SOME CONTRIBUTIONS TO SEMI-SUPERVISED LEARNING

By

Pavan Kumar Mallapragada

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Computer Science

2010

ABSTRACT

SOME CONTRIBUTIONS TO SEMI-SUPERVISED LEARNING

By

Pavan Kumar Mallapragada

Semi-supervised learning methods attempt to improve the performance of a supervised or an unsupervised learner in the presence of "side information". This side information can be in the form of unlabeled samples in the supervised case or pair-wise constraints in the unsupervised case. Most existing semi-supervised learning approaches design a new objective function, which in turn leads to a new algorithm rather than improving the performance of an already available learner. In this thesis, the three classical problems in pattern recognition and machine learning, namely, classification, clustering, and unsupervised feature selection, are extended to their semi-supervised counterparts.

Our first contribution is an algorithm that utilizes unlabeled data along with the labeled data while training classifiers. Unlike previous approaches that design specialized algorithms to effectively exploit the labeled and unlabeled data, we design a meta-semi-supervised learning algorithm called *SemiBoost*, which wraps around the underlying supervised algorithm and improve its performance using the unlabeled data and a similarity function. Empirical evaluation on several standard datasets shows a significant improvement in the performance of well-known classifiers (decision stump, decision tree, and SVM).

In the second part of this thesis, we address the problem of designing a mixture model for data clustering that can effectively utilize "side-information" in the form of pair-wise constraints. Popular mixture models or related algorithms (K-means, Gaussian mixture models) are too rigid (strong model assumptions) to result in different cluster partitions by utilizing the side-information. We propose a non-parametric mixture model for data clustering in order to be flexible enough to detect arbitrarily shaped clusters. Kernel density

estimates are used to fit the density of each cluster. The clustering algorithm was tested on a text clustering application, and performance superior to popular clustering algorithms was observed. Pair-wise constraints ("must-link" and "cannot-link" constraints) are used to select key parameters of the algorithm.

The third part of this thesis focuses on performing feature selection from unlabeled data using instance level pair-wise constraints (i.e., a pair of examples labeled as must-link pair or cannot-link pair). Using the dual-gradient descent method, we designed an efficient online algorithm. Pair-wise constraints are incorporated into the feature selection stage, providing the user with flexibility to use unsupervised or semi-supervised algorithms at a later stage. The approach was evaluated on the task of image clustering. Using pair-wise constraints, the number of features was reduced by around 80%, usually with little or no degradation in the clustering performance on all the datasets, and with substantial improvement in the clustering performance on some datasets.

To My Family

ACKNOWLEDGMENTS

I want to express my sincere gratitude to my thesis advisor, Prof. Anil Jain. He has been a wonderful mentor and motivator. Under his guidance, I have learned a lot about different aspects of conducting research, including finding a good research problem, writing a convincing technical paper, and prioritizing different tasks. I will always strive to achieve his level of discipline, scientific rigor, and attention to detail, at least asymptotically. I am also thankful to Prof. Rong Jin for working closely with me during the development of this thesis. From him, I have learned a lot about modeling techniques, how to formalize and solve a problem with the tools at hand, and how to be strong enough to handle adverse situations in research. I am thankful to the other members of my thesis committee, Prof. Lalita Udpa and Prof. William Punch. Their advice and suggestions have been very helpful. I am thankful to Martin (Hiu-Chung) Law for mentoring me through the first year of my studies and helping me make my first steps. I would like to thank Prof. Sarat Dass for the helpful discussions during the initial years of my Ph.D. program. Thanks to Prof. Jain for all the wonderful get-togethers and dinners at his home and in the lab. Special thanks to Prof. Stockman for all the Canoe trips he arranged, and also, for saving me and Biagio (with the help of Abhishek) from drowning in the freezing Grand River.

I am grateful to several other researchers who have mentored me during my various stages as a research student. I thank Dr. Jianchang Mao at Yahoo Labs for making the months I spent at Yahoo! Labs memorable and productive. The internships were a great learning experience. I would like to thank Dr. Sarangarajan Parthasarathy for guiding me through the internship at Yahoo. The experience was of immense help to me.

On a more personal side, I am grateful to all the friends I have made during the past few years. All my lab mates in the PRIP lab, including Abhishek, Soweon, Jung-Eun, Serhat, Unsang, Karthik, Meltem, Steve, Alessandra, Serhat, Kien, Jianjiang, Leo, Dirk, Miguel, Hong Chen, and Yi Chen. I thank Abhishek for all the helpful discussions we had when

I was stuck with some problem. I learned so much from all my colleagues in different aspects and had some of the best times in the lab. I will carry these experiences with me all through my life. I am grateful to Linda Moore, Cathy Davision, Norma Teague, and Debbie Kruch for their administrative support. I thank our technical editor Kim for her timely help. Many thanks to the CSE and HPCC admins.

I would like to thank Youtube, Google Video, Netflix and Hulu for keeping me entertained, albeit partially distracted at times. I would also like to thank Facebook and Orkut for helping me keep in touch with my friends within the comforts of the lab.

My special thanks to my dearest friends Mohar and Rajeev for all the countably infinite things they have helped me with. They have been family to me when I came to the United States. They have given me all that I needed to survive in the U.S., from furniture all the way to love, friendship, and affection. Much of the energy used to write this thesis was generated from Mohar's awesome and frequent dinners. Coming to music, one of the most important parts of my life: playing blues with Abhishek Bakshi; Gypsy Jazz with Chaitanya and Tim Williams; accompanying Mohar for Robindra Sangeet; Tabla jams with Anindya; live music with my super-talented singer friends – Mohar, Rajeev, Vidya Srinivasan, Srikanth, Bani, Abhishek Bakshi and Chandini; the Sargam practices; abstract jamming with Mayur; and discussing science and philosophy with Srikanth and Pradeep - are some of the most memorable parts of my personal life during my Ph.D. program. I would also like to thank Vidya, Abhisek Bakshi, Aparna Rajgopal, Ammar, Safa, Karthik, Aparna, Shaily, Saurabh (D), Tushar, Smiriti, Saurabh (S), and Raghav of the chummatp group, for providing me with great friendship during my internships at California, and the love and support they have shown to me during difficult times. I am indebted to Rupender and Suhasini for all the love and support they have provided me and my family during the thesis time.

Finally, without the nurturing, care and love, and above all, patience, from my mother and father, I could not have completed my doctoral degree. I am always deeply indebted to

them for all that they have given me. I thank my brother Bhaskar for the great support he was to me, and I am proud of him.

TABLE OF CONTENTS

LIST OF TABLES **x**

LIST OF FIGURES **xii**

1 Introduction **1**
- 1.1 Semi-supervised learning . . . 3
- 1.2 Thesis contributions . . . 5

2 Background **8**
- 2.1 Supervised learning . . . 8
- 2.2 Unsupervised learning . . . 10
- 2.3 Semi-supervised algorithms . . . 12
  - 2.3.1 Semi-supervised classification . . . 14
  - 2.3.2 Semi-supervised clustering . . . 22
- 2.4 Does side-information always help? . . . 27
  - 2.4.1 Theoretical observations . . . 28
- 2.5 Summary . . . 30

3 SemiBoost: Boosting for Semi-supervised Classification **32**
- 3.1 Introduction . . . 32
- 3.2 Related work . . . 35
- 3.3 Semi-supervised boosting . . . 38
  - 3.3.1 Semi-supervised improvement . . . 38
  - 3.3.2 SemiBoost . . . 39
  - 3.3.3 Algorithm . . . 42
  - 3.3.4 Implementation . . . 47
- 3.4 Results and discussion . . . 50
  - 3.4.1 Datasets . . . 50
  - 3.4.2 Experimental setup . . . 51
  - 3.4.3 Results . . . 52
  - 3.4.4 Convergence . . . 59
  - 3.4.5 Comparison with AdaBoost . . . 60
- 3.5 Performance on text-categorization . . . 61
- 3.6 Conclusions and future work . . . 64

4 Non-parametric Mixtures for Clustering **70**
- 4.1 Introduction . . . 70
- 4.2 Non-parametric mixture model . . . 74
  - 4.2.1 Model description . . . 74

4.2.2 Estimation of profile matrix Q by leave-one-out method 75
4.2.3 Optimization methodology 78
4.2.4 Implementation details 86
4.3 Results and discussion 87
4.3.1 Baseline methods: 87
4.3.2 Synthetic Datasets 88
4.3.3 UCI Datasets 88
4.3.4 Text Datasets 90
4.3.5 Discussion 92
4.3.6 Sensitivity to parameters: 92
4.4 Parameter selection 99
4.4.1 Maximum Pairwise Constraint Satisfaction Heuristic 99
4.5 Connection with K-means 100
4.5.1 Approximation to weighted K-means 108
4.6 Summary 109

5 Incremental Algorithm for Feature Selection 110
5.1 Introduction 110
5.2 Related work 113
5.2.1 Review of Online Learning 115
5.3 Problem formulation 119
5.4 Online algorithm using projections 121
5.4.1 Algorithm 121
5.4.2 Theoretical analysis 122
5.4.3 Implementation details 127
5.5 Experiments 128
5.5.1 Feature extraction 129
5.5.2 Results and discussion 131
5.6 Summary and conclusions 133

6 Summary and Conclusions 137
6.1 Contributions 137
6.2 Future Work 140
6.3 Conclusions 143

BIBLIOGRAPHY 145

LIST OF TABLES

1.1 Different kinds of semi-supervised settings considered in the literature. . . . 5

2.1 A comparison of different clustering algorithms proposed in the literature. Given the large number of available algorithms, only a few representative ones are shown here. 13

2.2 A summary of semi-supervised classification algorithms. T or I in the last column denotes Transductive or Inductive property of the algorithm, respectively. 16

2.3 Unsupervised learning algorithms and their corresponding semi-supervised counterparts. 20

3.1 Inductive performance of SemiBoost and the three benchmark algorithms. The first column shows the dataset and the two classes chosen. The number of samples n and the dimensionality d are shown below the name of each dataset. The algorithms chosen as base classifiers for boosting are Decision Stump (DS), Decision Tree (J48) and Support Vector Machine (SVM). For each algorithm, the SB- prefixed column indicates using the SemiBoost algorithm on the base classifier. The columns TSVM, ILDS and LapSVM show the inductive performance of the three benchmark algorithms. A '-' indicates that we could not finish running the algorithm in a reasonable time (20 hours) due to convergence issues. Each entry shows the mean classification accuracy and standard deviation (in parentheses) over 20 trials. 55

3.2 Performance of different classifiers and their boosted versions on 6 UCI datasets. X-small stands for the classifier trained on a set of 10 labeled samples chosen from the data. The prefix AB-X stands for AdaBoost with base classifier X. SB-X stands for SemiBoost with base classifier X. X-large stands for the classifier trained by labeling all the unlabeled data used in SB-X. 60

3.3 Comparison of the inductive performance (measured as % accuracy) of SemiBoost, TSVM, ILDS and LapSVM on pairwise binary problems created from 5 classes of the 20-newsgroups dataset. 62

4.1 Properties of different clustering methods. Prototype-based clustering methods are approaches that use a single data point to represent each of the clusters (e.g. K-means uses centroid of the cluster to represent each cluster) 73

4.2 Mean pairwise F_1 value of the performance of different clustering algorithms over 10 runs of each algorithm on 17 UCI datasets. The kernel width is chosen as the 5^{th} percentile of the pairwise Euclidean distances for Kernel based algorithms. The best performance for each dataset is shown in bold. The name of the dataset, number of samples (n), dimension (d), and the number of target clusters (G) are shown in the first 4 columns respectively. An entry of '-' indicates that the algorithm did not converge. The last column shows the best F_1 value achieved by Single (S), Complete (C) and Average (A) link algorithms. 89

4.3 Text datasets used in the evaluation. The datasets were composed of articles from three or four different newsgroups; the number of clusters (G) is assumed to be known. The number of samples and number of features is denoted by n and d respectively. 90

4.4 Mean pairwise F_1 value of the performance of different clustering algorithms over 10 runs of each algorithm on 9 high-dimensional text datasets. The kernel width is chosen as the 5^{th} percentile of the pairwise Euclidean distances for Kernel based algorithms. The best performance for each dataset is shown in bold. The name of the dataset, number of samples (n), dimension (d), and the number of target clusters (G) are shown in the first 4 columns, respectively. An entry of '-' indicates that the algorithm did not converge. The last column shows the best F_1 value achieved by Single (S), Complete (C) and Average (A) link algorithms. 91

4.5 Mean and standard deviation of the performance of Spectral Clustering vs. NMM approach. The bandwidth is selected using the maximum pairwise constraint satisfaction heuristic. Significant differences (paired t-test, 95% confidence) are shown in boldface. 101

5.1 Choice of potential and the resulting classical online learning algorithms . . 119

5.2 Performance of the proposed algorithm measured using pairwise-F1 measure. The first two columns show the target clusters, subsequent columns show the mean pairwise F_1 measure, expressed as percentage. Significant differences (paired t-test at 95% confidence) compared to the K-means algorithm are indicated by a $+$ or a $-$. 129

5.3 Mean and standard deviation of the number of visual words (from a total of 5,000) selected by the proposed LASSO and Group-LASSO method vs the L_2 DML algorithm. POLA is not a feature selection technique, and hence learns the weights for all the features. The batch mode forward search algorithm always selected 150 features, and hence is not reported in the table. The tasks are defined in Table 5.2 134

LIST OF FIGURES

Images in this dissertation are presented in color.

2.1 Utility of the unlabeled data in learning a classifier. (a) Classifier learned using labeled data alone. (b) Utility of unlabeled data. The filled dots show the unlabeled data. The gray region depicts the data distribution obtained from the unlabeled data. 15

2.2 Utility of pairwise constraints in data clustering. (a) Input unlabeled data to be clustered into two clusters. Figures (b) and (c) show two different clusterings of data in (a) obtained by using two different sets of pairwise constraints. 31

3.1 Block diagram of the proposed algorithm, SemiBoost. The inputs to SemiBoost are: labeled data, unlabeled data and the similarity matrix. 33

3.2 An outline of the SemiBoost algorithm for semi-supervised improvement. . 38

3.3 The SemiBoost algorithm 44

3.4 Decision boundary obtained by SemiBoost at iterations 1, 2, 3 and 12, on the two concentric rings dataset, using Decision Stump as the base classifier. There are 10 labeled samples per class (■,▲). The transductive performance (i.e., performance on the unlabeled data used for training) of SemiBoost is given at each iteration in parentheses. 48

3.5 Performance of baseline algorithm SVM with 10 labeled samples, with increasing number of unlabeled samples added to the labeled set (solid line), and with increasing number of labeled samples added to the training set (dashed line). 54

3.6 Performance of baseline algorithm SVM with 10 labeled samples, with increasing value of the parameter σ. The increments in σ are made by choosing the ρ-th percentile of the similarities, where ρ is represented on the horizontal axis. 56

3.7 The mean-margins over the iterations, on a single run of SemiBoost on optdigits dataset (classes 2,4), using Decision Stump as the base classifier. . 58

3.8 The mean-margins over the iterations, on a single run of SemiBoost on optdigits dataset (classes 2,4), using J48 as the base classifier. 58

3.9 The mean-margins over the iterations, on a single run of SemiBoost on optdigits dataset (classes 2,4), using SVM as the base classifier. 59

3.10 Distribution of the ensemble predictions $H_t(\mathbf{x}_i)$, over the unlabeled samples in the training data from optdigits dataset (classes 2,4) at the iteration t, where $t \in \{1, 2, 10, 20\}$. SVM is used as the base classifier. The light and dark bars in the histogram correspond to the two classes 2 and 4, respectively. 65

3.11 Continued from previous page. 66

3.12 Objective function of SemiBoost over the iterations, when run over two classes (2,4) of the optdigits dataset using Decision Stump as the base classifier. 67

3.13 The classifier combination weight α of SemiBoost over the iterations, when run over two classes (2,4) of the optdigits dataset using Decision Stump as the base classifier. 68

3.14 Accuracy of SemiBoost over the iterations, when run over two classes (2,4) of the optdigits dataset using Decision Stump as the base classifier. The accuracy of the base classifier is 65.9%. 69

4.1 Graphical model showing the data generation process using the NMM. . . . 77

4.2 Illustration of the non-parametric mixture approach and Gaussian mixture models on the "two-moon" dataset. (a) Input data with two clusters. (b) Gaussian mixture model with two components. (c) and (d) the iso-contour plots of non-parameteric estimates of the class conditional densities for each cluster. The warmer the color, the higher the probability. 84

4.3 Illustration of the (a) non-parametric mixture approach, (b) K-means and (c) spectral clustering on the example dataset from [1]. Input data contains 100 points each from three spherical two-dimensional Gaussian clusters with means (0,0), (6,0) and (8,0) and variances $4I_2$,$0.4I_2$ and $0.4I_2$, respectively. Spectral clustering and NMM use $\sigma = 0.95$. Plots (d)-(f) show the cluster-conditional densities estimated by the NMM. 85

4.4 Performance of the NMM on nine of the 26 datasets used, with varying value of the percentile (ρ) used for choosing the kernel bandwidth (σ). The NMM algorithm is compared with NJW (Spectral Clustering), K-means and the best of the three linkage based methods. 94

4.4 (Continued from previous page) 95

4.4 (Continued from previous page.) 96

4.4 (Continued from previous page.) 97

4.4 (Continued from previous page.) 98

4.5 Clustering performance and Constraint Satisfaction on UCI datasets for the proposed NPM algorithm. 102

4.5 (Continued from previous page...) Clustering performance and Constraint Satisfaction on UCI datasets for the proposed NPM algorithm. 103

4.6 Clustering performance and Constraint Satisfaction on UCI datasets for spectral clustering. 104

4.6 (Continued from previous page...) Clustering performance and Constraint Satisfaction on UCI datasets for spectral clustering. 105

4.7 Performance comparison between spectral clustering and the proposed non-parametric mixture model. 106

4.7 Performance comparison between spectral clustering and the proposed non-parametric mixture model. 107

5.1 Steps involved in constructing a bag-of-visual-words representation of images. (a) Given collection of images. (b) Features at key points are extracted (the SIFT operator [2] is a popular choice). (c) The key points are pooled togheter. (d) The key points are clustered using hierarchical K-means algorithm. The centroids obtained after clustering the key points are called the visual words. (e) The features in the key points in each image are assigned a cluster label, and the image is represented as a frequency histogram over the cluster labels. The centroids sharing common parent are considered similar to each other, and are called *visual synonyms*. Visual synonyms are shown using the same color in the table. 111

5.2 Illustration of SIFT key points, visual synonyms and feature selection at a group level. The first row shows a pair of images input for feature selection. Note that the key points occur in groups. Same colored marker is used for key points belonging to a group. Feature selection by the proposed feature selection algorithm acts at a group level by removing the entire group of unrelated features. 135

5.3 Feature selection using group-LASSO. In each pair of images, the left image shows the key points extracted from the original image and the right image shows the selected key points using the proposed feature selection algorithm. The number below each image indicates the number of key points (kp), and the number of groups (shown in brackets). 136

CHAPTER 1

Introduction

The advent of cheap sensors and storage devices has resulted in the generation, storage and consumption of massive amounts and variety of data in the form of text, video, image and speech. Multimedia content generation, which was once confined to recording studios or editorial offices, has now become a household activity. Internet users contribute to the media generation through blogs and vlogs (Blogspot), tweets (Twitter), photos (Flickr), audio and video recordings (Youtube). In addition, users unintentionally contribute to this surge through the records of their online activities such as Internet search logs (Google, Yahoo/Bing), advertisement clicks (Google and Yahoo), shopping (Amazon, Ebay), network logs (ISPs), browser logs (ISPs) to name just a prominent few!

It is not surprising then that automatic data analysis plays a central role for analyzing, filtering and presenting the user with the information he is searching for. Machine learning, pattern recognition and data mining are three related fields that study and develop algorithms to perform the necessary data analysis. Traditionally, data analysis has been studied in two settings:

Supervised Learning Given a set of input objects $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{n}$, and a set of corresponding outputs (class labels) for the i-th object, $\mathbf{y}_i = \{y_k^i\}_{k=1}^{K}$, where K is the number of output variables per input object, supervised learning aims to estimate a mapping $f : \mathcal{X} \rightarrow \mathcal{Y}$ such

that the output for a test object $\mathbf{x}$ (that was not seen during the training phase) may be predicted with high accuracy. For instance, $\mathcal{X}$ can be a collection of documents, and $\mathcal{Y}$ can be a collection of labels specifying if a user finds the corresponding document interesting (or not). The algorithm must learn a function f that predicts if the user will be interested in a particular document that has not yet been labeled.

Unsupervised Learning Given a set of objects, $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$, and a *similarity measure* between pairs of objects $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, the goal of unsupervised learning is to partition the set such that the objects within each group are more similar to each other than the objects between groups. For example, given a set of documents, the algorithm must group the documents into categories based on their content alone without any external labels. Unsupervised learning is popularly known as *clustering*.

Supervised learning expects training data that is completely labeled. On the other extreme, unsupervised learning is applied on completely unlabeled data. Unsupervised learning is more difficult problem than supervised learning due to the lack of a well-defined user-independent objective [3, 4]. For this reason, it is usually considered an ill-posed problem that is exploratory in nature [5]; that is, the user is expected to validate the output of the unsupervised learning process. Devising a fully automatic unsupervised learning algorithm that is applicable in a variety of data settings is an extremely difficult problem, and possibly infeasible. On the other hand, supervised learning is a relatively easier task compared to unsupervised learning. The ease comes with an added cost of creating a labeled training set. Labeling a large amount of data may be difficult in practice because data labeling:

1. *is expensive*: human experts are needed to perform labeling. E.g. Experts need to be paid to label, or tools such as Amazon's Mechanical turk [6] must be used.

2. *has uncertainty about the level of detail*: the labels of objects change with the granularity at which the user looks at the object. As an example, a picture of a person can

be labeled as “person”, or at a greater detail “face”,“eyes”,“torso” etc.

3. *is difficult*: sometimes objects must be sub-divided into coherent parts before they can be labeled. For example, speech signals and images have to be accurately segmented into syllables and objects, respectively before labeling can be performed.

4. *can be ambiguous*: objects might have non-unique labellings or the labellings themselves may be unreliable due to a disagreement among experts.

5. *uses limited vocabulary*: Typical labeling setting involves selecting a label from a list of pre-specified labels which may not completely or precisely describe an object. As an example, labeled image collections usually come with a pre-specified vocabulary that can describe only the images that are already present in the training and testing data.

Unlabeled data is available in abundance, but it is difficult to learn the underlying structure of the data. Labeled data is scarce but is easier to learn from. Semi-supervised learning is designed to alleviate the problems of supervised and unsupervised learning problems, and has gained significant interest in the machine learning research community [7].

1.1 Semi-supervised learning

Semi-supervised learning (SSL) works in situations where the available information in data is in between those considered by the supervised and unsupervised learners; i.e. it can be approached from both supervised and unsupervised learning problems by augmenting their traditional inputs. Various sources of side-information considered in the literature are summarized in Table 1.1.

Semi-supervised classification

Semi-supervised classification algorithms train a classifier given both labeled and unlabeled data. A special case of this is the well known transductive learning [8], where the goal is to label only the unlabeled data available during training. Semi-supervised classification can also be viewed as an unsupervised learning problem with only a small amount of labeled training data.

Semi-supervised clustering

Clustering is an ill-posed problem, and it is difficult to come up with a general purpose objective function that works satisfactorily with an arbitrary dataset [4]. If any side-information is available, it must be exploited to obtain a more useful or relevant clustering of the data. Most often, side-information in the form of pairwise constraints ("a pair of objects belong to the same cluster or different clusters") is available. The pairwise constraints are of two types: *must-link* and *cannot-link* constraints. The clustering algorithm must try to assign the same label to the pair of points participating in a must-link constraint, and assign different labels to a pair of points participating in a cannot-link constraint. These pairwise constraints may be specified by a user to encode his preferred clustering. Pairwise constraints can also be automatically inferred from the structure of the data, without a user having to specify them. As an example, web pages that are linked to one another may be considered as participating in a must-link constraint [9].

Semi-supervised feature selection

Feature selection can be performed for both supervised and unsupervised settings depending on the data available. Unsupervised feature selection is difficult for the same reasons that make clustering difficult – lack of a clear objective apart from the model assumptions. Supervised feature selection has the same limitations as classification, i.e. scarcity of labeled data. Semi-supervised feature selection aims to utilize pairwise constraints in order

Task	Typical input	Side-information for semi-supervised learning	References
Classification	Labeled data	Unlabeled data	[11]
Classification	Labeled data	Weakly related unlabeled data	[12]
Multilabel learning	Multi-label data	Unlabeled data	[13]
Multilabel learning	Multi-label data	Partially labeled examples	[14]
Clustering	Unlabeled data	Labeled data	[15]
Clustering	Unlabeled data	Pairwise constraints	[16]
Clustering	Unlabeled data	Group constraints	[17]
Clustering	Similarity metric	Balancing constraints	[18]
Ranking	Similarity metric	Partial ranking	[19]

Table 1.1: Different kinds of semi-supervised settings considered in the literature.

to identify a possibly superior subset of features for the task.

Many other learning tasks, apart from classification and clustering, have their semi-supervised counterparts as well (e.g., semi-supervised ranking [10]). For example, page ranking algorithms used by search engines can utilize existing partial ranking information on the data to obtain a final ranking based on the query.

1.2 Thesis contributions

Most semi-supervised learning algorithms developed in the literature (summarized in Chatper 2) attempt to modify existing supervised or unsupervised algorithms, or devise new approaches. This is often not desirable since a significant amount of effort may already have been invested in developing pattern recognition systems by fine tuning the parameters, or incorporating domain knowledge. The high-level goals of the thesis are as follows:

- *To design semi-supervised learning methods and algorithms that improve the existing and established supervised and unsupervised learning algorithms without having to modify them.*

- *To develop semi-supervised approaches following the above principle for each of the standard pattern recognition problems, namely, supervised learning, unsupervised learning and feature selection.*

The above goals can be achieved by using the side information in one of the following ways:

1. Design wrapper algorithms that use existing learning algorithms as components and improve them using the side information (e.g. unlabeled data for classification).

2. Use the side information to select critical parameters of the algorithm.

3. Incorporate side information directly into the data representation (features or similarity matrix) so that supervised and unsupervised algorithms can be directly used.

This thesis contributes to the field of semi-supervised classification and clustering by attempting to answer the following questions:

1. A meta semi-supervised-learning algorithm, called *SemiBoost* was developed that is presented in Chapter 3. It is designed to iteratively improve a given supervised classifier in the presence of a large number of unlabeled data.

2. A non-parametric mixture model using kernel density estimation is presented in Chapter 4. The resulting algorithm can discover arbitrary cluster structures in the data. Since the algorithm is probabilistic in nature, several issues like the number of clusters, incorporating side information etc., can be handled in a principled manner. Side-information in the form of pairwise constraints is used to estimate the critical parameters of the algorithm.

3. Curse of dimensionality is a well known problem in pattern recognition and machine learning. Many methods face challenges in analyzing high-dimensional data that are being generated in various applications (e.g, images and documents represented as

bag-of-words, gene microarray analysis etc.). Given a set of unlabeled examples, and an oracle that can label the pairwise constraints as must-link or cannot link, an algorithm is proposed to select a subset of relevant features from the data.

CHAPTER 2

Background

Most semi-supervised learning methods are extensions of existing supervised and unsupervised algorithms. Therefore, before introducing the developments in semi-supervised learning literature, it is useful to briefly review supervised and unsupervised learning approaches.

2.1 Supervised learning

Supervised learning aims to learn a mapping function $f : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are input and output spaces, respectively (e.g. classification and regression [20, 21]). The process of learning the mapping function is called *training* and the set of labeled objects used is called the *training data* or the *training set*. The mapping, once learned, can be used to predict the labels of the objects that were not seen during the training phase. Several pattern recognition [22, 20, 21] and machine learning [23, 21] textbooks discuss supervised learning extensively. A brief overview of supervised learning algorithms is presented in this section.

Supervised learning methods can be broadly divided into *generative* or *discriminative* approaches. Generative models assume that the data is independently and identically distributed and is generated by a parameterized probability density function. The parameters

are estimated using methods like the Maximum Likelihood Estimation (MLE), Maximum A Posteriori estimation (MAP) [20], Empirical Bayes and Variational Bayes [21]. Probabilistic methods could further be divided into *frequentist* or *Bayesian*. Frequentist methods estimate parameters based on the observed data alone, while Bayesian methods allow for inclusion of prior knowledge about the unknown parameters. Examples of this approach include the Naive Bayes classifier, Bayesian linear and quadratic discriminants to name a few.

Instead of modeling the data generation process, discriminative methods directly model the decision boundary between the classes. The decision boundary is represented as a parametric function of data, and the parameters are learned by minimizing the classification error on the training set [20]. Empirical Risk Minimization (ERM) is a widely adopted principle in discriminative supervised learning. This is largely the approach taken by Neural Networks [24] and Logistic Regression [21]. As opposed to probabilistic methods, these do not assume any specific distribution on the generation of data, but model the decision boundary directly.

Most methods following the ERM principle suffer from poor generalization performance. This was overcome by Vapnik's [25] Structural Risk Minimization (SRM) principle which adds a regularity criterion to the empirical risk that selects a classifier with good generalization ability. This led to the development of Support Vector Machines (SVMs) which regularize the complexity of classifiers while simultaneously minimizing the empirical error. Methods following ERM such as Neural networks, and Logistic Regression are extended to their regularized versions that follow SRM [21].

Most of the above classifiers implicitly or explicitly require the data to be represented as a vector in a suitable vector space, and are not directly applicable to nominal and ordinal features [26]. Also, most discriminative classifiers have been developed for only two classes. Multiclass classifiers are realized by combining multiple binary (2-class) classifiers, or using coding methods [20].

Decision trees is one of the earliest classifier [23], that can handle handle a variety of data with a mix of both real, nominal, missing features and multiple classes. It also provides interpretable classifiers, which give a user an insight about which features are contributing for a particular class being predicted for a given input example. Decision trees could produce complex decision rules, and are sensitive to noise in the data. Their complexity can be controlled by using approaches like pruning, however, in practice classifiers like SVM or Nearest Neighbor have been shown to outperform decision trees on vector data.

Ensemble classifiers are meta-classification algorithms that combine multiple component classifiers (called base classifiers) to obtain a meta-classifier with the hope that they will perform better than any of the individual component classifiers. Bagging [27] and Boosting [28, 29] are the two most popular methods in this class. Bagging is a short form for bootstrap aggregation, which trains multiple instances of a classifier on different subsamples (bootstrap samples) of the training data. The decision on an unseen test example is taken by a majority vote among the base classifiers. Boosting, on the other hand, samples training data more intelligently by sampling examples that are difficult for the existing ensemble to classify with a higher preference.

2.2 Unsupervised learning

Unsupervised learning or clustering, is a significantly more difficult problem than classification because of the absence of labels on the training data. Given a set of objects, or a set of pairwise similarities between the objects, the goal of clustering is to find *natural* groupings (clusters) in the data. The mathematical definition of what is considered a natural grouping defines the clustering algorithm. A very large number of clustering algorithms have already been published, and new ones continue to appear [30, 3, 31]. We broadly divide the clustering algorithms into groups based on their fundamental assumptions, and discuss a few representative algorithms in each group, ignoring minor variations within the

group.

K-means [30, 3], arguably, is the most popular and widely used clustering algorithm. K-means is an example of a sum of squared error (SSE) minimization algorithm. Each cluster is represented by its centroid. The goal of K-means is to find the centroids and the cluster labels for the data points such that the sum-of-squared error between each data point and its closest centroid is minimized. K-means is initialized with a set of random cluster centers, that are iteratively updated by assigning the closest data point to each center, and recomputing the centroids. ISODATA [32] and Linear Vector Quantization [33] are closely related SSE minimization algorithms that are independently proposed in different disciplines.

Parametric mixture models are well known in statistics and machine learning communities [34]. A mixture of parametric distributions, in particular, GMM [35, 36], has been extensively used for clustering. GMMs are limited by the assumption that each component is homogeneous, unimodal, and generated using a Gaussian density. Latent Dirichlet Allocation [37] is a multinomial mixture model that has become the de facto standard for text clustering.

Several mixture models have been extended to their non-parametric form by taking the number of components to infinity in the limit [38, 39, 40]. A non-parametric prior is used in the generative process of these infinite models (e.g. Dirichlet Process) for clustering in [38]. One of the key advantages offered by the non-parametric prior based approaches is that they adjust their complexity to fit the data by choosing the appropriate number of *parametric* components. Hierarchical Topic Models [39] are clustering approaches that have seen huge success in clustering text data.

Spectral clustering algorithms [41, 42, 43] are popular non-parametric models that minimize an objective function of the form $J(f) = f^T \Delta f$, where f is the function to be estimated, and Δ is the discrete graph Laplacian operator. Kernel K-means is a related kernel based algorithm, which generalizes the Euclidean distance based K-means to arbitrary

metrics in the feature space. Using the kernel trick, the data is first mapped into a higher dimensional space using a possibly non-linear map, and a K-means clustering is performed in the higher dimensional space. In [44], the explicit relation (equivalence for a particular choice of normalization of the kernel) between Kernel K-means, Spectral Clustering and Normalized Cut was established.

Non-parametric density based methods are popular in the data mining community. Mean-shift clustering [45] is a widely used non-parameteric density based clustering algorithm. The objective of Mean-shift is to identify the modes in the kernel-density, seeking the nearest mode for each point in the input space. Several density based methods like DBSCAN also rely on empirical probability estimates, but their performance degrades heavily when the data is high dimensional. A recent segmentation algorithm [46] uses a hybrid mixture model, where each mixture component is a convex combination of a parametric and non-parametric density estimates.

Hierarchical clustering algorithms are popular non-parametric algorithms that iteratively build a cluster tree from a given pairwise similarity matrix. Agglomerative algorithms such as Single Link, Complete Link, Average Link [4, 30], Bayesian Hierarchical Clustering [47], start with each data point in a single cluster, and merge them succesively into larger clusters based on different similarity criteria at each iteration. Divisive algorithms start with a single cluster, and successively divide the clusters at each iteration.

2.3 Semi-supervised algorithms

Semi-supervised learning algorithms (See Section 1.1) can be broadly classified based on the role the available side information plays in providing the solution to supervised or unsupervised learning.

Table 2.1: A comparison of different clustering algorithms proposed in the literature. Given the large number of available algorithms, only a few representative ones are shown here.

Method/Family	Algorithm	Cluster Definition
Non-parametric density estimation	Jarvis-Patrick [48], DBSCAN [49], MeanShift [45], DENCLUE [50]	Spatially dense and connected regions correspond to clusters.
Spectral Algorithms	Min Cut [51], Ratio Cut [52], Normalized Cut [41], Spectral Clustering [42]	Sparse regions correspond to the cluster separation boundaries.
Probabilistic Mixture models	Mixture of Gaussians [53, 36], Latent Dirichlet Allocation [37], PLSI [54]	Data comes from an underlying probabilistic mixture model.
Squared-Error	K-Means [20, 3], X-means [55], Vector Quantization [33], Kernel K-means [56]	Data points close to their cluster representative belong to the same cluster.
Hierarchical	Single Link, Complete Link and Average Link [20], Bayesian Hierarchical Clustering [57], COBWEB [58]	Data points close to each other fall in the same cluster.
Information Theoretic	Minimum Entropy [59, 60], Information Bottleneck [61], Maximum entropy [62]	Clustering is obtained by compressing the data to retain the maximum amount of information.

2.3.1 Semi-supervised classification

While semi-supervised classification is a relatively new field, the idea of using unlabeled samples to augment labeled examples for prediction was conceived several decades ago. The initial work in semi-supervised learning is attributed to Scudders for his work on "self-learning" [63]. An earlier work by Robbins and Monro [64] on sequential learning can also be viewed as related to semi-supervised learning. Vapnik's Overall Risk Minimization (ORM) principle [65] advocates minimizing the risk over the labeled training data as well as the unlabled data, as opposed to the Empirical Risk Minimization, and resulted in transductive Support Vector Machines.

Fig. 2.1 gives the basic idea of how unlabeled data could be useful in learning a classifier. Given a set of labeled data, a decision boundary may be learned using any of the supervised learning methods (Fig. 2.1(a)). When a large number of unlabeled data is provided in addition to the labeled data, the true structure of each class is revealed through the distribution of the unlabeled data (Fig. 2.1(b)). The unlabeled data defines a "natural region" for each class, and the region is labeled by the labeled data. The task now is no longer just limited to separating the labeled data, but to separate the regions to which the labeled data belong. The definition of this "region" constitutes some of the fundamental assumptions in semi-supervised learning.

Existing semi-supervised classification algorithms may be classified into two categories based on their underlying assumptions. An algorithm is said to satisfy the *manifold assumption* if it utilizes the fact that the data lie on a low-dimensional manifold in the input space. Usually, the underlying geometry of the data is captured by representing the data as a graph, with samples as the vertices, and the pairwise similarities between the samples as edge-weights. Several graph based algorithms such as Label propagation [11, 66], Markov random walks [67], Graph cut algorithms [68], Spectral graph transducer [69], and Low density separation [70] proposed in the literature are based on this assumption.

The second assumption is called the *cluster assumption* [71]. It states that the data

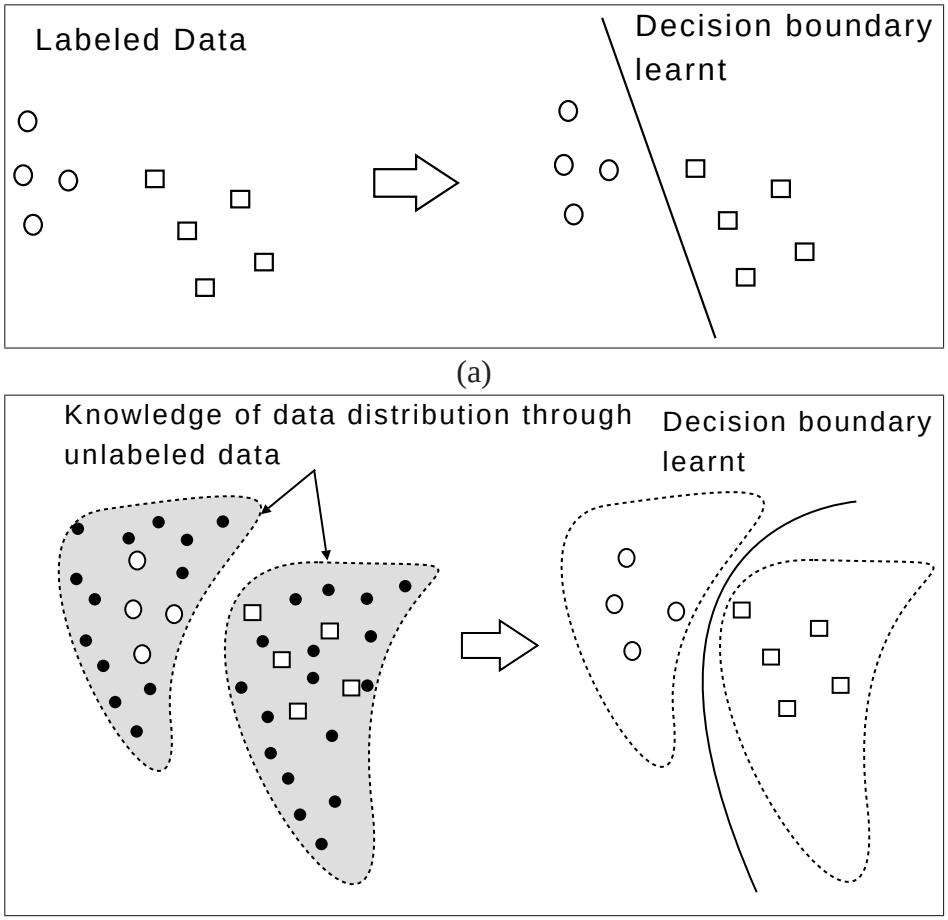


Figure 2.1: Utility of the unlabeled data in learning a classifier. (a) Classifier learned using labeled data alone. (b) Utility of unlabeled data. The filled dots show the unlabeled data. The gray region depicts the data distribution obtained from the unlabeled data.

samples with high similarity between them, must share the same label. This may be equivalently expressed as a condition that the decision boundary between the classes must pass through low density regions. This assumption allows the unlabeled data to regularize the decision boundary, which in turn influences the choice of the classification models. Many successful semi-supervised algorithms like TSVM [72] and Semi-supervised SVM [73] follow this approach. These algorithms assume a model for the decision boundary, resulting in an inductive classifier.

Table 2.2: A summary of semi-supervised classification algorithms. T or I in the last column denotes Transductive or Inductive property of the algorithm, respectively.

Group	Approach	Summary	T/I
Manifold Assumption	Label Propagation [11, 66]	Graph-based; Maximize label consistency using Graph Laplacian	T
	Min-cuts [68]	Edge-weight based graph-partitioning algorithm constraining nodes with same label to be in same partition	T
	MRF [67], GRF [74]	Markov random field and Gaussian random field models	T
	LDS [75]	TSVM trained on a dimensionality reduced data using graph-based kernel	T
	SGT [69]	Classification cost minimized with a Laplacian regularizer	T
	LapSVM [76]	SVM with Laplacian regularization	I
Cluster Assumption	Co-training [77]	Maximizes predictor consistency among two distinct feature views	I
	Self-training [78]	Assumes pseudo-labels as true labels and retrains the model	I
	SSMB [79]	Maximizes pseudo-margin using boosting	I
	ASSEMBLE [80]	Maximizes pseudo-margin using boosting	I
	Mixture of Experts [81]	EM based model-fitting of mixture models	I
	EM-Naive Bayes [82]	EM based model-fitting of Naive Bayes	I
	TSVM [72], S3VM [73]	Margin maximization using density of unlabeled data	I
	Gaussian processes [83]	Bayesian discriminative model	I
Manifold & Cluster Assumptions	SemiBoost (Proposed)	Boosting with a graph Laplacian inspired regularization	I

Bootstrapping Classifiers from Unlabeled data

One of the first uses of unlabeled data was to bootstrap an existing supervised learner using unlabeled data iteratively. The unlabeled data is labeled using a supervised learner trained on the labeled data, and the training set is augmented by the most confident labeled samples. This process is repeated until all the unlabeled data have been processed. This is popularly known as "Self-training", which was first proposed by Scudders [63]. Yarowsky [84] applied self-learning to the "word sense" disambiguation problem. Rosenberg et al. [85] applied self-training for object detection.

Several classifiers proposed later follow the bootstrapping architecture similar to that of self-training, but with a more robust and well-guided selection procedure for the unlabeled samples for inclusion in the training data. Semi-supervised generative models using EM [53], for instance, the Semi-supervised Naive Bayes [86], is a "soft" version of self-training. Many ensemble classification methods, in particular, those following the semi-supervised boosting approach [79, 87, 88] use specific selection procedures for the unlabeled data, and use a weighted combination of classifiers instead of choosing the final classifier.

Margin based classifiers

The success of margin based methods in supervised classification motivated a significant amount of research in their extension to semi-supervised learning. The key idea of margin based semi-supervised classifiers is to model the change in the definition of margin in the presence of unlabeled data. Margin based classifiers are usually extensions of Support Vector Machines (SVM). An SVM minimizes the empirical error on the training set, along with a regularization term that attemps to select the classifier with maximum margin. For a given set of labeled examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, and a loss function $\ell : \mathcal{X}, \mathcal{Y} \rightarrow \mathbb{R}$, SVM finds a classifier $f(x)$ minimizing the following objective function

$$J_{svm}(f) = ||f||^2_{\mathcal{H}} + \sum_{i=1}^{n} \ell(f(\mathbf{x}_i), y_i) \tag{2.1}$$

The first term in Eq (2.1) corresponds to the complexity of the function computed as the norm in an appropriate function space (Hilbert space), and the second term corresponds to the empirical error of the classifier f on the training set measured using a convex loss function $\ell(f(\mathbf{x}), y)$. The loss $\ell(f(\mathbf{x}), y)$ is defined only when the label y of the sample is known. The key idea behind the semi-supervised extensions of support vector machines is to define the loss for unlabeled data as $\ell(\mathbf{x}, y^u) = \min_{\hat{y}=\pm 1} \ell(\mathbf{x}, \hat{y})$, where $\hat{y}$ is the label assigned to the unlabeled example during learning (also called the pseudo-label).

Vapnik [8] first formulated this problem and proposed a branch and bound algorithm. A Mixed Integer Programming based solution is presented in [89], which is called Semi-supervised SVM or S^{3}VM. Fung and Mangasarian [73] proposed a successive linear approximation to the $\min(.)$ function in the loss function, and proposed VS^{3}VM. None of these methods are applicable to real datasets (even small size datasets) owing to their high computational complexity.

Transductive SVM (TSVM) [90] is one of the early attempts to develop a practically usable algorithm for semi-supervised SVM. TSVM provides an approximate solution to the combinatorial optimization problem of semi-supervised SVM by first labeling the unlabeled data with an SVM trained on the labeled data, followed by switching the individual labels of unlabeled data such that the objective function is minimized. Gradient descent was used in [75] to minimize the same objective function, while defining an appropriate subgradient for the $\min(.)$ function. This approach was called ∇TSVM, and its performance is shown to be comparable to that of the other optimization schemes discussed above.

Ensemble Methods

Almost all the early semi-supervised extensions to boosting algorithms relied on the margin interpretation of AdaBoost [28, 29]. It is well known that boosting algorithms minimize the following objective function:

$$J_{boost}(H) = \sum_{i=1}^{n_l} M(-y_i H(x_i)), \tag{2.2}$$

where $M(.)$ is a convex cost function. Choosing $M = \exp(.)$ results in the well-known AdaBoost algorithm. The quantity $y_i H(\mathbf{x}_i)$ is the classification margin by definition. Boosting algorithms like ASSEMBLE [89] and Semi-supervised Margin Boost (SSMB) [79] extend the definition of margin to unlabeled samples. Margin over unlabeled samples is defined as $|H(\mathbf{x}_i)|$ by ASSEMBLE and as $(H(\mathbf{x}_i))^2$ by SSMB. This definition of margin is reasonable since both the modulus and square functions are monotonically increasing functions of margin (Maximizing a monotonically increasing function of margin effectively maximizes the margin), and they conveniently eliminate the value of unknown label from the definition. More detailed discussion of the boosting algorithms with relevance to the proposed SemiBoost algorithm is presented in Chatper 3. In particular, we note that the margin over unlabeled data is not a sufficiently good measure for classification performance. Ideas from highly successful unsupervised methods are combined with the boosting algorithm in Chapter 3 to obtain a powerful boosting classifier, that is shown to improve the average margin.

Graph Connectivity

Graph theory has been known to be powerful tool for modeling unsupervised learning (clustering) problems since its inception [100, 101] to relatively recent Normalized Cuts [102] and Spectral clustering [103], and shown to perform well in practice [104, 105, 106]. Graph based methods represent the data as a weighted graph, where the nodes in the graph represent the data points, and the edge weights represent the similarity between the correspond-

Table 2.3: Unsupervised learning algorithms and their corresponding semi-supervised counterparts.

Method/Family	Original Unsupervised Algorithm	Semi-supervised Extension
Non-parametric density estimation	MeanShift [45]	Weakly-supervised Mean-Shift [91]
Spectral Algorithms	Min Cut [51] Ratio Cut [52] and Normalized Cut [41] Spectral Clustering [42]	 SS-Graph Clustering [92, 93, 94] Spectral Learning [95]
Probabilistic Mixture models	Mixture of Gaussians [53, 36], Latent Dirichlet Allocation [37], PLSI [54]	Penalized Probabilistic Clustering [96], Model based clustering with constraints [17]
Squared-Error	K-Means [20, 3]	COP-K-Means [97], HMRF-K-means [15], PC-K-means, MPCK-means [98]
Hierarchical	Single Link, Complete Link and Average Link [20] COBWEB [58]	[99] COP-COBWEB [16]
Information Theoretic	Minimum Entropy [59, 60], Information Bottleneck [61], Maximum entropy [62]	No semi-supervised extention proposed yet.

ing pair of data points. The success of graph based algorithms in unsupervised learning motivates its use in semi-supervised learning (SSL) problems.

A extension to Min-cut clustering algorithm for transduction is presented in [68]. The edge weight between a pair of samples is set to ∞ if they share the same label, to ensure that they remain in the same partition after partitioning the graph. Szummer and Jakkola [67] and Zhu and Ghaharamani [11] model the graph as a discrete Markov random field, where the nomalized weight of each edge represents the probability of a label (state) jumping from one data point to the other. The solution is modeled as the probability of a label (from a labeled data point) reaching an unlabeled data point in a finite number of steps. Zhu et al., [74] relax the Markov random field with a discrete state space (labels) to a Gaussian random field with continous state space, thereby achieveing an approximate solution with lower computational requirements.

Most graph based semi-supervised learning methods are non-parametric and transductive in nature, and can be shown as solutions to the discrete Green's function, defined using the discrete Graph Laplacian.

Definition 1. *For a weighted graph $\mathcal{G} = \langle V, W \rangle$, where W represents the edge weight matrix, the Graph Laplacian Δ is defined as $\Delta = (D - W)$, where D is a diagonal matrix containing the sums of rows of W.*

The quantity $\mathbf{y}^t \Delta \mathbf{y}$ measures the inconsistency between the similarity matrix W and the labeling $\mathbf{y}$, and plays a central role in graph based SSL algorithms. Given a similarity matrix W, where $[W]_{ij} = w_{ij}$, $\mathbf{y}^t \Delta \mathbf{y}$ can be expanded as

$$\mathbf{y}^t \Delta \mathbf{y} = -\frac{1}{2} \sum_{ij} w_{ij} (y_i - y_j)^2. \tag{2.3}$$

To minimize the inconsistency, the difference between y_i and y_j must be small whenever the similarity w_{ij} is large, and vice versa.

Eq (2.3) has several useful mathematical properties. Most importantly, it is a convex function of the labels, and hence has a unique minima. Normalized Cut [102] is an unsu-

pervised algorithm that minimizes a normalized version of Eq (2.3) using spectral methods. Spectral graph transducer [69] minimizes graph Laplacian over both labeled and unlabeled data, with an additional term penalizing the difference in prediction over the labeled samples. Manifold regularization [107] is a semi-supervised extension of SVMs that searches for a function that minimizes the graph Laplacian in addition to the standard SVM objective function. Unlike all other extensions of SVM, the resulting optimization function is convex, and can be optimized very efficiently.

2.3.2 Semi-supervised clustering

Clustering aims to identify groups of data such that the points within each group are more similar to each other than the points between different groups. Clustering problem is ill-posed, and hence multiple solutions exist that can be considered equally valid and acceptable. Semi-supervised clustering utilizes any additional information, called *side-information*, that is available to disambiguate between the solutions. The side information is usually present in the form of instance level *pairwise* constraints [16]. Pairwise constraints are of two types – *must-link* constraints and *cannot-link* constraints. Given a pair of points, must link constraints require the clustering algorithm to assign the same label to the points. On the other hand, cannot-link constraints require the clustering algorithm to assign different labels to the points. However, several other forms of side-information have been considered in the literature as summarized in Table 1.1. Figure 2.2 shows the utility of pairwise constraints in clustering.

Penalizing Constraints

One of the earliest constrained clustering algorithms was developed by Wagstaff and Cardie [16, 97], called the COP K-means algorithm. The cluster assignment step of K-means algorithm was modified with an additional check for constraint violations. However, when constraints are noisy or inconsistent, it is possible that there are some points that are

not assigned to any cluster. This was mitigated in an approach by Basu et. al. [109] which penalizes constraint violations instead of imposing them in a hard manner. A constrained clustering problem is modeled using a Hidden Markov Random Field (HMRF) which is defined over the data and the labels, with labels as the hidden states that generate the data points. The constraints are imposed on the values of the hidden states. Inference is carried out by an algorithm similar to that of K-means which penalizes the constraint violations.

Generative models are very popular in clustering. Gaussian mixture model (GMM) is one of the well-known models used for clustering [53, 36]. Shental et al. [108] incorporated pairwise constraints into the GMMs. To achieve this, groups of points connected by must-link constraints are defined as *chunklets* and each chunklet is treated as a single point for clustering purposes. Zhao and Miller [111] proposed an extension to GMM which penalizes constraint violations. A method to automatically estimate the number of clusters in the data using the constraint information was proposed. Lu and Leen [96] incorporate the constraints into the prior over all possible clusterings. In particular, for a clustering z, they use a prior of the form $P(z) = \sum_i \sum_j W_{ij} I(z_i, z_j)$, where $I(x, y)$ is the indicator function which is 1 when $x = y$ and 0, otherwise, and $W_{i,j}$ is a penalty for violating the constraint between the i-th and j-th data points. Gibbs sampling is used to infer the cluster labels.

In many approaches that enforce constraints in a hard manner (including those that penalize them), non-smooth solutions are obtained. A solution is called non-smooth when a data point takes a cluster label that is different from all of its surrounding neighbors. As noted in [112], it is possible that the hypothesis that fits the constraints well may not fit the data well. Therefore, a trade off between satisfying the constraints and fit to the data is required. Lange et al. [110] alleviate this problem by involving all the data points into a constraint through a smooth label.

Adapting the Similarity

Several semi-supervised clustering methods operate by directly modifying the entries of the pair-wise similarity matrix that are involved in constraints. All these algorithms, reudce the distance between data points connected by must-link constraints and increase the distance between those connected by must-not link by a small value. Spectral Learning algorithm by Kamvar et al. [95] modifies the normalized affinity matrix by replacing the values corresponding to must-link constraints by 1 and must-not link constraints by 0. The specific normalization they use ensures that the resulting matrix is positive definite. The remaining steps of the algorithm are the same as the Spectral clustering algorithm by Ng et al. [103]. Klien et. al. [99] modified the dissimilarity metric by replacing the entries participating in must-link constraints with 0 and replaced the entries participating in cannot-link constraints by maximum pairwise distance incremented by 1. This is followed by a complete link clustering on the modified similarity matrix. Kulis et al. [93] propose a generalzation of Spectral Learning via semi-supervised extensions to the popular normalized cut [102], ratio cut and ratio association [52]. To ensure positive definiteness of the similarity matrix, they simply add an arbitrary positive quantity to the diagonal.

The specific values of increments chosen in the above algorithms impacts the performance of the clustering algorithm. In order to apply spectral algorithms, we need the pairwise similarity matrix to be positive semi-definite. Arbitrary changes (especially decrements) to the similarity matrix may not retain its positive semi-definiteness. Some methods avoid using spectral algorithms, while some update the simliarity matrix carefully to retain the essential properties. The similarity adaptation methods are adhoc in nature, and are superseded by the similarity learning approaches presented in the next section.

Learning the Similarity

The performance of a clustering algorithm depends primarily on the similarity metric defined between the samples. It is usually difficult to design a similarity metric that suits

all the clustering scenarios. For this reason, attempts have been made to directly learn the similarity metric from the data using the side information. Similarity metric learning is not a new problem, and has been considered before in both unsupervised dimensionality reduction methods (LLE [113], ISOMAP [114]) and supervised methods like Fisher Linear Discriminant [20], Large Margin Distance Metric Learning [115] and Neighborhood Component Analysis [116]. Only those methods that learn the distance metric in a semi-supervised setting, i.e., using pairwise constraints and unlabeled data are reviewed here.

Once a similarity metric is learned, standard clustering/classification algorithms may later be applied with the learned similarity metric. The distance metric learning problem can be posed in its generality as follows: learn a function $f : \mathcal{X} \times \mathcal{X} \rightarrow R$ such that the distance between points linked by must-link constraints is smaller than that between the points linked by must-not link constraints overall. The distance function is usually parametrized in its quadratic form, i.e $f_A(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T A \mathbf{x}_j$, where A is the unknown parameter to be estimated from the constraints.

Xing et al. [117] formulated distance metric learning as a constrained optimization problem, where A is estimated such that the sum of distances between points connected by must-link constraints is minimized, while constraining the sum of distances between points connected by must-not link to be greater than a fixed constant. Bar-Hillel et al. [118] proposed Relevant Component Analysis (RCA), which estimates a global transformation of the feature space by reducing the weights of irrelevant features such that the groups of data points linked by must-link constraints (called *chunklets*) are closer to each other. A modified version of the constrained K-means algorithm that learns a parametrized distance function is presented in [119].

Yang et al. [120] learn a local distance metric by using an alternating optimization scheme that iteratively selects the local constraints, and fits the distance metric to the constraints. They parametrize the kernel similarity matrix in terms of the eigenvalues of the top few eigenvectors of the pairwise similarity matrix computed using the RBF kernel. Hoi

et al. [121] present a non-parametric distance metric learning algorithm that addresses the limitations of quadratic distance functions used by almost all the other approaches. Lee et al. [122] proposed an efficient distance metric learning algorithm and applied it to a content based image retrieval task showing significant performance gains.

There has been a recent surge in the interest in online learning algorithms due to the large volume of datasets that need to be processed. Shalev-shwartz et al. [123] present an online distance metric learning algorithm called POLA, that learns a quadratic distance function (parametrized by the covariance matrix) from pairwise constraints. A batch version of the algorithm is obtained by multiple epochs of the online algorithm on the training data. Davis et al. [124] present online and batch versions of an algorithm that searches for the parameterized covariance matrix A that satisfes the constraints maximally. Additionally, a log-determinant regularizer is added to prevent A from moving too far away from the initial similarity metric A_0.

Applications

Clustering with constraints has been applied succesfully to several real world problems. Bar-Hillel et al. [125] used pairwise constraints for clustering as an intermediate step to speaker identification in a conversation. An application to video surveillance, where the temporal similarity between frames is used to generate must-link constraints between the pixels is presented in [118]. Wagstaff et al. [97] applied constrained clustering for GPS lane finding. Yu and Shi [92] used the constraint information generated from the fact that pixels near image boundaries may represent background and pixels at the center of the image may represent the foreground. They automatically generate the pairwise constraints relating foreground and background pixels and showed that the segmentation is significantly improved with the side-information. Yang et al. [126] applied a local distance metric learning algorithm using pairwise constraints for interactive search assisted diagonostics (ISAD) of mammogram images and demonstrated an improved accuracy in identifying clusters of

similar patient cases in the database.

Acquiring the Constraints

Most of the papers in semi-supervised clustering literature describe how to utilize the constraints once they are available, but relatively few methods consider automatic acuqisition of constraints. While it is generally easier for a user to provide pairwise constraints compared to assigning class labels, it is still tedious if it has to be done for a large number of object pairs.

Automatic constraint acqusition aims at encoding human knowledge in the form of pairwise constraints or to minimize the number of constraints a user has to specify by selecting the most important set of pairs of points to be labeled. When the necessary domain knowledge is not available to automatically derive the pairwise constraints, it is desirable to present the user the most informative pairs of points to label. Active learning approaches [127, 128, 129] aim to select the most informative pairs of points or such that a large performance gain is obtained from as few constraints as possible.

2.4 Does side-information always help?

There is a significant gap between theoretical analysis and the practice of semi-supervised learning. Most theoretical analyses aim to derive the conditions under which the side-information will always improve the performance of the learning algorithm. The available results are limited and applicable to narrow and ideal learning scenarios. Most results emphasize that the relation between the underlying structure of both labeled and unlabeled data (which is different from label smoothness assumption) is a major factor in determining the performance of a semi-supervised learner.

2.4.1 Theoretical observations

Semi-supervised Classification

Castelli and Cover [130] provide an analysis of the utility of the unlabeled data from a Bayes Risk perspective for a two-class classification problem, with known class conditional densities $P(\mathbf{x}|y)$, where $\mathbf{x}\in \mathbb{R}^d$, and $y \in \{\omega_1, \omega_4\}$. In particular, they establish that the labeled samples reduce Bayes error exponentially, while unlabeled samples reduce Bayes error linearly. For instance, in a trivial scenario where no labeled samples are available, the Bayes risk of a classifier on the two-class problem is equal to $\frac{1}{2}$, since any example might be labeled as any class. However, when a single sample is known from each class, the Bayes risk becomes $2\epsilon(1-\epsilon)$, where ϵ is the Bayes risk for the two class problem if all the data are labeled.

Zhang [131] analyzed the utility of unlabeled data from the perspective of Fisher Information. Cramer-Rao inequality states that for any unbiased estimator t_n of α based on n i.i.d samples, the covariance of t_n satisfies $cov(t_n) \geq (nI(\alpha))^{-1}$. When the data distribution and the conditional label distribution share the parameters, unlabeled data help in reducing the variance ($cov(t_n)$) of the estimator. This is the case for generative models. However, in discriminative models, $P(y|x)$ is directly modeled disregarding the data density $P(x)$, and therefore, unlabeled data do not help in this situation. This analysis is not applicable to non-probabilistic discriminiative semi-supervised classification algorithms like TSVM since they use the $P(\mathbf{x})$ to avoid keeping the decision boundary where the value of $P(\mathbf{x})$ is very high, thereby following the input-dependent regularization framework of [132]

Semi-supervised learning algorithms incorporating side information may not necessarily result in improved performance. In many cases, the performance of a learner may even degrade with the use of side information. In the case of generative classifiers, Cozman and Cohen [133] summarize their empirical observations regarding the performance degradation as follows. Unlabeled data helps to improve the parameter estimates, and in turn the

predictive performance of classifiers, when the model assumptions match the data. However, when the model assumptions do not match the structure of the data, unlabeled data potentially degrade the performance significantly. However, in practice, it is not possible to evaluate the match between structure of the data and the model, neccessitating the need for caution when incorporating unlableled data into generative models.

Ben-David et al. [134] noted that under Probably Approximately Correct (PAC) learning setting, unlabeled data does not improve the worst case sample complexity [1] of a classifier compared to that of labeled data by more than a constant factor, unless strict assumptions are made about the label distribution.

Balcan and Blum [135] proposed the notion of "compatiability function" which measures how "nicely" the classifier fits the unlabeled data in terms of a measure (e.g. margin). Since this reduces the hypothesis class to only those functions that fit the unlabled data well, the generalization error bounds improve. However, if there is a mismatch between the label structure of the unlableled data and the labeled data, this reduction in the hypothesis class retains only the poor performing classifiers, resulting in a degradation in the empirical performance of the resulting classifier.

Semi-supervised Clustering

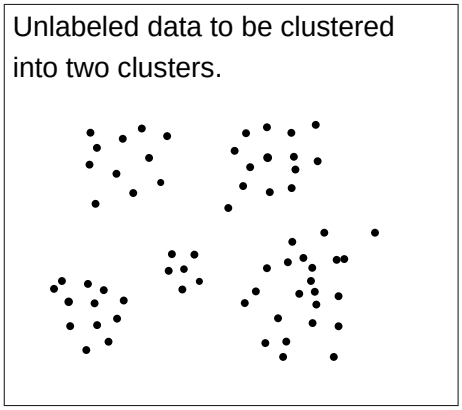
Semi-supervised clustering is a harder problem compared to semi-supervised classification, and it has not yet been amenable to theoretical analysis. Theoretical results pertaining to semi-supervised clustering aim to answer how and when the pairwise constraints are useful, or if they are useful at all. Davidson et al. [136] empirically observed that pairwise constraints can significantly degrade the performance of the clustering algorithm. They define two measures in an attempt to quantify the constraint set utility, called *informativeness* and *coherence*. Informativeness measures the mismatch between the unsupervised clustering algorithm and the constraint set. Coherence measures the internal consistency of the con-

[1] Sample complexity of a classifier is the number of samples m required to be sure with a probability $1-\delta$, that the test error of the classifier will not be more than a given ϵ.

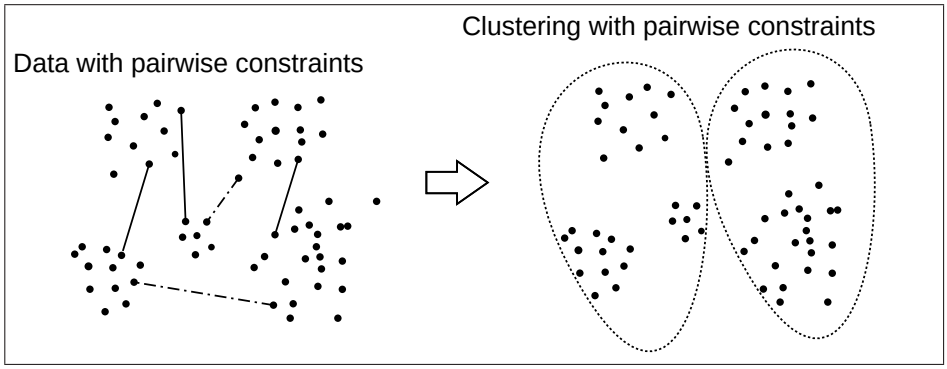
straint set. However, a greater value of informativeness does not necessarily mean that the clustering algorithm will perform better. Our experience also suggests that it is very difficult to conclude a priori whether a set of constraints will improve or degrade the clustering performance.

2.5 Summary

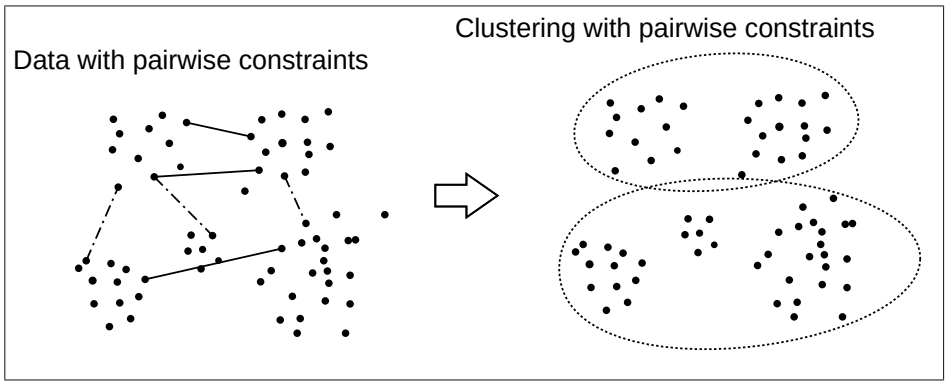
Semi-supervised classification has received significant amount of interest, as it provides a way to utilize the large amount of readily available unlabeled data for improving the classifier performance. Semi-supervised classification has been successfully applied to various applications in computer vision and machine learning, such as text classification [86], human computer interaction [137], content based image retrieval [138] , object detection [85], person identification [139], relevance feedback [140], computational linguistics [141] and protein categorization [7], to name a few. Similarity, side-information such as pairwise constraints has been utilized to improve the performance of clustering algorithms by aiding them in arriving at a clustering desired by the user. Semi-supervised learning continues to pose both theoretical and pracitcal questions to researchers in the machine learning. There is also an increasing interest in the fields of cognitive sciences and human psychology since there are demonstrated settings where humans performed semi-supervised learning [142].



(a) Input data to be clustered into 2 clusters.



(b) Clustering with a set of pairwise constraints. Solid line shows must-link constraints and dotted line shows cannot-link constraints.



(c) Different clustering of the same data obtained using a different set of pairwise constraints.

Figure 2.2: Utility of pairwise constraints in data clustering. (a) Input unlabeled data to be clustered into two clusters. Figures (b) and (c) show two different clusterings of data in (a) obtained by using two different sets of pairwise constraints.

CHAPTER 3

SemiBoost: Boosting for Semi-supervised Classification

3.1 Introduction

Most semi-supervised learning approaches, as discussed in Chapter 1, design specialized learning algorithms to effectively utilize both labeled and unlabeled data. However, it is often the case that a user already has a favorite (well-suited) supervised learning algorithm for his application, and would like to improve its performance by utilizing the available unlabeled data. In this light, a more practical approach is to design a technique to utilize the unlabeled samples, regardless of the underlying learning algorithm. Such an approach would accommodate for the task-based selection of a classifier, while providing it with an ability to utilize unlabeled data effectively. We refer to this problem of improving the performance of *any* supervised learning algorithm using unlabeled data as *Semi-supervised Improvement*, to distinguish our work from the standard semi-supervised learning problems.

To address the semi-supervised improvement, we propose a boosting framework, termed *SemiBoost*, for improving a given supervised learning algorithm with unlabeled data. Similar to most boosting algorithms [28], SemiBoost improves the classification accuracy iteratively. At each iteration, a number of unlabeled examples are selected and used

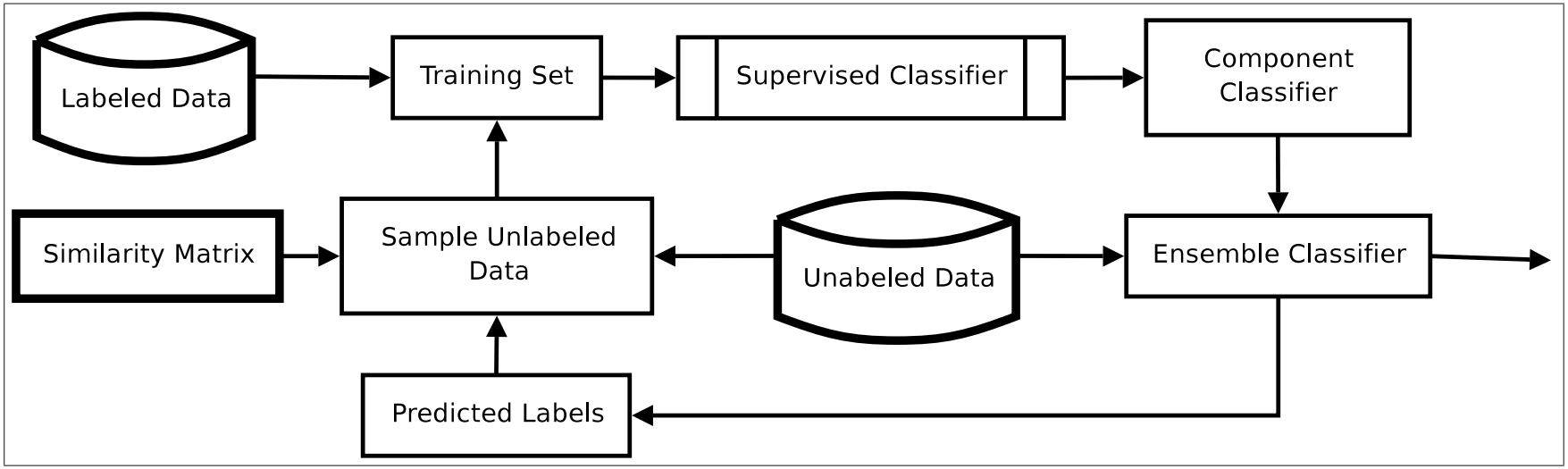


Figure 3.1: Block diagram of the proposed algorithm, SemiBoost. The inputs to SemiBoost are: labeled data, unlabeled data and the similarity matrix.

to train a new classification model using the given supervised learning algorithm. The trained classification models from each iteration are combined linearly to form a final classification model. An overview of the SemiBoost is presented in Figure 3.1. The key difficulties in designing SemiBoost are: (1) how to sample the unlabeled examples for training a new classification model at each iteration, and (2) what class labels should be assigned to the selected unlabeled examples. It is important to note that unlike supervised boosting algorithms where we select labeled examples that are difficult to classify, SemiBoost needs to select unlabeled examples, at each iteration.

One way to address the above questions is to exploit both the clustering assumption and the large margin criterion. One can improve the classification margin by selecting the unlabeled examples with the highest classification confidence, and assign them the class labels that are predicted by the current classifier. The assigned labels are hereafter referred to as the *pseudo-labels*. The labeled data, along with the selected pseudo-labeled data are utilized in the next iteration for training a second classifier. This is broadly the strategy adopted by approaches like Self-training [78], ASSEMBLE [80] and Semi-supervised MarginBoost [79]. However, a problem with this strategy is that the introduction of examples with predicted class labels may only help to increase the classification margin, without actually providing any novel information to the classifier. Since the selected unlabeled examples are the ones that can be classified confidently, they often are far away from the decision boundary. As a result, the classifier trained by the selected unlabeled examples is likely to share the same decision boundary with the original classifier that was trained only by the labeled examples. This is because by adjusting the decision boundary, the examples with high classification confidence will gain even higher confidence. This implies that we may need additional guidance for improving the base classifier, along with the maximum margin criterion.

To overcome the above problem, we propose to use the pairwise similarity measurements to guide the selection of unlabeled examples at each iteration, as well as for as-

signing class labels to them. For each unlabeled example $\mathbf{x}_i$, we compute the confidence of assigning the example $\mathbf{x}_i$ to the positive class as well as the confidence of assigning it to the negative class. These two confidences are computed based on the prediction made by the boosted classifier and the similarity among different examples. We then select the examples with the highest classification confidence together with the labeled examples to train a new classification model at each iteration. The new classification model will be combined linearly with the existing classification models to make improved predictions. Note that the proposed approach is closely related to graph-based semi-supervised learning approaches that exploit the manifold assumption. The following section discusses the existing semi-supervised learning methods, and their relationship with SemiBoost.

3.2 Related work

In Table 1.2 a brief summary of the existing semi-supervised learning methods and the underlying assumptions was presented. Recall that an inductive algorithm can be used to predict the labels of samples that are unseen during training (irrespective of it being labeled or unlabeled). On the other hand, transductive algorithms are limited to predicting only the labels of the unlabeled samples seen during training.

Graph-based approaches represent both the labeled and the unlabeled examples by a connected graph, in which each example is represented by a vertex, and pairs of vertices are connected by an edge if the corresponding examples have large similarity. The well known approaches in this category include Harmonic Function based approach [74], Spectral Graph Transducer (SGT) [69], Gaussian process based approach [83], Manifold Regularization [76] and Label Propagation approach [11, 66]. The optimal class labels for the unlabeled examples are found by minimizing their inconsistency with respect to both the supervised class labels and the graph structure.

A popular way to define the inconsistency between the labels $\mathbf{y} = \{y_i\}_{i=1}^{n}$ of the sam-

ples $\{\mathbf{x}_i\}_{i=1}^n$, and the pairwise similarities $S_{i,j}$ is the quadratic criterion,

$$F(\mathbf{y}) = \sum_{i=1}^{n}\sum_{j=1}^{n} S_{i,j}(y_i - y_j)^2 = \mathbf{y}^T L\mathbf{y}$$

where L is the combinatorial graph Laplacian. Given a semi-supervised setting, only a few labels in the above consistency measure are assumed to be known, and the rest are considered unknown. The task is to assign values to the unknown labels in such a way that the overall inconsistency is minimized. The approach presented in [68] considers the case when $y_i \in \{\pm 1\}$, thereby formulating it as a discrete optimization problem and solve it using a min-cut approach. Min-cuts are however prone to degenerate solutions, and hence the objective was minimized using a mixed integer programming approach in [89], which is computationally prohibitive [73]. A continuous relaxation of this objective function, where $y_i \in [0, 1]$ has been considered in several approaches, which is solved using Markov random fields [67], Gaussian random fields and harmonic functions [74].

The proposed framework is closely related to the graph-based approaches in the sense that it utilizes the pairwise similarities for semi-supervised learning. The inconsistency measure used in the proposed approach follows a similar definition, except that an exponential cost function is used instead of a quadratic cost for violating the labels. Unlike most graph-based approaches, we create a specific classification model by learning from both the labeled and the unlabeled examples. This is particularly important for semi-supervised improvement, whose goal is to improve a given supervised learning algorithm with massive amounts of unlabeled data.

The approaches built on cluster assumption utilize the unlabeled data to regularize the decision boundary. In particular, the decision boundary that passes through the region with low density of unlabeled examples is preferred to the one that is densely surrounded with unlabeled examples. These methods specifically extend SVM or related maximum margin classifiers, and are not easily extensible to non-margin based classifiers like decision trees. Approaches in this category include transductive support vector machine (TSVM) [72], Semi-supervised Support Vector Machine (S3VM) [73], and Gaussian processes with null

category noise model [83]. The proposed algorithm, on the other hand, is a general approach which allows the choice of a base classifier well-suited to the specific task.

Finally, we note that the proposed approach is closely related to the family of ensemble approaches for semi-supervised learning. Ensemble methods have gained significant popularity under the realm of supervised classification, with the availability of algorithms such as AdaBoost [143]. The semi-supervised counter parts of ensemble algorithms rely on the cluster assumption, and prime examples include ASSEMBLE [80] and Semi-supervised MarginBoost (SSMB) [79]. Both these algorithms work by assigning a pseudo-label to the unlabeled samples, and then sampling them for training a new supervised classifier. SSMB and ASSEMBLE are margin-based boosting algorithms which minimize a cost function of the form

$$J(H) = C(y_i H(x_i)) + C(|H(x_i)|),$$

where H is the ensemble classifier under construction, and C is a monotonically decreasing cost function. The term $y_i H(x_i)$ corresponds to the margin definition for labeled samples. A margin definition involves the true label y_i, which is not available for the unlabeled samples. A pseudo-margin definition is used such as $|H(x_i)|$ in ASSEMBLE, or $H(x_i)^2$ in SSMB, thereby getting rid of the y_i term in the objective function using the fact that $y_i \in \{\pm 1\}$. However, the algorithm relies on the prediction of pseudo-labels using the existing ensemble classifier at each iteration. In contrast, the proposed algorithm combines the similarity information along with the classifier predictions to obtain more reliable pseudo-labels, which is notably different from the existing approaches. SSMB on the other hand requires the base learner to be a semi-supervised algorithm in itself [79, 80]. Therefore, it is solving a different problem of boosting semi-supervised algorithms, in contrast with the proposed algorithm.

In essence, the SemiBoost algorithm combines the advantages of graph based and ensemble methods, resulting in a more general and powerful approach for semi-supervised learning.

- Start with an empty ensemble.
- Until $\alpha < 0$, at each iteration,
  - Compute the peusdolabel (and its confidence) for each unlabeled example (using existing ensemble, and the pairwise similarity).
  - Sample the most confident pseudolabeled examples; combine them with the labeled samples and train a component classifier using the supervised learning algorithm $\mathcal{A}$.
  - Update the ensemble by including the component classifier with an appropriate weight.

Figure 3.2: An outline of the SemiBoost algorithm for semi-supervised improvement.

3.3 Semi-supervised boosting

We first describe the semi-supervised improvement problem formally, and then present the SemiBoost algorithm.

3.3.1 Semi-supervised improvement

Let $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ denote the entire dataset, including both the labeled and the unlabeled examples. Suppose that the first n_l examples are labeled, given by $\mathbf{y}_l = (y_1^l, y_2^l, \ldots, y_{n_l}^l)$, where each class label y_i^l is either $+1$ or -1. We denote by $\mathbf{y}_u = (y_1^u, y_2^u, \ldots, y_{n_u}^u)$, the imputed class labels of unlabeled examples, where $n_u = n - n_l$. Let the labels for the entire dataset be denoted as $\mathbf{y} = [\mathbf{y}_l; \mathbf{y}_u]$. Let $S = [S_{i,j}]_{n \times n}$ denote the symmetric similarity matrix, where $S_{i,j} \geq 0$ represents the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$. Let $\mathcal{A}$ denote the given supervised learning algorithm. The goal of semi-supervised improvement is to improve the performance of $\mathcal{A}$ iteratively by treating $\mathcal{A}$ like a black box, using the unlabeled examples and the pairwise similarity S. A brief outline of the SemiBoost algorithm for semi-supervised improvement is presented in Figure 3.2.

It is important to distinguish the problem of semi-supervised improvement from the existing semi-supervised classification approaches. As discussed in section 2, any ensem-

ble based algorithm must rely on the pseudo-labels for building the next classifier in the ensemble. On the other hand, graph based algorithms use the pairwise similarities between the samples, and assign the labels to unlabeled samples such that they are consistent with the similarity. In the semi-supervised improvement problem, we aim to build an ensemble classifier which utilizes the unlabeled samples in the way a graph based approach would utilize.

3.3.2 SemiBoost

To improve the given learning algorithm $\mathcal{A}$, we follow the idea of boosting by running the algorithm $\mathcal{A}$ iteratively. A new classification model will be learned at each iteration using the algorithm $\mathcal{A}$, and the learned classification models at different iterations will be linearly combined to form the final classification model.

Objective function

The unlabeled samples must be assigned labels following the two main criteria: (a) the points with high similarity among unlabeled samples must share the same label, (b) those unlabeled samples which are highly similar to a labeled sample must share its label. Our objective function $F(\mathbf{y}, S)$ is a combination of two terms, one measuring the inconsistency between labeled and unlabeled examples $F_l(\mathbf{y}, S)$, and the other measuring the inconsistency among the unlabeled examples $F_u(\mathbf{y}_u, S)$.

Inspired by the harmonic function approach, we define $F_u(\mathbf{y}, S)$, the inconsistency between class labels $\mathbf{y}$ and the similarity measurement S, as

$$F_u(\mathbf{y}_u, S) \quad = \quad \sum_{i,j=1}^{n_u} S_{i,j} \exp(y_i^u - y_j^u). \tag{3.1}$$

Many objective functions using similarity or kernel matrices, require the kernel to be positive semi-definite to maintain the convexity of the objective function (e.g., SVM). However, since $\exp(x)$ is a convex function[1], and we assume that $S_{i,j}$ is non-negative $\forall i, j$, the func-

[1]Our choice of $F(y, S)$ is a mixture of exponential loss functions, and is motivated by the traditional

tion $F_u(\mathbf{y}_u, S)$ is convex irrespective of the positive definiteness of the similarity matrix. This allows similarity matrices which are asymmetric (e.g., similarity computed using KL-divergence) without changing the convexity of the objective function. Asymmetric similarity matrices arise when using directed graphs for modeling classification problems, and are shown to perform better in certain applications related to text categorization [144].

Though this approach can work for general similarity matrices, we assume that the similarity matrix provided is symmetric. Note that Eq (3.1) can be expanded as $F_u(\mathbf{y}_u, S) = \frac{1}{2}\sum S_{j,i}\exp(y_j^u - y_i^u) + \frac{1}{2}\sum S_{i,j}\exp(y_i^u - y_j^u)$, and due to the symmetry of S, we have

$$F_u(\mathbf{y}_u, S) \;=\; \sum_{i,j=1}^{n_u} S_{i,j}\cosh(y_i^u - y_j^u), \tag{3.2}$$

where $\cosh(y_i - y_j) = (\exp(-y_i + y_j) + \exp(y_i - y_j))/2$ is the hyperbolic cosine function. Note that $\cosh(x)$ is a convex function with its minimum at $x = 0$. Rewriting Eq (3.1) using the $\cosh(.)$ function reveals the connection between the quadratic penalty used in the graph Laplacian based approaches, and the exponential penalty used in the current approach. Using a $\cosh(.)$ penalty function not only facilitates the derivation of boosting based algorithms but also increases the classification margin. The utility of an exponential cost for boosting algorithms is well known [145].

The inconsistency between labeled and unlabeled examples $F_l(\mathbf{y}, S)$ is defined as

$$F_l(\mathbf{y}, S) = \sum_{i=1}^{n_l}\sum_{j=1}^{n_u} S_{i,j}\exp(-2y_i^l y_j^u). \tag{3.3}$$

Combining Eqs (3.1) and (3.3) leads to the objective function,

$$F(\mathbf{y}, S) = F_l(\mathbf{y}, S) + CF_u(\mathbf{y_u}, S). \tag{3.4}$$

The constant C is introduced to weight the importance between the labeled and the unlabeled data. Given the objective function in (3.4), the optimal class label $\mathbf{y}_u$ is found by minimizing F.

exponential loss used in Boosting and the resulting large margin classifier. However, any convex (monotonic) loss function should work with the current framework.

Let $\hat{y}_i^l, i = 1, \cdots, n_l$ denote the labels predicted by the learning algorithm over the labeled examples in the training data. Note that in Eq (3.4), there is no term corresponding to the inconsistency between predicted labels of the labeled samples and their true labels, which would be $F_{ll} = \sum_{i=1}^{n_l} \exp(y_i^l, \hat{y}_i^l)$. Adding this term would make the algorithm reduce to AdaBoost when no unlabeled samples are present. Since in practice, there is a limited amount of labeled data available, the F_{ll} term is usually significantly smaller than F_l and F_u, and therefore it is omitted in the formulation in Eq (3.4).

selecting an even smaller subset of samples to train the classifier may not be effective. Our approach, includes the prediction on the labeled data in the form of constraints, thereby utilizing all the available labeled data at each iteration of training a classifier for the ensemble. The problem can now be formally expressed as,

$$\begin{aligned} \min \quad & F(\mathbf{y}, S) \\ \text{s.t.} \quad & \hat{y}_i^l = y_i^l, i = 1, \cdots, n_l. \end{aligned} \tag{3.5}$$

This is a convex optimization problem, and therefore can be solved effectively by numerical methods. However, since our goal is to improve the given learning algorithm $\mathcal{A}$ by the unlabeled data and the similarity matrix S, we present a boosting algorithm that can efficiently minimize the objective function F. The following procedure is adopted to derive the boosting algorithm.

- The labels for the unlabeled samples y_i^u are replaced by the ensemble predictions over the corresponding data sample.
- A bound optimization based approach is then used to find the ensemble classifier minimizing the objective function.
- The bounds are simplified further to obtain the sampling scheme, and other required parameters.

The above objective function is strongly related to several graph based approaches, manifold regularization and ensemble methods.

3.3.3 Algorithm

We derive the boosting algorithm using the bound optimization approach. An alternate, conventional way to derive the boosting algorithm using the Function Gradient method is presented in [146]. This method may also be viewed as a relaxation that approximates the original objective function by a linear function. Such an approach however, involves specification of a parametric step size. In our derivation, the step size is automatically determined thus overcoming the difficulty in determining the step-size. SemiBoost algorithm is briefly summarized in Figure 3.3.

Let $h^{(t)}(\mathbf{x}) : \mathcal{X} \rightarrow \{-1, +1\}$ denote the 2-class classification model that is learned at the t-th iteration by the algorithm $\mathcal{A}$. Let $H(\mathbf{x}) : \mathcal{X} \rightarrow \mathbb{R}$ denote the combined classification model learned after the first T iterations. It is computed as a linear combination of the first T classification models, i.e.,

$$H(\mathbf{x}) \quad = \quad \sum_{t=1}^{T} \alpha_t h^{(t)}(\mathbf{x}),$$

where α_t is the combination weight. At the $(T+1)$-st iteration, our goal is to find a new classifier $h(\mathbf{x})$ and the combination weight α that can efficiently minimize the objective function F.

This leads to the following optimization problem:

$$\begin{aligned}
\underset{h(\mathbf{x}),\alpha}{\arg\min} \quad & \sum_{i=1}^{n_l}\sum_{j=1}^{n_u} S_{i,j} \exp(-2y_i^l(H_j + \alpha h_j)) \\
+ \quad & C \sum_{i,j=1}^{n_u} S_{i,j} \exp(H_i - H_j) \exp(\alpha(h_i - h_j)) \qquad (3.6) \\
\text{s.t.} \quad & h(\mathbf{x}_i) = y_i^l, i = 1, \cdots, n_l, \qquad (3.7)
\end{aligned}$$

where $H_i \equiv H(\mathbf{x}_i)$ and $h_i \equiv h(\mathbf{x}_i)$.

This expression involves products of variables α and h_i, making it non-linear and hence difficult to optimize. The constraints, however, can be easily satisfied by including all the labeled samples in the training set of each component classifier. To simplify the compu-

tation, we construct the upper bound of the objective function, described in Proposition 1.

Proposition 1. *Minimizing Eq* (3.7) *is equivalent to minimizing the function*

$$\overline{F}_1 = \sum_{i=1}^{n_u} \exp(-2\alpha h_i) p_i + \exp(2\alpha h_i) q_i \tag{3.8}$$

where

$$p_i = \sum_{j=1}^{n_l} S_{i,j} e^{-2H_i} \delta(y_j, 1) + \frac{C}{2} \sum_{j=1}^{n_u} S_{i,j} e^{H_j - H_i} \tag{3.9}$$

$$q_i = \sum_{j=1}^{n_l} S_{i,j} e^{2H_i} \delta(y_j, -1) + \frac{C}{2} \sum_{j=1}^{n_u} S_{i,j} e^{H_i - H_j} \tag{3.10}$$

and $\delta(x, y) = 1$ *when* $x = y$ *and* 0 *otherwise.*

Proof Sketch*: By substituting* $H_i \leftarrow H_i + \alpha h_i$ *into* $F(\mathbf{y}, S)$ *and regrouping the terms, we obtain the desired result.* □

The quantities p_i and q_i can be interpreted as the confidence in classifying the unlabeled example $\mathbf{x}_i$ into the positive class and the negative class, respectively.

The expression in Eq (3.8) is difficult to optimize since the weight α and the classifier $h(x)$ are coupled together. We simplify the problem using the upper bound stated in the following proposition.

Proposition 2. *Minimizing Eq (3.8) is equivalent to minimizing*

$$\overline{F}_1 \leq \sum_{i=1}^{n_u} (p_i + q_i)(e^{2\alpha} + e^{-2\alpha} - 1) - \sum_{i=1}^{n_u} 2\alpha h_i (p_i - q_i).$$

Proof*: See [147].* □

We denote the upper bound in the above equation by $\overline{F}_2$.

Proposition 3. *To minimize* $\overline{F}_2$*, the optimal class label* z_i *for the example* $\mathbf{x}_i$ *is* $z_i = sign(p_i - q_i)$*, and the weight for sampling example* $\mathbf{x}_i$ *is* $|p_i - q_i|$*. The optimal* α *that minimizes* $\overline{F}_1$ *is*

$$\alpha = \frac{1}{4} \ln \frac{\sum_{i=1}^{n_u} p_i \delta(h_i, 1) + \sum_{i=1}^{n_u} q_i \delta(h_i, -1)}{\sum_{i=1}^{n_u} p_i \delta(h_i, -1) + \sum_{i=1}^{n_u} q_i \delta(h_i, 1)}. \tag{3.11}$$

- Compute the pairwise similarity $S_{i,j}$ between any two examples.
- Initialize $H(\mathbf{x}) = 0$
- For $t = 1, 2, \ldots, T$
  - Compute p_i and q_i for every example using Equations (3.9) and (3.10)
  - Compute the class label $z_i = \text{sign}(p_i - q_i)$ for each example
  - Sample example $\mathbf{x}_i$ by the weight $|p_i - q_i|$
  - Apply the algorithm $\mathcal{A}$ to train a binary classifier $h_t(\mathbf{x})$ using the sampled examples and their class labels z_i
  - Compute α_t using Equation (3.11)
  - Update the classification function as $H(\mathbf{x}) \leftarrow H(\mathbf{x}) + \alpha_t h_t(\mathbf{x})$

Figure 3.3: The SemiBoost algorithm

Proof Sketch*: Expression in Eq 3.11 can be obtained by differentiating $\bar{F}_2$ w.r.t α and setting it equal to 0. Observe that the above function is linear in $h_i(p_i - q_i)$ and is minimized when we choose $h_i = sign(p_i - q_i)$, for maximum values of $|p_i - q_i|$.* □

Propositions 1-3 justify the relaxations made in the derivation of the SemiBoost. At each relaxation, the "touch-point" is maintained between the objective function and the upper bound. As a result, the procedure guarantees: (a) the objective function always decreases through iterations and (b) the final solution converges to a local minimum. For more details, see [148]. Proposition 3 establishes the key ingredients required for a boosting algorithm. Using these, the SemiBoost algorithm is presented in Figure 3.3.

Let ϵ_t be the weighted error made by the classifier, where

$$\epsilon_t = \frac{\sum_{i=1}^{n_u} p_i \delta(h_i, -1) + \sum_{i=1}^{n_u} q_i \delta(h_i, 1)}{\sum_i (p_i + q_i)}.$$

As in the case of AdaBoost [146], α can be expressed as

$$\alpha_t = \frac{1}{4} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right) \tag{3.12}$$

which is very similar to the weighting factor of AdaBoost, differing only by a constant factor of $\frac{1}{2}$. Also, if AdaBoost encounters a situation where the base classifier has an error rate more than random, i.e. $\epsilon_{t+1} \geq \frac{1}{2}$, it returns the current classifier H_t. This situation has a direct correspondence with the condition in SemiBoost where the algorithm stops when $\alpha \leq 0$. From Eq (3.11) (or rather directly from Eq (3.12)), we can see that this happens only when the denominator exceeds the numerator, which means $\epsilon_{t+1} \geq \frac{1}{2}$ is equivalent to the condition $\alpha \leq 0$. However, since this condition may not be satisfied until a large number of classifiers are trained, usually there is a parameter specifying the number of classifiers to be used. It has been empirically determined that using a fixed number (usually 20) of classifiers for AdaBoost gives good performance [145].

The sampling scheme used in SemiBoost is significantly different from that of AdaBoost. AdaBoost is given the true labels of the data, and hence can proceed to increase/decrease the weights assigned to samples based on the previous iteration. In SemiBoost we do not have the true class labels for the unlabeled data, which makes it challenging to estimate the difficulty of classification. However, Proposition 2 gives us the result that selecting the most confident unlabeled data samples is optimal for reducing the objective function. Intuitively, using the samples with highly confident labeling is a good choice because they are consistent with the pairwise similarity information along with their classifications. The values of p_i and q_i tend to be large if (i) $\mathbf{x}_i$ can't be classified confidently, i.e., $|H_i|$ is small, and one of its close neighbors is labeled. This corresponds to the first term in Eq. (3.9) and Eq. (3.10). (ii) the example $\mathbf{x}_i$ is very similar to some unlabeled examples that are already confidently classified, i.e., large $s_{i,j}$ and $|H_j|$ for unlabeled example x_j. This corresponds to the second term in Eq. (3.9) and Eq. (3.10). This indicates that the similarity information plays an important role in guiding the sample selection, in contrast to the previous approaches like ASSEMBLE and SSMB, where the samples are selected to increase the value of $|H_i|$ alone.

Similar to most boosting algorithms, we can show that the proposed semi-supervised

boosting algorithm reduces the original objective function F exponentially. This result is summarized in the following Theorem.

Theorem 1. *Let $\alpha_1, ..., \alpha_t$ be the combination weights that are computed by running the SemiBoost algorithm (Fig 1). Then, the objective function at $(t+1)$st iteration, i.e., F_{t+1}, is bounded as follows:*

$$F_{t+1} \leq \kappa_S \exp\left(-\sum_{i=1}^{t} \gamma_i\right),$$

where $\kappa_S = \left[\sum_{i=1}^{n_u}\left(\sum_{j=1}^{n_l} S_{i,j} + C\sum_{j=1}^{n_u} S_{i,j}\right)\right]$ and $\gamma_i = \log(\cosh(\alpha_i))$.

Proof. Similar to the supervised Boosting algorithms, we can show that the proposed semi-supervised boosting algorithm is able to reduce the objective function exponentially. Let F_t denote the objective function at the t-th iteration. Let $\alpha_t > 0$ denote the combination weight of the tth iteration. We first show that

$$F_{t+1} \quad = \quad F_t\left(\frac{2}{\exp(2\alpha_t) + \exp(-2\alpha_t)}\right). \tag{3.13}$$

The above equality indicates that the reduction factor is $2/(\exp(2\alpha_t) + \exp(-2\alpha_t))$, which is guaranteed to be less than 1 when α_t is positive. The proof of the equality in Eq (3.13) is straightforward. According to the derivation in the previous section,

$$F_{t+1} \quad = \quad \sum_{i=1}^{n_u} \exp(-2\alpha_t h_i)p_i + \exp(2\alpha_t h_i)q_i.$$

By replacing α_t with the expression in Eq (3.11) , and by defining the quantities $\epsilon_t = \sum_{i=1}^{n_u} p_i\delta(h_i, -1) + \sum_{i=1}^{n_u} q_i\delta(h_i, 1)$, and $\eta_t = \sum_i p_i + q_i$, we have

$$F_{t+1} = 2\sqrt{(\eta_t - \epsilon_t)\epsilon_t}\ . \tag{3.14}$$

Using the fact that $F_t = \eta_t$, we have

$$F_{t+1} = 2\frac{F_t}{\eta_t}\sqrt{(\eta_t - \epsilon_t)\epsilon_t} = 2F_t\frac{\epsilon_t}{\eta_t}\sqrt{\left(\frac{\eta_t - \epsilon_t}{\epsilon_t}\right)}. \tag{3.15}$$

Using the relation $\alpha_t = \frac{1}{4}\log(\eta_t - \epsilon_t/\epsilon_t)$, we prove the equality in Eq (3.13) as follows:

$$F_{t+1} = 2F_t\frac{1}{\exp(4\alpha t) + 1}\exp(2\alpha_t) = \frac{F_t}{\cosh(2\alpha_t)}. \tag{3.16}$$

Extending this equality to F_0, we have the final expression for F_{t+1} in terms of the initial value of the objective function F_0, i.e.,

$$F_{t+1} \quad = \quad F_0 \exp\left(-\sum_{i=1}^{t} \gamma_i\right), \tag{3.17}$$

where $F_0 = \sum_{i=1}^{n_u} \sum_{j=1}^{n_l} S_{i,j}$ and $\gamma_i = \log(\cosh(2\alpha_t))$. As indicated in Eq (3.17), the objective function F is reduced exponentially as the number of iterations is increased. □

The above theorem shows that the objective function follows an exponential decay, despite the relaxations made in the above propositions.

Corollary 1 *The objective function at* $(t+1)$*st iteration is bounded in terms of the error* ϵ_t *as* $F_{t+1} \leq \kappa_S \prod_{i=1}^{t} \left(\frac{1-\epsilon_t}{\epsilon_t}\right)^{1/4}$.

Proof. The corollary can be verified by substituting Eq. (3.12) for α_i in Theorem 1. The connection between α_i and the error ϵ_t may be used to bound the objective function in terms of classification error at each iteration. From the theorem, we have $\gamma_t = \log(\cosh(\alpha_t))$. Note that,

$$\gamma_t = \log(\cosh(\alpha_t)) \geq \log(\exp(\alpha_t)) = \alpha_t.$$

Using the definition of α_t from Eq (3.12), we have $\exp(-\gamma_t) \leq \left(\frac{1-\epsilon_t}{\epsilon_t}\right)^{1/4}$. Using this inequality with the bound in the Theorem 1 results in

$$F_{t+1} \leq \kappa_S \prod_{i=1}^{t} \left(\frac{1-\epsilon_t}{\epsilon_t}\right)^{1/4}.$$

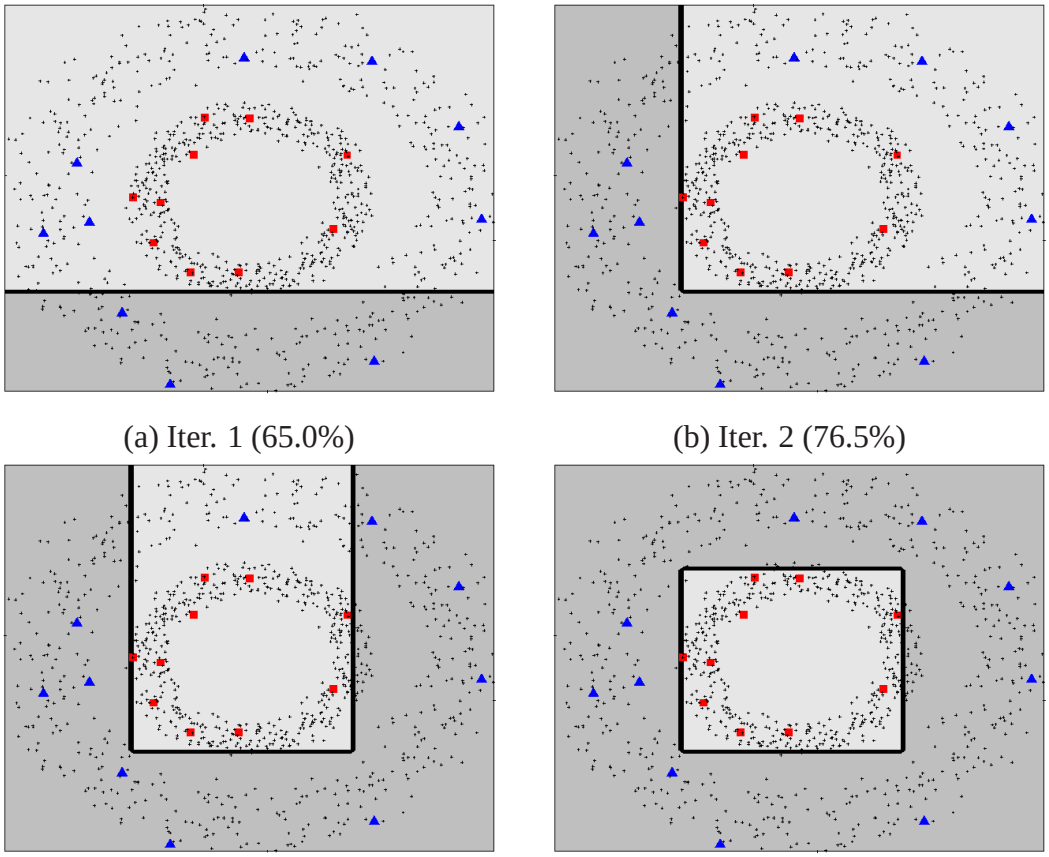
□

In the above derivation, we constrained the objective function such that the prediction of the classifier on the labeled samples must match the true labels provided. However, if the true labels are noisy, the resulting semi-supervised classifier might not perform its best. An algorithm similar to SemiBoost may be derived in such a case by including a term penalizing the solution, if the predicted labels of the labeled samples are different from the true labels. We assume that the given labels are correct, which is reasonable given the fact that there are very few labeled samples.

3.3.4 Implementation

Sampling

Sampling is the most important step in SemiBoost, just like any other boosting algorithm. The criterion for sampling usually considers the following issues: (a) How many samples



(a) Iter. 1 (65.0%) (b) Iter. 2 (76.5%)

(c) Iter. 3 (86.4%) (d) Iter. 12 (94.6%)

Figure 3.4: Decision boundary obtained by SemiBoost at iterations 1, 2, 3 and 12, on the two concentric rings dataset, using Decision Stump as the base classifier. There are 10 labeled samples per class (■,▲). The transductive performance (i.e., performance on the unlabeled data used for training) of SemiBoost is given at each iteration in parentheses.

must be selected from the unlabeled samples available for training? and (b) What is the distribution according to which the sampling must be done?

Supervised boosting algorithms like AdaBoost have the true labels available, which makes it easy to determine which samples to choose or not to choose. On the other hand, the labels assigned during the SemiBoost iteration are pseudo labels, and may be prone to errors. This suggests that we should choose only a small number of the most confident data points for SemiBoost. But selecting a small number of samples might make the convergence slow, and selecting too large a sample might include non-informative samples into the training set. The choice currently is made empirically; selecting the top 10% of the

samples seems to work well in practice. From Proposition 3, to reduce $\bar{F}_1$, it is preferable to select the samples with a large value of $|p_i - q_i|$. This selection provides highly reliable pseudo-labeled samples to the classifier. The sampling is probabilistically done according to the distribution,

$$P_s(\mathbf{x}_i) = \frac{|p_i - q_i|}{\sum_{i=1}^{n_l} |p_i - q_i|},$$

where $P_s(\mathbf{x}_i)$ is the probability that the data point $\mathbf{x}_i$ is sampled from the transduction set.

Stopping Criterion

According to the optimization procedure, SemiBoost stops when $\alpha \leq 0$, indicating that addition of that classifier would increase the objective function instead of decreasing it. However, the value of α decreases rapidly in the beginning, and eventually the rate of decrease falls down, taking a large number of iterations to actually make it negative. We currently use an empirically chosen fixed number of classifiers in the ensemble, specified as a parameter T. We set the value of $T = 20$.

Similarity Matrix

We use the Radial Basis Function similarity inspired from its success in graph based approaches. For any two samples $\mathbf{x}_i$ and $\mathbf{x}_j$, the similarity $S_{i,j}$ is computed as, $S_{i,j} = \exp(\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/\sigma^2)$, where σ is the scale parameter controlling the spread of the radial basis function. It is well known that the choice of σ has a large impact on the performance of the algorithm [74]. We set the scale parameter to the similarity values at the 10-th percentile to the 100-th percentile, varied in steps of 10, where μ_s is the average value of the similarity matrix S. Experimental results revealed that the transductive and inductive performances are stable for the chosen range of σ. This is a desirable property given the fact that choosing the right scale parameter is a difficult problem.

3.4 Results and discussion

The focus of SemiBoost is to improve any given (supervised) classifier using the unlabeled data. Therefore, our primary aim is to evaluate SemiBoost based on the improvement achieved in the inductive performance of base classifiers.

An illustration of improvement in the performance of a supervised learner (Decision Stump) using SemiBoost on the "ring-norm" dataset is shown in Figure 3.4. The dataset has 2 classes, with 500 samples each. There are 10 labeled samples per class, indicated by symbols (■, ▲). The solid line shows the decision boundary and the dark and light regions indicate the two class regions. The performance of SemiBoost at each iteration is given in parentheses below each of the plots. Figures 3.4(a)-(c) show the classifier obtained by SemiBoost at the first three iterations, and Figure 3.4(d) shows the final classifier obtained at the 12 iteration.

3.4.1 Datasets

SemiBoost was evaluated on 16 different datasets: 4 benchmark datasets provided in [71], 10 UCI datasets and 2 datasets from ethnicity classification from face images [149] and texture classification [150]. Since SemiBoost is applicable for two-class problems, we chose the two-class datasets from these benchmark datasets. The multiclass datasets in UCI are converted into two-class datasets by choosing the two most populated classes. The name of the dataset used, the classes chosen, the number of samples present in the selected classes n, and the dimensionality of the dataset d are summarized in the first column in Table 3.1. In addition to this, we also evaluated the proposed approach on text categorization problems.

The transductive performance of semi-supervised learning algorithms is well-studied [71, Chapter 21]. However, semi-supervised learning is not limited to transductive learning, and out-of-sample extensions have attracted significant attention. In fact, inductive learning is important, given that only a portion of the unlabeled samples are seen during the training phase. The real utility of learning in such cases lies in the ability to classify

unseen test samples. With this motivation, we compare the performance of SemiBoost with three state-of-the-art inductive semi-supervised algorithms: Transductive SVM [69], an inductive version of Low Density Separation (LDS) [70] and Laplacian-SVM from the Manifold Regularization approach [76]. LDS is not an inductive algorithm as it involves a graph-based dimensionality reduction step. We use the labels predicted by the LDS on the transduction set to train an inductive classifier on the original data.

3.4.2 Experimental setup

The experimental setup aims to study the improvement in performance of a supervised learner, by using unlabeled data and compare the performance of the SemiBoost algorithm with three state of the art semi-supervised learning algorithms.

We use classification accuracy as the evaluation measure. The mean and standard deviation of the accuracy are reported over 20 runs of each experiment, with different subsets of training and testing data. To measure the inductive performance, we randomly split the dataset into two halves. We call them the training and test sets. The training set has 10 labeled points along with all the given unlabeled samples. The ensemble classifier learnt by SemiBoost on the training set is evaluated by its performance on predicting the labels of the test set.

SemiBoost samples the unlabeled data, labels them at each iteration of boosting and builds a classifier $h_t(\mathbf{x})$. The number of such classifiers built will depend on the number of iterations T in boosting. T was set to 10 and we stop the boosting when weights α_t computed from Eq (3.11) become negative. We set the value of C in the objective function (Eq (3.4)) to be the ratio of number of labeled samples to the number of unlabeled samples $C = n_l/n_u$.

The first experiment studies the improvement in the performance of three different base classifiers ($h_t(\mathbf{x})$) after applying SemiBoost: Decision Stump (DS), the J48 decision tree algorithm (J48), and the Support Vector Machine with the sequential minimal optimization

(SVM) algorithm. Software WEKA [151] was used to implement all the three classifiers. All the algorithms are run with their default parameters (e.g. default C and a linear kernel was used for SVM algorithm). We chose decision trees (DS and J48) and SVM as the base classifiers because of their success in the supervised learning literature, for learning tasks.

3.4.3 Results

Choice of base classifier

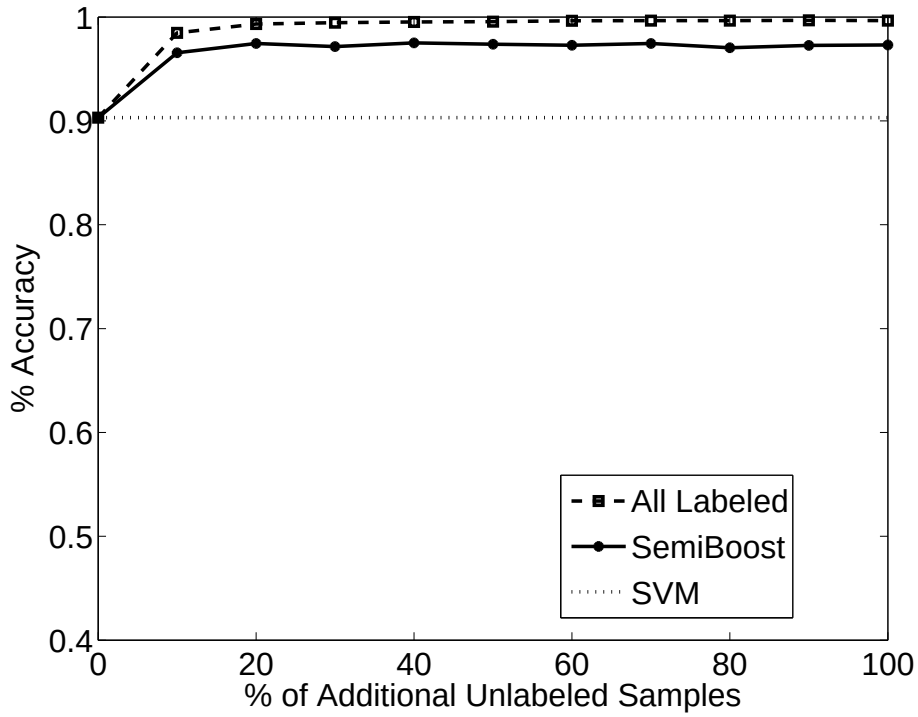
Table 3.1 compares the supervised and the three benchmark semi-supervised algorithms to the SemiBoost algorithm. The columns DS, J48 and SVM give the performances of the base classifiers on the induction set. The column SB-X gives the inductive performances of SemiBoost with base classifier X. The last three columns in Table 3.1 correspond to the inductive performances of benchmark semi-supervised algorithms TSVM, LDS and LapSVM. Note that the idea is not to build the best classifier for individual classification problem, but to show the possible improvement in the performance of supervised classifiers using SemiBoost on all the classification problems. Results indicate that SemiBoost significantly improves the performance of all the three base classifiers for nearly all the datasets. Using an independent sample paired t-test, we observed that SemiBoost significantly improved the performance of Decision Stump on 12 out of 16 datasets. The performance of J48 is improved significantly on 13 out of 16 datasets, with a significant degradation on the house dataset. For SVM, there is a significant improvement for 7 out of 16 datasets, while a significant degradation for 3 of the 16 datasets. The three cases where SVM with SemiBoost degraded, the benchmark algorithms performed poor compared to the supervised classifiers, suggesting that unlabeled data is not helpful in these cases. The ensemble classifier obtained using SemiBoost is relatively more stable, as its classification accuracy has lower standard deviation when compared to the base classifier.

Performance comparison of SemiBoost with Benchmark Algorithms

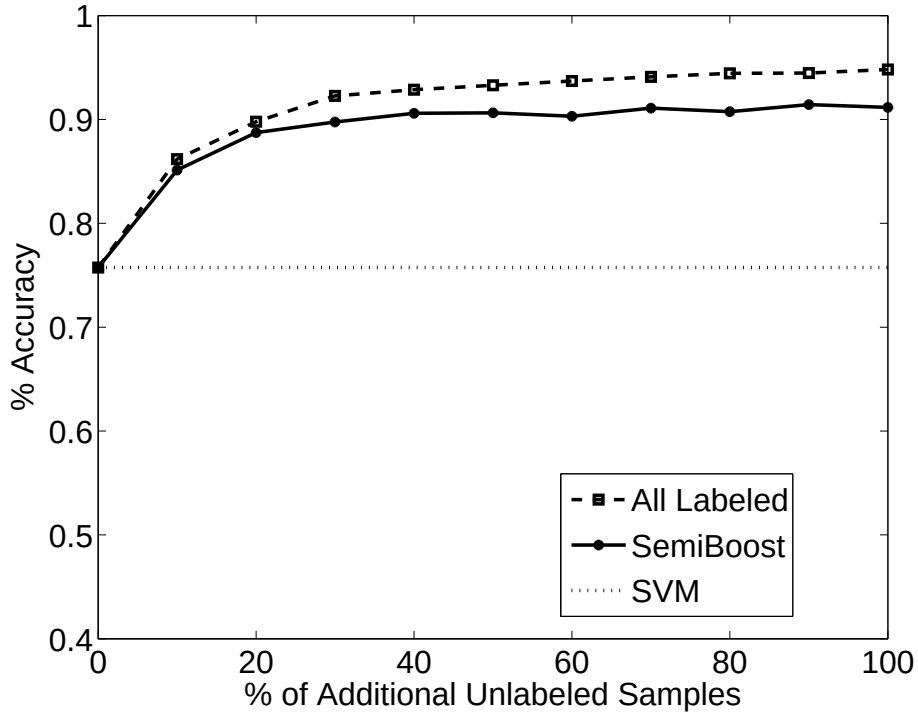
Performance of SemiBoost is compared with three different algorithms, namely TSVM, LapSVM and ILDS (inductive version of the LDS algorithm). SemiBoost achieves performance comparable to that of the benchmark algorithms. SemiBoost performs better than ILDS on almost all the datasets, and significantly better on 4 of the datasets, 2 using Decision Stump and 2 using SVM as the base classifier. SemiBoost significantly outperforms TSVM on 10 out of 16 datasets using SVM, and 8 out of 16 datasets using Decision Stump. Also, TSVM had difficulty converging on three datasets in a reasonable amount of time (20 hours). SemiBoost performs comparably to LapSVM; SB-DS outperformed LapSVM significantly on 2 datasets, and performed worse than LapSVM on 1 dataset. Similarly, SB-SVM and LapSVM significantly outperform each other on 3 out of the 16 datasets. There are datasets where one of the base classifiers outperforms SemiBoost. But in these cases, one of the base classifiers outperforms all the semi-supervised algorithms (e.g., SVM outperforms all the algorithms on COIL2, vehicle, sat and house datasets). This indicates that the unlabeled data do not always improve the base classifier, or in general, are not guaranteed to help in the learning process. When a base classifier outperforms the semi-supervised learning algorithms, we observed that the SemiBoost tends to perform close to the baseline compared to the other SSL algorithms in most cases.

Performance with respect to number of unlabeled data

Figs. 3.5(a)-(b) show the performance of SemiBoost on two of the UCI datasets. Each dataset is split into two equal parts, one for training and one for inductive testing. Ten samples in the training set are labeled. The performance of SVM, trained on the labeled data and with default parameters, on the test set is shown with a dotted line in each plot. The unlabeled examples in the training set are incrementally added to the labeled examples in units of 10%. The solid line shows the performance of the SemiBoost algorithm with addition of unlabeled data. The dashed line shows the performance obtained by the SVM



(a) optdigits (UCI)

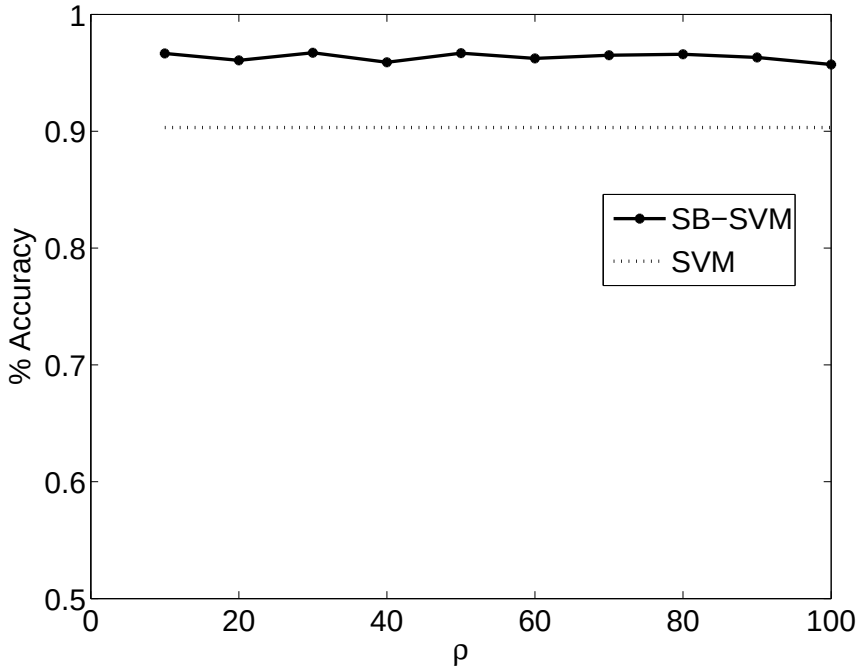


(b) wdbc (UCI)

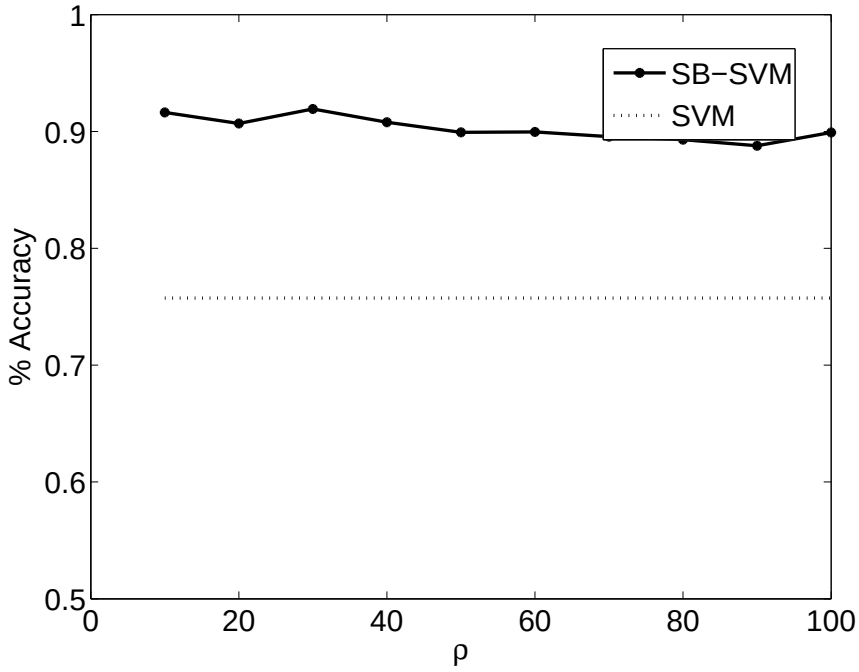
Figure 3.5: Performance of baseline algorithm SVM with 10 labeled samples, with increasing number of unlabeled samples added to the labeled set (solid line), and with increasing number of labeled samples added to the training set (dashed line).

Table 3.1: Inductive performance of SemiBoost and the three benchmark algorithms. The first column shows the dataset and the two classes chosen. The number of samples n and the dimensionality d are shown below the name of each dataset. The algorithms chosen as base classifiers for boosting are Decision Stump (DS), Decision Tree (J48) and Support Vector Machine (SVM). For each algorithm, the SB- prefixed column indicates using the SemiBoost algorithm on the base classifier. The columns TSVM, ILDS and LapSVM show the inductive performance of the three benchmark algorithms. A '-' indicates that we could not finish running the algorithm in a reasonable time (20 hours) due to convergence issues. Each entry shows the mean classification accuracy and standard deviation (in parentheses) over 20 trials.

Dataset (n, d)	DS	SB-DS	J48	SB-J48	SVM	SB-SVM	TSVM	ILDS	Lap SVM
Digit1 (1,2) (1500, 241)	57.15 (7.0)	78.09 (3.6)	57.21 (7.1)	74.97 (4.3)	74.81 (6.2)	77.89 (4.6)	79.52 (5.0)	79.53 (7.0)	74.06 (4.1)
COIL2 (1,2) (1500, 241)	55.14 (3.1)	55.84 (4.0)	54.81 (3.4)	55.27 (2.9)	59.75 (3.3)	55.42 (4.3)	50.23 (4.9)	54.62 (4.0)	55.64 (5.6)
BCI(1,2) (400, 117)	51.27 (4.2)	49.38 (2.9)	51.42 (4.1)	50.67 (3.8)	52.45 (3.1)	52.02 (4.1)	50.50 (3.6)	50.73 (2.4)	54.37 (3.6)
g241n (1,2) (1500, 241)	50.73 (3.1)	54.54 (2.8)	50.57 (2.9)	54.71 (2.5)	57.55 (2.6)	57.93 (3.4)	51.14 (3.5)	50.25 (1.5)	53.65 (3.1)
austra (1,2) (690, 15)	60.39 (13.0)	73.46 (7.9)	60.12 (12.7)	73.36 (7.4)	65.64 (8.2)	71.36 (8.8)	73.38 (12.6)	66.00 (14.5)	74.38 (8.7)
ethn (1,2) (2630, 30)	65.72 (8.6)	66.42 (6.4)	64.98 (7.9)	63.98 (5.3)	67.04 (4.8)	67.57 (5.7)	-	67.16 (16.7)	74.60 (5.8)
heart (1,2) (270, 9)	68.26 (14.3)	79.48 (3.6)	67.67 (15.0)	78.78 (3.8)	70.59 (7.9)	79.00 (4.1)	77.63 (6.6)	77.11 (9.6)	77.96 (4.8)
wdbc (1,2) (569, 14)	79.47 (16.3)	88.98 (6.5)	75.95 (17.1)	89.82 (4.0)	75.74 (9.7)	88.82 (9.9)	86.40 (8.6)	85.07 (8.7)	91.07 (3.4)
vehicle (2,3) (435, 26)	60.48 (7.6)	69.31 (6.7)	60.89 (8.1)	70.25 (7.7)	78.28 (6.2)	72.29 (9.4)	63.62 (8.6)	66.28 (8.5)	71.38 (6.7)
texture (2,3) (2026, 19)	95.67 (5.6)	98.90 (0.6)	89.46 (6.7)	98.50 (0.9)	98.44 (1.4)	99.91 (0.1)	-	98.38 (7.2)	99.11 (0.92)
image (1,2) (660, 18)	89.64 (11.2)	100.00 (0.0)	87.03 (9.3)	99.79 (0.3)	99.92 (0.2)	100.00 (0.0)	91.91 (8.2)	100 (0.0)	99.95 (0.2)
isolet (1,2) (600, 51)	64.23 (12.7)	91.92 (2.1)	64.48 (12.8)	90.20 (3.4)	89.58 (5.3)	95.12 (2.3)	90.38 (8.0)	92.07 (11.4)	93.93 (3.4)
mfeat (1,2) (400, 76)	82.25 (2.6)	96.25 (2.0)	85.90 (12.9)	96.00 (1.8)	98.78 (1.1)	99.85 (0.3)	95.32 (7.5)	96.5 10.8	100.00 (0.0)
optdigits (2,4) (1143, 42)	65.91 (13.4)	93.22 (3.0)	65.59 (13.1)	93.33 (2.6)	90.31 (3.6)	96.35 (2.4)	92.34 (9.0)	96.40 (11.1)	98.34 (2.4)
sat (1,6) (3041, 36)	82.77 (5.5)	85.99 (3.7)	83.80 (6.1)	86.55 (3.0)	99.13 (0.7)	87.71 (2.9)	-	94.20 (14.2)	99.12 (0.5)



(a) optdigits (UCI)



(b) wdbc (UCI)

Figure 3.6: Performance of baseline algorithm SVM with 10 labeled samples, with increasing value of the parameter σ. The increments in σ are made by choosing the ρ-th percentile of the similarities, where ρ is represented on the horizontal axis.

when all these added samples are labeled using their ground truth. It is observed that the performance of SemiBoost improves with the addition of more and more unlabeled data, whenever such an improvement is possible.

Sensitivity to parameter σ

Fig. 3.6 shows the performance of the SemiBoost-SVM, with varying value of the parameter σ. The parameter σ was chosen to be the ρ-th percentile of the distribution of similarities, with ρ varying between 10-th percentile to 100-th percentile. Selecting the value of σ is one of the most difficult aspects of graph construction, and several heuristics have been proposed to determine its value. On most of the datasets shown, SemiBoost is relatively stable with respect to the scale parameter. However, a choice of σ between 10-th percentile to 20-th percentile of the pairwise distances is recommended, based on empirical observations.

Margin and Confidence

In this experiment we empirically demonstrate that SemiBoost has a tendency to maximize the mean-margin. For unlabeled data, a popular definition of margin is $|H(\mathbf{x}_i)|$ [80, 79]. The mean margin is the empirical average of $|H(\mathbf{x}_i)|$ over the unlabeled data used for training. Figs. 3.7, 3.8, and 3.9 show the mean-margin value on optdigits dataset (classes 2,4) over the iterations using Decision Stump, J48 and SVM as the base classifiers, respectively. The value of the mean-margin increases over the iterations, irrespective of the choice of the base classifier. However, it is important to note that the minimum margin may not increase at each iteration, although the test error decreases. When the training data consists of a small number of labeled samples which can be perfectly classified, the margin is largely decided by the unlabeled data. Considering the margin over the unlabeled data, the classifier at iteration 1 has a margin of α_1 for all the unlabeled data, wheres at the second iteration, the minimum margin is $\min_{\mathbf{x}_i} |H^{(2)}(\mathbf{x}_i)| = |\alpha_1 - \alpha_2| \leq \alpha_1 = \min_{\mathbf{x}_i} |H^{(1)}(\mathbf{x}_i)|$. In fact, over the iterations, the value of the minimum margin may be traded off to obtain

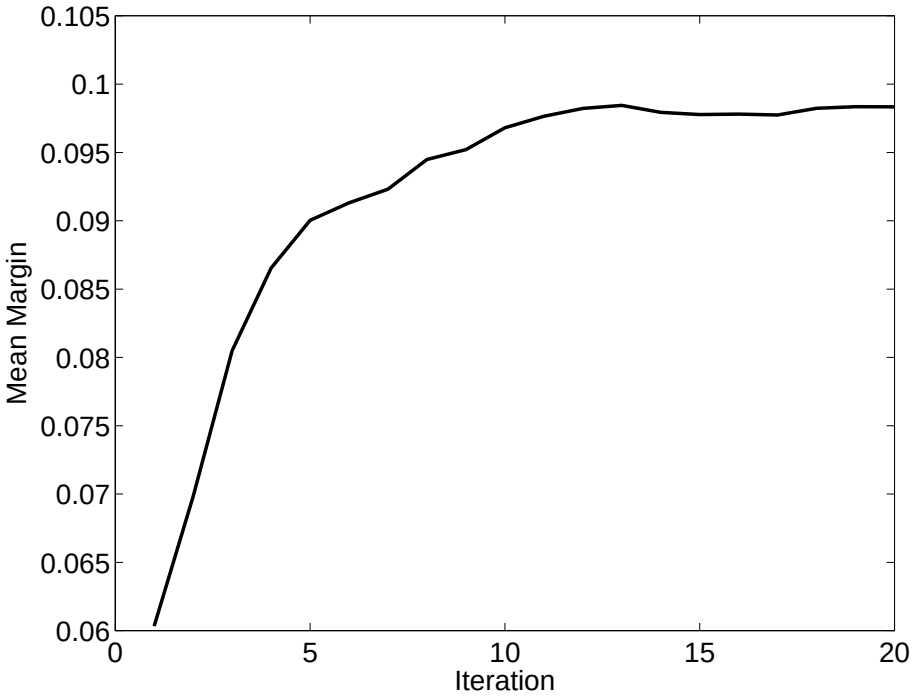


Figure 3.7: The mean-margins over the iterations, on a single run of SemiBoost on optdigits dataset (classes 2,4), using Decision Stump as the base classifier.

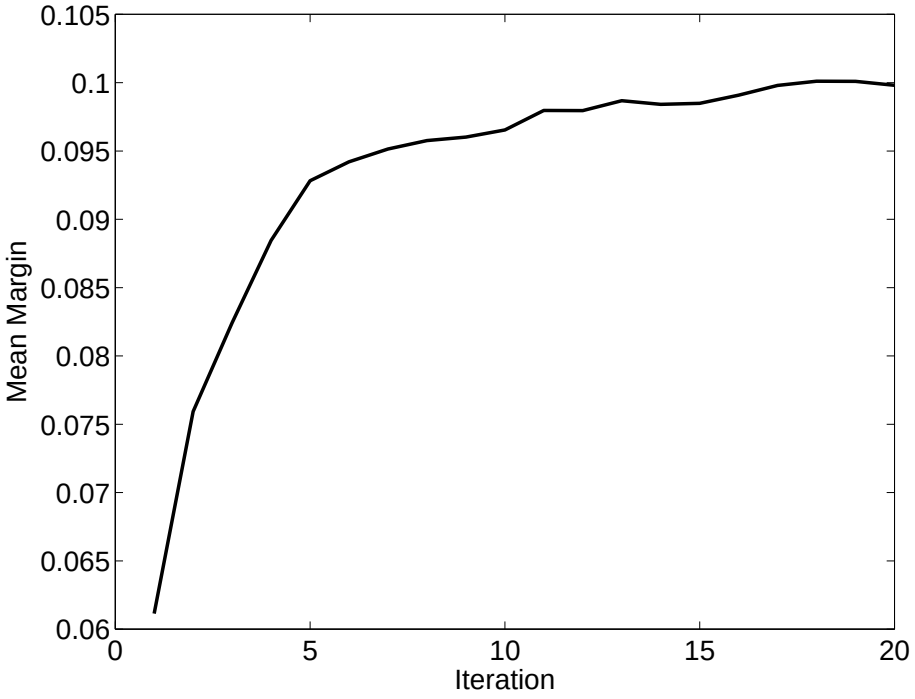


Figure 3.8: The mean-margins over the iterations, on a single run of SemiBoost on optdigits dataset (classes 2,4), using J48 as the base classifier.

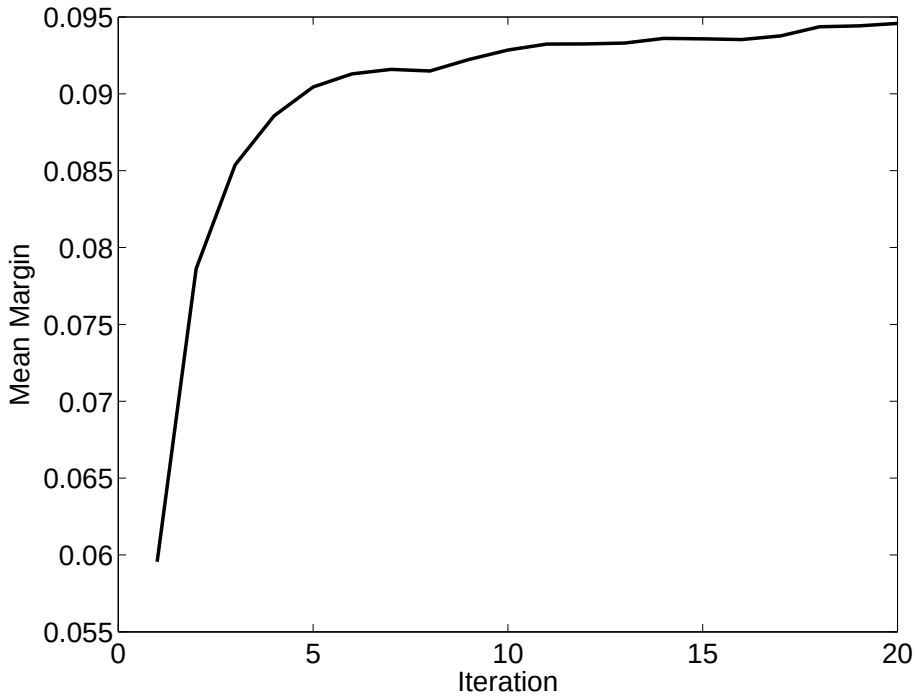


Figure 3.9: The mean-margins over the iterations, on a single run of SemiBoost on optdigits dataset (classes 2,4), using SVM as the base classifier.

a gain in the performance, i.e. being in agreement with the similarity matrix. It has been shown in [152] that maximizing the value of minimum margin does not necessarily translate to a better performance of a classifier. It is argued in the context of boosting that an approach that maximizes the mean-margin in a greedy fashion is preferable to those that maximize the minimum margin. Fig. 3.11 shows the distribution of the value of $H(\mathbf{x}_i)$ over the iterations. The light and dark bars in the histogram represent the two classes (2 and 4), in the optdigits dataset. Note that as iterations progress, the two classes get more and more separated.

3.4.4 Convergence

According to Theorem 1, SemiBoost converges exponentially. To illustrate the convergence, we chose the two most populous classes in the optdigits dataset, namely digits 2 and 4. The change in the objective function as new classifiers are added over iterations is

demonstrated in Fig. 3.13. which follows an exponential reduction. Fig. 3.12 shows the value of α over the iterations. Initially, the value of α falls rapidly, and after around 20 iterations, the value is insignificantly small relative to that of initial classifiers. This suggests that although SemiBoost still needs more iterations to converge, the new classifiers added in boosting will not significantly change the decision value. Fig. 3.14 shows the accuracy of the SemiBoost with Decision Stump as the base classifier, over the iterations.

3.4.5 Comparison with AdaBoost

Table 3.2: Performance of different classifiers and their boosted versions on 6 UCI datasets. X-small stands for the classifier trained on a set of 10 labeled samples chosen from the data. The prefix AB-X stands for AdaBoost with base classifier X. SB-X stands for SemiBoost with base classifier X. X-large stands for the classifier trained by labeling all the unlabeled data used in SB-X.

Classifier		austra	bupa	wdbc	optdigits	mfeat-fou	isolet
Decision Stump	small	60.39	54.94	79.47	65.91	82.25	64.23
	AB-small	62.55	56.45	70.02	63.20	77.62	64.82
	SemiBoost	73.46	55.78	88.98	93.22	96.25	91.92
	large	79.36	57.71	90.42	90.26	99.72	92.57
	AB-large	81.70	68.44	94.44	99.98	99.72	97.68
J48	small	60.12	54.97	75.95	65.59	85.90	64.48
	AB-small	60.68	55.09	68.86	61.40	75.80	65.33
	SemiBoost	73.36	54.74	89.82	93.33	96.00	90.20
	large	79.97	62.49	92.68	97.18	99.12	92.90
	AB-large	82.42	66.21	94.96	98.97	99.12	96.68
SVM	small	65.64	52.05	75.74	90.31	98.78	89.58
	AB-small	63.29	53.50	73.53	87.11	93.80	88.48
	SemiBoost	71.36	54.02	88.82	96.35	99.85	95.12
	large	85.57	58.15	94.81	99.66	100.00	99.72
	AB-large	84.29	65.64	95.89	99.65	100.00	99.72

To evaluate the contribution of unlabeled data in improving the performance of a base classifier, we compared the performance of SemiBoost with that of AdaBoost on the same base classifier (or weak learner) and using a similar experimental procedure as in Sec-

tion 3.4.2. Table 3.2 shows the performance of three base classifiers Decision Stump, J48 and SVM (shown in column 1) on 6 datasets shown in the top row. For each classifier, the first two rows show the inductive performance of the classifier and its boosted version (using AdaBoost) trained on 10 labeled samples. The third row shows the performance of SemiBoost when unlabeled data is added to the same set of labeled samples. The fourth and fifth rows, labeled *large* and *AB-large* show the performance of the classifier and its boosted version trained after labeling the unlabeled data used in SemiBoost.

From Table 3.2, we can see that the performance of SemiBoosted versions of the classifiers (SB-DS, SB-J48, and SB-SVM) is significantly better than classifiers trained using only labeled data, boosted (using AdaBoost) or unboosted (rows 1 and 2 for each classifier section). Naturally, when all the unlabeled data are labeled, the performance of the classifiers and their boosted versions are significantly better than SemiBoost (rows 4 and 5). The reduction in the inductive performance of AB-small compared to the base classifier on several datasets may be attributed to the overfitting due to small number of training samples. The addition of unlabeled data as a regularizing mechanism in SemiBoost avoids the overfitting, thereby achieving an improved classifier.

3.5 Performance on text-categorization

We further evaluate the SemiBoost algorithm on the Text Categorization problem using the popular 20-newsgroups dataset[2]. We performed the evaluation of SemiBoost algorithm with Decision Stump, J48 and SVM as the base classifiers on binary problems created using the 10 most popular classes of the 20-newsgroups dataset. Note that this experimental setup is different from some of the other studies in semi-supervised learning in which the one-vs-rest approach is used for evaluation. Compared to one-vs-rest, the one-vs-one evaluation has the following advantages:

[2]http://people.csail.mit.edu/jrennie/20Newsgroups/

Table 3.3: Comparison of the inductive performance (measured as % accuracy) of SemiBoost, TSVM, ILDS and LapSVM on pairwise binary problems created from 5 classes of the 20-newsgroups dataset.

Classes (d)	DS	SB DS	J48	SB J48	SVM	SB SVM	TSVM	ILDS	LapSVM
1, 2 (3736)	55.82 (14.0)	82.30 (12.3)	56.93 (15.5)	72.86 (8.0)	71.02 (8.7)	70.74 (5.6)	75.44 (13.2)	55.10 (16.6)	68.23 (3.9)
1, 3 (3757)	54.61 (10.6)	85.95 (9.6)	56.24 (10.7)	77.05 (8.0)	72.17 (8.2)	74.83 (5.4)	89.34 (5.9)	58.88 (20.2)	71.34 (4.8)
1, 4 (3736)	51.35 (7.1)	87.36 (11.4)	54.71 (13.4)	80.65 (5.7)	77.22 (9.0)	78.47 (3.6)	88.71 (6.8)	61.72 (9.4)	74.67 (3.1)
1, 5 (3979)	55.72 (11.4)	91.37 (7.8)	57.55 (12.6)	86.65 (6.7)	74.84 (9.8)	82.64 (4.3)	92.35 (5.5)	66.45 (16.7)	78.01 (4.1)
2, 3 (4154)	48.94 (2.3)	73.33 (11.6)	49.43 (2.1)	64.52 (7.8)	63.12 (5.2)	64.06 (4.5)	66.05 (10.6)	50.76 (1.8)	61.68 (3.8)
2, 4 (4143)	49.60 (4.0)	88.43 (9.5)	49.40 (3.8)	78.07 (5.0)	69.47 (7.0)	74.85 (3.2)	81.50 (13.5)	50.32 (2.1)	70.95 (3.2)
2, 5 (4406)	49.38 (1.9)	94.65 (6.5)	49.24 (1.6)	83.87 (6.0)	71.62 (6.6)	80.12 (4.8)	84.94 (12.4)	53.94 (7.3)	74.79 (3.4)
3, 4 (4130)	51.16 (3.1)	90.22 (8.6)	51.46 (3.7)	77.34 (7.2)	72.22 (5.4)	75.26 (5.1)	81.98 (12.7)	50.08 (2.7)	71.45 (3.8)
3, 5 (4426)	51.67 (4.0)	92.93 (6.5)	51.71 (4.9)	81.16 (7.0)	73.65 (8.3)	78.31 (3.9)	77.38 (16.2)	53.83 (8.1)	74.91 (4.2)
4, 5 (4212)	51.68 (4.1)	79.51 (11.5)	51.53 (3.9)	68.27 (8.4)	62.08 (5.8)	68.07 (5.3)	67.54 (12.7)	52.39 (6.5)	65.05 (5.0)

- There is a large variation in the best performing supervised classifier for the binary tasks. This enables us to show that when SVM is not the best classifier for a problem, then the methods that improve SVM using unlabeled data may not be the best semi-supervised algorithms to use.

- Semi-supervised learning algorithms rely on certain assumptions about the structure of the data and the classes. In one-vs-rest approaches, these assumptions are likely to be violated. For instance, many semi-supervised algorithms assume a large cluster gap between the two classes. By aggregating multiple classes into one negative class,

we expect to see a large cluster gap amongst the negative class itself. Violation of the manifold assumption can be explained similarily.

- There is a large imbalance in the data in a one-vs-rest classification. While a knowledge of priors may be used to incorporate this imbalance into semi-supervised learning to improve the performance, we assume no prior information is available about the data other than the similarity information and a small number of training examples.

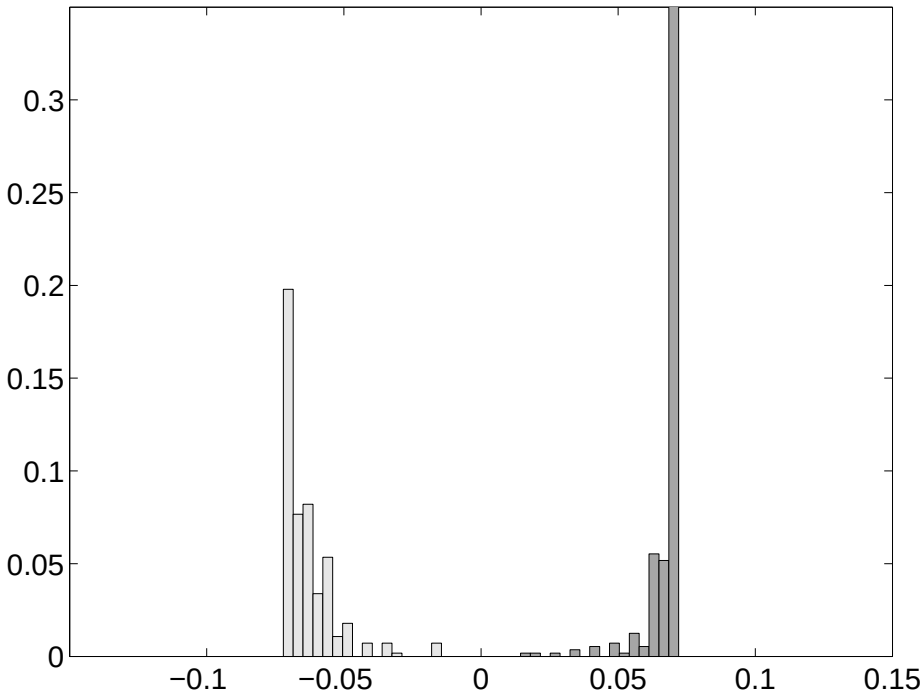
- One-vs-one has been a popular approach for creating multiclass classifiers. The testing time can be significantly reduced in a one-vs-one setting by using a DAG based architecture [153].

We generate all the 45 possible binary problems of the 10 classes. For simplicity, we include only results on 10 binary problems created from 5 of the classes in the 20 newsgroups, summarized in Table 3.3. These results are similar to the results on the other 35 binary problems.The first column in Table 3.3 shows the classes chosen for creating the binary problems. Each classification task contains a dataset with approximately 2,000 documents. We use the popular tf-idf features computed over the words which occur at least 10 times in total, in all the 2,000 documents. The tf-idf features are later normalized per document. The dimensionality of each dataset is shown in column 2 of Table 3.3. We follow the same inductive evaluation procedure as in Section 3.4.2. We use 2 labeled samples per class for training the classifier. We use the linear kernel (dot product between the feature vectors) as the similarity measure, popular in the text classification tasks. The inductive performance of the different algorithms Decision Stump, J48, SVM and their SemiBoosted versions, Transductive SVM, Inductive LDS, Laplacian SVM are shown in Table 3.3. To allow a fair comparison, the parameter value C for all SVMs is set to 1. The mean and standard deviation of the performance over 20 runs of the experiment are reported.

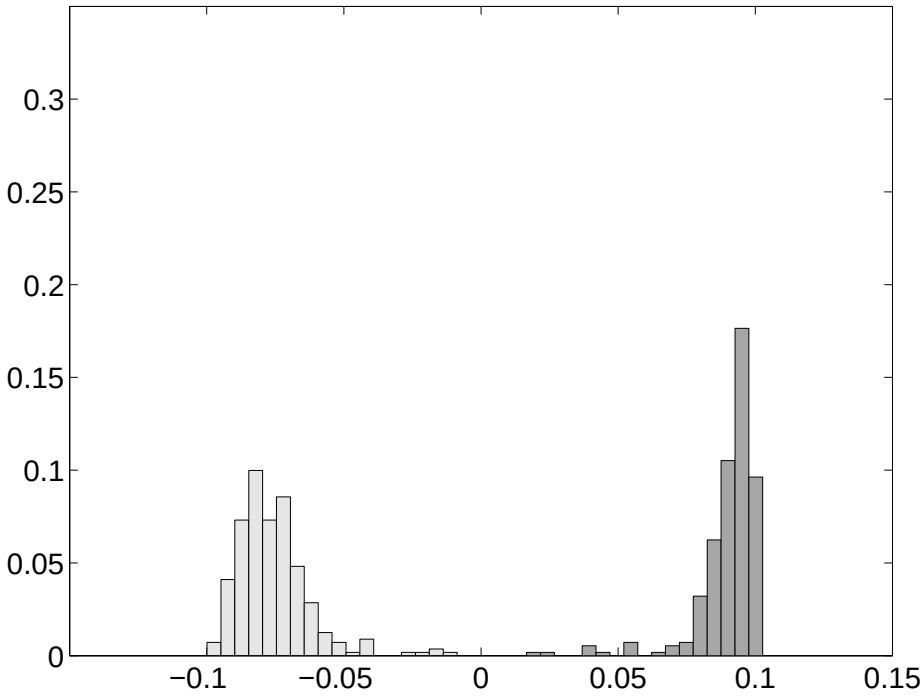
Table 3.3 shows that in the case of Decision Stump and J48, SemiBoost significantly (at a 95% confidence level, measured using independent sample paired t-test) improves the performance on all the pairs of classes. The performance of SVM is improved significantly on 5 out of the 10 class pairs. Also, we notice that SemiBoosted Decision stump performs signficantly better than SemiBoosted SVM on all the pairs of classes. Comparing the SVM based methods, SB-SVM significantly outperforms LapSVM on 7 class pairs and ILDS on all the 10 pairs. TSVM outperforms SB-SVM on 5 out of the 10 class pairs. Overall, the performance of SB-SVM is comparable to TSVM and it is significantly better than LapSVM and ILDS. SB-DS outperforms TSVM significantly on 5 out of the 10 class pairs, and LapSVM and ILDS on all the class pairs. The poor performance of ILDS may be ascribed to the use of a graph based kernel, which may not be as suitable for text classification based tasks as a linear kernel. These results show that SemiBoosting Decision Stumps is a viable alternative to the SVM based semi-supervised learning approaches.

3.6 Conclusions and future work

An algorithm for semi-supervised learning using a boosting framework is presented. The strength of SemiBoost lies in its ability to improve the performance of any given base classifier in the presence of unlabeled samples. Overall, the results on both UCI datasets and the text categorization using 20-newsgroups dataset demonstrate the feasibility of this approach. The performance of SemiBoost is comparable to the state-of-the-art semi-supervised learning algorithms. The observed stability of SemiBoost suggests that it can be quite useful in practice. SemiBoost, like almost all other semi-supervised classification algorithms, is designed for two-class classification. Multiclass extension of SemiBoost is presented in [154]. We are working towards obtaining theoretical results that will guarantee the performance of SemiBoost, when the similarity matrix reveals the underlying structure of data (e.g., the probability that two points may share the same class).

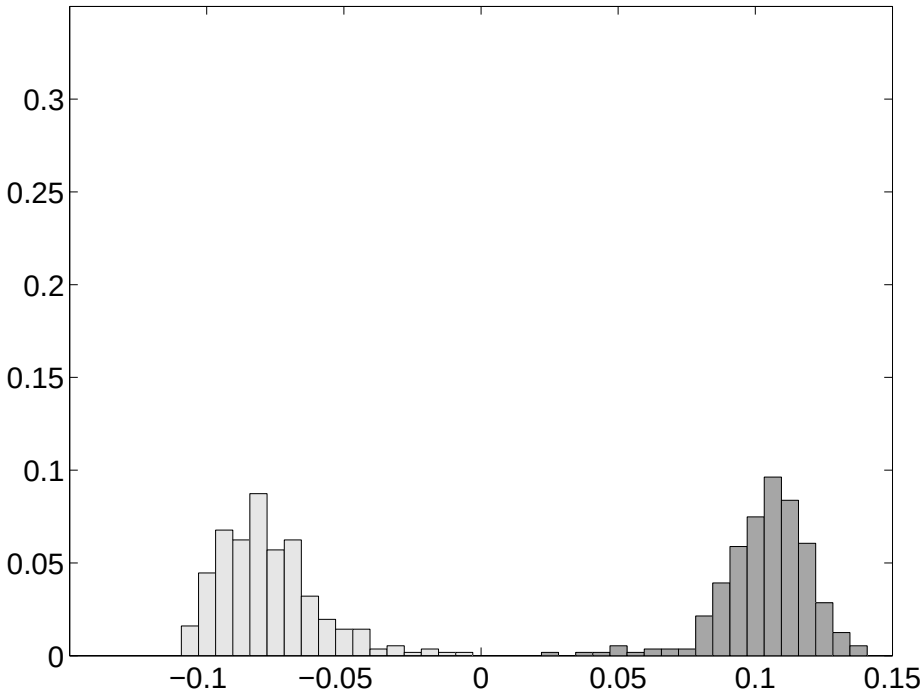


(a) Iteration 1

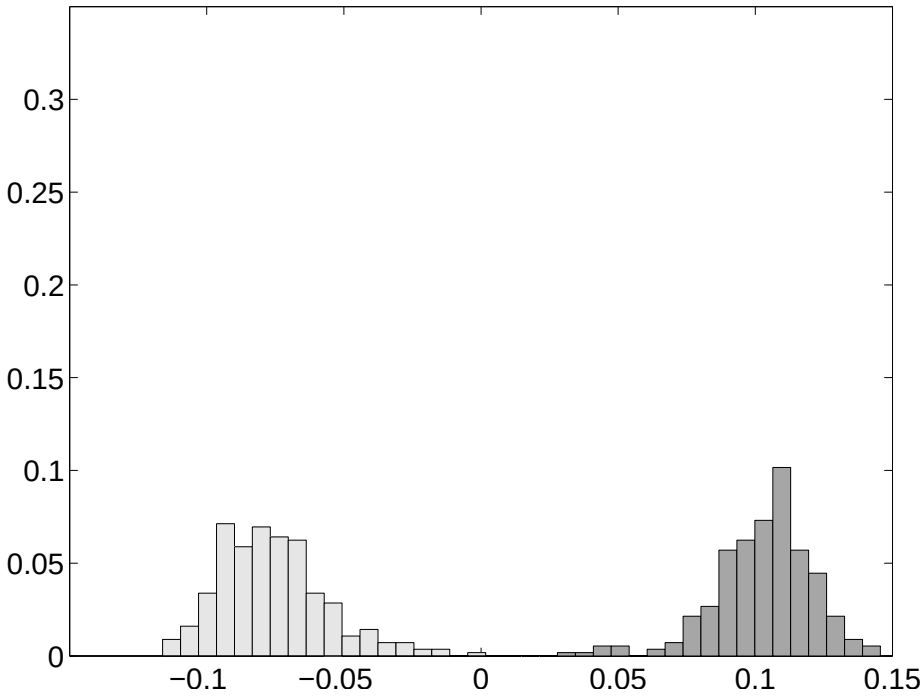


(b) Iteration 2

Figure 3.10: Distribution of the ensemble predictions $H_t(\mathbf{x}_i)$, over the unlabeled samples in the training data from optdigits dataset (classes 2,4) at the iteration t, where $t \in \{1, 2, 10, 20\}$. SVM is used as the base classifier. The light and dark bars in the histogram correspond to the two classes 2 and 4, respectively.



(c) Iteration 10



d) Iteration 20

Figure 3.11: Continued from previous page.

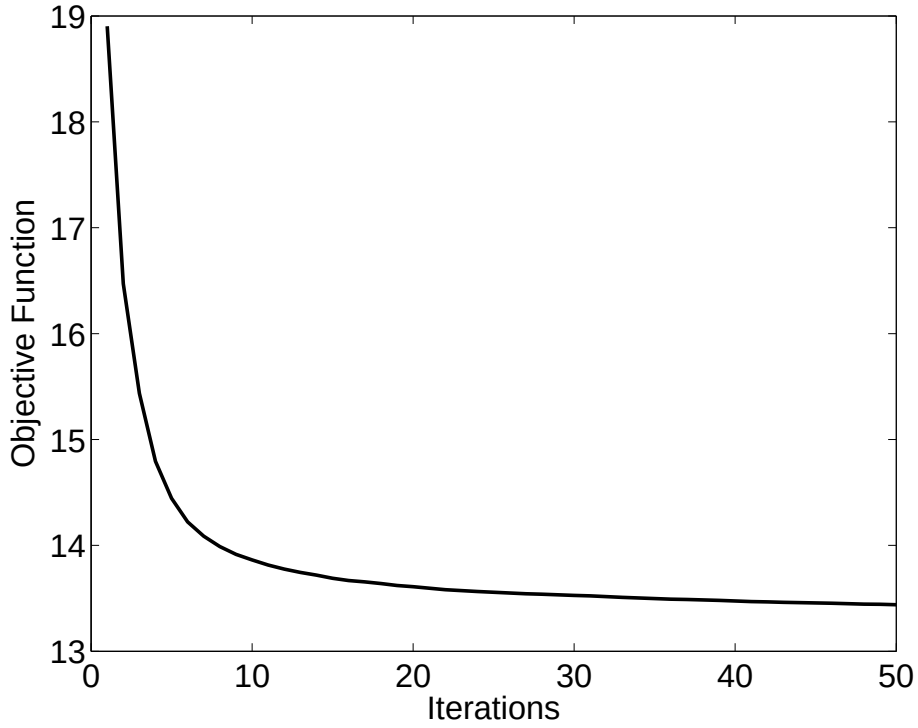


Figure 3.12: Objective function of SemiBoost over the iterations, when run over two classes (2,4) of the optdigits dataset using Decision Stump as the base classifier.

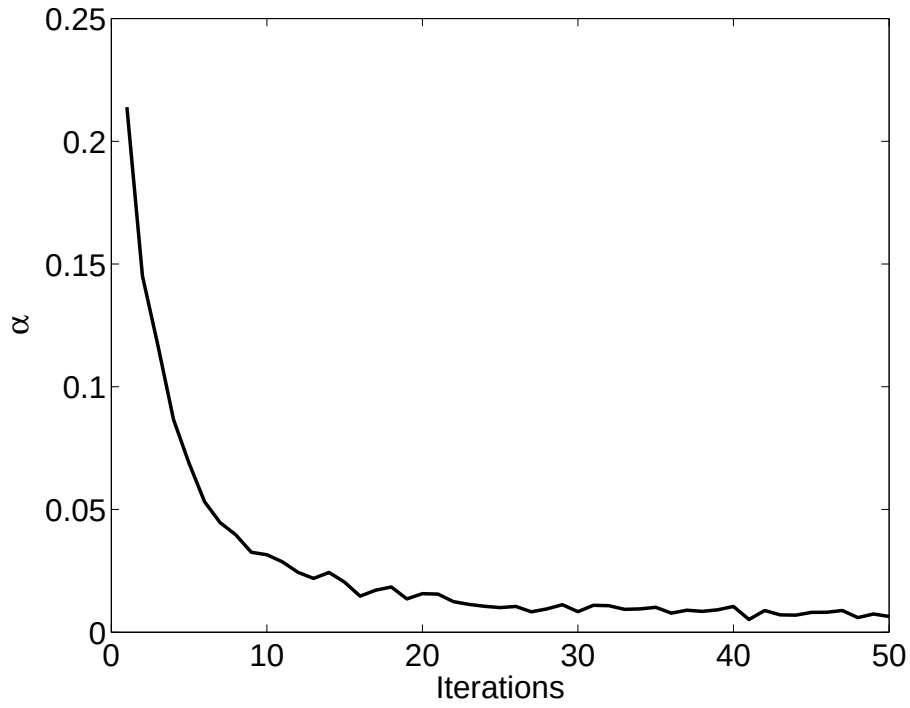


Figure 3.13: The classifier combination weight α of SemiBoost over the iterations, when run over two classes (2,4) of the optdigits dataset using Decision Stump as the base classifier.

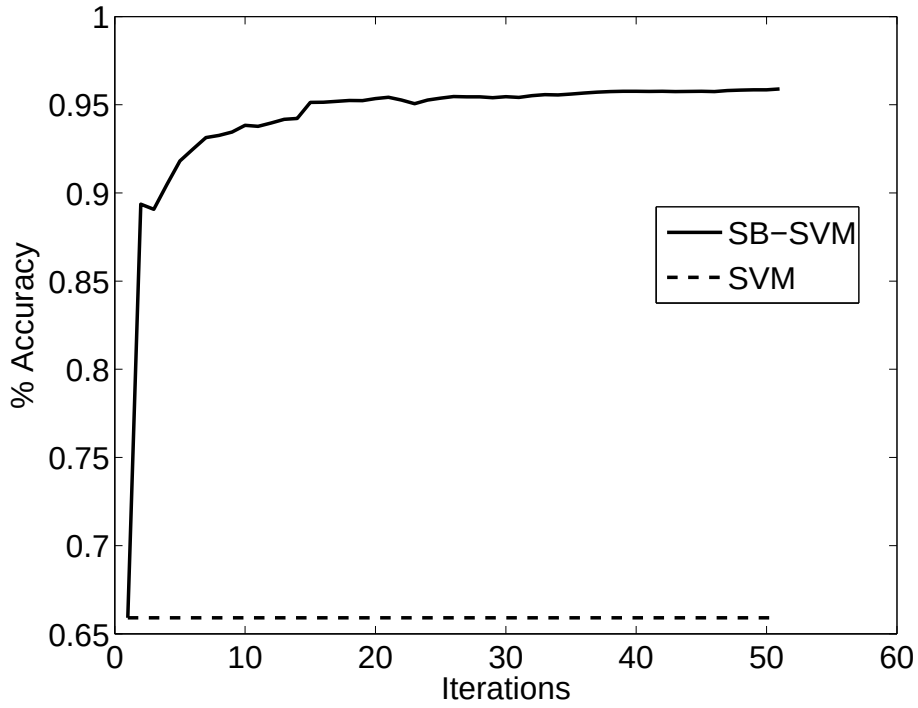


Figure 3.14: Accuracy of SemiBoost over the iterations, when run over two classes (2,4) of the optdigits dataset using Decision Stump as the base classifier. The accuracy of the base classifier is 65.9%.

CHAPTER 4

Non-parametric Mixtures for Clustering

4.1 Introduction

Clustering is applicable to many central problems in data analysis, specifically in computer vision such as image segmentation [155, 46], clustering images [156], visual word construction for image annotation [157], motion segmentation [158], image retrieval [159], and visual object recognition [160].

The lack of a universal definition of a cluster, and its task or data dependent nature has resulted in publication of a very large number of algorithms, each with slightly different assumptions about the cluster structure. A brief categorization of the existing algorithms was presented in Table 2.1, and some of their major properties are listed in Table 4.1. Broadly, the proposed approaches can be classified into *parametric* vs. *non parametric* approaches. Parametric approaches impose a structure on the data, whereas non-parametric methods infer the underlying structure from the data itself. Since parameteric models assume a specific structure in the dataset (e.g. K-means prefers spherical clusters), they tend to be less flexbile compared to the non-parameteric approaches.

Probabilistic models are highly effective when the underlying distribution of the data is either known, or can be closely approximated by the distribution assumed by the model. One of the most widely used probabilistic clustering methods is the finite mixture modeling [35, 36]. Several probabilistic models like Gaussian Mixture Models (GMM) [36]

and Latent Dirichlet Allocation [37] have been shown to be successful in a wide variety of applications concerning the analysis of continuous and discrete data, respectively. Probabilistic models are advantageous since they allow soft cluster memberships and provide a principled way to address issues like the number of clusters, missing feature values, etc. In addition, the probabilistic memberships are often used as an alternative representation of objects, leading to effective dimensionality reduction. The most well known example of this case is the Probabilistic Latent Semantic Indexing (PLSI) [54], which is widely used in the dimensionality reduction of text data. The main shortcoming of most mixture models is the assumption that data is generated from a finite mixture of parametric distributions e.g., Gaussian, multinomial etc. However, it is well known that clusters in real data are not always of the same shape and rarely follow a "nice" distribution like Gaussian [4]. In a general setting, each cluster may follow its own distribution that is unknown a priori. Therefore, there is a need for algorithms that are more flexbile in terms of their assumptions such that they are able to detect clusters of arbitrary shapes.

The limitations of parametric mixture models can be overcome by the use of non-parametric density estimation methods. Several algorithms, such as Mean-shift [45], DENCLUE [50] and DBSCAN [49] were developed to exploit non-parametric density estimates for data clustering. These methods find a single kernel-density estimate of the data, and detect clusters by identifying modes or regions of high density in the estimated density [45]. Despite their success, most of these approaches do not perform well consistently on high-dimensional datasets. The performance of DBSCAN is highly dependent on the parameters that are used to define the neighborhood. Since it is difficult to define the neighborhood of the data points in a high-dimensional space due to the curse of dimensionality [21], DBSCAN performs rather poorly in clustering high dimensional data; applications of DBSCAN cluster data with dimensionality upto 5 [161]. Further, many of these methods require specifying appropriate values for some parameters that often need to be decided in a rather ad-hoc manner. Finally, for most of these density estimation based approaches, it

is not possible to specify the number of clusters. Although this property may be viewed as an advantage since the number of clusters is determined implicitly, it is clearly a disadvantage as any prior information about the number of clusters can not be incorporated into the clustering algorithm. A hybrid clustering approach to segment images using a combination of parametric and non-parametric mixture models was presented in [46], which can incorporate the number of clusters into the non-parametric model.

In this chapter, an extension of the non-parametric density estimation to the mixture models for data clustering is presented. It is assumed that each cluster is generated by its own density that is unknown. The density of each cluster may be arbitrary and multimodal and hence it is modeled using a non-parametric kernel density estimate. The overall data is modeled as a mixture of the individual cluster densities. Since the NMM algorithm, unlike other non-parametric algorithms (e.g., Spectral clustering), constructs an explicit probabilistic model for each cluster, it can naturally handle out-of-sample[1] clustering by computing the posterior probabilities for new data points. Tables 4.2 and 4.4 compare the NMM algorithm to several well known clustering algorithms. In summary, we emphasize that:

- The NMM algorithm for data clustering offers several advantages compared to the other non-parametric approaches (e.g., hierarchical clustering, spectral clustering, etc.) that are not based on probabilistic models: (a) it allows for probabilistic assignments of data points to different clusters (b) it can effectively explore probabilistic tools such as Dirichlet process and Gaussian process for non-parametric priors, and (c) the model naturally supports out of sample cluster assignments

- Contrary to most existing mixture models, the NMM approach does not make any explicit assumption about the parametric form of the underlying density function, and therefore is flexible in modeling arbitrarily shaped clusters.

[1] A clustering algorithm can perform *out-of-sample* clustering if it can assign a cluster label to a data point unseen during the learning phase.

Table 4.1: Properties of different clustering methods. Prototype-based clustering methods are approaches that use a single data point to represent each of the clusters (e.g. K-means uses centroid of the cluster to represent each cluster)

Method/Family	Non-parametric	Prototype-based [4]	Out Of Sample	Output
Squared-Error	No	Yes	Yes	Labels
Parametric mixture models	No	Yes	Yes	Probabilities
Non-parametric density estimation	Yes	No	No	Labels
Spectral Algorithms	Yes	No	No	Scores
Hierarchical	Yes	No	No	Labels
Information Theoretic	Yes	Some	Some	Labels
Non-parametric Mixture	Yes	No	Yes	Probabilities

The performance of the NMM algorithm is shown on a large number of text and UCI datasets. Experimental results demonstrate that, compared to several widely used clustering algorithms such as K-means and spectral clustering, the NMM algorithm performs significantly better when data is of high dimensionality, as in text and image data.

There is no clustering algorithm that is optimal, i.e. it has the best performance on all the datasets [3]. However, depending on the data characteristics, different algorithms are appropriate for capturing the underlying structure of the data. Furthermore, each clustering algorithm has certain crucial pararameters that are critical to the performance of the algorithm. There is generally no guidance available to select these parameters. In this chapter, we propose to use any additional information to evaluate a goodness measure for each value of the parameter, and select the parameter with the highest goodness value. We call this approach *semi-supervised parameter selection*.

4.2 Non-parametric mixture model

4.2.1 Model description

Let $\mathcal{D} = \{x_1, \ldots, x_n\}$ be a collection of n data points to be clustered, where each $x_i \in \mathbb{R}^d$ is a vector of d dimensions. Let G be the number of clusters. The aim is to fit the data points in $\mathcal{D}$ by a non-parametric mixture model. Let $\kappa(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be the kernel function for density estimation. We further assume that the kernel function is stationary, i.e., $\kappa(x_i, x_j) = \kappa_s(x_i - x_j)$, where $\int \kappa_s(x)dx = 1$. Let the matrix $K = [\kappa(x_i, x_j)]_{n\times n} \in \mathbb{R}_+^{n\times n}$ denote the pairwise kernel similarity for data points in $\mathcal{D}$.

Let $\{c_g\}, g = 1, \ldots, G$ be the set of G clusters that forms a partition of $\mathcal{D}$. We specify the conditional density function $p_g(x|c_g, \mathcal{D})$ for each cluster c_g as follows:

$$p_g(x|c_g, \mathcal{D}) = \frac{1}{|c_g|} \sum_{x_i \in c_g} \kappa(x, x_i) \tag{4.1}$$

where $|c_g|$ is the number of samples in cluster c_g, and $\sum_g |c_g| = n$. The unconditional (on clusters) density $p(x|\mathcal{D})$ is then written as

$$p(x|\mathcal{D}) = \sum_{g=1}^{G} \pi_g p_g(x|c_g, \mathcal{D}) \tag{4.2}$$

where $\pi_g = P(c_g)$ is the mixture coefficient for cluster c_g. We generalize the cluster conditional density $p(x|c_g, \mathcal{D})$ in Eq. (4.1) by considering soft cluster memberships. We denote by $w^g = (w_1^g, \ldots, w_n^g)$ the probability of assigning data points to cluster c_g. Evidently, we have $w_i^g \geq 0, i = 1, \ldots, n, g = 1, \ldots, G$, and $\sum_{g=1}^{G} w_i^g = 1$. Using the soft memberships $w^g, g = 1, \ldots, G$, we can then generalize $p_g(x|c_g, \mathcal{D})$ as

$$p_g(x|c_g, \mathcal{D}) = \frac{1}{\sum_{i=1}^{n} w_i^g} \sum_{i=1}^{n} w_i^g \kappa(x_i, x) \tag{4.3}$$

Let $q_i^g = w_i^g / (\sum_{j=1}^{n} w_j^g)$. This simplifies $p_g(x|c_g, \mathcal{D})$ as

$$p_g(x|c_g, \mathcal{D}) = \sum_{i=1}^{n} q_i^g \kappa(x_i, x) \tag{4.4}$$

We refer to $q^g = (q_1^g, \ldots, q_n^g)$ as the *profile vector* for cluster c_g, and $Q = (q^1, \ldots, q^G)$ as the *profile matrix*. The objective of our clustering model is to learn the profile matrix Q for data set $\mathcal{D}$. We emphasize that due to the normalization step, i.e., $\sum_{j=1}^n q_j^g = 1$, q_j^g can no longer be interpreted as the probability of assigning x_j to cluster c_g. Instead, it only indicates the relative importance of x_j to the density function for cluster c_g. We finally note that density function in Eq. (4.4) is also referred to as the density estimate in "dual form" [162].

4.2.2 Estimation of profile matrix Q by leave-one-out method

To estimate the profile matrix Q, we follow the idea of maximum likelihood, i.e., to find the matrix Q by solving the optimization problem $\max_Q \sum_{i=1}^n \log p(x_i|\mathcal{D})$. One major problem with this approach is that, when estimating $p(x_i|\mathcal{D})$, x_i is already an observed data point in $\mathcal{D}$ that is used to construct the density function $P(x_i|\mathcal{D})$. As a result, simply maximizing the likelihood of data may lead to an overestimation of the parameter Q, a problem that is often referred to as overfitting in machine learning [20]. We resolve this problem by replacing $p(x_i|\mathcal{D})$ with its leave-one-out (LOO) estimate [163].

We first define $p_i(x_i|c_g, \mathcal{D}_{-i})$, the LOO conditional probability for each held out sample x_i, conditioned on the clusters and the remaining $n-1$ samples, as follows

$$p_i(x_i|c_g, \mathcal{D}_{-i}) = \frac{1}{\sum_{j=1}^n (1-\delta_{j,i}) q_j^g} \sum_{j=1}^n (1-\delta_{j,i}) q_j^g K_{i,j}, \tag{4.5}$$

where $D_{-i} = \mathcal{D} \backslash \{x_i\}$ denotes the subset of $\mathcal{D}$ that excludes sample x_i. Using the LOO cluster conditional probability $p_i(x_i|c_g, \mathcal{D}_{-i})$, we further define the LOO unconditional (on cluster) density for each held out sample x_i as follows:

$$p_i(x_i|\mathcal{D}_{-i}) = \sum_{g=1}^G \gamma_i^g p_i(x_i|c_g, D_{-i}), \tag{4.6}$$

where $\gamma_i^g = P(c_g|\mathcal{D}_{-i})$, and $\sum_g \gamma_i^g = 1, \forall i = 1, \ldots, n$. Note that unlike the mixture model in (4.2) where the same set of mixture coefficients $\{\pi_g\}_{g=1}^G$ is used for any x_i, in (4.6),

mixture coefficients $\{\gamma_i^g\}_{g=1}^G$ depend on sample x_i, due to the leave-one-out estimation. We denote by $\gamma_i = (\gamma_i^1, \cdots, \gamma_i^G)$ and $\Gamma = (\gamma_1, \ldots, \gamma_n)^\top \in \mathbb{R}_+^{n \times G}$.

To improve the robustness of the estimation of profile matrix Q, we introduce a Gaussian prior for profile matrix Q, i.e.,

$$p(Q) \quad \propto \quad \exp\left(-\lambda \sum_i \sum_g [q_i^g]^2\right), \tag{4.7}$$

where λ is a hyperparameter that will be determined empirically.

Remark One of the natural choices for the prior on each q^g would be the Dirichlet distribution, since we have $\sum_{i=1}^n q_i^g = 1$ for $g = 1, \cdots, G$, that is, $\text{Dir}(Q|\alpha) \propto \prod_{g=1}^G \prod_{i=1}^n (q_i^g)^{\alpha-1}$, where α is a hyper parameter. However, a Gaussian prior makes it convenient to incorporate the side information (e.g. instance level pairwise constraints [98]) into to the model. For example, if two samples are linked by a must-link constraint, it can be incorporated into the Gaussian prior with high correlation between the profile values of the two linked data points. However, since this prior is being specified on a matrix Q, it needs to use matrix-variate distributions.

Remark The maximum for $\log p(Q)$ under the constraint $\sum_{i=1}^n q_i^g$, g = 1,...,G , occurs when all $q_i^g = \frac{1}{n}$. This encodes a prior belief that all points equally contribute to all the clusters. A smaller value of λ results in a relatively sparser solution for Q. As the value of λ increases, the solution tends towards being uniform, i.e. $q_i^g \to \frac{1}{n}$ for $i = 1, \cdots, n$. However, choosing $\lambda = 0$ results in several numerical instabilities which are discussed in detail in Appendix A.

Bayesian interpretation NMM can also be interpreted as a Bayesian model by specifying the following data generation process:

1. Generate the mixture coefficients γ_i^g, $g = 1, \cdots, G$ from a Dirichlet prior with $\alpha = 1$.

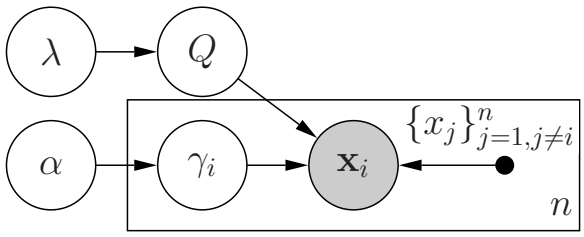


Figure 4.1: Graphical model showing the data generation process using the NMM.

2. Generate the profile coefficients $\mathbf{q}^g$ for the cluster c_g from the prior specified in Eq (4.7), given the parameter λ.

3. Using γ_i and Q, sample the point x_i from the density in Eq (4.5).

Figure 4.1 shows the graphical model corresponding to the data generation process.

For notational convenience, we set $K_{i,i} = 0$ in Eq (4.5). Now, using the condition $\sum_{i=1}^{n} q_i^g = 1$, the LOO log-likelihood of data, denoted by $\ell_{LOO}(\mathcal{D}; Q, \Gamma)$, can be expressed as follows

$$\begin{aligned} \ell_{LOO}(\mathcal{D}; Q, \Gamma) &= \log p(Q) + \sum_{i=1}^{n} \log p_i(x_i|\mathcal{D}_{-i}) \\ &= -\lambda \sum_{i=1}^{n}\sum_{g=1}^{G} (q_i^g)^2 + \sum_{i=1}^{n} \log\left(\sum_g \gamma_i^g \frac{\sum_{j=1}^{n} K_{i,j} q_j^g}{1 - q_i^g}\right). \end{aligned} \tag{4.8}$$

The parameters in the above simplified model are γ_i^g and q_i^g, for $i = 1, \cdots, n$ and $g = 1, \cdots, G$. They are estimated by maximizing the LOO log-likelihood $\ell_{LOO}(\mathcal{D}; Q, \Gamma)$. The optimal values of Q and Γ, denoted by Q^* and Γ^* can be obtained by solving the following optimization problem:

$$\{Q^*, \Gamma^*\} = \arg\max_{Q,\Gamma} \ell_{LOO}(\mathcal{D}; Q, \Gamma) \tag{4.9}$$

The optimization procedure used is discussed in the following section.

4.2.3 Optimization methodology

To determine the optimal values of Γ and Q that maximize the log-likelihood in Eq (4.9), we apply an alternating optimization strategy [164]. At each iteration, we first optimize Γ with fixed Q, and then optimize Q with fixed Γ. Below, we give the procedure for optimizing Γ and Q.

Optimizing Γ

By fixing Q, the optimal value of Γ can be obtained using the following proposition.

Proposition 4. *For a fixed Q, the LOO log-likelihood of a sample x_i is maximized when*

$$\gamma_i^g \quad = \quad \delta(g, \arg\max_{g'} p_i(x_i|c_{g'}, \mathcal{D}_{-i})), \tag{4.10}$$

where $\delta(.,.) = 1$ if the arguments are equal to each other, and 0, otherwise.

Proof. Collecting the terms containing γ_i^g from Eq (4.8), the maximization problem for Γ can be written as follows,

$$\max_{\Gamma} \quad \sum_{i=1}^{n} \log \left(\sum_g \gamma_i^g \frac{\sum_{j=1}^{n} K_{i,j} q_j^g}{1 - q_i^g} \right) \tag{4.11}$$

$$\text{s.t.} \quad \sum_{g=1}^{G} \gamma_i^g = 1, i = 1, \cdots, n \tag{4.12}$$

Since $\sum_{g=1}^{G} \gamma_i^g = 1$, Jensen's inequality can be used to write the above optimization problem as,

$$\max_{\Gamma} \quad \sum_{i=1}^{n} \sum_g \gamma_i^g \log \left(\frac{\sum_{j=1}^{n} K_{i,j} q_j^g}{1 - q_i^g} \right) \tag{4.13}$$

$$\text{s.t.} \quad \sum_{g=1}^{G} \gamma_i^g = 1, i = 1, \cdots, n \tag{4.14}$$

This is a linear programming problem, and the solution for Γ can be obtained as stated in Proposition 1.

□

The variable γ_i^g is closely related to the posterior distribution $\Pr(c_g|x_i)$, and therefore can be interpreted as the cluster label of the i-th sample, i.e., $\gamma_i^g = 1$ if $x_i \in c_g$ and 0, otherwise.

Optimizing Q

It is difficult to directly optimize the log-likelihood in Eq (4.8) with respect to Q. We therefore construct and minimize a convex variational upper bound on the negative log-likelihood for efficient inference. At each iteration, we maintain a touch point between the bound and the negative log-likelihood function, which guarantees convergence to at least a local minima [165].

The log of conditional probability $p_i(x_i|c_g, \mathcal{D}_i)$ in Eq (4.5) results in a log-sum form, and can be bounded as follows.

Proposition 5. *The logarithm of the conditional probability in Eq* (4.5) *satisfies the following concave lower bound,*

$$\log p_i(x_i|c_g, \mathcal{D}_i) = \log\left(\sum_{j=1}^{n} K_{i,j} q_j^g\right) - \log\left(1 - q_i^g\right) \quad (4.15)$$

$$\geq \sum_{j=1}^{n} \eta_{i,j}^g \log(K_{i,j} q_j^g) - \frac{1 - q_i^g}{z_i^g} - \log z_i^g + 1 + H(\eta_{i,\cdot}^g), \quad (4.16)$$

where $z_i^g \geq 0$ and $\eta_{i,j}^g, \sum_{j=1}^n \eta_{ij}^g = 1$ are the two variational distributions and $H(\eta_{i,\cdot}^g)$ corresponds to the Shannon entropy of the distribution $\eta_{i,\cdot}^g$.

Proof. Introducing variational distributions $\eta_{i,\cdot}^g, i = 1, \cdots, n; g = 1, \cdots, G$, whose j-th element is η_{ij}^g and $\sum_{j=1}^n \eta_{ij}^g = 1$ into the first term of Eq (4.15), and the variational

Bernoulli distribution $z_i^g, 0 \leq z_i^g \leq 1$ into the second term, we have,

$$\begin{aligned}
\log p_i(x_i|c_g, \mathcal{D}_i) &= \log\left(\sum_{j=1}^{n} K_{i,j} q_j^g\right) - \log\left(1 - q_i^g\right) && (4.17)\\
&= \log\left(\sum_{j=1}^{n} \left(\frac{\eta_{i,j}^g}{\eta_{i,j}^g}\right) K_{i,j} q_j^g\right) - \log\left(1 - q_i^g\right) && (4.18)\\
&= \log\left(\sum_{j=1}^{n} \eta_{i,j}^g \left(\frac{K_{i,j} q_j^g}{\eta_{i,j}^g}\right)\right) - \log\left(1 - q_i^g\right) && (4.19)\\
&\geq \sum_{j=1}^{n} \eta_{i,j}^g \log\left(\frac{K_{i,j} q_j^g}{\eta_{i,j}^g}\right) - \log\left(1 - q_i^g\right) && (4.20)\\
&= \sum_{j=1}^{n} \eta_{i,j}^g \log\left(K_{i,j} q_j^g\right) - \sum_{j=1}^{n} \eta_{i,j}^g \log \eta_{i,j}^g - \log\left(1 - q_i^g\right) && (4.21)\\
&= \sum_{j=1}^{n} \eta_{i,j}^g \log(K_{i,j} q_j^g) - \frac{1 - q_i^g}{z_i^g} - \log z_i^g + 1 + H(\eta_{i,\cdot}^g), && (4.22)
\end{aligned}$$

□

The bound introduced in the above proposition holds for any z and η. Clearly, the tightest bound is achieved by maximizing over z_i and η, whose solutions are given below.

Proposition 6. *The optimal values for the variables (z_i^g and $\eta_{i,j}^g$) introduced in Proposition 5 are:*

$$z_i^g = 1 - q_i^g \;\; \textit{and} \;\; \eta_{i,j}^g = \frac{K_{i,j} q_j^g}{\sum_{j'=1}^{n} K_{i,j'} q_{j'}^g}. \tag{4.23}$$

Proof. The optimal values of the variables in Proposition 2 can be obtained as the stationary points of the bound. Differentiating the bound from Proposition 2 w.r.t η_{ij}^g, and setting it to 0, we get

$$\log(K_{ij} q_j^g) - 1 - \log(\eta_{ij}^g) = 0, \quad \sum_j \eta_{ij}^g = 1. \tag{4.24}$$

Solving the above equation for η_{ij}^g, gives us

$$\eta_{ij}^g = \frac{K_{ij} q_j^g}{\sum_{j=1}^{n} K_{ij} q_j^g}. \tag{4.25}$$

The optimal value for z_i can be obtained similarly,

$$\frac{1-q_i^g}{(z_i^g)^2} - \frac{1}{z_i^g} = 0, \quad \text{or} \quad z_i^g = 1 - q_i^g.$$

□

Using the bound from Proposition 5, the maximization of log-likelihood in Eq (4.8) can be approximated by the following optimization problem.

Proposition 7. *Given γ_i^g, η_{ij}^g and z_i^g, the optimal value of Q at each iteration can be obtained by solving the following convex problem.*

$$\begin{aligned} \min_{Q} \quad & \lambda \sum_{i=1}^{n}\sum_{g=1}^{G} (q_i^g)^2 - \sum_{i=1}^{n}\sum_{g=1}^{G} \frac{\gamma_i^g}{z_i^g} q_i^g - \sum_{j=1}^{n}\sum_{g=1}^{G} \left[\left(\sum_{i=1}^{n} \gamma_i^g \eta_{i,j}^g \right) \log q_j^g \right] \\ \textit{s. t.} \quad & 0 \leq q_i^g, i = 1, \ldots, n, g = 1, \ldots, G, Q^\top \mathbf{1}_n = \mathbf{1}_G. \end{aligned} \tag{4.26}$$

Proof. Combining the bounds and results from propositions 1, 2 and 3, and simplifying the equation results in the optimization problem for Q. □

The convex optimization problem in Eq (4.26) can be solved as follows. Constructing a Lagrangian for the problem in Eq (4.26), and setting its derivative w.r.t. q_i^g to 0, we have

$$2\lambda q_i^g - \frac{\gamma_i^g}{z_i^g} - \frac{1}{q_i^g} \left(\sum_{j=1}^{n} \gamma_j^g \eta_{j,i}^g \right) - \theta_g = 0. \tag{4.27}$$

where θ_g, $g = 1, \cdots, G$ are the Lagrangian multipliers for constraints $\sum_{j=1}^{n} q_j^g = 1$. Defining

$$a_i^g = 2\lambda, \; b_i^g = \frac{\gamma_i^g}{z_i^g} + \theta_g, \; c_i^g = \sum_{j=1}^{n} \gamma_j^g \eta_{j,i}^g,$$

the above equation can be rewritten as,

$$a_i^g \, (q_i^g)^2 - b_i^g q_i^g - c_i^g = 0.$$

Given coefficients a_i^g, b_i^g, and c_i^g, the solution to q_i^g is found as

$$q_i^g = \frac{b_i^g + \sqrt{[b_i^g]^2 + 4a_i^g c_i^g}}{2a_i^g}.$$

We can estimate θ_g by solving the nonlinear equation $\sum_{i=1}^{n} q_i^g = 1$. Note that the function $\sum_{i=1}^{n} q_i^g$ is monotonically increasing in terms of θ_g, and therefore can be solved by bisection search on the interval $\theta_g \in [\min_i(a - b_i^g - nc_i^g), \max_i(b_i^g + nc_i^g - a)]$.

Remark One can interpret quantities b_i^g and c_i^g as follows: (i) b_i^g essentially measures the consistency between $\Pr(c_g|x_i)$ and q_i^g. In particular, if γ_i^g is large, which implies x_i must be assigned to cluster c_g, and z_i^g is small, which implies q_i^g is large at the current iteration, we then will have a large b_i^g, which will lead to larger value of q_i^g in the next iteration; (ii) c_i^g measures the consistency between assigning data point x_i to cluster c_g and its neighbors. In particular, a large $\eta_{j,i}^g$ indicates that x_i has a significant impact in driving x_j to cluster g.

Remark If $\lambda = 0$, then we can see that the solution of q_i^g is given by the following equation: $q_i^g = \frac{-c_i^g}{b_i^g}$. The following non-linear equation must be solved to obtain θ^g,

$$\sum_{i=1}^{n} \frac{c_i^g}{\frac{\gamma_i^g}{z_i^g} + \theta^g} + 1 = 0$$

Since $\theta^g \in R$, this function is discontinuous whenever $\theta^g = -\frac{\gamma_i^g}{z_i^g}$. Since we require $q_i^g \geq 0$, we need $\theta^g + \max_i \frac{\gamma_i^g}{z_i^g} < 0$. Also, γ_i^g takes a non-zero value for only one of the g values. This function is therefore well defined only in the range $\theta \in (-\infty, -\max(\gamma_i^g/z_i^g))$. Whenever a point contributes to a cluster by a large amount, the value of γ_i^g/z_i^g is really large. This results in several numerical problems while performing a bisection search. Apart from this, we have observed that the performance with $\lambda = 0$ is much inferior to that with a small positive λ; resulting function is smoother, and easier to numerically optimize.

The procedure for finding Q and Γ that maximizes the log-likelihood in Eq (4.8) is summarized in Algorithm 1. Upon convergence, the value of γ_i determines the cluster label for x_i.

Algorithm 1 $[Q, \Gamma]$ = NonParametricMixture($\mathcal{D}, G, \lambda, \sigma$)

Input: Dataset $\mathcal{D}$, no. of clusters G, parameters λ and σ

Output: Cluster labels Γ and the profile matrix Q

1: Compute the kernel matrix K for the points in $\mathcal{D}$ with bandwidth σ. Normalize K such that $\sum_k K_{ij} = 1$.
2: Set the iteration $t \leftarrow 0$.
3: Initialize $Q^{(t)} \leftarrow Q_0$, such that $Q_0 \succcurlyeq 0$, $Q_0^T 1_n = 1_G$.
4: **repeat**
5: $t \leftarrow t + 1$;
6: Compute the variables γ_i^g using Eq (4.10), $\eta_{i,j}^g$ and z_i^g using Eq (4.23) and $Q^{(t-1)}$.
7: Minimize Eq (4.26) to estimate $Q^{(t)}$.
8: $\Delta Q \leftarrow Q^{(t)} - Q^{(t-1)}$.
9: **until** $||\Delta Q||_2^2 \leq \epsilon$, ($\epsilon$ is pre-set to a desired precision)
10: **return** Q, Γ

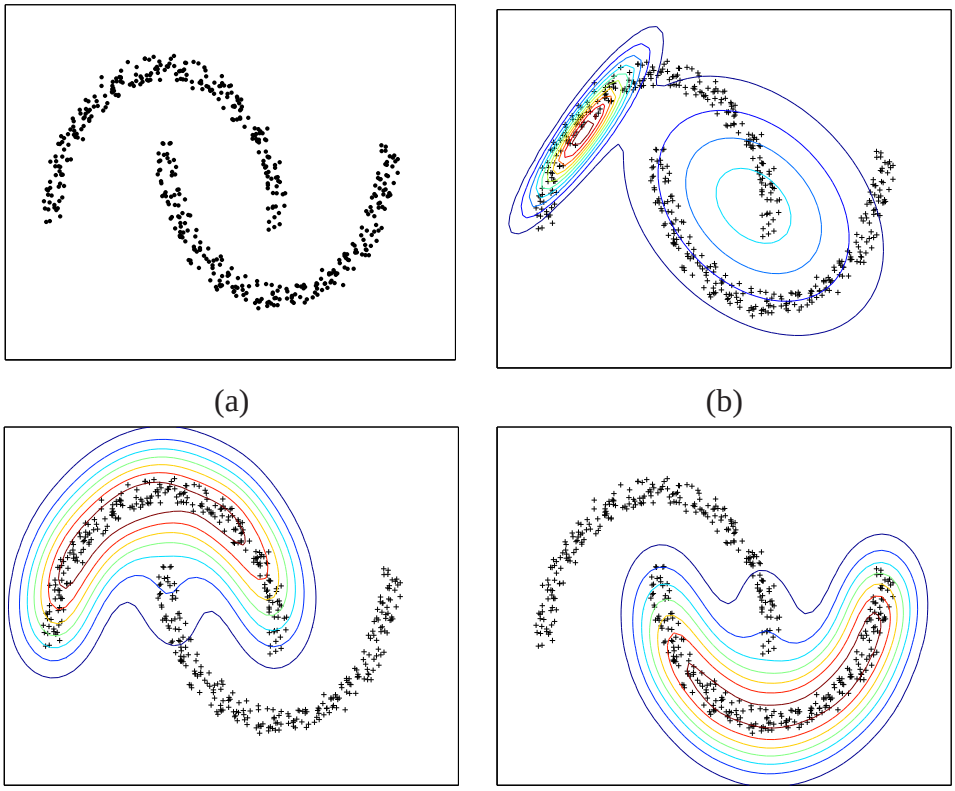
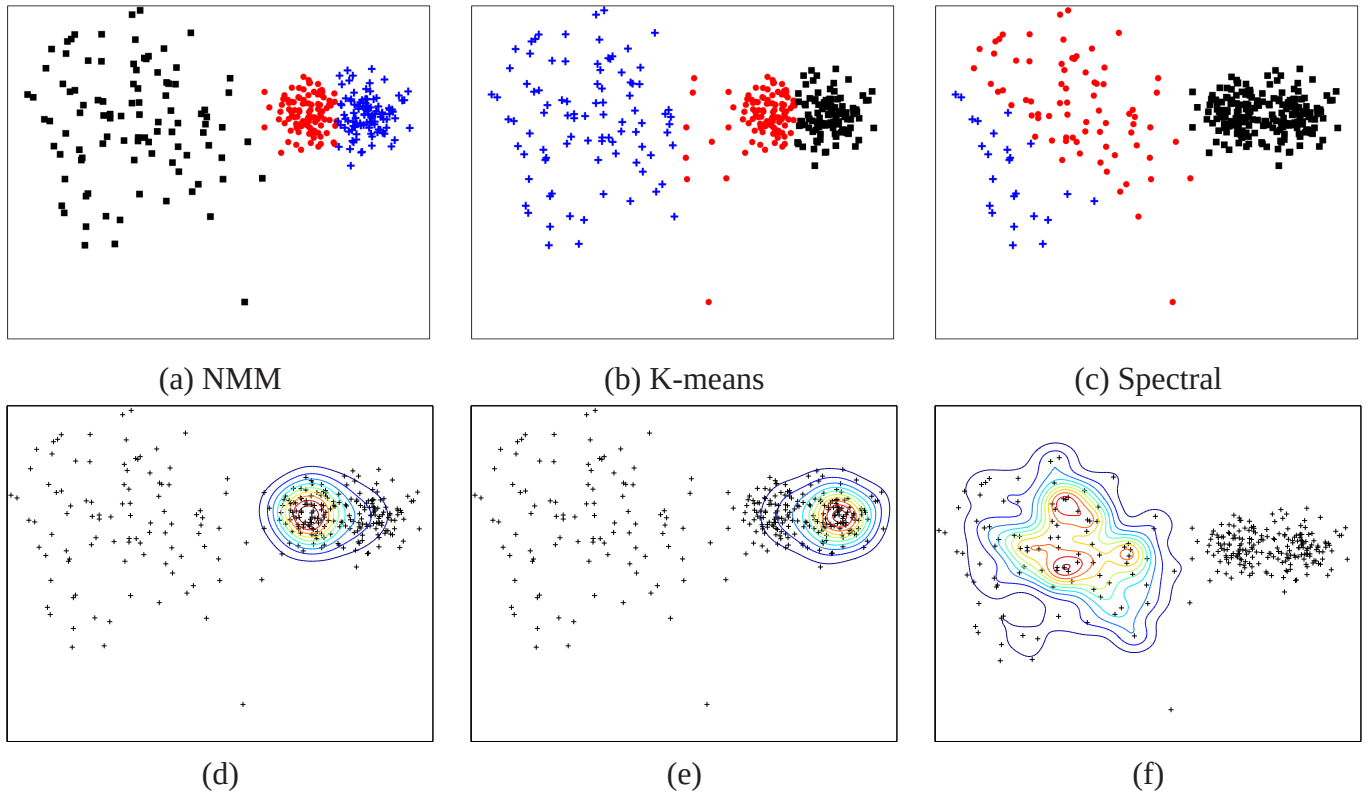


Figure 4.2: Illustration of the non-parametric mixture approach and Gaussian mixture models on the "two-moon" dataset. (a) Input data with two clusters. (b) Gaussian mixture model with two components. (c) and (d) the iso-contour plots of non-parameteric estimates of the class conditional densities for each cluster. The warmer the color, the higher the probability.



(a) NMM (b) K-means (c) Spectral

(d) (e) (f)

Figure 4.3: Illustration of the (a) non-parametric mixture approach, (b) K-means and (c) spectral clustering on the example dataset from [1]. Input data contains 100 points each from three spherical two-dimensional Gaussian clusters with means (0,0), (6,0) and (8,0) and variances $4I_2$,$0.4I_2$ and $0.4I_2$, respectively. Spectral clustering and NMM use $\sigma = 0.95$. Plots (d)-(f) show the cluster-conditional densities estimated by the NMM.

4.2.4 Implementation details

Normalization is one of the key issues in kernel density estimation. Conventionally, the kernel function is normalized over the entire domain of the data, $\kappa_\sigma(\mathbf{x}) = (\pi\sigma)^{-d}\exp\left(-||\mathbf{x}||^2/2\sigma^2\right)$. However, this may cause serious problems in density estimation for high-dimensional data (large values of d). To overcome this problem, we normalize the kernel matrix such that each of its columns sum to 1, i.e. $\sum_j K_{i,j} = 1$. This essentially nullifies the effect of dimensionality on the estimation process, and is useful in handling sparse datasets.

In all kernel based clustering algorithms, the kernel bandwidth σ is the most crucial parameter. Our empirical results show that, similar to spectral clustering, the choce of kernel bandwidth σ is critical to the success of the nonparametric mixture algorithm; λ is not very critical.

in all of our experiments we choose $\lambda = 10^{-4}$, which results in mild smoothing of the q_i^g values, and avoids any numerical instability in the algorithm due to the logarithm.

From Proposition 3, it appears as if the most memory and computationally intensive part of the algorithm is the storage and computation of $\eta_{i,j}^g$, which requires $O(n^2G)$ space. However, note that $\eta_{i,j}^g$ is always accompanied with the variable γ, as $\sum_{g=1}^G \gamma_i^g \eta_{i,j}^g$. By exploiting this, the space and computational requirements may be simplified by an order of magnitude. Since only one of the G possible γ_i^gs is equal to 1 for each $i = 1, \cdots, n$, the only $\eta_{i,j}^G$ that needs to be computed is when $\gamma_i^g = 1$; rest of them will be 0 since they will be multiplied by γ_i^g with value equal to 0. The overall space requirement for intermediate variables is therefore $O(nG)$, which is much smaller compared to the $O(n^2)$ space required to store the full kernel, in kernel based methods. Usage of sparse kernels can reduce the computational and space requirements further.

4.3 Results and discussion

The NMM algorithm is evaluated on datasets from three different sources (a) synthetic, (b) UCI and (c) text datasets derived from the 20-newsgroups[2] dataset [166]. In the following section, the baselines used for comparison are discussed. The results and discussion on the test datasets are presented in the subsequent sections.

4.3.1 Baseline methods:

The proposed non-parametric mixture algorithm is compared with three classes of well known clustering algorithms: (a) K-means and Gaussian mixture model (GMM) with diagonal and full covariance matrices, (b) one kernel-based algorithm, namely NJW spectral clustering [42], and (c) three non-parametric hierarchical clustering algorithms, including Single Link, Complete Link and Average Link. For (a) and (c), we use the implementations from the Matlab's Statistics Toolbox. For the linkage based methods, the number of clusters is externally specified. We chose the spectral clustering algorithm based on [42] because its performance is shown to be comparable to that of Normalized Cuts, and it has been shown to be equivalent to Kernel K-means. Comparison with Mean-shift, or related algorithms is difficult as the number of clusters is not specified apriori in these algorithms. Note that clustering approaches like Mean-shift were not designed for clustering high-dimensional text data and they are known to perform poorly on such datasets. Each algorithm is run 10 times and performance averaged over ten runs of the experiment is reported in Tables 4.2 and 4.4. The best performance for each dataset is shown in bold face in Tables 4.2 and 4.4.

At each run, the NMM, K-means, GMM, and Spectral clustering were initialized with 5 different starting points; only the best performance is reported. We only show the best performance among the three hierarchical linkage based algorithms, without specifying which algorithm achieved it.

[2]http://people.csail.mit.edu/jrennie/20Newsgroups/

4.3.2 Synthetic Datasets

The NMM algorithm aims at identifying clusters of arbitrary shapes, while estimating their conditional density. Figure 4.2 illustrates the performance of NMM on a dataset not suitable for GMM. Figure 4.2(a) shows the input data. Figure 4.2(b) is shown to contrast the proposed NMM agains the parametric Gaussian mixture model (GMM) with the number of mixture components set to two. Figures 4.2(c) and (d) show the class conditional densities for each of the two clusters. The proposed algorithm is able to recover the underlying clusters, as well as estimate the associated conditional densities, which is not possible for GMM as shown in Figure 4.2(b).

Figure 4.3 illustrates the performance of the proposed algorithm on a dataset that is known to be difficult for spectral clustering [1]. Both K-means and spectral clustering fail to recover the clusters due to the difference in the covariances of the three spherical clusters. Because of the local nature of the NMM (the cluster label of a point is affected only by the cluster labels of neighboring points), it can succesfully recover the clusters, as shown in Figure 4.3(a); the cluster conditional densities are shown in Figures 4.3(d)-(f).

4.3.3 UCI Datasets

The proposed algorithm is evaluated on 17 different datasets from the UCI ML repository [167]. The details of the datasets are summarized in the first four columns of Table 4.2. The choice of these datasets is motivated by their popularity, variety and previous usage.

For the large UCI datasets, 3000 points are randomly sampled for clustering for computational feasibility. The pairwise-F_1 measure was used for evaluating the clustering quality [168]. An RBF kernel $K(x_i, x_j) = \exp(-||x_i - x_j||^2/2\sigma^2)$ is used for the density estimation and also in the spectral clustering. The parameter σ in the RBF kernel is set to the 5-th percentile of the pairwise Euclidean distances for each dataset.

The performances of the baseline algorithms and the NMM algorithm are presented in

Table 4.2: Mean pairwise F_1 value of the performance of different clustering algorithms over 10 runs of each algorithm on 17 UCI datasets. The kernel width is chosen as the 5^{th} percentile of the pairwise Euclidean distances for Kernel based algorithms. The best performance for each dataset is shown in bold. The name of the dataset, number of samples (n), dimension (d), and the number of target clusters (G) are shown in the first 4 columns respectively. An entry of '-' indicates that the algorithm did not converge. The last column shows the best F_1 value achieved by Single (S), Complete (C) and Average (A) link algorithms.

				NMM	K-means	GMM		NJW-Spec	Linkage
Dataset	n	d	G			Diag	Free		max(S,C,A)
adult	3000	47	2	**68.94**	63.02	59.81	59.61	65.38	61.23
sat	3000	36	6	**71.29**	63.99	70.82	66.86	50.11	60.93
banana	2000	2	2	**86.87**	76.69	77.25	83.10	82.85	76.69
bupa	345	6	2	**52.78**	44.55	50.40	49.94	37.08	44.46
heart	270	9	2	**84.08**	84.07	53.62	74.89	82.63	82.26
wdbc	569	14	2	90.82	**90.92**	62.91	77.31	49.18	89.91
glass	214	10	6	66.25	67.07	-	-	**68.33**	66.99
ionosphere	351	34	2	71.54	**71.77**	-	-	50.73	71.36
austra	690	15	2	81.03	**82.50**	52.53	66.71	39.95	75.71
musk2	3000	166	2	62.02	61.79	60.10	**62.39**	61.81	61.52
house	232	16	2	88.80	88.80	-	68.11	88.80	**91.39**
digits-389	317	16	3	80.93	62.25	48.47	**89.34**	58.17	73.20
iris	150	4	3	93.26	89.18	93.98	**96.66**	90.53	83.90
letter-ijl	227	16	3	58.15	54.44	53.88	**64.21**	54.98	53.73
magic04	3000	10	2	53.90	50.25	**55.96**	54.58	52.34	50.40
musk1	476	166	2	52.57	52.67	**55.51**	52.67	52.57	51.57
german	1000	24	2	55.58	50.99	-	-	**58.70**	51.24

Table 4.2. Overall, the NMM algorithm yields compelling performance compared to the baseline algorithms.

- NMM significantly outperforms K-means on 9 of the 17 datasets; on the remaining 8 datasets, the performance of NMM is still comparable to the best performing algorithm.
- NMM significantly outperforms GMM with diagonal covariances on 14 of the 17 datasets, and outperforms GMM with full covariances on 11 of the 17 datasets.

- NMM performs significantly better than spectral clustering on 11 of the 17 datasets, and significantly worse on only 1 of the 17 datasets.
- Hierarchical clustering is outperformed significantly on 14 of the 17 datasets.

4.3.4 Text Datasets

Table 4.3: Text datasets used in the evaluation. The datasets were composed of articles from three or four different newsgroups; the number of clusters (G) is assumed to be known. The number of samples and number of features is denoted by n and d respectively.

Dataset	n	d	G
cmu-different-1000	2975	7657	3
cmu-similar-1000	2789	6665	3
cmu-same-1000	2906	4248	3
cmu-different-100	300	3251	3
cmu-similar-100	288	3225	3
cmu-same-100	295	1864	3
cmu-classic300	300	2372	3
cmu-classic400	400	2897	3
4newsgroups	3000	500	2

We use 9 high dimensional text datasets to show the efficacy of the algorithm. Eight of the text datasets are from [169], which are prefixed by `cmu`.We selected four of the 20-newsgroups datasets to create the dataset `4newsgroups` with multimodal clusters. The task is to partition the 4 newsgroups into two clusters, `politics` vs. `religion`. The `politics` cluster is a combination of documents from the newsgroups `talk.politics.mideast` and `talk.politics.guns`. The `religion` cluster contains documents from the newsgroups `talk.religion.misc` and `soc.religion.christianity`. Features are extracted from these datasets by considering only those words that appear at least 10 times in the whole dataset. This reduces the dimensionality of the dataset by restricting the vocabulary size. The `4newsgroups`

dataset is preprocessed to select 500 features with the maximum mutual information with the desired class labels. A summary of the text datasets is presented in Table 4.3. In addition, NMM performs particularly well for some of the text data sets compared to the other baseline clustering algorithms. The difficulty of clustering the text datasets arises from its high dimensionality. It is however generally believed that text data, despite its high dimensionality in the original space, tends to exist in a manifold of low dimensionality [170]. The success of the NMM algorithm with text data indicates that it is suitable for handling high dimensional data that are embedded in a manifold of low dimensions. These manifolds are rarely compact, and hence algorithms that prefer globular clusters cannot capture the structure of the data. This explains why K-means and GMM fail to find good clusters in the text data becuase these two algorithms are essentially designed to prefer compact clusters.

Table 4.4: Mean pairwise F_1 value of the performance of different clustering algorithms over 10 runs of each algorithm on 9 high-dimensional text datasets. The kernel width is chosen as the 5^{th} percentile of the pairwise Euclidean distances for Kernel based algorithms. The best performance for each dataset is shown in bold. The name of the dataset, number of samples (n), dimension (d), and the number of target clusters (G) are shown in the first 4 columns, respectively. An entry of '-' indicates that the algorithm did not converge. The last column shows the best F_1 value achieved by Single (S), Complete (C) and Average (A) link algorithms.

	NMM	K-means	GMM		NJW-Spec	Linkage
Dataset			Diag	Free		max(S,C,A)
cmu-different-1000	**95.86**	87.74	-	-	94.37	40.31
cmu-similar-1000	**67.04**	49.86	-	-	45.16	37.28
cmu-same-1000	**73.79**	49.40	-	-	48.04	30.01
cmu-different-100	**95.27**	79.22	-	-	87.47	75.74
cmu-similar-100	**50.89**	40.10	-	-	38.35	43.82
cmu-same-100	**48.97**	44.85	-	-	46.99	41.79
cmu-classic300	85.32	**86.32**	-	-	86.02	80.61
cmu-classic400	**61.26**	60.13	-	-	51.01	53.31
4newsgroups	**76.81**	73.88	-	-	74.13	68.25

Table 4.4 shows that the NMM algorithm performs significantly better (paired t-test,

95% confidence) than the baseline clustering methods on all the text datasets, except for `cmu-classic-300` where its performance is slightly inferior to K-means. Gaussian mixture models are prone to numerical estimation problems when the number of dimensions is larger than the number of samples. For this reason, these text datasets cannot be clustered by GMM, as indicated by the '-' entries in Table 4.4.

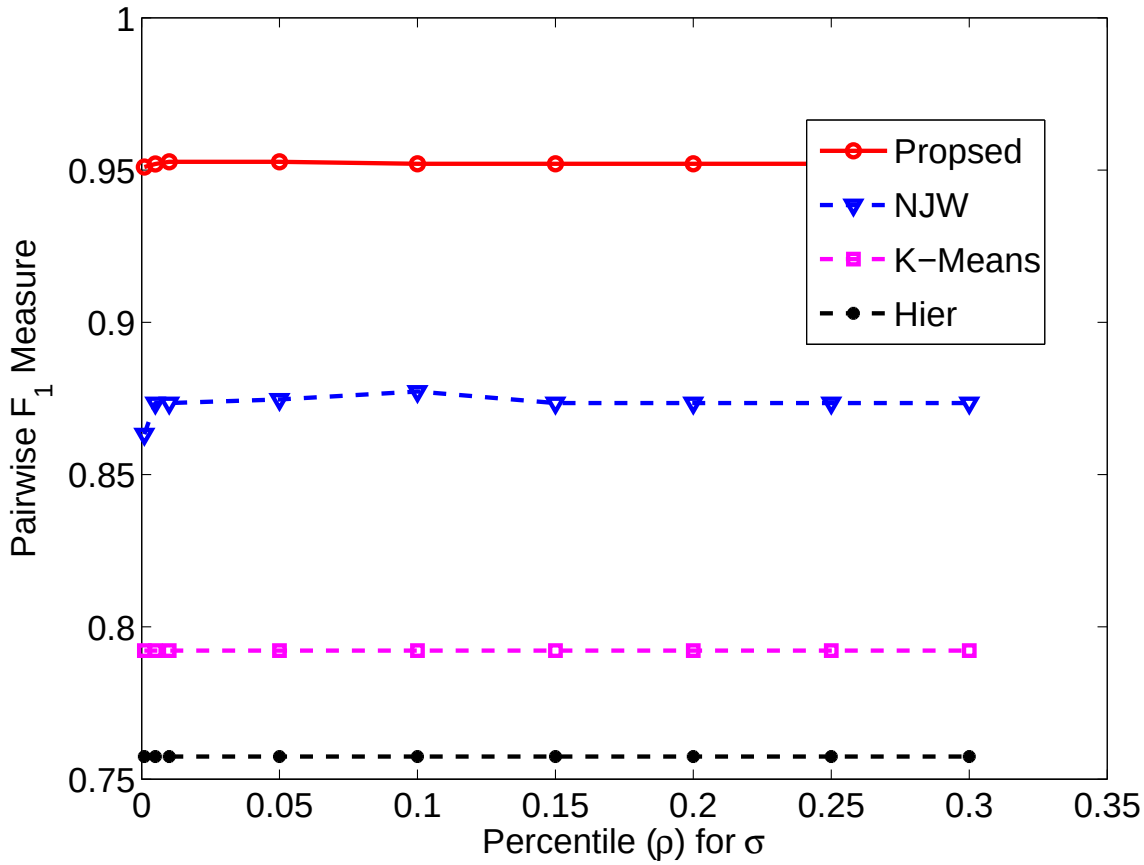
4.3.5 Discussion

Kernel density estimation (also known as non-parameteric density estimation, Parzen window estimation [20]) is a popular approach for density estimation. It aids in estimating the probability density of the data when the underlying density of the data is not known. In most real world data analysis tasks, the densities are not known, or may not even be closely approximated by any parameteric density. Furthermore, non-parameteric density estimates asymptotically converge to the true densities [163]. These estimates usually have a single parameter called the bandwidth, that controls the smoothness of the distribution. By changing the value of the bandwidth density estimates of varying smoothness can be obtained. In high dimensions, when the data is sparse, and when no assumptions about the density can be made, non-parameteric density estimates result in a smooth estimate of the density. This is one of the key reasons why the NPM algorithm performs well when the data is high-dimensional and sparse.

4.3.6 Sensitivity to parameters:

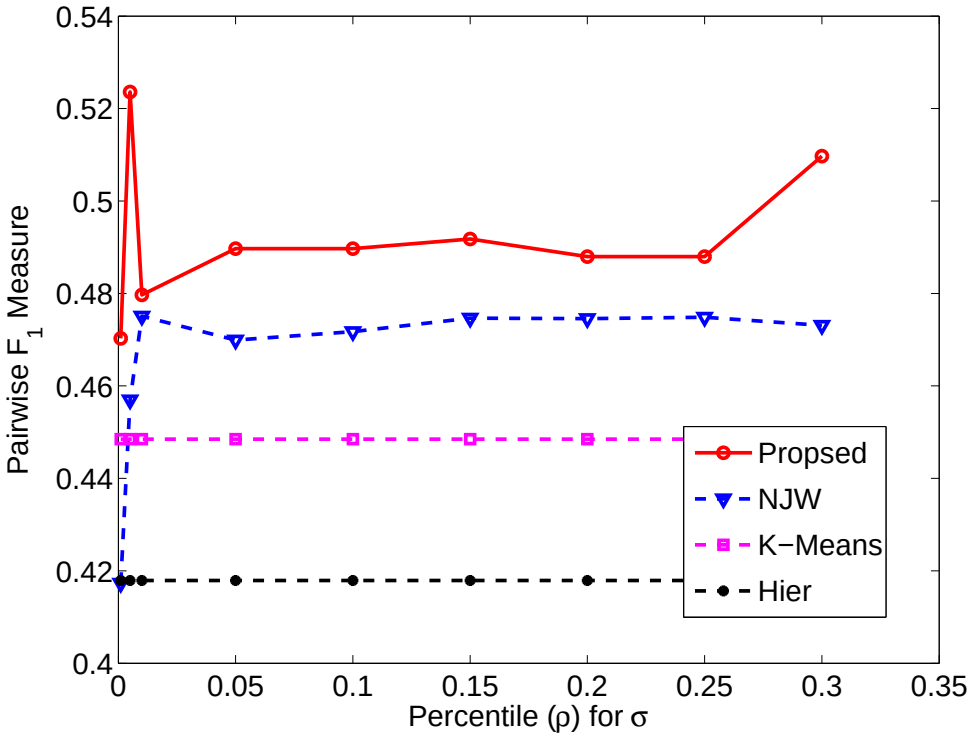
There are two parameters in the NMM algorithm: the regularizer weight λ and the kernel width σ. The parameter σ is set to the ρ^{th} percentile of the pairwise Euclidean distances. A useful range for ρ is 5-10%, as suggested in [41]. Figure 4.4 compares the performance of the NMM algorithm to K-means, Spectral clustering and Hierarchical clustering on 9 of the 26 datasets used (including both text and UCI) for different values of the kernel bandwidth.

Four of these datasets (Different-100, Similar-100, Same-100 and Classic-400) are chosen from the text datasets, and the remaining ones are chosen from the UCI datasets. For these datasets, the NMM algorithm exhibits superior performance over the baseline algorithms. The plots show that there exists a range of kernel bandwidth values for which the NMM algorithm performs significantly better than the competing methods. Figures 4.4(c), 4.4(d) and 4.4(i) show the datasets where the NMM algorithm performs better than the baseline algorithms only in a specific range of ρ, namely $(0, 0.1)$. For some datasets (e.g., Different-100, Classic-400), the NMM algorithm is more stable compared to that of other datasets. The NMM algorithm is not sensitive to the value of λ, over a larger range (10^{-4} to 10^{4}). For almost all the datasets, the change in performance is negligible with varying values of λ. However, λ does play a role in determining the sparsity of the profile matrix. As λ increases, the solution tends to get smoother. The performance of the clustering might degrade with larger values of λ. The key role of λ is to provide numerical stability to the algorithm. Therefore a small value of λ (10^{-4}) is preferred.

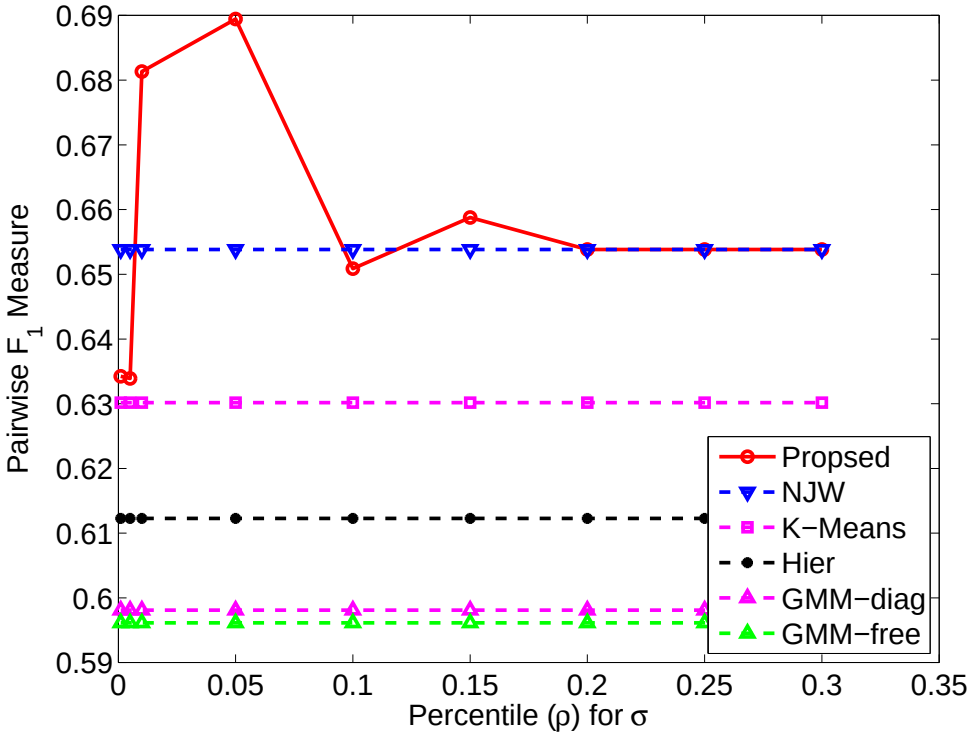


(a) Different-100

Figure 4.4: Performance of the NMM on nine of the 26 datasets used, with varying value of the percentile (ρ) used for choosing the kernel bandwidth (σ). The NMM algorithm is compared with NJW (Spectral Clustering), K-means and the best of the three linkage based methods.

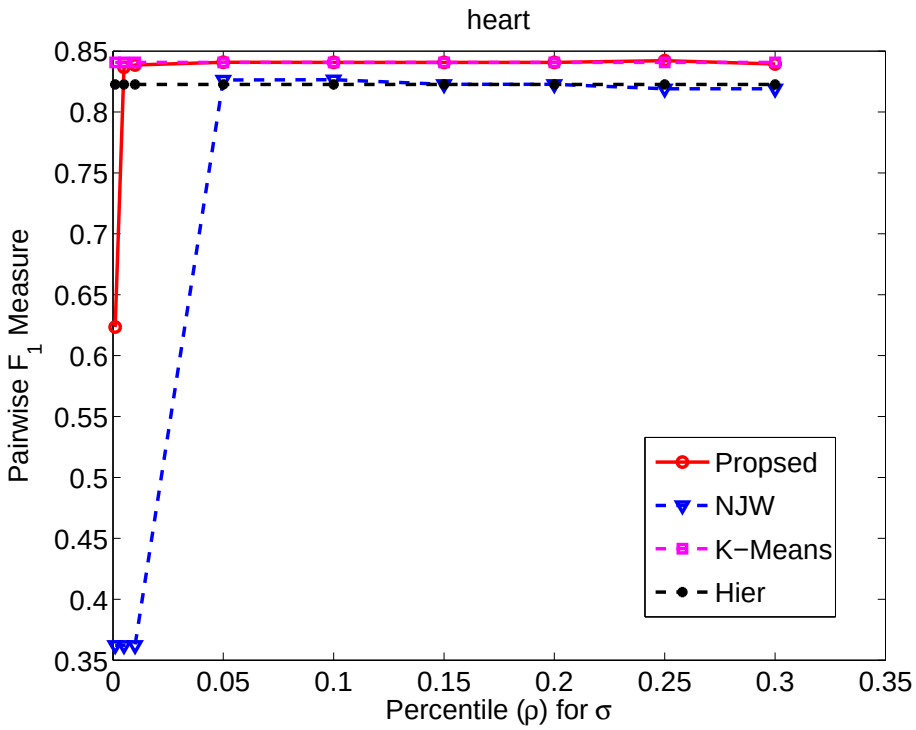


(b) Same-100

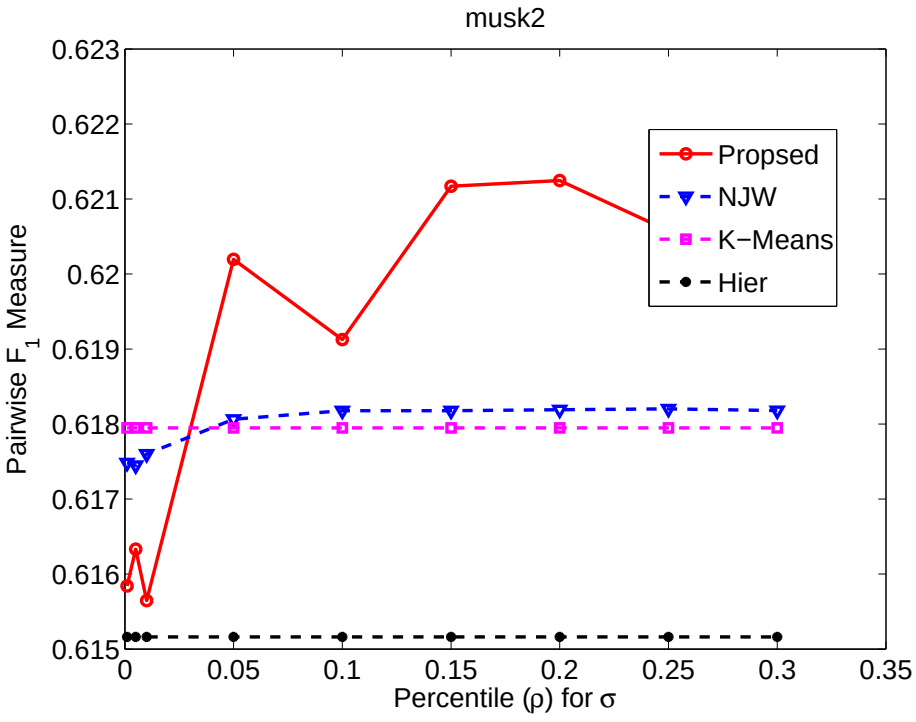


(c) Adult (UCI)

Figure 4.4: (Continued from previous page)

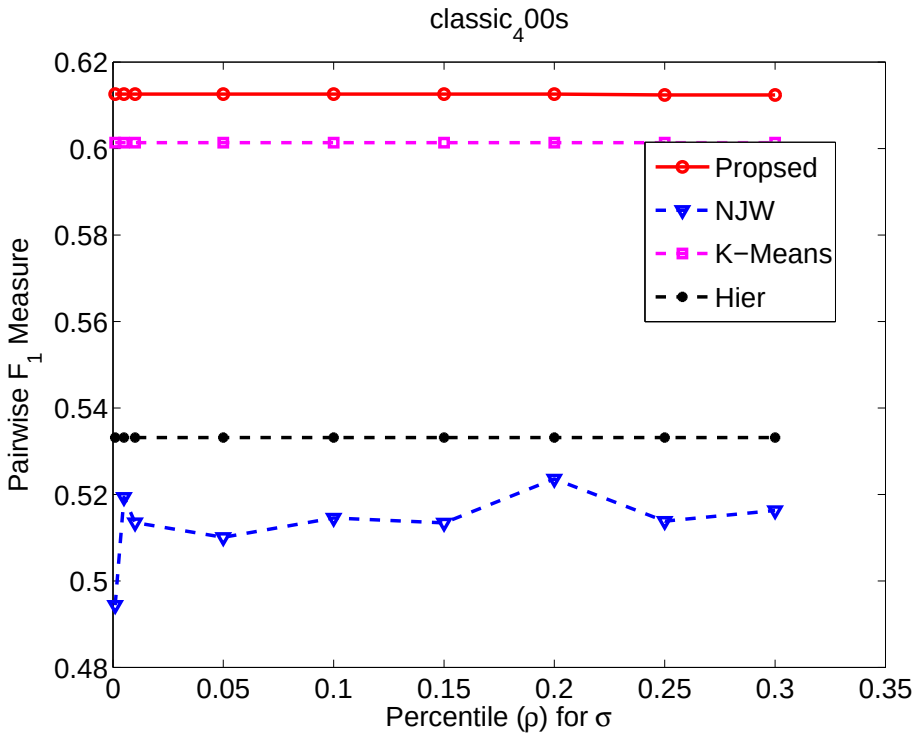


(d) Heart (UCI)

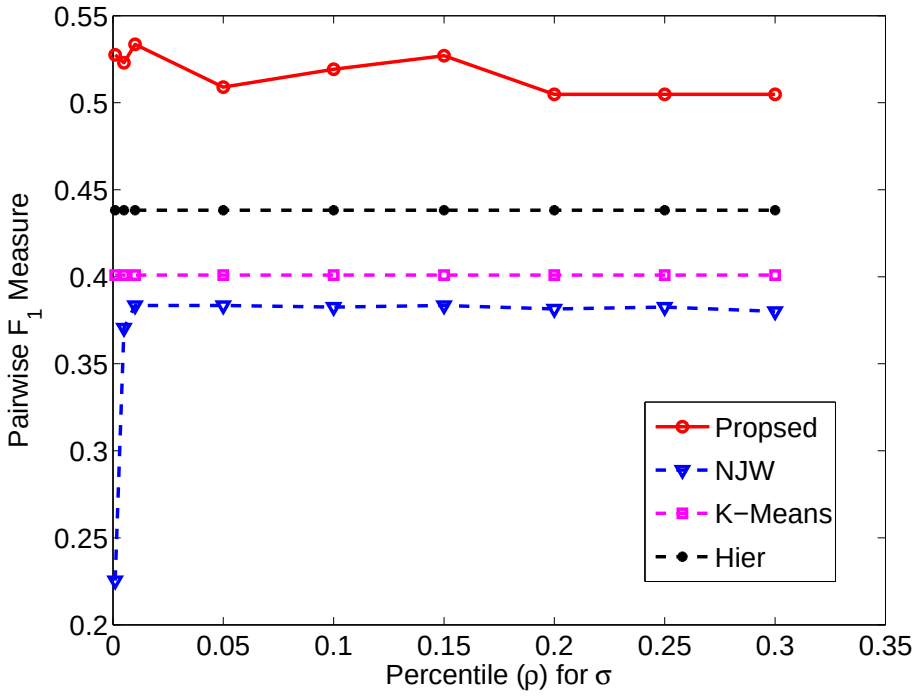


(e) Musk (UCI)

Figure 4.4: (Continued from previous page.)

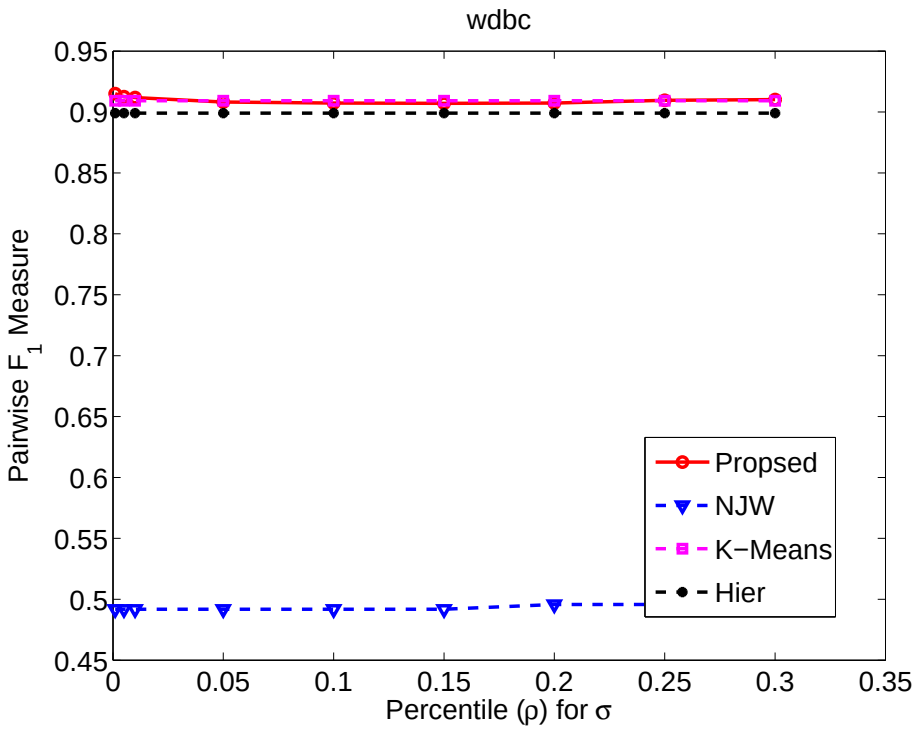


(f) Classic-400

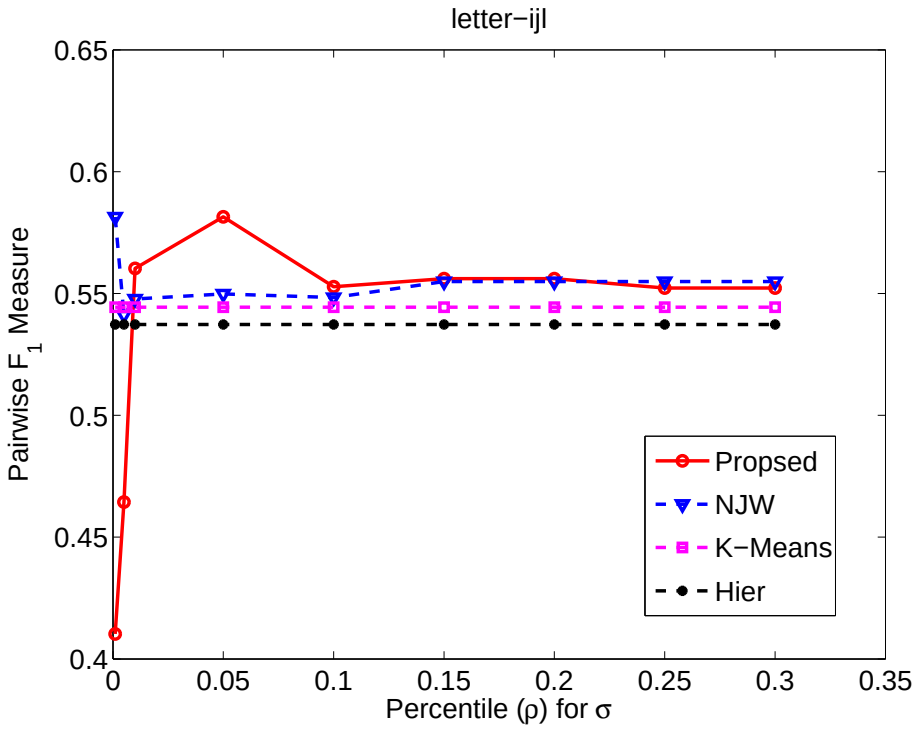


(g) Similar-100

Figure 4.4: (Continued from previous page.)



(h) wdbc



(i) letter-ijl

Figure 4.4: (Continued from previous page.)

4.4 Parameter selection

The kernel bandwidth σ is the most crucial parameter in kernel based clustering algorithms. Almost all kernels (except linear kernel) have a bandwidth parameter. In this section we define a heuristic for selecting the bandwidth parameter using pairwise constraints.

4.4.1 Maximum Pairwise Constraint Satisfaction Heuristic

For a given dataset with n data points, let m be the number of pairwise constraints that are randomly generated by sampling pairs of data points from the dataset with replacement. The set of must-link constraints is denoted by $\mathcal{M}$ and the set of cannot-link constraints is denoted by $\mathcal{C}$. The dataset is clustered by changing the value of σ each time. For each σ the clustering algorithm is run 10 times with different initializations. We define the pairwise constraint satisfaction measure $\xi_\sigma(\mathbf{y})$ as the fraction of constraints satisfied by the clustering algorithm for a particular value of σ and clustering $\mathbf{y}$. That is,

$$\xi_\sigma(\mathbf{y}) = \frac{\sum_{\mathbf{x}_i,\mathbf{x}_j \in \mathcal{M}} I(y_i = y_j) + \sum_{\mathbf{x}_i,\mathbf{x}_j \in \mathcal{C}} I(y_i \neq y_j)}{m} \tag{4.28}$$

Since RBF kernel is popularly used with spectral clustering and other kernel based methods, the parameter selection heuristic is evaluated for selecting the bandwidth of the RBF kernel for spectral clustering and the proposed non-parametric mixture approach. The value of σ is chosen as the ρ-th percentile of the pairwise similarities, and $\rho \in \{0.01, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30\}$. Figures 4.5 (a) and (b) show the performance of the clustering algorithm vs. ρ. In our experiments we set $m = \frac{n}{4}$. The clustering performance is measured using the pairwise F_1 measure [171] averaged over 10 runs of clustering algorithm with different initializations. In each plot, the horizontal axis shows the percentile ρ used for selecting the sigma and the vertical axis shows the performance. The solid line shows the performance of the clustering algorithm and the dashed line shows the constraint satisfaction ξ_σ. The value of σ corresponding to the value of ρ with the largest

constraint satisfaction is chosen as the parameter for the clustering algorithm. In almost all the cases, the constraint satisfaction plot has similar peaks and valleys as the clustering performance plot. In the cases where this does not hold, the clustering performance is very low, almost close to that of random assignment of labels. Figures 4.7 (a) and (b) compare the performance of Spectral Clustering with the proposed algorithm for the same range of values for σ. In almost all the datasets, even if the proposed non-parametric mixture approach does not outperform the spectral clustering for the whole range of σ, there exists a value of σ for which the proposed algorithm outperforms spectral clustering irrespective the σ value chosen by the spectral clustering. Since pairwise constraint based parameters selection provides a way to identify this particular σ, the proposed algorithm outperformed spectral clustering on almost all the datasets.

Table 4.5 compares the performance of spectral clustering with NPM. The bandwidth parameter σ is chosen using the maximum pairwise constraint satisfaction heuristic for both the approaches.

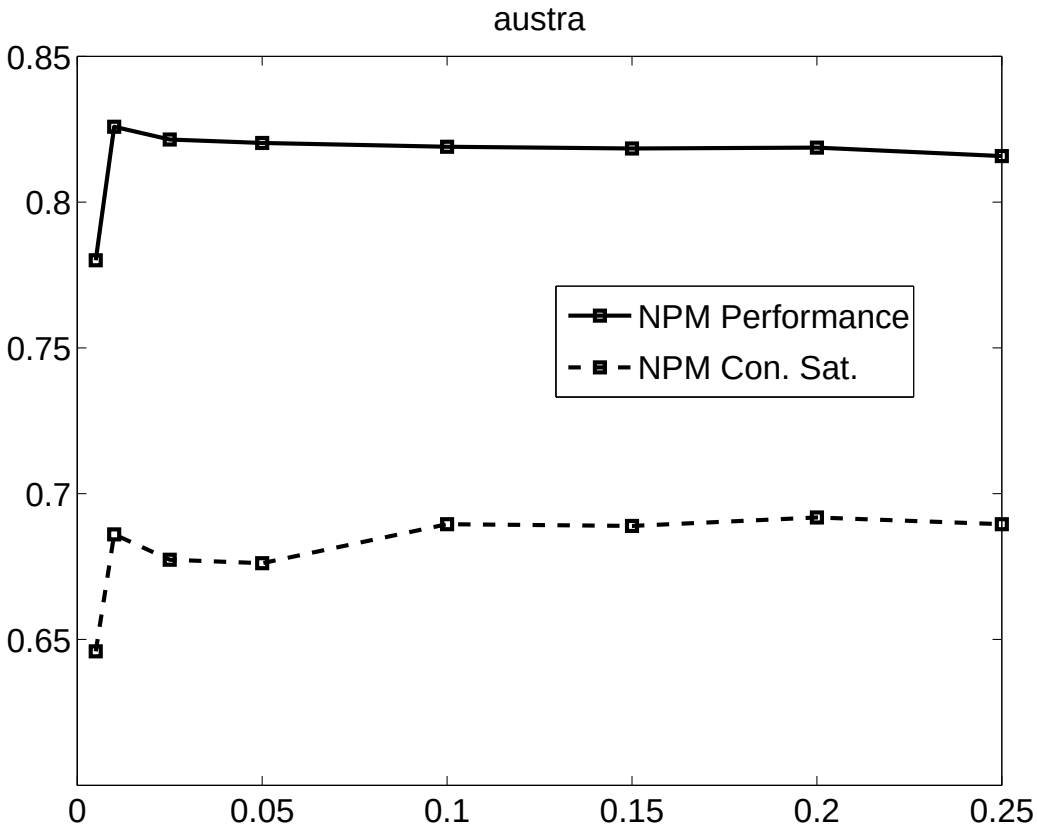
Goldberger and Roweis [172] present an algorithm to cluster a given dataset using a Mixture of Gaussians with a large number of components, and further clustering the components into a reduced mixture with lesser number of components. They achieve this by minimizing the KL divergence between the learnt mixture with large number of components, and another mixture with smaller number of components. The proposed approach can be related to an extreme case of [172], where each point is treated as a separate cluster with a Gaussian density of fixed variance and mean equal to the data point, and these clusters are further clustered.

4.5 Connection with K-means

In this section, we show that two simplifications of the proposed objective function (Eq (4.8)) lead to objective functions that correspond to K-means and Spectral cluster-

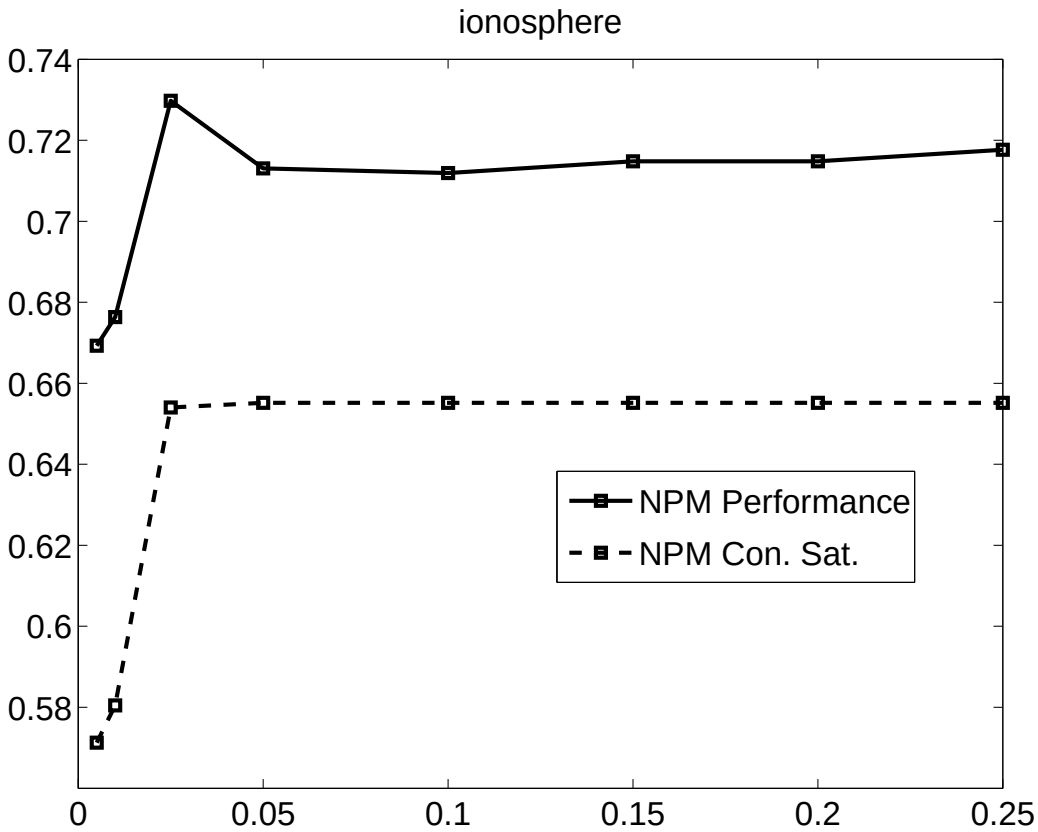
Dataset	Spectral Mean F_1	(std)	NMM Mean F_1	(std)
ionosphere	69.83	(0.0)	**71.31**	(0.1)
adult	79.43	(0.0)	76.21	(8.6)
austra	80.63	(0.0)	**81.87**	(0.1)
bupa	52.26	(0.0)	**53.25**	(0.3)
german	54.69	(0.0)	54.23	(0.0)
heart	73.44	(13.2)	**83.59**	(0.2)
texture	73.71	(10.7)	**95.68**	(0.0)
uci image	34.20	(4.7)	**56.62**	(1.7)
vehicle	31.39	(0.1)	**40.25**	(2.1)
classic300	86.02	(0.0)	86.36	(0.0)
different-100	93.31	(0.1)	**95.64**	(0.2)
same-100	53.95	(3.4)	52.38	(1.9)
similar-100	54.67	(1.1)	**62.01**	(3.1)
different-1000s	94.20	(0.1)	95.86	(0.0)
similar-1000s	65.58	(5.7)	67.72	(5.5)
same-1000s	66.80	(8.6)	**72.16**	(2.1)

Table 4.5: Mean and standard deviation of the performance of Spectral Clustering vs. NMM approach. The bandwidth is selected using the maximum pairwise constraint satisfaction heuristic. Significant differences (paired t-test, 95% confidence) are shown in boldface.



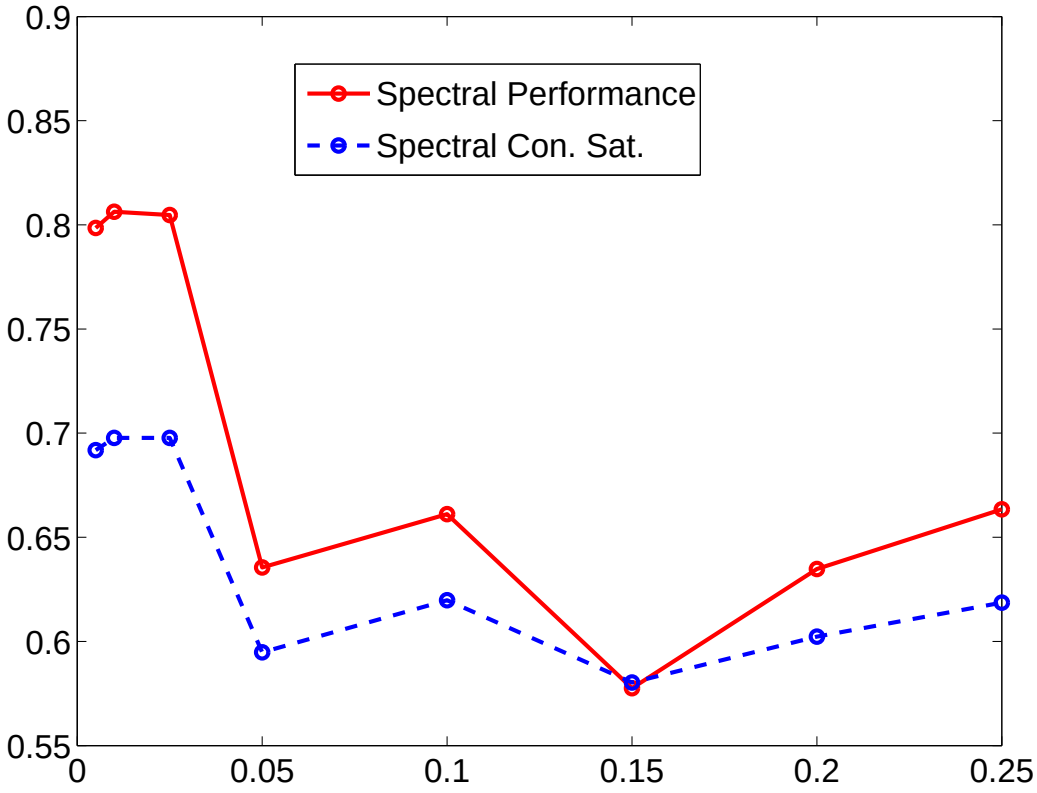
(a) Australian Credit (UCI austra)

Figure 4.5: Clustering performance and Constraint Satisfaction on UCI datasets for the proposed NPM algorithm.



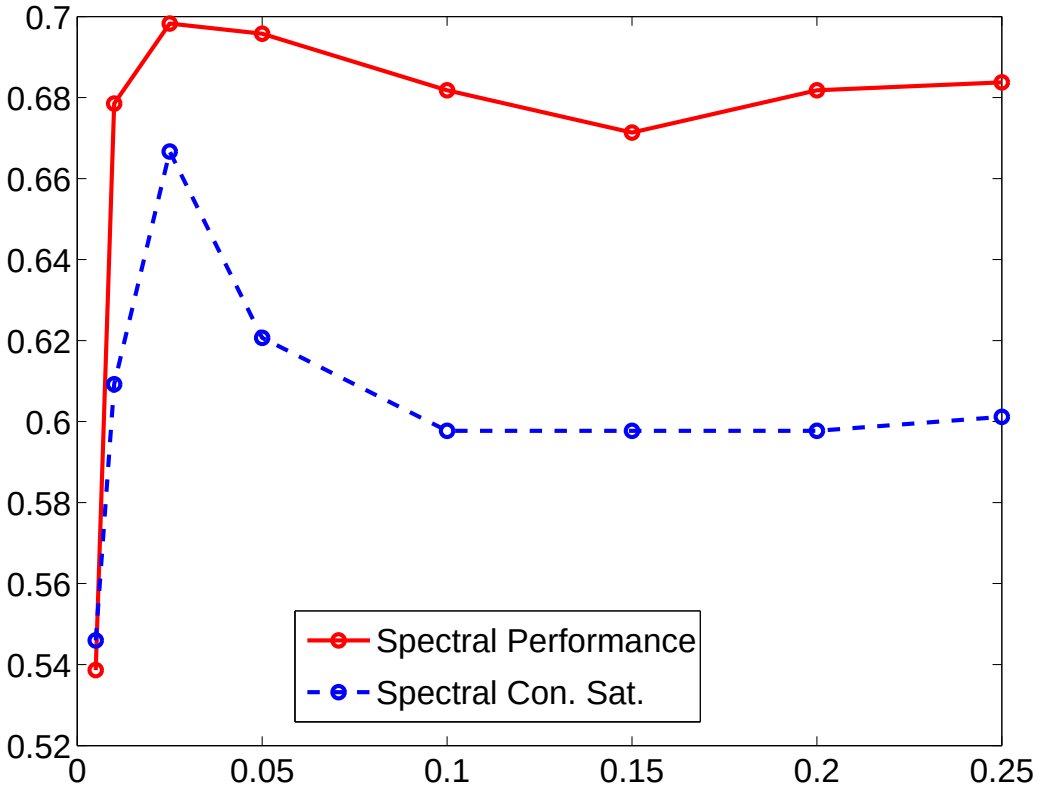
(b) Ionosphere (UCI)

Figure 4.5: (Continued from previous page...) Clustering performance and Constraint Satisfaction on UCI datasets for the proposed NPM algorithm.



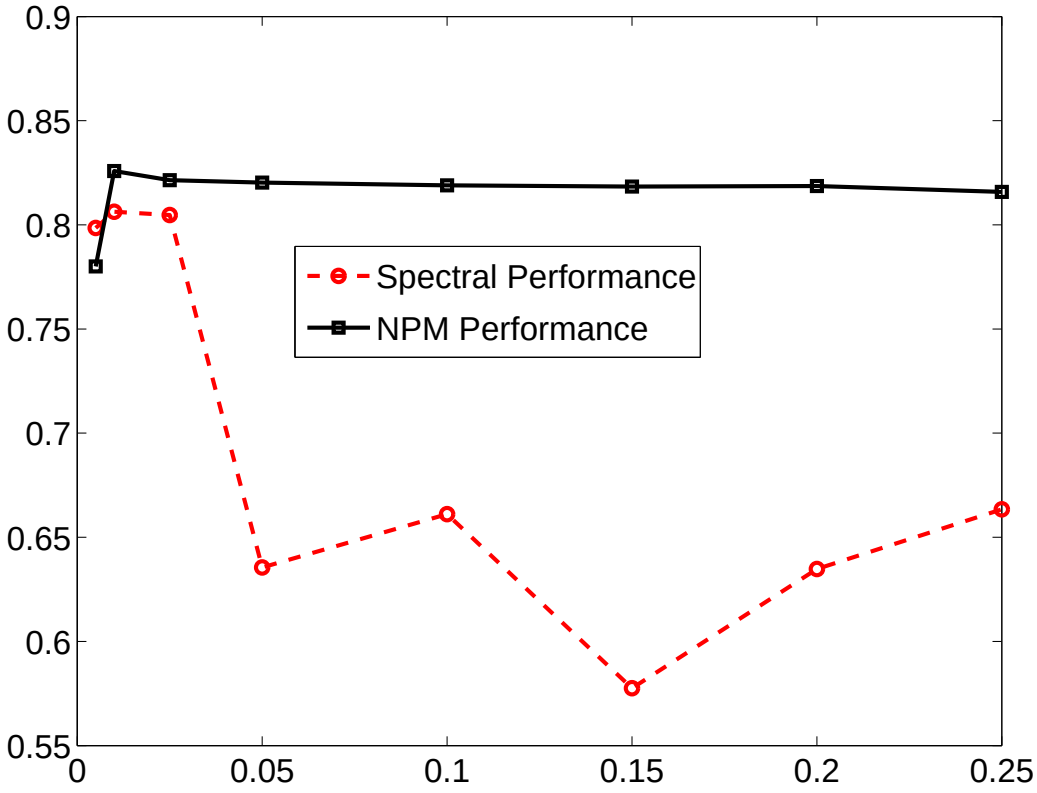
(a) Australian Credit (UCI austra)

Figure 4.6: Clustering performance and Constraint Satisfaction on UCI datasets for spectral clustering.



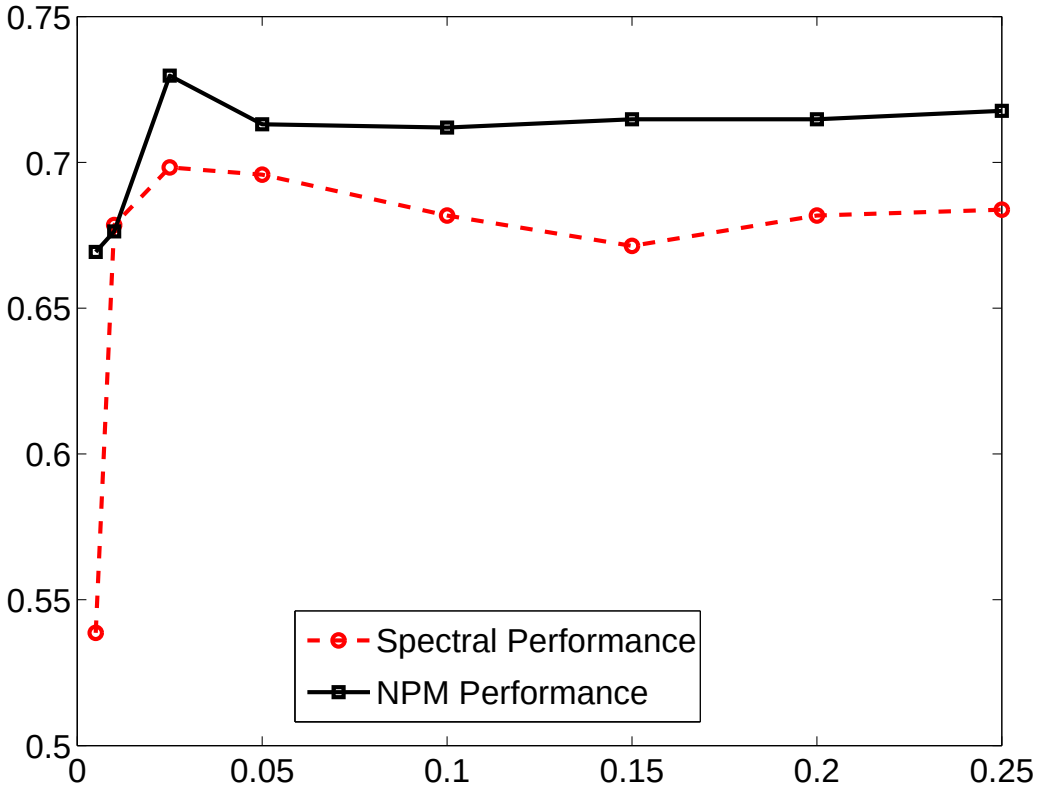
(b) Ionosphere (UCI)

Figure 4.6: (Continued from previous page...) Clustering performance and Constraint Satisfaction on UCI datasets for spectral clustering.



(a) Australian Credit (UCI austra)

Figure 4.7: Performance comparison between spectral clustering and the proposed non-parametric mixture model.



(b) Ionosphere (UCI)

Figure 4.7: Performance comparison between spectral clustering and the proposed non-parametric mixture model.

ing. However, these simplifications are based on some approximations, so we cannot claim the superiority of non-parametric mixture model for clustering over spectral clustering and K-means. Still, this connection at the objective function level between the three clustering algorithms opens up avenues for further exploration of non-parametric mixture models.

4.5.1 Approximation to weighted K-means

A slightly different relaxation of Eq (4.8) results in the conventional weighted K-means clustering. Since we have the constraint that $\sum_i q_i = 1$, we can use the Jensen's inequality to take the log inside,

$$\log P(\mathcal{D}) \geq \sum_{i=1}^{n} \sum_{g=1}^{G} \gamma_i^g \sum_{j=1}^{n} q_j^g \log K_{ij}.$$

For a Gaussian kernel, we have $\log K_{ij} = -(x_i - x_j)^T \Sigma^{-1} (x_i - x_j)$. Simplifying it further, we have $\log K_{ij} = -\frac{1}{\sigma^2} ||x_i - x_j||^2$. Therefore,

$$\log P(\mathcal{D}) \quad \geq \quad -\frac{1}{\sigma^2} \sum_{i=1}^{n} \sum_{g=1}^{G} \sum_{j=1}^{n} \gamma_i^g q_j^g ||x_i - x_j||^2.$$

Using the definition of γ_i^g and the inequality $\sum_{j=1}^{n} q_j^g |x_i - x_j|^2 \geq |x_i - \sum_{j=1}^{n} q_j^g x_j|^2$, we could further simplify the above equation as follows

$$-\log P(\mathcal{D}) \leq \sum_{g=1}^{G} \sum_{x_j \in C_g} ||x_j - \mu_g||^2,$$

where $\mu_k = \sum_{j=1}^{n} q_j^g x_j$. The right hand side in the above equation is the objective function for weighted K-means. Maximizing the likelihood in the proposed approach, therefore loosely minimizes the scatter between the data points. The key step here is the use of Jensen's inequality; the log eliminates the effect of the exponential in the kernel, thereby removing the "localization" introduced by the kernel. Note that the effect of σ is completely lost. As a result, all the points have the same effect on each other.

4.6 Summary

We have proposed a non-parametric mixture model, called NMM, for data clustering. It is a probablistic model that clusters the data by fitting kernel density estimate to each cluster. We have evaluated its performance on 26 standard datasets with large differences in dimensionality, number of data points and cluster structure. Experimental results show that the NMM based clustering performs well against some of the well known clustering algorithms (K-means, spectral, GMM and hierarchical). The non-parametric mixture model opens up a wide range of possible theoretical analysis related to data clustering.

CHAPTER 5

Incremental Algorithm for Feature Selection

5.1 Introduction

Online algorithms provide an efficient way to continuously learn from examples as and when they become available. In this chapter, we address the problem of feature selection using an incremental algorithm. In particular, we aim to perform feature selection on images represented as a *bag of visual words* [157]. The algorithm is derived using the framework of online learning [173]. Therefore, most of the chapter follows the online learning terminology. However, the algorithm is applied in an incremental fashion as described in Section 5.5.

Representing images using a bag of visual words [174] has received significant attention. In this approach, each image is represented as a distribution over a set of visual vocabulary. The vocabulary itself is a set of prototypes obtained by clustering the set of key points (e.g., using SIFT operator) pooled from a collection of training images. Several applications such as image clustering [156], large scale image [175] and video retrieval [159]

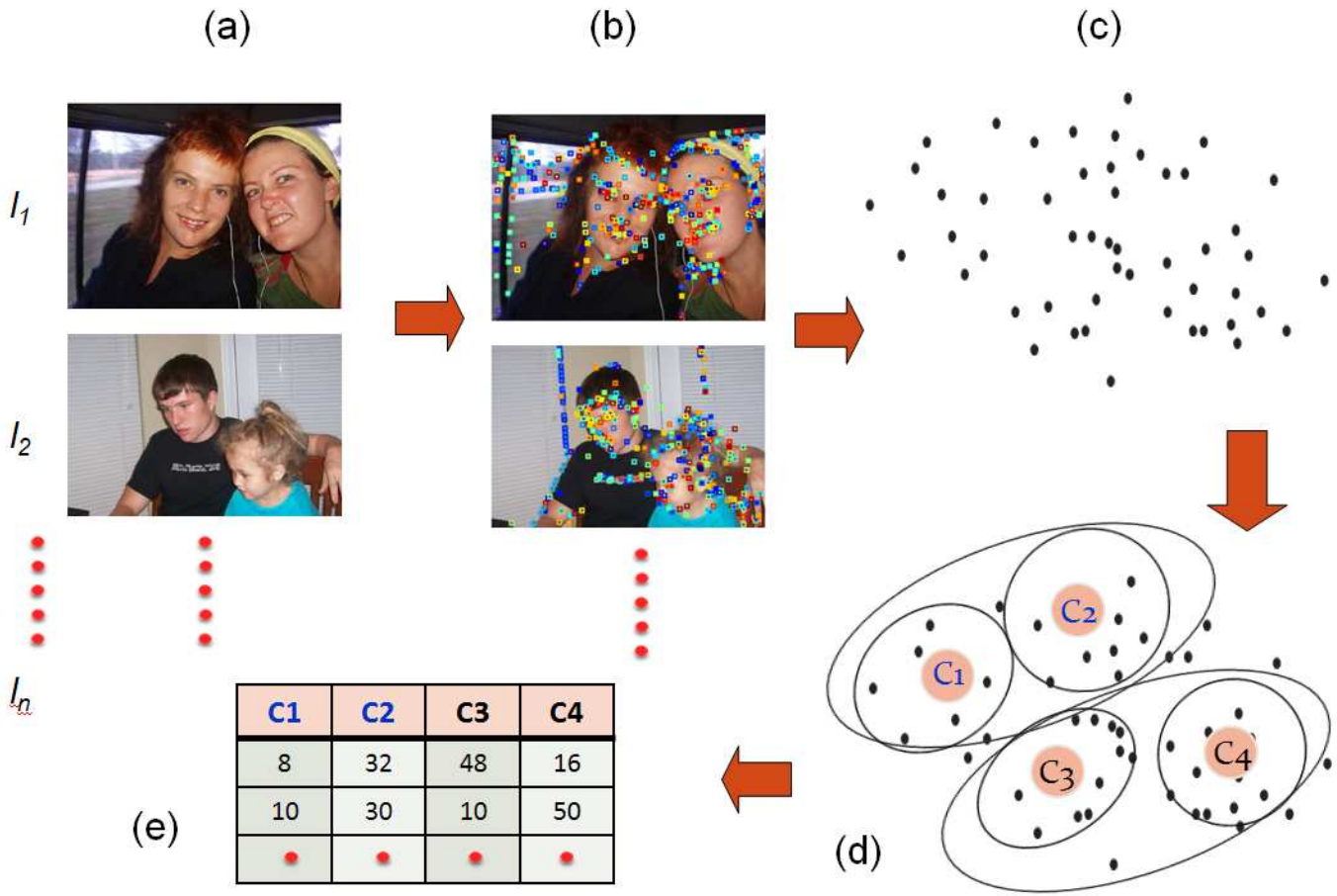


Figure 5.1: Steps involved in constructing a bag-of-visual-words representation of images. (a) Given collection of images. (b) Features at key points are extracted (the SIFT operator [2] is a popular choice). (c) The key points are pooled togheter. (d) The key points are clustered using hierarchical K-means algorithm. The centroids obtained after clustering the key points are called the visual words. (e) The features in the key points in each image are assigned a cluster label, and the image is represented as a frequency histogram over the cluster labels. The centroids sharing common parent are considered similar to each other, and are called *visual synonyms*. Visual synonyms are shown using the same color in the table.

have shown this method to be promising in both performance and scalability.

Recent studies have shown that the choice of vocabulary size can have a significant impact on the performance of learning algorithms [176]. A small vocabulary size may result in a feature space not rich enough to capture the variability in the images in the database, while a large vocabulary may cause two keypoints that are similar to be mapped to two different visual words leading to suboptimal performance. Further, a large number of visual words results in the well known problems of curse of dimensionality, complex hypothesis spaces and large computational requirements. Feature selection, or vocabulary pruning, is an important step that retains only those salient words needed for subsequent image matching or retrieval [177].

Visual vocabularies are usually constructed using recursive partitional clustering algorithms such as bisecting K-means, resulting in a cluster hierarchy [175, 160]. This causes the visual words at the leaf nodes that are children of a common parent to be similar to each other. If one of the visual words is not informative, it is an indication that its siblings may not be informative as well. One of the basic premises of this work is to exploit what we call *visual synonyms* for feature selection. Visual synonyms are identified as the visual words sharing a common parent in the cluster hierarchy.

We propose to use pairwise constraints to encode the relationship between images. The pairwise constraints are used to identify the subset of visual words that explain the similiarity or dissimilarity between the corresponding images. Since we perform feature selection using unlabeled images and the given pairwise constraints for a small number of images, the input setting resembles a semi-supervised clustering scenario. However, it should be noted that the feature selection is performed only using the labeled pairs, and hence is somewhat supervised. The pairwise constraints are of two types: *must-link* and *cannot-link*. A pairwise constraint is a natural way to encode a user's perceived visual similarity between a pair of images. It is easier to specify a constraint between two images than labeling them explicitly with some keywords based on all the objects present in it.

Figure 5.2 illustrates the goal of the proposed approach using an image pair labeled as *must-link*. Loosely speaking, the common key points between a pair of images need to be discarded if they form a cannot-link pair, and need to be retained if they are a must-link pair.

In this paper, we propose an efficient online algorithm that takes in a stream of images and the associated pairwise constraints, and selects a subset of visual words. Since each key point in an image is mapped to one of the visual words, pruning the vocabulary results in a reduction in the number of key-points in an image. The feature group information obtained from the cluster hierarchy is exploited to shrink the feature weights at a group level. The quality of the selected features is evaluated using an image clustering application.

5.2 Related work

Dimensionality reduction is a classical problem in multivariate statistics and pattern recognition. All disciplines of learning, i.e. supervised, unsupervised, and semi-supervised usually perform some sort of dimensionality reduction. Dimensionality reduction techniques can be broadly classified into feature selection or feature extraction. In feature selection, the goal is to obtain the most salient subset of features from the available feature set. The size of the subset is usually specified by the user. An introduction to feature or variable selection can be found in [178, 177, 179]. Feature extraction, in contrast with feature selection, identifies a (non)linear combination of existing features. This could be followed by a feature selection to reduce the dimensionality of the extracted feature vector Most feature extraction methods aim to learn a transformation from the input feature space to a sub-space of smaller dimensionality such that a desirable criterion is maximized. For example, Principal Component Analysis (PCA) [20] is an unsupervised linear dimensionality reduction method that retains the maximum possible variance of the data in the input space when projected into the sub-space. Linear Discriminant Analysis (LDA) [20] is a

supervised linear dimensionality reduction technique that finds a projection to a subspace such that the separation between the classes is maximized while the within class variance is minimized. In applications requiring interpretability of the features used, feature selection is preferred to feature extraction.

Feature selection methods can be broadly classified into *search based* methods (e.g. floating search [179]), *feature ranking*, and *shrinkage* methods such as LASSO [180] and Group LASSO [181]. Feature selection by ranking sorts the features based on a score, such as correlation coefficient or mutual information, computed between the feature and the class labels. While feature ranking is commonly used as a baseline, features that are correlated with the labels are possibly correlated among themselves as well, resulting in the selection of a set of redundant features [177].

Search based methods are further classified into *filter* and *wrapper* methods. They operate by incrementally modifying a selected set of features by adding or deleting features one by one. These approaches are greedy in nature, and are affected by the order of adding/deleting features to/from the set. Moreover, they are computationally expensive as the learning algorithm is run every time the selected feature set is modified. Branch and bound algorithms tend to be more accurate, but are limited in their ability to handle only a small set of features due to computational reasons. Search based algorithms are batch mode, and require all the labeled data examples be present before they can be used, and are not applicable to an online setting.

Shrinkage methods are widely used for variable selection in multivariate regression. These tend to be more principled, and amenable to theoretical analysis with a predictable behavior. In general, supervised learners such as SVM, learn the weights of features. Feature selection, however differs from feature weighting. Shrinkage methods such as LASSO perform feature selection by driving as many weights to zero as possible. In a supervised setting, several algorithms such as 1-norm SVMs [182], F_∞ SVM[183] and Lasso Boosting [184], ridge regression employ shrinkage strategy. To the best of our knowledge, there

is no feature selection method proposed in the literature that employs LASSO shrinkage with pairwise constraints.

Distance metric learning (DML) is another related area where the features weights are learnt from labeled examples [123, 185]. DML methods learn a quadratic distance function parameterized using a $d \times d$ weight matrix, where d is the dimensionality of the data. Online DML algorithms such as POLA [123] involve a projection step to ensure positive definiteness of the feature matrices, and are computationally expensive. Even using a diagonal weight matrix, they tend to prefer uniform feature weights, contrary to our goal. The proposed algorithm can be shown to be a generalization of the POLA algorithm with diagonal weight matrix, when all the visual words are put in a single group.

5.2.1 Review of Online Learning

Learning algorithms can be divided into two groups based on how they recieve (or utilize) the input data – *batch algorithms* and *online algorithms*. Batch techniques (e.g. MLE, MAP) require the processing of all the input training data at a time. This could be inefficient (or even infeasible) for large datasets. On the other hand, online learning techniques process only one data item at each time. This has two advantages:

- They are computationally more efficient for large datasets.
- They are applicable to scenarios where all the data is not available at once, but arrives in a stream (e.g. images uploaded to the Internet on a daily basis, video streams from surveilance cameras etc.).

Given a hypothesis space $\mathcal{H}$, batch mode algorithms select a classification function from $\mathcal{H}$ such that the training error on a given dataset $\mathcal{X}$ is minimized, while optionally satisfying an additional regularity condition on the hypothesis. This error minimization can be performed using gradient descent methods. Batch mode algorithms compute the

Algorithm 2 SupervisedOnlineLearning

input: examples $\mathbf{x}_t$, one per round of learning.
parameters: loss function $\ell(f(\mathbf{w}, \mathbf{x}), y)$, stepsize λ, hypothesis set $\mathcal{H}$
output: parameters of the classification model $\mathbf{w} \in \mathcal{H}$.
Initialize $\mathbf{w} \leftarrow 0, t \leftarrow 0$
for each round $t = 1, 2, \cdots$ **do**
  Observe $\mathbf{x}_t$ and Predict $y_t = f(\mathbf{w}, \mathbf{x}_t)$.
  if $\ell(f(\mathbf{x}_t, y_t), y_t) \geq 0$ **then**
    $\mathbf{w}'_t \leftarrow$ UpdateW$(\mathbf{w}_{t-1}, \mathbf{x}_t, \lambda)$
    $\mathbf{w}_t \leftarrow$ Project$(\mathbf{w}'_t, \mathcal{H})$
  end if
  $t \leftarrow t + 1$
end for

gradient of the training error (or a surrogate) on all the examples at each descent step. In contrast, online algorithms perform *stochastic gradient descent* (under the assumption that the samples are $i.i.d.$), where the gradient of the training error (or its surrogate) is computed using only one example each time. This example is selected or received randomly and hence the name *stochastic*. The gradient computed using one data item may be seen as a coarse approximation to the *true* gradient computed using all the exmaples.

Let the data point at time t be represented using $\mathbf{x}_t \in \mathcal{X}$ and the output space be represented by $\mathcal{Y}$. Let $f : \mathcal{X} \rightarrow \mathcal{Y} \in \mathcal{H}$ is a function that we want to estimate such that the loss $\ell(f(\mathbf{x}_t), y_t)$ is minimized. Further, let us assume that the functions in the space $\mathcal{H}$ are parametrized by parameters $\mathbf{w} \in \mathcal{S}$. For example, the class of binary linear classification functions, i.e., $\mathcal{X}, \mathcal{S} \subset \mathbb{R}^d$, where d is the dimension of the input feature space, where $f(\mathbf{w}, \mathbf{x}) = \mathbf{w}'\mathbf{x}$, $\mathbf{x} \in \mathcal{X}$ and $\mathbf{w} \in \mathcal{S}$. The general structure of the online algorithms for this setting is summarized in Algorithm 2.

The motivation for the further analysis and theory of online learning comes from the following special case. Consider a set up where there are d experts (corresponding to d-features) and the task is to learn a set of weights to combine the experts where the weighted average of the experts is a more accurate prediction than any of the individual experts. In this case, $\mathbf{x}_t in \mathcal{X} \subset \mathbb{R}^d$ is a d-dimensional vector containing decisions of experts decisions.

We can now define a useful quantity called the *instantaneous regret vector* $\mathbf{r}_t \in \mathbb{R}^d$ as the loss of individual experts at time t,

$$\mathbf{r}_t = (r_{1,t}, r_{2,t}, \ldots, r_{d,t}) \tag{5.1}$$

where $r_{i,t}$ is the regret (or loss) of not choosing the i-th expert's decision as final decision. The *cumulative regret vector* can now be defined as the sum of instantaneous regrets over all the rounds from $t = 1, \cdots, T$,

$$\mathbf{R}_t = \sum_{t=1}^{T} \mathbf{r}_t \tag{5.2}$$

These regret vectors perform an important function in determining the weights of individual experts. Naturally, the higher the regret of a particular expert, the more the weight it should be assigned in the next round. Therefore, one can assign the weights to the experts based on the cumulative regret vectors, that is,

$$\mathbf{w}_t = R_{t-1} \tag{5.3}$$

Let $\Phi : \mathcal{A} \subset \mathbb{R}^d \rightarrow \mathbb{R}$ be a strongly convex function with some additional properties (Legendre function). Since the gradient of $\Phi(.)$ is monotonically increasing, we may also assign the weights as

$$\mathbf{w}_t = \nabla\Phi(R_{t-1}). \tag{5.4}$$

The function Φ is called the potential function, and is useful in incorporating useful properties into $\mathbf{w}$. Many existing online learning algorithms such as Perceptron algorithm [20] or the Exponential-Gradient algorithm can be shown as specializations of a single algorithm with different choices of Φ.

Since Φ is chosen to be a Legendre function [173], we can compute the a Legendre dual of Φ, denoted by Φ^* as follows:

$$\Phi^*(\mathbf{u}) = \sup_{\mathbf{v} \in \mathcal{A}} \left(\mathbf{u} \cdot \mathbf{v} - \Phi(\mathbf{v})\right). \tag{5.5}$$

The following relationship holds true between a function Φ and its Legendre dual Φ^*, and is used heavily in the subsequent theory and algorithms:

$$\nabla\Phi^* = (\nabla\Phi)^{-1}. \tag{5.6}$$

Using the relation from Eq (5.6), the dual relationship between regret and the weight vector can be written as,

$$\mathbf{w}_t = \nabla\Phi(\mathbf{R}_t) \tag{5.7}$$

$$\mathbf{R}_t = \nabla\Phi^*(\mathbf{w}_t). \tag{5.8}$$

Potential-Based Gradient Descent

The goal of online learning is to minimize the regret. By definition of cumulative regret vector, we know that,

$$\mathbf{R}_t = \mathbf{R}_{t-1} + \mathbf{r}_t. \tag{5.9}$$

Using the duality relation, we can write

$$\nabla\Phi^*(\mathbf{w}_t) = \nabla\Phi^*(\mathbf{w}_{t-1}) + \mathbf{r}_t. \tag{5.10}$$

The key idea of potential based gradient descent is that the problem of *regret minimization* in the weight space $\mathbf{w}$ is converted to a gradient minimization in the regret space. Further, since the loss function is chosen to be convex, minimizing the gradient implies minimizing the loss. Since we want to minimize the regret, writing the cumulative regret vector $\mathbf{R}_t$ as a paratmeter θ_t (for convenience of notation), the dual gradient update can be written as,

$$\theta_t = \theta_{t-1} - \lambda\nabla\ell_t(\mathbf{w}_{t-1}). \tag{5.11}$$

From the relation between θ_t and $\mathbf{w}_t$ expressed in Eq (5.7), can be written as,

$$\nabla\Phi^*(\mathbf{w}_t) = \nabla\Phi^*(\mathbf{w}_{t-1}) - \lambda\nabla\ell_t(\mathbf{w}_{t-1}). \tag{5.12}$$

Potential	Name	Algorithm
$\frac{1}{2}\|\mathbf{u}\|^2$	Polynomial potential	Perceptron, Widrow-Hoff
$\sum_i \exp u_i$	Exponential potential	Exponential gradient

Table 5.1: Choice of potential and the resulting classical online learning algorithms

The weights at iteration t, $\mathbf{w}_t$, can be obtained using the following equation:

$$\mathbf{w}_t = (\nabla\Phi^*)^{-1}\left(\nabla\Phi^*(\mathbf{w}_{t-1}) - \lambda\nabla\ell_t(\mathbf{w}_{t-1})\right) \quad (5.13)$$

$$= \nabla\Phi\left(\nabla\Phi^*(\mathbf{w}_{t-1}) - \lambda\nabla\ell_t(\mathbf{w}_{t-1})\right). \quad (5.14)$$

Online algorithms can be derived by specifying appropriate Φ to encode the desired characteristics of the solution.

5.3 Problem formulation

Let $\mathcal{D} = \{\mathcal{I}_1, \ldots, \mathcal{I}_n\}$ be the given collection of n images represented as a distribution over the visual vocabulary $\mathcal{V} = (v_1, \ldots, v_m)$ containing m visual words. Since the visual words are often generated by a recursively bisecting K-means algorithm, we can derive a group structure for the visual words. In particular, we assume the visual words are divided into s groups. Let $\mathbf{v}^g = (v_{m_{g-1}+1}, \ldots, v_{m_g})$ be the collection of visual words belonging to the g-th group, for $g = 1, \cdots, s$, where $(m_{g-1} + 1)$ is the index of the first element in the g-th group. Note that even when no group structure is available, our method is still applicable where each feature forms its own group, and $s = m$. Given the visual words, each image $\mathcal{I}_i$ is represented by a vector of visual word histogram, denoted by $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,m})$. Further, let $\mathbf{x}_i^g$ denote the feature sub-vector of image $\mathcal{I}_i$ corresponding to the vocabulary $\mathbf{v}^g$. Let $\mathbf{w} = (w_1, \ldots, w_m)$ denote the weights for visual words. The squared distance between two visual word histograms $\mathbf{x}$ and $\mathbf{x}'$ given the feature weights $\mathbf{w}$, denoted by

$|\mathbf{x} - \mathbf{x}'|^2_{\mathbf{w}}$, is computed as

$$|\mathbf{x} - \mathbf{x}'|^2_{\mathbf{w}} = \sum_{i=1}^{m} w_i (x_i - x'_i)^2. \tag{5.15}$$

It is necessary that the weights are non-negative, $w_j \geq 0$, $j = 1, \ldots, m$, for Eq (5.15) to be a metric. The visual similarity between a pair of images is provided in the form of a pairwise constraint – a must-link constraint indicates two images are visually similar whereas a cannot-link constraint indicates two images are visually different. Let $\mathcal{T} = \{(\mathbf{x}_t, \mathbf{x}'_t, y_t), t = 1, \ldots, T\}$ denote the collection of pairwise constraints that will be used for learning the weights, where $\mathbf{x}_t$ and $\mathbf{x}'_t$ are visual word histograms corresponding to two images, and $y_t = \pm 1$, where $+1$ indicates the two images are visually similar and -1, otherwise.

The goal is to learn weights $\mathbf{w}$ for the visual words such that the following criteria are met:

1. The distance between the two images computed using Eq (5.15) reflects the visual similarity between the images.

2. Select a small subset of features by driving as many entries in the vector $\mathbf{w}$ to 0 as possible.

For a given a pairwise constraint $(\mathbf{x}_t, \mathbf{x}'_t, y_t)$, if $y_t = 1$, the distance between $\mathbf{x}_t$ and $\mathbf{x}'_t$ must be less than a threshold b (which can either be learnt, or specified by the user). On the other hand, if $y_t = -1$, the distance computed using the selected features must be greater than b. We define a loss function measuring the error made by a weight vector $\mathbf{w}$ on an example pair $(\mathbf{x}_t, \mathbf{x}'_t)$ with true label y_t as follows:

$$\ell(\mathbf{w}; \mathbf{x}_t, \mathbf{x}'_t, y_t) = \max\left(0, 1 - y_t(b - |\mathbf{x}_t - \mathbf{x}'_t|^2_{\mathbf{w}})\right). \tag{5.16}$$

In order to encode the hierarchical structure among visual words, we introduce a *mixed norm* for weight vector w, denoted by $\|w\|_{1,2}$, that is defined as follows:

$$\|\mathbf{w}\|_{1,2} = \sum_{g=1}^{s} \sqrt{\sum_{j=m_{g-1}+1}^{m_g} w_j^2} \tag{5.17}$$

The above norm is introduced to enforce feature selection at a group level, i.e., if multiple visual words within a group are assigned small weight, the entire group of visual words may be deemed irrelevant and can be discarded. This mixed norm is often referred to as group-lasso or the $L_{1,2}$ norm and is widely used for feature selection [186].

Using the norm defined in Eq (5.17) as the regularizer and the loss defined in Eq (5.16), the feature weights can be learnt by minimizing the following objective function:

$$\min_{w \in \mathbb{R}^m_+} \|\mathbf{w}\|^2_{1,2} + \lambda \sum_{t=1}^{T} \ell(\mathbf{w}; \mathbf{x}_t, \mathbf{x}'_t, y_t) \tag{5.18}$$

where $b > 0$ is a predefined constant. Our goal is to present an online algorithm to minimize Eq (5.18).

5.4 Online algorithm using projections

Our online feature selection algorithm is presented in Section 5.4.1, followed by a theoretical analysis in Section 5.4.2. For conciseness, we define $\Delta\mathbf{x}_t = \mathbf{x}_t - \mathbf{x}'_t$ and use the notation $\ell_t(\mathbf{w})$ to denote the loss $\ell(\mathbf{w}; \mathbf{x}_t, \mathbf{x}'_t, y_t)$ at the t-th round of learning. Algorithm 3 summarizes the general online feature selection framework.

5.4.1 Algorithm

Step 1. Given a pair of images $(\mathbf{x}_t, \mathbf{x}'_t)$, predict whether they are in the same cluster using the existing weight vector $\mathbf{w}_t$ and Eq (5.15). Observe the true output y_t, and compute the loss $\ell_t(\mathbf{w})$.

Step 2. For convenience, define temporary weights $\boldsymbol{\theta}_t = (\boldsymbol{\theta}^1_t, \boldsymbol{\theta}^2_t, \cdots, \boldsymbol{\theta}^s_t)$, where $\boldsymbol{\theta}^g$ is the subvector corresponding to group g, as follows:

$$\boldsymbol{\theta}^g_t = \|\mathbf{w}_t\|_{1,2} \frac{\mathbf{w}^g_t}{\|\mathbf{w}^g_t\|_2}, g = 1, \cdots s. \tag{5.19}$$

Step 3. Since the gradient of the loss function $\nabla_{\mathbf{w}}\ell_t(\mathbf{w}) = y_t\Delta\mathbf{x}_t$ indicates the direction for updating weights, the temporary weights are updated using the following rule

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \lambda\nabla_{\mathbf{w}}\ell_t(\mathbf{w}_t) = \boldsymbol{\theta}_t - \lambda y_t\Delta\mathbf{x}_t \tag{5.20}$$

where λ is a prespecified stepsize or the learning rate.

Step 4. To perform group level feature selection, each group is weighted by a factor that depends on the norm of the feature weights within the group. In particular, we compute the weight of each group using a soft-max function. That is, the weight of group g, q^g is obtained as,

$$q^g = \frac{\exp(\|\boldsymbol{\theta}^g\|_2^2/2\mu)}{\sum_{k=1}^{s}\exp(\|\boldsymbol{\theta}^k\|_2^2/2\mu)}. \tag{5.21}$$

The smoothing parameter μ in the softmax function controls the distribution of group weights. For a large μ all groups are weighted equally irrespective of their utility, and as μ goes to zero, only one group whose weights have the largest norm is selected.

Step 5. Since the weights for the features must be positive, replace all the negative elements in $\boldsymbol{\theta}$ with 0. Compute the weight vector $\mathbf{w}_{t+1}$from the temporary weights $\boldsymbol{\theta}$ as follows:

$$\mathbf{w}_{t+1}^g = q^g\frac{\boldsymbol{\theta}_t^g}{\|\boldsymbol{\theta}_t^g\|_2^2}, \quad g = 1, \cdots, s. \tag{5.22}$$

where $\mathbf{w}_{t+1}^g$ is the g-th subvector of $\mathbf{w}$ corresponding to the vocabulary of g-th group, $\mathbf{v}^g$.

Eq (5.22) gives the solution for the weight vector $\mathbf{w}_{t+1}$ for the next iteration. Steps 1-5 are repeated as each example pair (constraint) becomes available. The features corresponding to non-zero weights in $\mathbf{w}$ are considered relevant, and form the selected subset of features.

5.4.2 Theoretical analysis

Potential based gradient descent [173, Chapters 2,11] is an online learning framework that generalizes several classical algorithms like Widrow-Hoff, Winnow and the recent Expo-

nentiated Gradient (EG) algorithm [173]. However, classical analysis presented in [173] is applicable only to potential functions that are strictly convex. The potential generating the $L_{1,2}$ norm considered in the proposed approach is not strictly convex. In this section, we propose a smooth approximation to the mixed norm $\|\cdot\|_{1,2}^2$ for weight estimation. We begin with the following lemma that allows us to rewrite $\|w\|_{1,2}^2$ as a variational minimization problem.

Lemma 1. *The group-LASSO norm can be shown to be the exact minimizer of the variational problem*

$$\frac{1}{2}\|\mathbf{w}\|_{1,2}^2 = \frac{1}{2}\min_{p\in\mathbb{R}_+^s}\left\{\sum_{g=1}^{s}\frac{\|\mathbf{w}^g\|_2^2}{p_g} : \sum_{g=1}^{s} p_g = 1\right\} \tag{5.23}$$

Proof. Introducing the Lagrangian multiplier λ and setting the partial derivative of the Lagrangian function to be zero, we have

$$p_g = \frac{|w^j|_2}{\sum_{k=1}^{s}|w^k|_2}$$

Plugging the above expression into $\sum_{g=1}^{s}|w^g|_2^2/p_g$, we have the result in the lemma. □

A smoothing term is now introduced to ensure that the norm $\|\cdot\|_{1,2}^2$ is strictly convex. The smooth norm $\Phi(\mathbf{x},\mu)$ is defined as follows:

$$\Phi(\mathbf{w};\mu) = \min_{p\in\mathbb{R}_+^s}\left\{\sum_{g=1}^{s}\frac{\|\mathbf{w}^g\|_2^2}{2p_g} - \mu H(p) : \sum_{g=1}^{s} p_g = 1\right\} \tag{5.24}$$

where $H(p)$ is the Shannon entropy defined as $H(p) = -\sum_{g=1}^{s} p_g \ln p_g$, and μ is the *smoothness* parameter. Also, we have $\frac{1}{2}\|\mathbf{w}\|_{1,2}^2 - \mu\ln s \le \Phi(\mathbf{w};\mu) \le \frac{1}{2}\|\mathbf{w}\|_{1,2}^2$.

Lemma 2. *The approximate potential function $\Phi(\mathbf{w},\mu)$ is a strictly convex function.*

Proof. Define $Q = \{p \in \mathbb{R}_+^s : \sum_{g=1}^{s} p_g = 1\}$. According to the definition, we have

$\Phi^*(w,\mu)$ computed as

$$\begin{aligned}
\Phi^*(w,\mu) &= \max_v \langle w, v\rangle - \Phi(v,\mu) \\
&= \max_{p\in Q} \max_v \sum_{k=1}^{s} \langle w^k, v^k\rangle - \frac{|v^k|_2^2}{2p_k} - \mu p_k \ln p_k \\
&= \max_{p\in Q} \sum_{k=1}^{s} \frac{p_k}{2}|w^k|_2^2 - \mu p_k \ln p_k \\
&= \mu \ln \left(\sum_{k=1}^{s} \exp\left[\frac{|w^k|_2^2}{2\mu}\right]\right)
\end{aligned}$$

□

The following lemma shows the convex conjugate of the smooth norm, which is shown to be strictly convex in the subsequent lemma.

Lemma 3. *The convex conjugate of the smooth norm* $\Phi(\mathbf{w},\mu)$*, denoted by* $\Phi^*(w,\mu)$ *is computed as*

$$\Phi^*(\mathbf{w},\mu) = \mu \ln \left(\sum_{g=1}^{s} \exp\left[\frac{\|\mathbf{w}^g\|_2^2}{2\mu}\right]\right)$$

Proof. It suffices to show that for any v, we have $\langle v, H^*(w,\mu)v\rangle > 0$. We thus compute $\langle v, H^*(w,\mu)v\rangle$ as

$$\langle v, H^*(w,\mu)v\rangle = \sum_{k=1}^{s} q_k |v^k|_2^2 + \frac{1}{\mu}\left(\sum_{k=1}^{s} q_k [\langle v_k, w_k\rangle]^2 - \left[\sum_{k=1}^{s} q_k \langle v_k, w_k\rangle\right]^2\right)$$

where

$$q_k = \frac{\exp\left(|w^k|^2/[2\mu]\right)}{\sum_{j=1}^{s} \exp\left(|w^j|^2/[2\mu]\right)}$$

□

Note that as μ goes to zero, $\Phi^*(\mathbf{w},\mu)$ becomes $\max_{1\le g\le s} |\mathbf{w}^g|_2^2$, which is the square of the mixture of the L_∞ and L_2 norm. This is interesting since L_∞ norm is the dual of L_1 norm. Lemma 4 below shows that $\Phi^*(\mathbf{w},\mu)$ is a strict convex function

Lemma 4. *The Hessian matrix of $\Phi^*(\mathbf{w}, \mu)$, denoted by $H^*(\mathbf{w}, \mu)$, is positive definite i.e.,$H^*(\mathbf{w}, \mu) \succ 0$. Furthermore, if $\|\mathbf{w}\|_2 \leq R$, we have $H^*(\mathbf{w}, \mu) \preceq (1 + R^2/\mu)I$*

Proof. Note that

$$\left(\sum_{k=1}^{s} q_k[\langle v_k, w_k\rangle]^2\right)\left(\sum_{k=1}^{s} q_k\right) \geq \left(\sum_{k=1}^{s} q_k\langle v_k, w_k\rangle\right)^2$$

Hence,

$$\langle v, H^*(w, \mu)v\rangle \geq \sum_{k=1}^{s} q_k|v^k|_2^2 \geq \exp\left(-\frac{\|w\|_2^2}{2\mu}\right) > 0$$

To show the upper bound of $H^*(w, \mu)$, we have

$$\langle v, H^*(w, \mu)v\rangle \leq (1 + R^2/\mu)|v|_2^2$$

□

Given that both potential $\Phi(\mathbf{w}, \mu)$, and its convex conjugate $\Phi^*(\mathbf{w}, \mu)$ are strictly convex functions, the potential based gradient descent algorithm presented in [173, Chapter 11] can be used. The algorithm is described in Algorithm 4, where $\Omega = \{\mathbf{w} \in \mathbb{R}_+^m : |\mathbf{w}|_2 \leq R\}$ is the domain for feature weights and $R \in \mathbb{R}$ is a predefined constant. Step 4 involves a projection of an estimate of weight vector $\mathbf{w}'_{t+1}$ into Ω, such that the Bregman divergence generated by the potential function Φ, denoted by $D_\Phi(\mathbf{w}_{t+1}, \mathbf{w}_t)$ is minimized.

An online learning algorithm performs a weight update whenever it makes a mistake in its prediction. Online learning algorithms are characterized by mistake bounds [173], which bound the number of mistakes made by an algorithm compared to those made by the knowledge of optimal weight vector in retrospect. The following theorem shows the mistake bound for the above online algorithm.

Theorem 2. *For any convex loss function ℓ, learning rate λ, and $X_\infty = \max_t \|\Delta\mathbf{x}_t\|$ where $\Delta\mathbf{x}_t = \mathbf{x}_t - \mathbf{x}'_t$, let $\kappa = (1 + R^2/\mu)$, and $\lambda = \epsilon/(\kappa X_\infty^2)$. For all $\mathbf{u} \in \Omega$, the number of*

mistakes M made by the proposed algorithm is bounded as follows:

$$M \leq \frac{1}{1-\epsilon}\left(\frac{\kappa X_\infty^2\left(\|\mathbf{u}\|^2 + \mu \ln s\right)}{2\epsilon} + \sum_{t=1}^{T} \ell_t(\mathbf{u})\right) \tag{5.25}$$

Proof. For each training example $(\Delta\mathbf{x}_t, y_t)$ that was misclassified by $\mathbf{w}_t$, we consider

$$\begin{aligned}
\ell_t(\mathbf{w}_t) - \ell_t(\mathbf{u}) &\leq \langle \mathbf{u} - \mathbf{w}_t, \nabla\ell_t(\mathbf{w}_t)\rangle \\
&= \frac{1}{\lambda}\langle \mathbf{u} - \mathbf{w}_t, \boldsymbol{\theta}'_{t+1} - \boldsymbol{\theta}_t\rangle \\
&= \frac{1}{\lambda}\langle \mathbf{u} - \mathbf{w}_t, \nabla\Phi(\mathbf{w}'_{t+1}, \mu) - \nabla\Phi(\mathbf{w}_t, \mu)\rangle \\
&= \frac{1}{\lambda}\left(D_\Phi(\mathbf{u}, \mathbf{w}_t) - D_\Phi(\mathbf{u}, \mathbf{w}'_{t+1}) + D_\Phi(\mathbf{w}_t, \mathbf{w}'_{t+1})\right) \\
&\leq \frac{1}{\lambda}\left(D_\Phi(\mathbf{u}, \mathbf{w}_t) - D_\Phi(\mathbf{u}, \mathbf{w}_{t+1}) + D_\Phi(\mathbf{w}_t, \mathbf{w}'_{t+1})\right)
\end{aligned}$$

where $D_\Phi(\mathbf{u}, \mathbf{v})$ stands for the Bregman distance from v to u. The second step follows the fact $\boldsymbol{\theta}'_{t+1} = \nabla\Phi(\mathbf{w}'_{t+1}, \mu)$, and the third step follows the property of Bregman distance function, i.e.,

$$D_\Phi(\mathbf{u}, \mathbf{v}) + D_\Phi(\mathbf{v}, \mathbf{w}) = D_\Phi(\mathbf{u}, \mathbf{w}) + \langle \mathbf{u} - \mathbf{v}, \nabla\Phi(w) - \nabla\Phi(\mathbf{v})\rangle$$

The last step follows the fact that $\mathbf{w}_{t+1}$ is the projection of w'_{t+1} onto the domain $[0, +\infty)^m$ based on the Bregman distance $D_\Phi(\mathbf{u}, v)$. Let $\mathcal{S}$ include the indices of the trials where the training examples are misclassified. We have

$$\sum_{t\in\mathcal{S}} \ell_t(\mathbf{w}_t) \leq \sum_{t=1}^{T} \ell_t(\mathbf{u}) + \frac{1}{\lambda} D_\Phi(\mathbf{u}, \mathbf{w}_1) + \frac{1}{\lambda}\sum_{t\in\mathcal{S}} D_\Phi(\mathbf{w}_t, w'_{t+1})$$

Using the property $D_\Phi(\mathbf{w}_t, \mathbf{w}_{t+1}) = D_{\Phi^*}(\boldsymbol{\theta}_t, \boldsymbol{\theta}_{t+1})$, we have

$$\sum_{t\in\mathcal{S}} \ell_t(\mathbf{w}_t) \leq \sum_{t=1}^{T} \ell_t(\mathbf{u}) + \frac{1}{\lambda} D_\Phi(\mathbf{u}, \mathbf{w}_1) + \frac{1}{\lambda}\sum_{t\in\mathcal{S}} D_{\Phi^*}(\boldsymbol{\theta}_t, \boldsymbol{\theta}'_{t+1})$$

According to Lemma 3, we have

$$D_{\Phi^*}(\boldsymbol{\theta}_t, \boldsymbol{\theta}'_{t+1}) \leq (1 + R^2/\mu)|\Delta \mathbf{x}_t|^2$$

We thus have

$$M\left(1 - \lambda(1 + R^2/\mu)|\Delta \mathbf{x}|^2\right) \leq \sum_{t=1}^{T} \ell_t(\mathbf{u}) + \frac{1}{\lambda} D_{\Phi}(\mathbf{u}, \mathbf{w}_1)$$

where $X_\infty = |\Delta \mathbf{x}| = \max_t |\Delta \mathbf{x}_t|$. Finally, by choosing $\mathbf{w}_1 = \arg\min_{w \in \Omega} \Phi(w, \mu) = 0$, we have

$$D_{\Phi}(\mathbf{u}, \mathbf{w}_1) \leq \Phi(\mathbf{u}) - \Phi(\mathbf{w}_1) \leq \frac{1}{2}\|u\|^2 + \mu \ln s$$

Finally, we have

$$M \leq \frac{1}{1 - \lambda(1 + R^2/\mu)X_\infty^2} \min_{u \in \Omega} \left(\frac{1}{\lambda} \frac{\|u\|^2}{2} + \mu \ln s + \sum_{t=1}^{T} \ell_t(\mathbf{u}) \right)$$

Substituting the λ defined in the Theorem 1, will give the above bound in terms of ϵ. □

For $\epsilon = 0.5$, the above theorem shows that the number of mistakes M made by $\mathbf{w}$ is no more than twice the mistakes made by the optimal weight vector $\mathbf{u}$, and a constant depending on $\mathbf{u}$, the smoothing parameter μ and the logarithm of the number of groups s.

5.4.3 Implementation details

For the potential function defined in Eq (5.24), steps 3 and 4 of Algorithm 4 are computationally complex. In particular, the computation of $\nabla\Phi(\mathbf{w}_t, \mu)$ involves solving a non-linear optimization problem defined in Eq (5.24). To avoid this, we use the original $L_{1,2}$ norm instead of the smooth norm. Further, the projection step 4 in Algorithm 4 is difficult. The projection in $L_{1,2}$ is performed approximately by projecting weights in each group $\|\mathbf{w}^g\|$ into a unit ball using an L_2 norm. This results in significant computational gains, with negligible difference in empirical evaluation. This choice results in a normalized weight

Algorithm 3 IncrementalFeatureSelection

Initialize $\mathbf{w} \leftarrow 0, t \leftarrow 0$
for each round $t = 1, 2, \cdots$ **do**
  Observe $(\mathbf{x}_t, \mathbf{x}'_t)$ and Predict $d_t \leftarrow |\mathbf{x}_t - \mathbf{x}'_t|_{\mathbf{w}}$
  if $y_t(b - d_t) \leq 0$ **then**
    $\mathbf{w}_t \leftarrow$ DualGradientDescentStep($\mathbf{x}_t, \mathbf{x}'_t, \mathbf{w}_{t-1}$)
  end if
  $t \leftarrow t + 1$
end for

Algorithm 4 DualGradientDescentStep($\mathbf{w}_t$)

1. $\theta_t \leftarrow \nabla\Phi(\mathbf{w}_t, \mu)$
2. $\theta'_{t+1} \leftarrow \theta_t - \lambda\nabla\ell_t(\mathbf{w}_t)$
3. $\mathbf{w}'_{t+1} \leftarrow \nabla\Phi^*(\theta'_{t+1}, \mu)$
4. $\mathbf{w}_{t+1} \leftarrow \pi_\Omega(\mathbf{w}'_{t+1}, \Phi) = \arg\min_{\mathbf{w}\in\Omega} D_\Phi(\mathbf{w}, \mathbf{w}'_{t+1})$

vector, fixing the value of $R = 1$. The solution is given in Eq (5.22), and the detailed derivations of the solution are presented in [187].

5.5 Experiments

Datasets: The proposed algorithm is evaluated using the PASCAL VOC challenge 2007 dataset [188]. This dataset has 9,963 images labeled using 20 classes of objects. The training and validation set contains 5,011 images. A detailed description of the data including the number of images per class is provided in [188]. The images in the dataset have multiple labels, and hence it is not directly suitable for evaluating image clustering. We ignore infrequently occurring objects in images and consider only the images containing one of the 6 most popular classes in the dataset, namely, `bicycle` (243), `bird` (330), `car` (713), `cat` (337), `chair` (445), and `person` (2008). The number of samples in each of these six classes is shown in brackets. For objects with multiple labels, one of the labels is chosen randomly.

Table 5.2: Performance of the proposed algorithm measured using pairwise-F1 measure. The first two columns show the target clusters, subsequent columns show the mean pairwise F_1 measure, expressed as percentage. Significant differences (paired t-test at 95% confidence) compared to the K-means algorithm are indicated by a $+$ or a $-$.

Task	Classes			Proposed		Online Baseline	Batch Baseline
#	c_1	c_2	K-means	$L_{1,2}$	L_1	POLA(L_2)	BestFirst
1	bird	cat	34.25	54.77+	51.18+	41.89−	56.40+
2	bird	bicycle	46.88	45.79−	49.30+	46.55	48.83+
3	bird	chair	57.51	57.97	60.22+	50.55−	61.10+
4	bird	car	55.74	63.24+	66.99+	58.32+	66.01+
5	bird	person	79.34	78.78	76.54−	75.34−	73.47−
6	cat	bicycle	42.55	53.81+	61.73+	53.00+	59.73+
7	cat	chair	41.85	46.16+	48.04+	47.18+	55.24+
8	cat	car	55.37	55.10	55.72	55.50	55.15
9	cat	person	78.98	78.45	73.48−	74.92−	66.47−
10	bicycle	chair	62.83	64.18+	64.58+	60.73−	56.85−
11	bicycle	car	66.25	67.78+	68.97+	65.69−	66.76
12	bicycle	person	84.09	83.76	78.44−	79.96−	84.10
13	chair	car	50.35	51.03	52.02+	53.51+	55.73+
14	chair	person	73.67	76.68+	68.84−	71.87−	64.91−
15	car	person	62.65	62.73	59.97−	63.74+	57.03−
	Summary			8+/1−	9+/5−	5+/8−	7+/5−

5.5.1 Feature extraction

SIFT (Version 4) key points [2] are extracted from each image. Each key-point is represented using a 128-dimensional feature vector. The key-points extracted from images in the training set are pooled together resulting in around 4.5 million key points. These key-points are clustered into 5,000 clusters using approximate hierarchical K-means algorithm from the FLANN library [189], with a branching factor of 20, resulting in a visual vocabulary of size 5000. Key point histograms are computed for each image in the training set. The group information of the visual vocabulary is obtained during the clustering phase by identifying all the visual words with common parents.

Experimental setup: Group-LASSO is a general norm which can be specialized to both L_2 or L_1 using appropriate group definition. If the number of groups is equal to the number of features, then Group-LASSO is equivalent to performing feature selection using an L1 norm. If all the features are put in a single group, the proposed algorithm is equivalent to the online distance metric learning algorithm POLA [123], which uses an L_2 norm as a regularizer. The performance of proposed algorithm with and without group structure ($L_{1,2}$ and L_1) is evaluated. The proposed algorithm is compared with the L_2 distance metric learning algorithm POLA. To compare the performance of the online algorithm with the batch mode algorithms, the classical Best First Search algorithm is used. However, note that batch mode algorithms assume that all examples are available a priori, and therefore they usually have better performance. The performance of clustering can be compared after applying an unsupervised dimensionality reduction algorithm such as principle component analysis. However, the performance of the K-means clustering will not be significantly different on the data with reduced dimensionality compared to the original dimensionality. This is because, the clustering obtained by K-means algorithm lies in the span of the top few ($\geq K$ in number) eigenvectors of the similarity matrix (linear kernel) [190]. Therefore, we do not show the results of clustering the data with dimensionality reduced using unsupervised linear dimensionality reduction algorithms.

For each pair of classes from the PASCAL VOC dataset, 300 randomly selected pairwise constraints are specified. The online learning algorithm is run for 10 epochs with the same 300 constraints shuffled each time. The number of images used for generating the pairwise constraints is specified in the dataset description in Section 5.5. The number of constraints considered in our algorithm is orders of magnitude smaller than those considered by other approaches [123, 185], which used around 10,000 constraints.

K-means algorithm is used to cluster the images with the selected features. Different sub-tasks from the PASCAL VOC dataset are chosen based on their class labels. The pairwise constraints provided to the proposed feature selection algorithm are derived from

the true labels of the examples. To alleviate the variability due to local minima in the K-means algorithm, it is run with ten different initializations. The cluster labels corresponding to the run with the lowest value of objective function are used for evaluation. Pairwise-F measure is used as the evaluation metric.

Parameters: The proposed algorithm has two parameters – the learning rate (or step size in the sequential gradient descent) λ and the norm-smoothness parameter μ. We set $\lambda = \frac{s}{2}$ where s is the number of groups in the visual words. The value of μ is set to 1. The value of b is chosen empirically to be 4. Ideally, if $\mathbf{w}$ is unconstrained, the value of b does not matter since it compensates for a scale factor in $\mathbf{w}$. The approximation used for Step 4 of Algorithm 2 (see Section 5.4.3) results in $R = 1$ constraining the domain of $\mathbf{w}$ to the unit $L_{1,2}$ ball. In this case, for $b > X_\infty$, there is no $\mathbf{w}$ that satisfies any of the cannot link constraints. Therefore a choice of b must satisfy $0 < b < X_\infty$. The domain size R is not a parameter, and need not be specified.

The values of the parameter are selected using cross validation on one of the clustering tasks (`bird` vs `cat`), which are then used for all the tasks. The range of values for these parameters to perform cross validation was motivated by Theorem 2. It may appear that selecting μ close to 0 would reduce the $\mu \ln s$ term in the mistake bound in Eq (5.25). However, setting μ to be small results in a small λ small, rendering the updates insignificant. Moreover, too small or too large a value for λ increases the the bound significantly resulting in poor learning, and hence is not recommended.

5.5.2 Results and discussion

Figure 5.3 illustrates the features selected by the proposed algorithm on six example images from the VOC 2007 dataset. The left image in each pair shows the original set of key points extracted by the SIFT algorithm with its default settings. The right image in the pair shows the key points corresponding to the visual words, selected by the proposed algorithm. Note

that in almost all the images, the key points in the background are drastically reduced as a result of keyword selection. However, in the examples containing `bird` in Figure 5.3, the key points corresponding to the tree branches are also retained by the feature selection algorithm. In a large fraction of `bird` images in the dataset, tree branches co-occur with a bird. Unless a sufficient number of cannot-link constraints are given between images containing `birds` and tree-branches, corresponding key points would not be eliminated. Such cases did not occur frequently in the dataset considered.

Table 5.2 shows the performance of the K-means clustering algorithm on 15 clustering tasks created from the VOC dataset. Table 5.3 shows the mean and standard deviation of the visual words selected by the proposed algorithm and the baselines. Group-LASSO based feature selection always resulted in the least number of features, followed by LASSO. The variance of Group-LASSO is higher since the features are discarded in groups of large size. In most cases, the performance drop is not significant (using paired t-test at 95% confidence). The cases where there is a significant difference in performance are marked by $+$ or $-$ accordingly.

In three out of the five clustering tasks involving the `person` class, the performance after feature selection is lower than that of K-means. This is attributed to the large difference in the number of samples in each class in the dataset. The degradation of the proposed method however, is less severe compared to the baselines. The class `person` is not only the most frequent but also frequently co-occurs with the other classes in the dataset. This imbalance in the number of samples results in a large bias towards positive or negative constraints, resulting in relatively poor feature selection. This can be alleviated by balancing the number of positive and negative constraints.

Overall, the proposed feature selection method, using both group-LASSO and LASSO, results in a vocabulary pruning of about 75-80%, on average for two-class problems. A larger number of classes may retain a larger fraction of key-points. Since the key-points are clustered using a larger number of images than those considered for each clustering task,

one might observe that there are naturally irrelevant key points for each task. However, that is not the case. In almost all the clustering tasks, most of all the visual keywords are observed.

5.6 Summary and conclusions

An online algorithm is presented for pruning the vocabulary in image analysis tasks that use the bag of words representation. Online algorithms are computationally efficient since they learn incrementally. Vocabulary pruning aids in representing images using a feature vector of low dimensionality, which naturally reduces the computation time required for subsequent clustering, classification or retrieval. The quality of pruned vocabulary is evaluated using a clustering task, and is compared to the performance of batch learning algorithms. A controlled study was performed to evaluate the merits of the proposed algorithm. Although the proposed algorithm is evaluated on visual vocabulary pruning task, it is applicable to other feature selection tasks as well. It is possible to derive the pairwise constraints automatically in some application domains from auxiliary information (e.g. text in web pages), where one may also be able to exploit the degree of relation between images.

Task	Proposed				Baseline	
#	Group LASSO		LASSO(L_1)		POLA(L_2)	
1	1148	509	1263	91	4929	187
2	986	449	1082	113	4982	30
3	1133	602	1204	310	4995	2
4	816	372	1170	82	4994	2
5	682	536	943	64	4834	473
6	1134	363	1156	124	4991	4
7	1283	537	1268	102	4996	2
8	1050	446	1213	124	4799	616
9	682	377	971	100	4943	163
10	1118	435	1092	45	4994	2
11	790	336	1025	92	4985	23
12	495	198	847	245	4921	215
13	999	377	1180	92	4978	34
14	729	391	940	55	4992	9
15	665	347	969	84	4982	37

Table 5.3: Mean and standard deviation of the number of visual words (from a total of 5,000) selected by the proposed LASSO and Group-LASSO method vs the L_2 DML algorithm. POLA is not a feature selection technique, and hence learns the weights for all the features. The batch mode forward search algorithm always selected 150 features, and hence is not reported in the table. The tasks are defined in Table 5.2

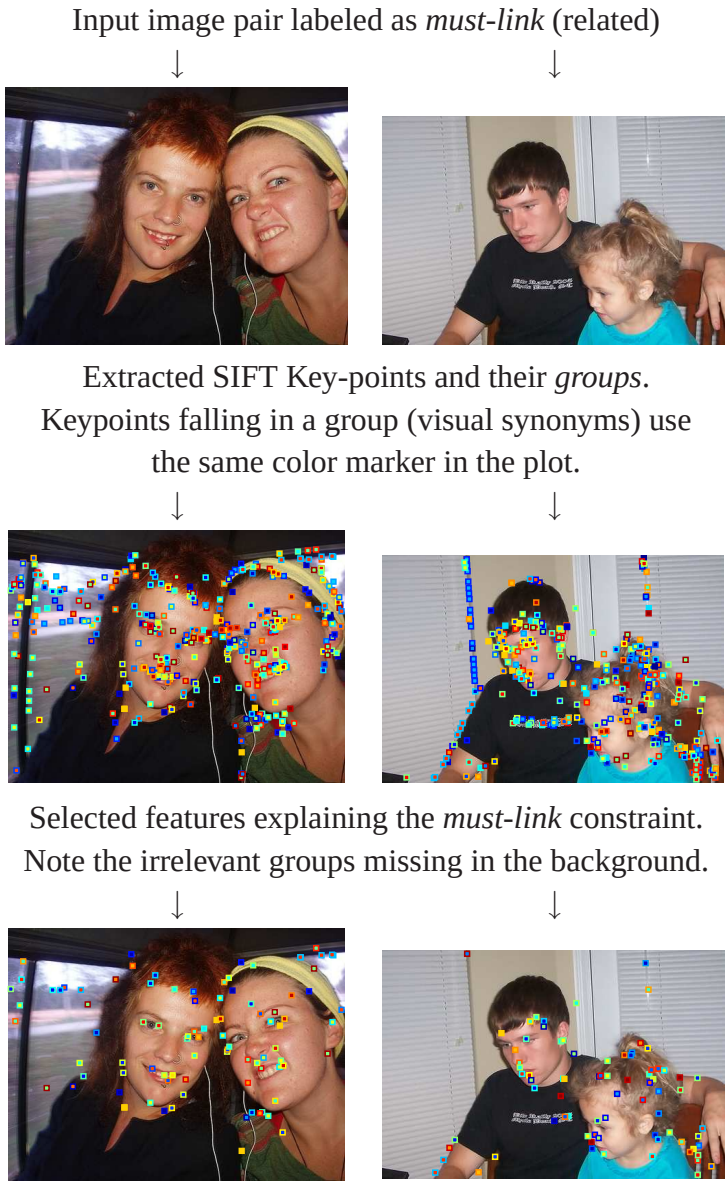


Figure 5.2: Illustration of SIFT key points, visual synonyms and feature selection at a group level. The first row shows a pair of images input for feature selection. Note that the key points occur in groups. Same colored marker is used for key points belonging to a group. Feature selection by the proposed feature selection algorithm acts at a group level by removing the entire group of unrelated features.

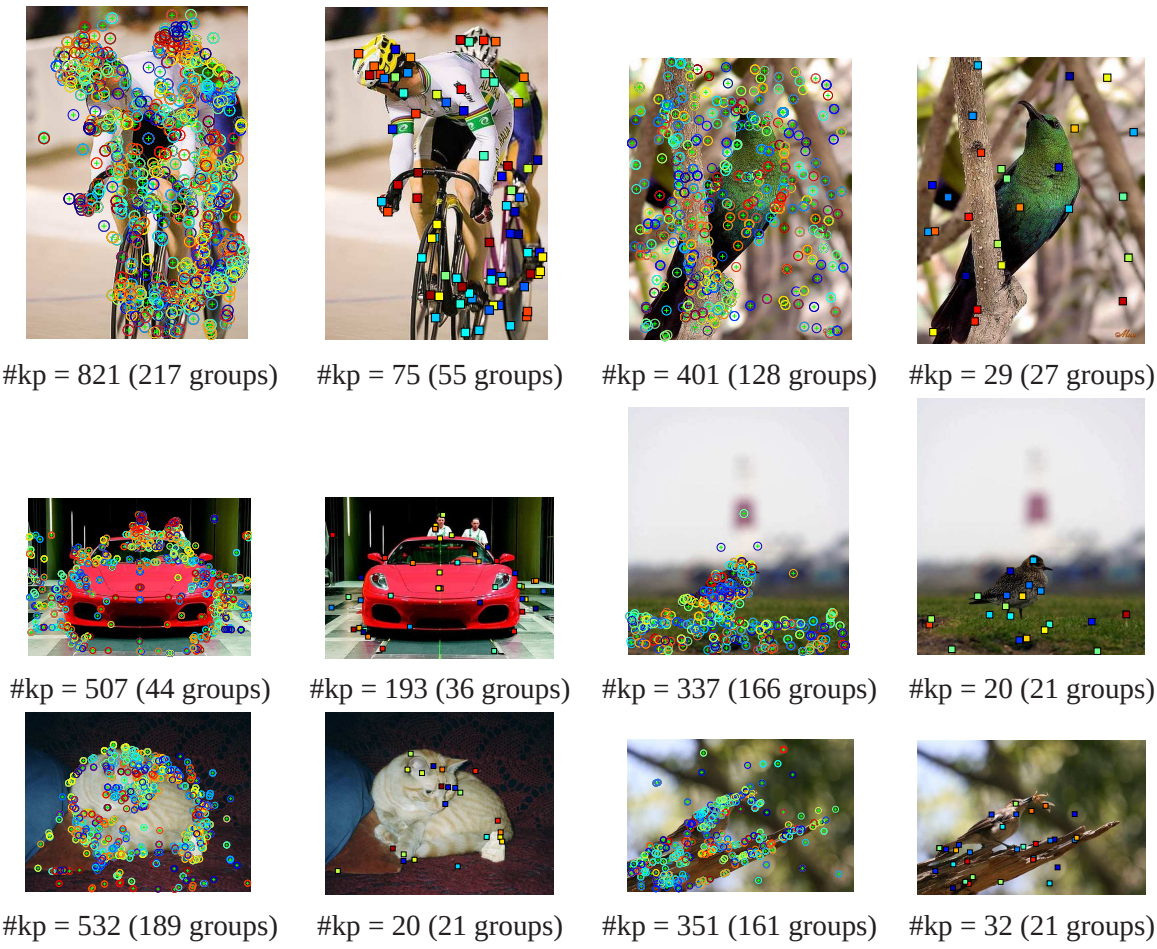


Figure 5.3: Feature selection using group-LASSO. In each pair of images, the left image shows the key points extracted from the original image and the right image shows the selected key points using the proposed feature selection algorithm. The number below each image indicates the number of key points (kp), and the number of groups (shown in brackets).

CHAPTER 6

Summary and Conclusions

In this thesis, three classical problems in pattern recognition, namely, classification, clustering and unsupervised feature selection have been extended to a semi-supervised learning setting.

6.1 Contributions

The semi-supervised boosting framework presented in Chapter 3 makes the following contributions:

- *Semi-supervised improvement framework:* A new semi-supervised learning setting, called *semi-supervised improvement* is proposed whose goal is to improve any existing supervised classifier when unlabeled data is available. Several well-known semi-supervised learning algorithms like self-training, ASSEMBLE etc. can be unified under this framework.

- *Semi-supervised Boosting Algorithm:* SemiBoost is a semi-supervised learning algorithm following the semi-supervised improvement framework. Drawing inspiration from successful ideas from graph based algorithms (Laplacian regularization), a new boosting algorithm was developed that utilizes the unlabeled data and a pairwise similarity matrix to improve the performance of any given supervised learning algorithms.

- *New regularizer for utilizing unlabeled data:* Graph Laplacian is a key concept in graph based semi-supervised learning. The quadratic cost function in the graph Laplacian is replaced with an exponential function. This helps in designing the SemiBoost algorithm along lines similar to the supervised boosting algorithms (e.g. AdaBoost).

The non-parametric mixtures algorithm presented in Chapter 4 makes the following contributions:

- *Mixture model based on kernel density estimates:* An extension of kernel density (Parzen window) estimation to the mixture models is proposed for data clustering. Each cluster is assumed to be generated by its own density function, and may have an arbitrary shape. Proposed model inherits the advantages of mixture models such as out-of-sample label assignment and probabilistic cluster assignments, without making any restrictive assumptions about the cluster shape.

- *Leave-one-out likelihood maximization:* A leave-one-out likelihood maximization approach is used to estimate the parameters of the algorithm, as opposed to the conventional maximum likelihood estimation.

- *Weighted kernel density estimate:* We introduced weighted kernel density estimates for modeling the density function of the clusters. It is possible to impose different constraints on the weights to achieve different clustering results. For instance, semi-supervision in the form of pairwise constraints can be incorporated by imposing the constraints on the weights.

- *Parameter selection from side-information:* Kernel bandwidth is one of the most important parameters of kernel based clustering algorithms. A novel approach for selecting critical parameters (bandwidth) of the kernel using pairwise constraints is presented. This approach is applicable to any kernel based algorithm (e.g., spec-

tral clustering, kernel k-means, etc) including the proposed non-parameteric mixture approach.

The feature selection algorithm proposed in Chapter 5 makes the following contributions:

- *Feature selection using pairwise constraints:* Given a set of unlabeled data points, and a set of pairwise constraints, an efficient online algorithm that selects a subset of features is proposed. The features are selected such that the distances between points in must-link constraints are less than the distances between points in cannot-link constraints.

- *Incorporating side information into the data:* While several approaches have been proposed to modify the clustering algorithms to include side-information, semi-supervised clustering is still largely an unsolved problem since the utility and the effect of the constraints are not well understood. We have proposed an alternate way to utilize the side-information, that is to incorporate it into the data itself through feature sub-space selection.

- *Online feature selection algorithm:* An online learning algorithm based on potential based gradient descent is presented to perform feature selection. Online algorithms are computationally efficient since they learn incrementally.

- *Group-LASSO:* A Group-LASSO regularizer is used to perform the feature selection. An application setting where the groups required in group LASSO are naturally available is explored.

- *Non-strongly convex regularizers in online learning:* Online learning algorithms and theory require the regularizer to be strongly convex. However, the proposed Group-LASSO regularizer is not strongly convex. A convex surrogate function for Group-

LASSO was introduced, and the related theoretical guarantees on the performance have been derived.

- *Application to Vocabulary Pruning:* The proposed online feature selection algorithm is applied to prune the vocabulary for clustering images represented using the bag of words representation. Given that the vocabularies constructed from general image collections are of large size, the proposed approach aids in representing images using smaller feature vectors. This reduces the computation time required for subsequent clustering, classification or retrieval. The quality of pruned vocabulary is evaluated using a clustering task that is comparable to that of batch learning algorithms.

- *Generality of the algorithm:* Although the proposed algorithm is evaluated on visual vocabulary pruning task, it is applicable to other feature selection tasks as well. Real world applications may derive the pairwise constraints automatically from auxiliary information (e.g. text in web pages), where one may also be able to exploit the degree of relationship between images.

6.2 Future Work

The studies conducted during the course of this dissertation raised questions that point to several important theoretical and empirical research directions.

- *Utility of unlabeled data for classification*

  Existing studies on the utility of unlabeled data for semi-supervised classification suggest that there are two sources of possible degradation in supervised algorithms:

  1. a mismatch between the *cluster structure* in the unlabeled data and that assumed by the model (similar to *cluster validation*)
  2. a misalignment between the true classes and the cluster structure.

The impact of unlabeled data on the classification algorithms can be analyzed by decomposing the generalization error of the classifier trained using partially labeled data, into the above two terms. Existing analyses do not explicitly study the decomposition of error into the aforementioned two terms, and is a potential area of study. Further, a priori evaluation of the two sources of error is difficult since it involves solving a harder problem of cluster validity. Currently, cross-validation against a labeled validation set is the only reliable way to assess the utility of side-information.

- *Utility of pairwise constraints:*

  Most empirical studies, including the experiments performed during the development of this thesis, indicate that the addition of pairwise constraints may occasionally degrade the performance of the corresponding supervised or unsupervised learner. An important question is *Can the effect of side-information be predicted a priori for a given learning task?*, or equivalently, *Can the factors affecting the utility of side-information to a clustering algorithm be explicitly enumerated, and its exact impact be studied?* This is related to the nature of the dataset and the quality of side-information.

- *Mixture of sparse kernel density estimates:*

  The mixture of kernel density estimates presented in Chapter 4 uses all the data points to define the density function of each cluster. However, most often a small subset of the examples should be sufficient to express the same density. This can be achieved by introducing an L_1 regularizer on the profile vectors instead of the current L_2 regularizer. This has several benefits such as increased efficiency in out-of-sample cluster assignment, and use of a simpler model as opposed to a complex model (Occam's razor).

- *Efficient algorithm for non-parameteric mixtures:*

The number of parameters in the non-parameteric mixture algorithm depends on the number of examples. While it is efficient compared to the Gaussian Mixture model and K-means for high-dimensional datasets, the number of parameters can be large for datasets with small dimension and large number of examples. This can be avoided by developing algorithms that use an L_1 regularizer to discard data points in each iteration that are not necessary to express the density function of a cluster.

- *Exploring relationship with spectral clustering:*

  The non-parameteric mixtures algorithm outperforms spectral clustering in most cases studied in our thesis, but under-performs in some cases. The proposed algorithm, however, has some advantages such as the ease of out-of-sample cluster assignment. It would be useful to study the theoretical relationship between spectral clustering and non-parameteric mixtures. The explanation may lead to a probabilistic interpretation for spectral clustering, and the intuition behind kernel density estimates may reveal why spectral clustering performs better or worse compared to the proposed algorithm.

- *Incorporating pairwise constraints into mixture of non-parameteric densities:*

  Side information such as pairwise constraints can be incorporated into the non-parameteric mixture algorithm.

- *Estimating the number of clusters:*

  Estimating the number of clusters present in the data is one of the most difficult problems in data clustering [3]. Any available side-information can only help in deciding the number of clusters. For instance, providing labels for a few examples may reveal the number of clusters. Most often, however, the labeled data could have originated only from a small number of the classes under consideration. The problem gets more complicated when a weaker form of side-information such as pairwise constraints is available, although it still is easier compared to the lack of

any side-information. This could be called *semi-supervised class discovery*, which is still an unexplored and difficult area. Throughout this thesis, we have assumed that the number of clusters is known.

- *Online algorithms using nested potentials:*

  The group-lasso based online learning algorithm presented in Chapter 5 can be generalized to nested potentials which provide a general framework for online learning with mixed norms.

- *Online clustering and applications:*

  Large scale data clustering is gaining interest due to the availability of massive datasets. Many applications such as visual vocabulary construction, use clustering algorithms to choose visual words. However, recent (unpublished) research shows that choosing these cluster centers randomly is equally helpful. In this light, a middle ground, where an efficient clustering algorithm is used to cluster the data to come up with visual vocabulary could provide the advantages of both – cluster quality and computational efficiency. Online algorithms can prove to be useful in large scale clustering applications where speed is more important compared to the quality of clusters.

- *Semi-supervised online learning:*

  Online learning algorithms are usually supervised. Exploiting the use of unlabeled data for online classification algorithms could be beneficial. No such algorithm exists at this point of time.

6.3 Conclusions

This thesis presents semi-supervised algorithms for three classical problems in pattern recognition and machine learning, namely, supervised learning, unsupervised learning,

and unsupervised feature selection. The thesis makes significant contributions to semi-supervised learning by developing algorithms that advance the state-of-the-art in semi-supervised classification and clustering. Several applications of semi-supervised learning algorithms such as text classification and image clustering were shown, and the potential of utilizing side-information to achieve better learning and prediction performance is demonstrated. Several directions for future work in semi-supervised learning are identified.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] B. Nadler and M. Galun, "Fundamental limitations of spectral clustering," in *Advances in Neural Information Processing Systems*, vol. 19, p. 1017, 2007.

[2] D. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the International Conference on Computer Vision*, vol. 2, pp. 1150–1157, 1999.

[3] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[4] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Prentice Hall, 1988.

[5] J. Tukey, *Exploratory data analysis*. Addison-Wesley, 1977.

[6] Amazon Mechanical Turk, "http://www.mturk.com."

[7] O. Chapelle, B. Schölkopf, and A. Zien, eds., *Semi-Supervised Learning*. MIT Press, 2006.

[8] V. Vapnik, *The Nature of Statistical Learning Theory*. Wiley-Interscience, 1998.

[9] B. Liu, *Web data mining*. Springer, 2007.

[10] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Scholkopf, "Ranking on data manifolds," in *Advances in Neural Information Processing Systems*, pp. 177–186, 2003.

[11] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002.

[12] L. Yang, R. Jin, and R. Sukthankar, "Semi-supervised learning with weakly-related unlabeled data: Towards better text categorization," in *Advances in Neural Information Processing Systems*, 2008.

[13] X. Zhu, "Semi-supervised learning literature survey," Tech. Rep. TR 1530, University of Wisconsin, Madison, 2006.

[14] Y. Liu, R. Jin, and L. Yang, "Semi-supervised multi-label learning by constrained non-negative matrix factorization," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 21, p. 421, 2006.

[15] S. Basu, A. Banerjee, and R. Mooney, "Semi-supervised clustering by seeding," in *Proceedings of the International Conference on Machine Learning*, pp. 27–34, 2002.

[16] K. Wagstaff and C. Cardie, "Clustering with instance-level constraints," in *Proceedings of the International Conference on Machine Learning*, pp. 1103–1110, 2000.

[17] M. H. Law, A. Topchy, and A. K. Jain, "Model-based clustering with probabilistic constraints," in *Proceedings of the SIAM Data Mining*, pp. 641–645, 2005.

[18] A. Banerjee and J. Ghosh, "Scalable clustering algorithms with balancing constraints," *Data Mining and Knowledge Discovery*, vol. 13, no. 3, pp. 365–395, 2006.

[19] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Scholkopf, "Ranking on data manifolds," in *Advances in Neural Information Processing Systems*, p. 169, 2004.

[20] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Wiley-Interscience Publication, 2000.

[21] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[22] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. Academic Pr, 1990.

[23] T. Mitchell, *Machine Learning*. McGraw Hill, 1997.

[24] C. Bishop, *Neural Networks for Pattern Recognition*. Oxford Univ Pr, 2005.

[25] V. Vapnik, *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, 1982.

[26] P. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Pearson Addison Wesley Boston, 2005.

[27] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[28] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proceedings of the International Conference on Machine Learning*, pp. 148–156, 1996.

[29] Y. Freund, "Boosting a weak learning algorithm by majority," *Information and Computation*, vol. 121(2), pp. 256–285, 1995.

[30] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

[31] P. Berkhin, *A Survey of Clustering Data Mining Techniques*, ch. 2, pp. 25–71. Springer-Verlag, 2006.

[32] G. Ball and D. Hall, "ISODATA, a novel method of data anlysis and pattern classification," Tech. Rep. NTIS AD 699616, Stanford Research Institute, Stanford, CA, 1965.

[33] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, pp. 129–137, 1982.

[34] G. L. McLachlan and K. E. Basford, *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, 1987.

[35] G. L. McLachlan and D. Peel, *Finite Mixture Models*. Wiley, 2000.

[36] M. Figueiredo and A. Jain, "Unsupervised learning of finite mixture models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 381–396, 2002.

[37] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[38] Y. Teh, M. Jordan, M. Beal, and D. Blei, "Hierarchical dirichlet processes," *Journal of the American Statistical Association*, vol. 101, no. 476, pp. 1566–1581, 2006.

[39] D. M. Blei and M. I. Jordan, "Hierarchical topic models and the nested chinese restaurant process," in *Advances in Neural Information Processing Systems*, 2004.

[40] C. E. Rasmussen, "The infinite gaussian mixture model," in *Advances in Neural Information Processing Systems*, pp. 554–560, 2000.

[41] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 888–905, 2000.

[42] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems*, pp. 849–856, 2001.

[43] Z. Li, J. Liu, S. Chen, and X. Tang, "Noise robust spectral clustering," in *Proceedings of the International Conference on Computer Vision*, pp. 1–8, 2007.

[44] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means: spectral clustering and normalized cuts," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 551–556, 2004.

[45] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 603–619, May 2002.

[46] M. Andreetto, L. Zelnik Manor, and P. Perona, "Non-parametric probabilistic image segmentation," in *Proceedings of the International Conference on Computer Vision*, pp. 1–8, 2007.

[47] K. Heller and Z. Ghahramani, "Bayesian hierarchical clustering," in *Proceedings of the International Conference on Machine Learning*, vol. 22, p. 297, 2005.

[48] R. A. Jarvis and E. A. Patrick, "Clustering using a similarity measure based on shared near neighbors," *IEEE Transactions on Computers*, vol. 22, p. 9, 1973.

[49] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 226–231, 1996.

[50] A. Hinneburg, E. Hinneburg, and D. A. Keim, "An efficient approach to clustering in large multimedia databases with noise," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 58–65, 1998.

[51] E. L. Johnson, A. Mehrotra, and G. L. Nemhauser, "Min-cut clustering," *Mathematical Programming*, vol. 62, pp. 133–151, 1993.

[52] L. Hagen and A. Kahng, "New spectral methods for ratio cut partitioning and clustering," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 11, no. 9, pp. 1074–1085, 1992.

[53] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society: Series B*, vol. 39, pp. 1–38, 1977.

[54] T. Hofmann, "Probabilistic latent semantic analysis," in *Proceedings of Uncertainty in Artificial Intelligence*, pp. 289–296, 1999.

[55] D. Pelleg and A. Moore, "X-means: Extending k-means with efficient estimation of the number of clusters," in *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 727–734, 2000.

[56] B. Sch olkopf, A. Smola, and K.-R. M uller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, pp. 1299–1319, 1998.

[57] K. A. Heller and Z. Ghahramani, "Bayesian hierarchical clustering," in *Proceedings of the Proceedings of the International Conference on Machine Learning*, vol. 119, 2005.

[58] D. Fisher, "Knowledge acquisition via incremental conceptual clustering," *Machine Learning*, vol. 2, no. 2, pp. 139–172, 1987.

[59] S. J. Roberts, R. Everson, and I. Rezek, "Minimum entropy data partitioning," in *Proceedings of International Conference on Artificial Neural Networks*, pp. 844–849, 1999.

[60] S. J. Roberts, C. Holmes, and D. Denison, "Minimum-entropy data clustering using reversible jump markov chain monte carlo," in *Proceedings of the International Conference on Artificial Neural Networks*, pp. 103–110, 2001.

[61] N. Tishby and N. Slonim, "Data clustering by markovian relaxation and the information bottleneck method," in *Advances in Neural Information Processing Systems*, pp. 640–646, 2000.

[62] J. M. Buhmann and T. Hofmann, "A maximum entropy approach to pairwise data clustering," in *Proceedings of the International Conference on Pattern Recognition*, pp. 207–212, 1994.

[63] H. J. Scudder, "Probability of error of some adaptive pattern-recognition machines," *IEEE Transactions on Information Theory*, vol. 11, pp. 363–371, 1965.

[64] H. Robbins and S. Monro, "A stochastic approximation method," *Annals of Mathematical Statistics*, vol. 22, pp. 400–407, 1951.

[65] V. Vapnik and A. Sterin, "On structural risk minimization or overall risk in a problem of pattern recognition," *Automation and Remote Control*, vol. 10, no. 3, pp. 1495–1503, 1977.

[66] Y. Bengio, O. B. Alleau, and N. Le Roux, "Label propagation and quadratic criterion," in *Semi-Supervised Learning* (O. Chapelle, B. Schölkopf, and A. Zien, eds.), pp. 193–216, MIT Press, 2006.

[67] M. Szummer and T. Jaakkola, "Partially labeled classification with Markov random walks," in *Advances in Neural Information Processing Systems*, pp. 945–952, 2001.

[68] A. Blum and S. Chawla, "Learning from labeled and unlabeled data using graph mincuts," in *Proceedings of the International Conference on Machine Learning*, pp. 19–26, 2001.

[69] T. Joachims, "Transductive learning via spectral graph partitioning," in *Proceedings of the International Conference on Machine Learning*, pp. 290–297, 2003.

[70] O. Chapelle and A. Zien, "Semi-supervised classification by low density separation," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 57–64, 2005.

[71] O. Chapelle, B. Scholkopf, and A. Zien, eds., *Semi-Supervised Learning*. MIT Press, 2006.

[72] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proceedings of the International Conference on Machine Learning*, pp. 200–209, 1999.

[73] G. Fung and O. Mangasarian, "Semi-supervised support vector machines for unlabeled data classification," *Optimization Methods and Software*, vol. 15, pp. 29–44, 2001.

[74] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the International Conference on Machine Learning*, pp. 912–919, 2003.

[75] O. Chapelle and A. Zien, "Semi-supervised classification by low density separation," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 57–64, 2005.

[76] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: a geometric framework for learning from examples," Technical Report TR-2004-06, University of Chicago, Dept of Computer Science, 2004.

[77] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the Workshop on Computational Learning Theory*, pp. 92–100, 1998.

[78] C. Rosenberg, M. Hebert, and H. Schneiderman, "Semi-supervised self-training of object detection models," in *Proceedings of the Workshopn on Applications of Computer Vision*, vol. 1, pp. 29–36, January 2005.

[79] F. d'Alche Buc, Y. Grandvalet, and C. Ambroise, "Semi-supervised marginboost," in *Advances in Neural Information Processing Systems*, pp. 553–560, 2002.

[80] K. P. Bennett, A. Demiriz, and R. Maclin, "Exploiting unlabeled data in ensemble methods," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 289–296, 2002.

[81] D. Miller and H. Uyar, "A mixture of experts classifier with learning based on both labeled and unlabeled data," in *Advances in Neural Information Processing Systems*, pp. 571–577, 1996.

[82] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using EM," *Machine Learning*, vol. 39, pp. 103–134, 2000.

[83] N. D. Lawrence and M. I. Jordan, "Semi-supervised learning via gaussian processes," in *Advances in Neural Information Processing Systems*, pp. 753–760, 2005.

[84] D. Yarowsky, "Unsupervised word sense disambiguation rivalling supervised methods," in *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, p. 189196, 1995.

[85] C. Rosenberg, M. Hebert, and H. Schneiderman, "Semi-supervised self-training of object detection models," in *Proc.eedings of the Workshop on Applications of Computer Vision*, vol. 1, pp. 29–36, January 2005.

[86] K. Nigam, A. K. McCallum, S. Thrun, and T. M. Mitchell, "Text classification from labeled and unlabeled documents using EM," *Machine Learning*, vol. 39, no. 2/3, pp. 103–134, 2000.

[87] K. Bennet, A. Demiriz, and R. Maclin, "Exploiting unlabeled data in ensemble methods," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 289–296, 2002.

[88] P. K. Mallapragada, R. Jin, A. K. Jain, and Y. Liu, "SemiBoost: boosting for semi-supervised learning.," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 11, p. 2000, 2009.

[89] K. Bennett and A. Demiriz, "Semi-supervised support vector machines," in *Advances in Neural Information Processing Systems*, pp. 368–374, 1998.

[90] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proceedings of the International Conference on Machine Learning* (I. Bratko and S. Dzeroski, eds.), pp. 200–209, 1999.

[91] O. Tuzel, F. Porikli, and P. Meer, "Kernel Methods for Weakly Supervised Mean Shift Clustering," in *International Conference on Computer Vision*, vol. 4, 2009.

[92] S. X. Yu and J. Shi, "Grouping with directed relationships," *Lecture Notes in Computer Science*, vol. 2134, pp. 283–293, 2001.

[93] B. Kulis, S. Basu, I. Dhillon, and R. Mooney, "Semi-supervised graph clustering: a kernel approach," in *Proceedings of the International Conference on Machine Learning*, pp. 457–464, 2005.

[94] B. Kulis, S. Basu, I. Dhillon, and R. Mooney, "Semi-supervised graph clustering: a kernel approach," *Machine Learning*, vol. 74, no. 1, pp. 1–22, 2009.

[95] S. Kamvar, D. Klein, and C. Manning, "Spectral learning," in *International Joint Conference On Artificial Intelligence*, vol. 18, pp. 561–566, Citeseer, 2003.

[96] Z. Lu and T. Leen, "Semi-supervised learning with penalized probabilistic clustering," in *Advances in Neural Information Processing Systems*, p. 849856, 2005.

[97] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl, "Constrained k-means clustering with background knowledge," in *Proceedings of the International Conference on Machine Learning*, pp. 577–584, 2001.

[98] S. Basu, I. Davidson, and K. Wagstaff, *Constrained Clustering: Advances in Algorithms, Theory, and Applications*. Chapman & Hall/CRC, 2008.

[99] D. Klien, S. D. Kamvar, and C. D. Manning, "From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering," in *Proceedings of the International Conference on Machine Learning*, 2002.

[100] J. G. Auguston and J. Minker, "An analysis of some graph theoretical clustering techniques," *Journal of the ACM*, vol. 17, no. 4, pp. 571–588, 1970.

[101] C. Zahn, "Graph theoretical methods for detecting and describing gestalt clusters," *IEEE Transactions on Computers*, vol. 20, pp. 68–86, 1971.

[102] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[103] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems*, vol. 2, pp. 849–856, 2002.

[104] Z. Wu and R. Leahy, "An optimal graph theoretic approach to data clustering: Theorey and its application to image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 1101–1113, 1993.

[105] S. Dongen, "Performance criteria for graph clustering and markov cluster experiments," tech. rep., CWI (Centre for Mathematics and Computer Science), Nederlands, 2000.

[106] U. Brandes, M. Gaertler, and D. Wagner, "Experiments on graph clustering algorithms," in *Proceedings of the 11th European Symposiom of Algorithms*, pp. 568–579, 2003.

[107] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *Journal of Machine Learning Research*, vol. 7, pp. 2399–2434, 2006.

[108] N. Shental, A. Bar-Hillel, T. Hertz, and D. Weinshall, "Computing gaussian mixture models with em using equivalence constraints," in *Advances in Neural Information Processing Systems*, 2004.

[109] S. Basu, M. Bilenko, and R. J. Mooney, "A probabilistic framework for semi-supervised clustering," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 59–68, 2004.

[110] T. Lange, M. H. Law, A. K. Jain, and J. Buhmann, "Learning with constrained and unlabelled data," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 730–737, 2005.

[111] Q. Zhao and D. Miller, "Mixture modeling with pairwise, instance-level class constraints," *Neural computation*, vol. 17, no. 11, pp. 2482–2507, 2005.

[112] M. H. C. Law, *Clustering, Dimensionality Reduction and Side Information*. PhD thesis, Michigan State University, 2006.

[113] S. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[114] V. De Silva and J. Tenenbaum, "Global versus local methods in nonlinear dimensionality reduction," in *Advances in Neural Information Processing Systems*, pp. 721–728, 2003.

[115] K. Weinberger, J. Blitzer, and L. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Advances in Neural Information Processing Systems*, vol. 18, p. 1473, 2006.

[116] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighbourhood components analysis," in *Advances in Neural Information Processing Systems*, vol. 17, pp. 513–520, 2005.

[117] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell, "Distance metric learning with application to clustering with side-information," in *Advances in Neural Information Processing Systems*, 2003.

[118] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall, "Learning a mahalanobis metric from equivalence constraints," *Journal of Machine Learning Research*, vol. 6, pp. 937–965, 2005.

[119] M. Bilenko, S. Basu, and R. J. Mooney, "Integrating constraints and metric learning in semi-supervised clustering," in *Proceedings of the International Conference on Machine Learning*, vol. 69, 2004.

[120] L. Yang, R. Jin, R. Sukthankar, and Y. Liu, "An efficient algorithm for local distance metric learning," in *Proceedings of the National Conference on Artificial Intelligence*, p. 543, 2006.

[121] S. Hoi, R. Jin, and M. Lyu, "Learning nonparametric kernel matrices from pairwise constraints," in *Proceedings of the International Conference on Machine Learning*, pp. 361–368, 2007.

[122] J. Lee, R. Jin, and A. Jain, "Rank-based distance metric learning: An application to image retrieval," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2008.

[123] S. Shalev-Shwartz, Y. Singer, and A. Ng, "Online and batch learning of pseudo-metrics," in *Proceedings of the International Conference on Machine Learning*, 2004.

[124] J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon, "Information-theoretic metric learning," in *Proceedings of the International Conference on Machine Learning*, pp. 209–216, 2007.

[125] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall, "Learning distance functions using equivalence relations," in *Proceedings of the International Conference on Machine Learning*, 2003.

[126] L. Yang, R. Jin, R. Sukthankar, B. Zheng, L. Mummert, M. Satyanarayanan, M. Chen, and D. Jukic, "Learning distance metrics for interactive search-assisted diagnosis of mammograms," in *Proceedings of the SPIE Medical Imaging*, 2007.

[127] S. Basu, A. Banerjee, and R. J. Mooney, "Active semi-supervision for pairwise constrained clustering.," in *Proceedings of the International Conference on Data Mining*, 2004.

[128] A. K. Jain, P. Mallapragada, and M. Law, "Bayesian feedback in data clustering," in *Proceedings of the International Conference on Pattern Recognition*, vol. 3, p. 378, 2006.

[129] P. Mallapragada, R. Jin, and A. Jain, "Active Query Selection for Semi-supervised Clustering," in *Proceedings of the 19th International Conference on Pattern Recognition*, pp. 1–4, 2008.

[130] V. Castelli and T. M. Cover, "On the exponential value of labeled samples," *Pattern Recognition Lettters*, vol. 16, no. 1, pp. 105–111, 1995.

[131] T. Zhang and F. Oles, "A probability analysis on the value of unlabeled data for classification problems," in *Proceedings of the International Conference on Machine Learning*, pp. 1191–1198, 2000.

[132] M. Seeger, "Learning with labeledand unlabeled data," tech. rep., University of Edinburgh, 2001.

[133] F. Cozman and I. Cohen, "Unlabeled data can degrade classification performance of generative classifiers," in *Proceedings of the Fifteenth International Florida Artificial Intelligence Society Conference*, pp. 327–331, 2002.

[134] S. Ben-David, T. Lu, and D. Pal, "Does unlabeled data provably help? worst-case analysis of the sample complexity of semi-supervised learning," in *Proceedings of the Annual Conference on Computational Learning Theory*, 2008.

[135] M.-F. Balcan and A. Blum, "A PAC-style model for learning from labeled and unlabeled data," in *Proceedings of the Annual Conference on Computational Learning Theory*, pp. 111 – 126, 2005.

[136] I. Davidson, K. Wagstaff, and S. Basu, "Measuring constraint-set utility for partitional clustering algorithms," *Lecture Notes in Computer Science*, vol. 4213, p. 115, 2006.

[137] I. Cohen, F. G. Cozman, N. Sebe, M. C. Cirelo, and T. S. Huang, "Semisupervised learning of classifiers: Theory, algorithms, and their application to human-computer interaction," *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, vol. 26, no. 12, pp. 1553–1567, 2004.

[138] Z.-H. Zhou, K.-J. Chen, , and Y. Jiang, "Exploiting unlabeled data in content-based image retrieval," in *Proceedings of the International Conference on Machine Learning*, pp. 525–536, 2004.

[139] M.-F. Balcan, A. Blum, P. P. Choi, J. Lafferty, B. Pantano, M. R. Rwebangira, and X. Zhu, "Person identification in webcam images: An application of semi-supervised learning," in *Proceedings of the Workshop on Learning with Partially Classified Training Data (ICML)*, 2005.

[140] Z.-H. Zhou, K.-J. Chen, and H.-B. Dai, "Enhancing relevance feedback in image retrieval using unlabeled data," *ACM Transactions on Information Systems*, vol. 24, pp. 219 – 244, 2006.

[141] S. Abney, *Semi-supervised Learning for Computational Linguistics*. Chapman and Hall-CRC, 2007.

[142] X. Zhu, T. Rogers, R. Qian, , and C. Kalish, "Humans perform semi-supervised classification too," in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 864–870, 2007.

[143] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, pp. 119 – 139, August 1997.

[144] D. Zhou, J. Huang, and B. Scholkopf, "Learning from labeled and unlabeled data on a directed graph," in *Proceedings of the International Conference on Machine Learning*, pp. 1036–1043, 2005.

[145] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *The Annals of Statistics*, vol. 28, pp. 337–374, April 2000.

[146] L. Mason, J. Baxter, P. Bartlett, and M. Frean, "Boosting algorithms as gradient descent in function space," in *Advances in Neural Information Processing Systems*, pp. 512–518, 1999.

[147] P. K. Mallapragada, R. Jin, A. K. Jain, and Y. Liu, "Semiboost: Boosting for semi-supervised learning," Tech. Rep. MSU-CSE-07-197, Michigan State University, 2007.

[148] T. Minka, "Expectation-maximization as lower bound maximization," *Tutorial available at http://www-white.media.mit.edu/ tpminka/papers/em.html*, 1998.

[149] A. K. Jain and X. Lu, "Ethnicity identification from face images," in *Proceedings of the SPIE Defense and Security Symposium*, vol. 5404, pp. 114–123, 2004.

[150] A. K. Jain and F. Farrokhina, "Unsupervised texture segmentation using gabor filters," *Pattern Recognition*, vol. 24, pp. 1167–1186, 1991.

[151] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2nd ed., 2005.

[152] L. Reyzin and R. E. Schapire, "How boosting the margin can also boost classifier complexity," in *Proceedings of the International Conference on Machine Learning*, pp. 753–760, 2006.

[153] J. Platt, N. Cristianini, and J. Shawe-Taylor, "Large margin dags for multiclass classification," in *Advances in Neural Information Processing Systems*, 2000.

[154] H. Valizadegan, R. Jin, and A. Jain, "Semi-supervised boosting for multi-class classification," *Machine Learning and Knowledge Discovery in Databases*, pp. 522–537, 2008.

[155] S. X. Yu and J. Shi, "Multiclass spectral clustering," in *Proceedings of the International Conference on Computer Vision*, pp. 313–319, 2003.

[156] K. Barnard, P. Duygulu, and D. Forsyth, "Clustering art," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 434–440, 2001.

[157] P. Duygulu, K. Barnard, N. De Freitas, and D. Forsyth, "Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary," in *Proceedings of the European Conference on Computer Vision*, pp. IV:97–112, 2002.

[158] J. Park, H. Zha, and R. Kasturi, "Spectral clustering for robust motion segmentation," in *Proceedings of the European Conference on Computer Vision*, pp. 390–401, 2004.

[159] J. Sivic and A. Zisserman, "Video google: a text retrieval approach to object matching in videos," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 17, pp. 1470–1477, 2003.

[160] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 5, 2006.

[161] J. Sander, M. Ester, H. Kriegel, and X. Xu, "Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 169–194, 1998.

[162] J. Shawe-Taylor and A. N. Dolia, "A framework for probability density estimation," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 468–475, 2007.

[163] M. P. Wand and M. C. Jones, *Kernel Smoothing*. Chapman & Hall/CRC, December 1994.

[164] I. Csiszár and G. Tusnády, "Information geometry and alternating minimization procedures," *Statistics and Decisions*, vol. 1, no. (Suppl.), pp. 205–237, 1984.

[165] T. S. Jaakkola, "Tutorial on variational approximation methods," in *Advanced Mean Field Methods: Theory and Practice*, pp. 129–159, 2000.

[166] N. Slonim and N. Tishby, "Agglomerative information bottleneck," in *Advances in Neural Information Processing Systems*, 2000.

[167] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz, "UCI repository of machine learning databases," 1998. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[168] C. J. V. Rijsbergen, *Information Retrieval*. Butterworth, 1979.

[169] A. Banerjee and J. Langford, "An objective evaluation criterion for clustering," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 515–520, 2004.

[170] H. Kim, P. Howland, and H. Park, "Dimension reduction in text classification with support vector machines," *Journal of Machine Learning Research*, vol. 6, pp. 37–53, 2005.

[171] R. Baeza-Yates, B. Ribeiro-Neto, *et al.*, *Modern Information Retrieval*. Addison-Wesley, 1999.

[172] J. Goldberger and S. Roweis, "Hierarchical clustering of a mixture model," in *Advances in Neural Information Processing Systems*, pp. 505–512, 2004.

[173] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. Cambridge Univ. Press, 2006.

[174] P. Duygulu, K. Barnard, J. de Freitas, and D. Forsyth, "Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary," *Lecture Notes in Computer Science*, vol. 2353, pp. 97–112, 2002.

[175] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 3613, pp. 1575–1589, 2007.

[176] J. Yang, Y. Jiang, A. Hauptmann, and C. Ngo, "Evaluating bag-of-visual-words representations in scene classification," in *ACM Workshop on MIR*, p. 206, 2007.

[177] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[178] M. Dash and H. Liu, "Feature selection for classification," *Intelligent Data Analysis*, vol. 1, no. 3, pp. 131–156, 1997.

[179] H. Liu and H. Motoda, *Computational methods of feature selection*. Chapman & Hall/CRC, 2008.

[180] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B*, vol. 58, no. 1, pp. 267–288, 1996.

[181] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of Royal Statistical Society: Series B*, vol. 68, no. 1, p. 49, 2006.

[182] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani, "1-norm support vector machines," in *Advances in Neural Information Processing Systems*, p. 49, 2004.

[183] H. Zou and M. Yuan, "The Fo-norm Support Vector Machine," *Statistica Sinica*, vol. 18, pp. 379–398, 2006.

[184] P. Zhao and B. Yu, "Boosted lasso," in *Proceedings of the SIAM Workshop on Feature Selection for Data Mining*, p. 35, 2005.

[185] P. Jain, B. Kulis, I. Dhillon, and K. Grauman, "Online metric learning and fast similarity search," in *Advances in Neural Information Processing Systems*, 2008.

[186] L. Meier, S. van de Geer, and P. Buhlmann, "The group lasso for logistic regression," *Journal of the Royal Statistical Society: Series B*, vol. 70, no. 1, p. 53, 2008.

[187] P. K. Mallapragada, R. Jin, and A. K. Jain, "Online feature selection using group-LASSO," Tech. Rep. MSU-CSE-10-8, Michigan State University, 2010.

[188] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL VOC Challenge 2007 (VOC2007) Results."

[189] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *VISAPP*, 2009.

[190] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means: spectral clustering and normalized cuts," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 551–556, 2004.