On Learning from Collective Data

Liang Xiong

December, 2013
CMU-ML-13-113

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

Thesis Committee:
Jeff Schneider, Chair
Aarti Singh
Eric Xing
Arthur Gretton, University College London

*Submitted in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy*

This research was sponsored by the Department of Energy under grant number DESC0002607; the Air Force Research Laboratory under grant numbers FA865010C7059 and FA87501220324; the National Science Foundation under grant number IIS0911032; the Association of Universities for Research in Astronomy, Inc. under award number C10625A; and a grant from ECCO.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Collective data; grouped data; point sets; low-rank decomposition; robust methods; anomaly detection; novelty detection; group anomaly; hierarchical probabilistic models; topic models; divergence estimation; distribution classification; efficient learning; distance completion; embedding; scientific data analysis.

To my beloved parents and dearest wife.

Abstract

In many machine learning problems and application domains, the data are naturally organized by groups. For example, a video sequence is a group of images, an image is a group of patches, a document is a group of paragraphs/words, and a community is a group of people. We call them the *collective data*.

In this thesis, we study how and what we can learn from collective data. Usually, machine learning focuses on individual objects, each of which is described by a feature vector and studied as a point in some metric space. When approaching collective data, researchers often reduce the groups into vectors to which traditional methods can be applied. We, on the other hand, will try to develop machine learning methods that respect the collective nature of data and learn from them directly.

Several different approaches were taken to address this learning problem. When the groups consist of unordered discrete data points, it can naturally be characterized by its sufficient statistics – the histogram. For this case we develop efficient methods to address the outliers and temporal effects in the data based on matrix and tensor factorization methods.

To learn from groups that contain multi-dimensional real-valued vectors, we develop both generative methods based on hierarchical probabilistic models and discriminative methods using group kernels based on new divergence estimators. With these tools, we can accomplish various tasks such as classification, regression, clustering, anomaly detection, and dimensionality reduction on collective data.

We further consider the practical side of the divergence based algorithms. To reduce their time and space requirements, we evaluate and find methods that can effectively reduce the size of the groups with little impact on the accuracy. We also proposed the conditional divergence along with an efficient estimator in order to correct the sampling biases that might be present in the data. Finally, we develop methods to learn in cases where some divergences are missing, caused by either insufficient computational resources or extreme sampling biases.

In addition to designing new learning methods, we will use them to help the scientific discovery process. In our collaboration with astronomers and physicists, we see that the new techniques can indeed help scientists make the best of data.

Acknowledgments

First and foremost, this thesis would not have been possible without the guidance and support from my advisor Jeff Scheinder. In addition to his academic advices, Jeff's constant encouragement and humor have made my life as a PhD student as enjoyable as I can imagine.

I am indebted to Aarti Singh, Eric Xing, and Arthur Gretton for serving on my thesis committee and giving me helpful suggestions. I would like to thank Barnabás Póczos for the insightful discussions and the help over the years. I also want to thank my external collaborators: Andrew Connolly, Jake Vanderplas, Scott Daniel, and Lauren Anderson from University of Washington; Alex Szalay and Charles Meneveau from Johns Hopinks University.

Life at the Machine Learning Department and the AutonLab has been wonderful. I want to show my deepest appreciation for my friends and those who have taught and helped me: Tom Mitchell, Geoff Gordon, Carlos Guestrin, Christos Faloutsos, Larry Wasserman, John Lafferty, Noah Smith, Ann Lee, Roni Rosenfeld, Artur Dubrawski, Daniel Neill, Diane Stidle, Michelle Martin, Karen Widmaier, Michael Baysek, Daria Sorokina, Madalina Fiterau, Roman Garnett, Tzu-Kuo Huang, Xuezhi Wang, Robin Sabhnani, Dougal Sutherland, Junier Oliva, Donghan Wang, Karen Chen, Yifei Ma, Brendan O'connor, Kumar Avinava Dubey, Xi Chen, Han Liu, Guang Xiang, Jieyue Li, Min Xu, Yang Xu, Robert Fisher, Haijie Gu, Guangyu Xia, Bin Zhao, Chong Wang, Jun Zhu, and more.

I thank my internship mentors and colleagues at Google and Yahoo! Labs: Max Ishutin, Ari Wilson, Youssef Billawala, Sudarshan Lamkhede, Jie Luo, and Yi Chang.

Most of all, I thank my parents and my wife for everything they have done for me. Without their love and care I would not have become who I am today.

Contents

1 Introduction **1**

1.1 Notations . . . 3

1.2 Learning from Discrete Data . . . 5

1.3 Learning from Continuous, Multidimensional Data . . . 7

1.4 Related Fields . . . 11

1.5 Challenges in Scientific Data . . . 12

1.6 Thesis Overview . . . 13

I Learning from Discrete Data **15**

2 Modeling Temporal Effects by Tensor Factorizations **16**

2.1 Introduction . . . 16

2.2 Preliminaries . . . 18

2.3 A Tensor Model for Temporal Data . . . 19

2.4 A Bayesian Treatment . . . 22

2.5 Related Work . . . 26

2.6 Experiments . . . 27

2.7 Summary . . . 32

3 Handling Outliers by Robust Factorization **37**

3.1 Introduction . . . 37

3.2 Direct Robust Factorization . . . 40

3.3 Related Work . . . 43

3.4 Discussion . . . 44

3.5 Experiments . . . 46

3.6 Summary . . . 52

3.7 Automatic Novelty Discovery for Astronomy . . . 52

II Learning from Multidimensional Data **58**

4 Generative Models for Collective Data **59**

4.1 Introduction 59
4.2 Background 61
4.3 Related Work 62
4.4 Multinomial Genre Models 63
4.5 Flexible Genre Models 67
4.6 Nonparametric Genre models 71
4.7 Discussion 75
4.8 Experiments 76
4.9 Summary 82

5 Discriminative Methods for Collective Data 86
5.1 Introduction 86
5.2 Related Work 87
5.3 Problem Definition 89
5.4 Nonparametric Kernel Estimation 89
5.5 Constructing Mercer Kernels 90
5.6 Experiments 91
5.7 Summary 98

6 Low-Rank Constructions of Mercer Kernels 100
6.1 Introduction 100
6.2 Related Work 101
6.3 Constructing Low-Rank Kernels 102
6.4 Constructing Low-Rank Divergences 103
6.5 Discussion 104
6.6 Experiments 105
6.7 Summary 108

7 Accelerated Learning by Condensing 111
7.1 Introduction 111
7.2 Background 112
7.3 Related Work 116
7.4 Condensing Methods 117
7.5 Empirical Evaluation 119
7.6 Discussion 127
7.7 Summary 128

8 Sampling Bias Correction by Conditional Divergences 129
8.1 Introduction 129
8.2 Background and Related Work 131
8.3 Conditional Divergences 132
8.4 Choosing $c(x)$ 134
8.5 Discussion 135

8.6 Experiments 136
8.7 Summary 140

9 Conclusion and Future Directions 142

Bibliography 144

List of Figures

1.1 A galaxy observation from SDSS. 12

1.2 Local motion patterns in the JTDC simulation. Several 2D slices are presented. . . 13

2.1 CP decomposition of a three-way tensor $\mathbf{X}$ 21

2.2 The graphical model for BPTF 23

2.3 Performance comparison of PMF, BPMF, and BPTF on 6 seasons of ECCO's sales data. BPTF outperforms others by a large margin. See text for details. 28

2.4 Convergence curves of BPTF and BPMF on the full Netflix data. As the number of samples increase, the RMSE of Bayesian methods drop monotonically. The RMSE of the Netflix's baseline and PMF are also presented. 30

2.5 Convergence curves of BPMF and BPTF with different number of factors on a subset of Netflix data. The accuracy increases when more factors are used, and no over-fitting is observed. Also, BPTF with 20 factors achieves similar performance as BPMF with 100 factors. 31

2.6 RMSE of PMF, BPMF, and BPTF. (a) On a subset of Netflix data. (b) On MovieLens data. Lower RMSE is better. 32

2.7 Prediction RMSE on the Yahoo Music data set using different methods and different number of latent factors. 33

3.1 An example where DRMF with initial $\mathbf{S} = \mathbf{0}$ would fail. Blue crosses are normal points and the red circle is the outlier. Blue arrow shows the true principle subspace and the red dashed arrow shows the wrong one DRMF would get starting from $\mathbf{S} = \mathbf{0}$. Note that when starting from an $\mathbf{S}$ that correctly indicates the circle as an outlier, DRMF is able to achieve the correct blue subspace. 46

3.2 Performances on noiseless data with entry outliers. Note that the running time is shown in log-scale. 48

3.3 Performances on noisy data with entry outliers. 48

3.4 Performances on noisy data with row outliers. 49

3.5 (a) Recovery RMSE of RPCA and DRMF versus the outliers' magnitude. (b) Recovery RMSE of DRMF versus the parameters K the rank and e the proportion of allowed outliers. Darker color indicates smaller error. 50

3.6 Video activity detection performance 51

3.7 USPS anomaly detection results. (a) the average precisions of detecting '7's among '1's. (b) images ranked by their anomaly scores in the descending order. 52

3.8 Video activity detection result frames. In each sub-figure, the images from left to right are: the original frame, background and foreground from DRMF, background and foreground from RPCA. 56
3.9 The frontpage of the SDSS collaborative SDSS website. 57
3.10 The UI for finding similar objects in the SDSS collaborative SDSS website. 57

4.1 Graphical model for latent Dirichlet allocation (LDA). 61
4.2 The Multinomial Genre Model (MGM). 64
4.3 The Flexible Genre Model (FGM). 68
4.4 The nonparametric genre model (NGM). 72
4.5 Detection results on the synthetic data. Black boxes are normal groups. Green dashed boxes are point-based anomalies. Red dashed boxes are distribution-based anomalies. The method postfix "-D" means distribution-based scores, and "-P" means point-based scores. 78
4.6 (a),(b),(c) show the density of the point-based anomaly estimated by LDA, MG-M, and FGM respectively. In LDA and MGM, topics must be shared globally, therefore their perform badly. (d) The genres in the synthetic data set learned by FGM. 79
4.7 Performances on detecting out-of-category images. See text for details. 80
4.8 Images samples. Green boxes (first row) contain natural images, and yellow boxes (second row) contain stitched anomalies. 81
4.9 Performances on detecting stitched images. 81
4.10 Detection results for the turbulence data. (a) & (b) FGM-DB anomaly score and vorticity visualized on one slice of the cube. (c) Correlations of the anomaly scores with the vorticity. 84

5.1 Densities of the two one-dimensional mixtures. 93
5.2 1D mixture classification accuracies. 94
5.3 Mean and standard deviation accuracies on the high-dimensional artificial data set. 95
5.4 Images of two objects from the ETH-80 data set. Each object has 5 different views. 96
5.5 Classification accuracies on ETH-80. 96
5.6 Classification accuracies on ETH-80 with Rényi-α for twenty α's, as well as the Hellinger distance. 97
5.7 Images from the 8 OT scene categories: *coast, forest, highway, inside city, mountain, open country, street, tall building*. 98
5.8 Accuracies on the OT data set. The horizontal line shows the best previously reported result. 99
5.9 Images from the 8 sports. 99
5.10 Accuracies on the Sport data set. The horizontal line shows the best previously reported result. 99

6.1 Recovery performances on the toy data set. LRDC-D and LRDC-K are for the distance matrix and the kernel matrix obtained by LRDC respectively. LRKC-K is for the kernel matrix obtained by LRKC. The X axis shows different values of the parameter r in LRDC. The correlations of the raw distance matrix $\mathbf{D}$ and kernel matrix $\mathbf{K}$ are also shown. 106
6.2 Example results from LRKC and LRDC. $\mathbf{K}^\star$ and $\mathbf{D}^\star$ are the groundtruths. $\mathbf{K}$ and $\mathbf{D}$ are the noisy observations. $\widehat{\mathbf{K}}$ and $\widehat{\mathbf{D}}$ are the low-rank results produced by LRKC and LRDC. 107
6.3 Recovery performances on the toy data set with missing data. LRDC-D and LRDC-K are for the distance matrix and the kernel matrix obtained by LRDC respectively. LRKC-K is for the kernel matrix obtained by LRKC. The X axis shows different values of the parameter r in LRDC. The correlations of the raw distance matrix $\mathbf{D}$ and kernel matrix $\mathbf{K}$ are also shown. 108
6.4 Classification accuracy on the ETH-80 data set. D is the baseline (green dashed line) using PSD projection. The “-I” postfix means incomplete data. LRKC with different λ's and LRDC with different r's are shown. 109
6.5 Classification accuracy on the Sports data set. 110
6.6 Classification accuracy on the OT data set. D is the baseline (green dashed line) using PSD projection. The “-I” postfix means incomplete data. LRKC with different λ's and LRDC with different r's are shown. 110

7.1 Condensing results of a 2D standard Gaussian point set using different condensers. 117
7.2 Accuracies on the OT data set using the original sets and the condensed sets. Green dashed lines are the accuracies of bag-of-words classifiers. 121
7.3 Performance of LNBNN on Scene-15 using the uniform covering condenser. 122
7.4 Scene-15 classification performances using different classifiers and condensers. . . 123
7.5 The impact of the number of checks in the NN search to different methods on the Scene-15 data set. 124
7.6 UIUC-Sports classification performances using different classifiers and condensers. 125
7.7 (a) CalTech-101 classification accuracies using LNBNN with different condensers. (b) CalTech-256 classification accuracies using LNBNN with different condensers. 126

8.1 The observation biases. 130
8.2 Estimated divergences on the synthetic data. 137
8.3 Divergences on the uneven sets. The goal is to recover the “Full D” given only the biased sets. 138
8.4 Divergences on the partial sets. The goal is to recover the “Full D” result shown in Figure 8.3. 138
8.5 Example temperature maps of the U.S. from the QCLCD. (a) and (c) are the original data. (b) and (d) are the artificially created uneven data. 139
8.6 Season classification results on the QCLCD weather data. 140
8.7 Image classification results on OT. 141

List of Tables

1.1 Symbols 4

2.1 RMSE of PMF, BPMF and BPTF on Netflix data. 29

3.1 Comparing the nuclear norm minimization (NNM) problem and DRMF. $\mathbf{L}$ is low-rank; $\mathbf{S}$ is the sparse outlier. $\|\cdot\|_*$ is the nuclear norm; σ is the allowed approximation error. 44

3.2 AP of detecting SIMBAD objects using the normalized spectrum feature. 55

7.1 Accuracies and running time of LNBNN on ImageNet. The training time is measured by CPU*minute per class and the testing time is measured by CPU*second per test image. 127

Chapter 1

Introduction

The current machine learning paradigms mostly focus on individual objects that have simple representations. For example in the tasks of classification, regression, and clustering, an object of interest is often described by a "feature vector", and abstracted as a point in certain metric space. The objective of learning is to estimate functions that map these points to target variables such as the class labels or cluster memberships, with the goal of achieving both empirical accuracies on the training data as well as the generalization power on unseen data. This "one point per object" abstraction has led to very concise representations, elegant mathematical theories, and very successful algorithms.

Nevertheless, we also realize that many of the interesting data in the real world can and should be treated as a collection of constituent items. For instance, in the field of language modeling and text processing, an article can be considered as a group of paragraph or sections, and further a paragraph is a group of words. In computer vision and image processing, a prevailing assumption is that a visual scene consists of a group of local image patches. In recommendation systems, a user is mainly described by the group of products he/she bought. In social network a community is a group of people. In these problems, the actual abstraction should be "one set of points per object". We call these kinds of data that are organized by groups as the *collective data*. In the following, we shall call the basic constituent entities as "points", and the aggregations of points as "groups" (or equivalently "sets" or "bags").

The task of learning from collective data arises in many application domains, yet our research is largely motivated by the demands from the scientific community. Due to the advancement of sensory systems and the ever increasing computation power, now the scientists are facing data that are at unprecedented scales. For example in astronomy, modern telescope pipelines like the *Sloan Digital Sky Survey*[1] (SDSS) can produce observations for a vast amount of celestial objects. In physics, large-scale simulation systems such as the *JHU Turbulence Database Clusters*[2] (JTDC) were implemented to study the dynamics of fluid and particles. In these problems, we have huge amount of collective data that are impossible to be examined by experts. Therefore, computational assistance is needed.

[1] `http://www.sdss.org`

[2] `http://turbulence.pha.jhu.edu`

In this thesis, we try to answer the question: *how and what can we learn from collective data*? We emphasize that it is important to look beyond the point-level behaviors of data when designing learning algorithms for collective data. Consider doing the task of novelty detection on an article that is represented as a bag (group) of words (points). While paragraphs talking about either "machine learning" or "gummy bears" are not novel on their own, an article containing both of the terms might be interesting. In computer vision, it is unreliable to classify a scene image just by the presence of a single object (*e.g.* we cannot say with certainty that an image is an city scene if it contains a building; it might also be a beach scene). Instead, we should consider the overall composition of different objects in this image.

The most straightforward method, and indeed the mostly used one, to learn from collective data is to treat the groups as single objects and transform them into vectors, so that the traditional point-wise learning techniques can be applied. However, in many situations this conversion is essentially a feature engineering process that can be domain specific and difficult. Moreover, it is likely that during the conversion some useful information is lost. Therefore, we aim at developing learning methods that inherently respect the collective nature of data and can learn from them directly.

We investigate several approaches to learn from different types of collective data. The first type of collective data are groups of one-dimensional discrete points that are modeled by categorical random variables. This kind of data are abundant in text processing and recommendation systems, where each word or item is considered a discrete symbol. It is popular to make the assumption that the points are *exchangeable i.e.* the order of the points within a group carries no information, so that we can summarize the points in a group by the histogram which is the sufficient statistics for any conceivable statistical methods. This approach provides a natural way of converting the groups into vectors, to which many existing learning methods can be applied. In this work, we learn from these data within the matrix factorization framework, where the matrices are formed by stacking the vector representations of the groups. We develop two algorithms. The first one introduces the extra "time" dimension to model the temporal effects of data using tensors. The second one is a robust method that enables reliable factorization in the presence of outliers, and we use it for anomaly detection purposes.

In more general and more common data sets, the groups contain points that are continuous, multidimensional vectors. Intuitively, each group is a point cloud in a multidimensional space. In this case, we no longer have a natural way of reducing the groups into the concise vector representations, and that raises more challenges into our learning tasks. To attack this problem, researchers often "encode" each vector by a discrete point, and then use the approach described in the previous paragraph. Indeed, "encoding" itself is a big and interesting topic. But even though many sophisticated algorithms have been proposed in the recent years, a significant amount of domain knowledge and human effort may be required depending on the specific problem.

We, on the other hand, try to learn from these groups directly without any conversion. Both generative and discriminative methods are studied. From the generative perspective, we can model the generating process of the groups and points, and then use the insights from the models to help us accomplish other learning tasks. Again assuming the exchangeability of points, we devise models that can capture the multi-level characteristics of the groups based on topic modeling, and then use them for group anomaly detection, clustering, and classification. These models consist of different probabilistic component to achieve the balance between flexibility, speed, and robustness.

Discriminative methods are also considered to learn from collective data based on the similarity or dissimilarity measures between groups. Assuming that the points within a group are *i.i.d.* samples from some underlying distribution, we construct novel estimators of kernels between the groups based on a class of new nonparametric divergence estimation methods. These kernel estimators are provably consistent and efficient to compute. Having them, we can take advantage of the existing methods that solely depends on similarities (*e.g. support vector machines* (SVM) and *spectral clustering* [122]) to accomplish various learning tasks including classifications, regression, clustering, dimensionality reduction, and anomaly detection. In our experiments on both synthetic and real-world data sets, these new methods has achieved the state-of-the-art performances.

These methods are further enhanced to cope with challenges we might face in real data sets. To increase the speed of the algorithms, we examined possible ways of reducing the size of the groups while preserving the learning performances. We also investigated the possibility of accomplishing learning tasks using only partial similarity measures so that less group kernel computations are needed. Finally, sampling biases in collective data are considered and we proposed novel divergence between groups to solve the problem.

We want our research to be truly useful. In addition to designing new machine learning methods, we developed practical tools to assist the scientific researchers. We analyze the real-time astronomy data generated by SDSS using the algorithms we developed. A website is built to present interesting celestial objects to the astronomers and to collect their feedbacks. Similar tools could be used to assist physicists in discovering and studying novel phenomena in large scale turbulence simulations.

This thesis contains several of our published work:

- Chapter 2: [172]
- Chapter 3: [103]
- Chapter 4: [173, 174]
- Chapter 5: [130, 131]
- Chapter 7: [175]

The rest of this chapter is organized as follows. In Section 1.1, we describe some notations that we use throughout this thesis. Section 1.2 and 1.3 introduce the background and literature on learning from discrete and continuous collective data respectively. Finally in Section 1.6 we state the purpose the our research and overview the structure of this thesis.

1.1 Notations

First, we list some symbols in Table 1.1 that we shall use through out this thesis. The common format of the data sets in this thesis is as follows. We consider a data set that consists of M groups of points $\{G_m\}_{m=1,\ldots,M}$, and each group G_m is a set of N_m points as $G_m = \{x_{mn}\}_{n=1,\ldots,N_m}$. The numbers of points in each group can be different. Note that we only consider cases where the groups are pre-defined, *i.e.* we are not addressing the clustering/partition problems of how to divide the points into groups.

Table 1.1: Symbols

Symbol	**Definition and Description**
M	The number of groups.
N_m	The number of points in group m.
D	For discrete data, D is the total number of categorical values. For continuous data, D is the dimensionality of the points.
x_{mn}	The nth point in group m. For discrete data, x_{mn} is a categorical variable taking values from $\{1, \ldots, D\}$. For continuous data, $\mathbf{x}_{mn} \in \mathbb{R}^D$.
G_m	Group/Set/Bag m. $G_m = \{x_{m,1}, \ldots, x_{m,N_m}\}$ contains the set of points in group m.
y_m	The label of G_m. In classification problems y_m is the class label, while in clustering problems y_m is the cluster membership.
f_m	If we assume that the points in G_m are random samples, then f_m is the distribution that generates these samples. Usually f_m is not observed.
$\mathbf{g}_m$	The vector representation of group G_m for discrete data. $\mathbf{g}_m \in \mathbb{R}^D$ is usually the histogram of points in G_m.
$\mathbf{X}$	The data matrix for discrete data. $\mathbf{X} = [\mathbf{g}_1, \ldots, \mathbf{g}_M]^T \in \mathbb{R}^{M \times D}$ is constructed by stacking the $\mathbf{g}_m$'s.
$\text{NN}_G(x)$	The nearest neighbor of point x in group G.
$\mathbb{S}^K$	The K-dimensional probability simplex.
$\mathbf{I}$	The identity matrix.
$I(c)$	The indicator function. $I(true) = 1$ and $I(false) = 0$.
$\odot$	The element-wise multiplication between vectors or matrices of the same size.
$< \cdot, \cdot >, \circ$	The inner and outer products of vectors.

The points in the groups can either be discrete or continuous and multidimensional. For discrete points, x_{mn} is a categorical variable taking values from $\{1, \ldots, D\}$, where D is the number of possible values. For continuous data, x_{mn} is a D-dimensional vector as $x_{mn} \in \mathbb{R}^D$, where D is the dimensionality. For example, in text modeling, we can consider a document as a group of words. In this case, G_m is a document, x_{mn} is the nth word in G_m, and x_{mn} can only take one of the values/words from the vocabulary. In computer vision, we can consider an image as a group of patches. In this case, G_m is an image, x_{mn} is the feature of the nth patch in G_m, and x_{mn} is a D-dimensional feature vector extracted to described that patch.

We often assume that the points in G_m are random samples from some distribution f_m. Usually f_m is not observed and has to be either inferred or estimated. For instance, in text modeling, under the bag-of-words (BoW) assumption which ignores the information carried by the order the words, we can say that a document G_m has an underlying multinomial word distribution f_m, and the document is realized by randomly sample points/words from f_m. Similar examples can also be found in continuous data sets based on appropriate assumptions.

In classification, the objects of interest are associated with class labels. We use y_m to denote the class label of group G_m. Note that we care only about learning on groups, therefore the class labels (as well as cluster memberships and other learning output) are only for groups and not for points. This is different from *multiple instance learning* [184], where the data are organized by groups but the learning is on points; see Section 1.4 for more details.

To denote the sub-matrices and sub-vectors, we use the Matlab® notation. For example, $\mathbf{X}_{1:100,:}$ denotes the first 100 rows of the matrix $\mathbf{X}$.

Nearest neighbors (NN) are also frequently used. We use $\text{NN}_G(x)$ to denote the NN of point x in group G. If x is in G then it excludes itself during the search. Ties, if any, are broken arbitrarily.

1.2 Learning from Discrete Data

Many collective data sets contains discrete/categorical points. For example, in text processing documents comprises discrete words that take values from a vocabulary. In recommendation problems, a user is characterized by the set of items he/she bought, which are discrete symbols.

If we assume that these discrete points are infinitely exchangeable, *i.e.* the order of the points does not affect the nature of the groups, then we can succinctly represent the group by its *sufficient statistics*: the histogram. Let a group of points be $G = \{x_1, \ldots, x_N\}$ with $x_n \in \{1, \ldots, D\}$. Then G can be represented by a histogram $\mathbf{g} = \left[\sum_{n=1}^{N} I(x_n = 1), \ldots, \sum_{n=1}^{N} I(x_n = D)\right] \in \mathbb{R}^D$. This approach reduces groups to vectors for which we have mature analysis tools and learning techniques. Note that no information is lost during this reduction process under the exchangeability assumption.

In some problems, even when the points are not discrete, researchers would still discretize them using techniques like *vector quantization* [59] or *sparse coding* [94, 181], and then use aggregation methods that are similar to histogram reduction. A well-know example is the *bag-of-words* representation used in image processing and computer vision (*e.g.* [19, 50, 136]). Inevitably, the information carried by the original data might be compromised during this reduction. Nevertheless, the resulting vectorial representation is compact and familiar, and researchers can still develop good learning methods based on this reduced representation. In fact, the problem of how to effectively discretize the points makes its own field, but it is out of our focus.

Nevertheless, effectively algorithms to learn from groups with discrete points is very important, and they can be also applied to many traditional problems with individual vectorial points. In the later part of this thesis, we develop algorithms that can learn from collective data directly without the discretization or other conversions.

1.2.1 Factorization Methods

In this research, the learning of discrete collective data that can be converted into vectors is done under the factorization framework for its simplicity, speed, and wide applicability.

Matrices are very useful in representing data. Vectors can be stacked to form matrices such as the *design matrices* in regression and classification, and the *document-word matrix* in language modeling and text processing. Matrices are also used to describe *networks and graphs*, as well as *preference data* in collaborative filtering [85, 138].

We denote the data matrix as $\mathbf{X} \in \mathbb{R}^{M \times D}$, where each row represents a groups and each column represents one discrete value. One of the most common analysis for $\mathbf{X}$ is factorization/decomposition, such as the *principal component analysis* (PCA). For design matrices, PCA reveals the intrinsic linear structure of data. For text data, *latent semantic indexing* (LSI) [72] and *non-negative matrix factorization* (NMF) [43] are often applied. The low-rank assumption is also useful in *matrix completion* [23, 111] and *collaborative filtering* [138, 141].

To do a low-rank factorization, we assume that $\mathbf{X}$ has a low rank K and decomposes as

$$\mathbf{X} \approx \mathbf{U}\mathbf{V}^T, \quad \mathbf{U} \in \mathbb{R}^{M \times K}, \mathbf{V} \in \mathbb{R}^{D \times K}. \tag{1.1}$$

There are several ways to interpret the factor matrices $\mathbf{U}$ and $\mathbf{V}$. They can be thought of as the small number of bases and coefficients to reconstruct the matrix. They can be also interpreted as the low-dimensional latent factors/features for the rows and columns of the matrix $\mathbf{X}$, such that the inner-products of the factors approximate the matrix's entries. In terms of our learning problem, the bases interpretation means that each group can be approximated by a linear combination of "basis groups"; The factor interpretation means that each group and discrete value have a latent feature, and the "compatibility" between the features of a group and a value determines how often that value appears in the group.

1.2.2 Temporal Modeling

Successful as they are, one limitation of most existing factorization algorithms is that they are static models in which groups are assumed to be stationary over time. However, real data is often evolving over time and exhibits strong temporal patterns, and traditional static methods are incapable of learning the shift of a group's composition of points. In other words, a group's properties and behavior may change over time, but it can only be represented one fixed latent factor in traditional factorization models.

Another outstanding problem for many data set is that they are often very sparse. For example, a document only contains a very small set of words compared to the whole vocabulary. Taking the Netflix [3] data for instance, there are $17,770$ movies and $480,189$ users, but only $99,072,112$ training ratings. This means that we are learning from a matrix with only 1.16% of its entries given. This phenomenon presents two challenges for us. The first one is how to avoid over-fitting, and the second is how to take advantage of this sparsity to accelerate computation.

To solve the problems, we propose a factorization method that is able to model time-evolving data. In addition to the factors that are used to characterize groups and values, we introduce another set of latent factors for time itself. Intuitively, these additional factors represent the population-level modulation of latent features at each particular time. This kind of modeling allows us to introduce flexibility into the time dimension without further sparsifying the data, which would happen if we where to estimate different models at different time. We further enhance the method by Bayesian

[3] http://www.netflixprize.com/

techniques to avoid overfitting. Finally, the speed of the new algorithm is not much slower than static methods.

1.2.3 Robust Factorization

Another aspect of data we are interested in is the outliers/anomalies. Real-world problems almost always involve anomalies or outliers that do not conform to our assumptions. They can severely degrade the models' quality, or lead to novel discoveries. Thus we want robust methods that can produce high-quality models as well as find the outliers. The definition of outlier varies depending on specific problems, but in general outliers lie in the low-density regions of data distributions. [26] surveyed outlier detection problems. In our factorization work, we consider *subspace outliers* and assume that normal data reside in a low-dimensional linear subspace (the row/column space of the low-rank matrix). For instance in signal processing, a normal signal can be reconstructed by a few bases. If a signal cannot be well reconstructed, it is an outlier.

Factorization methods are often not robust due to the L_2-norm used to measure approximation errors [86, 176]. Many robust estimators has been proposed (*e.g.* [67, 79, 86, 88, 99]). A common approach is to replace the L_2-norm with robust norms are insensitive to outliers. For example L_1 norm is widely used for robustness [15, 22]. Other measures like the *Huber loss* [74] and the *Geman-McClure* function have also been employed [88, 123]. Another strategy is to exclude the outliers: we first guess which data are outliers, and then reduce their influences [86, 176].

In this thesis, we took the approach of using robust norms. Specifically, we use the L_0-norm, which counts the number of outliers disregarding their magnitudes, to replace the L_2-norm to measure the errors. The resulting algorithm is simple, fast, and effective. It can also be shown that the recently popular algorithms using the L_1-norm and the *nuclear* norm [22, 177] are relaxations of this method.

1.3 Learning from Continuous, Multidimensional Data

In many other problems, points are multi-dimensional vectors with continuous values. In this case, it is not easy to summarize a group by a vector or get sufficient statistics. Current learning of these data are often done by discretization. However, this conversion step may lose valuable information and might need significant domain knowledge. Therefore, we want to attack this problem directly. Our basic assumption is that the points in group G_m are infinitely exchangeable samples from an underlying probability distribution f_m. To learn from the groups, we learn the f_m's.

1.3.1 Generative Models

To motivate the characterization of groups we consider the problem of finding group anomalies. We consider two types of group anomalies. A *point-based* group anomaly is a group that contains individually anomalous points *e.g.* an image containing a two-headed wolf. A *distribution-based* anomaly is a group where the points are relatively normal, but as a whole they are unusual *e.g.* an image containing a pack of wolves and a flock of sheep together.

Most existing work on group anomaly detection focuses on point-based anomalies. They first identify anomalous points and then find their aggregations. Clearly this paradigm will not work for distribution-based anomalies. One solution is to design problem-specific features for groups. However, it relies on feature engineering that is domain specific and can be difficult.

In this thesis, we design several probabilistic models to capture the generating process of the collective data. By training these models, we can learn the "normality" of the data, and hence detect unusual behaviors. We can also infer the important latent attributes of the groups that can help us find both types of anomalies. The tools we use are developed based on topic modeling.

1.3.2 Topic models

We can learn the generating process of the groups using probabilistic models. For this purpose, particular useful are the *topic models*, among which the *probabilistic latent semantic analysis* (PLSA) [72] and *latent Dirichlet allocation* (LDA) [14] are the most well-known.

Topic models are originally proposed for text modeling, where we have words as points and document as groups. They are hierarchical mixture models built upon the assumption of exchangeable points *i.e.* the order of points/words does not matter. Essentially in LDA, a group G_m is modeled by a mixture density $f_m = \sum_{k=1}^{K} \theta_{mk}\beta_k$, where K is the number of mixture components. We call the mixture components β_k's as the *topics* and the mixing weights $\theta_m \in \mathbb{S}^K$ as the *topic weights*. LDA forces all groups to share the same topics $\{\beta_k\}_{k=1}^K$ so as to share information and enhance the statistical power.

Topic models are often described by generative schemes. For example a LDA model with topics $\{\beta_k\}_{k=1}^K$ and prior Dirichlet topic weight distribution $\mathcal{D}ir(\alpha)$ can be described by Algorithm 1, and the resulting complete likelihood is

$$p\left(G_m, \theta_m, \mathbf{z}_m | \alpha, \beta\right) = \mathcal{D}ir(\theta_m|\alpha) \prod_n \mathcal{M}(z_{mn}|\theta_m)\beta_{z_{mn}}(x_{mn}).$$

Algorithm 1 The generative process of LDA.

```
For group $m = 1$ to $M$:
    1. Choose the topic weight $\theta_m \in \mathbb{S}^K, \theta_m \sim \mathcal{D}ir(\alpha)$.
    2. For points $n = 1$ to $N_m$:
        (a) Choose a topic $z_{mn} \sim \mathcal{M}(\theta_m), z_{mn} \in \{1, \dots, K\}$.
        (b) Generate a point $x_{mn} \sim \beta_{z_{mn}}$.
```

Although topic models are proposed for discrete data like text, it is straightforward to use them for continuous multi-dimensional points by using multivariate distributions such as Gaussians as the topics $\{\beta_k\}$. The idea stays the same: we approximate the underlying distribution f_m with a mixture model $\sum_k \theta_{mk}\beta_k$, and try to figure out how this mixture model was generated.

1.3.3 Enhanced Topic Models

The challenge we face in using generative models for group anomaly detection is how to devise flexible models to fully characterize the data. In traditional topic models, the model parameters, such as the topic weight distribution $\mathcal{D}ir(\alpha)$, are considered "priors", and their role is incorporate prior knowledge to assist the estimation of latent variables. In our problem, however, we need the model parameters to describe the detailed behaviors of data in ordered to differentiate what is normal and what is not.

Since LDA, various improvements have been proposed. Many of them enhanced the flexibility of generating mechanism of topic distributions [50, 80] or capture correlation between topics [13, 102]. On the other hand, [45] allow the topics to vary for different groups in order to account for the burstiness of words. These ideas are helpful ingredients for creating a model that can thoroughly capture how groups are generated.

We proposed several enhanced topic models for group anomaly detection. We study the models' flexibility, robustness, learning and inference speed, *etc*, and present our findings. Based on these models, we proposed several novel scoring functions to detect both the point-based and distribution-based anomalies.

1.3.4 Discriminative Methods

Discriminative methods can also be used to learn from collective data, in which we circumvent the need of the generating process of data and aim directly at what we want to learn, *e.g.* the class label of a group. Here we focus on learning methods that are based on pairwise (dis)similarity measures, such as the SVM.

Several methods has been proposed to measure the similarity between sets of vectors. [171] used several traditional sets distances such as the *Hausdorff* distance for this purpose. [63, 64] proposed the *pyramid matching kernels* between vector sets based on hierarchical approximate matching of points. [170] measures group similarities based on the angles between the subspaces spanned by the points from different groups. [151] proposed *algebraic kernels* between matrices that represent sets of vectors.

[65, 155] tries to embed probability distributions into *reproducing kernel Hilbert spaces* (RKHS). In these methods, a density f is mapped to a *mean function* μ_f in a RKHS $\mathcal{H}_k$ induced by a kernel $k(\cdot,\cdot)$ as $\mu_f(\cdot) = \mathbb{E}_{x\sim f}[k(x,\cdot)]$. Then the inner-product between densities is just the inner-product of the mean functions in $\mathcal{H}_k$, which is equivalent to the average kernel values between each pair of inter-group point pairs. The discrepancy between densities is defined as the distance between the mean functions in $\mathcal{H}_k$.

Alternatively, we can measure the divergences between the underlying distribution f_m's. In statistics, this question can be answered by the results from two-sample tests (*e.g.* the probability of rejecting the null hypothesis) such as the *Student t-test*, the *Kolmogorov-Smirnov test*, and the *permutation test*. However these methods either rely on parametric assumptions, use only limited statistics, or have difficulties in high-dimensions.

Another approach is to first estimate densities f_m first and then measure similarities. [140] compute divergences by discretizing the continuous densities. [75] defines *Fisher kernels* between parametric densities. [119] fits *Gaussian mixture models* (GMM) to compute the Kullback-Leiber (KL) divergences. [76] fits exponential family densities, and then compute *product kernels* between these densities in an RKHS. [42] defined a kernel on the level-sets of fitted densities. The problem with these methods is that density estimation is itself notoriously difficult and parametric methods often introduce biases.

We took the distributional approach in this work. Based on the work of [129], we propose a nonparametric method to estimate a family of kernels between distributions based on observed samples, while avoiding explicit density estimates. These estimators are both efficient and accurate, and is able to achieve the state-of-the-art performance on real data sets.

1.3.5 Accelerated Learning

One major disadvantage of the algorithms that learn from collective data is their high computational cost compared to those operate on vectors. For example, in computer vision, by discretizing the local features and aggregating them by the "bag of visual words" method [50], typically a 256×256 image can be characterized by one $1,000$-dimensional vector, amounting to just 1KB of data. On the other hand, to represent an image as a group of SIFT features, we typically need more than $1,500$ 128-dimensional vectors, amounting to about 190KB of data. The further computation needed to process such groups is likely to be also orders of magnitudes larger. In order to make the learning algorithms in this work truly useful, this hurdle of computational efficiency must be overcome.

We explore different ways to improve the speed of the group similarity based methods. In most cases, the cost to train, store, and apply the model is determined by the sizes/cardinalities of the groups. Therefore, one approach is to directly attack the crux of the problem by reducing the size of groups while maintaining the learning performance, in an unsupervised way. We call such an operation *condensing*. We analyze and evaluate several possible ways of decrease the size of a group, and discover that distribution approximation via k-Means can successfully achieve the goal of condensing.

Another way to improve the speed of similarity-based algorithms is to reduce the number of similarity evaluations. In SVM or *spectral clustering*, we need a full kernel matrix or distance matrix, which means that similarities are needed for every pair of groups. In many problems, structures exist in the similarity/divergence matrices that allows us to infer the full matrix based on only part of the entries. By exploiting such structures, we can compare only some pairs of groups and "complete" the similarities between the other pairs. In this work, we study methods to complete both the kernel matrices and the divergence matrices, and compare their empirical performances.

1.3.6 Sampling Bias

One factor that can significantly affect the effectiveness of learning is the sampling bias. In realistic situations, sampling bias alters the way we collect points from the underlying distribution, and makes the observed sample not representative of the true distribution. In other words, even though the group G_m has a underlying distribution f_m, the actual points in G_m may not be faithful samples from f_m, but rather drawn from some distorted version of f_m. Therefore, it undermines the fundamental validity of learning algorithms, including all the methods we proposed in this work. Though been extensively studied in statistics, this key problem has been largely ignored by the previous research on learning from collective data.

We propose conditional divergences to correct these distortions and learn from biased groups effectively. Traditional divergences mainly compares the joint distribution of the random variables. On the other hand, conditional divergences focus on the conditional distributions of some variables given the rest, and is insensitive to the distribution of the variables that we are conditioning on. As long as the conditional distributions are intact, the conditional divergences will be accurate. An efficient estimator is also developed for the conditional divergences.

1.4 Related Fields

Several other research fields are closely related to learning from collective data. Statistical relational learning (SRL) [60] enhances point-centered machine learning by consider a group of point and their relationships altogether. For example, SRL studies the *collective classification* problem, where the goal is to simultaneously classify several objects based on their attributes and relations. This problem is also studied under the name of *structural prediction* typically using *Markov networks* [158] or *large-margin* approaches [161]. Even though collective behaviors are considered, SRL still tries to learn the labels of the points, whereas our research shall only focus on the groups.

Multiple instance learning (MIL) [184] also tries to classify groups of points. In MIL, a group is positive if at least one of its point is positive; otherwise it is negative. Consequently, in MIL the nature of a group is determined by a few of its points. By comparison, we assume that it is the holistic behavior of all the points that characterizes a group. Nevertheless, sometimes the methods for learning from collective data indeed overlap with methods for MIL. For example, some kernels [12, 55] can be used to do MIL as well as learning from collective data.

Another related field is on *graph kernels*. Graph kernels studies the similarities between graphs, which are defined by a set of nodes and edges. Graphs can also be considered as collective data. But unlike before, graph data are structured and the elements in a graph can no longer be considered *i.i.d.* or exchangeable. Graph kernels often count the intersection of sub-structures between graphs. For example, the *random walk kernel* [54, 165] measures the path similarity of random walks in different graphs. [6] designed kernels between groups based on graph kernels.

Finally, quantization/discretization/encoding has always been the traditional way of learning from collective data. These methods first turns vectors into discrete points, and then reduce the groups into vectors. When such conversion is done, our algorithms of learning from discrete data can be applied. But our main focus of this thesis is to avoid such conversions and learn high-quality results from collective data directly.

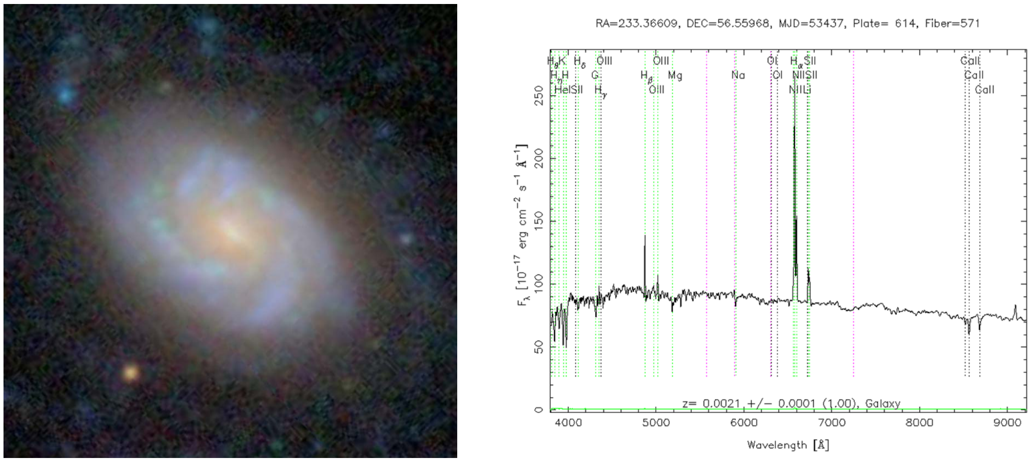
1.5 Challenges in Scientific Data

This research is motivated by the need of analyzing scientific data sets. The proposed algorithms are intended to help the scientists learn about the data sets they observe. Here we describe two projects that involves astronomical surveys and particle/fluid simulation systems. In these projects, we want to conduct explorative study to obtain the overall data profile, and pick out potentially interesting things using *novelty detection*.

Astronomical Surveys

Astronomical surveys provide a holistic view of the universe by imaging a large portion of the sky. The *Sloan Digital Sky Survey*[4] (SDSS) project has imaged more than 35% of the sky and gives millions of observations for stars, galaxies, quasars, and other celestial objects. Astronomers are also planning for even more powerful survey telescopes such as the *Large Synoptic Survey Telescope*[5] (LSST) that can scan the sky deeper and faster. The massive amount of data produced by these surveys calls for the assistance of computational methods.

We focus on the spectroscopic observations in the SDSS data set. SDSS provides for each object a 3700-dimensional spectrum as shown in Figure 1.1b, along with its 3D spatial location for distant objects like galaxies and quasars. We directly take these spectra as the feature vectors for these objects.



(a) Photometric

(b) Spectroscopic

Figure 1.1: A galaxy observation from SDSS.

Two novelty detection tasks are currently considered on this data set. The first one is to find *individually anomalous objects* such as planetary nebulae. This task is traditional in that we want to find unusual vectorial points, yet it still poses unique challenges. For example, the spectra have high-dimensionality which makes it difficult to use density-based methods, and they usually

[4] `http://www.sdss.org`
[5] `http://www.lsst.org`

contain *emission lines* (see the spikes in Figure 1.1b) that could easily distort models. We focus on subspace outliers assuming that normal spectra can be reconstructed by a few bases, and develop robust factorization methods to address the emission lines.

The second task is to detect *special clusters of galaxies*. Based on the 3D spatial locations, we can find nearby galaxies and put them into clusters/groups. These clusters could shed light on the development of the universe [166], and it will be valuable to find interesting clusters for the astronomers. In this case, each cluster contains a set of spectrum vectors, and we shall address this problem using group anomaly detection methods.

Particle/Fluid Simulation

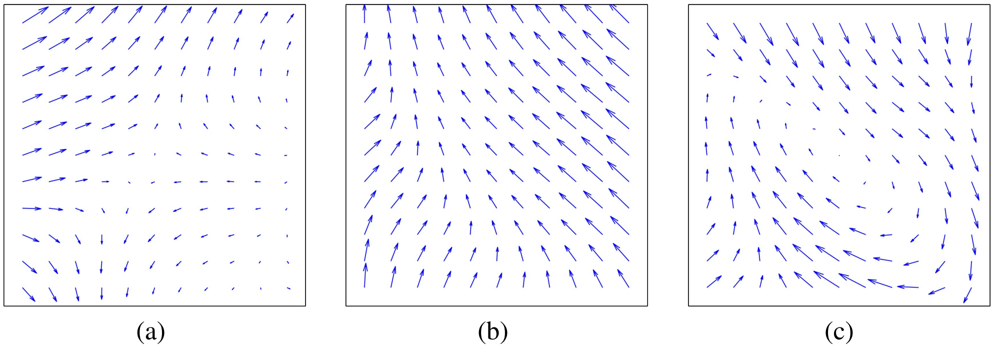

(a) (b) (c)

Figure 1.2: Local motion patterns in the JTDC simulation. Several 2D slices are presented.

In physics, researchers often simulate particle or fluid systems at a very large scale. For example, the *JHU Turbulence Database Clusters*[6] (JTDC) provides open access to 1024^4 space-time points in fluid simulations. At each points, the 3D velocity as well as other information including pressure and temperature are recorded. See Figure 1.2 for some examples. Our task is again to detect interesting phenomena in such massive data sets.

In these systems, a single particle is seldom interesting, but a group of particles can form interesting phenomena like the vortices as in Figure 1.2c. This can again be framed as a group anomaly detection problem. We treat points in a local region as a group, and aims at finding interesting collective motion patterns characterized by the distribution of locations, velocities, and other relevant features. In other scenarios, the researchers could give some examples of interesting phenomena, and then our supervised methods will classify the local regions accordingly.

1.6 Thesis Overview

Motivated by numerous practical problems, we propose various algorithms and models that can learn from collective data effectively, and present our answers to the question *how and what we*

[6] `http://turbulence.pha.jhu.edu`

can learn from collective data. We study different data types (discrete and continuous), as well as different learning approaches (generative and discriminative). Efforts are also made to improve the practicality of the proposed methods from multiple angles.

The rest of this thesis is organized as follows. In Chapter 2 and 3 we describe two algorithms to learn from discrete collective data focusing on modeling the temporal effects and the outliers. Chapter 4 and 5 describe the generative and discriminative methods of learning from continuous multi-dimensional data. Chapter 6 further studies how to construct Mercer kernels for collective data. Then we describe how to accelerate the learning while maintaining accuracies in Chapter 7. Chapter 8 proposed conditional divergences to correct the sampling biases in collective data. Finally in Chapter 9 we summarize the thesis and discuss future directions for this research.

Part I

Learning from Discrete Data

Chapter 2

Modeling Temporal Effects by Tensor Factorizations

In the chapter and the next, we are learning from groups that contain discrete points and can be converted into vectors. Specifically in this chapter, we shall discuss the recommendation *a.k.a.* collaborative filtering problems, in which a user rates a set of items, and the goal is to find out which other items this user might like. But the proposed factorization algorithms can also be applied to other problems of similar natures.

We consider the temporal dynamics in collaborative filtering problems. Real-world data are seldom stationary, yet traditional collaborative filtering algorithms generally rely on this assumption. Motivated by our sales prediction problem, we propose a factor-based algorithm that is able to take time into account. By introducing additional factors for time, we formalize this problem as a tensor factorization with a special constraint on the time dimension. Further, we provide a fully Bayesian treatment to avoid fine-tuning the parameters and achieve automatic model complexity control. To learn this model we develop an efficient sampling procedure that is capable of analyzing large-scale data sets. This new algorithm, called *Bayesian Probabilistic Tensor Factorization* (BPTF), is evaluated on several real-world problems including sales prediction, movie recommendation, and music recommendation. Empirical results demonstrate the superiority of the temporal model.

2.1 Introduction

Nowadays, recommendation *a.k.a.* collaborative filtering algorithms play a vital role in various automatic recommendation systems and has been used in many online applications such as *Amazon.com, eBay,* and *Netflix*. The set up of the problem is as follows. Different users rates different items based on their preferences. Suppose that we have observed for each user the set of items he/she has rated in the past. Then, collaborative filtering tries to predict the how a user would rate a currently unrated item. Note that here rating can mean either the actual preference such as movie ratings, or other indications such as the quantity of the consumed item.

Successful as they are, one limitation of most existing methods is that they are static models in which the statistical properties of data are assumed to be the same at different time. However, real data is often evolving over time and exhibits strong temporal patterns. To motivate our research, let us consider the following problem. A shoe production company sells many types of shoes to its retailers around the world. Now this company wants to predict the demand of different shoes by different retailers for the ongoing season based on this season's initial orders and historical sales data. Having this prediction, the company can make more informed decisions on the marketing strategy and inventory planning. Obviously, the behavior of the market and the retailers is changing over time.

The traditional way to solve this problems is to use statistical regression models or time-series forecasting techniques for each shoe-retailer pair. Ideally, regression models can predict the order using the features of the retailers and the shoes. But the reality is that few retailer and product attributes are available due to the complexity of the domain knowledge and policy issues. What we have is only the transaction data recording the retailer, product, and quantity of each order. Therefore, it is more convenient to treat the products as discrete symbols, and represent the retailers by the sets of products they ordered. On the other hand, typical time-series models such as *autoregressive moving average* (ARMA) and *exponential smoothing* [20] use past data to make predictions. But they are not suitable for our problem neither because the data are extremely scarce and each season many new products are introduced, for which no historical data exist. Moreover, both of these two paradigms cannot exploit the "collaboration" between entities and hence are expected to perform poorly when the data is sparse. For these reasons, we use collaborative filtering to make the prediction.

Even if collaborative filtering is able to handle our data, traditional static methods are incapable of learning the shift of product designs and customers' preferences, especially considering that we are facing the volatile and fast-moving fashion business. The preference of the market can change from season to season and even within each season. In this case, trying to explain all the data with one fixed global model would be ineffective. On the other hand, if we only use the recent data or down-weigh the past, a lot of useful information would be lost, making the already very sparse data set even worse.

To solve this problem, we propose a factorization based method that is able to model time-evolving data. This method is based on probabilistic latent factor models [141, 142]. In addition to the factors that are used to characterize retailers and shoes, we introduce another set of latent features for each different time period. Intuitively, these additional factors represent the population-level preference of different (latent) features of the shoes at each particular time, so that they are able to capture concepts like "high-heeled shoes lost their popularity this fall" or "orders of golf shoes tend to arrive late". A special constraint is imposed on the time factors to ensure that the evolution of factors is smooth. This model learns the features of the entities using all the available data, while adapts these features to different time periods. It can be formulated as a probabilistic tensor factorization problem, thus is widely applicable to other similar data sets.

The modeling of temporal effects is also useful in other collaborative filtering problems, since often the preferences of users are subject to change. Remarkably, the remarkable progress in the *Netflix Prize* contest is attributed to a temporal model [85]. The winner identifies strong temporal patterns in the data, and exploits them to achieve a significant improvement leading to the best performance attained by a single algorithm.

One outstanding problem for many data is that they are often very sparse. A retailer usually only order a small subset of shoes from the whole product line. In the Netflix Prize[1] data set, there are $17,770$ movies and $480,189$ users, but only $99,072,112$ training ratings. This means that on average each user has only rated 1.16% of the movies. This phenomenon presents two challenges for us. The first one is how to avoid over-fitting, and the second is how to take advantage of this sparsity to accelerate computation. To address the first problem, we extend our approach using Bayesian techniques. By introducing priors on the parameters, we can effectively average over various models and ease the pain of tuning parameters. We call the resulting algorithm *Bayesian Probabilistic Tensor Factorization* (BPTF). And for scalability, we develop an efficient *Markov Chain Monte Carlo* (MCMC) procedure for the learning process so that this algorithm can be scaled to problems like Netflix.

In our experiments we applied our BPTF model to the sales prediction problem as well as movie and music recommendation problems. The empirical results show that using the temporal modeling, consistent improvement of prediction accuracy can be achieved over static methods at the cost of little extra complexity and computation.

The rest of this chapter is organized as follows. First we introduce some preliminaries about factorization methods in Section 2.2. In Section 2.3 we describe the proposed model, which are enhanced in Section 2.4 by Bayesian techniques. Some related work is discussed in Section 2.5. Section 2.6 presents the empirical performance and efficiency of our method. Finally we make our conclusions.

2.2 Preliminaries

First we introduce some background and notations. Our data is stored in a matrix $\mathbf{X} \in \mathbb{R}^{M \times D}$, which can be considered as a rating matrix in collaborative filtering, or simply a data matrix formed by stacking the vectors by row. Corresponding to the rows and columns of $\mathbf{X}$, there are two types of entities $\{u_i\}$ and $\{v_j\}$. In collaborative filtering, we call them “user” and “item” respectively. They can also be “group” and “value” in the context of learning from discrete collective data. Obviously, there are M users and D items.

The $(i, j)th$ element of $\mathbf{X}$, denoted as x_{ij}, is the “rating” that user i gave to item j (or it could mean the number of times value j appeared in group i). Note that in collaborative filtering a large portion of the entries in $\mathbf{X}$ is not observed.

Typical collaborative filtering algorithms can be categorized into two classes: neighborhood methods and factorization methods. Generally factor-based algorithms are considered more effective than those based on neighborhood. But these two class are often complementary and the best performance is often obtained by blending them [9]. A practical survey of this field can be found in [84].

One representative factor-based method for collaborative filtering is the *probabilistic matrix factorization* (PMF) [141]. PMF assigns a Z-dimensional latent feature vector for each user and

[1] http://www.netflixprize.com/

item, denoted as $\mathbf{u}_i, \mathbf{v}_j \in \mathbb{R}^Z$, and model each rating as the inner-product of corresponding latent features, *i.e.* $x_{ij} \approx \mathbf{u}_i^T \mathbf{v}_j$ where $\mathbf{u}_i^T$ is the transpose of $\mathbf{u}_i$. Formally, the following conditional distribution is assumed:

$$p(\mathbf{X}|\mathbf{U}, \mathbf{V}, \alpha) = \prod_{i=1}^{M} \prod_{j=1}^{D} \mathcal{N}\left(x_{ij}|\mathbf{u}_i^T \mathbf{v}_j, \alpha^{-1}\right)^{I_{ij}}, \tag{2.1}$$

where $\{\mathbf{u}_i\}, \{\mathbf{v}_j\}$ are columns of $\mathbf{U} \in \mathbb{R}^{Z \times M}$ and $\mathbf{V} \in \mathbb{R}^{Z \times D}$, α is the observation precision, and I_{ij} is the indicator that x_{ij} has been observed. Zero-mean Gaussian prior are imposed on $\mathbf{u}_i$ and $\mathbf{v}_j$ as a regularization.

This model can be learned by estimating the value of $\mathbf{U}$ and $\mathbf{V}$ using the *maximum a posteriori* (MAP) principle. It turns out that this learning procedure actually corresponds to the following weighted regularized matrix factorization:

$$\mathbf{U}, \mathbf{V} = \arg\min_{\mathbf{U},\mathbf{V}} \sum_{i=1}^{M} \sum_{j=1}^{D} I_{ij} \left(x_{ij} - \mathbf{u}_i^T \mathbf{v}_j\right)^2 + \lambda_U \sum_{i=1}^{M} \|\mathbf{u}_i\|^2 + \lambda_V \sum_{j=1}^{D} \|\mathbf{v}_j\|^2. \tag{2.2}$$

These formulations reflect the basic ideas of factorization based collaborative filtering.

The optimization problem (2.2) can be done efficiently using gradient descent. This model is very successful in the Netflix Prize contest in terms of speed-accuracy trade-off. The drawback though is that it requires fine tuning of both the model and the training procedure to predict accurately. This process is computationally expensive on large data sets.

We present the proposed method in two parts. First we extend PMF to *tensor factorization* to model temporal data, and formulate a *maximum a posteriori* (MAP) scheme for estimating the factors. Then we apply a fully Bayesian treatment to deal with the tuning of the prior parameters and derive an almost *parameter-free* probabilistic tensor factorization algorithm. Finally an efficient learning procedure is developed.

2.3 A Tensor Model for Temporal Data

In PMF each rating is determined by the inner product of the user feature and the item feature. To model their time-evolving behavior, we make use of the tensor notation. We can denote a rating as x_{ij}^k where i, j index users and items as before, and k indexes the *time slice* when the rating was given. Then similar to the static case, we can organize these ratings into a *three-dimensional tensor* $\mathbf{X} \in \mathbb{R}^{M \times D \times K}$, whose three dimensions correspond to user, item, and time respectively.

Based on the idea of PMF, we assume that each entry x_{ij}^k can be expressed as the inner-product of three Z-dimensional vectors:

$$x_{ij}^k \approx < \mathbf{u}_i, \mathbf{v}_j, \mathbf{t}_k > \equiv \sum_{z=1}^{Z} u_{zi} v_{zj} t_{zk}, \tag{2.3}$$

where $\mathbf{u}_i$, $\mathbf{v}_j$ are the factors for users and items while $\mathbf{t}_k$ is the additional latent feature vector/factors for the kth time slice. Using matrix representations $\mathbf{U} \equiv [\mathbf{u}_1, \ldots, \mathbf{u}_N]$, $\mathbf{V} \equiv [\mathbf{v}_1, \ldots, \mathbf{v}_M]$, and $\mathbf{T} \equiv [\mathbf{t}_1, \mathbf{t}_K, \ldots,]$, we can also express Eq. (2.3) as a *three-way tensor factorization* of $\mathbf{X}$:

$$\mathbf{X} \approx \sum_{z=1}^{Z} \mathbf{U}_{z,:} \circ \mathbf{V}_{z,:} \circ \mathbf{T}_{z,:}, \tag{2.4}$$

where $\mathbf{U}_{z,:}, \mathbf{V}_{z,:}$ and $\mathbf{T}_{z,:}$ represent the zth rows of $\mathbf{U}, \mathbf{V}$ and $\mathbf{T}$, and $\circ$ denotes the vector outer product. This is an instance of the CANDECOMP/PARAFAC (CP) decomposition [82], for which a illustration is in Figure 2.1. We prefer this model over the one that assigns a separate factor to each entity at each time slice because it will increase the number of factors dramatically and does not implement a tensor factorization.

An interpretation of the factorization (2.3) is that a rating depends not only on how well a user's preferences and an item's features match, but also on how much these features match with the "current trend" reflected in the time factors. For instance, if a user likes green shoes but the overall trend of this year is that few people wears them on the street, then this user is probably not going to buy them neither.

To account for the randomness in ratings, we consider the following probabilistic model:

$$x_{ij}^{k} \sim \mathcal{N}\left(< \mathbf{u}_i, \mathbf{v}_j, \mathbf{t}_k >, \alpha^{-1}\right), \tag{2.5}$$

i.e. the conditional distribution of x_{ij}^{k} given $\mathbf{U}, \mathbf{V}$, and $\mathbf{T}$ is a Gaussian distribution with mean $< \mathbf{u}_i, \mathbf{v}_j, \mathbf{t}_k >$ and precision α. Note that if $\mathbf{t}_k$ is an all-one vector then this model is equivalent to PMF. Since many entries in $\mathbf{X}$ are missing, estimation based on the model (2.5) may over-fit the observed entries and fail to predict the missing entries well. To deal with this issue, we place prior distributions on $\mathbf{U}, \mathbf{V}$, and $\mathbf{T}$ to regularize. Specifically we impose zero-mean Gaussian priors on user and item factors:

$$\mathbf{u}_i \sim \mathcal{N}\left(0, \sigma_U^2 \mathbf{I}\right), \qquad i = 1, \ldots, M, \tag{2.6}$$

$$\mathbf{v}_j \sim \mathcal{N}\left(0, \sigma_V^2 \mathbf{I}\right), \qquad j = 1, \ldots, D, \tag{2.7}$$

where $\mathbf{I}$ is the $D \times D$ identity matrix.

As for the time factors, since they account for the evolution of global trends, a reasonable prior belief is that they change smoothly over time. We further assume that each time feature vector depends only on its immediate predecessor. Therefore, we can use the following conditional prior for factors $\mathbf{T}$:

$$\mathbf{t}_k \sim \mathcal{N}\left(\mathbf{t}_{k-1}, \sigma_{dT}^2 \mathbf{I}\right), \qquad k = 1, \ldots, K. \tag{2.8}$$

For the initial time feature vector $\mathbf{t}_0$, we assume

$$\mathbf{t}_0 \sim \mathcal{N}\left(\mu_T, \sigma_0^2 \mathbf{I}\right), \tag{2.9}$$

where $\mu_T \in \mathbb{R}^Z$ is the prior mean. We call this model the *Probabilistic Tensor Factorization* (PTF).

Having the observational model (2.5) and the priors, we can estimate the latent features $\mathbf{U}, \mathbf{V}$, and $\mathbf{T}$ by maximizing the logarithm of the posterior distribution, which takes the following form

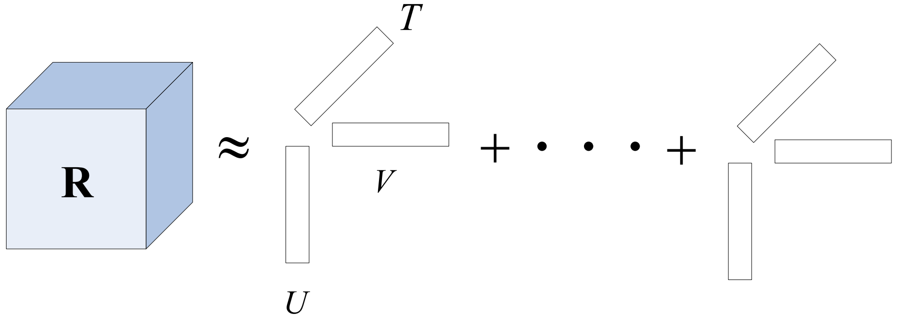


Figure 2.1: CP decomposition of a three-way tensor $\mathbf{X}$

assuming ratings are independent given the latent factors:

$$
\begin{aligned}
&\log p\left(\mathbf{U},\mathbf{V},\mathbf{T}|\mathbf{X}\right) \propto \log p\left(\mathbf{X}|\mathbf{U},\mathbf{V},\mathbf{T}\right) + \log p\left(\mathbf{U},\mathbf{V},\mathbf{T}\right)\\
=&\sum_{k=1}^{K}\sum_{i=1}^{M}\sum_{j=1}^{D} I_{ij}^{k} \log p\left(x_{ij}^{k}|\mathbf{u}_i,\mathbf{v}_j,\mathbf{t}_k\right) + \sum_{i=1}^{M}\log p(\mathbf{u}_i) + \sum_{j=1}^{D}\log p(\mathbf{v}_j) + \sum_{k=1}^{K}\log p(\mathbf{t}_k|\mathbf{t}_{k-1}) + \log p(\mathbf{t}_0)\\
=&-\sum_{k=1}^{K}\sum_{i=1}^{M}\sum_{j=1}^{D}\frac{I_{ij}^{k}(x_{ij}^{k}- <\mathbf{u}_i,\mathbf{v}_j,\mathbf{t}_k>)^2}{2\alpha^{-1}} + \frac{(\#\text{nz})\log\alpha}{2}\\
&-\sum_{i=1}^{M}\frac{\|\mathbf{u}_i\|^2}{2\sigma_U^2} - N\log\sigma_U - \sum_{j=1}^{D}\frac{\|\mathbf{v}_j\|^2}{2\sigma_V^2} - M\log\sigma_V - \sum_{k=1}^{K}\frac{\|\mathbf{t}_k-\mathbf{t}_{k-1}\|^2}{2\sigma_{dT}^2} - K\log\sigma_{dT} - \frac{\|\mathbf{t}_0-\mu_T\|^2}{2\sigma_0^2}\\
&-\log\sigma_0 + C,
\end{aligned}
$$

where I_{ij} indicates the presence of x_{ij}^{k}, #nz is the total number of ratings, and C is a constant. Under fixed values of $\alpha, \sigma_U, \sigma_V, \sigma_{dT}$, σ_0 and μ_T, which are usually referred to as hyper-parameters, maximizing the log-posterior with respect to $\mathbf{U}, \mathbf{V}, \mathbf{T}$ is equivalent to minimizing the following regularized sum of squared errors:

$$
\begin{aligned}
&\sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{D} I_{ij}^{k}(x_{ij}^{k}- <\mathbf{u}_i,\mathbf{v}_j,\mathbf{t}_k>)^2 + \sum_{i=1}^{N}\frac{\lambda_U\|\mathbf{u}_i\|^2}{2} +\\
&+\sum_{j=1}^{D}\frac{\lambda_V\|\mathbf{v}_j\|^2}{2} + \sum_{k=1}^{K}\frac{\lambda_{dT}\|\mathbf{t}_k-\mathbf{t}_{k-1}\|^2}{2} + \frac{\lambda_0\|\mathbf{t}_0-\mu_T\|^2}{2},
\end{aligned}
\tag{2.10}
$$

where $\lambda_U = (\alpha\sigma_U^2)^{-1}, \lambda_V = (\alpha\sigma_V^2)^{-1}, \lambda_{dT} = (\alpha\sigma_{dT}^2)^{-1}, \lambda_0 = (\alpha\sigma_0^2)^{-1}$.

This objective function (2.10) is non-convex, and we may only be able to find a local minimum. To optimize it, common choices include *stochastic gradient descent* and *block coordinate descent*, both of which update the latent feature vectors iteratively. The estimations $\mathbf{U}^*$, $\mathbf{V}^*$, and $\mathbf{T}^*$ can be used to predict an unobserved rating x_{ij}^{k} by the distribution (2.5).

One issue with the aforementioned approach is the tuning of the hyper-parameters $\alpha, \sigma_U, \sigma_V, \sigma_{dT}$, σ_0 and μ_T. Since there are many, the usual approach of hyper-parameter selection, such as cross-validation, is infeasible even for a modest problem size. We thus propose in the next section a fully Bayesian treatment to integrate out the hyper-parameters in the model, leading to an almost *parameter-free* estimation procedure.

2.4 A Bayesian Treatment

The performance of PTF depends the careful tuning of the hyper-parameters when model parameters are estimated by maximizing the posterior probability, as pointed out in [142]. The point estimate obtained by MAP is often vulnerable to over-fitting when hyper-parameters are not properly tuned, and is more likely so when the data is sparse.

An alternative scheme that may help alleviate over-fitting is the Bayesian estimation, which integrates out all model parameters, arriving at a *predictive distribution* of future observations given observed data. Because this predictive distribution is obtained by averaging all models in the model space specified by the priors, it is less likely to overfit a given set of observations.

However, when integrating over parameters one often cannot obtain an analytical solution, thus we will need to apply sampling-based approximation methods, such as Markov Chain Monte Carlo (MCMC). For large-scale problems, sampling-based methods are usually not preferred due to their computational cost and convergence-related issues. Nevertheless, [142] devises an MCMC procedure for PMF that can run efficiently on large data sets like Netflix. The main trick is choosing proper distributions for hyper-parameters so that sampling can be carried out efficiently.

Inspired by the work of [142], we present in the following a fully Bayesian treatment to the PTF model proposed in Section 2.3. We refer to the resulting method as BPTF for *Bayesian Probabilistic Tensor Factorization*.

2.4.1 Model Specification for BPTF

A graphical overview of our entire model is in Figure 2.2, and each component is described below. The model for generating ratings is the same as Eq. (2.5):

$$x_{ij}^k | \mathbf{U}, \mathbf{V}, \mathbf{T} \sim \mathcal{N}\left(< \mathbf{u}_i, \mathbf{v}_j, \mathbf{t}_k >, \alpha^{-1}\right). \tag{2.11}$$

As before, the prior distributions for the user and the item feature vectors are assumed to be Gaussian, but the mean and the precision matrix (inverse of the covariance matrix) may take arbitrary values:

$$\mathbf{u}_i \sim \mathcal{N}\left(\mu_U, \Lambda_U^{-1}\right), i = 1, \ldots, M,$$
$$\mathbf{v}_j \sim \mathcal{N}\left(\mu_V, \Lambda_V^{-1}\right), j = 1, \ldots, D.$$

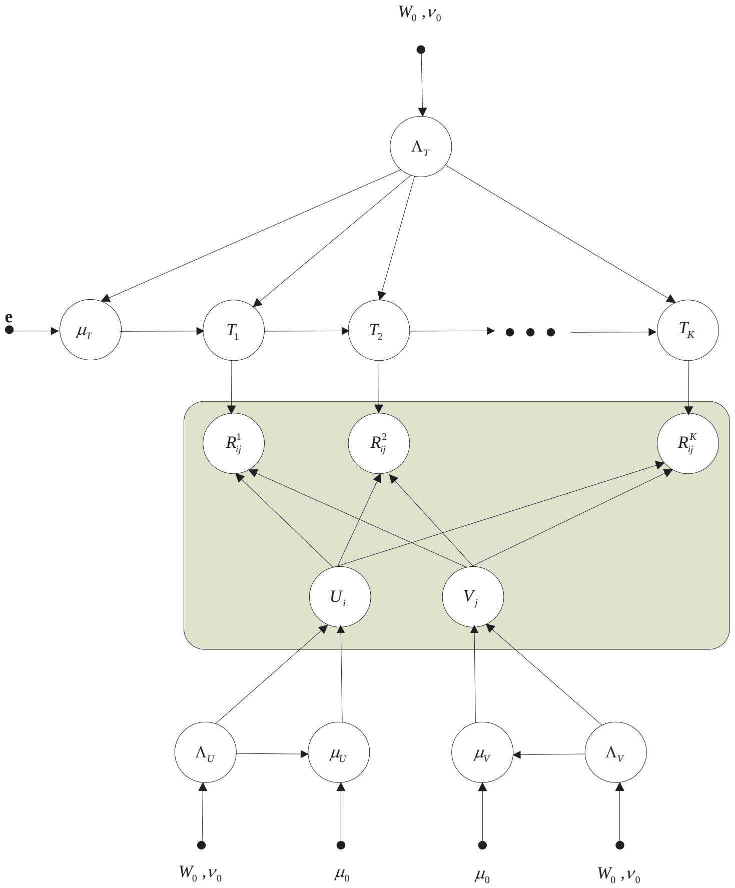


Figure 2.2: The graphical model for BPTF

For the time factors, we make the same Markovian assumption as in Section 2.3 and consider the priors:

$$\mathbf{t}_1 \sim \mathcal{N}\left(\mu_T, \Lambda_T^{-1}\right),$$
$$\mathbf{t}_k \sim \mathcal{N}\left(\mathbf{t}_{k-1}, \Lambda_T^{-1}\right), \quad k = 2, \ldots, K.$$

The key ingredient of our fully Bayesian treatment is to view the hyper-parameters $\alpha, \Theta_U \equiv \{\mu_U, \Lambda_U\}, \Theta_V \equiv \{\mu_V, \Lambda_V\}$, and $\Theta_T \equiv \{\mu_T, \Lambda_T\}$ also as *random variables*, leading to a *predictive distribution* for an unobserved rating x_{ij}^k,

$$p\left(\hat{x}_{ij}^k|\mathbf{X}\right) = \int p\left(\hat{x}_{ij}^k|\mathbf{u}_i, \mathbf{v}_j, \mathbf{t}_k, \alpha\right) p\left(\mathbf{U}, \mathbf{V}, \mathbf{T}, \alpha, \Theta_U, \Theta_V, \Theta_T|\mathbf{X}\right) \mathrm{d}\{\mathbf{U}, \mathbf{V}, \mathbf{T}, \alpha, \Theta_U, \Theta_V, \Theta_T\} \tag{2.12}$$

that integrates over the parameters, as opposed to the scheme in Section 2.3 which simply plugs the MAP estimates into Eq. (2.5).

We then need to choose prior distributions for the hyper-parameters (the so-called *hyper-priors*). For the Gaussian parameters, we choose their conjugate priors that simplifies subsequent computations:

$$\begin{aligned}
p(\alpha) &= \mathcal{W}(\alpha|\tilde{W}_0, \tilde{\nu}_0), \\
p(\Theta_U) &= p(\mu_U|\Lambda_U)p(\Lambda_U) = \mathcal{N}\left(\mu_0, (\beta_0\Lambda_U)^{-1}\right)\mathcal{W}(\Lambda_U|W_0, \nu_0), \\
p(\Theta_V) &= p(\mu_V|\Lambda_V)p(\Lambda_V) = \mathcal{N}\left(\mu_0, (\beta_0\Lambda_V)^{-1}\right)\mathcal{W}(\Lambda_V|W_0, \nu_0), \\
p(\Theta_T) &= p(\mu_T|\Lambda_T)p(\Lambda_T) = \mathcal{N}\left(\rho_0, (\beta_0\Lambda_T)^{-1}\right)\mathcal{W}(\Lambda_T|W_0, \nu_0).
\end{aligned}$$

Here $\mathcal{W}$ is the Wishart distribution over $Z \times Z$ random matrix Λ with ν_0 degrees of freedom and a $Z \times Z$ scale matrix W_0:

$$\mathcal{W}(\Lambda|W_0, \nu_0) = \frac{|\Lambda|^{(\nu_0-D-1)/2}}{B}\exp\left(-\frac{\text{tr}(W_0^{-1}\Lambda)}{2}\right), \tag{2.13}$$

where B is the normalizing constant. There are several parameters in the hyper-priors: $\mu_0, \rho_0, \beta_0, W_0, \nu_0, \tilde{W}_0$, and $\tilde{\nu}_0$; These parameters should reflect our prior knowledge about the specific problem and are treated as constants during training. Nevertheless, slightly varying their values usually has little impact on the final prediction performance, as often observed in Bayesian learning. Note that we use the same hyper-parameters for all the factors for convenience, while in fact different priors can be used for different factors if appropriate or necessary. In our experiment we set $\mu_0 = 0$, $\beta_0 = 1$, $W_0 = \mathbf{I}$, $\nu_0 = D$, $\tilde{W}_0 = 1$, $\tilde{\nu}_0 = 1$. For notational convenience, we aggregate the parameters as $\Theta_0 \equiv \{\mu_0, \beta_0, W_0, \nu_0, \tilde{W}_0\,\tilde{\nu}_0\}$ and $\Theta \equiv \{\Theta_U, \Theta_V, \Theta_T, \Theta_0\}$.

2.4.2 Learning by Markov Chain Monte Carlo

The predictive distribution (2.12) involves a multi-dimensional integral that cannot be computed analytically. We thus resort to approximation techniques. The main idea is to view Eq. (2.12) as an *expectation* of $p\left(\hat{x}_{ij}^k|\mathbf{u}_i, \mathbf{v}_j, \mathbf{t}_k, \alpha\right)$ over the posterior distribution $p\left(\mathbf{U}, \mathbf{V}, \mathbf{T}, \alpha, \Theta_U, \Theta_V, \Theta_T|\mathbf{X}\right)$, and approximate the expectation by an average of samples drawn from the posterior distribution. Since the posterior is too complex to directly sample from, we apply a widely-used indirect sampling technique, Markov Chain Monte Carlo (MCMC) [66, 116, 117]. The method works by drawing a sequence of samples from some *proposal distribution* such that each sample depends only on the previous one, thus forming a Markov chain. When the sampling step obeys certain properties, the most notably being *detailed balance*, the chain converges to the desired distribution. Then we collect a number of samples and approximate the integral in Eq. (2.12) by

$$p(\hat{x}_{ij}^k|\mathbf{X}) \approx \sum_{l=1}^{L} p(\hat{x}_{ij}^k|\mathbf{u}_i^{(l)}, \mathbf{v}_j^{(l)}, \mathbf{t}_k^{(l)}, \alpha^{(l)}), \tag{2.14}$$

where L denotes the number of samples collected and $\mathbf{u}_i^{(l)}, \mathbf{v}_j^{(l)}, \mathbf{t}_k^{(l)}$, and $\alpha^{(l)}$ are the lth samples. A detailed description of MCMC can be found in [139].

There are quite a few different flavors of MCMC. Here we choose to use the Gibbs sampling paradigm [56]. In Gibbs sampling, the target random variables are decomposed into several disjoint subsets or blocks, and at each iteration a block of random variables is sampled while all the others are fixed. All the blocks are iteratively sampled until convergence. Such a scheme is very similar to the *nonlinear Gauss-Seidel method* (Chapter 2.7, [10]) for nonlinear optimization, which optimizes iteratively over blocks of variables.

As indicated by its parametrization, our target distribution $p\left(\mathbf{U}, \mathbf{V}, \mathbf{T}, \alpha, \Theta_U, \Theta_V, \Theta_T | \mathbf{X}\right)$ has an inherent block structure of the random variables. In the appendix of this chapter we show that such a block structure, together with our choice of model components in Section 2.4.1, gives rise to conditional distributions that are easy to sample from, leading to an efficient Gibbs sampling procedure as outlined in Algorithm 2. It has two notable features: 1) the only distributions that need to be sampled are multivariate Gaussian distributions and the Wishart distribution; 2) individual user feature vectors can be sampled in parallel, and so can individual item vectors.

Algorithm 2 Gibbs sampling for BPTF

```
Initialize model parameters $\{\mathbf{U}^{(1)}, \mathbf{V}^{(1)}, \mathbf{T}^{(1)}\}$.
For $l$=1, ..., $L$,
    • Sample the hyper-parameters $\alpha^{(l)}, \Theta_U^{(l)}, \Theta_V^{(l)}, \Theta_T^{(l)}$ according to (2.16), (2.17), (2.18) and
      (2.19), respectively.
    • For $i = 1, \ldots, M$, sample the user factors $\{\mathbf{u}_i\}$ (in parallel) according to (2.20).
    • For $j = 1, \ldots, D$, sample the item factors $\{\mathbf{v}_j\}$ (in parallel) according to (2.21).
    • For $k = 1, \ldots, K$, sample the time factors $\{\mathbf{t}_k\}$ according to (2.22).
```

2.4.3 Scalability and Practical Issues

In our implementation, the PTF model is optimized using *alternating least squares*, which is a block coordinate descent algorithm that optimizes one user or one item at each time. The BPTF model is learned using Gibbs sampling as described in Algorithm 2. Both of them are efficient and scalable for large data sets.

Let #nz be the number of observed ratings in the training data. For each iteration, the time complexity for both PTF and BPTF is $O(\#\text{nz} \times Z^2 + (M + D + K) \times Z^3)$. Typically, the term $(\#\text{nz} \times Z^2)$ is much larger than the others so the complexity grows linearly with respect to the number of observations. For the choice of Z, in our experience using tens of latent features usually achieves a good balance between speed and accuracy. Inevitably, the running time of BPTF is slower than the non-Bayesian PMF, which has a complexity of $O(\#\text{nz} \times Z)$ for each iteration using stochastic gradient descent. But using PMF involves a model selection problem. Typically parameters λ_U and λ_V have to be tuned along with the early-stopping strategy. This process can be prohibitive for large data sets. On the other hand, BPTF eliminates the existence of hyper-parameters by introducing priors for them. Therefore, we can set the priors according to our knowledge and let the algorithm adapt them to the data. Empirically, good results can be obtained without any tuning.

When using MCMC, a typical issue is the convergence of sampling. Theoretically, the results generated are only accurate when the chain has reached its equilibrium. This however would usually take a long time and there is no effective way to diagnose the convergence. To alleviate this, we use the MAP result from PMF to initialize the sampling. Then the chain usually converges within a few hundreds samples from our experience. Moreover, we found that the accuracy increases monotonically as the number of samples increases. Therefore in practice we can just monitor the performance on validation sets and stop sampling when the improvement from more samples is diminishing.

2.5 Related Work

There is a lot of work on factorization methods for collaborative filtering, among which the most well-known one is *Singular Value Decomposition* (SVD), which is also called *Latent Semantic Analysis* (LSA) in the language and information retrieval communities. Based on the LSA, *probabilistic LSA* [72] was proposed to provide the probabilistic modeling, and further *latent Dirichlet allocation* (LDA) [14] provides a Bayesian treatment of the generative process. Along another direction, methods like [28, 138, 142] improve the SVD using more sophisticated factorization.

Bayesian PMF (BPMF) [142] provides a Bayesian treatment for PMF to achieve automatic model complexity control. It demonstrates the effectiveness and efficiency of Bayesian methods and MCMC in real-world large-scale data mining tasks, and inspired our research. However, as mentioned before, BPMF is a static model that cannot handle evolving data. BPTF enhance it by adapting the latent features to include the time information. From the algorithmic perspective, BPTF extends BPMF so that it can deal with multi-dimensional tensor data and the time dimension is specially taken care of. Although BPTF gives more flexibility over BPMF, the increase of parameters is negligible considering that the number of time slices are often much smaller than the number of entities. Another difference is that BPMF leaves the observation precision α as a tuning parameter while our Bayesian treatment covers all the parameters. There are also other probabilistic tensor factorizations such as Multi-HDP [132], Probabilistic Non-negative Tensor Factorization [144], and Probabilistic polyadic factorization [30]. Yet they are neither designed for prediction purpose nor modeling temporal effects.

Temporal modeling has been largely neglected in the collaborative filtering community until Koren [85] proposed their award winning algorithm timeSVD++. The timeSVD++ method assumes that the latent features consist of some components that are evolving *linearly* over time and some others that are dedicated bias for each user at each specific time period. This model can effectively capture local changes of user preferences (*i.e.* each user is involving independently) which the authors claim to be vital for improving the performance. On the other hand, BPTF tries to capture the global effect of time that are shared among all users and items. For our sales prediction purpose we argue that modeling the evolution of the overall market would be more effective since the behavior of retailers are not very localized and the data is very sparse.

Real data sets are rarely stationary. Recently, several algorithms aimed at learning the evolution of data were proposed. Tong *et al.* [160] proposed an online algorithm to efficiently compute the proximity in a series of evolving bipartite graphs. Ahmed and Xing [2] added dynamic components to the LDA to track the evolution of topics in a text corpus. Sarkar *et al.* [143] considers the dynamic graph embedding problem and uses *Kalman Filter* to track the embedding coordinates through time. All these works reveal the dynamic nature of various problems.

2.6 Experiments

We conducted several experiments on three real world data sets to test the effectiveness of BPTF. In these data sets, a timestamp is available for each rating, which can thus be denoted by the tuple $(u_i, v_j, t_k, x_{ij}^k)$. The experimental domains include sales prediction and online movie recommendation.

For comparison, we also implemented and report the performance of PMF and BPMF. When training the non-temporal models, the time information is dropped so the actual tuple used is (u_i, v_j, x_{ij}^k). For PMF, stochastic gradient descent with a fixed learning rate ($lrate$) is adopted for training, and its parameters are obtained by hand tuning to achieve the best accuracy. For BPMF and BPTF, Gibbs sampling is used for training and the results from PMF are used to initialize the sampling. Similar to [142], parameters for Bayesian methods are set according to prior knowledge without tuning. Unless indicated otherwise, parameters used for priors are $\mu_0 = 0, \nu_0 = D, \beta_0 = 1, W_0 = \mathbf{I}, \rho_0 = \mathbf{1}, \tilde{\nu}_0 = 1$, where $\mathbf{1} \in \mathbb{R}^Z$ is a vector of 1's.

The algorithms are implemented in MATLAB with embedded *C* functions.

2.6.1 Sales Prediction

In this section we evaluate the performance of BPTF on a sales prediction task for ECCO$^{®}$, a shoe company selling thousands of kinds of shoes to thousands of retailer customers from all over the world. For the consistency of expression we still use “user” to represent “customer” and “item” to represent ECCO’s product: shoes.

ECCO sells its shoes in two seasons each year. Here we use “2008.1” to denote the spring season of 2008 and “2006.2” for the fall season of 2006. For each season there is a period for accepting orders. Suppose we are in the middle of current ordering period, our problem is: in the following part of this season, how many orders of an item can be expected from a particular user? The data we have is only the existing sales record. No attributes for the items or users are available. As mentioned in section 2.1, this is a data set characterized by changing preferences and the fast emergence and disappearance of entities. On average we have thousands of items and users with only 2% of the possible entries observed. Moreover, in each season $75 - 80\%$ of the items and around 20% of the users are new arrivals compared to the last corresponding season. All these characteristics render it a particular challenging problem for collaborative filtering.

The data specification is as follows. We have the sales record from years 2005 to 2008 so there are 4 spring seasons and 4 fall seasons, which are handled separately. For each season, we select a week as the cut-off point so that orders before this week will be used for training and the rest are for testing. For example, if we want to predict for orders of season 2008.1 after week 40 of 2007 (the cut-off point), then the training data will be orders in seasons {2005.1, 2006.1, 2007.1, 2008.1} that happened before the cut-off and the testing data are orders of 2008.1 after the cut-off. We use a single cut-off point for all spring seasons and another one for fall seasons. The resulting test set contains $15 - 20\%$ of the orders. Note that this choice is arbitrary in the sense that the

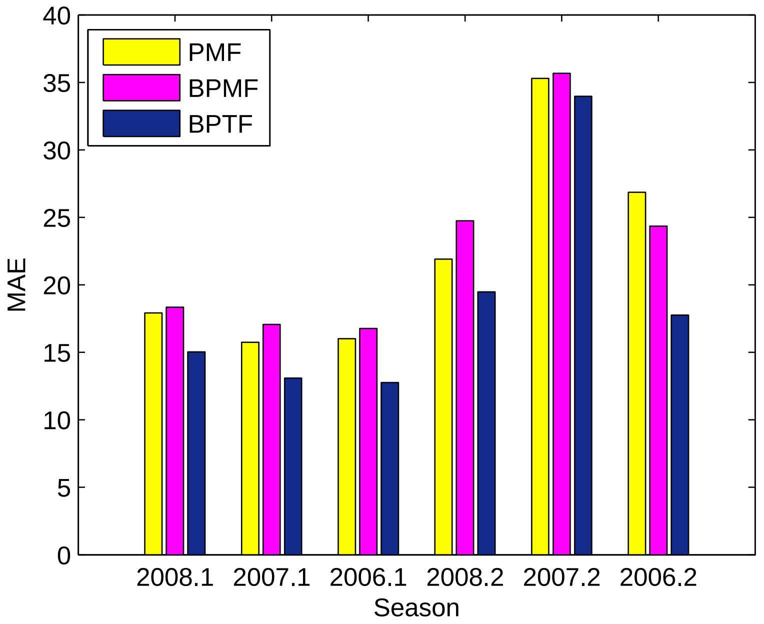


Figure 2.3: Performance comparison of PMF, BPMF, and BPTF on 6 seasons of ECCO's sales data. BPTF outperforms others by a large margin. See text for details.

progress of the sales varies from season to season. We measure the performance of algorithms using *mean absolute error* (MAE) for each order since it is the most relevant quantity for ECCO.

We observed that the within-season variability of data is much larger than the cross-season one. This means that trends like "Customers tend to order formal shoes early and golf shoes late" are strong. Therefore, we assign the timestamp of each order according to the cut-off week so that the latent factors can evolve within seasons. Concretely, every season is divided into *early season* and *late season* by the cut-off week, resulting in two time slices. Note that the data are not grouped by seasons, and all the test data are in the *late season* slice. For each test tuple, we use the time factor for the late season to make the prediction.

We test the performance of three algorithms on all the seasons except 2005.1 and 2005.2 since they do not have previous seasons. The parameters are $\lambda_U = \lambda_V = 0.1, lrate = 1 \times 10^{-5}$ for PMF, $\alpha = 0.04$ for BPMF, and $\tilde{W}_0 = 0.04$ for BPTF. BPMF and BPTF both use the same initialization from PMF. 50 samples are generated in sampling when the accuracy stabilizes.

The prediction accuracies are reported in Figure 2.3. We conclude that our prediction has an average error of 20 pairs for each order, and the accuracy for spring seasons are much lower than fall. For all the seasons BPTF consistently outperforms the static methods by a fairly large margin. Note that PMF appears better than BPMF here. The reason may be that for this moderately sized problem we are able tune the PMF parameters to get the best results, while for BPMF and BPTF we assign the parameters by prior knowledge. The results verify that we can enhance the prediction by modeling the temporal effects of data using BPTF.

	PMF	BPMF	BPTF
RMSE	0.9166	0.9083	0.9044

Table 2.1: RMSE of PMF, BPMF and BPTF on Netflix data.

2.6.2 Movie Rating Prediction

To make the comparison more transparent, we also did experiments on benchmark movie rating problems: Netflix[2] and MovieLens[3]. These large-scale data sets consist of users' ratings to various movies on a 5-star scale, and our task is to predict the rating for new user-movie pairs.

To measure the accuracy, we adopt the *root mean squared error* (RMSE) criterion as commonly used in collaborative filtering literature and the Netflix Prize. For all models, raw user ratings are used as the input. Prediction results are clipped to fit between $[1, 5]$.

Netflix

The Netflix data set contains $100, 480, 507$ ratings from $M = 480, 189$ users to $D = 17, 770$ movies between 1999 and 2005. Among these ratings, $1, 408, 395$ are selected uniformly over the users as the *probe* set for validation. Time information is provided in days. The ratio of observed ratings to all entries of the rating matrix is 1.16%. As a baseline, the score of Netflix's *Cinematch* system is RMSE $= 0.9514$.

Basically the timestamps we used for BPTF correspond to calendar months. However, since the ratings in the early months are much more scarce than that in the later months, we aggregated several earlier months together so that every time slice contains an approximately equal number of ratings. In practice we found that in a fairly large range, the slicing of time does not affect the performance much. In the end, we have $K = 27$ time slices for the entire data set.

Following the settings in the BPMF paper [142], we use $Z = 30$ latent features to model each entity and set $\lambda_U = \lambda_V = 0.015, lrate = 0.001$ for PMF, $\alpha = 2$ for BPMF, and $\tilde{W}_0 = 2$ for BPTF. These parameters for Bayesian methods are set as constant based on prior knowledge and not tuned for best accuracy. 100 samples are used to generate the final prediction.

The prediction accuracies of PMF, BPMF, and BPTF on the probe set are presented in Table 2.1. Figure 2.4 shows the change of accuracies as the number of sample increases. BPMF shows a large improvement over its non-Bayesian ancestor PMF, and BPTF further provides a steady increment in accuracy. However, BPTF does not beat the RMSE $= 0.8891$ result of 20-dimensional timeSVD++ (quoted from their paper), which is the state-of-the-art temporal model for the Netflix. As pointed out by the authors of timeSVD++, the most important trait of the Netflix data is that there are many local changes of preference which could just affect one user in one day. BPTF on the other hand aims at learning the global evolution thus cannot capture these changes. However, modeling the global changes still gives us improved performance.

To generate one sample, BPTF with $Z = 30$ latent features took about 9 minutes using about

[2] http://archive.ics.uci.edu/ml/datasets/Netflix+Prize

[3] http://www.grouplens.org/node/73

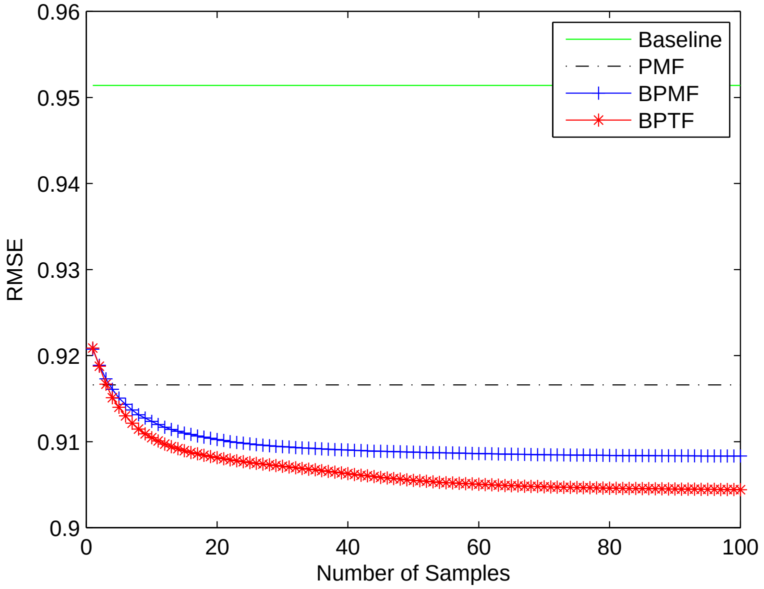


Figure 2.4: Convergence curves of BPTF and BPMF on the full Netflix data. As the number of samples increase, the RMSE of Bayesian methods drop monotonically. The RMSE of the Netflix's baseline and PMF are also presented.

5GB RAM. For comparison, BPMF uses 6 minutes for one sample. We ran our experiments in a single-threaded MATLAB process on a 2.4 GHz AMD Opteron CPU with 64 GB RAM. We did not use the parallel implementation because it involves distributing a large amount of data and the computational model provided by MATLAB does not handle it well. However, since each user and movie latent feature vector can be sampled independently, we believe that on more sophisticated platforms, BPTF can work nicely with *MapReduce*-style parallel processing.

We also did a group of experiments on a subset of the Netflix data constructed by randomly selecting 20% of the users and 20% of the movies. It consists of $M = 95,992$ users, $D = 3,565$ movies, and $4,167,600$ ratings. This subset is further divided into training and testing sets by randomly selecting 10 ratings (or $1/3$ of the total ratings, whichever is smaller) from each user as the testing set. This sampling strategy is similar to the way that the Netflix Prize did it. Finally the new data set contains about 4% of the original data set and is thus suitable for detailed experimental analysis. In the training process, parameters are $\lambda_U = \lambda_V = 0.03, lrate = 0.001$ for PMF, and for Bayesian methods the same parameters as for full data are adopted.

Firstly, we investigate the performance of algorithms as the number of factors varies. For dimensions 10, 20, 50, and 100, the curves of convergence are shown in Figure 2.5. The RMSE steadily decreases as the number of factors increase, and no over-fitting is observed. When using 100 factors, there are on average two parameters for a single rating. This clearly shows the effect of model averaging using Bayesian technique. Also by comparing the curves of BPTF and BPMF, we see that BPTF with 20 factors performs similarly to BPMF with 100 factors. This demonstrates the advantage of temporal modeling considering that the number of parameters in BPMF is about 5 times more than BPTF.

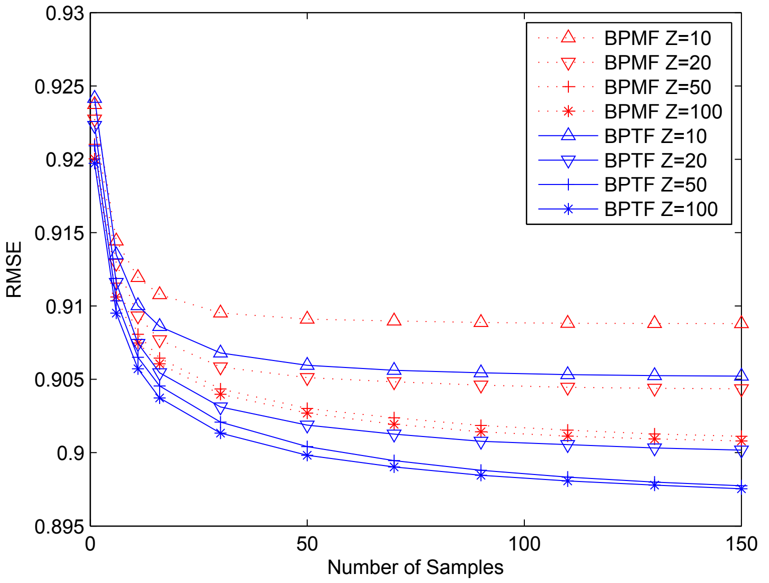


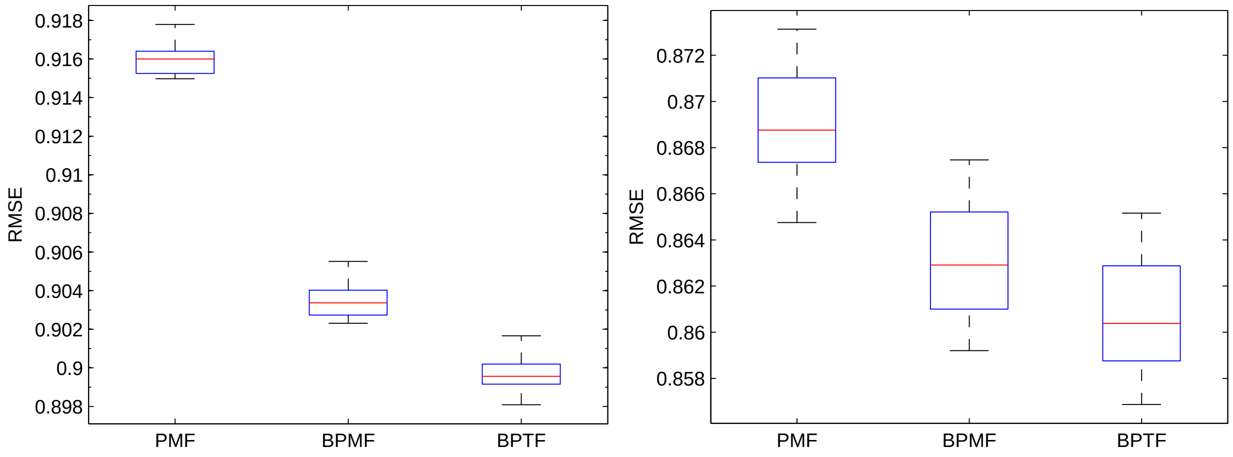
Figure 2.5: Convergence curves of BPMF and BPTF with different number of factors on a subset of Netflix data. The accuracy increases when more factors are used, and no over-fitting is observed. Also, BPTF with 20 factors achieves similar performance as BPMF with 100 factors.

We further examine the significance of the improvement of BPTF over the BPMF by repeating the prediction tasks 20 times using different random test sets. The resulting *box plot* of RMSEs are shown in figure 2.6a. The p-value of *paired t-test* between the results of BPMF and BPTF is 1.3×10^{-22}. In fact, in all runs, BPTF always produce better results than BPMF.

MovieLens

The MovieLens data set contains $1,000,209$ movie ratings from $M = 6,040$ users and $D = 3,706$ movies between April, 2000 and February, 2003, with the restriction that each user has at least 20 ratings. The ratio of observed ratings is round 4.5%. Time information is provided in seconds. We randomly select 10 ratings from each user as the test set, which is roughly 6.5% as large as the training set. The timestamp used for BPTF corresponds to calendar months. We also use $Z = 30$ latent features here. The parameters for PMF are $\lambda_U = \lambda_V = 0.05, lrate = 0.001$ as in [31], and the parameters for Bayesian methods are the same as for Netflix.

Figure 2.6b shows the performance of three algorithms from 20 random runs. This result is similar to what we have for the Netflix data. BPTF still consistently outperforms BPMF, and the p-value of *paired t-test* between them is 8.9×10^{-15}.



(a) (b)

Figure 2.6: RMSE of PMF, BPMF, and BPTF. (a) On a subset of Netflix data. (b) On MovieLens data. Lower RMSE is better.

2.6.3 Music Recommendation

We also test the algorithms' performances on the recently released *Yahoo Music*[4] data sets. This data set is very similar to the previous movie rating data sets, except that instead of movies we are dealing with music records. It contains $1,000,990$ users, $624,961$ music records, and $252,800,275$ ratings on $3,974$ different days. The ratings are given on the scale between 0 and 100.

A subset of this data set is used. As in the Netflix experiment, we first randomly sample 20% users and music from the data set, and then remove users that has less than 50 ratings. The resulting data set contains $38,685$ users, $24,502$ music, and $6,798,119$ ratings. In this subset, only 0.7% of the ratings are observed. The time index of ratings are discretized into 30 equally sized bins. The ratings are scaled to the range $[1,5]$, the same as the movie recommendation problems.

To evaluate the performance, in each round we randomly select 10 ratings from each user to construct the training set. Performances of 20 random runs are reported in Figure 2.7. BPTF still consistently performs the best, showing the advantage of temporal models. We can also see that the non-Bayesian PMF model exhibits clear over-fitting behaviors when the number of factors is large.

2.7 Summary

We present the Bayesian Probabilistic Tensor Factorization (BPTF) algorithm for modeling temporally evolving data. By introducing a set of additional time features to traditional factorization algorithms, and imposing a smoothness constraint on those factors, BPTF is able to learn the global evolution of latent factors. An efficient MCMC procedure is proposed to realize automatic model averaging and largely eliminates the need for tuning parameters on large-scale data. We show

[4] http://kddcup.yahoo.com/datasets.php

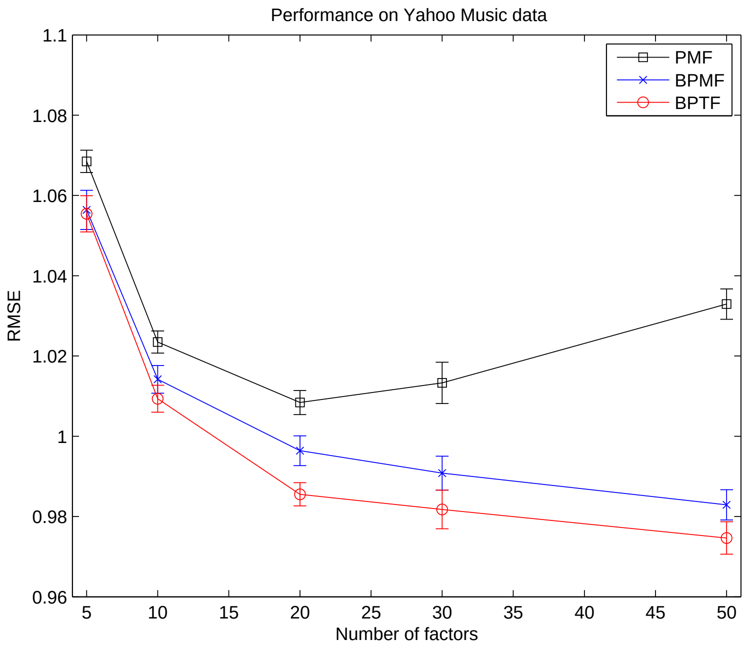


Figure 2.7: Prediction RMSE on the Yahoo Music data set using different methods and different number of latent factors.

extensive empirical results on several real-world data sets to illustrate the advantage of temporal model over static models.

In future works, we may adopt other types of observational models such as the *exponential family* distributions. Gaussian model has been extensively used for rating data and proved to be very effective. However, it might be better to use transformations [141] or other distributions to handle the ratings that are discrete and have limited support. Similarly for the sale prediction problem, a Poisson model might be better suited. However, these changes may lead to more complicated posterior distributions. We can then consider the more general *Metropolis-Hastings* sampling techniques such as [134].

Detailed Derivation

In this section we give explicit forms for the conditional distributions used in Algorithm 2. According to our model assumption in Figure 2.2, the joint posterior distribution can be factorized as

$$\begin{aligned} & p(\mathbf{U}, \mathbf{V}, \mathbf{T}, \alpha, \Theta_U, \Theta_V, \Theta_T | \mathbf{X}) \\ \propto\; & p(\mathbf{X}|\mathbf{U}, \mathbf{V}, \mathbf{T}, \alpha) p(\mathbf{U}|\Theta_U) p(\mathbf{V}|\Theta_V) p(\mathbf{T}|\Theta_T) p(\Theta_U) p(\Theta_V) p(\Theta_T) p(\alpha). \end{aligned} \tag{2.15}$$

By plugging into Eq. (2.15) all the model components described in Section 2.4.1 and carrying out proper marginalization, we derive the desired conditional distributions in the following two subsections.

2.7.1 Hyper-parameters

By using the conjugate prior for the rating precision α, we have that the conditional distribution of α given $\mathbf{X}$, $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{T}$ follows the Wishart distribution:

$$
\begin{aligned}
p(\alpha|\mathbf{X},\mathbf{U},\mathbf{V},\mathbf{T}) &= \mathcal{W}(\alpha|W_0^*,\nu_0^*), \qquad (2.16)\\
\nu_0^* &= \tilde{\nu}_0 + \sum_{k=1}^{K}\sum_{i=1}^{M}\sum_{j=1}^{D} I_{ij}^k,\\
(\tilde{W}_0^*)^{-1} &= \tilde{W}_0^{-1} + \sum_{k=1}^{K}\sum_{i=1}^{M}\sum_{j=1}^{D} I_{ij}^k \left(x_{ij}^k - <\mathbf{u}_i, \mathbf{v}_j, \mathbf{t}_k>\right)^2.
\end{aligned}
$$

For $\Theta_U \equiv \{\mu_U, \Lambda_U\}$, our graphical model assumption in Figure 2.2 suggests that it is conditionally independent of all the other parameters given $\mathbf{U}$. We thus integrate out all the random variables in Eq. (2.15) except $\mathbf{U}$ and obtain the Gaussian-Wishart distribution:

$$
\begin{aligned}
p(\Theta_U|\mathbf{U}) &= \mathcal{N}(\mu_U|\mu_0^*, (\beta_0^*\Lambda_U)^{-1})\mathcal{W}(\Lambda_U|W_0^*,\nu_0^*), \qquad (2.17)\\
\mu_0^* &= \frac{\beta_0\mu_0 + M\bar{U}}{\beta_0 + M}, \qquad \beta_0^* = \beta_0 + M, \qquad \nu_0^* = \nu_0 + M;\\
(W_0^*)^{-1} &= W_0^{-1} + M\bar{S} + \frac{\beta_0 M}{\beta_0 + M}(\mu_0 - \bar{U})(\mu_0 - \bar{U})^T,\\
\bar{U} &= \frac{1}{M}\sum_{i=1}^{M}\mathbf{u}_i, \qquad \bar{S} = \frac{1}{M}\sum_{i=1}^{M}(\mathbf{u}_i - \bar{U})(\mathbf{u}_i - \bar{U})^T.
\end{aligned}
$$

Similarly, $\Theta_V \equiv \{\mu_V, \Lambda_V\}$ is conditionally independent of all the other parameters given $\mathbf{V}$, and its conditional distribution has the same form:

$$
\begin{aligned}
p(\Theta_V|\mathbf{V}) &= \mathcal{N}(\mu_V|\mu_0^*, (\beta_0^*\Lambda_V)^{-1})\mathcal{W}(\Lambda_V|W_0^*,\nu_0^*), \qquad (2.18)\\
\mu_0^* &= \frac{\beta_0\mu_0 + D\bar{V}}{\beta_0 + D}, \qquad \beta_0^* = \beta_0 + D, \qquad \nu_0^* = \nu_0 + D;\\
(W_0^*)^{-1} &= W_0^{-1} + D\bar{S} + \frac{\beta_0 D}{\beta_0 + D}(\mu_0 - \bar{V})(\mu_0 - \bar{V})^T,\\
\bar{V} &= \frac{1}{D}\sum_{j=1}^{D}\mathbf{v}_j, \qquad \bar{S} = \frac{1}{D}\sum_{j=1}^{D}(\mathbf{v}_j - \bar{V})(\mathbf{v}_j - \bar{V})^T.
\end{aligned}
$$

Finally, $\Theta_T \equiv \{\mu_T, \Lambda_T\}$ is conditionally independent of all other parameters given $\mathbf{T}$, and its conditional distribution also follows Gaussian-Wishart distributions:

$$
\begin{aligned}
p(\Theta_T|\mathbf{T}) &= \mathcal{N}(\mu_T|\mu_0^*, (\beta_0^*\Lambda_U)^{-1})\mathcal{W}(\Lambda_T|W_0^*,\nu_0^*), \qquad (2.19)\\
\mu_0^* &= \frac{\mathbf{t}_1 + \beta_0\rho_0}{\beta_0 + 1}, \qquad \beta_0^* = \beta_0 + 1, \qquad \nu_0^* = \nu_0 + K;\\
(W_0^*)^{-1} &= W_0^{-1} + \sum_{k=2}^{K}(\mathbf{t}_k - \mathbf{t}_{k-1})(\mathbf{t}_k - \mathbf{t}_{k-1})^T + \frac{\beta_0}{1 + \beta_0}(\mathbf{t}_1 - \rho_0)(\mathbf{t}_1 - \rho_0)^T.
\end{aligned}
$$

2.7.2 Model parameters

We first consider the user features $\mathbf{U}$. According to the graphical model in Figure 2.2, its conditional distribution factorizes with respect to individual users:

$$p(\mathbf{U}|\mathbf{X},\mathbf{V},\mathbf{T},\alpha,\Theta)=\prod_{i=1}^{N}p(\mathbf{u}_i|\mathbf{X},\mathbf{V},\mathbf{T},\alpha,\Theta_U).$$

We then have, for each user feature vector,

$$\begin{aligned}
p(\mathbf{u}_i|\mathbf{X},\mathbf{V},\mathbf{T},\alpha,\Theta_U) &= \mathcal{N}(\mathbf{u}_i|\mu_i^*,(\Lambda_i^*)^{-1}), \qquad (2.20)\\
\mu_i^* &\equiv (\Lambda_i^*)^{-1}\Big(\Lambda_U\mu_U+\alpha\sum_{k=1}^{K}\sum_{j=1}^{D}I_{ij}^k x_{ij}^k Q_{jk}\Big),\\
\Lambda_i^* &\equiv \Lambda_U+\alpha\sum_{k=1}^{K}\sum_{j=1}^{D}I_{ij}^k Q_{jk}Q_{jk}^T,
\end{aligned}$$

where $Q_{jk}\equiv\mathbf{v}_j\odot\mathbf{t}_k$ is the element-wise product of $\mathbf{v}_j$ and $\mathbf{t}_k$. For the item features $\mathbf{V}$ the conditional distribution factorizes with respect to individual items, and for each item feature vector we have

$$\begin{aligned}
p(\mathbf{v}_j|\mathbf{X},\mathbf{U},\mathbf{T},\alpha,\Theta_V) &= \mathcal{N}(\mathbf{v}_j|\mu_j^*,(\Lambda_j^*)^{-1}), \qquad (2.21)\\
\mu_j^* &\equiv (\Lambda_j^*)^{-1}\Big(\Lambda_V\mu_V+\alpha\sum_{k=1}^{K}\sum_{i=1}^{M}I_{ij}^k x_{ij}^k P_{ik}\Big),\\
\Lambda_j^* &\equiv \Lambda_V+\alpha\sum_{k=1}^{K}\sum_{i=1}^{M}I_{ij}^k P_{ik}P_{ik}^T,
\end{aligned}$$

where $P_{ik}\equiv\mathbf{u}_i\odot\mathbf{t}_k$.

Regarding the time features, the conditional distribution of $\mathbf{t}_k$ is also a Gaussian distribution:

$$p(\mathbf{t}_k|\mathbf{X},\mathbf{U},\mathbf{V},\mathbf{T}_{-k},\alpha,\Theta_T)=\mathcal{N}(\mathbf{t}_k|\mu_k^*,(\Lambda_k^*)^{-1}), \qquad (2.22)$$

where $\mathbf{T}_{-k}$ denotes all the time feature vectors except $\mathbf{t}_k$. The mean vectors and the precision matrices depend on k in the following way:
For $k=1$,

$$\mu_1^*=\frac{\mathbf{t}_2+\mu_T}{2},\quad \Lambda_1^*=2\Lambda_T+\alpha\sum_{i=1}^{M}\sum_{j=1}^{D}I_{ij}^1 O_{ij}O_{ij}^T,$$

where $O_{ij}\equiv\mathbf{u}_i\odot\mathbf{v}_j$ (the same for the following).
For $2\leqslant k\leqslant K-1$,

$$\begin{aligned}
\mu_k^* &= (\Lambda_k^*)^{-1}\left(\Lambda_T(\mathbf{t}_{k-1}+\mathbf{t}_{k+1})+\alpha\sum_{i=1}^{M}\sum_{j=1}^{D}I_{ij}^k x_{ij}^k O_{ij}\right),\\
\Lambda_k^* &= 2\Lambda_T+\alpha\sum_{i=1}^{M}\sum_{j=1}^{D}I_{ij}^k O_{ij}O_{ij}^T.
\end{aligned}$$

For $k = K$,

$$\mu_K^* = (\Lambda_K^*)^{-1} \left(\Lambda_T \mathbf{t}_{K-1} + \alpha \sum_{i=1}^{M} \sum_{j=1}^{D} I_{ij}^K x_{ij}^K O_{ij} \right),$$

$$\Lambda_K^* = \Lambda_T + \alpha \sum_{i=1}^{M} \sum_{j=1}^{D} I_{ij}^K O_{ij} O_{ij}^T.$$

Chapter 3

Handling Outliers by Robust Factorization

This chapter focuses on the outliers/anomalies in the data and proposes an algorithm to improves robustness of factorization methods.

Matrix factorization methods are extremely useful in many data mining tasks, yet their performances are often degraded by outliers. In order to alleviate the influence of outliers, we directly formulate the robust factorization problem as a matrix approximation problem with constraints on the rank of the matrix and the cardinality of the outlier set. Then, unlike existing methods that resort to relaxations, we solve this problem directly and efficiently. In addition, structural knowledge about the outliers can be incorporated to find outliers more effectively. We applied this method in anomaly detection tasks on various data sets. Empirical results show that this new algorithm is effective in robust modeling and anomaly detection, and our direct solution achieves superior performance over the state-of-the-art methods based on the L_1-norm and the nuclear norm of matrices.

3.1 Introduction

Real world problems almost always involve data that do not conform to the assumptions we made in our models. These data are called outliers or anomalies. These outliers can severely degrade the models' quality and performances, therefore we want robust methods to reduce the impact of outliers. In novelty detection problems, we are also interested in finding and studying these outliers since they might lead to discoveries. To do this, we also need reliable models that are not distorted by outliers.

The definition of outlier varies depending on the application and the behavior of data we want to capture. A popular assumption is that the normal data are close together, and consequently outliers are far away from the others *i.e.* lie in the low-density region of the data distribution [21, 182]. For a survey of the outlier detection field readers can refer to [26]. In this work, we consider another common definition called the *subspace outlier*, which comes from the assumptions that the normal data reside in a low-dimensional linear subspace, which the outliers lie outside of. This means, for example in signal processing, that a normal signal can be reconstructed by a few bases. If a signal cannot be well reconstructed by these bases, it is an outlier. This subspace-based modeling is widely used in various problems such as dimensionality reduction, signal/image processing, time series analysis, and collaborative filtering.

Matrix factorization techniques, such as *principal component analysis* (PCA) and *non-negative matrix factorization* (NMF) [93], are extremely useful in learning subspace structures from data. However, traditional methods are prone to be distorted by outliers [86]. Since factorizations are usually done by minimizing the error made by the model, a popular way of achieving robustness is to use error measurements that are insensitive to outliers. Though being pervasively used, the *mean squared error* or the L_2 error measure is known to be vulnerable to outliers [176]. In machine learning and statistics, the L_1 error measure (mean absolute error) is widely used for the purpose of robustness [15, 22]. Other measures like the *Huber loss* [74] and the *Geman-McClure* function have also been employed [88, 123]. These robust measurements usually increase the algorithms' complexities significantly. Another strategy is to exclude the outliers: we can first guess which data are outliers, and then reduce their influences to the model [86, 176].

The contribution of this work is to propose a novel algorithm for learning robust subspace models based on matrix factorization. For a data matrix $\mathbf{X}$, we assume that it is approximately low-rank, and a small portion of this matrix has been corrupted by some arbitrary outliers. The goal of the proposed algorithm is to get a reliable estimation of the true low-rank structure of this matrix. To achieve this, our basic idea is to exclude the outliers from the model estimation. Specifically, the proposed algorithm directly answers the question: if you are allowed to ignore some data (outliers), what is the best low-rank model you can get?

We formulate this problem as a constrained optimization problem. This formulation aims at minimizing the L_2 error of the low-rank approximation subject to that the number of ignored outliers is small, without any further assumptions. This formulation reflects our direct understanding of outliers and robust estimation. Thus we call it *direct robust matrix factorization* (DRMF).

It can be shown that DRMF is the original problem that the recently popular *nuclear norm* based methods (*e.g.* [22, 177]) are trying to solve. However, unlike these methods that resort to relaxation techniques, we directly form these constraints in terms of the *matrix rank* and the *cardinality of the outlier set*. Despite that matrix rank and set cardinality are often very difficult to handle in optimization, we are able to solve this problem directly in its original form. We observe that better quality results are produced by this direct solution compared to the relaxed methods.

We adopt *block coordinate descent* to solve the DRMF problem. The resulting algorithm is based on existing factorization routines such as the *singular value decomposition* (SVD), and efficient thresholding procedures. Therefore DRMF is simple to implement, efficient, and easy to use. DRMF is also very flexible: we can impose additional constraints on both the factorization (*e.g.* nonnegative factors [93]) and the outliers (*e.g.* outlier columns instead of entries [177]) to incorporate knowledge for better performance.

We applied DRMF to both synthetic and real-world data sets for the purpose of robust modeling and anomaly detection. We compare DRMF to its state-of-the-art competitors based on the nuclear norm and the L_1 error measurement. Based on extensive empirical results we conclude that DRMF is able get better performance than these relaxed methods. In addition, the parameters of DRMF are intuitive and easy to tune, making it a practical tool for robust analysis.

The rest of this chapter is structured as follows. We introduce background and notations in

Section 3.1.1. Section 3.2 describes the proposed algorithm. Related work and discussions are in Section 3.3 and 3.4. Experiments are presented in section 3.5. Finally our conclusions are made.

3.1.1 Background and Notation

Matrices are very useful in representing data. For example, in regression and classification, samples are often organized into a *design matrix* in which each row represents a sample and each column represents a feature. The document-word matrix is often used for text data. In recommendation systems we have the rating matrix. *Connectivity matrices* are widely used to express network and graph data. We denote a data matrix as $\mathbf{X} \in \mathbb{R}^{M \times D}$. $x_{i,j}$ denotes the (i, j)th entry of $\mathbf{X}$. We also use the operator $\mathcal{D}_l(\cdot)$ to return an $l \times l$ diagonal matrix whose diagonal is the input vector.

One of the most common analysis we can do on $\mathbf{X}$ is factorization, as in *principal component analysis* (PCA). We assume that $\mathbf{X}$ has a low rank and can be factorized as

$$\mathbf{X} \approx \mathbf{U}\mathbf{V}^T, \mathbf{U} \in \mathbb{R}^{M \times K}, \mathbf{V} \in \mathbb{R}^{D \times K}, \tag{3.1}$$

where K is the rank of the factorization. For design matrices, factors given by PCA/SVD reveals the linear structure and intrinsic dimensionality of the data. For text data, *latent semantic indexing* (LSI) [72] and *nonnegative matrix factorization* (NMF) [43] is often applied. The low-rank assumption is also useful in *matrix completion* [23, 111] and *collaborative filtering* [138, 141] (See Chapter 2).

In a more general form, low-rank matrix factorization can be written as the following optimization problem

$$\begin{aligned} \min_{\mathbf{L}} \quad & \|\mathbf{X} - \mathbf{L}\|_F \\ \text{s.t.} \quad & \text{rank}(\mathbf{L}) \leq K, \end{aligned} \tag{3.2}$$

where $\|\cdot\|_F$ is the *Frobenius norm*, $\mathbf{L}$ is the low-rank approximation of $\mathbf{X}$, and K is the maximal rank of $\mathbf{L}$.

Singular value decomposition (SVD) is perhaps the most commonly used tool for low-rank analysis. SVD decomposes a matrix into three factors:

$$\mathbf{X} = \mathbf{U}\mathcal{D}(\mathbf{s})\mathbf{V}^T = \sum_{i=1}^{l} s_i \mathbf{u}_i \mathbf{v}_i^T, \tag{3.3}$$

where $l = \min(M, D)$, $\mathbf{s} = [s_1, \ldots, s_l]$ is the vector of $\mathbf{X}$'s singular values in descending order, columns of $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_l] \in \mathbb{R}^{M \times l}$ and $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_l] \in \mathbb{R}^{D \times l}$ are the left and right singular vectors. The significance of SVD is reflected in the following theorem [48]:

Theorem 1 (Eckart-Young)**.** *Let the SVD of* $\mathbf{X}$ *be* (3.3)*. For any* K *with* $0 \leq K \leq \text{rank}(\mathbf{X})$*, let*

$$\hat{\mathbf{L}}_K = \mathbf{U}_{:,1:K}\mathcal{D}(\mathbf{s}_{1:K})\mathbf{V}_{:,1:K}^T = \sum_{i=1}^{K} s_i \mathbf{u}_i \mathbf{v}_i^T, \tag{3.4}$$

then

$$\|\mathbf{X} - \hat{\mathbf{L}}_K\|_F = \min_{\mathrm{rank}(\mathbf{L}) \leq K} \|\mathbf{X} - \mathbf{L}\|_F.$$

In other words, the rank-K truncated SVD approximation $\hat{\mathbf{L}}_K$ is a globally optimal solution to problem (3.2).

From SVD we can derive the *nuclear norm* of matrices. The nuclear norm of matrix $\mathbf{X}$ is defined as $\|\mathbf{X}\|_* = \sum_{i=1}^{l} s_i$ *i.e.* the sum of $\mathbf{X}$'s singular values. The nuclear norm can serve as a convex relaxation of the matrix rank, and has attracted much research interest recently. We shall discuss more in Section 3.3.

Next we consider robust error measurement. Let the error matrix be $\mathbf{E} = \mathbf{X} - \mathbf{L}$. In (3.2), we used the Frobenius norm, $\|\mathbf{E}\|_F = \sqrt{\sum_{i,j} E_{i,j}^2}$ *a.k.a.* the L_2**-norm**, to measure $\mathbf{E}$. The L_2-norm is pervasively used but is known to be sensitive to outliers [86]. A common robust alternative is the L_1**-norm** $\|\mathbf{E}\|_1 = \sum_{i,j} |E_{i,j}|$ [15, 22], in which the errors are not squared so the impact of large errors is reduced. A more aggressive choice is the L_0**-norm**[1] $\|\mathbf{E}\|_0 = \sum_{i,j} I(E_{i,j} \neq 0)$. The L_0-norm only counts the number of errors disregarding their magnitudes.

Recently, *structured norms* become popular in handling problems such as *group lasso* [180] and *multitask learning* [104] with structural knowledge. These norms can also be used to incorporate knowledge about the structure of outliers (*e.g.* when outlier entries in the same row is correlated) [177]. Here we introduce the $L_{2,1}$**-norm** and $L_{2,0}$**-norm**. The $L_{2,1}$-norm $\|\mathbf{E}\|_{1,2} = \sum_{i=1} \|\mathbf{E}_{i,:}\|_2$ is the sum of the L_2-norm of rows of $\mathbf{E}$ (*i.e.* the sum of the lengths of the row vectors), and $L_{2,0}$-norm $\|\mathbf{E}\|_{2,0} = \sum_{i=1} I(\|\mathbf{E}_{i,:}\|_2 \neq 0)$ is the number of non-zero rows in $\mathbf{E}$. These two norms compare similarly as the L_1 and L_2 norms except that errors are measured in groups according to the assumed structure.

3.2 Direct Robust Factorization

We adopt the common assumption that there is only a small amount of outliers in the data matrix $\mathbf{X}$. Then, we define the robust low-rank approximation of $\mathbf{X}$ as the answer to the question: *if you are allowed to ignore some data as outliers, what is the best low-rank approximation?*

A directly formulation of the above question is the following problem *direct robust matrix factorization* (DRMF):

$$\begin{aligned} \min_{\mathbf{L},\mathbf{S}} \quad & \|(\mathbf{X} - \mathbf{S}) - \mathbf{L}\|_F \\ \text{s.t.} \quad & \mathrm{rank}(\mathbf{L}) \leq K \\ & \|\mathbf{S}\|_0 \leq e, \end{aligned} \tag{3.5}$$

where $\mathbf{L}$ is the low-rank approximation as before, K is the rank, $\mathbf{S}$ is the matrix of *outliers*, and e is the maximal number of non-zeros entries in $\mathbf{S}$ *i.e.* the maximal number of entries that can be ignored as outliers. Comparing DRMF to the regular problem (3.2), we can see that the only difference is that we allow the outliers $\mathbf{S}$ to be excluded from the low-rank approximation, as long

[1] Rigorously this L_0 measurement is not a norm.

as the number of outliers is not too large *i.e.* $\mathbf{S}$ is sufficiently *sparse*. Note that we do not need the actual number of outliers. Instead, we only use e to put an upper limit on it.

By excluding the outliers from the low-rank approximation, we can ensure the reliability of the estimated low-rank structure. On the other hand, the number of outliers is constrained so that the estimation is still faithful to the data. DRMF is advantageous over existing methods in its simplicity and directness: no special robust error measurement is introduced, nor do we make assumptions about the outliers beyond necessity. In fact, several state-of-the-art methods are relaxed versions of DRMF, as we shall discuss in section 3.3.

3.2.1 DRMF Algorithm

Usually, optimization problems involving the *rank* or the L_0-norm *i.e. set cardinality* are difficult to solve. Nevertheless, the DRMF problem admits a simple solution due to its decomposable structure *w.r.t.* variables $\mathbf{L}$ and $\mathbf{S}$. To take advantage of this property, we adopt the *block coordinate descent* strategy, and the resulting algorithm is described in Algorithm 3: We first fix $\mathbf{S}$ the current estimate of outliers, exclude them from $\mathbf{X}$ to get the "clean" data $\mathbf{C}$, and fit $\mathbf{L}$ based on $\mathbf{C}$. Then, we update the outliers $\mathbf{S}$ based on the error $\mathbf{E} = \mathbf{X} - \mathbf{L}$.

Algorithm 3 Direct Robust Matrix Factorization (DRMF)

```
1. Input:
    • $\mathbf{X}$ the data matrix.
    • $K$ the maximal rank of the factorization.
    • $e$ the maximal number of outliers.
    • $\mathbf{S}$ the initial outliers.
2. While not converged:
    (a) Solve the factorization problem:
            $\mathbf{L} = \arg\min_{\mathbf{L}} \|\mathbf{C} - \mathbf{L}\|_F, \mathbf{C} = \mathbf{X} - \mathbf{S}$        (3.6)
            s.t. $\mathrm{rank}(\mathbf{L}) \leq K$
    (b) Solve the outlier detection problem:
            $\mathbf{S} = \arg\min_{\mathbf{S}} \|\mathbf{E} - \mathbf{S}\|_F, \mathbf{E} = \mathbf{X} - \mathbf{L}$        (3.7)
            s.t. $\|\mathbf{S}\|_0 \leq e$
3. Output:
    • $\mathbf{L}$ the robust low-rank approximation.
    • $\mathbf{S}$ the outliers.
```

It is easy to see that the solution to the low-rank approximation problem (3.6) is directly given by SVD according to Theorem 1. Therefore, the solution to $\mathbf{L}$ is simply the truncated SVD approximation to $\mathbf{C}$ given in (3.4), which can be obtained efficiently. Since only the first K singular vectors are required, we can further accelerate the computation using *partial SVD* algorithms such as PROPACK [91].

The outlier detection problem (3.7) can also be solved efficiently. To solve the general problems of L_0-norm constrained minimization of decomposable objectives, we give the following theorem which extends the work of [107]:

Theorem 2. *Let $\mathcal{A}$ be a domain with $0 \in \mathcal{A}$; $A = \{a_1, \ldots, a_n\} \in \mathcal{A}^n$; $\{f_i | f_i : \mathcal{A} \to \mathbb{R}, i = 1, \ldots, n\}$ be a set of n functions mapping from $\mathcal{A}$'s elements to real numbers. Also, let $\hat{a}_i = \arg\min_{a_i} f_i(a_i)$; $b_i = f_i(0) - f_i(\hat{a}_i) \geq 0$; $\|A\|_0$ be the number of non-zero elements in A; e be a positive integer. Then for the following problem*

$$\begin{aligned} \min_{A} \quad & f(A) = \sum_{i=1}^{n} f_i(a_i) \\ \textit{s.t.} \quad & \|A\|_0 \leq e, \end{aligned} \tag{3.8}$$

an optimal solutions is given by $A^* = \{a_1^*, \ldots, a_n^*\}$ with

$$a_i^* = \begin{cases} \hat{a}_i & b_i \geq b_{(e)} \\ 0 & \textit{otherwise} \end{cases}$$

where $b_{(e)}$ is the eth largest element in $\{b_i\}_{i=1,\ldots,n}$.

Proof. This theorem is a slight generalization of [107] and can be derived similarly. Denote index set $J = \{j | b_j \geq b_{(e)}\}$. Clearly, A^* is a feasible solution to (3.8) since $\|A^*\|_0 \leq |J| = e$. For any other feasible A' with $\|A'\|_0 \leq e$ and $A' \neq A^*$, we denote index set $J' = \{j | a_j' \neq 0\}$, $|J'| \leq e$, and $\bar{J}$ be the complement of J. Then

$$\begin{aligned} & f(A') - f(A^*) \\ = & \sum_{j \in J' \cap J} f_j(a_j') - f_j(a_j^*) + \sum_{j \in \bar{J}' \cap \bar{J}} f_j(a_j') - f_j(a_j^*) + \sum_{j \in \bar{J}' \cap J} f_j(a_j') - f_j(a_j^*) + \sum_{j \in J' \cap \bar{J}} f_j(a_j') - f_j(a_j^*) \\ \geq & \sum_{j \in J' \cap J} f_j(a_j^*) - f_j(a_j^*) + \sum_{j \in \bar{J}' \cap \bar{J}} f_j(0) - f_j(0) + \sum_{j \in \bar{J}' \cap J} f_j(0) - f_j(a_j^*) + \sum_{j \in J' \cap \bar{J}} f_j(a_j') - f_j(0) \\ \geq & \sum_{j \in \bar{J}' \cap J} f_j(0) - f_j(\hat{a}_j) - \sum_{j \in J' \cap \bar{J}} f_j(0) - f_j(\hat{a}_j) \\ \geq & \sum_{j \in \bar{J}' \cap J} b_j - \sum_{j \in J' \cap \bar{J}} b_j \geq |\bar{J}' \cap J| b_{(e)} - |J' \cap \bar{J}| b_{(e)} \\ = & \left((|J| - |J' \cap J|) - (|J'| - |J' \cap J|) \right) b_{(e)} \\ \geq & 0 \end{aligned}$$

Therefore, for any feasible A', we have$f(A') \geq f(A^*)$. □

Based on Theorem 2, problem (3.7) can easily be solved by letting $a_{i,j} = S_{i,j}$ and $f_{i,j}(S_{i,j}) = (S_{i,j} - E_{i,j})^2$. Specifically, the solution is

$$S_{i,j} = \begin{cases} E_{i,j} & b_{i,j} \geq b_{(e)} \\ 0 & \text{otherwise.} \end{cases} \tag{3.9}$$

where $b_{i,j} = E_{i,j}^2$ and $b_{(e)}$ is the eth largest value in $\{b_{i,j}\}_{i=1,\ldots,M,j=1,\ldots,D}$. This result is very intuitive: in each round, large errors are considered outliers, and are put into $\mathbf{S}$ to be excluded from the low-rank fitting in the next round.

The above results give the global optima to step (a) and (b) in Algorithm 3. They are guaranteed to improve the objective value within the feasible region, and thus the algorithm is going to converge. In each iteration, we do one rank-K partial SVD plus one $\frac{e}{MD}$ quantile computation, so the total complexity is $O(MD(K + \log(e)))$. Since K and e are usually fixed and small, DRMF can handle large-scale problems.

The DRMF problem (3.5) is not convex due to the constraints on the rank of $\mathbf{L}$ and the L_0-norm of $\mathbf{S}$. Therefore, local minima exist depending on the starting point. This fact is reflected in that the algorithm starts with an initial guess of outliers. However, in experiments we found that DRMF is quite stable *w.r.t.* starting point, and good initialization methods exist. More details can be found in Section 3.4.

DRMF has two parameters K and e that need the user's attention. Yet, their clear meanings (the rank and the maximally allowed number of outliers) help the user select their values. Particularly, we emphasize that the value of e does not need to match the actual number of outliers. It is only used as a safeguard to ensure that not too many data are regarded as outliers. For this purpose we can easily set e to be say 5% of the whole data set. From Eq. (3.9) we can see that normal data with small factorization errors will not be thrown as outliers. On the other hand, if there are more than 5% outliers, the ones with largest errors will be taken care of. We will show that this default behavior gives us good performance in various situations in Section 3.5.

3.3 Related Work

Matrix factorization is widely used in data mining and machine learning, and robust subspace analysis methods are of great value in practical situations. Many robust estimators has been proposed (*e.g.* [67, 79, 86, 88, 99]). They usually involves alternative error measurements, complex estimation procedures, or problem specific heuristics. On the other hand, the DRMF algorithm is both conceptually and computationally simple: it excludes some data and fit the rest, and the solution is obtained by iteratively applying SVD and thresholding the errors.

Another limitation of traditional robust methods is that performance cannot be guaranteed in high dimensions [44, 177]. Recently, constraining the nuclear norm [23, 111] of the matrix instead of its rank becomes a popular strategy for overcoming this problem [22, 177, 185], and has been shown to outperform traditional algorithms. These methods can be summarized as the *nuclear norm minimization* (NNM) problem. To compare, we also rewrite DRMF to one of its equivalent lagrangian form, and show them in Table 3.1.

NNM	$\min_{\mathbf{L},\mathbf{S}} \quad \|\mathbf{L}\|_* + \lambda\|\mathbf{S}\|_1$ $\text{s.t.} \quad \|\mathbf{X}-\mathbf{L}-\mathbf{S}\|_F \leq \sigma$
DRMF	$\min_{\mathbf{L},\mathbf{S}} \quad \mathrm{rank}(\mathbf{L}) + \lambda\|\mathbf{S}\|_0$ $\text{s.t.} \quad \|\mathbf{X}-\mathbf{L}-\mathbf{S}\|_F \leq \sigma$

Table 3.1: Comparing the nuclear norm minimization (NNM) problem and DRMF. $\mathbf{L}$ is low-rank; S is the sparse outlier. $\|\cdot\|_*$ is the nuclear norm; σ is the allowed approximation error.

We can immediately see the relationship between DRMF and the NNM methods: DRMF minimizes the *rank*, while NNM minimizes the *nuclear norm*; DRMF measures outliers by the L_0-norm, while NNM uses the L_1-norm. In fact, the nuclear norm and the L_1-norm in the NNM problem are proposed as convex relaxations of the rank and the L_0-norm in the first place. In this sense, DRMF is the "original problem" that NNM is trying to solve.

By using the relaxations, NNM is convex and the globally optimal solutions can be found. In addition, theories have been provided for choosing λ to guarantee the correct recovery of the principal subspace under certain conditions [22, 177]. Yet, it is unknown how well these relaxations approximate the original problem in general. On the other hand, the original DRMF problem is non-convex and has the local-minima problem. As a remedy, we can initialize DRMF with the NNM results to obtain results that are better than using either NNM or DRMF alone. We expand this point further in Section 3.4. The theoretical properties of DRMF are difficult to analyze due to the non-continuous and non-convex nature of the L_0-norm and the matrix rank. Yet we shall show that DRMF can achieve better empirical performance than the relaxed NNM methods.

The NNM methods often set $\sigma = 0$ for exact recovery [22, 177]. Yet real-world noisy data invalidate this choice and make the algorithm inefficient. When NNM uses $\sigma > 0$ (*e.g.* in [177, 185]), it needs more assumptions to ensure the theoretical soundness and introduces extra parameters (*e.g.* the amount of Gaussian noise) that need careful tuning. On the other hand, DRMF can be applied in both situations, thanks to the fact that it solves the problem in the constrained form (3.5) the only difference between noisy and noiseless data is that the former will have non-zero objective values.

3.4 Discussion

3.4.1 Extensions to Incorporating Prior Knowledge

In many situations, additional knowledge is available for us to find outliers. For example, in a *design matrix*, if one sample point has been corrupted, then it is very likely that most of the entries in its corresponding row are outliers. In collective data, we may also face the situation where if the observation of a group is disrupted, then all of its points are affected. In this case, we should look for outlier rows so that evidences of anomalies can aggregated to enhance the performance. DRMF can easily be extended to handle this situation. Here, we consider the *outlier patterns* to be groups of entries that are anomalous. Instead of counting the number of outlier entries, we can count the number of outlier patterns using structured norms such as the $L_{2,0}$-norm. Concretely, the following DRMF-Row (DRMF-R) problem handles row outliers:

$$\begin{aligned} \min_{\mathbf{L},\mathbf{S}} \quad & \|(\mathbf{X}-\mathbf{S})-\mathbf{L}\|_F \\ \text{s.t.} \quad & \operatorname{rank}(\mathbf{L}) \leq K \\ & \|\mathbf{S}\|_{2,0} \leq e, \end{aligned} \tag{3.10}$$

where e is the maximal number of outlier rows allowed. DRMF-R can be solved by replacing step (b) in Algorithm 3 with the following problem:

$$\begin{aligned} \mathbf{S} = \quad & \arg\min_{\mathbf{S}} \|\mathbf{E}-\mathbf{S}\|_F, \mathbf{E} = \mathbf{X}-\mathbf{L} \\ \text{s.t.} \quad & \|\mathbf{S}\|_{2,0} \leq e. \end{aligned} \tag{3.11}$$

Row-wise outliers has also been considered in *outlier pursuit* (OP) [177]. OP extends the NNM algorithm by using the $L_{2,1}$-norm to capture outlier rows. Not surprisingly, OP is the convex relaxation of the DRMF-R problem (3.10).

Problem (3.11) can also be solved based on Theorem 2 by treating each row of $\mathbf{S}$ as an element. Without giving details, we show that the solutions is:

$$\mathbf{S}_{i,:} = \begin{cases} \mathbf{E}_{i,:} & l_i \geq l_{(e)} \\ \mathbf{0} & \text{otherwise}, \end{cases} \tag{3.12}$$

where $l_i = \|\mathbf{E}_{i,:}\|_F$ and $l_{(e)}$ is the eth largest value among $\{l_i\}_{i=1,\dots,M}$. Again, the solution is obtained efficiently by thresholding. In fact, it is very easy to capture arbitrarily shaped outlier patterns to accommodate specific problems.

Finally, the low-rank component of DRMF can also be extended. For example, we can require the factor matrices in (3.1) to be non-negativity as in *non-negative matrix factorization* (NMF) [43]. To do this, we replace the constraint $\operatorname{rank}(\mathbf{L}) \leq K$ in (3.5) by the explicit factorization form $\mathbf{L} = \mathbf{U}\mathbf{V}^T$ and then impose non-negativity constraints on $\mathbf{U}$ and $\mathbf{V}$. DRMF can also easily be extended to handle missing values in collaborative filtering. Fast and pass-efficient algorithms such as [124] can also be integrated into DRMF to do robust analysis on massive data sets.

3.4.2 Implementation

When applying DRMF, we need to answer several important practical questions: how to choose the parameters e the maximal number of outliers allowed, K the rank of the factorization, and the starting point *i.e.* the initial guess of outliers $\mathbf{S}$. As discussed in Section 3.2, we can set e to be *e.g.* 5% of the whole data set so that the algorithm is not ignoring to much data.

Like most matrix factorization methods, in DRMF the rank of the factorization K is selected according to prior knowledge, cross-validation, or other heuristics. For example, we can observe the singular values of the data matrix, and choose a K to preserve certain amount of data variability. In some situations, the value of K is constrained by available computational resources, so we have to make trade-offs between accuracy and running time.

The initial guess of outliers $\mathbf{S}$ affects the final solution, since DRMF is non-convex and can be trapped in local minima. For many moderate situations we found that the simple choice of $\mathbf{S} = \mathbf{0}$ works well. But in extreme cases where the regular SVD is completely disrupted by outliers, this simple heuristic would lead DRMF into irrecoverable local minima. One such example is shown in Figure 3.1.

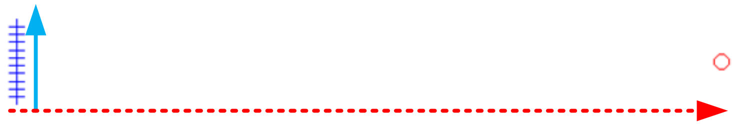


Figure 3.1: An example where DRMF with initial $\mathbf{S} = \mathbf{0}$ would fail. Blue crosses are normal points and the red circle is the outlier. Blue arrow shows the true principle subspace and the red dashed arrow shows the wrong one DRMF would get starting from $\mathbf{S} = \mathbf{0}$. Note that when starting from an $\mathbf{S}$ that correctly indicates the circle as an outlier, DRMF is able to achieve the correct blue subspace.

We found that an effective way is to solve this problem is to leverage the convexity of nuclear norm minimization (NNM) methods. Since NNM is a convex relaxation of DRMF, we can first compute the NNM solution of $\mathbf{S}$, and then use it to initialize DRMF. This strategy is similar to the case where the *linear programming* relaxation is used to approximate the original *integer programming* problems. In practice, we can run NNM for a few iterations and terminate before convergence. This is usually enough to guide DRMF to a good convergence region. In this way, we can get results that are better than using either NNM of DRMF alone. Other methods (*e.g.* [176]) can also be used to initialize DRMF. Using these initialization schemes, DRMF is able to overcome the problem posed in Figure 3.1 and get higher quality results than NNM.

Very recently we noticed a parallel work *GoDec* [183] that shares the same idea with DRMF. By comparison, DRMF extends to structured outliers as discussed in Section 3.4.1. In addition, the non-convexity of DRMF/GoDec is not addressed in [183] and the GoDec algorithm in its original form would likely get stuck in the extreme case in Figure 3.1.

3.5 Experiments

In this section we show the empirical effectiveness of DRMF on both simulation and real-world data sets. We compare DRMF to the following state-of-the-art competitors:

- **Robust PCA (RPCA) [22]** We use the code from `http://perception.csl.uiuc.edu/matrix-rank`. The efficient "inexact augmented Lagrange multiplier" implementation is used.

- **Stable principal component pursuit (SPCP) [185]** We implemented SPCA in Matlab using the *proximal gradient* method according to [52].
- **Outlier Pursuit (OP) [177]** We implemented OP in Matlab using the *proximal gradient* method according to [177].

In terms of Table 3.1, RPCA and SPCP solve the NNM problem with $\sigma = 0$ and $\sigma > 0$ respectively; OP solves NNM when the outlier is measured by $\|\mathbf{S}\|_{2,1}$ and $\sigma = 0$. The truncated SVD results (3.4) are also provided as a baseline.

DRMF and DRMF-R are implemented in Matlab. Partial SVD is done using PROPACK [91]. We terminate the iteration when the relative change of the objective value is diminishing.

DRMF, SPCP, and OP are all initialized by the solution produced by 10 iterations of RPCA. For DRMF, we *always* set the maximal number of allowed outliers to be $e = 0.05MD$ without tuning unless indicated otherwise.

3.5.1 Simulation Data

First, we study the performances of different methods on simulated data sets. We follow the set up in [22] to create the data matrix. Let $\mathcal{N}(\mu, \sigma^2)$ denote the Gaussian distribution with mean μ and variance σ^2, and $\mathcal{U}(a, b)$ denote the uniform distribution on the interval $[a, b]$. We generate the rank-K matrix as $\mathbf{L} = \mathbf{U}\mathbf{V}^T \in \mathbb{R}^{M\times M}$, where entries of the factor matrices $\mathbf{U}$ and $\mathbf{V}$ are *i.i.d.* samples from Gaussian distributions as $\mathbf{U} \in \mathbb{R}^{M\times K} \sim \mathcal{N}(0, 1/K), \mathbf{V} \in \mathbb{R}^{M\times K} \sim \mathcal{N}(0, 1/K)$. To generate the outlier matrix $\mathbf{S}$, we first select γM^2 entries from $\mathbf{S}$ and then draw their values from the uniform distribution $\mathcal{U}(-\sigma_o, +\sigma_o)$, where σ_o is the magnitude of outliers. Finally, we put them together and add *i.i.d.* Gaussian noise for each entry to get $\mathbf{X} = \mathbf{L} + \mathbf{S} + \mathcal{N}(0, \sigma_n^2)$, where σ_n is the level of the Gaussian observation noise.

Recovery Quality and Detection Rate

In this part we test how well the methods can detect the outliers and recover the underlying low-rank $\mathbf{L}$ accurately. We compare the performances on three different indices. To measure the accuracy of robust modeling, we compute the *root mean squared error* (RMSE) of the recovered $\hat{\mathbf{L}}$ *w.r.t.* the true $\mathbf{L}$. NNM results are "debiased" as in [110] to compensate the shrunken singular values. Outlier scores are computed as the absolute difference between the estimated $\hat{\mathbf{L}}$ and observation $\mathbf{X}$, and *average precision* (AP) is used to measure the detection performance. The simulation parameters we use are $K = 0.05n$, $\gamma = 0.05$, $\sigma_o = 1$. Finally, the running time is also compared.

First we examine the *entry outliers, noiseless observation* case by selecting uniformly random entries in $\mathbf{S}$ to be outliers and setting $\sigma_n = 0$. This situation satisfies the assumption made by RPCA. We compare the performances of SVD, RPCA, and DRMF. Note that SPCP and OP cannot be applied to this data set. For RPCA, we use parameter $\lambda = 1/\sqrt{M}$ as suggested in [22]. For SVD and DRMF, the true K is used for factorization. Matrices with sizes M between $[100, 2000]$ are used. Mean performances of 20 random runs are reported in Figure 3.2. We see that both RPCA and DRMF achieved much better performances than plain SVD, showing the necessity

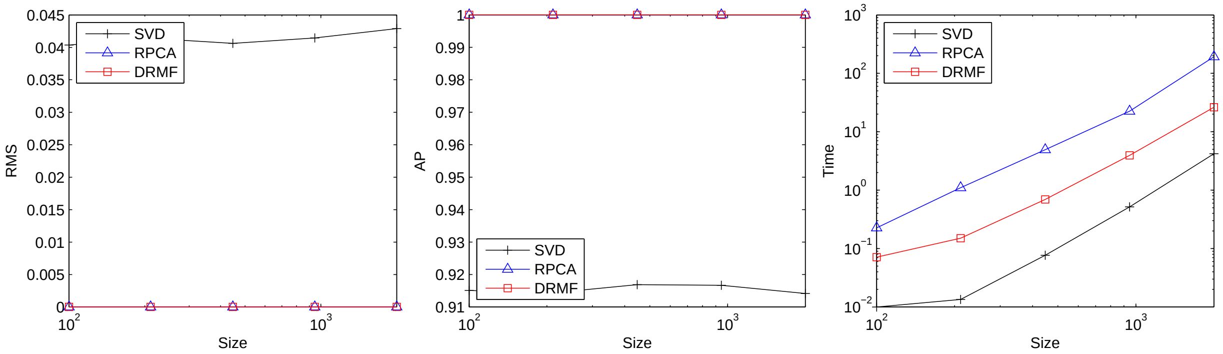


Figure 3.2: Performances on noiseless data with entry outliers. Note that the running time is shown in log-scale.

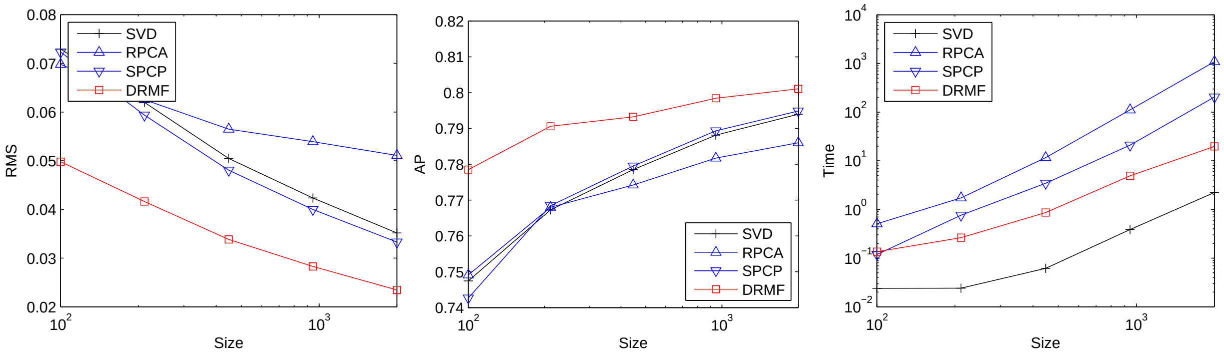


Figure 3.3: Performances on noisy data with entry outliers.

and effectiveness of robust factorization. Further, even in this noiseless case, DRMF is able to outperform RPCA consistently, using much less running time (only slightly slower than partial SVD).

Next we examine the *entry outliers, noisy observation* case. Compared to the previous simulation, we use $\sigma_n = 0.1$ and other settings remain the same. Note that this situation violates the assumption made by RPCA. We compare SVD, RPCA, SPCP, and DRMF here. The same settings for SVD, RPCA, and DRMF are used as before. For SPCP, the parameter regarding the level of regular Gaussian noise is set as suggested by [185]. Mean performances of 20 random runs are reported in Figure 3.3. On this data set, we see DRMF achieves the best performance again. RPCA performs poorly because of the noise, which inflates the estimated rank dramatically. SPCP, which is essentially an extended version of RPCA to handle noisy data, shows much better accuracy here, but is still worse than DRMF. Based on these two experiments, we conclude that DRMF can handle both noisy or noiseless data sets, and is able to achieve better results than RPCA and SPCP.

Further we examine the *row outliers, noisy observation* case. Unlike the entry outlier case, here we randomly select γM ($\gamma = 0.05$) rows in $\mathbf{S}$ and fill them with outliers from $\mathcal{U}(-1, 1)$. Note that this situation violates the assumptions made by RPCA and SPCP. We compare SVD, RPCA, SPCP, OP, DRMF, and DRMF-R here. For OP, we use parameter $\lambda = 0.4/\sqrt{\gamma M}$ as suggested by [177]. For DRMF-R, we directly specify that there can be γM outlier rows. Mean performances

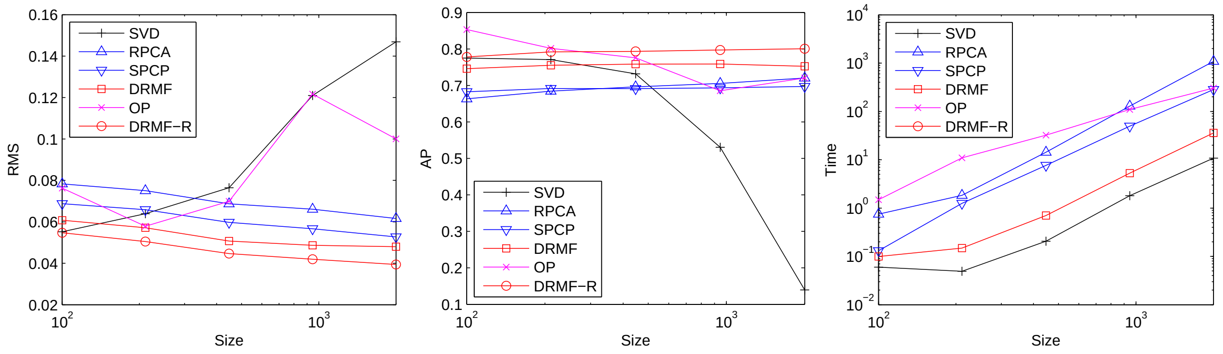


Figure 3.4: Performances on noisy data with row outliers.

of 20 random runs are reported in Figure 3.4. In the presence of row outliers, SVD and SPCP failed to work for large matrices. By contrast, OP performs poorly for small matrices, but then catches up as M grows larger. The reason could be that OP's suggested settings are not suitable for small problems, where tuning the parameters by cross-validation might give better results. RPCA, DRMF, and DRMF-R show stable performances and DRMF-R beats the others by a large margin. This verifies that utilizing additional knowledge about outlier patterns helps robust modeling and finding outliers. It is also interesting to see even though DRMF is design to handle entry outliers, its recovery quality is not affected by row outliers as SPCP is.

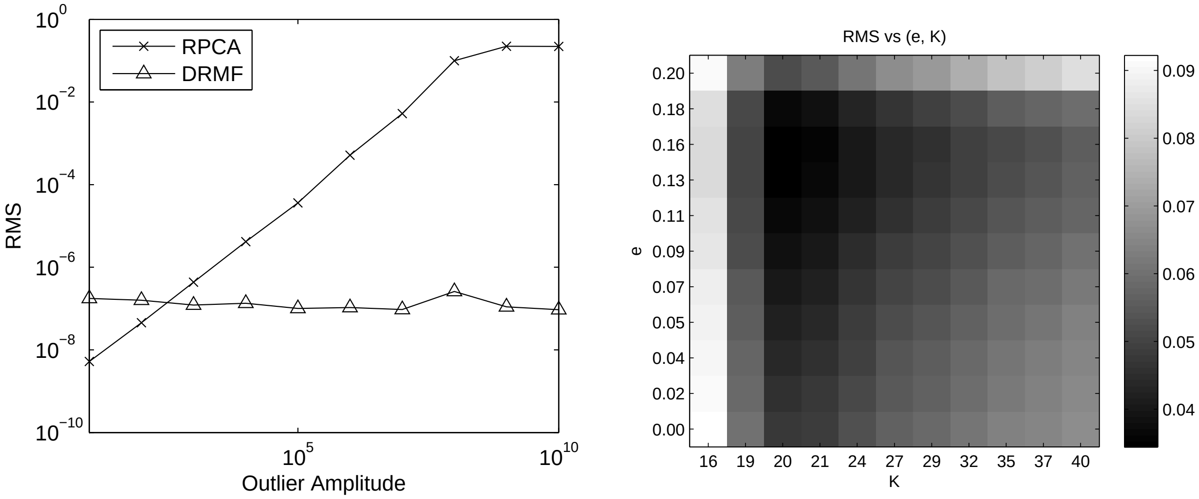
Based on these results, we conclude that DRMF outperforms the NNM methods in various cases, including noiseless and noisy cases as well as different outlier patterns.

Sensitivity

In this section, we study the sensitivity of DRMF's performance *w.r.t.* the magnitude of outliers and values of parameters.

First we examine how the magnitude of outliers affects the recovery quality. We simulate noiseless matrices with entry outliers, using $M = 400$, $K = 20$, and $\gamma = 0.05$. Then we change σ_o the magnitude of outliers from 1 to 10^5, and calculate the RMSE between the recovered $\hat{\mathbf{L}}$ and $\mathbf{L}$. Results produced by RPCA and DRMF are shown in Figure 3.5a. We can see that the recovery quality of DRMF is not affected by the magnitude of outliers at all. This is expected: the L_0-norm used in DRMF totally disregards the magnitude of outliers and only counts the number of them. On the other hand, though being robust, the L_1-norm used in RPCA is still influenced by large outliers, and we observe that this influence grows linearly with the magnitude of outliers.

We also examine how the recovery quality of DRMF is affected by the choice of parameters K the rank and e the number (or equivalently the proportion) of allowed outliers. The matrices are generated in the same way as the previous experiment with $\sigma_o = 1$. Then we run DRMF with different Ks between $[14, 60]$ and different e's between $[0, 0.2]$. Recovery RMSE are shown in Figure 3.5b. We can see that in a large range of parameters the performance is stable. It is especially interesting to see that moderately larger values of e actually produces better results. This behavior verifies the role of e in DRMF: it is only a safeguard to prevent excessive data being regarded as outliers, and it does not need to be same as the true number of outliers. We also observe



(a) (b)

Figure 3.5: (a) Recovery RMSE of RPCA and DRMF versus the outliers' magnitude. (b) Recovery RMSE of DRMF versus the parameters K the rank and e the proportion of allowed outliers. Darker color indicates smaller error.

that performance can be degraded when using too small Ks and too large e's ($\geq 20\%$). This is expected: when e is too large, a large portion of data can be treated as outliers (this actually violates our definition of outliers) and thus the results become unfaithful. When K is too small, the model lacks the capability to capture the normal variability of data.

3.5.2 Video Background Modeling and Activity Detection

In this experiment we consider the problem of modeling the background of videos. Estimating the background in videos is important for many computer vision tasks such as activity detection, yet also difficult because of the variability of the background (*e.g.* due to lighting conditions) and the presence of foreground objects such as moving people.

Here we apply robust matrix factorization methods to solve this problem. We assume that the background variations in videos are of low-rank (*i.e.* the background scenes can be approximated by linear combinations of several "basis" images), and the foreground objets are sparse outliers. By applying robust factorization methods to these video data, we want that the low-rank component will capture the background and its variations, while the foreground activities will be recognized as outliers so that they will not interfere the estimation of background.

Video sequences "Hall" (size 128×160, frames 2100-2400), "Lobby" (size 144×176, frames 1300-1700), "Restaurant" (size 120×160, frames 2500-3000), and "Shopping Mall" (size 128×160, frames 1500-2000) from [101] are used. "Hall" contains a relatively static background and many foreground activities. "Lobby" contains few foreground activities and large background variations. "Restaurant" and "Shopping Mall" are noisier and contain much more foreground activities. Sample images are shown in Figure 3.8.

	Hall	Lobby	Restaurant	Mall
SVD	0.669	0.695	0.547	0.721
RPCA	0.768	0.754	0.596	0.713
SPCA	0.746	0.770	0.549	0.753
DRMF	*0.805*	*0.792*	*0.666*	*0.774*

(a) Average Precision

	Hall	Lobby	Restaurant	Mall
SVD	4.9	1.54	2.00	3.24
RPCA	405.9	451.97	618.08	572.74
SPCA	18.6	17.40	25.06	42.34
DRMF	*23.76*	*16.92*	*29.60*	*57.41*

(b) Running Time (seconds)

Figure 3.6: Video activity detection performance

We flatten and stack the video frames into a matrix, with one row corresponds to a frame. Then we use SVD, RPCA, SPCA, DRMF to estimate the background. The anomaly scores of pixels are computed as the absolute difference between the estimated background and the observation, so our hope is that pixels corresponding to foreground activities will receive high scores. The performance is measured by the average precision of detecting foreground pixels on the ground truth frames. We use the suggested parameters for RPCA and SPCA (For SPCA, the median of pixels' standard deviation is used to estimate the Gaussian noise level). For SVD and DRMF, rank-5 models are used for "Hall", "Lobby" and rank-7 models are used for "Restaurant", "Shopping Mall" to capture the background variations.

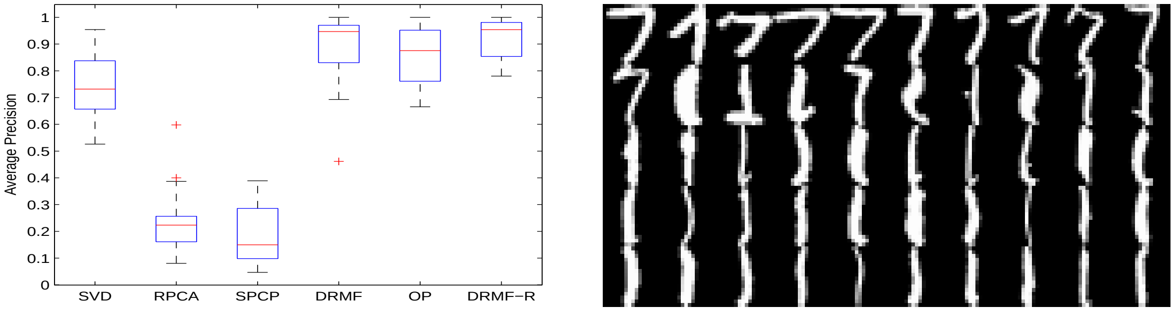
Detection results on some ground-truth frames using DRMF and RPCA are shown in Figure 3.8. Both methods are able to separate the foreground and background and produce good results. By more detailed examination, we can see that the backgrounds images captured by DRMF are smoother and contains less artifacts than RPCA. Figure 3.6 shows the detection performance and running time of different methods. Again, we see that DRMF consistently gives better detection performance than RPCA and SPCP.

3.5.3 Hand-written Digit Modeling

In the last experiment, we use these factorization methods to find anomalous digit images. The assumption is that images of the same digits have a low-rank structure (*i.e.* these images reside in a low-dimensional subspace), and if we inject in a small amount of different digits, these injections will violate the low-rank structure and stand out as outliers.

We use digits '1' and '7' from the USPS data set as in [177]. The image size is 16×16. We select a data set that is a mixture of 220 images of '1' and 11 images of '7'. The goal is to detect all the '7's in an unsupervised way. To do this, we flatten all images as row vectors and stack them into a 231×256 matrix $\mathbf{X}$. Then, factorization methods are applied to estimate low-rank matrices $\hat{\mathbf{L}}$ which are expected to capture the '1's. Finally, each image (a row of $\mathbf{X}$) is scored by the L_2-norm of its corresponding row in the error matrix $\mathbf{X} - \mathbf{L}$. Ideally, '7's should receive higher scores than '1's.

We compare SVD, RPCA, SPCP, DRMF, OP, DRMF-R on this task. for SVD and DRMF methods, rank $K = 3$ is used. For NNM methods, suggested parameters are used as before. Performances are measured by the average precision of detecting '7's. In each run, we randomly re-select the images. Results of 20 random runs are shown in Figure 3.7a.



(a)

(b)

Figure 3.7: USPS anomaly detection results. (a) the average precisions of detecting '7's among '1's. (b) images ranked by their anomaly scores in the descending order.

We can see that DRMF-R gives the best results, showing the advantage of our direct solution, and the benefit from incorporating knowledge about the outliers' structure. On the other hand, RPCA and SPCP failed in this case, since the non-uniform and non-random outliers in this data set violate their basic assumptions. The difference between OP and DRMF-R is significant: a *paired t-test* gives a p-value of 0.95×10^{-6}. Figure 3.7b shows a list of images ranked by their anomaly scores. We conclude that the '1's are clearly captured by the low-rank structure in DRMF, and it is interesting to observe the behavior of the $(1, 2)$th and the $(2, 5)$th image.

3.6 Summary

We proposed the direct robust matrix factorization (DRMF) algorithm as a simple and effective way for robust low-rank factorizations and outlier detection. We start from the fundamental notion of outliers and use a direct formulation to address this problem. DRMF is conceptually simple (SVD + error thresholding), easy to implement (about 10 lines of Matlab code), efficient (linear complexity *w.r.t.* number of entries), and flexible to incorporate prior knowledge about both the outliers and the low-rank structure.

DRMF is compared to the recently proposed nuclear norm minimization (NNM) family methods. We show that NNM methods are in fact convex relaxations of DRMF. In extensive empirical evaluations we find that the solutions given by DRMF achieve better performances over the state-of-the-art competitors that use relaxations, showing the advantage of our direct formulation.

The factorization algorithms proposed in this Chapter and Chapter 2 are widely useful when learning from discrete collective data as well as other vector or matrix based data format.

3.7 Automatic Novelty Discovery for Astronomy

Using the factorization techniques, we have developed an automatic system for real-time novelty discovery in astronomical survey data described in Section 1.5. From the ongoing SDSS III project[2], we can get daily updates of the new objects observed by the telescope. The goal is to develop a system that can examine these new objects in real-time, and automatically pick out the potentially interesting ones to present them to the astronomers for further examination. Then we collect feedback from the astronomers to support further studies.

One of the goals of this system is to find objects with unusual spectra. For this purpose we detect *subspace outliers* as mentioned in Section 3.1. The assumption behind this choice is that normal spectra lie in a low-dimensional linear subspace. In other words, we can find a small number of bases whose linear combinations can approximate normal spectra. On the contrary, anomalous spectra contain unusual spectral patterns that cannot be reconstructed by these bases. For example, if a spectrum has an unusual emission line that the normal bases do no have, then this spectrum cannot be approximated by the bases and thus will be detected. Many subspace approaches has been proposed to study spectra in astronomy. [37] provides a brief survey of researches that used PCA to accomplish tasks including spectra classification, visualization, and physical property extraction. [113, 178, 179] analyzed the quasars, galaxies, and stars in SDSS using PCA. [33] uses PCA to repair corrupted pixels in spectra. These researches indicate that low-dimensional subspace can indeed capture the main characteristics of the SDSS spectra.

Subspace outliers can be detected by first modeling the normal subspace and then finding points outside of it. We can use factorization algorithms to accomplish this goal. In this process, robustness is a desirable property of the low-rank model because of the severe challenges presented by outliers, especially for our automatic detection purpose. First of all, the survey observations usually have limited quality. There are plenty of bad pixels, interference from the sky, and other sources of noise such as galaxies mislabeled as stars. Second, the emission lines (Figure 1.1b) in the spectra may vary dramatically and cause problems to regular algorithms. As analyzed by [163], emission features form a main source of the inadequacy of linear subspaces given by PCA. Finally and most importantly, in our automated pipeline, the potentially novel objects are hidden among all the other regular ones. These novel objects are usually also outliers that can bend the model towards their side and thus make themselves harder to detect. Therefore, the low-rank models need to be robust, so that when they remain reliable when trained from a set of "dirty" spectra including potential novelties, bad pixels, large emission features, and other corruptions.

To answer these challenges, we use the DRMF algorithm to find robust normal subspaces. DRMF is able to alleviate the impact of emission lines in the spectra and obtain reliable subspace models. It is also efficient enough to process the large amount of data. We can use the existing data to learn a reliable subspace that contains most of the normal spectra. Once we have this subspace model, anomalous spectra can be found outside of this subspace.

In order to find the anomalies, we propose to use the following scoring function:

$$f_p(\mathbf{x}_i) = \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_p = \left(\sum_{j=1}^{D} |x_{ij} - \hat{x}_{ij}|^p \right)^{1/p} . \tag{3.13}$$

where $\hat{\mathbf{x}}_i$ is the projection of $\mathbf{x}_i$ in the normal subspace, or equivalently the low-rank reconstruction of $\mathbf{x}_i$. Therefore, $f_p(\cdot)$ is calculating the L_p distance from a point to its projection in the normal

[2] http://www.sdss3.org

subspace. The parameter p controls weather the scoring function should focus on a few badly reconstructed pixels or the mismatch of the overall shape of the spectrum. To apply the scoring function (3.13) with DRMF, we just have to train a DRMF model on the data and obtain the robust low-rank approximation $\mathbf{L}$. Then, we can use the ith row of $\mathbf{L}$ as $\hat{\mathbf{x}}_i$.

3.7.1 Results on SIMBAD Objects

In order to test he performance of our proposed detector, we looked up in the SIMBAD[3] database to find stars that have been assigned class labels. We test the detector's ability to find objects that are labeled by SIMBAD. Even if these SIMBAD-labeled objects are only a small portion of the whole data set, and that they might not correspond to the true novelties in the whole data set precisely, we trust that on average these labeled objects are more interesting than the rest of the data set, and we would want the detector to find them out.

Our data set contains $49,529$ stars from SDSS. These stars are selected to have a high enough signal-to-noise ratio. We normalize the stars' spectra so that each spectrum vector sums to 1, therefore only the shape of a spectrum matters. We found that $6,454$ out these stars have been assigned a label from one of the 42 SIMBAD classes, which are listed at `http://simbad.u-strasbg.fr/guide/chF.htx`. For the ease of presentation and analysis, we collapse these 42 classes into 15 according to the class hierarchy specified by SIMBAD.

We compare the performance of DRMF to PCA and RPCA as in Section 3.5. The parameters of different algorithms are hand tuned to achieve their respective optima. We found that the rank-5 DRMF combined with scoring function $f_{10}(\cdot)$ produces the best results. This indicate that the spectra indeed have a low rank, and the scoring function should focus on a few erroneously reconstructed pixels instead of counting small errors on all pixels.

Table 3.2 shows the labeled the classes and the APs of different methods on detecting these classes. We can see that the DRMF based detector achieved that best result, and the RPCA based detector is only slightly worse. Both robust methods are significantly between than the plain PCA, showing the necessity of robust modeling. The achieved AP of 56.72 also shows that the proposed detection scheme is very effective in finding interesting objects. Concretely, about 80 of the top 100, or 3000 of the top 5000, detection results are interesting enough to be labeled by SIMBAD.

3.7.2 Collaborative System

Anomaly detection is just the first step in analyzing the astronomical data. It is vital for learning systems to get supervision from experts in the forms of class labels, *etc*. Therefore, we present the detection results to the astronomers and let them provide feedbacks. After the astronomers have examined and labeled these anomalies, we shall have the "seed" label information to support further learning tasks such as *active learning* and *classification*.

To facilitate this process, We developed a real-time detection system as well as a website to detect anomalies from the real-time data provided by SDSS-III and present the detection results and to collect feedbacks. The backend system receives new data from the SDSS-III data source

[3] http://simbad.u-strasbg.fr/simbad

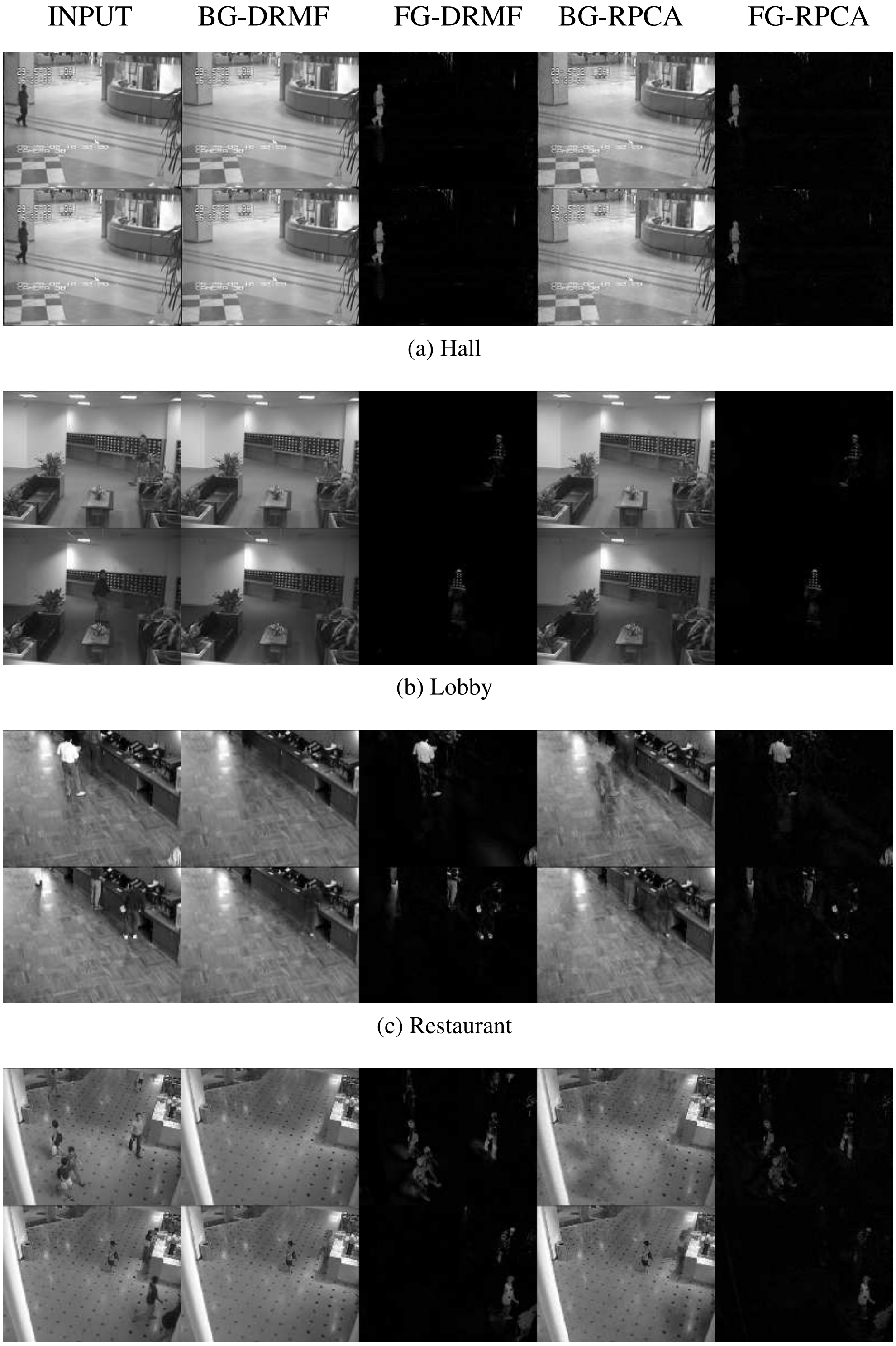
Class	Description	Class Size	PCA	RPCA	DRMF
All	All labeled objects	5611	31.01	54.39	56.72
?	Unknown object	46	0.80	1.11	1.09
IR	Infra-Red source	35	0.07	0.06	0.06
LM*	Low-Mass	71	2.96	0.89	0.69
PM*	High proper-motion	85	1.34	1.38	1.33
UV	UV-emission source	44	0.11	1.18	1.18
blu	Blue object	140	0.50	1.54	1.73
CLU	Cluster	45	0.60	0.96	0.90
G	Galaxy	316	7.18	9.96	9.83
CV	Cataclysmic Variable	96	21.04	27.46	26.50
PEC	Peculiar star	800	2.94	9.70	9.52
NEB	Nebula	24	59.25	75.79	75.66
V	Variable star	172	0.59	0.54	0.70
WD	White dwarf	3617	25.69	52.48	56.46
RAD	Radio source	105	1.27	1.17	1.13
X	X-ray source	15	0.07	0.34	0.32

Table 3.2: AP of detecting SIMBAD objects using the normalized spectrum feature.

daily. It then update the robust low-rank model and detection results to reflect the recent change. Various features and detection methods are implemented for diversity and evaluation purposes.

A snapshot of the website[4] is shown in Figure 3.9. The goal of the website is to easy communication and collaboration. Through it, we can inform the astronomers of the latest detection results and they can give us feedbacks on what these results means and how good they are. The users is able to select detection results based on different object types, features, and detection algorithms. Each detected object has a block showing the essential information and useful links for easy labeling. Comments and feedbacks can be collected from multiple users to enable discussion and collaboration. We also provide other functionalities such as finding look-alike objects based on spectrum similarity, as shown in Figure 3.10. In the future, this website can be enhanced to facilitate other learning tasks such as classification, clustering, finding group anomalies (See Chapter 4), and so on.

[4] Currently hosted at `http://www.autonlab.org/sdss`



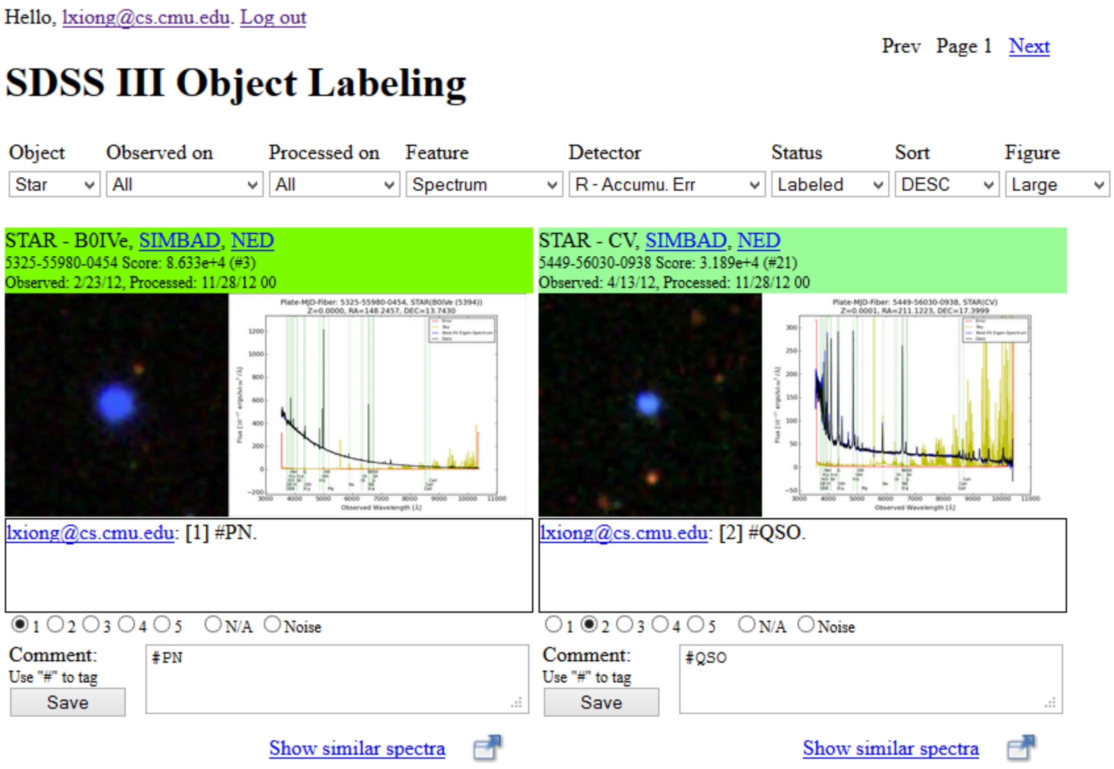
(a) Hall

(b) Lobby

(c) Restaurant

(d) Shopping Mall

Figure 3.8: Video activity detection result frames. In each sub-figure, the images from left to right are: the original frame, background and foreground from DRMF, background and foreground from RPCA.



Hello, lxiong@cs.cmu.edu. Log out

Prev Page 1 Next

SDSS III Object Labeling

Object	Observed on	Processed on	Feature	Detector	Status	Sort	Figure
Star	All	All	Spectrum	R - Accumu. Err	Labeled	DESC	Large

STAR - B0IVe, SIMBAD, NED
5325-55980-0454 Score: 8.633e+4 (#3)
Observed: 2/23/12, Processed: 11/28/12 00

lxiong@cs.cmu.edu: [1] #PN.

● 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ N/A ○ Noise

Comment: Use "#" to tag: #PN

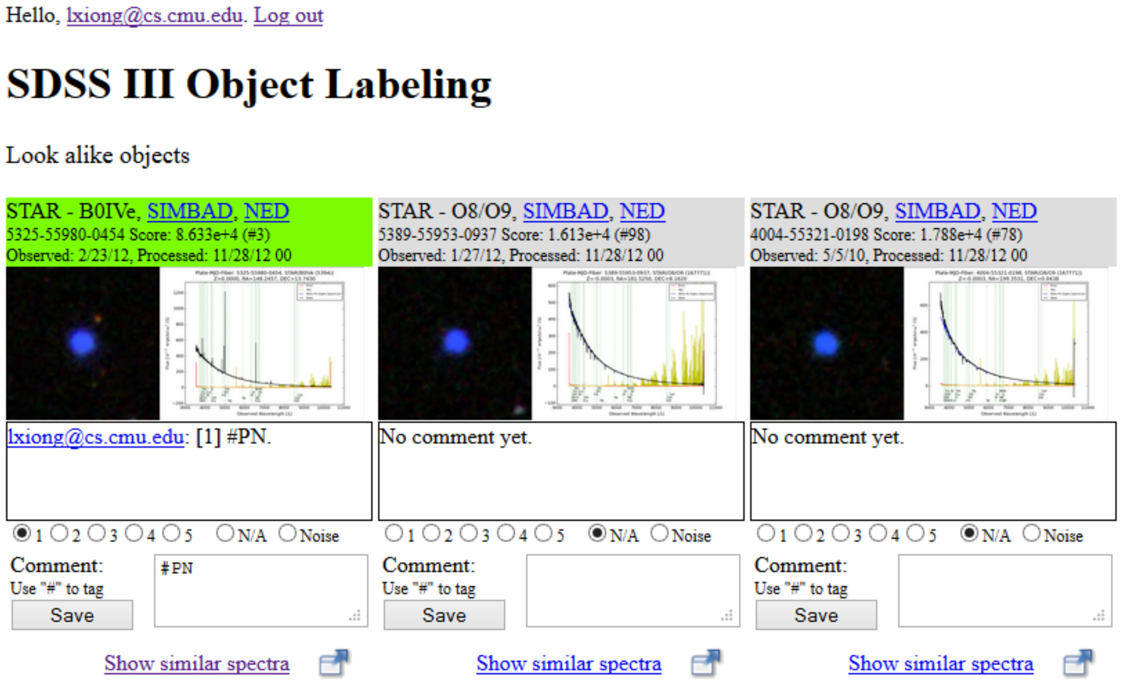
STAR - CV, SIMBAD, NED
5449-56030-0938 Score: 3.189e+4 (#21)
Observed: 4/13/12, Processed: 11/28/12 00

lxiong@cs.cmu.edu: [2] #QSO.

○ 1 ● 2 ○ 3 ○ 4 ○ 5 ○ N/A ○ Noise

Comment: Use "#" to tag: #QSO

Figure 3.9: The frontpage of the SDSS collaborative SDSS website.



Hello, lxiong@cs.cmu.edu. Log out

SDSS III Object Labeling

Look alike objects

STAR - B0IVe, SIMBAD, NED
5325-55980-0454 Score: 8.633e+4 (#3)
Observed: 2/23/12, Processed: 11/28/12 00

lxiong@cs.cmu.edu: [1] #PN.

● 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ N/A ○ Noise

Comment: Use "#" to tag: #PN

STAR - O8/O9, SIMBAD, NED
5389-55953-0937 Score: 1.613e+4 (#98)
Observed: 1/27/12, Processed: 11/28/12 00

No comment yet.

○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ● N/A ○ Noise

Comment: Use "#" to tag

STAR - O8/O9, SIMBAD, NED
4004-55321-0198 Score: 1.788e+4 (#78)
Observed: 5/5/10, Processed: 11/28/12 00

No comment yet.

○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ● N/A ○ Noise

Comment: Use "#" to tag

Figure 3.10: The UI for finding similar objects in the SDSS collaborative SDSS website.

Part II

Learning from Multidimensional Data

Chapter 4

Generative Models for Collective Data

From now on, we consider groups of real-valued, multi-dimensional points. For these groups, there is no easy way to reduce them into vectorial representations. In this chapter, we describe parametric generative models to directly capture the generating process of the groups. These models can then facilitate us to do classification, clustering, anomaly detection, and so on. In the following, however, our models will be motivated by the group anomaly detection problem.

4.1 Introduction

Given a data set, anomaly/novelty detection aims at finding things that "surprise" us. These things can either interfere with the learning process, in which case they should be removed, or they may have values for being novel. Section 3.1 gave a brief introduction on anomaly detection. Traditional anomaly detection typically focuses on finding individual point anomalies. But often the most interesting or unusual things in a data set are not odd individual points, but rather larger scale phenomena that only become apparent when groups of points are considered. We call these unusual groups the *group anomalies*.

Group anomalies exist in many real-world problems. For example, as mentioned in Chapter 1, astronomy surveys such as the *Sloan Digital Sky Survey* (SDSS) produce descriptions for a vast amount of celestial objects. We not only want to pick out the scientifically valuable objects like planetary nebulae, but also special clusters of galaxies that could shed light on the development of the universe [166]. Also in the particle simulation systems in physics, a single particle is seldom interesting, but a group of particles can exhibit interesting motion patterns like the interweaving vortices. In computer vision, an unusual image can be a strange group of local patches, and an anomalous video sequence can be thought of as an odd group of image frames. Other examples are abundant in the fields of text processing, time series, and spatial data analysis.

Two types of group anomalies are considered. A point-based group anomaly is a group of individually anomalous points. A distribution-based anomaly is a group where the points are relatively normal, but as a whole they are unusual. Most existing work on group anomaly detection focuses on point-based anomalies. A common way to detect point-based anomalies is to first identify anomalous points and then find their aggregations using scanning or segmentation methods

[38, 39, 68]. This paradigm clearly does not work well for distribution-based anomalies, where the individual points are normal. To handle distribution-based anomalies, we can design features for groups and then treat them as points [25, 81]. However, this approach relies on feature engineering that is domain specific and can be difficult.

We take a generative approach to address the group anomaly detection problem. If we have a probabilistic model that generates the normal data, then we can mark the groups that have small probabilities under this model as anomalies. The "bag-of-points" assumption is made, *i.e.* , points in the same group are unordered and infinitely *exchangeable*. Under this assumption, mixture models are often used to model the data due to *De Finetti*'s theorem [40]. The most famous class of generative models for modeling group data is the family of *topic models* [14, 72]. In topic models, distributions of points in different groups are mixtures of simple components called the "topics", which are shared among all the groups.

We propose the *genre models* based on topic models. Genre models are specifically designed for the purposes of detailed characterization of the groups and detecting distribution-based group anomalies. Flexible probabilistic structures based on the mixture of "genres" is employed to describe how the topic weights are generated for each group so that complex normal behaviors can be modeled. These genres capture the high-level distributional behavior of the groups, and therefore are ideal for detecting distribution-based anomalies.

In order to achieve more precise modeling of the groups, we further add the flexibility to allow each group to have their own topics in order to accommodate the variations of the points' distributions in different groups. Meanwhile, information is still shared among groups via a global mechanism called the "topic generators" to help estimate the topics. Topic generators can also capture the behavior of the topics and detect the presence of unusual topics that cause point-based anomalies.

Having the genre models, we can examine if a test group conforms to the normal behavior defined by the learned genres and topic generators. We show that straightforward scoring functions have their limitations and may lead to unstable results. Instead, several specifically designed scoring functions are used to detect both the point-based and distribution-based group anomalies. These scoring function are also developed to be robust against noise, and be able to ameliorate the weaknesses of the simpler models to achieve a good balance between the speed and flexibility.

Exact inference and learning for the genre models are generally intractable, so we resort to approximate methods. Both *variational inference* or *Gibbs sampling* [58] methods are developed to learn the genre models. We test the performance of the genre models along with the scoring functions on both synthetic and on real-world data sets including scene images, astronomical surveys, and turbulence simulations. Empirical results show that the proposed methods are effective in modeling collective data and finding group anomalies.

The chapter is structured as follows. We introduce some background and define the problem set-up Section 4.2. In Section 4.3 we describe some related work. The proposed models and scoring functions are described in Section 4.4, 4.5, and 4.6. In Section 4.7, we then make detailed discussion about the models and the scoring functions. Experimental results are shown in Section 4.8. We finish with a short discussion and conclusions (Section 4.9).

4.2 Background

In this section, we provide background about topic models and define our group anomaly detection problem. For intuition, we introduce the problem in the context of detecting anomalous images, rare galaxy clusters, and unusual motion in a dynamic fluid simulation, but the methods can be used for other collective data.

We consider a data set with M groups $G_1, \ldots, G_M$ (e.g. spatial clusters of galaxies, patches in an image, or fluid motions in a local region). Group G_m contains N_m points (galaxies, image patches, simulation grid points) as $G_m = \{\mathbf{x}_{m,1}, \ldots, \mathbf{x}_{m,N_m}\}, \mathbf{x}_n \in \mathbb{R}^D$, where D is the dimensionality of the points. We further assume that points in the same group are unordered and exchangeable.

Topic models such as the *latent dirichlet allocation* (LDA) [14] are widely used to model data having this kind of group structure. The original LDA model was proposed for text processing. It represents the distribution of points (words) in a group (document) as a mixture of K global topics $p(\mathbf{x}; \beta_1), \ldots, p(\mathbf{x}; \beta_K)$, where β_k is the parameter of the kth topic. When the points are discrete (*e.g.* words in text documents), $p(\mathbf{x}; \beta_k)$ can be the multinomial distribution $\mathcal{M}(\beta_k)$ with $\beta_i \in \mathbb{S}^D$, where $\mathbb{S}^D$ is the D-dimensional probability simplex. Let $\mathcal{M}(\theta)$ be the multinomial distribution parameterized by $\theta \in \mathbb{S}^K$ and $\mathcal{D}ir(\alpha)$ be the prior Dirichlet distribution with parameter $\alpha \in \mathbb{R}^K_+$. LDA generates the mth group by first drawing its topic weight θ_m from the prior distribution $\mathcal{D}ir(\alpha)$. Then for each point x_{mn} it draws one of the K topics from $\mathcal{M}(\theta_m)$ (*i.e.* , $z_{mn} \sim \mathcal{M}(\theta_m)$) and then generates the point according to this topic ($x_{mn} \sim \mathcal{M}(\beta_{z_{mn}})$). A description of the LDA model can also be found in Figure 1. Figure 4.1 shows the graphical model of LDA.

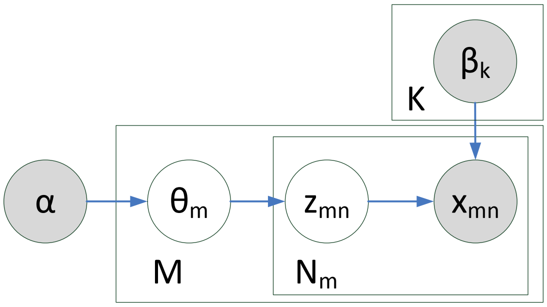


Figure 4.1: Graphical model for latent Dirichlet allocation (LDA).

Essentially, topic models capture each group by a mixture model. The key to successful modeling is that the topics (mixture components) $\{p(\mathbf{x}; \beta_k)\}_{k=1,\ldots,K}$ are shared among all groups, therefore all the information are used to learn them. The shared topics also provide a common basis to compare the groups. Another important ingredient is that the mixing weights are governed by the global distribution $p(\theta; \alpha) = \mathcal{D}ir(\alpha)$, which is use to convey prior information to help the estimation of the topics and the topic weights in each group.

The topic models can also help us model groups of real-valued, multidimensional points with a slight change. In our examples, we want the topics to represent concepts such as the galaxy types (*e.g.* “blue”,“red”, or “emissive”, with $K = 3$ topics), objects in the images, or common motion patterns in the fluid (go left, go right, etc), each of which can be captured by a distribution of the points. To do this, we can choose to model the topics by Gaussian distributions (*i.e.* $p(\mathbf{x}; \beta_k) = \mathcal{N}(\beta_k)$ where β_k contains both the mean vector and the covariance matrix), so that each point is generated from one of the K Gaussian distributions.

Now we ask the question whether the distribution of points in group G_m is normal. At a higher level, a group is characterized by the topic weight θ_m, *i.e.* , the proportion of different topics in the group G_m. At a lower level, we should also look at how the actual points are generated from the topics. This two-level characterization can help us define the group anomalies: a *point-based* group anomaly contains points that are unlikely to be from any of the topics, and a *distribution-based* group anomaly has a topic weight θ_m that is anomalous. For example, when detecting group anomalies the astronomy data, we are looking for galaxy clusters containing galaxies that do not fall into the common types (red, blue, and emissive), or clusters in which the proportion of different types of galaxy is strange.

Although topic models are very useful in estimating the topics and topic weights in the groups, the existing methods are *incapable of detecting group anomalies comprehensively*. In order to detect anomalies, the model should be flexible enough to capture complex normal behaviors. For example, it should be able to model complex and multi-modal distributions of the topic weight θ. LDA, however, only uses a single Dirichlet distribution to generate topic weights, and cannot define what is the normal and what is not with precision. It also uses the same K topics for all groups, which might makes groups indifferentiable when looking at their topics. Moreover, these shared topics are not adapted to each group either.

The genre models and the their corresponding scoring functions are developed to address these problems. Based on *latent Dirichlet allocation* (LDA) [14], we progressively propose three probabilistic hierarchical models designed specifically for the purpose of detecting group anomalies. The first model is simple and focuses on *distribution-based* anomalies and enriches LDA by allowing a flexible way of generating topic weights. The second model further takes both *distribution-based* and *point-based* anomalies into account, forming a very elastic topic model and a comprehensive anomaly detector. The third one finds the trade-off between model flexility and learning speed to make the most practical model.

4.3 Related Work

Typically, the notion of "anomaly" depends heavily on the specific problem, and various algorithms have been developed for their own purposes. Quite often they are based only on the simple idea that a data point is anomalous if it falls in a low density region of the feature space. For example, [182] uses the distances to nearest neighbors as an anomaly score. [21] consider the case of non-uniform density of the normal data, and propose a local outlier factor for detecting anomalous instances. We can also explicitly estimate the underlying density function and use statistical tests to find anomalies. To see a comprehensive summary, readers can refer to the survey by [26].

Detecting group anomalies is not a new problem, but only a few results have been published on it. One idea is to represent each group as a point, and then apply point anomaly detectors for these groups. To do this, we need to define a feature vector for the groups [25, 81]. A problem with this approach is that it relies heavily on feature engineering, which can be domain specific and difficult. We believe that directly modeling the generative process of the data is more natural, and can help us explore the data sets.

Another approach is to first identify the individual anomaly points, and then try to find aggregations of these points. Scan and segmentation methods are often used for this purpose. On image data, [68] applied a point anomaly detector to find anomalous pixels, and then segment the image to find the anomalous group of pixels. [38] first detects interesting points, and then find subsets of the data with a high ratio of anomalous points. [39] proposed a scan statistic-based method to find anomalous subsets of points. In these approaches the anomalousness of a group is determined by the anomalousness of its member points, therefore they cannot find anomalous groups that are unusual only at the group level *i.e.* the distribution-based anomalies.

The proposed genre models belongs to the family of topic models. The goal of traditional topic models is to estimate the topics and the topic weights in each group, and the model parameters are used to facilitate the estimations. They lack the ability to do group anomaly detection because we need models that capture the details of how the groups are generated, so that they can differentiate unusual behaviors from the normality. Many enhanced topic models have been proposed to increase the modeling power. [13] and [102] enhanced the prior distribution of the topic weights to model the correlations between topics. [80] and [50] use mixture models to generate topic weights to do clustering on the groups. These ideas are useful for modeling group-level behaviors but fails to capture anomalous point behaviors. On the other hand, [45] proposed to use different topics for different groups in order to account for the burstiness of the points. These adaptive topics are useful in recognizing point-level anomalies, but cannot be used to detect anomalous behavior at the group level. Finally, it was unclear how topic modeling should be used to find group anomalies. To address the above problems, we use ingredients from the topic modeling research and propose new models to characterize groups both at the group-level and the point-level. Corresponding scoring functions are also designed to be sensitive to anomalies but also robust against insignificant noise. We demonstrate that they are able to solve the issues above and performs better than the existing algorithms.

4.4 Multinomial Genre Models

We extend LDA to address several of its weakness in detailed modeling of complex collective data. To address the problem of simplistic distribution of topic weights in LDA, we introduce the concept "genres" to characterize the topic weights so that complex normal behaviors can be recognized. A genre, intuitively, is a type of typical/normal topic weight, and we allow a group to derive its topic weight from one of the many genres. The combination of these genres is very flexible and thus can accurately describe what normal topic weights should be like in the data set. In the next sections, the assumption that topics are shared globally in topic models will also be relaxed to further enhance the modeling power.

To start, we let the genres be the typical topic weights themselves. In other words, we construct a dictionary of typical topic weights (*i.e.* multinomial distributions), and each group can select one of them as its own topic weight. We call this model the *Multinomial Genre Model* (MGM).

We assume that there are K topics $\{p(\mathbf{x}; \beta_k)\}_{k=1,\dots,K}$, and the points are generated from one of the K Gaussian topics as $p(\mathbf{x};\beta_k) = \{\mathcal{N}(\mu_k, \Sigma_k)\}$, where $\beta_k = \mu_k, \Sigma_k$ is the mean and covariance of the Gaussian. But we shall still use the general notation to cope with other types of topics. Also let the tth genre be $\alpha_t \in \mathbb{S}^K$ denoting a typical topic weight vector, and $\alpha = \{\alpha_1, \ldots, \alpha_T\}_{t=1}^T$ denote the set of T genres. $\rho \in \mathbb{S}^T$ is a distribution over the genres (weights of the genres). The generative process of MGM is described in Algorithm 4, and the corresponding graphical model is shown in Figure 4.2.

Algorithm 4 The generative process of MGM.

```
for groups $m = 1$ to $M$ do
  • Choose a genre $y_m \in \{1, \ldots, T\}$, $y_m \sim \mathcal{M}(\rho)$. Let the topic weight $\theta_m = \alpha_{y_m} \in \mathbb{S}^K$.
  for $n = 1$ to $N_m$ do
    • Choose a topic $z_{mn} \in \{1, \ldots, K\}$, $z_{mn} \sim \mathcal{M}(\theta_m)$.
    • Generate a point $\mathbf{x}_{mn} \in \mathbb{R}^D$, $\mathbf{x}_{mn} \sim P(\mathbf{x}_{mn}|\beta, z_{mn})$.
```

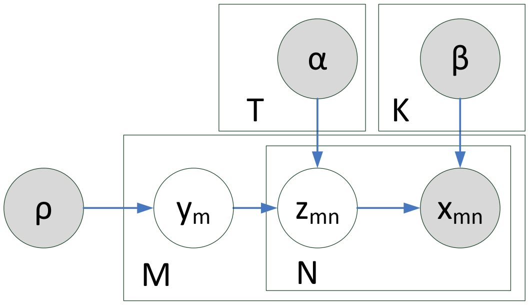


Figure 4.2: The Multinomial Genre Model (MGM).

Our strategy for group anomaly detection is as follows. Using the training set, we first learn the model parameters $\Theta = \{\rho, \alpha, \beta\}$. If a test group G is not compatible with our model, then it will lead to a small likelihood $P(G|\Theta)$ compared to normal groups like those in the training data. Hence we can detect it as an anomalous group.

Under MGM, the complete and marginal likelihood of group G_m are

$$P\left(y_m, z_m, G_m|\Theta\right) = \mathcal{M}(y_m|\rho) \prod_{n=1}^{N_m} \mathcal{M}(z_{mn}|y_m, \alpha) P(x_{mn}|z_{mn}, \beta), \tag{4.1}$$

$$P(G_m|\Theta) = \sum_{t=1}^{T} \rho_t \prod_{n=1}^{N_m} \sum_{k=1}^{K} \alpha_{tk} P(x_{mn}|\beta_k). \tag{4.2}$$

To learn the parameters using maximum likelihood estimation, we want

$$\Theta = \underset{\rho,\alpha,\beta}{\arg\max} \log \prod_{m=1}^{M} P(G_m|\rho, \alpha, \beta).$$

Unlike the LDA model, direct maximization of the likelihood function is possible. We can use the *Expectation-Maximization* (EM) method. However in practice we found that the *variational EM* [77] method was able to learn high quality models faster than EM. Therefore in the following we shall describe the variational method.

4.4.1 Inference and Learning

According to the Jensen's inequality, for any variational distribution $q_m(y, z)$ we have that

$$\begin{aligned}\sum_m \log P(G_m|\Theta) &\geq \sum_m \int \mathrm{d}(y, z) q_m(y, z) \log \frac{P(y, z, G_m|\Theta)}{q_m(y, z)} \\ &= \sum_m \mathbb{E}_{q_m}\left[\log P(y, z, G_m|\Theta)\right] - \mathbb{E}_{q_m}\left[\log q_m(y, z)\right], \end{aligned} \qquad (4.3)$$

with equality iff $q_m(y, z) = P(y, z|G_m, \Theta)$, and $\mathbb{E}_q[\cdot]$ denotes the expected value *w.r.t.* the distribution q. The posterior distribution $P(G_m|\Theta)$ might difficult to compute, thus instead of directly attacking of $\log P(G_m|\Theta)$, we will maximize its lower bound as

$$\Theta = \underset{\Theta, \{q_m\}}{\arg\max} \sum_m \mathbb{E}_{q_m}\left[\log P(y, z, G_m|\Theta)\right] - \mathbb{E}_{q_m}\left[\log q_m\right], \qquad (4.4)$$

where we look for the variational distributions q_m in the parametric form:

$$q(y_m, z_m|\gamma_m, \phi_m) = q(y_m|\gamma_m) \prod_{n=1}^{N_m} q(z_{mn}|\phi_{mn}). \qquad (4.5)$$

Here $\gamma_m \in \mathbb{S}^T$ and $\phi_{mn} \in \mathbb{S}^K$ are the variational parameters, and $q(y_m|\gamma_m) = \mathcal{M}(\gamma_m)$, $q(z_{mn}|\phi_{mn}) = \mathcal{M}(\phi_{mn})$ are multinomial distributions. Combining Eq. (4.1),(4.4) and (4.5), we have that the variational learning problem

$$\Theta = \underset{\{\gamma_m\}, \{\phi_m\}, \Theta}{\arg\max} \sum_{m=1}^{M} L_m\left(\gamma_m, \phi_m, \Theta\right), \qquad (4.6)$$

where L_m is

$$\begin{aligned} L_m(\gamma_m, \phi_m; \rho, \alpha, \beta) &= \mathbb{E}_q\left[\log P(y_m, z_m, G_m|\rho, \alpha, \beta)\right] - \mathbb{E}_q\left[\log q(y_m, z_m)\right] \qquad (4.7) \\ &= \mathbb{E}_q\left[\log P(y_m|\rho)\right] + \sum_{n=1}^{N_m} \mathbb{E}_q\left[\log P(z_{mn}|y_m, \alpha)\right] \\ &\quad + \sum_{n=1}^{N_m} \mathbb{E}_q\left[\log P(x_{mn}|z_{mn}, \beta)\right] - \mathbb{E}_q\left[\log q(y_m|\gamma_m)\right] \\ &\quad - \sum_{n=1}^{N_m} \mathbb{E}_q\left[\log q\left(z_{mn}|\phi_{mn}\right)\right]. \end{aligned}$$

We omit the derivation and show that each of the solutions below maximizes L_m when the other variables are fixed:

$$\phi^*_{mnk} \propto \exp\left(\sum_{t=1}^{T} \gamma_{mt} \log \alpha_{tk} + \log P(x_{mn}|\beta_k)\right) \tag{4.8}$$

$$\gamma^*_{mt} \propto \exp\left(\log \rho_t + \sum_{n=1}^{N_m}\sum_{k=1}^{K} \phi_{mnk} \log \alpha_{tk}\right) \tag{4.9}$$

$$\alpha^*_{tk} \propto \sum_{m=1}^{M} \gamma_{mt} \sum_{n=1}^{N_m} \phi_{m,n,k}. \tag{4.10}$$

Note that the multinomial parameters need to be normalized to sum to one. Finally, to calculate $\{\beta_k\}$, we need to solve

$$\beta_k = \arg\max_{\beta_k} \sum_{m=1}^{M}\sum_{n=1}^{N_m}\sum_{k=1}^{K} \phi_{mnk} \log P(x_{mn}|\beta_k). \tag{4.11}$$

Specially, when $P(\mathbf{x}_{mn}|\beta_k) = \mathcal{N}(\mathbf{x}_{mn}|\mu_k, \Sigma_k)$, then learning $\{\mu_k, \Sigma_k\}$ is the same as fitting Gaussians in a mixture of Gaussians model with ϕ being the weights of the samples [114].

In inference, we seek for the variational posterior distributions $q(\gamma)$ and $q(\phi)$. We can fix the value of the parameters ρ, α, β and update the values of γ, ϕ using Eq. (4.9) and (4.8) iteratively until convergence. When learning the MGM model, we iteratively update all the model parameters together with the variational parameters until convergence.

In order to use the MGM model we need to determine T the number of genres and K the number of topics. To automatically determine their values, we can either use model scoring methods such as BIC [149], or AIC [3], or we can resort to cross-validation to find the best parameter values that can maximize the specific learning performances. The definition of BIC score is given by $BIC(X, \Theta) = \ln L(X, \Theta) - \frac{1}{2}\ln(|\Theta|)$, where $|\cdot|$ stands for the number of free parameters. Similarly, the AIC score is given by $AIC(X, \Theta) = \ln L(X, \Theta) - |\Theta|$. We can then use these criteria to search for the best T and K values. In practice, we first determine K using $T = 1$, and then determine T fixing K.

4.4.2 Scoring Functions

In this section we discuss how to define scoring functions that can detect group anomalies based on MGM. Having learned the parameters Θ, a natural choice is to score a group by its likelihood $-\ln P(G|\Theta)$. In theory, this likelihood score is able to find anomalous groups that either contain anomalous points or have strange topic weights. However, directly using (4.2) may produce dubious results. In fact, the likelihood is problematic even when used to find point-based anomalies. First, if a group only contains points from the centers of the topics, then it would receive a low score, even if such a behavior never appeared in the training data. Second, if there is a single point not belonging the any of the topics, then the score of the whole group will be inflated to infinity. It is debatable that this behavior is correct, but we argue that such anomalies can easily be found by other much simpler methods, and will overshadow the truly anomalous collective behaviors.

To find the distribution-based anomalies, we propose to score only the topic weights in each group: we first infer the posterior distributions of the topics given the data, and then compute the expected likelihood of the topic weights. Unlike LDA, MGM does not give each group a topic weight variable θ_m, so we use the collection of topic variables $\mathbf{z}_m = \{z_{m,1}, \ldots, z_{m,N_m}\}$ instead. Formally, for the MGM model the distribution-based score $\chi_d(G_m)$ is defined as

$$\chi_d(G_m) = \mathbb{E}_{\mathbf{z}_m}\left[-\log P(\mathbf{z}_m|\Theta)\right] = -\sum_{\mathbf{z}_m} P(\mathbf{z}_m|\Theta, G_m)\log P(\mathbf{z}_m|\Theta), \tag{4.12}$$

$$P(\mathbf{z}_m|\Theta) = \sum_t \rho_t \mathcal{M}(\mathbf{h}_m; \alpha_t)$$

where $\mathbf{h}_m$ is obtained by aggregating the values in $\mathbf{z}_m$ in to a histogram. This score finds groups whose topic variables $\mathbf{z}_m$ are not compatible with any of the genres (stereotypical topic weights) in α learned by MGM.

To simplify the computation, we use the variational distributions $q_m(z_m|\phi_m)$ to approximate the corresponding posterior distributions $P(z_m|\Theta, G_m)$ in (4.12). The integrations then can be done by *Monte Carlo* method using samples drawn from the approximate posteriors. Similarly, the point-based score χ_p can also be approximated by the variational lower-bound.

4.5 Flexible Genre Models

In the previous section, MGM enhances the LDA model by allowing groups to select their topic weights from the dictionary of genres. With enough number of genres, complex normal behaviors of the topic weights can be captured, and topic weights that deviate from the learned genres will be marked as anomalies.

Although MGM addressed some flexibility issues of LDA, it is still inadequate for group anomaly detection. By modeling the genres by a dictionary of multinomial distributions, MGM does not take the uncertainty of topic weights into account. MGM also inherits the assumption that the topics are shared globally, therefore it cannot capture the point level behaviors. In this section we further improve our model to form a comprehensive model for detecting both distribution-based and point-based anomalies.

At the group level, "genres" are still used to model the topic distributions. Instead of multinomials, we use one Dirichlet distributions for each genre to model a typical distribution of topic weights. At the point level, each group has its own topics to accommodate the variations of its points, and these topics are generated by the global *topic generator*. We call this model the *Flexible Genre Model* (FGM). Given a group of points, we can examine whether or not it conforms to the normal behavior defined by the learned genres and topics; A *point-based* anomaly contains points from unusual topics are unlikely given the normal topic generators, while a *distribution-based* anomaly has a unusual topic weight θ_m given the normal genres.

The generative process of FGM is presented in Algorithm 5. A graphical representation of FG-M is given in Figure 4.3. We let $\mathcal{M}(\rho)$ be distribution of genres. Each genre is a Dirichlet distribu-

Algorithm 5 The generative process of FGM.

```
for groups $m = 1$ to $M$ do
    • Choose a genre $y_m \in \{1, \dots, T\}, y_m \sim \mathcal{M}(\rho)$.
    • Choose a topic weight from the genre $y_m$: $\theta_m \in \mathbb{S}^K, \theta_m \sim \mathcal{D}ir(\alpha_{y_m})$.
    • Choose $K$ topics $\{\beta_{m,k} \sim P(\beta_{m,k}|\eta_k)\}_{k=1,\dots,K}$.
    for points $n = 1$ to $N_m$ do
        • Choose a topic $z_{mn} \in \{1, \dots, K\}, z_{mn} \sim \mathcal{M}(\theta_m)$.
        • Generate a vector $x_{mn} \sim P(x_{mn}|\beta_{m,z_{mn}})$.
```

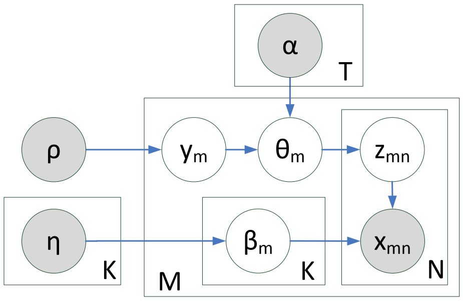


Figure 4.3: The Flexible Genre Model (FGM).

tion $P(\theta|\alpha_t)$ for generating the topic weights θ_m, and $\alpha = \{\alpha_t\}_{t=1,\dots,T}$ is the set of genre parameters. Each group has K topics $\beta_m = \{\beta_{m,k}\}_{k=1,\dots,K}$. The topic generators, $\{P(\beta_k|\eta_k)\}_{k=1,\dots,K}$, are the global distributions for generating the topics for each group. Having the topic distribution θ_m and the topics $\{\beta_{m,k}\}$, points are generated as in LDA.

By comparing FGM to LDA, we can observe that: (i) in FGM, each group has a latent genre y_m, which determines how its topic weight *should* look like ($Dir(\alpha_{y_m})$), and (ii) each group has its own topics $\{\beta_{m,k}\}_{k=1,\dots,K}$, but they are still tied through the generators $P(\beta|\eta)$. Thus, the topics can adapted to local group data, but information is still shared globally to enhance estimation results. Moreover, the topic generators $P(\beta|\eta)$ determine how the topics $\{\beta_{m,k}\}$ *should* look like. If a group uses unusual topics to generate its points, it can be identified.

For computational convenience, the topic generators are chosen to be *Gaussian-Inverse-Wishart* (GIW) distributions parameterized by $\eta_k = \{\mu_{0k}, \kappa_{0k}, \Psi_{0k}, \nu_{0k}\}$ [57]. The GIW distribution are conjugate to the Gaussian topics. Let $\Theta = \{\rho, \alpha, \eta\}$ denote the model parameters. The complete likelihood of data and latent variables in group G_m under FGM is:

$$\begin{aligned} &P(G_m, y_m, \theta_m, z_m, \beta_m|\Theta) \\ &= \mathcal{M}(y_m|\rho)\mathcal{D}ir(\theta_m|\alpha_{y_m})\prod_k GIW(\beta_{m,k}|\eta_k)\prod_n \mathcal{M}(z_{mn}|\theta_m)\mathcal{N}(\mathbf{x}_{mn}|\beta_{m,z_{mn}}). \end{aligned} \tag{4.13}$$

By integrating out θ_m, β_m and summing out $y_m, \mathbf{z}$, we get the marginal likelihood of G_m:

$$P(G_m|\Theta) = \sum_t \rho_t \int_{\theta_m,\beta_m} \mathcal{D}ir(\theta_m|\alpha_t)\prod_k GIW(\beta_{m,k}|\eta_k)\prod_n\sum_k \theta_{mk}\mathcal{N}(\mathbf{x}_{mn}|\beta_{m,k})d\beta_m d\theta_m. \tag{4.14}$$

4.5.1 Inference and Learning

The parameters of FGM can be learned via the maximum-likelihood method. The inferred values for the latent variables θ_m, β_m can be used for detecting anomalies and exploring the data. Nonetheless, the inference and learning under FGM are intractable, so we develop approximate method described below.

Inference Similar to the MGM model, approximate inference in FGM can also be done via variational EM. Yet due to the use of the GIW distributions, the formulae become very complicated. Alternatively, we can use *Gibbs sampling* [58] to learn FGM. In Gibbs sampling, we iteratively update one variable at a time by drawing samples from its conditional distribution when all the other parameters are fixed. Thanks to the use of conjugate distributions, Gibbs sampling in FGM is simple and easy to implement. The sampling distributions of the latent variables in group m are given below. We use $P(\cdot|\cdots)$ to denote the distribution of one variable conditioned on all the others.

For the genre membership y_m we have that:

$$P(y_m = t|\cdots) \propto P(\theta_m|\alpha_t)P(y_m = t|\rho) = \rho_t \mathcal{D}ir(\theta_m|\alpha_t). \tag{4.15}$$

For the topic distribution θ_m:

$$P(\theta_m|\cdots) \propto P(\mathbf{z}_m|\theta_m)P(\theta_m|\alpha, y_m) = \mathcal{M}(\mathbf{z}_m|\theta_m)\mathcal{D}ir(\theta_m|\alpha_{y_m}) = \mathcal{D}ir(\alpha_{y_m} + \mathbf{h}_m), \tag{4.16}$$

where $\mathbf{h}_m$ denotes the histogram of the K values in vector $\mathbf{z}_m$. The last equation follows from the Dirichlet-Multinomial conjugacy.

For $\beta_{m,k}$, the kth topic in group m, one can find that:

$$P(\beta_{m,k}|\cdots) \propto P(G_{mk}|\beta_{m,k})P(\beta_{m,k}|\eta_k) = \mathcal{N}(G_{mk}|\beta_{m,k})GIW(\beta_{m,k}|\eta_k) = GIW(\beta_{m,k}|\eta'_k), \tag{4.17}$$

where G_{mk} are the points in group G_m from the kth topic according to $\mathbf{z}_m$. The last equation follows from the Gaussian-Inverse-Wishart-Gaussian conjugacy. η'_k is the parameter of the posterior GIW distribution given the prior parameters η and G_{mk}; its form can be found in standard statistics textbooks *e.g.* [57].

For z_{mn}, the topic membership of point n in group m is sampled follows:

$$P(z_{mn} = k|\cdots) \propto P(x_{mn}|z_{mn} = k, \beta_m)P(z_{mn} = k|\theta_m) = \theta_{m,k}\mathcal{N}(x_{mn}|\beta_{m,k}). \tag{4.18}$$

Note that the multinomial parameters should be normalized to sum to 1.

Learning Learning the parameters of FGM helps us identify the groups' and points' normal behaviors. Each of the genres $\alpha = \{\alpha_t\}_{t=1,\ldots,T}$ captures one typical distribution of topic weights as $\theta \sim \mathcal{D}ir(\alpha_t)$. The topic generators $\eta = \{\eta_k\}_{k=1,\ldots,K}$ determine how the normal topics $\{\beta_{m,k}\}$ should look like. We use single-sample Monte Carlo EM [24] to learn parameters from the samples provided by the Gibbs sampler. Given sampled latent variables, we update the parameters to their maximum likelihood estimations: we learn α from $\mathbf{y}$ and θ; η from β; and ρ from $\mathbf{y}$.

ρ can easily be estimated from the histogram of y's. α_t is learned by the MLE of a Dirichlet distribution given the topic weights of groups in genre t *i.e.* $\{\theta_m | y_m = t, m = 1, \dots, M\}$. The MLE of Dirichlet can be solved using the *Newton–Raphson* method [118].

The kth topic-generator's parameter $\eta_k = \{\mu_{0k}, \kappa_{0k}, \Psi_{0k}, \nu_{0k}\}$ is the MLE of a GIW distribution given the parameters $\{\beta_{m,k} = (\mu_{m,k}, \Sigma_{m,k})\}_{m=1,\dots,M}$ (the kth topics of all groups). We have derived an efficient solution for this MLE problem. The details can be found at the end of this chapter.

The overall learning algorithm works by repeating the following procedure until convergence or equilibrium: (1) do Gibbs sampling to infer the states of the latent variables; (2) update the model parameters using the estimators above. If we only want to infer the posterior distributions of the latent variables, we can only repeat step (1) until enough samples are gathered to form the approximate empirical distribution.

Like for MGM, to select appropriate values for the parameters T and K (the number of genres and topics), we can apply the Bayesian information criterion (BIC) [149], or use cross-validation to find values that maximize the learning performances.

4.5.2 Scoring Functions

FGM can easily be used for group anomaly detection. We can first infer a group's latent states including the topics β and the topic weight θ, and then examine if they are compatible with the topic generators and genres in the model.

Point-based anomalies can be found by examining the topics. If a group contains anomalous points, then the topics that generated these points will deviate from the topic generators η. Let $P(\beta_m|\Theta) = \prod_{k=1}^{K} GIW(\beta_{m,k}|\eta_k)$. We define the *point-based anomaly score* as

$$f_p(G_m) = \mathbb{E}_{\beta_m}\left[-\log P(\beta_m|\Theta)\right] = -\int_{\beta_m} P(\beta_m|\Theta, G_m) \log P(\beta_m|\Theta) \mathrm{d}\beta_m. \tag{4.19}$$

The posterior distribution $P(\beta_m|\Theta, G_m)$ can again be approximated by the samples from Gibbs sampling, and the expectation can be done by Monte Carlo integration. Compared to the finding point-based anomalies using the point-wise likelihood as in Section 4.4.2, this scoring function examines the topic instead of the points. The topics are used as a summarization of the points as a whole, so that problems raised in Section 4.4.2 can be solved.

Distribution-based anomalies can be detected by examining the topic weights like in MGM. The genres $\{\alpha_t\}_{t=1,\dots,M}$ capture the typical distribution of topic weights. If a group's topic weight θ_m is unlikely under these genres, we call it anomalous. Let $P(\theta_m|\Theta) = \sum_{t=1}^{T} \rho_t \mathcal{D}ir(\theta_m|\alpha_t)$. The *distribution-based anomaly score* is

$$f_d(G_m) = \mathbb{E}_{\theta_m}\left[-\log P(\theta_m|\Theta)\right] = -\int_{\theta_m} P(\theta_m|\Theta, G_m) \log P(\theta_m|\Theta) \mathrm{d}\theta_m. \tag{4.20}$$

Again, this expectation can be approximated using Gibbs sampling and Monte Carlo integration.

4.6 Nonparametric Genre models

FGM provides us with great flexibility to model the group-level and point-level behaviors of the groups. It also inspires us to design effective scoring function to find different types of group anomalies. However, such flexibility comes with the a price. First, the inference and learning of FGM is slower than MGM. Specifically, the use of the multivariate Gaussian-Wishart distribution greatly increases the computation needed for both inference and learning. Second, the conjugate priors are chosen merely for the computational convenience rather than correctness. Thirdly, the conjugate priors involves a large number of free parameters that needs to be set or learned, causing volatile performance and difficulties in practical use. Therefore, we want other prior distributions that can implement similar flexibilities as in FGM, but runs much simpler and faster.

The nonparametric empirical Bayes (NPEB) method can be used to solve this problem. Instead of methods that use parametric conjugate priors, NPEB does not assume the form of the prior distribution to allow for minimum prior knowledge and restrictions. Applying NPEB to FGM, we can replace the mixture of Dirichlet distribution for topic weights $P(\theta|\rho, \alpha)$ and the Gaussian-Wishart distributions for the topics $\{P(\beta_k|\eta_k)\}_k$ with simpler $P(\theta|F_\theta)$ and $P(\beta|F_\beta)$ respectively, where F_θ, F_β are the nonparametric distributions for θ and β without further assumptions.

We use the nonparametric maximum likelihood (NPML) technique proposed by [89, 90]. It can be proved that the maximum likelihood estimates (MLE) of F are step functions in the parameter space *i.e.* the probability mass of $\hat{F}$ only exists at a finite number of discrete points in the parameters space, and number of steps in F grows as the data become more complex.

The above result means that, for a given data set, the MLE $\hat{F}_\theta$ contains a number of values for θ. To simplify the computational, we specify the number of steps in $\hat{F}_\theta$ to a relatively large value beforehand, instead of computing it from the data. Similar modeling can also be applied to the topic generators. In this case, this NPML becomes very similar to the mechanism we used in MGM.

We use the simplified NPML method above to improve the FGM and get the nonparametric genre model (NGM). Suppose that $\hat{F}_\theta$ the prior of the topic weights has T elements, $\hat{F}_{\beta_k}$ the prior of the kth topic has S elements. The generative process of NGM can be described in Algorithm 6, and its graphical representation is shown in Figure 4.4.

Algorithm 6 The generative process of NGM.

```
for groups $m = 1$ to $M$ do
    • Choose a genre $y_m \in \{1, \dots, T\}, y_m \sim \mathcal{M}(\rho)$. Let the topic weight $\theta_m = \alpha_{y_m} \in \mathbb{S}^K$.
    for topics $k = 1$ to $K$ do
        • Choose an $r_{mk} \in \{1, \dots, S\}, r_{mk} \sim \mathcal{M}(\pi_k), \pi_k \in \mathbb{S}^S$. Let the $k$th topic be $\beta_k = \eta_{k,r_{mk}}$.
    for $n = 1$ to $N_m$ do
        • Choose a topic $z_{mn} \in \{1, \dots, K\}$, $z_{mn} \sim \mathcal{M}(\theta_m)$.
        • Generate a point $\mathbf{x}_{mn} \in \mathbb{R}^D$, $\mathbf{x}_{mn} \sim P(\mathbf{x}_{mn}|\beta_{z_{mn}})$.
```

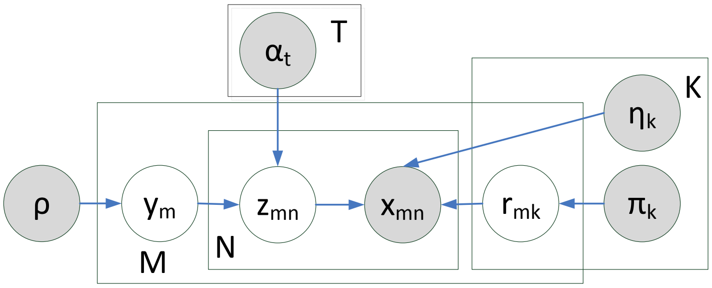


Figure 4.4: The nonparametric genre model (NGM).

The complete and marginal likelihood of data under NGM are:

$$
\begin{aligned}
p(y_m, r_m, z_m, G_m|\rho, \alpha, \pi, \eta) &= p(y_m|\rho) \prod_k p(r_{mk}|\pi_k) \prod_n p(z_{mn}|y_m, \alpha) p(x_{mn}|z_{mn}, r_m, \eta) \quad (4.21)\\
&= \rho_{y_m} \prod_k \pi_{k,r_{mk}} \prod_n \alpha_{y_m,z_{mn}} p(x_{mn}|\eta_{z_{mn},r_{m,z_{mn}}})
\end{aligned}
$$

$$
p(G_m|\rho, \alpha, \pi, \eta) = \sum_{y_m} \rho_{y_m} \sum_{r_m} \prod_k \pi_{k,r_{mk}} \prod_n \sum_{z_{mn}} \alpha_{y_m,z_{mn}} p(x_{mn}|\eta_{z_{mn},r_{m,z_{mn}}}). \quad (4.22)
$$

NGM does not assume the specific forms of the prior distributions for topic eights θ and the topics β, therefore with a suitable choice of T and S it can model complex behaviors the data represent. The model only involves simple distributions such as Gaussians, and therefore is easy and fast to learn.

4.6.1 Inference and Learning

When NGM is learned from the data, the nonparametric priors embodied in the parameters $\{\rho, \alpha\}$ and $\{\pi_k, \eta_k\}_k$ will contain the typical topic weights as well as topics in the training set. These priors can then help the inference of the topic weights θ and topics $\{\beta_k\}_{k=1,\ldots,K}$.

Like other genre models, the NGM model can be learned via the variational EM algorithm. We define for $P(y_m, r_m, z_m|G_m, \Theta)$ the posterior marginal distribution of latent variables a factorized variational distribution

$$
\begin{aligned}
q(y_m, r_{mk}, z_{nk}|\gamma_m, \tau_m, \phi_m) &= q(y_m|\gamma_m) \prod_k q(r_{mk}|\tau_{mk}) \prod_n q(z_{nk}|\phi_{mn}) \quad (4.23)\\
&= \mathcal{M}(y_m|\gamma_m) \prod_k \mathcal{M}(r_{mk}|\tau_{mk}) \prod_n \mathcal{M}(z_{nk}|\phi_{mn})
\end{aligned}
$$

where variational distributions $q(y|\gamma)$ models the genre y, $q(r|\tau)$ models how the topic was sampled from the nonparametric topic generators, and $q(z|\phi)$ models which topic generated a point.

The model and variational parameters can be obtained by maximizing the variational lower-bound to the marginal likelihood of data as described in Section 4.4.1. The actual derivation is similar to the one used in MGM. In the following we only show the update formulae for the iterative solution.

$$\phi_{mnk} \propto \exp\left(\sum_t \gamma_{mt} \log \alpha_{tk} + \sum_s \tau_{mks} \log p(x_{mn}|\eta_{ks})\right) \tag{4.24}$$

$$\gamma_{mt} \propto \exp\left(\log \rho_t + \sum_{n,k} \phi_{mnk} \log \alpha_{tk}\right) \tag{4.25}$$

$$\tau_{mks} \propto \exp\left(\log \pi_{ks} + \sum_n \phi_{mnk} \log p(x_{mn}|\eta_{ks})\right) \tag{4.26}$$

$$\alpha_{tk} \propto \sum_m \gamma_{mt} \sum_n \phi_{mnk} \tag{4.27}$$

$$\rho = \frac{1}{M} \sum_m \gamma_m \tag{4.28}$$

$$\pi_k = \frac{1}{M} \sum_m \tau_{mk} \tag{4.29}$$

To learn the topic generators, we need to maximize the following objective function:

$$\eta_{ks} = \arg\min_{\eta_{ks}} \sum_{m,n} \tau_{mks} \phi_{mnk} \log p(x_{mn}|\eta_{ks}). \tag{4.30}$$

Therefore, estimating η_{ks} is the same as fitting a Gaussian distribution using samples weighted by $\tau_{mks}\phi_{mnk}$.

4.6.2 Scoring Functions

Finding anomalies based on the inferred latent variables is not straightforward under the NGM. Due to the use of nonparametric priors, we do not have explicit latent variables such as the topic weight θ and the topics β for each group. So instead of scoring the latent variables, we find ways to score the data directly.

Point-based Anomaly Similar as before, we can find point-based anomalies by looking for groups that use unusual topics to generate the points. Since there is no explicit latent variable for the topics, we can use the following scoring function to score the points:

$$\begin{aligned}\chi_p(G_m) &= -\sum_{\mathbf{z}_m} p(\mathbf{z}_m|G_m, \Theta) \log p(G_m|\mathbf{z}_m, \Theta) \approx -\sum_{\mathbf{z}_m} q(\mathbf{z}_m|\phi_m) \log p(G_m|\mathbf{z}_m, \pi, \eta) \\ &= -\sum_{\mathbf{z}_m} q(\mathbf{z}_m|\phi_m) \sum_k \log p(G_{mk}|\pi_k, \eta_k) \\ &= -\sum_{\mathbf{z}_m} q(\mathbf{z}_m|\phi_m) \sum_k \log \sum_s \pi_{ks} p(G_{mk}|\eta_{ks}).\end{aligned} \tag{4.31}$$

where G_{mk} contains all the points from topic k according to $\mathbf{z}_m$. The above equations used the variational distributions to approximate the actual posterior distributions.

The key quantity in χ_p is how to compute $p(G_{mk}|\eta_{ks})$, which means how likely the set of points G_{mk} were generated by the Gaussian distribution $\mathcal{N}(\eta_{ks})$. The most intuitive option is the likelihood of *i.i.d.* points as $p(G|\eta) = \prod_n p(x_n|\eta)$, but it is flawed. For example, if the test group contains points that are only at the centers of the Gaussians, then the likelihood will be mistakenly high, because the distribution of the points is not normal. This problem can only be addressed by considering the points G_{mk} as a whole. This is also why in Eq. (4.12) we have to score the topic variables $\mathbf{z}_m$ as a whole instead of individually.

Essentially, we need $p(G_{mk}|\eta_{ks})$ to be a goodness-of-fit (GoF) measurement. Unfortunately, GoF tests in high-dimensions are notoriously difficult. Here we take a parametric approach. First, we construct a prior distribution for η, denoted as $\Omega(\eta)$. Then, we estimate a distribution for G_{mk} that has the same parametric form as η using $\Omega(\eta)$ as the Bayesian prior, denoted as $f(G_{mk}, \Omega(\eta))$. Finally, we use $p(f(G_{mk}, \Omega(\eta))|\Omega(\eta))$ as a surrogate of $p(G_{mk}|\eta)$. Intuitively, this approach uses the parametric distribution $f(G_{mk}, \Omega(\eta))$ to summarize G_{mk}, and then evaluate how probable $f(G_{mk}, \Omega(\eta))$ is generated from the model. We call this approach the *pseudo-prior* method.

We choose Ω to be the conjugate prior of η, and set the mode of $\Omega(\eta)$ to η so that $\Omega(\eta)$ can reflect what eta should be like. Since $\Omega(\eta)$ is conjugate to η, estimating $f(G_{mk}, \Omega(\eta))$ and evaluate its likelihood under $\Omega(\eta)$ is straightforward. Another advantage of this approach is that, the conjugate prior distributions usually have a degrees-of-freedom parameter to specify how strong the prior is. With a suitable strength, the Bayesian estimate $f(G_{mk}, \Omega(\eta))$ can be robust against random individual points and focus more on the collective behavior. In addition, if two groups have the same amount of anomalous points, then the larger group would receive a higher score.

Concretely for our NGM model where each η_{ks} is a Gaussian distributions, we let $\Omega(\eta_{ks}, \lambda)$ be the GIW distribution whose mode is at η_{ks}, where λ is a parameter specifying its degrees-of-freedom. λ acts as the "pseudo counts" in the prior distribution, and larger λ makes the score more insensitive to individual points or smaller groups. To evaluate $p(G_{mk}|\eta_{ks})$, we first estimate the Gaussian distribution $f(G_{mk}, \Omega(\eta, \lambda))$ based on the data G_{mk} and the prior $\Omega(\eta_{ks}, \lambda)$, and then calculate the GIW likelihood $p\left(f(G_{mk}, \Omega(\eta, \lambda))|\Omega(\eta_{ks}, \lambda)\right)$.

Note the resemblance between the pseudo-prior scoring function and the point-based scoring function f_p (4.19) for FGM. They are very simpler in that they both use adaptive topics to summarize the points and then score the topics. FGM explicitly learns the adaptive topics during training. On the other hand, NGM construct the adaptive topics only during detection time using pseudo-priors, making the training much simpler while achieving similar results.

Distribution-based Anomaly The above pseudo-prior approach can also be used to find distribution based anomalies using the topic variables $\mathbf{z}_m$. The pseudo-prior scoring function to find distribution-based anomalies is

$$\chi_d(G_m) = -\sum_{\mathbf{z}_m} p(\mathbf{z}_m|G_m, \Theta) \log p(\mathbf{z}_m|\Theta) \approx -\sum_{\mathbf{z}_m} q(\mathbf{z}_m) \log \sum_t \pi_t p(\mathbf{z}_m|\alpha_t) \tag{4.32}$$

To evaluate $p(z_m|\alpha_t)$, we first construct the pseudo-prior $\Omega(\alpha_t, \lambda) = \mathcal{D}ir(\alpha_t, \lambda)$ where λ is the pseudo-counts in the Dirichlet, then estimate the posterior topic weights $\theta_m = f(\mathbf{z}_m, \Omega(\alpha_t, \lambda))$, and finally evaluate the Dirichlet likelihood $p\left(f(\mathbf{z}_m, \Omega(\alpha_t, \lambda))|\Omega(\alpha_t, \lambda)\right) = \mathcal{D}ir(\theta_m|\alpha_t, \lambda)$.

Note the scoring function based on multinomial likelihood (4.12) can still work well in NGM. In fact, (4.12) is simpler and more natural to score the discrete variables $\mathbf{z}_m$. The pseudo-prior approach was only proposed as a remedy to the difficulty of evaluating the goodness-of-fit for continuous multidimensional data.

4.7 Discussion

In the previous sections, we progressively proposed three models: the *multinomial genre models* (MGM), the *flexible genre models* (FGM), and the *nonparametric genre models* (NGM). MGM is a basic model that introduces the concept of genre and use it the enhance LDA's modeling capability of topic weights. FGM enhances MGM by using more flexible probabilistic components and allowing the groups to have different topics. Inspired by FGM, NGM uses nonparametric priors to further remove modeling assumptions and enable faster learning.

The computational cost of the genre models mainly comes from the inference procedures, where we have to compute the point likelihood given the topic/topic generator *i.e.* $p(x|\beta)$ or $p(x|\eta)$. For the D-dimensional Gaussians used in this chapter, computing the likelihood for N points *w.r.t.* to all topics costs $O(NKD^2)$ time (or $O(NKSD^2)$ for NGM). To make it faster, we can first reduce D the dimensionality of the points using reduction algorithms such as PCA. Alternatively, we can use Gaussians with diagonal covariances so that the time complexity can be reduced to $O(NKD)$. Further, when there are many groups or the groups are very large, we can use a subset of the groups/points to get initial estimates of the models that can be refined later.

The parameters T and K are needed to specify the genre models. Usually these values can be selected by AIC/BIC scores or cross-validation. Theoretically, all the parameters of the prior distributions in FGM (specifically the Dirichlet parameters α and GIW parameters η) can be learned via the empirical Bayes methods. When these parameter are learned correctly good performance can be achieved. However, in practice we found that such an approach is usually unstable and may lead to bad local minima. On the other hand, finding a good fixed value for them involves much tuning. NGM partly solves this problem by controlling the prior complexity via the number steps in the prior distribution, which is more intuitive and easier to tune.

Multiple scoring functions were proposed for each model to find both the point-based and distribution-based anomalies. Although it is tempting to find a ubiquitous scoring function, such attempts are usually futile as the definition of anomalies depends on specific problem *e.g.* the importance of point-based and distribution-based anomalousness are different in different problems or for different users. In practice, we suggest try multiple scoring functions to find the anomalies that are particularly interesting.

Apparently, when $S = 1$ NGM is equivalent to MGM. To further see their relationship, note that the marginal likelihood of NGM (4.22) can also be written as

$$p(G_m|\rho, \alpha, \pi, \eta) = \sum_{r_m} \left(\prod_k \pi_{k,r_{mk}} \right) \sum_{y_m} \rho_{y_m} \prod_n \sum_{z_{mn}} \alpha_{y_m, z_{mn}} p(x_{mn}|\eta_{z_{mn}, r_{m,z_{mn}}}). \tag{4.33}$$

Comparing (4.33) to the MGM marginal likelihood (4.2), we see that NGM can be considered as a mixture of $K \times S$ dependent MGM models (using only $K \times S$ independent Gaussian components), with structured mixing weights specified by π. We found that NGM behaves like an MGM model with $K \times S$ topics. Further, note that the pseudo-prior scores for NGM described in Section 4.6.2 can also be applied to MGM. Considering all their similarities, we conclude that when simplicity and efficiency are important, NGM can be replaced by MGM when the number of topics is large. Indeed in practice we found that the difference between MGM and NGM are insignificant, which makes the simpler MGM a more cost-effective choice.

The genre models are able to learn from groups with different numbers of points effectively. In training, the smaller groups have less influence on the likelihood of the data. In prediction, the scoring functions assume that a group is normal unless some evidence of anomaly is observed, and the anomalousness increases as the group size becomes larger (between two anomalous groups with the same distribution of points, the larger group has a higher anomaly score). These behaviors help us ignore the noises and focus on the real anomalies.

In addition to detecting group anomalies, the genre models can also be used to accomplish other learning tasks. For example, as in [50] the genres can be used to cluster the groups together. Using techniques similar to naïve Bayes, we can use genre models to classify groups. In addition, the genres and topic generators provides a natural summary of the data that can help us explore the data sets.

Finally, the generative methods can also be used learn structured groups, where a point might depend on other points in the same group. The idea is that, first for each group we find a generative model to generate the points in it, then we find a global mechanism to generate those generative models. In genre models, the generative models are mixtures of topics, while the global mechanism is realized by the genres and the topic generators. To handle structured groups, we can use generative models such as the *hidden Markov models* (HMM) and the *random Markov field* (MRF), and then try to design suitable mechanism to generate the HMMs and MRFs .

4.8 Experiments

In this section we provide empirical results produced by the genre models on both synthetic and real data. We demonstrate the behaviors of different models, and show their effectiveness in detecting various group anomalies.

4.8.1 Synthetic Data

First, we demonstrate the behaviors of the genre models and the scoring functions on a synthetic data set. The data set is described below. We generated the data using 2-dimensional *Gaussian mixture models* (GMM). Each group has a GMM to generate its points. All GMMs share three Gaussian components with covariance $0.2 \times \mathbf{I}_2$ and centered at points $(-1.7, -1)$, $(1.7, -1)$, and $(0, 2)$, respectively. A group's mixing weights are randomly chosen from $w_1 = [0.33, 0.33, 0.33]$ or $w_2 = [0.84, 0.08, 0.08]$. Thus, a group is *normal* if its points are sampled from these three Gaussians, and their mixing weights are close to either w_1 or w_2. To test the detectors, we injected both point-based and distribution-based anomalies. point-based anomalies were groups of points sampled from $\mathcal{N}((0,0),\mathbf{I})$. Distribution-based anomalies were generated by GMMs consisting of normal Gaussian components but with mixing weights $[0.33, 0.64, 0.03]$ and $[0.08, 0.84, 0.08]$, which were different from w_1 and w_2. We generated $M = 100$ groups, each of which had $N_m \sim Poisson(100)$ points. One point-based anomalous group and two distribution-based anomalous groups were injected into the data set.

The detection results of MGM, FGM, NGM, as well as LDA are shown in Fig. 4.5. For LDA, we use FGM's point-based score (4.20). We show 12 out of the 100 groups. Normal groups are surrounded by black solid boxes, point-based anomalies have green dashed boxes, and distribution-based anomalies have red dashed boxes. Points are colored by the anomaly scores of the groups (darker color means more anomalous). An ideal detector would make dashed boxes' points dark and solid boxes' points light gray. The method postfix "-D" means distribution-based scores, and "-P" means point-based scores.

We can see that the genre models can all find the distribution-based anomalies since they are able to learn the complex distribution of the topic weights. But LDA lacks the flexibility to capture the simple yet multi-modal distribution of the topic weights in this data set. When we merge the results of the point-based and distribution-based scores, all the injected group anomalies can be found by the genre models. We notice that MGM is less sensitive to the point-based anomaly. The explanation is simple; the anomalous points are distributed in the middle of the topics, thus the inferred topic weight is around $[0.33, 0.33, 0.33]$, which is exactly w_1. As a result, MGM infers this group to be normal, although it is not. This example shows one possible problem of scoring groups based on topic weights only. On the other hand, with adaptive topics, FGM and NGM managed to identify the point-based anomaly even with the distribution-based scoring function. We also observed that the MGM-P score is slightly more noisy than FGM-P and NGM-P, probably because MGM is scoring individual points while FGM and NGM are scoring topics that are more stable.

Figures 4.6b – 4.6c show the density estimations produced by LDA, MGM, and FGM, respectively, for the point-based anomalous group. We can see that FGM gives a better estimation due to its adaptive topics, while LDA and MGM are limited to use their global topics. Figure 4.6d shows the learned genres visualized as the distribution $\sum_t \rho_t \mathcal{D}ir(\cdot|\alpha_t)$ on the topic simplex. This distribution summarizes the normal topic weights in this data set. Observe that the two peaks in the probability simplex are very close to w_1 and w_2 indeed.

4.8.2 Image Data

In this experiment we test the performances of the genre models on detecting anomalous scene images. We use the OT data set from [126], which contains 8 outdoor scene categories. There are $2,688$ images in total, each having about 256×256 pixels. A more detailed description of this data set can be found in Section 5.6.3.

We use the first 100 images from each category in our experiments. The images are represented as in [50]: we treat each image as a group of local patches. We densely sample about 400 patches on a regular grid from each image, and on each patch extract the 128-dimensional SIFT [106] feature vector, and then reduce its dimension to 10 using PCA.

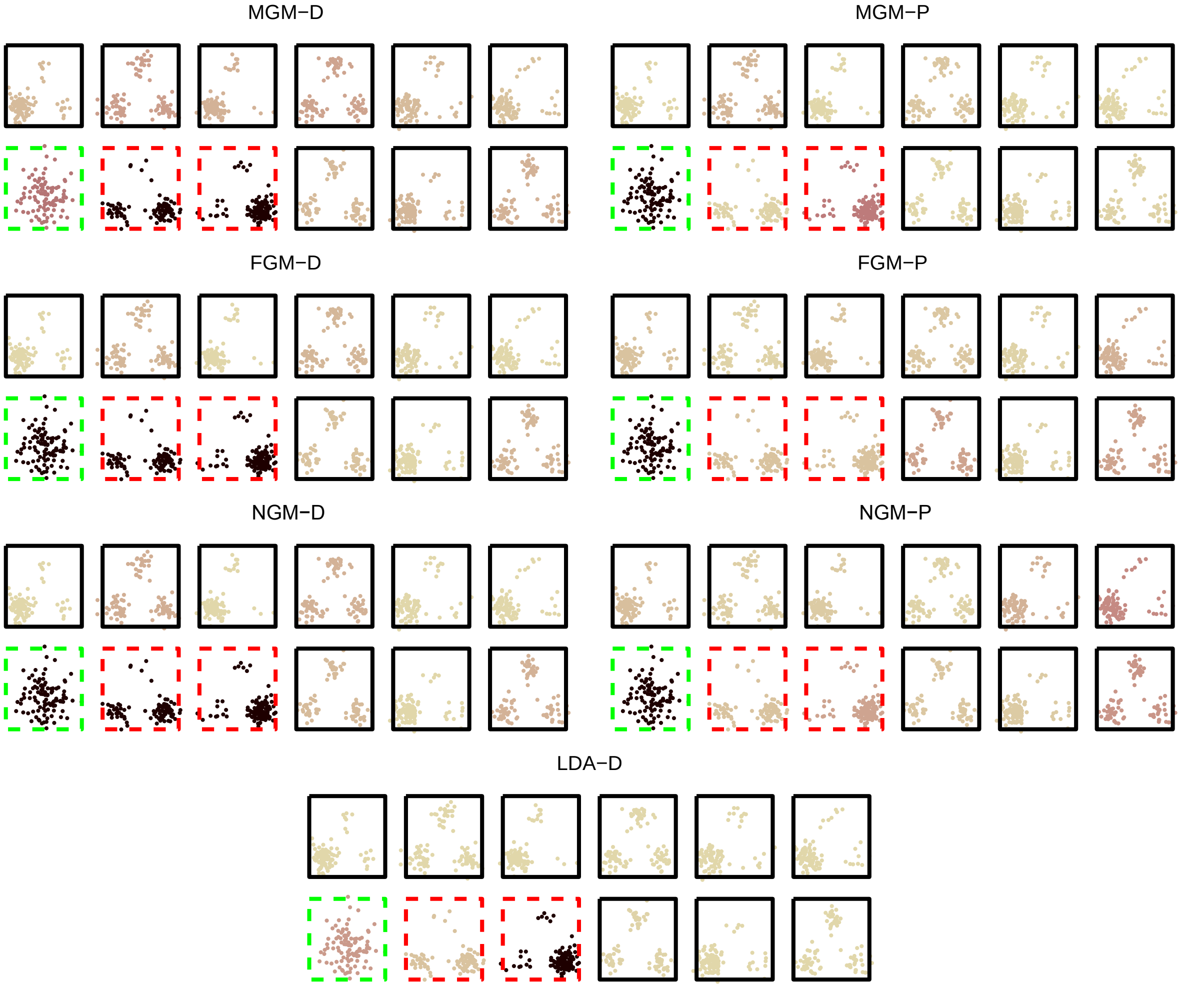


Figure 4.5: Detection results on the synthetic data. Black boxes are normal groups. Green dashed boxes are point-based anomalies. Red dashed boxes are distribution-based anomalies. The method postfix “-D” means distribution-based scores, and “-P” means point-based scores.

In addition to the genre models, we also test several other simple detector to compare. A Gaussian mixture models (GMM) based method is implemented to detect point-based anomalies. This method flatten the groups and fits a GMM to all the training data points. Then it computes the points’ likelihood in the test groups under the GMM as their anomaly scores, and finally scores a group by averaging the points’ scores. In other words, the GMM method finds groups with the most points in the low-density regions. To be able to detect distribution-based anomalies, we also implemented another competitor called LDA-KNN. LDA-KNN uses LDA to estimate the topic weights in the groups and treats these topic weights (parameter vectors of the multinomials) as the groups’ features. Then, a KNN based point anomaly detector [182] is used to score the groups’ feature vectors. Finally, we examine an adaptation of the Theme Model (ThM) [50]. The original ThM handles only discrete data and was proposed for clustering. To handle continuous data, we modified ThM by using Gaussian topics. Essentially, ThM is a simplified version of FGM

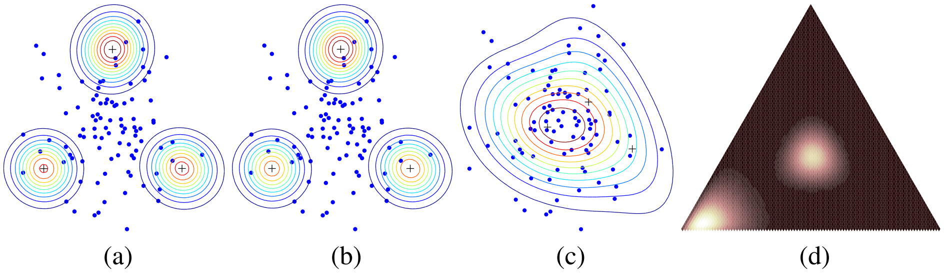


Figure 4.6: (a),(b),(c) show the density of the point-based anomaly estimated by LDA, MGM, and FGM respectively. In LDA and MGM, topics must be shared globally, therefore their perform badly. (d) The genres in the synthetic data set learned by FGM.

without the adaptive topics. We can then apply the scoring function (4.20) to find distribution-based anomalies, and the data likelihood to find the point-based anomalies. The following list summarizes the detectors:

- **P**: point-based detector using GMM.
- **MGM-P**: point-based detector using the MGM likelihood.
- **MGM-PP**: point-based detector using MGM with the pseudo-prior scorer (4.31).
- **ThM-P**: point-based detector using the ThM likelihood.
- **NGM-P**: point-based detector using NGM with the pseudo-prior scorer (4.31).
- **FGM-P**: point-based detector using FGM with the scorer (4.19).
- **LDA-KNN**: distribution-based detector using KNN and the topic weights learned by LDA.
- **MGM-D**: distribution-based detector using MGM with scorer (4.12).
- **ThM-D**: distribution-based detector using ThM with scorer (4.20).
- **NGM-D**: distribution-based detector using NGM with scorer (4.12).
- **FGM-D**: distribution-based detector using FGM with scorer (4.20).

For all the models we used $K = 8$ topics and $T = 6$ genres as suggested by BIC searches. For FGM, we set $\kappa_0 = \nu_0 = \bar{N}$ where $\bar{N}$ is the average size of the groups (hence the topic generators in FGM have low variances). For NGM, we set $S = 3$ so that each topic generator contains 3 possible elements. The performance is measured by the *area under the ROC curve* (AUC) of retrieving the anomalies from the test set.

Finding Out-of-Category Anomalies

The first type of image anomalies we test are out-of-category anomalies. In each run, we randomly select one category as the normal class and use its images to train the genre models. At test time, we mix images from another category into some normal images as anomalies, and ask the models to find them. The anomalies in this experiment are not controlled and can be of any type. Note that the training and testing images do not overlap. In each run, we select one normal category and one abnormal category. Then, we use 80% of the images in the normal category for training, and use the rest 20% combined with the images in the abnormal category for testing.

The results of 56 random runs are reported in Figure 4.7. In general, point-based detectors did better than the distribution-based detectors, which is expected since different scene images are likely to have distinctive patches. Our method MGM-PP, NGM-P, and FGM-P did significantly better than others. Note that the point-based detector MGM-PP performed better than the MGM-P, showing the advantage of the pseudo-prior approach. FGM-P performed the best. Though not apparent in the boxplot due to the high variance of data, the advantage of FGM-P is significant: between FGM-P and the next best NGM-P, the p-value of the Wilcoxon signed rank test is 0.035 (the signed rank test was used because the distribution of the accuracies were highly skewed. For reference the paired t-test has a p-value of 0.028). On the other hand, several distribution-based detectors also did well, but ThM-D and FGM-D failed this task because in this complex data set the estimation of Dirichlet genres became unstable. Finally, we observe that MGM and NGM performed very similarly.

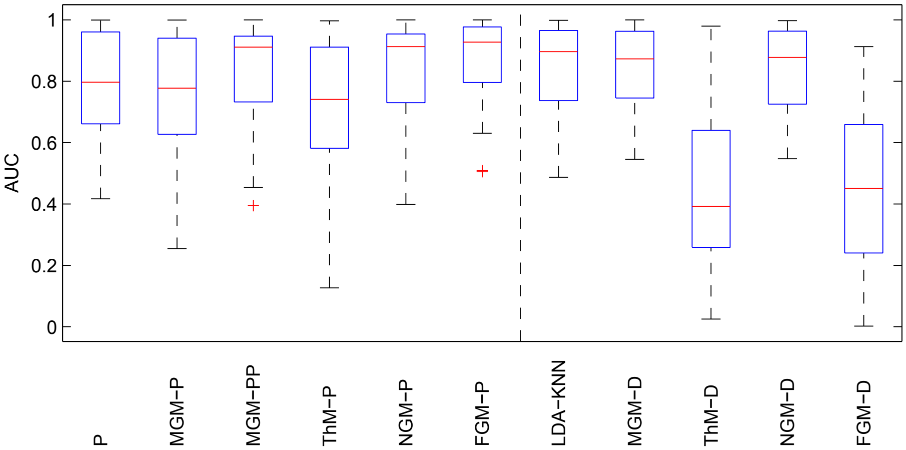


Figure 4.7: Performances on detecting out-of-category images. See text for details.

Finding Stitched Images

The second type of image anomalies are stitched images. The purpose here is to find unnatural images. In each run, we select two categories as the normal classes, and then divide the images in these two classes into training and testing sets. We create anomalies by stitching random pairs of images (horizontally half-by-half) from different categories in the testing set. The stitched images are then added to the testing set, and the goal is to find these unnatural synthesized images. Note that unlike the previous experiment, the anomalies here are controlled; the normal test images and the anomalies consist of exactly the same patches, and none of them overlap with the training images. For instance, an anomaly may be a picture that is half mountain and half city street. Some examples are shown in Figure 4.8. When extracting the SIFT features, points near the stitching boundaries are discarded to avoid boundary artifacts.

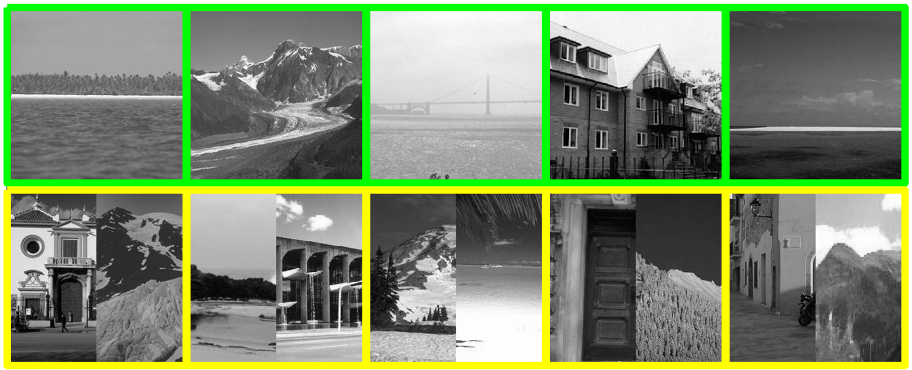


Figure 4.8: Images samples. Green boxes (first row) contain natural images, and yellow boxes (second row) contain stitched anomalies.

We use the same data as the previous experiment. In each run, we randomly select two categories and use 80% of the images in both categories for training. The rest 20%, combined with the synthesized stitched images, are used for testing. The number of normal testing images and the number of anomalies are equal.

The performances from 56 random runs are shown in Figure 4.9. As expected, in contrary to the previous experiment, the distribution based methods are much better than the point-based methods since by construction there is no point-based anomalies. Particularly, the methods that are based on Dirichlet genres, including ThM-D and FGM-D, lead the performance by a large margin. The difference between ThM-D and FGM-D are negligible, meaning that the adaptive topics of FGM had little use since there are no point anomalies. Again, MGM and NGM performed similarly.

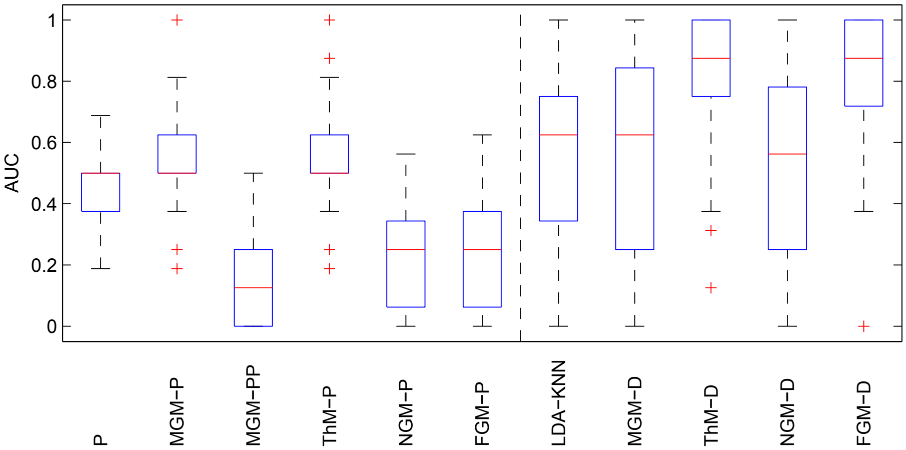


Figure 4.9: Performances on detecting stitched images.

4.8.3 Turbulence Data

We present an explorative study of detecting group anomalies on turbulence data from the JHU Turbulence Database Cluster[1] (TDC) [127]. TDC simulates fluid motion through time on a 3-dimensional grid, and here we perform our experiment on a continuous 128^3 sub-grid. In each time step and each vertex of the grid, TDC records the 3-dimensional velocity of the fluid. We consider the vertices in a local cubic region as a group, and the goal is to find groups of vertices whose velocity distributions (*i.e.* moving patterns) are unusual and potentially interesting. The following steps were used to extract the groups: (1) We chose the $\{(8i, 8j, 8k)\}_{i,j,k}$ grid points as centers of our groups. Around these centers, the points in 7^3 sized cubes formed our groups. (2) The feature of a point in the cube was its velocity relative to the velocity at its cube's center point. After these pre-processing steps, we had $M = 4\,096$ groups, each of which had 342 3-dimensional feature vectors.

We applied MGM-D, ThM-D, and FGM-D to find anomalies in this group data. $T = 4$ genres and $K = 6$ topics were used for all methods. We do not have a groundtruth for anomalies in this data set. However, we can compute the "vorticity score" [115] for each vertex that indicates the tendency of the fluid to "spin". Vortices and especially their interactions are uncommon and of great interest in the field of fluid dynamics. This vorticity can be considered as a hand crafted anomaly score based on expert knowledge of this fluid data. We do not want an anomaly detector to match this score perfectly because there are other "non-vortex" anomalous events it should find as well. However, we do think higher correlation with this score indicates better anomaly detection performance.

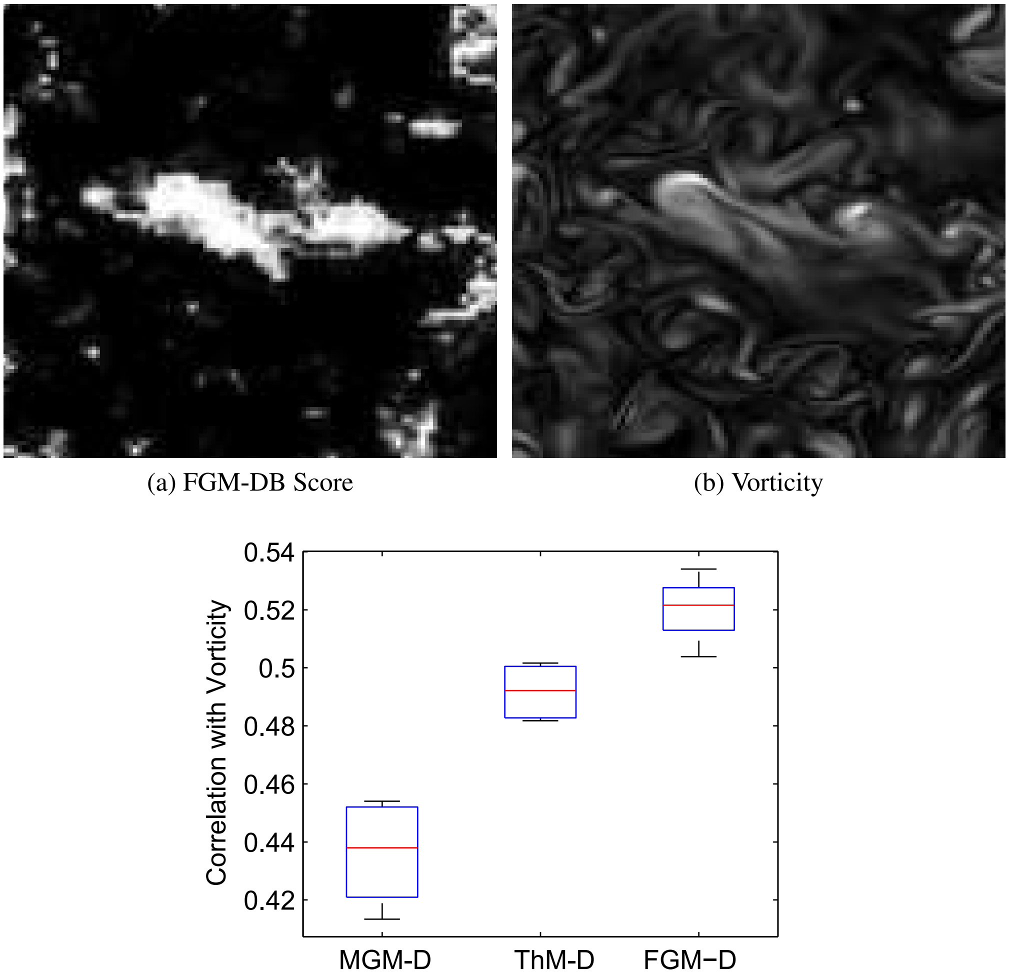
Figure 4.10 visualizes the anomaly scores of FGM and the vorticity. We can see that these pictures are highly correlated, which implies that FGM was able to find interesting turbulence activities based on velocity only and without using the definition of vorticity or any other expert knowledge. Correlation values between vorticity and the MGM, ThM, and FGM scores from 20 random runs are displayed in Fig. 4.10c, showing that FGM is better at finding regions with high vorticity.

4.9 Summary

We presented an parametric, generative approach to model collective data, and use it for the group anomaly detection problem. Using topic modeling techniques, we proposed the *Multinomial genre models* (MGM), the *flexible genre models* (FGM), and the *nonparametric genre models* (NGM) that are able to capture complex group behaviors at multiple levels, while archiving a better balance between the model flexibility and learning efficiency progressively. Several scoring function are also proposed specifically to exploit the capability of the models to detect group anomalies. Empirical results show that genre models can model the generating process of the collective data and detect various group anomalies well. However, since the anomalies vary a lot depending on the data sets, we need to choose the best model and scoring function in order to achieve the best results.

[1] http://turbulence.pha.jhu.edu

In the future, we would like to make the genre models robust. So far we have used the genre models to find outliers in the testing set. However, when the training set is contaminated by outliers, the learned model might be distorted towards the outliers. Therefore, we want the model to only learn normal behaviors even if it was trained on a data set that contains a few outliers. Initial attempt has been made in this direction using long-tail distributions (*e.g.* the *student-t*'s distribution) as the model parameters, yet the resulting model is overly complex, involving many free parameters, and unstable during practice. We shall continue to investigate more reliable approaches. Using techniques from *Gaussian processes* [137], we can also extend the genre models to functional observations where each point is a noisy observation of a function.



(a) FGM-DB Score

(b) Vorticity

(c)

Figure 4.10: Detection results for the turbulence data. (a) & (b) FGM-DB anomaly score and vorticity visualized on one slice of the cube. (c) Correlations of the anomaly scores with the vorticity.

MLE of the Gaussian-Inverse-Wishart Distribution

We present the MLE for a Gaussian-Inverse-Wishart (GIW) distribution $GIW(\mu_0, \kappa_0, \Psi_0, \nu_0)$ given a set of Gaussian distributions with parameters $\beta = \{\beta_m = (\mu_m, \Sigma_m)\}_{m=1,\ldots,M}$. By iteratively updating the parameters, we converge to a stationary point of the likelihood function $L_{GIW}(\beta; \mu_0, \kappa_0, \Psi_0, \nu_0)$. For μ_{0k}, κ_{0k}, and Ψ_{0k}, we can derive direct solutions by setting the partial derivatives of the log-likelihood to zero:

$$\mu_0 = \left\{ \sum_m \Sigma_m^{-1} \right\}^{-1} \sum_m \Sigma_m^{-1} \mu_m \tag{4.34}$$

$$\kappa_0 = \frac{MD}{\sum_m (\mu_m - \mu_0)^T \Sigma_m^{-1} (\mu_m - \mu_0)}, \tag{4.35}$$

$$\Psi_0 = \nu_0 \left\{ \frac{1}{M} \sum_m \Sigma_m^{-1} \right\}^{-1}, \tag{4.36}$$

where D denotes the feature dimension.

The partial derivative *w.r.t.* ν_0 is given as follows:

$$\frac{\partial L}{\partial \nu_0} = \frac{1}{2} \left\{ M \log \frac{|\Psi_0|}{2^D} - \sum_m \log |\Sigma_m| - \psi_D(\nu_0/2) \right\}, \tag{4.37}$$

where $\psi_D(\cdot) = \frac{\Gamma'_D(\cdot)}{\Gamma_D(\cdot)}$ stands for the first order derivative of the multivariate log-gamma function. ν_0 does not have an analytical solution. To address this issue, observe that ν_0 is a scalar, and Ψ_0 has a simple linear dependency on ν_0. Thus, we can apply a one dimensional search to find the optimal ν_0, *e.g.* , using numerical differentiation. Having ν_0 we can compute Ψ_0.

Chapter 5

Discriminative Methods for Collective Data

We introduce a new discriminative learning method for classification on collective data. Unlike generative models, discriminative methods tries to learn target concepts directly regardless of the generating mechanism of data. In this chapter we describe discriminative ways of learning from collective data based on similarity or dissimilarity measures between groups. The advantage of this approach is its great flexibility, since we can take advantage of the existing tools that rely on similarities to accomplish a vast variety of tasks on groups. We use consistent nonparametric divergence estimators to define new kernels over the groups/sets, and then apply them in kernel classifiers. Our results on image classification demonstrate that in many cases this approach can outperform state-of-the-art competitors on both simulated and challenging real-world datasets.

5.1 Introduction

We propose new methods for the *classification* of distributions. In the classification problem our goal is to find a map from the space of distributions to the space of class, while in the anomaly detection problem we want to find distributions that are unlike others. Note that only finite *i.i.d.* samples are observed from these distributions. For this purpose we extend the *support vector machines* (SVM) to the space of distributions. In our framework, some of the distributions in the training data will play the role of support vectors.

We consider this problem in the context of image classification. There are numerous examples in computer vision where images are represented by unordered sets of feature vectors. For example, the shapes of an object can be represented by sets of local descriptors at edges and corner points [63]. Human faces can also be described by sets of local image patches containing certain facial parts. The SIFT [106], HOG [36], and PHOG [8] features extractors find stable image representations by detecting sets of local-affine invariant regions and other regions of interest.

To compare images represented by feature sets, a straightforward approach is to treat the sets as if they contained instances sampled from an unknown and possibly high-dimensional distribution. A common way to handle these distributions is to use (high-dimensional) histograms through discretization, and compare these histograms. The popular “Bag-of-words” (BoW) algorithms use this approach: they treat each image as a set of visual words, where the words are obtained by clustering local image patches [50, 97].

Histogram-based representations have been used in many state-of-the-art computer vision algorithms. However, they have some obvious limitations. When we discretize continuous distributions into bins, we might lose valuable information. This problem is especially severe in high dimensions, where the curse of dimensionality makes histogram-based density estimators unreliable. Selecting the bin sizes (or number of bins) for the histograms are also difficult model selection problems.

In this chapter we propose new classification algorithms that operate directly on the set-of-vectors representation of the images. We assume that the elements of these sets are *i.i.d.* sample points from unknown distributions that characterize the images. In order to classify the images, we classify these distributions based on their *i.i.d.* sample set representations. The kernel-based approach is adopted: we introduce and estimate the kernel functions between these distributions. Having the estimated kernel matrix, we then apply kernel classifiers such as SVM for classification. The proposed kernels avoid the traditional clustering, quantization, or histogram building steps that could lead to loss of information.

These kernel functions on sets will be defined in terms of divergences/distances, just as the Euclidean distance is used to define Gaussian/RBF kernels on vectors. To this end, we will need to estimate the divergences between distributions. A straightforward approach would be to estimate the underlying densities and plug them into the corresponding divergence formulae. In fact, histogram and BoW approaches follow this paradigm. Density estimation, however, is among the most difficult problems in statistics due to the curse of dimensionality. To avoid this problem, we develop our kernels based on a direct (no density estimation required) and nonparametric (minimal assumptions about the true distributions) approach. We show how to estimate a large family of divergences that includes the Rényi, Tsallis, Hellinger, Bhattacharyya, KL, L_2, and many other divergences. The estimator is provably consistent, nonparametric, and does not use histograms, kernel density estimators (KDE), or any other density estimators. It depends on only simple k-nearest neighbor (KNN) statistics.

We evaluate the empirical performance of the proposed kernels on both simulated and real-world datasets, and compare them to alternatives based on density estimation or parametric approximations. We show that our kernels achieve performances that match or beat the state of the art in several image classification tasks.

The chapter is organized as follows. In the next section we review some related work. We formally introduce the distribution classification problem and show how to define kernels on distributions in Section 5.3. Section 5.4 and 5.5 describes how to estimate the kernels on distributions when the densities are unknown. Section 5.6 presents the results of numerical experiments. We conclude with a discussion in Section 5.7.

5.2 Related Work

Although several methods exist to measure the distance between sample sets, and kernels have also been defined on sets, all of these previous methods have their shortcomings. We will now review the most popular methods.

Nguyen et al. recently proposed a method for f-divergence estimation using its so-called "variational characterization properties" [125]. This approach involves an intractable optimization over an infinite-dimensional function space. When this function space is chosen to be a reproducing kernel Hilbert space (RKHS), this optimization problem reduces to an N-dimensional convex problem, where N is the sample size. This can be very demanding in practice for a only few thousand sample points, which is quite common in computer vision applications.

There are RKHS based approaches for defining kernels on unordered sets as well. The method proposed by Smola *et al.* [155] uses the interaction between pairs in the sample set, and hence its computation time is $O(m^2)$. The divergence estimator we propose, by contrast, uses only KNN distances in the sample set, a well-studied problem with efficient solutions such as k-d trees. Note also that choosing an appropriate kernel function for the RKHS can be a difficult model selection problem, a challenge not faced by our proposed divergence estimator.

Sricharan et al. [157] developed k-nearest-neighbor based methods similar to our method for estimating non-linear functionals of the density, of which divergences are a special case. In contrast to our approach, however, their method requires k to increase with the sample size N and diverge to infinity. KNN computations for large k values can be very computationally demanding. In our approach we fix k on a small number (typically between 1 and 5), and are still able to prove that the divergence estimator is consistent.

Jebara and Kondor [76] have also studied the question of how to define kernels on distributions. Their approach fits a parametric family (*e.g.* exponential family) density to each set of points, and then using these fitted parameters estimates the inner products between the densities. Moreno et al. [119] also fit a parametric density to the data and use it to define a KL divergence-based kernel. Parametric approaches can work better than nonparametric methods when the sample size N is small, or if we know from prior knowledge that the true densities belong to these parametric families. When the assumptions do not hold, however, parametric methods introduce bias in estimating the inner products between densities. In contrast, our proposed method is completely nonparametric and provides provably asymptotically unbiased kernel estimations for certain kernels.

Kondor and Jebara [83] earlier introduced a kernel between distributions defined as Bhattacharyya's measure of affinity between finite dimensional Gaussians in a Hilbert space. This approach fits a Gaussian distribution to the features in a Hilbert space, but it can lead to a large bias when the data in the Hilbert spaces is not Gaussian. Furthermore, the approach is developed only for Bhattacharyya's measure. Our proposed method is asymptotically unbiased and can be used for many other divergences.

The Pyramid Matching Kernel [63], which also operates over unordered sets, has recently become popular in computer vision. In this approach each feature set is mapped to a multi-resolution histogram. These histogram pyramids are compared using a so-called "weighted histogram intersection computation." A shortcoming of this approach is that it needs to calculate D-dimensional histograms, which can become very inefficient for large D due to the curse of dimensionality. Selecting appropriate bin sizes is also a difficult problem for which only heuristics are known [150].

Póczos *et al.* [130] used a slightly less general version of our nonparametric divergence estimator similar to solve certain machine learning problems in the space of distributions. That work studied only simple KNN based classifiers, however. Here we use kernel methods that are more discriminative in classification tasks, and evaluate their performance on various image datasets.

5.3 Problem Definition

In this section we formally define our set classification problem and show how kernel classifiers can be generalized to sample sets of distributions. Assume we have M inputs $\{G_1, \ldots, G_M\}$ each representing one image, where the mth input G_m contains *i.i.d.* samples from some underlying density f_m. That is, G_m is a set of sample points, and $x_{m,j} \sim f_m$ for $j = 1, \ldots, N_m$. Let $\mathcal{G}$ denote the set of all such sample sets *i.e.* $G_m \in \mathcal{G}$, $m = 1, \ldots, M$.

Further assume we are given M labels for these inputs $\{(G_m, y_m)\}_{m=1}^{M}$. Here $y_m \in \mathcal{Y} \doteq \{y_1, \ldots, y_c\}$ denotes the class label of the mth set. We seek a function $h : \mathcal{G} \to \mathcal{Y}$ such that for a new input and output pair $(G, y) \in \mathcal{G} \times \mathcal{Y}$ we ideally have that $h(G) = y$. For simplicity, we discuss only binary classification. The ideas below can be extended to c-class classification in the standard ways.

SVM is one the most successful methods in estimating such functions. In order to use SVM, we need to be able to evaluate the kernel between the inputs. In our case, we need a kernel function on $\mathcal{G} \times \mathcal{G}$ that returns real values. Once we evaluated such kernels and obtained the kernel matrix *a.k.a.* Gram matrix, existing SVM algorithm can be used for classification. Having the kernel matrix, we can also accomplish many other learning tasks on sets such as clustering using spectral clustering[122], dimensionality reduction using kernel PCA [147], anomaly detection using one-class SVM [148], and so on. All of these urge us to find a good kernel matrix for the groups.

5.4 Nonparametric Kernel Estimation

Having two finite *i.i.d.* sample sets from densities f_1 and f_2, we need to estimate $k(f_1, f_2)$, the kernel value between them. Many kernels, *i.e. positive semi-definite* (PSD) functionals of f_1 and f_2 can be constructed from

$$D_{\alpha,\beta}(f_1 \| f_2) = \int f_1^{\alpha}(x)\, f_2^{\beta}(x)\, f_1(x)\, \mathrm{d}x, \tag{5.1}$$

where $\alpha, \beta \in \mathbb{R}$. For example, we can use Eq. (5.1) to construct *Linear* $\big(k(f_1, f_2) = \int f_1 f_2\big)$, *polynomial* $\big(k(f_1, q) = \big(\int f_1 f_2 + c\big)\big)^s$), and *Gaussian* $\Big(k(f_1, f_2) = \exp(-\frac{1}{2}\mu^2(f_1, f_2)/\sigma^2), \mu(f_1, f_2) = \int f_1^2 + f_2^2 - 2f_1 f_2\Big)$ kernels.

For the Gaussian kernel, which we primarily use in this chapter, one can also use other "distances". For example, we can use the *Hellinger distance* with $\mu(f_1, f_2) = 1 - \int \sqrt{f_1 f_2}$. Another important family of divergences is the *Rényi-α divergence*, where

$$\mu(f_1, f_2) = \frac{1}{\alpha - 1} \log \int f_1^{\alpha} f_2^{1-\alpha}.$$

Note that the KL-divergence is a special case of the Rényi divergence when $\alpha \to 1$. These divergences are nonnegative and vanish iff $p = q$ almost surely. Nonetheless, the divergences are usually not symmetric, do not satisfy the triangle inequality, and do not lead to PSD kernel matrices. In Section 5.5 we will show how to address this problem.

To estimate $D_{\alpha,\beta}(f_1\|f_2)$ for some α, β values, we use the tools that have been applied for Rényi entropy [96], Shannon entropy [61], KL divergence [168], and Rényi divergence estimation [129]. We show how to estimate $D_{\alpha,\beta}(f_1\|f_2)$ in an efficient, nonparametric, and consistent way.

Let $G_1 \doteq \{x_1, \ldots, x_{N_1}\}$ be an *i.i.d.* sample from f_1, and similarly let $G_2 \doteq \{z_1, \ldots, z_{N_2}\}$ be an *i.i.d.* sample from f_2. Let $\rho_k(i)$ denote the Euclidean distance between x_i and its kth nearest neighbor in G_1, and similarly let $\nu_k(i)$ denote the distance between x_i and its the kth nearest neighbor in G_2. Based on [130], we can use the following estimate

$$\widehat{D}_{\alpha,\beta} = \frac{B_{k,\alpha,\beta}}{N_1(N_1-1)^{\alpha}N_2^{\beta}} \sum_{i=1}^{N_1} \rho_k^{-d\alpha}(i)\, \nu_k^{-d\beta}(i), \tag{5.2}$$

where $B_{k,\alpha,\beta} \doteq \bar{c}^{-\alpha-\beta} \frac{\Gamma(k)^2}{\Gamma(k-\alpha)\Gamma(k-\beta)}$. Under certain conditions, we can prove that $\widehat{D}_{\alpha,\beta}$ is a consistent estimator of $D_{\alpha,\beta}$, and thus by plugging these estimators into kernels we get consistent estimators for those kernels. It means that the more sample points we have the better the quality of the kernel estimation is, and eventually it is converging to the correct value.

To compute the estimate (5.2), all we need are the KNN distances $\rho_k(i)$ and $\nu_k(i)$ for every point x_i in group G_1. In low dimensions, nearest neighbors can be found in logarithm time using tree structures such as the KD-Tree, resulting in a time complexity of $O\left(\bar{N}\log(\bar{N})\right)$ for one pair of groups, where $\bar{N}$ is around the average size of the groups. In high dimensions, however, efficient search for neighbors becomes difficult and generally we can only examine the points one by one using linear time, resulting in a quadratic time complexity $O(\bar{N}^2)$. Since we have to compute for each pair of groups the estimated kernel to use kernel machines, the overall complexity becomes $O(M^2\bar{N}^2)$. As a remedy, we can parallelize the computations of different pairs of groups. Another solution is to use approximate nearest-neighbor search algorithms such as [121]. More discussions and solutions to the efficiency problem can be found in Chapter 7.

This estimator can also work on groups with different sizes and the consistency result still holds. However, since larger sample size tends to give more accurate estimate, in theory working with groups of different sizes might give estimates of different qualities in the same kernel matrix. Nevertheless, in practice we found this is usually not a problem; We show empirical results in Section 5.6 on groups of similar sizes as well as groups of very different sizes.

5.5 Constructing Mercer Kernels

Kernels constructed from $\widehat{D}_{\alpha,\beta}$ are not ready to be plugged into kernel machines like SVM. Even though the estimation is consistent, any particular estimated kernel/Gram matrix might not be *positive semi-definite* (PSD), which is required by SVM. There are two reasons for this problem: 1) the divergences themselves might not be Hilbertian metrics which is a necessary condition of producing PSD kernels [145]; 2) Estimation errors exist given the finite sample size. We therefore need to transform the raw estimated kernel matrix into a PSD matrix so that the underlying kernel is a valid Mercer kernel.

Here we project the raw kernel matrix to the cone of PSD matrices, and use that projected image, which is a PSD matrix, as the input to SVM. In other words, we are seeking for the PSD matrix that can best approximate the raw kernel matrix. To do this, we first symmetrize the estimated kernel matrix by taking half the sum of it and its transpose, and then project it to the cone of PSD matrices by discarding any negative eigenvalues (*i.e.* setting them to zeros) from its spectrum [71].

Rather than projecting the estimated kernel matrix and then solving an SVM, one can actually combine these two steps into a single convex problem [108]. We do not pursue this approach in this work, however.

When structures exist in the kernel matrix (*e.g.* the kernel matrix has a low rank), we can find better ways to construct PSD/Mercer kernels based on the raw estimations. This direction is further studied in Chapter 6.

We could also estimate distribution divergences that are Hilbertian metrics, and then use them to construct kernels. By doing this, we only have to deal with the estimation errors, and may obtain higher quality kernel matrices. [69, 70] proposed a family of Hilbertian metrics between probability distributions. These metrics can also be estimated using similar techniques as in (5.2). The consistency of such a family of estimates is yet to be studied, but several interesting special cases of this family, notably the *Jensen-Shannon divergence*, either coincide with the divergences mentioned in the previous section, or can be derived from (5.2) and [168]. We shall leave this possibility for the future work.

5.6 Experiments

In this section, we show the empirical performance of the proposed kernels in both simulation studies and real-world image classification tasks. Code and datasets used here are available at `autonlab.org/autonweb/20680.html`.

In all these tasks, the objects of interest are represented as "bags of vectors" (BoV), *i.e.* unordered sets of feature vectors. The proposed kernel estimators as well as several other kernels between sets of points are used to calculate kernel matrices for these sets. The full kernel matrices are projected to be symmetric positive semi-definite and given to a multi-class SVM for classification.

Nonparametric divergence kernels These kernels are based on the proposed nonparametric Rényi-α divergence estimators (**NPR-**α) and Hellinger distance estimators (**NPH**). We use the $k = 5$th nearest neighbors in these estimators, except in Section 5.6.1, where small sample sizes necessitate $k = 1$. For NPR, we test the performance with $\alpha \in \{0.5, 0.7, 0.9, 0.99\}$. Note that when $\alpha = 0.99$ the Rényi-divergence approximates the KL divergence, and when $\alpha = 0.5$ it is twice the Bhattacharyya distance.

Parametric kernels These kernels are based on a Gaussian or Gaussian Mixture Model (GMM) assumption. We first fit the density to each group, and then compute the KL-divergence (**G-KL**,

GMM-KL) [119] and *product probability kernels* (**G-PPK**, **GMM-PPK**) [76] with $\alpha = 0.5$ between the groups (therefore they are actually the *Bhattacharyya Coefficients* between Gaussians). Tuning the number of GMM components for each group is not feasible, so we always use 3 components. GMM-KL has no analytic form, so we use the *Monte Carlo* approximation with 500 samples.

BoW kernels To convert BoV to BoW, we quantize the feature to "visual words," and then compute the histogram of words for each group. The *chi-square distance* between these BoW histograms is used to construct the Gaussian kernel. The histograms can be further processed by PLSA [72] and then used in kernels based on Euclidean distance.

Pyramid matching kernel (PMK) We also use the *vocabulary-guided pyramid matching kernel* [64]; this variant performs better for high-dimensional data. We use the authors' implementation *libpmk*[1] with the suggested parameters.

Mean map kernel (MMK) We also consider the *mean map kernel* [155], also known as the *mean match kernel* [109] to the computer vision community. The MMK between two groups of vectors $G_1 = \{x_1, \ldots, x_{N_1}\}$ and $G_2 = \{z_1, \ldots, z_{N_2}\}$ is defined as $k_{MM}(G_1, G_2) = \frac{1}{N_1 N_2} \sum_{i=1,j=1}^{N_1,N_2} k(x_i, z_j)$. In other words, MMK is the average kernel matching score between every pair of points between the two groups. We let the point-wise matching kernel be the Gaussian kernel $k(x, y) = \exp\left(-\|x - y\|_2^2/\sigma^2\right)$, where the kernel width σ is tuned in the same way as other parameters using cross-validation. To avoid the high computational cost of MMK ($O(N_1 N_2)$ for each pair of groups), we randomly choose at most 500 points from each group to compute the MMK, so that the computation is affordable while the approximation error is small.

We use *LibSVM* [27]'s multi-class SVM for classification. All kernel matrices are projected to be symmetric PSD as in Section 5.5 before use. The penalty to points within the margin C is chosen from $\{2^{-9}, 2^{-6}, \cdots, 2^{18}\}$. For PPK and PMK, we use their kernel values directly. For other kernels, we use Gaussian kernels $\exp\left(-\frac{1}{2}\mu^2/\sigma^2\right)$, where μ is the divergence/distance. The kernel width σ is chosen from $\sigma_0 \times \{2^{-4}, 2^{-2}, \cdots, 2^{10}\}$, where σ_0 is the mean of the pairwise divergences. C and (when used) σ are chosen through joint 3-fold cross-validation on the training set.

For the image experiments, we extract features as follows unless indicated otherwise. The BoV representation we use is based on the *dense SIFT* descriptors. We put a regular 2D grid with step size 10 on each image, and compute SIFT descriptors on each grid node. These descriptors are 128-dimensional. In an attempt for scale invariance, we usually compute three SIFT descriptors with bin sizes of $\{6, 9, 12\}$ pixels at each point. After the feature extraction, each image is represented by a variable number of 128-dimensional feature vectors. Following [19], we can also include color information in the SIFT features by converting the images to HSV color space and separately extracting SIFT features from each color channel. Then SIFT features with the same location and bin size are concatenated together to construct the more descriptive "color SIFT" feature with dimensionality 384. Finally, we use PCA to reduce the feature vectors' dimensionality. Our implementation uses the PHOW function of the VLFeat package [164] for feature extraction.

[1] `people.csail.mit.edu/jjl/libpmk`

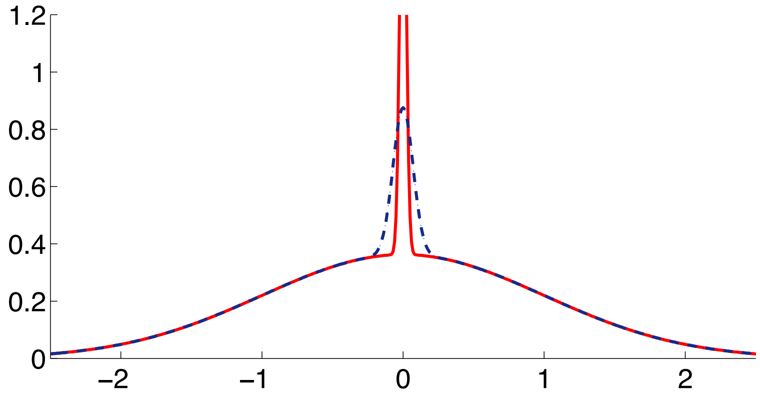


Figure 5.1: Densities of the two one-dimensional mixtures.

For BoW, these SIFT vectors are quantized by k-means into *visual words*, for which the vocabulary size (number of clusters) is 1000 for color images and 500 for grayscale images. The number of PLSA topics is 25, as in [19]. Following common practice in computer vision, the visual words are based on the original (uncompressed) feature vectors. Therefore the BoW methods do not compare to BoV kernels directly, as they are based on different features. In comparison, BoW loses information in the discretization step, while BoV kernels lose information when the feature dimension is reduced. We will show that our non-parametric kernels outperform BoW in most cases, perhaps indicating that less information is lost in PCA than in quantization.

We report kernel matrix construction times using 40 cores of a machine with four 12-core 2.3 GHz Opteron K10.5 processors. In this high-dimensional setting, k-d trees are ineffective, so we use simple brute-force search. Established techniques for approximate KNN should result in significant speedups with limited loss of performance. In each case, we estimated divergences for the Hellinger distance and Rényi-α divergence with 20 values of α: -1, -.5, -.2, .1, .2, .3, …, .9, .99, 1.01, 1.1, 1.2, 1.3, 1.4, 1.5, and 2.

5.6.1 Artificial Gaussian Mixture Classification

We first compare the proposed kernels to others on artificial problems, to demonstrate two advantages of our kernel: its relatively few parameters requiring fine-tuning and its effectiveness in high-dimensional problems.

Consider the problem of distinguishing between the two Gaussian mixtures illustrated in Figure 5.1. The two mixtures each have a standard normal distribution with mixture coefficient $\frac{10}{11}$; the two classes are distinguished by the variance of the other component, which can be either .005 or .0005. Our task is to learn a classifier which can distinguish samples of size 30 from these two mixtures. (Although most feature sets will have substantially more than 30 data points for a real-world image, having a low number of sample points parallels having a moderate number of sample points in a high-dimensional space.) Note that this problem is quite difficult, as the expected number of samples from the distinguishing mixture is below 3.

Figure 5.2 shows accuracies from 8 runs of 10-fold cross-validation accuracies for several ker-

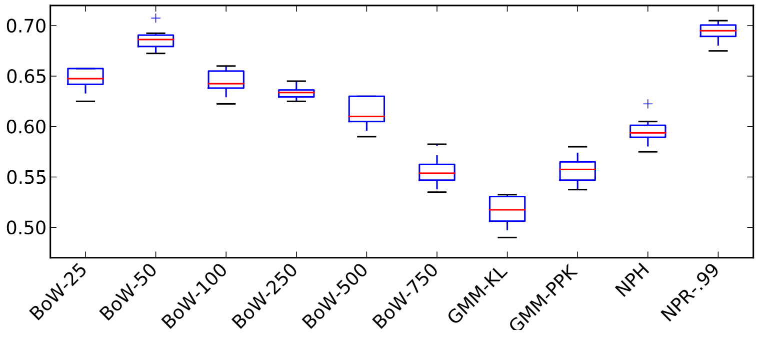


Figure 5.2: 1D mixture classification accuracies.

nels on a data set consisting of 200 samples from each mixture. The BoW method with codebook size K is denoted by BoW-K. The classification performance obtained by the Bayes-optimal classifier that chooses which mixture had a higher likelihood of generating the sample is 75%. The BoW kernel performs at its best only for codebook size 50; smaller and larger sizes both perform worse, some of them considerably so. In contrast, the proposed NPR and NPH methods perform well with minimal parameter selection, though it seems the Rényi divergence is better for this problem than the Hellinger.

We also show that our proposed kernel is capable of scaling up to higher-dimensional problems with small sample sets. This problem is similar, but the samples are of size 15 in $\mathbb{R}^D$. The common Gaussian has diagonal components 1 and off-diagonal components 0.2, while the distinguishing Gaussian has covariance matrix equal to either I_d or $I_d/2$, where I_d stands for the d-dimensional identity matrix. Each component has mean zero and mixture coefficient $1/2$. The distributions are more distinguishable in higher dimensions, as the components overlap less.

The results of 16 runs of 10-fold cross-validation for several kernels, as well as that of the Bayes-optimal classifier, are shown in Figure 5.3. The proposed NPR method outperformed its competitors in this experiment, and indeed achieved near-optimal results for all ds. $BoW - 500$ is the only BoW method shown, but other codebook sizes performed similarly. The dimensionality at which performance peaked varied with the codebook size, so that *e.g.* BoW-100 peaked at dimension 8, and BoW-1000 at 14.

5.6.2 Object Classification

In the following sections we compare the performances of various kernels on real-world image datasets. We first examine object classification in the ETH-80 [95] data set. This data set contains 8 categories of objects; each category has 10 different objects, and each object has 41 images from different view angles. Following [63], we use a subset of 400 images for the experiment, selecting 5 images per object that capture its appearance from different angles. Sample images of two objects are shown in Figure 5.4. Our goal is to classify these objects into the 8 categories.

For this data set, we extract the color SIFT features with bin size fixed at 6 pixels, as scale invariance is not necessary for this problem. We then reduce the SIFT features to 18 dimensions

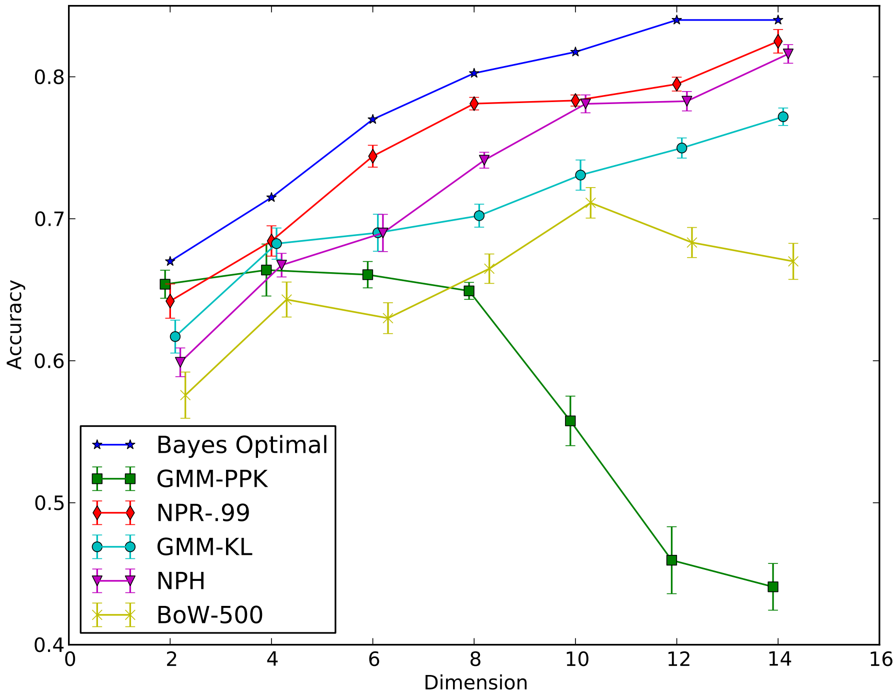


Figure 5.3: Mean and standard deviation accuracies on the high-dimensional artificial data set.

using PCA, preserving 50% of variance. Each image is then represented by 576 18-dimensional points. Constructing our proposed kernels took 47 seconds.

We report the performance of 16 random runs of 2-fold cross-validation in Figure 5.5. We can see that our Rényi-divergence kernels perform better than BoW, and much better than the other methods. We note that BoW achieved impressive results only when properly tuned, as in the simulation study of Section 5.6.1. The improvement of NPR-0.9 (mean accuracy 90.9%) over BoW (88.3%) is statistically significant: a *paired t-test* shows a p-value below 10^{-3}. It is also interesting to see that GMM-based methods perform worse than simple Gaussian-based methods. This may be because it is harder to choose the parameters of a GMM, or because divergences between GMMs could not be obtained precisely; both of those problems are infeasible to remedy. PMK is not very accurate here, though fast to compute.

Figure 5.6 shows the performance of the Rényi-α kernel for many values of α, along with the Hellinger performance for context. The best α values are clearly near 1, *i.e.* near the KL divergence, though performance seems to degrade faster when greater than 1 than when below.

5.6.3 Scene Classification

Scene classification using BoV/BoW representations is a well-studied problem for which many methods have been proposed (*e.g.* [19, 50, 135]). Here we test the performance of our non-parametric kernels against state-of-the-art methods.

We use the OT data set from [126], which contains 8 outdoor scene categories: *coast, forest,*

Figure 5.4: Images of two objects from the ETH-80 data set. Each object has 5 different views.

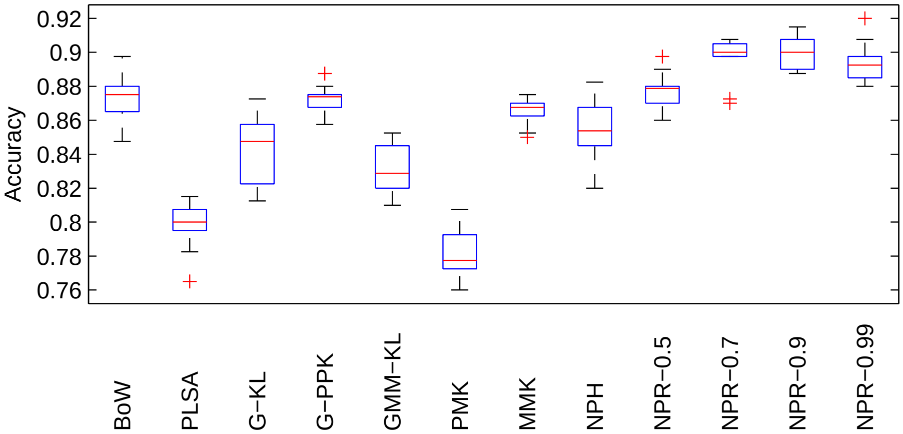


Figure 5.5: Classification accuracies on ETH-80.

highway, inside city, mountain, open country, street, and *tall building*. There are 2688 images in total, each about 256×256 pixels. Sample images are shown in Figure 5.7. The goal is to classify test images into one of the 8 categories.

We used the color SIFT features, and also append the relative y location of each patch (0 meaning the top of the image and 1 the bottom) onto the local feature vectors, allowing the use of some information about objects locations in the images in classification. (We chose not to include x coordinates, because horizontal locations of objects generally carry little information in these scene images). We used bin sizes of $\{6, 12, 18, 24, 30\}$. The larger patches are used so that more global information, such as the co-occurrences of local objects, can be captured. Using the above features, a typical image contains $1,815$ SIFT vectors, each of dimensionality 384; these are reduced by PCA to 53 dimensions preserving 70% of the variance, and then y coordinates are appended. Each dimension of the feature vectors was finally normalized to have zero mean and unit variance. Computing the nonparametric kernels on these larger, higher-dimensional points took $283,599$ seconds (about 3 days).

The accuracies of 16 random runs are shown in Figure 5.8. Here results of 10-fold cross-validations are used so that we can directly compare to other published results. GMM-PPK is not shown because it is too low. NPR-0.99 achieved the best average accuracy of 92.11%, which is

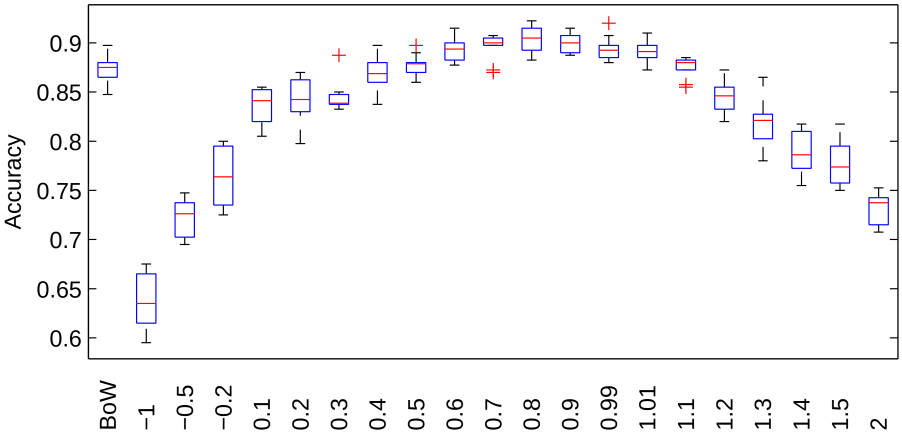


Figure 5.6: Classification accuracies on ETH-80 with Rényi-α for twenty α's, as well as the Hellinger distance.

much better than BOW's 90.26%. Notably, this 92.11% accuracy (std dev 0.18%) surpasses the best previous result of which we are aware, 91.57% [48]. For comparison, in 2-fold cross-validations the mean accuracies of NPR-0.99 and BOW are 90.85% and 88.21% respectively.

5.6.4 Sport Event Classification

The BoV kernels can also be used for visual event classification [100] in the same manner as for scene classification. We use the data set from [100], which contains Internet images of 8 sport event categories: *badminton, bocce, croquet, polo, rock climbing, rowing, sailing,* and *snowboarding*. This data set is considered more difficult than traditional scene classification, as it involves much more widely varying foreground activity than does *e.g.* the OT data set.

We use the first 130 images from each category, as in [100]. We use color SIFT features with dimensionality reduced to 57, and add spatial information in the form the patches' x and y coordinates. As image sizes vary, each BoV group contains 295 to $1,542$ vectors. Constructing our proposed kernels took $9,327$ seconds (2.5 hours).

Figure 5.10 shows the accuracies of 16 random 2-fold cross-validations. We again see the kernel based on the Rényi-$.9$ divergence achieve the best accuracy of 87.1% (std dev $.4\%$). This performance is at the same level as state-of-the-art methods such as [181], which attained 86.7%. It is worth noting that we used only PCA SIFT without further feature learning, as opposed to other methods which achieved significant performance increases by learning features. Compared to previous results, we can see that the performance of PPK methods decreased; we did not show GMM-PPK here because its accuracy is too low. The BoW method, though worse than Rényi-$.9$ with 83.5%, again performs reasonably well, showing its wide applicability.

Another interesting observation based on all the above results is that the nonparametric estimates of the Rényi divergences usually perform the best when α is close to 1 *i.e.* when it is close to the KL divergence. This can be viewed as an empirical support for the theoretical soundness of the KL divergences. On the other hand, in many cases the optimal α is usually slightly smaller than 1,

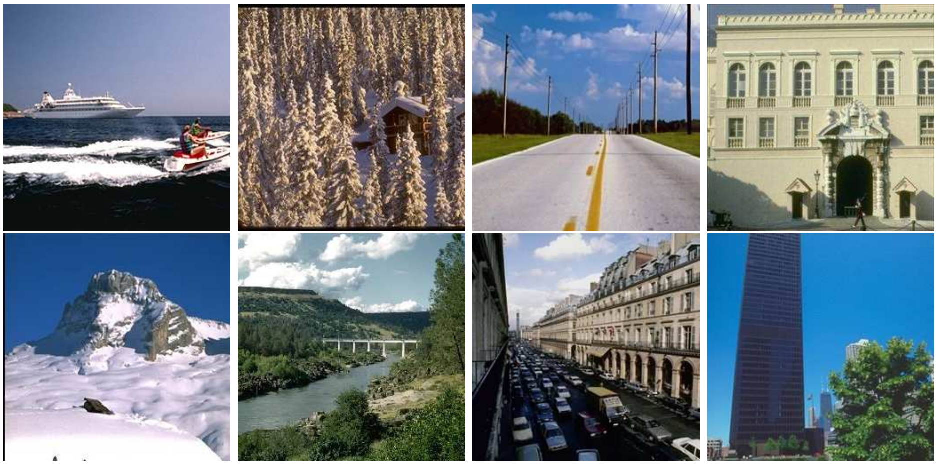


Figure 5.7: Images from the 8 OT scene categories: *coast, forest, highway, inside city, mountain, open country, street, tall building*.

showing that flexibility of the Rényi divergence can be rewarding.

5.7 Summary

In this work we proposed a novel discriminative method for set and distribution classification. We defined new kernels on sets of vectors and used consistent nonparametric divergence estimators for estimating the kernel values. Our goal was not to introduce new features; instead we were interested in improving the performance of bag of vectors image representations through better dissimilarity measures.

Parametric methods for divergence estimation are usually biased, since the true distributions may not belong to assumed parametric families. Our nonparametric divergence estimator, however, is asymptotically unbiased. It is also easy to compute, requiring only certain k-NN distances.

For bag-of-words methods, setting the appropriate codebook size is a difficult model selection problem. It is similarly unknown how to choose the bin sizes for histogram-based methods. Our algorithm has comparably fewer parameters to tune, and avoids the inherent approximations of histograms, quantization, and clustering, which can lead to loss of information and decreased performance.

In our experiments, we demonstrated that the proposed method can outperform its state-of-the-art competitors on several challenging datasets, both artificial and real.

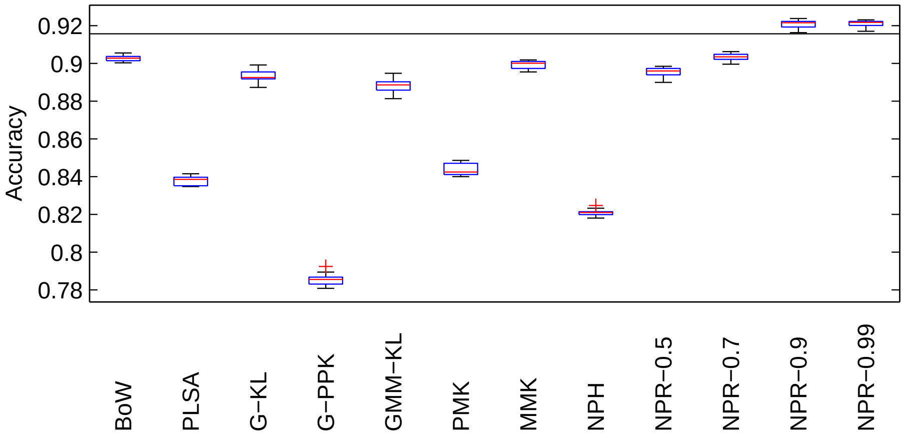


Figure 5.8: Accuracies on the OT data set. The horizontal line shows the best previously reported result.

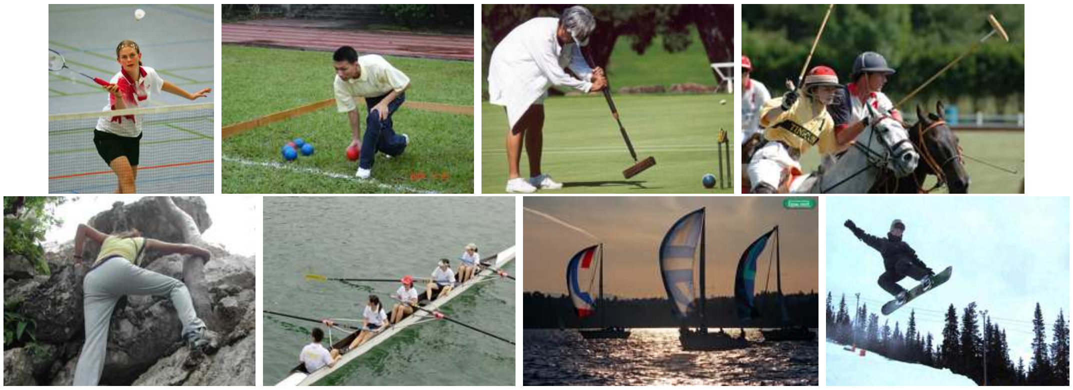


Figure 5.9: Images from the 8 sports.

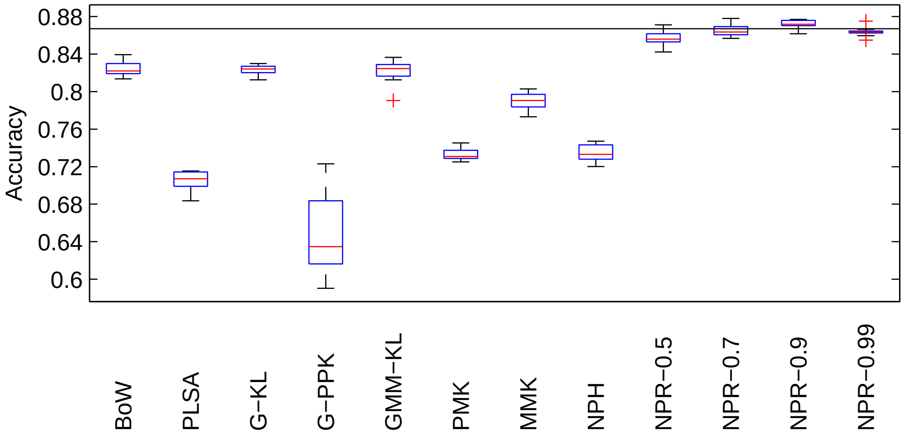


Figure 5.10: Accuracies on the Sport data set. The horizontal line shows the best previously reported result.

Chapter 6

Low-Rank Constructions of Mercer Kernels

In this chapter we describe more ways of constructing Mercer kernels based on estimated set kernels. Unlike the approach used in Chapter 5 which simply seeks best approximations, methods in this Chapter further exploit the low-rank structures that might exist in the divergence/kernel matrices. By using these structures, we are able to construct higher-quality kernel matrices, make the subsequent SVM training faster, cope with missing/unreliable kernel values, and make the computation faster.

6.1 Introduction

Many learning algorithms for collective data is based on pairwise similarities between groups. The advantage of this approach is evident: once we have the similarities, many excellent off-the-shelf algorithms such as the k-nearest-neighbors, SVM, spectral clustering, and so on can be used to accomplish various learning tasks. In this chapter, we focus on the kernel methods.

In Chapter 5, we described a large class of divergences that can be used to measure the dissimilarity between groups of points. Traditional set distances such as the *Hausdorff* distance can also be used for learning purposes. For a brief survey refer to Section 5.2. One shortcoming for many of these divergences is that they do not behave like Euclidean distances; they do not satisfy the triangle inequality, and they even do not provide symmetry. Moreover, these similarities might have been estimated as in Chapter 5, and thus are mixed with estimation errors. Consequently, the kernels (most commonly Gaussian kernels) based on these divergences are not valid Mercer kernels, and thus cannot be directly applied in many kernel methods such as the SVM.

To address this problem, we take the approach of replacing the raw kernel matrix constructed from the divergences by a refined kernel matrix that is *positive semi-definite* (PSD) and thus corresponding to a valid Mercer kernel. Chapter 5 did this by using the PSD projection method, in which the refined kernel matrix is the best L_2 approximation to the raw kernel matrix. That method is purely based on numerical approximation. In the following, we shall show that when structures exist in the kernel matrix, more effective methods can be used to construct the refined kernel matrix.

The structure we exploit in this Chapter is that the kernel matrices are often approximately low-rank. A low-rank matrix can be reconstructed by the linear combination of a few row/column basis vectors. Another useful interpretation is that the objects can be embedded as points in some low-dimensional Hilbert space, so that the inner-products between the points equals the kernels between the objects. In machine learning, the low-rank structure of kernel matrices has been used to accelerate the training of SVM, or enhance the kernels' discriminative power when combined with supervision. In our work, we use the low-rank technique to construct valid kernels from raw estimations. With a low-rank, the refined matrix can be more robust against errors in the raw matrix, and make the subsequent SVM training more efficient.

In addition to construct the refined kernel matrices directly, we can also construct refined divergence matrices that will lead to valid kernel matrices when converted to the Gaussian kernels. Again this can be achieved by embedding the objects into some low-dimensional Euclidean space, so that the distances between the points are close to the divergences between the objects.

The low-rank methods are also useful in dealing with missing data and accelerating the computations. In low-rank methods, the degrees-of-freedom in the matrices are limited, and thus the entries become redundant and the missing ones can be inferred based on the observed ones. This method can also be used to speed up the computation of kernel matrices for collective data; instead of computing the kernel between every pair of groups, we can skip some pairs and still get a high-quality kernel matrix.

In the experiments, we examined the performance of the refined kernels constructed from the raw kernel and the raw divergence matrices on image classification tasks. They may both provide superior results than the PSD projection method in Chapter 5 depending on the data. We also tested their effectiveness in the presence of missing entries in the kernel/divergence matrices, and find that the kernel matrix based method can achieve good classification accuracies using a small number of kernel evaluations.

The rest of this chapter is organized as follows. Related work on described in Section 6.2. In Section 6.3 we describe how to complete a kernel matrix, and Section 6.4 shows how to complete a divergence matrix. In Section 6.6 we examine the performances of different approaches, and finally we make our conclusion in Section 6.7.

6.2 Related Work

The low-rank structure of kernel matrices has been frequently assumed and used in machine learning. A number of papers have used low-rank approximations of the kernel matrix to accelerate the training of SVM. [51] described how to train SVMs efficient when given low-rank kernel matrices. [47] proposed and analyzed sampling based method that can construct low-rank approximations efficiently. [87] proposed an alternative way to optimize low-rank approximations efficiently. These methods operate on valid kernel matrices already and the goal was to accelerate the training of SVMs. On the other hand we are constructing kernels from noisy non-PSD matrices. But indeed common techniques can be used, and the low-rank matrices we construct can also be used to train SVMs efficiently.

Other works were also proposed to learn the low-rank kernel matrix using heuristics or supervised information enhance the kernels. [169] learns a PSD matrix under neighborhood constraints while maximizing the embedding's variance. By doing this, the data can be "spreaded out" to create clear visualizations. [7] incorporated class labels to increase the discriminative power of the kernels. These enhancements can also be used for our purposes of constructing valid kernels.

The classical technique of creating Euclidean embeddings based on given distances is the metric multidimensional scaling (MDS) [35]. MDS has been studied for decades and many variations has been created to deal with different problems. The method we use in this chapter is based on metric MDS, and specifically tuned for the divergences proposed in Chapter 5. The effectiveness of different settings are also empirically evaluated.

Several papers have been proposed to learn the kernel matrices given missing data. [62] used *semi-definite programming* (SDP) to learn the complete PSD matrix based on partially observed kernel matrix, and [4] used an alternative constrained optimization to obtain sparse PSD matrices. Both methods can not handle noisy and non-PSD raw matrices, which are what we are facing in this chapter. [1] described a way to ignore some entries in the kernel matrix, but their method need to know the full kernel matrix beforehand.

6.3 Constructing Low-Rank Kernels

First we define our problem. Suppose that we have observed a kernel matrix $\mathbf{K} \in \mathbb{R}^{M\times M}$, where each entry $k_{ij} = k(G_i, G_j)$ is the kernel value for the (i, j)th pair of groups. We mainly consider the kernels constructed from estimated divergences defined in Section 5.4, but the techniques below are applicable to general kernel matrices. This raw kernel matrix $\mathbf{K}$ might have been derived from non-metric divergences, and the kernel values might be inaccurate due to estimation errors. As a result, $\mathbf{K}$ is not PSD and hence not a valid Mercer kernel to be used in kernel machines. Our goal is to find a refined kernel matrix $\widetilde{\mathbf{K}} \in \mathbb{R}^{M\times M}$ so that $\widetilde{\mathbf{K}}$ is close to $\mathbf{K}$ and is PSD.

By definition, $\widetilde{\mathbf{K}}$ is PSD iff it can be decomposed as $\widetilde{\mathbf{K}} = \mathbf{U}\mathbf{U}^T, \mathbf{U} \in \mathbb{R}^{l\times M}$, where $\mathbf{U}$ is the factor matrix and l is the rank of $\widetilde{\mathbf{K}}$. A brief introduction of matrix factorizations can be find in Section 3.1.1. The columns of $\mathbf{U}$, which are denoted as $\{\mathbf{u}_i\}_{i=1,\dots,M}$, can be interpreted as points in a l-dimensional space, where the points are the embeddings of the groups. The inner-product between two points equals their corresponding groups' kernel value as $\mathbf{u}_i^T\mathbf{u}_j = k(G_i, G_j)$.

A good $\widetilde{\mathbf{K}}$ should be close to $\mathbf{K}$, so we want to minimize the element-wise difference between $\widetilde{\mathbf{K}}$ and $\mathbf{K}$. To cope with missing or unreliable entries, we further weigh the errors on different entries differently by a weight matrix $\mathbf{W} = \{w_{ij}\}_{i,j=1,\dots,M} \in \mathbb{R}^{M\times M}$. To reduce the degrees-of-freedom, we constrain the rank l of $\widetilde{\mathbf{K}}$, as well as minimize the norm of $\mathbf{U}$. All these terms can be summarized by the following optimization problem which we call the *low-rank kernel construction* (LRKC) problem:

$$\mathbf{U} = \underset{\mathbf{U}\in\mathbb{R}^{l\times M}}{\arg\max} \sum_{i,j} w_{ij}\left(\mathbf{u}_i^T\mathbf{u}_j - K_{ij}\right)^2 + \lambda\|\mathbf{U}\|_F^2 \tag{6.1}$$

where $\|\cdot\|_F$ is the Frobenius norm, λ is the penalty on the norm of $\mathbf{U}$, and $\mathbf{U}$ is the factor matrix for $\widetilde{\mathbf{K}}$. The weight matrix $\mathbf{W}$ controls the importance of the entries in $\mathbf{K}$. Entries with zero weight are ignored in the optimization and are thus regarded as missing. This problem can easily be solved by local descend algorithms such as gradient descend and L-BFGS. Upon convergence, the refined kernel matrix can be obtained by $\widetilde{\mathbf{K}} = \mathbf{U}^T\mathbf{U}$.

We choose to minimize $\|\mathbf{U}\|_F^2$ because it equals the nuclear norm of $\widetilde{\mathbf{X}}$ (the sum of singular values of $\widetilde{\mathbf{X}}$) , which is a good surrogate of the its rank [138]. Therefore, we can gain more control over $\widetilde{\mathbf{K}}$'s complexity by penalizing $\|\mathbf{U}\|_F^2$. Usually, we let the rank l be a relatively large value and control the complexity of $\widetilde{\mathbf{K}}$ by varying λ.

6.4 Constructing Low-Rank Divergences

The refined kernel matrices are based on the raw kernel matrices, which are derived from divergences. Therefore, once the kernel parameters (*e.g.* the width of the Gaussian kernel) change, we have to refine the kernel matrix again. This problem can be solved by constructing low-rank distance matrices instead. In addition, refined distances behave different from refined kernels, and may lead to better learning performances.

Formally, given a raw divergence matrix $\mathbf{D} \in \mathbb{R}^{M\times M}$ for M groups, we want to find a refined distance matrix $\widetilde{\mathbf{D}} \in \mathbb{R}^{M\times M}$, so that $\widetilde{\mathbf{D}}$ can lead to a PSD Gaussian kernel matrix. The following Lemma 1 reveals a way of satisfying this requirement. That is, any distance function that leads to PSD Gaussian kernels can be realized in some real Hilbert space. Another way of arriving at this is that, since the a distance d is *conditional negative definite* (CND) iff $e^{-\lambda d}, \lambda > 0$ is PSD [146], and any non-negative symmetric CND matrix with a zero diagonal must be a squared Euclidean distance matrix [145], therefore $e^{-\lambda d}, \lambda > 0$ is PSD iff d is a Euclidean distance. In short, the refined distance matrix $\widetilde{\mathbf{D}}$ should be a Euclidean distance matrix.

Lemma 1 (Embeddability and PSD Functions [145])**.** *A necessary and sufficient condition that a separable space with a distance function that is non-negative, symmetric, and discernible, be isometrically embeddable in the real Hilbert space, is that the family of functions* $e^{-\lambda t^2}, \lambda > 0$ *be positive definite, where* t *corresponds to the distances.*

Meanwhile, the rank of the Euclidean distance matrix is constrained by Lemma 2. It shows that to find a low-rank Euclidean distance matrix $\widetilde{\mathbf{D}}$ we only need to find a low-dimensional embedding of the groups.

Lemma 2 (Rank of the Euclidean Distance Matrix [46])**.** *Let* $\mathbf{D}$ *be a squared Euclidean distance matrix for* l*-dimensional points. Then the rank of* $\mathbf{D}$ *is at most* $l + 2$*.*

The above reasonings lead us to a formulation of finding low-rank distance matrices that is similar to finding low-rank kernel matrices in the previous section. Concretely, we seek for $\mathbf{U} \in \mathbb{R}^{l\times M}$ the low-dimensional embedding of the groups via the following *low-rank distance construction* (LRDC) problem

$$\mathbf{U} = \underset{\mathbf{U}\in\mathbb{R}^{l\times M}}{\arg\max} \sum_{ij} w_{ij} \left(D_{ij}^r - \|\mathbf{u}_i - \mathbf{u}_j\|_2^r \right)^2 . \tag{6.2}$$

Similar to finding low-rank kernel matrices, the weight matrix $\mathbf{W} = \{w_{ij}\}_{i,j=1,\dots,M}$ gives the importance/presence of the entries, and $\mathbf{u}_i$ the ith column of $\mathbf{U}$ is the embedding of the ith group. Again, this problem can be solved by gradient descend algorithms. Once having the embedding $\mathbf{U}$, we can calculate the refined distance matrix $\widetilde{\mathbf{D}}$ that gives a PSD kernel matrix.

Note the presence of the power parameter r, which transforms the elements of the matrices before computing their differences. This is important because Euclidean distances and the divergence estimates may behave very differently (*e.g.* the KL-divergence can go to infinity easily; in practice we found that the estimators in Chapter 5 tend to underestimate large divergences), and it may be difficult to match them directly even when using high-dimensional embeddings. Intuitively, a small $r < 1$ emphasizes the approximation for small divergences, while a large $r > 1$ emphasizes large divergences.

Problem (6.2) essentially solves the metric multidimensional scaling (MDS) [35] problem. Indeed, if $r = 1$ then (6.2) is MDS with the *stress* objective, and if $r = 2$ it is MDS with the *s-stress* objective. It is interesting to note that we arrive at MDS from the initial motivation of finding a low-rank divergence matrix that leads to PSD Gaussian kernels. Compared to traditional MDS, our formulation uses the parameter r to accommodate different divergences.

6.5 Discussion

Section 6.3 and 6.4 described ways of finding low-rank kernel and distance matrices as the refinement of the raw kernel and divergence matrices. These refined matrices will given us PSD kernel matrices that can be used in various kernel machines.

Both problems (6.1) and (6.2) are non-convex and local minima might exist. Therefore, finding a good starting point is important. For the kernel matrix construction problem (6.1), note that when $\mathbf{W} = \mathbf{1}, \lambda = 0$ the global optimum can be obtained by either SVD or eigen-decompositions. In this work we shall use this particular solution to initialize the optimization. For the distance matrix construction problem (6.2), we can use the solutions of the classical MDS, which also admits global optima, as the initializations.

Missing entries can easily be handled by setting the corresponding entries in the weight matrix $\mathbf{W}$ to zero. By doing this, the missing entries will be effectively ignored in the objective function and the embedding $\mathbf{U}$ is derived only based on the observed entries. When initializing the optimization using eigen-decomposition or classical MDS, we set the missing entries to the average kernel/divergence value.

The ability to cope with missing entries provides us with a way of speed up the construction of the kernel matrix. Even though the divergence computation only needs KNN statistics, comparing two groups is still slow relative to comparing two vectors. Instead of computing every entry in the divergence matrix, we can intentionally skip some of them and let the low-rank construction infer them. This is possible because the low-rank-ness greatly reduces the degrees-of-freedom in the divergence/kernel matrices, hence the information carried by the entries become redundant and we can impute the missing entries. The nature of this imputation is the same as that of Chapter 2.

We can use different ways to determine the suitable rank l in order to balance the approximation accuracy and the ability to filter out the noisy and infer the missing entries. For the kernel matrix, we can directly use the number of dominant eigenvalues to guess the rank as in PCA, while using λ to further control the model complexity. As for the divergence matrix, according to Lemma 2, we can look at the singular values of $\mathbf{D}^2$ to determine a sensible rank. The above indications can be used as guidelines to guess the rank of the refined matrices, but in general cross-validation should be used to determine the best choice of parameters.

6.6 Experiments

In this section, we shall evaluate the empirical performances of the refined kernel matrices constructed by the low-rank methods. First we use synthetic data sets to demonstrate them, and then test their effectiveness in image classification tasks.

6.6.1 Synthetic Data

We synthesize a toy data set using the following steps. We randomly choose $M = 100$ points in the 2D space half from the Gaussian distribution $\mathcal{N}(-3, 1)$ and half from $\mathcal{N}(3, 1)$, and use them calculate the groundtruth distance matrix $\mathbf{D}^*$ and kernel matrix $\mathbf{K}^* = \exp\left(-\frac{\mathbf{D}^{*2}}{2\sigma^2}\right)$ where σ is the kernel width. Then we impose independent Gaussian noise on the entries of $\mathbf{D}^*$ to get the noisy raw distance matrix $\mathbf{D}$ as $D_{ij} = D^*_{ij} + \beta\mathcal{N}(0, D^{*2}_{ij})$ where β controls the noise level. We let the noise level increase as the distance become larger in order to emulate the behavior of the divergence estimators. The raw kernel matrix hence is $\mathbf{K} = \exp\left(-\frac{\mathbf{D}^2}{2\sigma^2}\right)$. $\beta = 0.3, \sigma = 2$ is used. Note that the noise level is very high here, and the noise can make $\mathbf{D}$ asymmetric.

Our goal is to obtain refined kernel matrix $\widetilde{\mathbf{K}}$ from the noisy observation $\mathbf{D}$ and $\mathbf{K}$, so that $\widetilde{\mathbf{K}}$ is close to the groundtruth $\mathbf{K}^*$. We do this in two ways: 1) get refined distance matrix $\widetilde{\mathbf{D}}$ from $\mathbf{D}$ using LRDC (6.1), and then derive $\widetilde{\mathbf{K}}$ from $\widetilde{\mathbf{D}}$; 2) directly get refined $\widetilde{\mathbf{K}}$ from $\mathbf{K}$ using LRKC (6.2). For LRDC, we set the rank $l = 2$. For LRKC, we use the rank that preserves 95% data variance using SVD (usually $l = 9$), and set the penalty $\lambda = 0$. All the weights are set to one. Different settings of the divergence transformation r are tested. To measure the quality of recovery, we compute the element-wise correlation between $\widetilde{\mathbf{K}}$ and $\mathbf{K}*$ since scaling the kernel matrix will not affect learning.

The result of 20 random runs are shown in Figure 6.1. We can see that the quality of the recoveries are very high, with correlations above 0.99. Examples of the matrices are shown in Figure 6.2. For LRDC, we include the results of different r values to show the importance of transforming the divergences. Indeed, r being to small or too large will produce bad results. The optimal range for r seems to be within $[0.2, 1]$, with a weak peak at $r = 0.7$. We can also see that the results LRDC can outperform the results of LRKC given proper r values.

Next, we test how the methods perform given missing entries. We randomly pick a portion p of entries (diagonal entries are always picked to help bound the LRKC problem) in the weight matrix and set the rest to zero. Then we do the same test as above. Figure 6.3a and 6.3b show the results of picking $p = 50\%$ and $p = 20\%$ entries. We can see that both methods still produce satisfactory results even if a majority of the data are missing. We observe that in this test the LRDC results are significantly better than the LRKC results. The reason might be that the Gaussian kernel matrix are inherently full rank, and with the presence missing entries it becomes more difficult to guess a good rank for LRKC. On the other hand, we know that the optimal rank for the distance matrix is 2. It should also be noted that better result might be available if we tune the λ parameter for LRKC.

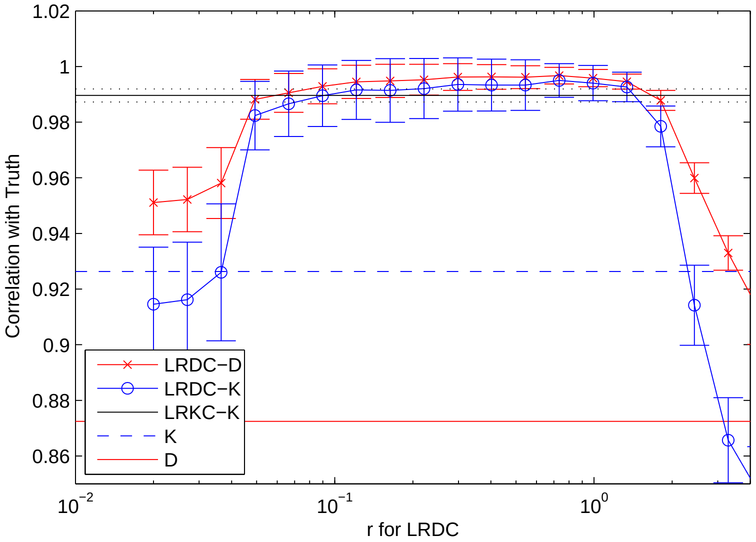


Figure 6.1: Recovery performances on the toy data set. LRDC-D and LRDC-K are for the distance matrix and the kernel matrix obtained by LRDC respectively. LRKC-K is for the kernel matrix obtained by LRKC. The X axis shows different values of the parameter r in LRDC. The correlations of the raw distance matrix $\mathbf{D}$ and kernel matrix $\mathbf{K}$ are also shown.

The optimal r values for these two cases are 0.5 and 0.4 respectively, showing a very consistent behavior.

6.6.2 Image Classification

In this section, we test the refined kernels' performances in real-world image classification tasks. The data and the setup of experiments here are the same as in Section 5.6, therefore we omit the details here. Instead of using the PSD projection method to get the valid PSD kernels as in Section 5.5, we use LRDC and LRKC to accomplish the same task.

Since the number of possible settings is too large (*e.g.* divergence types, divergence estimators, kernel width, rank of the refined matrices, and so on), we shall use heuristics and limit our attention to the most interesting ones. We only use the raw estimated KL-divergences and the resulting Gaussian kernels given in Chapter 5 since it is the most common choice and often leads to near-optimal results. For LRKC, we avoid using cross-validation to select the kernel width, which is computationally demanding, by heuristically setting the kernel width to $2\sigma^0$, where σ^0 is the average divergence from a group to its 3rd nearest neighbor group. In these data sets, finding a good guess of the rank l becomes increasingly difficult because the spectra of the matrices become highly concentrated. Instead, we found that $l = 100$ works well for smaller data sets with less than $M = 1,000$ groups, and the $l = 150$ works well for larger data sets. To generate incomplete divergence/kernel matrices, we randomly mark 50% of their entries as missing and set the corresponding weights to zero.

In each run, we use half of the groups for training and the other half for testing. SVM parame-

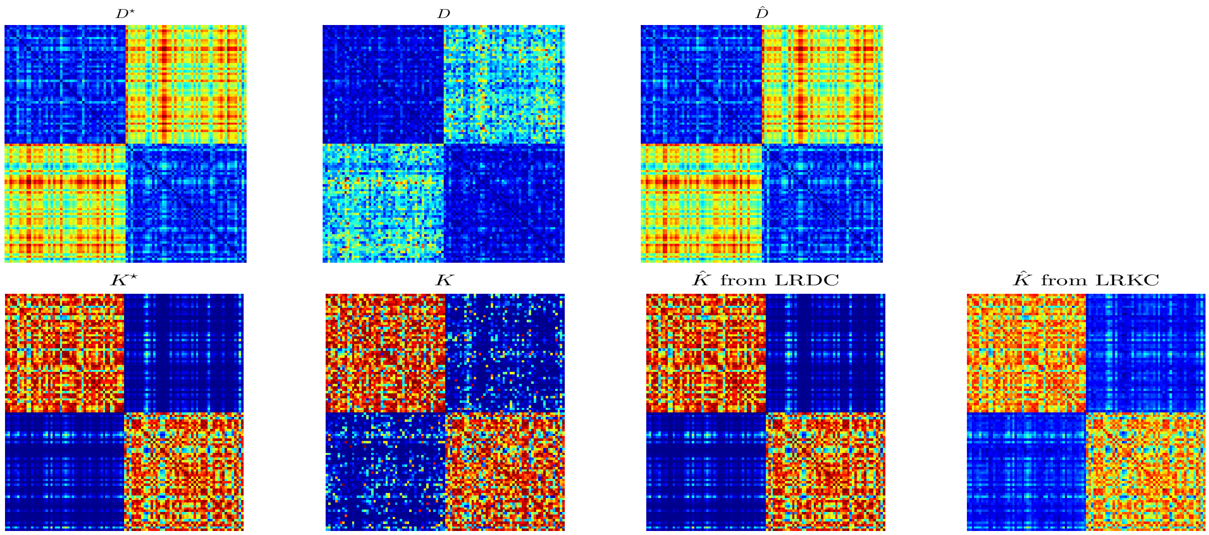
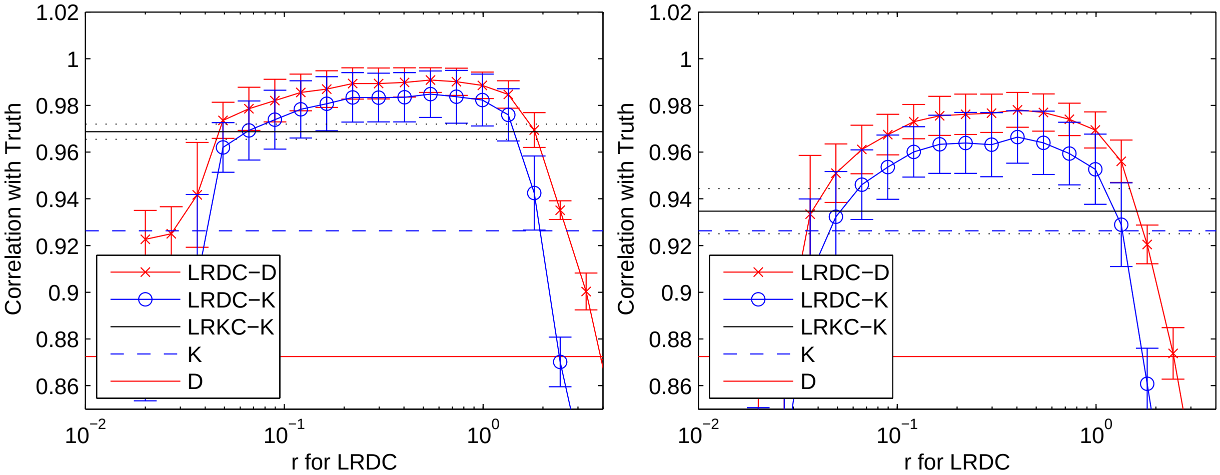


Figure 6.2: Example results from LRKC and LRDC. $\mathbf{K}^\star$ and $\mathbf{D}^\star$ are the groundtruths. $\mathbf{K}$ and $\mathbf{D}$ are the noisy observations. $\widehat{\mathbf{K}}$ and $\widehat{\mathbf{D}}$ are the low-rank results produced by LRKC and LRDC.

ters (the slack penalty C and the kernel width σ when the input is a distance matrix) are tuned by 3-fold cross-validation on the training set. We report the results of LRKC with different λ's and LRDC with different r's. Accuracies from 10 random runs are reported. In the figures, the method label "D" denotes the results from the PSD projection method which is used as our baseline, and method labels with a "-I" postfix means incomplete data.

ETH-80 First, we report the results on the ETH-80 [95] object recognition data set. The results with both full and incomplete divergences/kernels are shown in Figure 6.4. We can see that LRKC causes a slight degradation to the accuracies, while LRDC is able to slightly outperform the baseline using much less degrees-of-freedom. On the other hand, LRKC is more robust against missing entries, showing a decrease of only about 2% to the accuracy. The sensitivities to the parameters are generally small. LRKC prefers a smaller λ while the performance of LRDC slightly peaks around $r = 0.8$.

Sports Results on the Sports scene [100] data set are shown in Figure 6.5. On this more complex data set, the low-rank methods LRKC and LRDC can both outperform the baseline. The optimal performance of LRDC is achieved with around $r = 0.5$ or $r = 2$. Notably, the incomplete LRKC method achieved the same performance as the baseline method using only half of the entries, and is again only 2% worse than itself with full data. The incomplete LRDC, however, failed to maintain its effectiveness facing missing entries. Observe that on the data set the performance of the incomplete LRKC is sensitive to the parameter λ, meaning that its ability to cope with missing entries depends on a proper model complexity.



(a) Observe 50% entries.

(b) Observe 20% entries.

Figure 6.3: Recovery performances on the toy data set with missing data. LRDC-D and LRDC-K are for the distance matrix and the kernel matrix obtained by LRDC respectively. LRKC-K is for the kernel matrix obtained by LRKC. The X axis shows different values of the parameter r in LRDC. The correlations of the raw distance matrix $\mathbf{D}$ and kernel matrix $\mathbf{K}$ are also shown.

OT Finally, results on the OT [126] data set, which contains natural scene images, are shown in Figure 6.6. Again, we see that LRDC is able to significantly outperform the baseline, and incomplete LRKC achieved high accuracies that are less than 1% worse than the baseline. The optimal performance of LRDC is achieved with around $r = 0.5$ and $r = 2$.

Based on the above results, we conclude that LRDC can achieve better performance than L-RKC and the PSD projection method on the full matrices given suitable parameters. On the other hand, the incomplete LRKC is robust against missing values, usually being able to save half of the computation while only losing 2% accuracy. LRKC usually performs similarly as the PSD projection on full matrices, and LRDC can have difficulties handling missing data.

6.7 Summary

In this chapter we investigated more ways of constructing Mercer kernels based on the kernels estimators in Chapter 5. Exploiting the low-rank structures of the raw kernel and divergence matrices, we are able to derive better kernels for kernel machines.

Both kernel-based and divergence-based approaches are proposed based on the low-dimensional embedding of the groups. The performances vary depending on the specific data set, but in general both can produce better results than the PSD projection method in Chapter 5, thanks to their robustness against errors and the discrepancy between the PSD kernels and the divergences.

We also tested the performance of using the methods to speed up the learning by skipping divergence computations. We found that the kernel-based approach is very robust against missing

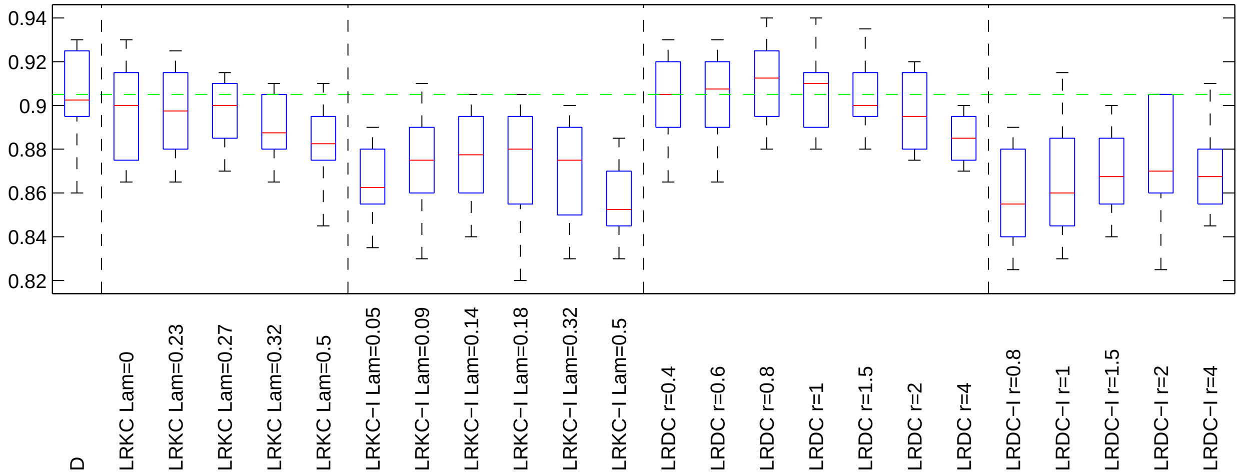


Figure 6.4: Classification accuracy on the ETH-80 data set. D is the baseline (green dashed line) using PSD projection. The “-I” postfix means incomplete data. LRKC with different λ’s and LRDC with different r’s are shown.

kernel values, so that we can skip a large portion of the kernel computations while preserving the learning performances. Meanwhile, the divergence based approach is less effective in this case. Another interesting direction to explore is that we can borrow the ideas of active learning and purposefully choose which kernel values to observe, so that we can recover a good low-rank kernel matrix using as less observations as possible.

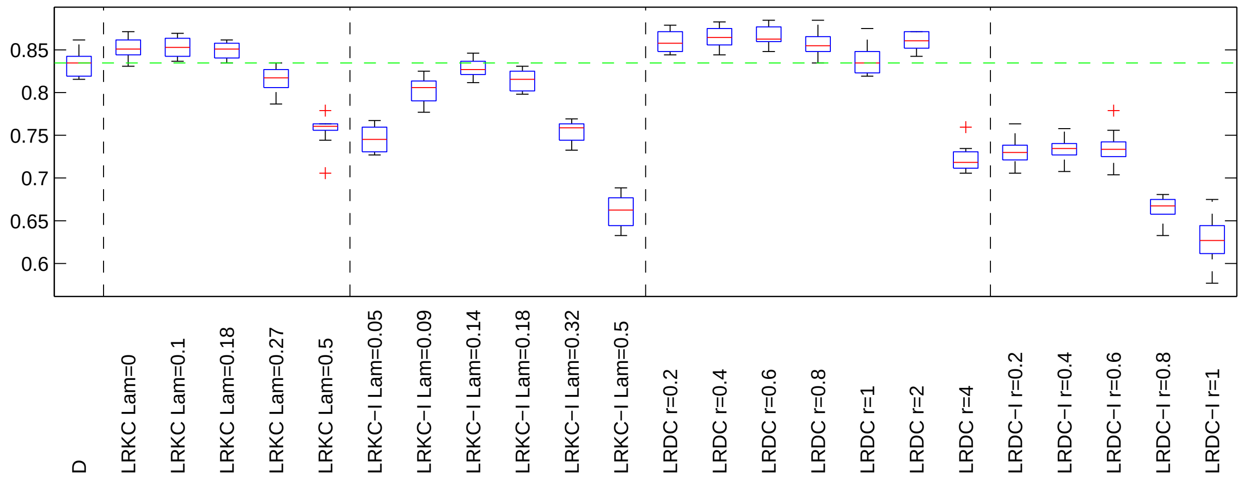


Figure 6.5: Classification accuracy on the Sports data set.

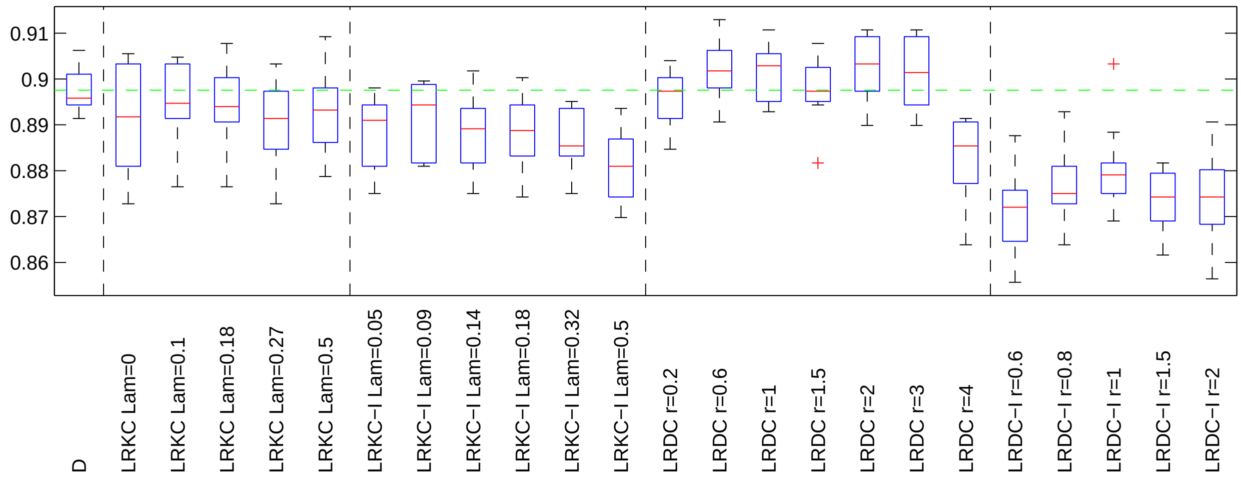


Figure 6.6: Classification accuracy on the OT data set. D is the baseline (green dashed line) using PSD projection. The “-I” postfix means incomplete data. LRKC with different λ's and LRDC with different r's are shown.

Chapter 7

Accelerated Learning by Condensing

In addition to methods proposed in this thesis, recently several other algorithms have been proposed to learn from data that are represented as sets/groups of vectorial points. Such algorithms usually suffer from the high demand of computational resources, making them impractical on large-scale problems. We propose to solve this problem by *condensing i.e.* reducing the sizes of the sets while maintaining the learning performance. Three methods are examined and evaluated with a wide spectrum of set learning algorithms on several large-scale image data sets. We discover that k-Means can successfully achieve the goal of condensing. In many cases, k-Means condensing can improve the algorithms' speed, space requirements, and surprisingly, learning performances simultaneously.

7.1 Introduction

In many problems the object of interest can be represented by a set of multidimensional vectors. For instance, in computer vision an image is often treated as a set of patches [50]. In text processing and retrieval, we can also think of a document as a set of sections/paragraphs to cope with its structure. A convenient and indeed frequently used way to deal with point sets is to discretize the points and construct a feature vector for each set. However, the conversion process is often problem-specific, sometimes complicated, and involves much human effort. More importantly, this set-to-vector reduction can cause loss of information.

On the other hand, the development of algorithms that handle sets directly has been largely left behind. One major disadvantage of such algorithms is their high computational cost compared to those that operate on vectors. Despite the difficulties, recently several methods have been proposed to deal with point sets directly. They learn from the sets without the set-to-vector reduction, so that the researchers do not have to design the feature vector for a set, and the loss of information caused by the reduction can be avoided. For example, our Chapter 5 design a novel kernel between point sets based on consistent estimators of divergences between distributions, and achieved the state-of-the-art classification performance on a couple of datasets. [17] proposed an extremely simple classifier for point sets based on group-to-class matching, and showed that it could compete with classifiers based on very sophisticated set features on images. These successes demonstrate the advantage of learning directly from point sets over the reduction approach.

Early set learning algorithms (more specifically set similarities), such as the *Hausdorff* distance and the *mean map kernels* by [65], rely on the similarities between every pair of points and are thus computationally expensive. Recent improvements such as Chapter 5 and [17, 112, 131] gained efficiency by designing algorithms based on information from the points' local neighborhoods, which can be obtained via efficient search algorithms. [17, 112] proposed a new classification paradigm by comparing images to classes and significantly accelerated the prediction. Details of these methods are described in Section 7.2. Nevertheless, they still demand much time and storage space, making them not suitable for large-scale problems and less likely to be adopted by practitioners.

In this work, we aim to further improve the computational efficiency of set learning algorithms. In most set learning algorithms, the cost to train, store, and apply the model is determined by the sizes/cardinalities of the sets. Therefore, our approach is to directly attack the crux of the problem by reducing the size of sets while maintaining the learning performance in an unsupervised way. We call such an operation *condensing*.

To achieve this goal, we analyze and evaluate three possible ways of decrease the size of a set: random sampling, uniform covering, and distribution approximation using k-Means. These three methods are chosen because they are easy to implement and efficient to run in large data scenarios. Our discovery is that distribution approximation via k-Means is the only method that can successfully achieve the goal of condensing.

In our experiments, we apply the k-Means condensing as a pre-processing step to various point-set learning methods on several image classification tasks, and find that the performance is surprisingly good and consistent. In most problems, we do not have to make a speed-accuracy tradeoff; condensing can actually *improve both speed and accuracy simultaneously*. In addition, this condensing step can be easily implemented and parallelized for large-scale problems. We believe this discovery is useful to practitioners that have large-scale point-set data.

The rest of this chapter is organized as follows. In Section 7.2 we introduce the notation and common learning algorithms on point sets to provide a context to this study. Section 7.3 briefly reviews related work. Section 7.4 describes in detail the condensing methods we are examining. In Section 7.5, we thoroughly evaluate the performance of different methods on different data sets and discuss our findings. Finally, we discuss and conclude this chapter in Section 7.6 and Section 7.7.

7.2 Background

Most learning tasks can easily be accomplished if we know the similarities between the point sets. Many set learning algorithms assume that a set has an unknown underlying distribution, and the points in the set are *i.i.d.* samples from that distribution. Then the similarity measures between point sets can be designed based on the divergences between their underlying distributions. For example, [65, 155] proposed the *mean map* set kernel to test if two point sets have the same underlying distribution. The same technique has been used by *multiple-instance learning* [55] and computer vision [109]. [17] uses a simplified *kernel density estimator* to estimate the divergence between a set and the classes, and assign the set to the class with the most similar distribution. [130, 131] and Chapter 5 use a consistent nonparametric estimator to get the divergences between the point sets and use these dissimilarities to construct Gaussian kernels so that SVM can be used for classification.

From the computational perspective, many of the set similarity measures can be considered as aggregations of the similarities between the individual points of the sets. We will discuss in more details how these similarities are measured and aggregated in Section 7.2.1. The key point is that these point-level pairwise comparisons make the speed of the algorithms crucially depend on the sizes of the sets. This is why reducing the sets' sizes by condensing would greatly improve their computational efficiency. Generative methods such as [173, 174] and Chapter 4 have also been developed to model point sets. Condensing the sets will also benefit these methods. In this Chapter we will focus on the similarity based approaches and their applications in set classification problems.

We again consider a data set with M point sets $\{G_m\}_{m=1,\dots,M}, G_m = \{x_{mn}\}_{n=1,\dots,N_m}, x_{mn} \in \mathbb{R}^D$. We also assume that each G_m has an unknown underlying distribution f_m, and the points $\{x_{mn}\}$ are *i.i.d.* samples from f_m. For instance, in the context of image classification, each G_m is an image, and vector x_{mn} is the feature of the nth patch in this image.

Nearest neighbors (NN) are frequently used in set learning algorithms. We use $\text{NN}_G(x)$ to denote the NN of x in point set G. If x is in G then it excludes itself during the search. Ties, if any, are broken arbitrarily.

7.2.1 Set Similarities

Set Kernels [65, 155] proposed the following kernel (similarity) for two sets of points G_1 and G_2:

$$K(G_1, G_2) = \frac{1}{N_1 N_2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} k(x_{1i}, x_{2j}) \tag{7.1}$$

where $k(x, y)$ is a kernel between points x and y. One particularly popular example is the Gaussian kernel $k(x, y) = \exp(-\|x - y\|_2^2/\sigma^2)$. The underlying principle for this kernel is that if the point-level kernel $k(x, \cdot)$ induces a feature map $\phi(x)$ for x in a *reproducing kernel Hilbert space* (RKHS), then this corresponding set-level kernel $K(G, \cdot)$ will induce the feature map $\Phi(G) = \frac{1}{N}\sum_{n=1}^{N} \phi(x_n)$ for G in the same RKHS. Since $\Phi(G)$ is the empirical mean of the mapped features of the points, K is called the *mean map kernel* (MMK).

We can see that MMK essentially is a way of aggregating the point-level similarities between two sets. It possesses many nice theoretical properties such as positive definiteness [65, 120]. The same idea has also been used in computer vision [109] and *multiple-instance learning* [55]. Yet since it averages the similarities between every pair of points, MMK will slow down quadratically *w.r.t.* the sets' sizes, and become infeasible for even moderately sized problems.

[167] used a variation of MMK that only considers the most similar pairs of points:

$$K(G_1, G_2) = \frac{1}{N_1} \sum_{i=1}^{N_1} k(x_{1i}, \text{NN}_{G_2}(x_{1i})) + \frac{1}{N_2} \sum_{j=1}^{N_2} k(x_{2j}, \text{NN}_{G_1}(x_{2j})). \tag{7.2}$$

Computationally it improves MMK by only using the points' NNs instead of dealing with all pairs. Unfortunately, this kernel is no longer a proper *Mercer kernel*, but it still serves well as a similarity measure. Other related methods include [63], which uses multi-resolution histograms to approximates MMK, and [16], which constructs explicit approximate feature maps for the MMK so that linear classifiers can used to achieve faster computation.

Set Divergences Another class of dissimilarity measures is defined based on the statistical divergences between two distributions. In [130, 131, 168], the authors provided consistent NN based estimators for various divergences including the *Kullback-Leibler* (KL), Rényi, and the L_2 divergences. They have been successfully applied to image classification problems when used in SVMs; See Chapter 5.

For example, the KL divergence between two point sets can be estimated by [168]

$$\hat{\text{KL}}(G_1||G_2) = \frac{D}{N_1}\sum_{i=1}^{N_1} \ln \frac{\|x_{1i} - \text{NN}_{G_2}(x_{1i})\|_2}{\|x_{1i} - \text{NN}_{G_1}(x_{1i})\|_2} + \ln \frac{N_2}{N_1 - 1} \tag{7.3}$$

where D is the dimensionality of x_{1i}. It was proved that these estimators are consistent, *i.e.* under regularity conditions, (7.3) converges to KL$(f_1||f_2)$ as the sample sizes N_1 and N_2 approach infinity.

[17] proposed an alternative estimate of the KL divergences. Consider *kernel density estimation* at point x_{1i} given the points in G_2 with a Gaussian kernel of width σ:

$$\hat{f}(x_{1i}; G_2) \propto \frac{1}{\sigma N_2}\sum_{j=1}^{N_2} \exp\left(-\|\frac{x_{1i} - x_{2j}}{\sigma}\|_2^2\right). \tag{7.4}$$

This estimator is computationally demanding since we have to consider every pair of points. In [17], the authors let the width σ be small enough, so that the summation will be dominated by its largest term *i.e.* the nearest neighbor, and the estimator becomes

$$\ln \hat{f}(x_{1i}; G_2) \approx -\|x_{1i} - \text{NN}_{G_2}(x_{1i})\|_2^2/\sigma^2 - \ln N_2\sigma + \text{const}, \tag{7.5}$$

which again is based on NNs. The corresponding estimated KL divergence is

$$\hat{\text{KL}}(G_1||G_2) \propto \sum_{i=1}^{N_1} \|x_{1i} - \text{NN}_{G_2}(x_{1i})\|_2^2 - \sum_{i=1}^{N_1} \|x_{1i} - \text{NN}_{G_1}(x_{1i})\|_2^2 \tag{7.6}$$

up to a constant. Note the resemblance between (7.6) and (7.3). Unlike (7.3), this estimator is not consistent even with infinite points, but in practice it can still produce good results.

These set similarities follow a similar pattern. They generally use nearest neighbor statistics and can be efficient in low dimensions where various search trees work well. However, in high dimensions (as few as 10) the computational speed of the estimators will deteriorate rapidly.

7.2.2 Set Classification Schemes

Having the similarities between the sets, we can easily apply SVM, KNN, or other techniques to accomplish tasks like classification, ranking, and clustering as in *e.g.* [120, 130, 131]. For example, the KL divergences estimated using (7.3) can be used construct Gaussian kernels *e.g.*

$$K_{\text{KL}}(G_1||G_2) = \exp\left(-\hat{\text{KL}}(G_1||G_2)/\sigma^2\right), \tag{7.7}$$

where σ is the width of the kernel. Due to the properties of the KL divergence, this kernel is neither symmetric nor positive definite. Therefore in Chapter 5 proposed to approximate this "pseudo" kernel matrix by the closest positive definite matrix, and then use this approximation as the input to SVMs.

The drawback of this set-vs-set approach is that the training cost grows quadratically and the prediction cost grows linearly with the number of sets for training. Considering that comparing a pair of sets already requires significant work, this scheme quickly becomes infeasible in larger problems. To solve this problem, [17] proposed a set-vs-class paradigm for set classification. Assume that there are C classes indexed by c, and class c is represented by the merged set

$$H_c = \bigcup_{G_m \in c} G_m,$$

which contains all the points in the sets that belong to class c. The classification rule is to assign a test set G to the class with the smallest divergence

$$c(G) = \arg\min_c \hat{\text{KL}}(G||H_c), \tag{7.8}$$

where the KL-divergence is estimated by (7.6). The assumption under this scheme is that all the sets in the same class c have the same distribution f_c, from which H_c are the *i.i.d.* samples. Though it is debatable if this assumption is valid in real-world problems, the resulting algorithm called the *Naive Bayes nearest neighbor* (NBNN) [17] is extremely simple and performs well empirically.

More importantly, NBNN discards the training phase and makes the prediction cost of (7.8) only proportional to the number of classes as $O(NC\zeta_{H_c})$, where N is the size of the test set and ζ_{H_c} denotes the complexity of one NN search in H_c. *Local NBNN* (LNBNN) [112] further improves NBNN by merging all classes into one large set $H = \bigcup_c H_c$ and decreases the complexity to $O(N\zeta_H)$. As a result, LNBNN can classify many classes very efficiently. Interested readers are encouraged to see [112] for more details. Yet in large, high-dimensional problems, $\{H_c\}$ and H can easily become huge, making ζ_{H_c} and ζ_H unaffordable. Another problem is that H might become so large that it cannot be held in memory, making even the approximate NN search methods infeasible.

7.2.3 Computational Issues

Looking at the above algorithms, we realize that one would face severe challenges due to both computational time and space demand if one were to apply these algorithms to large-scale problems. These computational requirements are determined by the sizes of the sets. If we could reduce them by half, the space complexity would drop by half, and ideally the running time would drop to only a quarter. Therefore, our approach to make the "learning on sets" problem more efficient is to directly reduce the size of the sets, *condensing* the information into a much smaller amount of data while preserving the learning performance.

Even though NBNN and LNBNN have successfully made the complexity linear *w.r.t.* the number of sets, they create huge point sets $\{H_c\}$ for each class that are difficult to store and use. To put it in context, suppose we have $1,000$ images for training. Typically each image is characterized by around $2,000$ densely sampled 128-dimensional single precision SIFT vectors. Using Local NBNN, this relatively small training set would result in a model consisting of 2×10^6 points and 1GB of data. Additionally, in high dimensions searching for nearest neighbors in such a large set can be very slow.

In practice one could also consider using the approximate NN search algorithms. One popular method, for example, is the randomized KDTree [154] algorithm implemented in the *FLANN*[1] [121] package. It checks in multiple KDTrees for a fixed number of leaf nodes, which is the budget set by the user, and then returns the best results it can find. Its exact approximation accuracy and time complexity is unclear and dependent on the data. When using such approximate methods, it is rare to achieve quadratic improvement of speed by reducing the size, yet condensing can still greatly help the construction, use, and storage of models. When the data is too large to fit into memory, then even approximate search is infeasible, but condensing can make it possible.

7.3 Related Work

As far as we know, there is little previous work that thoroughly studies how to reduce the sets' sizes in set-based learning. One reason might be that set-based learning itself is rather new. Random sampling is a common practice. In [162], the authors used an *asymmetric approach* for image classification. The reasoning is that we can find good matching patches between two similar images as long as one of them is densely sampled. This approach is actually subsampling the set on one side of the similarity/divergence computation. It can speed up the computation but will also deteriorate the classification performance. In the following sections we show that by using more carefully chosen condensing methods, we can improve both the speed and the accuracy of their algorithm. Recently, the *kernel herding* [29] algorithm was proposed to accelerate MMK (7.1) by selecting a small number points to represent the group. Essentially, herding is letting the selected points to approximate the original distribution of the group. It is similar to the k-Means condenser described in our next Section, but it is much more complex and less practical in large data scenarios.

In point-based learning, condensing point sets is embodied in the *prototype selection* (PS) problem [53]. In prototype selection, we are given one training set of labeled points, and the goal is to reduce the size of the training set while maximizing the performance of the resulting classifier. We can see that PS is very different from condensing in the context of set-based learning. PS handles *one* set and focuses on the behavior of individual points. In contrast, set-based learning handles multiple sets and focuses on the set as a whole. PS methods usually focus on discriminating the

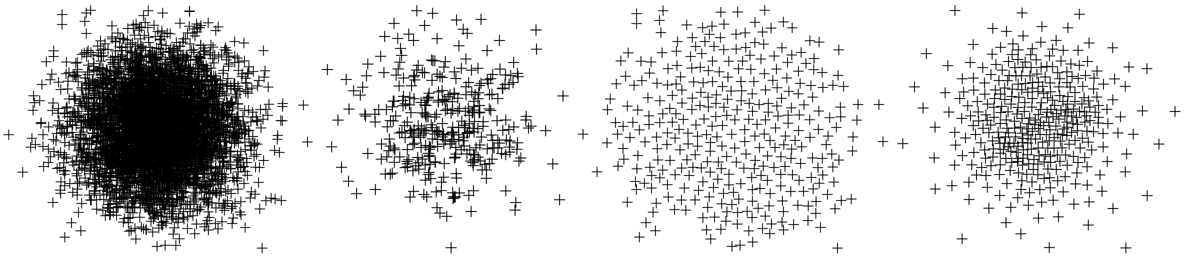
[1] `http://www.cs.ubc.ca/~mariusm/index.php/FLANN`

points instead of preserving the statistical properties of the sets that are needed in set-based learning. Several underlying techniques are shared between PS and condensing in set-based learning. Below we will comprehensively evaluate the techniques that are suitable for set-based learning. Our contribution is the discovery of a good condensing method for set-based learning algorithms that can *improve their speed, space requirement, and accuracy all at the same time*.

7.4 Condensing Methods

We examine three potential ways of condensing a point set G of size N to a smaller point set $\tilde{G}$ of size K, so that the properties of G that are useful to set learning are preserved in $\tilde{G}$. These three methods are selected for the following reasons:

- They have sound theoretical bases.
- They are easy and efficient to use in large-scale problems.
- They are universal *i.e.* not coupled with the subsequent learning algorithms, and thus widely applicable.



(a) All points (b) Random sampling (c) Uniform covering (d) k-Means

Figure 7.1: Condensing results of a 2D standard Gaussian point set using different condensers.

7.4.1 Random Sampling

Statistically, random subsampling can create smaller point sets $\tilde{G}$ with the same statistical properties as the original set G. This is the common method used to make trade-offs between speed and accuracy [133], and it is essentially the *asymmetric approach* used in [162]. It is easy to implement with little computational cost. However, when given only one chance to subsample, the result is random with possibly high variance, and might lead to poor results. Figure 7.1b shows an example in which the subsampled set is far from an ideal representation of the original set. We will use this method as our baseline.

7.4.2 Uniform Covering

We can also control the condensing quality at the point level. Since many learning algorithms are based on NNs, we require for any point x, the change of distance to its NN in G is bounded individually after condensing. Formally, we are looking for a $\tilde{G}$ such that

$$|\|x - \text{NN}_{\tilde{G}}(x)\|_2 - \|x - \text{NN}_G(x)\|_2| \le r, \forall x,$$

where r is the error threshold.

To find such a $\tilde{G}$ with minimal size is NP-hard [11]. [159] proposed *kernel vector quantization* which used the l_1 relaxation and provided an approximate solution using *linear programming*, but it can be too slow when condensing large sets. Instead, we adopt the following simple solution. Starting from an empty $\tilde{G}$, a) pick a point x from G and move x into $\tilde{G}$; b) Remove points in G whose distance to x is less than r; Repeat a) and b) until G is empty. The points in $\tilde{G}$ form the centers of radius-r spheres that uniformly cover the support of G's underlying distribution. Therefore, we call this *uniform covering* condensing. An illustration is given in Figure 7.1c.

To minimize the size $\tilde{G}$, a heuristic is to pick points in denser regions early so that more points can be removed. [78] implemented such a heuristic. It runs *Mean Shift* [32] to find a local peak of the density, and then picks a point from the peak. The complexity is $O(N^2)$ for a suitable r.

Though it seems well motivated, this approach is flawed. First, we cannot control the number of points to cover the support. To find a proper r we would need trial runs and tuning. Secondly, it only captures the support of the underlying distribution and ignores the actual density levels. The most severe problem, however, is the curse of dimensionality. In high dimensions, the neighborhood of a point becomes so large that we need a very large r to effectively trim the size down. Sometimes r is not much smaller than the diameter of the support, making the bound useless. We shall demonstrate this effect in the experiments.

7.4.3 k-Means

As a clustering algorithm, k-Means can also serve our condensing purposes well. We can run k-Means on the set G, and then use the set of cluster centers as $\tilde{G}$. Recall that k-Means minimizes the following objective

$$\tilde{G} = \{\tilde{x}_k\}_{k=1,\dots,K} = \arg\min_{\tilde{G}} \sum_{i=1}^{N} \|x - \text{NN}_{\tilde{G}}(x)\|_2^2. \tag{7.9}$$

In other words, it is trying to find a point set $\tilde{G}$ to best reconstruct G using the nearest neighbor method given the budget of K points.

We can prove that the k-Means condensing is actually maximizing the performance of NBNN. Recall that NBNN assumes that sets from the same class c share the same distribution f_c that is characterized by the merged class representation H_c, as described in Section 7.2.2. Then we assign a test set G to the class with the smallest KL-divergence according to (7.8). Hence, to find a condensed set $\tilde{H}_c$ for H_c that can maximize the classification performance of class c, we should minimize $\text{KL}(G||\tilde{H}_c)$ for any G with the distribution f_c. Since f_c is characterized by H_c, the objective should be

$$\tilde{H}_c = \arg\min_{\tilde{H}_c} \text{KL}(H_c||\tilde{H}_c). \tag{7.10}$$

We can see that $\tilde{H}_c$ should approximate the distribution of H_c. Further, by plugging (7.6) – the KL-divergence estimator used by NBNN – into (7.10), we see that (7.10) is indeed equivalent to the k-Means objective (7.9). In this sense, k-Means is the *ideal* condenser for NBNN classifiers. In the experiments we show that it is also generally good for other set learning algorithms.

Another advantage of k-Means is its efficiency. Being an extensively used and studied method, k-Means can be implemented with highly efficient algorithms for even massive data sets (*e.g.* [49, 153]). In practice, we also found that the exact solutions or even the local minima of k-Means is not required. Usually running k-Means for only tens of iterations is enough to achieve good condensing performance.

To sum up, k-Means provides a well-justified and efficient way to condense point sets. Figure 7.1d shows the visually appealing result of k-Means compared to other condensers. In the experiments, we will also show that it performs surprising well in classification tasks.

7.5 Empirical Evaluation

In this section we evaluate the performances of the above condensing methods when applied to various set learning algorithms in different image classification tasks. The three condensing methods described in Section 7.4 are tested. They are denoted as **Rand:K**, **Unif:K**, and **KMeans:K** respectively, where **K** is the condensed size. **All** is used to denoted the original set. We only evaluate the uniform covering condenser on a small problem to demonstrate its deficiencies. For the k-Means condenser, we run k-Means once using the random initialization for 20 iterations.

Five image data sets of different scales and natures are used for evaluation. For classification, as described in Section 7.2.2, we consider both the set-to-set scheme using the **MMK** (7.1) and the KL divergence based Gaussian kernel (7.7) (**KLK**), and the set-to-class scheme using **NBNN** [17], **LNBNN** [112], and **NPKL** which is the NBNN classifier paired with the divergence estimator (7.3).

For MMK and KLK, we use the condenser to reduce the size of every training and testing point set separately. For NBNN, LNBNN, and NPKL, the condensers are used to reduce the class representations $\{H_c\}$. When using KLK with SVM, the resulting kernel matrices are projected to the closest positive definite matrix as in [131]. For MMK and SVMs, the kernel width σ and slack parameter C are tuned on the training data using 3-fold cross-validation.

MMK, KLK, and Uniform Covering is only applied to small problems because they are slow compared to other methods. Only LNBNN is used for very large scale problems, since it is the only one that is fast enough.

We extract multiscale *dense SIFT* features called *PHOW* [18] for images. Images are resized so that the longest side is no larger than 256 pixels. The step size 10 is used to sample patches and the patch sizes are $[24, 36, 48]$. This setting will produce about $1,500$ 128-dimensional points for a 256×256 image. We also append the patches' spatial position in the image to the feature vectors to enable spatial matching, making the points 130 dimensional. The weight of the coordinates in distance calculation is roughly tuned on a small subset and kept the same throughout all the algorithms and runs.

The *VlFeat* [164] software is used for k-Means and dense SIFT feature extraction. The *FLANN*

[121] software is used for NN search. Exact NNs are used for KLK on small scale data sets. In large experiments of NBNN, LNBNN, and NPKL, we use approximate NNs with four randomized KDTrees. The number of *leaf node checks*, which controls the precision of the approximate NN search, is stated in each experiment and figures. Experiments are done on *Opteron K10 2.3 GHz* CPUs.

7.5.1 Scene-15

The first data set we consider is the *Scene-15* data set [92], which is a widely used benchmark for scene classification. This data set contains $4,485$ images from 15 classes. In general a scene image is characterized by the distribution of features *e.g.* the proportion of sky, water, flat surfaces, etc. This is quite different from images for object recognition that we will present later. For this data set, the weight of the spatial coordinates is set to 3. In each run, we randomly choose 100 images for training and 100 images for testing unless stated otherwise.

Set-to-Set Classification To use MMK and KLK for classification, we first calculate the kernel values between every pair of sets by (7.1) and (7.7), and then given them to SVM and KNN for classification. For KNN the number of neighbors is tuned based on the training data. Because this approach is very computationally expensive, we test it only on the first 8 classes, known as the *OT* data set [5]. Each image is a set of $1,542$ points, and the condensers Rand:100 and KMeans:100 are compared, meaning that we use subsampling and k-Means to reduce the sets' sizes from $1,542$ to 100. In each run, we randomly choose 50 images for training and 50 for testing.

The performance of 10 runs are reported in Figure 7.2. We can see that random sampling significantly decreased the accuracies of the classifiers. However, using the KMeans:100 condensing, the performances of SVMs are very close to using the original sets. For MMK, the decrease of mean accuracies is around than 0.8% and the Wilcoxon signed rank test shows a p-value of 0.07. For KLK, the difference is slightly larger at about 2%. This is a very good performance considering that only $1/15$ of the data is kept. It is interesting to see that KNN's accuracy using the k-Means condensed sets is significantly better than the uncondensed result. We shall see later this is not a random effect. As a reference, we also provide the results based on the distance between bag-of-words representations (500 visual words) using the same features and classifiers.

The time to compute both MMK and KLK using all the points is about $6,700$ CPU×minutes. After condensing it took about 26 CPU×minutes to compute MMK and 33 CPU×minutes to compute KLK, which are $200 – 250$ times faster. k-Means condensing only took about 0.1 CPU×second for each set.

The Uniform Covering Condenser Here we test the performance of uniform covering condenser paired with LNBNN classification on the full Scene-15 data set. The original H in LNBNN contains about 2×10^6 points. Note that with this condenser we cannot specify the size but only the radius r.

As mentioned in Section 7.4, this condenser is problematic in high dimensional spaces. Figure 7.3b shows the relationship between the condensed size and the radius r. To reduce the size to

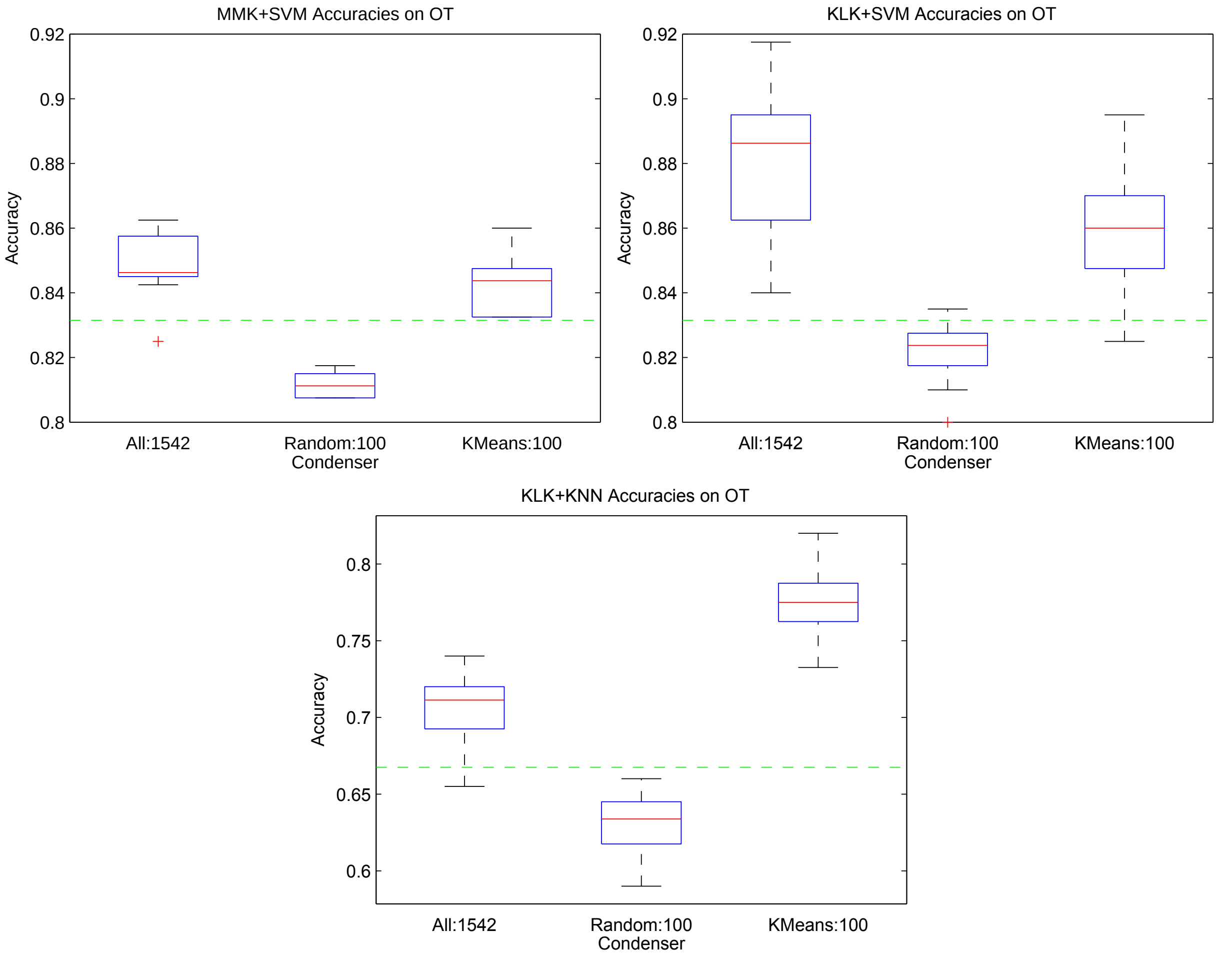


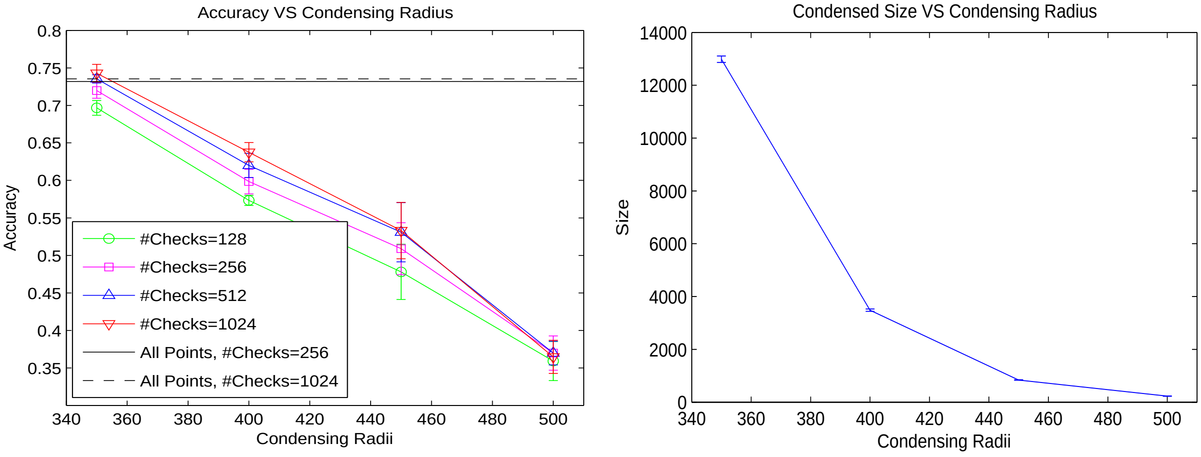
Figure 7.2: Accuracies on the OT data set using the original sets and the condensed sets. Green dashed lines are the accuracies of bag-of-words classifiers.

e.g. $3,000$, we need $r \approx 400$ which is very large. Either increasing or decreasing r would result in a dramatic change of size. This phenomenon is a natural result of the curse of dimensionality. In practice, it is very hard to get a good sense of how large the radius should be in high dimensions.

Figure 7.3a shows the relationship between the accuracy and the radius from 10 runs. Different number of checks in the approximate NN search is tried. We can see that the decreases of accuracy is unacceptable when the sets are condensed to a reasonable size. In fact, as we will show later, its performance is even worse than random sampling.

This experiment confirms that the uniform covering condenser has ill-formed behavior and bad performance. Moreover, it is also slower than other condensers. Therefore, we shall exclude it from the subsequent experiments.

The Random and k-Means Condenser Now we evaluate the sampling and k-Means condensers with the NBNN, LNBNN, and NPKL classifiers. Again the original classifiers contain about $2 \times$



(a) (b)

Figure 7.3: Performance of LNBNN on Scene-15 using the uniform covering condenser.

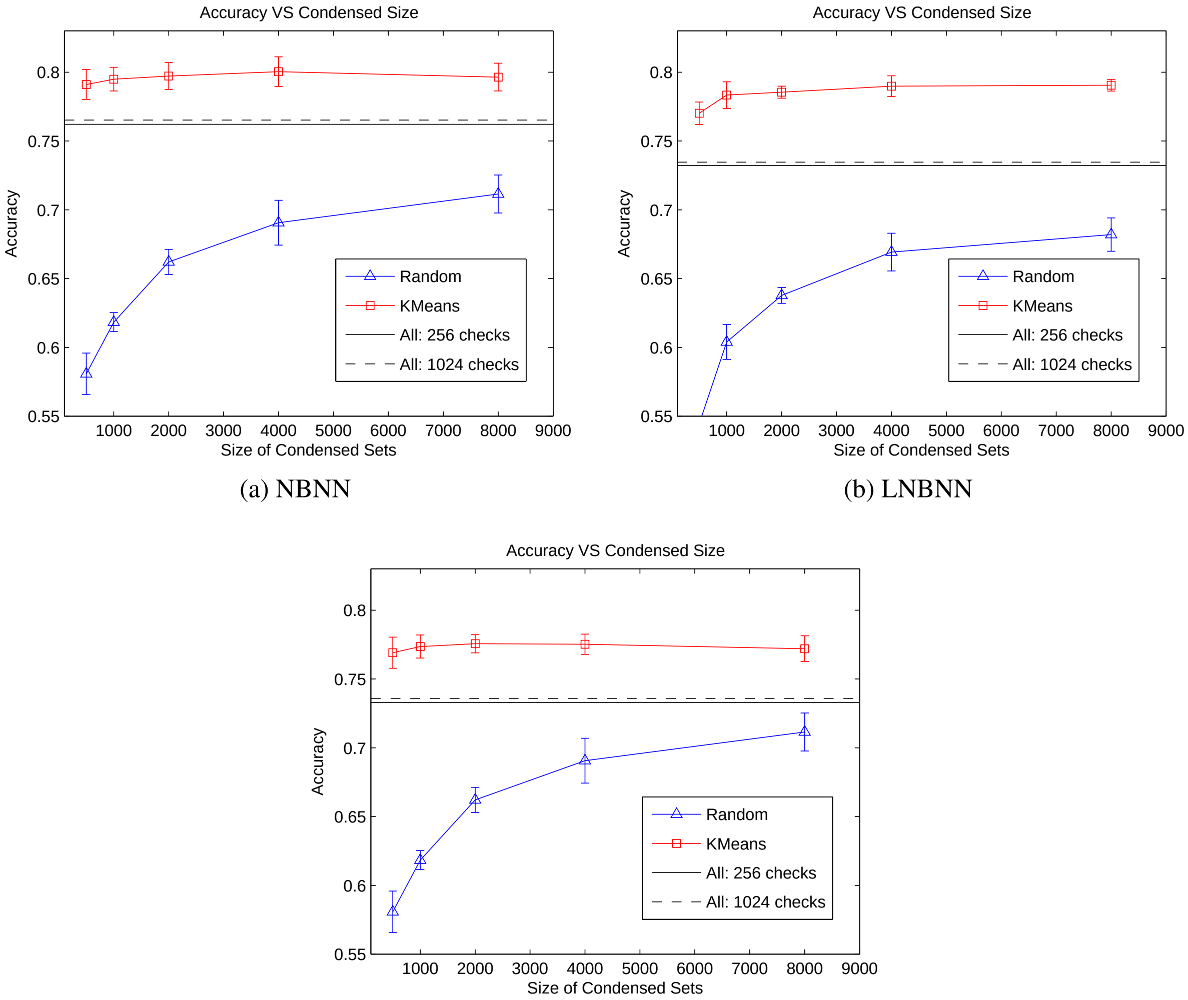
10^6 points. 512 checks are used in NN search. Figure 7.4 shows the accuracies from 5 random runs using different condensers and different classifiers. We can see that k-Means condensing is much better than random sampling. More surprisingly, classifiers using data condensed by k-Means consistently and significantly outperform those using the original uncondensed data. In other words, we *improved the speed and the accuracy simultaneously*, as opposed to make tradeoffs between them. The explanation might be that the condenser removes some of the noisy and outlier points in the original sets. In Figure 7.4a we can observe that the accuracy decreases a little when the condensed size is very large. We shall see that this behavior is consistent throughout most of our experiments.

We also examine the impact of the number of checks in NN search in Figure 7.5. The approximate search algorithm performs very well. The impact of the number of checks is minimum, and the performance usually saturates with 512 checks.

7.5.2 UIUC-Sports

The UIUC-Sports data set [100] contains $1,030$ images from 8 sport events. In order to test the performance in higher dimensions, we use the color version of the PHOW feature which is 384 dimensional. The image sizes vary so that each one contains 295 to $1,542$ points. In each run 70 images are used for training and 60 for testing. As a result, the original classifiers contain about 6×10^5 points. The weight of spatial coordinates is 0.6. Other settings remain the same as the Scene-15 experiment.

Accuracies of 5 random runs are reported in Figure 7.6. We can observe again that the k-Means condensing is better than both sampling and no-condensing. This verifies again the benefit of removing noise brought by condensing. We also noticed that using all the points, the accuracy of NPKL is worse than NBNN and LNBNN. But after condensing, these three algorithms performs almost the same. This shows that k-Means condensing could make the data less sensitive to



(a) NBNN

(b) LNBNN

(c) NPKL

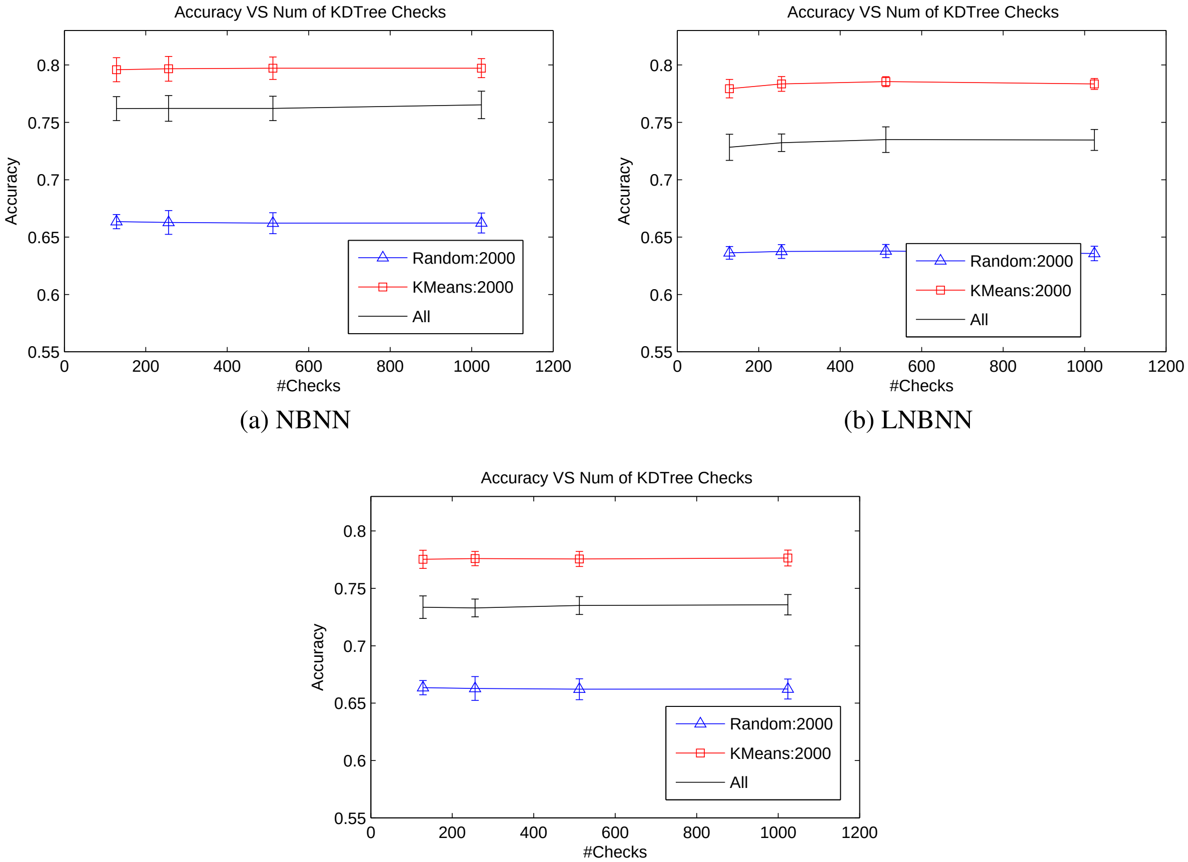
Figure 7.4: Scene-15 classification performances using different classifiers and condensers.

different algorithms.

7.5.3 CalTech-101

The CalTech-101 data set [98] is a standard benchmark for object recognition. This data set contains $9,144$ images of 102 different object classes. Unlike the Scene-15 and UIUC-Sports data set, the class of an object's image is more determined by the presence of a few distinctive local features (intuitively the object parts) than the distribution of features.

We follow the standard protocol and use $10, 15, 30$ images per class for training and the rest for testing. We only test the performance of LNBNN using different condensers as it is the only classifier that scales well with this problem. We compare the accuracies of Rand:4000 and K-Means:4000 condensers to the accuracy without condensing. Without condensing, the classifier



(a) NBNN

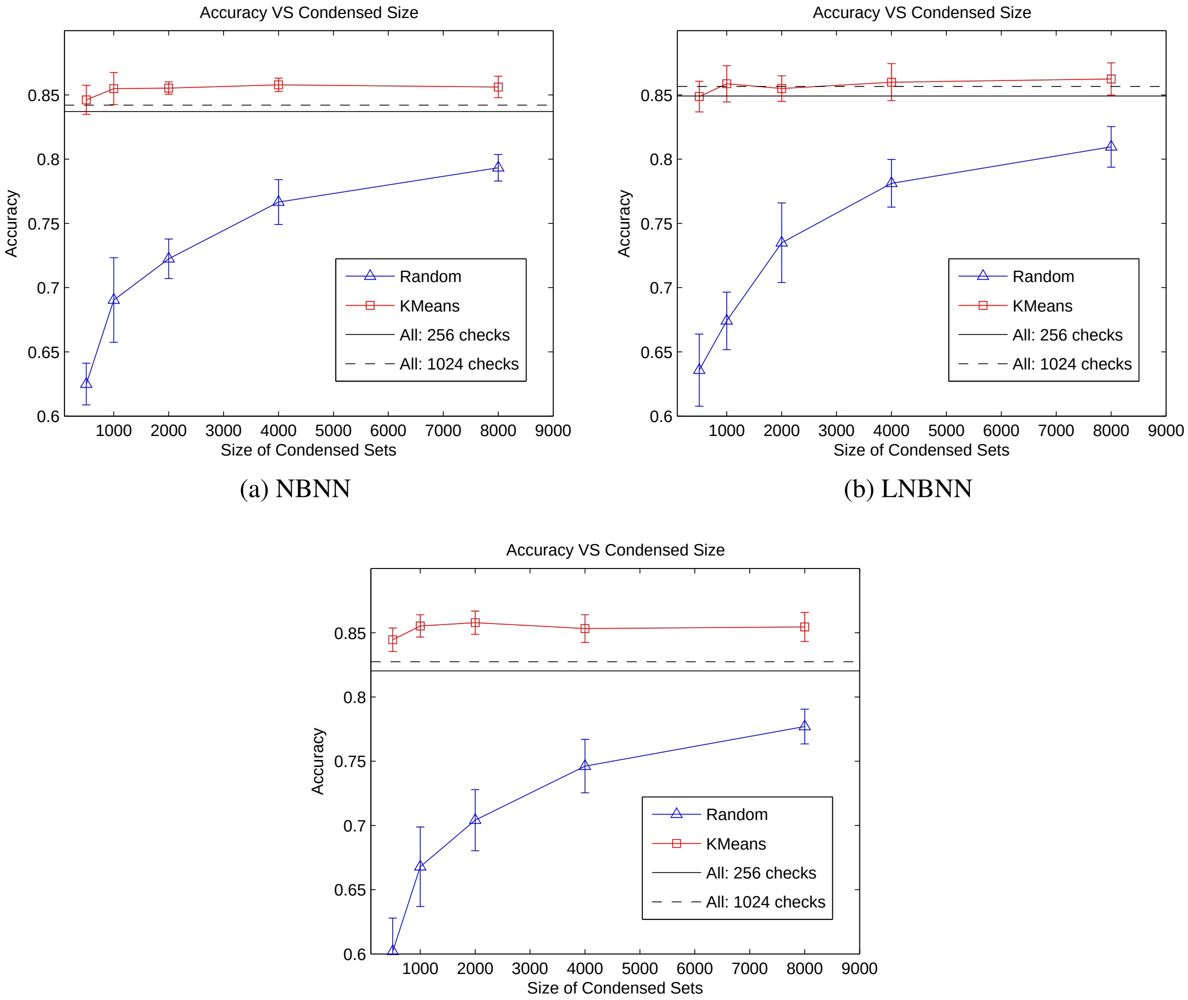
(b) LNBNN

(c) NPKL

Figure 7.5: The impact of the number of checks in the NN search to different methods on the Scene-15 data set.

contains 6×10^6 points taking about 3GB memory given 30 training images, while the condensed classifier only takes 202MB memory irrespective of the number of training images. Note that we use a larger condensed size of $4,000$ expecting that more points are needed to accurately capture the distinctive features of the objects. The weight of spatial coordinates is 1.5.

Figure 7.7a shows the performance of 10 random runs. We can see that k-Means is much better than the random condensing. k-Means slightly outperforms the uncondensed results again, even though the difference is insignificant. Observe that the improvement of k-Means condensing over using the all points is becoming smaller as more training images are used. This may be because as more images are added, $4,000$ points is becoming insufficient to capture all the information contained in the training set. The time used for k-Means condensing is 4 CPU*minutes per class with 30 training images. The prediction takes 100 CPU*minutes using the condensed classifier, while without condensing it takes 161 CPU*minutes. The acceleration is not much here because the highly efficient approximate NN search is used. Yet reducing the space requirement by 93% is still a significant benefit.



(a) NBNN

(b) LNBNN

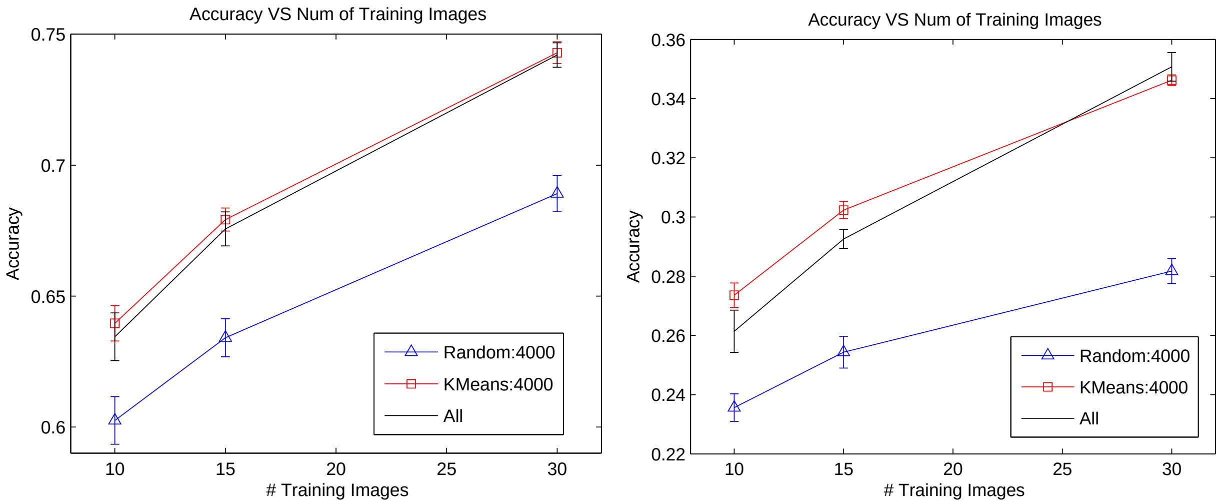
(c) NPKL

Figure 7.6: UIUC-Sports classification performances using different classifiers and condensers.

7.5.4 CalTech-256

CalTech-256 is an enlarged version of the previous CalTech-101 data set, containing $30,607$ images from 257 object classes. The same settings are used as for CalTech-101 except that the weight of spatial coordinates is 0.6. Note that without condensing, the LNBNN classifier contains 1.4×10^7 points taking 7GB memory, which is approaching the limit of readily available machines. After the condensing, the classifier takes only 500MB memory, irrespective of the number of training images.

Figure 7.7b shows the performance of 5 random runs. The behaviors of the condensers are basically the same as in the CalTech-101 experiment, showing the consistency of the condensers. Notably, when 30 training images are used, the condenser seems to have reached the limit and causes a slight decrease of accuracy. It shows that the information carried by the training set is



(a) (b)

Figure 7.7: (a) CalTech-101 classification accuracies using LNBNN with different condensers. (b) CalTech-256 classification accuracies using LNBNN with different condensers.

finally exceeding the capacity of the KMeans:4000 condenser and more points are needed to maintain performance. The time used for k-Means condensing is again 4 CPU$\times$minutes per class with 30 training images. The prediction takes 440 CPU$\times$minutes using the condensed classifier, while without condensing it takes 791 CPU$\times$minutes. In this larger problem the condenser's acceleration effect is becoming more prominent, even if the approximate NN searcher is used.

7.5.5 ImageNet Challenge 2012

The ImageNet Challenge 2012[2] [41] provides a massive object image classification task, containing $1,261,406$ images from $1,000$ object classes retrieved by crawling the Internet. The large amount of variations in the perspective, object appearance, and background clutter make it a extremely challenging task. Because of the large number of classes and possible ambiguities, 5 guesses are allowed when predicting an image's label.

We use the dense SIFT features provided by the organizer[3], which provides about 800 SIFT vectors per image. We apply LNBNN to this classification task with 500 images per class for training and 500 images per class for testing, resulting in an experiment that involves 1 million images. This experiment is too large to be feasible on reasonable machines without condensing; the training set alone would take 140GB memory.

Rand:2000 and KMeans:2000 condensers are used in this task. The size of the classifier after condensing is about 1GB. Since we are not able to complete the task with all the training points, the condensed result by Rand:20000 is used as a surrogate, which is feasible but already runs very

[2] `http://www.image-net.org/challenges/LSVRC/2012/index`
[3] `http://www.image-net.org/download-features`

slow. For the k-Means condensing, we first use random sampling to reduce the input sets' sizes to 10^5 and then run k-Means. 256 checks are used in the NN search, and the weight of spatial coordinates is 0.9.

The results from 5 runs of KMeans:2000, Rand:2000 and 2 runs of Rand:20000 are shown in Table 7.1. We can see that k-Means condenser performs around 70% better than the sampling condenser using the same amount of data, and also 11% better than the sampling condenser that uses 10 times more data, showing the effectiveness of k-Means condensing in optimizing the classification performance.

The running time for different condensers are also reported. Note that here "Training" is just the condensing step. We see that even if the approximate NN searcher is used, condensing can still make the prediction speed 6 times faster, and this improvement will become significantly larger if more accurate NN search is needed. The sampling condenser basically costs no time except for the disk IO. On the other hand, the k-Means condensing takes less than 4 minutes per class. In large-scale parallel computation, this extra cost is acceptable, and the improvement to the prediction speed and accuracy is significant. In all, again, condensing makes the classifier smaller, faster, and more accurate.

Note that our results here are mainly to show the effectiveness of the k-Means condenser and not comparable to the ImageNet Challenge's top performers. We used the provided features instead of doing feature engineering/learning, and the algorithm used here is extremely simple and efficient.

Condenser	Rand:2000	Rand:20000	KMeans:2000
Accuracy (%)	14.04 ± 0.05	21.18 ± 0.07	23.7 ± 0.55
Training Time	0.07	0.17	3.7
Testing Time	0.52	2.91	0.53

Table 7.1: Accuracies and running time of LNBNN on ImageNet. The training time is measured by CPU*minute per class and the testing time is measured by CPU*second per test image.

7.6 Discussion

Typically when facing large point sets, a popular approach is to subsample them to make a trade-off between accuracy and speed [133]. Our experiments show that this approach often compromises too much accuracy. However, when we use the k-Means condenser, we can often improve the speed, space requirement, and the accuracy all at the same time.

Depending on the data set, the k-Means condensing can have different impact on the performance. When the sets are mainly characterized by the holistic characteristics of its points, k-Means condensing can not only reduce the size significantly while retaining the information, but it can also possibly remove noise and outliers to enhance the accuracy. Examples of such data sets include the Scene-15 and the UIUC-Sports. If the sets are mainly characterized by a few distinctive points, like in the CalTech data sets, approximation error on the individual points plays a bigger role and condensing is usually less effective. Nonetheless, even in those data sets, we see that k-Means can still at least maintain the accuracy while greatly improve the time and space efficiency.

To use the condensing algorithms, we need to choose the size of the condensed set. A general guideline is to set the budget of time and space and use the largest number of points allowed. Our experience shows that $1,000$ – $5,000$ points usually works well for set-vs-class classifiers, and 100 – 500 points should work for set-vs-set classifiers. If the purpose is to use condensing to remove the noise and improve the accuracy, then we can use cross-validation to determine the appropriate size.

The cost of k-Means is not trivial but very manageable. The condensing of different sets are independent. In our experiments, we used Elkan's algorithm [49] in VLFeat [164], which is not the fastest algorithm like [153] but can still condense 10^5 points to $2,000$ points in less than 4 minutes. In a large-scale parallel computation environment like *MapReduce*, this is very acceptable. We believe that, given the budget of time and space, it is almost always beneficial to apply k-Means condensing before a learning algorithm on point sets. Compared to random sampling, it will result in a much better accuracy within acceptable time.

For algorithms that need all point-wise similarities or exact NNs in high dimensions, condensing can easily provide quadratic improvement for speed and linear improvement for space requirement. In this latter case, condensing can turn impossible tasks into possibilities. When approximate NN search is used and we have a small data set, the improvement for speed is not significant, such as in the Scene-15 and UIUC-Sports data set. However, when the data set becomes as big as the CalTech-256 or even the ImageNet datasets, then condensing can provide substantial improvement on top of the approximate NN search.

7.7 Summary

Efficient algorithms for learning from point sets are important and useful, yet existing methods suffer from high time and space demand. In this work we tried to condense the point sets (reduce the size of the sets) in order to make these methods faster and better. We found that condensing is in general a better strategy than imputing the kernels as described in Chapter 6. Particularly, we discovered that the k-Means algorithm does this job very well.

On a wide range of classification methods and image data sets, we evaluated three different practical condensing strategies and found the k-Means is the only one that can successfully reduce the size of point sets without much loss of accuracy. In many cases it even improves the accuracy by removing the noise and outliers. This success seems to be universal despite the differences across various classifiers and data sets. We hope our discovery could help the adoption of the point-set based methods by the practitioners in large scale problems.

Chapter 8

Sampling Bias Correction by Conditional Divergences

Many objects can be represented as sets of multi-dimensional points. A common approach to learning from these point sets is to assume that each set is an *i.i.d.* sample from an unknown underlying distribution, and then estimate the similarities between these distributions. In realistic situations, however, the point sets are often subject to sampling biases. These biases can fundamentally change the distributions and distort the results of estimation and learning. In this work we propose to use conditional divergences to correct these distortions and learn from biased point sets effectively. Our empirical study shows that the proposed method can successfully correct the biases and achieve satisfactory learning performance.

8.1 Introduction

Traditional learning algorithms deal with vectors/points, but many real objects are actually sets of points that are multi-dimensional, real-valued vectors. For instance, in monitoring problems, each sensor produces one set of measurements for a particular region within a time period. A traditional way to deal with point sets is to construct feature vectors for the sets through discretization so that standard learning techniques can be applied. However, this conversion process often relies on human effort and is prone to information loss. Recently, several algorithms were proposed to directly learn from point sets based on the assumption that each point set is a sample from an unknown distribution, including methods proposed in this thesis. [130, 131] proposed novel kernels between point sets based on efficient and consistent divergence estimators. [65, 120] took a similar approach and designed a kernel based on the kernel embedding of distributions. [17, 112] developed extremely simple classifiers for point sets based on divergences between sets and classes. These methods achieved impressive empirical successes, showing the advantage of learning directly from point sets.

One factor that can significantly affect the effectiveness of learning is sampling bias. Sampling bias comes from the way we collect points from the underlying distributions, and makes the observed sample not representative of the true distribution. It undermines the fundamental validity of learning because the points are no longer *i.i.d.* samples from a distribution conditioned only on the object's type. Though it has been extensively studied in statistics, this key problem has been largely ignored by the previous research on learning from sets. The goal of this paper is to alleviate the impact of sampling bias when measuring similarities between point sets.

We consider point sets with the following structure. Let each point be described by two groups of random variables: the independent variables (*i.v.*) and dependent variables (*d.v.*). A point is collected by first specifying the value of the *i.v.*, and then observing a sample from the distribution of the *d.v.* conditioned on the given *i.v.* Figure 8.1 shows a synthetic example where the *i.v.* is sampled uniformly, and the *d.v.* is from the Gaussian distribution whose mean is proportional to the value of *i.v.*, forming the black line-shaped point set. Many real world situations, including surveys and mobile sensing, produce point sets of this type. In patch-based image analysis, we first specify the location of the patches as the *i.v.* and then extract their features as the *d.v.* In traffic monitoring, a helicopter is sent to specific locations at specific times (*i.v.*) and measures the traffic volume (*d.v.*).

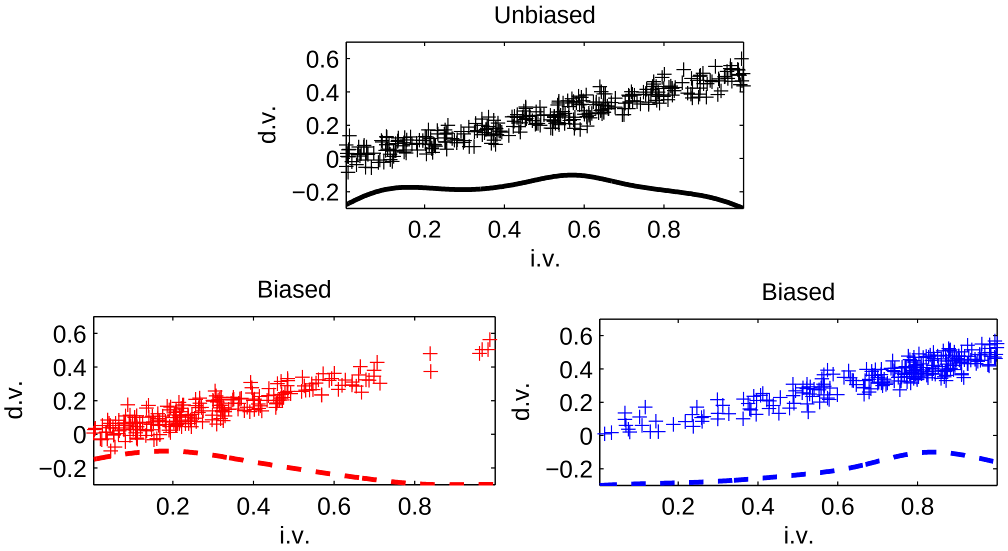


Figure 8.1: The observation biases.

We assume that the sampling bias affects the way we observe *i.v.* , yet the observation of *d.v.* given *i.v.* remains intact. This assumption is compatible with the covariate shift model [73, 152]. As shown in Figure 8.1, an unbiased observer will sample *i.v.* uniformly and get the black set. Biased observers might focus more on the smaller or larger values of the *i.v.* and create the biased red and blue sets, where the curves show the observed marginal densities of the *i.v.* The joint and marginal distributions of the biased sets now look very different from each other and the unbiased set. Nevertheless, no matter what the distribution of *i.v.* is, the distribution of *d.v.* given *i.v.* is always the same Gaussian that does not change with the observer. In traffic monitoring, the helicopter may be tasked with other, non-traffic, jobs that create different patrol schedules each day, thus creating an uneven profile of the city's traffic. But the measured traffic volumes at the patrolled locations are still accurate.

To correct sampling biases of this kind, we propose to use conditional divergences. Existing divergence-based methods use the joint distribution of the *i.v.* and the *d.v.* to measure the differences between point sets. On the other hand, conditional divergences focus on the conditional distributions of *d.v.* given *i.v.* and are insensitive to the distribution of *i.v.*, which is distorted by the sampling bias in our setting. As long as the conditional distributions are intact, the conditional divergences will be reliable. Moreover, it can be shown that the divergence between joint distributions is a special case of the conditional divergence. A fast and consistent estimator is developed for the conditional divergences. We also discuss specific examples of correcting sampling biases, including some extreme cases.

We evaluate the effectiveness of conditional divergences on both synthetic and real world data sets. On synthetic data sets, we show that the proposed estimator is accurate and the conditional divergences are capable of correcting sampling biases. We also demonstrate their performance on real-world climate and image classification problems.

The rest of this paper is organized as follows. The background and some related work is introduced in Section 8.2. Section 8.3 defines the conditional divergence and describes its properties and estimation. Section 8.4 describes how to use conditional divergence to correct various sampling biases. In Section 8.5 we make a discussion about the conditional divergences. In Section 8.6, we evaluate the effectiveness of the proposed methods on both synthetic and real data sets. We conclude the paper in Section 8.7.

8.2 Background and Related Work

We consider a data set that consists of M point sets $\{G_m\}_{m=1,\ldots,M}$, and each point set G_m is a set of d-dimensional vectors, $G_m = \{z_{mn}\}_{n=1,\ldots,N_m}$, $z_{mn} \in \mathbb{R}^d$. Each point $z_{mn} = [x_{mn}; y_{mn}]$ is a concatenation of two shorter vectors $x_{mn} \in \mathbb{R}^{d_x}$ and $y_{mn} \in \mathbb{R}^{d_y}$ representing the independent variables *i.v.* and the dependent variables *d.v.* respectively. We assume that each G_m has an underlying distribution $f_m(z) = f_m(x, y)$, and the points $\{z_{mn}\}$ are *i.i.d.* samples from $f_m(z)$. f_m can be written as $f_m(z) = f_m(y|x)f_m(x)$. In the context of image classification, each G_m is an image, and x_{mn} is the location of the nth patch and y_{mn} is the feature of that patch.

We can learn from these sets by estimating the divergence between the f_m's as the dissimilarity between the G_m's. Having the dissimilarities, various problems can be solved by using similarity based learning algorithms, including *k-nearest neighbors* (KNN), *spectral clustering* [122], and *support vector machines* (SVM). In this direction, several divergence-based methods have been proposed [17, 120, 131], and both empirical and theoretical successes were achieved.

In the presence of sampling bias that affects the distribution of *i.v.*, $f_m(x)$ is transformed into $f'_m(x)$. Consequently the observed G_ms represent the biased joint distribution $f'_m(z) = f_m(y|x)f'_m(x)$. Therefore naïvely learning from the point sets using joint distributions will lead us to the distorted f'_m's instead of the true f_m's. To correct the sampling bias, we need to either 1) modify the point sets to restore $f(z)$, or 2) use similarity measures that are insensitive to $f(x)$.

Existing correction methods often reweigh the points in the training set so that its effective distribution matches the distribution in the test set [34, 73, 152]. Our proposed conditional divergences are insensitive to the biased distributions of the independent variables and thus robust against sampling biases.

Traditionally in statistics and machine learning, sampling bias is considered between the training set and the test set. In contrast, we consider problems consisting of a large number of point sets, and our goal is to learn from the sets themselves. This extension raises many important challenges, including how to find a common basis to compare all pairs of distributions, how to deal with unobserved segments of distributions, and how to design efficient algorithms.

To our knowledge, this is first time sampling bias is addressed in the context of learning from sets of points. Algorithms such as [17, 65, 76, 112, 120, 130, 131] all directly compare the joint distributions of the observed points, making them susceptible to sample bias. [128] proposed the use of conditional divergence, yet sampling bias was still not considered.

8.3 Conditional Divergences

We propose to measure the dissimilarity between two distributions $p(z) = p(x, y)$ and $q(z) = q(x, y)$ using the *conditional divergence* (**CD**) based on the *Kullback-Leibler* (KL) divergence:

$$\mathrm{CD}_{c(x)}\left(p(z)||q(z)\right) = \mathbb{E}_{c(x)}\left[\mathrm{KL}\left(p(y|x)||q(y|x)\right)\right] \tag{8.1}$$

where $c(x)$ is a user-specified distribution over which the expectation is taken. CD is the average KL divergence between the conditional distributions $p(y|x)$ and $q(y|x)$ over possible values of x, and $c(x)$ can be considered as the importance of the divergences at different x's. CD's definition is free of the *i.v.* distributions $p(x)$ and $q(x)$, which are vulnerable to sampling biases. By definition, CD has a lot in common with the KL divergence: it is non-negative, and equals zero if and only if $p(y|x) = q(y|x)$ for every x within the support of $c(x)$. CD is also not a metric and not even symmetric.

In the form of (8.1), CD is hard to compute because the divergences $\mathrm{KL}\left(p(y|x)||q(y|x)\right)$ are not available for arbitrary continuous distributions. Also note that $c(x)$ is a distribution specified by the user. To make CD more accessible, we can rewrite it as

$$\mathrm{CD}_{c(x)}\left(p(z)||q(z)\right) = \mathbb{E}_{p(z)}\left[\frac{c(x)}{p(x)}\left(\ln\frac{p(z)}{q(z)} - \ln\frac{p(x)}{q(x)}\right)\right]. \tag{8.2}$$

Now, CD is defined in terms of the density ratios of the input distributions and the expectation over $p(z)$.

An interesting case of (8.2) occurs when we choose $c(x) = p(x)$, which gives the result

$$\mathrm{CD}_{p(x)}\left(p(z)||q(z)\right) = \mathrm{KL}(p(z)||q(z)) - \mathrm{KL}(p(x)||q(x)). \tag{8.3}$$

We can see this special CD is equal to the *joint divergence* (divergence between joint distributions) minus the divergence between the marginal distributions of x. Intuitively, CD is removing the effect of $p(x)$ and $q(x)$ from the joint divergence, so that the net results are free from the sampling bias. Moreover, when $p(x)$ and $q(x)$ are the same, $\mathrm{KL}(p(x)||q(x))$ vanishes and this CD equals the joint divergence. In other words, when there is no sampling bias, $\mathrm{CD}_{p(x)}\left(p(z)||q(z)\right) = \mathrm{KL}(p(z)||q(z))$.

8.3.1 Estimation

In this section we give an estimator for CD (8.2). Suppose we have two sets G_p and G_q with underlying distributions $p(z)$ and $q(z)$ respectively. We can approximate the expectation (8.2) with the empirical mean and estimated densities:

$$\widehat{\text{CD}}_{c(x)}\left(p(z)||q(z)\right) = \frac{1}{N_p}\sum_{n=1}^{N_p}\frac{c(x_{p,n})}{\hat{p}(x_{p,n})}\left(\ln\frac{\hat{p}(z_{p,n})}{\hat{q}(z_{p,n})} - \ln\frac{\hat{p}(x_{p,n})}{\hat{q}(x_{p,n})}\right), \tag{8.4}$$

where N_p is the size of G_p, $\hat{p}, \hat{q}$ are the estimates of p, q.

$c(t)$ is an arbitrary input from the user and we can see that its role is to reweight the log-density-ratios at different points in G_p. To generalize this notion, we define the *generalized conditional divergence* (GCD) and its estimator as the weighted average of the log-density-ratios:

$$\text{GCD}_w\left(p(z)||q(z)\right) = \sum_{n=1}^{N_p} w(x_{p,n})\left(\ln\frac{p(z_{p,n})}{q(z_{p,n})} - \ln\frac{p(x_{p,n})}{q(x_{p,n})}\right) \tag{8.5}$$

$$\widehat{\text{GCD}}_w\left(p(z)||q(z)\right) = \sum_{n=1}^{N_p} w(x_{p,n})\left(\ln\frac{\hat{p}(z_{p,n})}{\hat{q}(z_{p,n})} - \ln\frac{\hat{p}(x_{p,n})}{\hat{q}(x_{p,n})}\right) \tag{8.6}$$

$$\sum_{n=1}^{N_p} w(x_{p,n}) = 1, w(x_{p,n}) \geq 0,$$

where $w(x)$ is the weight function and the constraint $\sum_n w(x_n) = 1$ is induced by the fact that

$$\lim_{N_p\to\infty}\sum_{n=1}^{N_p} w(x_{p,n}) = \lim_{N_p\to\infty}\frac{1}{N_p}\sum_{n=1}^{N_p}\frac{c(x_{p,n})}{p(x_{p,n})} = \mathbb{E}_{p(x)}\left[\frac{c(x)}{p(x)}\right] = \int\frac{c(x)}{p(x)}p(x)dx = 1.$$

To obtain the density estimates $\hat{p}, \hat{q}$, we use the *k-nearest-neighbor* (KNN) based estimator [105]. Let the $f(z)$ be the d-dimensional density function to be estimated and $Z = \{z_n\}_{n=1,\ldots,N} \in \mathbb{R}^d$ be samples from $f(z)$. Then the density estimate at the point z' is

$$\hat{f}(z') = \frac{k}{Nc_1(d)\phi^d_{Z,k}(z')}, \tag{8.7}$$

where $c_1(d)$ is the volume of the unit ball in the d-dimensional space, and $\phi_{Z,k}(z')$ denotes the distance from z' to its kth nearest neighbor in Z (if z' is already in Z then it is excluded). This estimator is chosen over other options such as the *kernel density estimation* because it is simple, fast, and leads to a provably convergent estimator as shown below.

By plugging in (8.7) into (8.6), we can get the following estimator for GCD:

$$\widehat{\text{GCD}}_w\left(p(z)||q(z)\right) = \sum_{n=1}^{N_p} w(x_{p,n})\left(d\ln\frac{\phi_{G_q,k}(z_{p,n})}{\phi_{G_p,k}(z_{p,n})} - d_x\ln\frac{\phi_{G_q,k}(x_{p,n})}{\phi_{G_p,k}(x_{p,n})}\right), \tag{8.8}$$

where d_x is the dimensionality of the x. We can see that the resulting estimator has a simple form and can be calculated based only on the KNN statistics ϕ, which are efficient to compute using space-dividing trees or even approximate KNN algorithms such as [121]. Also note that even though the estimator (8.8) is obtained using the density estimator (8.7), its final form only involves simple combinations of the log-KNN-statistics $\ln \phi$. Thus, this GCD estimator effectively avoids explicit density estimation which is notoriously difficult, especially in high dimensions.

More importantly, the GCD estimator (8.8) has stronger convergence properties than the density estimator from which it is derived. Standard convergence results have that the density estimator (8.7) is statistically consistent only if $k/n \to 0, k \to \infty$ simultaneously. However, for estimator (8.8) convergence can be achieved even for a fixed finite k. This means that we can always use a small k to keep the nearest neighbor search fast and still get good estimates. Specifically, following the work of [129, 168], the following theorem can be proved:

Theorem 3. *Suppose the density function pairs* $(p(z), q(z))$ *and* $(p(x), q(x))$ *are both* 2*-regular (as defined in [168]). Also suppose that the weight function satisfies* $\lim_{N_p \to \infty} w(x_{p,n}) = 0, \forall n$*. Then the estimator* (8.8) *is* L^2 *consistent for any fixed* k*. That is*

$$\lim_{N_p, N_q \to \infty} \mathbb{E}\left[\widehat{GCD}_w(p(z)||q(z)) - GCD_w(p(z)||q(z))\right]^2 = 0 \qquad (8.9)$$

The proof of Theorem 3 is similar to what was used in [168]. The condition $\lim\limits_{N_p \to \infty} w(x_{p,n}) = 0$ ensures that the weight function does not concentrate on only a few points. We omit the detailed proof here. Note that the convergence of GCD does not carry to CD (8.4) because the weight function $w(x_{p,n}) = \frac{c(x_{p,n})}{\hat{p}(x_{p,n})}$ is no longer deterministic. However, empirically we found that (8.4) exhibits the behavior of a consistent estimator and produces satisfactory results.

8.4 Choosing $c(x)$

To use CD, we have to choose the appropriate $c(x)$ or $w(x)$. When learning from point sets, it is preferable to use the same $c(x)$ to compute the CDs between all pairs of sets, so that they have a common basis to compare. However, this is not always necessary or possible. Even though the choice of $c(x)$ and $w(x)$ can be arbitrary, we consider 3 options below.

First, we can let $c(x) \propto 1$ so that $w(x_{p,n}) \propto p^{-1}(x_{p,n})$ to treat every value of x equally. The disadvantage is that $p^{-1}(x_{p,n})$ has to be estimated, which is error prone. We can also use $c(x) = p(x)$ and $w(x_{p,n}) \propto 1$, leading to (8.3). In this case, different pairs of sets can have different $c(x)$'s. When the sampling bias is small, these differences might be acceptable considering the possible errors in $w(x)$ otherwise. Thirdly, $c(x) \propto p(x)q(x)$ and $w(x_{p,n}) \propto q(x_{p,n})$ puts the focus on regions where both $p(x)$ and $q(x)$ are high. It means that we should put larger weights in dense regions and avoid scarce regions to get reliable estimates.

One caveat is that the weight function and the log-density-ratios in CD should not use the same density estimate, otherwise the estimation errors will correlate and cause systematic over-estimations. Using different estimators can help decouple the errors and avoid accumulation. In practice, we use the estimator (8.7) with a different k.

Some extreme cases of sampling bias are when whole segments of the distribution are missing from the sample and therefore unobserved. Two sets can even have disjoint supports of x. With the CD, we can choose $c(x) \propto p(x)q(x)$ or $c(x) \propto I(p(x)q(x) > 0)$, where $I(\cdot)$ is the indicator function, and only compare two sets in their overlapping regions. The result may not be accurate with respect to the true divergence, but it is still a valid measurement of the differences between conditional distributions. When $f(y|x)$ only weakly depends on x, this estimate can be an adequate approximation to the original divergence. If $f(y|x)$ varies drastically for different x's without any regularity then only comparing the overlapping regions might be the best we can do.

When two sets have disjoint supports in x, no useful information can be extracted and the corresponding divergence has to be regarded as missing without further assumptions. Nevertheless, in our settings where a large number of point sets are available, it is likely that each set will share its support in x with at least some others to provide a few reliable divergence estimates. We might be able to infer the divergence between disjoint sets using the idea of triangulation. We shall leave this possibility for future investigation.

8.5 Discussion

In CD, $c(x)$ conveys prior knowledge about the importance of different x's. It should be carefully chosen based on the data, and poor results can happen when the assumptions made in $c(x)$ are not valid. For example, $c(x) \propto 1$ assumes that all the x's are equally important. This could be a bad assumption when the supports of two sets do not overlap, because at some x's one of the densities will be zero, making the conditional densities $f(y|x)$ not well-defined. Similar problems might occur in regions where one of the densities is very low. Numerically the estimator can still work but usually produces poor results. In this scenario, $c(x) \propto p(x)q(x)$ suits the data better.

The CD estimator (8.8) relies on the KNN statistics ϕ which is the distance between nearest neighbors. Usually we use Euclidean distance to measure the difference between points and find nearest neighbors. However, the estimator does not prevent the use of other distances. In fact, [105] shows that alternative distances can be used and the consistency results will generally still hold. A common choice of adaptive distance measure is the *Mahalanobis distance* [156], which is equivalent to applying a linear transformation to the random variables. It is even possible to learn the distance metric for ϕ in a supervised way to maximize the learning performance. We leave this possibility as future work.

The estimated conditional divergences can be used in many learning algorithms to accomplish various tasks. In this paper, we use kernel machines to classify point sets as in [130, 131]. Having the divergence estimates, we convert them into Gaussian kernels and then use SVM for classification. When constructing kernels, all the divergences are symmetrized by taking the average $\mu(p, q) = \frac{d(p||q)+d(q||p)}{2}$. The symmetrized divergences μ are then exponentiated to get the Gaussian kernel $k(p, q) = \exp(-\gamma\mu(p, q))$ and the kernel matrix $\mathbf{K}$, where γ is the width parameter. Unfortunately, $\mathbf{K}$ usually does not represent a valid Mercer kernel because the divergence is not a metric and random estimation errors exist. As a remedy, we discard the negative eigenvalues from the kernel matrix $\mathbf{K}$ to convert it to its closest *positive semi-definite* (PSD) matrix $\tilde{\mathbf{K}}$. This $\tilde{\mathbf{K}}$ then is a valid kernel matrix and can be used in an SVM for learning.

8.6 Experiments

We examine the empirical properties of the conditional divergences and their estimators. The tested divergences are listed below.

- **Full D**: Divergence between full unbiased sets as the groundtruth.
- **D**: Divergence between biased sets.
- **D-DV**: Divergence between biased sets while ignoring the *i.v.*.
- **CD-P,CD-U,CD-PQ**: conditional divergences with $c(x) \propto p(x), c(x) \propto 1, c(x) \propto p(x)q(x)$ respectively between biased sets.

Full D, D, D-DV are estimated using the KL divergence estimator proposed by [168]. Unless stated otherwise, we use $k = 3$ for GCD estimation using (8.8), and use k values between 30 and 50 to compute the weight function.

We consider two types of sampling biases. The first type creates different $f(x)$'s for different sets, yet they still share the same support of x as the original unbiased data. Based on the first type, the second type of sampling bias is more extreme and can hide certain segments of the true distributions, and thus causes different sets to have different supports of x. We call the resulting test sets from these two sampling biases *uneven sets* and *partial sets* respectively.

In order to evaluate the quality of the bias correction by the CDs, we use controlled sampling biases in our experiments. The original point set data are collected from real problems without any sampling bias. Then we resample each set to create artificial sampling biases. By doing this, we can compare the results using the biased sets to the divergences using the unbiased data which is the groundtruth.

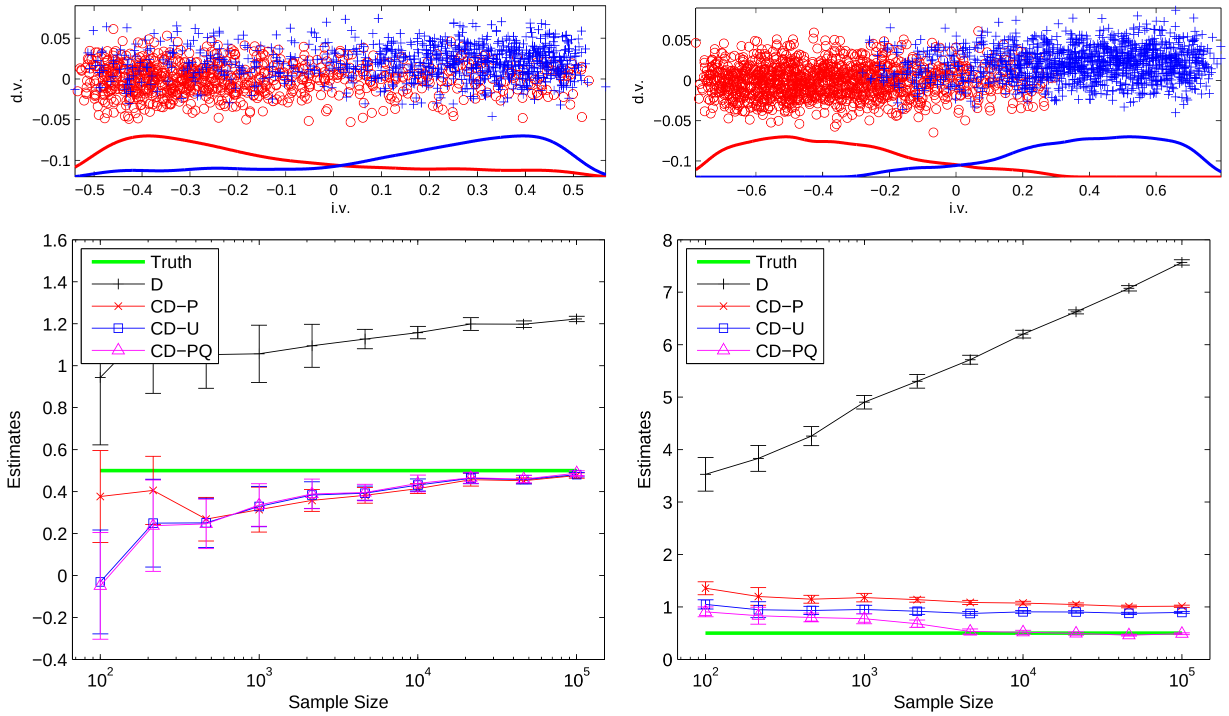
An SVM is used to classify the point sets using the method described in Section 8.5. When using the SVM, we tune the width parameter γ and the slack penalty C by 3-fold cross-validation on the training set.

8.6.1 Synthetic Data

Estimation Accuracy

We generate synthetic data to test the accuracy of the proposed conditional divergence estimators. The data set consists of 2-dimensional (one as *i.v.* and one as *d.v.*) Gaussian noise along two horizontal lines as the two classes, as shown in Figure 8.2a and 8.2b. The Gaussians have fixed spherical covariance, and the mean of the blue class is slightly higher than the red class, resulting in an analytical KL divergence of 0.5. Then the *i.v.* (x axis) is resampled to create sampling bias and the red and blue curves show the resulting marginal densities $f_{\text{red}}(x), f_{\text{blue}}(x)$. The task is to recover the true divergence value 0.5 from this biased sample. We vary the sample size to see the empirical convergence, and the results of 10 random runs are reported. The shortcut for this problem is to ignore the *i.v.*, but we do not let the estimators take it and force them to recover from the sampling bias.

Figure 8.2a shows the results on the uneven sets. As expected, the joint divergences are corrupted by the sampling bias and are far from the truth. The three CDs all converge to the true value. Figure 8.2b shows the results on the partial sets. The joint divergence diverges in this case. CD-P and CD-U are closer but not converging to the correct value, and the reason is that the non-overlapping supports violate the assumptions made by them. CD-PQ successfully achieved the true value. This shows the advantage of only measuring CD within the overlapping region in this example. Overall, the CDs are effective against sampling bias and the estimators converge to the true values.



(a) Uneven

(b) Partial

Figure 8.2: Estimated divergences on the synthetic data.

Handling Point Sets

Here we test the estimators using a large number of point sets. The full data of two classes are shown in Figure 8.4a. To create partial sets, we use a sliding window, whose width is half of the data's span, to scan the full data and at each position put the points within the window together as a set. The uneven sets are then created by combining the partial sets with a small number of random samples from the original data. 100 sets are created for each class and each set contains $200 - 300$ points.

This data set is more challenging: the marginal distribution of *d.v.* cannot differentiate the two classes; the conditional distributions $f(y|x)$ are dependent on x; near the center of the data the conditional distributions of the two classes are very close. The different divergence matrices on the uneven sets are shown in Figure 8.3, in which we sorted the sets according to their classes and window positions to show the structures. We see that the joint divergence is severely affected by the sampling bias, while the CDs are quite insensitive. The result of CD-U is especially impressive: the similarity structure of the original data is perfectly recovered. Figure 8.4 shows the results on the partial sets. The joint divergence is now dominated by the sampling bias. CDs again are able to recover from this severe disruption and achieve reasonable results. The result of CD-PQ is the cleanest on this data set.

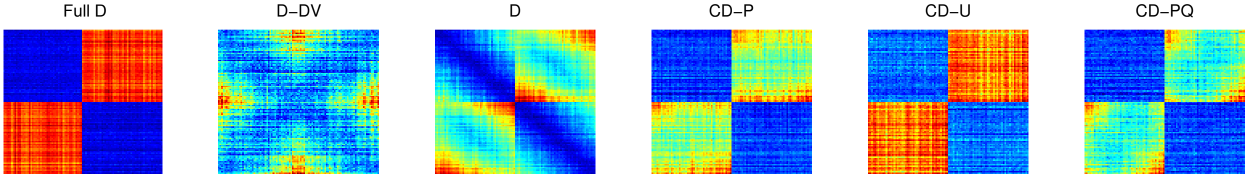
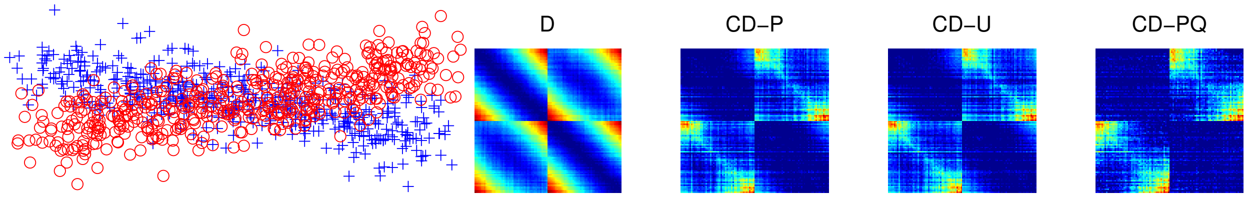


Figure 8.3: Divergences on the uneven sets. The goal is to recover the "Full D" given only the biased sets.



(a) Original data.

(b) Divergences

Figure 8.4: Divergences on the partial sets. The goal is to recover the "Full D" result shown in Figure 8.3.

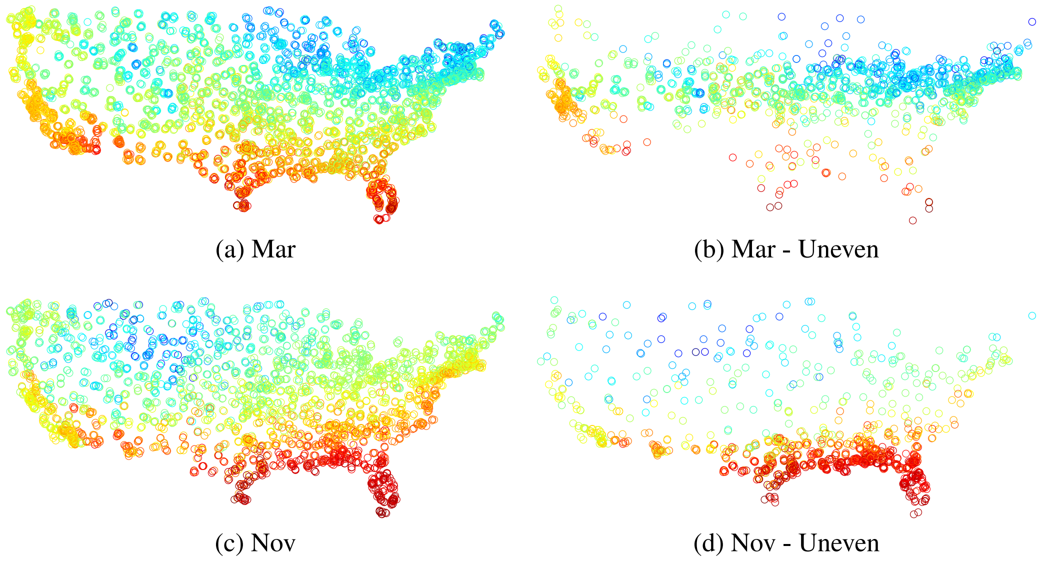
8.6.2 Season Classification

In this section we use the divergences in SVM to classify real world point sets generated by sensor networks. We gathered the data from the QCLCD climate database at NCDC [1]. We use a subset of QCLCD that contains daily climatological data from May 2007 to May 2013 measured by $1,164$ weather stations in the continental U.S. Each of these weather station produces various measurements such as the temperature, humidity, precipitation, *etc*, at its location. We aggregate these data into point sets, so that each set contains the measurements from all stations in one week.

We consider the problem of predicting the season of a set based on the average temperature measurement. Specifically, we want to know if a set corresponds to Spring or Fall based on the average temperatures over the U.S. Note that classifying Summer and Winter would be too easy, while differentiating Spring and Fall can be challenging since they have similar average temperatures. Nevertheless, it is still possible based on the geographical distribution of the temperatures. Figure 8.5 shows the temperature maps in a first week of March and a first week of November.

[1] http://www.ncdc.noaa.gov

Again, we create uneven and partial sets based on the original data by randomly positioning a full-width window whose height is 20% of the data's vertical span, as shown in Figure 8.5. This injection of sampling bias is simulating the scenario where we only have a sensoring satellite orbiting parallel to the equator. In this problem, the location is the *i.v.* and the temperature is the *d.v.*. This procedure gives us 160 3-dimensional (latitude, longitude, temperature) point sets with sizes around 2, 000.


(a) Mar

(b) Mar - Uneven

(c) Nov

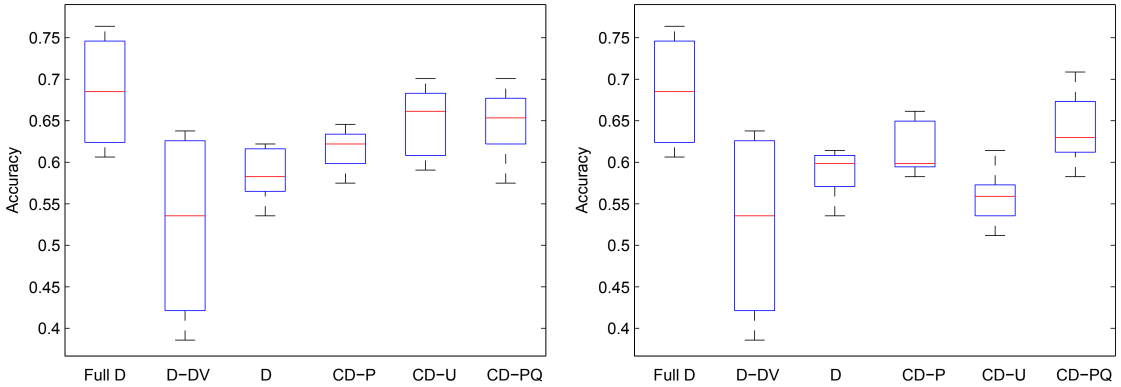
(d) Nov - Uneven

Figure 8.5: Example temperature maps of the U.S. from the QCLCD. (a) and (c) are the original data. (b) and (d) are the artificially created uneven data.

In each run, 20% of the random point sets are used for training and the rest are used for testing. Classification results of 10 runs are reported in Figure 8.6. On the uneven sets, we see that both CD-U and CD-PQ are able to recover from the sampling bias and achieve results that are only 3% worse than the full divergence. On the partial sets, however, the performance CD-U dropped significantly. This indicates that it can be risky to apply CD in regions where two sets do not overlap. It is interesting to see that D-DV, which ignores the locations, barely does better than random since Spring and Fall indeed have similar temperatures. Yet by considering the geographical distribution of temperatures we can achieve 70% accuracy.

8.6.3 Image Classification

We can also use CDs to classify scene images. We construct one point set for each image, where each point describes one patch including its location (*i.v.*) and the feature (*d.v.*). The OT [5] scene images are used, which contain 2, 688 grayscale images of size 256 × 256 from 8 categories. The patches are sampled densely on a grid and multiscale SIFT features are extracted using VLFeat [164]. The points are reduced to 20-dimensions using PCA, preserving 70% of variance.



(a) QCLCD, uneven.

(b) QCLCD, partial.

Figure 8.6: Season classification results on the QCLCD weather data.

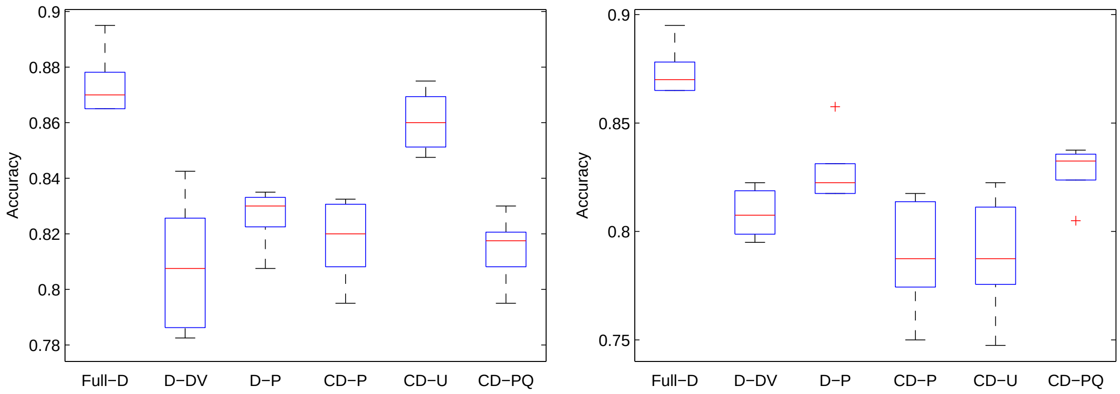
Again, we create both uneven and partial point sets by randomly positioning a full-width window whose height is 60% of the image. By doing this, the injected sampling bias forces a set to focus on a specific horizontal part of the scene. For instance in a beach scene, the biased observer focuses either on the sky or the sand, and only see a small part of the rest of the scene. After the above processing, the full data set contains $2,688$ sets of 20-dimensional points, and the sets' sizes are around $1,600$. In the biased data, each partial set has about 950 points and each uneven set has about $1,100$. In each run, we randomly select 50 images per class for training and another 50 for testing.

Results of 10 random runs are shown in Figure 8.7. In these results, CDs again successfully restore the accuracies to a high level even in the face of harsh sampling biases. We see that CD-U impressively beats the other methods by a large margin on the uneven sets, and is only 1% worse than the full divergence. CD-PQ is the best on partial sets. These results show the CDs' corrective power when the correct assumptions are made about the sampling biases.

We also observe that CD-U and CD-P did not perform well on the partial sets, which is expected since their assumptions were invalid on the data. In general, the impact of sampling bias on this data set is small (less than 10% decrease in accuracies) because the patch features (*d.v.*) only weakly depend on the patch locations (*i.v.*). In fact, many patch-based image analyses such as [50] do not include the locations. This might explain why both D-DV and D-P did reasonably well in this task and the corrected results by CD-PQ are only slightly better.

8.7 Summary

In this paper we described various aspects of dealing with sampling bias when learning from point sets. We proposed the conditional divergence (CD) to measure the difference between point sets and alleviate the impact of sampling bias. An efficient and convergent estimator of CD was provided. We then discussed how to deal with various types of sampling biases using CD. In the experiments we show that these methods are effective against sampling bias on both synthetic and



(a) Image, uneven.

(b) Image, partial.

Figure 8.7: Image classification results on OT.

real data.

Several directions can be explored in the future. We can extend the definition of conditional divergence from KL divergence to the more general Rényi divergences. The generalized conditional divergences provide the possibility of learning the weights of the density ratios in a supervised ways in order to maximize the discriminative power of the resulting divergences. The distance between points used in estimating the CDs could also be learned. Finally for extreme cases that cause missing divergences, we may infer them by exploiting the relationships among the sets using low-rank matrix completion techniques described in Chapter 6.

Chapter 9

Conclusion and Future Directions

Many real world problems generate a large amount collective data that are organized by groups. To effectively learn from them we need new tools from machine learning. How and what we can learn from collective data? In this thesis, we describe the research we have conducted in answering this question.

We considered different types of collective data and different ways to approach them. The first type consists of groups of discrete points. These groups can naturally be reduced to vectors and we proposed two novel factorization models to learn from them. The first model, called *Bayesian probabilistic tensor factorizations*, is able to capture the temporal dynamics of the data, and uses Bayesian techniques to avoid overfitting and parameter tuning. The second model *direct robust matrix factorization* addresses the outliers in the data, and seek to find robust factors/subspaces as well as identify the outliers. Both of these methods are simple and efficient for practical usages.

The main focus of this thesis, however, is on the more commonly encountered collective data: groups of real-valued multidimensional points. We developed both generative and discriminative methods to learn from them. From the generative perspective, we can first learn the generating process of the data and then use it to accomplish various learning tasks. Motivated by the group anomaly problem, and facilitated by the topic modeling techniques, we developed two flexible *genre models* to characterize how a collective data set was generated. We further designed several scoring functions based on these models to find different types of group anomalies.

We also took the kernel approach to learn from collective data discriminatively. Thanks to the newly proposed non-parametric divergence estimators, we can derive a new class of consistent and efficient *kernel estimators for collective data*. These kernels achieved the state-of-the-art performances in image classification task. Further efforts were made to study different ways of constructing Mercer kernels from the raw divergences in order to exploit the information in the divergence matrices.

We then addressed several practical problems in the kernel methods. The kernel estimators in this work, though relatively efficient, are still slow in practice. In order to accelerate, we studied different ways of reducing sizes of the groups, and discovered that k-Means was able to *condense* the information in the original groups into much smaller ones. As a result, the computation can be orders-of-magnitude faster and the learning performance can be preserved. The second practical issue we considered was the sampling bias. In the presence of sampling bias, the observed groups of points are not representative of the groups' underlying attributes. To solve this problem, we improved the traditional divergences and proposed the *conditional divergences*, for which an efficient estimator was also developed. Under certain assumptions, conditional divergences are insensitive to common sampling biases in data.

The methods we proposed are widely applicable in many real-world problems. In this thesis our attention is paid primarily to the scientific discovery process. We developed automatic discovery and learning systems for data sets from astronomy and physics based on the research in this thesis. This system facilitates the collaboration between us and the scientists by presenting the learning results to and collecting feedbacks from the experts. We believe this is the first step in building more powerful systems in the future.

The majority of the research in this thesis depends on the assumption that the points in a group are either *exchangeable* or *i.i.d.* samples from the underlying distribution. As of now this assumption has been prevailing in various areas including text modeling and computer vision, and has achieved great successes. Nevertheless, in the future we would like study the particularly interesting case of structured groups, in which points dependent on other points. For example, we can consider the Markovian dependencies between words in the same document, or between patches in the same image. With these additional characterizations of data, better results on learning from collective data can be expected.

There are a lot of other problems that remain to be studied based on this work. For example, considering that real-world data sets almost always contains outliers, we wish to make our methods robust so that the results are more reliable. This requires the development of robust topic models and kernel estimators. We also want to expanded the research to situations where the points are functions. This is quite common in astronomy where a spectrum is considered as a noisy observations of the object's underlying characteristic spectral function within a certain wavelength range. Finally, continuous effort has to be made in improving the algorithms' speed to gain actual practicality.

We believe that our current work is just a beginning and much remains to be done in the future. Learning from collective data directly has been a less active topic in machine learning probably because it requires a large amount of computational resources and the mathematical representations of the problems are less concise and elegant than the point-wise learning. However, the vast advancements of computer hardware and parallel computing tools have largely cleared the obstacles and it is interesting and useful and further explore this realm.

Bibliography

[1] Dimitris Achlioptas, Frank Mcsherry, and Bernhard Schölkopf. Sampling techniques for kernel methods. In *Advances in Neural Information Processing Systems (NIPS)*, 2002. 6.2

[2] Amr Ahmed and Eric P. Xing. Dynamic non-parametric mixture models and the recurrent chinese restaurant process. In *Proceedings of SDM 2008*, 2008. 2.5

[3] Hirotugu Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 1974. 4.4.1

[4] Martin S. Andersen and Lieven Vandenberghe. Support vector machine training using matrix completion techniques. Technical report, University of California, Los Angeles, 2010. 6.2

[5] A.Oliva and A. Torralba. Modmodel the shape of the scene: a holistic representation of spatial envelope. *International Journal of Computer Vision (IJCV)*, 42, 2001. 7.5.1, 8.6.3

[6] Francis R. Bach. Graph kernels between point clouds. In *ICML*, 2008. 1.4

[7] Francis R. Bach and Michael I. Jordan. Predictive low-rank decomposition for kernel methods. In *International Conference on Machine Learing (ICML)*, 2005. 6.2

[8] Y. Bai, L. Guo, L. Jin, and Q. Huang. A novel feature extraction method using pyramid histogram of orientation gradients for smile recognition. In *ICIP*, 2009. 5.1

[9] Robert Bell, Yehuda Koren, and Chris Volinsky. The bellkor 2008 solution to the netflix prize, 2008. Available at www.research.att.com/˜volinsky/netflix/. 2.2

[10] Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA 02178-9998, second edition, 1999. 2.4.2

[11] Jacob Bien and Robert Tibshirani. Prototype selection for interpretable classification. *The Annals of Applied Statistics*, 2011. 7.4.2

[12] M.B. Blaschko and T. Hofmann. Conformal multi-instance kernels. In *NIPS Workshop on Learning to Compare Examples*, 2006. 1.4

[13] David M. Blei and John D. Lafferty. Correlated topic models. In *NIPS*, 2006. 1.3.3, 4.3

[14] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *JMLR*, 3: 993–1022, 2003. 1.3.2, 2.5, 4.1, 4.2, 4.2

[15] P. Bloomfield and W. L. Steiger. *Least Absolute Deviations: Theory, Applications, and Algorithms (Progress in Probability)*. Birkh´auser Boston, Mass, USA, 1983. 1.2.3, 3.1,

3.1.1

[16] Liefeng Bo and Cristian Sminchisescu. Efficient matching kernels between sets of features for visual recognition. In *Neural Information Processing Systems (NIPS)*, 2009. 7.2.1

[17] Oren Boiman, Eli Shechtman, and Michal Irani. In defense of nearest neighbor based image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 7.1, 7.2, 7.2.1, 7.2.1, 7.2.2, 7.2.2, 7.5, 8.1, 8.2

[18] A. Bosch and and X. Munoz A. Zisserman. Image classifcation using random forests and ferns. In *International Conference on Computer Vision*, 2007. 7.5

[19] Anna Bosch, Andrew Zisserman, and Xavier Munoz. Scene classification using a hybrid generative/discriminative approach. *IEEE Trans. PAMI*, 30(4), 2008. 1.2, 5.6, 5.6.3

[20] George Box, Gwilym M. Jenkins, and Gregory C. Reinsel. *Time Series Analysis: Forecasting and Control.* Pretice-Hall, 1994. 2.1

[21] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and J´org Sander. Lof: Identifying density-based local outliers. In *ACM SIGMOD Record*, 2000. 3.1, 4.3

[22] E. Candes, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *preprint*, 2009. 1.2.3, 3.1, 3.1.1, 3.3, 3.3, 3.5, 3.5.1, 3.5.1

[23] Emmanuel J. Candès and Terence Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Trans. Information Theory*, 56(5):2053–2080, 2009. 1.2.1, 3.1.1, 3.3

[24] Gilles Celeux, Didier Chaveau, and Jean Diebolt. Stochastic version of the em algorithm: An experimental study in the mixture case. *J. of Statistical Computation and Simulation*, 55, 1996. 4.5.1

[25] Philip K. Chan and Matthew V. Mahoney. Modeling multiple time series for anomaly detection. In *IEEE International Conference on Data Mining*, 2005. 4.1, 4.3

[26] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):1–72, 2009. 1.2.3, 3.1, 4.3

[27] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. 5.6

[28] Gang Chen, Fei Wang, and Changshui Zhang. Collaborative filtering using orthogonal nonnegative matrix tri-factorization. *Information Processing & Management*, 42:2863–2875, 2009. 2.5

[29] Yutian Chen, Max Welling, and Alex J. Smola. Super-samples from kernel herding. In *Uncertainty in Artificial Intelligence (UAI)*, 2010. 7.3

[30] Yun Chi, Shenghuo Zhu Yihong Gong, and Yi Zhang. Probabilistic polyadic factorization and its application to personalized recommendation. In *CIKM*, 2008. 2.5

[31] Ma Chih-Chao. Large-scale collaborative filtering algorithms. Master's thesis, National Taiwan University, 2008. 2.6.2

[32] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. PAMI*, 24, 2002. 7.4.2

[33] ANDREW J. CONNOLLY and Alex S. SZALAY. A robust classification of galaxy spectra: Dealing with noisy and incomplete data. *THE ASTRONOMICAL JOURNAL*, 117:2052 – 2062, 1999. 3.7

[34] Corinna Cortes, Mehryar Mohri, Michael Riley, and Afshin Rostamizadeh. Sample selection bias correction theory. In *Algorithmic Learning Theory*, 2008. 8.2

[35] Trevor F. Cox and M. A. A. Cox. *Mulitidimensional Scaling*. Chapman and Hall, 2000. 6.2, 6.4

[36] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages 886–893, 2005. 5.1

[37] Scott F. Daniel, Andrew Connolly, Jeff Schneider, Jake VanderPlas, and Laing Xiong. Classification of stellar spectra with local linear embedding. *Astronomical Journal*, 142:203, 2011. 3.7

[38] Kaustav Das, Jeff Schneider, and Daniel Neill. Anomaly pattern detection in categorical datasets. In *Knowledge Discovery and Data Mining (KDD)*, 2008. 4.1, 4.3

[39] Kaustav Das, Jeff Schneider, and Daniel Neill. Detecting anomalous groups in categorical datasets. Technical Report 09-104, CMU-ML, 2009. 4.1, 4.3

[40] B. de Finetti. Funzione caratteristica di un fenomeno aleatorio. *Atti della R. Academia Nazionale dei Lincei, Serie 6. Memorie, Classe di Scienze Fisiche, Mathematice e Naturale*, 4:251–299, 1931. 4.1

[41] J. Deng, W. Dong, R. Socher, Li-Jia Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 7.5.5

[42] Frd́éric Desobry, Manuel Davy, and William J. Fitzgerald. A class of kernels for sets of vectors. In *Proceedings of the 13th European Symposium on Artificial Neural Networks*, 2005. 1.3.4

[43] Chris Ding, Tao Li, and Michael I. Jordan. Convex and semi-nonnegative matrix factorizations. *IEEE T-PAMI*, 32(1):45–55, 2010. 1.2.1, 3.1.1, 3.4.1

[44] David L. Donoho. *Breakdown Properties of Multivariate Location Estimators*. PhD thesis, Harvard University, 1982. 3.3

[45] Gabriel Doyle and Charles Elkan. Accounting for burstiness in topic models. In *International Conference on Machine Learning*, 2009. 1.3.3, 4.3

[46] P. Drineas, A. Javed, M. Magdon-Ismail, G. Pandurangan, R. Virrankoski, and A. Savvides. Distance matrix reconstruction from incomplete distance information for sensor network localization. In *Sensor and Ad Hoc Communications and Networks*, 2006. 2

[47] Petros Drineas and Michael W. Mahoney. On the nyström method for apprxoimating a gram matrix for improved kernel-based learning. *journal of Machine Learning Research (JMLR)*,

6:2153–2175, 2005. 6.2

[48] C. Eckart and G Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1:211–218, 1936. 3.1.1

[49] Charles Elkan. Using the triangle inequality to accelerate k-means. In *International Conference on Machine Learning (ICML)*, 2003. 7.4.3, 7.6

[50] Li Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. *IEEE Conf. CVPR*, pages 524–531, 2005. 1.2, 1.3.3, 1.3.5, 4.3, 4.7, 4.8.2, 5.1, 5.6.3, 7.1, 8.6.3

[51] Shai Fine and Katya Scheinberg. Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research (JMLR)*, 2:243–264, 2001. 6.2

[52] Arvind Ganesh, Zhouchen Lin, John Wright, Leqin Wu, Minming Chen, and Yi Ma. Fast algorithms for recovering a corrupted low-rank matrix. In *International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, 2009. 3.5

[53] Salvador García, Joaquín Derrac, José Ramón Cano, and Francisco Herrera. Prototype selection for nearest neighbor classification: Taxonomy nd empirical study. *IEEE Trans. PAMI*, 34, 2012. 7.3

[54] T. Gärtner, P. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In *CoLT*, 2003. 1.4

[55] Thomas Gärtner, Peter A. Flach, Adam Kowalczyk, and Alex J. Smola. Multi-instance kernels. In *ICML*, 2002. 1.4, 7.2, 7.2.1

[56] Alan E. Gelfand and Adrian F. M. Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85(410):398–409, 1990. 2.4.2

[57] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 2003. 4.5, 4.5.1

[58] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. PAMI*, 6:721 – 741, 1984. 4.1, 4.5.1

[59] Allen Gersho and Robert M. Gray. *Vector Quantization and Signal Compression*. Springer, 1991. 1.2

[60] Lise Getoor and Ben Taskar, editors. *Introduction to statistical relational learning*. MIT Press, 2007. 1.4

[61] M. Goria, N. Leonenko, V. Mergel, and N. Inverardi. A new class of random vector entropy estimators and its applications in testing statistical hypotheses. *Journal of Nonparametric Statistics*, 17:277–297, 2005. 5.4

[62] Thore Graepel. Kernel matrix completion by semidefinite programming. In *International Conference on Neural Networks*, 2002. 6.2

[63] Kristen Grauman and Trevor Darrell. The pyramid matching kernel: Discriminative classification with sets of image features. In *Internation Conference on Computer Vision (ICCV)*,

2005. 1.3.4, 5.1, 5.2, 5.6.2, 7.2.1

[64] Kristen Grauman and Trevor Darrell. Approximate correspondences in high dimensions. In *NIPS*, 2006. 1.3.4, 5.6

[65] Arthur Gretton, Karsten M. Borgwardt, Malte Rasch, Bernard Schölkopf, and Alex J. Smola. A kernel method for the two sample problem. In *Neural Information Processing Systems (NIPS)*, 2007. 1.3.4, 7.1, 7.2, 7.2.1, 7.2.1, 8.1, 8.2

[66] W. Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57:97–109, 1970. 2.4.2

[67] Douglas M Hawkins, Li Liu, and S. Stanley Young. Robust singular value decomposition. Technical report, National Institute of Statistical Sciences, 2001. 1.2.3, 3.3

[68] Geoffrey G. Hazel. Multivariate gaussian MRF for multispectral scene segmentation and anomaly detection. *IEEE Trans. Geoscience and Remote Sensing*, 38-3:1199 – 1211, 2000. 4.1, 4.3

[69] Matthias Hein and Olivier Bousquet. Hilbertian metrics and positive definite kernels on probability measures. In *AI and Statistics (AISTATS)*, 2005. 5.5

[70] Matthias Hein, Thomas Navin Lal, and Olivier Bousquet. Hilbertian metrics on probability measures and their application in svms. In *Proceedings of the 26th DAGM Symposium*, 2004. 5.5

[71] Nicholas J. Higham. Computing the Nearest Correlation Matrix a Problem From Finance. *IMA Journal of Nummerical Analysis*, pages 329–343, 2002. 5.5

[72] Thomas Hofmann. Unsupervised learning with probabilistic latent semantic analysis. *Machine Learning Journal*, 2001. 1.2.1, 1.3.2, 2.5, 3.1.1, 4.1, 5.6

[73] Jiayuan Huang, Alexander J. Smola, Arthur Gretton, Karsten M. Borgwardt, and Bernhard Schölkopf. Correcting sample selection bias by unlabeled data. In *NIPS*, 2007. 8.1, 8.2

[74] Huber and Peter J. Robust estimation of a location parameter. *Annals of Statistics*, 53: 73–101, 1964. 1.2.3, 3.1

[75] Tommi Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In *NIPS*, 1998. 1.3.4

[76] T. Jebara, R. Kondor, A. Howard, K. Bennett, and N. Cesa-bianchi. Probability product kernels. *Journal of Machine Learning Research*, 5:819–844, 2004. 1.3.4, 5.2, 5.6, 8.2

[77] Michael I. Jordan, editor. *Learning in Graphical Models*. MIT Press, Cambridge, MA, 1999. 4.4

[78] Frederic Jurie and Bill Triggs. Creating efficient codebooks for visual recognition. In *International Conference on Computer Vision (ICCV)*, 2005. 7.4.2

[79] Qifa Ke and Takeo Kanade. Robust l1 norm factorization in the presence of outliers and missing data by alternative convex programming. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 1.2.3, 3.3

[80] Mikaela Keller and Samy Bengio. Theme-topic mixture model for document representation.

In *Learning Methods for Text Understanding and Mining*, 2004. 1.3.3, 4.3

[81] Eamonn Keogh, Jessica Lin, and Ada Fu. Hot sax: Efficiently finding the most unusual time series subsequence. In *IEEE International Conference on Data Mining*, 2005. 4.1, 4.3

[82] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3), September 2009 (to appear). 2.3

[83] R. Kondor and T. Jebara. A kernel between sets of vectors. In *ICML*, 2003. 5.2

[84] Yehuda Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008. 2.2

[85] Yehuda Koren. Collaborative filtering with temporal dynamics. In *KDD-09*, 2009. 1.2.1, 2.1, 2.5

[86] Hans-Peter Kriegel, Peer Kroger, Erich Schubert, and Arthur Zimek. A general framework for increasing the robustness of pca-based correlation clustering algorithms. In *Scientific and Statistical Database Management Conference (SSDBM)*, 2008. 1.2.3, 3.1, 3.1.1, 3.3

[87] Brain Kulis, Mátyás Sustik, and Inderjit Dhillon. Learning low-rank kernel matrices. In *International Conference on Machine Learing (ICML)*, 2006. 6.2

[88] Fernando De la Torre and Michael J. Black. A framework for robust subspace learning. *International Journal of Computer Vision*, 54:117–142, 2003. 1.2.3, 3.1, 3.3

[89] Nan Laird. Nonparametric maximum likelihood estimation of a mixing distribution. *Journal of American Statistical Association*, 73:805–811, 1978. 4.6

[90] Nan Laird. Empirical bayes estimates using the nonparametric maximum likelihood estimate for the prior. *JOURNAL OF STATISTICAL COMPUTATION AND SIMULATION*, 55: 211–220, 1982. 4.6

[91] R. M. Larsen. Propack - software for large and sparse svd calculations, 2001. URL `http://soi.stanford.edu/~rmunk/PROPACK`. 3.2.1, 3.5

[92] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006. 7.5.1

[93] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999. 3.1

[94] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng. Efficient sparse coding algorithms. In *NIPS*, 2006. 1.2

[95] Bastian Leibe and Bernt Schiele. Analyzing appearance and contour based methods for object categorization. In *CVPR*, 2003. 5.6.2, 6.6.2

[96] Nikolai Leonenko, Luc Pronzato, and Vippal Savani. A class of Rényi information estimators for multidimensional densities. *Annals of Statistics*, 36(5):2153–2182, 2008. 5.4

[97] Thomas Leung and Jitendra Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *IJCV*, 43:29–44, 2001. 5.1

[98] Fei-Fei Li, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Workshop on Generative-Model based Vision*, 2004. 7.5.3

[99] Guoying Li and Zhonglian Chen. Projection-pursuit approach to robust dispersion matrices and principal components: Primary theory and monte carlo. *J. of American Statistical Association*, 80(391):759 – 766, 1985. 1.2.3, 3.3

[100] Li-Jia Li and Fei-Fei Li. What, where and who? classifying events by scene and object recognition. In *International Conference on Computer Vision*, 2007. 5.6.4, 5.6.4, 6.6.2, 7.5.2

[101] Liyuan Li, Weimin Huang, Irene Yu-Hua Gu, and Qi Tian. Statistical modeling of complex backgrounds. *IEEE Trans. Image Processing*, 13(11):1459–1472, 2004. 3.5.2

[102] Wei Li and Andrew McCallum. Pachinko allocation: Dag-structured mixture models of topic correlations. In *ICML*, 2006. 1.3.3, 4.3

[103] Jeff Schneider Liang Xiong, Xi Chen. Direct robust matrix factorization for anomaly detection. In *IEEE International Conference on Data Mining (ICDM)*, 2011. 1

[104] Jun Liu, Shuiwang Ji, and Jieping Ye. Multi-task feature learning via efficient l2,1-norm minimization. In *The Twenty-fifth Conference on Uncertainty in Artificial Intelligence (UAI)*, 2009. 3.1.1

[105] D. O. Loftsgaarden and C. P. Quesenberry. A nonparametric estimate of a multivariate density function. *The Annals of Mathematical Statistics*, 36(3), 1965. 8.3.1, 8.5

[106] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91 – 110, 2004. 4.8.2, 5.1

[107] Zhaosong Lu and Yong Zhang. Penalty decomposition methods for $l0$-norm minimization. Technical report, Department of Mathematics, Simon Fraser University, 2010. 3.2.1, 3.2.1

[108] R. Luss and A. d'Aspremont. Support vector machine classification with indefinite kernels. *Mathematical Programming Computation*, 1(2-3):97–118, 2009. 5.5

[109] Siwei Lyu. Mercer kernels for object recognition with local features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 5.6, 7.2, 7.2.1

[110] Shiqian Ma, Donald Goldfarb, and Lifeng Chen. Fixed point and bregman iterative methods for matrix rank minimization. *Math. Program., Ser. A*, to appear. 3.5.1

[111] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research*, 2009. 1.2.1, 3.1.1, 3.3

[112] Sancho McCann and David G. Lowe. Local naive bayes nearest neighbor for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 7.1, 7.2.2, 7.5, 8.1, 8.2

[113] Rosalie C. McGurk, Amy E. Kimball, and Zeljko Ivezic. Principal component analysis of

sloan digital sky survey stellar spectra. *The Astronomical Journal*, 139:1261, 2010. 3.7

[114] Geoffrey J. Mclachlan and Thriyambakam Krishnan. *The EM Algorithm and Extensions*. John Wiley and Sons, 1996. 4.4.1

[115] Charles Meneveau. Lagrangian dynamics and models of the velocity gradient tensor in turbulent flows. *Annual Review of Fluid Mechanics*, 43:219–45, 2011. 4.8.3

[116] Nicholas Metropolis and S. Ulam. The monte carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949. 2.4.2

[117] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1091, 1953. 2.4.2

[118] Thomas P. Minka. Estimating a dirichlet distribution. `http://research.microsoft.com/en-us/um/people/minka/papers/dirichlet`, 2009. 4.5.1

[119] Pedro J. Moreno, Purdy P. Ho, and Nuno Vasconcelos. A kullback-leibler divergence based kernel for svm classification in multimedia applications. Technical report, HP Lab Cambridge, 2004. 1.3.4, 5.2, 5.6

[120] Krikamol Muandet, Kenji Fukumizu, Francesco Dinuzzo, and Bernhard Schölkopf. Learning from distributions via support measure machines. In *Neural Information Processing Systems (NIPS)*, 2013. 7.2.1, 7.2.2, 8.1, 8.2

[121] Marius Muja and David G. Lowe. Fast approximate nearest nneighbor with automatic algorithms configuration. In *International Conference on Computer Vision THeory and Applications (VISAPP)*, 2009. 5.4, 7.2.3, 7.5, 8.3.1

[122] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2001. 1, 5.3, 8.2

[123] Minh Hoai Nguyen and Fernando De la Torre. Robust kernel principal components analysis. In *NIPS*, 2009. 1.2.3, 3.1

[124] Nam H. Nguyen, Thong T. Do, and Trac D. Tran. A fast and efficient algorithm for low-rank approximation of a matrix. In *STOC*, 2009. 3.4.1

[125] X. Nguyen, M.J. Wainwright, and M.I. Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory, To appear.*, 2010. 5.2

[126] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3), 2001. 4.8.2, 5.6.3, 6.6.2

[127] E. Perlman, R. Burns, Y. Li, and C. Meneveau. Data exploration of turbulence simulations using a database cluster. In *Supercomputing SC*, 2007. 4.8.3

[128] Barnabás Póczos. Nonparametric estimation of conditional information and divergences. In *AI and Statistics (AISTATS)*, 2012. 8.2

[129] Barnabás Póczos and Jeff Schneider. On the estimation of alpha divergence. In *AI and*

Statistics (AISTATS), 2011. 1.3.4, 5.4, 8.3.1

[130] Barnabás Póczos, Liang Xiong, and Jeff Schneider. Nonparametric divergence estimation with applications to machine learning on distributions. In *Uncertainty in Artificial Intelligence (UAI)*, 2011. 1, 5.2, 5.4, 7.2, 7.2.1, 7.2.2, 8.1, 8.2, 8.5

[131] Barnabás Póczos, Liang Xiong, Dougal Sutherland, and Jeff Schneider. Nonparametric kernel estimators for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 1, 7.1, 7.2, 7.2.1, 7.2.2, 7.5, 8.1, 8.2, 8.5

[132] Ian Porteous, Evgeniy Bart, and Max Welling. Multi-hdp: A non-parametric bayesian model for tensor factorization. In *AAAI*, 2008. 2.5

[133] Foster Provost, D. Jensen, and T. Oates. Efficient progressive sampling. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, 2009. 7.4.1, 7.6

[134] Yuan Qi and Thomas P. Minka. Hessian-based markov chain monte-carlo algorithms. In *First Cape Cod Workshop on Monte Carlo Methods*, 2002. 2.7

[135] Jianzhao Qin and Nelson H.C. Yung. Scene categorization via contextual visual words. *Pattern Recognition*, 43, 2010. 5.6.3

[136] Guoping Qiu. ”indexing chromatic and achromatic patterns for content-based colour image retrieval. *Pattern Recognition*, 35, 2002. 1.2

[137] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. 4.9

[138] Jasson D. M. Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *ICML*, 2005. 1.2.1, 2.5, 3.1.1, 6.3

[139] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer-Verlag New York, LLC, 2nd edition, 2004. 2.4.2

[140] Yasushi Sakurai and Rosalynn Chong. Efficient distribution mining and classification. In *SIAM Data Mining*, 2008. 1.3.4

[141] Ruslan Salakhutdinov and Andriy Minh. Probabilistic matrix factorization. In *NIPS*, 2007. 1.2.1, 2.1, 2.2, 2.7, 3.1.1

[142] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *ICML*, 2008. 2.1, 2.4, 2.5, 2.6, 2.6.2

[143] Purnamrita Sarkar, Sajid M. Siddiqi, and Geoffrey J. Gordon. A latent space approach to dynamic embedding of co-occurrence data. In *AISTAT-07*, 2007. 2.5

[144] Mikkel N. Schmidt and Shakir Mohamed. Probabilistic non-negative tensor factorization using markov chain monte carlo. In *European Signal Processing Conference (EUSIPCO)*, 2009. 2.5

[145] I. J. Schoenberg. Metric spaces and positive definite functions. *Transactions of the American Mathematical Society*, 44:522–536, 1938. 5.5, 6.4, 1

[146] B. Schölkopf and A. Smola. *Learning with kernels : support vector machines, regularization, optimization, and beyond*. The MIT Press, 2002. 6.4

[147] Bernhard Scholkopf, Alexander Smola, and Klaus-Robert MÏler. Kernel principal component analysis. In *ADVANCES IN KERNEL METHODS - SUPPORT VECTOR LEARNING*, 1999. 5.3

[148] Bernhard Schölkopf, John C. Platt, John Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13:1443–1471, 2001. 5.3

[149] Gideon E. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 1974. 4.4.1, 4.5.1

[150] D. Scott. On optimal and data-based histograms. *Biometrika*, 66:605–610, 1979. 5.2

[151] Amnon Shashua and Tamir Hazan. Algebraic set kernels with application to inference over local image representations. In *NIPS*, 2004. 1.3.4

[152] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2), 2000. 8.1, 8.2

[153] Michael Shindler, Alex Wong, and Adam Meyerson. Fast and accurate k-means for large datasets. In *Neural Information Processing Systems (NIPS)*, 2011. 7.4.3, 7.6

[154] Chanop SilpaAnan and Richard Hartley. Optimized kd-trees for fast image descriptor matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008. 7.2.3

[155] Alex Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf. A hilbert space embedding for distributions. In *Algorithmic Learning Theory (ALT)*, 2007. 1.3.4, 5.2, 5.6, 7.2, 7.2.1

[156] Springer. *Pattern Recognition and Machine Learning*. Springer, 2007. 8.5

[157] K. Sricharan, R. Raich, and A. Hero. Empirical estimation of entropy functionals with confidence. Technical Report, `arxiv.org/abs/1012.4188`, 2010. 5.2

[158] Ben Taskar, Pieter Abbeel, and Daphne Koller. Discriminative probabilistic models for relational data. In *Uncertainty in Artificial Intelligence*, 2002. 1.4

[159] Michael E. Tipping and Bernhard Schölkopf. A kernel approach for vector quantization with guaranteed distortion bounds. In *AI and Statistics (AISTATS)*, 2001. 7.4.2

[160] Hanghang Tong, Spiros Papadimitriou, Philip s. Yu, and Christos Faloutsos. Proximity tracking on time-evolving bipartite graphs. In *SDM-08*, 2008. 2.5

[161] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the 21st International Conference on Machine Learning*, New York, NY, USA, 2004. ACM. ISBN 1-58113-838-5. 1.4

[162] Tinne Tuytelssars, Mario Fritz, Kate Saenko, and Trevor Darrell. The nbnn kernel. In *International Conference on Computer Vision (ICCV)*, 2011. 7.3, 7.4.1

[163] Jake VanderPlas and Andrew J. Connolly. Reducing the dimensionality of data: Locally linear embedding of sloan galaxy spectra. *Astronomical Journal*, 138:1365, 2009. 3.7

[164] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision

algorithms. http://www.vlfeat.org, 2008. 5.6, 7.5, 7.6, 8.6.3

[165] S.V.N. Vishwanathan, Karsten M. Borgwardt, Imre Risi Kondor, and Nicol N. Schraudolph. Graph kernels. *JMLR*, 2004. 1.4

[166] G. Mark Voit. Tracing cosmic evolution with clusters of galaxies. *Reviews of Modern Physics*, 77(1):207 – 258, 2005. 1.5, 4.1

[167] C. Wallraven, B. Caputo, and A. Graf. Recognition with local features: the kernel recipe. In *International Conference on Computer Vision (ICCV)*, 2003. 7.2.1

[168] Qing Wang, Sanjeev R. Kulkarni, and Sergio Verdú. Divergence estimation for multidimensional densities via k-nearest-neighbor distances. *IEEE Trans. on Information Theory*, 55, 2009. 5.4, 5.5, 7.2.1, 8.3.1, 3, 8.3.1, 8.6

[169] K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision (IJCV)*, 70:77–90, 2004. 6.2

[170] Lior Wolf and Amnon Shashu. Kernel principal angles for classification machines with applications to image sequence interpretation. Technical report, Hebrew University, School of CSE, 2002. 1.3.4

[171] Adam Wońica, Alexandros Kalousis, and Melanie Hilario. Distances and (indefinite) kernels for sets of objects. In *IEEE International Conference on Data Mining*, 2006. 1.3.4

[172] Liang Xiong, Xi Chen, Tzu kuo Huang, Jeff Schneider, and Jaime Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SIAM Data Mining*, 2010. 1

[173] Liang Xiong, Barnabás Póczos, and Jeff Schneider. Group anomaly detection using flexible genre models. In *Neural Information Processing Systems (NIPS)*, 2011. 1, 7.2

[174] Liang Xiong, Barnabás Póczos, and Jeff Schneider. Hierarchical probabilistic models for group anomaly detection. In *AI and Statistics (AISTATS)*, 2011. 1, 7.2

[175] Liang Xiong, Barnabás Póczos, and Jeff Schneider. Efficient learning from point sets. In *IEEE International Conference on Data Mining (ICDM)*, 2013. 1

[176] Huan Xu, Constantine Caramanis, and Shie Mannor. Principal component analysis with contaminated data: The high dimensional case. In *Annual Conference on Learning Theory (CoLT)*, 2010. 1.2.3, 3.1, 3.4.2

[177] Huan Xu, Constantine Caramanis, and Sujay Sanghavi. Robust pca via outlier pursuit. In *NIPS*, 2010. 1.2.3, 3.1, 3.1.1, 3.3, 3.3, 3.4.1, 3.5, 3.5.1, 3.5.3

[178] C. W. Yip, A. J. Connolly, A. S. Szalay, T. Budavari, M. SubbaRao, J. A. Frieman, R. C. Nichol, A. M. Hopkins, D. G. York, S. Okamura, J. Brinkmann, I. Csabai, A. R. Thakar, M. Fukugita, and Z. Ivezic. Distributions of galaxy spectral types in the sloan digital sky survey. *The Astronomical Journal*, 128:585, 2004. 3.7

[179] C. W. Yip, A. J. Connolly, D. E. Vanden Berk, Z. Ma, J. A. Frieman, M. SubbaRao, A. S. Szalay, G. T. Richards, P. B. Hall, D. P. Schneider, A. M. Hopkins, J. Trump, and J. Brinkmann. Spectral classification of quasars in the sloan digital sky survey: Eigenspec-

tra, redshift, and luminosity effects. *The Astronomical Journal*, 128:2603, 2004. 3.7

[180] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68:49–67, 2006. 3.1.1

[181] Chunjie Zhang, Jing Liu, Qi Tian, Changsheng Xu, Hanqing Lu, and Songde Ma. Image classification by non-negative sparse coding, low-rank and sparse decomposition. In *CVPR*, 2011. 1.2, 5.6.4

[182] Manqi Zhao and Venkatesh Saligrama. Anomaly detection with score functions based on nearest neighbor graphs. In *NIPS*, 2009. 3.1, 4.3, 4.8.2

[183] Tianyi Zhou and Dacheng Tao. Godec: Randomized low-rank & sparse matrix decomposition in noisy case. In *International Conference on Machine Learning*, 2011. 3.4.2

[184] Zhihua Zhou. Multi-instance learning: A survey. Technical report, Department of Computer Science & Technology, Nanjing University, 2004. 1.1, 1.4

[185] Zihan Zhou, Xiaodong Li, John Wright, Emmanuel Candès, and Yi Ma. Stable principal component pursuit. In *International Symposium on Information Theory*, 2010. 3.3, 3.3, 3.5, 3.5.1