|_2^2. \tag{2.69}
$$
For the vectors on both sides of Eq. (2.68), taking the square of their $\ell_2$ norm, we get
\[(2\gamma_2 + \rho)^2 + \gamma_3^2 = \|y_2^{(r)} + \rho \tilde{w}_{q(r)}\|^2_2.\] (2.70)

Taking the inner product of the two vectors on the left hand sides of Eq. (2.67, 2.68), and those on the right hand sides, we get
\[(2\rho + 2\gamma_1 + 2\gamma_2)\gamma_3 = (y_1^{(r)} + \rho \tilde{w}_{p(r)})^\top (y_2^{(r)} + \rho \tilde{w}_{q(r)}).\] (2.71)

Solving the system of equations consisting of Eq. (2.69, 2.71), we obtain the optimal values of $\gamma_1$, $\gamma_2$, and $\gamma_3$. Plugging them into Eq. (2.67) and Eq. (2.68), we obtain a solution of $v_1^{(r)}$ and $v_2^{(r)}$. This is an optimal solution.

### 2.4 Diversity in the RKHS: Orthogonality-promoting Regularization of Kernel Methods

In previous sections, we study how to promote diversity among finite-dimensional vectors. In this section, we extend the study to infinite-dimensional functions [374] in the reproducing kernel Hilbert space (RKHS) [294], which is presumably more challenging.

Kernel methods perform learning in reproducing kernel Hilbert spaces (RKHSs) of functions [294]. The RKHS represents a high-dimensional feature space that can capture nonlinear patterns in the lower-dimensional observed data. This Hilbert space is associated with a kernel function $k$, and the inner product in the RKHS can be implicitly computed by evaluating $k$ in the lower-dimensional input space (known as the kernel trick). Well-established kernel methods include support vector machine [294], kernel principal component analysis [293], kernel independent component analysis [25], to name a few. Even though their large model-capacity leads to high representational power, it also incurs substantial risk of overfitting.

One key ingredient in kernel methods is regularization, which reduces overfitting by controlling the complexity of the RKHS functions [251, 294]. Regularizers proposed previously such as RKHS norm, derivatives, green functions, and splines mostly focus on encouraging a small norm [251] and smoothness of functions [294]. Notably, the most widely-used regularizer is the squared RKHS norm.

We are interested in whether diversity-promoting regularization can outperform the existing regularizers in reducing overfitting and whether its ability of better capturing infrequent patterns and reducing model size without sacrificing modeling power holds in the RKHS. Similar to Section 2.2 and 2.3, we use near-orthogonality to characterize diversity. Similar to Section 2.2.1 to promote near-orthogonality among a set of RKHS functions $\{f_i\}_{i=1}^K$, we compute their Gram matrix $G$ where $G_{ij} = \langle f_i, f_j \rangle$, and encourage $G$ to be close to an identity matrix $I$ where the closeness is measured using the Bregman matrix divergences [100]. We apply the proposed BMD regularizers to two kernel methods – kernel distance metric learning (KDML) [176, 335] and kernel sparse coding (KSC) [120], and develop an optimization algorithm based on the alternating direction method of multipliers (ADMM) [52] where the RKHS functions are learned using functional gradient descent (FGD) [89]. Experimental results show that the proposed near-orthogonality regularizers (1) greatly improve the generalization performance of KDML and
KSC; (2) can reduce model size without sacrificing modeling power; (3) can better capture infrequent patterns in the data; and (4) outperform other orthogonality-promoting regularizers and the squared Hilbert norm.

2.4.1 Bregman Matrix Divergence Regularized Kernel Methods

We consider kernel methods that are parameterized by a set of RKHS functions $\mathcal{F} = \{f_i\}_{i=1}^K$. Examples include kernel principal component analysis (PCA) [293], kernel independent component analysis (ICA) [25], kernel distance metric learning [176, 335] and kernel sparse coding [120], to name a few. We promote diversity among these functions by encouraging them to be close to being orthogonal. Similar to the vector case in Section 2.2.1, we compute a Gram matrix $G$ where $G_{ij} = \langle f_i, f_j \rangle_{\mathcal{H}}$ and encourage $G$ to be close to an identity matrix $I$ by minimizing the Bregman matrix divergences (BMD) between $G$ and $I$. A family of BMD regularizers can be defined, based on the (1) squared Frobenius norm (SFN): $\|G - I\|_F^2$; (2) von Neumann divergence (VND): $\text{tr}(G \log G - G)$; (3) log determinant divergence (LDD): $\text{tr}(G) - \log \det(G)$. To apply the VND and LDD regularizers, the Gram matrix $G$ is required to be positive definite. In our experiments, this condition is always satisfied since VND and LDD encourage the RKHS functions to be close to being orthogonal (therefore linearly independent). We use these regularizers to encourage near-orthogonality among RKHS functions, and define BMD regularized kernel methods (BMD-KM):

$$\min_{\mathcal{F}} \mathcal{L}(\mathcal{F}) + \lambda \Omega(\mathcal{F}), \quad (2.72)$$

where $\mathcal{L}(\mathcal{F})$ is the objective function of the kernel method, and $\lambda$ is the regularization parameter. Compared to kernel PCA and ICA in which the functions are required to be strictly-orthogonal, BMD-KM can be seen as a relaxed counterpart where the functions are encouraged to be close to, but not necessarily strictly, being orthogonal. As we will demonstrate in the experiments, strict-orthogonality can compromise performance in certain applications.

We apply the BMD regularizers to two instances of kernel methods: kernel distance metric learning and kernel sparse coding.

Case study I: kernel distance metric learning (KDML) KDML [176, 335] is equipped with $K$ RKHS functions $\mathcal{F} = \{f_i\}_{i=1}^K$ that map a data example $x$ into a vector $h(x)$ in a $K$-dimensional latent space, where $h(x) = f_i(x)$. Given two examples $x$ and $y$, their distance is defined as $d_\mathcal{F}(x, y) = \|h(x) - h(y)\|_2^2$, which is parameterized by $\mathcal{F}$. Given $N$ training examples, $\{x_n, y_n, t_n\}_{n=1}^N$, where $x_n$ and $y_n$ are similar if the label $t_n$ equals 1 and dissimilar if $t_n = 0$, following [146], we learn the distance metric by minimizing $\sum_{n=1}^N \log(1 + \exp((2t_n - 1)d_\mathcal{F}(x_n, y_n)))$. Using $\Omega(\mathcal{F})$ to promote near-orthogonality, we obtain the BMD-regularized KDML (BMD-KDML) problem:

$$\min_{\mathcal{F}} \sum_{n=1}^N \log(1 + \exp((2t_n - 1)d_\mathcal{F}(x_n, y_n))) + \lambda \Omega(\mathcal{F}). \quad (2.73)$$

Case study II: kernel sparse coding (KSC) In KSC [120], a data example $x$ is mapped into $k(x, \cdot)$ in an RKHS induced by the kernel function $k(\cdot, \cdot)$ and a dictionary of RKHS functions
\[ F = \{ f_i \}_{i=1}^K \] are learned to reconstruct \( k(x, \cdot) \). Given the training data \( \{ x_n \}_{n=1}^N \), \( F \) can be learned by minimizing \( \frac{1}{2} \sum_{n=1}^N \| k(x_n, \cdot) - \sum_{i=1}^K a_n f_i \|_H^2 + \lambda_1 \sum_{n=1}^N \| a_n \|_1 + \frac{\lambda_2}{2} \sum_{i=1}^K \| f_i \|_H^2 \), where the reconstruction error is measured by the squared Hilbert norm and \( a_n \) are the linear coefficients. The \( \ell_1 \) regularizer \( \| a_n \|_1 \) is applied to encourage the coefficients to be sparse. To avoid the degenerated case where the RKHS functions are of large norm while the coefficients are close to zero, the squared Hilbert norm regularizer \( \| f_i \|_H^2 \) is applied to the RKHS functions to keep their magnitude small. By adding \( \Omega(F) \), we obtain the BMD-regularized KSC (BMD-KSC) problem:

\[
\min_{F, A} \ \frac{1}{2} \sum_{n=1}^N \| k(x_n, \cdot) - \sum_{i=1}^K a_n f_i \|_H^2 + \lambda_1 \sum_{n=1}^N \| a_n \|_1 + \frac{\lambda_2}{2} \sum_{i=1}^K \| f_i \|_H^2 + \lambda_3 \Omega(F),
\]

where \( A \) denotes all the sparse codes.

### 2.4.2 A Functional Gradient Descent Algorithm

In this section, we develop an ADMM [52] based algorithm to solve the BMD-KM problem. First, by introducing auxiliary variables \( \tilde{F} = \{ \tilde{f}_i \}_{i=1}^K \) which are a set of RKHS functions and \( A \in \mathbb{R}^{K \times K} \), we rewrite the BMD-KM problem into an equivalent form that is amenable for developing ADMM-based algorithms:

\[
\begin{align*}
\min_{F, \tilde{F}, A} & \ \mathcal{L}(F) + \lambda D_\phi(A, I) \\
\text{s.t.} & \ \forall i, f_i = \tilde{f}_i \\
\end{align*}
\]

where \( A \) is required to be positive definite when \( D_\phi(A, I) \) is a VND or LDD regularizer. The constraints \( (f_i, \tilde{f}_i) = A_{ij} \) and \( (f_i, \tilde{f}_j) = A_{ji} \) imply that \( A \) is symmetric. We define the augmented Lagrangian with parameter \( \rho > 0 \): \( \mathcal{L}(F) + \lambda D_\phi(A, I) + \sum_{i=1}^K \| g_i - \tilde{f}_i \|_H + \sum_{i=1}^K \sum_{j=1}^K (P_{ij} (f_i, \tilde{f}_j) - A_{ij}) + Q_{ij} ((f_i, \tilde{f}_j) - A_{ij}) + \frac{\rho}{2} ((f_i, \tilde{f}_j) - A_{ij}) + \frac{\rho}{2} ((f_i, \tilde{f}_j) - A_{ij})^2 \), where \( G = \{ g_i \}_{i=1}^K \) is another set of RKHS functions, and \( P, Q \in \mathbb{R}^{K \times K} \) are Lagrange multipliers. Then we minimize this Lagrangian function by alternating among \( F, \tilde{F}, G, A, P, Q \).

**Solve A** Given \( H \in \mathbb{R}^{K \times K} \) where \( H_{ij} = \langle f_i, \tilde{f}_j \rangle \), we learn \( A \) by minimizing \( \lambda D_\phi(A, I) - \langle P, A \rangle - \langle Q^T, A \rangle + \frac{\rho}{2} \| H - A \|_F^2 + \frac{\rho}{2} \| H^T - A \|_F^2 \), which is a convex problem. \( D_\phi(A, I) \) has three cases, which we discuss separately. When \( D_\phi(A, I) \) is the VND, we first perform an eigen-decomposition of \( D = P + Q^T + \rho (H + H^T) : D = \Phi \Sigma \Phi^{-1} \), then the optimal solution of \( A \) can be obtained as \( A = \Phi \Sigma \Phi^{-1} \) where

\[
\Sigma_{ii} = \frac{\lambda \omega \left( \frac{\Sigma_{ii}}{\lambda} - \log \left( \frac{\Sigma_{ii}}{2\rho} \right) \right)}{2\rho},
\]

and \( \omega(\cdot) \) is the Wright omega function [137]. It can be shown that \( A \) is positive definite. When \( D_\phi(A, I) \) is the LDD, the optimal solution is:

\[
A = -\frac{1}{2} B + \frac{1}{2} \sqrt{B^2 - 4C},
\]

54
where $B = \frac{1}{\rho}(\lambda I - P - Q^T - \rho(H + H^T))$ and $C = -\frac{2}{\rho}I$. It can be verified that $A$ is positive definite. When $D_\phi(A, I)$ is the SFN, the optimal solution for $A$ is:

$$A = (2\lambda I + P + Q^T + \rho(H + H^T))/(2\lambda + 2\rho).$$  \hspace{1cm} (2.78)

**Solve $f_i$**  We solve $f_i$ by minimizing $\Upsilon = L(\mathcal{F}) + \langle g_i, f_i \rangle + \sum_{j=1}^K (P_{ij} + Q_{ij})\langle f_i, \hat{f}_j \rangle + \frac{\rho}{2}\sum_{j=1}^K((\langle f_i, \hat{f}_j \rangle - A_{ij})^2 + ((f_i, \hat{f}_j) - A_{ij})^2)$. The first issue we need to address is how to represent $f_i$. When $f \in \mathcal{F}$ is regularized by the RKHS norm, according to the representer theorem [294], the optimal solution $f^*$ can be expressed as a linear combination of kernel functions evaluated at the training data: $f^*(x) = \sum_{n=1}^N \alpha_n k(x_n, x)$, which we refer to as *representer theorem representation* (RTR). This endows $f^*$ an explicit parametrization that greatly eases learning: the search space of $f^*$ is reduced from the infinite-dimensional RKHS $\mathcal{H}$ to an $N$-dimensional space of coefficients $\{\alpha_n\}_{n=1}^N$. However, in $\Upsilon$, due to the presence of the inner products between $f_i$ with other functions, the representer theorem does not hold and $f_i$ does not admit an RTR form. To address this issue, we learn $f_i$ directly using functional gradient descent [89]. A functional $F : \mathcal{H} \to \mathbb{R}$ maps functions in $\mathcal{H}$ to real numbers. A functional gradient $\nabla F[f]$ is defined implicitly as the linear term of the change in a function due to a small perturbation $\epsilon$ in its input: $F[f + \epsilon g] = F[f] + \epsilon \langle \nabla F[f], g \rangle + O(\epsilon^2)$. Of particular interest is the *evaluation functional* $F_x[f]$ which is parameterized by an input vector $x$ and evaluates $f$ at $x$: $F_x[f] = f(x)$. The functional gradient of $F_x[f]$ is $k(x, \cdot)$ [89] where $k$ is the kernel associated with the RKHS. The gradient of an inner product functional $F_y[f] = \langle f, g \rangle$ is $g$.

The form of $L(\mathcal{F})$ depends on a specific kernel method. Here, we consider KDML where $L(\mathcal{F})$ is given in Eq. (2.73). In KDML, $f_i$ appears in two types of functionals: evaluation functionals in $d_F(x_n, y_n)$ (such as $f_i(x_n)$) and inner product functionals (such as $\langle g_i, f_i \rangle$). The functional gradient of $\Upsilon$ is $\nabla \Upsilon = 2 \sum_{n=1}^N \int d_F(x_n, y_n) \int d_F(x_n, y_n) \int d_F(x_n, \cdot) - k(y_n, \cdot)) + g_i + \sum_{j=1}^K (P_{ij} + Q_{ij} + \rho(2\langle f_i, \hat{f}_j \rangle - A_{ij} - A_{ji}))\hat{f}_j$, where $\sigma(x) = 1/(1 + \exp(-x))$ is a sigmoid function. Given this functional gradient, we can perform gradient descent to update $f_i$ until convergence: $f_i \leftarrow f_i - \eta \nabla \Upsilon f_i$, where $\eta$ is the learning rate. In the algorithm, we initialize $f_i$, $g_i$, and $\hat{f}_j$ as zero functions. Then as will be proven later on, during the algorithm execution, $f_i$, $g_i$, and $\hat{f}_j$ are all in the form of RTR. So updating $f_i$ amounts to updating the linear coefficients in the RTR.

**Solve $\hat{f}_j$**  The sub-problem defined over $\hat{f}_j$ is:

$$\min_{\hat{f}_j} -\langle g_j, \hat{f}_j \rangle + \sum_{i=1}^K ((P_{ij} + Q_{ij})\langle f_i, \hat{f}_j \rangle + \frac{\rho}{2}((\langle f_i, \hat{f}_j \rangle - A_{ij})^2 + ((f_i, \hat{f}_j) - A_{ij})^2), \hspace{1cm} (2.79)$$

which is a convex problem. Setting the derivative of the objective function to zero, we get an equation:

$$0 = (2\rho \sum_{i=1}^K f_i \otimes f_i)\hat{f}_j = \sum_{i=1}^K (\rho(A_{ij} + A_{ji}) - (P_{ij} + Q_{ij}))f_i + g_j,$$ \hspace{1cm} (2.80)

where $\otimes$ denotes the outer product in the RKHS. As will be proven later on, $\hat{f}_j$, $g_j$, and $f_i$ are all in the form of RTR. Let $\Phi = [k(x_1, \cdot), \ldots, k(x_N, \cdot)]$, then $f_i = \Phi a_i$, $\sum_{i=1}^K (\rho(A_{ij} + A_{ji}) -$
\( (P_{ij} + Q_{ij})f_i + g_j = \Phi b, \ \hat{f}_j = \Phi c, \) where \( a_i, b, c \) are coefficient vectors. \( a_i \) and \( b \) are known and \( c \) is to be estimated. Then Eq. (2.80) can be written as

\[
(2\rho \sum_{i=1}^{K} a_i a_i^\top)\Phi^\top \Phi c = b, \tag{2.81}
\]

where \( (\Phi^\top \Phi)_{ij} = k(x_i, x_j) \) and \( c = ((2\rho \sum_{i=1}^{K} a_i a_i^\top)\Phi^\top \Phi)^{-1}b \). In practice, inverting the \( N \times N \) matrix \( (2\rho \sum_{i=1}^{K} a_i a_i^\top)\Phi^\top \Phi \) is computationally prohibitive when \( N \) is large. In that case, we can switch to a stochastic FGD method to solve this problem.

The update rules for \( P, Q \) and \( g_j \) are simple:

**Update P** \( P = P + \rho(H - A) \)

**Update Q** \( Q = Q + \rho(H - A^\top) \)

**Update \( g_j \)** \( g_j = g_j + \rho(f_j - \hat{f}_j) \)

**RTR form of the RKHS functions** Next, we present the proof that as long as the RKHS functions \( f_i, g_i, \) and \( \hat{f}_j \) are initialized to be zero, they are always in the RTR form during the entire execution of the algorithm. We prove this by induction. For the base case (iteration \( t = 0 \)), these functions are all zero, hence admitting the RTR form. For the inductive step, assuming the statement is true at iteration \( t - 1 \), we prove it holds for iteration \( t \). We begin with \( f_i \), which is solved by functional gradient descent (FGD). At iteration \( t \), the input of the algorithm is \( f_i^{(t-1)} \) and the output is \( f_i^{(t)} \). The first term of the functional gradient \( \triangle f_i \) is in the RTR form, so are \( g_i \) and \( \hat{f}_j \) (according to the inductive hypothesis). Then \( \triangle f_i \) is in the RTR form. Starting from \( f_i^{(t-1)} \) which is in the RTR form according to the inductive hypothesis, \( f_i \) is updated iteratively in the following way: \( f_i^{(s)} \leftarrow f_i^{(s-1)} - \eta \triangle f_i^{(s-1)} \) (where \( s \) indexes the FGD iterations), resulting in \( f_i^{(t)} \) which is also in the RTR form.

Next, we prove that if \( g_j^{(t-1)} \) and \( f_i^{(t-1)} \) are in the RTR form, so will be \( \hat{f}_j \). The proof is similar to that of the representer theorem \([294]\). We decompose \( \hat{f}_j \) into \( \hat{f}_j^\parallel \) and \( \hat{f}_j^\perp \), where \( \hat{f}_j^\parallel \) is in \( S = \{ \sum_{n=1}^{N} \alpha_n k(x_n, x) \} \) (i.e., in the RTR form) and \( \hat{f}_j^\perp \) is perpendicular to \( S \), hence \( \langle g_j, \hat{f}_j^\parallel \rangle = 0, \langle f_i, \hat{f}_j^\parallel \rangle = 0 \). The problem defined in Eq. (2.79) can be equivalently written as:

\[
\min_{\hat{f}_j^\parallel} - \langle g_j, \hat{f}_j^\parallel \rangle + \sum_{i=1}^{K} ((P_{ij} + Q_{ij})\langle f_i, \hat{f}_j^\parallel \rangle + \frac{\rho}{2}((\langle f_i, \hat{f}_j^\parallel \rangle - A_{ij})^2 + (\langle f_i, \hat{f}_j^\parallel \rangle - A_{ij})^2)). \tag{2.82}
\]

Hence the optimal solution of \( \hat{f}_j \) is in the RTR form. For \( g_j \), from its update equation \( g_j = g_j + \rho(f_j - \hat{f}_j) \), it is easy to see that if \( f_j^{(t-1)} \) and \( \hat{f}_j^{(t-1)} \) are in the RTR form, so will be \( g_j^{(t)} \). Note that these RKHS functions are in the RTR form because of the algorithmic procedure (namely, initializing these functions as zero and using FGD to solve \( f \)) rather than the representer theorem. If we choose another way of initialization, \( f \) may not be in the RTR form.
Scalable representation of the RKHS functions based on random Fourier features

When the RKHS functions are in the RTR form, \(O(N^2 D)\) computational cost is incurred where \(N\) is the number of training examples and \(D\) is the input feature dimension. On large-sized datasets, this is not scalable. In this section, we investigate a scalable representation of RKHS functions based on random Fourier features (RFFs) \([284]\). Given a shift-invariant kernel \(k(x, y) = k(x - y)\) such as the radial basis function (RBF) kernel, it can be approximated with RFFs: 

\[
k(x, y) = \langle k(x, \cdot), k(y, \cdot) \rangle \approx z(x)^\top z(y),
\]

where \(z(x) \in \mathbb{R}^Q\) is the RFF transformation of \(x\), and can be seen as an approximation of \(k(x, \cdot)\). 

\(z(x)\) is generated in the following way: (1) compute the Fourier transform \(p(\omega)\) of the kernel \(k\); (2) draw \(Q\) i.i.d samples \(\omega_1, \ldots, \omega_Q \in \mathbb{R}^D\) from \(p(\omega)\) and \(Q\) i.i.d samples \(b_1, \ldots, b_Q \in \mathbb{R}\) from the uniform distribution on \([0, 2\pi]\); (3) let 

\[
z(x) = \sqrt{\frac{2}{Q}} [\cos(\omega_1^\top x + b_1), \ldots, \cos(\omega_Q^\top x + b_Q)]^\top.
\]

For \(f \in \mathcal{H}\) where \(\mathcal{H}\) is an RKHS induced by a shift-invariant kernel, we know that \(f(x) = \langle f, k(x, \cdot) \rangle\). Using \(z(x)\) to approximate \(k(x, \cdot)\) and \(w \in \mathbb{R}^Q\) to approximate \(f\), we get 

\[
f(x) \approx w^\top z(x).
\]

As such, the infinite-dimensional function \(f\) can be approximately represented as a finite-dimensional vector \(w\), and the BMD-KM problem defined in Eq.(2.72) can be written as:

\[
\min_W \mathcal{L}(W) + \lambda \Omega(W), \tag{2.83}
\]

where \(W = \{w_i\}_{i=1}^K\) and \(\Omega(W) = D(G, I)\) with \(G_{ij} = w_i^\top w_j\). Now, learning can be conducted over \(W\), and the computational cost is reduced from \(O(N^2 D)\) to \(O(NQ)\), where \(Q\) is the number of RFFs and is much smaller than \(ND\).

2.4.3 Evaluation

In this section, we present experimental results on BMD-KDML and BMD-KSC.

Datasets

We used six datasets in the experiments: an electronic health record dataset MIMIC-III \([184]\); five image datasets including Stanford-Cars (Cars) \([202]\), Caltech-UCSD-Birds (Birds) \([358]\), Scenes-15 \([212]\), Caltech-256 \([141]\), and UIUC-Sports \([224]\). The first three were used for KDML and the last three for KSC. Their statistics are summarized in Table 2.19. For each dataset, five random train/test splits were performed, and the results were averaged over the five runs. For the MIMIC-III dataset, we extracted features from demographics (including age and gender), clinical notes (including bag-of-words and Word2Vec \([256]\)), and lab tests (including zero-order, first-order, and second-order temporal features). The total feature dimension is 7207. The features for Cars and Birds datasets were extracted using the VGG16 \([303]\) convolutional neural network trained on the ImageNet \([95]\) dataset, which were the outputs of the second fully-connected layer with 4096 dimensions. For Scenes-15, Caltech-256, and UIUC-Sport, we extracted pixel-level dense SIFT \([237]\) features where the step size and patch size were 8 and 16, respectively.

Experimental setup

For BMD-KDML and BMD-KSC, we experimented with six combinations between three regularizers including SFN, LDD, and VND, and two representations of RKHS functions including RTR and RFF. In DML experiments, two data examples were labeled as similar if belonging to the same class and dissimilar if otherwise. The learned distance
Metrics were applied for retrieval whose performance was evaluated using precision@k. Precision@k is defined as \( \frac{n}{k} \) where \( n \) is the number of examples (among the top \( k \) retrieved examples) that have the same class label with the query. We compared with three groups of baseline methods: (1) KDML (Eq. (2.73) without the regularizer \( \Omega(\mathcal{F}) \)) and its variants under different regularizers including squared Hilbert norm (SHN), DPP [429] and Angle [369]; (2) other kernel DML methods including the ones proposed in [335] (Tsang) and [176] (Jain), multiple kernels DML (MK-DML) [345] and pairwise constrained component analysis (PCCA) [252]; (3) non-kernel metric learning (ML) methods, including information theoretic ML (ITML) [92], logistic discriminant ML (LDML) [146], DML with eigenvalue optimization (DML-Eig) [398], information-theoretic semi-supervised ML via entropy regularization (Seraph) [265] and geometric mean ML (GMML) [405]; (4) Euclidean distance (EUC). For methods in group (1), the RKHS functions are represented in the RTR form. In sparse coding experiments, on top of the SIFT features, we used kernel sparse coding to learn a set of RKHS functions and represented each SIFT feature into a sparse code. To obtain image-level features, we applied max-pooling [392] and spatial pyramid matching [212, 392] over the pixel-level sparse codes. The following baselines were compared with: sparse coding (SC) [392], unregularized kernel SC (KSC) [120], KSC regularized by SHN, DPP, and Angle. In these methods, the RKHS functions are represented in the RTR form. We used 5-fold cross validation to tune the regularization parameters in \( \{10^{-5}, 10^{-4}, \ldots, 10^5\} \), the number of RKHS functions in \( \{50, 100, 200, \ldots, 500\} \), and the dimension of RFF in \( \{1, 2, \ldots, 10\} \times D \) where \( D \) is the feature dimension of the input data. The kernel function was chosen to be the radial basis function \( \exp(-\gamma \|x - y\|^2) \) where the scale parameter \( \gamma \) was tuned in \( \{10^{-3}, 10^{-2}, \ldots, 10^3\} \). The parameter \( \rho \) in the ADMM-based algorithm was set to 1. The learning rate in functional gradient descent was set to 0.001.

Results

Table 2.20 shows the retrieval precision@10 on three datasets, where we observe the following. First, BMD-KDML methods including KDML-(SFN,VND,LDD)-(RTR,RFF) greatly outperform unregularized and SHN-regularized KDML, which demonstrates that near-orthogonality regularization is an effective way to reduce overfitting. Second, BMD regularizers including SFN, VND, and LDD outperform other near-orthogonality regularizers including DPP and Angle, possibly because they are insensitive to vector scaling and amenable for optimization. Third, VND and LDD achieve comparable performance and outperform SFN, possibly because they measure near-orthogonality in a global way while SFN conducts that in a pairwise fashion. Fourth, the RFF representation of RKHS functions performs comparably to RTR, in spite of the fact that it is an approximation method. Finally, the BMD-KDML methods achieve better performance than non-kernel DML methods and other kernel DML methods, suggesting their competitive ability in learning effective distance metrics.

Table 6.2 shows the number of RKHS functions under which the precision@10 in Table 2.20 is achieved. It can be seen that the BMD-KDML methods utilize much fewer functions than KDML while achieving better precision@10. For instance, KDML-VND-RTR achieves 77.1% precision@10 with 100 functions on the MIMIC-III dataset while KDML achieves 73.8% precision@10 with 300 functions. These results demonstrate the ability of the BMD regularizers in reducing model size without sacrificing modeling power. By encouraging the functions to be near-orthogonal, the BMD regularizers decrease the redundancy among functions and make

58
the functions highly complementary. As a result, a small number of such functions are able to capture the patterns in the data sufficiently well. In addition, the BMD regularizers achieve better precision@10 with fewer functions than other near-orthogonality regularizers including DPP and Angle, suggesting their better efficacy in promoting near-orthogonality.

In the next experiment, we investigate whether near-orthogonality regularization can better capture infrequent patterns. We select 3 frequent diseases (patterns) and 5 infrequent ones from the MIMIC-III dataset. A disease is regarded as being frequent if the number of hospital admissions diagnosed with this disease is greater than 300. Table 2.22 shows the precision@10 on the 8 diseases, from which we observe that: (1) on the 5 infrequent diseases (labeled as D4–D8), the BMD-KDML methods achieve much higher precision@10 than the unregularized KDML, suggesting that by encouraging the functions to be close to being orthogonal, the BMD regularizers can better capture infrequent patterns; (2) on the 3 frequent diseases (labeled as D1–D3), the precision@10 achieved by the BMD-KDML methods is comparable with that achieved by the unregularized KDML, indicating that the BMD regularizers do not compromise the modeling efficacy on the frequent patterns. On the infrequent diseases, the BMD-KDML methods outperform KDML-DPP and KDML-Angle, suggesting that the BMD regularizers have better ability in promoting near-orthogonality than DPP and Angle.

Further, Figure 2.6 shows how the precision@10 on MIMIC-III varies as we increase the regularization parameter $\lambda$ in KDML-VND-RTR. As can be seen, the best precision@10 is achieved under a modest $\lambda$. A very large $\lambda$ would make the functions strictly orthogonal, as kernel PCA and ICA do, which would result in excessively strong regularization and therefore poor performance.

We also compare the convergence time of BMD-KDML under two representations: RTR and RFF. As shown in Table 2.23, RFF results in much faster convergence since this representation does not depend on the training data. While computationally efficient, RFF does not sacrifice much modeling power. Table 2.20 shows that RFF achieves precision@10 that is comparable to RTR.

Next, we present the kernel sparse coding results. Table 2.24 shows the classification accuracy on three datasets, from which we observe similar results as in the KDML experiments. First, KSC-(SFN,VND,LDD)-(RTR,RFF) achieve better accuracy than the unregularized and the SHN-regularized KSC. Second, the BMD regularizers outperform other near-orthogonality reg-

Figure 2.6: Precision@10 versus the regularization parameter $\lambda$ on MIMIC-III.
ularizers including DPP and Angle. Third, VND and LDD are superior to SFN. Fourth, the RFF representation of RKHS functions performs comparably to RTR. Finally, the BMD-KSC methods outperform the non-kernel SC methods and other kernel SC methods. These observations demonstrate the efficacy of the BMD regularizers in reducing overfitting and promoting near-orthogonality.

2.4.4 Appendix: Details of Algorithms

Detailed Derivation of Solving A

Given $H \in \mathbb{R}^{K \times K}$ where $H_{ij} = \langle f_i, \hat{f}_j \rangle$, the sub-problem defined over $A$ is

$$\min_A \lambda D_\phi(A, I) - \langle P, A \rangle - \langle Q^T, A \rangle + \frac{\rho}{2} ||H - A||_F^2 + \frac{\rho}{2} ||H^T - A||_F^2. \quad (2.84)$$

$D_\phi(A, I)$ has three cases, which we discuss separately. When $D_\phi(A, I)$ is the squared Frobenius norm, the problem becomes

$$\min_A \lambda ||A - I||_F^2 - \langle P, A \rangle - \langle Q^T, A \rangle + \frac{\rho}{2} ||H - A||_F^2 + \frac{\rho}{2} ||H^T - A||_F^2. \quad (2.85)$$

Taking the derivative and setting it to zero, we get the optimal solution for $A$:

$$A = \left(2\lambda I + P + Q^T + \rho(H + H^T)\right)/(2\lambda + 2\rho). \quad (2.86)$$

When $D_\phi(A, I)$ is the log-determinant divergence, the problem is specialized to

$$\min_A \lambda(\text{tr}(A) - \log \det(A)) - \langle P, A \rangle - \langle Q^T, A \rangle + \frac{\rho}{2} ||H - A||_F^2 + \frac{\rho}{2} ||H^T - A||_F^2 \quad s.t. \quad A \succ 0. \quad (2.87)$$

Taking the derivative of the objective function w.r.t $A$ and setting it to zero, we get

$$A^2 + \frac{1}{\rho}(\lambda I - P - Q^T - \rho(H + H^T))A - \frac{\lambda}{\rho}I = 0. \quad (2.88)$$

Let $B = \frac{1}{\rho}(\lambda I - P - Q^T - \rho(H + H^T))$ and $C = -\frac{\lambda}{\rho}I$, Eq.(2.88) can be written as

$$A^2 + BA + C = 0. \quad (2.89)$$

According to [156], since $B$ and $C$ commute, the solution of this equation is

$$A = -\frac{1}{2}B + \frac{1}{2}\sqrt{B^2 - 4C}. \quad (2.90)$$

Taking an eigen-decomposition of $B = \Phi \Sigma \Phi^{-1}$, we can compute $A$ as $A = \Phi \hat{\Sigma} \Phi^{-1}$, where $\hat{\Sigma}$ is a diagonal matrix with

$$\hat{\Sigma}_{kk} = -\frac{1}{2}\Sigma_{kk} + \frac{1}{2}\sqrt{\Sigma_{kk}^2 + \frac{4\lambda}{\rho}}. \quad (2.91)$$

Since $\sqrt{\Sigma_{kk}^2 + \frac{4\lambda}{\rho}} > \Sigma_{kk}$, we know $\hat{\Sigma}_{kk} > 0$. Hence $A$ is positive definite.
When $D_{\psi}(A, I)$ is the von Neumann divergence, the problem becomes
\begin{equation}
\min_A \lambda \text{tr}(A \log A - A) - \langle P, A \rangle - \langle Q^T, A \rangle + \frac{\rho}{2} \|H - A\|_F^2 + \frac{\rho}{2} \|H^T - A\|_F^2.
\end{equation}
\begin{equation}
\text{s.t.} \quad A \succ 0.
\end{equation}
(2.91)

Setting the gradient of the objective function w.r.t $A$ to zero, we get
\begin{equation}
\lambda \log A + 2\rho A = P + Q^T + \rho(H + H^T).
\end{equation}
(2.92)

Let $D = P + Q^T + \rho(H + H^T)$. We perform an eigen-decomposition of $D = \Phi \Sigma \Phi^{-1}$ and parameterize $A$ as $A = \Phi \Sigma \Phi^{-1}$, then we obtain
\begin{equation}
\lambda \log A + 2\rho A = \Phi(\lambda \log \hat{\Sigma} + 2\rho \hat{\Sigma}) \Phi^{-1}.
\end{equation}
(2.93)

Plugging this equation into Eq.(2.92), we get the following equation
\begin{equation}
\lambda \log \hat{\Sigma} + 2\rho \hat{\Sigma} = \Sigma,
\end{equation}
(2.94)

which amounts to solving $K$ independent one-variable equations taking the form
\begin{equation}
\lambda \log \hat{\Sigma}_{ii} + 2\rho \hat{\Sigma}_{ii} = \Sigma_{ii},
\end{equation}
(2.95)

where $i = 1, \cdots, K$. This equation has a closed-form solution
\begin{equation}
\hat{\Sigma}_{ii} = \frac{\lambda \omega(\frac{\Sigma_{ii}}{\lambda} - \log(\frac{\lambda}{2\rho}))}{2\rho},
\end{equation}
(2.96)

where $\omega(\cdot)$ is the Wright omega function \cite{137}. Due to the presence of log, $\hat{\Sigma}_{ii}$ is required to be positive and the solution always exists since the range of $\lambda \log \hat{\Sigma}_{ii} + 2\rho \hat{\Sigma}_{ii}$ and $\Sigma_{ii}$ are both $(-\infty, \infty)$. Hence $A$ is guaranteed to be positive definite.

**An ADMM-based Algorithm for BMD-KSC**

The BMD-KSC model has two set of parameters: sparse codes $\{a_n\}_{n=1}^N$ and a dictionary of RKHS functions $\{f_i\}_{i=1}^K$. We use a coordinate descent algorithm to learn these two parameter sets, which iteratively performs the following two steps: (1) fixing $\{f_i\}_{i=1}^K$, solving $\{a_n\}_{n=1}^N$; (2) fixing $\{a_n\}_{n=1}^N$, solving $\{f_i\}_{i=1}^K$, until convergence. We first discuss step (1). The sub-problem defined over $a_n$ is
\begin{equation}
\min_{a_n} \frac{1}{2} \|k(x_n, \cdot) - \sum_{i=1}^K a_{ni} f_i\|_H^2 + \lambda_1 \|a_n\|_1.
\end{equation}
(2.97)

$\|k(x_n, \cdot) - \sum_{i=1}^K a_{ni} f_i\|_H^2 = k(x_n, x_n) - 2a_n^T h + a_n^T G a_n$ where $h \in \mathbb{R}^K$, $h_i = \langle f_i, k(x_n, \cdot) \rangle$, $G \in \mathbb{R}^{K \times K}$ and $G_{ij} = \langle f_i, f_j \rangle$. This is a standard lasso \cite{331} problem and can be solved using many algorithms. Next we discuss step (2), which learns $\{f_i\}_{i=1}^K$ using the ADMM-based algorithm outlined in the main paper. The updates of all variables are the same as those in BMD-KDML, except $f_i$. Let $b_{ni} = k(x_n, \cdot) - \sum_{j \neq i}^K a_{nj} f_j$, the sub-problem defined over $f_i$ is:
\begin{equation}
\min_{f_i} \frac{1}{2} \sum_{n=1}^N \|b_{ni} - a_{ni} f_i\|_H^2 + \lambda_2 \frac{1}{2} \|f_i\|_H^2 + \|g_i, f_i\| + \sum_{j=1}^K P_{ij} (f_i, \hat{f}_j) + \sum_{j=1}^K Q_{ij} (f_i, \hat{f}_j) \left(\sum_{j=1}^K (f_i, \hat{f}_j) - A_{ij}\right)^2.
\end{equation}
(2.98)
This problem can be solved with functional gradient descent. The functional gradient of the objective function w.r.t $f_i$ is

$$
\sum_{n=1}^{N} a_{ni}(a_{ni}f_i - b_{ni}) + \lambda_2 f_i + g_i + \sum_{j=1}^{K} (P_{ij} + Q_{ij} + 2\rho(f_i, \hat{f}_j) - \rho(A_{ij} + A_{ji}))\hat{f}_j. \quad (2.99)
$$

### 2.5 Diversity and Sparsity: Nonoverlapness-promoting Regularization

In this section, we combine diversity with sparsity, which jointly bring in an effect of nonoverlapness [380], for the sake of selecting less-overlapped variables. Variable selection [331] is a classic problem in machine learning, widely used to find important explanatory factors, and improve generalization performance and interpretability of ML models.

Among the many criteria of evaluating model quality, two are typically considered: (1) accuracy of prediction on unseen data; (2) interpretation of the model. For (2), scientists prefer a simpler model because it puts more light on the relationship between the response and covariates. Parsimony [331] is especially an important issue when the number of predictors is large. With a large number of predictors, we often would like to determine a smaller subset that exhibits the strongest effects.

To produce accurate prediction while selecting a subset of important factors, regularization-based variable-selection methods [331, 404, 428] have been widely studied. The most notable one is $\ell_1$ regularization [331], which encourages the model coefficients to be sparse. Its variants including $\ell_1/\ell_2$-norm [404] that brings in a group sparsity effect and elastic net [428] which encourages strongly correlated predictors to be in or out of the model together, among many others.

In many ML problems, multiple responses are to be predicted based on the same set of covariates. For example, in multi-task classification, the classifiers of $m$ classes are built on top of a shared feature set and each classifier has a class-specific coefficient vector. In topic modeling [49], multiple topics are learned over the same vocabulary and each topic has a unique multinomial distribution on the words. Different responses are relevant to different subsets of covariates. For example, an education topic is relevant to words like student, university, professor while a political topic is relevant to words like government, president, election, etc. To account for the difference between different responses when performing variable selection, we desire the selected variables for different responses to be less-overlapped.

The problem is formally formulated as follows. Consider $m$ responses sharing $d$ covariates. Each response has a specific $d$-dimensional weight vector $w$ where each dimension corresponds to a covariate. Let $s(w) = \{k | w_k \neq 0\}$ – the support of $w$ – index the selected variables for a response. For any two responses $i$ and $j$, we desire their selected variables $s(w_i)$ and $s(w_j)$ are less overlapped, where the overlapness is measured by \[ \frac{|s(w_i) \cap s(w_j)|}{|s(w_i) \cup s(w_j)|}. \] To achieve this effect, we propose a regularizer that simultaneously encourages different weight vectors to be close to being orthogonal and each vector to be sparse, which jointly encourage vectors’ supports to have small overlap. Empirically, we verify that minimizing this regularizer reduces overlap among
selected variables. We apply this regularizer to four models: multiclass logistic regression, distance metric learning, sparse coding, and deep neural networks. Efficient algorithms are derived to solve these regularized problems. In particular, we develop an algorithm based on ADMM and coordinate descent for regularized sparse coding. In experiments, we demonstrate the empirical effectiveness of this regularizer in selecting non-overlap variables and improving generalization.

**Related works** Variable selection based on regularization has been widely studied. Lasso \([331]\) uses \(\ell_1\)-norm to encourage the coefficient vector of the linear regression model to be sparse. The lasso is able to recover the exact support of a sparse model from data generated by this model if the covariates are not too correlated \([417]\). Elastic net \([428]\) uses the weighted sum of the \(\ell_1\) and \(\ell_2\) norm to encourage strongly-correlated variables to be co-selected. Group lasso \([404]\) uses the \(\ell_1/\ell_2\) penalty, which is defined as the sum of the \(\ell_2\) norms of sub-weight-vectors corresponding to predefined groups, to select groups of variables. It recovers the support of a model if the support is a union of groups and if covariates of different groups are not too correlated. Zhao et al. \([418]\) proposed composite absolute penalties for hierarchical selection of covariates, e.g., when one has a hierarchy over the covariates and wants to select covariates only if their ancestors in the hierarchy are also selected. Graphical lasso \([113]\) uses matrix \(\ell_1\) norm for covariance (or neighborhood) selection.

In the context of group variable selection, several works \([24,174,418]\) consider the cases where variables from different groups are (non)overlapping. Their problem settings are different from ours. In their problems, the (non)overlapping structure is with respect to groups and is known as a prior while in our problem it is with respect to different responses and is unknown.

### 2.5.1 Nonoverlap-promoting Regularization

We assume the model has \(m\) responses and each is parameterized by a weight vector. For a vector \(w\), its support \(s(w)\) is defined as \(\{i|w_i \neq 0\}\) – the indices of nonzero entries in \(w\). And the support contains indexes of the selected variables. We first define a score \(\tilde{d}(w_i, w_j)\) to
measure the overlap between selected variables of two responses:

\[
\tilde{o}(w_i, w_j) = \frac{|s(w_i) \cap s(w_j)|}{|s(w_i) \cup s(w_j)|},
\]

which is the Jaccard index of the supports. The smaller \(\tilde{o}(w_i, w_j)\) is, the less overlapped the two sets of selected variables are. For \(m\) variable sets, the overlap score is defined as the sum of pairwise scores

\[
o(\{w_i\}_{i=1}^m) = \frac{1}{m(m-1)} \sum_{i \neq j} \tilde{o}(w_i, w_j).
\]

This score function is not smooth, which will result in great difficulty for optimization if used as a regularizer. Instead, we propose a smooth function that is motivated from \(\tilde{o}(w_i, w_j)\) and can achieve a similar effect as \(o(W)\). The basic idea is: to encourage small overlap, we can encourage (1) each vector has a small number of non-zero entries and (2) the intersection of supports among vectors is small. To realize (1), we use an L1 regularizer to encourage the vectors to be sparse. To realize (2), we encourage the vectors to be close to being orthogonal. For two sparse vectors, if they are close to being orthogonal, then their supports are landed on different positions. As a result, the intersection of supports is small. To promote orthogonality, similar to Section 2.2.1, we encourage the Gram matrix \(G (G_{ij} = w_i^\top w_j)\) of weight vectors to be close to an identity matrix \(I\) and measure the “closeness” between two matrices using the log-determinant divergence (LDD), which results in the LDD regularizer \(\text{tr}(G) - \log \det(G)\). Combining the orthogonality-promoting LDD regularizer with the sparsity-promoting L1 regularizer together, we obtain the following LDD-L1 regularizer

\[
\Omega(W) = \text{tr}(G) - \log \det(G) + \gamma \sum_{i=1}^m |w_i|_1,
\]

where \(\gamma\) is a tradeoff parameter between these two regularizers. As verified in experiments, this regularizer can effectively promote nonoverlap. The formal analysis of the relationship between Eq. (2.102) and Eq. (2.101) will be left for future study. It is worth noting that either L1 or LDD alone is not sufficient to reduce overlap. As illustrated in Figure 2.7a where only L1 is applied, though the two vectors are sparse, their supports are completely overlapped. In Figure 2.7b where the LDD regularizer is applied, though the two vectors are very close to being orthogonal, their supports are completely overlapped since they are dense. In Figure 2.7c where the LDD-L1 regularizer is used, the two vectors are sparse and are close to being orthogonal. As a result, their supports are not overlapped.

We apply LDD-L1 to four ML models: multiclass logistic regression, distance metric learning, sparse coding, and neural networks.

- **Multiclass logistic regression (MLR)** aims at classifying a data example \(x \in \mathbb{R}^d\) (whose features are treated as covariates) into one of the \(m\) classes (treated as responses). It is parameterized by a coefficient matrix \(W \in \mathbb{R}^{d \times m}\) where the \(i\)-th column is the coefficient vector of class \(i\) and \(d\) is the feature dimension of \(x\). In inference, MLR calculates \(p = \text{softmax}(W^\top x + b) \in \mathbb{R}^m\) where \(p_i\) denotes the probability that \(x\) belongs to class \(i\) and \(b \in \mathbb{R}^m\) is a bias vector. \(x\) is assigned to the class yielding the largest probability. Given
$N$ training examples $\{x_n, y_n\}_{n=1}^N$, MLR learns $W$ by minimizing the cross-entropy loss between $p_n$ and the ground-truth class label $y_n$. The LDD-L1 regularizer can be applied to encourage the coefficient vectors of different classes to have less-overlapped supports.

- **Distance metric learning (DML)** has wide applications in classification, clustering, and information retrieval [92, 146, 383]. Given data pairs labeled as similar or dissimilar, DML aims at learning a distance metric such that similar pairs would be placed close to each other and dissimilar pairs are separated apart. Following [357], we define the distance metric between $x, y \in \mathbb{R}^d$ as $\|W^T x - W^T y\|_2^2$ where $W \in \mathbb{R}^{d \times m}$ contains $m$ projection vectors which are treated as responses. The features in a data example are treated as covariates. Given $N$ training examples, $\{x_n, y_n, t_n\}_{n=1}^N$, where $x_n$ and $y_n$ are similar if the label $t_n$ equals 1 and dissimilar if $t_n = 0$, following [146], we learn the distance metric by minimizing $\sum_{n=1}^N \log(1 + \exp((2t_n - 1)\|W^T x - W^T y\|_2^2))$. Using LDD-L1 to promote nonoverlap among the projection vectors in $W$, we obtain the LDD-L1 regularized DML problem:

$$\min_F \sum_{n=1}^N \log(1 + \exp((2t_n - 1)\|W^T (x - y)\|_2^2) + \lambda \Omega(W). \quad (2.103)$$

- **Sparse Coding (SC) and Deep Neural Networks (DNNs)** have been introduced in Section 2.3.1. In SC, the features of data examples are treated as covariates and the basis vectors are treated as responses. We apply the LDD-L1 regularizer to encourage the supports of the basis vectors to have small overlap. In DNNs, hidden units in a layer $l$ are treated as a group of responses. Their weight vectors are encouraged to have less-overlapped supports by the LDD-L1 regularizer. In the experiments, we study two popular instances of DNNs: long short-term memory (LSTM) network [163] and convolutional neural network (CNN).

### 2.5.2 A Coordinate Descent Algorithm

For LDD-L1-regularized MLR, NN, and DML problems, we solve them using proximal gradient descent [270]. The proximal operation is with respect to the L1 regularizer in LDD-L1. The algorithm iteratively performs the following three steps until convergence: (1) calculate gradient of $\mathcal{L}(W) + \lambda(\text{tr}(W^T W) - \log\det(W^T W))$ where $\mathcal{L}(W)$ is the loss function of the unregularized ML model and $\text{tr}(W^T W) - \log\det(W^T W)$ is the LDD regularizer in LDD-L1; (2) perform gradient descent update of $W$; (3) apply the proximal operator of the L1 regularizer to $W$.

For LDD-L1-SC, we solve it by alternating between $A$ and $W$: (1) updating $A$ with $W$ fixed; (2) updating $W$ with $A$ fixed. These two steps alternate until convergence. With $W$ fixed, the sub-problem defined over $A$ is $\min_A \frac{1}{2}\|X - WA\|_F^2 + \lambda_1 |A|_1$, which can be decomposed into $n$ lasso problems ($P1$): for $i = 1, \ldots, n$, $\min_{a_i} \frac{1}{2}\|x_i - Wa_i\|_2^2 + \lambda_1 |a_i|_1$ where $a_i$ is the coefficient vector of the $i$-th sample. Lasso can be solved by many algorithms, such as proximal gradient descent (PGD). Fixing $A$, the sub-problem defined over $W$ is ($P2$): $\min_W \frac{1}{2}\|X - WA\|_F^2 + \lambda_2 \|W\|_F^2 + \lambda_4 |W|_1 + \frac{\lambda_5}{2}(\text{tr}(W^T W) - \log \det(W^T W))$. We solve this problem using
an ADMM-based algorithm. First, we write the problem into an equivalent form:

$$\min_{\mathbf{W}} \frac{1}{2} \| \mathbf{X} - \mathbf{W} \mathbf{A} \|^2_F + \frac{\lambda_2}{2} \| \mathbf{W} \|^2_F + \lambda_4 |\mathbf{W}|_1 + \frac{\lambda_3}{2} (\text{tr}(\mathbf{W}^T \mathbf{W}) - \log \det(\mathbf{W}^T \mathbf{W}))$$

$$\text{s.t.} \quad \mathbf{W} = \mathbf{\tilde{W}}.$$  \hspace{1cm} (2.104)

Then we write down the augmented Lagrangian function (P3):

$$\min_{\mathbf{W}} \frac{1}{2} \| \mathbf{X} - \mathbf{W} \mathbf{A} \|^2_F + \frac{\lambda_2}{2} \| \mathbf{W} \|^2_F + \lambda_4 |\mathbf{W}|_1 + \langle \mathbf{U}, \mathbf{W} - \mathbf{\tilde{W}} \rangle + \frac{\rho}{2} \| \mathbf{W} - \mathbf{\tilde{W}} \|^2_F + \frac{\lambda_3}{2} (\text{tr}(\mathbf{W}^T \mathbf{W}) - \log \det(\mathbf{W}^T \mathbf{W})).$$

We minimize this Lagrangian function by alternating among $\mathbf{W}$, $\mathbf{\tilde{W}}$, and $\mathbf{U}$.

**Update $\mathbf{W}$** The subproblem defined on $\mathbf{W}$ is (P4):

$$\min_{\mathbf{W}} \frac{1}{2} \| \mathbf{X} - \mathbf{W} \mathbf{A} \|^2_F + \frac{\lambda_2}{2} \| \mathbf{W} \|^2_F + \langle \mathbf{U}, \mathbf{W} \rangle + \frac{\lambda_3}{2} (\text{tr}(\mathbf{W}^T \mathbf{W}) - \log \det(\mathbf{W}^T \mathbf{W})) + \frac{\rho}{2} \| \mathbf{W} - \mathbf{\tilde{W}} \|^2_F,$$

which can be solved using a coordinate descent (CD) algorithm. In each iteration of CD, one basis vector is chosen for update while the others are fixed. Without loss of generality, we assume it is $w_1$. The loss function defined over $w_1$ is $\frac{1}{2} \sum_{i=1}^n \| \mathbf{x}_i - \sum_{t=2}^m a_{it} w_t - a_{i1} w_1 \|^2_F + \lambda_2 w_1^2 + \lambda_3 \log \det(\mathbf{W}^T \mathbf{W}) + \mathbf{u}^T \mathbf{w}_1 + \frac{\rho}{2} \| \mathbf{w}_1 - \mathbf{\tilde{w}}_1 \|^2_F$. The optimal solution can be obtained via the following procedures: (1) calculate $\mathbf{M} = \mathbf{I} - \mathbf{W}_{-1}(\mathbf{W}_{-1}^T \mathbf{W}_{-1})^{-1} \mathbf{W}_{-1}^T$, where $\mathbf{W}_{-1} = \big[\mathbf{w}_2, \ldots, \mathbf{w}_m\big]$; (2) perform eigen-decomposition: $\mathbf{M} = \mathbf{U} \Sigma \mathbf{U}^T$; (3) solve the scalar quadratic equation $\gamma \sum_{s=1}^d (\mathbf{U}^T \mathbf{b})_s^2 = (\gamma c - \lambda_3)^2$ w.r.t $\gamma$, where $c = \sum_{i=1}^n a_{i1}^2 + \lambda_2 + \lambda_3 + \rho$ and $\mathbf{b} = \sum_{i=1}^n a_{i1} (\mathbf{x}_i - \sum_{t=2}^m a_{it} \mathbf{w}_t) - \mathbf{u} + \rho \mathbf{\tilde{w}}_j$; (4) calculate $\mathbf{w}_1$ as:

$$\mathbf{w}_1 = \gamma \mathbf{U}(\gamma c \mathbf{I} - \lambda_3 \mathbf{\Sigma})^{-1} \mathbf{U}^T \mathbf{b}.$$  \hspace{1cm} (2.106)

The detailed derivation is deferred to Section 2.5.4

**Update $\mathbf{\tilde{W}}$** The subproblem defined on $\mathbf{\tilde{W}}$ is (P5): $\min_{\mathbf{\tilde{W}}} \lambda_4 |\mathbf{\tilde{W}}|_1 - \langle \mathbf{U}, \mathbf{\tilde{W}} \rangle + \frac{\rho}{2} \| \mathbf{W} - \mathbf{\tilde{W}} \|^2_F$, which is an lasso problem and can be solved using proximal gradient descent [270].

**Update $\mathbf{U}$** The update equation of $\mathbf{U}$ is simple: $\mathbf{U} = \mathbf{U} + (\mathbf{W} - \mathbf{\tilde{W}})$.

### 2.5.3 Evaluation

In this section, we present experimental results.

**Simulation Study**

The simulation study was performed on the multiclass logistic regression model. We set the number of classes to 10. Each class is relevant to 5 variables. The variables of different classes have no overlap. We generated 1000 data samples from a multivariate Gaussian distribution with zero mean and the covariance matrix was set to an identity matrix. In the coefficient vector of each class, the entries corresponding to the relevant variables were uniformly sampled from $[-1, 1]$. 

66
Algorithm 1: Algorithm for solving the LDD-L1-SC problem.

Initialize \( W \) and \( A \)
repeat
    Update \( A \) with \( W \) being fixed, by solving \( n \) lasso problems (P1).
    repeat
        for \( i \leftarrow 1 \) to \( m \) do
            Update the \( i \)-th column vector \( w_i \) of \( W \) using Eq.(2.106)
        end for
    until convergence of the problem (P4)
    Update \( \tilde{W} \) by solving the lasso problem (P5)
    \( U \leftarrow U + (W - \tilde{W}) \)
until convergence of the problem (P3)
until convergence of the LDD-L1-SC problem

and the rest entries were set to zero. Given a generated sample \( x \) and the generated coefficient matrix \( W \in \mathbb{R}^{10 \times 50} \), the class label of sample \( x \) was determined as \( y = \arg \max_k [Wx + b]_k \), where \( b \in \mathbb{R}^{10} \) was a randomly generated bias vector whose entries were sampled independently from a univariate normal distribution. We split the dataset into train/validation/test set with 600/200/200 examples respectively. The regularization parameter was tuned on the validation set to achieve the best prediction performance. We generated 50 simulated datasets. The performance was averaged over these 50 datasets. We compared our method with L1-regularization \[331\] and elastic net \[428\]. We did not compare with LDD since it is not able to select variables.

Following \[193\], we use sensitivity (true positive rate) and specificity (true negative rate) to measure the performance of recovering the true supports of the coefficient vectors, shown in the second and third column of Table 2.25. Our method outperforms the baselines with a large margin. LDD-L1 encourages the supports of different weight vectors to have less overlap, which makes it more suitable to select nonoverlapping variables. We also compare the performance of different methods in terms of prediction errors, shown in the fourth column of Table 2.25. LDD-L1 achieves the lowest error rate. Since the variables selected by our method are closer to the ground-truth, the predictions made by our method based upon these selected variables are more accurate.

Experiments on Real Data

We applied the LDD-L1 regularizer to three ML models and four datasets and verified whether it is able to improve generalization performance. In each experiment, the hyperparameters were tuned on the validation set.

Sparse coding for text representation learning  The SC experiments were conducted on two text datasets: 20-Newsgroups (20-News) \[1\] and Reuters Corpus Volume 1 (RCV1) \[10\]. The 20-News dataset contains newsgroup documents belonging to 20 categories, where 11314, 3766,
and 3766 documents were used for training, validation, and testing respectively. The original RCV1 dataset contains documents belonging to 103 categories. Following [60], we chose the largest 4 categories which contain 9625 documents, to carry out the study. The number of training, validation, and testing documents are 5775, 1925, 1925 respectively. For both datasets, stopwords were removed and all words were changed into lower-case. Top 1000 words with the highest document frequency were selected to form the vocabulary. We used tf-idf to represent documents and the feature vector of each document was normalized to have unit L2 norm. For 20-News, the number of basis vectors in LDD-L1-SC was set to 50. \( \lambda_1, \lambda_2, \lambda_3, \text{ and } \lambda_4 \) were set to 1, 1, 0.1, and 0.001 respectively. For RCV1, the number of basis vectors was set to 200. \( \lambda_1, \lambda_2, \lambda_3, \text{ and } \lambda_4 \) were set to 0.01, 1, 1 and 1 respectively. We compared LDD-L1 with LDD-only and L1-only.

To evaluate the model performance quantitatively, we applied the dictionary learned on the training data to infer the linear coefficients (\( \mathbf{A} \) in SC) of test documents, then performed \( k \)-nearest neighbors (KNN) classification on \( \mathbf{A} \). Table 2.26 shows the classification accuracy on test sets of 20-News and RCV1 and the gap between the accuracy on training and test sets. Without regularization, SC achieves a test accuracy of 0.592 on 20-News, which is lower than the training accuracy by 0.119. This suggests that an overfitting to training data occurs. With LDD-L1 regularization, the test accuracy is improved to 0.612 and the gap between the accuracy on training and test sets is reduced to 0.099, demonstrating the ability of LDD-L1 in alleviating overfitting. Though LDD alone and L1 alone improve test accuracy and reduce train/test gap, they perform less well than LDD-L1, which indicates that for overfitting reduction, encouraging nonoverlap is more effective than solely promoting orthogonality or solely promoting sparsity. Similar observations are made on the RCV1 dataset. Interestingly, the test accuracy achieved by LDD-L1-SC on RCV1 is better than the training accuracy.

Table 2.27 shows the selected variables (words that have nonzero weights) for 9 exemplar basis vectors learned by LDD-L1-SC on the 20-News dataset. From the selected words, we can see basis vector 1-9 represent the following semantics respectively: crime, faith, job, war, university, research, service, religion, and Jews. The selected words of different basis vectors have no overlap. As a result, it is easy to associate each vector with a unique concept, in other words, easy to interpret. Figure 2.8 visualizes the learned vectors where the black dots denote vectors’ supports. As can be seen, the supports of different basis vectors are landed on different words and their overlap is small.

**LSTM for language modeling** We applied LSTM networks [163] to learn language models on the Penn Treebank (PTB) dataset [244], which consists of 923K training, 73K validation, and 82K test words. Following [254], top 10K words with the highest frequency were selected.
to form the vocabulary. All other words were replaced with a special token UNK. The LSTM network architecture follows the word language model (PytorchTM) provided in Pytorch [3]. The number of hidden layers was set to 2. The embedding size was 1500. The size of hidden state was 1500. Following [279], the word embedding and softmax weights were tied. The number of training epochs was 40. Dropout with 0.65 was used. The initial learning rate was 20. Gradient clipping threshold was 0.25. The size of mini-batch was 20. In LSTM training, the network was unrolled for 35 iterations. Perplexity was used for evaluating language modeling performance (lower is better). The weight parameters were initialized uniformly between [-0.1, 0.1]. The bias parameters were initialized as 0. We compared with the following regularizers: (1) L1 regularizer; (2) orthogonality-promoting regularizers based on cosine similarity (CS) [402], incoherence (IC) [29], mutual angle (MA) [369], decorrelation (DC) [82], angular constraint (AC) (Section 2.3.1), and LDD (Section 2.2.1).

Table 2.28 shows the perplexity on the PTB test set. Without regularization, PytorchLM achieves a perplexity of 72.3. With LDD-L1 regularization, the perplexity is significantly reduced to 71.1. This shows that LDD-L1 can effectively improve generalization performance. Compared with the sparsity-promoting L1 regularizer and orthogonality-promoting regularizers, LDD-L1 – which promotes nonoverlap by simultaneously promoting sparsity and orthogonality – achieves lower perplexity. For the convenience of readers, we also list the perplexity achieved by other state of the art deep learning models. The LDD-L1 regularizer can be applied to these models as well to potentially boost their performance.

**CNN for image classification** The CNN experiments were performed on the CIFAR-10 dataset [5]. It consists of 32x32 color images belonging to 10 categories, where 50,000 images were used for training and 10,000 for testing. 5000 training images were used as the validation set for hyperparameter tuning. We augmented the dataset by first zero-padding the images with 4 pixels on each side, then randomly cropping the padded images to reproduce 32x32 images. The CNN architecture follows that of the wide residual network (WideResNet) [406]. The depth and width were set to 28 and 10 respectively. The networks were trained using SGD, where the epoch number was 200, the learning rate was set to 0.1 initially and was dropped by 0.2 at 60, 120, and 160 epochs, the minibatch size was 128 and the Nesterov momentum was 0.9. The dropout probability was 0.3 and the L2 weight decay was 0.0005. Model performance was measured using error rate, which was the median of 5 runs. We compared with (1) the L1 regularizer; (2) orthogonality-promoting regularizers including CS, IC, MA, DC, AC, LDD and the one based on locally constrained decorrelation (LCD) [291].

Table 2.29 shows classification errors on the CIFAR-10 test set. Compared with the unregularized WideResNet which achieves an error rate of 3.89%, the proposed LDD-L1 regularizer greatly reduces the error to 3.60%. LDD-L1 outperforms the L1 regularizer and orthogonality-promoting regularizers, demonstrating that encouraging nonoverlap is more effective than encouraging sparsity alone or orthogonality alone in improving generalization performance. The error rates achieved by other state of the art methods are also listed.
LDD-L1 and Nonoverlap

We verified whether the LDD-L1 regularizer is able to promote nonoverlap. The study was performed on the SC model and the 20-News dataset. The number of basis vectors was set to 50. For 5 choices of the regularization parameter of LDD-L1: $\{10^{-4}, 10^{-3}, \ldots, 1\}$, we ran the LDD-L1-SC model until convergence and measured the overlap score (defined in Eq. (2.101)) of the basis vectors. The tradeoff parameter $\gamma$ inside LDD-L1 was set to 1. Figure 2.9 shows that the overlap score consistently decreases as the regularization parameter of LDD-L1 increases, which implies that LDD-L1 can effectively encourage nonoverlap. As a comparison, we replaced LDD-L1 with LDD-only and L1-only, and measured the overlap scores. As can be seen, for LDD-only, the overlap score remains to be 1 when the regularization parameter increases, which indicates that LDD alone is not able to reduce overlap. This is because under LDD-only, the vectors remain dense, which renders their supports to be completely overlapped. Under the same regularization parameter, LDD-L1 achieves lower overlap score than L1, which suggests that LDD-L1 is more effective in promoting nonoverlap. Given that $\gamma$ – the tradeoff parameter associated with the L1 norm in LDD-L1 – was set to 1, the same regularization parameter $\lambda$ imposes the same level of sparsity for both LDD-L1 and L1-only. Since LDD-L1 encourages the vectors to be mutually orthogonal, the intersection between vectors’ supports is small, which consequently results in small overlap. This is not the case for L1-only, which hence is less effective in reducing overlap.

2.5.4 Appendix: Details of Algorithms

In each iteration of the CD algorithm, one basis vector is chosen for update while the others are fixed. Without loss of generality, we assume it is $w_1$. The sub-problem defined over $w_1$ is

$$
\min_{w_1} \frac{1}{2} \sum_{i=1}^{n} \| x_i - \sum_{l=2}^{m} a_{il} w_l - a_{i1} w_1 \|_2^2 + \frac{\lambda_1 + \lambda_2}{2} \| w_1 \|_2^2 - \frac{\lambda_2}{2} \log \det(W^T W) + u^T w_1 + \frac{\rho}{2} \| w_1 - \tilde{w}_1 \|_2^2.
$$

(2.107)
To obtain the optimal solution, we take the derivative of the objective function and set it to zero. First, we discuss how to compute the derivative of $\logdet(W^TW)$ w.r.t $w_1$. According to the chain rule, we have
\[
\frac{\partial \logdet(W^TW)}{\partial w_1} = 2W(W^TW)_{:,1}^{-1},
\]
where $(W^TW)_{:,1}^{-1}$ denotes the first column of $(W^TW)^{-1}$. Let $W_{-1} = [w_2, \cdots, w_m]$, then
\[
W^TW = \begin{bmatrix} w_1^Tw_1 & w_1^Tw_{-1} \\ W_{-1}^Tw_1 & W_{-1}^TW_{-1} \end{bmatrix}
\]
(2.109)
According to the inverse of block matrix
\[
\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} \tilde{A} & \tilde{B} \\ \tilde{C} & \tilde{D} \end{bmatrix},
\]
(2.110)
where $\tilde{A} = (A - BD^{-1}C)^{-1}$, $\tilde{B} = -(A - BD^{-1}C)^{-1}BD^{-1}$, $\tilde{C} = -D^{-1}C(A - BD^{-1}C)^{-1}$, $\tilde{D} = D^{-1} + D^{-1}C(A - BD^{-1}C)^{-1}BD^{-1}$, we have $(W^TW)_{:,1}^{-1}$ equals $[a \ b]^T$ where
\[
a = (w_1^Tw_1 - w_1^Tw_{-1}(W_{-1}^TW_{-1})^{-1}W_{-1}^Tw_1)^{-1},
\]
(2.111)
\[
b = -(W_{-1}^TW_{-1})^{-1}W_{-1}^tw_1.
\]
(2.112)
Then
\[
W(W^TW)_{:,1}^{-1} = [w_1 \ W_{-1}] \begin{bmatrix} a \\ b \end{bmatrix} = \frac{Mw_1}{w_1^TMw_1},
\]
(2.113)
where
\[
M = I - W_{-1}(W_{-1}^TW_{-1})^{-1}W_{-1}^T.
\]
(2.114)
To this end, we obtain the full gradient of the objective function in Eq.(2.107):
\[
\sum_{i=1}^{n} a_{i1}(a_{i1}w_1 + \sum_{l=2}^{m} a_{il}w_l - x_i) + (\lambda_2 + \lambda_3)w_1 - \lambda_3 \frac{Mw_1}{w_1^TMw_1} + \rho(w_1 - \bar{w}_1) + u.
\]
(2.115)
Setting the gradient to zero, we get
\[
((\sum_{i=1}^{n} a_{i1}^2 + \lambda_2 + \lambda_3 + \rho)I - \lambda_3 M/(w_1^TMw_1))w_1 = \sum_{i=1}^{n} a_{i1}(x_i - \sum_{l=2}^{m} a_{il}w_l) - u + \rho \bar{w}_1.
\]
(2.116)
Let $\gamma = w_1^TMw_1$, $c = \sum_{i=1}^{n} a_{i1}^2 + \lambda_2 + \lambda_3 + \rho$, $b = \sum_{i=1}^{n} a_{i1}(x_i - \sum_{l=2}^{m} a_{il}w_l) - u + \rho \bar{w}_j$, then $(cI - \frac{\lambda_3}{\gamma}M)w_1 = b$ and $w_1 = (cI - \frac{\lambda_3}{\gamma}M)^{-1}b$. Let $U \Sigma U^T$ be the eigen decomposition of $M$, we have
\[
w_1 = \gamma U(\gamma cI - \lambda_3 \Sigma)^{-1}U^Tb.
\]
(2.117)
Then
\[
w_1^TMw_1
= \gamma^2 b^TU(\gamma cI - \lambda_3 \Sigma)^{-1}U^T \Sigma U^TU(\gamma cI - \lambda_3 \Sigma)^{-1}U^Tb
= \gamma^2 b^TU(\gamma cI - \lambda_3 \Sigma)^{-1} \Sigma(\gamma cI - \lambda_3 \Sigma)^{-1}U^Tb
= \gamma^2 \sum_{s=1}^{d} \frac{(U^Tb)^2_{ss} \Sigma_{ss}}{(\rho - \lambda_3 \Sigma_{ss})^2} = \gamma.
\]
(2.118)
The matrix $A = W^{-1} (W^\top_1 W^{-1})^{-1} W^\top_1$ is idempotent, i.e., $AA = A$, and its rank is $m - 1$. According to the property of idempotent matrix, the first $m - 1$ eigenvalues of $A$ equal one and the rest equal zero. Thereafter, the first $m - 1$ eigenvalues of $M = I - A$ equal zero and the rest equal one. Based on this property, Eq.(2.118) can be simplified as

$$\gamma \sum_{s=m}^{d} \frac{(U^\top b)^2_s}{(r_c - \lambda_3)^2} = 1. \quad (2.119)$$

After simplification, it is a quadratic function where $\gamma$ has a closed form solution. Then we plug the solution of $\gamma$ into Eq.(2.117) to get the solution of $w_1$. 

72
<table>
<thead>
<tr>
<th>Method</th>
<th>MIMIC AUC-All</th>
<th>MIMIC AUC-F</th>
<th>MIMIC AUC-IF</th>
<th>EICU AUC-All</th>
<th>EICU AUC-F</th>
<th>EICU AUC-IF</th>
<th>Reuters AUC-All</th>
<th>Reuters AUC-F</th>
<th>Reuters AUC-IF</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCDML</td>
<td>0.634</td>
<td>0.654</td>
<td>0.608</td>
<td>0.671</td>
<td>0.690</td>
<td>0.637</td>
<td>0.949</td>
<td>0.963</td>
<td>0.916</td>
</tr>
<tr>
<td>CDML</td>
<td>0.641</td>
<td>0.659</td>
<td>0.617</td>
<td>0.677</td>
<td>0.693</td>
<td>0.652</td>
<td>0.952</td>
<td>0.961</td>
<td>0.929</td>
</tr>
<tr>
<td>EUC</td>
<td>0.559</td>
<td>0.558</td>
<td>0.562</td>
<td>0.583</td>
<td>0.584</td>
<td>0.581</td>
<td>0.887</td>
<td>0.888</td>
<td>0.885</td>
</tr>
<tr>
<td>LMNN [357]</td>
<td>0.628</td>
<td>0.643</td>
<td>0.609</td>
<td>0.662</td>
<td>0.678</td>
<td>0.633</td>
<td>0.943</td>
<td>0.951</td>
<td>0.913</td>
</tr>
<tr>
<td>LDML [146]</td>
<td>0.619</td>
<td>0.638</td>
<td>0.594</td>
<td>0.667</td>
<td>0.678</td>
<td>0.647</td>
<td>0.934</td>
<td>0.946</td>
<td>0.906</td>
</tr>
<tr>
<td>MLEC [200]</td>
<td>0.621</td>
<td>0.633</td>
<td>0.605</td>
<td>0.679</td>
<td>0.692</td>
<td>0.656</td>
<td>0.927</td>
<td>0.933</td>
<td>0.916</td>
</tr>
<tr>
<td>GMML [405]</td>
<td>0.607</td>
<td>0.621</td>
<td>0.588</td>
<td>0.668</td>
<td>0.679</td>
<td>0.648</td>
<td>0.931</td>
<td>0.938</td>
<td>0.905</td>
</tr>
<tr>
<td>ILHD [63]</td>
<td>0.577</td>
<td>0.590</td>
<td>0.560</td>
<td>0.633</td>
<td>0.652</td>
<td>0.610</td>
<td>0.905</td>
<td>0.919</td>
<td>0.893</td>
</tr>
<tr>
<td>ℓ₂-CDML</td>
<td>0.648</td>
<td>0.664</td>
<td>0.627</td>
<td>0.695</td>
<td>0.706</td>
<td>0.676</td>
<td>0.953</td>
<td>0.967</td>
<td>0.930</td>
</tr>
<tr>
<td>ℓ₂,₁-CDML [399]</td>
<td>0.646</td>
<td>0.658</td>
<td>0.630</td>
<td>0.703</td>
<td>0.727</td>
<td>0.661</td>
<td>0.963</td>
<td>0.970</td>
<td>0.936</td>
</tr>
<tr>
<td>Tr-CDML [234]</td>
<td>0.659</td>
<td>0.672</td>
<td>0.642</td>
<td>0.696</td>
<td>0.709</td>
<td>0.673</td>
<td>0.961</td>
<td>0.966</td>
<td>0.934</td>
</tr>
<tr>
<td>IT-CDML [92]</td>
<td>0.653</td>
<td>0.675</td>
<td>0.626</td>
<td>0.692</td>
<td>0.705</td>
<td>0.668</td>
<td>0.954</td>
<td>0.964</td>
<td>0.920</td>
</tr>
<tr>
<td>Dropout-CDML</td>
<td>0.647</td>
<td>0.660</td>
<td>0.630</td>
<td>0.701</td>
<td>0.718</td>
<td>0.670</td>
<td>0.959</td>
<td>0.968</td>
<td>0.937</td>
</tr>
<tr>
<td>OS-CDML [116]</td>
<td>0.649</td>
<td>0.665</td>
<td>0.626</td>
<td>0.689</td>
<td>0.711</td>
<td>0.679</td>
<td>0.957</td>
<td>0.961</td>
<td>0.938</td>
</tr>
<tr>
<td>DCM-NCDML [242]</td>
<td>0.652</td>
<td>0.662</td>
<td>0.639</td>
<td>0.706</td>
<td>0.717</td>
<td>0.686</td>
<td>0.962</td>
<td>0.976</td>
<td>0.943</td>
</tr>
<tr>
<td>CS-NCDML [402]</td>
<td>0.661</td>
<td>0.676</td>
<td>0.641</td>
<td>0.712</td>
<td>0.736</td>
<td>0.670</td>
<td>0.967</td>
<td>0.973</td>
<td>0.954</td>
</tr>
<tr>
<td>DPP-NCDML [429]</td>
<td>0.659</td>
<td>0.679</td>
<td>0.632</td>
<td>0.714</td>
<td>0.725</td>
<td>0.695</td>
<td>0.958</td>
<td>0.971</td>
<td>0.937</td>
</tr>
<tr>
<td>IC-NCDML [29]</td>
<td>0.660</td>
<td>0.674</td>
<td>0.642</td>
<td>0.711</td>
<td>0.728</td>
<td>0.685</td>
<td>0.972</td>
<td>0.984</td>
<td>0.954</td>
</tr>
<tr>
<td>DC-NCDML [82]</td>
<td>0.648</td>
<td>0.666</td>
<td>0.625</td>
<td>0.698</td>
<td>0.711</td>
<td>0.675</td>
<td>0.965</td>
<td>0.977</td>
<td>0.960</td>
</tr>
<tr>
<td>VGF-NCDML [177]</td>
<td>0.657</td>
<td>0.673</td>
<td>0.634</td>
<td>0.718</td>
<td>0.730</td>
<td>0.697</td>
<td>0.974</td>
<td>0.985</td>
<td>0.952</td>
</tr>
<tr>
<td>MA-NCDML [567]</td>
<td>0.659</td>
<td>0.670</td>
<td>0.644</td>
<td>0.721</td>
<td>0.733</td>
<td>0.703</td>
<td>0.975</td>
<td>0.983</td>
<td>0.959</td>
</tr>
<tr>
<td>OC-NCDML [233]</td>
<td>0.651</td>
<td>0.663</td>
<td>0.636</td>
<td>0.705</td>
<td>0.716</td>
<td>0.685</td>
<td>0.955</td>
<td>0.966</td>
<td>0.931</td>
</tr>
<tr>
<td>OS-NCDML [116]</td>
<td>0.639</td>
<td>0.658</td>
<td>0.614</td>
<td>0.675</td>
<td>0.691</td>
<td>0.641</td>
<td>0.951</td>
<td>0.962</td>
<td>0.928</td>
</tr>
<tr>
<td>SFN-NCDML [74]</td>
<td>0.662</td>
<td>0.677</td>
<td>0.642</td>
<td>0.724</td>
<td>0.736</td>
<td>0.701</td>
<td>0.973</td>
<td>0.984</td>
<td>0.947</td>
</tr>
<tr>
<td>VND-NCDML</td>
<td>0.667</td>
<td>0.676</td>
<td>0.655</td>
<td>0.733</td>
<td>0.748</td>
<td>0.706</td>
<td>0.976</td>
<td>0.983</td>
<td>0.971</td>
</tr>
<tr>
<td>LDD-NCDML</td>
<td>0.664</td>
<td>0.674</td>
<td>0.651</td>
<td>0.731</td>
<td>0.743</td>
<td>0.711</td>
<td>0.973</td>
<td>0.981</td>
<td>0.964</td>
</tr>
<tr>
<td>CSFN-CDML</td>
<td>0.668</td>
<td>0.679</td>
<td>0.653</td>
<td>0.728</td>
<td>0.741</td>
<td>0.705</td>
<td>0.978</td>
<td>0.991</td>
<td>0.968</td>
</tr>
<tr>
<td>CVND-CDML</td>
<td>0.672</td>
<td>0.678</td>
<td>0.664</td>
<td>0.735</td>
<td>0.744</td>
<td>0.718</td>
<td>0.984</td>
<td>0.996</td>
<td>0.982</td>
</tr>
<tr>
<td>CLDD-CDML</td>
<td>0.669</td>
<td>0.678</td>
<td>0.658</td>
<td>0.739</td>
<td>0.750</td>
<td>0.719</td>
<td>0.981</td>
<td>0.993</td>
<td>0.980</td>
</tr>
</tbody>
</table>

Table 2.10: On three imbalanced datasets – MIMIC, EICU, Reuters, we show the mean AUC (averaged on 5 random train/test splits) on all classes (AUC-All), frequent classes (AUC-F), and infrequent classes (AUC-IF). On the second panel (EUC, etc.) are well established or state of the art baselines. On the third panel (ℓ₂-CDML, etc.) are CDML methods regularized by non-diversity regularizers. On the fourth panel (DCM-NCDML, etc.) are NCDML methods regularized by previously proposed diversity-promoting regularizers. On the fifth panel (VND-NCDML, etc.) are NCDML methods regularized by our proposed nonconvex BMD regularizers. On the sixth panel (CSFN-CDML, etc.) are CDML methods regularized by our proposed convex BMD regularizers.
Table 2.11: AUC-All on 4 balanced datasets.

<table>
<thead>
<tr>
<th></th>
<th>MIMIC</th>
<th>EiCU</th>
<th>Reuters</th>
<th>News</th>
<th>Cars</th>
<th>Birds</th>
<th>Act</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NC</td>
<td>CF</td>
<td>NC</td>
<td>CF</td>
<td>NC</td>
<td>CF</td>
<td></td>
</tr>
<tr>
<td>NCDML</td>
<td>0.50</td>
<td>1.7</td>
<td>0.7</td>
<td>2.4</td>
<td>0.3</td>
<td>9.6</td>
<td>5.9</td>
</tr>
<tr>
<td>LMNN</td>
<td>0.52</td>
<td>1.7</td>
<td>0.7</td>
<td>2.4</td>
<td>0.3</td>
<td>9.6</td>
<td>5.9</td>
</tr>
<tr>
<td>LDML</td>
<td>0.48</td>
<td>1.7</td>
<td>0.7</td>
<td>2.4</td>
<td>0.3</td>
<td>9.6</td>
<td>5.9</td>
</tr>
<tr>
<td>MLE</td>
<td>0.50</td>
<td>1.7</td>
<td>0.7</td>
<td>2.4</td>
<td>0.3</td>
<td>9.6</td>
<td>5.9</td>
</tr>
<tr>
<td>GMML</td>
<td>0.48</td>
<td>1.7</td>
<td>0.7</td>
<td>2.4</td>
<td>0.3</td>
<td>9.6</td>
<td>5.9</td>
</tr>
<tr>
<td>IHD</td>
<td>0.50</td>
<td>1.7</td>
<td>0.7</td>
<td>2.4</td>
<td>0.3</td>
<td>9.6</td>
<td>5.9</td>
</tr>
</tbody>
</table>

Table 2.12: The numbers of components that achieve the AUCs in Table 6.1 and 2.11.

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MIMIC</td>
<td>EiCU</td>
<td>Reuters</td>
<td>News</td>
<td>Cars</td>
<td>Birds</td>
<td>Act</td>
</tr>
<tr>
<td></td>
<td>NC</td>
<td>CF</td>
<td>NC</td>
<td>CF</td>
<td>NC</td>
<td>CF</td>
<td></td>
</tr>
<tr>
<td>NCDML</td>
<td>0.50</td>
<td>1.7</td>
<td>0.7</td>
<td>2.4</td>
<td>0.3</td>
<td>9.6</td>
<td>5.9</td>
</tr>
<tr>
<td>LMNN</td>
<td>0.52</td>
<td>1.7</td>
<td>0.7</td>
<td>2.4</td>
<td>0.3</td>
<td>9.6</td>
<td>5.9</td>
</tr>
<tr>
<td>LDML</td>
<td>0.48</td>
<td>1.7</td>
<td>0.7</td>
<td>2.4</td>
<td>0.3</td>
<td>9.6</td>
<td>5.9</td>
</tr>
<tr>
<td>MLE</td>
<td>0.50</td>
<td>1.7</td>
<td>0.7</td>
<td>2.4</td>
<td>0.3</td>
<td>9.6</td>
<td>5.9</td>
</tr>
<tr>
<td>GMML</td>
<td>0.48</td>
<td>1.7</td>
<td>0.7</td>
<td>2.4</td>
<td>0.3</td>
<td>9.6</td>
<td>5.9</td>
</tr>
<tr>
<td>IHD</td>
<td>0.50</td>
<td>1.7</td>
<td>0.7</td>
<td>2.4</td>
<td>0.3</td>
<td>9.6</td>
<td>5.9</td>
</tr>
<tr>
<td>Method</td>
<td>MIMIC</td>
<td>EICU</td>
<td>Reuters</td>
<td>News</td>
<td>Cars</td>
<td>Birds</td>
<td>Act</td>
</tr>
<tr>
<td>-------------</td>
<td>-------</td>
<td>-------</td>
<td>---------</td>
<td>------</td>
<td>-----</td>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td>NCDML</td>
<td>0.175</td>
<td>0.145</td>
<td>0.043</td>
<td>0.095</td>
<td>0.149</td>
<td>0.075</td>
<td>0.045</td>
</tr>
<tr>
<td>CDML</td>
<td>0.187</td>
<td>0.142</td>
<td>0.045</td>
<td>0.087</td>
<td>0.124</td>
<td>0.066</td>
<td>0.042</td>
</tr>
<tr>
<td>LMNN [357]</td>
<td>0.183</td>
<td>0.153</td>
<td>0.031</td>
<td>0.093</td>
<td>0.153</td>
<td>0.073</td>
<td><strong>0.013</strong></td>
</tr>
<tr>
<td>LDML [200]</td>
<td>0.159</td>
<td>0.139</td>
<td>0.034</td>
<td>0.079</td>
<td>0.131</td>
<td>0.072</td>
<td>0.068</td>
</tr>
<tr>
<td>MLEC [200]</td>
<td>0.162</td>
<td>0.131</td>
<td>0.042</td>
<td>0.088</td>
<td>0.151</td>
<td>0.039</td>
<td>0.043</td>
</tr>
<tr>
<td>GMML [405]</td>
<td>0.197</td>
<td>0.157</td>
<td>0.051</td>
<td>0.063</td>
<td>0.118</td>
<td>0.067</td>
<td>0.036</td>
</tr>
<tr>
<td>ILHD [63]</td>
<td>0.164</td>
<td>0.162</td>
<td>0.048</td>
<td>0.077</td>
<td>0.117</td>
<td>0.045</td>
<td>0.059</td>
</tr>
<tr>
<td>θ₂-CDML</td>
<td>0.184</td>
<td>0.136</td>
<td>0.037</td>
<td>0.072</td>
<td>0.105</td>
<td>0.053</td>
<td>0.041</td>
</tr>
<tr>
<td>θ₁-CDML</td>
<td>0.173</td>
<td>0.131</td>
<td>0.042</td>
<td>0.064</td>
<td>0.113</td>
<td>0.061</td>
<td>0.026</td>
</tr>
<tr>
<td>θ₂,₁-CDML</td>
<td>0.181</td>
<td>0.129</td>
<td>0.034</td>
<td>0.073</td>
<td>0.121</td>
<td>0.044</td>
<td>0.024</td>
</tr>
<tr>
<td>Tr-CDML [254]</td>
<td>0.166</td>
<td>0.138</td>
<td>0.024</td>
<td>0.076</td>
<td>0.111</td>
<td>0.058</td>
<td>0.037</td>
</tr>
<tr>
<td>IT-CDML</td>
<td>0.174</td>
<td>0.134</td>
<td>0.033</td>
<td>0.061</td>
<td>0.109</td>
<td>0.036</td>
<td><strong>0.013</strong></td>
</tr>
<tr>
<td>Dropout-CDML</td>
<td>0.182</td>
<td>0.140</td>
<td>0.021</td>
<td>0.076</td>
<td>0.114</td>
<td>0.063</td>
<td>0.024</td>
</tr>
<tr>
<td>OS-CDML [116]</td>
<td>0.166</td>
<td>0.133</td>
<td>0.032</td>
<td>0.063</td>
<td>0.108</td>
<td>0.057</td>
<td>0.031</td>
</tr>
<tr>
<td>DCM-NCDML</td>
<td>0.159</td>
<td>0.131</td>
<td>0.035</td>
<td>0.069</td>
<td>0.127</td>
<td>0.064</td>
<td>0.035</td>
</tr>
<tr>
<td>CS-NCDML</td>
<td>0.163</td>
<td>0.135</td>
<td>0.031</td>
<td>0.083</td>
<td>0.103</td>
<td>0.045</td>
<td>0.033</td>
</tr>
<tr>
<td>DPP-NCDML</td>
<td>0.147</td>
<td>0.140</td>
<td>0.038</td>
<td>0.067</td>
<td>0.117</td>
<td>0.072</td>
<td>0.041</td>
</tr>
<tr>
<td>IC-NCDML</td>
<td>0.155</td>
<td>0.127</td>
<td>0.018</td>
<td>0.075</td>
<td>0.116</td>
<td>0.074</td>
<td>0.029</td>
</tr>
<tr>
<td>DC-NCDML</td>
<td>0.164</td>
<td>0.123</td>
<td>0.023</td>
<td>0.082</td>
<td>0.125</td>
<td>0.051</td>
<td>0.033</td>
</tr>
<tr>
<td>VGF-NCDML</td>
<td>0.158</td>
<td>0.136</td>
<td>0.014</td>
<td>0.064</td>
<td>0.136</td>
<td>0.035</td>
<td>0.028</td>
</tr>
<tr>
<td>MA-NCDML</td>
<td>0.143</td>
<td>0.128</td>
<td>0.023</td>
<td>0.078</td>
<td>0.102</td>
<td>0.031</td>
<td>0.042</td>
</tr>
<tr>
<td>OC-NCDML</td>
<td>0.161</td>
<td>0.142</td>
<td>0.032</td>
<td>0.061</td>
<td>0.111</td>
<td>0.063</td>
<td>0.034</td>
</tr>
<tr>
<td>OS-NCDML</td>
<td>0.169</td>
<td>0.137</td>
<td>0.015</td>
<td>0.083</td>
<td>0.119</td>
<td>0.058</td>
<td>0.042</td>
</tr>
<tr>
<td>SFN-NCDML</td>
<td>0.153</td>
<td>0.126</td>
<td>0.022</td>
<td>0.069</td>
<td>0.127</td>
<td>0.043</td>
<td>0.028</td>
</tr>
<tr>
<td>VND-NCDML</td>
<td>0.148</td>
<td>0.135</td>
<td>0.019</td>
<td>0.078</td>
<td>0.116</td>
<td>0.067</td>
<td>0.035</td>
</tr>
<tr>
<td>LDD-NCDML</td>
<td>0.146</td>
<td>0.121</td>
<td>0.017</td>
<td><strong>0.054</strong></td>
<td>0.111</td>
<td>0.036</td>
<td>0.021</td>
</tr>
<tr>
<td>CSFN-CDML</td>
<td>0.142</td>
<td>0.124</td>
<td>0.019</td>
<td>0.062</td>
<td>0.092</td>
<td>0.043</td>
<td><strong>0.019</strong></td>
</tr>
<tr>
<td>CVND-CDML</td>
<td>0.137</td>
<td><strong>0.115</strong></td>
<td><strong>0.008</strong></td>
<td>0.055</td>
<td>0.094</td>
<td>0.038</td>
<td><strong>0.013</strong></td>
</tr>
<tr>
<td>CLDD-CDML</td>
<td>0.131</td>
<td>0.118</td>
<td>0.012</td>
<td>0.058</td>
<td><strong>0.089</strong></td>
<td><strong>0.026</strong></td>
<td>0.016</td>
</tr>
</tbody>
</table>

Table 2.13: The gap of training AUC and testing AUC.

<table>
<thead>
<tr>
<th>Method</th>
<th>Scene</th>
<th>Caltech</th>
<th>Sports</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC [392]</td>
<td>83.6 ± 0.2</td>
<td>42.3 ± 0.4</td>
<td>87.4 ± 0.5</td>
</tr>
<tr>
<td>DCM-SC [242]</td>
<td>85.4 ± 0.5</td>
<td>44.7 ± 0.8</td>
<td>89.6 ± 0.1</td>
</tr>
<tr>
<td>CS-SC [402]</td>
<td>84.8 ± 0.6</td>
<td>45.4 ± 0.5</td>
<td>88.3 ± 0.3</td>
</tr>
<tr>
<td>DPP-SC [429]</td>
<td>84.6 ± 0.3</td>
<td>43.5 ± 0.3</td>
<td>88.1 ± 0.2</td>
</tr>
<tr>
<td>IC-SC [29]</td>
<td>85.5 ± 0.1</td>
<td>43.9 ± 0.7</td>
<td>90.2 ± 0.7</td>
</tr>
<tr>
<td>MA-SC [369]</td>
<td>86.1 ± 0.5</td>
<td>45.6 ± 0.4</td>
<td>89.7 ± 0.4</td>
</tr>
<tr>
<td>AC-SC</td>
<td><strong>86.5 ± 0.7</strong></td>
<td><strong>46.1 ± 0.3</strong></td>
<td><strong>90.9 ± 0.3</strong></td>
</tr>
</tbody>
</table>

Table 2.14: Classification accuracy (%) on three datasets. We compare the proposed ACs with existing diversity-promoting regularizers.
<table>
<thead>
<tr>
<th>Network</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segmental NN [14]</td>
<td>21.9</td>
</tr>
<tr>
<td>MCRBM [88]</td>
<td>20.5</td>
</tr>
<tr>
<td>DSR [326]</td>
<td>19.9</td>
</tr>
<tr>
<td>Rectifier NN [332]</td>
<td>19.8</td>
</tr>
<tr>
<td>DBN [313]</td>
<td>19.7</td>
</tr>
<tr>
<td>Shallow CNN [22]</td>
<td>19.5</td>
</tr>
<tr>
<td>Structured DNN [391]</td>
<td>18.8</td>
</tr>
<tr>
<td>Posterior Modeling [278]</td>
<td>18.5</td>
</tr>
<tr>
<td>Kaldi [277]</td>
<td>18.53</td>
</tr>
<tr>
<td>CS-Kaldi [402]</td>
<td>18.48</td>
</tr>
<tr>
<td>IC-Kaldi [29]</td>
<td>18.46</td>
</tr>
<tr>
<td>MA-Kaldi [369]</td>
<td>18.51</td>
</tr>
<tr>
<td>DC-Kaldi [82]</td>
<td>18.50</td>
</tr>
<tr>
<td>AC-Kaldi</td>
<td>18.41</td>
</tr>
<tr>
<td>CTC [139]</td>
<td>18.4</td>
</tr>
<tr>
<td>RNN Transducer [139]</td>
<td>17.7</td>
</tr>
<tr>
<td>Attention RNN [80]</td>
<td>17.6</td>
</tr>
<tr>
<td>CTC+SCRF [239]</td>
<td>17.4</td>
</tr>
<tr>
<td>Segmental RNN [238]</td>
<td>17.3</td>
</tr>
<tr>
<td>RNNDrop [259]</td>
<td>16.9</td>
</tr>
<tr>
<td>CNN [333]</td>
<td>16.7</td>
</tr>
</tbody>
</table>

Table 2.15: Phone error rate (%) on the TIMIT test set. On the second panel (Kaldi, etc.), we compare AC with previously proposed diversity-promoting regularizers. On the first panel (Segmental NN, etc.) and the third panel (CTC, etc.) are other state of the art baselines.
<table>
<thead>
<tr>
<th>Network</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maxout [135]</td>
<td>9.38</td>
</tr>
<tr>
<td>NiN [229]</td>
<td>8.81</td>
</tr>
<tr>
<td>DSN [217]</td>
<td>7.97</td>
</tr>
<tr>
<td>Highway Network [314]</td>
<td>7.60</td>
</tr>
<tr>
<td>All-CNN [310]</td>
<td>7.25</td>
</tr>
<tr>
<td>ResNet [153]</td>
<td>6.61</td>
</tr>
<tr>
<td>ELU-Network [81]</td>
<td>6.55</td>
</tr>
<tr>
<td>LSUV [257]</td>
<td>5.84</td>
</tr>
<tr>
<td>Fract. Max-Pooling [138]</td>
<td>4.50</td>
</tr>
<tr>
<td>WideResNet [406]</td>
<td>3.89</td>
</tr>
<tr>
<td>CS-WideResNet [402]</td>
<td>3.81</td>
</tr>
<tr>
<td>IC-WideResNet [29]</td>
<td>3.85</td>
</tr>
<tr>
<td>MA-WideResNet [369]</td>
<td>3.68</td>
</tr>
<tr>
<td>DC-WideResNet [82]</td>
<td>3.77</td>
</tr>
<tr>
<td>LCD-WideResNet [291]</td>
<td>3.69</td>
</tr>
<tr>
<td>AC-WideResNet</td>
<td>3.63</td>
</tr>
<tr>
<td>ResNeXt [382]</td>
<td>3.58</td>
</tr>
<tr>
<td>PyramidNet [307]</td>
<td>3.48</td>
</tr>
<tr>
<td>DenseNet [169]</td>
<td>3.46</td>
</tr>
<tr>
<td>PyramidSepDrop [388]</td>
<td>3.31</td>
</tr>
</tbody>
</table>

Table 2.16: Classification error (%) on the CIFAR-10 test set.
<table>
<thead>
<tr>
<th></th>
<th>CNN Dev</th>
<th>CNN Test</th>
<th>DailyMail Dev</th>
<th>DailyMail Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hermann et al. [154]</td>
<td>61.6</td>
<td>63.0</td>
<td>70.5</td>
<td>69.0</td>
</tr>
<tr>
<td>Hill et al. [157]</td>
<td>63.4</td>
<td>6.8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Kadlec et al. [188]</td>
<td>68.6</td>
<td>69.5</td>
<td>75.0</td>
<td>73.9</td>
</tr>
<tr>
<td>Kobayashi et al. [197]</td>
<td>71.3</td>
<td>72.9</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Sordoni et al. [309]</td>
<td>72.6</td>
<td>73.3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Trischler et al. [334]</td>
<td>73.4</td>
<td>74.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Chen et al. [68]</td>
<td>73.8</td>
<td>73.6</td>
<td>77.6</td>
<td>76.6</td>
</tr>
<tr>
<td>Cui et al. [86]</td>
<td>73.1</td>
<td>74.4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Shen et al. [301]</td>
<td>72.9</td>
<td>74.7</td>
<td>77.6</td>
<td>76.6</td>
</tr>
<tr>
<td>BIDAF [296]</td>
<td>76.31</td>
<td>76.94</td>
<td>80.33</td>
<td>79.63</td>
</tr>
<tr>
<td>CS-BIDAF [402]</td>
<td>76.43</td>
<td>77.10</td>
<td>80.37</td>
<td>79.71</td>
</tr>
<tr>
<td>IC-BIDAF [29]</td>
<td>76.41</td>
<td>77.21</td>
<td>80.49</td>
<td>79.83</td>
</tr>
<tr>
<td>MA-BIDAF [369]</td>
<td>76.49</td>
<td>77.09</td>
<td>80.42</td>
<td>79.74</td>
</tr>
<tr>
<td>DC-BIDAF [82]</td>
<td>76.35</td>
<td>77.15</td>
<td>80.38</td>
<td>79.67</td>
</tr>
<tr>
<td>AC-BIDAF</td>
<td>76.62</td>
<td>77.23</td>
<td>80.65</td>
<td>79.88</td>
</tr>
<tr>
<td>Dhingra et al. [101]</td>
<td>77.9</td>
<td>77.9</td>
<td>81.5</td>
<td>80.9</td>
</tr>
<tr>
<td>Dhingra et al. [102]</td>
<td>79.2</td>
<td>78.6</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 2.17: Accuracy (%) on two QA datasets.

<table>
<thead>
<tr>
<th></th>
<th>TIMIT</th>
<th>CIFAR-10</th>
<th>CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>No regularization</td>
<td>1.1</td>
<td>6.3</td>
<td>69.4</td>
</tr>
<tr>
<td>CS [402]</td>
<td>1.2</td>
<td>6.8</td>
<td>74.8</td>
</tr>
<tr>
<td>IC [29]</td>
<td>1.2</td>
<td>6.7</td>
<td>76.1</td>
</tr>
<tr>
<td>MA [369]</td>
<td>1.3</td>
<td>7.0</td>
<td>78.6</td>
</tr>
<tr>
<td>DC [82]</td>
<td>1.5</td>
<td>7.6</td>
<td>82.9</td>
</tr>
<tr>
<td>LCD [291]</td>
<td>-</td>
<td>6.8</td>
<td>-</td>
</tr>
<tr>
<td>AC</td>
<td>1.3</td>
<td>7.1</td>
<td>79.7</td>
</tr>
</tbody>
</table>

Table 2.18: Average runtime (hours). We compare AC with existing diversity-promoting regularizers.

<table>
<thead>
<tr>
<th></th>
<th>#Train</th>
<th>#Test</th>
<th>Dimension</th>
<th>#Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIMIC-III</td>
<td>40K</td>
<td>18K</td>
<td>7207</td>
<td>2833</td>
</tr>
<tr>
<td>Cars</td>
<td>8144</td>
<td>8041</td>
<td>4096</td>
<td>196</td>
</tr>
<tr>
<td>Birds</td>
<td>9000</td>
<td>2788</td>
<td>4096</td>
<td>200</td>
</tr>
<tr>
<td>Scenes-15</td>
<td>3140</td>
<td>1345</td>
<td>-</td>
<td>15</td>
</tr>
<tr>
<td>Caltech-256</td>
<td>20846</td>
<td>8934</td>
<td>-</td>
<td>256</td>
</tr>
<tr>
<td>UIUC-Sports</td>
<td>1254</td>
<td>538</td>
<td>-</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 2.19: Dataset statistics.
Table 2.20: Retrieval precision@10 (%) on three datasets. On the first panel (EUC, etc.) are non-kernel DML methods. On the second panel (Tsang, etc.) are well established or state of the art kernel DML methods. On the third panel (KDML, etc.) are KDML methods regularized by existing regularizers. On the fourth panel (KDML-SFN-RTR, etc.) are KDML methods regularized by our proposed near-orthogonality regularizers.

<table>
<thead>
<tr>
<th>Method</th>
<th>MIMIC-III</th>
<th>Cars</th>
<th>Birds</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>EUC</td>
<td>58.3 ± 0.1</td>
<td>37.8 ± 0.0</td>
<td>43.2 ± 0.0</td>
<td></td>
</tr>
<tr>
<td>ITML [92]</td>
<td>69.3 ± 0.4</td>
<td>50.1 ± 0.0</td>
<td>52.9 ± 0.3</td>
<td></td>
</tr>
<tr>
<td>LDML [146]</td>
<td>70.9 ± 0.9</td>
<td>51.3 ± 0.0</td>
<td>52.1 ± 0.2</td>
<td></td>
</tr>
<tr>
<td>DML-Eig [398]</td>
<td>70.6 ± 0.7</td>
<td>50.7 ± 0.0</td>
<td>53.3 ± 0.8</td>
<td></td>
</tr>
<tr>
<td>Seraph [265]</td>
<td>71.7 ± 0.2</td>
<td>53.6 ± 0.0</td>
<td>52.9 ± 0.2</td>
<td></td>
</tr>
<tr>
<td>GMML [405]</td>
<td>71.2 ± 0.3</td>
<td>54.2 ± 0.0</td>
<td>53.7 ± 0.6</td>
<td></td>
</tr>
<tr>
<td>Tsang [335]</td>
<td>73.5 ± 0.2</td>
<td>55.8 ± 0.0</td>
<td>53.9 ± 0.4</td>
<td></td>
</tr>
<tr>
<td>MKDML [345]</td>
<td>75.1 ± 0.9</td>
<td>53.5 ± 0.0</td>
<td>54.4 ± 0.1</td>
<td></td>
</tr>
<tr>
<td>Jain [176]</td>
<td>74.9 ± 1.1</td>
<td>53.9 ± 0.0</td>
<td>55.9 ± 0.6</td>
<td></td>
</tr>
<tr>
<td>PCCA [252]</td>
<td>73.4 ± 0.5</td>
<td>56.4 ± 0.0</td>
<td>55.1 ± 0.9</td>
<td></td>
</tr>
<tr>
<td>KDML</td>
<td>73.8 ± 0.9</td>
<td>54.9 ± 0.0</td>
<td>54.7 ± 0.5</td>
<td></td>
</tr>
<tr>
<td>KDML-SHN [294]</td>
<td>74.2 ± 0.6</td>
<td>55.4 ± 0.0</td>
<td>54.8 ± 0.9</td>
<td></td>
</tr>
<tr>
<td>KDML-DPP [429]</td>
<td>75.5 ± 0.8</td>
<td>56.4 ± 0.0</td>
<td>57.3 ± 0.3</td>
<td></td>
</tr>
<tr>
<td>KDML-Angle [369]</td>
<td>75.9 ± 0.2</td>
<td>56.8 ± 0.0</td>
<td>57.1 ± 0.6</td>
<td></td>
</tr>
<tr>
<td>KDML-SFN-RTR</td>
<td>76.3 ± 0.7</td>
<td>56.6 ± 0.0</td>
<td>56.4 ± 0.1</td>
<td></td>
</tr>
<tr>
<td>KDML-VND-RTR</td>
<td><strong>77.1</strong> ± 0.6</td>
<td><strong>57.7</strong> ± 0.0</td>
<td>58.9 ± 0.7</td>
<td></td>
</tr>
<tr>
<td>KDML-LDD-RTR</td>
<td>76.7 ± 0.3</td>
<td>57.4 ± 0.0</td>
<td><strong>59.2</strong> ± 0.3</td>
<td></td>
</tr>
<tr>
<td>KDML-SFN-RFF</td>
<td>75.9 ± 0.1</td>
<td>56.5 ± 0.0</td>
<td>56.0 ± 0.2</td>
<td></td>
</tr>
<tr>
<td>KDML-VND-RFF</td>
<td>76.9 ± 0.4</td>
<td>57.2 ± 0.0</td>
<td>58.8 ± 0.6</td>
<td></td>
</tr>
<tr>
<td>KDML-LDD-RFF</td>
<td>76.8 ± 0.8</td>
<td>57.1 ± 0.0</td>
<td>58.5 ± 0.4</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.21: The number of RKHS functions achieving the precision@10 in Table 2.20.

Table 2.21: The number of RKHS functions achieving the precision@10 in Table 2.20.
D4 (204) D5 (176) D6 (148) D7 (131) D8 (121)
82.9 14.8 83.7 0.74 0.72 14.4 83.1 9.4 11.5 82.3 7.2 82.1 82.2 0.82 0.75 80.3 7.4 3.7 8.3 15.1 15.0 82.5 0.30 0.31 0.76 0.71

are frequent diseases, while that D4–D8 are infrequent ones. The number next to a disease ID is its frequency. Note that diseases D1–D3

Table 2.22: Retrieval precision@10 (%) on three frequent and five infrequent diseases in the MIMIC-III dataset. The number next to a disease ID is its frequency. Note that diseases D1–D3 are frequent diseases, while that D4–D8 are infrequent ones.

<table>
<thead>
<tr>
<th>MIMIC-III</th>
<th>Cars</th>
<th>Birds</th>
</tr>
</thead>
<tbody>
<tr>
<td>KDML-SFN-RTR</td>
<td>69.4</td>
<td>17.2</td>
</tr>
<tr>
<td>KDML-VND-RTR</td>
<td>69.8</td>
<td>17.4</td>
</tr>
<tr>
<td>KDML-LDD-RTR</td>
<td>69.9</td>
<td>17.5</td>
</tr>
<tr>
<td>KDML-SFN-RFF</td>
<td>12.6</td>
<td>2.7</td>
</tr>
<tr>
<td>KDML-VND-RFF</td>
<td>12.9</td>
<td>2.8</td>
</tr>
<tr>
<td>KDML-LDD-RFF</td>
<td>12.8</td>
<td>2.8</td>
</tr>
</tbody>
</table>

Table 2.23: Convergence time (hours) on three datasets.

<table>
<thead>
<tr>
<th></th>
<th>Scenes-15</th>
<th>Caltech-256</th>
<th>UIUC-Sports</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC [266]</td>
<td>83.6 ± 0.2</td>
<td>42.3 ± 0.4</td>
<td>87.4 ± 0.5</td>
</tr>
<tr>
<td>KSC [120]</td>
<td>85.4 ± 0.5</td>
<td>44.7 ± 0.8</td>
<td>88.2 ± 0.1</td>
</tr>
<tr>
<td>KSC-SHN [294]</td>
<td>85.8 ± 0.6</td>
<td>45.4 ± 0.5</td>
<td>88.3 ± 0.3</td>
</tr>
<tr>
<td>KSC-DPP [429]</td>
<td>86.3 ± 0.3</td>
<td>47.3 ± 0.8</td>
<td>89.3 ± 0.2</td>
</tr>
<tr>
<td>KSC-Angle [369]</td>
<td>86.8 ± 0.1</td>
<td>46.1 ± 0.8</td>
<td>89.5 ± 0.5</td>
</tr>
<tr>
<td>KSC-SFN-RTR</td>
<td>87.1 ± 0.5</td>
<td>47.2 ± 0.5</td>
<td>89.9 ± 0.3</td>
</tr>
<tr>
<td>KSC-VND-RTR</td>
<td>87.9 ± 0.7</td>
<td><strong>48.6</strong> ± 0.3</td>
<td><strong>91.3</strong> ± 0.5</td>
</tr>
<tr>
<td>KSC-LDD-RTR</td>
<td>87.4 ± 0.4</td>
<td>48.1 ± 0.6</td>
<td>90.7 ± 0.2</td>
</tr>
<tr>
<td>KSC-SFN-RFF</td>
<td>86.8 ± 0.6</td>
<td>46.5 ± 0.1</td>
<td>89.5 ± 0.7</td>
</tr>
<tr>
<td>KSC-VND-RFF</td>
<td>87.5 ± 0.5</td>
<td>48.2 ± 0.4</td>
<td>90.4 ± 0.8</td>
</tr>
<tr>
<td>KSC-LDD-RFF</td>
<td>87.2 ± 0.1</td>
<td>47.8 ± 0.2</td>
<td>90.2 ± 0.4</td>
</tr>
</tbody>
</table>

Table 2.24: Classification accuracy (%) on three datasets.

<table>
<thead>
<tr>
<th></th>
<th>Sensitivity</th>
<th>Specificity</th>
<th>Error rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 [331]</td>
<td>0.76</td>
<td>0.71</td>
<td>0.31</td>
</tr>
<tr>
<td>Elastic Net [428]</td>
<td>0.74</td>
<td>0.72</td>
<td>0.30</td>
</tr>
<tr>
<td>LDD-L1</td>
<td>0.82</td>
<td>0.75</td>
<td>0.24</td>
</tr>
</tbody>
</table>

Table 2.25: Sensitivity and specificity for support recovery and error rate for prediction.
<table>
<thead>
<tr>
<th>Method</th>
<th>20-News Test</th>
<th>20-News Gap</th>
<th>RCV1 Test</th>
<th>RCV1 Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC [266]</td>
<td>0.592</td>
<td>0.119</td>
<td>0.872</td>
<td>0.009</td>
</tr>
<tr>
<td>LDD-SC</td>
<td>0.605</td>
<td>0.108</td>
<td>0.886</td>
<td>0.005</td>
</tr>
<tr>
<td>L1-SC [331]</td>
<td>0.606</td>
<td>0.105</td>
<td>0.897</td>
<td>0.005</td>
</tr>
<tr>
<td>LDD-L1-SC</td>
<td><strong>0.612</strong></td>
<td><strong>0.099</strong></td>
<td><strong>0.909</strong></td>
<td>-0.015</td>
</tr>
</tbody>
</table>

Table 2.26: Classification accuracy on the test sets of 20-News and RCV1, and the gap between training and test accuracy.

<table>
<thead>
<tr>
<th>Basis Vector</th>
<th>Selected Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>crime, guns</td>
</tr>
<tr>
<td>2</td>
<td>faith, trust</td>
</tr>
<tr>
<td>3</td>
<td>worked, manager</td>
</tr>
<tr>
<td>4</td>
<td>weapons, citizens</td>
</tr>
<tr>
<td>5</td>
<td>board, uiuc</td>
</tr>
<tr>
<td>6</td>
<td>application, performance, ideas</td>
</tr>
<tr>
<td>7</td>
<td>service, quality</td>
</tr>
<tr>
<td>8</td>
<td>bible, moral</td>
</tr>
<tr>
<td>9</td>
<td>christ, jews, land, faq</td>
</tr>
</tbody>
</table>

Table 2.27: Selected words of 9 exemplar basis vectors.
<table>
<thead>
<tr>
<th>Network</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN [253]</td>
<td>124.7</td>
</tr>
<tr>
<td>RNN+LDA [253]</td>
<td>113.7</td>
</tr>
<tr>
<td>Deep RNN [273]</td>
<td>107.5</td>
</tr>
<tr>
<td>Sum-Product Network [70]</td>
<td>100.0</td>
</tr>
<tr>
<td>RNN+LDA+KN-5+Cache [253]</td>
<td>92.0</td>
</tr>
<tr>
<td>LSTM (medium) [408]</td>
<td>82.7</td>
</tr>
<tr>
<td>CharCNN [194]</td>
<td>78.9</td>
</tr>
<tr>
<td>LSTM (large) [408]</td>
<td>78.4</td>
</tr>
<tr>
<td>Variational LSTM [115]</td>
<td>73.4</td>
</tr>
<tr>
<td>CS-PytorchLM [402]</td>
<td>71.8</td>
</tr>
<tr>
<td>IC-PytorchLM [29]</td>
<td>71.9</td>
</tr>
<tr>
<td>MA-PytorchLM [369]</td>
<td>72.0</td>
</tr>
<tr>
<td>DC-PytorchLM [82]</td>
<td>72.2</td>
</tr>
<tr>
<td>AC-PytorchLM</td>
<td>71.5</td>
</tr>
<tr>
<td>LDD-PytorchLM</td>
<td>71.6</td>
</tr>
<tr>
<td>L1-PytorchLM [331]</td>
<td>71.8</td>
</tr>
<tr>
<td>LDD-L1-PytorchLM</td>
<td>71.1</td>
</tr>
<tr>
<td>Pointer Sentinel LSTM [250]</td>
<td>70.9</td>
</tr>
<tr>
<td>Ensemble of 38 Large LSTMs [408]</td>
<td>68.7</td>
</tr>
<tr>
<td>Variat. LSTM Ensem. [115]</td>
<td>68.7</td>
</tr>
<tr>
<td>Variational RHN [426]</td>
<td>68.5</td>
</tr>
<tr>
<td>Variational LSTM + REAL [170]</td>
<td>68.5</td>
</tr>
<tr>
<td>Neural Architecture Search [427]</td>
<td>67.9</td>
</tr>
<tr>
<td>Variational RHN + RE [170]</td>
<td>66.0</td>
</tr>
<tr>
<td>Variational RHN + WT [426]</td>
<td>65.4</td>
</tr>
<tr>
<td>Variational RHN+WT+Dropout [426]</td>
<td>64.4</td>
</tr>
<tr>
<td>Architecture Search + WT V1 [427]</td>
<td>64.0</td>
</tr>
<tr>
<td>Architecture Search + WT V2 [427]</td>
<td>62.4</td>
</tr>
</tbody>
</table>

Table 2.28: Word-level perplexities on the PTB test set. On the second panel (PytorchLM, etc.), we compare LDD-L1 with the L1 regularizer and orthogonality-promoting regularizers. On the first panel (RNN, etc.) and the third panel (Pointer Sentinel LSTM, etc.) are other state of the art baselines.
<table>
<thead>
<tr>
<th>Network</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maxout [135]</td>
<td>9.38</td>
</tr>
<tr>
<td>NiN [229]</td>
<td>8.81</td>
</tr>
<tr>
<td>DSN [217]</td>
<td>7.97</td>
</tr>
<tr>
<td>Highway Network [314]</td>
<td>7.60</td>
</tr>
<tr>
<td>All-CNN [310]</td>
<td>7.25</td>
</tr>
<tr>
<td>ResNet [153]</td>
<td>6.61</td>
</tr>
<tr>
<td>ELU-Network [81]</td>
<td>6.55</td>
</tr>
<tr>
<td>LSUV [257]</td>
<td>5.84</td>
</tr>
<tr>
<td>Fract. Max-Pooling [138]</td>
<td>4.50</td>
</tr>
<tr>
<td>WideResNet [406]</td>
<td>3.89</td>
</tr>
<tr>
<td>CS-WideResNet [402]</td>
<td>3.81</td>
</tr>
<tr>
<td>IC-WideResNet [29]</td>
<td>3.85</td>
</tr>
<tr>
<td>MA-WideResNet [369]</td>
<td>3.68</td>
</tr>
<tr>
<td>DC-WideResNet [82]</td>
<td>3.77</td>
</tr>
<tr>
<td>LCD-WideResNet [291]</td>
<td>3.69</td>
</tr>
<tr>
<td>AC-WideResNet</td>
<td>3.63</td>
</tr>
<tr>
<td>LDD-WideResNet</td>
<td>3.65</td>
</tr>
<tr>
<td>L1-WideResNet [331]</td>
<td>3.81</td>
</tr>
<tr>
<td>LDD-L1-WideResNet</td>
<td>3.60</td>
</tr>
<tr>
<td>ResNeXt [382]</td>
<td>3.58</td>
</tr>
<tr>
<td>PyramidNet [307]</td>
<td>3.48</td>
</tr>
<tr>
<td>DenseNet [169]</td>
<td>3.46</td>
</tr>
<tr>
<td>PyramidSepDrop [388]</td>
<td>3.31</td>
</tr>
</tbody>
</table>

Table 2.29: Classification error (%) on the CIFAR-10 test set.
Chapter 3

Diversity-promoting Learning II – Bayesian Inference

In the last chapter, we have studied diversity-promoting learning under a frequentist-style regularized optimization framework, where the component vectors are deterministic and are learned via point estimation [352]. In this chapter, we study how to promote diversity under an alternative learning paradigm: Bayesian inference [46, 172, 260], where the components are considered as random variables of which a posterior distribution shall be computed from data under certain priors. Compared with point estimation, Bayesian learning offers complementary benefits. First, it offers a “model-averaging” [46, 172] effect for ML models when they are used for decision-making and prediction because the parameters shall be integrated under a posterior distribution, thus can potentially alleviate overfitting on training data. Second, it provides a natural way to quantify uncertainties of model parameters, and downstream decisions and predictions made thereupon [46, 172, 260]. Affandi et al. [16] investigated the “diversification” of Bayesian models using the determinantal point process (DPP) [204] prior. DPP has two drawbacks. First, it is not applicable to Bayesian nonparametric models where the number of components is infinite. Second, it is not amenable to developing variational-inference [340] based posterior inference algorithms which are usually more efficient than sampling-based [127] methods. In this chapter, we develop new diversity-promoting priors to address these limitations. Bayesian models can be classified into parametric models where the number of components is finite and nonparametric [111] models where the number of components is infinite in principle and can dynamically grow with data. To promote diversity among a finite number of components in parametric models, we propose a prior called mutual angular process [371] that encourage the components to have large mutual angles. In nonparametric models, we study an infinite mutual angular process [381] which encourages infinitely many components to be diverse.

3.1 Diversity-promoting Learning of Bayesian Parametric Models

We start with promoting diversity in Bayesian parametric models where the number of components is finite and fixed. We investigate two approaches: (1) prior control, which defines
diversity-promoting priors and uses them to affect the posterior via the Bayes rule; (2) posterior regularization, which directly performs diversity-promoting regularization over post-data distributions \[424\]. These two approaches have complementary advantages which will be discussed in detail below.

Following Section 2.3, we adopt a notion of diversity that component vectors are more diverse provided the pairwise angles between them are larger. In the prior control method, we define a mutual angular process (MAP) \[371\], which assigns higher probability density to components that have larger mutual angles. Specifically, we build a Bayesian network \[198\] where the nodes represent the directional vectors of the components and the local probabilities are parameterized by von Mises-Fisher \[245\] distributions that entail an inductive bias towards vectors with larger mutual angles. The MAP is amenable for approximate posterior inference of model components. In particular, they facilitate variational inference (VI) \[340\], which is usually more efficient than sampling-based \[127\] methods. In contrast, it is difficult to derive VI-based algorithms for the determinant point process (DPP) \[204\] prior. Bardenet and Titsias \[30\] derived bounds on the likelihood of a DPP and use these bounds for variational inference. This method is applicable when DPP is used to define the likelihood. However, when DPP is applied as a prior of components in Bayesian models, it is unclear how to apply this method for variational inference of components’ posterior.

It is not always the case that a Bayesian prior can be defined to capture a certain desired effect. For example, Xie et al. \[369\] posit that one ingredient of diversity is evenness: the vectors are desired to uniformly spread out to different directions and each vector is evenly different from all other vectors. To achieve this, they defined a regularizer to encourage the variance of angles between vectors to be small. However, it is very difficult to define a Bayesian prior to achieve such an effect. The major technical difficulty lies in how to ensure the prior integrates to one. To address this issue, we adopt a posterior regularization approach \[424\], in which a diversity-promoting regularizer is directly imposed over the post-data distributions to encourage diversity and the regularizer can be flexibly defined to accommodate various desired diversity-promoting goals.

We apply these two approaches to “diversify” the “experts” (components) in the Bayesian mixture of experts model \[354\]. Posterior inference algorithms based on both variational inference and Markov chain Monte Carlo sampling are developed. Experiments demonstrate the effectiveness and efficiency of our methods.

3.1.1 Mutual Angular Process

We start with defining diversity-promoting Bayesian priors. We desire the priors to have two traits. First, to favor diversity, they assign a higher density to components having larger mutual angles. Second, the priors should facilitate posterior inference. In Bayesian learning, the easiness of posterior inference relies heavily on the prior \[47\], \[341\]. One possible solution is to turn a diversity-promoting regularizer \( \Omega(A) \) (e.g., the uniform eigenvalue regularizer in Section 2.1) into a distribution \( p(A) = \frac{1}{Z} \exp(\Omega(A)) \) based on the Gibbs measure \[195\], where \( Z \) is the partition function guaranteeing \( p(A) \) integrates to one. However, it is not sure whether \( Z = \int_A \exp(\Omega(A)) dA \) is finite, i.e., whether \( p(A) \) is proper. When an improper prior is utilized in Bayesian learning, the posterior is also highly likely to be improper, except in a few special cases
Performing inference on improper posteriors is problematic. At a high-level, we define such a prior in a sequential manner: each time we add one component to the component pool and encourage the new component to be different from the existing ones. This sequential manner has two benefits. First, it can easily define both parametric and nonparametric priors by controlling the number of components to be added. Repeating the adding process \( K \) times, we obtain a parametric prior defined on \( K \) components. Likewise, a nonparametric prior on an infinite set of components can be obtained by repeating the adding process infinitely many times. Second, priors defined in this sequential manner can be naturally decomposed into a sequence of “simpler” distributions, which makes the derivation of variational-inference-based algorithms relatively easier.

Following [369], we adopt a notion of diversity that component vectors are more diverse provided the pairwise angles between them are larger. Given a component vector \( \mathbf{a} \), let \( g = \|\mathbf{a}\|_2 \) be its magnitude and \( \tilde{\mathbf{a}} \) be the directional vector (\( \|\tilde{\mathbf{a}}\|_2 = 1 \)), where \( a = g\tilde{a} \). Noting that the angle between two vectors is invariant to their magnitudes, we can construct the angle-based prior by sequentially adding the directional vectors (DVs) and encouraging the new DV to have large angles with the existing ones. We use a Bayesian network (BNs) (shown in Figure 3.1) to model this adding process, where nodes represent the DVs and the parents of \( \tilde{\mathbf{a}}_i \) are \( \tilde{\mathbf{a}}_1, \ldots, \tilde{\mathbf{a}}_{i-1} \). To encourage large angles between \( \tilde{\mathbf{a}}_i \) and \( \tilde{\mathbf{a}}_1, \ldots, \tilde{\mathbf{a}}_{i-1} \), we define a local probability based on the von Mises-Fisher (vMF) [243] distribution. The probability density function of the vMF distribution is \( f(\mathbf{x}) = C_p(\kappa) \exp(\kappa \mathbf{\mu}^\top \mathbf{x}) \), where the random variable \( \mathbf{x} \in \mathbb{R}^p \) lies on a \( p - 1 \) dimensional sphere (\( \|\mathbf{x}\|_2 = 1 \)), \( \mathbf{\mu} \) is the mean direction with \( \|\mathbf{\mu}\|_2 = 1, \kappa > 0 \) is the concentration parameter, and \( C_p(\kappa) \) is the normalization constant. The vMF-based local probability \( p(\tilde{\mathbf{a}}_i|\text{pa}(\tilde{\mathbf{a}}_i)) \) is defined as

\[
p(\tilde{\mathbf{a}}_i|\text{pa}(\tilde{\mathbf{a}}_i)) = C_p(\kappa) \exp \left( \kappa \left( -\frac{\sum_{j=1}^{i-1} \tilde{\mathbf{a}}_j}{\sum_{j=1}^{i-1} \|\tilde{\mathbf{a}}_j\|_2} \right)^\top \tilde{\mathbf{a}}_i \right),
\]

which encourages large angles: \( \tilde{\mathbf{a}}_i^\top \tilde{\mathbf{a}}_j \) is the cosine of the angle between \( \tilde{\mathbf{a}}_i \) and \( \tilde{\mathbf{a}}_j \); if \( \tilde{\mathbf{a}}_i \) has larger angles with \( \{\tilde{\mathbf{a}}_j\}_{j=1}^{i-1} \), then the average negative cosine similarity \( -\sum_{j=1}^{i-1} \tilde{\mathbf{a}}_j^\top \tilde{\mathbf{a}}_i \) would be larger; accordingly \( p(\tilde{\mathbf{a}}_i|\text{pa}(\tilde{\mathbf{a}}_i)) \) would be larger. For the magnitudes \( \{g_i\}_{i=1}^{K} \) of the components, we sample them independently from a gamma distribution with shape parameter \( \alpha_1 \) and rate parameter \( \alpha_2 \). The generative process of \( \mathbf{A} \) is summarized as follows:

- Draw \( \tilde{\mathbf{a}}_1 \sim \text{vMF}(\mathbf{\mu}_0, \kappa) \)
- For \( i = 2, \ldots, K \), draw \( \tilde{\mathbf{a}}_i \sim \text{vMF}-\frac{\sum_{j=1}^{i-1} \tilde{\mathbf{a}}_j}{\sum_{j=1}^{i-1} \|\tilde{\mathbf{a}}_j\|_2}, \kappa \)
- For \( i = 1, \ldots, K \), draw \( g_i \sim \Gamma(\alpha_1, \alpha_2) \)

Figure 3.1: A Bayesian network representation of the mutual angular process.
• For \( i = 1, \ldots, K \), \( a_i = \tilde{a}_i g_i \).

The probability distribution over \( A \) can be written as

\[
p(A) = C_p(\kappa) \exp(\kappa \mu_0^\top \tilde{a}_1) \prod_{i=2}^{K} C_p(\kappa) \exp \left( \kappa \left( -\frac{\sum_{j=1}^{i-1} \tilde{a}_j}{\| \sum_{j=1}^{i-1} \tilde{a}_j \|_2} \right)^\top \tilde{a}_i \right) \prod_{i=1}^{K} \frac{a_i^\alpha_1 a_i^{1-\alpha} e^{-a_i^\alpha_2}}{\Gamma(\alpha_1)}. \tag{3.2}
\]

According to the factorization theorem [198] of Bayesian networks, it is easy to verify \( \int_A p(A) dA = 1 \), thus \( p(A) \) is a proper prior.

When inferring the posterior of model components using a variational inference method, we need to compute the expectation of \( 1/\| \sum_{j=1}^{i-1} \tilde{a}_j \|_2 \) appearing in the local probability \( p(\tilde{a}_i | p(\tilde{a}_i)) \), which is extremely difficult. To address this issue, we define an alternative local probability that achieves similar modeling effect as \( p(\tilde{a}_i | p(\tilde{a}_i)) \), but greatly facilitates variational inference:

\[
p(\tilde{a}_i | p(\tilde{a}_i)) \propto \exp(\kappa(-\sum_{j=1}^{i-1} \tilde{a}_j)^\top \tilde{a}_i) \\
\propto \exp \left( \kappa \| \sum_{j=1}^{i-1} \tilde{a}_j \|_2 \left( -\frac{\sum_{j=1}^{i-1} \tilde{a}_j}{\| \sum_{j=1}^{i-1} \tilde{a}_j \|_2} \right)^\top \tilde{a}_i \right) \\
= C_p(\kappa \| \sum_{j=1}^{i-1} \tilde{a}_j \|_2) \exp \left( \kappa \| \sum_{j=1}^{i-1} \tilde{a}_j \|_2 \left( -\frac{\sum_{j=1}^{i-1} \tilde{a}_j}{\| \sum_{j=1}^{i-1} \tilde{a}_j \|_2} \right)^\top \tilde{a}_i \right) \\
= C_p(\kappa \| \sum_{j=1}^{i-1} \tilde{a}_j \|_2) \exp(\kappa(-\sum_{j=1}^{i-1} \tilde{a}_j)^\top \tilde{a}_i),
\]

which is another vMF distribution with mean direction \( -\sum_{j=1}^{i-1} \tilde{a}_j / \| \sum_{j=1}^{i-1} \tilde{a}_j \|_2 \) and concentration parameter \( \kappa \| \sum_{j=1}^{i-1} \tilde{a}_j \|_2 \). This local probability is proportional to \( (-\sum_{j=1}^{i-1} \tilde{a}_j)^\top \tilde{a}_i \), which measures the negative cosine similarity between \( \tilde{a}_i \) and its parents. Thereby, \( p(\tilde{a}_i | p(\tilde{a}_i)) \) still encourages large mutual angles. The difference between \( p(\tilde{a}_i | p(\tilde{a}_i)) \) and \( p(\tilde{a}_i | p(\tilde{a}_i)) \) is that in \( p(\tilde{a}_i | p(\tilde{a}_i)) \) the term \( \| \sum_{j=1}^{i-1} \tilde{a}_j \|_2 \) is moved from the denominator to the normalizer, thus we can avoid computing the expectation of \( 1/\| \sum_{j=1}^{i-1} \tilde{a}_j \|_2 \). Though it incurs a new problem that we need to compute the expectation of \( \log C_p(\kappa \| \sum_{j=1}^{i-1} \tilde{a}_j \|_2) \), which is also difficult due to the complex form of the \( C_p(\cdot) \) function, we managed to resolve this problem as detailed in Section 3.1.2. We refer to the mutual angular process defined in Eq. (3.2) as type-I MAP and that with local probability defined in Eq. (3.3) as type-II MAP.

### 3.1.2 Approximate Inference Algorithms

We develop algorithms to infer the posteriors of components under the MAP priors. Since exact posteriors are intractable, we resort to approximate inference techniques. Two main paradigms of approximate inference methods are: (1) variational inference (VI) [340]; (2) Markov chain Monte Carlo (MCMC) sampling [127]. These two approaches possess benefits that are mutually complementary. MCMC can achieve a better approximation of the posterior than VI since it generates samples from the true posterior while VI seeks an approximation of the posterior. However, VI can be computationally more efficient [164]. In this section, we focus on deriving the VI-based algorithm and leave the details of MCMC-based algorithms (which is relatively straightforward) to the appendix.
The basic idea of VI [340] is to use a “simpler” variational distribution $q(A)$ to approximate the true posterior by minimizing the Kullback-Leibler divergence between these two distributions, which is equivalent to maximizing the following variational lower bound w.r.t $q(A)$:

$$
\mathbb{E}_{q(A)}[\log p(D|A)] + \mathbb{E}_{q(A)}[\log p(A)] - \mathbb{E}_{q(A)}[\log q(A)],
$$

(3.4)

where $p(A)$ is the MAP prior and $p(D|A)$ is the data likelihood. Here we choose $q(A)$ to be a mean field variational distribution $q(A) = \prod_{k=1}^{K} q(\tilde{a}_k)q(g_k)$, where $q(\tilde{a}_k) = \text{vMF}(\tilde{a}_k|\tilde{\kappa}_k, \tilde{\kappa}_k)$ and $q(g_k) = \text{Gamma}(g_k|r_k, s_k)$. Given the variational distribution, we first compute the analytical expression of the variational lower bound, in which we particularly discuss how to compute $\mathbb{E}_{q(A)}[\log p(A)]$. If choosing $p(A)$ to be the type-I MAP prior (Eq.(3.2)), we need to compute $\mathbb{E}([-\sum_{j=1}^{i-1} \tilde{a}_j]/\lambda)|\tilde{a}_i]$ which is very difficult to deal with due to the presence of $1/\sum_{j=1}^{i-1} \tilde{a}_j\|_2$. Instead we choose the type-II MAP prior. Under type-II MAP, we need to compute $\mathbb{E}_{q(A)}[\log Z_i]$ for all $i$, where $Z_i = 1/C_p(\kappa)\|\sum_{j=1}^{i-1} \tilde{a}_j\|_2$ is the partition function of $p(\tilde{a}_i|p(\tilde{a}_i|\gamma))$. The analytical form of this expectation is difficult to derive as well due to the complexity of the $C_p(x)$ function: $C_p(x) = \int_{p/2-1}^{p/2} I_p(x)\frac{dx}{y^{p/2-1}}$ where $I_p(x)$ is the modified Bessel function of the first kind at order $p/2 - 1$ [245]. To address this issue, we derive an upper bound of $\log Z_i$ and compute the expectation of the upper bound, which is relatively easy to do. Consequently, we obtain a further lower bound of the variational lower bound and learn the variational and model parameters w.r.t the new lower bound. Now we proceed to derive the upper bound of $\log Z_i$, which equals $\log \int \exp(\kappa(-\sum_{j=1}^{i-1} \tilde{a}_j)|\tilde{a}_i)\,d\tilde{a}_i$. Applying the inequality $\log \int \exp(x)\,dx \leq \gamma + \int \log(1+\exp(x-\gamma))\,dx$ [50], where $\gamma$ is a variational parameter, we have

$$
\log Z_i \leq \gamma + \int \log(1+\exp(\kappa(-\sum_{j=1}^{i-1} \tilde{a}_j|\tilde{a}_i-\gamma))\,d\tilde{a}_i.
$$

(3.5)

Then applying the inequality $\log(1+e^{-x}) \leq \log(1+e^{-\xi}) - \frac{x-\xi}{2} - \frac{1/2-g(\xi)}{2} (x^2 - \xi^2)$ [50], where $\xi$ is another variational parameter and $g(\xi) = 1/(1+\exp(-\xi))$, we have

$$
\log Z_i \leq \gamma + \int \log(1+e^{-\xi}) - (\kappa(\sum_{j=1}^{i-1} \tilde{a}_j|\tilde{a}_i + \gamma - \xi)/2 - \frac{1/2-g(\xi)}{2} ((\kappa(\sum_{j=1}^{i-1} \tilde{a}_j|\tilde{a}_i + \gamma)2 - \xi^2))\,d\tilde{a}_i.
$$

(3.6)

Finally applying the following integrals on a high-dimensional sphere: (1) $\int_{\|y\|=1} \frac{1\,dy}{(\frac{1}{2} + \frac{p}{2})}$, (2) $\int_{\|y\|=1} x\cdot y\,dy = 0$, (3) $\int_{\|y\|=1} (x\cdot y)^2\,dy \leq \|x\|^2\frac{2\pi^{(p+1)/2}}{\Gamma(p/2+1)}$, we get

$$
\log Z_i \leq -\frac{1/2-g(\xi)}{2}\kappa^2\sum_{j=1}^{i-1} \tilde{a}_j\|_2^2 \frac{2\pi^{(p+1)/2}}{\Gamma(p/2+1)} + \gamma + \log(1+e^{-\xi}) + \frac{\xi-\gamma}{2} + \frac{1/2-g(\xi)}{2} (\xi^2 - \gamma^2)\frac{2\pi^{(p+1)/2}}{\Gamma(p/2+1)}.
$$

(3.7)

The expectation of this upper bound is much easier to compute. Specifically, we need to tackle $\mathbb{E}_{q(A)}[\|\sum_{j=1}^{i-1} \tilde{a}_j\|_2^2]$, which can be computed as

$$
\mathbb{E}_{q(A)}[\|\sum_{j=1}^{i-1} \tilde{a}_j\|_2^2] = \mathbb{E}_{q(A)}[\sum_{j=1}^{i-1} \tilde{a}_j \tilde{a}_j] + \sum_{j=1}^{i-1} \sum_{k \neq j}^{i-1} \tilde{a}_j \tilde{a}_k + \sum_{j=1}^{i-1} \sum_{k \neq j}^{i-1} \mathbb{E}_{q(\tilde{a}_j|\tilde{a}_i)}\mathbb{E}_{q(\tilde{a}_k)}\tilde{a}_j \tilde{a}_k
$$

$$
= \sum_{j=1}^{i-1} \text{tr}(\mathbb{C}(\tilde{a}_j)) + \sum_{j=1}^{i-1} \sum_{k \neq j}^{i-1} \mathbb{E}_{q(\tilde{a}_j|\tilde{a}_i)}\mathbb{E}_{q(\tilde{a}_k)}\tilde{a}_j \tilde{a}_k,
$$

(3.8)
where $E_q(\tilde{a}_j) = A_p(\tilde{\kappa})\tilde{a}_j$, $\text{cov}(\tilde{a}_j) = \frac{h(\tilde{\kappa})}{\tilde{\kappa}}I + (1 - 2\frac{\nu + 1}{\tilde{\kappa}} - h^2(\tilde{\kappa}))\tilde{a}_j\tilde{a}_j^T$, $h(\tilde{\kappa}) = \frac{I_{\nu + 1}(\tilde{\kappa})}{I_{\nu}(\tilde{\kappa})}$, $A_p(\tilde{\kappa}) = \frac{I_{\nu/2}(\tilde{\kappa})}{I_{\nu/2 - 1}(\tilde{\kappa})}$ and $\nu = p/2 - 1$.

3.1.3 Diversity-promoting Posterior Regularization

In practice, one may desire to achieve more than one diversity-promoting effects. For example, the mutual angular regularizer [369] aims at encouraging the pairwise angles between components to have not only large mean, but also small variance such that the components are uniformly “different” from each other and evenly spread out to different directions in the space. It would be extremely difficult, if ever possible, to define a proper prior that can accommodate all desired effects. To overcome such inflexibility of the prior control method, we resort to a posterior regularization approach [424]. Instead of designing a Bayesian prior to encode the diversification desideratum and indirectly influencing the posterior, posterior regularization directly imposes a control over the post-data distributions to achieve diversity.

The basic idea of posterior regularization [424] is: formulate the posterior estimation problem as an optimization problem defined over a family of distributions; add a regularizer, which encodes the desideratum, on the distributions to be optimized. In general, designing a regularizer is much easier than designing a proper prior. Giving the prior $\pi(A)$ and the data likelihood $p(D|A)$, computing the posterior $p(A|D)$ is equivalent to solving the following optimization problem [424]

$$\sup_{q(A)} \mathbb{E}_q[A] \left[ \log p(D|A)\pi(A) \right] - \mathbb{E}_q(A) \left[ \log q(A) \right],$$

where $q(A)$ is any valid probability distribution. To bring in diversity into the posterior, we impose a diversity-promoting regularizer $\mathcal{R}(q(A))$ over $q(A)$ and solve the following regularized problem:

$$\sup_{q(A)} \mathbb{E}_q(A) \left[ \log p(D|A)\pi(A) \right] - \mathbb{E}_q(A) \left[ \log q(A) \right] + \lambda \mathcal{R}(q(A)),$$

where $\lambda$ is a tradeoff parameter.

Here we present a specific example of $\mathcal{R}(q(A))$ while noting that many other choices are applicable. Gaining insight from [369], we define $\mathcal{R}(q(A))$ as

$$\Omega(\{E_q(a_i)\}, \{a_i\}) = \frac{1}{K(K-1)} \sum_{i=1}^K \sum_{j \neq i} \theta_{ij} - \gamma \frac{1}{K(K-1)} \sum_{i=1}^K \sum_{j \neq i} \left( \theta_{ij} - \frac{1}{K(K-1)} \sum_{p=1}^K \sum_{q \neq p} \theta_{pq} \right)^2,$$

where $\theta_{ij} = \arccos \left( \frac{E[a_i] \cdot E[a_j]}{||E[a_i]||_2 ||E[a_j]||_2} \right)$ is the angle measuring the dissimilarity between $E[a_i]$ and $E[a_j]$, and the regularizer is defined as the mean of pairwise angles minus their variance. The intuition behind this regularizer is: if the mean of angles is larger (indicating these vectors are more different from each other on the whole) and the variance of the angles is smaller (indicating these vectors evenly spread out to different directions), then these vectors are more diverse.

While posterior regularization is more flexible, it lacks some strengths possessed by the prior control method. First, prior control is a more natural way of incorporating prior knowledge, with solid theoretical foundation. Second, prior control can facilitate sampling-based algorithms that are not applicable in posterior regularization.
3.1.4 Case Study: Bayesian Mixture of Experts Model

In this section, we apply the two approaches developed above to “diversify” the Bayesian mixture of experts model (BMEM) \[354\].

**BMEM with mutual angular process** The mixture of experts model (MEM) \[186\] has been widely used for machine learning tasks where the distribution of input data is so complicated that a single model (“expert”) cannot be effective for all the data. The MEM model assumes that the input data inherently belongs to multiple latent groups and one single “expert” is allocated to each group to handle the data therein. Here we consider a binary classification task. Let \(D = \{(x_i, y_i)\}_{i=1}^N\) be the training data where \(x\) is the input feature vector and \(y_i \in \{1, 0\}\) is the class label. MEM assumes there are \(K\) latent experts where each expert is a classifier with a coefficient vector \(\beta\). Given a test sample \(x\), it first goes through a gate function that decides which expert is best suitable to classify this sample and the decision is made in a probabilistic way. A discrete variable \(z\) is utilized to indicate the selected expert and the probability of assigning sample \(x\) to expert \(k\) \((z = k)\) is \(\frac{\exp(\eta_k^\top x)}{\sum_{j=1}^K \exp(\eta_j^\top x)}\) where \(\eta_k\) is a coefficient vector characterizing the selection of expert \(k\). Given the selected expert, the sample is classified using the coefficient vector \(\beta\) corresponding to that expert. As described in Figure 3.2, the generative process of \(\{(x_i, y_i)\}_{i=1}^N\) is as follows

- For \(i = 1, \cdots, N\)
  - Draw \(z_i \sim \text{Multi}(\zeta)\), \(\zeta_k = \frac{\exp(\eta_k^\top x_i)}{\sum_{j=1}^K \exp(\eta_j^\top x_i)}\)
  - Draw \(y_i \sim \text{Bernoulli}\left(\frac{1}{1 + \exp(-\beta_i^\top x_i)}\right)\)

As of now, the model parameters \(B = \{\beta_k\}_{k=1}^K\) and \(H = \{\eta_k\}_{k=1}^K\) are deterministic variables. Next we place a prior over them to enable Bayesian learning \[354\] and desire this prior to be able to promote diversity among the experts to retain the advantages of “diversification” as stated.
before. The mutual angular processes can be applied to achieve this goal

\[
p(B) = C_p(\kappa) \exp(\kappa \mu_0^\top \hat{\beta}_1) \prod_{i=2}^K C_p(\kappa) \exp \left( -\frac{\sum_{j=1}^{i-1} \hat{\beta}_j}{\| \sum_{j=1}^{i-1} \hat{\beta}_j \|_2} \right)^\top \hat{\beta}_i \prod_{i=1}^K \frac{\alpha_1^i \Gamma(\alpha_1^i - 1)}{\Gamma(\alpha_1)} e^{-\alpha_1^i \| \hat{\beta}_i \|_2^2},
\]

(3.12)

\[
p(H) = C_p(\kappa) \exp(\kappa \xi_0^\top \tilde{\eta}_1) \prod_{i=2}^K C_p(\kappa) \exp \left( -\frac{\sum_{j=1}^{i-1} \tilde{\eta}_j}{\| \sum_{j=1}^{i-1} \tilde{\eta}_j \|_2} \right)^\top \tilde{\eta}_i \prod_{i=1}^K \frac{\omega_1^i \Gamma(\omega_1^i - 1)}{\Gamma(\omega_1)} e^{-\omega_1^i \| \tilde{\eta}_i \|_2^2},
\]

(3.13)

where \( \beta_k = g_k \hat{\beta}_k \) and \( \eta_k = h_k \tilde{\eta}_k \).

**BMEM with diversity-promoting posterior regularization** As an example approach, the diversity among the experts can be imposed by placing the mutual angular regularizer (Eq. (3.11)) over the post-data posteriors. The latent variables in BMEM include \( B, H, z = \{z_i\}_{i=1}^N \). The post-data distribution over them is defined as \( q(B, H, z) = q(B)q(H)q(z) \). For computational tractability, we define \( q(B) \) and \( q(H) \) to be: \( q(B) = \prod_{k=1}^K q(\beta_k)q(g_k) \) and \( q(H) = \prod_{k=1}^K q(\tilde{\eta}_k)q(h_k) \) where \( q(\beta_k), q(\tilde{\eta}_k) \) are von Mises-Fisher distributions and \( q(g_k), q(h_k) \) are gamma distributions, and define \( q(z) \) to be multinomial distributions: \( q(z) = \prod_{i=1}^N q(z_i|\phi_i) \) where \( \phi_i \) is a multinomial vector. The priors over \( B \) and \( H \) are specified to be: \( \pi(B) = \prod_{k=1}^K \beta_k p(\beta_k) \) and \( \pi(H) = \prod_{k=1}^K h_k p(h_k) \) where \( p(\beta_k), p(h_k) \) are vMF distributions and \( p(g_k), p(h_k) \) are gamma distributions. Under such a parameterization, we solve the following diversity-promoting posterior regularization problem

\[
\sup_{q(B, H, z)} \mathbb{E}_{q(B, H, z)}[\log p(y_i|z_i)_{i=1}^N, z|B, H] - \mathbb{E}_{q(B, H, z)}[\log q(B, H, z)] + \lambda_1 \Omega(\{q(\beta_k)\}_{k=1}^K) + \lambda_2 \Omega(\{q(\tilde{\eta}_k)\}_{k=1}^K).
\]

(3.14)

Note that other parameterizations are also valid, such as placing Gaussian priors over \( B \) and \( H \) and setting \( q(B), q(H) \) to be Gaussian.

### 3.1.5 Evaluation

Using the Bayesian mixture of experts model as an instance, we conducted experiments to verify the effectiveness and efficiency of our proposed approaches.

**Datasets** We used two binary-classification datasets. The first one is the Adult-9 \[275\] dataset, which has \( \sim 33K \) training instances and \( \sim 16K \) testing instances. The feature dimension is 123. The other dataset is SUN-Building compiled from the SUN \[364\] dataset, which contains \( \sim 6K \) building images and 7K non-building images randomly sampled from other categories, where 70% of images were used for training and the rest for testing. We used SIFT-based \[236\] bag-of-words to represent the images with a dimension of 1000.

**Experimental settings** To understand the effects of diversification in Bayesian learning, we compared with the following methods: (1) mixture of experts model (MEM) with L2 regularization (MEM-L2) where the L2 regularizer is imposed over “experts” independently; (2) MEM
with diversity-promoting mutual angular regularization [369] (MEM-MAR); (3) Bayesian MEM with a Gaussian prior (BMEM-G) where the “experts” are independently drawn from a Gaussian prior; (4) BMEM with a diversity-promoting determinantal point process [429] prior (BMEM-DPP); (5) BMEM with type-(I,II) mutual angular process priors (BMEM-MAP-I, BMEM-MAP-II); (6) BMEM with posterior regularization (BMEM-PR). BMEM-MAP-I was inferred with MCMC sampling and BMEM-MAP-II was inferred with variational inference. In BMEM-DPP [204], a Metropolis-Hastings sampling algorithm was adopted (variational inference and Gibbs sampling [16] are not applicable). The key parameters involved in the above methods are: (1) the regularization parameter $\lambda$ in MEM-L2, MEM-MAR, and BMEM-PR; (2) the concentration parameter $\kappa$ in the mutual angular processes and variational distributions; (3) the scale parameter of the radial basis function kernel in DPP. All parameters were tuned using 5-fold cross validation. Besides internal comparison, we also compared with 5 well-established classification methods achieving state of the art performance. They are: (1) kernel support vector machine (KSVM) [58]; (2) random forest (RF) [55]; (3) deep neural network (DNN) [160]; (4) infinite SVM (ISVM) [423].

**Results** Table 3.1 and 3.2 show the classification accuracy under different number of “experts” on the Adult-9 and SUN-Building dataset respectively. From these two tables, we observe that: (1) MAP-(I,II) outperform Gaussian, demonstrating the effectiveness of promoting diversity in Bayesian models. (2) MAP-(I,II) outperform DPP, showing their better capability in promoting diversity than DPP. (3) Bayesian learning achieves better performance than point estimation, which is manifested by comparing BMEM-G with MEM-L2, and comparing BMEM-MAP-

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>82.6</td>
<td>85.3</td>
<td>83.4</td>
<td>85.1</td>
<td><strong>87.1</strong></td>
<td>86.4</td>
<td>86.2</td>
</tr>
<tr>
<td>10</td>
<td>83.8</td>
<td>86.4</td>
<td>84.2</td>
<td>85.8</td>
<td><strong>88.3</strong></td>
<td>87.8</td>
<td>87.9</td>
</tr>
<tr>
<td>20</td>
<td>84.3</td>
<td>86.6</td>
<td>84.9</td>
<td>86.5</td>
<td><strong>88.6</strong></td>
<td>88.1</td>
<td>88.7</td>
</tr>
<tr>
<td>30</td>
<td>84.7</td>
<td>87.1</td>
<td>84.9</td>
<td>86.1</td>
<td><strong>88.9</strong></td>
<td>88.4</td>
<td>88.1</td>
</tr>
</tbody>
</table>

Table 3.1: Classification accuracy (%) on the Adult-9 dataset.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>76.2</td>
<td>81.3</td>
<td>76.5</td>
<td>80.4</td>
<td><strong>82.1</strong></td>
<td>80.9</td>
<td>81.7</td>
</tr>
<tr>
<td>10</td>
<td>78.8</td>
<td>82.1</td>
<td>78.6</td>
<td>81.6</td>
<td><strong>83.6</strong></td>
<td>82.8</td>
<td>84.1</td>
</tr>
<tr>
<td>20</td>
<td>79.4</td>
<td>82.7</td>
<td>80.2</td>
<td>83.7</td>
<td><strong>85.3</strong></td>
<td>84.9</td>
<td>83.8</td>
</tr>
<tr>
<td>30</td>
<td>79.7</td>
<td>82.3</td>
<td>80.4</td>
<td>84.5</td>
<td><strong>85.2</strong></td>
<td>84.9</td>
<td>84.9</td>
</tr>
</tbody>
</table>

Table 3.2: Classification accuracy (%) on the SUN-Building dataset.
Table 3.3: Training time (hours) of different methods with $K = 30$.

<table>
<thead>
<tr>
<th>Category ID</th>
<th>Adult-9</th>
<th>SUN-Building</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMEM-DPP</td>
<td>8.2</td>
<td>11.7</td>
</tr>
<tr>
<td>BMEM-MAP-I</td>
<td>7.5</td>
<td>10.5</td>
</tr>
<tr>
<td>BMEM-MAP-II</td>
<td>2.9</td>
<td>4.1</td>
</tr>
<tr>
<td>BMEM-PR</td>
<td>3.3</td>
<td>4.9</td>
</tr>
</tbody>
</table>

Table 3.4: Accuracy on 9 subcategories of the CCAT category in the RCV1.Binary dataset.

<table>
<thead>
<tr>
<th>Category ID</th>
<th>C18</th>
<th>C17</th>
<th>C12</th>
<th>C14</th>
<th>C22</th>
<th>C34</th>
<th>C23</th>
<th>C32</th>
<th>C16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Documents</td>
<td>5281</td>
<td>4125</td>
<td>1194</td>
<td>741</td>
<td>611</td>
<td>483</td>
<td>262</td>
<td>208</td>
<td>192</td>
</tr>
<tr>
<td>BMEM-G Accuracy (%)</td>
<td>87.3</td>
<td>88.5</td>
<td>75.7</td>
<td>70.1</td>
<td>71.6</td>
<td>64.2</td>
<td>55.9</td>
<td>57.4</td>
<td>51.3</td>
</tr>
<tr>
<td>BMEM-MAP-I Accuracy (%)</td>
<td>88.1</td>
<td>86.9</td>
<td>74.7</td>
<td>72.2</td>
<td>70.5</td>
<td>67.3</td>
<td>68.9</td>
<td>70.1</td>
<td>65.5</td>
</tr>
<tr>
<td>Relative Improvement (%)</td>
<td>1.0</td>
<td>-1.8</td>
<td>-1.3</td>
<td>2.9</td>
<td>-1.6</td>
<td>4.6</td>
<td>18.9</td>
<td>18.1</td>
<td>21.7</td>
</tr>
</tbody>
</table>

(I,II)/BMEM-PR with MEM-MAR. (4) BMEM-MAP-I works better than BMEM-MAP-II and BMEM-PR. The possible reason is that BMEM-MAP-I inferred with MCMC draws samples from the true posterior while BMEM-MAP-II and BMEM-PR inferred with variational inference (VI) seek an approximation of the posterior.

However, BMEM-MAP-II is computationally more efficient than BMEM-MAP-I. Table 3.3 compares the time (hours) taken by each method to achieve convergence, with $K$ set to 30. BMEM-MAP-II is more efficient than BMEM-MAP-I, due to the higher efficiency of VI than MCMC. Since VI is not applicable for DPP, BMEM-DPP is inferred using MCMC sampling, hence is less efficient than BMEM-MAP-II.

To verify whether our methods can better capture infrequent patterns, from the RCV1 [220] dataset we picked up a subset of documents such that the frequencies of categories have a power-law distribution. Specifically, we chose documents from 9 subcategories (shown in the 1st row of Table 3.4) of the CCAT category as the positive instances, and randomly sampled 15K documents from non-CCAT categories as negative instances. The 2nd row shows the number of documents in each of the 9 categories. The distribution of these document frequencies is in a power-law fashion, where frequent categories (such as C18 and C17) have a lot of documents while infrequent categories (such as C32 and C16) have a small number of documents. The 3rd and 4th row show the accuracy achieved by BMEM-G and BMEM-MAP-I on each category. The 5th row shows the relative improvement of BMEM-MAP-I over BMEM-G, which is defined as $\frac{A_{map} - A_g}{A_g}$, where $A_{map}$ and $A_g$ denote the accuracy achieved by BMEM-MAP-I and BMEM-G respectively. While achieving accuracy comparable to BMEM-G on the frequent categories, BMEM-MAP-I obtains much better performance on the infrequent categories. For example, the relative improvements on infrequent categories C32 and C16 are 18.1% and 21.7%. This demonstrates that BMEM-MAP-I can effectively capture infrequent patterns.

Our proposed diversity-promoting methods can reduce model size without sacrificing modeling power. As can be seen from Table 3.1 and 3.2, using a smaller number of components, BMEM-MAP-(I,II) and BMEM-PR achieve accuracy that is comparable to or even better than the non-diversified BMEM-G model. For example, on the Adult-9 dataset (Table 3.2), with 5
experts BMEM-MAP-I is able to achieve an accuracy of 87.1%, which cannot be achieved by BMEM-G with even 30 experts.

Table 3.5 presents the comparison with state of the art classification methods. Our methods outperform the baselines.

3.1.6 Appendix: Details of Algorithms

Variational Inference for Type I MAP

In this section, we present details on how to derive the variational lower bound

$$E_{q(A)}[\log p(D|A)] + E_{q(A)}[\log p(A)] - E_{q(A)}[\log q(A)],$$

(3.15)

where the variational distribution $q(A)$ is chosen to be

$$q(A) = \prod_{k=1}^{K} q(\tilde{a}_k)q(g_k) = \prod_{k=1}^{K} \text{vMF}(\tilde{a}_k|\hat{a}_k, \hat{\kappa})\text{Gamma}(g_k|r_k, s_k).$$

(3.16)

Among the three expectation terms, $E_{q(A)}[\log p(A)]$ and $E_{q(A)}[\log q(A)]$ are model-independent and we discuss how to compute them in this section. $E_{q(A)}[\log p(D|A)]$ depends on the specific model and a concrete example will be given later on.

First we introduce some equalities and inequalities. Let $a \sim \text{vMF}(\mu, \kappa)$, then

(I) $E[a] = A_p(\kappa)\mu$ where $A_p(\kappa) = I_{p/2}^2(\kappa)$, and $I_v(\cdot)$ denotes the modified Bessel function of the first kind at order $v$.

(II) $\text{cov}(a) = \frac{h(\kappa)}{\kappa}I + (1 - 2\frac{\nu+1}{\kappa}h(\kappa) - h^2(\kappa))\mu\mu^T$, where $h(\kappa) = \frac{I_{\nu+1}}{I_{\nu}(\kappa)}$ and $\nu = p/2 - 1$.

Please refer to [15] for the derivation of $E[a]$ and $\text{cov}(a)$.

(III) $E[a^T a] = \text{tr}(\text{cov}(a)) + A_p^2(\kappa)\mu^T \mu$.

Proof

$$E[\text{tr}(a^T a)] = E[\text{tr}(aa^T)] = \text{tr}(E[aa^T])$$

$$= \text{tr}(\text{cov}(a) + E[a]E[a]^T) = \text{tr}(\text{cov}(a)) + \text{tr}(E[a]E[a]^T)$$

$$= \text{tr}(\text{cov}(a)) + A_p^2(\kappa)\mu^T \mu$$

(3.17)

Let $g \sim \text{Gamma}(\alpha, \beta)$, then

(IV) $E[g] = \frac{\alpha}{\beta}$. 

Table 3.5: Classification accuracy (%) on two datasets.
(V) $\mathbb{E}[\log g] = \psi(\alpha) - \log \beta$.

(VI) $\log \sum_{k=1}^K \exp(x_k) \leq \gamma + \sum_{k=1}^K \log(1 + \exp(x_k - \gamma))$ and $\log \int \exp(x)dx \leq \gamma + \int \log(1 + \exp(x - \gamma))dx$, where $\gamma$ is a variational parameter. See [50] for the proof.

(VII) $\log(1 + e^{-x}) \leq \log(1 + e^{-\xi}) - \frac{x - \xi}{2} - \frac{1}{2} - \frac{\gamma}{2} \log(1 + e^{-\xi}) + \frac{x - \xi}{2} - \frac{1}{2} - \frac{\gamma}{2} \log(1 + e^{-\xi})$, where $\xi$ is a variational parameter and $g(\xi) = 1/(1 + \exp(-\xi))$. See [50] for the proof.

(VIII) $\int_{\|y\|=1} 1dy = \frac{2^{p+1/2}}{\Gamma(\frac{p+1}{2})}$, which is the surface area of the $p$-dimensional unit sphere. $\Gamma(\cdot)$ is the gamma function.

(IX)$\int_{\|y\|=1} x^T y dy = 0$, which can be shown according to the symmetry of the unit sphere.

(X)$\int_{\|y\|=1} (x^T y)^2 dy \leq \|x\|^2 \frac{2\pi^{(p+1)/2}}{\Gamma(\frac{p+1}{2})}$.

\[\text{Proof}\]

\[\int_{\|y\|=1} (x^T y)^2 dy = \|x\|^2 \int_{\|y\|=1} ((\frac{x}{\|x\|})^T y)^2 dy = \|x\|^2 \int_{\|y\|=1} (\hat{y})^2 dy \]

\[\text{(according to the symmetry of unit sphere)} \]

\[\leq \|x\|^2 \int_{\|y\|=1} 1dy = \|x\|^2 \frac{2\pi^{(p+1)/2}}{\Gamma(\frac{p+1}{2})} \]

Given these equalities and inequalities, we can upper bound $\log Z_i$ and use this upper bound to lower-bound $\mathbb{E}_{q(A)}[\log p(A)]$

\[\mathbb{E}_{q(A)}[\log p(A)] = \mathbb{E}_{q(A)}[\log p(\hat{a}_1) \prod_{i=2}^K p(\hat{a}_i | \{\hat{a}_j\}_{j=1}^{i-1}) \prod_{i=1}^K q(g_i)] = \mathbb{E}_{q(A)}[\log p(\hat{a}_1) \prod_{i=2}^K \exp(\sum_{j=1}^{i-1} \hat{a}_j) \hat{a}_i] \prod_{i=1}^K \frac{2\pi^{(i-1)/2} \Gamma(\frac{i-1}{2})}{\Gamma(\frac{p+1}{2})} + K(\alpha_1 \log \alpha_2)

- \log \Gamma(\alpha_1) + \sum_{i=1}^K (\alpha_1 - 1) \mathbb{E}_{q(g_i)}[\log g_i] - \alpha_2 \mathbb{E}_{q(g_i)}[g_i] + \text{const}

\[\geq \kappa A_p(\hat{k}) \mu_0 \hat{a}_1 + \sum_{i=2}^K (-\kappa A_p(\hat{k})^2 \sum_{j=1}^{i-1} \hat{a}_j \cdot \hat{a}_i - \gamma_i - (\log(1 + e^{-\xi_i}) + \frac{\xi_i - \gamma_i}{2}) \sum_{j=1}^{i-1} \hat{a}_j \cdot \hat{a}_i + \sum_{j=1}^{i-1} \text{tr}(A_j)) + K(\alpha_1 \log \alpha_2)

- \log \Gamma(\alpha_1) + \sum_{i=1}^K (\alpha_1 - 1) \mathbb{E}_{q(g_i)}[\log g_i] - \alpha_2 \mathbb{E}_{q(g_i)}[g_i] + \text{const} \]

where $\Lambda_j = \frac{h(\hat{k})^2}{\widehat{k}} \mathbf{I} + (1 - 2 \frac{\nu+1}{\hat{k}} h(\hat{k}) - h^2(\hat{k})) \hat{a}_j \hat{a}_j^T$.

The other expectation term $\mathbb{E}_{q(A)}[\log q(A)]$ can be computed as

\[\mathbb{E}_{q(A)}[\log q(A)] = \mathbb{E}_{q(A)}[\log \prod_{k=1}^K \text{vMF}(\hat{a}_k | \hat{a}_k, \hat{k})] \Gamma(\alpha_k | r_k, s_k)\]

\[= \sum_{k=1}^K \kappa A_p(\hat{k}) \|\hat{a}_k\|^2 + r_k \log s_k - \log \Gamma(r_k) + (r_k - 1)(\psi(r_k) - \log(s_k)) - r_k. \]

95
Variational Inference for BMEM-MAP-II

In this section, we discuss how to derive the variational lower bound for BMEM-MAP-II. The latent variables are \( \{ \beta_k \}_{k=1}^K, \{ \eta_k \}_{k=1}^K \), and \( \{ z_n \}_{n=1}^N \). The joint distribution of all variables is

\[
p(\{ \beta_k \}_{k=1}^K, \{ \eta_k \}_{k=1}^K, \{ x_n, y_n, z_n \}_{n=1}^N) = p(\{ y_n \}_{n=1}^N | \{ x_n \}_{n=1}^N, \{ \beta_k \}_{k=1}^K) p(\{ z_n \}_{n=1}^N | \{ x_n \}_{n=1}^N, \{ \eta_k \}_{k=1}^K) p(\{ \beta_k \}_{k=1}^K) p(\{ \eta_k \}_{k=1}^K).
\]

To perform variational inference, we employ a mean field variational distribution

\[
Q = q(\{ \beta_k \}_{k=1}^K, \{ \eta_k \}_{k=1}^K, \{ z_n \}_{n=1}^N) = \prod_{k=1}^K q(\beta_k) q(\eta_k) \prod_{n=1}^N q(z_n) = \prod_{k=1}^K \text{vMF}(\beta_k | \bar{\beta}_k, \hat{k}) \Gamma(\eta_k | r_k, s_k) \text{vMF}(\hat{\eta}_k | \hat{\eta}_k, \hat{k}) \Gamma(h_k | t_k, u_k) \prod_{n=1}^N q(z_n | \phi_n).
\]

Accordingly, the variational lower bound is

\[
\mathbb{E}_Q \left[ \log p(\{ \beta_k \}_{k=1}^K, \{ \eta_k \}_{k=1}^K, \{ x_n, y_n, z_n \}_{n=1}^N) \right] - \mathbb{E}_Q \left[ \log q(\{ \beta_k \}_{k=1}^K, \{ \eta_k \}_{k=1}^K, \{ z_n \}_{n=1}^N) \right]
\]

\[
= \mathbb{E}_Q \left[ \log p(\{ y_n \}_{n=1}^N | \{ x_n \}_{n=1}^N, \{ \beta_k \}_{k=1}^K) \right] + \mathbb{E}_Q \left[ \log p(\{ z_n \}_{n=1}^N | \{ x_n \}_{n=1}^N, \{ \eta_k \}_{k=1}^K) \right] + \mathbb{E}_Q \left[ \log q(\{ \beta_k \}_{k=1}^K) \right] - \mathbb{E}_Q \left[ \log q(\{ \eta_k \}_{k=1}^K) \right] - \mathbb{E}_Q \left[ \log q(\{ z_n \}_{n=1}^N) \right].
\]

where \( \mathbb{E}_Q \left[ \log p(\{ \beta_k \}_{k=1}^K) \right] \) and \( \mathbb{E}_Q \left[ \log p(\{ \eta_k \}_{k=1}^K) \right] \) can be lower bounded in a similar way as that in Eq. (3.19). \( \mathbb{E}_Q \left[ \log q(\{ \beta_k \}_{k=1}^K) \right] \) and \( \mathbb{E}_Q \left[ \log q(\{ \eta_k \}_{k=1}^K) \right] \) can be computed in a similar manner as that in Eq. (3.20). Next we discuss how to compute the remaining expectation terms.

**Compute** \( \mathbb{E}_Q \left[ \log p(\{ z_n \}_{n=1}^N | \{ \eta_k \}_{k=1}^K, \{ x_n \}_{n=1}^N) \right] \)\( \mathbb{E}_Q \left[ \log p(\{ z_n \}_{n=1}^N | \{ \eta_k \}_{k=1}^K, \{ x_n \}_{n=1}^N) \right] \) is defined as

\[
p(\{ z_n \}_{n=1}^N | \{ \eta_k \}_{k=1}^K, \{ x_n \}_{n=1}^N) = \prod_{n=1}^N p(z_n | x_n, \{ \eta_k \}_{k=1}^K)
\]

\[
= \prod_{n=1}^N \frac{\exp(\eta_k x_n)^{z_{nk}}}{\sum_{j=1}^K \exp(\eta_j x_n)}.
\]
\[
\log p(\{z_n\}_{n=1}^N|\{\eta_k\}_{k=1}^K, \{x_n\}_{n=1}^N) \text{ can be lower bounded as}
\]
\[
\log p(\{z_n\}_{n=1}^N|\{\eta_k\}_{k=1}^K, \{x_n\}_{n=1}^N) = \sum_{n=1}^N \sum_{k=1}^K z_{nk} \eta_k^T x_n - \log(\sum_{j=1}^K \exp(\eta_j^T x_n))
\]
\[
\geq \sum_{n=1}^N \sum_{k=1}^K z_{nk} \eta_k^T x_n - c_n - \sum_{j=1}^K \log(1 + \exp(\eta_j^T x_n - c_n))(\text{Using Inequality VI})
\]
\[
\geq \sum_{n=1}^N \sum_{k=1}^K z_{nk} \eta_k^T x_n - c_n - \sum_{j=1}^K \log(1 + e^{-d_{nj}}) - \frac{c_n - \eta_j^T x_n - d_{nj}}{2} - \frac{1/2 - g(d_{nj})}{2d_{nj}}((\eta_j^T x_n - c_n)^2 - d_{nj}^2)
\]
(Using Inequality VII)

The expectation of \(\log p(\{z_n\}_{n=1}^N|\{\eta_k\}_{k=1}^K, \{x_n\}_{n=1}^N)\) can be lower bounded as

\[
\mathbb{E}[\log p(\{z_n\}_{n=1}^N|\{\eta_k\}_{k=1}^K, \{x_n\}_{n=1}^N)] = A_p(\hat{\kappa}) \sum_{n=1}^N \sum_{k=1}^K \phi_{nk} \eta_k^T x_n - c_n - \sum_{j=1}^K \log(1 + e^{-d_{nj}}) - (c_n - A_p(\hat{\kappa}) \frac{\beta_j^T}{u_j} \eta_j^T x_n - d_{nj})/2
\]
\[
- \frac{1/2 - g(d_{nj})}{2d_{nj}}(\frac{\beta_j^T}{u_j} \mathbb{E}[\eta_j^T x_n x_n^T \eta_j] - 2c_n A_p(\hat{\kappa}) \frac{\beta_j^T}{u_j} \eta_j^T x_n + c_n^2 - d_{nj}^2),
\]
(3.26)

where

\[
\mathbb{E}[\eta_k^T x_n x_n^T \eta_k] = \mathbb{E}[\text{tr}(\eta_k^T x_n x_n^T \eta_k)] = \mathbb{E}[\text{tr}(x_n x_n^T \eta_k \eta_k^T)] = \text{tr}(x_n x_n^T \mathbb{E}[\eta_k \eta_k^T])
\]
\[
= \text{tr}(x_n x_n^T (\mathbb{E}[\eta_k]^T \mathbb{E}[\eta_k]) + \text{cov}(\eta_k))).
\]

\[
\text{Compute } \mathbb{E}[\log p(\{y_n\}_{n=1}^N|\{\beta_k\}_{k=1}^K, \{z_n\}_{n=1}^N)] = p(\{y_n\}_{n=1}^N|\{\beta_k\}_{k=1}^K, \{z_n\}_{n=1}^N) \text{ is defined as}
\]
\[
p(\{y_n\}_{n=1}^N|\{\beta_k\}_{k=1}^K, \{z_n\}_{n=1}^N) = \prod_{n=1}^N p(y_n|z_n, \{\beta_k\}_{k=1}^K)
\]
\[
= \prod_{n=1}^N \frac{1}{\prod_{k=1}^K [1 + \exp(-(2y_n - 1)\beta_k^T x_n)^{\varepsilon_{nk}}]}.
\]

log \(p(\{y_n\}_{n=1}^N|\{\beta_k\}_{k=1}^K, \{z_n\}_{n=1}^N)\) can be lower bounded by

\[
\log p(\{y_n\}_{n=1}^N|\{\beta_k\}_{k=1}^K, \{z_n\}_{n=1}^N) = - \sum_{n=1}^N \sum_{k=1}^K z_{nk} \log(1 + \exp(-(2y_n - 1)\beta_k^T x_n))
\]
\[
\geq \sum_{n=1}^N \sum_{k=1}^K z_{nk} [- \log(1 + e^{-\varepsilon_{nk}}) + ((2y_n - 1)\beta_k^T x_n - \varepsilon_{nk})/2 + 1/2 - g(\varepsilon_{nk})/(2\varepsilon_{nk})((\beta_k^T x_n)^2 - \varepsilon_{nk}^2)].
\]
(3.29)
\[
\mathbb{E}[\log p(\{y_n\}_{n=1}^N | \{\beta_k\}_{k=1}^K, \{z_n\}_{n=1}^N)] \text{ can be lower bounded by} \\
\geq \sum_{n=1}^N \sum_{k=1}^K \phi_{nk} \left[ -\log(1 + e^{-e_{nk}}) + \left( \frac{1}{2} \cdot \frac{1}{2} + \sigma(e_{nk}) \right) \frac{1}{2} \cdot \frac{1}{2} - \sigma(e_{nk}) \right]
\]
(Using Inequality VII)

\[ (3.30) \]

where \( \mathbb{E}[\hat{\beta}_k^T x_n x_n^T \hat{\beta}_k] \) can be computed in a similar way as that in Eq.\( (3.27) \).

**Compute** \( \mathbb{E}[\log q(z_i)] \)

\[
\mathbb{E}[\log q(z_i)] = \sum_{k=1}^K \phi_{ik} \log \phi_{ik}. 
\]
(3.31)

In the end, we can get a lower bound of the variational lower bound, then learn all the parameters by optimizing the lower bound via coordinate ascent. In each iteration, we select one parameter \( x \) and fix all other parameters, which leads to a sub-problem defined over \( x \). Then we optimize the sub-problem w.r.t \( x \). For some parameters, the optimal solution of the sub-problem is in closed form. If not the case, we optimize \( x \) using the gradient ascent method. This process iterates until convergence. We omit the detailed derivation here since it only involves basic algebra and calculus, which can be done straightforwardly.

**MCMC Sampling for Type-I MAP**

The type-I MAP prior is not amenable for variational inference. Instead, we use an alternative approximation inference method — Markov chain Monte Carlo (MCMC) \[127\] sampling, which draws samples directly from the true posterior distribution and uses the samples to represent the posterior. Specifically we choose the Metropolis-Hastings (MH) algorithm \[127\] which generates samples from an adaptive proposal distribution, computes acceptance probabilities based on the unnormalized true posterior, and uses the acceptance probabilities to decide whether a sample should be accepted or rejected. The most commonly used proposal distribution is based on the random walk: the newly proposed sample \( t + 1 \) comes from a random perturbation around the previous sample \( t \). For the directional variables \( \{\tilde{a}_i\}_{i=1}^K \) and magnitude variables \( \{g_i\}_{i=1}^K \), we define the proposal distributions to be a von Mises-Fisher distribution and a normal distribution respectively:

\[
q(\tilde{a}_{i}^{(t+1)} | \tilde{a}_{i}^{(t)}) = C_p(\kappa) \exp \left( \frac{\kappa}{2} \tilde{a}_{i}^{(t+1)} \cdot \tilde{a}_{i}^{(t)} \right) \]
\[
q(g_{i}^{(t+1)} | g_{i}^{(t)}) = \frac{1}{\sigma \sqrt{2\pi}} \exp \left\{ -\frac{(g_{i}^{(t+1)} - g_{i}^{(t)})^2}{2\sigma^2} \right\}. 
\]
(3.32)

\( g_{i}^{(t+1)} \) is required to be positive, but the Gaussian distribution may generate non-positive samples. To address this problem, we adopt a truncated sampler \[360\] which repeatedly draws samples until a positive value is obtained. Under such a truncated sampling scheme, the MH acceptance ratio needs to be modified accordingly.

The MAP is parameterized by several deterministic parameters including \( \kappa, \mu_0, \alpha_1, \alpha_2 \). Among them, we tune \( \kappa \) manually via cross validation and learn the others via an expectation
maximization (EM) framework. Let \( x \) denote the observed data, \( z \) denote all random variables, and \( \theta \) denote the deterministic parameters \( \{\mu_0, \alpha_1, \alpha_2\} \). EM is an algorithm aiming at learning \( \theta \) by maximizing the log-likelihood \( p(x; \theta) \) of data. It iteratively performs two steps until convergence. In the E step, \( \theta \) is fixed and the posterior \( p(z|x) \) is inferred using the aforementioned Metropolis-Hastings algorithm. In the M step, \( \theta \) is learned by optimizing a lower bound of the log-likelihood \( \mathbb{E}_{p(x|z)}[\log p(x, z; \theta)] \), where the expectation is computed w.r.t the posterior \( p(z|x) \) inferred in the E step. The parameters \( \kappa \) and \( \sigma \) in the proposal distributions are tuned manually.

**Algorithm for Posterior-Regularized BMEM**

In this section, we present the algorithm details of the posterior-regularized BMEM. Recall that the problem is

\[
\sup_{q(B, H, z)} \mathbb{E}_{q(B, H, z)}[\log p(\{y_i\}_{i=1}^{N}, z|B, H)p(B, H)] - \mathbb{E}_{q(B, H, z)}[\log q(B, H, z)] + \lambda_1 \Omega(\{\mathbb{E}_{q(\beta_k)}[\tilde{\beta}_k]\}_{k=1}^{K}) + \lambda_2 \Omega(\{\mathbb{E}_{q(\eta_k)}[\tilde{\eta}_k]\}_{k=1}^{K}),
\]

(3.33)

where \( B = \{\beta_k\}_{k=1}^{K}, H = \{\eta_k\}_{k=1}^{K}, \) and \( z = \{z_i\}_{i=1}^{N} \) are latent variables and the post-data distribution over them is defined as \( q(B, H, z) = q(B)q(H)q(z) \). For computational tractability, we define \( q(B) \) and \( q(H) \) to be: \( q(B) = \prod_{k=1}^{K} q(\beta_k)q(g_k) \) and \( q(H) = \prod_{k=1}^{K} q(\eta_k)q(h_k) \) where \( q(\beta_k), q(\eta_k) \) are von-Mises Fisher distributions and \( q(g_k), q(h_k) \) are gamma distributions, and define \( q(z) \) to be multinomial distributions: \( q(z) = \prod_{i=1}^{N} q(z_i|\phi_i) \) where \( \phi_i \) is a multinomial vector. The priors over \( B \) and \( H \) are specified to be: \( \pi(B) = \prod_{k=1}^{K} p(\beta_k)p(g_k) \) and \( \pi(H) = \prod_{k=1}^{K} p(\eta_k)p(h_k) \) where \( p(\beta_k), p(\eta_k) \) are von-Mises Fisher distributions and \( p(g_k), p(h_k) \) are gamma distributions.

The objective in Eq.(3.33) can be further written as

\[
\mathbb{E}_{q(B, H, z)}[\log p(\{y_i\}_{i=1}^{N}, z|B, H)p(B, H)] - \mathbb{E}_{q(B, H, z)}[\log q(B, H, z)] + \lambda_1 \Omega(\{\mathbb{E}_{q(\beta_k)}[\tilde{\beta}_k]\}_{k=1}^{K}) + \lambda_2 \Omega(\{\mathbb{E}_{q(\eta_k)}[\tilde{\eta}_k]\}_{k=1}^{K})
\]

(3.34)

Among these expectation terms, \( \mathbb{E}_{q(B, z)}[\log p(\{y_i\}_{i=1}^{N}, z|B)] \) can be computed via Eq.(3.28-3.30) and \( \mathbb{E}_{q(H, z)}[\log p(z|H)] \) can be computed via Eq. (3.24-3.27). \( \mathbb{E}_{q(H)}[\log \pi(H)], \mathbb{E}_{q(B)}[\log \pi(B)], \mathbb{E}_{q(B)}[\log q(B)], \mathbb{E}_{q(H)}[\log q(H)] \) can be computed in a way similar to that in Eq.(3.20). \( \mathbb{E}_{q(z)}[\log q(z)] \) can be computed via Eq.(3.31). Given all these expectations, we can get an analytical expression of the objective in Eq.(3.33) and learn the parameters by optimizing this objective. Regarding how to optimize the mutual angular regularizers \( \Omega(\{\mathbb{E}_{q(\beta_k)}[\tilde{\beta}_k]\}_{k=1}^{K}) \) and \( \Omega(\{\mathbb{E}_{q(\eta_k)}[\tilde{\eta}_k]\}_{k=1}^{K}) \), please refer to [369] for details.
3.2 Diversity-promoting Learning of Bayesian Nonparametric Models

In the last section, we study how to promote diversity among a finite number of components in parametric models. In this section, we investigate how to achieve this goal in Bayesian nonparametric (BNP) models where the component number is infinite in principle, by extending the mutual angular process (Section 3.1.1) to an infinite MAP (IMAP).

Different from parametric models where the component number is set to a finite value and does not change throughout the entire execution of algorithm, in BNP models the number of components is unlimited and can reach infinite in principle. As more data accumulates, new components are dynamically added. Compared with parametric models, BNP models possess the following advantages: (1) they are highly flexible and adaptive: if new data cannot be well modeled by existing components, new components are automatically invoked; (2) in BNP models, the “best” number of components is determined according to the fitness to data, rather than being manually set which is a challenging task even for domain experts.

A BNP model consists of an infinite number of components, each parameterized by a vector. For example, in the Dirichlet process Gaussian mixture model (DP-GMM) [48, 286], the components are called clusters, each parameterized with a Gaussian mean vector. In the Indian buffet process latent feature model (IBP-LFM) [143], the components are called features, each parameterized by a weight vector. Given infinitely many components, BNP models design some proper mechanism to select one or a finite subset of components to model each observed data example. For example, in DP-GMM, a Chinese restaurant process (CRP) [18] is designed to assign each data example to one of the infinitely many clusters. In IBP-LFM, an Indian buffet process (IBP) [143] is utilized to select a finite set of features from the infinite feature pool to reconstruct each data example. A BNP model typically consists of two priors. One is a base distribution from which the parameter vectors of components are drawn. The other is a stochastic process – such as CRP and IBP – which designates how to select components to model data. The IMAP prior proposed in this section belongs to the first regime. It is commonly assumed that the parameter vectors of components are independently drawn from the same base distribution. For example, in both DP-GMM and IBP-LFM, the mean vectors and weight vectors are independently drawn from a Gaussian distribution. In our work, we aim at designing a prior that encourages the component vectors to be mutually different and “diverse”, under which the component vectors are not independent anymore, which presents great challenges for posterior inference.

To “diversify” BNP models, we extend the mutual angular process to an infinite MAP (IMAP) that encourages infinitely many components to have large angles. In this prior, the components are mutually dependent, which is challenging to perform posterior inference. We develop an efficient posterior inference algorithm based on slice sampling [329] and Riemann manifold Hamiltonian Monte Carlo [129]. We apply the IMAP to induce diversity in the infinite latent feature model (ILFM) [142] and experiments on various datasets demonstrate that the IMAP is able to (1) achieve better performance with fewer components, (2) better capture infrequent patterns, and (3) reduce overfitting.
Related works Several works studied how to promote diversity in Bayesian nonparametric models. In Xu et al. [386], the component number is treated as a random variable. To obtain a unknown number of “diverse” components, they first sample a component number $K$ from a prior, then generate $K$ “diverse” component vectors using DPP. They apply this approach to “diversify” the cluster centers in the Dirichlet process mixture model (DPMM) [111] and feature allocation vectors in the infinite latent feature model (ILFM). However, it is unclear how to apply this method to “diversify” the weight vectors of the latent features in ILFM when it is coupled with the Indian buffet process [142]. They use a reversible jump MCMC algorithm for posterior inference, which is not scalable for high-dimensional problems. Xie and Xu [366] proposed a Bayesian repulsive Gaussian mixture model where the cluster centers in DPMM are encouraged to be “diverse”. Their approach for inducing diversity is not applicable to the ILFM model.

3.2.1 Infinite Mutual Angular Process

In this section, we aim at designing a Bayesian nonparametric prior to promote diversity among infinitely many components, which is very challenging and presumably much more difficult than inducing diversity among a finite set of components. Taking the determinantal point process (DPP) [204] as example, in the finite case, a kernel matrix is computed over the component vectors and the determinant of this matrix is employed as a measure of diversity. In the infinite case, the kernel matrix would have an infinite number of rows and columns, whose determinant cannot be defined.

One possible reason that renders DPP fails to be extended to the infinite case is that DPP measures the diversity of components in a batch mode: the similarity between each pair of components is measured in one shot, then these similarity scores is aggregated into a diversity score (via the determinant). In the infinite case, the number of similarity scores is infinite. How to aggregate infinitely many scores into one single score and meanwhile guarantee the prior distribution derived thereafter is proper (integrating to one) is a highly challenging issue. As discussed in Section 3.1.1, one way to solve this problem is to bring in diversity in a sequential mode: each time we add one component to the component pool and encourage the new component to be different from the existing ones. The number of existing components is always finite, hence the (dis)similarity measurement always occurs between one component (the new one) and a finite set of components (the existing ones), which stands in contrast with the batch mode where each component is measured against infinitely many components. This adding process can be repeated infinitely many times, providing a natural way to accommodate an infinite number of components into the BNP models.

Similar to the mutual angular process in Section 3.1.1, we use a Bayesian network (BN) [198] to model the adding process of components and design local probability at each node of the BN to encourage the components to have large mutual angles. Repeating the adding process infinitely many times, we end up with a prior that encourages an infinite number of components to have large mutual angles

\[
p(\{\tilde{a}_i\}_{i=1}^\infty) = p(\tilde{a}_1) \prod_{i=2}^\infty p(\tilde{a}_i|\text{pa}(\tilde{a}_i)), \tag{3.35}
\]
where \( \{ \tilde{a}_k \}^\infty_{k=1} \) are the directional vectors of components. The factorization theorem \cite{198} of Bayesian networks ensures that \( p(\{ \tilde{a}_k \}^\infty_{k=1}) \) integrates to one. The magnitudes \( \{ g_i \}^\infty_{i=1} \) do not affect angles, which can be generated independently from a gamma distribution.

The generative process of components \( \{ a_i \}^\infty_{i=1} \) can be summarized as follows:

- Sample \( \bar{a}_1 \sim \text{vMF}(\mu_0, \kappa) \)
- For \( i = 2, \ldots, \infty \), sample \( \bar{a}_i \sim \text{vMF}( - \frac{\sum_{j=1}^{i-1} \bar{a}_j}{\| \sum_{j=1}^{i-1} \bar{a}_j \|_2}, \kappa) \)
- For \( i = 1, \ldots, \infty \), sample \( g_i \sim \text{Gamma}(\alpha_1, \alpha_2) \)
- For \( i = 1, \ldots, \infty \), \( a_i = \tilde{a}_i g_i \)

The probability distribution of \( \{ a_i \}^\infty_{i=1} \) can be written as

\[
p(\{ a_i \}^\infty_{i=1}) = C_G(\kappa) \exp(\kappa \mu_0^T \tilde{a}_1) \prod_{i=2}^\infty C_G(\kappa) \exp \left( \kappa \left( - \frac{\sum_{j=1}^{i-1} \bar{a}_j}{\| \sum_{j=1}^{i-1} \bar{a}_j \|_2} \right) ^T \tilde{a}_i \right) \prod_{i=1}^\infty \frac{\alpha_2 g_i^{\alpha_1 - 1} e^{-g_i \alpha_2}}{\Gamma(\alpha_1)}.
\]

(3.36)

### 3.2.2 Case Study: Infinite Latent Feature Model

In this section, using the infinite latent feature model (ILFM) \cite{142} as a study case, we show how to promote diversity among the components therein with the IMAP prior. Given a set of data examples \( X = \{ x_n \}^N_{n=1} \) where \( x_n \in \mathbb{R}^D \), ILFM aims at invoking a finite subset of features from an infinite feature collection \( A = \{ a_k \}^\infty_{k=1} \) to construct these data examples. Each feature (which is a component in this BNP model) is parameterized by a vector \( a_k \in \mathbb{R}^D \). For each data example \( x_n \), a subset of features are selected to construct it. The selection is denoted by a binary vector \( z_n \in \{ 0, 1 \}^\infty \) where \( z_{nk} = 1 \) denotes the \( k \)-th feature is invoked to construct the \( n \)-th example and \( z_{nk} = 0 \) otherwise. Given the parameter vectors of features \( \{ a_k \}^\infty_{k=1} \) and the selection vector \( z_n \), the example \( x_n \) can be represented as: \( x_n \sim \mathcal{N}(\sum_{k=1}^\infty z_{nk} a_k, \sigma^2 I) \). The binary selection vectors \( Z = \{ z_n \}^N_{n=1} \) can be either drawn from an Indian buffet process (IBP) \cite{143} or a stick-breaking construction \cite{329}. Let \( \mu_k \) be the prior probability that feature \( k \) is present in a data example and the features are permuted such that their prior probabilities are in a decreasing ordering: \( \mu(1) > \mu(2) > \ldots \). According to the stick-breaking construction, these prior probabilities can be generated in the following way: \( \nu_k \sim \text{Beta}(\alpha, 1), \mu(k) = \nu_k \mu(k-1) = \prod_{l=1}^k \nu_l \). Given \( \mu(k) \), the binary indicator \( z_{nk} \) is generated as \( z_{nk} | \mu(k) \sim \text{Bernoulli}(\mu(k)) \). To reduce the redundancy among the features, we impose the IMAP over their parameter vectors \( A \) to encourage them to be mutually different, which results in an IMAP-LFM model.

### 3.2.3 A Sampling-based Inference Algorithm

In this section, we develop a sampling algorithm to infer the posteriors of \( A \) and \( Z \) in the IMAP-LFM model. Two major challenges need to be addressed. First, the prior over \( A \) is not conjugate to the likelihood function \( p(x) \). Second, the parameter vectors \( A \) are usually high-dimensional, rendering slow mixing. To address the first challenge, we adopt a slicing sampling algorithm \cite{329}. This algorithm introduces an auxiliary slice variable \( s | Z, \mu(1: \infty) \sim \text{uniform}[0, \mu^*] \), where \( \mu^* = \min \{ 1, \min_{k : \exists n, z_{nk} = 1} \mu_k \} \) is the prior probability of the last active feature. A feature \( k \) is active
Algorithm 2: A manifold Hamiltonian Monte Carlo algorithm for sampling $\tilde{\alpha}$.

1. $v \sim \mathcal{N}(0, I)
2. v \leftarrow v - (I - \tilde{a}\tilde{a}^\top)v
3. h \leftarrow \log p(\tilde{a}|\text{rest}) - \frac{1}{2}v^\top v
4. \tilde{a}^* \leftarrow \tilde{a}
5. \textbf{for } \tau = 1, \cdots, T \textbf{ do}
6. \quad v \leftarrow v + \frac{1}{2}\nabla_{\tilde{a}^*} \log p(\tilde{a}^*|\text{rest})
7. \quad v \leftarrow v - \tilde{a}\tilde{a}^\top v
8. \quad \tilde{a}^* \leftarrow \cos(\epsilon\|v\|_2)\tilde{a}^* + \|v\|_2^{-1} \sin(\epsilon\|v\|_2)v
9. \quad v \leftarrow -\|v\|_2 \sin(\epsilon\|v\|_2)\tilde{a}^* + \cos(\epsilon\|v\|_2)v
10. \quad v \leftarrow v + \frac{1}{2}\nabla_{\tilde{a}^*} \log p(\tilde{a}^*|\text{rest})
11. \quad v \leftarrow v - \tilde{a}\tilde{a}^\top v
12. \textbf{end for}
13. h^* \leftarrow \log p(\tilde{a}^*|\text{rest}) - \frac{1}{2}v^\top v
14. u \sim \text{uniform}(0, 1)
15. \textbf{if } u < \exp(h^* - h) \textbf{ then}
16. \quad \tilde{a} \leftarrow \tilde{a}^*
17. \textbf{end if}

if there exists an example $n$ such that $z_{nk} = 1$ and is inactive otherwise. In the sequel, we discuss the sampling of other variables.

Sample new features Let $K^*$ be the maximal feature index with $\mu(K^*) > s$ and $K^+$ be the index such that all active features have index $k < K^+$ ($K^+$ itself would be an inactive feature). If the new value of $s$ makes $K^* \geq K^+$, then we draw $K^* - K^+ + 1$ new (inactive) features, including the parameter vectors and prior probabilities. The prior probabilities $\{\mu(k)\}$ are drawn sequentially from $p(\mu(k)|\mu(k-1)) \propto \exp(\alpha \sum_{n=1}^{N} \frac{1}{p}(1 - \mu(k))^n)\mu(k-1)^{-1}(1 - \mu(k))^{N}(0 \leq \mu(k) \leq \mu(k-1))$ using the adaptive rejection sampling (ARS) [128] method. The parameter vectors are drawn sequentially from

$$p(\tilde{a}_k|\{\tilde{a}_j\}_{j=1}^{k-1}) = p(\tilde{a}_k|\{\tilde{a}_j\}_{j=1}^{k-1})p(g_k) = C_p(\kappa) \exp \left( -\frac{\sum_{j=1}^{k-1} \tilde{a}_j}{\sum_{j=1}^{k} \tilde{a}_j/2} \right) \frac{\tilde{a}_2^{\alpha_1} \tilde{a}_1^{\alpha_1-1} e^{-\tilde{a}_2/2}}{\Gamma(\alpha_1)} \right),$$

where we draw $\tilde{a}_k$ from $p(\tilde{a}_k|\{\tilde{a}_j\}_{j=1}^{k-1})$ which is a von Mises-Fisher distribution and draw $g_k$ from a gamma distribution, then multiply $\tilde{a}_k$ and $g_k$ together since they are independent. For each new feature $k$, the corresponding binary selection variables $z_{:,k}$ are initialized to zero.

Sample existing $\mu(k)$ ($1 \leq k \leq K^+ - 1$) We sample $\mu(k)$ from $p(\mu(k)|\text{rest}) \propto \mu(k)^{-1}(1 - \mu(k))^{N-m_k} \mu(k+1) \leq \mu(k) \leq \mu(k-1)$, where $m_k = \sum_{n=1}^{N} z_{nk}$.

Sample $z_{nk}$ ($1 \leq n \leq N, 1 \leq k \leq K^*$) Given $s$, we only need to sample $z_{nk}$ for $k \leq K^*$ from $p(z_{nk} = 1|\text{rest}) \propto \mu(k)^{s}p(x_n|z_{n,-k}, z_{nk} = 1, \{a_j\}_{j=1}^{K^+}),$ where $p(x_n|z_{n,-k}, z_{nk} = 1, \{a_j\}_{j=1}^{K^+}) = \cdots$
the Riemann manifold, which in our case is a unit sphere. After a transformation of the coordi-

\[ \mathcal{N}(x_n|a_k + \sum_{j \neq k}^{K^+} z_{nj}a_j, \sigma I) \]

and \( z_{n,-k} \) denotes all other elements in \( z \) except the \( k \)-th one and \( a_j = a_j g_j \).

**Sample \( a_k (k = 1, \cdots, K^+) \)** We draw \( a_k = \tilde{a}_k g_k \) from the following conditional probability

\[ p(\tilde{a}_k g_k | \text{rest}) \propto p(\tilde{a}_k g_k | \{a_j\}_{j \neq k}^{K^+}) \prod_{n=1}^{N} p(x_n | z_{n:1:K^+}, \{a_j\}_{j \neq k}^{K^+}, \tilde{a}_k g_k), \]

(3.38)

where \( p(\tilde{a}_k g_k | \{a_j\}_{j \neq k}^{K^+}) \propto p(\tilde{a}_k g_k | \{a_j\}_{i=1}^{k-1}) \prod_{j=k+1}^{K^+} p(a_j | \{a_i\}_{i \neq k}^{j-1}, \tilde{a}_k g_k) \) and \( p(x_n | z_{n:1:K^+}, \{a_j\}_{j \neq k}^{K^+}, \tilde{a}_k g_k) = \mathcal{N}(x_n | \tilde{a}_k g_k + \sum_{j \neq k}^{K^+} z_{nj}a_j, \sigma I) \). In the vanilla IBP latent feature model [143], the prior over \( a_k \) is a Gaussian distribution, which is conjugate to the Gaussian likelihood function. In that case, the posterior \( p(a_k | \text{rest}) \) is again a Gaussian, from which samples can be easily drawn. But in Eq. (3.38), the posterior does not have a closed form expression since the prior \( p(\tilde{a}_k g_k | \{a_j\}_{j \neq k}^{K^+}) \) is no longer a conjugate prior, making the sampling very challenging.

We sample \( \tilde{a}_k \) and \( g_k \) separately. \( g_k \) can be efficiently sampled using the Metropolis-Hastings (MH) [152] algorithm. For \( \tilde{a}_k \) which is a random vector, the sampling is much more difficult. The random walk based MH algorithm suffers slow mixing when the dimension of \( \til|a_k \) is large (which is typically the case). In addition, \( \tilde{a}_k \) lies on a unit sphere. The sampling algorithm should preserve this geometric constraint. To address these two issues, we study a Riemann manifold Hamiltonian Monte Carlo (RM-HMC) method [59, 129]. HMC leverages the Hamiltonian dynamics to produce distant proposals for the Metropolis-Hastings algorithm, enabling a faster exploration of the state space and faster mixing. The RM-HMC algorithm introduces an auxiliary vector \( v \) and defines a Hamiltonian function \( H(\tilde{a}_k, v) = -\log p(\tilde{a}_k | \text{rest}) + \log |G(\tilde{a}_k)| + \frac{1}{2} v^T G(\tilde{a}_k)^{-1} v \), where \( G \) is the metric tensor associated with the Riemann manifold, which in our case is a unit sphere. After a transformation of the coordi-

<table>
<thead>
<tr>
<th>Dataset</th>
<th>IBP-LFM</th>
<th>PYP-LFM</th>
<th>IMAP-LFM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yale</td>
<td>447±7</td>
<td>432±3</td>
<td>419±4</td>
</tr>
<tr>
<td>Block (×10^{-2})</td>
<td>6.3±0.4</td>
<td>5.8±0.1</td>
<td>4.4±0.2</td>
</tr>
<tr>
<td>AR</td>
<td>926±4</td>
<td>939±11</td>
<td>871±7</td>
</tr>
<tr>
<td>EEG (+2.1 × 10^6)</td>
<td>5382±34</td>
<td>3731±15</td>
<td>575±21</td>
</tr>
<tr>
<td>Piano (×10^{-4})</td>
<td>5.3±0.1</td>
<td>5.7±0.2</td>
<td>4.2±0.2</td>
</tr>
</tbody>
</table>

Table 3.6: L2 test error.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>IBP-LFM</th>
<th>PYP-LFM</th>
<th>IMAP-LFM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yale</td>
<td>-16.4±0.3</td>
<td>-14.9±0.4</td>
<td>-12.7±0.1</td>
</tr>
<tr>
<td>Block-Image</td>
<td>-2.1±0.2</td>
<td>-1.8±0.1</td>
<td>-1.4±0.1</td>
</tr>
<tr>
<td>AR</td>
<td>-13.9±0.3</td>
<td>-14.6±0.7</td>
<td>-8.5±0.4</td>
</tr>
<tr>
<td>EEG</td>
<td>-14133±54</td>
<td>-12893±73</td>
<td>-9735±32</td>
</tr>
<tr>
<td>Piano</td>
<td>-6.8±0.6</td>
<td>-6.9±0.2</td>
<td>-4.2±0.5</td>
</tr>
</tbody>
</table>

Table 3.7: Test log-likelihood.
nate system, $H(\tilde{\alpha}_k, v)$ can be re-written as

$$H(\tilde{\alpha}_k, v) = -\log p(\tilde{\alpha}|rest) + \frac{1}{2} v^T v. \quad (3.39)$$

$p(\tilde{\alpha}_k|\text{rest})$ needs not to be normalized and $\log p(\tilde{\alpha}_k|\text{rest}) \propto \kappa(-\frac{\sum_{j=1}^{k-1} \tilde{\alpha}_j}{\sum_{j=1}^{k-1} \|	ilde{\alpha}_j\|_2})^T \tilde{\alpha}_k + \sum_{j=k+1}^{K^+} \kappa(-\frac{\sum_{i=1}^{k} \tilde{\alpha}_i + \tilde{\alpha}_k g_k}{\sum_{i=1}^{k} \tilde{\alpha}_i + \|	ilde{\alpha}_k g_k\|_2})^T \tilde{\alpha}_j + \sum_{n=1}^{N} \frac{1}{\sigma} (\tilde{x}_n - \sum_{j \neq k} \tilde{z}_{nj} a_j)^T \tilde{\alpha}_k g_k - \frac{1}{2\sigma^2} \|	ilde{\alpha}_k g_k\|^2_2$. A new sample of $\tilde{\alpha}_k$ can be generated by approximately solving a system of differential equations characterizing the Hamiltonian dynamics on the manifold [129]. Following [59], we solve this problem based upon geodesic flow, which is shown in Line 6-11 in Algorithm [2]. Line 6 performs an update of $v$ according to the Hamiltonian dynamics, where $\nabla_{\tilde{\alpha}^*} \log p(\tilde{\alpha}^*|rest)$ denotes the gradient of $\log p(\tilde{\alpha}^*|rest)$ w.r.t $\tilde{\alpha}^*$. Line 7 performs the transformation of the coordinate system. Line 8-9 calculate the geodesic flow on the unit sphere. Line 10-11 repeat the update of $v$ as done in Line 6-7. These procedures are repeated $T$ times to generate a new sample $\tilde{\alpha}^*$, which then goes through an acceptance/rejection procedure (Line 3, 13-17).

### 3.2.4 Evaluation

We evaluated the effectiveness of the IMAP prior in alleviate overfitting, reducing model size without sacrificing modeling power, and capturing infrequent patterns, on a wide range of datasets.

#### Datasets

We used ten datasets from different domains including texts, images, audio, and EEG signals: Yale [125], Block-Images [361], AR [249], EEG [165], Piano [276], Reuters [10], TDT [4], 20-News [1], 15-Scenes [212], and Caltech-101 [110]. The first five datasets were represented as raw data without feature extraction. The documents in Reuters, TDT, and 20-News were represented with bag-of-words vectors, weighted using tf-idf. The images in 15-

<table>
<thead>
<tr>
<th>Dataset</th>
<th>IBP-LFM Mean</th>
<th>IBP-LFM Stdev</th>
<th>PYP-LFM Mean</th>
<th>PYP-LFM Stdev</th>
<th>IMAP-LFM Mean</th>
<th>IMAP-LFM Stdev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reuters</td>
<td>45.4 ± 0.3</td>
<td>43.1 ± 0.4</td>
<td>48.2 ± 0.6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TDT</td>
<td>48.3 ± 0.7</td>
<td>47.5 ± 0.3</td>
<td>53.2 ± 0.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20-News</td>
<td>21.5 ± 1.1</td>
<td>23.7 ± 0.2</td>
<td>25.2 ± 0.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15-Scenes</td>
<td>22.7 ± 0.2</td>
<td>21.9 ± 0.4</td>
<td>25.3 ± 0.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Caltech-101</td>
<td>11.6 ± 0.4</td>
<td>12.1 ± 0.1</td>
<td>14.7 ± 0.2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.8: Clustering accuracy (%).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>IBP-LFM Mean</th>
<th>IBP-LFM Stdev</th>
<th>PYP-LFM Mean</th>
<th>PYP-LFM Stdev</th>
<th>IMAP-LFM Mean</th>
<th>IMAP-LFM Stdev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reuters</td>
<td>41.7 ± 0.5</td>
<td>38.6 ± 0.2</td>
<td>45.4 ± 0.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TDT</td>
<td>44.2 ± 0.1</td>
<td>46.7 ± 0.3</td>
<td>49.6 ± 0.6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20-News</td>
<td>38.9 ± 0.8</td>
<td>35.2 ± 0.5</td>
<td>44.6 ± 0.9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15-Scenes</td>
<td>42.1 ± 0.7</td>
<td>44.9 ± 0.8</td>
<td>47.5 ± 0.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Caltech-101</td>
<td>34.2 ± 0.4</td>
<td>36.8 ± 0.4</td>
<td>40.3 ± 0.3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.9: Normalized mutual information (%).
Table 3.10: Log-likelihood on the training and testing sets of Yale and AR.

<table>
<thead>
<tr>
<th>Method</th>
<th>Yale</th>
<th>AR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train</td>
<td>Test</td>
</tr>
<tr>
<td>IBP-LFM</td>
<td>-9.2</td>
<td>-16.4</td>
</tr>
<tr>
<td>IMAP-LFM</td>
<td>-9.8</td>
<td>-12.7</td>
</tr>
</tbody>
</table>

Table 3.11: Number of latent features and L2 test errors.

<table>
<thead>
<tr>
<th></th>
<th>Yale</th>
<th>AR</th>
</tr>
</thead>
<tbody>
<tr>
<td># Features</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IBP-LFM</td>
<td>162</td>
<td>175</td>
</tr>
<tr>
<td>IMAP-LFM</td>
<td>165</td>
<td>176</td>
</tr>
<tr>
<td>L2 Error</td>
<td>445</td>
<td>922</td>
</tr>
<tr>
<td></td>
<td>419</td>
<td>871</td>
</tr>
</tbody>
</table>

Scenes [212] and Caltech-101 [110] were represented with visual bag-of-words vectors based on the SIFT [237] features. The train/test split of each dataset was 70%/30%. The results were averaged over five random splits.

Experimental setup For each dataset, we used the IMAP-LFM model to learn the latent features $\mathcal{A}$ on the training set, then used $\mathcal{A}$ to reconstruct the test data. The reconstruction performance was measured using log-likelihood (the larger, the better). Meanwhile, we used $\mathcal{A}$ to infer the representations $\mathcal{Z}$ of test data and performed data clustering on $\mathcal{Z}$. Clustering performance was measured using accuracy and normalized mutual information (NMI) (the higher, the better) [61]. We compared with two baselines: the Indian buffet process LFM (IBP-LFM) [142] and the Pitman-Yor process LFM (PYP-LFM) [327]. In IBP-LFM, posterior inference was performed with the slice sampling algorithm [329]. For PYP-LFM inference, we used the algorithm proposed in [327]. To attain a better local optimal, for each experiment run, we performed 5 random restarts and chose the best solution. The initialization for all models were the same. Following [105], all datasets were centered to have zero mean. We used grid search and 5-fold cross-validation to tune the hyper-parameters for all methods. In our method, $\kappa$ in Eq. (3.36) was set to 1. $\sigma^2$ was set to $0.25\hat{\sigma}$ where $\hat{\sigma}$ is the standard deviation of data across all dimensions. $\alpha$ was set to 2. The hyper-parameters of the RM-HMC algorithm followed those in [59].

Results We first verify whether the features learned by IMAP-LFM are diverse. We computed the average of the cosine similarities between each pair of latent features. Features with smaller average cosine similarity (ACS) are considered to be more diverse. On the Reuters dataset, the ACS for IBP-LFM, PYP-LFM, and IMAP-LFM are 0.72, 0.69, 0.61, respectively. IMAP-LFM also achieves smaller ACS on other datasets. This shows that IMAP effectively promotes diversity.

Table 3.6 and 3.7 present the L2 error and log-likelihood (mean $\pm$ standard error) on the test set of the first five datasets. The standard errors were computed from the 5 random train/test splits for each dataset. As can be seen, IMAP-LFM achieves much lower L2 error and higher log-likelihood than IBP-LFM and PYP-LFM. Table 3.8 and 3.9 show the clustering accuracy and NMI (mean $\pm$ standard error) on the last 5 datasets which have class labels. IMAP-LFM
Table 3.12: Number of features.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>IBP-LFM</th>
<th>PYP-LFM</th>
<th>IMAP-LFM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yale</td>
<td>201±5</td>
<td>220±8</td>
<td>165±4</td>
</tr>
<tr>
<td>Block-Image</td>
<td>8±2</td>
<td>9±4</td>
<td>11±4</td>
</tr>
<tr>
<td>AR</td>
<td>257±11</td>
<td>193±5</td>
<td>176±8</td>
</tr>
<tr>
<td>EEG</td>
<td>14±2</td>
<td>9±2</td>
<td>12±1</td>
</tr>
<tr>
<td>Piano</td>
<td>37±4</td>
<td>34±6</td>
<td>28±3</td>
</tr>
<tr>
<td>Reuters</td>
<td>354±12</td>
<td>326±5</td>
<td>294±7</td>
</tr>
<tr>
<td>TDT</td>
<td>297±6</td>
<td>311±9</td>
<td>274±3</td>
</tr>
<tr>
<td>20-News</td>
<td>442±8</td>
<td>408±3</td>
<td>369±5</td>
</tr>
<tr>
<td>15-Scenes</td>
<td>192±3</td>
<td>218±5</td>
<td>171±8</td>
</tr>
<tr>
<td>Caltech-101</td>
<td>127±7</td>
<td>113±6</td>
<td>96±6</td>
</tr>
</tbody>
</table>

outperforms the two baseline methods with a large margin.

Regarding why IMAP-LFM outperforms the baselines, we conjecture the reasons are two-fold. First, IMAP is more capable of alleviating overfitting. To verify this, we show the log-likelihood on the training and testing sets of Yale and AR in Table 3.10. Compared with IBP-LFM, IMAP-LFM achieves lower training log-likelihood and higher test log-likelihood. In IMAP-LFM, the gap between training and testing log-likelihood is smaller than that in IBP-LFM. This demonstrates that IMAP can better reduce overfitting. In both IBP-LFM and PYP-LFM, the weight vectors of latent features are drawn independently from a Gaussian distribution, which is unable to characterize the relations among features. In contrast, IMAP-LFM imposes a structure over the features, encouraging them to be “diverse” and less redundant. This diversity-biased structural constraint reduces model complexity of LFM, thus alleviating overfitting on the training data and achieving better reconstruction of the test data. Second, “diversified” features presumably have higher representational power and are able to capture richer information and subtle aspects of data, thus achieving a better modeling effect.

The L2 error and log-likelihood depend on two major factors: (1) model size, which is proportional to the number of latent features; (2) the level of “diversity” among the latent features. To clearly see the effect of (2), we removed the impact of (1) by tuning the hyperparameters of IBP-LFM and IMAP-LFM so that they have roughly the same feature number. Table 3.11 shows the number of features learned on the training set and the L2 error on the test set of Yale and AR. IMAP-LFM has almost the same feature number as IBP-LFM, but achieves significantly lower L2 error on the test set. This indicates that diversified features are better than non-diversified features.

Table 3.12 shows the number of features (mean±standard error) obtained by each model when the algorithm converges. Analyzing Table 3.6-3.9 and 3.12 simultaneously, we see that IMAP-LFM uses much fewer features to achieve better performance than the baselines. For instance, on the Reuters dataset, with 294 features, IMAP-LFM achieves a 48.2% clustering accuracy. In contrast, IBP-LFM uses 60 more features, but achieves 2.8% (absolute) lower accuracy. This suggests that IMAP is able to reduce the size of LFM (the number of features) without sacrificing modeling power. Because of IMAP’s diversity-promoting mechanism, the learned features bear less redundancy and are highly complementary to each other. Each feature captures
<table>
<thead>
<tr>
<th>Categories</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequencies of categories</td>
<td>3713</td>
<td>2055</td>
<td>321</td>
<td>298</td>
<td>245</td>
<td>197</td>
<td>142</td>
<td>114</td>
<td>110</td>
</tr>
<tr>
<td>Precision@100 (%) of IBP</td>
<td>73.7</td>
<td>45.1</td>
<td>7.5</td>
<td>8.3</td>
<td>6.9</td>
<td>7.2</td>
<td>2.6</td>
<td>3.8</td>
<td>3.4</td>
</tr>
<tr>
<td>Precision@100 (%) of IMAP</td>
<td>81.5</td>
<td>78.4</td>
<td>36.2</td>
<td>37.8</td>
<td>29.1</td>
<td>20.4</td>
<td>8.3</td>
<td>13.8</td>
<td>11.6</td>
</tr>
<tr>
<td>Relative improvement (%)</td>
<td>11</td>
<td>74</td>
<td>382</td>
<td>355</td>
<td>321</td>
<td>183</td>
<td>219</td>
<td>263</td>
<td>241</td>
</tr>
</tbody>
</table>

Table 3.13: Per-category precision@100 on the Reuters dataset.

<table>
<thead>
<tr>
<th>IBP-LFM [142]</th>
<th>Feature 1</th>
<th>Feature 2</th>
<th>Feature 3</th>
<th>Feature 4</th>
<th>Feature 5</th>
<th>Feature 1</th>
<th>Feature 2</th>
<th>Feature 3</th>
<th>Feature 4</th>
<th>Feature 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>government</td>
<td>saddam</td>
<td>nuclear</td>
<td>turkish</td>
<td>game</td>
<td>president</td>
<td>clinton</td>
<td>olympic</td>
<td>school</td>
<td>space</td>
<td></td>
</tr>
<tr>
<td>house</td>
<td>white</td>
<td>iraqi</td>
<td>soviet</td>
<td>gold</td>
<td>government</td>
<td>team</td>
<td>team</td>
<td>great</td>
<td>operation</td>
<td></td>
</tr>
<tr>
<td>baghdad</td>
<td>clinton</td>
<td>weapons</td>
<td>government</td>
<td>team</td>
<td>legal</td>
<td>lewinsky</td>
<td>hockey</td>
<td>institute</td>
<td>shuttle</td>
<td></td>
</tr>
<tr>
<td>weapons</td>
<td>united</td>
<td>weapons</td>
<td>season</td>
<td>years</td>
<td>nuclear</td>
<td>good</td>
<td>program</td>
<td>research</td>
<td>space</td>
<td></td>
</tr>
<tr>
<td>tax</td>
<td>time</td>
<td>work</td>
<td>number</td>
<td>april</td>
<td>state</td>
<td>work</td>
<td>baseball</td>
<td>study</td>
<td>life</td>
<td></td>
</tr>
<tr>
<td>years</td>
<td>president</td>
<td>spkr</td>
<td>enemy</td>
<td>man</td>
<td>baghdad</td>
<td>minister</td>
<td>gold</td>
<td>japanese</td>
<td>satellite</td>
<td></td>
</tr>
<tr>
<td>white</td>
<td>baghdad</td>
<td>president</td>
<td>good</td>
<td>number</td>
<td>church</td>
<td>weapons</td>
<td>ball</td>
<td>office</td>
<td>launch</td>
<td></td>
</tr>
<tr>
<td>united</td>
<td>iraqi</td>
<td>people</td>
<td>don</td>
<td>work</td>
<td>white</td>
<td>india</td>
<td>medal</td>
<td>reading</td>
<td>lunar</td>
<td></td>
</tr>
<tr>
<td>state</td>
<td>un</td>
<td>baghdad</td>
<td>citizens</td>
<td>baseball</td>
<td>united</td>
<td>years</td>
<td>april</td>
<td>level</td>
<td>device</td>
<td></td>
</tr>
<tr>
<td>bill</td>
<td>lewinsky</td>
<td>state</td>
<td>due</td>
<td>years</td>
<td>iraqi</td>
<td>white</td>
<td>winter</td>
<td>number</td>
<td>program</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.14: Visualization of features learned on the 20-News dataset.

a significant amount of information. As a result, a small number of such features are sufficient to model data well. In contrast, the features in IBP-LFM and PYP-LFM are drawn independently from a base distribution, which lacks the mechanism to reduce redundancy. IMAP achieves more significant reduction of feature numbers on datasets that have larger feature dimensions. This is possibly because higher-dimensional data contains more redundancy, giving IMAP a larger room to improve.

To verify whether IMAP helps to better capture infrequent patterns, on the learned features of the Reuters dataset we performed a retrieval task and measured the precision@100 on each category. For each test document, we retrieved 100 documents from the training set based on the Euclidean distance. Precision@100 is defined as \( n / 100 \), where \( n \) is the number of retrieved documents that share the same category label with the query document. We treat each category as a pattern and define its frequency as the number of documents belonging to it. A category with more than 1000 documents is labeled as being frequent. Table 3.13 shows the per-category precision. The last row shows the relative improvement of IMAP-LFM over IBP-LFM, defined as \( (P_{imap} - P_{ibp}) / P_{ibp} \). As can be seen, on the infrequent categories 3-9, IMAP-LFM achieves much better precision than IBP-LFM, while on the frequent categories 1 and 2, their performance are comparable. This demonstrates that IMAP is able to better capture infrequent patterns without losing the modeling power on frequent patterns.

On the 20-News dataset, we visualized the learned features. For a latent feature with parameter vector \( a \), we picked up the top 10 words corresponding to the largest values in \( a \). Table 3.14 shows 5 exemplar features learned by IBP-LFM and IMAP-LFM. As can be seen, the features learned by IBP-LFM have much overlap and redundancy and are hard to distinguish, whereas those learned by IMAP-LFM are more diverse.

Figure 3.3 shows how the L2 reconstruction error on the Yale test dataset varies as the parameter \( \kappa \) increases. \( \kappa \) controls the level of “diversity”. A larger \( \kappa \) results in more “diversity”.
As can be seen from this figure, the L2 error decreases as $\kappa$ increases from 0.01 to 1, which suggests that increasing the “diversity” among latent features is beneficial. This is possibly because the “diversity” effect acts as a type of inductive bias that controls the complexity of the model class, which thereby improves the generalization performance on unseen data. However, further increasing $\kappa$ from 1 to 100 causes the L2 error to increase. A possible explanation is: if $\kappa$ is too large, the diversity-promoting inductive bias is excessively strong, which compromises the quality of data-fitting.

We compared the convergence speed of the RM-HMC algorithm (Algorithm 2) with a random-walk based Metropolis Hastings (RW-MH) algorithm. The runtime is given in Table 3.15. As can be seen, RM-HMC is much more efficient than RW-MH, since it leverages the gradient information of $p(\tilde{a}_k | \text{rest})$ to search for “better” proposals of $\tilde{a}_k$. Compared with IBP-LFM, IMAP-LFM has an additional overhead of running the RM-HMC algorithm to sample $\tilde{a}$. Fortunately, RM-HMC converges very quickly, therefore this extra overhead is not substantial. For example, on the Block-Images dataset, the runtime of IBP-LFM and IMAP-LFM is 3.6 and 4.4 hours, respectively. On the 20-News dataset, the runtime of IBP-LFM and IMAP-LFM is 43.3 and 58.1 hours, respectively.

As can be seen from this figure, the L2 error decreases as $\kappa$ increases from 0.01 to 1, which suggests that increasing the “diversity” among latent features is beneficial. This is possibly because the “diversity” effect acts as a type of inductive bias that controls the complexity of the model class, which thereby improves the generalization performance on unseen data. However, further increasing $\kappa$ from 1 to 100 causes the L2 error to increase. A possible explanation is: if $\kappa$ is too large, the diversity-promoting inductive bias is excessively strong, which compromises the quality of data-fitting.

We compared the convergence speed of the RM-HMC algorithm (Algorithm 2) with a random-walk based Metropolis Hastings (RW-MH) algorithm. The runtime is given in Table 3.15. As can be seen, RM-HMC is much more efficient than RW-MH, since it leverages the gradient information of $p(\tilde{a}_k | \text{rest})$ to search for “better” proposals of $\tilde{a}_k$. Compared with IBP-LFM, IMAP-LFM has an additional overhead of running the RM-HMC algorithm to sample $\tilde{a}$. Fortunately, RM-HMC converges very quickly, therefore this extra overhead is not substantial. For example, on the Block-Images dataset, the runtime of IBP-LFM and IMAP-LFM is 3.6 and 4.4 hours, respectively. On the 20-News dataset, the runtime of IBP-LFM and IMAP-LFM is 43.3 and 58.1 hours, respectively.

![Figure 3.3: L2 error on the Yale test set versus the concentration parameter $\kappa$.](image)

Table 3.15: Runtime (hours) of RM-HMC and RW-MH.

<table>
<thead>
<tr>
<th></th>
<th>Block-Images</th>
<th>20-News</th>
</tr>
</thead>
<tbody>
<tr>
<td>RW-MH</td>
<td>7.6</td>
<td>93.7</td>
</tr>
<tr>
<td>RM-HMC</td>
<td>4.4</td>
<td>58.7</td>
</tr>
</tbody>
</table>
Chapter 4

Diversity-promoting Learning III – Analysis

In the previous two chapters, we have demonstrated the effectiveness of the proposed regularizers and Bayesian priors in (1) better capturing infrequent patterns, (2) achieving better generalization performance, and (3) reducing model size without sacrificing modeling power, via empirical studies. In this chapter, we provide theoretical analysis on why these regularizers/priors can achieve such effects.

4.1 Analysis of Better Capturing of Infrequent Patterns

On the distance metric learning model, we analyze why encouraging diversity using the nonconvex Bregman matrix divergence (NCBMD) regularizers [379] (Section 2.2.1) can better capture infrequent patterns [379]. In the analysis, we first define a score (the lower, the better) to quantify how well the infrequent patterns are captured. Then we upper bound this score using increasing functions of the NCBMD regularizers. Putting the two pieces together, we can conclude that reducing the NCBMD regularizers leads to better capturing of the infrequent patterns.

The analysis focuses on the following projection matrix:

\[ A^* = \arg \min_{A} \mathbb{E}_{S,D} \left[ \frac{1}{|S|} \sum_{(x,y) \in S} \| Ax - Ay \|_2^2 + \frac{1}{|D|} \sum_{(x,y) \in D} \max(0, \tau - \| Ax - Ay \|_2^2) + \gamma \Omega_{\phi}(A) \right]. \] (4.1)

We assume the data examples in \( S \) and \( D \) are from \( K \) classes (which are considered as patterns), where class \( k \) has a distribution \( \pi_k \) and the corresponding expectation is \( \mu_k \). Let \( p_k \) denote the prior probability of drawing a data example from class \( k \). We assume these classes are “imbalanced” in the sense that the variance of prior probabilities \( \{p_k\}_{k=1}^{K} \) is large. A class is considered as being “frequent” if it has a large \( p \) and “infrequent” if otherwise. We use \( \mu_k \) to represent class \( k \) and define the Mahalanobis distance between two classes \( j \) and \( k \) as: \( d_{jk} = (\mu_j - \mu_k)^\top A^\top A^*(\mu_j - \mu_k) \).

An imbalance factor (IF; the lower, the better) is defined to quantify how well the infrequent
Figure 4.1: Large and small circles denote frequent and infrequent classes respectively. The objective function of DML tends to make the inter-class distance to be large. (Left) The frequent classes are dominant in the training objective of DML, and hence are given higher priority for being pushed far away from each other. The infrequent classes are paid less attention to. Their distance ends up being small since DML is constrained by inner-class data pairs which need to have small distances. (Right) By promoting diversity, the infrequent classes are paid more attention to and their distance $d_2$ is enlarged. As a result, the ratio between $d_1$ and $d_2$ decreases.

The classes are captured:

$$\eta = \frac{\max_{j \neq k} d_{jk}}{\min_{j \neq k} d_{jk}}, \quad (4.2)$$

which is the ratio between the largest and smallest inter-class distance. Figure 4.1 illustrates why $\eta$ can reflect how well the infrequent classes are captured. For two frequent classes (denoted by large circles), since they have more training examples and hence contributing more in learning $A^*$, DML intends to make their distance ($d_1$) large; whereas for two infrequent classes (denoted by small circles), since they contribute less in learning (and DML is constrained by similar pairs which need to have small distances), their distance ($d_2$) may end up being small. Consequently, if classes are imbalanced, some between-class distances can be large while others small, resulting in a large IF (the ratio between $d_1$ and $d_2$), as shown in Figure 4.1(left). By promoting diversity, the DML model pays more attention to infrequent classes and tries to make their inter-class distance ($d_2$) large, as shown Figure 4.1(right). As a result, the IF decreases. To sum up, if the infrequent patterns are better captured, the imbalance factor is smaller.

Next, we upper bound the IF using increasing functions of the NCBMD regularizers. We define $\xi_k = \mathbb{E}_{x \sim \pi_k} [\sup_{\|v\|_2 = 1} |v^T (x - \mu_k)|]$ and $\xi = \max_k \xi_k$. Further, we assume $A^*$ has full rank $R$ (which is the number of the projection vectors), and let $U\Lambda U^T$ denote the eigen-decomposition of $A^*A^{*\top}$, where $\Lambda = \text{diag}(\lambda_1, \lambda_2, \cdots, \lambda_R)$ with $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_R$. The basic idea of deriving the upper bounds is as follows. First, we define $G = \text{span}\{\mu_j - \mu_k : j \neq k\}$ and prove that $G \subset \text{span}(A^*A^{*\top})$ under certain conditions where $\text{span}(A^*A^{*\top})$ denotes the column space of matrix $A^*A^{*\top}$. Next, we show that if $G \subset \text{span}(A^*A^{*\top})$ holds, we can bound $\eta$ with the condition number of $A^*A^{*\top}$. Finally, we bound this condition number with the NCBMD regularizers, including the von Neumann divergence (VND) regularizer and the log-determinant divergence (LDD) regularizer. Putting the pieces together, we obtain NCBMD-based upper bounds of the IF.

**Theorem 1.** Let $C$ denote the ratio between $\max_{j \neq k} \|\mu_j - \mu_k\|_2^2$ and $\min_{j \neq k} \|\mu_j - \mu_k\|_2^2$ and
assume $\max_{j,k} \|\mu_j - \mu_k\|_2 \leq B_0$. Suppose the regularization parameter $\gamma$ and distance margin $\tau$ are sufficiently large: $\gamma \geq \gamma_0$ and $\tau \geq \tau_0$, where $\gamma_0$ and $\tau_0$ depend on $\{p_k\}_{k=1}^{K}$ and $\{\mu_k\}_{k=1}^{K}$. If $R \geq K - 1$ and $\xi \leq (-B_0 + \sqrt{B_0^2 + \lambda_{K-1} \beta_{K-1} / (2\text{tr}(A))}) / 4$, then we have the following bounds for the IF.

- For the VND regularizer $\Omega_{vnd}(A^*)$, if $\Omega_{vnd}(A^*) \leq 1$, the following bound of the IF $\eta$ holds:

$$\eta \leq Cg(\Omega_{vnd}(A^*)),$$

where $g(\cdot)$ is an increasing function defined in the following way. Let $f(c) = c^{1/(c+1)}(1+1/c)$, which is strictly increasing on $(0, 1]$ and strictly decreasing on $[1, \infty)$ and let $f^{-1}(c)$ be the inverse function of $f(c)$ on $[1, \infty)$, then $g(c) = f^{-1}(2 - c)$ for $c < 1$.

- For the LDD regularizer $\Omega_{ldd}(A^*)$, we have

$$\eta \leq 4Ce^{\Omega_{ldd}(A^*)}.$$

As can be seen, the bounds are increasing functions of the NCBMD regularizers $\Omega_{vnd}(A^*)$ and $\Omega_{ldd}(A^*)$. Decreasing these regularizers would reduce the upper bounds of the imbalance factor, hence abridging the gap between infrequent and frequent classes. For the squared Frobenius norm (SFN) regularizer which is also an instance of the NCBMD regularizers, such a bound cannot be derived.

### 4.2 Analysis of Generalization Errors

In this section, we analyze why diversity-promoting regularization can improve the generalization performance on unseen data.

#### 4.2.1 Generalization Error Analysis for the Angular Constraints

Using neural networks as a study case, we analyze how the angular constraints \cite{[372]} (Section 2.3) affect the generalization performance. The generalization error of a hypothesis $f$ represented with a neural network is defined as $L(f) = \mathbb{E}_{(x,y) \sim p^*}[\ell(f(x), y)]$, where $p^*$ is the distribution of the input-output pair $(x, y)$ and $\ell(\cdot, \cdot)$ is the loss function. The training error is $\hat{L}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x^{(i)}), y^{(i)})$, where $n$ is the number of training samples. Let $f^* \in \text{argmin}_{f \in \mathcal{F}} L(f)$ be the true risk minimizer and $\hat{f} \in \text{argmin}_{f \in \mathcal{F}} \hat{L}(f)$ be the empirical risk minimizer. We aim at analyzing the generalization error $L(\hat{f})$ of the empirical risk minimizer $\hat{f}$. $L(\hat{f})$ can be decomposed into $L(\hat{f}) = L(\hat{f}) - L(f^*) + L(f^*)$, where $L(\hat{f}) - L(f^*)$ is the estimation error and $L(f^*)$ is the approximation error.

For simplicity, we start with a “simple” fully connected network with one hidden layer of $m$ units, used for univariate regression (one output unit) with squared loss. Analysis for more complicated NNs with multiple hidden layers can be achieved in a straightforward way by cascading our analysis for this “simple” NN. Let $x \in \mathbb{R}^d$ be the input vector and $y$ be the response value. For simplicity, we assume $\max\{\|x\|_2, |y|\} \leq 1$. Let $w_j \in \mathbb{R}^d$ be the weights connecting the $j$-th hidden unit with the input units, with $\|w_j\|_2 \leq C$. Let $A$ be a vector where $a_j$ is the weight
Let the activation function $h(t)$ applied on the hidden units is Lipschitz continuous with a constant $L$. Commonly used activation functions such as rectified linear $h(t) = \max(0, t)$, tanh $h(t) = (e^t - e^{-t})/(e^t + e^{-t})$, and sigmoid $h(t) = 1/(1 + e^{-t})$ are all Lipschitz continuous with $L = 1, 1, 0.25$, respectively. Consider the hypothesis set

$$
\mathcal{F} = \{ x \mapsto \sum_{j=1}^m \alpha_j h(w_j^\top x) \mid \|\alpha\|_2 \leq B, \|w_j\|_2 \leq C, \forall i \neq j, |w_i \cdot w_j| \leq \tau \|w_i\|_2 \|w_j\|_2 \}.
$$

The estimation error represents how well the algorithm is able to learn. We bound it in the following way. We first upper bound the estimation error using the Rademacher complexity (RC) \cite{32} $\mathcal{R}(\mathcal{F})$ of $\mathcal{F}$, then further bound $\mathcal{R}(\mathcal{F})$ using the hyperparameter $\tau$ in the angular constraints. When bounding the RC, we leverage the composition property \cite{32}: if a scalar function $\phi$ is Lipschitz with constant $L_\phi$ and satisfies $\phi(0) = 0$, then $\mathcal{R}(\phi \circ \mathcal{F}) \leq 2L_\phi \mathcal{R}(\mathcal{F})$. $L_\phi$ involves pairwise interactions between the weight vectors of hidden units, thus can be upper bounded using $\tau$. Putting the pieces together, we obtain the following estimation error bound.

**Theorem 2.** Let the activation function $h$ be $L$-Lipschitz continuous and the loss $\ell(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$. Then, with probability at least $1 - \delta$:

$$
L(\hat{f}) - L(f^*) \leq \frac{\gamma^2 \sqrt{2 \ln(4/\delta)} + 4\gamma B(2CL + \|h(0)\|) \sqrt{m}}{\sqrt{n}},
$$

where $\gamma = 1 + BCL\sqrt{(m - 1)\tau + 1} + \sqrt{mB}\|h(0)\|$.

Note that $\gamma$, hence the above bound on estimation error, decreases as $\tau$ becomes smaller. The bound goes to zero as $n$ (sample size) goes to infinite. The inverse square root dependence on $n$ matches existing results \cite{32}. We note that it is straightforward to extend our bound to any bounded Lipschitz continuous loss $\ell$.

The approximation error indicates how capable the hypothesis set $\mathcal{F}$ is to approximate a target function $g = \mathbb{E}[y|x]$, where the error is measured by $\min_{f \in \mathcal{F}} \|f - g\|_{L^2}$ and $\|f - g\|_{L^2} = \int (f(x) - g(x))^2 P(dx)$. Following \cite{31}, we assume the target function $g$ satisfies certain smoothness condition that is expressed in the first moment of its Fourier representation: $\int |\omega|_2 \|\hat{g}(\omega)\| d\omega \leq B/2$ where $\hat{g}(\omega)$ is the Fourier representation of $g(x)$. To study the approximation error, we first define an auxiliary function class $\tilde{\mathcal{F}} = \{ x \mapsto \sum_{j=1}^m \alpha_j h(w_j^\top x) \mid \|\alpha\|_2 \leq B, \|w_j\|_2 \leq C \}$. $\tilde{\mathcal{F}}$ differs from $\mathcal{F}$ in that the angular constraints $\forall i \neq j, |w_i \cdot w_j| \leq \tau \|w_i\|_2 \|w_j\|_2$ are removed. Note that for $f \in \mathcal{F}$ and $\tilde{f} \in \tilde{\mathcal{F}}$, we have $\|g - \tilde{f}\|_{L^2} \leq \|g - f\|_{L^2} + \|\tilde{f} - f\|_{L^2}$. For $\|\tilde{f} - f\|_{L^2}$, we derive an upper bound which is a quantity involving $\tau$. For $\|g - \tilde{f}\|_{L^2}$, its upper bound has been derived in \cite{31}. Consequently, we obtain the upper bound of $\|g - f\|_{L^2}$, which is given in the following theorem. In order to derive explicit constants we restrict $h$ to be the sigmoid function, but other Lipschitz continuous activation function can be similarly handled.

**Theorem 3.** Let $C > 1$, $m \leq 2\left(\frac{2 - \theta}{\theta} + 1\right)$, where $\theta = \arccos(\tau)$, and $h(t) = 1/(1 + e^{-t})$. Then, there is a function $f \in \mathcal{F}$ such that

$$
\|f - g\|_{L^2} \leq B\left(\frac{1}{\sqrt{m}} + \frac{\min(3m\theta, \pi)}{C}\right) + 2\sqrt{mBC} \sin\left(\frac{\min(3m\theta, \pi)}{2}\right).
$$
This theorem implies that whether to use the angular constraint (AC) or not has a significant influence on the approximate error bound: without using AC ($\tau = 1$), the bound is a decreasing function of $m$ (the number of hidden units); using AC ($\tau < 1$), the bound increases with $m$. This striking phrase-change indicates the impact of AC. Given a fixed $m$, the bound decreases with $\tau$, implying that a stronger regularization (smaller $\tau$) incurs larger approximation error. When $\tau = 1$, the second term in the bound vanishes and the bound is reduced to the one in [31], which is a decreasing function of $m$ and $C$, the upper bound on the weights. When $\tau < 1$, the second term increases with $m$ up to the upper bound $2(\lceil \frac{\pi - \theta}{\theta} \rceil + 1)$. This is because a larger number of hidden units bear a larger difficulty in satisfying the pairwise ACs, which causes the function space $F$ to shrink rapidly; accordingly, the approximation power of $F$ decreases quickly.

The analysis in the two theorems shows that $\tau$ incurs a tradeoff between the estimation error and the approximation error: decreasing $\tau$ reduces the estimation error and enlarges the approximation error. Since the generalization error is the sum of the two errors, $\tau$ has an optimal value to yield the minimal generalization error.

### 4.2.2 Estimation Error Analysis for the Nonconvex Bregman Matrix Divergence Regularizers

In this section, we analyze how the nonconvex Bregman matrix divergence (BMD) regularizers [379] (Section 2.2.1) affect the estimation error of ML models. We use the distance metric learning (DML) model to perform the study. In DML, the hypothesis function is $u(x, y) = \|W^T(x - y)\|_2^2$ and the loss function $\ell$ is the logistic loss $\ell(u(x, y), t) = \log(1 + \exp((2t - 1)u(x, y)))$. Let $U = \{u : (x, y) \mapsto \|W^T(x - y)\|_2^2, \Omega(W) \leq \tau\}$ denote the hypothesis set and $A = \{\ell : (x, y, t) \mapsto \ell(u(x, y), t), u \in U\}$ denote the loss class, which is the composition of the loss function with each of the hypotheses. In $U$, we add the constraint $\Omega(W) \leq \tau$ to incorporate the impact of the BMD regularizers $\Omega(W)$. $\tau$ controls the strength of regularization. A smaller $\tau$ entails stronger promotion of diversity. $\tau$ is controlled by the regularization parameter $\lambda$ in Eq.(2.14). Increasing $\lambda$ reduces $\tau$. Given the joint distribution $p^*$ of the input data pair $(x, y)$ and the binary label $t$ indicating whether this data pair is similar or dissimilar, the risk of the hypothesis $u$ is $L(u) = \mathbb{E}_{(x, y, t) \sim p^*}[\ell(u(x, y), t)]$. Its empirical counterpart (training error) can be defined as $\hat{L}(u) = \frac{1}{N} \sum_{n=1}^{N} \ell(u(x_n, y_n), t_n)$. The estimation error of a hypothesis $u$ is defined as $L(u) - \hat{L}(u)$, which represents how well the algorithm can learn and usually depends on the complexity of the hypothesis class and the number of training examples.

To upper bound the estimation error, we define a capacity variable on $W$.

**Definition 1 (Capacity Variable).** Let $\pi_1, \cdots, \pi_m$ be the eigenvalues of $W^TW$. Then the capacity variable is defined as:

$$C(W) = \sum_{j=1}^{m} |\pi_j - 1|.$$

The following inequality helps us to understand the intuitive meaning of $C(W)$:

$$\frac{1}{m} \|W^TW - I_m\|_1 \leq C(W).$$

(4.4)
\[ \frac{1}{m} \| W^T W - I_m \|_1 \] measures the closeness between the Gram matrix \( W^T W \) and an identity matrix using L1 norm. The smaller this quantity is, the closer to being orthogonal the vectors in \( W \) are. Being an upper bound of this quantity, \( C(W) \) essentially determines the level of near-orthogonality among vectors.

We first prove that the estimation error can be upper bounded by an increasing function of the capacity variable. Then we show that the capacity variable can be upper bounded by an increasing function of the nonconvex BMD regularizers, including the log-determinant divergence (LDD) regularizer, the von Neumann divergence (VND) regularizer, and the squared Frobenius norm (SFN) regularizer. Combining these two steps we reveal the relationship between the estimation errors and the BMD regularizers, which is given in the following theorem.

**Theorem 4.** Suppose \( \sup_{(x,y)} \|x-y\|_2 \leq B_0 \) and any column vector \( w \) of \( W \) satisfies \( \|w\|_2 \leq D \). With probability at least \( 1 - \delta \), we have:

- For the LDD regularizer,
  \[
  L(u) - \hat{L}(u) \leq 4B_0^2D \sqrt{m g_{idd}(\tau/m)^m + 1} \left\{ \begin{array}{ll}
  \frac{1}{N} \log(1/\delta) & \text{if } \tau < 1/m \\
  \frac{1}{N} \log(1/\delta) + \frac{1}{N} & \text{if } \tau \geq 1/m
  \end{array} \right.
  \] (4.5)
  where \( g_{idd}(x) = x - \log(x+1) \).

- For the VND regularizer,
  \[
  L(u) - \hat{L}(u) \leq 4B_0^2D \sqrt{m g_{vnd}(\tau/m)^m + 1} \left\{ \begin{array}{ll}
  \frac{1}{N} \log(1/\delta) & \text{if } \tau < 1/m \\
  \frac{1}{N} \log(1/\delta) + \frac{1}{N} & \text{if } \tau \geq 1/m
  \end{array} \right.
  \] (4.6)
  where \( g_{vnd}(x) = (x+1) \log(x+1) - x \).

- For the SFN regularizer,
  \[
  L(u) - \hat{L}(u) \leq (4 + \sqrt{2 \log(1/\delta)}) B^2 \sqrt{\frac{\tau m}{N} + \frac{4B_0^2m + (B^2m + 1) \sqrt{2 \log(1/\delta)}}{N}}
  \] (4.7)

From these estimation error bounds (EEBs), we can see two major implications. First, BMD regularizers can effectively control the EEBs. Increasing the strength of BMD regularization (by enlarging \( \lambda \)) reduces \( \tau \), which decreases the EEBs since they are increasing functions of \( \tau \). Second, the EEBs converge with rate \( O(1/\sqrt{N}) \), where \( N \) is the number of training data pairs. This rate matches with that in [37, 338].

### 4.2.3 Estimation Error Analysis for the Convex Bregman Matrix Divergence Regularizers

In this section, we analyze how the convex BMD (CBMD) regularizers [379] (Section 2.2.2) affect the estimation error of CBMD-regularized distance metric learning problem (Eq. (2.22)). Following [338], we use distance-based error to measure the quality of a Mahalanobis distance matrix \( M \). Given the sample \( S \) and \( D \) where the total number of data pairs is \( m = |S| + |D| \), the empirical error is defined as \( \hat{L}(M) = \frac{1}{|S|} \sum_{(x,y) \in S} (x-y)^\top M (x-y) + \frac{1}{|D|} \sum_{(x,y) \in D} \text{max}(0, \tau - (x-y)^\top M (x-y)) \) and the expected error is \( L(M) = E_{S,D}[\hat{L}(M)] \). Let \( \hat{M}^* \) be the optimal matrix learned by minimizing the empirical error: \( \hat{M}^* = \arg\min_M \hat{L}(M) \). We are interested in
how well $\hat{M}^*$ performs on unseen data. The performance is measured using the estimation error:

$$E = L(\hat{M}^*) - \hat{L}(\hat{M}^*).$$

To incorporate the impact of the CBMD regularizers $\Omega_\phi(M)$, we define the hypothesis class of $M$ to be $\mathcal{M} = \{M \succeq 0 : \Omega_\phi(M) \leq C\}$. The upper bound $C$ controls the strength of regularization. A smaller $C$ entails stronger promotion of orthogonality. $C$ is controlled by the regularization parameter $\gamma$ in Eq. (4.1). Increasing $\gamma$ reduces $C$. We perform the estimation error analysis as follows. We first establish an upper bound of the estimation error based on the Rademacher complexity $R(M)$ of $M$. Then we upper bound $R(M)$ using the trace of $M$. Finally, we derive upper bound of the trace based on the CBMD regularizers. Combining the three steps together, we establish upper bounds of the estimation error based on the CBMD regularizers, including the convex von Neumann divergence (CVND) regularizer, the convex log-determinant divergence (CLDD) regularizer, and the convex squared Frobenius norm (CSFN) regularizer:

**Theorem 5.** Suppose $\sup_{\|v\|_2 \leq 1, (x, y) \in S} |v^\top (x - y)| \leq B$, then with probability at least $1 - \delta$, we have:

- For the CVND regularizer,
  $$E \leq \frac{4B^2C + \max(\tau, B^2\sqrt{\log(1/\delta)})}{\sqrt{m}}.$$

- For the CLDD regularizer,
  $$E \leq \frac{4B^2C}{\log(1/\epsilon) - 1} + \max(\tau, \frac{C - D\epsilon}{\log(1/\epsilon) - 1}) \frac{\sqrt{2\log(1/\delta)}}{\sqrt{m}}.$$

- For the CSFN regularizer,
  $$E \leq \frac{2B^2 \min(2C, \sqrt{C}) + \max(\tau, C) \sqrt{2\log(1/\delta)}}{\sqrt{m}}.$$

From these estimation error bounds (EEBs), we can see two major implications. First, CBMD regularizers can effectively control the EEBs. Increasing the strength of CBMD regularization (by enlarging $\gamma$) reduces $C$, which decreases the EEBs since they are all increasing functions of $C$. Second, the EEBs converge with rate $O(1/\sqrt{m})$, where $m$ is the number of training data pairs. This rate matches with that in [37, 338].

### 4.3 Appendix: Proofs

#### 4.3.1 Proof of Theorem 1

**Proof Sketch**

We make the following two assumptions.

- The size of the similar and dissimilar set $|S|$ and $|D|$ are fixed.
- $A^*$ has full row rank $R$.

Denote the $K$ classes as $C_1, C_2, \cdots, C_K$. The probability that a sample is drawn from the $k$-th class is $p_k$, and $\sum_{k=1}^K p_k = 1$. Denote the class membership of an example $x$ as $c(x)$. Denote the probability that $x \in C_j, y \in C_k$ where $(x, y) \in D$ as $p_{jk} = p_j p_k / (1 - \sum_{l=1}^K p_l^2)$. Define
the singular value decomposition (SVD) of the matrix $A^*$ as $U \sqrt{\Lambda} V^\top$ where $U \in \mathbb{R}^{R \times R}$, $\Lambda \in \mathbb{R}^{R \times R}$, and $V \in \mathbb{R}^{D \times R}$. Let $\Omega = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_R)$. Then $A^* A^* = V \Lambda V^\top$. Denote $V = [v_1, v_2, \cdots, v_R]$. Then $\forall z = x - y$, we have $z^\top A^* A^* z = \sum_{r=1}^{R} \lambda_r (v_r^\top z)^2$. We see $z^\top A^* A^* z$ can be written as a sum of $R$ terms. Inspired by this, we define a vector function $\alpha(\cdot)$ as $\alpha(u) = \sum_{j,k=1}^{K} p_{jk} (u^\top (\mu_j - \mu_k))^2$. This function measures the weighted sum of $(u^\top (\mu_j - \mu_k))^2$ across all classes. Define $\mathcal{G} = \text{span}\{\mu_j - \mu_k : j \neq k\}$.

**Definition 2** (feature values and feature vectors). For a linear space $\mathcal{G}$, define vectors $w_1, w_2, \cdots, w_{K-1}$ and positive real numbers $\beta_1, \beta_2, \cdots, \beta_{K-1}$ as

$$w_1 = \arg \min_{\|u\|=1, u \in \mathcal{G}} \alpha(u), \quad \beta_1 = \alpha(w_1),$$

$$w_r = \arg \min_{\|u\|=1, u \in \mathcal{G}} \alpha(u), \quad \beta_r = \alpha(w_r), \quad \forall r > 1$$

$\forall r > K - 1$, define $\beta_r = 0$, and $w_r$ as an arbitrary vector which has norm 1 and is orthogonal to $w_1, w_2, \cdots, w_{r-1}$. $w_1, w_2, \cdots, w_{K-1}$ are called feature vectors of $\mathcal{G}$, and $\beta_1, \beta_2, \cdots, \beta_{K-1}$ are called feature values of $\mathcal{G}$.

We give a condition on the regularizers.

**Condition 1.** For a regularizer $\Omega_{\phi}(\cdot)$, there exists a unique matrix function $\varphi(\cdot)$ such that for any $A^*$,

$$\Omega_{\phi}(A^*) = \varphi(A^* A^* \top) = \varphi(\Lambda).$$

The VND and LDD regularizers satisfy this condition. For the VND regularizer, $\varphi(\Lambda) = \text{tr}(\Lambda \log \Lambda - \Lambda) + R$; for the LDD regularizer, $\varphi(\Lambda) = \text{tr}(\Lambda) - \log \det(\Lambda) - R$. The SFN regularizer does not satisfy this condition.

Now we have enough preparation to give the following lemma. It shows that the linear space $\mathcal{G}$ can be recovered if the second moment of noise is smaller than a certain value.

**Lemma 5.** Suppose $R \geq K - 1$, $\max_{j \in \mathcal{C}} \|\mu_j - \mu_k\|_2 \leq B_0$, and the regularization parameter $\gamma$ and distance margin $\tau$ satisfy $\gamma \geq \gamma_0, \tau \geq \tau_0$. If $\xi \leq \frac{-B_0 + \sqrt{B_0^2 + 2\gamma K - 1 B_{K-1} \gamma_{K-1}}}{4}$, then

$$\mathcal{G} \subseteq \text{span}(A^* A^* \top).$$

Here $\text{span}(A^* A^* \top)$ denotes the column space of matrix $A^* A^* \top$. Both $\gamma_0$ and $\tau_0$ depend on $\mu_1, \mu_2, \cdots, \mu_K$ and $p_1, p_2, \cdots, p_K$.

The next lemma shows that if Eq. (4.8) holds, we can bound the imbalance factor $\eta$ with the condition number of $A^* A^* \top$ (denoted by $\text{cond}(A^* A^* \top)$). Note that the BMD regularizers $\Omega_{\phi}(A^*)$ encourage $A^* A^* \top$ to be close to an identity matrix, i.e., encouraging the condition number to be close to 1.

**Lemma 6.** If Eq. (4.8) holds, and there exists a real function $g$ such that

$$\text{cond}(A^* A^* \top) \leq g(\Omega_{\phi}(A^*)), $$

then we have the following bound for the imbalance factor

$$\eta \leq g(\Omega_{\phi}(A^*)) \frac{\max_{j \neq k} \|\mu_j - \mu_k\|^2}{\min_{j \neq k} \|\mu_j - \mu_k\|^2}. $$

117
Next, we derive the explicit forms of $g$ for the VND and LDD regularizers.

**Lemma 7.** For the VND regularizer $\Omega_{\text{vnd}}(A^*)$, define $f(c) = c^{1/(c+1)}(1 + 1/c)$, then $f(c)$ is strictly increasing on $(0, 1]$ and strictly decreasing on $[1, \infty)$. Define the inverse function of $f(\cdot)$ on $[1, \infty)$ as $f^{-1}(\cdot)$. Then if $\Omega_{\text{vnd}}(A^*) < 1$, we have

$$\text{cond}(A^*A^{*\top}) \leq f^{-1}(2 - \Omega_{\text{vnd}}(A^*)).$$  

For the LDD regularizer $\Omega_{\text{ldd}}(A^*)$, we have

$$\text{cond}(A^*A^{*\top}) \leq 4e^{\Omega_{\text{ldd}}(A^*)}.$$  

Combining Lemma 5, 6, and 7 we finish the proof of Theorem 1.

**Proof of Lemma 8.**

In order to prove Lemma 8, we first need some auxiliary lemmas on the properties of the function $\alpha(\cdot)$. Denote $\mu_{jk} = \mu_j - \mu_k, \forall j \neq k$.

**Lemma 8.** Suppose $u_1, u_2, \cdots, u_r$ and $v_1, v_2, \cdots, v_r$ are two sets of standard orthogonal vectors in $\mathbb{R}^d$, and $\text{span}(u_1, u_2, \cdots, u_r) = \text{span}(v_1, v_2, \cdots, v_r)$, then we have

$$\sum_{l=1}^{r} \alpha(u_l) = \sum_{l=1}^{r} \alpha(v_l).$$

**Proof.** By the definition of these two sets of vectors, there exists a $r \times r$ standard orthogonal matrix $B = (b_{jk})$, such that $(u_1, u_2, \cdots, u_r) = (v_1, v_2, \cdots, v_r)B$. Then we have

$$\sum_{l=1}^{r} \alpha(u_l) = \sum_{l=1}^{r} \sum_{j \neq k} p_{jk} ((\sum_{s=1}^{r} b_{ls}v_s)^\top \mu_{jk})^2$$

$$= \sum_{j \neq k} p_{jk} \sum_{s,t=1}^{r} b_{ls}b_{lt}v_s^\top \mu_{jk}v_t^\top \mu_{jk}$$

$$= \sum_{s=1}^{r} \sum_{j \neq k} p_{jk} (v_s^\top \mu_{jk})^2 \sum_{l=1}^{r} b_{ls}^2 + \sum_{j \neq k} p_{jk} \sum_{s,t=1}^{r} v_s^\top \mu_{jk}v_t^\top \mu_{jk} \sum_{l=1}^{r} b_{ls}b_{lt}.$$  

Since $B$ is a standard orthogonal matrix, we have $\forall s, \sum_{l=1}^{r} b_{ls}^2 = 1$ and $\forall s \neq t, \sum_{l=1}^{r} b_{ls}b_{lt} = 0$. Further, we have

$$\sum_{l=1}^{r} \alpha(u_l) = \sum_{l=1}^{r} \alpha(v_l).$$

\[ \square \]

**Lemma 9.** For any positive integer $r$, any set of standard orthogonal vectors $u_1, u_2, \cdots, u_r \in \mathbb{R}^d$, and real numbers $\gamma_1 \geq \gamma_2 \geq \cdots \geq \gamma_r \geq 0$, we have

$$\sum_{l=1}^{r} \gamma_l \alpha(u_l) \leq \sum_{l=1}^{r} \gamma_l \beta_l,$$

where $\beta_l$ is the $l$-th feature value.
Proof. We first prove the situation that \( \gamma_1 = \gamma_2 = \cdots = \gamma_r = 1 \), i.e.,

\[
\sum_{l=1}^{r} \alpha(u_l) \leq \sum_{l=1}^{r} \beta_l. \tag{4.10}
\]

We prove it by induction on \( r \). For \( r = 1 \), by the definition of feature values and feature vectors, Eq. (4.10) holds. Now supposing Eq. (4.10) holds for \( r = s \), we prove it holds for \( r = s + 1 \) by contradiction. If Eq. (4.10) does not hold, then there exist standard orthogonal vectors \( u_1, u_2, \cdots u_{s+1} \in \mathbb{R}^d \), such that

\[
\sum_{l=1}^{s+1} \alpha(u_l) > \sum_{l=1}^{s+1} \alpha(w_l), \tag{4.11}
\]

where \( w_l \) are feature vectors. Since the dimension of \( \text{span}(u_1, u_2, \cdots u_{s+1}) \) is \( s + 1 \), there exists \( \tilde{w}_{s+1} \in \text{span}(u_1, u_2, \cdots u_{s+1}) \), such that \( \tilde{w}_{s+1} \perp w_l, \forall 1 \leq l \leq s \). By the definition of the feature vector \( w_{s+1} \), we have

\[
\sum_{l=1}^{s+1} \alpha(w_l) \geq \sum_{l=1}^{s} \alpha(w_l) + \alpha(\tilde{w}_{s+1}). \tag{4.12}
\]

Let \( \tilde{w}_1, \tilde{w}_2, \cdots \tilde{w}_{s+1} \) be a set of standard orthogonal basis of \( \text{span}(u_1, u_2, \cdots u_{s+1}) \), by Lemma \[8\] we have

\[
\sum_{l=1}^{s+1} \alpha(u_l) = \sum_{l=1}^{s+1} \alpha(\tilde{w}_l). \tag{4.13}
\]

Combining equation (4.11), (4.12) and (4.13) we get

\[
\sum_{l=1}^{s+1} \alpha(\tilde{w}_l) > \sum_{l=1}^{s} \alpha(w_l) + \alpha(\tilde{w}_{s+1}).
\]

Thus we have

\[
\sum_{l=1}^{s} \alpha(\tilde{w}_l) > \sum_{l=1}^{s} \alpha(w_l).
\]

This contradicts with our induction assumption. The proof for the \( \gamma_1 = \gamma_2 = \cdots = \gamma_r = 1 \) case completes.

Next, we prove the situation that \( \gamma_l \) are not all equal to 1, by utilizing Eq. (4.10).

\[
\sum_{l=1}^{r} \gamma_l \alpha(u_l) = \sum_{l=1}^{r-1} \left[ (\gamma_l - \gamma_{l+1}) \sum_{t=1}^{l} \alpha(u_t) \right] + \gamma_r \sum_{t=1}^{r} \alpha(u_t)
\leq \sum_{l=1}^{r-1} \left[ (\gamma_l - \gamma_{l+1}) \sum_{t=1}^{l} \beta_t \right] + \gamma_r \sum_{t=1}^{r} \beta_t
\leq \sum_{l=1}^{r} \gamma_l \beta_l
\]

The proof completes. \( \square \)
Note that in Lemma 9, \( r \) can be larger than the number of nonzero feature values \( K - 1 \). This will be used in the proof of Lemma 5 later on.

Another auxiliary lemma needed to prove Lemma 5 is given below.

**Lemma 10.** Suppose \( w_0 \in \mathcal{G} \), define a linear space \( \mathcal{H} = \{v \in \mathcal{G} : v \perp w_0\} \). Then there are \( K - 2 \) nonzero feature values of \( \mathcal{H} \). Denote them as \( \beta_1', \beta_2', \ldots \beta_{K-2}' \), then \( \forall \gamma_1 \geq \gamma_2 \geq \cdots \geq \gamma_r \geq 0, \)

\[
\sum_{l=1}^{r} \gamma_l \beta_l' \leq \sum_{l=1}^{r} \gamma_l \beta_l.
\]

**Proof.** Note that the dimension of \( \mathcal{H} \) is \( K - 2 \), then there are \( K - 2 \) nonzero feature values. The feature vectors of \( \mathcal{H} \) are also standard orthogonal vectors of the linear space \( \mathcal{G} \). By Lemma 9, we have \( \sum_{l=1}^{r} \gamma_l \beta_l' \leq \sum_{l=1}^{r} \gamma_l \beta_l, \forall r \leq K - 2. \)

Now we are ready to prove Lemma 5.

**Proof.** (of Lemma 5) We conduct the proof by contradiction. Assuming Eq. (4.8) does not hold, we prove \( A^* \) can not be the global optimal solution. Let \( U \sqrt{\Lambda V^\top} \) be the SVD of \( A^* \). Define \( W = (w_1, w_2, \ldots w_R) \) as a matrix whose columns contain the feature vectors. Let \( \tilde{A} = U \sqrt{\Lambda W}^\top \). Then by Condition 6 we have \( \Omega_\phi(A^*) = \Omega_\phi(\tilde{A}) \). Define

\[
L(A) = E\left[ \frac{1}{|S|} \sum_{(x,y) \in S} \|Ax - Ay\|_2^2 + \frac{1}{|D|} \sum_{(x,y) \in D} \max(0, \tau - \|Ax - Ay\|_2^2) \right].
\]

Assuming Eq. (4.8) does not hold, we prove \( L(A^*) > L(\tilde{A}) \), i.e., \( A^* \) is not the optimal solution. We consider two cases: \( \xi = 0 \) and \( \xi \neq 0 \). Define \( h(A^*, \xi) = L(A^*) \) and \( h(\tilde{A}, \xi) = L(\tilde{A}) \). When \( \xi = 0 \), we have:

\[
h(A^*, 0) = E\left[ \frac{1}{|S|} \sum_{(x,y) \in S} \|A^* \mu_{c(x)} - A^* \mu_{c(y)}\|_2^2 + \frac{1}{|D|} \sum_{(x,y) \in D} \max(0, \tau - \|A^* \mu_{c(x)} - A^* \mu_{c(y)}\|_2^2) \right]
\]

\[
= \sum_{j \neq k} p_{jk} \max(0, \tau - \|A^* (\mu_j - \mu_k)\|_2^2),
\]

and

\[
h(\tilde{A}, 0) = \sum_{j \neq k} p_{jk} \max(0, \tau - \|\tilde{A} (\mu_j - \mu_k)\|_2^2).
\]

Since Eq. (4.8) does not hold by assumption, there exists a \( w_0 \) satisfying \( w_0 \in \mathcal{G} \) and \( w_0 \notin \text{span}(A^*) \). Denote \( \mathcal{H} = \{v \in \mathcal{G} : v \perp w_0\} \) and its \( K - 2 \) nonzero feature values as \( \beta_1', \beta_2', \ldots \beta_{K-2}' \). \( \forall u \in \text{span}(A^*) \), let \( u' \) be the projection of \( u \) to the space \( \mathcal{H} \) and \( u' \) is rescaled to have norm 1. Then \( \alpha(u') \geq \alpha(u) \). Thus, \( \forall r, \) the \( r \)-th feature value of \text{span}(A^*) \) is no larger than the \( r \)-th feature value of \( \mathcal{G} \). By Lemma 10, we have \( \sum_{l=1}^{r} \gamma_l \beta_l' \leq \sum_{l=1}^{r} \gamma_l \beta_l \). By the definition of feature values, we have

\[
\sum_{j \neq k} p_{jk} \|A^* (\mu_j - \mu_k)\|_2^2 = \sum_{l=1}^{R} \gamma_l \alpha(a_l) \leq \sum_{l=1}^{R} \gamma_l \beta_l' \cdot 120
\]
Since $H$ has only $K - 2$ nonzero feature values, we have
\[
\sum_{l=1}^{K-2} \gamma_l \beta'_l = \sum_{l=1}^{K-2} \gamma_l \beta'_l \leq \sum_{l=1}^{K-2} \gamma_l \beta_l = \sum_{l=1}^{K-1} \gamma_l \alpha(w_l) - \gamma_{K-1} \beta_{K-1} = \sum_{j \neq k} p_{jk} \| \tilde{A}(\mu_j - \mu_k) \|_2^2 - \gamma_{K-1} \beta_{K-1}.
\]
So we have
\[
\sum_{j \neq k} p_{jk} \| \tilde{A}(\mu_j - \mu_k) \|_2^2 \geq \sum_{j \neq k} p_{jk} \| A^*(\mu_j - \mu_k) \|_2^2 + \gamma_{K-1} \beta_{K-1}.
\]

Next, we establish a relationship between $h(A^*, 0)$ and $h(\tilde{A}, 0)$, which is given in the following lemma.

**Lemma 11.** There exist constants $\tau_0, \gamma_0$ which are determined by $p_1, p_2, \ldots, p_K$ and $\mu_1, \mu_2, \ldots, \mu_K$, such that if $\tau \geq \tau_0, \gamma \geq \gamma_0$, then we have
\[
h(A^*, 0) - h(\tilde{A}, 0) > \frac{1}{2} \gamma_{K-1} \beta_{K-1}.
\]

**Proof.** If $\| \tilde{A}(\mu_j - \mu_k) \|_2 \leq \tau$ and $\| A^*(\mu_j - \mu_k) \|_2 \leq \tau$ for all $j \neq k$, we have $h(A^*, 0) - h(\tilde{A}, 0) = \gamma_{K-1} \beta_{K-1}$. Since $\max_{j \neq k} \| \mu_j - \mu_k \|_2 = B_0$, we have
\[
\| A^*(\mu_j - \mu_k) \|_2 \leq \text{tr}(A)B_0^2, \quad \| \tilde{A}(\mu_j - \mu_k) \|_2 \leq \text{tr}(A)B_0^2, \quad \forall j \neq k.
\] (4.14)

Select $\tau_0$ such that $\tau_0 \geq K(1 + \epsilon_0)B_0^2$, where $\epsilon_0$ is any positive constant. For the VND and LDD regularizers, as $\gamma \to \infty$, $A \to I_R$. Thereby, there exists $\gamma_0$, such that if $\gamma \geq \gamma_0, \forall j, |\lambda_j - 1| \leq \epsilon$. Hence, if $\gamma \geq \gamma_0, \tau \geq \tau_0$,
\[
\text{tr}(A)B_0^2 \leq K(1 + \epsilon_0)B_0^2 \leq \tau_0.
\]
Combining this inequality with Eq. (4.14), we finish the proof. \qed

Now we continue to prove Lemma 5. In Lemma 11, we have already proved that $h(A^*, 0)$ is strictly larger than $h(\tilde{A}, 0)$. We then prove that if the noise is smaller than a certain value, $h(A^*, \xi)$ is strictly larger than $h(\tilde{A}, \xi)$. By the definition of $\xi$, we have
\[
|h(A^*, \xi) - h(A^*, 0)| \leq E \left| \frac{1}{|S|} \sum_{(x,y) \in S} \| A^*(x - y) \|_2^2 + E \frac{1}{|D|} \sum_{(x,y) \in D} \| A^*(x - y) \|_2^2 - \| A^*(\mu_c(x) - \mu_c(y)) \|_2^2 \right|
\leq 4\text{tr}(A)\xi^2 + (4B_0\xi + 4\xi^2)\text{tr}(A)
\leq 8\xi^2\text{tr}(A) + 4B_0\xi\text{tr}(A).
\] (4.15)
Similarly, we have
\[
|h(A^*, \xi) - h(A^*, 0)| \leq 8\xi^2\text{tr}(A) + 4B_0\xi\text{tr}(A).
\] (4.16)
Combining Lemma 11 with Eq. (4.15) and Eq. (4.16), we have if $\xi \leq -B_0 + \sqrt{B_0^2 + \gamma_{K-1} \beta_{K-1}/(2\text{tr}(A))}$, then $L(A^*) > L(\tilde{A})$, i.e., $A^*$ is not the global optimal solution. By contradiction, Eq. (4.8) holds. The proof completes. \qed
Proof of Lemma 6

Proof. For any vector $u \in G$, since the condition of Lemma 5 is satisfied, we have $u \in \text{span}(A^*)$. Recall $A^\top A^* = VTV^\top$ and $V = [v_1, v_2, \cdots v_R]$. We can denote $u$ as $u = \|u\| \sum_{j=1}^R t_j v_j$, where $\sum_{j=1}^R t_j^2 = 1$. Then we have $\forall u \in G$,

$$
    u^\top A^\top A^* u = \sum_{j=1}^R \langle v_j, u \rangle^2 t_j \lambda_j = \sum_{j=1}^R \|u\|^2 t_j^2 \lambda_j \leq \|u\|^2 \lambda_1.
$$

Similarly, we have $u^\top A^\top A^* u \geq \|u\|^2 \lambda_R$. Noting $\forall j \neq k, \mu_j - \mu_k \in G$, we have

$$
    \eta \leq \text{cond}(A^\top A^*) \frac{\max_{j \neq k} \|\mu_j - \mu_k\|^2}{\min_{j \neq k} \|\mu_j - \mu_k\|^2}.
$$

Combining this inequality with $\text{cond}(A^\top A^*) \leq g(\Omega_\phi(A^*))$, we complete the proof. \qed

Proof of Lemma 7

Proof. We first prove the result about the VND regularizer. Define a scalar function $s(x) = x \log x - x + 1$ and denote $\text{cond}(A^\top A^*) = c$. Since $s'(x) = \log x$, and $s(1) = 0$, we have

$$
    \Omega_{vnd}(A^*) = \sum_{j=1}^R s(\lambda_j) \geq s(\lambda_1) + s(\lambda_R) = \lambda_1 \log \lambda_1 - \lambda_1 + \frac{\lambda_1}{c} \log \frac{\lambda_1}{c} - \frac{\lambda_1}{c} + 2.
$$

Define $F(\lambda_1, c) = \lambda_1 \log \lambda_1 - \lambda_1 + \frac{\lambda_1}{c} \log \frac{\lambda_1}{c} - \frac{\lambda_1}{c} + 2$. We aim at maximizing $c$, so

$$
    \frac{\partial}{\partial \lambda_1} F(\lambda_1, c) = 0.
$$

This equation has a unique solution: $\log \lambda_1 = \frac{\log c}{c+1}$. Therefore we have

$$
    c^{1/(c+1)} (1 + \frac{1}{c}) \geq 2 - \Omega_{vnd}(A^*).
$$

Define $f(c) = c^{1/(c+1)}(1 + \frac{1}{c})$. Its derivative is: $f'(c) = -\frac{\log c}{c(c+1)} c^{1/(c+1)}$. Analyzing $f'(c)$, we know that $f(c)$ increases on $(0, 1]$, decreases on $[1, \infty)$, and $f(1) = 2$. Also we have the following limits:

$$
    \lim_{c \to 0} f(c) = 0, \quad \lim_{c \to \infty} f(c) = 1.
$$

We denote the inverse function of $f(\cdot)$ on $[1, \infty)$ as $f^{-1}(\cdot)$. Then for any $\Omega_{vnd}(A^*) < 1$, we have

$$
    \text{cond}(A^\top A^*) \leq f^{-1}(2 - \Omega_{vnd}(A^*)).$$
Next we prove the result for the LDD regularizer $\Omega_{ldd}(A^*)$. Define a scalar function $s(x) = x - \log x - 1$ and denote $\text{cond}(A^*A^{*\top}) = c$. Since $s'(x) = 1 - \frac{1}{x}$ and $s(1) = 0$, we have

$$\Omega_{ldd}(A^*) = \sum_{j=1}^{R} s(\lambda_j) \geq s(\lambda_1) + s(\lambda_R) = \lambda_1 - \log \lambda_1 + \frac{\lambda_1}{c} - \log \frac{\lambda_1}{c} - 2.$$ 

Therefore we have

$$\log c \leq \Omega_{ldd}(A^*) + 2 \log \lambda_1 - \lambda_1 (1 + \frac{1}{c}) + 2 \leq \Omega_{ldd}(A^*) + 2 \log \lambda_1 - \lambda_1 + 2 \leq \Omega_{ldd}(A^*) + 2 \log 2 - 2 + 2 = \Omega_{ldd}(A^*) + 2 \log 2.$$

The third inequality is obtained from the following fact: the scalar function $\log x - x$ gets its maximum when $x = 2$. Further, we have

$$c \leq 4 e^{\Omega_{ldd}(A^*)}.$$

The proof completes.

4.3.2 Proof of Theorem 2

A well established result in learning theory is that the estimation error can be upper bounded by the Rademacher complexity [32]. We start from the Rademacher complexity, seek a further upper bound of it, and show how the diversity of the hidden units affects this upper bound. The Rademacher complexity $R_n(\ell \circ F)$ of the function class $F$ composed with the loss function $\ell$ is defined as

$$R_n(\ell \circ F) = E \sup_{f \in F} \left| \frac{1}{n} \sum_{i=1}^{n} \sigma_i \ell(f(x^{(i)}), y^{(i)}) \right|, \quad (4.17)$$

where $\{\sigma_i\}$ are independent and uniform over $\{-1, 1\}$ and $\{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$ are i.i.d samples drawn from $p^*$. Note that for later convenience, we take the absolute value in the above definition.

We can use the Rademacher complexity to upper bound the estimation error, as shown in Lemma 12.

**Lemma 12** ([298, Theorem 26.5]). With probability at least $1 - \delta$:

$$L(\hat{f}) - L(f^*) \leq 4R_n(\ell \circ F) + 2\gamma \sqrt{\frac{2 \log(4/\delta)}{n}}, \quad (4.18)$$

for $\gamma \geq \sup_{x,y,f} |\ell(f(x), y)|$. 

123
Our analysis starts from this lemma and we seek a further upper bound on \( R_n(\ell \circ \mathcal{F}) \). The analysis needs an upper bound of the Rademacher complexity of the hypothesis set \( \mathcal{F} \), which is given in Lemma 13.

**Lemma 13.** Let \( R_n(\bar{\mathcal{F}}) \) denote the Rademacher complexity of the hypothesis set \( \bar{\mathcal{F}} = \{ f(x) = \sum_{j=1}^{m} \alpha_j h(w_j^T x) \mid \| \alpha \|_2 \leq B, \| w_j \|_2 \leq C, \| x \|_2 \leq 1 \} \), and \( h \) be \( L \)-Lipschitz, then

\[
R_n(\bar{\mathcal{F}}) \leq B(2CL + |h(0)|) \sqrt{\frac{m}{n}}.
\] (4.19)

**Proof.** The proof is analogous to that of [32, Theorem 18]. First, we have from [298, Lemma 26.11] that the Rademacher complexity of the class of linear functions \( \mathcal{L} = \{ x \mapsto w^T x : \| w \|_2 \leq C, w \in \mathbb{R}^d \} \) is

\[
R_n(\mathcal{L}) \leq \frac{C}{\sqrt{n}}.
\] (4.20)

Since composing with a 1-Lipschitz function \( h \) that vanishes at 0 can only increase the Rademacher complexity by a factor of two [216, Theorem 4.12], we have

\[
R_n((h \circ \mathcal{L})) \leq R_n((h - h(0)) \circ \mathcal{L}) + |h(0)|/\sqrt{n}
\] (4.21)

\[
\leq (2CL + |h(0)|)/\sqrt{n}.
\] (4.22)

Lastly, since taking the (absolute) convex hull of a set of functions does not increase the Rademacher complexity [32, Theorem 12.2], we have

\[
R_n(\bar{\mathcal{F}}) \leq R_n(\sqrt{mB} \cdot \text{absconv}(h \circ \mathcal{L})) \leq B(2CL + |h(0)|) \sqrt{\frac{m}{n}}.
\] (4.23)

We note that if instead we consider \( \mathcal{F}' = \{ f(x) = \sum_{j=1}^{m} \alpha_j h(w_j^T x) \mid \| \alpha \|_1 \leq B, \| w_j \|_1 \leq C, \| x \|_2 \leq 1 \} \), then a similar argument confirms that

\[
R_n(\mathcal{F}') \leq B(2CL + |h(0)|) \sqrt{\frac{2\ln(2d)}{n}}.
\]

In addition, we need the following uniform bound on the output of the network.

**Lemma 14.** For any \( \| x \|_2 \leq 1, \| \alpha \|_2 \leq B \) and \( L \)-Lipschitz continuous \( h \), we have

\[
\sup_{\| w_j \|_2 \leq C, \forall i \neq j, \| w_j^T w_{j'} \|_2 \leq \tau} \left| \sum_{j=1}^{m} \alpha_j h(w_j^T x) - h(0) \right| \leq BCL\sqrt{1 - \tau + m\tau} + \sqrt{mB|h(0)|}.
\] (4.24)

**Proof.** Indeed, let \( u_j = h(w_j^T x) - h(0) \). Then, \( |u_j| \leq L\| w_j^T x \| \), hence

\[
\left| \sum_{j=1}^{m} \alpha_j h(w_j^T x) - h(0) \right| \leq \| \alpha \|_2 \| u \|_2 \leq B\| W^T x \|_2 \leq B\| W \|_{sp},
\] (4.25)

(4.26)

(4.27)
where $W = [w_1, \ldots, w_m]$, and $\|W\|_{sp}$ is the spectral norm, which we bound as follows:

$$\|W\|_{sp}^2 = \max_{\|x\|_2 \leq 1} x^T W^T W x$$  \hspace{1cm} (4.28)

$$\leq \max_{\|x\|_2 \leq 1} \sum_{i,j} |x_i||x_j||w_i^T w_j|$$  \hspace{1cm} (4.29)

$$\leq \max_{\|x\|_2 \leq 1} (1 - \tau) \sum_i x_i^2 \|w_i\|^2 + \tau \sum_{i,j} |x_i||x_j|\|w_i\|_2\|w_j\|_2$$  \hspace{1cm} (4.30)

$$\leq C^2[(1 - \tau) + m\tau].$$  \hspace{1cm} (4.31)

Plugging the bound on $\|W\|_{sp}$ into (4.27) and applying the triangle inequality completes the proof.

We are now ready to prove Theorem 2. The square loss function $\ell_y(t) = \ell(t, y) = \frac{1}{2}(t - y)^2$ is Lipschitz continuous with constant

$$\gamma = \sup_{\|x\|_2 \leq 1, |y| \leq 1, f \in F} |f(x) - y| \leq \sup_{x,y,f} (|f(x)| + |y|)$$  \hspace{1cm} (4.32)

$$\leq 1 + BCL\sqrt{1 - \tau + m\tau} + \sqrt{mB}|h(0)|,$$  \hspace{1cm} (4.33)

where the last inequality follows from Lemma 14. Applying the contraction property [298, Lemma 26.9] of the Rademacher complexity, we have

$$R_n(\ell \circ F) \leq (1 + BCL\sqrt{1 - \tau + m\tau} + \sqrt{mB}|h(0)|)R_n(F)$$  \hspace{1cm} (4.34)

$$\leq (1 + BCL\sqrt{1 - \tau + m\tau} + \sqrt{mB}|h(0)|)R_n(\bar{F}).$$  \hspace{1cm} (4.35)

Using Lemma 13 and Lemma 12 the claim follows.

In our proof we bound the Rademacher complexity of $F$ using that of $\bar{F}$ which simply ignores the pairwise correlation constraints. This step is likely very loose and it would be interesting to refine the bound so that it enjoys a reasonable dependence on $\tau$.

### 4.3.3 Proof of Theorem 3

We need the following approximation error bound due to [31]:

**Lemma 15 ([31, Theorem 3]).** Consider the function class

$$\bar{F} = \{x \mapsto \sum_{j=1}^m \alpha_j h(\bar{w}_j^T x) \mid \|\alpha\|_2 \leq B, \|\bar{w}_j\|_2 \leq C, \|x\|_2 \leq 1\};$$  \hspace{1cm} (4.36)

where the activation function $h(t) = 1/(1 + e^{-t})$. Then, for any square integratable function $g$ with

$$\int_\omega \|\omega\|_2 |\bar{g}(\omega)| d\omega \leq B/2,$$

where $\bar{g}(\omega)$ is the Fourier representation of $g(x)$, there exists $\bar{f} \in \bar{F}$ such that

$$\|g - \bar{f}\|_{L^2} \leq B\left(\frac{1}{\sqrt{m}} + \frac{1 + 2\ln C}{C}\right).$$  \hspace{1cm} (4.37)
We remark that [31] used instead the $\ell_1$ constraint $\|\alpha\|_1 \leq B$ in the definition of the function class $\tilde{F}$. For our purpose the $\ell_2$ constraint on $\alpha$ is more natural, and it is clear that we can only increase the function class $\tilde{F}$ by using the $\ell_2$ constraint. As $C$ and $m$ tend to infinity, the above bound indicates that any square integratable function $g$ can be approximated (under the $L_2$ metric) by the function class $\tilde{F}$. This is the well-known universal approximation property of neural networks. Moreover, if we choose $C \geq \sqrt{m \ln m}$, then the approximation error decreases at the usual rate $1/\sqrt{m}$.

Our key idea is to approximate the function class $\tilde{F}$ by the function class $F$, where the pairwise angle constraint $\theta_{jk} \geq \theta$ is enforced. Of course, this is possible only when $m$ is not too large.

**Lemma 16.** Let $\theta = \arccos(\tau) \in (0, \frac{\pi}{2})$ and $m \leq 2(\lfloor \frac{\pi}{2} - \theta \rfloor + 1)$. For any unit vectors $(\tilde{w}_j)_{j=1}^m$, there exist unit vectors $(w_j)_{j=1}^m$ such that

\begin{align}
\forall j \neq k \in \{1, \ldots, m\}, \quad &|w_j \cdot w_k| \leq \tau, \quad (4.38) \\
\forall j \in \{1, \ldots, m\}, \quad &w_j \cdot \tilde{w}_j \geq \cos[\min(3m\theta, \pi)]. \quad (4.39)
\end{align}

Assuming Lemma 16 holds for now we can proceed to prove Theorem 3. According to Lemma 15, there exists some $\tilde{f} = \sum_{j=1}^m \alpha_j h(\tilde{w}_j^\top x) \in \tilde{F}$ such that

$$\|g - \tilde{f}\|_{L^2} \leq B \left( \frac{1}{\sqrt{m}} + \frac{1 + 2 \ln C}{C} \right). \quad (4.40)$$

Since

$$\|g - f\|_{L^2} \leq \|g - \tilde{f}\|_{L^2} + \|\tilde{f} - f\|_{L^2}, \quad (4.41)$$

we need only bound the last term as follows. Let us define $f = \sum_{j=1}^m \alpha_j h(w_j^\top x)$, where $(w_j)_{j=1}^m$ are given in Lemma 16 and scaled so that $\|w_j\|_2 = \|\tilde{w}_j\|_2$. Straightforward calculation shows:

\begin{align}
\|f - \tilde{f}\|_{L^2}^2 &= \int_{\|x\|_2 \leq 1} (f(x) - \tilde{f}(x))^2 d\mu(x) \quad (4.42) \\
&= \int_{\|x\|_2 \leq 1} \left( \sum_j \alpha_j \left( h(w_j^\top x) - h(\tilde{w}_j^\top x) \right) \right)^2 d\mu(x) \quad (4.43) \\
&\leq \int_{\|x\|_2 \leq 1} \|\alpha\|_2^2 \sum_j |w_j^\top x - \tilde{w}_j^\top x|^2 d\mu(x) \quad (4.44) \\
&\leq B^2 \sum_j \|w_j - \tilde{w}_j\|_2^2 \quad (4.45) \\
&= 2B^2 \sum_j \|w_j\|_2^2 - \tilde{w}_j^\top \tilde{w}_j \quad (4.46) \\
&\leq 2B^2 (1 - \cos[\min(3m\theta, \pi)]) \sum_j \|w_j\|_2^2 \quad (4.47) \\
&\leq 4mB^2 C^2 \sin^2 \frac{\min(3m\theta, \pi)}{2}. \quad (4.48)
\end{align}
Substituting back to (4.41) we have
\[
\|g - f\|_{L^2} \leq \|g - \bar{f}\|_{L^2} + \|\bar{f} - f\|_{L^2} \\
\leq B(\frac{1}{\sqrt{m}} + \frac{1 + 2\ln C}{C}) + 2\sqrt{mBC} \sin \frac{\min(3m\theta, \pi)}{2},
\]
which was to be proved.

**Proof of Lemma 16**

The following simple lemma will be useful in our proof.

**Lemma 17.** For any three unit vectors \(u_1, u_2, \) and \(u_3:\)
\[
\angle(u_1, u_3) \leq \angle(u_1, u_2) + \angle(u_2, u_3),
\]
with the convention \(\angle(x, y) \in [0, \pi).\)

**Proof.** Without loss of generality, let \(u_2 = e_1.\) Let \(\theta_{12} = \angle(u_1, u_2), \theta_{23} = \angle(u_2, u_3), \) and \(\theta_{13} = \angle(u_1, u_3).\) Then, \(u_1 = \cos \theta_{12} \cdot u_2 + \sin \theta_{12} \cdot v\) where the first element of the unit vector \(v\) equals 0. Similarly, \(u_3 = \cos \theta_{23} \cdot u_2 + \sin \theta_{23} \cdot w\) where the first element in the unit vector \(w\) equals 0. Therefore,
\[
\cos \theta_{13} = u_1^\top u_3 \\
= (\cos \theta_{12} \cdot u_2 + \sin \theta_{12} \cdot v)^\top (\cos \theta_{23} \cdot u_2 + \sin \theta_{23} \cdot w) \\
= \cos \theta_{12} \cos \theta_{23} + \sin \theta_{12} \sin \theta_{23} v^\top w \\
\geq \cos \theta_{12} \cos \theta_{23} - \sin \theta_{12} \sin \theta_{23} \\
= \cos(\theta_{12} + \theta_{23}).
\]

If \(\theta_{12} + \theta_{23} < \pi,\) \(\arccos(\cos(\theta_{12} + \theta_{23})) = \theta_{12} + \theta_{23}.\) As \(\arccos(\cdot)\) is monotonically decreasing, we have \(\theta_{13} \leq \theta_{12} + \theta_{23}.\) Otherwise, \(\theta_{13} < \pi \leq \theta_{12} + \theta_{23}.\) \(\square\)

We now begin to prove Lemma 16. Let \(\phi(a, b) = \arccos(\frac{a \cdot b}{\|a\|\|b\|}),\) \(\rho(a, b) = \arccos(\frac{|a \cdot b|}{\|a\|\|b\|}).\)

We begin our proof by considering a 2-dimensional case \((d = 2).\) Let \(k = \lfloor \frac{\pi - \theta}{\theta} \rfloor.\) Define an index set \(I = \{-k+1, -k, \cdots, -1, 1, 2, \cdots, k + 1\}.\) We define a set of vectors \((e_i)_{i \in I}:\)
\[
e_i = (\sin \theta_i, \cos \theta_i),\] where \(\theta_i \in (-\frac{\pi}{2}, \frac{\pi}{2})\) is defined as follows:
\[
\theta_i = \text{sgn}(i)(\frac{\theta}{2} + (|i| - 1)\theta). 
\]

From the definition we have:
\[
\forall i \neq j \in I, \rho(e_i, e_j) \geq \theta, \\
-\frac{\pi}{2} + \frac{\theta}{2} \leq \theta_{-k+1} < \frac{\pi}{2} + \frac{\theta}{2}, \\
\frac{\pi}{2} - \frac{3\theta}{2} < \theta_{k+1} \leq \frac{\pi}{2} - \frac{\theta}{2}. 
\]

And we can further verify that \(\forall i \in I,\) there exist different \(i_1, \cdots, i_{2k+1} \in I \setminus i\) such that \(\phi(e_i, e_{i_j}) \leq j\theta.\) For any \(e = (\sin \beta, \cos \beta)\) with \(\beta \in [-\frac{\pi}{2}, \frac{\pi}{2}],\) we can find an \(i \in I\) such that \(\phi(e_i, e) \leq \frac{3}{2}\theta:\)

127
• If $\beta \geq \theta_{k+1}$, taking $i = k + 1$, we have $\phi(e_i, e) \leq \frac{\pi}{2} - \theta_{k+1} < \frac{3}{2}\theta$.

• If $\beta \leq \theta_{-(k+1)}$, taking $i = -(k + 1)$, we also have $\phi(e_i, e) \leq \frac{3}{2}\theta$.

• Otherwise, taking $i = \text{sgn}(\beta)[\frac{\beta - \theta}{2\theta}]$, we also have $\phi(e_i, e) \leq \theta < \frac{3}{2}\theta$.

Recall that for any $i$, there exist different $i_1, \cdots, i_{2k+1} \in I \setminus i$ such that $\phi(e_i, e_{i_j}) \leq j \theta$, and using Lemma 17, we can draw the conclusion that for any $e = (\sin \beta, \cos \beta)$ with $\beta \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$, there exist different $i_1, \cdots, i_{2k+2}$ such that $\phi(e_i, e_{i_j}) \leq \frac{3}{2}\theta + (j - 1)\theta = (j + \frac{1}{2})\theta$. For any $(w^j_j)_{j=1}^m$, assume $w_j^{'} = \|w_j^0\|_2\sin \beta_j, \cos \beta_j)$, and assume $\beta_j \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$. Using the above conclusion, for $w_j^0$, we can find some $r_1$ such that $\phi(w_j^0, e_{r_1}) \leq \frac{3}{2}\theta$. For $w_j^0$, we can find different $i_1, i_2$ such that $\phi(w_j^0, e_{i_1}) \leq \frac{3}{2}\theta < (\frac{3}{2} + 1)\theta$ and $\phi(w_j^0, e_{i_2}) \leq (\frac{3}{2} + 1)\theta$. So we can find $r_2 \neq r_1$ such that $\phi(w_j^0, e_{r_2}) \leq (\frac{3}{2} + 1)\theta$. Following this scheme, we can find $r_j \notin \{r_1, \cdots, r_{j-1}\}$ and $\phi(w_j^0, e_{r_j}) \leq (j + \frac{1}{2})\theta < 3m\theta$ for $j = 1, \cdots, m$, as we have assumed that $m \leq (k + 1)$. Let $w_j = \|w_j^0\|_2 e_{r_j}$, then we have constructed $(w_j)_{j=1}^m$ such that

$$\forall j \in \{1, \cdots, m\}, \phi(w_j^0, w_j) \leq 3m\theta, \quad (4.59)$$

$$\forall j \in \{1, \cdots, m\}, \|w_j^0\|_2 = \|w_j\|_2, \quad (4.60)$$

$$\forall j \neq k, \rho(w_j, w_k) \geq \theta. \quad (4.61)$$

Note that we have assumed $\forall j = 1, \cdots, m, \beta_j \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$. In order to show that the conclusion holds for general $w_j^0$, we need to consider the cases where $\beta_j \in \left[-\frac{3}{2}\pi, -\frac{\pi}{2}\right]$. For those cases, we can let $\beta_j = \beta_j + \pi$, then $\beta_j \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$. Let $w_j'' = \|w_j''\|_2\sin \beta_j', \cos \beta_j')$, we can find an $e_{r_j}$ such that $\phi(w_j'', e_{r_j}) \leq m\theta$ following the same procedure. Let $w_j = -\|w_j''\|_2 e_{r_j}$, then $\phi(w_j, w_j) = \phi(w_j'', e_{r_j}) \leq 2m\theta$. Since $\rho(-e_{r_j}, e_k) = \rho(e_{r_j}, e_k)$, the $\rho(w_j, w_k) \geq \theta$ condition is still satisfied. Also noting that $\phi(a, b) \leq \pi$ for any $a, b$, we complete the proof for the 2-dimensional case.

Now we consider a general $d$-dimensional case. Similar to the 2-dimensional case, we construct a set of vectors with unit $L_2$ norm such that the pairwise angles $\rho(w_j, w_k) \geq \theta$ for $j \neq k$.

We perform the construction in two phases. In the first phase, we construct a sequence of unit vector sets indexed by $I = \{-(k + 1), \cdots, 1, 1, \cdots, k + 1\}$:

$$\forall i \in I, \mathcal{E}_i = \{e \in \mathbb{R}^d ||e||_2 = 1, e \cdot (1, 0, \cdots, 0) = \cos \theta_i\}, \quad (4.62)$$

where $\theta_i = \text{sgn}(i)(\frac{\theta}{2} + (|i| - 1)\theta)$ is defined in Eq. (4.57). It can be shown that $\forall i \neq j, \forall e_i \in \mathcal{E}_i, e_j \in \mathcal{E}_j$, $\rho(e_i, e_j) \geq \theta$. \quad (4.63)

The proof is as follows. First, we write $e_i$ as $e_i = (\cos \theta_i, 0, \cdots, 0) + r_i$, where $\|r_i\|_2 = |\sin \theta_i|$. Similarly, $e_j = (\cos \theta_j, 0, \cdots, 0) + r_j$, where $\|r_j\|_2 = |\sin \theta_j|$. Hence we have

$$e_i \cdot e_j = \cos \theta_i \cos \theta_j + r_i \cdot r_j, \quad (4.64)$$

Hence

$$\cos(\rho(e_i, e_j)) = |e_i \cdot e_j| \quad (4.65)$$

$$\leq \cos \theta_i \cos \theta_j + |\sin \theta_i \sin \theta_j| \quad (4.66)$$

$$= \max(\cos(\theta_i + \theta_j), \cos(\theta_i - \theta_j)). \quad (4.67)$$
We have shown in the 2-dimensional case that \( \cos(\theta_i + \theta_j) \geq \cos \theta \) and \( \cos(\theta_i - \theta_j) \geq \cos \theta \), hence \( \rho(e_i, e_j) \geq \theta \). In other words, we have proved that for any two vectors from \( E_i \) and \( E_j \), their pairwise angle is lower bounded by \( \theta \). Now we proceed to construct a set of vectors for each \( E_i \) such that the pairwise angles can also be lower bounded by \( \theta \). The construction is as follows.

First, we claim that for any \( E_i \), if \( \mathcal{W} \subset E_i \) satisfies \( \forall w_j \neq w_k \in \mathcal{W}, \phi(w_j, w_k) \geq \theta \), then \( |\mathcal{W}| \) is finite. In order to prove that, we first define \( B(x, r) = \{ y \in \mathbb{R}^n : ||y - x||_2 < r \} \). Then \( E_i \subset \bigcup_{e \in E_i} B(e, \frac{1 - \cos \theta}{2}) \). According to the definition of \( E_i \), it is a compact set. Therefore the open cover has a finite subcover. Therefore we have \( \exists V \subset E_i \) with \( |V| \) being finite and

\[
E_i \subset \bigcup_{v \in V} B \left( v, \frac{1 - \cos \theta}{1 + \cos \frac{\theta}{2}} \right). \tag{4.69}
\]

Furthermore, we can verify that \( \forall v \in V, \forall e_1, e_2 \in B(v, \frac{1 - \cos \theta}{1 + \cos \frac{\theta}{2}}), \phi(e_1, e_2) \leq \theta \). So if \( \mathcal{W} \subset E_i \) satisfies \( \forall w_j \neq w_k \in \mathcal{W}, \phi(w_j, w_k) \geq \theta \), then for each \( v \in V \), \( |B(v, \frac{1 - \cos \theta}{1 + \cos \frac{\theta}{2}}) \cap \mathcal{W}| = 1 \). As \( \mathcal{W} \subset E_i \), we have

\[
|\mathcal{W}| = |\mathcal{W} \cap E_i| \tag{4.70}
\]

\[
= \left| \mathcal{W} \cap \left( \bigcup_{v \in V} B \left( v, \frac{1 - \cos \theta}{1 + \cos \frac{\theta}{2}} \right) \right) \right| \tag{4.71}
\]

\[
= \left| \bigcup_{v \in V} \mathcal{W} \cap B \left( v, \frac{1 - \cos \theta}{1 + \cos \frac{\theta}{2}} \right) \right| \tag{4.72}
\]

\[
\leq \sum_{v \in V} \left| \mathcal{W} \cap B \left( v, \frac{1 - \cos \theta}{1 + \cos \frac{\theta}{2}} \right) \right| \tag{4.73}
\]

\[
\leq \sum_{v \in V} 1 \tag{4.74}
\]

\[
= |V|. \tag{4.75}
\]

Therefore, we have proved that \( |\mathcal{W}| \) is finite. Using this fact, we can construct a sequence of vectors \( w_j \in E_i (j = 1, \ldots, l) \) in the following way:

1. Let \( w_1 \in E_i \) be any vector in \( E_i \).
2. For \( j \geq 2 \), let \( w_j \in E_i \) be any vector satisfying

\[
\forall k = 1, \ldots, j - 1, \phi(w_j, w_k) \geq \theta, \tag{4.76}
\]

\[
\exists k \in \{1, \ldots, j - 1\}, \phi(w_j, w_k) = \theta, \tag{4.77}
\]

until we cannot find such vectors any more.

3. As we have proved that \( |\mathcal{W}| \) is finite, the above process will end in finite steps. Assume that the last vector we found is indexed by \( l \).
We will verify that such constructed vectors satisfy
\[ \forall j \neq k \in \{1, \cdots, l\}, \rho(w_j, w_k) \geq \theta. \tag{4.78} \]

According to the construction, \( \phi(w_j, w_k) \geq \theta \). As \( \rho(w_j, w_k) = \min(\phi(w_j, w_k), \pi - \phi(w_j, w_k)) \), we only need to show that \( \pi - \phi(w_j, w_k) \geq \theta \). To show this is true, we use the definition of \( E_i \) to write \( w_j \) as \( w_j = (\cos \theta_i, 0, \cdots, 0) + r_j \), where \( \|r_j\|_2 = |\sin \theta_i| \). Similarly, \( w_k = (\cos \theta_i, 0, \cdots, 0) + r_k \), where \( \|r_k\|_2 = |\sin \theta_i| \). Therefore \( \cos(\phi(w_j, w_k)) = w_j \cdot w_k \geq \cos^2 \theta_i - \sin^2 \theta_i = \cos(2\theta_i) \geq \cos(\pi - \theta) \), where the last inequality follows from the construction of \( \theta_i \). So \( \pi - \phi(w_j, w_k) \geq \theta \), the proof for \( \rho(w_j, w_k) \geq \theta \) is completed.

Now we will show that \( \forall e \in E_i \), we can find \( j \in \{1, \cdots, l\} \) such that \( \phi(e, w_j) \leq \theta \). We prove it by contradiction: assume that there exists \( e \) such that \( \min_{j \in \{1, \cdots, l\}} \phi(e, w_j) > \theta \), then as \( E_j \) is a connected set, there is a path \( q : t \in [0, 1] \rightarrow E_j \) connecting \( e \) to \( w_1 \). When \( t = 0 \), the path starts at \( q(0) = e \); when \( t = 1 \), the path ends at \( q(1) = w_1 \). We define functions \( r_j(t) = \phi(q(t), w_j) \) for \( t \in [0, 1] \) and \( j = 1, \cdots, l \). It is straightforward to see that \( r_j(t) \) is continuous. As \( \min_j(r_j(0)) \geq \theta \) and \( \min_j(r_j(1)) = 0 < \theta \), there exists \( t^* \in (0, 1) \) such that \( \min_j(r_j(t^*)) = \theta \). Then \( q(t^*) \) satisfies Condition 4.76, which contradicts the construction in \( W \) as the construction only ends when we cannot find such vectors. Hence we have proved that
\[ \forall e \in E_i, \exists j \in \{1, \cdots, l\}, \phi(e, w_j) \leq \theta. \tag{4.79} \]

Now we can proceed to prove the main lemma. For each \( i \in I \), we use Condition 4.76 to construct a sequence of vectors \( w_{ij} \). Then such constructed vectors have pairwise angles greater than or equal to \( \theta \). Then for any \( e \in \mathbb{R}^d \) with \( \|e\|_2 = 1 \), we write \( e \) in sphere coordinates as \( e = (\cos r_1, \sin r_1 \cos r_2, \cdots, \prod_{j=1}^d \sin r_j) \). Use the same method as we did for the 2-dimensional case, we can find \( \theta_i \) such that \( |\theta_i - r_i| \leq \frac{3}{2} \theta \). Then \( e' = (\cos \theta_i, \sin \theta_i \cos r_2, \cdots, \prod_{j=2}^d \sin r_j) \in E_i \). It is easy to verify that \( \phi(e, e') = |\theta_i - r| \leq \frac{3}{2} \theta \). As \( e' \in E_i \), there exists \( w_{ij} \) as we constructed such that \( \phi(e', w_{ij}) \leq \theta \). So \( \phi(e, w_{ij}) \leq \phi(e, e') + \phi(e', w_{ij}) \leq \frac{5}{2} \theta < 3 \theta \). So we have proved that for any \( e \in \mathbb{R}^d \) with \( \|e\|_2 = 1 \), we can find \( w_{ij} \) such that \( \phi(e, w_{ij}) < 3 \theta \).

For any \( w_{ij} \), assume \( i + 1 \in I \), we first project \( w_{ij} \) to \( w^* \in E_{i+1} \). We have proved that \( \phi(w_{ij}, w^*) \leq \frac{3}{2} \theta \) and also proved that we can find \( w_{i+1,j'} \in E_{i+1} \) such that \( \phi(w_{i+1,j'}, w^*) \leq \theta \). So we have found \( w_{i+1,j'} \) such that \( \phi(w_{ij}, w_{i+1,j'}) \leq \frac{3}{2} \theta < 3 \theta \). We can use similar scheme to prove that \( \forall w_{ij} \), there exist different \( w_{i_1,j_1} \cdots, w_{i_{2k+1},j_{2k+1}} \) such that \( (i_1, j_1) \neq (i, j) \) and \( \phi(w_{ij}, w_{i,j'}) \leq 3r \theta \). Following the same proof as the 2-dimensional case, we can prove that if \( m \leq 2k + 1 \), then we can find a set of vectors \( (w_j)_{j=1}^m \) such that
\[ \forall j \in \{1, \cdots, m\}, \phi(w_j, w_j) \leq \min(3m \theta, \pi), \tag{4.80} \]
\[ \forall j \in \{1, \cdots, m\}, \|w_j\|_2 = \|w_j\|_2, \tag{4.81} \]
\[ \forall j \neq k, \rho(w_j, w_k) \geq \theta. \tag{4.82} \]

The proof completes.
4.3.4 Proof of Theorem 4

Proof Sketch

Define \( \mathcal{W} = \{ W \in \mathbb{R}^{d \times m} | \Omega(W) \leq \tau \} \) and \( \tilde{C}(\mathcal{W}) = \sup_{W \in \mathcal{W}} C(W) \). The following lemma shows that the estimation error can be upper bounded using \( \tilde{C}(\mathcal{W}) \).

**Lemma 18.** Suppose \( \sup_{(x,y)} \| x - y \|_2 \leq B_0 \) and \( \| w \|_2 \leq D \), then with probability at least \( 1 - \delta \), we have

\[
L(u) - \hat{L}(u) \leq 4B_0^2 D \sqrt{\frac{\tilde{C}(\mathcal{W})+1}{m}} + [B_0^2(\tilde{C}(W) + m) + 1] \sqrt{\frac{2 \log(1/\delta)}{m}}. \tag{4.83}
\]

The upper bound is an increasing function of \( \tilde{C}(\mathcal{W}) \). The following lemma shows that \( C(W) \) can be upper bounded by an increasing function of the LDD regularizer \( \Omega_{\text{ldd}}(W) \).

**Lemma 19.** Let \( g(x) = x - \log(x+1) \). Then we have

\[
C(W) \leq g^{-1}(\Omega_{\text{ldd}}(W)/m)m.
\]

where \( g^{-1}(\cdot) \) is the inverse function of \( g \) on \([0, \infty)\) and is an increasing function.

Since \( \Omega_{\text{ldd}}(W) \leq \tau \), we have \( C(W) \leq g^{-1}(\tau/m)m \) for any \( W \in \mathcal{W} \), i.e., \( \tilde{C}(\mathcal{W}) \leq g^{-1}(\tau/m)m \). Substituting this inequality into Lemma 18, we obtain the bound in Eq.(4.5) in Theorem 4.

**Proof of Lemma 18**

*Proof.* Let \( \mathcal{U} = \{ u : (x, y) \rightarrow \| W(x - y) \|^2_2 \} \) be the set of hypothesis \( u(x, y) = \| W(x - y) \|^2_2 \), and \( \mathcal{R}(\mathcal{U}) \) be the Rademacher complexity \([32]\) of \( \mathcal{U} \) which is defined as:

\[
\mathcal{R}(\mathcal{U}) = \mathbb{E}_{S_N, \sigma} \sup_{u \in \mathcal{U}} \frac{1}{N} \sum_{n=1}^{N} \sigma_n \| W(x_n - y_n) \|^2_2,
\]

where \( S_N = ((x_1, y_1, t_1), (x_2, y_2, t_2) \cdots (x_N, y_N, t_N)) \) are the training examples, \( \sigma_n \in \{-1, 1\} \) are the Rademacher variables, and \( \sigma = (\sigma_1, \sigma_2, \cdots, \sigma_N) \).

Lemma 20 shows that the estimation error can be bounded using the Rademacher complexity. Its proof is adapted from \([32]\). Readers only need to notice \( x + 1 \) is an upper bound of \( \log(1 + \exp(x)) \) for \( x > 0 \).

**Lemma 20.** With probability at least \( 1 - \delta \), we have

\[
L(u) - \hat{L}(u) \leq 2 \mathcal{R}(\mathcal{U}) + \sup_{x,y,W \in \mathcal{W}} (\| W'(x - y) \|^2_2 + 1) \sqrt{\frac{2 \log(1/\delta)}{N}}. \tag{4.84}
\]

We then bound \( \mathcal{R}(\mathcal{U}) \) and \( \sup_{x,y,W \in \mathcal{W}} \| W'(x - y) \|^2_2 \). The result is in the following lemma.

**Lemma 21.** Suppose \( \sup_{(x,y)} \| x - y \|_2 \leq B_0 \) and \( \| w \|_2 \leq D \), then we have

\[
\mathcal{R}(\mathcal{U}) \leq \frac{2B_0^2 D \sqrt{m}}{\sqrt{N}} \sqrt{\tilde{C}(\mathcal{W})} + 1,
\]

131
and 
\[ \sup_{x,y,w \in W} \| W'(x - y) \|_2^2 \leq B_0^2(\tilde{C}(W) + m). \]

**Proof.** We first give bound on \( R(U) \). Let \( R(S) = \{ s : (x, y) \rightarrow \sum_{j=1}^m |\langle w_j, x - y \rangle|, W \in W \} \) be the set of hypothesis \( s(x, y) = \sum_{n=1}^N \sum_{j=1}^m |\langle w_j, x_n - y_n \rangle| = \langle w_j, a_{n,j}(x_n - y_n) \rangle \), where \( a_{n,j} \in \{-1, 1\} \). Then

\[ R(S) = E_{S_N, \sigma} \sup_{W \in W} \frac{1}{N} \sum_{n=1}^N \sum_{j=1}^m \langle w_j, a_{n,j}(x_n - y_n) \rangle. \]  

We first bound \( R(S) \).

\[ R(S) = E_{S_N, \sigma} \sup_{W \in W} \frac{1}{N} \sum_{j=1}^m \sum_{n=1}^N \sigma_n a_{n,j}(x_n - y_n) \]

\[ = E_{S_N, \sigma} \sup_{W \in W} \frac{1}{N} \langle \sum_{j=1}^m w_j, \sum_{n=1}^N \sigma_n a_{n,j}(x_n - y_n) \rangle \]

\[ \leq E_{S_N, \sigma} \sup_{W \in W} \frac{1}{N} \| \sum_{j=1}^m w_j \|_2 \| \sum_{n=1}^N \sigma_n a_{n,j}(x_n - y_n) \|_2 \]

\[ = E_{S_N, \sigma} \sup_{W \in W} \frac{1}{N} \left( \sum_{j=1}^m \| w_j \|_2 \right)^2 \left( \sum_{n=1}^N \| \sigma_n a_{n,j}(x_n - y_n) \|_2 \right)^2 \]

Applying Jensen’s inequality to Eq. (4.86), we have

\[ R(S) \leq E_{S_N} \sup_{W \in W} \frac{1}{N} \sum_{j,k=1}^m | \langle w_j, w_k \rangle | \left( \sum_{n=1}^N \sigma_n a_{n,j}(x_n - y_n) \right) \left( \sum_{n=1}^N \sigma_n a_{n,j}(x_n - y_n) \right) \]

Combining Eq. (4.87) with the inequality \( \sum_{j,k=1}^m | \langle w_j, w_k \rangle | - \delta_{j,k} \leq mC(W) \), we have

\[ R(S) \leq E_{S_N} \sup_{W \in W} \frac{1}{N} \sqrt{mC(W)} + m \sqrt{\sum_{n=1}^N \| x_n - y_n \|_2} \]

\[ \leq \sqrt{ \frac{m}{N} } \sup_{w \in W} (C(W) + 1) B_0. \]

Let \( w \) denote any column vector of \( W \in W \) and \( x \) denote any data example. According to the
composition property of Rademacher complexity (Theorem 12 in [32]), we have

\[ \mathcal{R}(U) \leq 2 \sup_{w, x} \mathcal{R}(S) \]
\[ \leq 2 \sup_{w} \|w\|_2 B_0 \mathcal{R}(S) \]
\[ \leq \frac{2B_0^2D\sqrt{m}}{\sqrt{N}} \sqrt{\bar{C}(W)} + 1. \]

Next we give bound on \( \sup_{x, y, W' \in W} \|W'(x - y)\|_2^2 \):

\[ \sup_{x, y, W' \in W} \|W'(x - y)\|_2^2 \leq \sup_{W' \in W} \sum_{j=1}^{m} \langle w'_j, w'_j \rangle \sup_{(x, y)} \|x - y\|_2^2 \]
\[ = \sup_{W' \in W} \text{tr}(W'^\top W') \sup_{(x, y)} \|x - y\|_2^2 \]
\[ \leq (\bar{C}(W) + m)B_0^2 \]

Combining Lemma 21 with Lemma 20, we complete the proof of Lemma 18.

Proof of Lemma 19

Proof. The function \( g(x) = x - \log(x + 1) \) is decreasing on \((-1, 0]\), increasing on \([0, \infty)\), \( g(0) = 0 \), and \( g(-t) > g(t) \) for \( \forall 0 \leq t < 1 \). We have

\[ \Omega_{\text{add}}(W) = \text{tr}(W^\top W) - \log \det(W^\top W) - m \]
\[ = \sum_{j=1}^{m} g(\pi_j - 1) \]
\[ \geq \sum_{j=1}^{m} g(|\pi_j - 1|) \]
\[ \geq g\left( \frac{1}{m} \sum_{j=1}^{m} |\pi_j - 1| \right) m \]
\[ = g(C(W)/m) m. \]

The first inequality is due to \( g(-t) > g(t) \), and the second inequality can be attained by Jensen’s inequality. Finally we have

\[ g(C(W)/m) m \leq \Omega_{\text{add}}(W). \]

Thus, we have

\[ C(W) \leq g^{-1}(\Omega_{\text{add}}(W)/m) m. \]
Extend the Analysis to the VND Regularizer

Next, we extend the estimation error analysis to the VND regularizer. The following lemma shows that $C(W)$ can be upper bounded by an increasing function of the VND regularizer $\Omega_{vnd}(W)$.

**Lemma 22.** Define a real function $g = (x + 1) \log(x + 1) - x$, which is increasing on $[0, \infty)$. Then we have:

$$C(W) \leq mg^{-1}(\Omega_{vnd}(W)/m),$$

where $g^{-1}$ is the inverse function of $g$.

**Proof.** $g$ is decreasing on $(-1, 0]$ and increasing on $[0, \infty)$. $g(0) = 0$, $g(-t) > g(t)$ for $\forall 0 \leq t < 1$. We have

$$\Omega_{vnd}(W) = \text{tr}(G \log G) - \text{tr}(G) + m$$

$$= \sum_{j=1}^{m} g(\pi_j - 1)$$

$$\geq \sum_{j=1}^{m} g(|\pi_j - 1|)$$

$$\geq mg\left(\frac{1}{m} \sum_{j=1}^{m} |\pi_j - 1|\right)$$

$$= mg(C(W)/m),$$

where $G = W^T W$ and $\{\pi_j\}_{j=1}^{m}$ are the eigenvalues of $G$. The first inequality is due to $g(-t) > g(t)$, and the second inequality can be attained by Jensen’s inequality. Finally we have

$$g(C(W)/m) \leq \Omega_{vnd}(W).$$

Thus, we have

$$C(W) \leq mg^{-1}(\Omega_{vnd}(W)/m).$$

Combining Lemma 22 with Lemma 18, we obtain the bound in Eq.(4.6) in Theorem 4.

**Analysis of the SFN Regularizer**

For SFN, the analysis approach is different. Instead of leveraging the intermediate capacity variable, we directly bound the estimation error with SFN. The following lemma gives an upper bound of the Rademacher complexity.

**Lemma 23.** Suppose $\sup_{(x, y)} \|x - y\| \leq B$, then we have

$$\mathcal{R}(\mathcal{U}) \leq \frac{2B^2}{\sqrt{N}} \left(\|W^T W - I\|_F + \sqrt{m}\right),$$

and

$$\sup_{x, y, W' \in \mathcal{W}} \|W'(x - y)\|_2^2 \leq \sup_{W' \in \mathcal{W}} (\sqrt{m}\|W'^T W - I\|_F + m) B^2.$$
Proof. We first prove the bound of $\mathcal{R}(U)$. Recall the definition of $\mathcal{R}(S)$ in Eq. (4.85). By the proof of Lemma 21, we have

$$
\mathcal{R}(S) \leq \frac{B}{\sqrt{N}} E_{S_N} \sup_{w \in \mathcal{W}} \sqrt{\sum_{j,k=1}^{m} |\langle w_j, w_k \rangle|},
$$

(4.88)

$$
\mathcal{R}(U) \leq 2 \sup_{w \in \mathcal{W}} \|w_j\|_2 B \mathcal{R}(S).
$$

(4.89)

By the properties of matrix norm, we have

$$
\sup_{W \in \mathcal{W}} \|W\|_F \leq \|W^T W - I\|_F + \sqrt{m}.
$$

Further, by Jensen’s inequality, we have

$$
\sup_{W \in \mathcal{W}} \|w_j\|_2 \leq \sqrt{\langle w_j, w_j \rangle} \leq \sqrt{\|W^T W\|_F} \leq \sqrt{\|W^T W - I\|_F + \sqrt{m}}.
$$

(4.91)

Combining Eq. (4.88) and Eq. (4.91), we have

$$
\mathcal{R}(U) \leq \frac{2B^2 \sqrt{m}}{\sqrt{N}} (\|W^T W - I\|_F + \sqrt{m}).
$$

We then drive the bound on $\sup_{x,y,w \in \mathcal{W}} \|W'(x - y)\|_2^2$.

$$
\sup_{x,y,w \in \mathcal{W}} \|W'(x - y)\|_2^2 \leq \sup_{w \in \mathcal{W}} \text{tr}(W^T W) \sup_{(x,y)} \|x - y\|^2 \\
\leq \sqrt{m} \|W^T W\|_F^2 B^2 \\
\leq (\sqrt{m} \|W^T W - I\|_F + m) B^2
$$

Combining Lemma 23 with Lemma 20 and noting that $\|W - I\|_F \leq \tau$, we complete the proof of the estimation error bound w.r.t SFN given in Eq. (4.7).

### 4.3.5 Proof of Theorem 5

**Proof Sketch**

Part of the proof is tailored to the CVND regularizer. Extensions to CSFN and CLDD are given later. The proof is based on the Rademacher complexity (RC) [32], which measures the complexity of a hypothesis class. The Rademacher complexity $\mathcal{R}(\mathcal{M})$ of the function class $\mathcal{M}$ is defined as:
\( \mathcal{R}(\mathcal{M}) = \mathbb{E}_{S,D,\sigma} \max_{M \in \mathcal{M}} \frac{1}{m} \sum_{i=1}^{m} \sigma_i (x_i - y_i) \top M (x_i - y_i), \)

where \( m \) is the number of data pairs in the training data (\( m = |S| + |D| \)), \( \sigma_i \in \{-1, 1\} \) is the Rademacher variable, and \( \sigma = (\sigma_1, \sigma_2, \cdots, \sigma_m) \).

We first establish an upper bound of the estimation error based on RC. Intuitively, a less-complicated hypothesis class generalizes better on unseen data. Then we upper bound the RC based on the CBMD regularizers. Combining the two steps together, we establish upper bounds of the estimation error based on the CBMD regularizers. The following lemma presents the RC-based upper bound of the estimation error. Its proof is adapted from [32].

**Lemma 24.** With probability at least \( 1 - \delta \), we have

\[
\sup_{M \in \mathcal{M}} (L(M) - \hat{L}(M)) \leq 2 \mathcal{R}(\mathcal{M}) + \max(\tau, \sup_{(x,y) \in S} (x - y) \top M (x - y)) \sqrt{\frac{2 \log(1/\delta)}{m}}. \tag{4.92}
\]

For the second term in the bound, it is easy to verify

\[
\sup_{(x,y) \in S} (x - y) \top M (x - y) \leq \sup_{M \in \mathcal{M}} \sup_{(x,y) \in S} \|x - y\|_2^2. \tag{4.93}
\]

Now we focus on the first term. We denote \( z = x - y \), \( z_i = x_i - y_i \).

**Lemma 25.** Suppose \( \sup_{\|v\|_2 \leq 1, z} |v \top z| \leq B \), then we have

\[
\mathcal{R}(\mathcal{M}) \leq \frac{2B^2}{\sqrt{m}} \sup_{M \in \mathcal{M}} \text{tr}(M). \tag{4.94}
\]

We next show that \( \text{tr}(M) \) can be bounded by the CVND regularizer \( \tilde{\Omega}_{vnd}(M) \).

**Lemma 26.** For the convex VND regularizer \( \tilde{\Omega}_{vnd}(M) \), for any positive semidefinite matrix \( M \), we have

\[
\text{tr}(M) \leq \tilde{\Omega}_{vnd}(M).
\]

Combining Lemma 24, 25, 26 and Eq. (4.93) and noting that \( \mathcal{E} = L(\hat{M}^*) - \hat{L}(\hat{M}^*) \leq \sup_{M \in \mathcal{M}} (L(M) - \hat{L}(M)) \) and \( \tilde{\Omega}_{vnd}(M) \leq C \) (\( C \) is the upper bound in the hypothesis class \( \mathcal{M} \)), we complete the proof of the first bound in Theorem 5. In the sequel, we present detailed proofs of these lemmas and extend the results to CSNF and CLDD.

**Proof of Lemma 25**

*Proof.* For any \( M \in \mathcal{M} \), denote its spectral decomposition as \( M = V \Pi V^\top \), where \( V \) is a standard orthogonal matrix and \( \Pi \) is a diagonal matrix. Denote \( V = (v_1, v_2, \cdots, v_D), \Pi = \)
diag(\(\pi_1, \pi_2, \cdots, \pi_D\)), then we have

\[
\mathcal{R}(\mathcal{M}) = E_{S,D,\sigma} \sup_{M \in \mathcal{M}} \left[ \frac{1}{m} \sum_{i=1}^{m} \sigma_i z_i^T M z_i \right]
\]

\[
= \frac{1}{m} E_{S,D,\sigma} \sup_{M \in \mathcal{M}} \left[ \sum_{i=1}^{m} \sigma_i \sum_{j=1}^{D} \pi_j (v_j^T z_i)^2 \right]
\]

\[
= \frac{1}{m} E_{S,D,\sigma} \sup_{M \in \mathcal{M}} \left[ \sum_{j=1}^{D} \pi_j \sum_{i=1}^{m} \sigma_i (v_j^T z_i)^2 \right]
\]

\[
= \frac{1}{m} E_{S,D,\sigma} \sup_{M \in \mathcal{M}} \sup_{\|v\|_2 \leq 1} \left[ \sum_{j=1}^{D} \pi_j \sum_{i=1}^{m} \sigma_i (v_j^T z_i)^2 \right]
\]

\[
\leq \frac{1}{m} \sup_{M \in \mathcal{M}} \text{tr}(M) E_{S,D,\sigma} \sup_{\|v\|_2 \leq 1} \sum_{i=1}^{m} \sigma_i (v^T z_i)^2.
\]

Since \((v^T z)^2\) is Lipschitz continuous w.r.t \(v^T z\) with constant \(2 \sup_{\|v\|_2 \leq 1} z^T v\), according to the composition property [32] of Rademacher complexity on Lipschitz continuous functions, we have

\[
\mathcal{R}(\mathcal{M}) \leq \frac{1}{m} 2 \sup_{\|v\|_2 \leq 1} (v^T z) \sup_{M \in \mathcal{M}} \text{tr}(M) E_{S,D,\sigma} \sup_{\|v\|_2 \leq 1} \sum_{i=1}^{m} \sigma_i v^T z_i
\]

\[
= \frac{2}{m} \sup_{M \in \mathcal{M}} \text{tr}(M) E_{S,D,\sigma} \sup_{\|v\|_2 \leq 1} \sum_{i=1}^{m} \sigma_i v^T z_i
\]

\[
\leq \frac{2}{m} \sup_{M \in \mathcal{M}} \text{tr}(M) E_{S,D,\sigma} \sup_{\|v\|_2 \leq 1} \sum_{i=1}^{m} \sigma_i z_i \|z_i\|_2
\]

\[
= \frac{2}{m} \sup_{M \in \mathcal{M}} \text{tr}(M) E_{S,D,\sigma} \sqrt{\sum_{i=1}^{m} (\sigma_i z_i)^2}.
\]

By Jensen’s inequality, we have

\[
\mathcal{R}(\mathcal{M}) \leq \frac{2}{\sqrt{m}} \sup_{M \in \mathcal{M}} \text{tr}(M) E_{S,D} \sqrt{E_{\sigma} \left( \sum_{i=1}^{m} \sigma_i z_i \right)^2}
\]

\[
\leq \frac{2B^2}{\sqrt{m}} \sup_{M \in \mathcal{M}} \text{tr}(M).
\]
Proof of lemma 26

Proof. By the definition of the convex VND regularizer, we have

\[
\hat{\Omega}_{vnd}(M) = \Gamma_{vnd}(M + \epsilon I_D, I_D) + \text{tr}(M)
\]

\[
= \text{tr}[(M + \epsilon I_D) \log(M + \epsilon I_D) - (M + \epsilon I_D) \log I_D - (M + \epsilon) + I_D] + \text{tr}(M)
\]

\[
= \sum_{j=1}^{D} [\pi_j + \epsilon \log(\pi_j + \epsilon) - (\pi_j + \epsilon) + 1] + \sum_{j=1}^{D} \pi_j
\]

\[
= \sum_{j=1}^{D} [\lambda_j + \epsilon \log(\lambda_j + \epsilon) - \epsilon + 1].
\]

Denote \( \bar{\pi} = (\sum_{j=1}^{D} \pi_j)/D = \text{tr}(M)/D \), then by Jensen’s inequality, we have

\[
\sum_{j=1}^{D} (\lambda_j + \epsilon \log(\lambda_j + \epsilon) \geq D(\bar{\pi} + \epsilon) \log(\bar{\pi} + \epsilon).
\]

Since \( \forall x \in \mathbb{R}_+, x - 1 \leq x \log x \), we have

\[
\bar{\pi} + \epsilon - 1 \leq (\bar{\pi} + \epsilon) \log(\bar{\pi} + \epsilon)
\]

\[
\leq \frac{1}{D} \sum_{j=1}^{D} (\lambda_j + \epsilon \log(\lambda_j + \epsilon)
\]

\[
\leq \frac{1}{D} \hat{\Omega}_{vnd}(M) + \epsilon - 1.
\]

Therefore we have

\[
\text{tr}(M) \leq \hat{\Omega}_{vnd}(M).
\]

Estimation Error Bound for the Convex SFN Regularizer

In this section we prove estimation error bounds for the convex SFN regularizer. The CSFN is composed of two parts. One is the squared Frobenius norm of \( M - I_D \) and the other is the trace of \( M \). We have already established a relationship between \( \text{tr}(M) \) and \( \mathcal{R}(\mathcal{M}) \). Now we analyze the relationship between \( \|M - I_D\|_F \) and \( \mathcal{R}(\mathcal{M}) \), which is given in the following lemma.

Lemma 27. Suppose \( \sup_{\|v\|_2 \leq 1} |v^T z| \leq B \), then we have

\[
\mathcal{R}(\mathcal{M}) \leq \frac{B^2}{\sqrt{m}} \sup_{M \in \mathcal{M}} \|M - I_D\|_F. \quad (4.95)
\]

138
Proof. Denote \( M(j, k) = a_{jk} \), and \( \delta_{jk} = I_{\{j=k\}} \), \( z_i = (z_{i1}, z_{i2}, \ldots, z_{id}) \), then we have

\[
\mathcal{R}(\mathcal{M}) = \frac{1}{m} E_{\mathcal{S}, \mathcal{D}, \sigma} \sup_{\mathcal{M} \in \mathcal{M}} \left[ \sum_{j,k} a_{jk} \sum_{i=1}^{m} \sigma_i z_{ij} z_{ik} \right]
\]

\[
= \frac{1}{m} E_{\mathcal{S}, \mathcal{D}, \sigma} \sup_{\mathcal{M} \in \mathcal{M}} \left[ \sum_{j,k} (a_{jk} - \delta_{jk}) \sum_{i=1}^{m} \sigma_i z_{ij} z_{ik} + \delta_{jk} \sum_{i=1}^{m} \sigma_i z_{ij} z_{ik} \right]
\]

\[
\leq \frac{1}{m} E_{\mathcal{S}, \mathcal{D}, \sigma} \sup_{\mathcal{M} \in \mathcal{M}} \left[ \| \mathbf{M} - \mathbf{I}_D \|_F \sqrt{\sum_{j,k} \left( \sum_{i=1}^{m} \sigma_i z_{ij} z_{ik} \right)^2} \right].
\]

Here the inequality is attained by Cauchy’s inequality. Applying Jensen’s inequality, we have

\[
\mathcal{R}(\mathcal{M}) \leq \frac{1}{\sqrt{m}} \sup_{\mathcal{M} \in \mathcal{M}} \| \mathbf{M} - \mathbf{I}_D \|_F E_{\mathcal{S}, \mathcal{D}} \left[ \sqrt{\sum_{j,k} \left( \sum_{i=1}^{m} \sigma_i z_{ij} z_{ik} \right)^2} \right].
\]

Recalling the definition of \( B \), we have

\[
\mathcal{R}(\mathcal{M}) \leq \frac{B^2}{\sqrt{m}} \sup_{\mathcal{M} \in \mathcal{M}} \| \mathbf{M} - \mathbf{I}_D \|_F.
\]

We now bound the estimation error with the convex SFN regularizer, which is given in the following lemma.

Lemma 28. Suppose \( \sup_{\| \mathbf{v} \|_2 \leq 1, \mathbf{z}} | \mathbf{v}^\top \mathbf{z} | \leq B \), then with probability at least \( 1 - \delta \), we have

\[
\sup_{\mathcal{M} \in \mathcal{M}} (L(M) - \hat{L}(M)) \leq \frac{2B^2}{\sqrt{m}} \min(2\hat{\Omega}_{sfn}(M), \sqrt{\hat{\Omega}_{sfn}(M)}) + \max(\tau, \hat{\Omega}_{sfn}(M)) \sqrt{\frac{2\log(1/\delta)}{m}}.
\]

Proof. For the convex SFN regularizer \( \hat{\Omega}_{sfn}(M) \), we have \( \text{tr}(\mathbf{M}) \leq \hat{\Omega}_{sfn}(\mathbf{M}) \) and \( \| \mathbf{M} - \mathbf{I}_D \| \leq \hat{\Omega}_{sfn}(\mathbf{M}) \). By Eq.(4.93), we have

\[
\sup_{(x, y) \in \mathcal{S}, \mathbf{M} \in \mathcal{M}} (x - y)^\top \mathbf{M}(x - y) \leq \sup_{\mathbf{M} \in \mathcal{M}} \hat{\Omega}_{sfn}(\mathbf{M}) B^2. \tag{4.96}
\]

By Lemma 25 and 27, we have

\[
\mathcal{R}(\mathcal{M}) \leq \frac{B^2}{\sqrt{m}} \min(2\hat{\Omega}_{sfn}(M), \sqrt{\hat{\Omega}_{sfn}(M)}). \tag{4.97}
\]

139
Substituting \(\text{Eq.(4.97)}\) and \(\text{Eq.(4.96)}\) into Lemma 24 we have

\[
\sup_{M \in M} (L(M) - \hat{L}(M)) \leq \frac{2B^2}{\sqrt{m}} \min(2\hat{\Omega}_{sfn}(M), \sqrt{\hat{\Omega}_{sfn}(M)}) + \max(\tau, \hat{\Omega}_{sfn}(M)) \sqrt{\frac{2\log(1/\delta)}{m}}.
\]

\[
\square
\]

Noting that \(\mathcal{E} = L(\hat{M}^*) - \hat{L}(\hat{M}^*) \leq \sup_{M \in M} (L(M) - \hat{L}(M))\) and \(\hat{\Omega}_{sfn}(M) \leq C\), we conclude \(\mathcal{E} \leq \frac{2B^2}{\sqrt{m}} \min(2C, \sqrt{C}) + \max(\tau, C) \sqrt{\frac{2\log(1/\delta)}{m}}\).

**Estimation Error Bound for the Convex LDD Regularizer**

Starting from Lemma 24 we bound \(\mathcal{R}(\mathcal{M})\) and \(\sup_{M \in \mathcal{M}} \text{tr}(M)\) which are given in the following two lemmas.

**Lemma 29.** Suppose \(\sup_{\|\mathbf{v}\|_2 \leq 1, \mathbf{z}} |\mathbf{v}^\top \mathbf{z}| \leq B\), then we have

\[
\mathcal{R}(\mathcal{M}) \leq \frac{B}{\sqrt{m \log(1/\epsilon)}} - 1.
\]

**Proof.** We first perform some calculation on the convex LDD regularizer.

\[
\hat{\Omega}_{ldd}(M) = \Gamma_{ldd}(M + \epsilon I_D, I_D) - (1 + \log \epsilon) \text{tr}(M) = \text{tr}((M + \epsilon I_D)I_D^{-1}) - \log \det((M + \epsilon I_D)I_D^{-1}) - D - (1 + \log \epsilon) \text{tr}(M) = \sum_{j=1}^D (\pi_j + \epsilon) - \sum_{j=1}^D \log(\pi_j + \epsilon) - D - (1 + \log \epsilon) \sum_{j=1}^D \pi_j
\]

\[
= \log(\frac{1}{\epsilon}) \sum_{j=1}^D \pi_j - \sum_{j=1}^D \log(\pi_j + \epsilon) - D(1 - \epsilon). \tag{4.98}
\]

Now we upper bound the Rademacher complexity using the CLDD regularizer.

\[
\log(\frac{1}{\epsilon}) \mathcal{R}(\mathcal{M}) = \frac{\log(\frac{1}{\epsilon})}{m} E_{S,D,\sigma} \sup_{M \in \mathcal{M}} \left[ \sum_{j=1}^D \pi_j \sum_{i=1}^m \sigma_i (\mathbf{v}_j^\top \mathbf{z}_i)^2 \right]
\]

\[
\leq \frac{1}{m} E_{S,D,\sigma} \sup_{\Pi} \sum_{j=1}^D \left[ (\log(\frac{1}{\epsilon})\pi_j - \log(\pi_j + \epsilon)) + \log(\pi_j + \epsilon) \right] \sup_{\|\mathbf{v}\|_2 \leq 1} \sum_{i=1}^m \sigma_i (\mathbf{v}_j^\top \mathbf{z}_i)^2.
\]

Similar to the proof of Lemma 25, we have

\[
\log(\frac{1}{\epsilon}) \mathcal{R}(\mathcal{M}) \leq \frac{2B^2}{\sqrt{m}} \sup_{\Pi} \sum_{j=1}^D [ (\log(\frac{1}{\epsilon})\pi_j - \log(\pi_j + \epsilon)) + \log(\pi_j + \epsilon) ] \leq \frac{2B^2}{\sqrt{m}} [ \sup_{M \in \mathcal{M}} \hat{\Omega}_{ldd}(M) + \sup_{M \in \mathcal{M}} \sum_{j=1}^D \log(\pi_j + \epsilon) ]. \tag{4.99}
\]

Denoting \(A = \sum_{j=1}^D \log(\pi_j + \epsilon)\), we bound \(A\) with \(\hat{\Omega}_{ldd}(M)\). Denoting \(\bar{\pi} = (\sum_{j=1}^D \pi_j)/D = \text{tr}(M)/D\), by Jensen’s inequality, we have

\[
A \leq D \log(\bar{\pi} + \epsilon), \tag{4.100}
\]
then $\bar{\pi} \geq e^{A/D} - \epsilon$. Replacing $\bar{\pi}$ with $A$ in Eq.(4.98), we have
\[
\hat{\Omega}_{ldd}(M) \geq D \log(1/\epsilon)(e^{A/D} - \epsilon) - A - D(1 - \epsilon)
\geq D \log(1/\epsilon)(\frac{A}{D} + 1 - \epsilon) - A - D(1 - \epsilon)
= (\log(1/\epsilon) - 1)A + \lceil \log(\frac{1}{\epsilon}) - 1 \rceil D(1 - \epsilon).
\]
Further,
\[
A \leq \frac{\hat{\Omega}_{ldd}(M)}{\log(\frac{1}{\epsilon}) - 1} - D(1 - \epsilon).
\]
Substituting this upper bound of $A$ into Eq.(4.99), we have
\[
\mathcal{R}(M) \leq \frac{2B^2 \sup_{M \in \mathcal{M}} \hat{\Omega}_{ldd}(M)}{\sqrt{m} \log(\frac{1}{\epsilon}) - 1}.
\]

The next lemma shows the bound of $\text{tr}(M)$.
**Lemma 30.** For any positive semidefinite matrix $M$, we have
\[
\text{tr}(M) \leq \frac{\hat{\Omega}_{ldd}(M) - D\epsilon}{\log(\frac{1}{\epsilon}) - 1}.
\]
**Proof.**
\[
\hat{\Omega}_{ldd}(M) \geq D \log(1/\epsilon)\bar{\pi} - D \log(\bar{\pi} + \epsilon) - D(1 - \epsilon)
\geq D \log(1/\epsilon)\bar{\pi} + D(1 - \bar{\pi}) - D(1 - \epsilon)
= D\lceil \log(1/\epsilon) - 1 \rceil \bar{\pi} + D\epsilon.
\]
Then
\[
\text{tr}(M) = D\bar{\pi} \leq \frac{\hat{\Omega}_{ldd}(M) - D\epsilon}{\log(\frac{1}{\epsilon}) - 1}.
\]

Combining Lemma 29, 30, and 24, we get the following estimation error bound w.r.t the convex LDD regularizer.
**Lemma 31.** Suppose $\sup_{\|v\| \leq 1, z} |v^\top z| \leq B$, then with probability at least $1 - \delta$, we have
\[
\sup_{M \in \mathcal{M}} (L(M) - \hat{L}(M)) \leq \frac{4B^2 \hat{\Omega}_{ldd}(M)}{\sqrt{m} \log(1/\epsilon) - 1} + \max \left(\tau, \frac{\hat{\Omega}_{ldd}(M) - D\epsilon}{\log(\frac{1}{\epsilon}) - 1}\right) \sqrt{\frac{2\log(1/\delta)}{m}}.
\]
Chapter 5

Large-scale Learning via System and Algorithm Co-design

In this chapter, we present another line of research in this thesis: large-scale machine learning, where we design efficient distributed systems \[370, 377\] for healthcare applications at scale.

To execute ML algorithms efficiently on a distributed system, it is important to perform system and algorithm co-design. On one hand, ML algorithms possess unique properties such as iterativeness and error tolerance \[162\], which can be leveraged to design efficient systems. On the other hand, based on the features of the system such as parallelism and (a)synchronicity, ML algorithms can be adjusted for more efficient execution.

Through a principled system and algorithm co-design, we build the Orpheus system for training matrix-parameterized models (MPMs), a very general class of ML models including multi-class logistic regression (MLR), deep neural networks, sparse coding \[266\], collaborative filtering \[54\], topic models \[49\], etc. In MPMs, model parameters can be represented by a matrix \(W\). For example, in MLR, rows of \(W\) represent the classification coefficient vectors corresponding to different classes; whereas in SC rows of \(W\) correspond to the basis vectors used for reconstructing the observed data. A learning algorithm, such as stochastic gradient descent (SGD), would iteratively compute an update \(\Delta W\) from data, to be aggregated with the current version of \(W\).

Learning MPMs in large scale ML problems is challenging: ML application scales have risen dramatically, a good example being the ImageNet \[95\] compendium with millions of images grouped into tens of thousands of classes. To ensure fast running times when scaling up MPMs to such large problems, it is desirable to turn to distributed computation; however, a unique challenge to MPMs is that the parameter matrix grows rapidly with problem size, causing straightforward parallelization strategies to perform less ideally. Consider a data-parallel algorithm, in which every worker uses a subset of the data to update the parameters — a common paradigm is to synchronize the full parameter matrix and update matrices amongst all workers \[78, 93, 136, 222, 304\]. However, this synchronization can quickly become a bottleneck: take MLR for example, in which the parameter matrix \(W\) is of size \(J \times D\), where \(J\) is the number of classes and \(D\) is the feature dimensionality. In one application of MLR to Wikipedia \[272\], \(J \approx 325k\) and \(D > 10,000\), thus \(W\) contains several billion entries (tens of GBs of memory). Because typical computer cluster networks can only transfer a few GBs per second at the
most, inter-machine synchronization of $W$ can dominate and bottleneck the actual algorithmic computation. In addition to the communication overhead, checkpointing these matrices for fault tolerance purposes incurs substantial disk IO.

In recent years, many distributed frameworks have been developed for large scale machine learning, including bulk synchronous parallel (BSP) systems such as Hadoop [93] and Spark [407], graph computation frameworks such as GraphLab [132], and bounded-asynchronous key-value stores such as TensorFlow [13], Bosen [162, 355], Project Adam [78], and [226]. When using these systems to learn MPMs, it is common to transmit the full parameter matrices $W$ and/or matrix updates $\Delta W$ between machines, usually in a server-client style [78, 93, 94, 136, 222, 304]. As the matrices become larger due to increasing problem sizes, so do communication costs and synchronization delays — hence, reducing such costs is a key priority when using these frameworks.

To facilitate system-algorithm co-design, we begin by investigating the structure of matrix parameterized models. Many MPMs bear the following property: when the parameter matrix $W$ is optimized with SGD [78, 94, 162] or stochastic dual coordinate ascent (SDCA) [167, 299], the update $\Delta W$ computed over one (or a few) data sample(s) is of low-rank, e.g. it can be written as the outer product of two vectors $u$ and $v$: $\Delta W = uv^\top$. The vectors $u$ and $v$ are sufficient factors (SF, meaning that they are sufficient to reconstruct the update matrix $\Delta W$). A rich set of models [78, 218, 266, 383] fall into this family: for instance, when solving an MLR problem using SGD, the stochastic gradient is $\Delta W = uv^\top$, where $u$ is the prediction probability vector and $v$ is the feature vector. Similarly, when solving an $\ell_2$ regularized MLR problem using SDCA, the update matrix $\Delta W$ also admits such a structure, where $u$ is the update vector of a dual variable and $v$ is the feature vector. Other models include neural networks [78], distance metric learning [383], sparse coding [266], non-negative matrix factorization [218], and principal component analysis, to name a few.

Leveraging this property, we propose a computation model called sufficient factor broadcasting (SFB). The basic idea is to send sufficient factors (SFs) between workers, which then reconstruct matrix updates $\Delta W$ locally, thus greatly reducing inter-machine parameter communication. This stands in contrast to the well-established parameter server idiom [78, 226], a centralized design where workers maintain a “local” image of the parameters $W$, which are synchronized with a central parameter image $W$ (stored on the “parameter servers”). In existing parameter server designs, the (small, low-rank) updates $\Delta W$ are accumulated into the central parameter server’s $W$, and the low-rank structure of each update $\Delta W$ is lost in the process. Thus, the parameter server can only transmit the (large, full-rank) matrix $W$ to the workers, inducing extra communication that could be avoided. We address this issue by a decentralized, peer-to-peer architecture that performs SFB, where each worker keeps its own image of the parameters $W$ (either in memory or on local disk), and sends sufficient factors to other workers for parameter synchronization. SFB is highly communication-efficient; transmission costs are linear in the dimensions of the parameter matrix, and the resulting faster communication greatly reduces waiting time in synchronous systems (e.g. Hadoop and Spark), or improves parameter freshness in (bounded) asynchronous systems (e.g. GraphLab, Petuum-PS and [226]). Most importantly, such savings are achieved without compromising the mathematical equivalence of the new ML solution to the original one. SFs have been used to speed up some (but not all) network communication in deep learning [78]; our work differs primarily in that we always transmit SFs,
never full matrices.

In addition to the SFB computation model, we propose a battery of system designs (SD) and algorithm designs (ADs) for efficient communication, lightweight fault tolerance and recovery, and easiness of programming. In communication, an AD – SF selection and a SD – random multicast are used to further reduce the size of each network message and the number of messages. In fault tolerance, based on an AD – using SFs to represent parameter states, an SD – incremental SF checkpoint (ISFC) – is used, which continuously saves new SFs computed at each clock to stable storage. Compared with checkpointing parameter matrices, ISFC greatly reduces disk I/O, avoids compute-cycle waste, and provides fine-grained (per-clock) rollbacks. In addition, we provide an easy-to-use programming model where the generation of SFs is automatically identified rather than being specified by users, which reduces users’ programming efforts.

We provide theoretical analysis which shows that the algorithms executed using SFB are guaranteed to converge. We also extend SFB to a hybrid communication model where decentralized SFB and centralized parameter server are used together to train convolutional neural networks. Evaluations on a wide range of ML applications demonstrate the efficiency and scalability of our systems.

5.1 Sufficient Factor Property

The core goal of sufficient factor broadcasting (SFB) is to reduce network communication costs for matrix-parameterized models; specifically, those that follow an optimization formulation

\[
(P) \quad \min_{W} \frac{1}{N} \sum_{i=1}^{N} f_i(Wa_i) + h(W),
\]

where the model is parametrized by a matrix \( W \in \mathbb{R}^{J \times D} \). The loss function \( f_i(\cdot) \) is typically defined over a set of training samples \( \{(a_i, b_i)\}_{i=1}^{N} \), with the dependence on \( b_i \) being suppressed. We allow \( f_i(\cdot) \) to be either convex or nonconvex, smooth or nonsmooth (with subgradient everywhere); examples include \( \ell_2 \) loss and multiclass logistic loss, amongst others. The regularizer \( h(W) \) is assumed to admit an efficient proximal operator \( \text{prox}_h(\cdot) \). For example, \( h(\cdot) \) could be an indicator function of convex constraints, \( \ell_1-, \ell_2-, \text{trace-norm} \), to name a few. The vectors \( a_i \) and \( b_i \) can represent observed features, supervised information (e.g., class labels in classification, response values in regression), or even unobserved auxiliary information (such as sparse codes in sparse coding [266]) associated with data sample \( i \). The key property we exploit below ranges from the matrix-vector multiplication \( Wa_i \). This optimization problem \( (P) \) can be used to represent a rich set of ML models [78, 218, 266, 383], such as the following:

- Distance metric learning (DML) [383] improves the performance of other ML algorithms, by learning a new distance function that correctly represents similar and dissimilar pairs of data samples; this distance function is a matrix \( W \) that can have billions of parameters or more, depending on the data sample dimensionality. The vector \( a_i \) is the difference of the feature vectors in the \( i \)-th data pair and \( f_i(\cdot) \) can be either a quadratic function or a hinge loss function, depending on the similarity/dissimilarity label \( b_i \) of the data pair. In both cases, \( h(\cdot) \) can be an \( \ell_1-, \ell_2-, \text{trace-norm} \) regularizer or simply \( h(\cdot) = 0 \) (no regularization).
• Sparse coding (SC) \cite{266} learns a dictionary of basis from data, so that the data can be represented sparsely (and thus efficiently) in terms of the dictionary. In SC, \( W \) is the dictionary matrix, \( a_i \) are the sparse codes, \( b_i \) is the input feature vector, and \( f_i(·) \) is a quadratic function \cite{266}. To prevent the entries in \( W \) from becoming too large, each column \( W_k \) must satisfy \( \| W_k \|_2 \leq 1 \). In this case, \( h(W) \) is an indicator function which equals 0 if \( W \) satisfies the constraints and equals \( \infty \) otherwise.

To solve the optimization problem \((P)\), it is common to employ either (proximal) stochastic gradient descent (SGD) \cite{78, 94, 162, 222} or stochastic dual coordinate ascent (SDCA) \cite{167, 299}, both of which are popular and well-established parallel optimization techniques.

**Proximal SGD**  In proximal SGD, a stochastic estimate of the gradient, \( \Delta W \), is first computed over one data sample (or a mini-batch of samples), in order to update \( W \) via \( W ← W − \eta \Delta W \) (where \( \eta \) is the learning rate). Following this, the proximal operator \( \text{prox}_{\eta h}(·) \) is applied to \( W \).

Notably, the stochastic gradient \( \Delta W \) in \((P)\) can be written as the outer product of two vectors \( \Delta W = uv^\top \), where \( u = \partial_f(Wa_i,b_i) \), \( v = a_i \), according to the chain rule. Later, we will show that this low rank structure of \( \Delta W \) can greatly reduce inter-worker communication.

**Stochastic DCA**  SDCA applies to problems \((P)\) where \( f_i(·) \) is convex and \( h(·) \) is strongly convex (e.g. when \( h(·) \) contains the squared \( \ell_2 \) norm); it solves the dual problem of \((P)\), via stochastic coordinate ascent on the dual variables. Introducing the dual matrix \( U = [u_1, \ldots, u_N] \in \mathbb{R}^{J \times N} \) and the data matrix \( A = [a_1, \ldots, a_N] \in \mathbb{R}^{D \times N} \), the dual problem of \((P)\) can be written as

\[
(D) \quad \min_U \frac{1}{N} \sum_{i=1}^{N} f_i^*(−u_i) + h^*(\frac{1}{N}UA^\top),
\]

where \( f_i^*(·) \) and \( h^*(·) \) are the Fenchel conjugate functions of \( f_i(·) \) and \( h(·) \), respectively. The primal-dual matrices \( W \) and \( U \) are connected by \( W = ∇h^*(Z) \), where the auxiliary matrix \( Z := \frac{1}{N}UA^\top \). Algorithmically, we need to update the dual matrix \( U \), the primal matrix \( W \), and the auxiliary matrix \( Z \): every iteration, we pick a random data sample \( i \), and compute the stochastic update \( \Delta u_i \) by minimizing \((D)\) while holding \( \{u_j\}_{j≠i} \) fixed. The dual variable is updated via \( u_i ← u_i − \Delta u_i \), the auxiliary variable via \( Z ← Z − \Delta u_i a_i^\top \), and the primal variable via \( W ← ∇h^*(Z) \). Similar to SGD, the update of \( Z \) is also the outer product of two vectors: \( \Delta u_i \) and \( a_i \), which can be exploited to reduce communication cost.

**Sufficient factor property in SGD and SDCA**  In both SGD and SDCA, the parameter matrix update can be computed as the outer product of two vectors — we call these sufficient factors (SFs). This property can be leveraged to improve the communication efficiency of distributed ML systems: instead of communicating parameter/update matrices among machines, we can communicate the SFs and reconstruct the update matrices locally at each machine. Because the SFs are much smaller in size, synchronization costs can be dramatically reduced.

More generally, the update matrix \( \Delta W \) may not be exactly rank-1, but still of very low rank. For example, when each machine uses a mini-batch of size \( K \), \( \Delta W \) is of rank at most \( K \); in restricted Boltzmann machines, the update of the weight matrix is computed from four vectors.
\( u_1, v_1, u_2, v_2 \) as \( u_1 v_1^\top - u_2 v_2^\top \), \textit{i.e.} rank-2; for the BFGS algorithm \cite{40}, the update of the inverse Hessian is computed from two vectors \( u, v \) as \( \alpha uu^\top - \beta (uv^\top + vu^\top) \), \textit{i.e.} rank-3. Even when the update matrix \( \Delta W \) is not genuinely low-rank, to reduce communication cost, it might still make sense to send only a certain low-rank \textit{approximation}. We intend to investigate these possibilities in future work.

### 5.2 Orpheus: A Light-weight Peer-to-peer System

In this section, we present a peer-to-peer framework Orpheus where the system design is driven by the sufficient factor property. Orpheus is a decentralized system that executes data-parallel \cite{78,85,94,226,355} distributed training of matrix-parameterized ML models. Orpheus runs on a group of worker machines connected via a peer-to-peer (P2P) network. Unlike the client-server architectures including the parameter server \cite{78,85,94,162,226,355}, machines in Orpheus play equal roles without server/client asymmetry and every pair of machines can communicate. Each machine holds one shard of the data and a replica of the model parameters. Machines synchronize their model replicas to ensure consistency, by exchanging parameter-(pre)updates via network communication. Under this general framework, Orpheus applies a battery of system-algorithm co-designs to achieve efficiency in communication and fault tolerance.

For efficient communication, the main idea is to represent the parameter update matrices by their corresponding SFs, which can be understood as “pre-updates”, meaning that the actual update matrices must be computed on each client upon receiving fresh SFs, and the update matrices themselves are never transmitted. Since the size of SFs is much smaller than matrices, the communication cost can be substantially reduced. Under a P2P architecture, in addition to void transmitting update matrices, Orpheus can also avoid transmitting parameter matrices, while still achieving synchrony. Besides, \textit{random multicast}, under which each machine sends SFs to a randomly-chosen subset of machines, is leveraged to reduce the number of messages. \textit{SF selection}, which chooses a subset of representative SFs to communicate, is used to further reduce the size of each message.

Orpheus uses \textit{incremental SF checkpoint} for fault tolerance, motivated by the fact that the parameter states can be represented as a dynamically growing set of SFs. Machines continuously save the new SFs computed in each logical time onto stable storage. To recover a parameter state, Orpheus transforms the saved SFs into a matrix. Compared with checkpointing parameter matrices \cite{13,407}, saving vectors requires much less disk IO and does not require the application program to halt. Besides, the parameters can be rollbacked to the state in any logical time.

In programming abstraction, the SFs are explicitly exposed such that system-level optimizations based on SFs can be exploited. Orpheus is able to automatically identify the symbolic expressions representing SFs and updates, relieving users’ burden to manually specify them.

Orpheus supports two consistency models: bulk synchronous parallel (BSP) \cite{42} and staleness synchronous parallel (SSP) \cite{162}. BSP sets a global barrier at each clock. A worker cannot proceed to the next clock until all workers reach this barrier. SSP allows workers to have different paces as long as their difference in clock is no more than a user-defined \textit{staleness} threshold.
5.2.1 Communication: Sufficient Factor Broadcasting (SFB)

Orpheus explores system-algorithm co-design to perform efficient communication. To ensure the consistency among different parameter replicas, the updates computed at different machines need to be exchanged. One popular system architecture that enables this is parameter server (PS) \[78, 85, 94, 226, 355\], which conceptually consists of a server machine that maintains shared state of the parameters and a set of worker machines each having a local cache of the parameters. In PS, the updates computed at worker machines are aggregated at the server and applied to the shared state. The shared state is subsequently sent back to workers to refresh their local caches. When PS is used to train MPMs, the update and parameter matrices – which could contain billions of elements \[214\] – are transferred, incurring substantial communication overhead.

Sufficient Factor Broadcasting

Leveraging the SF property of the update matrix in problems (P) and (D), we propose a sufficient factor broadcasting (SFB) computation model that supports efficient (low-communication) distributed learning of the parameter matrix \(W\). We assume a setting with \(P\) workers, each of which holds a data shard and a copy of the parameter matrix \(W\). Stochastic updates to \(W\) are generated via proximal SGD or SDCA, and communicated between machines to ensure parameter consistency. In proximal SGD, on every iteration, each worker \(p\) computes SFs \((u_p, v_p)\), based on one data sample \(x_i = (a_i, b_i)\) in the worker’s data shard. The worker then broadcasts \((u_p, v_p)\) to all other workers; once all \(P\) workers have performed their broadcast (and have thus received all SFs), they re-construct the \(P\) update matrices (one per data sample) from the \(P\) SFs, and apply them to update their local copy of \(W\). Finally, each worker applies the proximal operator \(\text{prox}_h(\cdot)\). When using SDCA, the above procedure is instead used to broadcast SFs for the auxiliary matrix \(Z\), which is then used to obtain the primal matrix \(W = \nabla h^*(Z)\). Figure 5.1 illustrates the SFB operation: 4 workers compute their respective SFs \((u_1, v_1), \ldots, (u_4, v_4)\), which are then

---

1For simplicity, we assume each worker has enough memory to hold a full copy of the parameter matrix \(W\). If \(W\) is too large, one can either partition it across multiple machines \[92, 226\], or use local disk storage (i.e. out of core operation). We plan to investigate these strategies as future work.
broadcast to the other 3 workers. Each worker $p$ uses all 4 SFs $(u_1, v_1), \ldots, (u_4, v_4)$ to exactly reconstruct the update matrices $\Delta W_p = u_p v_p^\top$, and update their local copy of the parameter matrix: $W_p \leftarrow W_p - \sum_{q=1}^{4} u_q v_q^\top$. While the above description reflects synchronous execution, it is easy to extend to (bounded) asynchronous execution.

The communication cost of transmitting SFs is $O(J + D)$ which is linear in matrix dimensions while that of transmitting update matrices (UMs) is $O(JD)$ which is quadratic in matrix dimensions. Hence SF-transfer can greatly reduce communication overhead. The transformation from SFs to a UM is mathematically exact, without compromising computational correctness.

In PS, the one-sided communication cost from workers to the server can be reduced by transmitting SFs [78]: each worker sends the new SFs to the server, where the received SFs are transformed to UM to update the shared parameter state. However, since the parameter matrix cannot be computed from a few SFs, from server to workers the newly-updated parameters needs to be sent as a matrix, which still incurs high communication overhead. To avoid transmitting parameter matrices, Orpheus adopts a decentralized peer-to-peer architecture where worker machines synchronize their parameter replicas by exchanging updates in the form of SFs. Unlike PS, the P2P architecture does not maintain the shared parameter state and can completely avoid transmitting any type of matrices.

While SF-transfer greatly reduces communication cost, it increases computation overhead. Each group of SFs are transformed into the same update multiple times (on different receivers). However, in-memory computation is usually much more efficient than inter-machine network communication, especially with the advent of GPU computing, hence the reduction in communication cost overshadows the increase of computation overhead.

**Random Multicast**

While P2P transfer of SFs greatly reduces the size of each message (from a matrix to a few vectors), its limitation is SFs need to be sent from each machine to every other machine, which renders the number of messages per clock to be quadratic in the number of machines $P$. To address this issue, Orpheus adopts random multicast: in each clock, each machine randomly selects $Q (Q < P - 1)$ machines to send SFs to. This cuts messages per clock from $O(P^2)$ to $O(PQ)$. Figure 5.2 shows an example. In each iteration $t$, an update $U_p^t$ generated by machine $p$ is sent only to machines that are directly connected with $p$ (and the update $U_p^t$ takes effect at
iteration $t + 1$). The effect of $U^t_p$ is *indirectly and eventually* transmitted to every other machine $q$, via the updates generated by machines sitting between $p$ and $q$ in the topology. This happens at iteration $t + \tau$, for some delay $\tau > 1$ that depends on $Q$ and the location of $p$ and $q$ in the network topology. Consequently, the $P$ machines will not have the exact same parameter image $W$, even under bulk synchronous parallel execution — yet this does not empirically (Section 5.2.5) compromise algorithm accuracy as long as $Q$ is not too small. We hypothesize that this property is related to the tolerance of ML algorithms to bounded-asynchronous execution and random error, and we provide a theoretical analysis in Section 5.4.

Two random selection methods are provided. One is *uniform selection*: each machine has the same probability to be selected. The other is *prioritized selection* for load-balancing purpose. Each machine is assigned a priority score based on its progress (measured by clock). A machine with faster progress (higher priority) is selected with higher probability and receives more SFs from slower machines. It spends more compute cycles to consume these remote SFs and slows down the computation of new SFs, giving the slower machines a grace time to catch up.

Unlike a deterministic multicast topology [222] where each machine communicates with a fixed set of machines throughout the application run, random multicast provides several benefits. First, dynamically changing the topology in each clock gives every two machines a chance to communicate directly, which facilitates more symmetric synchronization. Second, random multicast is more robust to network connection failures since the failure of a network connection between two machines will not affect their communication with other one. Third, random multicast makes resource elasticity simpler to implement: adding and removing machines require minimal coordination with existing ones, unlike a deterministic topology which must be modified every time a worker joins or leaves.

**SF Selection**

In ML practice, parameter updates are usually computed over a small batch (whose size typically ranges from tens to hundreds) of examples. At each clock, a batch of $K$ training examples are selected and an update is generated with respect to each example. When represented as matrices, these $K$ updates can be aggregated into a single matrix to communicate. Hence the communication cost is independent of $K$. However, this is not the case in sufficient factors transfer: the $K$ SFGs cannot be aggregated into one single SFG; they must be transferred individually. Therefore, communication cost grows linearly with $K$. To alleviate this cost, Orpheus provides SF selection (SFS), which chooses a subset of $C$ SFGs (where $C < K$) — that best represent the entire batch — to communicate.

We design an efficient sampling-based algorithm called *joint matrix column subset selection* (JMCSS) to perform SFS. Given the $P$ matrices $X^{(1)}, \ldots, X^{(P)}$ where $X^{(p)}$ stores the $p$-th SF of all SFGs, JMCSS selects a subset of non-redundant column vectors from each matrix to approximate the entire matrix. The selection of columns in different matrices are tied together, i.e., if the $i$-th column is selected in one matrix, for all other matrices their $i$-th column must be selected as well to atomically form an SFG. Let $I = \{i_1, \ldots, i_C\}$ index the selected SFGs and $S_I^{(p)}$ be a matrix whose columns are from $X^{(p)}$ and indexed by $I$. The goal is to find out the optimal selection $I$ such that the following approximation error is minimized: $\sum_{p=1}^{P} \| X^{(p)} - S_I^{(p)}(S_I^{(p)})^\dagger X^{(p)} \|_2$, 149
Algorithm 3: Joint matrix column subset selection.

Input: \{\{X^{(p)}\}_{p=1}^{P}\}
Initialization: \forall p, X_0^{(p)} = X^{(p)}, S_0^{(p)} = []
for \(t \in \{1, \ldots, C\}\) do

- Compute the squared L2 norm of column vectors in \{\{X_{t-1}^{(p)}\}_{p=1}^{P}\}
- Sample a column index \(i_t\)
- \(\forall p, X_t^{(p)} \leftarrow X_{t-1}^{(p)}/\{x_{i_t}^{(p)}\}\)
- \(\forall p, S_t^{(p)} \leftarrow S_{t-1}^{(p)} \cup \{x_{i_t}^{(p)}\}\)

end for
Output: \{\{S^{(p)}\}_{p=1}^{P}\}

where \((S_I^{(p)})^\dagger\) is the pseudo-inverse of \(X_I^{(p)}\).

Finding the exact solution of this problem is NP-hard. To address this issue, we develop a sampling-based method (Algorithm 3), which is an adaptation of the iterative norm sampling algorithm [99]. Let \(S^{(p)}\) be a dynamically growing matrix that stores the column vectors to be selected from \(X^{(p)}\) and \(S_t^{(p)}\) denote the state of \(S^{(p)}\) at iteration \(t\). Accordingly, \(X^{(p)}\) is dynamically shrinking and its state is denoted by \(X_t^{(p)}\). At the \(t\)-th iteration, an index \(i_t\) is sampled and the \(i_t\)-th column vectors are taken out from \{\{X^{(p)}\}_{p=1}^{P}\} and added to \{\{S^{(p)}\}_{p=1}^{P}\}. \(i_t\) is sampled in the following way. First, we compute the squared L2 norm of each column vector in \{\{X_{t-1}^{(p)}\}_{p=1}^{P}\}. Then sample \(i_t\) (1 ≤ \(i_t\) ≤ \(K + 1 - t\)) with probability proportional to \(\prod_{p=1}^{P} \|x_{i_t}^{(p)}\|^2\), where \(x_{i_t}^{(p)}\) denotes the \(i_t\)-th column vector in \(X_{t-1}^{(p)}\). The selected column vector is removed from \(X_{t-1}^{(p)}\) and added to \(S_{t-1}^{(p)}\). Then a back projection is utilized to transform \(X_t^{(p)}\): \(X_t^{(p)} \leftarrow X_t^{(p)} - S_t^{(p)}(S_t^{(p)})^\dagger X_t^{(p)}\). After \(C\) iterations, we obtain the selected SFs contained in \{\{S^{(p)}\}_{p=1}^{P}\} and pack them into SFGs, which are subsequently sent to other machines. Under JMCSS, the aggregated update generated from the \(C\) SFGs is close to that computed from the entire batch. Hence SFS does not compromise parameter-synchronization quality.

The selection of SFs is pipelined with their computation and communication to increase throughput. Two FIFO queues (denoted by A and B) containing SFs are utilized for coordination. The computation thread adds newly-computed SFs into queue A. The selection thread dequeues SFs from A, executes the selection and adds the selected SFs to queue B. The communication thread dequeues SFs from B and sends them to other machines. The three modules operate asynchronously: for each one, as long as its input queue is not empty and output queue is not full, the operation continues. The two queues can be concurrently accessed by their producer and consumer.

Cost Analysis

Figure [5, 3] compares the communications, space and time (to apply updates to \(W\)) costs of peer-to-peer SFB, against parameter server (PS) architectures [94, 226, 355]. For SFB with a broadcasting scheme, in each mini-batch, every worker broadcasts \(K\) SF pairs \((u, v)\) to \(P - 1\) other workers, i.e. \(O(P^2K(J + D))\) values are sent per iteration — linear in matrix dimensions \(J, D\).
and quadratic in $P$. For SFB with a multicast scheme, every worker communicates SF pairs with $Q < P$ peers, hence the communication cost is reduced to $O(PQK(J + D))$. Because SF pairs cannot be aggregated before transmission, the cost has a dependency on $K$. In contrast, the communication cost in PS is $O(PJD)$, linear in $P$, quadratic in matrix dimensions, and independent of $K$. For both SFB and PS, the cost of storing $W$ is $O(JD)$ on every machine. As for the time taken to update $W$ per iteration, PS costs $O(PJD)$ at the server (to aggregate $P$ client update matrices) and $O(PKJD)$ at the $P$ clients (to aggregate $K$ updates into one update matrix). By comparison, SFB bears a cost of $O(P^2KJD)$ under broadcasting and $O(PQKJD)$ under multicast due to the additional overhead of reconstructing each update matrix $P$ or $Q$ times.

Compared with PS, SFB achieves communication savings by paying an extra computation cost. In a number of practical scenarios, such a tradeoff is worthwhile. Consider large problem scales where $\min(J, D) \geq 10000$, and moderate mini-batch sizes $1 \leq K \leq 100$ (as studied in this work); when using a moderate number of machines (around 10-100), the $O(P^2K(J + D))$ communications cost of SFB is lower than the $O(PJD)$ cost for PS, and the relative benefit of SFB improves as the dimensions $J, D$ of $W$ grow. In data center scale computing environments with thousands of machines, we can adopt the multicast scheme. As for the time needed to apply updates to $W$, it turns out that the additional cost of reconstructing each update matrix $P$ or $Q$ times in SFB is negligible in practice — we have observed in our experiments that the time spent computing SFs, as well as communicating SFs over the network, greatly dominates the cost of reconstructing update matrices using SFs. Overall, the communication savings dominate the added computational overhead, which we validated in experiments.

### 5.2.2 Fault Tolerance

In this section, further exploring the SF update property, we propose to represent the parameter matrix using SFs. Based on such a representation, a light-weight fault tolerance approach is developed.

**SF-based Representation of Parameters**

We first show the parameter matrix can be represented as a set of SFs. At clock $T$, the parameter state $W_T$ is mathematically equal to $W_0 + \sum_{t=1}^{T} \Delta W_t$ where $\Delta W_t$ is the update matrix computed

<table>
<thead>
<tr>
<th>Computational Model</th>
<th>Total comms, per iter</th>
<th>$W$ storage per machine</th>
<th>$W$ update time, per iter</th>
</tr>
</thead>
<tbody>
<tr>
<td>SFB (peer-to-peer, broadcast)</td>
<td>$O(P^2K(J + D))$</td>
<td>$O(JD)$</td>
<td>$O(P^2KJD)$</td>
</tr>
<tr>
<td>SFB (peer-to-peer, multicast)</td>
<td>$O(PQK(J + D))$</td>
<td>$O(JD)$</td>
<td>$O(PQKJD)$</td>
</tr>
<tr>
<td>PS (client-server)</td>
<td>$O(PJD)$</td>
<td>$O(JD)$</td>
<td>$O(PJD)$ at server, $O(PKJD)$ at clients</td>
</tr>
</tbody>
</table>

Figure 5.3: Cost of using SFB versus PS. $K$ is the mini-batch size, $J, D$ are the dimensions of $W$, and $P$ is the number of workers.
at clock $t$ and $W_0$ is the initialization of the parameters. As noted earlier, $\Delta W_t$ can be computed from an SFG $G_t$: $\Delta W_t = h(G_t)$, using a transformation $h$. Following the standard practice of initializing ML models using randomization, we randomly generate an SFG $G_0$, then let $W_0 = h(G_0)$. To this end, the parameter state can be represented as $W_T = \sum_{t=0}^{T} h(G_t)$, using a set of SFGs.

**Incremental SF Checkpoint**

Based on the SF-representation (SFR) of parameters and inspired by the asynchronous and incremental checkpointing methods [17, 263], Orpheus provides an *incremental SF checkpoint* (ISFC) mechanism for fault tolerance and recovery: each machine continuously saves the new SFGs computed in each clock to stable storage and restores the parameters from the saved SFGs when machine failure happens. Unlike existing systems [13, 407] which checkpoint large matrices, saving small vectors consume much less disk bandwidth. To reduce the frequency of disk write, the SFGs generated after each clock are not immediately written onto the disk, but staged in the host memory. When a large batch of SFGs are accumulated, Orpheus writes them together.

ISFC does not require the application program to halt while checkpointing the SFs. The IO thread reads the SFs and the computing thread writes the parameter matrix. There is no read/write conflict. In contrast, in matrix-based checkpointing, the IO thread reads the parameter matrix, which requires the computation thread to halt to ensure consistency, incurring waste of compute cycles.

ISFC is able to rollback the parameters to the state at any clock. To obtain the state at clock $T$, Orpheus collects the SFGs computed up to $T$ and transforms them into a parameter matrix. This granularity is much more fine-grained than checkpointing parameter matrices. Since saving large-sized matrices to disk is time-consuming, the system can only afford to perform a checkpoint periodically and the parameter states between two checkpoints are lost. The $\text{restore}(T)$ API is used for recovery where $T$ is a user-specified clock which the parameters are to be rolled-back to. The default $T$ is the latest clock.

**5.2.3 Programming Model**

The Orpheus programming model provides a data abstraction called sufficient factor group (SFG) and two user-defined functions that generate and consume SFGs to update model parameters. Each SFG contains a set of SFs that are generated with respect to one data example and atomically produces a parameter update. The SFs are immutable and dense, and their default type is $\text{float}$. Inside an SFG, each SF has an index. To program an Orpheus application, users specify two functions: (1) $\text{compute_svg}$ which takes the current parameter state and one data example as inputs and computes vectors that collectively form an SFG; (2) $\text{compute_update}$ which takes an SFG and produces a parameter update. These two functions are invoked by the Orpheus engine to perform *data-parallel* distributed ML: each of the $P$ machines holds one shard of the training data and a replica of parameters; different parameter replicas are synchronized across machines to retain consistency (consistency means different replicas are encouraged to be as close as possible). Every machine executes a sequence of operations iteratively: in each clock, a small batch of training examples are randomly selected from the data shard and $\text{compute_svg}$
Algorithm 4: Execution semantics.

Allocate an empty SFG set \( \mathcal{S} = \{\} \)
Select a small batch of training examples \( \mathcal{X} \)
for each \( x \in \mathcal{X} \)
    Compute an SFG \( s = \text{compute}\_\text{svg}(W^{(\text{old})}, x) \)
    Add \( s \) to \( \mathcal{S} \)
end for
Send \( \mathcal{S} \) to other machines
Receive the SFG set \( \mathcal{T} \) from other machines
for each \( s \in \mathcal{S} \cup \mathcal{T} \)
    Compute an update \( u = \text{compute}\_\text{update}(s) \)
    Update parameters \( W^{(\text{new})} \leftarrow W^{(\text{old})} + u \)
end for

is invoked to compute an SFG with respect to each example; the SFGs are then sent to other machines for parameter synchronization; \text{compute}\_\text{update} is invoked to transform locally-generated SFGs and remotely-received SFGs into updates which are subsequently added to the parameter replica. The execution semantics (per-clock) of the Orpheus engine is shown in Algorithm 4. Unlike existing systems which directly compute parameter updates from training data, Orpheus breaks this computation into two steps and explicitly exposes the intermediate SFs to users, which enables SF-based system-level optimizations to be exploited.

Below shows how these two functions are implemented in MLR. The inputs of the \text{compute}\_\text{svg} function include the parameter replica Parameters and a data example Data and the output is a SFG. A SFG is declared via \( \text{SFG}([d_1, \cdots, d_J]) \) where \( d_j \) is the length of the \( j \)-th SF. In MLR, an SFG contains two SFs: the first one is the difference between the prediction vector \( \text{softmax}(W^{(\text{old})} smp.\text{feats}) \) and the label vector \( smp.\text{label} \); the second one is the feature vector \( smp.\text{feats} \). The update matrix is computed as the outer product between the two SFs.

```python
def compute_svg(Parameters W, Data smp):
    svg=SFG([W.nrows, W.ncols])
    x=softmax(W*smp.feats)-smp.label
    svg.sv[0]=x
    svg.sv[1]=smp.feats
    return svg
def compute_update(SFG svg):
    return outproduct(svg.sv[0],svg.sv[1])
```

**Automatic Identification of SFs and Updates**

When ML models are trained using gradient descent or quasi-Newton algorithms, the computation of SFGs and updates can be automatically identified by the Orpheus engine, which relieves users from writing the two functions \text{compute}\_\text{svg} and \text{compute}\_\text{update}. The only input required from users is a symbolic expression of the loss function, which is in general much easier
to program compared with the two functions. Note that this is not an extra burden: in most ML applications, users need to specify this loss function to measure the progress of execution.

The identification procedure of SFs depends on the optimization algorithm – either gradient descent or quasi-Newton – specified by the users for minimizing the loss function. For both algorithms, automatic differentiation techniques [13, 34] are needed to compute the gradient of variables. Given the symbolic expression of the loss function, such as $f = \text{cross}\_\text{entropy} (\text{softmax}(W \times x), y)$ in MLR, the Orpheus engine first parses it into an expression graph [43] as shown in Figure 5.4. In the graph, circles denote variables including terminals such as $W, x, y$ and intermediate ones such as $a = W \times x, b = \text{softmax}(a)$; boxes denote operators applied to variables. According to their inputs and outputs, operators can be categorized into different types, shown in Table 5.1. Given the expression graph, Orpheus uses automatic differentiation to compute the symbolic expressions of the gradient $\frac{\partial f}{\partial z}$ of $f$ w.r.t to each unknown variable $z$ (either a terminal or an intermediate one). The computation is executed recursively in the backward direction of the graph. For example, in Figure 5.4, to obtain $\frac{\partial f}{\partial a}$, we first compute $\frac{\partial f}{\partial b}$, then transform it into $\frac{\partial f}{\partial a}$ using an operator-specific matrix $A$. For a type-2 operator (e.g., softmax) in Table 5.1, $A_{ij} = \frac{\partial b_j}{\partial a_i}$.

If $W$ is involved in a type-5 operator (Table 5.1) which takes $W$ and a vector $x$ as inputs and produces a vector $a$ and the gradient descent algorithm is used to minimize the loss function, then the SFG contains two SFs which can be automatically identified: one is $\frac{\partial f}{\partial a}$ and the other is $x$. Accordingly, the update of $W$ can be automatically identified as the outer product of the two SFs.

If quasi-Newton methods are used to learn ML models parameterized by a vector $x$, Orpheus can automatically identify the SFs of the update of the approximated Hessian matrix $W$. First of all, automatic differentiation is applied to compute the symbolic expression of the gradient $g(x) = \frac{\partial f}{\partial x}$. To identify the SFs at clock $k$, we plug in the states $x_{k+1}$ and $x_k$ of the parameter vector in clock $k + 1$ and $k$ into $g(x)$ and calculate a vector $y_k = g(x_{k+1}) - g(x_k)$. We also
compute another vector $s_k = x_{k+1} - x_k$. Then based on $s_k$, $y_k$, and $W_k$ (the state of $W$ at clock $k$), we can identify the SFs which depend on the specific quasi-Newton algorithm instance. For BFGS, the procedures are: (1) set $y_k \leftarrow y_k / \sqrt{y_k^T s_k}$; (2) compute $v_k = W_k s_k$; (3) set $y_k \leftarrow y_k / \sqrt{s_k^T v_k}$. Then the SFs are identified as $y_k$ and $v_k$ and the update of $W_k$ is computed as $y_k y_k^T - v_k v_k^T$. For DFP, the procedures are: (1) set $s_k \leftarrow s_k / \sqrt{y_k^T s_k}$; (2) compute $v_k = W_k y_k$; (2) set $v_k \leftarrow v_k / \sqrt{y_k^T v_k}$. Then the SFs are identified as $s_k$ and $v_k$ and the update of $W_k$ is computed as $s_k s_k^T - v_k v_k^T$.

### 5.2.4 Implementation

Orpheus is a decentralized system, where workers are symmetric, running the same software stack, which is conceptually divided into three layers (Figure 5.5): (1) an ML application layer including ML programs implemented on top of Orpheus, such as multiclass logistic regression, topic models, deep learning models, etc.; (2) a service layer for automatic identification of SFs, SF selection, fault tolerance, etc.; (3) a P2P communication layer for sufficient factor transfer and random multicast.

The major modules in the Orpheus engine include: (1) an interpreter that automatically identifies the symbolic expressions of SFs and parameter updates; (2) a SF generator that selects training examples from local data shard and computes an SFG for each example using the symbolic expressions of SFs produced by the interpreter; (3) an SF selector that chooses a small subset of most representative SFs out of those computed by the generator for communication; (4) a communication manager that transfers the SFs chosen by the selector using broadcast or random multicast and receives remote SFs; (5) an update generator which computes update matrices from locally-generated and remotely-received SFs and updates the parameter matrix; (6) a central coordinator for periodic centralized synchronization, parameter-replicas rotation, and elasticity.
**Heterogeneous computing**  The Orpheus programming interface exposes a rich set of operators, such as matrix multiplication, vector addition, and softmax, through which users write their ML programs. To support heterogeneous computing, each operator has a CPU implementation and a GPU implementation built upon highly optimized libraries such as Eigen, cuBLAS, and cuDNN. In the GPU implementation, Orpheus performs *kernel fusion* which combines a sequence of kernels into a single one, to reduce the number of kernel launches that bear large overhead. The Orpheus engine generates a dependency graph of operators by parsing users’ program and traverses the graph to fuse consecutive operators into one CUDA kernel.

**Elasticity**  Orpheus is elastic to resource adjustment. Adding new machines and preempting existing machines do not interrupt the current execution. To add a new machine, the central coordinator executes the following steps: (1) launching the Orpheus engine and application program on the new machine; (2) averaging the parameter replicas of existing machines and placing the averaged parameters on the new machine; (3) taking a chunk of training data from each existing machine and assigning the data to the new machine; (4) adding the new machine into the P2P network. When an existing machine is preempted, it is taken off from the P2P network and its data shard is re-distributed to other machines.

**Periodic centralized synchronization**  Complementary to the P2P decentralized parameter synchronization, Orpheus performs a centralized synchronization periodically. The centralized coordinator sets a global barrier every $R$ clocks. When all workers reach this barrier, the coordinator calls the `AllReduce(average)` interface to average the parameter replicas and set each replica to the average. After that, workers perform decentralized synchronization until the next barrier. Centralized synchronization effectively removes parameter-replicas’ discrepancy accumulated during decentralized execution and it will not incur substantial communication cost since it is invoked periodically.

**Rotation of parameter replicas**  Orpheus adopts data parallelism, where each worker has access to one shard of the data. Since computation is usually much faster than communication, the updates computed locally are much more frequent than those received remotely. This would render imbalanced updating of parameters: a parameter replica is more frequently updated based on the local data residing in the same machine than data shards on other machines. This is another cause of out-of-synchronization. To address this issue, Orpheus performs *parameter-replica rotation*, which enables each parameter replica to explore all data shards on different machines. Logically, the machines are connected via a ring network. Parameter-replicas rotate along the ring periodically (every $S$ iterations) while each data shard sits still on the same machine during the entire execution. We choose to rotate the parameters rather than data since the size of parameters is much smaller than data. A centralized coordinator sets a barrier every $S$ iterations. When all workers reach the barrier, it invokes the `Rotate` API which triggers the rotation of parameter replicas.

**Data prefetching**  The loading of training data from CPU to GPU is overlapped with the SF generator via a data queue. The next batches of training examples are prefetched into the queue
while the generator is processing the current one. In certain applications, each training example is associated with a data-dependent variable (DDV). For instance, in topic model, each document has a topic proportion vector. The states of DDVs need to be maintained throughout execution. Training examples and their DDVs are stored in consecutive host/device memory for locality and are prefetched together. At the end of a clock, GPU buffer storing examples is immediately ready for overwriting. The DDVs are swapped from GPU memory to host memory, which is pipelined using a DDV queue.

**Hardware/software-aware SF Transfer**

Orpheus provides a communication library for efficient message broadcasting. It contains a collection of broadcast methods designed for different hardware and software configurations, including (1) whether the communication is CPU-to-CPU or GPU-to-GPU; (2) whether InfiniBand [6] is available; (3) whether the consistency model is BSP or SSP.

- **CPU-to-CPU, BSP** In this case, we use the MPI Allgather [8] routine to perform all-to-all broadcast. In each clock, it gathers the SFs computed by each machine and distributes them to all machines. MPI Allgather is a blocking operation (i.e. the control does not return to the application until the receiving buffer is ready to receive SFs from all machines). This is in accordance with the BSP consistency model where the execution cannot proceed to the next clock until all machines reach the global barrier.

- **CPU-to-CPU, SSP** Under SSP, each machine is allowed to have a different pace to compute and broadcast SFs. To enable this, the all-to-all broadcast is decomposed into multiple one-to-all broadcast. Each machine separately invokes the MPI Bcast routine to broadcast its messages to others. MPI Bcast is a blocking operation: the next message cannot be sent until the current one finishes. This guarantees the SFs are received in order: SFs generated at clock \( t \) arrive early than those at \( t + 1 \). This order is important for the correctness of ML applications: the updates generated earlier should be applied first.

- **CPU-to-CPU, BSP, InfiniBand** An all-gather operation is executed by leveraging the Remote Direct Memory Access (RDMA) feature [318] provided by InfiniBand, which supports zero-copy networking by enabling the network adapter to transfer data directly to or from application memory, without going through the operating system. The recursive doubling (RD) [145] algorithm is used to implement all-gather, where pairs of processes exchange their SFs via point-to-point communication. In each iteration, the SFs collected during all previous iterations are included in the exchange. RDMA is used for the point-to-point transfer during the execution of RD.

- **CPU-to-CPU, SSP, InfiniBand** Each machine performs one-to-all broadcast separately, using the hardware supported broadcast (HSB) in InfiniBand. HSB is topology-aware: packets are duplicated by the switches only when necessary; therefore network traffic is reduced by avoiding the cases that multiple identical packets travel through the same physical link. The limitation of HSB is that messages can be dropped or arrive out of order, which degrades the correctness of ML execution. To retain reliability and in-order delivery, on top of HSB another layer of network protocol [232] is added, where (1) receivers send ACKs back to the root machine to confirm message delivery; (2) a message is re-transmitted using point-to-point
reliable communication if no ACK is received before timeout; (3) receivers use a continuous
clock counter to detect out-of-order messages and put them in order.

• **GPU-to-GPU** To reduce the latency of inter-machine SF transfer between two GPUs, we use
the GPUDirect RDMA [2] provided by CUDA, which allows network adapters to directly
read from or write to GPU device memory, without staging through host memory. Between
two network adaptors, the SFs are communicated using the methods listed above.

Similar to broadcast, several multicast methods tailored to different system configurations
are provided.

• **CPU-to-CPU** We use MPI group communication primitives for CPU-to-CPU multicast. In
each clock, `MPI_Comm_split` is invoked to split the communicator `MPI_COMM_WORLD` into
a target group (containing the selected machines) and a non-target group. Then the message is
broadcast to the target group.

• **CPU-to-CPU, InfiniBand** The efficient but unreliable multicast method supported by Infini-
Band at hardware level and a reliable point-to-point network protocol are used together. Infini-
Band combines the selected machines into a single *multicast address* and sends the message
to it. Point-to-point re-transmission is issued if no ACK is received before timeout. Since the
selection of receivers is random, any machine does not receive messages in continuous clocks
from another machine, making it difficult to detect out-of-order messages. We adopt a simple
approach: a message is discarded if it arrives late.

• **GPU-to-GPU** GPUDirect RDMA is used to copy buffers from GPU memory to network adaptor.
Then the communication between network adaptors is handled using the two methods
given above.

5.2.5 Evaluation

We evaluate Orpheus on three ML applications: multiclass logistic regression (MLR), topic
model (TM) [207], and long short-term memory (LSTM) [163] network.

Experimental Setup

**Cluster setup** We used two clusters: (1) a CPU cluster having 34 machines each with 64 cores
and 128 GB memory, connected by FDR10 Infiniband; (2) a GPU cluster having 40 machines
each with one TitanX GPU and 64GB memory, connected by 40Gbps Ethernet. We trained MLR
and TM on the CPU cluster and trained LSTM on the GPU cluster. Unless otherwise noted, the
experiments were performed using all machines in each cluster.

**Baseline systems** We compared with (1) Spark-MLR from Spark MLlib-2.0.0, which is based
on an L-BFGS algorithm (Spark-MLR has an SGD-based implementation, which converges
slower than L-BFGS); (2) Bosen-MLR and Bosen-TM: MLR and TM implemented using a
recent parameter server (PS) Bosen [355], with additional system features (e.g., parameter-
update filtering) borrowed from another PS [226]; (3) TensorFlow-MLR and TensorFlow-LSTM:
TensorFlow-1.0 implementation of MLR and TensorFlow-1.7 implementation of LSTM, which
are partially based on PS; (4) MXNet-MLR and MXNet-LSTM: MXNet-0.7 [73] implementation of MLR and MXNet-1.1 implementation of LSTM, using PS-based data parallelism; (5) Gopal-MLR [136]: a model-parallel MLR based on Hadoop; (6) FlexiFaCT-TM [207]: a Hadoop implementation of TM. Note that Spark and TensorFlow implement a probabilistic topic model – latent Dirichlet allocation [49], which is different from the non-probabilistic TM in Orpheus.

Utilization of system features All system features except elasticity were used in the experiments. Most broadcast and multicast methods were utilized since our experiments span various hardware-level configurations (CPU, GPU, InfiniBand, Ethernet) and software-level settings (broadcast, multicast, BSP, SSP).

Datasets Three datasets were used in the experiments: Wikipedia [272], PubMed [9], and 1B-Word [67]. The Wikipedia dataset contains ∼2.4 million documents from 325K classes. Documents are represented with 20K-dimensional bag-of-words vectors. The PubMed dataset contains 8.2M documents and ∼0.74B words. The vocabulary size is 141K. The 1B-Word dataset contains ∼0.8B words with a vocabulary size of ∼0.8M. The MLR, TM, and LSTM experiments were conducted on the Wikipedia, PubMed, and 1B-Word dataset respectively.

ML and system hyper-parameter setup The topic number in TM and the state-vector dimension in LSTM was set to 50K and 40K respectively. As a result, the parameter matrix in MLR, TM, and LSTM has a size of 325K×20K, 50K×141K, 40K×40K, containing ∼6.5B, ∼7.1B, ∼1.6B entries respectively. The parameters in all applications were trained using the SGD algorithm with a mini-batch size of 100. In SF selection, the number of selected SFs was set to 25. In random multicast, the number of destinations each machine sends message to was set to 4. The consistency model was set to SSP with a staleness value of 4.

Overall Results

Comparison with other systems We first compare the convergence speed of these systems. In this comparison, no system uses fault tolerance. By convergence, it means the loss function
Table 5.2: The second and third columns show the convergence time (hours) of each system, under a different number of machines. The third column shows the speedup of each system when the number of machines is increased from 12 to 34 (in MLR and TM), or from 12 to 40 (in LSTM).

value (e.g., cross-entropy loss in MLR, negative log-likelihood in TM and LSTM) on the training set levels off. A better system takes less time to converge. In each of our experiments, different systems converged to the same loss value. Figure 5.6 shows the convergence curves for MLR (34 machines). The second and third columns of Table 5.2 shows the convergence time of each system, under a different number of machines. On all three models, Orpheus converges faster than the baseline systems.

We first compare Orpheus with parameter server (PS) based systems including Bosen [355], TensorFlow-1.0 [13], and MXNet-0.7 [73]. On MLR, with 34 CPU machines, the speedup of Orpheus over Bosen, TensorFlow-1.0, MXNet-0.7 and SFB is 3.7x, 4.7x, 4.4x and 1.4x respectively. On TM, with 34 CPU machines, Orpheus is 4.4x and 1.8x faster than Bosen and SFB respectively. On LSTM, with 40 GPU machines, Orpheus is 1.4x faster than TensorFlow-1.7 and 1.1x faster than MXNet-1.1. As we will further show later, these speedups are achieved mainly because Orpheus is more efficient in communication. Compared with SFB, Orpheus reduces the number of sent SFs using SF selection and provides a random multicast scheme which works better than the deterministic multicast scheme utilized in SFB. Similar to SFB, Orpheus transmits small-sized vectors instead of large-sized matrices, which greatly reduces network traffic, compared with PS-based systems.
Next, we compare Orpheus with the rest of baseline systems. Using 34 CPU machines, Orpheus is 10.3x and 7.9x faster than Spark and Gopal on MLR, and 6.3x faster than FlexiFaCT on TM. Gopal outperforms Spark possibly because it uses a better distributed-algorithm which is based on model-parallelism. However, it is slower than PS systems due to the disk IO overhead of Hadoop. So is FlexiFaCT, which is a Hadoop-based system. Spark is at least two times slower than PS systems (implemented with C++) due to the overhead incurred by resilient distributed datasets and Java virtual machine.

Scalability We evaluated how Orpheus scales up as the number of machines increases. The results on MLR and LSTM are shown in Figure 5.7. With 34 CPU machines, Orpheus achieved a 27.3x speedup on MLR. With 40 GPU machines, a 32.2x speedup is achieved on LSTM. The scalability of Orpheus is better than baseline systems, as shown in the fourth column of Table 5.2, where we measured the speedups for MLR and TM when the number of CPU/GPU machines increases from 12 to 34 and the speedups for LSTM when the number of GPU machines increases from 12 to 40. For example, in LSTM experiments with 40 GPU machines, Orpheus achieves a speedup of 2.9 compared with using 12 machines.

Evaluation of individual components

In this section, we evaluate the impact of each individual component. We compare the following systems: (1) Matrix+PS: synchronizing parameter copies by transmitting full matrices using a parameter server (PS) architecture; (2) SFB: SF broadcasting (SFB); (3) SFB+SFS: adding SF selection (SFS) to SFB; (4) SFB+SFS+RM: adding random multicast (RM); (5) SFB+SFS+RM+PCS: adding periodic centralized synchronization (PCS); (6) SFB+SFS+RM+PRR: adding parameter-replicas rotation (PRR). In these systems, no checkpointing is used. The number of machines in MLR, TM, and LSTM is set to 34, 34 and 40 respectively. Table 5.3 shows the convergence time of these systems, where we make the following observations. First, SFB is much more efficient than Matrix+PS. SFB is 2.6x, 2.4x and 2x faster than Matrix+PS on MLR, TM, and LSTM respectively. The reason is: SFB transmits small-sized vectors while Matrix+PS transmits large matrices; the communication cost of SFB is much smaller. Second, adding SFS...
<table>
<thead>
<tr>
<th>Configuration</th>
<th>MLR</th>
<th>TM</th>
<th>LSTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix+PS</td>
<td>7.1</td>
<td>23.5</td>
<td>16.8</td>
</tr>
<tr>
<td>SFB</td>
<td>2.7</td>
<td>9.7</td>
<td>8.6</td>
</tr>
<tr>
<td>SFB+SFS</td>
<td>2.3</td>
<td>7.9</td>
<td>6.9</td>
</tr>
<tr>
<td>SFB+SFS+RM</td>
<td>2.1</td>
<td>6.6</td>
<td>5.5</td>
</tr>
<tr>
<td>SFB+SFS+RM+PCS</td>
<td>2.0</td>
<td>5.9</td>
<td>4.9</td>
</tr>
<tr>
<td>SFB+SFS+RM+PCS+PRR</td>
<td>1.9</td>
<td>5.4</td>
<td>4.1</td>
</tr>
</tbody>
</table>

Table 5.3: Convergence time (hours) of different system configurations.

Figure 5.8: Breakdown of network waiting time (hours) and computation time (hours).

to SFB further reduces the runtime – by 15%, 19%, and 20% – on MLR, TM, and LSTM respectively. SFS selects a subset of SFGs for communication, which reduces network traffic. Third, incorporating RM reduces the runtime of SFB+SFS by 9%, 16%, and 20% on the three applications. Under RM, in each clock, each machine selects a subset of machines to send SFs to, which reduces the number of network messages. Fourth, adding PCS further speeds up convergence. Via PCS, the incoherence among different parameter copies is alleviated, which reduces noise and improves convergence quality. Fifth, comparing the last two rows of this table, we confirm the effectiveness of PRR in reducing convergence time. Under PRR, each parameter replica has the chance to explore all data shards on different machines, which facilitates symmetric update of parameters.

Figure 5.8 shows the breakdown of network waiting time and computation time for four configurations. Compared with Matrix+PS, SFB greatly reduces network waiting time by avoiding transmitting matrices. It slightly increases the computation time since the same SFG needs to be converted into an update matrix at each worker. Adding SFS further decreases network time since it reduces the number of transmitted SFs. SFS causes the increase of computation time because of the overhead of executing the JMCSS algorithm (Algorithm 3). The network time is further reduced by using RM, which decreases the number of network messages. RM has little impact on the computation time.

In the sequel, we present more detailed evaluations of several key components.
**Random multicast (RM)** Figure 5.9 (left) shows the convergence time of MLR under varying $Q$ – the number of destinations each machine sends messages to. As can be seen, communicating with all machines (i.e., $Q = 33$) incurs much more running time than using random multicast ($Q = 4$), which demonstrates the effectiveness of RM in improving communication efficiency. The efficiency results from RM’s ability of reducing the number of network messages. However, $Q$ cannot be too small. Otherwise, the running time increases (e.g., when $Q = 1$), due to the severe synchronization delays.

We compared RM with two multicast schemes: (1) deterministic multicast (DM) [222] where each sending machine sends messages to a fixed set of 4 receiving machines; the set of 4 receiving machines is different for each sending machine and is chosen to balance network load across the cluster and prevent network hot spots; (2) round-robin multicast (RRM) [222, 353] where every two workers communicate with each other periodically according to a deterministic circular order. Figure 5.10 (left) shows the convergence time of MLR and LSTM under DM, RRM, and RM. In both applications, RM takes less time to converge. Compared with DM, RM uses a randomly changing multicast topology to enable each pair of machines to have some chance to communicate directly, thus facilitating more symmetric (hence faster) synchronization of parameter replicas. Compared with RRM, the randomness of RM facilitates faster “mixing” [148] of parameter copies and hence speeds up convergence.

To examine the robustness of RM against network connection failures, we simulated the effect that in each clock the connections between 10% of machine pairs are “broken” randomly. Figure 5.10 (right) shows the relative increase of convergence time when failure happens. As can be seen, the relative increase under RM is much smaller than that under DM and RRM, confirming that RM is more robust, due to its random nature.

**SF selection** Figure 5.9 (right) shows the convergence time for MLR under varying $C$ – the number of selected SFs. Compared with using the entire batch ($C = 100$), selecting a subset (e.g., $C = 25$) of SFs to communicate significantly speeds up convergence, thanks to the reduced network traffic. On the other hand, $C$ cannot be too small, which otherwise incurs large approximation errors in parameter updates. For example, the convergence time under $C = 5$ is
Figure 5.10: (Left) Convergence time in Orpheus under deterministic, round-robin, and random broadcast for MLR and LSTM. (Right) Relative increase of convergence time when network connection fails.

Table 5.4: Convergence time (hours) under different configurations of fault tolerance and recovery.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>MLR</th>
<th>LSTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>No checkpoint</td>
<td>1.9</td>
<td>4.1</td>
</tr>
<tr>
<td>Matrix-based checkpoint</td>
<td>2.4</td>
<td>5.2</td>
</tr>
<tr>
<td>SF-based checkpoint</td>
<td>1.9</td>
<td>4.2</td>
</tr>
<tr>
<td>Matrix-based recovery</td>
<td>2.9</td>
<td>6.7</td>
</tr>
<tr>
<td>SF-based recovery</td>
<td>2.3</td>
<td>4.8</td>
</tr>
</tbody>
</table>

worse than that under $C = 100$.

**Fault tolerance and recovery** We compare the following configurations: (1) no checkpoint; (2) matrix-based checkpoint: every 100 clocks, Orpheus saves the parameter matrix onto the disk; when saving matrices, the computation halts to ensure consistency; (3) SF-based checkpoint; (4) matrix-based recovery: we simulated the effect that in each clock, each machine fails with a probability of 0.01; when failure happens, a recovery is performed based on the checkpointed matrices; (5) SF-based recovery: machine failure is simulated in the same way as (4) and recovery is based on the saved SFs. For (2) and (3), no machine failure happens.

Table 5.4 shows the convergence time of MLR (34 machines) and LSTM (40 machines) under different configurations. From this table, we observe the following. First, compared with no-checkpoint, matrix-based checkpoint method substantially increases the convergence time while our SF-based checkpointing incurs very little increase. The reasons are twofold: (1) saving vectors consumes much less disk bandwidth than saving matrices; (2) checkpointing SFs does not halt computation, wasting no compute cycles, which is not the case in checkpointing matrices. Second, in the case where machine failure happens, matrix-based recovery causes more slowing-down of convergence than SF-based recovery. This is because in matrix-based recovery, the parameters can only be rolled back to the state that is saved every 100 clocks. If the failure happens at clock 199, the parameters are rolled back to the state saved at clock 100. The com-

\footnote{Choosing a number smaller than 100 would entail more disk-IO waiting.}
Computation from clock 101 to 198 are wasted. However, in SF-based recovery, the parameter state after every clock is preserved. It can always roll back to the latest parameter state (e.g., the state after clock 198) and no computation is wasted.

5.3 Poseidon: A Hybrid Architecture of SFB and Parameter Server

Many ML models are parameterized by multiple matrices. For example, in a convolutional neural network (CNN), the convolution layers and fully-connected layers have their own layer-specific weight matrices. Some of these matrices are of large size (in the sense that they have thousands of rows and columns at least) while others are small-sized (hundreds of rows/columns). For instance, in the first convolutional layer of AlexNet [203], the weight matrix of 96 filters has a size of $96 \times 363$ which is relatively small while that in the fully-connected (FC) layer is $4096 \times 4096$ which is much larger.

When these weight matrices are trained using stochastic gradient descent (SGD), their updates can be constructed by vectors, i.e., having the sufficient factor property. Take the $J \times D$ weight matrix $W$ between two FC layers $l_i$ and $l_{i+1}$ as an example. In the forward pass, one data sample is fed into the network and the activations of layer $l_i$ is produced as $a_i$. During back-propagation (BP), an error message $e_{i+1}$, which is a $J$-dimensional vector, is passed back from $l_{i+1}$ to $l_i$. The gradients $\Delta W$ thus can be exactly reconstructed by two vectors $e_{i+1}$ and $a_i$: $\Delta W = e_{i+1} a_i^T$. As a result, these weight matrices can be learned using the sufficient factor broadcasting (SFB) architecture of Orpheus.

In Section 5.2.1, we present a cost analysis of SFB and parameter server (PS) architectures [94, 226, 355]. The analysis shows that SFB outperforms PS if the matrix size is significantly larger (at least 10 times) than the mini-batch size in SGD and the number of worker machines, and PS has an advantage over SFB if otherwise. In CNN, both cases exist, which motivates us to design a hybrid communication protocol that adopts both SFB and PS, based on which a distributed deep learning system called Poseidon [411] is built.

5.3.1 Structure-aware Message Passing (SACP) Protocol

We propose the SACP [411] protocol, which hybridizes the client-server PS scheme with the peer-to-peer SFB scheme, for GPU-based distributed deep learning. In addition to PS and SFB, SACP also supports SF-based PS (SFPS) [78]. SFPS is much like a PS except that instead of transmitting update matrices from workers to the server as PS does, SFPS transmits SFs generated on the workers to the server and transform these SFs into update matrices at the server. Figure 5.11 illustrates the three communication schemes.

The SACP is structure-aware, as it intelligently determines the optimal communication method, according to the matrix size $J \times D$, the SGD batch size $K$, and the number of workers $P$. Recall that under the bulk synchronous parallel consistency model where all workers proceed at the same pace, the communication costs of PS, SFB, and SFPS are $2PJD$, $(P - 1)^2 K(J + D)$, $PK(J + D) + PJ$ respectively. In general, SFB and SFPS have a lower cost than PS when the
matrix size is large, and the mini-batch size and the number of workers are small.

In CNN, a convolution layer has a $J \times D$ weight matrix, where $J$ is the number of filters and the $D$ is the size of the local receptive field. $J$ and $D$ are usually small. For example, in the first convolution layer of AlexNet, $J$ and $D$ are 96 and 363 respectively. These filters slide over a grid of locations in each image. At each location, an update matrix of the weights is generated, which can be written as the outer-product of two vectors (SFs). Let $L$ denote the number of locations in each image, then the costs of SFB and SFPS would be $(P - 1)^2 KL(J + D), PKL(J + D) + PJD$. The value of $L$ could be several thousand or more (e.g., in the first convolution layer of AlexNet, $L$ equals 3136), which renders the costs of SFB and SFPS to be larger than PS’s cost that is independent of $L$. Therefore, we choose PS for synchronizing weight matrices in convolution layers.

For fully-connected (FC) layers, their weight matrices are in general much larger than those in convolutional layers. In AlexNet, $J$ and $D$ are 4096 in the FC layers. Different from the convolutional layers where multiple update matrices are generated at different locations of an image, in FC layers there is only one update matrix for each image. The mini-batch size $K$ is usually around 100. Putting these pieces together, we conclude that in most cases the costs of SFB and SFPS are lower than that of PS. Hence, we choose either SFB or SFPS to communicate matrices in FC layers. To choose between SFB and SFPS, we directly compare their costs $(P - 1)^2K(J + D), PK(J + D) + PJD$, and select the one yielding lower cost. Algorithm 5 summarizes how SACP works.

5.3.2 Evaluation

We evaluated Poseidon on image classification tasks and used three datasets: CIFAR-10 [5], ILSFRC-2012 [95] which contains about one million ImageNet [95] images from 1000 classes, and ImageNet-22K which contains all ImageNet images from 22K classes. The experiments show that Poseidon significantly accelerates the training of CNNs, while guaranteeing the correct convergence. Ablation studies were performed to justify the effectiveness of SACP.
Algorithm 5: Structure-aware communication protocol (SACP).

At iteration $t$ on worker $p$:

Input: Layer $l_i$, $J \times D$ gradient matrix $\triangle W^p_i$, number of workers $P$, batch size $K$.

Task: Push out gradients $\triangle W^p_i$ and then update $W^p_i$.

if $l_i$ is not an FC layer then
  Send $\triangle W^p_i$ to the master node.
  Synchronize updated $W_i$ from the master node.
else
  Recast $\triangle W^p_i$ into two SFs, i.e. $\triangle W^p_i = u^p_i v^p_i \top$.
  if $(P-1)^2 K (J+D) \leq PK (J+D) + PJD$ then
    Broadcast $u^p_i, v^p_i$ to all other workers.
    Receive SFs $u^j_i, v^j_i, j \neq p$ from all other workers.
    Update $W_i$: $W_i \leftarrow W_i + \sum_j u^j_i v^j_i \top$.
  end
else
  Send $u^p_i, v^p_i$ to the master node.
  Synchronize updated $W_i$ from the master node.
end
end

Cluster Configuration

We conducted all experiments on the PRObE Susitna cluster [235], where each node has a 4×16-core 2.1GHz AMD Opteron 6272 CPU, 128GB of RAM, and an NVIDIA Tesla K20C GPU with 4799MB memory. All cluster nodes have shared access to a network file system (NFS) with 1x Hitachi 1.0 TB HDD and 2x Hitachi 3.0 TB HDD. We used the 40GbE network. The Caffe framework [182] (the Oct 2014 version) was used as the single node baseline, and it was modified using Poseidon’s client library API for distributed execution.

Image Classification

We demonstrate Poseidon’s performance on three benchmark datasets.

Classification on CIFAR-10  We first evaluate Poseidon on the CIFAR-10 [5] dataset, which contains $32 \times 32$ images from 10 classes. Each class has 6K images. An official train/test split is provided where 50K images are used for training and 10K for testing.

We employed the built-in `cifar10_quick_solver` and `cifar10_quick_train_test` network structure in Caffe[3], consisting of 3 convolutional (CONV) layers and 1 fully-connected (FC) layer followed by a 10-way softmax classifier. This network has 145,578 parameters in total. It converges to a 70% test accuracy with 4 epochs of training in a single machine without decreasing

`github.com/BVLC/caffe/tree/master/examples/cifar10`
Figure 5.12: The comparison of training different CNNs to convergence between Poseidon on 8 GPU nodes and Caffe on a single node: (a)-(b) cifar10_quick_train_test; (c)-(d) bvlc_alexnet; (e)-(f) bvlc_googlenet. Test errors are shown with respect to (left) training time, and (right) training iterations.

The learning rate. We deployed Poseidon onto 8 Susitna nodes. Since a larger batch size usually hurts the SGD performance, we reduced the batch size from 100 to 50 and also slightly decreased the base learning rate from 0.01 to 0.007, while keeping other hyperparameter settings in cifar10_quick Solver unchanged. For better comparison, in the distributed setting, we used the bulk synchronous parallel consistency model.

Similar to the single machine setting, we trained the network to convergence without adjusting the learning rate. The test accuracy achieves nearly 75%. Figure 5.12(a)-(b) plot how the test error decreases along with training time and iterations for Poseidon on 8 nodes and Caffe on a single node. Under the same setting, the single-machine Caffe takes more than 4 times of training time to converge to 70% accuracy, while Poseidon quickly converges to 72% in 19 seconds and attains a higher accuracy 75% in 25 seconds with 8 GPU nodes.
<table>
<thead>
<tr>
<th>Model</th>
<th>Speedup</th>
<th>Top-1 accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10 quick</td>
<td>4×</td>
<td>75%</td>
</tr>
<tr>
<td>AlexNet</td>
<td>4.5×</td>
<td>56.5%</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>4×</td>
<td>67.1%</td>
</tr>
</tbody>
</table>

Table 5.5: Speedups and converged top-1 accuracies of Poseidon by training models on 8 GPU nodes and on the CIFAR-10 and ILSFRC-2012 dataset, compared to Caffe on a single machine.

**Classification on ILSFRC-2012** We then experiment on ImageNet ILSFRC-2012, consisting of 1.28 million training images and 50K validation images on 1,000 categories. We downsampled all images to 256 × 256 × 3 before feeding them into the networks, and report the top-1 accuracy on the validation set.

We evaluate Poseidon on AlexNet [203] and GoogLeNet [319]. The AlexNet has 5 CONV layers, 2 FC layers, and a 1000-class softmax classifier. It contains 61.3 million parameters in total. GoogLeNet has 22 layers and 5 million parameters. For fair comparisons, we employed the open implementations of AlexNet [203] and GoogLeNet [319] provided in Caffe, denoted as bvlc_alexnet and bvlc_googlenet. In single machine training of AlexNet, we used the standard solver in Caffe, where the batch size was 256 and the number of epochs was 70. During training, the learning rate was decreased 3 times. For the training of GoogLeNet, we employed the quick_solver, which uses the polynomial learning rate policy, and trains for 60 epochs with a batch size of 32. In the distributed setting, we deployed both AlexNet and GoogLeNet onto 8 GPU nodes using Poseidon, and kept the network structure and the batch size exactly the same as those in the single machine setting. Specifically, for AlexNet, we trained it on 8 nodes for about 60K iterations, with the base learning rate set to 0.005, which was decreased 5 times during the entire training process. For GoogLeNet, we used a standard step learning rate policy by setting the base learning rate to 0.005 and decreasing it 90 times during training.

Figure 5.12(c)-(d) and Figure 5.12(e)-(f) show the performance of training AlexNet and GoogLeNet using Poseidon with a GPU cluster of 8 nodes, compared to single-machine Caffe, respectively. For AlexNet, Poseidon attains 56.5% top-1 accuracy on the validation set after 27 hours of training, achieving a 4.5× speedup against single machine Caffe which needs 5 days to achieve such an accuracy. For GoogLeNet, Poseidon converges to 67.1% top-1 accuracy after 130 hours of training, whereas single-machine Caffe only achieves 50% top-1 accuracy after 250 hours of training and 57% after near 350 hours of training (Poseidon only needs less than 48 hours to achieve 50% and 75 hours to achieve 57%, with a near 5× speedup).

The speedups achieved by Poseidon in terms of algorithm convergence are summarized in Table 5.5. Besides, we report the speedups on throughput (i.e. number of images processed per seconds) in Figure 5.13 when training AlexNet and GoogLeNet using Poseidon on different numbers of GPU nodes, compared to single machine Caffe. The staleness synchronous parallel [162] consistency model was used and the throughput speedups are compared under different settings of the staleness value.

[github.com/BVLC/caffe/tree/master/models/bvlc_alexnet](https://github.com/BVLC/caffe/tree/master/models/bvlc_alexnet)
[github.com/BVLC/caffe/tree/master/models/bvlc_googlenet](https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet)
Figure 5.13: The speedups on throughput with different values of staleness, when training using Poseidon on 8 nodes, compared to Caffe on a single node. (a) AlexNet with batch size 256, and (b) GoogLeNet with batch size 32.

<table>
<thead>
<tr>
<th>Framework</th>
<th>Data</th>
<th># machines/cores</th>
<th>Time</th>
<th>Train accuracy</th>
<th>Test accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Poseidon</td>
<td>7.1M ImageNet-22K images for training, 7.1M for testing</td>
<td>8/8 GPUs</td>
<td>3 days</td>
<td>41%</td>
<td>23.7%</td>
</tr>
<tr>
<td>Adam [78]</td>
<td>7.1M ImageNet-22K images for training, 7.1M for testing</td>
<td>62 machines/?</td>
<td>10 days</td>
<td>N/A</td>
<td>29.8%</td>
</tr>
<tr>
<td>MxNet [73]</td>
<td>All ImageNet-22K images for training, no testing results</td>
<td>1/4 GPUs</td>
<td>8.5 days</td>
<td>37.2%</td>
<td>N/A</td>
</tr>
<tr>
<td>Le et al. [214] w/ pretrain</td>
<td>7.1M ImageNet-22K and 10M unlabeled images for training, 7.1M for testing</td>
<td>1,000/1,600 CPU cores</td>
<td>3 days</td>
<td>N/A</td>
<td>15.8%</td>
</tr>
</tbody>
</table>

Table 5.6: Comparisons of image classification results on ImageNet-22K.

Classification on ImageNet-22K

ImageNet-22K is the largest public dataset for image classification, including 14,197,087 labeled images from 21,841 categories, which is rarely touched by the research community due to its massive data size and complexity. We experiment on ImageNet-22K to demonstrate the scalability of Poseidon. As no official test data exists for evaluation, following previous settings in [78, 94, 214], we randomly split the whole set into two parts, and used the first 7.1 million images for training and the rest for testing. Similar to ILSFRC-2012, we resized all images to $256 \times 256$ and report the top-1 test accuracy.

We designed an AlexNet-like architecture. Specifically, the CNN takes a random $227 \times 227$ crop from the original image as input, and forwards it into 5 CONV layers and 2 FC layers before prediction. The CNN has convolution filters with sizes $7 \times 7$, $5 \times 5$, and $3 \times 3$. Similar to AlexNet, the first, second, and fifth CONV layers are followed by max pooling layers with size $3 \times 3$ and stride 2. Two FC layers with 3,000 neurons each are put at the top of the network, followed by a softmax layer for 21,841-way classification. We trained the CNN with data-parallelism by equally partitioning the training data into 8 GPU nodes. The batch size was set to 256. The network was trained using the step learning rate policy, with a base learning rate of 0.005, which is decreased 6 times during training.

Table 5.6 compares our results to those achieved by previous works including Adam [78],

170
MXNet [73], and Le et al. [214]. Note that at this point a complete fair comparison between different frameworks is not possible, because the experiment protocol of ImageNet-22K is not standardized, the source codes are not fully available yet, and large variations exist in system configurations, models, and implementation details. However, in general Poseidon achieves a competitive accuracy 23.7% with shorter training time and less machines. Our system achieves 23.7% accuracy which is close to that achieved by Adam [78], using a model whose size is close to that of Adam. But our system takes much less training time (30% of Adam) and much fewer machines (13% of Adam). Compared with MXNet which achieves 37.2% training accuracy with 8.5 days of training on 4 GPUs, our system achieves a higher training accuracy (41%) with 3 days of training on 8 GPUs.

**Internal Comparison**

In this section, we conduct internal comparison to validate the effectiveness of SACP in reducing communication cost for GPU-based distributed deep learning. We measure the speedups against the single-machine Caffe in terms of throughput, which is defined as the number of images processed per second. Figure 5.14 compares the speedups for training AlexNet and GoogLeNet under the following two settings with different number of nodes: (1) w/o SACP: weight matrices in all layers are synchronized using PS; (2) w/ SACP: SACP is enabled. We used the BSP consistency model and set the batch size to 256 for AlexNet and 32 for GoogLeNet (different batch sizes will lead to slightly different speedups on throughput). With SACP enabled, the speedup on throughput is improved. Particularly, when training on 8 nodes, SACP greatly increases the speedups of AlexNet training from 4 to 6, with a 50% improvement. For GoogLeNet with fewer FC layers, SACP achieves approximately 20% improvement on the speedup.

### 5.4 Convergence Analysis

In this section, we analyze the convergence of algorithms executed using SFB. We first study the convergence of mini-batch SGD under full broadcasting SFB for nonconvex optimization. Since SFB is a peer-to-peer decentralized computation model, we need to show that parameter copies
on different workers converge to the same limiting point without a centralized coordination, even under delays in communication due to bounded asynchronous execution. In this respect, our analysis differs from that of centralized parameter server systems [162], which instead show convergence of global parameters on the central server. More specifically, We are interested in solving the optimization problem \( \min_W f(W) \), where \( f(W) \) corresponds to the loss of a set of training samples.

Consider a distributed system with \( P \) machines. Each machine \( p \) keeps a local variable \( W_p \). We assume that the total loss \( f \) is distributed among the \( P \) worker machines, i.e., \( f = \sum_{p=1}^P f_p \), and \( f_p \) denotes the loss stored in the \( p \)-th machine. Given a parameter matrix \( W \) and a random variable \( \xi \), each machine \( p \) has access to a stochastic gradient oracle \( G_p(W, \xi) \) that satisfies \( E\xi G_p(W, \xi) = \nabla f_p(W) \). We further assume that every machine calls the stochastic oracle \( K \) times at each iteration (\( K \) is also referred to as the mini-batch size) and generates the stochastic gradient \( \frac{1}{K} \sum_{j=1}^K G_p(W_p, \xi_{c,j}) \), where \( W_p \) denotes the local variable of machine \( p \) at iteration \( c \).

At each iteration, machine \( p \) updates its local variable by aggregating the stochastic gradients from all machines and performing a proximal update. Specifically, the aggregated stochastic gradients on machine \( p \) at iteration \( c \) is \( G_p^c = \sum_{q=1}^P \frac{1}{K} \sum_{j=1}^K G_q(W_{q}^{\tau_q^p(c)}, \xi_{q,c,j}) \), where \( \tau_q^p(c) \) is the number of iterations machine \( q \) has transmitted to machine \( p \) when machine \( p \) conducts its \( c \)-th iteration. We assume that \( \tau_q^p(c) = c \) and \( 0 \leq c - \tau_q^p(c) \leq s \), i.e., machine \( p \) has access to the fresh information on itself, and receives delayed updates of machine \( q \) that are within \( s \) iterations. Denote the stepsize as \( \eta > 0 \), the local update rule of machine \( p \) is expressed as

\[
W_{p}^{c+1} = W_{p}^{c} - \eta_c G_p^c, \quad \text{where } 0 \leq c - \tau_q^p(c) \leq s. \tag{5.3}
\]

This formulation, in which \( s \) is the maximum “staleness” allowed between any update and any worker, covers bulk synchronous parallel broadcasting (\( s = 0 \)) and bounded-asynchronous broadcasting (\( s > 0 \)). We also define an undelayed version of the aggregated stochastic gradient \( G^c = \sum_{q=1}^P \frac{1}{K} \sum_{j=1}^K G_q(W_{q}^c, \xi_{c,j}) \). We adopt the following assumptions for our analysis.

**Assumption 1.** (1) For all \( p = 1, \ldots, P \), \( f_p \) is continuously differentiable and \( F \) is bounded from below; (2) \( \nabla f_p \), \( \nabla f_p \) are Lipschitz continuous with constants \( L_f \) and \( L_p \), respectively, and let \( L = \sum_{p=1}^P L_p \); (3) There exists \( B, \sigma^2 > 0 \) such that for all \( p \) and \( c \), we have \( \|W_p^c\|_F \leq B \) and \( E\|G^c - \sum_{p=1}^P \nabla f_p(W_p^c)\|^2_F \leq \frac{\sigma^2}{K_P} \) almost surely.

The assumptions in item 1 and 2 are standard smoothness assumptions for gradient methods. In the case of full synchronization (i.e., \( W_p^c = W^c \) for all \( p \)), the assumption in item 3 reduces to the standard bounded variance assumption for mini-batch SGD methods. Our analysis is based on the following auxiliary update sequence

\[
W_{p}^{c+1} = W_{p}^{c} - \eta_c G_p^c. \tag{5.4}
\]

Compare to the local update in Eq. (5.3) on machine \( p \), essentially this auxiliary update uses the undelayed aggregated stochastic gradients \( G_p^c \) generated by all machines, instead of the delayed ones \( G_p^c \). We show that all local machine parameter sequences \( \{W_p^c\} \) are asymptotically consistent with this auxiliary sequence \( \{W^c\} \), and their limit points are all stationary points of \( f \).

**Theorem 6.** Denote \( \{W_p^c\}, p = 1, \ldots, P \), and \( \{W^c\} \) as the local sequences and the auxiliary sequence generated by SFB, respectively. Let Assumption 1 hold. Then, with learning rate \( \eta_c = O\left(\frac{K_P}{(L_S + L_f)\sigma^2\sqrt{c}}\right) \), we have
\begin{itemize}
  \item \( \lim_{c \to \infty} \max_p \| W^c - W_p^c \|_F = 0 \), i.e. the maximal disagreement between all local sequences and the auxiliary sequence converges to 0;
  \item We have \( \min_{c \leq C} \mathbb{E}\| \sum_{p=1}^P \nabla f_p(W_p^c) \|^2 \leq O \left( \frac{(L_s + L_f)\sigma^2}{KPC} \right) \). Moreover, there exists a common subsequence of \( \{ W_p^c \} \) and \( \{ W^c \} \) that converges almost surely to a stationary point.
\end{itemize}

Intuitively, Theorem 6 says that, given a properly-chosen learning rate, all local worker parameters \( \{ W_p^c \} \) eventually converge to stationary points (i.e. local minima) of the objective function \( F \), despite the fact that SF transmission can be delayed by up to \( s \) iterations. Thus, SFB learning is robust even under bounded-asynchronous communication (such as SSP). Our analysis differs from [41] in two ways: (1) Bertsekas and Tsitsiklis [41] explicitly maintain a consensus model which would require transmitting the parameter matrix among worker machines — a communication bottleneck that we were able to avoid; (2) we allow subsampling in each worker machine. Accordingly, our theoretical guarantee is probabilistic, instead of the deterministic one in [41].

We now consider the multicast setting, where each machine updates its local variable by aggregating the stochastic gradients from a subset of the other machines. Specifically, we denote \( \delta_p^q(c) \in \{ 0, 1 \} \) as whether machine \( p \) and machine \( q \) are connected at \( c \)-th iteration on machine \( p \). The aggregated stochastic gradients on machine \( p \) at iteration \( c \) is denoted as \( G_p^c(\delta) = \sum_{q=1}^P \delta_p^q(c) \frac{1}{K} \sum_{j=1}^K G_q(W_q^t(c), \xi_{p,q}(c), j) \). We assume that each machine is connected to \( Q \) number of other machines at each iteration, i.e., \( \sum_{q=1}^P \delta_p^q(c) = Q \) for all \( p \) and \( c \). Denote the stepsize as \( \eta > 0 \), the local update rule of machine \( p \) is expressed as

\[
W_p^{c+1} = W_p^c - \eta c G_p^c(\delta), \quad \text{where} \quad 0 \leq c - \tau_p^q(c) \leq s, \quad \sum_{q=1}^P \delta_p^q(t) = Q. \tag{5.5}
\]

We obtain the following result for the multicast setting.

**Corollary 1.** Denote \( \{ W_p^c \}, p = 1, \ldots, P, \) and \( \{ W^c \} \) as the local sequences and the auxiliary sequence generated by multicast SFB, respectively, and assume that each machine aggregates the stochastic gradients from \( Q < P \) other machines at every iteration. Let Assumption 7 hold. Then, with learning rate \( \eta_c \equiv O\left( \frac{1}{L(P-Q)C} \right) \), we have

\[
\min_{c=1, \ldots, C} \mathbb{E}\| \sum_{p=1}^P \nabla f_p(W_p^c) \|^2 \leq O \left( L(P-Q) + \frac{(L_s + L_f)\sigma^2}{KPL(P-Q)C} \right). \]

In the multicast setting, we need to use a smaller stepsize \( \eta_c = O\left( \frac{1}{L(P-Q)C} \right) \) to compensate the errors caused by multicast SFB. The algorithm eventually converges to an \( O(L(P-Q)) \) neighbourhood of stationary point, i.e., if we broadcast the stochastic gradient updates to more machines, the variable sequence is more close to a stationary point. In the experiments, we observe that SFB under multicast converges to the same objective value as SFB under broadcast with a comparable convergence, while multicast can significantly reduce the communication cost.
5.4.1 Appendix: Proof of Theorem 6

Proof. Throughout, we denote $g_c := \sum_{p=1}^{P} \nabla f_p(W^c_p) = \mathbb{E} G^c$. By the descent lemma [41], we have

$$f(W^{c+1}) - f(W^c) \leq \langle W^{c+1} - W^c, \nabla f(W^c) \rangle + \frac{L_f}{2} \| W^{c+1} - W^c \|_F^2$$

where (i) follows from the observation that $W^{c+1} - W^c = -\eta_c G^c$. Taking expectations on both sides, and note that $\mathbb{E} G^c = g_c$, $\mathbb{E} \| G^c \|^2 \leq \| g_c \|^2 + \frac{\sigma^2}{K_P}$, we obtain that

$$\mathbb{E} f(W^{c+1}) - \mathbb{E} f(W^c) \leq -\eta \mathbb{E} \langle g_c, \nabla f(W^c) \rangle + \frac{L_f \eta_c^2 \sigma^2}{2K_P} \left[ \mathbb{E} \| G^c \|^2 + \frac{\sigma^2}{K_P} \right]$$

$$= -\eta \mathbb{E} \| g_c \|^2 + \frac{L_f \eta_c^2 \sigma^2}{2K_P} \left[ \mathbb{E} \| G^c \|^2 + \frac{\sigma^2}{K_P} \right]$$

$$\leq (\frac{L_f \eta_c^2 - \eta_c}{2K_P}) \mathbb{E} \| g_c \|^2 + \eta_c \mathbb{E} \| g_c \|^2 \sum_{p=1}^{P} L_p \| W^c_p - W^c \| + \frac{L_f \eta_c^2 \sigma^2}{2K_P}.$$

(5.6)

Next, we bound the term $\| W^c_p - W^c \|$. Note that $W^c = \sum_{t=0}^{c-1} \eta_t G^t$, $W^c_p = \sum_{t=0}^{c-1} \eta_t G^t_p$, both of which are aggregations of the stochastic gradient updates. Since $G^t_p$ contains the delayed stochastic gradients with maximum staleness $s$, the discrepancy between $W^c_p$ and $W^c$ is at most the updates $\{ G^{c-s+1}, \ldots, G^{c-1} \}$, and we obtain that

$$\| W^c - W^c_p \| \leq \sum_{t=c-s+1}^{c-1} \eta_t \| G^t \|. \tag{5.7}$$

We note that this is only a coarse estimate for simplicity of the derivation. In general, less discrepancy may occur and does not affect the order of the final convergence rate. With Eq.(5.7), Eq.(5.6) can be further bounded as

$$\mathbb{E} f(W^{c+1}) - \mathbb{E} f(W^c)$$

$$\leq \left( \frac{L_f \eta_c^2 - \eta_c}{2K_P} \right) \mathbb{E} \| g_c \|^2 + L \sum_{t=c-s+1}^{c-1} \mathbb{E} \eta_t \| g_c \| \| G^t \| + \frac{L_f \eta_c^2 \sigma^2}{2K_P}$$

$$\leq \left( \frac{L_f \eta_c^2 - \eta_c}{2K_P} \right) \mathbb{E} \| g_c \|^2 + L \sum_{t=c-s+1}^{c-1} \mathbb{E} [\eta_t^2 \| g_c \|^2 + \eta_t^2 \| G^t \|^2] + \frac{L_f \eta_c^2 \sigma^2}{2K_P}$$

$$\leq \left( \frac{L_f + 2Ls \eta_c^2 - \eta_c}{2K_P} \right) \mathbb{E} \| g_c \|^2 + L \sum_{t=c-s+1}^{c-1} \eta_t^2 [\mathbb{E} \| G^t \|^2 + \frac{\sigma^2}{K_P}] + \frac{L_f \eta_c^2 \sigma^2}{2K_P}. \tag{5.6}$$

174
Telescoping over $c$ from 0 to $C - 1$ and note that 

$$\sum_{c=0}^{C-1} \sum_{t=0}^{c-1} r_t \leq s \sum_{c=0}^{C-1} r_c$$

for the positive sequence $\{r_t\}$, we obtain that

$$\inf f(W) - \mathbb{E} f(W^0) \leq \sum_{c=0}^{C-1} \left( \frac{1}{2} L_f \eta_c^2 - \frac{1}{2} \eta_c \right) \mathbb{E} \|g_c\|^2 + \mathbb{E} \sum_{c=0}^{C-1} \eta_c^2 \frac{\sigma^2}{KP} + \mathbb{E} \sum_{c=0}^{C-1} L_f \eta_c^2 \frac{\sigma^2}{2KP}$$

$$\leq \sum_{c=0}^{C-1} \left( \frac{1}{2} L_f \eta_c^2 - \frac{1}{2} \eta_c \right) \mathbb{E} \|g_c\|^2 + \sum_{c=0}^{C-1} \eta_c \frac{(L_c + L_f)\sigma^2}{KP}.$$  

(5.8)

It then follows that

$$\min_{c=1,\ldots,C} \mathbb{E} \|g_c\|^2 \leq \frac{f(W^0) - \inf f(W) + \frac{(L_c + L_f)\sigma^2}{KP} \sum_{c=0}^{C-1} \eta_c^2}{\sum_{c=0}^{C-1} \eta_c} \approx \frac{f(W^0) - \inf f(W) + \frac{(L_c + L_f)\sigma^2}{KP} \sum_{c=0}^{C-1} \eta_c^2}{\sum_{c=0}^{C-1} \eta_c},$$

where we ignore the higher order term in the last equation for simplicity, as we will use a diminishing stepsize $\eta_c = O(1/\sqrt{c})$ and this does not affect the order of the final estimate. Now we can apply [35, Theorem 4.2] to the above equation to conclude that

$$\min_{c=1,\ldots,C} \mathbb{E} \|g_c\|^2 \leq \sqrt{\frac{f(W^0) - \inf f(W)(L_c + L_f)\sigma^2}{KP C}},$$

with the choice of stepsize

$$\eta_c = \frac{4[f(W^0) - \inf f(W)]KP \cdot 1}{(L_c + L_f)\sigma^2 \sqrt{c}}.$$  

(5.9)

Hence we must have \( \liminf_{c \to \infty} \mathbb{E} \|g_c\|^2 = 0 \). By our assumption, \( \{W_p\}_{p,c} \) and \( \{W_c\}_{c} \) are bounded and hence have limit points. Also note that the gradient of \( f_j \) is continuous. Thus, \( \|G^t\| \) is bounded. Since \( \eta_c = \frac{1}{\sqrt{c}} \to 0 \), Eq.(5.7) implies that \( \|W^c - W^c_p\| \to 0 \). This further implies that \( \{W^c\} \) and \( \{W^c_p\} \) share the same set of limit points. Lastly, by the fact that \( \|W^c - W^c_p\| \to 0 \) and the Lipschitz continuity of \( \nabla f_p \), we conclude that \( \liminf \mathbb{E} \|\nabla f(W^c)\|^2 \to 0 \). Thus, there exists a common subsequence of \( \{W^c\} \) and \( \{W^c_p\} \) that converges to a stationary point almost surely.

**Proof of Corollary**

The proof strategy is similar to that of Theorem [6]. Specifically, the main difference from the proof of Theorem [6] is the bound of \( \|W^c - W^c_p\| \). In the multicast setting, we have \( W^c = \sum_{t=0}^{c-1} \eta_t G^t \), \( W^c_p = \sum_{t=0}^{c-1} \eta_t G^t_p(\delta) \). Despite the discrepancy of gradients cause by the delay \( s \), the multicast scheme makes \( G^t_p(\delta) \) only aggregates \( Q \) stochastic gradients from other machines at
each iteration, resulting in a $P - Q$ stochastic gradient discrepancy compared to $G'$. Thus, we obtain

$$\|W^c - W^c_p\| \leq \sum_{t=c-s+1}^{c-1} \eta_t \|G^t\| + \sum_{t=1}^{c-s} \eta_t (P - Q) R,$$

(5.10)

where $R$ is an absolute constant that bounds the norm of the gradients. The rest of the proof follows that of Theorem 6. Specifically, with Eq. (5.10), Eq. (5.6) further becomes

$$\mathbb{E} f(W^{c+1}) - \mathbb{E} f(W^c)$$

$$\leq \left( \frac{L_f}{2} \eta_c^2 - \eta_c \right) \mathbb{E} \|g_c\|^2 + L \sum_{t=c-s+1}^{c-1} \mathbb{E} \eta_t \|g_c\| \|G^t\| + L \sum_{t=1}^{c-s} \mathbb{E} \eta_t \|g_c\| (P - Q) R + \frac{L_f \eta_c^2 \sigma^2}{2KP}$$

$$\leq \left( \frac{L_f}{2} \eta_c^2 - \eta_c \right) \mathbb{E} \|g_c\|^2 + L \sum_{t=c-s+1}^{c-1} \eta_t^2 \|G^t\|^2 + L \sum_{t=1}^{c-s} \mathbb{E} \eta_t^2 \|g_c\|^2 (P - Q) R + \eta_t^2 (P - Q) R$$

$$+ \frac{L_f \eta_c^2 \sigma^2}{2KP}$$

$$\leq \left( \frac{L_f + 2Lc(P - Q) R}{2} \eta_c^2 - \eta_c \right) \mathbb{E} \|g_c\|^2 + L \sum_{t=c-s+1}^{c-1} \eta_t^2 \|G^t\|^2 + \frac{\sigma^2}{KP} + L(P - Q) R \sum_{t=1}^{c-s} \eta_t^2 + \frac{L_f \eta_c^2 \sigma^2}{2KP}.$$

Telescoping over $c$ from 0 to $C - 1$ and noting that $\sum_{c=0}^{C-1} \sum_{t=c-s+1}^{c-1} r_t \leq s \sum_{c=0}^{C-1} r_c, \sum_{c=0}^{C-1} \sum_{t=1}^{c-s} r_t \leq (c - s) \sum_{c=0}^{C-1} r_c$ for the positive sequence $\{r_t\}$, we obtain

$$\inf_{c} f(W) - \mathbb{E} f(W^0)$$

$$\leq \sum_{c=0}^{C-1} \left( \frac{L_f}{2} + 2Lc(P - Q) R + s \right) \eta_c^2 - \eta_c \mathbb{E} \|g_c\|^2 + L(P - Q) R \sum_{c=1}^{C-s} \eta_c^2 + \sum_{c=0}^{C-1} \eta_c^2 \frac{(Ls + Lf) \sigma^2}{KP}.$$

It then follows that

$$\min_{c=1,\ldots,C} \mathbb{E} \|g_c\|^2 \leq \frac{f(W^0) - \inf f(W) + L(P - Q) R \sum_{c=1}^{C-s} \eta_c^2 + \frac{(Ls + Lf) \sigma^2}{KP} \sum_{c=0}^{C-1} \eta_c^2}{\sum_{c=0}^{C-1} (\eta_c - \frac{L_f + 2Lc(P - Q) R + s}{2} \eta_c^2)}$$

$$\approx \frac{f(W^0) - \inf f(W) + [L(P - Q) R + \frac{(Ls + Lf) \sigma^2}{KP}] \beta^2}{\beta - \frac{\beta^2 L(P - Q) R}{2}},$$

where we have chosen the constant stepsize $\eta_c \equiv \beta/C$. Setting $\beta = \frac{1}{L(P - Q) R}$, we obtain that

$$\min_{c=1,\ldots,C} \mathbb{E} \|g_c\|^2 \leq O \left( L(P - Q) + \frac{(Ls + Lf) \sigma^2}{KPL(P - Q)C} \right).$$

\[ \square \]
Chapter 6
Applications in Healthcare

In this chapter, we design machine learning models for several healthcare applications: (1) retrieving similar patients, (2) discovering topics from medical texts, (3) tagging pathology and radiology images, and (4) assigning ICD codes, which involve various data modalities, including texts, images, time series, and tabular data. We apply diversity-promoting learning (DPL) techniques (developed in Chapters 2-4) for the first application to better capture infrequent patterns, reduce overfitting, shrink model size without sacrificing modeling power and apply large-scale distributed learning systems (developed in Chapter 5) for the second application to improve training efficiency.

6.1 Diversity-promoting Learning for Similar-patient Retrieval

Patient similarity measurement (PSM) \[315, 343\], which decides whether two patients are similar or dissimilar based on their electronic health records (EHRs), is a critical task for patient cohort identification and finds wide applications in clinical decision-making. For instance, with an effective similarity measure in hand, one can perform case-based retrieval of similar patients for a target patient, who can be subsequently diagnosed and treated by synthesizing the diagnosis outcomes and treatment courses of the retrieved patients. Other applications powered by patient similarity include classification of epidemiological data on hepatic steatosis \[155\], patient risk prediction \[261, 281\], personalized treatment for hypercholesterolemia \[413\], personalized mortality prediction \[219\], near-term prognosis \[106, 344\], to name a few. Several approaches have applied distance metric learning (DML) \[383\] to learn patient-similarity measures, where physicians provide feedback on whether two patients are similar or dissimilar, and the algorithm learns a distance metric such that similar patients would have small distance while dissimilar patients are separated apart.

When using DML for patient similarity measurement, we encountered the following problems. First, it performs less well on infrequent diseases. As an example, the top and bottom plots in Figure 6.1(a) show the frequencies of eight diseases and the retrieval performance (precision@K) on individual diseases. As can be seen, the performance on infrequent diseases is much worse than that on frequent diseases. Second, in clinical decision-making, timeliness is very important. For PSM-based applications, how to retrieve similar patients efficiently is critical.
Figure 6.1: (a) Retrieval performance on frequent and infrequent diseases. (b) Retrieval AUCs on the training and test sets.

The time complexity of retrieval increases linearly with the dimension of the latent representations learned in DML. An immediate solution for fast retrieval is to reduce the latent dimension. However, this would sacrifice the expressivity of DML and compromise retrieval performance. Third, DML generalizes less well on unseen patients. As can be seen from Figure 6.1(b), the performance on test set is much worse than that on training set.

To address these issues, we resort to diversity-promoting regularization that has been demonstrated to have the ability of better capturing infrequent patterns, reducing model size without sacrificing modeling power, and improving generalization performance (Chapter 2–4). Among the various diversity-promoting regularizers, we choose to use the convex Bregman matrix divergence regularizers (Section 2.2.2) since they guarantee to achieve the global optimal solution in optimization. Specifically, we solve the following problem:

$$\min_M \frac{1}{|S|} \sum_{(x,y) \in S} (x - y)^T M (x - y) + \frac{1}{|D|} \sum_{(x,y) \in D} \max(0, 1 - (x - y)^T M (x - y)) + \lambda \tilde{\Omega}_\phi(M)$$

s.t. \( M \succeq 0 \).  

(6.1)

6.1.1 Experimental Settings

The evaluation has been reported in Section 2.2.4. For readers’ convenience, we re-summarize them here. Two clinical datasets are used: MIMIC-III [184] and EICU [131]. MIMIC-III contains 58K hospital admissions of patients who stayed within the intensive care units at the Beth Israel Deaconess Medical Center between 2001 and 2012. We extracted 7207-dimensional features: (1) 2 dimensions from demographics, including age and gender; (2) 5300 dimensions from clinical notes, including 5000-dimensional bag-of-words (weighted using tf-idf) and 300-dimensional word2vec [256]; (3) 1905-dimensions from lab tests where the zero-order, first-order, and second-order temporal features are extracted for each of the 635 lab items. In the
extraction of bag-of-words features from clinical notes, we removed stop words, then counted the document frequency (DF) of the remaining words. Then we selected the largest 5000 words to form the dictionary. Based on this dictionary, we extracted tfidf features. In the extraction of word2vec features, we trained a 300-dimensional embedding vector for each word using an open source word2vec tool. To represent a clinical note, we averaged the embeddings of all words in this note. In lab tests, there are 635 test items in total. An item is tested at different time points for each admission. For an item, we extracted three types of temporal features: (1) zero-order: averaging the values of this item measured at different time points; (2) first-order: taking the difference of values at every two consecutive time points \( t \) and \( t - 1 \), and averaging these differences; (3) second-order: for the sequence of first-order differences generated in (2), taking the difference (called second-order difference) of values at every two consecutive time points \( t \) and \( t - 1 \), and averaging these second-order differences. If an item is missing in an admission, we set the zero-order, first-order, and second-order feature values to 0.

The EICU dataset contains hospital admissions of patients who were treated as part of the Philips eICU program across intensive care units in the United States between 2014 and 2015. There are 474 lab test items and 48 vital sign items. Each admission has a past medical history, which is a collection of diseases. There are 2644 unique past diseases. We extracted the following features: (1) age and gender; (2) zero-, first- and second- order temporal features of lab test and vital signs; (3) past medical history: we used a binary vector to encode them; if an element in the vector is 1, then the patient had the corresponding disease in the past. For both datasets, the features were normalized using min-max normalization along each dimension. We used PCA to reduce the feature dimension to 1000 to reduce the computational complexity (performing eigen-decomposition of the Mahalanobis matrix bears cubic complexity in term of feature dimension). In either dataset, each admission has a primary diagnosis (a disease). MIMIC-III and EICU have 2833 and 2175 unique diseases respectively.

Two admissions are considered as similar if they have the same primary diagnosis and dissimilar if otherwise. In similar-patient retrieval, we used each test example to query the rest of the test examples based on the learned distance metric. If the distance between \( x \) and \( y \) is smaller than a threshold \( s \) and they have the same primary diagnosis, then this is a true positive. By choosing different values of \( s \), we obtained a receiver operating characteristic (ROC) curve. We evaluated the retrieval performance using the area under (ROC) curve (AUC) which is the higher, the better. We compared the proposed convex BMD regularizers with two sets of baseline regularizers. The first set aims at promoting orthogonality, which are based on determinant of covariance (DC) [242], cosine similarity (CS) [402], determinantal point process (DPP) [204, 429], InCoherence (IC) [29], variational Gram function (VGF) [177, 420], decorrelation (DeC) [82], mutual angles (MA) [369], squared Frobenius norm (SFN) [74, 114, 122, 347], von Neumann divergence (VND), log-determinant divergence (LDD), and orthogonal constraint (OC) \( \text{AA}^\top = I \) [233, 342]. All these regularizers were applied to the non-convex projection matrix-based DML (PDML, Eq. (2.14) in Section 2.2.1). The other set of regularizers are not designed particularly for promoting orthogonality but are commonly used, including \( \ell_2 \) norm, \( \ell_1 \) norm [280], \( \ell_{2,1} \) norm [399], trace norm (Tr) [234], information theoretic (IT) regularizer \(-\logdet(M) + \text{tr}(M)\) [92], and dropout (Drop) [313]. All these regularizers were applied to the

\[1\text{https://code.google.com/archive/p/word2vec/}\]
<table>
<thead>
<tr>
<th>Method</th>
<th>AUC-All</th>
<th>AUC-F</th>
<th>AUC-IF</th>
<th>AUC-All</th>
<th>AUC-F</th>
<th>AUC-IF</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCDML</td>
<td>0.634</td>
<td>0.654</td>
<td>0.608</td>
<td>0.671</td>
<td>0.690</td>
<td>0.637</td>
</tr>
<tr>
<td>CDML</td>
<td>0.641</td>
<td>0.659</td>
<td>0.617</td>
<td>0.677</td>
<td>0.693</td>
<td>0.652</td>
</tr>
<tr>
<td>EUC</td>
<td>0.559</td>
<td>0.558</td>
<td>0.562</td>
<td>0.583</td>
<td>0.584</td>
<td>0.581</td>
</tr>
<tr>
<td>LMNN [357]</td>
<td>0.628</td>
<td>0.643</td>
<td>0.609</td>
<td>0.662</td>
<td>0.678</td>
<td>0.633</td>
</tr>
<tr>
<td>LDML [146]</td>
<td>0.619</td>
<td>0.638</td>
<td>0.594</td>
<td>0.667</td>
<td>0.678</td>
<td>0.647</td>
</tr>
<tr>
<td>MLEC [200]</td>
<td>0.621</td>
<td>0.633</td>
<td>0.605</td>
<td>0.679</td>
<td>0.692</td>
<td>0.656</td>
</tr>
<tr>
<td>GMML [405]</td>
<td>0.607</td>
<td>0.621</td>
<td>0.588</td>
<td>0.668</td>
<td>0.679</td>
<td>0.648</td>
</tr>
<tr>
<td>ILHD [63]</td>
<td>0.577</td>
<td>0.590</td>
<td>0.560</td>
<td>0.637</td>
<td>0.652</td>
<td>0.610</td>
</tr>
<tr>
<td>ℓ₂-CDML</td>
<td>0.648</td>
<td>0.664</td>
<td>0.627</td>
<td>0.695</td>
<td>0.706</td>
<td>0.676</td>
</tr>
<tr>
<td>ℓ₁-CDML [280]</td>
<td>0.643</td>
<td>0.666</td>
<td>0.615</td>
<td>0.701</td>
<td>0.715</td>
<td>0.677</td>
</tr>
<tr>
<td>ℓ₂,₁-CDML [159]</td>
<td>0.646</td>
<td>0.658</td>
<td>0.630</td>
<td>0.703</td>
<td>0.727</td>
<td>0.661</td>
</tr>
<tr>
<td>Tr-CDML [334]</td>
<td>0.659</td>
<td>0.672</td>
<td>0.642</td>
<td>0.696</td>
<td>0.709</td>
<td>0.673</td>
</tr>
<tr>
<td>IT-CDML [92]</td>
<td>0.653</td>
<td>0.675</td>
<td>0.626</td>
<td>0.692</td>
<td>0.705</td>
<td>0.668</td>
</tr>
<tr>
<td>Dropout-CDML [282]</td>
<td>0.647</td>
<td>0.660</td>
<td>0.630</td>
<td>0.701</td>
<td>0.718</td>
<td>0.670</td>
</tr>
<tr>
<td>OS-CDML [116]</td>
<td>0.649</td>
<td>0.665</td>
<td>0.626</td>
<td>0.697</td>
<td>0.711</td>
<td>0.679</td>
</tr>
<tr>
<td>DCM-NCDML [242]</td>
<td>0.652</td>
<td>0.662</td>
<td>0.639</td>
<td>0.706</td>
<td>0.717</td>
<td>0.686</td>
</tr>
<tr>
<td>CS-NCDML [402]</td>
<td>0.661</td>
<td>0.676</td>
<td>0.641</td>
<td>0.712</td>
<td>0.736</td>
<td>0.670</td>
</tr>
<tr>
<td>DPP-NCDML [429]</td>
<td>0.659</td>
<td>0.679</td>
<td>0.632</td>
<td>0.714</td>
<td>0.725</td>
<td>0.695</td>
</tr>
<tr>
<td>IC-NCDML [29]</td>
<td>0.660</td>
<td>0.674</td>
<td>0.642</td>
<td>0.711</td>
<td>0.728</td>
<td>0.685</td>
</tr>
<tr>
<td>DC-NCDML [53]</td>
<td>0.648</td>
<td>0.666</td>
<td>0.625</td>
<td>0.698</td>
<td>0.711</td>
<td>0.675</td>
</tr>
<tr>
<td>VOF-NCDML [427]</td>
<td>0.657</td>
<td>0.673</td>
<td>0.634</td>
<td>0.718</td>
<td>0.730</td>
<td>0.697</td>
</tr>
<tr>
<td>MA-NCDML [367]</td>
<td>0.659</td>
<td>0.670</td>
<td>0.644</td>
<td>0.721</td>
<td>0.733</td>
<td>0.703</td>
</tr>
<tr>
<td>OC-NCDML [233]</td>
<td>0.651</td>
<td>0.663</td>
<td>0.636</td>
<td>0.705</td>
<td>0.716</td>
<td>0.685</td>
</tr>
<tr>
<td>OS-NCDML [116]</td>
<td>0.639</td>
<td>0.658</td>
<td>0.614</td>
<td>0.675</td>
<td>0.691</td>
<td>0.641</td>
</tr>
<tr>
<td>SFN-NCDML [74]</td>
<td>0.662</td>
<td>0.677</td>
<td>0.642</td>
<td>0.724</td>
<td>0.736</td>
<td>0.701</td>
</tr>
<tr>
<td>VND-NCDML</td>
<td>0.667</td>
<td>0.676</td>
<td>0.655</td>
<td>0.733</td>
<td>0.748</td>
<td>0.706</td>
</tr>
<tr>
<td>LDD-NCDML</td>
<td>0.664</td>
<td>0.674</td>
<td>0.651</td>
<td>0.731</td>
<td>0.743</td>
<td>0.711</td>
</tr>
<tr>
<td>CSFN-CDML</td>
<td>0.668</td>
<td>0.679</td>
<td>0.653</td>
<td>0.728</td>
<td>0.741</td>
<td>0.705</td>
</tr>
<tr>
<td>CVND-CDML</td>
<td>0.672</td>
<td>0.678</td>
<td>0.664</td>
<td>0.735</td>
<td>0.744</td>
<td>0.718</td>
</tr>
<tr>
<td>CLDD-CDML</td>
<td>0.669</td>
<td>0.678</td>
<td>0.658</td>
<td>0.739</td>
<td>0.750</td>
<td>0.719</td>
</tr>
</tbody>
</table>

Table 6.1: On MIMIC and EICU, we show the mean AUC (averaged on 5 random train/test splits) on all diseases (AUC-All), frequent diseases (AUC-F), and infrequent diseases (AUC-IF). On the second panel (EUC, etc.) are well established or state of the art baselines. On the third panel (ℓ₂-CDML, etc.) are CDML methods regularized by non-diversity regularizers. On the fourth panel (DCM-NCDML, etc.) are NCDML methods regularized by previously proposed diversity-promoting regularizers. On the fifth panel (VND-NCDML, etc.) are NCDML methods regularized by our proposed nonconvex BMD regularizers. On the sixth panel (CSFN-CDML, etc.) are CDML methods regularized by our proposed convex BMD regularizers.

Mahalanobis distance-based DML (MDML, Eq. (6.1) in Section 2.2.2). In addition, we compared with the vanilla Euclidean distance (EUC) and other distance learning methods including large margin nearest neighbor (LMNN) metric learning, information theoretic metric learning (ITML) [92], logistic discriminant metric learning (LDML) [146], metric learning from equivalence constraints (MLEC) [200], geometric mean metric learning (GMML) [405], and independent Laplacian hashing with diversity (ILHD) [63].

For computational efficiency, in MDML-based methods, we did not use \((x - y)^\top M (x - y)\) to compute distance directly. Given the learned matrix \(M\) (which is of rank \(k\)), we can decompose it into \(L^\top L\) where \(L \in \mathbb{R}^{k \times d}\). Let \(U \Lambda U^\top\) be the eigen-decomposition of \(M\). Let \(\lambda_1, \ldots, \lambda_k\) denote the \(k\) nonzero eigenvalues and \(u_1, \ldots, u_k\) denote the corresponding eigenvectors. Then \(L\) is the transpose of \([\sqrt{\sigma_1} u_1, \ldots, \sqrt{\sigma_k} u_k]\). Given \(L\), we can use it to transform each input \(d\)-dimensional feature vector \(x\) into a new \(k\)-dimensional vector \(Lx\), then perform retrieval on the...
new vectors based on Euclidean distance. Note that only when computing Euclidean distance between $L_x$ and $L_y$, we have that $\|L_x - L_y\|_2^2$ is equivalent to $(x - y)^\top M (x - y)$. For other distances or similarity measures between $L_x$ and $L_y$, such as $L_1$ distance and cosine similarity, this does not hold. Performing retrieval based on $\|L_x - L_y\|_2^2$ is more efficient than that based on $(x - y)^\top M (x - y)$ when $k$ is smaller than $d$. Given $m$ test examples, the computation complexity of $\|L_x - L_y\|_2^2$-based retrieval is $O(mkd + m^2k)$, while that of $(x - y)^\top M (x - y)$-based retrieval is $O(m^2d^2)$.

### 6.1.2 Results

First, we verify whether CSFN, CVND, and CLDD are able to improve the performance on infrequent diseases. On MIMIC and EICU, we consider a disease as being “frequent” if it contains more than 1000 examples, and “infrequent” if otherwise. We measure AUCs on all diseases (AUC-All), infrequent diseases (AUC-IF), and frequent diseases (AUC-F). As can be seen, in most DML methods, the AUCs on infrequent diseases are worse than those on frequent diseases, showing that DML is sensitive to the imbalance of diseases’ frequencies and tends to be biased towards frequent diseases and is less capable of capturing infrequent diseases. This is in accordance with the previous findings [369]. Adding our proposed CSFN, CVND, CLDD regularizers to CDML, the AUCs on infrequent diseases are greatly improved. By encouraging the components to be close to being orthogonal, our methods can reduce the redundancy among vectors. Mutually complementary vectors can achieve a broader coverage of latent features, including

<table>
<thead>
<tr>
<th></th>
<th>MIMIC</th>
<th></th>
<th>EICU</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PDML</td>
<td>0.654</td>
<td>300</td>
<td>2.1</td>
<td>0.671</td>
</tr>
<tr>
<td>MDML</td>
<td>0.641</td>
<td>247</td>
<td>2.6</td>
<td>0.677</td>
</tr>
<tr>
<td>EUC</td>
<td>0.639</td>
<td>1000</td>
<td>0.6</td>
<td>0.583</td>
</tr>
<tr>
<td>LMNN</td>
<td>0.628</td>
<td>200</td>
<td>3.1</td>
<td>0.662</td>
</tr>
<tr>
<td>LDML</td>
<td>0.619</td>
<td>300</td>
<td>2.1</td>
<td>0.667</td>
</tr>
<tr>
<td>MLEC</td>
<td>0.621</td>
<td>487</td>
<td>1.3</td>
<td>0.679</td>
</tr>
<tr>
<td>GMML</td>
<td>0.607</td>
<td>1000</td>
<td>0.6</td>
<td>0.668</td>
</tr>
<tr>
<td>IHD</td>
<td>0.577</td>
<td>100</td>
<td>5.8</td>
<td>0.637</td>
</tr>
<tr>
<td>MDML-$\ell_2$</td>
<td>0.648</td>
<td>269</td>
<td>2.4</td>
<td>0.695</td>
</tr>
<tr>
<td>MDML-$\ell_1$</td>
<td>0.643</td>
<td>341</td>
<td>1.9</td>
<td>0.701</td>
</tr>
<tr>
<td>MDML-$\ell_2,1$</td>
<td>0.646</td>
<td>196</td>
<td>3.3</td>
<td>0.703</td>
</tr>
<tr>
<td>MDML-Tr</td>
<td>0.659</td>
<td>148</td>
<td>4.5</td>
<td>0.696</td>
</tr>
<tr>
<td>MDML-IT</td>
<td>0.653</td>
<td>1000</td>
<td>0.7</td>
<td>0.692</td>
</tr>
<tr>
<td>MDML-Drop</td>
<td>0.647</td>
<td>183</td>
<td>3.5</td>
<td>0.701</td>
</tr>
<tr>
<td>PDML-DC</td>
<td>0.652</td>
<td>100</td>
<td>6.5</td>
<td>0.706</td>
</tr>
<tr>
<td>PDML-CS</td>
<td>0.661</td>
<td>200</td>
<td>3.3</td>
<td>0.712</td>
</tr>
<tr>
<td>PDML-DPP</td>
<td>0.659</td>
<td>100</td>
<td>6.6</td>
<td>0.714</td>
</tr>
<tr>
<td>PDML-IC</td>
<td>0.660</td>
<td>200</td>
<td>3.3</td>
<td>0.711</td>
</tr>
<tr>
<td>PDML-DeC</td>
<td>0.648</td>
<td>200</td>
<td>3.2</td>
<td>0.698</td>
</tr>
<tr>
<td>PDML-VGF</td>
<td>0.657</td>
<td>200</td>
<td>3.3</td>
<td>0.718</td>
</tr>
<tr>
<td>PDML-MA</td>
<td>0.659</td>
<td>200</td>
<td>3.3</td>
<td>0.721</td>
</tr>
<tr>
<td>PDML-OC</td>
<td>0.651</td>
<td>100</td>
<td>6.5</td>
<td>0.705</td>
</tr>
<tr>
<td>PDML-SFN</td>
<td>0.662</td>
<td>100</td>
<td>6.6</td>
<td>0.724</td>
</tr>
<tr>
<td>PDML-VND</td>
<td>0.667</td>
<td>100</td>
<td>6.7</td>
<td>0.733</td>
</tr>
<tr>
<td>PDML-LDD</td>
<td>0.664</td>
<td>100</td>
<td>6.6</td>
<td>0.731</td>
</tr>
<tr>
<td>MDML-CSFN</td>
<td>0.668</td>
<td>143</td>
<td>4.7</td>
<td>0.728</td>
</tr>
<tr>
<td>MDML-CVND</td>
<td>0.672</td>
<td>53</td>
<td>12.7</td>
<td>0.735</td>
</tr>
<tr>
<td>MDML-CLDD</td>
<td>0.669</td>
<td>76</td>
<td>8.8</td>
<td>0.739</td>
</tr>
</tbody>
</table>

Table 6.2: Area under ROC curve (AUC), number of projection vectors (NPV), and compactness score (CS, $\times 10^{-3}$).
those associated with infrequent diseases. As a result, these vectors improve the performance on infrequent diseases.

Thanks to their convexity nature, our methods can achieve the global optimal solution and outperform the non-convex ones that can only achieve a local optimal and hence a sub-optimal solution. (C)VND and (C)LDD outperform (C)SFN, possibly because they measure near-orthogonality in a global way while (C)SFN conducts that in a pairwise fashion. Comparing OS-(NCDML,CDML) with the unregularized NCDML/CDML, we can see that over-sampling indeed improves performance on infrequent diseases. However, this improvement is less significant than that achieved by our methods. In general, the diversity-promoting (DP) regularizers outperform the non-DP regularizers, suggesting the effectiveness of promoting diversity. The orthogonal constraint (OC) \cite{233,342} imposes strict orthogonality, which may be too restrictive that hurts performance. ILHD \cite{63} learns binary hash codes, which makes retrieval more efficient. However, it achieves lower AUCs due to the quantization errors. (CSFN,CVND,CLDD)-CDML outperform popular DML approaches including LMNN, LDML, MLEC, and GMML, demonstrating their competitive standing in the DML literature.

Table 6.2 shows the area under ROC curve (AUC) and the numbers of the projection vectors (NPVs) that achieve the AUCs. For MDML-based methods, the NPV equals the rank of the Mahalanobis matrix since $M = A^\top A$. We define a compactness score (CS) which is the ratio between AUC and NPV. A higher CS indicates achieving higher AUC by using fewer projection vectors. From Table 6.2, we can see that on both datasets, MDML-(CSFN,CVND,CLDD) achieve larger CSs than the baseline methods, demonstrating their better capability in learning
compact distance metrics. CSFN, CVND, and CLDD perform better than non-convex regularizers, and CVND, CLDD perform better than CSFN. The reduction of NPV improves the efficiency of retrieval since the computational complexity grows linearly with this number.

Table 6.3 shows the difference between training AUC and testing AUC. Our methods have the smallest gap between training and testing AUCs. This indicates that our methods are better capable of reducing overfitting and improving generalization performance.

6.2 Large-scale Distributed Learning for Medical Topic Discovery

Discovering medical topics from clinical documents has many applications, such as consumer medical search [84], mining FDA drug labels [44], investigating drug repositioning opportunities [45], to name a few. In practice, the clinical text corpus can contain millions of documents and the medical dictionary is comprised of hundreds of thousands of terminologies. These large-scale documents contain rich medical topics, whose number can be tens of thousands. How to efficiently discover so many topics from such a large dataset is computationally challenging. We resort to the Orpheus system (Section 5.2) we have developed.

We begin with introducing how to learn the topics. For each document, we represent it with a bag-of-words vector \( d \in \mathbb{R}^V \) where \( V \) is the dictionary size. Similar to [207], we assume each document is approximately a linear combination of \( K \) topics: \( d \approx W\theta \). \( W \in \mathbb{R}^{V \times K} \) is the topic matrix, where each topic is represented with a vector \( w \in \mathbb{R}^V \). \( w_v \geq 0 \) denotes the association strength between the \( v \)-th word and this topic, and \( \sum_{v=1}^{V} w_v = 1 \). \( \theta \in \mathbb{R}^K \) are the linear combination weights satisfying \( \theta_k \geq 0 \) and \( \sum_{k=1}^{K} \theta_k = 1 \). \( \theta_k \) denotes how relevant topic \( k \) is to this document. Given the unlabeled documents \( \{d_i\}_{i=1}^{N} \), we learn the topics by solving the following problem [207]:

\[
\min_{\{\theta_i\}_{i=1}^{N}, W} \frac{1}{2} \sum_{i=1}^{N} \|d_i - W\theta_i\|_2^2
\]

\[
s.t. \quad \forall k = 1, \ldots, K, v = 1, \ldots, V, W_{kv} \geq 0, \sum_{v=1}^{V} W_{kv} = 1; \quad \forall i = 1, \ldots, N, k = 1, \ldots, K, \theta_{ik} \geq 0, \sum_{k=1}^{K} \theta_{ik} = 1, \quad (6.2)
\]

where \( \{\theta_i\}_{i=1}^{N} \) denotes all the linear coefficients.

This problem can be solved by alternating between \( \{\theta_i\}_{i=1}^{N} \) and \( W \). The \( N \) sub-problems defined on \( \{\theta_i\}_{i=1}^{N} \) can be solved independently by each machine based on the data shard it has. No inter-machine communication is needed. For the sub-problem defined on \( W \), we use the Orpheus system to solve it. Each machine maintains a local copy of \( W \) and different copies are synchronized to ensure convergence. The projected stochastic gradient descent algorithm is applied, which iteratively performs two steps: (1) stochastic gradient descent; (2) projection onto the probability simplex. In the first step, the stochastic gradient matrix computed over one document can be written as the outer product of two vectors: \( (W\theta_i - d_i)\theta_i^T \). In other words, this problem has a sufficient factor (SF) property and fits into the sufficient factor broadcasting framework. In each iteration, each machine computes a set of SFs, sends them to other machines and uses the SFs to update its own parameter copy. On the receiver side, the received SFs are
Table 6.4: Convergence time (hours) in Orpheus and baseline systems for topic model.

<table>
<thead>
<tr>
<th>System</th>
<th>Runtime (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FlexiFaCT [207]</td>
<td>33.9</td>
</tr>
<tr>
<td>Bosen [355]</td>
<td>23.5</td>
</tr>
<tr>
<td>Orpheus</td>
<td>5.4</td>
</tr>
</tbody>
</table>

Figure 6.2: (a) A medical image has multiple tags which are correlated clinically or biologically. (b) The medical concepts are organized into a hierarchical tree. (c) The abnormal regions (marked by red contours) are small.

6.3 Hierarchical Multi-label Tagging of Medical Images

Medical images generated from radiography, computed tomography (CT) scans, magnetic resonance imaging (MRI), ultrasound, biopsy etc. are widely used in hospitals and clinics for diagnosis, treatment, and surgery. Once read, these images are dumped into the picture archiving and communication system (PACS). Lacking accurate and rich textual labels, these images are difficult to index and search. As a result, the utilization of these images is under-explored. To
address this issue, we study the automatic tagging of medical images, where the obtained textual tags improve the accessibility of images. For example, physicians can search images that are tagged with “chest x-ray” and “pneumonia” and perform a cohort study on them.

Automatically tagging clinical images with medical concepts is challenging. First, a medical image usually contains rich information. As a result, it can be simultaneously tagged with multiple concepts. Figure 6.2 shows an example where the image is tagged with “capillary”, “fibroblast”, and “mononuclear”. These concepts have strong clinical or biological correlations. For example, mononuclear cell and fibroblast have an interaction in scleroderma [133]. Capillaries and fibroblast have a dynamic interaction in the heart [51]. Such correlations can be explored to improve tagging accuracy. Here is an example. Since capillary is visually similar to Mallory bodies, it is difficult to distinguish these two based on image features. But Mallory bodies have little correlation with fibroblast and mononuclear cell. Leveraging the concept-correlations, we can correctly choose capillary rather than Mallory body as the label. Technically, how to capture these correlations is nontrivial.

Second, medical concepts are usually organized by physicians into a hierarchical ontology. On the top of the hierarchy are general concepts. From top to bottom, each concept is divided into more fine-grained sub-concepts. Figure 6.2b shows an example. In this concept tree, each node represents a disease, whose children represent subtypes of this disease. For instance, meningomyelocele and anencephaly are both subtypes of neural tube defects. This hierarchical structure can be leveraged to improve tagging accuracy. On one hand, if A and B are both children of C, then it is unlikely to use A and B to simultaneously tag an image. For instance, low power microscopic, middle power microscopic, and high power microscopic are children of microscopic. A pathology image can only be taken by one of these microscopic techniques. On the other hand, if the distance between A and B in the concept tree is smaller than that between A and C and we know A is the correct label, then B is more likely to be the correct label than C, since concepts with smaller distance are more relevant. For example, abscess is closer to phagocytosis than omphalocele. The former two are both under the sub-tree rooted with inflammation. As a result, if an image is tagged with abscess, it is more likely to be tagged with phagocytosis than omphalocele. How to explore the hierarchical structure among concepts for better tagging is technically demanding.

Third, in images showing abnormalities, the abnormal regions are usually very small, occupying a small proportion of the entire image. As shown in Figure 6.2c, the dark round areas (marked with red contours) show lymph nodes which are involved by the neoplasm. It is difficult to tag these abnormalities because of the smallness of the abnormal regions. How to detect these regions and properly tag them is challenging.

We aim at addressing these three challenges. To cope with the first challenge, we design an adversarial learning [134] approach where a discriminative network is used to distinguish the predicted labels from physician-provided labels while the predictive network tries to make them indistinguishable (i.e., cannot tell whether a label vector is predicted or human-provided). Such indiscernibility ensures the correlations among physician-provided labels are transferred to the predicted labels. To cope with the second challenge, we design a tree-of-sequences long short-term memory (LSTM) model which uses sequential LSTMs [163] to encode the individual concepts and uses a tree-structured LSTM [320, 368] to capture the hierarchical relationship among concepts. To address the third challenge, we design a contextual attention mechanism.
Figure 6.3: Our model consists of three major modules. Given the medical image, a contextual attention encoder is leveraged to learn a representation vector for this image. The tree-of-sequences LSTM encoder is used to learn a representation for each concept in the ontology. The image representation and concept representations are fed into the adversarial multi-label tagger to generate the tags for this image.

which calculates an importance score of each image-patch according to its contrast with the entire image. Patches with larger importance are paid more attention in predicting the tags. We demonstrate the effectiveness of our methods on two medical image datasets.

6.3.1 Methods

Figure 6.3 shows the overview of our model, where the inputs include a medical image and an ontology of medical concepts, and the outputs are multiple concepts that can be used to tag this image. In the concept ontology, only the leaf nodes are used for tagging. The model consists of three major modules: (1) a contextual attention encoder which takes the image as input, localizes abnormal regions which are given higher importance weights, and generates a weighted representation for this image; (2) a tree-of-sequences LSTM encoder which takes the concept ontology as input, uses sequential LSTMs to encode individual concepts and utilizes a bidirectional tree-structured LSTM to incorporate the hierarchical relationship among concepts into their representations; (3) an adversarial multi-label tagger which takes the representations of the image and the concepts as inputs, incorporates label-correlations using adversarial learning, and outputs multiple concepts to tag this image.

Adversarial Multi-label Tagging

It is often the case that a medical image can be tagged with multiple medical concepts that exhibit clinical or biological correlations. These correlations can be leveraged to distinguish concepts that are difficult to be differentiated using visual clues. In this section, we present an adversarial learning approach to capture such correlations for better multi-label tagging.

When assigning multiple concepts to an image, we need to consider two orthogonal factors: (1) concept-image relevance – how the content of the image is relevant to each concept; (2) concept-concept correlation – how strongly these concepts are correlated. To achieve (1), we build a predictive network which takes the image representation \( x \) (produced by the contextual attention encoder) and representation \( c_i \) of each concept \( i \) (produced by the tree-of-sequences
Figure 6.4: Adversarial learning for capturing correlations among concepts. The discriminative network (DN) aims at distinguishing the labels produced by the predictive network (PN) from the groundtruth labels provided by physicians. The PN tries to make the two sets of labels indistinguishable.

Figure 6.5: Tree-of-sequences LSTM. For each concept name, we build a sequential LSTM (SLSTM) to capture the semantics of words and the sequential structure among words. Over the concept hierarchy, we build a tree LSTM (TLSTM) to capture the hierarchical relationship among concepts. The encodings produced by the SLSTM are the inputs of the TLSTM.

LSTM encoder) as inputs and calculates a score \( s_i \) that measures how relevant this image is to this concept. To achieve (2), we leverage adversarial learning [134]. The basic idea is to use a discriminative network to tell which labels are produced by the predictive network and which are provided by the physicians. The predictive network tries to produce labels in a way that the discriminative network cannot tell whether they are predicted or human-provided. Such indistinguishability transfers the correlations manifested in human-provided labels into those predicted by the network.

Figure 6.4 illustrates the details. The training set is divided into two partitions: A and B. Partition A is used for learning the relevance between images and concepts. Partition B is used for learning the correlation among concepts. The groundtruth tagging of each image is represented by a binary vector \( t \in \mathbb{R}^k \) where \( k \) is the number of unique leaf concepts in the ontology and \( t_i = 1 \) denotes that the \( i \)-th concept is utilized by the physician to label this image. For each image
x in A, the predictive network (PN) predicts a score vector \( s = f_P(x; W_P) \) where \( s_i \in [0, 1] \) denotes the confidence that this image should be tagged by concept \( i \) and \( f_P(\cdot; W_P) \) denotes the PN parameterized by weight parameters \( W_P \). Then a sigmoid cross entropy (SCE) loss is defined on \( s \) and \( t \) to measure the discrepancy between the prediction and the groundtruth.

For each image in B, similarly the predictive network first generates the score vector \( s \). We use a discriminative network (DN) to differentiate the predicted score vectors \( \{s^{(n)}\}_{n=1}^{N_B} \) and the groundtruth label vectors \( \{t^{(n)}\}_{n=1}^{N_B} \), where \( N_B \) is the number of images in B. This is a binary classification task. The input to the DN (denoted by \( f_D(\cdot; W_D) \) where \( W_D \) are the weight parameters) is a \( k \)-dimensional vector and the output is the probability that the vector is predicted. As for the predictive network, it aims at making \( \{s^{(n)}\}_{n=1}^{N_B} \) indistinguishable from \( \{t^{(n)}\}_{n=1}^{N_B} \), such that the correlations reflected in \( \{t^{(n)}\}_{n=1}^{N_B} \) can be transferred to \( \{s^{(n)}\}_{n=1}^{N_B} \). Overall, we solve the following optimization problem:

\[
\min_{W_P} \mathcal{L}_{SCE}(W_P, A) + \max_{W_D} - \mathcal{L}_{BC}(W_P, W_D, B).
\] (6.3)

\( \mathcal{L}_{SCE}(W_P, A) \) is the sigmoid cross-entropy loss defined on the training set A:

\[
\mathcal{L}_{SCE}(W_P, A) = \sum_{n=1}^{N_A} \ell(f_P(x^{(n)}; W_P), t^{(n)}),
\] (6.4)

where \( \ell(s, t) = -\sum_{i=1}^{k} t_i \log s_i + (1 - t_i) \log (1 - s_i) \) is the SCE loss on a single image and \( N_A \) is the number of images in A. \( \mathcal{L}_{BC}(W_P, W_D, B) \) is the binary classification loss defined on training set B:

\[
\mathcal{L}_{BC}(W_P, W_D, B) = \sum_{n=1}^{2N_B} \ell(f_D(a^{(n)}; W_D), b^{(n)}),
\] (6.5)

where \( a^{(n)} \) can be either a predicted score vector \( f_P(x; W_P) \) or a physician-provided label vector \( t \). In the former case, \( b = 1 \). In the latter case, \( b = 0 \).

In the overall loss, the PN learns its weight parameters \( W_P \) by minimizing the SCE loss and maximizing the BC loss. The DN learns its weights parameters \( W_D \) by minimizing the BC loss. Note that the SCE loss and the BC loss cannot be defined on the same training image. By overfitting the training set, the SCE loss can make the predicted label vector to be the same as the groundtruth vector. In this case, it is futile to distill concept-correlations using the BC loss. In AL, designing the architecture of the prediction and discrimination networks is required. However, this is in general much easier than designing a comprehensive objective function for multi-label classification. In our experiments, we simply use common CNNs and feedforward NNs.

Previously, adversarial learning was applied for domain adaptation [119]: a discriminator is learned to judge whether an example is from the source domain or target domain; an encoder is learned so that after being encoded, a sample cannot be identified as being from the source domain or target domain. By doing this, the discrepancy between two domains are eliminated so that data from the source domain can be utilized to improve the task in the target domain. In our approach of using adversarial learning for multi-label classification, a similar idea is explored: using adversarial learning to eliminate the discrepancy between the predicted labels and ground-truth labels in terms of label-correlation patterns, so that the correlations existing in the ground-truth labels can be transferred to the predicted labels.
Tree-of-sequences LSTM Encoder

In this section, we aim at learning embedding vectors for the medical concepts. Each concept has a name (a sequence of words) that tells the semantics of this concept. We use a sequential LSTM (SLSTM) to encode this name. Meanwhile, in the hierarchical ontology, the concepts possess hierarchical relationships. To capture such relationships, on top of the encodings generated by the SLSTMs, we build a tree LSTM (TLSTM) [320] along the hierarchy, ending up with a tree-of-sequences LSTM model (Figure 6.5).

Sequential LSTM A sequential LSTM (SLSTM) [163] network is a special type of recurrent neural network that (1) learns the latent representation (which usually reflects certain semantic information) of words; (2) models the sequential structure among words. In the word sequence, each word \( t \) is allocated with an SLSTM unit, which consists of the following components: input gate \( i_t \), forget gate \( f_t \), output gate \( o_t \), memory cell \( c_t \), and hidden state \( s_t \). These components (vectors) are computed as follows:

\[
\begin{align*}
i_t &= \sigma(W^{(i)}s_{t-1} + U^{(i)}x_t + b^{(i)}) \\
f_t &= \sigma(W^{(f)}s_{t-1} + U^{(f)}x_t + b^{(f)}) \\
o_t &= \sigma(W^{(o)}s_{t-1} + U^{(o)}x_t + b^{(o)}) \\
c_t &= i_t \odot \tanh(W^{(c)}s_{t-1} + U^{(c)}x_t + b^{(c)}) + f_t \odot c_{t-1} \\
s_t &= o_t \odot \tanh(c_t),
\end{align*}
\]

where \( x_t \) is the embedding vector of word \( t \). \( W, U \) are component-specific weight matrices and \( b \) are bias vectors.

Tree-of-sequences LSTM We use a bi-directional tree LSTM (TLSTM) [320, 368] to capture the hierarchical relationships among concepts. The inputs of this LSTM include the tree structure of the concept ontology and hidden states of individual concepts produced the SLSTMs. It consists of a bottom-up TLSTM and a top-down TLSTM, which produce two hidden states \( h_\uparrow \) and \( h_\downarrow \) at each node in the tree.

In the bottom-up TLSTM, an internal node (representing a concept \( C \), having \( M \) children) is comprised of the following components: an input gate \( i_\uparrow \), an output gate \( o_\uparrow \), a memory cell \( c_\uparrow \), a hidden state \( h_\uparrow \), and \( M \) child-specific forget gates \( \{ f^{(m)}_\uparrow \}_{m=1}^M \) where \( f^{(m)}_\uparrow \) corresponds to the \( m \)-th child. The transition equations among components are:

\[
\begin{align*}
i_\uparrow &= \sigma(\sum_{m=1}^M W^{(i,m)}_\uparrow h^{(m)}_\downarrow + U^{(i)}s + b^{(i)}_\uparrow) \\
f^{(m)}_\uparrow &= \sigma(W^{(f,m)}_\uparrow h^{(m)}_\downarrow + U^{(f,m)}s + b^{(f,m)}_\uparrow) \\
o_\uparrow &= \sigma(\sum_{m=1}^M W^{(o,m)}_\uparrow h^{(m)}_\downarrow + U^{(o)}s + b^{(o)}_\uparrow) \\
u_\uparrow &= \tanh(\sum_{m=1}^M W^{(u,m)}_\uparrow h^{(m)}_\downarrow + U^{(u)}s + b^{(u)}_\uparrow) \\
c_\uparrow &= i_\uparrow \odot u_\uparrow + \sum_{m=1}^M f^{(m)}_\uparrow \odot c^{(m)}_\downarrow \\
h_\uparrow &= o_\uparrow \odot \tanh(c_\uparrow),
\end{align*}
\]

where \( s \) is the SLSTM hidden state that encodes the name of the concept \( C \). \( \{ b^{(m)}_\uparrow \}_{m=1}^M \) and \( \{ c^{(m)}_\uparrow \}_{m=1}^M \) are the bottom-up TLSTM hidden states and memory cells of the children. \( W, U, b \)
Figure 6.6: Contextual attention module. For each patch, an attention score is calculated by comparing the representation of this patch and that of the entire image. These patches are re-weighted using the corresponding attention scores to generate an attentional representation for this image.

Figure 6.7: Attentional convolution. Each pixel is weighted by the attention score of the patch containing this pixel. Then the weighted pixels are convoluted with the filter weights.

are component-specific weight matrices and bias vectors. For a leaf node having no children, its only input is the SLSTM hidden state \( s \) and no forget gates are needed. The transition equations are:

\[
\begin{align*}
    i_t &= \sigma(U^{(i)}s + b_t^{(i)}) \\
    o_t &= \sigma(U^{(o)}s + b_t^{(o)}) \\
    u_t &= \tanh(U^{(u)}s + b_t^{(u)}) \\
    c_t &= i_t \odot u_t \\
    h_t &= o_t \odot \tanh(c_t).
\end{align*}
\]  

(6.8)

In the top-down TLSTM, for a non-root node, it has the following components: an input gate \( i_t \), a forget gate \( f_t \), an output gate \( o_t \), a memory cell \( c_t \), and a hidden state \( h_t \). The transition
The equations are:

\[
\begin{align*}
    i_\downarrow &= \sigma(W^{(i)}_\downarrow h^{(p)}_\downarrow + b^{(i)}_\downarrow) \\
    f_\downarrow &= \sigma(W^{(f)}_\downarrow h^{(p)}_\downarrow + b^{(f)}_\downarrow) \\
    o_\downarrow &= \sigma(W^{(o)}_\downarrow h^{(p)}_\downarrow + b^{(o)}_\downarrow) \\
    u_\downarrow &= \tanh(W^{(u)}_\downarrow h^{(p)}_\downarrow + b^{(u)}_\downarrow) \\
    c_\downarrow &= i_\downarrow \odot u_\downarrow + f_\downarrow \odot c^{(p)}_\downarrow \\
    h_\downarrow &= o_\downarrow \odot \tanh(c_\downarrow),
\end{align*}
\]  

(6.9)

where \(h^{(p)}_\downarrow\) and \(c^{(p)}_\downarrow\) are the top-down TLSTM hidden state and memory cell of the parent of this node. For the root node which has no parent, \(h_\downarrow\) cannot be computed using the above equations. Instead, we set \(h_\downarrow\) to \(h_\uparrow\) (the bottom-up TLSTM hidden state generated at the root node). \(h_\uparrow\) captures the semantics of all concepts in this ontology, which is then propagated downwards to each individual concept via the top-down TLSTM dynamics.

We concatenate the hidden states of the two directions to obtain the bidirectional TLSTM encoding of each concept \(h = [h_\downarrow; h_\uparrow]\). The bottom-up TLSTM composes the semantics of children (representing sub-concepts) and merges them into the current node, which hence captures child-to-parent relationship. The top-down TLSTM makes each node inherit the semantics of its parent, which captures parent-to-child relation. As a result, the hierarchical relationship among concepts is encoded in the hidden states.

**Contextual Attention**

When reading a medical image, the physicians care more about the regions that show abnormalities. These abnormalities are usually indicators of diseases and prompt the physicians to take treatment actions. The abnormal regions are usually small, which are difficult to detect and tag. It is important to localize these regions and encourage the tagger to pay more attention to them.

The abnormal regions can be spotted by comparing them with their context – the entire image. It is often the case that majority of an image contains normal tissues, whose visual appearance differs from the abnormalities. This observation motivates us to design a contextual attention mechanism that calculates the level of abnormality in each region by contrasting this region with the entire image.

Figure 6.6 presents an illustration. We first divide the input image into patches. For each patch, we use a patch encoder to extract its representation. For the entire image (which is deemed as context), an image encoder is used to capture the holistic information of this image. For each patch, its representation and the image representation are concatenated and fed into an attention module which generates a score indicating how abnormal this patch is. Then we use these attention scores to weight pixels. The weighted pixels are fed into an attentional encoder to generate an attentional representation for this image. For simplicity, we assume the patches are non-overlapped, hence each pixel belongs to exactly one patch. The value of a pixel is weighted by the attention score of the patch containing it. In the attentional encoder, attentional convolution is performed: the filters take weighted pixels as inputs to calculate the feature map. Figure 6.7 shows an example. The \(6 \times 6\) image is divided into 4 patches, whose attention scores are 0.6, 0.1, 0.3, and 0.7. When a \(3 \times 3\) filter (denoted by a red box) is applied, each pixel in the receptive field
is weighted using the attention score of the patch containing this pixel. Then the convolution is performed between the weighted pixels and the filter weights, by calculating the following equation:

\[ c = f \left( \sum_{i=1}^{n} a_i p_i w_i + b \right) \]

where \( \{a_i\} \) are the attention scores, \( \{p_i\} \) are the pixel values, and \( \{w_i\} \) are the weights of filters. Extending the method to overlapped patches is straightforward: the weight of a pixel is calculated by averaging the attention scores of overlapping patches that contain this pixel.

### 6.3.2 Evaluation

**Datasets**

We used two medical image datasets obtained from a teaching hospital. The first dataset contains 47933 pathology images which are split into a training set with 33553 images, a validation set with 7190 images, and a test set with 7190 images. The second dataset contains 39827 radiology images, which are split into train/validation/test sets with 27879, 5974, and 5974 images respectively. Labels in the first dataset are organized by doctors into a 4-level hierarchy. The number of labels in level 1-4 are 16, 38, 137, 249 respectively. The number of leaf nodes is 272. The average labels each image has is 3.7. Each label has a name. The minimum, average, and maximum number of words in the label names are 1, 5.3, 9 respectively. Labels in the second dataset are organized by doctors into a 3-level hierarchy. The number of labels in level 1-3 are 12, 51, 196 respectively. The number of leaf nodes is 211. The average labels each image has is 3.1. The minimum, average, and maximum number of words in the label names are 1, 4.4, 8 respectively. Each image is labeled by at least 3 doctors and the labels are decided using majority vote. The kappa statistics (range is \([-1,1]\)) for measuring inter-annotator reliability is 0.83, indicating high consistency among annotators. Since it is too costly to annotate fine-grained hierarchical labels for all raw images in the PACS, we did not use all images there. The distribution of classes’ frequencies is imbalanced: some classes appear in many images while others are less frequent. The number of tags that each image has follows a Gaussian distribution.

**Experimental Setup**

Data augmentation by cropping and rotating was applied. The images were resized to 3000×2000. Each image was divided into 16×16 overlapping patches with a stride of 8. In the contextual attentional encoder, both the image encoder and patch encoder were set to ResNet-50 [153], where the 1000-dimensional vector produced by the fully-connected layer is used as the representation of an image or a patch. The attention module was a feed-forward network with 2 hidden layers where the number of unit was 300 and 100 respectively. The activation function was set to ReLU. The attentional encoder was also set to ResNet-50. In the tree-of-sequences LSTM encoder, the hidden state size of both the sequential and tree-structured LSTM was set to 100. The size of word embedding was set to 128. In the adversarial multi-label tagger, 70% training images were used as set A to learn label-image relevance. The rest were used as set B to learn label-correlations. Both the predictive and discriminative networks were feed-forward networks with 2 hidden layers where the number of units was 300 and 100 respectively. We used AdaGrad [255] with a learning rate of 0.1 and batch size of 32 to learn model parameters. In LSTM
### Table 6.5: Average sensitivity and specificity.

The first, second, and third panels compare baselines in the ablation study of (1) adversarial learning for multi-label tagging, (2) tree-of-sequences LSTM for capturing hierarchical relationship, and (3) contextual attention for identifying abnormalities. The fourth panel compares baselines for holistic comparison.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Pathology</th>
<th></th>
<th>Radiology</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sensitivity</td>
<td>Specificity</td>
<td>Sensitivity</td>
<td>Specificity</td>
</tr>
<tr>
<td>CRF [419]</td>
<td>0.30</td>
<td>0.34</td>
<td>0.48</td>
<td>0.50</td>
</tr>
<tr>
<td>SPEN [36]</td>
<td>0.32</td>
<td>0.34</td>
<td>0.46</td>
<td>0.48</td>
</tr>
<tr>
<td>CNN-RNN [346]</td>
<td>0.31</td>
<td>0.35</td>
<td>0.48</td>
<td>0.51</td>
</tr>
<tr>
<td>DPP [375]</td>
<td>0.32</td>
<td>0.34</td>
<td>0.47</td>
<td>0.49</td>
</tr>
<tr>
<td>No-AL</td>
<td>0.30</td>
<td>0.33</td>
<td>0.46</td>
<td>0.48</td>
</tr>
<tr>
<td>LET [38]</td>
<td>0.30</td>
<td>0.35</td>
<td>0.48</td>
<td>0.50</td>
</tr>
<tr>
<td>HD-CNN [389]</td>
<td>0.31</td>
<td>0.35</td>
<td>0.46</td>
<td>0.49</td>
</tr>
<tr>
<td>HybridNet [166]</td>
<td>0.32</td>
<td>0.36</td>
<td>0.45</td>
<td>0.49</td>
</tr>
<tr>
<td>B-CNN [425]</td>
<td>0.30</td>
<td>0.34</td>
<td>0.48</td>
<td>0.51</td>
</tr>
<tr>
<td>No-TLSTM</td>
<td>0.29</td>
<td>0.34</td>
<td>0.45</td>
<td>0.49</td>
</tr>
<tr>
<td>Bottom-up TLSTM</td>
<td>0.33</td>
<td>0.36</td>
<td>0.48</td>
<td>0.51</td>
</tr>
<tr>
<td>LPA [385]</td>
<td>0.30</td>
<td>0.35</td>
<td><strong>0.50</strong></td>
<td>0.51</td>
</tr>
<tr>
<td>SA [71]</td>
<td>0.31</td>
<td>0.35</td>
<td>0.49</td>
<td>0.49</td>
</tr>
<tr>
<td>No-CA</td>
<td>0.30</td>
<td>0.34</td>
<td>0.48</td>
<td>0.49</td>
</tr>
<tr>
<td>DTHMLC [337]</td>
<td>0.20</td>
<td>0.23</td>
<td>0.39</td>
<td>0.40</td>
</tr>
<tr>
<td>DenseNet-LSTM [396]</td>
<td>0.27</td>
<td>0.29</td>
<td>0.44</td>
<td>0.46</td>
</tr>
<tr>
<td>OS [171]</td>
<td>0.22</td>
<td>0.23</td>
<td>0.41</td>
<td>0.44</td>
</tr>
<tr>
<td>Our full method</td>
<td><strong>0.33</strong></td>
<td><strong>0.37</strong></td>
<td><strong>0.50</strong></td>
<td><strong>0.52</strong></td>
</tr>
</tbody>
</table>

Table 6.5: Average sensitivity and specificity. On the first, second, and third panel are baselines compared in the ablation study of (1) adversarial learning for multi-label tagging, (2) tree-of-sequences LSTM for capturing hierarchical relationship, and (3) contextual attention for identifying abnormalities. On the fourth panel are baselines for holistic comparison.

### Ablation Study

We perform ablation study to verify the effectiveness of each module.

**Adversarial learning for multi-label tagging** To evaluate the efficacy of adversarial learning (AL), we removed it from the model. In Figure 6.4, the branch associated with training set B (including the discriminative network and binary classification loss) was taken away. All training images were put into set A to learn label-image relevance. In addition, we compared with four state-of-the-art methods proposed for capturing label-correlation in multi-label classification, by replacing AL with each of them while keeping the other modules intact. These baselines include (1) conditional random field (CRF) [419]; (2) structured prediction energy network (SPEN) [36]; (3) CNN-RNN [346]; (4) determinantal point process (DPP) [205, 375]. In training, the network was unrolled for 60 iterations. For performance evaluation, we used sensitivity (true positive rate) and specificity (true negative rate), which are the most widely used evaluation metrics in clinical trial. To address the imbalance issue of classes’ frequency, in the loss function in Eq. (6.4), we gave infrequent classes larger weights, where the weight of a class is proportional to the reciprocal of the frequency of this class.

**Ablation Study**

We perform ablation study to verify the effectiveness of each module.
SPEN, The local and global energy networks are chosen to be feed-forward networks consisting of two hidden layers, with 300 and 150 units in each layer. Gradient descent was applied to make predictions, where the momentum and learning rate were set to 0.95 and 0.01 respectively. In CNN-RNN, the size of hidden states in LSTM was set to 128. The network was trained using SGD with momentum 0.9, weight decay 1e-4, and dropout rate 0.5. In the DPP baseline, we feed the image representation produced by the contextual attention encoder and concept representations produced by the tree-of-sequence LSTM encoder into a conditional kernel function represented by a label-input dependency network (LIDN) and a label-correlation network (LCN). The LIDN is configured to be a fully-connected network with 2 hidden layers where the number of units in the first and second layer is 200 and 100 respectively and the activation function is ReLU. The LCN has two hidden layers as well, each having 100 and 50 units respectively.

The first panel of Table 6.5 shows the results, from which we observe the following. First, after AL is removed (denoted by No-AL), the sensitivity and specificity are significantly dropped. No-AL ignores label-correlations and selects labels purely based on their relevance to the image, hence leading to inferior performance. Second, compared with the four baselines designed for capturing label-correlations, our full method (which uses AL) achieves better performance, suggesting that AL is more effective in capturing correlations. These baselines design explicit objective functions to encode correlations. The correlations might be very complicated and diverse, which are difficult to be captured by a single objective function. Instead, our approach implicitly learns such correlation by making the predicted labels indistinguishable from the groundtruth ones. It has more flexibility and expressivity to capture various types of correlations. The advantage of DPP is that it is able to capture high-order correlations among labels while being able to perform the exact learning of model parameters (without approximation) with cubic computational complexity (in terms of the number of unique classes). Its drawback is exact inference cannot be achieved in polynomial time. Approximate inference may bear large errors. The disadvantage of CRF is both inference and parameter learning cannot be performed exactly.

Tree-of-sequences LSTM To evaluate this module, we compared with the two configurations: (1) No TLSTM, which removes the tree LSTM and directly uses the hidden states produced by the sequential LSTM as final representations of concepts; (2) Bottom-up TLSTM, which removes the hidden states generated by the top-down TLSTM. In addition, we compared with four hierarchical classification baselines including (1) hierarchical deep CNN (HD-CNN) [389], (2) HybridNet [166], (3) Branch CNN (B-CNN) [425], (4) label embedding tree (LET) [38], by using them to replace the bidirectional tree LSTM while keeping other modules untouched. The second panel of Table 6.5 shows the sensitivity and specificity achieved by these methods. We make the following observations. First, removing tree LSTM (No TLSTM) greatly degrades performance since the hierarchical relationship among labels is ignored. Second, the bottom-up tree LSTM alone performs less well than our full method that uses the bi-directional tree LSTM. This demonstrates the necessity of the top-down TLSTM, which ensures every two labels are connected by directed paths and can more expressively capture label-relations in the hierarchy. Third, our full method outperforms the four baselines. The possible reason is that our method directly builds labels’ hierarchical relationship into their representations while the baselines perform representation-learning and relationship-capturing separately.
### Table 6.6: Performance versus patch sizes and stride sizes.

<table>
<thead>
<tr>
<th></th>
<th>Pathology</th>
<th></th>
<th></th>
<th>Radiology</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IoU</td>
<td>Sensitivity</td>
<td>Specificity</td>
<td>IoU</td>
<td>Sensitivity</td>
<td>Specificity</td>
</tr>
<tr>
<td>Patch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.25</td>
<td>0.32</td>
<td>0.36</td>
<td>0.40</td>
<td>0.47</td>
<td>0.49</td>
</tr>
<tr>
<td>16</td>
<td>0.29</td>
<td>0.33</td>
<td>0.37</td>
<td>0.41</td>
<td>0.49</td>
<td>0.50</td>
</tr>
<tr>
<td>32</td>
<td>0.28</td>
<td>0.32</td>
<td>0.34</td>
<td><strong>0.43</strong></td>
<td><strong>0.50</strong></td>
<td><strong>0.52</strong></td>
</tr>
<tr>
<td>64</td>
<td>0.24</td>
<td>0.30</td>
<td>0.31</td>
<td>0.39</td>
<td>0.47</td>
<td>0.48</td>
</tr>
<tr>
<td>Stride</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.28</td>
<td>0.30</td>
<td><strong>0.37</strong></td>
<td>0.40</td>
<td>0.48</td>
<td>0.51</td>
</tr>
<tr>
<td>8</td>
<td><strong>0.29</strong></td>
<td><strong>0.33</strong></td>
<td><strong>0.37</strong></td>
<td><strong>0.43</strong></td>
<td><strong>0.50</strong></td>
<td><strong>0.52</strong></td>
</tr>
<tr>
<td>16</td>
<td>0.27</td>
<td>0.31</td>
<td>0.36</td>
<td>0.42</td>
<td>0.47</td>
<td>0.50</td>
</tr>
</tbody>
</table>

**Contextual attention** In the evaluation of this module, we compared with No-CA which removes the contextual attention (CA) module, and two other attention models: (1) label-patch attention (LPA) [69, 385, 394, 400]; (2) scale attention (SA) [71]. The third panel of Table 6.5 shows the performance scores achieved by these methods. As can be seen, the sensitivity and specificity under No-CA is lower than our full method which uses CA, which demonstrates the effectiveness of contextual attention. CA is able to localize the small abnormal regions and pays more attention to them, which leads to the successful tagging of abnormalities. Compared with other attention baselines, our full method with CA achieves better performances. This indicates that for medical images, the contextual information embodied in the entire image is very valuable for distilling “attention”.

We asked doctors to label the abnormal regions in 100 pathology and 100 radiology images. We compared the abnormal regions detected by our contextual attention model with the groundtruth using Intersection over Union (IoU). We evaluated how patch size affects performance. Table 6.6 shows IoU and sensitivity/specificity of tagging under different patch sizes (the stride size is set to 8). As can be seen, a patch size in the middle ground (that best matches the scale of abnormal regions) yields the best abnormality detection and tagging performance. We also evaluated how the stride size in overlapping patches affects performance. Fixing the patch size to 16, we tried stride sizes of 4, 8, and 16 (equivalent to nonoverlap). As can be seen from the table, overlap with a stride size 8 works better than nonoverlap.

**Holistic Comparison with Other Baselines**

In addition to evaluating the three modules individually, we also compared the entire model with three other baselines, including (1) decision trees for hierarchical multi-label classification (DTHMLC) [337], (2) DenseNet-LSTM designed in [396] for chest x-ray classification, (3) Occlusion sensitivity (OS) used in [171] for abnormality localization in x-rays.

The fourth panel of Table 6.5 shows the comparison with these baselines. As can be seen, our approach achieves much better performances on both datasets than the baselines. DTHMLC performs less well probably because it lacks the ability to learn deep visual features. DenseNet-LSTM lacks the ability to explore concept-hierarchy. OS cannot be trained in an end-to-end fashion which leads to inferior performance.

We evaluated the effects of tagging on image search. We used the learned models to tag
images in the PACS. For either pathology or radiology images, we randomly sampled 100 tags as queries, then performed retrieval by tag matching. We compared with a baseline: retrieval by checking whether the query tag is contained in the image reports. Retrieval performance was evaluated using precision@10: among the top 10 retrieved images, how many are relevant to the query (whether being relevant is labeled by doctors). The precision@10 achieved by report-based retrieval is 0.24 for pathology and 0.32 for radiology. Our tagging method improves the precision@10 to 0.29 for pathology and 0.40 for radiology.

On the pathology dataset, our method is significantly better than CRF, No-AL, LET, B-CNN, No-TLSTM, LPA, No-CA, DTHMLC, DenseNet-LSTM, OS with p-value < 0.01, and is significantly better than SPEN, CNN-RNN, DPP, HD-CNN, HybridNet, SA with p-value < 0.05. On the pathology dataset, our method is significantly better than SPEN, DPP, No-AL, HD-CNN, HybridNet, No-TLSTM, No-CA, DTHMLC, DenseNet-LSTM, OS with p-value < 0.01, and is significantly better than CRF, CNN-RNN, LET, B-CNN, Bottom-up TLSTM, SA with p-value < 0.05.

6.3.3 Related works

Medical image categorization Many works [107, 208, 230, 321] have been devoted to the classification of medical images related to skin cancer [107], chest x-ray [349], pathology [178], to name a few. In [396], label-correlation is considered for more accurate classification of multi-label chest x-ray images. The localization of abnormality in chest x-rays is studied in [171] using the occlusion sensitivity method [410]. The problem setting studied in our work is different from the existing ones. We simultaneously consider two important facts: (1) each image may have multiple labels; (2) the classes have a hierarchical structure. The existing works consider neither or only one of them.

Multi-label classification (MLC) In computer vision and machine learning, MLC has been widely studied [56, 57, 123, 124, 189, 190, 285, 317, 351, 351, 362, 387, 390, 415, 416]. In MLC, one fundamental problem is how to capture the correlations among labels. Many approaches have been proposed based on graphical models [36, 70, 126, 147, 151, 201, 227, 316, 322, 412], latent space projection [36, 180, 183, 221, 228, 356, 397], and class chaining [77, 289, 346]. To retain computational efficiency, these methods typically make strong assumptions, such as the order of correlation is less than three [70, 126, 151, 201, 227], the dependency relationship among labels is linear [147], the labels are independent conditioned on a latent space [36, 228, 356, 397], etc. Some of these methods [316, 322, 412] lack the flexibility to perform deep visual feature learning in an end-to-end manner.

The existing methods define explicit objective functions to encode label correlations. In reality, the correlations might be of a variety of complicated forms which are unknown to the human modelers. As a result, it is difficult to specify a comprehensive objective to capture all of them. In this paper, we propose a different approach: instead of manually defining such an objective, we use adversarial learning to implicitly transfer the correlations manifested in human-provided labels into predicted labels by making the two types of labels indistinguishable.
Hierarchical classification  Leveraging the hierarchical structure among classes to improve classification accuracy has been widely studied in [38, 97, 98, 166, 247, 389]. Many methods [28, 121, 166, 231, 389, 425, 430] propose to learn coarse-to-fine classifiers or features along the class hierarchy. In [38], a method is proposed to learn label embeddings where the learning is guided by the label tree. Our work also aims at embedding labels into a latent space and uses LSTM networks to capture long-range nonlinear dependencies among labels. Several works [98, 312, 414] encode the hierarchical relationship among labels as structural constraint, Bayesian priors and potential functions in conditional random field. A variety of works [96, 140, 181, 225, 248] study how to automatically learn the tree-structure among labels. Several studies [33, 64, 65, 337] have been conducted on hierarchical multi-label classification based on decision trees, neural networks, and hierarchical support vector machines.

Visual attention  Visual attention [23, 26, 69, 71, 91, 258, 297, 300, 385, 400] has been widely utilized in computer vision. In many works [69, 385, 394, 400], the attention is computed between an image patch and a label (such as a sentence, a word, an attribute) by comparing their similarity. In a video-based activity recognition task, Sharma et al. [300] use the previous frame to predict the importance of regions in the current frame. Chen et al. [71] design an attention model which takes images at different scales as inputs and predicts the importance of regions at different positions and scales.

6.4 A Neural Architecture for Automated ICD Coding

The International Classification of Diseases (ICD) is a healthcare classification system maintained by the World Health Organization [268]. It provides a hierarchy of diagnostic codes of diseases, disorders, injuries, signs, symptoms, etc. It is widely used for reporting diseases and health conditions, assisting in medical reimbursement decisions, collecting morbidity and mortality statistics, to name a few.

While ICD codes are important for making clinical and financial decisions, medical coding – which assigns proper ICD codes to a patient visit – is time-consuming, error-prone and expensive. Medical coders review the diagnosis descriptions written by physicians in the form of textual phrases and sentences and (if necessary) other information in the electronic medical record of a clinical episode, then manually attribute the appropriate ICD codes by following the coding guidelines [267]. Several types of errors frequently occur. First, the ICD codes are organized in a hierarchical structure. For a node representing a disease \( C \), the children of this node represents the subtypes of \( C \). In many cases, the difference between disease subtypes is very subtle. It is common that human coders select incorrect subtypes. Second, when writing diagnosis descriptions, physicians often utilize abbreviations and synonyms, which causes ambiguity and imprecision when the coders are matching ICD codes to those descriptions [302]. Third, in many cases, several diagnosis descriptions are closely related and should be mapped to a single ICD code. However, unexperienced coders may code each disease separately. Such errors are called unbundling. The cost incurred by coding errors and the financial investment spent on improving coding quality are estimated to be $25 billion per year in the US [109, 210].

197
To reduce coding errors and cost, we aim at building an ICD coding model which automatically and accurately translates the free-text diagnosis descriptions into ICD codes. To achieve this goal, several technical challenges need to be addressed. First, there exists a hierarchical structure among the ICD codes. This hierarchy can be leveraged to improve coding accuracy. On one hand, if code A and B are both children of C, then it is unlikely to simultaneously assign A and B to a patient. On the other hand, if the distance between A and B in the code tree is smaller than that between A and C and we know A is the correct code, then B is more likely to be a correct code than C, since codes with smaller distance are more clinically relevant. How to explore this hierarchical structure for better coding is technically demanding. Second, the diagnosis descriptions and the textual descriptions of ICD codes are written in quite different styles even if they refer to the same disease. In particular, the textual description of an ICD code is formally and precisely worded, while diagnosis descriptions are usually written by physicians in an informal and ungrammatical way, with telegraphic phrases, abbreviations, and typos. Third, it is required that the assigned ICD codes are ranked according to their relevance to the patient. How to correctly determine this order is technically nontrivial. Fourth, as stated earlier, there does not necessarily exist an one-to-one mapping between diagnosis descriptions and ICD codes, and human coders should consider the overall health condition when assigning codes. In many cases, two closely related diagnosis descriptions need to be mapped onto a single combination ICD code. On the other hand, physicians may write two health conditions into one diagnosis description which should be mapped onto two ICD codes under such circumstances.

In this section, we design a neural architecture to automatically perform ICD coding given the diagnosis descriptions. This architecture uses a tree-of-sequences LSTM network (Section 6.3.1) to simultaneously capture the hierarchical relationship among codes and the semantics of each code. An adversarial learning approach is utilized to reconcile the heterogeneous writing styles of diagnosis descriptions and ICD code descriptions. We use isotonic constraints to preserve the importance order among codes and develop an algorithm based on ADMM and isotonic projection to solve the constrained problem. An attentional matching mechanism is employed to perform many-to-one and one-to-many mappings between diagnosis descriptions and codes. On a clinical datasets with 59K patient visits, we demonstrate the effectiveness of the proposed methods.

6.4.1 Methods

Figure 6.8 shows the overview of our approach. The proposed ICD coding model consists of five modules. The model takes the ICD-code tree and diagnosis descriptions (DDs) of a patient as inputs and assigns a set of ICD codes to the patient. The encoder of DDs generates a latent representation vector for a DD. The encoder of ICD codes is the tree-of-sequences long short-term memory (LSTM) network introduced in Section 6.3.1. It takes the textual descriptions of the ICD codes and their hierarchical structure as inputs and produces a latent representation for each code. The representation aims at simultaneously capturing the semantics of each code and the hierarchical relationship among codes. By incorporating the code hierarchy, the model can avoid selecting codes that are subtypes of the same disease and promote the selection of codes that are clinically correlated. The writing styles of DDs and code descriptions (CDs) are largely different, which makes the matching between a DD and a CD error-prone. To address this...
issue, we develop an adversarial learning approach to reconcile the writing styles. On top of the latent representation vectors of the descriptions, we build a discriminative network to distinguish which ones are DDs and which are CDs. The encoders of DDs and CDs try to make such a discrimination impossible. By doing this, the learned representations are independent of the writing styles and facilitate more accurate matching. The representations of DDs and CDs are fed into an attentional matching module to perform code assignment. This attentional mechanism allows multiple DDs to be matched to a single code and allows a single DD to be matched to multiple codes. During training, we incorporate the order of importance among codes as isotonic constraints. These constraints regulate the model’s weight parameters so that codes with higher importance are given larger prediction scores.

**Representation Learning**

We use the tree-of-sequences LSTM network (Section 6.3.1) to encode the ICD codes. Each code has a description (a sequence of words) that tells the semantics of this code. A sequential LSTM (SLSTM) is used to encode this description. To capture the hierarchical relationship among codes, we build a tree-structured LSTM (TLSTM) along the code tree. The inputs of this LSTM include the code hierarchy and the hidden states of individual codes produced by the SLSTMs. It consists of a bottom-up TLSTM and a top-down TLSTM, which produce two hidden states $h_↑$ and $h_↓$ at each node in the tree.

For the diagnosis descriptions of a patient, we use an SLSTM network to encode each description individually. The weight parameters of this SLSTM are tied with those of the SLSTM used for encoding code descriptions.
Attentional Matching

Next, we introduce how to map the DDs to codes. We denote the hidden representations of DDs and codes as \( \{h_m\}_{m=1}^M \) and \( \{u_n\}_{n=1}^N \) respectively, where \( M \) is the number of DDs of one patient and \( N \) is the total number of codes in the dataset. The mapping from DDs to codes is not one-to-one. In many cases, a code is assigned only when a certain combination of \( K \) \((1 < K \leq M)\) diseases simultaneously appear within the \( M \) DDs and the value of \( K \) depends on this code. Among the \( K \) diseases, their importance of determining the assignment of this code is different. For the rest \( M - K \) DDs, we can consider their importance score to be zero. We use a soft-attention mechanism \([26]\) to calculate these importance scores. For a code \( u_n \), the importance of a DD \( h_m \) to \( u_n \) is calculated as \( a_{nm} = u_n^\top h_m \). We normalize the scores \( \{a_{nm}\}_{m=1}^M \) of all DDs into a probabilistic simplex using the softmax operation: \( \tilde{a}_{nm} = \exp(a_{nm}) / \sum_{l=1}^M \exp(a_{nl}) \). Given these normalized importance scores \( \{\tilde{a}_{nm}\}_{m=1}^M \), we use them to weight the representations of DDs and get a single attentional vector of the \( M \) DDs: \( \tilde{h}_n = \sum_{m=1}^M \tilde{a}_{nm} h_m \). Then we concatenate \( \tilde{h}_n \) and \( u_n \), and use a linear classifier to predict the probability that code \( n \) should be assigned: \( \tilde{p}_n = \text{sigmoid}(\tilde{w}_n^\top [\tilde{h}_n; u_n] + b_n) \), where the coefficients \( \tilde{w}_n \) and bias \( b_n \) are specific to code \( n \).

We train the weight parameters \( \Theta \) of the proposed model using the data of \( L \) patient visits. \( \Theta \) includes the sequential LSTM weights \( W_s \), tree LSTM weights \( W_t \), and weights \( W_p \) in the final prediction layer. Let \( c^{(l)} \in \mathbb{R}^N \) be a binary vector where \( c_n^{(l)} = 1 \) if the \( n \)-th code is assigned to this patient and \( c_n^{(l)} = 0 \) if otherwise. \( \Theta \) can be learned by minimizing the following prediction loss:

\[
\min_{\Theta} \mathcal{L}_{\text{pred}}(\Theta) = \sum_{l=1}^L \sum_{n=1}^N \text{CE}(\tilde{p}_n^{(l)}, c_n^{(l)}), \quad (6.10)
\]

where \( \tilde{p}_n^{(l)} \) is the predicted probability that code \( n \) is assigned to patient visit \( l \) and \( p_n^{(l)} \) is a function of \( \Theta \). CE(·, ·) is the cross-entropy loss.

Adversarial Reconciliation of Writing Styles

We use an adversarial learning \([134]\) approach to reconcile the different writing styles of diagnosis descriptions (DDs) and code descriptions (CDs). The basic idea is: after encoded, if a description cannot be discerned to be a DD or a CD, then the difference in their writing styles is eliminated. We build a discriminative network which takes the encoding vector of a description as input and tries to identify it as a DD or CD. The encoders of DDs and CDs adjust their weight parameters so that such a discrimination is difficult to be achieved by the discriminative network. Consider all the descriptions \( \{t_r; y_r\}_{r=1}^R \) where \( t_r \) is a description and \( y_r \) is a binary label. \( y_r = 1 \) if \( t_r \) is a DD and \( y_r = 0 \) if otherwise. Let \( f(t_r; W_s) \) denote the sequential LSTM (SLSTM) encoder parameterized by \( W_s \). This encoder is shared by the DDs and CDs. Note that for CDs, a tree LSTM is further applied on top of the encodings produced by the SLSTM. We use the SLSTM encoding vectors of CDs as the input of the discriminative network rather than using the TLSTM encodings since the latter are irrelevant to writing styles. Let \( g(f(t_r; W_s); W_d) \) denote the discriminative network parameterized by \( W_d \). It takes the encoding vector \( f(t_r; W_s) \) as input and produces the probability that \( t_r \) is a DD. Adversarial learning is performed by solving this
The discriminative network tries to differentiate DDs from CDs by minimizing this classification loss while the encoder maximizes this loss so that DDs and CDs are not distinguishable.

**Isotonic Constraints**

Next, we incorporate the importance order among ICD codes. For the $D^{(l)}$ codes assigned to patient $l$, without loss of generality, we assume the order is $1 \succ 2 \cdots \succ D^{(l)}$ (the order is given by human coders as groundtruth in the MIMIC-III dataset). We use the predicted probability $p_i$ ($1 \leq i \leq D^{(l)}$) defined in Section 6.4.1 to characterize the importance of code $i$. To incorporate the order, we impose an isotonic constraint on the probabilities: $p_1^{(l)} \succ p_2^{(l)} \cdots \succ p_{D^{(l)}}^{(l)}$, and solve the following problem:

$$\min_\Theta \mathcal{L}_{\text{pred}}(\Theta) + \max_{W_d} (-\lambda \mathcal{L}_{\text{adv}}(W_s, W_d))$$

$$s.t. \quad p_1^{(l)} \succ p_2^{(l)} \cdots \succ p_{D^{(l)}}^{(l)}$$

$$\forall l = 1, \cdots, L$$

where the probabilities $p_i^{(l)}$ are functions of $\Theta$ and $\lambda$ is a tradeoff parameter.

We develop an algorithm based on the alternating direction method of multiplier (ADMM) [52] to solve the problem defined in Eq. (6.12). Let $p^{(l)}$ be a $|D^{(l)}|$-dimensional vector where the $i$-th element is $p_i^{(l)}$. We first write the problem into an equivalent form

$$\min_\Theta \mathcal{L}_{\text{pred}}(\Theta) + \max_{W_d} (-\lambda \mathcal{L}_{\text{adv}}(W_s, W_d))$$

$$s.t. \quad p^{(l)} = q^{(l)}$$

$$q_1^{(l)} \succ q_2^{(l)} \cdots \succ q_{|D^{(l)}|}^{(l)}$$

$$\forall l = 1, \cdots, L$$

Then we write down the augmented Lagrangian

$$\min_{\Theta, q, v} \mathcal{L}_{\text{pred}}(\Theta) + \max_{W_d} (-\lambda \mathcal{L}_{\text{adv}}(W_s, W_d)) + \langle p^{(l)} - q^{(l)}, v^{(l)} \rangle + \frac{\rho}{2} \| p^{(l)} - q^{(l)} \|_2^2$$

$$s.t. \quad q_1^{(l)} \succ q_2^{(l)} \cdots \succ q_{|D^{(l)}|}^{(l)}$$

$$\forall l = 1, \cdots, L$$

We solve this problem by alternating between $\{p_i^{(l)}\}_{i=1}^L$, $\{q_i^{(l)}\}_{i=1}^L$ and $\{v_i^{(l)}\}_{i=1}^L$. The sub-problem defined over $q_i^{(l)}$ is

$$\min_{q_i^{(l)}} -\langle q_i^{(l)}, v_i^{(l)} \rangle + \frac{\rho}{2} \| p_i^{(l)} - q_i^{(l)} \|_2^2$$

$$s.t. \quad q_1^{(l)} \succ q_2^{(l)} \cdots \succ q_{|D^{(l)}|}^{(l)}$$

which is an isotonic projection problem and can be solved via the algorithm proposed in [403]. With $\{q_i^{(l)}\}_{i=1}^L$ and $\{v_i^{(l)}\}_{i=1}^L$ fixed, the sub-problem is $\min_\Theta \mathcal{L}_{\text{pred}}(\Theta) + \max_{W_d} (-\lambda \mathcal{L}_{\text{adv}}(W_s, W_d))$ which can be solved using stochastic gradient descent (SGD). The update of $v_i^{(l)}$ is simple: $v_i^{(l)} = v_i^{(l)} + \rho (p_i^{(l)} - q_i^{(l)})$. 201
Diagnosis Descriptions
1. Prematurity at 35 4/7 weeks gestation
2. Twin number two of twin gestation
3. Respiratory distress secondary to transient tachypnea of the newborn
4. Suspicion for sepsis ruled out

Assigned ICD Codes
1. V31.00 (Twin birth, mate liveborn, born in hospital, delivered without mention of cesarean section)
2. 765.18 (Other preterm infants, 2,000-2,499 grams)
3. 775.6 (Neonatal hypoglycemia)
4. 770.6 (Transitory tachypnea of newborn)
5. V29.0 (Observation for suspected infectious condition)
6. V05.3 (Need for prophylactic vaccination and inoculation against viral hepatitis)

Table 6.7: The diagnosis descriptions of a patient visit and the assigned ICD codes. Inside the parentheses are the descriptions of the codes. The codes are ranked according to descending importance.

6.4.2 Evaluation

In this section, we present experiment results.

Dataset

We performed the study on the publicly available MIMIC-III dataset [184], which contains de-identified electronic health records (EHRs) of 58,976 patient visits in the Beth Israel Deaconess Medical Center from 2001 to 2012. Each EHR has a clinical note called discharge summary, which contains multiple sections of information, such as ‘discharge diagnosis’, ‘past medical history’, etc. From the ‘discharge diagnosis’ and ‘final diagnosis’ sections, we extract the diagnosis descriptions (DDs) written by physicians. Each DD is a short phrase or a sentence, articulating a certain disease or condition. Medical coders perform ICD coding mainly based on DDs. Following such a practice, in this paper, we set the inputs of the automated coding model to be the DDs while acknowledging that other information in the EHRs is also valuable and is referred to by coders for code assignment. For simplicity, we leave the incorporation of non-DD information to future study.

Each patient visit is assigned with a list of ICD codes, ranked in descending order of importance and relevance. For each visit, the number of codes is usually not equal to the number of diagnosis descriptions. These groundtruth codes serve as the labels to train our coding model. The entire dataset contains 6,984 unique codes, each of which has a textual description, describing a disease, symptom, or condition. The codes are organized into a hierarchy where the top-level codes correspond to general diseases while the bottom-level ones represent specific diseases. In the code tree, children of a node represent subtypes of a disease. Table 6.7 shows the
DDs and codes of an exemplar patient.

Experimental Settings

Out of the 6,984 unique codes, we selected 2,833 codes that have the top frequencies to perform the study. We split the data into a train/validation/test dataset with 40k/7k/12k patient visits respectively. The hyperparameters were tuned on the validation set. The SLSTMs are bidirectional and dropout with 0.5 probability [313] was used. The size of hidden states in all LSTMs was set to 100. The word embeddings were trained on the fly and their dimension was set to 200. The tradeoff parameter $\lambda$ was set to 0.1. The parameter $\rho$ in the ADMM algorithm was set to 1. In the SGD algorithm for solving $\min_{\Theta} \mathcal{L}_{\text{pred}}(\Theta) + \max_{W_d}( -\lambda \mathcal{L}_{\text{adv}}(W_s, W_d))$, we used the Adam [196] optimizer with an initial learning rate 0.001 and a mini-batch size 20. Sensitivity (true positive rate) and specificity (true negative rate) were used to evaluate the code assignment performance. We calculated these two scores for each individual code on the test set, then took a weighted (proportional to codes’ frequencies) average across all codes. To evaluate the ranking performance of codes, we used the normalized discounted cumulative gain (NDCG) [179].

Ablation Study

We performed ablation study to verify the effectiveness of each module in our model. To evaluate module X, we remove it from the model without changing other modules and denote such a baseline by No-X. The comparisons of No-X with the full model are given in Table 6.8.

Tree-of-sequences LSTM

To evaluate this module, we compared with the two configurations: (1) No-TLSTM, which removes the tree LSTM and directly uses the hidden states produced by the sequential LSTM as the final representations of codes; (2) Bottom-up TLSTM, which removes the hidden states generated by the top-down TLSTM. In addition, we compared with four hierarchical classification baselines including (1) hierarchical network (HierNet) [389], (2) HybridNet [166], (3) branch network (BranchNet) [425], (4) label embedding tree (LET) [38], by using them to replace the bidirectional tree LSTM while keeping other modules untouched. Table 6.8 shows the average sensitivity and specificity scores achieved by these methods on the test set. We make the following observations. First, removing tree LSTM largely degrades performance: the sensitivity and specificity of No-TLSTM is 0.23 and 0.28 respectively while our full model (which uses bidirectional TLSTM) achieves 0.29 and 0.33 respectively. The reason is No-TLSTM ignores the hierarchical relationship among codes. Second, the bottom-up tree LSTM alone performs less well than the bidirectional tree LSTM. This demonstrates the necessity of the top-down TLSTM, which ensures every two codes are connected by directed paths and can more expressively capture code-relations in the hierarchy. Third, our method outperforms the four baselines. The possible reason is our method directly builds codes’ hierarchical relationship into their representations while the baselines learn representations and capture hierarchical relationships separately.

Next, we present some qualitative results. For a patient (admission ID 147798) having a DD ‘E Coli urinary tract infection’, without using tree LSTM, two sibling codes 585.2 (chronic kidney disease, stage II (mild)) – which is the groundtruth – and 585.4 (chronic kidney disease, stage
<table>
<thead>
<tr>
<th>Baseline</th>
<th>Sensitivity</th>
<th>Specificity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Larkey and Croft [211]</td>
<td>0.15</td>
<td>0.17</td>
</tr>
<tr>
<td>Franz et al. [112]</td>
<td>0.19</td>
<td>0.21</td>
</tr>
<tr>
<td>Pestian et al. [274]</td>
<td>0.12</td>
<td>0.21</td>
</tr>
<tr>
<td>Kavuluru et al. [191]</td>
<td>0.09</td>
<td>0.11</td>
</tr>
<tr>
<td>Kavuluru et al. [192]</td>
<td>0.21</td>
<td>0.25</td>
</tr>
<tr>
<td>Koopman et al. [199]</td>
<td>0.18</td>
<td>0.20</td>
</tr>
<tr>
<td>LET [38]</td>
<td>0.23</td>
<td>0.29</td>
</tr>
<tr>
<td>HierNet [389]</td>
<td>0.26</td>
<td>0.30</td>
</tr>
<tr>
<td>HybridNet [166]</td>
<td>0.25</td>
<td>0.31</td>
</tr>
<tr>
<td>BranchNet [425]</td>
<td>0.25</td>
<td>0.29</td>
</tr>
<tr>
<td>No-TLSTM</td>
<td>0.23</td>
<td>0.28</td>
</tr>
<tr>
<td>Bottom-up TLSTM</td>
<td>0.27</td>
<td>0.31</td>
</tr>
<tr>
<td>No-AL</td>
<td>0.26</td>
<td>0.31</td>
</tr>
<tr>
<td>No-IC</td>
<td>0.24</td>
<td>0.29</td>
</tr>
<tr>
<td>No-AM</td>
<td>0.27</td>
<td>0.29</td>
</tr>
<tr>
<td>Our full model</td>
<td><strong>0.29</strong></td>
<td><strong>0.33</strong></td>
</tr>
</tbody>
</table>

Table 6.8: Weighted sensitivity and specificity on the test set. On the first panel are baselines for holistic comparison. On the second panel are baselines compared in the ablation study of tree-of-sequences LSTM for capturing hierarchical relationship. On the third panel are baselines compared in the ablation study of adversarial learning for writing-style reconciliation, isotonic constraints for ranking, and attentional matching.

IV (severe)) are simultaneously assigned possibly because their textual descriptions are very similar (only differ in the level of severity). This is incorrect because 585.2 and 585.4 are children of 585 (chronic kidney disease) and the severity level of this disease cannot simultaneously be mild and severe. After the tree LSTM is added, the false prediction of 585.4 is eliminated, which demonstrates the effectiveness of tree LSTM in incorporating one constraint induced by the code hierarchy: among the nodes sharing the same parent, only one should be selected.

For patient 197205, No-TLSTM assigns the following codes: 462 (subacute sclerosing panencephalitis), 790.29 (other abnormal glucose), 799.9 (unspecified viral infection), and 285.21 (anemia in chronic kidney disease). Among these codes, the first three are the groundtruth and the fourth one is incorrect (the groundtruth is 401.9 (unspecified essential hypertension)). Adding tree LSTM fixes this error. The average distance between 401.9 and the rest of groundtruth codes is 6.2. For the incorrectly assigned code 285.21, such a distance is 7.9. This demonstrates that tree LSTM is able to capture another constraint imposed by the hierarchy: codes with smaller tree-distance are more likely to be assigned together.

**Adversarial learning** To evaluate the efficacy of adversarial learning (AL), we remove it from the full model and refer to this baseline as No-AL. Specifically, in Eq. (6.12), the loss term \( \max W_d (−L_{adv}(W_s, W_d)) \) is taken away. Table 6.8 shows the results, from which we observe that after AL is removed, the sensitivity and specificity are dropped from 0.29 and 0.33 to 0.26 and 0.31 respectively. No-AL does not reconcile different writing styles of diagnosis descriptions.
Table 6.9: Comparison of NDCG scores in the ablation study of isotonic constraints.

<table>
<thead>
<tr>
<th>Position</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>No-IC</td>
<td>0.27</td>
<td>0.26</td>
<td>0.23</td>
<td>0.20</td>
</tr>
<tr>
<td>IC</td>
<td>0.32</td>
<td>0.29</td>
<td>0.27</td>
<td>0.23</td>
</tr>
</tbody>
</table>

(DDs) and code descriptions (CDs). As a result, a DD and a CD that have similar semantics may be mismatched because their writing styles are different. For example, a patient (admission ID 147583) has a DD ‘h/o DVT on anticoagulation’, which contains abbreviation DVT (deep vein thrombosis). Due to the presence of this abbreviation, it is difficult to assign a proper code to this DD since the textual descriptions of codes do not contain abbreviations. With adversarial learning, our model can correctly map this DD to a groundtruth code: 443.9 (peripheral vascular disease, unspecified). Without AL, this code is not selected. As another example, a DD ‘coronary artery disease, STEMI, s/p 2 stents placed in RCA’ was given to patient 148532. This DD is written informally and ungrammatically, and contains too much detailed information, e.g., ‘s/p 2 stents placed in RCA’. Such a writing style is quite different from that of CDs. With AL, our model successfully matches this DD to a groundtruth code: 414.01 (coronary atherosclerosis of native coronary artery). On the contrary, No-AL fails to achieve this.

**Isotonic constraint (IC)** To evaluate this ingredient, we remove the ICs from Eq.(6.12) during training and denote this baseline as No-IC. We used NDCG to measure the ranking performance, which is calculated in the following way. Consider a testing patient-visit \( l \) where the groundtruth ICD codes are \( M^{(l)} \). For any code \( c \), we define the relevance score of \( c \) to \( l \) as 0 if \( c \notin M^{(l)} \) and as \( |M^{(l)}| - r(c) \) if otherwise, where \( r(c) \) is the groundtruth rank of \( c \) in \( M^{(l)} \). We rank codes in descending order of their corresponding prediction probabilities and obtain the predicted rank for each code. We calculated the NDCG scores at position 2, 4, 6, 8 based on the relevance scores and predicted ranks, which are shown in Table 6.9. As can be seen, using IC achieves much higher NDCG than No-IC, which demonstrates the effectiveness of IC in capturing the importance order among codes.

We also evaluated how IC affects the sensitivity and specificity of code assignment. As can be seen from Table 6.8, No-IC degrades the two scores from 0.29 and 0.33 to 0.24 and 0.29 respectively, which indicates that IC is helpful in training a model that can more correctly assign codes. This is because IC encourages codes that are highly relevant to the patients to be ranked at top positions, which prevents the selection of irrelevant codes.

**Attentional matching (AM)** In the evaluation of this module, we compared with a baseline – No-AM, which performs an unweighted average of the \( M \) DDs: \( \hat{h}_n = \frac{1}{M} \sum_{m=1}^{M} h_m \), concatenates \( \hat{h}_n \) with \( u_n \), and feeds the concatenated vector into the final prediction layer. From Table 6.8 we can see our full model (with AM) outperforms No-AM, which demonstrates the effectiveness of attentional matching. In determining whether a code should be assigned, different DDs have different importance weights. No-AM ignores such weights, therefore performing less well.

AM can correctly perform the many-to-one mapping from multiple DDs to a CD. For exam-
ple, patient 190236 was given two DDs: ‘renal insufficiency’ and ‘acute renal failure’. AM maps them to a combined ICD code: 403.91 (hypertensive chronic kidney disease, unspecified, with chronic kidney disease stage V or end stage renal disease), which is in the groundtruth provided by medical coders. On the contrary, No-AM fails to assign this code. On the other hand, AM is able to correctly map a DD to multiple CDs. For example, a DD ‘congestive heart failure, diastolic’ was given to patient 140851. AM successfully maps this DD to two codes: (1) 428.0 (congestive heart failure, unspecified); (2) 428.30 (diastolic heart failure, unspecified). Without AM, this DD is mapped only to 428.0.

Holistic comparison with other baselines

In addition to evaluating the four modules individually, we also compared our full model with four other baselines proposed by [112, 191, 192, 199, 211, 274] for ICD coding. Table 6.8 shows the results. As can be seen, our approach achieves much better sensitivity and specificity scores. The reason that our model works better is two-fold. First, our model is based on deep neural networks, which has arguably better modeling power than linear methods used in the baselines. Second, our model is able to capture the hierarchical relationship and importance order among codes, can alleviate the discrepancy in writing styles, and allows flexible many-to-one and one-to-many mappings from DDs to CDs. These merits are not possessed by the baselines.

6.4.3 Related Works

Larkey and Croft [211] studied the automatic assignment of ICD-9 codes to dictated inpatient discharge summaries, using a combination of three classifiers: k-nearest neighbors, relevance feedback, and Bayesian independence classifiers. This method assigns a single code to each patient visit. However, in clinical practice, each patient is usually assigned with multiple codes. Franz et al. [112] investigated the automated coding of German-language free-text diagnosis phrases. This approach performs one-to-one mapping between diagnosis descriptions and ICD codes. This is not in accordance with the coding practice where one-to-many and many-to-one mappings widely exist [267]. Pestian et al. [274] studied the assignment of ICD-9 codes to radiology reports. Kavuluru et al. [191] proposed an unsupervised ensemble approach to automatically perform ICD-9 coding based on textual narratives in electronic health records (EHRs) Kavuluru et al. [192] developed multi-label classification, feature selection, and learning to rank approaches for ICD-9 code assignment of in-patient visits based on EHRs. Koopman et al. [199] explored the automatic ICD-10 classification of cancers from free-text death certificates. These methods did not consider the hierarchical relationship or importance order among codes.

The tree LSTM network was first proposed by [320] to model the constituent or dependency parse trees of sentences. Teng and Zhang [330] extended the unidirectional tree LSTM to a bidirectional one. Xie and Xing [368] proposed a sequence-of-trees LSTM network to model a passage. In this network, a sequential LSTM is used to compose a sequence of tree LSTMs. The tree LSTMs are built on the constituent parse trees of individual sentences and the sequential LSTM is built on the sequence of sentences. Our proposed tree-of-sequences LSTM network differs from the previous works in two-fold. First, it is applied to a code tree to capture the
hierarchical relationship among codes. Second, it uses a tree LSTM to compose a hierarchy of sequential LSTMs.

Adversarial learning [134] has been widely applied to image generation [134], domain adaptation [118], feature learning [104], text generation [401], to name a few. In this paper, we use adversarial learning for mitigating the discrepancy among the writing styles of a pair of sentences.

The attention mechanism was widely used in machine translation [26], image captioning [385], reading comprehension [295], text classification [395], etc. In this work, we compute attention scores between sentences to perform many-to-one and one-to-many mappings.
Chapter 7

Conclusions and Future Directions

In this thesis, we focused on developing diversity-promoting learning methods and large-scale distributed learning systems for automatic smart data-driven medical predictions, recommendations, and decision-making, to assist physicians and hospitals to improve the quality and efficiency of healthcare.

7.1 Contributions

We recapitulate the key contributions.

Diversity-promoting Learning

- In frequentist learning, we proposed a number of diversity-promoting regularizers, including uniform-eigenvalue regularizer (UER), nonconvex and convex Bregman matrix divergence (BMD) regularizers, angular constraints (ACs), and nonoverlap-promoting regularization. We discussed their advantages, disadvantages, and inter-connections. We validated their effectiveness in a wide spectrum of ML models including distance metric learning (DML), long short-term memory network (LSTM), convolutional neural network (CNN), feedforward neural network (FNN), sparse coding (SC), and multiclass logistic regression (MLR). We extended the study of diversity-promoting regularization (DPR) from finite-dimensional vector space to infinite-dimensional reproducing kernel Hilbert space and investigated the nonoverlap-promoting effect of DPR when jointly used with sparsity-promoting regularization.

- In Bayesian learning, we proposed diversity-promoting priors, including the mutual angular process (MAP) and infinite MAP. We validated their effectiveness on two models: Bayesian mixture of experts model (BMEM) and infinite latent feature model (ILFM).

- In theoretical analysis, we formally justified why promoting diversity can better capture infrequent patterns, using nonconvex BMD regularizers as study cases. To understand why DPR can improve generalization performance, we conducted analysis to show that (1) angular constraints can exploit the tradeoff between estimation error and approximation
error, and (2) reducing the nonconvex and convex BMD regularizers can reduce estimation errors.

- In algorithm development, we derived efficient optimization and Bayesian inference methods, based on projected gradient descent, proximal gradient descent, functional gradient descent, coordinate ascent, ADMM, KKT conditions, variational inference, and MCMC sampling, to solve the problems regularized or biased by diversity-promoting regularizers or priors.

**Large-scale Distributed Learning**

- We proposed a sufficient factor broadcasting (SFB) computation model by exploring the sufficient factor property of a large family of ML models. SFB can greatly reduce communication cost. We also proposed a hybrid computation model that leverages the best of SFB and parameter-server architectures.
- We provided theoretical guarantee on the convergence of algorithms executed using SFB. The analysis was extended to partial SFB where each machine communicates with a subset of neighbors.
- Based on SFB, we built a distributed system that provides efficient communication, lightweight fault tolerance, and easy-to-use programming interface. We validated its efficiency and scalability on three large-scale ML models containing billions of parameters.

**Applications in Healthcare**

- In terms of problems, we studied similar patient retrieval, medical topic discovery, medical image tagging, and automated ICD coding.
- In terms of methods, we applied the diversity-promoting and distributed learning techniques for two applications. For the other two applications we proposed tree-of-sequences encoding for hierarchy modeling, adversarial learning for multi-label classification, contextual attention for abnormality localization, adversarial learning for writing-style reconciliation, isotonic constraints for order preservation, etc.
- In terms of evaluation, we validated the effectiveness of our methods on various clinical datasets including electronic health records from ICU, radiology and pathology images, medical literature, to name a few.

**7.2 Conclusions**

We have the following conclusions.

- Promoting diversity can effectively capture infrequent patterns. This is justified both empirically and theoretically. The experiments on (1) DML regularized by UER, nonconvex and convex BMD, (2) BMEM biased by MAP, and (3) ILFM biased by infinite MAP demonstrate that promoting diversity improve the performance on infrequent patterns. In
theory, we proved that decreasing the nonconvex BMD regularizers can reduce an imbalance factor (IF) in DML and a smaller IF indicates better capturing of infrequent patterns.

- Promoting diversity can improve generalization performance, as validated in experiments and theory. In each section of Chapter 2 and 3, the evaluation shows that adding diversity-promoting regularizers or priors improves the performance on test data. In theory, our analysis showed that: (1) a stronger angular constraint yields smaller estimation error and larger approximation error, which hence can explore the tradeoff between these two types of errors and seeks the optimal generalization error; (2) decreasing the nonconvex and convex BMD regularizers can reduce the estimation error.

- Promoting diversity can improve interpretability. It is demonstrated that by encouraging diversity using the LDD-L1 regularizer and the infinite MAP prior, the basis vectors in sparse coding and the latent features in the infinite latent feature model are more mutually distinct and have less overlap, making them easier to interpret.

- Promoting diversity can reduce model size without sacrificing modeling power. In the following experiments: UER-regularized DML, nonconvex/convex BMD regularized DML, BMD-regularized kernel DML, MAP-biased BMEM, infinite MAP biased ILFM, it is shown that with diversity-promoting regularization, an ML model can achieve better performance with fewer components.

- System-algorithm co-design is essential for building efficient distributed ML systems, as evidenced in the Orpheus system. On one hand, we leveraged an algorithmic insight – sufficient factor property – to drive the design of efficient communication and light-weight fault tolerance. On the other hand, we designed new algorithms (e.g., joint matrix column subset selection) to support the realization of system designs (e.g., sufficient factor selection). Empirical evaluation on various large-scale applications demonstrate the benefits of system-algorithm co-design.

- When applied to healthcare applications, the proposed diversity-promoting and distributed learning techniques demonstrate great effectiveness in solving the problems mentioned in the introduction chapter.

- The additionally proposed ML techniques for healthcare applications are effective, as demonstrated in extensive evaluations. The tree-of-sequences LSTM model can effectively capture the hierarchical relationship among classes. Adversarial learning is successful applied to capture correlations among classes and reconcile the discrepancy between writing styles. Contextual attention shows great efficacy in identifying abnormal regions in medical images. Isotonic constraints effectively improve ranking performance.

### 7.3 Future Directions

We identify the following directions for future work.
Diversity-promoting Learning

- **Promoting structured diversity** The diversity-promoting regularizers/priors developed in this thesis do not consider the structural relationships among components. In many ML models, the components admit a certain structure, such as multiview, grouping, and hierarchy. For example, in cross-modal distance metric learning [283], each feature modality has a separate set of components. In clustered multi-task learning [175], the tasks (components) are partitioned into latent groups. In hierarchical topic models [144], the topics (components) are organized into a tree. In the presence of such structures, how to promote diversity deserves investigation.

- **Provably consistent parameter estimation** Most diversity-promoting regularizers developed in this thesis are nonconvex, rendering the regularized problems to be nonconvex and therefore vulnerable to local optima. Further, they are difficult, if possible, to be convexified. It is intriguing to study whether the global optima of these nonconvex problems can be achieved. We plan to investigate methods of moments and spectral algorithms [168], which were demonstrated to be computationally efficient and provably consistent estimators in a number of nonconvex problems [20, 66, 269].

- **Posterior contraction analysis** In Bayesian models, it is interesting to analyze how diversity-promoting priors affect the contraction rate of components’ posterior. Inspired by [262], we plan to study how fast the convex hull \( G \) of the estimated components contracts to the convex hull \( G_0 \) of the “true” components, where the “closeness” between \( G \) and \( G_0 \) can be measured by the “minimum-matching” Euclidean distance or Hausdorff metric. In the mutual angular process, the concentration parameter \( \kappa \) represents the level of diversity. Our goal is to establish a relationship between \( \kappa \) and the contraction rate.

- **Information geometry analysis** In probabilistic models, there are two types of manifolds. First, these models essentially define distributions over the observed data and these distributions have a manifold structure [215] which is studied by information geometry [19]. Second, the components in ML models also lie in a manifold. We are interested in how diversity-promoting regularizers/priors affect the component manifold and further how they affect the distribution manifold (indirectly via the component manifold).

Large-scale Distributed Learning

- **Model parallelism** In this thesis, we assume the parameter matrix (PM) can fit into the memory of each machine, which may not hold when the PM is excessively large. One way to address this issue is to divide the PM into sub-matrices so that each of which can fit into the memory of one machine. Multiple machines collaboratively update a single PM via model parallelism. On the other hand, we retain the parallelism on data by adopting a hierarchical architecture. The machines are divided into groups. Each group holds one replica of the PM and one data shard. Machines within the same group hold different sub-matrices of the PM replica. Data parallelism is executed between machine groups and model parallelism is conducted within each group.
Applications in Healthcare

- **Never-ending construction of medical knowledge graph** Medical knowledge graph (MKG) which consists of medical entities (e.g., symptoms, diseases, medications) and their relationships (e.g., medication A can treat disease B, medication A and C have adverse interactions) is a fundamental asset that empowers many clinical applications, such as medical named entity recognition, symptom checking, medication-prescription safety alert, to name a few. Manually building an MKG is time-consuming and not scalable. There have been efforts to automatically construct MKGs by analyzing literature and clinical notes [292]. However, they lack the mechanism to continuously grow the MKGs from medical texts that are newly generated every day. We plan to build a reinforcement learning system that constantly digests the new medical texts and ceaselessly expands and refines the MKG.

- **Discharge summary generation** When discharging patients, physicians need to collect a large amount of information accumulated during inpatient stay (e.g., hospital course, problem list, medications, test results) and consolidate them into a discharge summary. This process is very time-consuming. It would be helpful to build an ML software that automatically pulls data from different sources, extracts key information from the raw data, and summarizes these information into a highly precise and readable discharge note.
Bibliography

[1] 20-Newsgroups dataset. http://www.daviddlewis.com/resources/testcollections/reuters21578/ 2.2.4 2.5.3 3.2.4

[2] GPUDirect RDMA. http://docs.nvidia.com/cuda/gpudirect-rdma/ #axzz4eWBuf9Rj 5.2.4


[5] CIFAR-10 dataset. https://www.cs.toronto.edu/~kriz/cifar.html 2.3.3 2.5.3 5.3.2 5.3.2


[7] MAGMA matrix algebra on GPU. In http://icl.cs.utk.edu/magma/ 2.1.3 2.2.3 2.2.4

[8] MPI. https://www.mpich.org/ 5.2.4


[10] Reuters dataset. http://www.daviddlewis.com/resources/testcollections/reuters21578/ 2.2.4 2.5.3 3.2.4


Andrew R Barron. Universal approximation bounds for superpositions of a sigmoidal


[49] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 2003. 1.1 1.3 2.5 5 5.2 5

[50] Guillaume Bouchard. Efficient bounds for the softmax function, applications to inference in hybrid models. 2007. 3.1.2 3.1.2 3.1.6


[53] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004. 2.1.3 2.1.3 2.2.2 2.3.2 2.3.2 2.3.2


[58] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 1998. 3.1.5 3.1.5

[59] Simon Byrne and Mark Girolami. Geodesic monte carlo on embedded manifolds. *Scandinavian Journal of Statistics*, 2013. 3.2.3 3.2.3 3.2.4

[60] Deng Cai and Xiaofei He. Manifold adaptive experimental design for text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 2012. 2.5.3

[61] Deng Cai, Xiaofei He, and Jiawei Han. Document clustering using locality preserving indexing. *IEEE Transactions on Knowledge and Data Engineering*, 2005. 3.2.4

[62] Emmanuel Candes and Benjamin Recht. Exact matrix completion via convex optimization. *Communications of the ACM*, 2012. 2.1.1 2.2.2

[63] Miguel A Carreira-Perpiñan and Ramin Raziperchikolaei. An ensemble diversity approach to supervised binary hashing. In *Advances in Neural Information Processing Systems*, 2016. 2.2.4 2.2.4 2.2.4 2.2.4 2.2.4 2.2.4 2.2.4 6.1.1 6.1.2

[64] Ricardo Cerri, Rodrigo C Barros, and André CPLF De Carvalho. Hierarchical multi-label


[70] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. *International Conference on Learning Representations*, 2014. [6.3.3]

[71] Liang-Chieh Chen, Yi Yang, Jiang Wang, Wei Xu, and Alan L Yuille. Attention to scale: Scale-aware semantic image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3640–3649, 2016. [6.3.2] [6.3.2] [6.3.3]


[73] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naikwan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *NIPS 2015 Workshop*, 2015. [1.3.2] [5.2.5] [5.2.5] [5.3.2] [5.3.2]

[74] Yong Chen, Hui Zhang, Yongxin Tong, and Ming Lu. Diversity regularized latent semantic match for hashing. *Neurocomputing*, 2017. [2.2] [2.2.4] [2.2.4] [2.2.4] [2.2.4] [2.2.4] [6.1.1] [6.1.2]

[75] Yukun Chen, Hongxin Cao, Qiaozhu Mei, Kai Zheng, and Hua Xu. Applying active learning to supervised word sense disambiguation in MEDLINE. *Journal of the American Medical Informatics Association*, 2013. [1.3.3] [1.3.3]


[92] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, 2007. 2.1.2, 2.1.4, 2.2.2, 2.2.4, 2.4.3, 2.5.1, 2.2.4, 2.2.4, 2.2.4, 2.2.4, 2.4.3, 6.1.1, 6.1.2


[95] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009. 2.1.4, 2.4.3, 5, 5.3.2


[100] Inderjit S Dhillon and Joel A Tropp. Matrix nearness problems with bregman divergences. *SIAM Journal on Matrix Analysis and Applications*, 29(4):1120–1146, 2007. 1.2.1, 2.2.1, 2.4


[103] Son Doan, Ko-Wei Lin, Mike Conway, Lucila Ohno-Machado, Alex Hsieh,


[110] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer vision and Image understanding*, 2007. 3.2.4


[114] Xiping Fu, Brendan McCane, Steven Mills, and Michael Albert. Nokmeans: Non-orthogonal k-means hashing. In *Asian Conference on Computer Vision*, 2014. 2.2.4, 6.1.1


<table>
<thead>
<tr>
<th>Reference</th>
<th>Authors</th>
<th>Title</th>
<th>Journal/Conference</th>
<th>Year</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>[131]</td>
<td>Ary L Goldberger et al.</td>
<td>Physiobank, physiotoolkit, and physionet.</td>
<td>Circulation</td>
<td>2000</td>
<td>2.2.4, 6.1.1</td>
</tr>
<tr>
<td>[134]</td>
<td>Ian Goodfellow et al.</td>
<td>Generative adversarial nets.</td>
<td>Advances in Neural Information Processing Systems</td>
<td>2014</td>
<td>1.2.3, 6.3, 6.3.1, 6.4.1</td>
</tr>
<tr>
<td>[136]</td>
<td>Rudolf Gorenflo et al.</td>
<td>Analytical properties and applications of the Wright function.</td>
<td>Fractional Calculus and Applied Analysis</td>
<td>1999</td>
<td>2.2.3, 2.4.2, 2.4.4</td>
</tr>
<tr>
<td>[139]</td>
<td>Alex Graves et al.</td>
<td>Speech recognition with deep recurrent neural networks.</td>
<td>IEEE International Conference on Acoustics, Speech and Signal Processing</td>
<td>2013</td>
<td>2.3.3, 2.3.3</td>
</tr>
<tr>
<td>[141]</td>
<td>Gregory Griffin et al.</td>
<td>Caltech-256 object category dataset.</td>
<td>Caltech-256 object category dataset</td>
<td>2007</td>
<td>2.3.3, 2.4.3</td>
</tr>
<tr>
<td>[142]</td>
<td>Thomas Griffiths and Zoubin Ghahramani</td>
<td>Infinite latent feature models and the Indian buffet process.</td>
<td>Advances in Neural Information Processing Systems</td>
<td>2005</td>
<td>3.2, 3.2.2, 3.2.3, 3.2.3, 3.2.4, 3.2.4, 3.2.4, 3.2.4</td>
</tr>
<tr>
<td>[143]</td>
<td>Thomas L Griffiths and Zoubin Ghahramani</td>
<td>The Indian buffet process: An introduction and review.</td>
<td>Journal of Machine Learning Research</td>
<td>2011</td>
<td>1.2.7, 3.2, 3.2.2, 3.2.3</td>
</tr>
</tbody>
</table>

222

[146] Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. Is that you? metric learning approaches for face identification. In IEEE International Conference on Computer Vision. IEEE, 2009. 2.1.2, 2.1.4, 2.1.4, 2.2.4, 2.4.1, 2.4.3, 2.5.1, 2.2.4, 2.2.4, 2.2.4, 2.4.3, 6.1.1, 6.1.2


[149] Yoni Halpern, Steven Horng, Youngduck Choi, and David Sontag. Electronic medical record phenotyping using the anchor and learn framework. Journal of the American Medical Informatics Association, 2016. 1.3.3


[154] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In Advances in Neural Information Processing Systems, 2015. 2.1.2, 2.1.4, 2.3.3, 2.3.3, 2.3.3


223


[163] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 1.2.1, 1.3.3, 2.1, 2.1.2, 2.1.2, 2.3.3, 2.5.1, 2.5.3, 5.2.5, 6.3, 6.3.1


224


[176] P. Jain, B. Kulis, J. Davis, and I. Dhillon. Metric and kernel learning using a linear transformation. *Journal of machine Learning research*, 2012. 2.4, 2.4.1, 2.4.3, 2.4.3

[177] Amin Jalali, Lin Xiao, and Maryam Fazel. Variational Gram functions: Convex analysis and optimization. *Siam Journal on Optimization*, 2017. 1.3.1, 2.2.4, 2.2.4, 2.2.4, 2.2.4, 6.1.1, 6.1.2


[185] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002. 2.1, 2.1.1


[187] Shalmali Joshi, Suriya Gunasekar, David Sontag, and Joydeep Ghosh. Identifiable pheno-


[230] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen AWM van der Laak, Bram van Ginneken,
and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 2017. 6.3.3


[233] Wei Liu, Steven Hoi, and Jianzhuang Liu. Output regularized metric learning with side information. *European conference on computer vision*, 2008. 2.2.4, 2.2.4, 2.2.4, 2.2.4, 6.1.1, 6.1.2

[234] Wei Liu, Cun Mu, Rongrong Ji, Shiqian Ma, John R Smith, and Shih-Fu Chang. Low-rank similarity metric learning in high dimensions. In *AAAI Conference on Artificial Intelligence*, 2015. 2.2.4, 2.2.4, 2.2.4, 2.2.4, 2.2.4, 6.1.1, 6.1.2


[242] Jonathan Malkin and Jeff Bilmes. Ratio semi-definite classifiers. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4113–4116. IEEE, 2008. 1.3.1, 1.3.1, 2.1, 2.1.1, 2.1.4, 2.1.4, 2.1.4, 2.1.4, 2.1.4, 2.1.4, 2.1.4, 2.2.4, 2.3.3, 2.2.4, 2.2.4, 2.2.4, 6.1.1, 6.1.2

[243] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to information retrieval*, volume 1. 2008. 2.2.4, 6.1.1


[249] Aleix M Martín and Avinash C Kak. Pca versus lda. IEEE transactions on pattern analysis and machine intelligence, 23(2):228–233, 2001. 3.2.4


[252] A. Mignon and F. Jurie. PCCA: A new approach for distance learning from sparse pairwise constraints. In IEEE Conference on Computer Vision and Pattern Recognition, 2012. 2.4.3 2.4.3


[257] Dmytro Mishkin and Jiri Matas. All you need is a good init. International Conference on Learning Representations, 2015. 2.3.3 2.5.3


[259] Taesup Moon, Heeyoul Choi, Hoshik Lee, and Inchul Song. Rndrop: A novel dropout for RNNs in ASR. In IEEE Automatic Speech Recognition and Understanding Workshop, 2015. 2.3.3


[264] Michael A Nielsen and Isaac L Chuang. *Quantum computation and Quantum information*. [2.2.2]


[266] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 1997. [1.2.1] [2.3.1] [2.4.3] [2.5.3] [5.1]


[282] Qi Qian, Juhua Hu, Rong Jin, Jian Pei, and Shenghuo Zhu. Distance metric learning using dropout: a structured regularization approach. In *ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014. \[2.1.4\] [2.1.4] [2.1.4] [2.1.4] [2.2.4] [2.2.4] [2.2.4] [2.2.4]


[288] Narges Razavian, Jake Marcus, and David Sontag. Multi-task prediction of disease onsets from longitudinal lab tests. *Machine Learning for Healthcare Conference*, 2016. \[1.3.3\]

[289] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for


[291] Pau Rodríguez, Jordi Gonzàlez, Guillem Cucurull, Josep M Gonfaus, and Xavier Roca. Regularizing cnns with locally constrained decorrelations. *International Conference on Learning Representations*, 2017. 2.3.3 2.5.3 2.3.3 2.3.3


[293] B. Schölkopf, A. Smola, and K. Müller. Kernel principal component analysis. In *International Conference on Artificial Neural Networks*, 1997. 2.4 2.4.1

[294] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press, 2002. 2.2.4 2.4 2.4.2 2.4.2 2.4.3 2.4.3 2.4.3


[296] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *International Conference on Learning Representations*, 2017. 2.1.4 2.1.4 2.1.4 2.3.3 2.3.3 2.3.3


[298] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014. 12 4.3.2 4.3.2


[301] Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. Reasonet: Learning to stop reading in machine comprehension. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017. 2.1.4 2.3.3


[304] Vikas Sindhwani and Amol Ghoting. Large-scale distributed non-negative sparse coding and sparse dictionary learning. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2013. 6.3.3

233


Sayantan Sur, Uday Kumar Reddy Bondhugula, Amith Mamidala, H-W Jin, and Dhabaleswar K Panda. High performance RDMA based all-to-all broadcast for InfiniBand
clusters. In *International Conference on High-Performance Computing*, 2005. 5.2.4


[323] Buzhou Tang, Hongxin Cao, Yonghui Wu, Min Jiang, and Hua Xu. Recognizing clinical entities in hospital discharge summaries using structural support vector machines with word representation features. *BMC medical informatics and decision making*, 2013. 1.3.3

[324] Buzhou Tang, Yonghui Wu, Min Jiang, Yukun Chen, Joshua C Denny, and Hua Xu. A hybrid system for temporal information extraction from clinical text. *Journal of the American Medical Informatics Association*, 2013. 1.3.3


[327] Yee W Teh and Dilan Gorur. Indian buffet processes with power-law behavior. In *Advances in neural information processing systems*, 2009. 3.2.4, 3.2.4

[328] Yee Whye Teh and Zoubin Ghahramani. Stick-breaking construction for the Indian buffet process. 2007. 1.2.1

[329] Yee Whye Teh and Zoubin Ghahramani. Stick-breaking construction for the Indian buffet process. 2007. 3.2, 3.2.2, 3.2.3, 3.2.4


Darren Wilkinson. Metropolis hastings mcmc when the proposal and target have differing
support. In https://darrenjw.wordpress.com/2012/06/04/metropolis-hastings-mcmc-when-the-proposal-and-target-have-differing-support/, 2015. 3.1.6


[365] Bo Xie, Yingyu Liang, and Le Song. Diversity leads to generalization in neural networks. International Conference on Artificial Intelligence and Statistics, 2017. 1.3.1 1.3.1


[367] Pengtao Xie. Learning compact and effective distance metrics with diversity regularization. In European Conference on Machine Learning, 2015. 1.1 2.1.2 2.1.4 2.1.4 2.1.4 2.1.4 2.1.4 2.2 2.2.4 2.2.4 2.2.4 2.2.4 2.2.4 6.1.1 6.1.2


[369] Pengtao Xie, Yuntian Deng, and Eric Xing. Diversifying restricted boltzmann machine for document modeling. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 1315–1324. ACM, 2015. 1.1 1.3.1 1.3.1 2.1 2.1.1 2.1.4 2.1.4 2.2.4 2.2.4 2.2.4 2.3.3 2.4.3 2.5.3 2.3.3 2.3.3 2.3.3 2.3.3 2.4.3 2.4.3 2.4.3 2.5.3 3.1 3.1.1 3.1.3 3.1.3 3.1.3 3.1.5 3.1.5 3.1.5 3.1.6 6.1.1 6.1.2

[370] Pengtao Xie, Jin Kyu Kim, Yi Zhou, Qirong Ho, Abhimanu Kumar, Yaoliang Yu, and Eric Xing. Distributed machine learning via sufficient factor broadcasting. In Conference on Uncertainty in artificial intelligence, 2016. 1.1 1.2.2 1.3.2 5


[372] Pengtao Xie, Yuntian Deng, Yi Zhou, Abhimanu Kumar, Yaoliang Yu, James Zou, and Eric P. Xing. Learning latent space models with angular constraints. In Proceedings of the 34th International Conference on Machine Learning, pages 3799–3810, 2017. 1.1 1.2.1 1.2.1 2 2.3 4.2.1


238
[374] Pengtao Xie, Barnabas Poczos, and Eric P Xing. Near-orthogonality regularization in kernel methods. 2017. 1.2.1, 1.2.1, 2.4


[378] Pengtao Xie, Haoran Shi, Ming Zhang, and Eric P Xing. A neural architecture for automated icd coding. The 56th Annual Meeting of the Association for Computational Linguistics, 2018. 1.1, 1.2.3

[379] Pengtao Xie, Wei Wu, Yichen Zhu, and Eric P Xing. Orthogonality-promoting distance metric learning: Convex relaxation and theoretical analysis. International Conference on Machine Learning, 2018. 1.1, 1.2.1, 1.2.1, 1.2.3, 2, 2.2, 4.1, 4.2.2, 4.2.3


[382] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. IEEE Conference on Computer Vision and Pattern Recognition, 2017. 2.3.3, 2.5.3

[383] Eric P Xing, Michael I Jordan, Stuart Russell, and Andrew Y Ng. Distance metric learning with application to clustering with side-information. In Advances in neural information processing systems, 2002. 1.2.1, 2.1, 2.1.2, 2.1.2, 2.2, 2.2.3, 2.5.1, 5, 5.1, 6.1

[384] Eric P Xing, Qirong Ho, Wei Dai, Jin-Kyu Kim, Jinliang Wei, Seunghak Lee, Xun Zheng, Pengtao Xie, Abhimanu Kumar, and Yaoliang Yu. Petuum: A new platform for distributed machine learning on big data. In ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015. 1.1

[385] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In International Conference on Machine Learning, 2015. 6.3.2, 6.3.2, 6.3.3, 6.4.3

[386] Yanxun Xu, Peter Müller, and Donatello Telesca. Bayesian inference for latent biologic structure with determinantal point processes (dpp). Biometrics, 72(3):955–964, 2016. 3.2


[429] James Y Zou and Ryan P Adams. Priors for diversity in generative latent variable models. In Advances in Neural Information Processing Systems, 2012. 1.3.1 1.3.1 2.1 2.1.1 2.1.4 2.1.4 2.1.4 2.1.4 2.1.4 2.2.4 2.2.4 2.2.4 2.2.4 2.2.4 2.2.4 2.2.4 2.2.4 2.4.3 2.4.3 3.1.5 3.1.5 3.1.5 6.1.1 6.1.2

[430] Alon Zweig and Daphna Weinshall. Exploiting object hierarchy: Combining models from different category levels. In The IEEE International Conference on Computer Vision,