The Pennsylvania State University
The Graduate School

IMPROVED GENERATIVE MODELING APPROACHES FOR SEMI-SUPERVISED AND DOMAIN ADAPTIVE CLASSIFIER LEARNING FROM LABELS AND CONSTRAINTS

A Dissertation in
Electrical Engineering
by
Jayaram Raghuram

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

December 2014

The dissertation of Jayaram Raghuram was approved* by the following:

David J. Miller
Professor of Electrical Engineering
Dissertation Advisor, Co-Chair of Committee

George Kesidis
Professor of Electrical Engineering and Computer Science and Engineering
Co-Chair of Committee

Kenneth Jenkins
Professor of Electrical Engineering
Committee Member

Yu Zhang
Associate Professor of Statistics
Committee Member

Kultegin Aydin
Professor of Electrical Engineering
Head of the Department of Electrical Engineering

*Signatures on file in the Graduate School.

Abstract

This dissertation makes contributions towards the following three closely related, important problems in machine learning: *1. Semi-supervised classification, 2. Semi-supervised learning with instance-level constraints, and 3. Semi-supervised domain adaptation of classifiers*. Semi-supervised learning, which has been an active area of research for more than a decade now, attempts to mitigate the problem of label scarcity or limited supervision in practical machine learning and statistical modeling tasks, such as classification and clustering, by exploiting the relative abundance and easy availability of unlabeled data to improve upon model solutions which would otherwise be based only on the limited data having supervision. Domain adaptation of classifiers is a relatively recent area of research, where the goal is to leverage the availability of a large labeled database and/or an existing classifier model to adaptively learn a better classifier model for a target domain where the underlying distribution of data is different.

In the case of *semi-supervised classification*, where the partial supervision is in the form of class labels, we developed a method for improving upon the class posterior probability model given the feature vector (set of features) for a generative mixture model based classifier. In particular, based on novel stochastic data generation methods, we allow the class posterior probability models within the mixture components to be non-trivial functions of the feature vector, addressing a significant limitation of existing methods which only allow a single class per component or a single, feature vector independent probability mass function per component. This allows for more *fine-grained* component conditional class modeling, leading to potentially better classification performance as we demonstrate on synthetic and real data sets in Chapter 2.

In the case of *semi-supervised constraint-based learning*, where supervision is in the form of constraints on pairs of data samples indicating whether the sample pairs are from the same underlying class (a must-link constraint) or from different underlying classes (a cannot-link constraint), we developed a method for predicting the grouping of data samples into (unknown) classes, which *not only* satisfies a majority of the constraints, but also ensures that the spatial implications of the constraints are consistently enforced in the solution, leading to better grouping solutions and generalization on unseen data. Most of the prior works addressing this problem do not provide a serious treatment of this requirement to enforce the spatial implications of the constraints in the solution. Also, they make a typically unrealistic assumption that the number

of underlying classes is known and provided as input, while our method does not require this knowledge, and in fact provides an estimate of the number of classes as a by-product of the model learning. In Chapter 3, we demonstrate that our method can lead to significant performance improvements on a variety of synthetic and real data sets.

In the case of *semi-supervised domain adaptation of classifiers*, we have a scenario similar to semi-supervised learning in one of the data domains (called target domain), with scarcity of labeled data and relative abundance of unlabeled data. However, in addition, there is easy availability of labeled data from a different domain (called source domain) for which it is possible that the underlying probability distribution of the data (features and class labels) may be different from that of the target domain. Under the assumption that the underlying data distributions of the two domains are not *very* different, we leverage an existing generative mixture model based classifier learned solely using the labeled data from the source domain, and adapt its parameters using both the labeled and unlabeled data sets from the target domain. The formulation and solution approach adopted by this method and the semi-supervised constraint based learning method are similarly motivated by the need to achieve label propagation (or constraint propagation) by imposing space-partitioning in the solution. In chapter 4, using publicly available Internet packet-flow traffic data from different temporal and spatial domains, we demonstrate significant classification performance improvements in the setting of semi-supervised domain adaptation.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I am very grateful to my advisor Dr. David Miller and my co-advisor Dr. George Kesidis for having kindly supported my entire PhD studies and for giving me the opportunity to work on a number of interesting research problems. I would like to thank Dr. David Miller for his constant encouragement and mentorship during the course of this work. Starting out as an amateur researcher, I have learned many valuable lessons from him about how to perform high quality research and communicate it cogently and precisely. This work would not have been possible without his vision, enthusiasm, and tireless work ethic, qualities which have also helped shape me as a better researcher and person. I would like to thank Dr. George Kesidis for his active involvement and interest in this work amidst his busy academic schedule. The insightful comments and discussions stemming from his vast knowledge have brought forth interesting ideas and perspectives, which improved the scope and quality of this work.

I would like to thank Dr. Kenneth Jenkins and Dr. Yu Zhang for being a part of my PhD committee and providing valuable comments and feedback on this work. The time and effort they spent on reviewing this work is much appreciated.

My venture into higher studies and research would not at all have been possible without the many sacrifices and selfless support of my parents and sister. I am indeed fortunate that they took care of so many aspects of my life so that I could focus on my studies.

I would like to mention my good friends Vivek Balasubramaniam, Nithin Sugavanam, and Chandrasekhar Mothali for their cheerful presence in my life. I am also thankful for the good camraderie with my friends and colleagues at Penn State - Anand, Vamshi Krishna, Chandraprakash, Umamahesh, Harisha, Mahesh, Aditya, Girish, Sushant, Ajit, Fatih Kocak, Fatih Tutuk, Berkay, Hossein, Abdullah, Zhicong, Korkut, Ashkan, and Ibrahim.

Dedication

Sarvam Sri Narayanaarpanamasthu

To my parents, grandparents, and my sister

Chapter 1

Introduction

Statistical learning methods attempt to learn functions, models, or rules from finite data collections in order to automatically make predictions or inference on unobserved data in a reliable manner. Different practical problems encounter different types of data and also have different learning goals. But broadly, the goal can be described as one of learning a relation between a variable (possibly vector valued) of interest (usually referred to as the *target*) and a number of other potentially informative variables (usually referred to as *features* or *covariates*). When the target variable takes values from a small set of categories, the learning problem is called *classification*, and when the target variable(s) takes values on the real line the learning problem is called *regression*. There are also other types of problems such as *time-series prediction*, where the goal is to predict the future values of a time-series based on its current and past values, and *clustering*, where is goal is find a grouping of points such that points within a group are more similar (based on a measure of similarity) to each other than to points in other groups. The work in this dissertation will focus largely on classification problems, and to a smaller extent on clustering problems.

Statistical learning problems can also be categorized based on the type and amount of supervision information (usually targets) available in a data set. Two such well-known, commonly addressed problems are that of *supervised learning* and *unsupervised learning* [51]. In the case of supervised learning, all the samples in a data set have a target value corresponding to the feature vector. The goal of this learning problem is to learn an accurate model for making target predictions on the entire feature space. Most of the theory and methods underlying this problem are well understood and are readily available to be applied to real world applications. In the problem setting of unsupervised learning, *none* of the samples in a data set have a target value (or supervision information) corresponding to the feature vector. The goal in this case is to learn

about the underlying structure or distribution of the feature vectors in a data set. Problems like clustering, anomaly detection, density estimation, and dimensionality reduction come under the category of unsupervised learning [51], and they also have been widely used in real world applications. It seems natural to consider the intermediate problem setting known as *semi-supervised learning*, where only a small fraction of samples in the entire data set have supervision information corresponding to their feature vectors, while the rest of the samples do not (they only have the feature vector). The goal, similar to supervised learning, is to learn a model which can provide accurate target predictions on unobserved data, but in case of semi-supervised learning it is of interest to utilize the samples which *do not* have targets to potentially learn a better model for target predictions than one that would be solely based on the small number of samples having targets (supervision). Typically, classification and clustering are the problems of interest in semi-supervised learning. This important problem setting is motivated and discussed in more detail in the sequel.

1.1 Generative and Discriminative learning

Consider a classification problem where the goal is to learn a class prediction function $f(\underline{x}) : \mathbb{R}^d \rightarrow \{1, \ldots, K\}$ from a data set $\mathcal{X} = \{(\underline{x}_i, c_i) \ : \ i = 1, \ldots, N\}$ with N samples, where $\underline{x}_i = [x_{i1}, \ldots, x_{id}]^T$ is the d-dimensional feature vector and c_i is the class label (target) for sample i. From statistical decision theory, it is well known that the optimal Bayes classification rule [51], [127] which minimizes the misclassification rate (error rate) is given by

$$f_{\text{bayes}}(\underline{x}) \ = \ \underset{c \in \{1,\ldots,K\}}{\arg\max} \ P(C = c \,|\, \underline{x}). \tag{1.1}$$

Since the true class posterior probability (given the feature vector) $P(C = c \,|\, \underline{x})$ for any real problem would be unknown, classification methods whose objective is to minimize the error rate try to find a good approximation to the true class posterior.

In the generative learning framework, an approximation to the true class posterior is found by learning a model $P(\underline{x}, c\,; \theta) = \pi_c \, P(\underline{x} \,|\, c\,; \theta)$ (with parameter set θ) for the joint probability distribution of the feature vector and the class [1], where π_c is the prior probability of class c and $P(\underline{x} \,|\, c\,; \theta)$ is the class-conditional density of the feature vector. Based on this model, the plug-in

[1] For convenience, we will sometimes use the overloaded notation for probability distributions or density functions, such as $P(\underline{x}, c)$ to denote $P(\underline{X} = \underline{x}, C = c)$.

Bayes class posterior

$$\widehat{P}(C = c \,|\, \underline{x}\,; \theta) \;=\; \frac{\pi_c \, P(\underline{x} \,|\, c\,; \theta)}{\sum_{k=1}^{K} \pi_k \, P(\underline{x} \,|\, k\,; \theta)} \tag{1.2}$$

can be calculated and used to make class predictions. The defining feature of generative learning or modeling is that based on the learned probability model it is possible to generate new data samples. Also, there is an always an underlying *stochastic data generation mechanism* underlying the model assumptions. This can be useful and provide some insights about addressing model limitations. The model for the class-conditional density of the feature vector $P(\underline{x} \,|\, c\,; \theta)$ is chosen based on the type of values taken by the features, the number of features, and the size of the data available to estimate the model parameters. Common choices include the multivariate Gaussian density, multivariate student-t density, mixture of multivariate Gaussian densities, mixture of Bernoulli or Multinomial distributions, and mixture of densities from the Exponential family [22], [127]. Methods such as linear discriminant analysis, quadratic discriminant analysis, mixture model based discriminant analysis, and their regularized variants [127] are generative classification methods obtained by making specific assumptions about the class-conditional density of the feature vector. Hidden markov models (HMM) are an example of a more complex generative modeling method [142]. A variant of the EM algorithm known as the Baum-Welch algorithm is used for parameter estimation of HMMs. Bayesian networks, which are probabilistic graphical models represented by directed acyclic graphs, are another example of generative models used for classification.

The parameters of the model are usually estimated using methods like maximum likelihood (ML) estimation and maximum-a-posteriori (MAP) estimation [51], [127]. For many parametric distributions, these estimation problems have closed form solutions. For mixture models and also certain multivariate densities (like Student-t), the Expectation-Maximization (EM) algorithm [48], [112], [125] is a computationally efficient, iterative method for ML and MAP estimation, which has good convergence properties. The ML and MAP estimation may be called *joint learning* methods, because they maximize the data log-likelihood based on the joint probability (of the feature vector and class) model. There is also a different flavor of parameter estimation called *conditional learning*, where the conditional distribution of the class given the feature vector is directly optimized in an estimation objective such as the conditional maximum likelihood [80]. The motivation here is that, since it is the class posterior distribution which is finally used for classification, it may be more efficient to work with this distribution directly than to work with the joint distribution (as done by ML and MAP objectives). While ML and MAP estimation for problems with latent variables and incomplete data can be handled tractably using

the EM algorithm, conditional learning methods usually have to resort to gradient based methods for maximizing the estimation objective. In [80] a method called the conditional expectation-maximization (CEM), an extension of the EM algorithm, has been developed to address this problem.

In the discriminative learning framework, no explicit attempt is made to model the underlying distribution of the features. Instead the entire effort is focused on learning the class posterior distribution or on learning the classifier mapping between the features and the class variable. In this sense, it is more consistent with what is called *Vapnik's principle* [33], [163] - "When trying to solve some problem, one should not try to solve a more difficult problem as an intermediate step". In one class of discriminative classification approaches, instead of approximating the Bayes class posterior via an intermediate model of $P(\underline{x}, c)$ (as done in generative learning), a suitable parametric model is directly chosen for the class posterior $\widehat{P}(C = c \,|\, \underline{x}, \theta)$ and the parameters are learned to minimize an objective such as the empirical error rate on training data, or more generally an empirical risk function [163]. For example, a linear logistic classification method uses the following parametric model for the class posterior:

$$\widehat{P}(C = c \,|\, \underline{x}\,; \theta) \;=\; \frac{e^{\underline{w}_c^T \underline{x} + b_c}}{\sum_{k=1}^{K} e^{\underline{w}_k^T \underline{x} + b_k}}, \;\; c = 1, \ldots, K\,.$$

The multilayer perceptron with suitably chosen activation functions in its output layer can be trained on labeled training data set to learn a function which approximates the true class posterior. Other examples of discriminative parametric classifiers include radial basis function networks, conditional random fields (CRFs), Gaussian processes, and maximum entropy discrimination [127], [51]. Instead of learning a model for the class posterior probability, some classification methods learn discriminant functions $G_c(\underline{x}\,; \theta_c),\; c = 1, \ldots, K$, one per class, which effectively define the classification rule and the boundaries between classes in the feature space. The classification rule for a feature vector $\underline{x}$ is $\widehat{C}(\underline{x}) = \arg\max_{c \in \{1,\ldots,K\}} G_c(\underline{x}\,; \theta_c)$ and the boundary between any two classes i and j is defined by $G_i(\underline{x}\,; \theta_i) = G_j(\underline{x}\,; \theta_j)$. Popular classification methods which fall under this category include linear and kernel based Fisher discriminant analysis, support vector machines, K-nearest neighbor classifiers, and decision trees (although the later two are actually non-parametric, rule based classifiers) [127], [51].

An important issue to understand is under what conditions is it suitable to choose one of the learning approaches (discriminative or generative) versus the other. The principle in support of discriminative learning is that, by directly modeling the class posterior distribution (or the class decision boundaries), it avoids having to solve a potentially more complex intermediate prob-

lem (as generative approaches do by modeling the joint distribution $P(\underline{x}, c)$) as a step towards learning the final classifier. This can lead to a better utilization of the labeled training data in estimating the classifier parameters. On the other hand, generative models are attractive for certain problem domains [79] because they can incorporate some elegant probabilistic concepts such as latent variables, stochastic data generation, and prior probabilities (in a Bayesian framework). They can usually handle missing values in the data along with parameter learning in an integrated manner [127]. Also, the relationship between the feature variables and the class can be made explicit in generative models and visualized using probabilistic graphical models. In [86], the issue of discriminative versus generative classifier learning is discussed in a supervised learning setting, and experimental results on the relative performance of the logistic regression classifier and a classifier based on naive Bayes, class conditional Gaussian density model are provided. The experiments suggest that discriminative learning methods tend to have lower asymptotic error rate (as the number of labeled data increases) compared to generative learning methods [86]. However, generative learning methods tend to converge more quickly to their (higher) asymptotic error rate, which implies that they may have better performance when the number of labeled data is small [86].

It may be noted that there has been some work on hybrid approaches which combines both the discriminative and generative learning principles into a single method for learning classifiers [77], [70], [71], [96], [56]. A comprehensive discussion on generative and discriminative learning can be found in chapter 2 of [79].

1.2 Inductive and Transductive inference

Recall that we defined a classification problem as one that learns a function $f(\underline{x}) : \mathbb{R}^d \rightarrow \{1, \ldots, K\}$, such that the classifier can make predictions on any point $\underline{x}$ in the feature space. In this sense, the classifier is learning a function to induce class decisions on the entire feature space based on a finite training data set. This method of class inference by learning to generalize on any sample point is commonly referred to as *inductive inference*. On the other hand, there may be classification problems where it is only required to make class predictions on a finite set of feature vector samples $\mathcal{X}'$ outside the training data set. These samples are also provided as input to the classification method (without class labels of course), and the objective in this case is to learn a function which makes class predictions only on the finite set of samples $\mathcal{X}'$, *i.e.*, $f(\underline{x}) : \mathcal{X}' \rightarrow \{1, \ldots, K\}$. This approach is commonly known as *transductive inference*, and in some problem settings it may be simpler or a more natural mode of inference compared to inductive inference [121], [123]. Chapter 25 of [33] provides an interesting discussion on

transductive inference in the context of semi-supervised learning.

1.3 Background on semi-supervised learning

Traditionally, supervised classification problems use only labeled data (set of feature vector, class label pairs) for learning classifiers. However, it is often expensive and time-consuming to obtain class labels for the entire data set since this may require expertise from human annotators, special devices, or expensive and slow experiments [185]. To appreciate this, consider a few practical problems [185]. In speech recognition, the task of transcribing long records of speech data can be time consuming and requires human expertise. In the problem of spam email filtering, the availability of labels (spam or not-spam) depends upon the patience and commitment of individual email users towards provide feedback about spam emails. In video surveillance data, it may be of interest to locate and identify objects or people of interest in a video, which can be a time consuming task requiring expertise. On the other hand, unlabeled data consisting of just a set of feature vectors is comparatively easy to collect and likely to be available in large quantity. However, unlabeled data have no use in the supervised learning framework. So a natural question to ask is whether the performance of classification (or other prediction tasks) can be improved by making use of the readily available unlabeled data. Semi-supervised learning addresses precisely this question and indeed, when the underlying model assumptions made by a semi-supervised learning method are valid for a particular data set, unlabeled data can be helpful in improving prediction performance. From a different standpoint, it may be said that semi-supervised learning methods try to achieve a similar performance level as supervised learning methods, but with a smaller number of labeled samples, hence reducing the cost required to label the data [185].

We next describe two common problem variants in semi-supervised learning, namely *semi-supervised classification* and *semi-supervised constraint-based learning*. The semi-supervised classification problem consists of a labeled data set $\mathcal{X}_l = \{(\underline{x}_i, c_i),\ i = 1, \ldots, L\}$ and an unlabeled data set $\mathcal{X}_u = \{\underline{x}'_i,\ i = 1, \ldots, U\}$, where $L \ll U$. The goal is the same as that of supervised classification, *i.e.*, to learn a class posterior or a decision function on the entire feature space in the case of inductive inference, and only the unlabeled data set $\mathcal{X}_u$ in the case of transductive inference. In the semi-supervised constraint-based learning problem, an unlabeled data set $\mathcal{X}_u$ is augmented with a set of constraints on the samples in $\mathcal{X}_u$. Constraints are usually in the form of whether pairs of samples in $\mathcal{X}_u$ are from the same class (*must-link constraints*) or from different classes (*cannot-link constraints*). Such constraints are also known as *pairwise-sample constraints*, and the number of such constraints available in a typical problem will be

small, *i.e.*, a large number of samples may not be involved in any constraints. The learning objective of this problem is to find an accurate grouping of the samples into a set of classes such that all or most of the provided constraints are satisfied. Note that such kind of pairwise-sample constraints only provide class label information of sample pairs relative to each other. Accordingly, a learning method will only be able to predict the class assignments of samples that are consistent relative to a labeling convention chosen by the method.

1.3.1 Early work on semi-supervised learning

One of the earliest approaches (around the year 1967) of combining unlabeled data into the learning of a classifier is the so called *self-learning* or *self-training* method [54], [172], [147]. This is an iterative method, where initially a supervised learning method is trained using only the available labeled data. Based on its decision function, a subset of the unlabeled samples (usually of small size) which have the most confident class predictions are treated as labeled samples in subsequent iterations with their predicted class treated as their class label. These two steps of supervised classifier learning and augmenting the labeled data subset are repeated until all the unlabeled samples have been assigned class labels and added to the labeled training set. This method is simple and can be based on any classifier model that can be learned in a supervised fashion. However, a major limitation of self-learning is that, if there are quite a few class predictions errors made by the initial classifier model (based only on the small set of labeled data), then these errors tend to quickly multiply in the subsequent iterations creating more class prediction errors because of the erroneous labeling, and so forth. This is illustrated with a propagating 1-nearest neighbor self learning method in [33]. Also, it becomes difficult to detect when such a cascading effect of erroneous labeling may occur withing self-learning.

A few related semi-supervised learning methods based on modeling the class-conditional density of the features as a multivariate Gaussian with a shared covariance matrix across the classes has been explored in some early works (around the period after 1975) such as [72], [114], and [137]. With this model , the classifier is restricted to learning linear decision boundaries between the classes. However, the real interest in semi-supervised learning started picking up after the year 1995 [33]. Initial work in semi-supervised learning has largely focused on generative modeling because given a model for the distribution of the feature vector, they find direct value for utilizing unlabeled data in the learning process. In [154], [122], and [135], generative semi-supervised learning methods based of mixture models have been proposed. All three methods allow the samples from each class to be modeled with multiple mixture components, however the methods of [154] and [135] constrains the components/clusters to be class pure, while the method of [122] allows a probabilistic assignment of classes to components. These methods

utilize the unlabeled data by including the log-likelihood component of the unlabeled data in the estimation objective, and the Expectation-Maximization (EM) algorithm is used for parameter estimation.

The co-training method for semi-supervised learning proposed in [19] and the transductive support vector machine (TSVM) method proposed in [84] represent some early work in semi-supervised learning based on the discriminative learning framework [2]. In the co-training method [19], it is assumed that the set of features can be divided into two subsets (called views) such that they are conditionally independent given the class, and such that each subset of features by itself has sufficient predictive information about the class variable. Given two such views, the co-training method iteratively trains two classifiers, one based on each view (subset of features). Initially each classifier is trained using only the labeled data, and the trained classifier is used to make class predictions on the unlabeled data (based on their respective views). Then a certain number k of the most confident predictions of each classifier on the unlabeled data are added to the labeled data subset of the *other classifier* and this process of training and augmenting each other's labeled data subset is iteratively repeated until all the unlabeled data is exhausted. The idea behind such an iterative approach is to make the predictions of the two classifiers (based on different views) agree closely on the unlabeled data [3]. The co-training method, like self-training, is also a wrapper method for semi-supervised learning, meaning that any classifier model which can assign confidence values to their class predictions is suitable for co-training. Also, note that the assumptions made by co-training about the two feature subsets are important for it to perform well in practice. We discuss the TSVM method in the next section after providing some context for the discussion.

1.3.2 Motivations for semi-supervised learning

We briefly mentioned why it may be more natural to incorporate unlabeled data into the learning objective of a generative model based classifier. Unlabeled data provide knowledge about the distribution of the features vector, and since generative classifiers model this distribution as $P(\underline{x}\,;\theta) \;=\; \sum_{k=1}^{K} \pi_k \, P(\underline{x} \,|\, k\,;\theta)$, it should be evident from plug-in Bayes rule for the class posterior (1.2) that $P(\underline{x}\,;\theta)$, and hence the unlabeled data also affect the learning of the class posterior model. In other words, the class posterior distribution and the feature vector distribution share parameters. It is believed that this is an important condition to be satisfied by a semi-supervised learning method, without which there cannot be much value from the unlabeled

[2] Actually the co-training method can be based on either a discriminative or a generative classifier model.

[3] There is theory behind the idea of learning multiple classifiers or hypotheses such that they have close agreement on their predictions [102]

data [153], [182].

On the other hand, discriminative learning methods completely by-pass the learning or modeling of the feature vector distribution . Therefore, it is not obvious how the unlabeled data can be utilized for semi-supervised learning of a discriminative classifier. As we next discuss, discriminative semi-supervised methods establish this connection between the class posterior distribution and the feature vector distribution by making some assumptions about the problem (sometimes explicitly and sometimes tacitly) [33], [185].

1.3.3 Categorization and survey of prior work

We discuss some of the common assumptions underlying semi-supervised learning methods which enable them to take advantage of unlabeled data [33], [182].

Cluster assumption

The cluster assumption for semi-supervised learning [33] is that data points (samples) from the same cluster are likely to have the same class labels, and hence the prediction function (or class posterior) of a semi-supervised learning method on points from the same cluster should be close in value. Formally, a *cluster* may be defined as a group of points such that any two points in the group can be connected via a path that passes through a high density region of points [33]. To see why how the cluster assumption can be useful, consider a simple (but sub-optimal) classification method which learns in two stages. In the first stage, the method finds a clustering of all points (both labeled and unlabeled) based on some popular clustering method such as K-means, hierarchical clustering, or mixture modeling. In the second stage, for each cluster, the method counts the number of labeled samples from each class and assigns the cluster to the majority class [4]. For this method, the unlabeled samples inform the method about the cluster structure of the data. Then the method invokes the cluster assumption to assign the same class label to all samples withing a cluster. Note that the method ignores information about the diversity of class labels withing a cluster.

Low density separation assumption

The low density separation assumption for semi-supervised learning [33] is that data points from different classes should have a region of low sample density separating them. In terms of semi-supervised classifier learning, this would imply that methods should try to learn a solution where the class boundaries end up in low density regions of the feature space. Semi-supervised learning

[4] If there are ties, they are broken randomly. If there are no labeled samples within a cluster found by the method, then it is assigned the class label assignment of the nearest cluster (based on a notion of distance between two clusters).

methods can exploit the information provided by unlabeled data about the low and high density regions of the feature space, so that they can learn a solution that is consistent with this assumption. The low density separation assumption is closely related to the cluster assumption, and in fact, methods which are consistent with one assumption will also be consistent with the other. To see this, note that the low-density separation assumption implies that class boundaries (which are surfaces in a high-dimensional space) should not cut across regions with high sample density, *i.e.*, they should not cut across a cluster of points. This implies that the low-density separation assumption would *not* be consistent with classification solutions where points within a cluster have different class predictions. This is also the idea behind the cluster assumption.

Manifold assumption

The manifold assumption for semi-supervised learning [33] is based on the idea that real world data often approximately lie on low dimensional manifolds in the feature space. This is useful when the feature space is high dimensional, because both direct estimation of parameters (in density models) and the direct calculation of distances (Euclidean) between points in the high dimensional feature space is prone to be inaccurate. Also, the number of samples required to make reliable parameter estimation increases exponentially with the feature dimensionality. These problems of learning in high dimensions are commonly referred to as the *curse of dimensionality* [51]. In the case of semi-supervised learning, unlabeled data may be useful in informing the method about any low dimensional manifold structure, which can be exploited by methods to handle the high dimensionality problem.

These three learning assumptions are sometimes grouped under the broader idea of the *semi-supervised smoothness* assumption [33], where the idea is that points in a high density region of the feature space are likely to have the same class labels (or targets), and hence a learning method should make similar class predictions for them.

Generative model assumption

As we discussed earlier, when a semi-supervised learning method assumes that the class conditional density of the feature vector can be modeled well according to a parametric distribution (or mixture distribution), the unlabeled data are helpful in learning the parameter estimates for the feature vector distribution and also the class posterior distribution. Some methods based on generative mixtures model classes as consisting of one or more components (clusters) [135], [154]. In this case, since points from a cluster will have the same class prediction, it can also be said that such methods are based on the cluster assumption.

There are a number of methods in literature which are either directly based on one of these

assumptions or whose learning method is implicitly consistent with one these assumptions.

Literature survey

In [135], a mixture of Multinomial distributions learned using the EM algorithm is used for text classification. Each component in the mixture is mapped to one of the classes, thus making its solution consistent with the cluster assumption. They show improvements in text classification performance by the use of unlabeled data in addition to the labeled data. In [122], a method called the semi-supervised mixture of experts classifier is proposed. Here also, a generative classifier based on a mixture model is used, with a component-conditional probability mass function (pmf) over the classes (one per component) which allows samples from a component to be probabilistically associated with multiple classes. This may allow more flexibility in the model compared to [135]. In [120], a novel type of semi-supervised generative mixture was proposed, wherein unlabeled samples may arise from one of the known (observed) classes or they may arise from an unknown class. Also, in the mixture model, the fact of label presence or absence is treated as observed data (and modeled appropriately) in addition to the feature vector and the class label. These extensions allow the model to handle classification problems with unlabeled data in a more robust way and also to possibly discover new/unknown classes in the data. A generalized mixture model for unknown class discovery in semi-supervised learning based on the work of [120] was also proposed in [55].

Earlier, we discussed the co-training method for semi-supervised learning. In [134], extensive experiments show that co-training performs well provided the two assumptions on the feature subsets (views) are satisfied. They also propose some improvements and extensions to co-training in a method called co-EM. Multiview learning [20], [156] is a learning framework similar to co-training, where multiple prediction models (like support vector machines, decision trees) are trained on the labeled data such that there is agreement in their predictions on the unlabeled data.

In [84], a method for learning a support vector machine from both labeled and unlabeled data called TSVM was proposed. The goal is to find a linear separating hyperplane (possibly in a higher dimensional feature space via a kernel mapping) and the class predictions of the unlabeled data such that the margin is maximized on both the labeled and the unlabeled data. Since the method only finds class predictions on the unlabeled data set, it is called transductive. However, its separating hyperplane solution can be used to inductively predict on unseen data points. Also, the optimization problem of TSVM is non-convex and an NP-hard problem, and a number of methods have focused on finding efficient locally optimal solutions [84], [32], [34], [40]. Finally, note that the TSVM method follows the *low density separation* assumption since

it places its decision boundary such that the margin relative to both labeled and unlabeled data is maximized.

An information theoretic regularization framework for semi-supervised learning is proposed in [161]. Given the feature vector distribution $P(\underline{x})$, either based on an estimated model or based on the empirical distribution of data, the idea is that the class labels should be more homogeneous in regions of high sample density (similar to the idea of the semi-supervised smoothness assumption). The mutual information between the feature vector and the class variable gives a measure of class homogeneity because, for regions where the class labels are nearly the same the mutual information will be close to 0, and the mutual information increases as the variety of class labels increases. By dividing the feature space into multiple overlapping regions, an objective for minimizing the maximum mutual information over all the regions, subject to a constraint of accurate prediction on the labeled data is proposed. The method provides a framework for linking the feature vector distribution and the class posterior distribution without making any parametric model assumptions about the class posterior. This work is further developed in [41], where a regularization penalty is added to the class posterior log-likelihood in the learning objective in order to incorporate the ideas of class homogeneity in regions of high sample density. They also develop the theory behind information regularization in a learning theoretic framework.

In [60], an entropy minimization semi-supervised learning framework for discriminative classifiers is proposed. Specifically, the learning objective is to maximize the class posterior log-likelihood of the labeled data regularized by the negative conditional entropy of the class posterior (given the feature vector) calculated on the unlabeled data. The conditional entropy regularizer encourages the method to find solutions whose class boundaries lie along low sample density regions of the feature space (consistent with the low density separation assumption). The contribution of unlabeled data can be controlled via the constant factor multiplying the entropy regularization term, but this constant has to be set carefully, for example using cross-validation. This learning objective is suitable for any parametric discriminative classifier, and in [60] they experiment with a logistic regression model. In [74], the conditional entropy regularization framework is used to train a Gaussian mixture model classifier from combined labeled and unlabeled data. The parameter optimization is solved using a pre-conditioned conjugate gradient method, since it cannot be iteratively solved using the EM algorithm or its extensions. However, the component covariance matrices of the Gaussian mixture are not optimized. They experiment on the task of phonetic classification and compare against the maximum likelihood objective for learning a Gaussian mixture classification model.

Graph based semi-supervised learning methods represent another important class of methods which are usually based on the manifold assumption. The combined labeled and unla-

beled data are represented by the nodes of an undirected graph, and the edges are assigned a similarity weight calculated from a metric of similarity between the two nodes (in an edge). For example, the graph could be fully connected with the edge weights decaying according to the distance between the points represented by the nodes. A commonly used edge weight is $w_{ij} = \exp(-\|\underline{x}_i - \underline{x}_j\|^2 / \sigma^2)$. Other types of edge weights such as the k-nearest neighbor graph and the ϵ-nearest neighbor graph are also possible [185]. The goal is to propagate the label information from the labeled nodes of the graph to the unlabeled nodes. This is made possible through the notion of adjacency between nodes based on the edge weights. The key idea or assumption among graph based methods is that the labels are smooth with respect to the graph, *i.e.*, two nodes that are connected by a large weight will tend to have the same class labels (or similar targets) [185]. Also, if the distance between two nodes is calculated as the shortest distance among all paths connecting the nodes, then this distance may be taken as an approximation of the geodesic distance between the points on an underlying manifold [10]. This way graph based methods can capture any underlying low dimensional manifold structure in the data. These intuitive ideas of smoothness and regularization are made concrete using spectral graph theory [10], [185]. A number of graph based semi-supervised learning methods such as mincut [17], [18], [168], local and global consistency [180], [61], label propagation [183], Gaussian fields and harmonic functions [184], and manifold regularization [10], [8], [9] have been developed. Finally, we note that graph based semi-supervised learning methods are typically non-parametric and discriminative, and performs transductive inference on the unlabeled samples.

There are some works which address the theoretical aspects of semi-supervised learning, such as the value of unlabeled data, and generalization error bounds on classifier learning. In general, semi-supervised learning does not always help to improve the classification performance. This can happen when the assumptions made by a semi-supervised learning method are not appropriate for a problem. For example, if the samples from the different classes are not well separated and have significant overlap, then a method based on the low density separation assumption will not be appropriate for the problem. Likewise, if the feature vector distribution in samples is not well-modeled by the parametric density assumed by a generative semi-supervised learning method, it will be mismatched to the problem. In such cases, adding unlabeled data may in fact lead to degradation of classification performance. This phenomenon has been studied in the context of generative classifiers by [43], [44], and [38]. The effect of model mis-specification for semi-supervised classification has been studied in [171]. The value of unlabeled data for classification problems in general has been studied in [28], [29], and [177]. Some works addressing the generalization error performance of semi-supervised classification are [2], [3], and [144].

It is important to mention that the discussion on semi-supervised learning presented here is based on the comprehensive discussions and literature surveys found in [33], [185], and [182], which are also valuable resources for an in-depth study of this vast area of research. We have attempted to provide an overview of the different flavors of semi-supervised learning in order to connect and relate the large number of methods addressing this problem.

1.4 Background on domain adaptation of classifiers

In this section, we provide background and motivation for the classifier domain adaptation problem, and review some of the existing work in literature which address this problem. In standard statistical learning problems, a prediction function or rule is learned based on a training data set such that it can predict or generalize well on new data samples that are generated according the same underlying distribution that generated that training data set. This assumption is reasonable to make in certain problem settings like supervised, semi-supervised, and unsupervised learning. Classifier domain adaptation addresses problems where such an assumption may not be valid, *i.e.*, the joint distribution of the feature vector and the class may be different in the training and test data sets.

We next consider some realistic problem scenarios where domain adaptation may be useful. (i) Suppose the data collection (or recording) is based on a number of sensors, each of which has some associated recording noise. If the time instants of recording the training and test data sets are significantly different, it is likely that the distribution of the test data will be different (from that of the training data) because of factors such as changes in the sensor noise distributions and changes in the recording conditions. (ii) Consider the problem of object recognition in images where the goal is to classify objects in an image into a set of predefined categories. It is quite possible that images in the training and test data sets get captured under different conditions such as varying illumination, varying perspectives, varying degrees of occlusion, differences in the type of camera *etc.*. Now if regions in the images are represented as feature vectors with a corresponding object category (class) label, then due to the changes created during the image capture, we would expect the class-conditional distribution of the feature vector to be different in the training and test data sets. It is also possible that the proportion of samples from the different object categories could be different in the training and test data sets thus creating class prior imbalance. (iii) In some problems domains, the data might consist of a categorical valued variable (or variables), conditioned on which the joint distribution of the feature vector and the class changes. For example, the categorical variable could represent some contextual information such as the race or gender of people in a demographic data set. In this case, if

the training and the test data sets consist of records of people from different races (or gender), then the context difference would create a difference between their distributions. (iv) Finally, consider the problem of personalizing email spam filters specific to the users inbox. In this problem, the training data for learning a generic spam classifier will be based on an aggregated (public) collection of emails and their class labels (spam or not-spam) from many users. Since the goal is to personalize this generic spam classifier to a specific user's emails, the test data will consist of emails of the specific user and any partial labeling provided by that user for his own emails. It is obvious that the distribution of words and word sequences found in the emails of a specific user are likely to be different from those found in the training data. Also, the specific user may have somewhat different criteria for classifying an email as spam compared to the aggregated criteria of multiple users.

Consider a classification problem with d dimensional feature vector $\underline{X} \in \mathbb{R}^d$ and a class variable $C \in \{1, \ldots, K\}$. Consistent with the terminology commonly used in the domain adaptation literature [81], [82], [83], we will refer to the domain on which the classifier is trained as the *source domain* and the domain where the classifier will be deployed to make predictions as the *target domain*, although sometimes they are also referred to as the *training domain* and *test domain* respectively. Suppose the joint probability distribution of the feature vector and the class in the source and target domains are denoted by $P_s(\underline{x}, c)$ and $P_t(\underline{x}, c)$ respectively. Similarly, we denote the distribution of the feature vector in the source and target domains by $P_s(\underline{x})$ and $P_t(\underline{x})$ respectively [5], and the distribution of the class variable in the two domains by $P_s(c)$ and $P_t(c)$. Similar notation is used to denote the conditional distributions. In the most general problem of domain adaptation $P_s(\underline{x}, c)$ and $P_t(\underline{x}, c)$ are different. Hence, a classifier trained on labeled data from the source domain, cannot be expected to make good predictions for data from the target domain without some adaptation of its parameters. There are many variants of this problem and also some special problems that have been addressed in earlier work for specific application areas.

1.4.1 Related problem setting

In domain adaptation problems, the target domain is always assumed to have a large unlabeled data set. When there is no labeled data available in the target domain, the scenario is called unsupervised domain adaptation, whereas the scenario where the target domain consists of a small labeled data set is called semi-supervised domain adaptation. Both problem settings have been

[5] For simplicity, sometimes we denote probability distributions with an overloaded notation such as $P(c)$ instead of $P(C = c)$, and let the arguments of $P(.)$ distinguish the distributions. When this is not clear, for example the probability of a specific value k, then we denote it as $P(C = k)$.

addressed in past works. There are a few closely related problems to domain adaptation, such as transfer learning, multi-task learning, class imbalance, sample selection bias, and covariance shift [81]. In *transfer learning*, the knowledge available from one or more tasks or domains is utilized in an efficient way to solve related problems in other domains. *Multi-task learning* attempts to solve multiple related tasks at the same time using some kind of a shared representation. This kind of learning often leads to a better model for the different tasks by exploiting the commonality among the tasks. *Class imbalance* refers to the situation where it is assumed or known that $P_t(\underline{x} \,|\, c) = P_s(\underline{x} \,|\, c),\ \forall c \in [K]$, but $P_t(c) \neq P_s(c),\ \forall c \in [K]$. That is class prior distributions may be quite different in the two domains and needs to corrected for during classifier adaptation. *Sample selection bias* or sampling bias is said to occur in a data set when the underlying data generation mechanism is biased in one domain compared to the other, and certain regions of the feature space are less likely to contain samples than others. Hence, a classifier or predictor model learned on such a data distribution may not have good performance in those regions of the feature space. *Covariance shift* is a problem where it is known or assumed that $P_t(c \,|\, \underline{x}) = P_s(c \,|\, \underline{x}),\ \forall \underline{x}$, but $P_t(\underline{x}) \neq P_s(\underline{x})$. That is the class posterior distribution given a feature vector is the same in the two domains, but the marginal distribution of the feature vector is different in the two domains. Class imbalance, sample selection bias, and covariance shift are all special kinds of domain adaptation problems, but these related problems might have simpler solutions.

1.4.2 Survey of prior work

In [25], [50], [49], labeled data in the source domain is used to learn a parametric model for the class conditional density function of features (in [50] and [49] specifically a Gaussian mixture model), and to estimate the class prior probabilities. This classifier model is adapted to the target domain by maximizing the log-likelihood of features on the target domain data. This is done using an EM algorithm [48], which is initialized with the parameters of the source domain classifier and iteratively updated to a reach a local maximum of the log-likelihood. The authors in [25] use this method in the context of remote sensing image classification for land cover maps, while [50] and [49] use this method in the context of speaker adaptation. In [23], this method is extended by the use of a cascade-classifier approach to capture the temporal correlation between remote sensing images acquired from the same area at different times, where labeled data is available only at the initial time instant. The joint density of the image feature vectors at two successive time instants are assumed to be conditionally independent given the class, and a Bayes class posterior is used to make predictions. The class conditional density of features at the initial time is estimated using the labeled data. An EM algorithm is used to estimate the class

conditional density of features in the image at the next time instant, and also the joint probability mass function of classes in the two images considered.

In [24], a domain adaptation technique for support vector machines called DASVM is proposed. Also, a method for validating the domain adapted classifier, *i.e.* identifying reliable solutions for the target domain by using only labeled data from the source domain, is proposed. The method draws from principles of semi-supervised and transductive SVMs [84], where the learning algorithm makes use of both labeled and unlabeled data which are assumed to be obtained from the same underlying probability distribution. For the domain adaptation problem this assumption does not hold. In their method, the labeled samples from the source domain are used to learn an initial, presumably unreliable SVM solution for the target domain. Then unlabeled samples from the target domain are used to adjust the decision function iteratively, obtaining *candidate* labels in the process. The labeled samples from the source domain are gradually erased, and the final SVM solution is determined only on the basis of the target domain samples(which obtain proxy labels during the learning).

In [150], a domain adaptation method is proposed which selects a subset of features for which the deviation between the source and target domains (measured using some metric) is minimized, while maximizing the likelihood of the source domain data. Instead of actually selecting the feature subset, they use an objective function that penalizes features which have large deviation between source and target domains, so that less distorted features have more influence on the solution. Note that the features they define are actually composite feature functions of the input and labels. In [16], [11], a method called Structural Correspondence Learning (SCL) is proposed. The key idea is to identify correspondences among features from different domains by modeling their correlations with what are called *pivot* features, which behave in the same way for discriminative learning in the two domains. A model trained on the source domain using this common feature representation is shown to generalize better to the target domain.

The methods discussed so far are unsupervised domain adaptation methods. We now look at a few semi-supervised domain adaptation methods. In [37], a Maximum Entropy classifier is first trained on the source domain data, and the weights of the classifier are used as the mean value for a Gaussian prior which is used for MAP estimation on the target domain. In [82], an instance weighting framework is proposed for domain adaptation, where several heuristics are proposed to remove misleading labeled samples from the source domain, assign more weight to labeled target domain samples than labeled source domain samples, and augment the training set with predicted target domain samples. In [121], a transductive approach is used to address the distributed ensemble classification problem, where the individual classifiers are built using different training data sets, and possibly use different feature spaces. Also, there is no common

pool of labeled data for supervised learning of the aggregation function. Maximum likelihood and information theoretic approaches are developed to account for the mismatch between the class priors used by individual classifiers and those reflected in the test data. The information theoretic approach also accounts for missing classes in the test data. In [45], a domain adaptation technique for maximum entropy classifiers is developed, where the joint probability distribution of the input features and output is modeled as a mixture of three components, one specific to the source domain, one specific to the target domain, and one common to both domains. The conditional expectation maximization (CEM) algorithm is used to solve the problem.

With this we complete our somewhat brief background and literature survey of classifier domain adaptation. In the next section we review some statistical methods which are relevant to the methods proposed in this dissertation.

1.5 Maximum likelihood estimation and Expectation-Maximization

As we discussed in section 1.1, ML and MAP estimation are widely used for learning generative classification models. For a large number of distributions belonging to the exponential family, the ML and MAP estimates can be directly found as closed form expressions [127]. However, this is not the case when the data is modeled using mixture distributions such as a mixture of Gaussians or a mixture of Multinomials. For such distributions, and also in general when there are missing/latent/hidden variables in the data, the Expectation-Maximization (EM) algorithm and its variants [48], [111], [145], [125] are widely used to find the ML or MAP parameter estimates. We next review this approach in the context of ML estimation for a model with latent variables. Mixture distributions are a special case, where the mixture components of origin of the samples are the latent variables.

Consider a data set of observed samples $X = \{x_1, \ldots, x_n\}$, where $x_i \in \mathcal{X},\ i \in [n]$ [6] can be vector valued in general. Let $Z = \{z_1, \ldots, z_n\}$ be the set of latent/missing values corresponding to the samples in X such that $z_i \in \mathcal{Z},\ \forall i \in [n]$. Suppose it is of interest to model the data according to a probability distribution $P(x \,|\, \theta)$, which can also be written as

$$P(x \,|\, \theta) \;=\; \sum_{z \in \mathcal{Z}} P(x, z \,|\, \theta) \;=\; \sum_{z \in \mathcal{Z}} P(z \,|\, \theta)\, P(x \,|\, z, \theta).$$

For example, in a mixture distribution with K components, the latent variable $z \in \{1, \ldots, K\}$ is the component of origin of x, and $P(z = k \,|\, \theta)$ is the prior probability of component k. Assuming that the samples in X are generated independently according to $P(x \,|\, \theta)$, the log-

[6] For notational convenience, we define $[m] = \{1, \ldots, m\}$ for any positive integer m

likelihood of the observed data (or incomplete data) X under the model is given by

$$\mathcal{L}_{\text{inc}}(\theta) \;=\; \log P(X \,|\, \theta) \;=\; \sum_{i=1}^{n} \log \sum_{z_i \in \mathcal{Z}} P(z_i \,|\, \theta)\; P(x_i \,|\, z_i, \theta), \tag{1.3}$$

and the log-likelihood of the complete data $Y = \{y_i = (x_i, z_i),\ i \in [n]\}$ is given by

$$\mathcal{L}_{\text{com}}(\theta) \;=\; \log P(Y \,|\, \theta) \;=\; \sum_{i=1}^{n} \log[P(z_i \,|\, \theta)\; P(x_i \,|\, z_i, \theta)]. \tag{1.4}$$

In maximum likelihood estimation, the goal is to maximize (1.3) with respect to the model parameters. This may not have a direct closed-form solution because of the summation term inside the logarithm in the incomplete data log-likelihood. When the distribution $P(x_i \,|\, z_i, \theta)$ is from the Exponential family, maximizing the complete data log-likelihood (1.4) is straightforward and has closed-form solutions. However, this is not practical because the complete data log-likelihood is based on missing data, and cannot be calculated. The Expectation-Maximization (EM) algorithm [48], [111], [145], [125] provides a tractable iterative approach for solving the maximum likelihood problem. The EM algorithm and its extensions have some attractive theoretical properties and are used to solve a number of problems, especially those with missing data.

We next show how the EM algorithm is developed for a log-likelihood maximization problem. Assuming independent generation, consider the posterior distribution of the latent variable z_i given the feature vector x_i and parameters θ for any sample $i \in [n]$, given by

$$P(z_i \,|\, x_i, \theta) \;=\; \frac{P(z_i \,|\, \theta)\; P(x_i \,|\, z_i, \theta)}{\sum_{z \in \mathcal{Z}} P(z \,|\, \theta)\; P(x_i \,|\, z, \theta)}. \tag{1.5}$$

Also consider an unspecified posterior distribution $\overline{P}(z_i \,|\, x_i) \in [0, 1]$ such that $\sum_{z \in \mathcal{Z}} \overline{P}(z \,|\, x_i) = 1$. The incomplete data log-likelihood can be re-written as

$$\mathcal{L}_{\text{inc}}(\theta) \;=\; \sum_{i=1}^{n} \log \sum_{z \in \mathcal{Z}} P(z \,|\, \theta)\; P(x_i \,|\, z, \theta) \;=\; \sum_{i=1}^{n} \sum_{z_i \in \mathcal{Z}} \overline{P}(z_i \,|\, x_i)\; \log \sum_{z \in \mathcal{Z}} P(z \,|\, \theta)\; P(x_i \,|\, z, \theta)$$

Using the expression $\sum_{z \in \mathcal{Z}} P(z \,|\, \theta)\; P(x_i \,|\, z, \theta) \;=\; \dfrac{P(z_i \,|\, \theta)\; P(x_i \,|\, z_i, \theta)}{P(z_i \,|\, x_i, \theta)}$ obtained from (1.5) inside the logarithm, and then multiplying and dividing by the $\overline{P}(z_i \,|\, x_i)$, the incomplete data

log-likelihood can be further rewritten as

$$\begin{aligned}
\mathcal{L}_{\text{inc}}(\theta) &= \sum_{i=1}^{n} \sum_{z_i \in \mathcal{Z}} \overline{P}(z_i \,|\, x_i) \, \log[\frac{P(z_i \,|\, \theta) \, P(x_i \,|\, z_i, \theta)}{P(z_i \,|\, x_i, \theta)}] \\
&= \sum_{i=1}^{n} \sum_{z_i \in \mathcal{Z}} \overline{P}(z_i \,|\, x_i) \, \log[\frac{P(z_i \,|\, \theta) \, P(x_i \,|\, z_i, \theta)}{P(z_i \,|\, x_i, \theta)} \frac{\overline{P}(z_i \,|\, x_i)}{\overline{P}(z_i \,|\, x_i)}] \\
&= \sum_{i=1}^{n} \sum_{z_i \in \mathcal{Z}} \overline{P}(z_i \,|\, x_i) \, \log[\frac{P(z_i \,|\, \theta) \, P(x_i \,|\, z_i, \theta)}{\overline{P}(z_i \,|\, x_i)}] + \sum_{i=1}^{n} \sum_{z_i \in \mathcal{Z}} \overline{P}(z_i \,|\, x_i) \, \log \frac{\overline{P}(z_i \,|\, x_i)}{P(z_i \,|\, x_i, \theta)}
\end{aligned}$$

In the above decomposition of the log-likelihood, the second term on the right hand side of the last equation is the Kullback-Leibler (KL) distance [42] between the distributions $P(x_i \,|\, z_i, \theta)$ and $\overline{P}(x_i \,|\, z_i)$, which has the property that it is always non-negative and is equal to 0 if and only if the two distributions are equal. This implies that $\mathcal{L}_{\text{inc}}(\theta)$ is always an upper bound of the first term on the right hand side, with equality if and only if the two distributions are equal.

Starting from initial parameter estimates $\theta^{(0)}$, the EM algorithm iteratively updates the parameter estimates such that the incomplete data log-likelihood increases monotonically. Suppose $\theta^{(t)}$ are the parameter estimates at iteration t. By setting $\theta = \theta^{(t)}$ and $\overline{P}(z_i \,|\, x_i) = P(z_i \,|\, x_i, \theta^{(t)})$ in the final expression for the incomplete data log-likelihood, we see that

$$\begin{aligned}
\mathcal{L}_{\text{inc}}(\theta^{(t)}) &= \sum_{i=1}^{n} \sum_{z_i \in \mathcal{Z}} P(z_i \,|\, x_i, \theta^{(t)}) \, \log[\frac{P(z_i \,|\, \theta^{(t)}) \, P(x_i \,|\, z_i, \theta^{(t)})}{P(z_i \,|\, x_i, \theta^{(t)})}] \\
&\geq \sum_{i=1}^{n} \sum_{z_i \in \mathcal{Z}} P(z_i \,|\, x_i, \theta^{(t)}) \, \log[\frac{P(z_i \,|\, \theta) \, P(x_i \,|\, z_i, \theta)}{P(z_i \,|\, x_i, \theta^{(t)})}] \\
&\triangleq Q(\theta, \theta^{(t)}). \qquad (1.6)
\end{aligned}$$

The function $Q(\theta, \theta^{(t)})$ is a lower bound of the incomplete data log-likelihood such that $\mathcal{L}_{\text{inc}}(\theta^{(t)})$ $= Q(\theta^{(t)}, \theta^{(t)})$, and it can be expressed as the sum of two terms,

$$\begin{aligned}
Q(\theta, \theta^{(t)}) &= \sum_{i=1}^{n} \sum_{z_i \in \mathcal{Z}} P(z_i \,|\, x_i, \theta^{(t)}) \, \log[P(z_i \,|\, \theta) \, P(x_i \,|\, z_i, \theta)] \\
&+ \sum_{i=1}^{n} \sum_{z_i \in \mathcal{Z}} P(z_i \,|\, x_i, \theta^{(t)}) \, \log \frac{1}{P(z_i \,|\, x_i, \theta^{(t)})}. \qquad (1.7)
\end{aligned}$$

From (1.4) it should be clear that the first term in the above equation is the expectation of the complete data log-likelihood with respect to the posterior distribution (of the latent variables) $P(z \,|\, x_i, \theta^{(t)}), \ \forall i \in [n]$, and the second term is the entropy of the posterior distribution. Also,

note that the second term is not a function of θ.

As we mentioned earlier, maximizing the complete data log-likelihood has closed form solutions for most parametric distributions from the exponential family because it does not have a summation term inside the logarithm (unlike $\mathcal{L}_{\text{inc}}(\theta^{(t)})$). But the complete data log-likelihood requires knowledge of the latent variables. On the other hand, the expected complete data log-likelihood term in $Q(\theta, \theta^{(t)})$ is an expected quantity which does not require knowledge of the latent variables, and it has the same advantages in terms of maximization because of the absence of the summation term inside the logarithm. Suppose we find updated parameter estimates, $\theta^{(t+1)} = \arg\max_\theta Q(\theta, \theta^{(t)})$, the following inequalities are true:

$$\mathcal{L}_{\text{inc}}(\theta^{(t)}) \;=\; Q(\theta^{(t)}, \theta^{(t)}) \;\leq\; Q(\theta^{(t+1)}, \theta^{(t)}) \;\leq\; Q(\theta^{(t+1)}, \theta^{(t+1)}) \;=\; \mathcal{L}_{\text{inc}}(\theta^{(t+1)}).$$

This constitutes one iteration of the EM algorithm which monotonically increases the incomplete data log-likelihood. Given parameters $\theta^{(t)}$, the E-step calculates the posterior distributions $P(z_i \,|\, x_i, \theta^{(t)}),\ \forall z_i \in \mathcal{Z}, \forall i \in [n]$ which makes the lower bound function equal to the incomplete data log-likelihood. In the M-step the lower bound $Q(\theta, \theta^{(t)})$ is maximized with respect to the parameters, which also necessarily increases the incomplete data log-likelihood. These two steps are repeated until a local maximum of $\mathcal{L}_{\text{inc}}(\theta)$ is found. Note that convergence of the monotonically increasing sequence is guaranteed as long as $\mathcal{L}_{\text{inc}}(\theta)$ has a finite upper bound. The EM algorithm described above is general and can be applied for maximization or minimization problems in the presence of missing/latent variables. Note that the method is only guaranteed to find a local maximum depending on the initialization of the parameters. It is always a good idea to repeat the EM algorithm from a number of different initializations and choose the largest of the local maximum solutions.

There are a number of extensions or variants of the EM algorithms to make it applicable on a larger number of problems. One such is the generalized EM (GEM) [52], [119], [59], which we have used for solving some of our proposed problems. At iteration t, instead of performing a single update of all the parameters from $\theta^{(t)}$ to $\theta^{(t+1)}$ using a maximization step, GEM allows the parameters to be updated sequentially in smaller subsets. To be precise, suppose $\theta = \{\theta_1, \ldots, \theta_L\}$ (for some $L > 1$). Let $\theta^{(t+m/L)},\ \ m = 1, \ldots, L$ denote the partially updated parameter set in going from $\theta^{(t)}$ to $\theta^{(t+1)}$, where m of the L parameter subsets have been updated., *i.e.*, $\theta^{(t+m/L)} = \{\theta_1^{(t+1)}, \ldots, \theta_m^{(t+1)}, \theta_{m+1}^{(t)}, \ldots, \theta_L^{(t)}\}$. Also let $\theta^{(t+L/L)} = \theta^{(t+1)}$. In the M-step of GEM, also known as the generalized M-step, the parameter updates of the individual subsets are monotonically increasing in the lower bound function such that

$$Q(\theta^{(t)}, \theta^{(t)}) \;\leq\; Q(\theta^{(t+1/L)}, \theta^{(t)}) \;\leq\; Q(\theta^{(t+2/L)}, \theta^{(t)}) \;\leq\; \ldots \;\leq\; Q(\theta^{(t+L/L)}, \theta^{(t)}).$$

The individual updates need not perform a maximization but they just need to increase the lower bound, which can be done using methods such as gradient ascent. There are some other extensions of the EM algorithm [145] such as the Expectation-Conditional maximization (ECM) [118], [178] and the Alternating ECM (AECM) [116] which are suitable for certain problems and may help in speeding up convergence. For example, the mixture of factor analyzers (MFA) model [58], [116] can be solved using the ECM and AECM algorithms to achieve faster convergence and simpler update equations [178], [116].

Finally, we note MAP estimation problems can also be solved using the EM algorithm. The only difference is that in MAP estimation the objective consists of the log-likelihood term plus the log-prior term (with prior distribution over the parameters). The log-prior term also gets added to the lower bound $Q(\theta, \theta^{(t)})$ and it only changes the M-step equations.

1.6 Contributions

The main contributions of this dissertation are as follows:

1. Semi-supervised mixture model based classification with fine-grained component conditional class labeling

Typically, semi-supervised classification methods based on the cluster assumption (or the low-density separation assumption), in particular those based on mixture modeling, assume that the class labels of samples originating from a particular cluster or component (a region of high sample density) are likely to be the same, or that they are distributed according to a component specific, but feature vector independent, probability mass function. When this assumption is not satisfied by the data set used for semi-supervised learning, existing generative model-based methods may not have the flexibility to learn a suitably complex class posterior model, while existing discriminative classification methods may not be able to learn accurate class decision boundaries for the problem. We address this limitation by proposing generative semi-supervised mixture models whose underlying stochastic data generation mechanisms allow more fine-grained within-component class label distributions than existing semi-supervised mixture models. The within-component class posterior distributions are inspired by the K-nearest-neighbor and K-nearest-prototype classification methods, which achieve accurate classification in the vicinity of labeled samples.

2. Semi-supervised mixture model based learning from pairwise-sample constraints with imposed space-partitioning

In the semi-supervised learning problem, the partial supervision can also be in the form of class-label constraints on sample pairs (must-links and cannot-links), in which case the objective is to learn a model that fits well to the unlabeled data while satisfying all or a majority of the constraints. Many existing methods which address this problem suffer from the limitation that, while their solutions may closely satisfy the given set of pairwise constraints, they generally do not achieve propagation of the constraint information to all the samples. This can lead to poor generalization and also discontinuities in the predicted class posterior of spatially proximate sample points. We propose a method which effectively addresses this limitation by imposing space-partitioning in the solution to a constrained mixture model learning objective via a parametric mean-field approximation. The proposed method also has the advantage that it does not require knowledge of the number of underlying classes in the data, while this is a requirement for most of the existing methods.

3. Semi-supervised domain adaptation of mixture model based classifiers with imposed space-partitioning

We consider the problem of adapting the parameters of a generative mixture discriminant analysis classifier that has been learned using labeled data from a source domain, to a related target domain where the underlying data distribution may have small differences (relative to the source domain). The target domain consists of a small number of labeled samples and a relatively large number of unlabeled samples available as adaptation data. Motivated by the solution approach adopted for the pairwise-constraint based semi-supervised learning method, here we formulate a learning objective and solution approach for the domain adaptation problem that attempts to make effective utilization of a small number of labeled samples in the target domain. Specifically, in the EM algorithm used to minimize the objective function, we constrain the posterior distribution over latent variables in the E-step to be a smooth parametric function and optimize over these parameters. This ensures that, during adaptation, the components of the class-conditional mixture densities actually have to *move* (in the feature space) in order to minimize the error count on the labeled target domain samples. This in turn ensures that there will be effective *label-propagation* from even a few labeled samples to the unlabeled samples in the target domain classifier solution, as we demonstrate through illustrative examples and experimental results on real data.

The rest of the dissertation is organized as follows. In chapter 2, we present the semi-supervised classification method based on mixture models with fine-grained component conditional class labeling. In chapter 3, we present the semi-supervised learning method from pairwise-sample constraints which imposes space-partitioning in the solution. In chapter 4, we

present the semi-supervised domain adaptation method for mixture model based classifiers. In chapter 5, we conclude with a summary of the dissertation and provide future directions for extending and improving upon this work.

Chapter 2

Semi-supervised mixture model based classification with fine-grained component conditional class labeling

2.1 Introduction

In this chapter, we introduce two related inductive semi-supervised classification methods based on generative mixture modeling, with more fine-grained class label generation mechanisms compared to previous works. Earlier works on semi-supervised learning based on mixture models frequently assume that points from a cluster (or mixture component) are likely to have the same class label. This is consistent with the cluster assumption for semi-supervised learning discussed in chapter 1. A model with a less restrictive assumption has also been proposed earlier, which assumes that the probability distribution of the class labels within a component or cluster can be well represented by a fixed, feature space independent probability mass function. This model is well-suited for problems which are consistent with the cluster assumption (class-pure clusters), and also for problems where the underlying clusters are not fully class-pure but have probabilistic association with two or more classes (perhaps due to cluster overlap). However, both this model and models which assume class-pure clusters will *not* be suitable for modeling data where the within-cluster class distribution is a non-trivial function of the feature vector. Our fine-grained labeling models address this important limitation by combining the advantages of semi-supervised mixtures, which achieve label extrapolation over a component, and nearest-

neighbor (NN)/nearest-prototype (NP) classification, which achieve accurate classification in the vicinity of labeled samples.

For our first "NN-based" fine-grained model, we propose a novel two-stage stochastic data generation mechanism, with the feature vectors (and implicitly their mixture components of origin) of all samples (labeled and unlabeled) first generated independently according to a standard finite mixture density, and then the class labels of labeled samples jointly generated conditioned on the feature vectors and their components of origin. This mechanism entails an underlying Markov random field over the class labels of labeled samples generated by each mixture component, given that the feature vectors and the underlying mixture components of all samples have been observed. As we will see, for this model, the stochastic data generation specifies a non-parametric component-conditional class posterior, whose complexity is automatically determined by the number and variety of class labels in the labeled data subset. Hence we refer to this method as NFGL (non-parametric fine-grained labeling). We invoke the pseudo-likelihood formulation (commonly applied to make MRF based model learning tractable), which forms the basis for an approximate generalized Expectation-Maximization model learning (parameter estimation) algorithm. Our second, "NP-based" fine-grained model is a parametric counterpart of NFGL, which we refer to as PFGL (parametric fine-grained labeling). Importantly, it overcomes a problem with the NFGL model that sometimes manifests when there are very few sparsely distributed labeled samples. The PFGL model is motivated by a simpler, independent stochastic data generation mechanism, which does not entail a Markov random field over the class labels of the labeled samples. Hence, for PFGL, it is possible to directly maximize the data log-likelihood in order to estimate its model parameters, unlike NFGL which maximizes the pseudo-log-likelihood approximation. Also, different from the NFGL model, the complexity of the within-component class posteriors in the PFGL model are not automatically determined by the labeled data subset, but they need to be carefully specified using model order selection. Both our models are advantageous when the distribution of classes is not constant over the feature space region "owned by" a component or cluster of points.

Recall that there are two common objectives for semi-supervised learning. In the first and perhaps more common objective, a small labeled training set is augmented with a large set of unlabeled samples, with the aim of learning a more accurate statistical classifier than that learned solely using the labeled training set. In the second, less common objective, an unlabeled data set may be augmented by a small set of labeled samples, with the objective to more accurately cluster the data or model the data density, *i.e.*, semi-supervised extension of the unsupervised learning problem. In this work, we primarily focus on the former (statistical classification) objective.

As discussed in chapter 1, semi-supervised learning approaches can be broadly categorized

into generative and discriminative learning frameworks. There has been a substantial amount of work in discriminative semi-supervised learning, such as co-training [19], transductive support vector machines [84], [32], Gaussian process approaches [100], information regularization [161], [41], and a number of graph-based methods [10], [186]. On the other hand, generative, mixture-based models [122], [135], [154] represent some of the earliest approaches to semi-supervised learning. There have also been some recent works in this area which propose different learning objectives to estimate the mixture model parameters [73], [74]. The mixture-based semi-supervised learning models fundamentally vary in the assumptions they make about the connection between the mixture components and the classes. For instance, they may assume that samples from each class are well-modeled by one or more mixture components [135], [73], [74], or they may assume that the samples from each component are probabilistically associated with more than one class [122]. The methods [122], [135], and [154] maximize data log-likelihood based learning objectives for parameter estimation, while the methods [73] and [74] maximize discriminative classification objectives such as the class posterior log-likelihood of the labeled data regularized by the feature vector marginal log-likelihood of the unlabeled data [73], or the class posterior log-likelihood of the labeled data regularized by the conditional entropy of the class given the feature vector [74].

The generative mixture-based semi-supervised learning approaches are suitable only when the feature data is well-modeled by a finite mixture of the assumed parametric density form. If the mixture model assumption is not suitable or mis-matched to the data, semi-supervised mixtures may degrade the classification performance [43], [44], [38]. However, as we discussed in chapter 1, all semi-supervised methods make some underlying assumptions about the distribution of the data, and they may not be suitable when the assumptions are not satisfied by the data. For example, as noted in the review article [182], co-training requires a (natural) partitioning of the feature space into two subsets; transductive SVMs and the works in [100] and [35] place the decision boundary in regions of low data density. However, this may not be the proper choice if there is significant overlap between the true class-conditional densities. Many graph-based methods assume the data follows an underlying manifold structure and they also require transductive inference [182], which may be of high complexity.

Moreover, mixture models, *e.g.* Gaussian mixtures, are ubiquitous, are appropriate, and are used very effectively in a variety of application domains –modeling speech [165], [143], image content [159], [141], in bioinformatics [117], and in modeling many types of scientific data, *e.g.* [64], [57]. Mixtures of other distributions, *e.g.* the multinomial, naturally model text documents [135]. Accordingly, generative, mixture-based semi-supervised learning is highly suitable for a large number of real-world application domains. Likewise, *improvements* to ex-

isting semi-supervised mixtures should find significant application. Thus, in this work we focus on generative semi-supervised mixtures and in particular on improving the model for class label generation within semi-supervised mixtures. Previewing our framework, we will cast model learning as a *maximum likelihood* problem, with model parameters treated as deterministic (but whose values are unknown), rather than as a Bayesian learning/inference problem (which hypothesizes stochastic generation of the parameters as well as the data) [51].

The rest of the chapter is organized as follows. In section 2.2, we review generative semi-supervised learning methods, focusing particularly on semi-supervised mixture modeling [122], [135], [154] and on the limitations of these past models. We identify that the class label generation mechanism in these models is crude and may introduce severe model bias. In previous work [38], it was demonstrated that, if statistical modeling assumptions are incorrect, semi-supervised learning may in fact *degrade* classification performance relative to supervised learning, which solely makes use of the labeled data. We put forward the crude label generation mechanisms in[122], [135], and [154] as, at any rate, one of the main sources of modeling error that may lead to poor performance. We also illustrate that when the goal is data clustering or density estimation, the use of labeled data in [122], [135], and [154] may bias the learned solution away from the ground-truth mixture solution. In sections 2.3 and 2.4, we develop our new generative semi-supervised approaches, which achieve more fine-grained class probability modeling within each mixture component/cluster than previous works. In section 2.6, we discuss the relationship to previous work. In section 2.7, we present experimental comparisons against alternative semi-supervised and supervised learning methods. The results will show that our models' fine-grained class modeling yields significantly better classification accuracy than the previous semi-supervised mixtures [122], [135], and [154]. Moreover, when the number of labeled samples is quite small, our models also achieve overall better classification accuracy than supervised linear and nonlinear kernel support vector machines.

2.2 Limitations of Previous semi-supervised Mixtures

2.2.1 Notation

Consider a random feature vector $\underline{X} \in \mathbb{R}^d$, a random class label $C \in \mathcal{C} \equiv \{1, \ldots, N_c\}$, and a random component label $M \in \mathcal{M} \equiv \{1, \ldots, L\}$. Let $\{P[M = l] \equiv \alpha_l,\ l = 1, \ldots, L\}$, satisfying $0 \leq \alpha_l \leq 1$, $\sum_{l=1}^{L} \alpha_l = 1$, be the masses (proportions) for an L-component mixture model [112] with component densities $f_{\underline{X}\,|\,l}(\underline{x}\,|\,\theta_l), l = 1, \ldots, L$, where θ_l is the parameter set for the l-th component density. The mixture density for $\underline{X}$ is $f_{\underline{X}}(\underline{x}) = \sum_{l=1}^{L} \alpha_l\, f_{\underline{X}\,|\,l}(\underline{x}\,|\,\theta_l)$. In semi-supervised mixture modeling, one jointly models the feature vector $\underline{X}$ and the class label

C, *i.e.* one learns the joint density $f_{\underline{X},C}(\underline{x}, c) = \sum_{l=1}^{L} \alpha_l \; f_{\underline{X},C\,|\,l}(\underline{x}, c\,|\,\theta_l)$, based on a pool of both labeled and unlabeled data samples. For future reference, we denote the unlabeled data subset by $\mathcal{X}_u = \{\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_{N_u}\}$, $\underline{x}_i \in \mathbb{R}^d$; the labeled subset by $\mathcal{X}_l = \{(\underline{x}_{N_u+1}, c_{N_u+1}), (\underline{x}_{N_u+2}, c_{N_u+2}), \ldots, (\underline{x}_{N_u+N_l}, c_{N_u+N_l})\}$, $\underline{x}_i \in \mathbb{R}^d$, $c_i \in \mathcal{C}$; the set of class labels considered by themselves, $\mathcal{C}_l = \{c_{N_u+1}, \ldots, c_{N_u+N_l}\}$; the set of all feature vectors considered by themselves $\overline{\mathcal{X}} = \{\underline{x}_1, \ldots, \underline{x}_{N_u}, \underline{x}_{N_u+1}, \ldots, \underline{x}_{N_u+N_l}\}$; and the *complete* pooled data set by $\mathcal{X} = \{\mathcal{X}_u, \mathcal{X}_l\}$. We also denote the index set of the labeled samples by $\mathcal{I}_l = \{N_u + 1, \ldots, N_u + N_l\}$, and the set of first n positive integers by $[n] \triangleq \{1, 2, \ldots, n\}$.

2.2.2 Hard Versus Soft Class-to-Component Assignments

Early work on semi-supervised mixture modeling includes [122], [135], and [154]. There has also been a significant amount of followup interest in this area, *e.g.* [36], [87], [95], [149], [153], and [155]. There is also related work on semi-supervised clustering [167], [7], and semi-supervised mixture modeling [155], [97], [179] where rather than labels, the supervision takes the form of class "must-link" and "cannot-link" constraints. Most of these approaches are similar to [135] and [154] in that they essentially make a "one-class-per-component" assumption, *i.e.* all clusters or mixture components are each assigned to a single class, with all data samples within the cluster/component assumed to belong to this (assigned) class. The model in [122] differs from these approaches in not making this assumption. We next first briefly summarize [122] and how it differs from [135] and [154]. We then identify two fundamental limitations common to all three of these methods, which will motivate development of our new semi-supervised mixtures in sections 2.3 and 2.4.

In [122], the following stochastic data generation was assumed for the pooled data $\mathcal{X}$: 1) Independently, for each $\underline{x}_i \in \mathcal{X}_u$:
i) randomly select a mixture component, $M = j$, according to $\{\alpha_l\}$; ii) randomly generate $\underline{x}_i$ according to $f_{\underline{X}\,|\,j}(\underline{x}\,|\,\theta_j)$.
2) Independently, for each $(\underline{x}_i, c_i) \in \mathcal{X}_l$:
i) randomly select a mixture component, $M = j$, according to $\{\alpha_l\}$; ii) randomly generate $\underline{x}_i$ according to $f_{\underline{X}\,|\,j}(\underline{x}\,|\,\theta_j)$. iii) randomly select the class label c_i according to the component-conditional (multinomial) probability mass function (pmf) $\{P(C = c\,|\,M = j) \equiv \beta_{c\,|\,j}\}$.

Thus, it was assumed class labels are stochastically generated, conditioned on the sample's component of origin (M) and that $\underline{X}$ and C are conditionally independent, given M. In this case, the joint density is $f_{\underline{X},C}(\underline{x}, c) = \sum_{l=1}^{L} \alpha_l \; f_{\underline{X}\,|\,l}(\underline{x}\,|\,\theta_l) \; \beta_{c\,|\,l}$ and the associated class *a posteriori*

probabilities, used for class decision making, take the "mixture of experts" (MOE) form:

$$P(C = c \,|\, \underline{x}) \;=\; \sum_{l=1}^{L} \beta_{c\,|\,l}\; P(M = l \,|\, \underline{x}) = \sum_{l=1}^{L} \beta_{c\,|\,l}\; \frac{\alpha_l\; f_{\underline{X}\,|\,l}(\underline{x} \,|\, \theta_l)}{\sum_{m=1}^{L} \alpha_m\; f_{\underline{X}\,|\,m}(\underline{x} \,|\, \theta_m)}. \tag{2.1}$$

The model parameters $\Theta = \{\theta_l, \alpha_l, \{\beta_{c\,|\,l},\ \forall c \in \mathcal{C}\},\ \forall l \in [L]\}$ were estimated by maximizing the joint data likelihood over $\mathcal{X}$,

$$P(\mathcal{X} \,|\, \Theta) \;=\; P(\mathcal{X}_u, \mathcal{X}_l \,|\, \Theta) = \left(\prod_{i=1}^{N_u} \sum_{l=1}^{L} \alpha_l\; f_{\underline{X}\,|\,l}(\underline{x}_i \,|\, \theta_l)\right) \left(\prod_{i=N_u+1}^{N_u+N_l} \sum_{l=1}^{L} \alpha_l\; f_{\underline{X}\,|\,l}(\underline{x}_i \,|\, \theta_l)\; \beta_{c_i\,|\,l}\right). \tag{2.2}$$

This was achieved in a locally optimal fashion via an Expectation-Maximization (EM) algorithm [48], [111], [119] developed in [122].

The methods in [135] and [154] are closely related to [122]. Like [122], [135] learned the mixture model to maximize the joint data likelihood over $\mathcal{X}$ (whereas [154]) maximized the class-conditional likelihood). Also like [122], [135] developed an EM algorithm for parameter learning. However, a distinction between [122], and [135], [154] lies in how these methods hypothesize class label generation. Unlike [122], in [135], [154] it was assumed each mixture component is exclusively assigned to a single class, *i.e.* it only generates labeled samples from this (assigned) class. Thus, in [135] and [154], once the component of origin for a sample is selected, its class label is also determined. More specifically, let $z_{k\,|\,l} \in \{0, 1\}$ be a binary parameter with $z_{k\,|\,l} = 1$ indicating that component l is assigned to class k. These parameters must be chosen to satisfy $\sum_{k \in \mathcal{C}} z_{k\,|\,l} \;=\; 1, \quad \forall l$ (each component is assigned to a single class). It is also highly desirable to satisfy $\sum_{l=1}^{L} z_{k\,|\,l} \geq 1 \;\; \forall k \in \mathcal{C}$ (every class is assigned at least one component); otherwise, the classifier will completely misclassify unrepresented classes. Clearly, $\{z_{k\,|\,l}\}$ is a special, restricted case of the multinomial pmf $\{\beta_{k\,|\,l}\}$. Accordingly, the joint mixture density $f_{\underline{X},C}(\underline{x}, c) = \sum_{l=1}^{L} \alpha_l\; f_{\underline{X}\,|\,l}(\underline{x} \,|\, \theta_l)\; z_{c\,|\,l}$, the associated class posterior probability $P(C = c \,|\, \underline{x}) = \frac{f_{\underline{X},C}(\underline{x},c)}{f_{\underline{X}}(\underline{x})}$, the joint data likelihood $P(\mathcal{X} \,|\, \Theta)$, and the associated EM algorithm from [135] are specializations of [122], achieved by constraining $\beta_{c\,|\,l} \in \{0, 1\}$[1].

Allowing *stochastic* (soft) label generation, given the mixture component of origin, *i.e.*

[1] Actually, in [135] the authors did not re-estimate $\{z_{k\,|\,l}\}$ within their EM algorithm – component assignments to classes were simply initialized and then left unchanged during the EM optimization. Presumably, optimizing the $\{z_{k\,|\,l}\}$ should yield more accurate models. One cannot form a *batch* (in parallel) M-step update of all the $\{z_{k\,|\,l}\}$ parameters since this will not ensure $\sum_l z_{k\,|\,l} \geq 1 \;\; \forall k$. One can, however, apply a *generalized* M-step [52], [119], [59], optimizing $\{z_{k\,|\,l} \;\; \forall k\}$ for a given l, consistent with the constraint $\sum_m z_{k\,|\,m} \geq 1 \;\; \forall k$, while holding $\{z_{k\,|\,l'} \;\; \forall k\}$ fixed for all $l' \neq l$. One can in this way cyclically optimize over each component's class assignments, given all other assignments held fixed, until there are no further improvements.

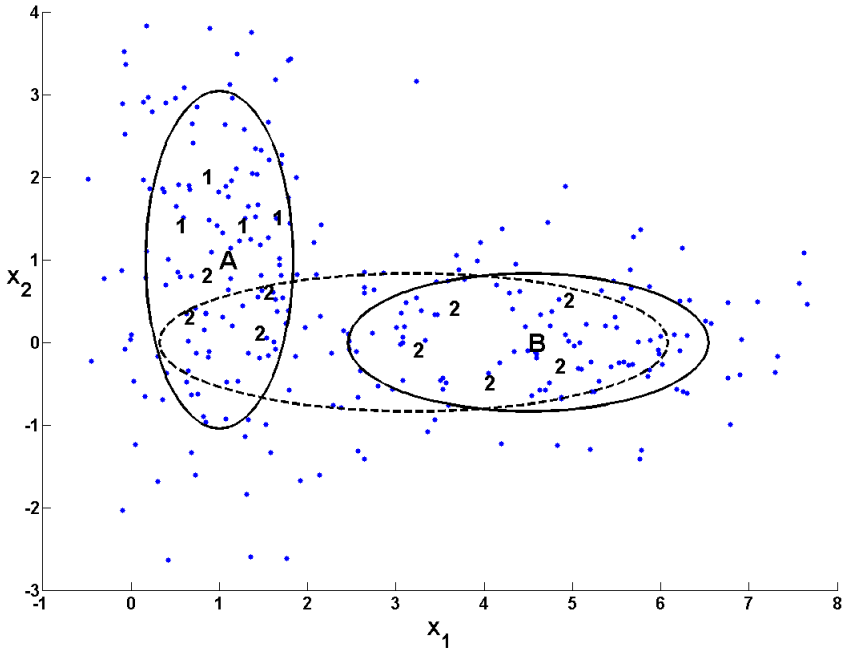


Figure 2.1. A 2-D mixture example with two components, A and B, one of which (A) generates labeled samples from two different classes. Also shown (dashed contour) is an (incorrect) learned component that exclusively generates samples from class 2.

$\beta_{k\,|\,l} \in [0,1]$, $\sum_{k\in\mathcal{C}} \beta_{k\,|\,l} = 1$, as in [122], is advantageous in some cases because the assumption of exclusive component-to-class assignments in [135] and [154] may be too inflexible to accurately model the given semi-supervised data and capture its underlying cluster structure. Consider the illustrative example shown in Fig. 2.1. Here, we depict a 2-D semi-supervised data set with two classes '1' and '2', a few labeled samples, and with the ground-truth components A and B indicated by solid ellipses[2]. It is clear from the figure that while component B does generate labeled samples exclusively from class 2, component A generates samples from both classes. The models in [135] and [154] cannot accurately learn the ground-truth mixture in this case – one of the two learned components must capture all the labeled points from class 2, and the class 2 "outliers" (those from component A) would degrade parameter estimates for this learned component. A solution based on [135] is depicted via a dashed ellipse for the learned component that generates labeled samples from class 2. The model in [122], on the other hand, will have no difficulty learning the ground-truth mixture components in this example (learned component 1, corresponding to true component A, will estimate $\beta_{1\,|\,1} = \beta_{2\,|\,1} = \frac{1}{2}$ to accurately

[2]Labeled samples are indicated by the symbols '1' or '2', with the labeled sample's location indicated by the symbol's location in the figure. Unlabeled samples are simply indicated by points.

reflect the proportion of labeled samples from each class within component 1 (4 labeled samples from each class)).

In order to understand whether mixture models fit to real world data sets form class-pure clusters, or whether they have within-component class impurities, we evaluated the following measure of *average class purity* [3] over the components of a mixture model. Consider a fully labeled data set $\{(\underline{x}_n, c_n) \,:\, n = 1, \ldots, N\}$ with feature vectors $\underline{x}_n \in \mathbb{R}^d$ and class labels $c_n \in \mathcal{C}$. Let L denote the number of components (clusters) in a mixture model fit to the data set (in an unsupervised fashion), and $P(M_n = j \,|\, \underline{x}_n, \Theta),\ \forall j \in [L], \forall n \in [N]$ denote the component posterior probabilities of the samples based on the estimated mixture model parameters Θ. To calculate the average class-purity we utilize the true class labels of all the points in the data set. The probabilistic count of the number of samples with class labels i that are assigned to component j is given by

$$n_{ij} \;=\; \sum_{n=1:c_n=i}^{N} P(M_n = j \,|\, \underline{x}_n, \Theta), \quad \forall i \in \mathcal{C},\ \forall j \in [L].$$

The probabilistic count of the total number of samples assigned to component j is given by $n_j \;=\; \sum_{i \in \mathcal{C}} n_{ij},\ \ \forall j \in [L]$. The average class purity of the components of a mixture model is then defined as

$$P \;=\; \sum_{j=1}^{L} \frac{n_j}{N} \max_{i \in \mathcal{C}} \frac{n_{ij}}{n_j} \;=\; \frac{1}{N} \sum_{j=1}^{L} \max_{i \in \mathcal{C}} n_{ij}. \tag{2.3}$$

Intuitively, this class purity measure calculates the weighted sum of the proportion of samples from the majority-class within each component, where the weight for each component term is the proportion of the total number of samples (probabilistically) assigned to that component. Therefore, if the mixture components capture class-pure clusters, then P will have a value close to 1. We evaluated this class purity measure on three data sets from the UC Irvine machine learning repository [1], namely (i) Wall-following robot navigation which has 4 classes, (ii) Balance scale which has 3 classes, and (iii) Iris which has 3 classes. For each of the these data sets, we estimated Gaussian mixture models with the number of components increased starting from the number of classes in the data set. For a fixed number of mixture components, the parameters of the mixture model were initialized using the K-means clustering algorithm, following which the EM algorithm was applied to find a local maximum of the log-likelihood. This was repeated from ten random initializations based on K-means clustering, and the solution with largest data log-likelihood was chosen. Based on the chosen mixture model solution, the

[3] This measure of class purity is used in the definition of the F-score, which is used as a measure of clustering performance (*e.g.*, see [155] and [179]).

average class purity score (2.3) was calculated and plotted as a function of the number of mixture components. This is shown in Fig. 2.2 for the three UC Irvine data sets. We observe that for two of the data sets P has values smaller than 0.75, which suggests that the mixture components are not likely to be class-pure. We would expect that the method of [122], which allows components

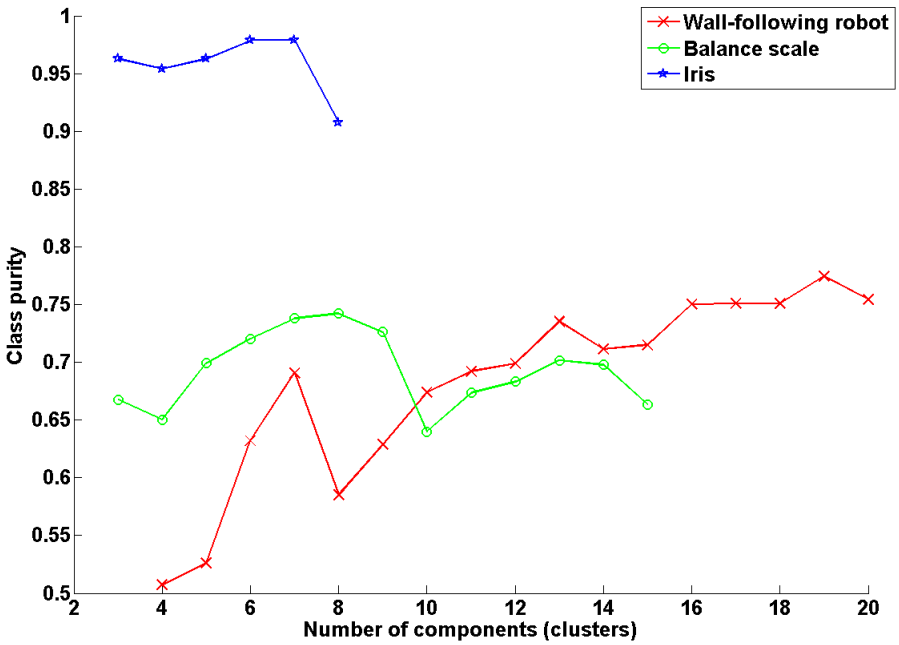


Figure 2.2. Plots of the average class purity as a function of the number of components in the mixture model for three data sets from the UC Irvine machine learning repository.

to generate samples from multiple classes, is more suitable than [135] and [154] for learning the underlying class structure within components from semi-supervised data sets such as this, where the underlying clusters in the data may consist of samples from more than one class. However, while [122] gives *more* flexible within-component class label generation than [135] and [154], this degree of flexibility is still insufficient, as we next elaborate by highlighting fundamental limitations that [122], [135], and [154] all share in seeking to build a statistical classifier based on semi-supervised data when the "one class per cluster" assumption is violated.

2.2.3 Crude Within-Component Class Labeling

First, again consider the data in Fig. 2.1. Although [122] can accurately capture the ground-truth mixture components, this model will not be an effective *classifier* for data originating from component A – as noted earlier $\beta_{1\,|\,1} = \beta_{2\,|\,1} = \frac{1}{2}$, and thus for $\underline{x}$ originating from component

A, [122] will estimate $P(C = c \,|\, \underline{x}) = \frac{1}{2},\ \ c = 1, 2$ even though it would be far better to choose $P(C = 1 \,|\, x_1, x_2 > 1) = 1$ and $P(C = 2 \,|\, x_1, x_2 < 1) = 1$ for $\underline{x}$ from component A, *i.e.* more fine-grained *within-component* class labeling is needed to give accurate classification in this example.

There are several ways this can be achieved in principle. First, as suggested in [135], one can simply introduce more components into the model, splitting components with high class-impurity into multiple smaller components that do exhibit class purity. This can certainly improve classification accuracy. However, there are two potential problems with this remedy. First, in the Fig. 2.1 example, there really are two Gaussian components – it is only the class label, not the feature vector, which requires more complex modeling; *i.e.*, by introducing (many) extra components, the solution may no longer capture the natural clustering structure present in the data. Second, this solution may also give sub-optimal classification accuracy, especially if the within-cluster class structure is complicated. Consider the example in Fig. 2.3. Here, it is somewhat reasonable to split natural cluster A into two class-pure Gaussian components. However, such a split will still be suboptimal for classification, since the class 2 distribution (eyebrow-shaped) within cluster A does not well-conform to an elliptical (bivariate Gaussian) shape. The situation may be even worse in natural cluster B. One can split this cluster into two components, one assigned to class 1 and the other to class 2. However, the "outlier" labeled sample from class 1 at the top of the cluster must be owned by the (class-pure) class-1 component – this sample will corrupt parameter estimates for the class-1 component, resulting in a sub-optimal class decision boundary within cluster B. In our experimental results, we will explore the classification benefit achieved simply by using a large number of components (larger than the estimated number of natural clusters) in the model.

A second strategy for improving classification accuracy involves retaining the modeling of the natural cluster structure in the data, but with *fine-grained*, "discriminative" class modeling *within* each natural cluster/component. A crude, heuristic method for achieving this fine-graining, which we dub "within-component NN" (WC-NN), is as follows: 1) perform unsupervised clustering/mixture modeling, ignoring the available class labels; 2) make exclusive (0-1) assignments of each labeled sample to its best-fitting component; 3) for each component, directly form a *nearest-neighbor classifier* [51] based simply on the labeled samples assigned to that component. An (accurate) classifier obtained in this way is depicted via an illustrative example in Fig. 2.4.

In summary, from the above examples, we anticipate that a semi-supervised mixture model with more fine-grained ("discriminative") within-component class modeling may give improved classification accuracy, compared with previous semi-supervised mixtures. Moreover, we em-

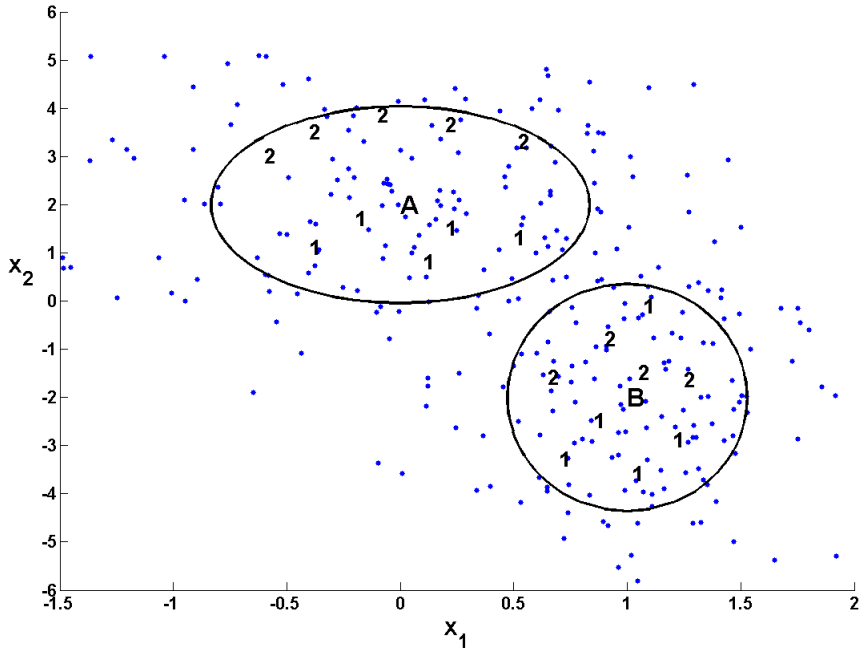


Figure 2.3. An illustrative example with two ground-truth components, each of which generates labeled samples from two classes. Assuming a two-component model, the method from [135], with exclusive class-to-component assignments, will fail to learn an accurate classifier on this example. Moreover, splitting each natural component into two "class-pure" components will also give suboptimal classification accuracy in this example.

phasize that while the examples shown in this section (excepting Fig. 2.2) were synthetically generated and thus merely illustrative, the practical benefits of fine-grained labeling will be demonstrated on real-world data sets in our experimental results.

In sections 2.3 and 2.4, we will show that fine-grained labeling can be achieved in a much less heuristic fashion (and, as shown in our experimental results, with better classification accuracy) than WC-NN, via new generative semi-supervised mixtures with more involved stochastic data generation than previous proposals. The model in section 2.3 performs fine-grained class modeling in a *non-parametric* fashion, with the resulting model based on Markov random fields (MRFs) [103], and with the model learning based on maximization of the pseudo-likelihood function, associated with MRFs [103]. The method in section 2.4, alternatively, takes a *parametric* approach to fine-grained class modeling, and is specifically designed to overcome a problem experienced by the non-parametric method at very low labeled fractions. Moreover, the model learning maximizes a true likelihood function, rather than the pseudo-likelihood considered in section 2.3. On the other hand, unlike the method in section 2.3, the parametric approach may require the integration of a model order selection procedure within the model learning, in order

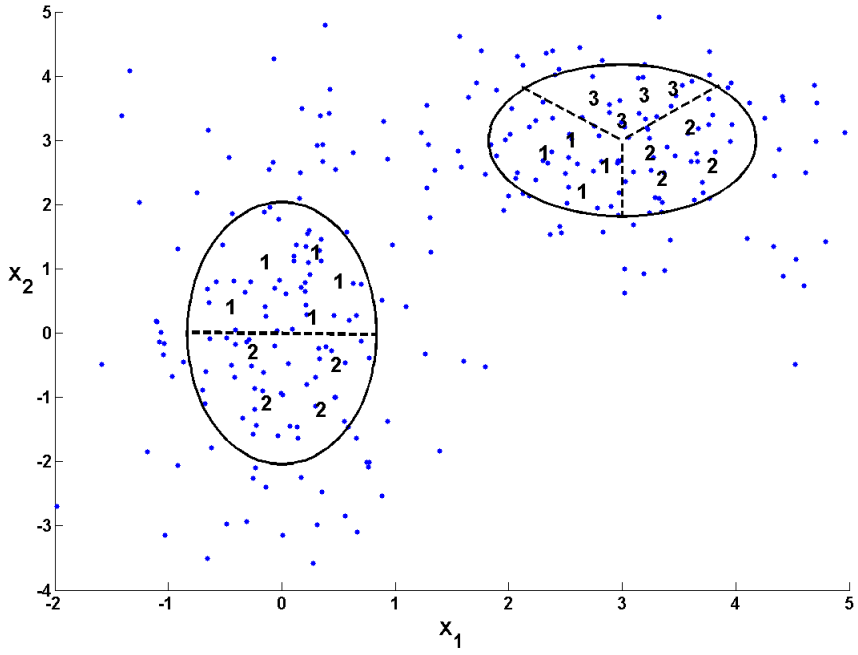


Figure 2.4. Nearest neighbor classification applied within each (learned) cluster achieves accurate within-cluster classification in this 2-D example.

to introduce the "right" level of class modeling complexity, within each mixture component. Both of our fine-grained approaches model the joint distribution $P(\underline{X}, C)$ and, in this sense, strictly speaking, are purely generative. However, since these models give more detailed within-component class posterior modeling than previously proposed semi-supervised mixtures, one might also reasonably describe them as hybrid "generative/discriminative".

In section 2.7, we will demonstrate on a number of real-world UC Irvine data set domains that both of these new models yield more accurate classifiers than both previous semi-supervised mixtures and WC-NN. They also give better overall accuracy than supervised classifiers (support vector machines) when the fraction of labeled training data is sufficiently low.

2.3 A Non-Parametric Fine-Grained Labeling Model (NFGL)

2.3.1 Stochastic Data Generation for the New Model

As in [122], we treat class labels as data and model their stochastic generation, along with the feature data. Different from previous methods, we hypothesize generation of the data $\mathcal{X}$ in two (wholly distinct) stages. In the first stage, the feature vectors for *all* samples, both unla-

beled and labeled ($\overline{\mathcal{X}}$), are generated, i.i.d., according to a standard finite mixture model density, $f_{\underline{X}}(\underline{x}) = \sum_{l=1}^{L} \alpha_l \; f_{\underline{X}\,|\,l}(\underline{x}\,|\,\theta_l)$. Then, the class labels $\mathcal{C}_l$ are generated in a non-independent fashion from their joint distribution, conditioned on the knowledge of the feature vectors, $\overline{\mathcal{X}}$, and on the knowledge of the component of origin of the feature vectors, $\mathcal{V} \triangleq \{V_{j\,|\,i},\ \forall j \in [L], \forall i \in [N_u + N_l]\}$, where $V_{j\,|\,i} \in \{0, 1\}$ and $\sum_{j=1}^{L} V_{j\,|\,i} = 1,\ \forall i$.

In our approach, the class labels in $\mathcal{C}_l$ are generated according to a non-trivial within-component class posterior model, $P(C = c\,|\,V_{j\,|\,i} = 1, \underline{x})$. This is what makes our approach "fine-grained". The modeling framework we develop in this section is *in principle* amenable to many different choices for this fine-grained posterior. For example, we could model the class posterior, conditioned on $\underline{x}$ belonging to a given component, as a logistic function (*i.e.*, a randomized (soft) linear discriminant function), a nonlinear (*e.g.* kernelized) logistic function (*i.e.* a randomized *generalized* linear discriminant function), or even a complex parametric model such as a multilayer perceptron (MLP) with the MLP's outputs (one per class) normalized to produce a valid posterior (pmf). However, each of these choices entails drawbacks pertaining to finding the "right" level of description in modeling the class labels. The logistic function may introduce model bias, *e.g.* a single linear discriminant function does not have sufficient power to discriminate the three classes present in one cluster in the Fig. 2.4 example. Likewise, a kernelized logistic function may increase the model variance, giving an overly complex within-component class model for a component in which a linear decision boundary would suffice (the other cluster in the Fig. 2.4 example). For the MLP, careful model order selection would be required to customize the number of hidden units used by each within-component class posterior.

Alternatively, in this section, we propose a posterior which *automatically* sets the complexity of the class modeling within any given cluster, based simply on the number (and class variety) of the labeled samples that fall within the cluster. Specifically, we propose to use a *randomized KNN* within-component class posterior. To develop this model, let us suppose that the first stage of data generation has already been applied, generating $\overline{\mathcal{X}}$ and determining a mixture component of origin for each sample, $\mathcal{V}$. Let $\mathcal{V}_l$ denote the set of mixture components of the labeled samples and $\mathcal{V}_u$ denote the set of mixture components of the unlabeled samples, such that $\mathcal{V}_l \cup \mathcal{V}_u = \mathcal{V}$. Further, for the moment, suppose that we have *already* generated the class labels for $N_l - 1$ of the N_l labeled samples, *i.e.* it is only left to generate the label for the remaining sample, i^*. Consider the set of labeled samples originating from the same mixture component as i^*, *i.e.*, $\{i \in \mathcal{I}_l \backslash \{i^*\} : V_{j\,|\,i} = V_{j\,|\,i^*}, \forall j \in [L]\}$, of size $N_{i^*}(\mathcal{V}_l)$. These labeled samples could be used to form a hard, within-component, K-nearest-neighbor classifier (with $K = N_{i^*}(\mathcal{V}_l)$), similar to the method discussed in section 2.2. Alternatively, we define a *randomized* version of this

classifier:

$$
\begin{aligned}
& P(C_{i^*} = c \,|\, \{c_i : i \in \mathcal{I}_l \backslash \{i^*\}\}, \overline{\mathcal{X}}, \mathcal{V}_l) \\
= \quad & \prod_{j=1}^{L} \left[\, \delta\left(M_{jc}(i^*) > 0\right) \; R_{j,i^*}(c \,|\, \mathcal{C}_l, \overline{\mathcal{X}}, \mathcal{V}_l) \; + \; \delta\left(M_{jc}(i^*) = 0\right) \; \beta_{c\,|\,j} \, \right]^{V_{j\,|\,i^*}}, \quad (2.4)
\end{aligned}
$$

where $R_{j,i^*}(c \,|\, \mathcal{C}_l, \overline{\mathcal{X}}, \mathcal{V}_l)$ is the randomized nearest neighbor (RNN) class posterior for component j, defined as

$$
R_{j,i^*}(c \,|\, \mathcal{C}_l, \overline{\mathcal{X}}, \mathcal{V}_l) \; = \; \frac{\sum\limits_{\substack{i=N_u+1: \\ c_i=c,\, i \neq i^*}}^{N_u+N_l} V_{j\,|\,i} \; e^{-a_j \, \|\underline{x}_i - \underline{x}_{i^*}\|^2}}{\sum\limits_{\substack{i=N_u+1: \\ i \neq i^*}}^{N_u+N_l} V_{j\,|\,i} \; e^{-a_j \, \|\underline{x}_i - \underline{x}_{i^*}\|^2}}, \quad \forall c \in \mathcal{C}, \qquad (2.5)
$$

$M_{jc}(i^*)$ is the number of labeled samples (excluding i^*) generated from component j that have class label c, $\delta(a)$ is the binary indicator which takes a value $1(0)$ when condition a is satisfied (not satisfied), and $\beta_{c\,|\,j}, \;\; \forall c \in \mathcal{C}, \;\; \forall j \in [L]$ are fixed component conditional class probabilities as defined by the MOE model [122]. Note that $R_{j,i^*}(c \,|\, \mathcal{C}_l, \overline{\mathcal{X}}, \mathcal{V}_l)$ is 0 or undefined when there are very small number of labeled samples generated by a mixture component. For example, when i^* is the only labeled sample from component j with class label c, or when there are no labeled samples from component j with class label c, or worse when there are no labeled samples at all from component j. In order to handle such degenerate cases, we include the component conditional class probabilities estimated by the MOE model, which are well defined and sum to 1 even when $M_{jc}(i^*) = 0$. Note that we do not consider $\beta_{c\,|\,j}, \;\; \forall c \in \mathcal{C}, \;\; \forall j \in [L]$ to be part of the parameter set of our model, but use their maximum likelihood estimates found using the EM algorithm in [122]. However, the significant term in (2.4) is the RNN class posterior, which comes into play most of the time when the number of labeled samples is not very small.

From (2.5) we see that C_{i^*} is in fact independent of the class labels of all labeled samples that do not originate from component j, conditioned on the class labels of all other labeled samples that do originate from component j. Here, a_j is a component-specific scale parameter. As $a_j \rightarrow \infty, \;\; R_{j,i^*}(c \,|\, \mathcal{C}_l, \overline{\mathcal{X}}, \mathcal{V}_l) \rightarrow \{0, 1\}$, consistent with a hard K-nearest-neighbor rule applied within component j (where $K = N_{i^*}(\mathcal{V}_l)$). Note that the complexity of this within-component posterior (decision rule) is not determined by how many parameters we (*e.g.*, arbitrarily) decide to include in the model – it is *automatically* determined by the number of labeled samples (and the class variety of same) falling within the component. If there are few labeled samples within a

component, there is no basis for a complex decision boundary in this component and, consistent with this, the RNN rule is automatically very simple. If there are many labeled samples in a given component, the RNN rule will be more complex. This is automatically determined/controlled, based on the amount of available labeled data, rather than requiring careful model selection to control the model complexity. Moreover, the posterior (2.4) introduces very few additional parameters – just one scaling parameter a_j for each component. Different soft nearest neighbor rules, achieved *e.g.* by replacing the (implicit) kernel choice in (2.4), $k(\underline{x}, \underline{y}) = e^{-a_j \|\underline{x}-\underline{y}\|^2}$ by specialized data-dependent kernels can also be considered, especially when some or all of the features in the data take on categorical (discrete) values.

As noted above, a key point about (2.4) is that it specifies that the probability for any label C_i, conditioned on $\{C_m : m \in \mathcal{I}_l \backslash \{i\}\}$ equals the probability of C_i conditioned on the labels for the *subset* of labeled samples that originate from the same mixture component as i. There are two implications of this. The first is that the joint label distribution $P(\mathcal{C}_l \,|\, \overline{\mathcal{X}}, \mathcal{V}_l)$ factorizes as a product of distributions over mutually exclusive label subsets, one per mixture component. Specifically, let $\mathcal{C}_l^{(j)}$ denote the subset of labels of samples originating from component j, such that $\mathcal{C}_l = \cup_{j=1}^{L} \mathcal{C}_l^{(j)}$ and $\mathcal{C}_l^{(j)} \cap \mathcal{C}_l^{(k)} = \emptyset,\ \forall j, k \in [L]$. Then we have $P(\mathcal{C}_l \,|\, \overline{\mathcal{X}}, \mathcal{V}_l) = \prod_{j=1}^{L} P(\mathcal{C}_l^{(j)} \,|\, \overline{\mathcal{X}}, \mathcal{V}_l)$. The second implication of (2.4) is that, conditioned on knowledge of the results from step 1 in the stochastic data generation ($\overline{\mathcal{X}}$ and $\mathcal{V}$), each component's label subset $\mathcal{C}_l^{(j)}$ forms a Markov random field (MRF) [103], *i.e.* it forms a *conditional* Markov random field, defined over the subset of class labels which occur within the component. For a given component's MRF, the neighborhood for a label C_i belonging to this component is the *full subset* of labels associated with samples originating from this same mixture component (excluding C_i itself), *i.e.* if $V_{j\,|\,i} = 1$, the neighborhood for C_i is $\{C_{i'},\ i' \in \mathcal{I}_l \backslash \{i\} : V_{j\,|\,i'} = 1\}$. Moreover, while it is not obvious in what precise form to express a component's MRF-based joint label subset distribution $P(\mathcal{C}_l^{(j)} \,|\, \overline{\mathcal{X}}, \mathcal{V}_l)$ consistent with the conditional distributions (2.4), such a joint distribution is guaranteed to exist so long as a strict positivity condition holds on the conditionals (2.4) [12]. Inspecting (2.4), one can see strict positivity of the conditionals indeed holds for the subset of class labels belonging to the given component (which is the same as the subset of class labels over which the component's MRF is defined). Thus, existence of $P(\mathcal{C}_l^{(j)} \,|\, \overline{\mathcal{X}}, \mathcal{V}_l)$ is guaranteed. Accordingly, so too the joint label distribution $P(\mathcal{C}_l \,|\, \overline{\mathcal{X}}, \mathcal{V}_l) = \prod_{j=1}^{L}\ P(\mathcal{C}_l^{(j)} \,|\, \overline{\mathcal{X}}, \mathcal{V}_l)$, is guaranteed to exist.

Having established this, we can now propose stochastic data generation for $\mathcal{X}$ as follows.

Stochastic Data Generation

(i) Generate $\overline{\mathcal{X}}$ (and, in the process $\mathcal{V}$) by independently randomly generating each sample according to a standard finite mixture density. Note that the samples in $\overline{\mathcal{X}}$ can be generated in any

order.

(ii) Given the results from step (i), generate the class labels $\{c_{N_u+1}, \ldots, c_{N_u+N_l}\}$ by sampling from the joint label distribution $P(C_{N_u+1}, \ldots, C_{N_u+N_l} \,|\, \overline{\mathcal{X}}, \mathcal{V}_l)$.

2.3.2 Pseudo-likelihood Function

Ideally, we would like to develop a strict EM algorithm, choosing the model parameters $\Theta = \{\alpha_j, \theta_j, a_j \;\; | \; j \in [L]\}$ to maximize the joint data likelihood $P(\overline{\mathcal{X}}, \mathcal{C}_l \,|\, \Theta) = P(\mathcal{C}_l \,|\, \overline{\mathcal{X}}, \Theta)$ $P(\overline{\mathcal{X}} \,|\, \Theta)$ [4]. However, in the last subsection we showed that, even if we assume knowledge of the hidden component assignments of the labeled samples $\mathcal{V}_l$, $P(\mathcal{C}_l \,|\, \overline{\mathcal{X}}, \mathcal{V}_l, \Theta)$ is a product of joint label subset distributions, associated with component-specific MRFs. It is typically intractable to work directly with the joint distribution of an MRF, due to the partition function in its denominator, which sums over all possible joint realizations of class values of the labeled samples. Our situation is more complicated because although we know that the joint distribution over class labels factorizes as a product over terms associated with each mixture component, the joint distributions $P(\mathcal{C}_l^{(j)} \,|\, \overline{\mathcal{X}}, \mathcal{V}_l, \Theta), \; j = 1, \ldots, L$ are unknown – the MRFs are only defined via the conditional distributions given by (2.4). One standard strategy for handling intractabilities when performing learning or inference involving MRFs is to work with the *pseudo-likelihood* function [13], [103], rather than the true joint likelihood. For a collection of statistically dependent random variables $\{A_1, \ldots, A_N\}$, the pseudo-likelihood employs the approximation $\widetilde{P}(A_1, \ldots, A_N) = \prod_{i=1}^{N} P(A_i \,|\, \{A_m : m \neq i\})$. The pseudo-likelihood approximation applied to our problem gives:

$$\widetilde{P}(\mathcal{C}_l \,|\, \overline{\mathcal{X}}, \mathcal{V}_l, \Theta) \;=\; \prod_{i=N_u+1}^{N_u+N_l} P(C_i = c_i \,|\, \{c_m : m \in \mathcal{I}_l \backslash \{i\}\}, \overline{\mathcal{X}}, \mathcal{V}_l, \Theta). \tag{2.6}$$

Note, conveniently, that the required conditional probabilities in (2.6) are precisely those given in (2.4). In what follows, we develop a generalized EM algorithm [52], [119], [59] to estimate the model parameters Θ to maximize the data pseudo-log-likelihood. Approaches that employ maximum pseudo-likelihood for paramater estimation can also be found in [67] and [139].

[4] We have included Θ in the set of conditioning variables of the joint-probabilities in order make the dependence on the parameters explicit during the estimation process. This also allows for an extension to MAP estimation, where the parameters are treated as random variables with a prior probability distribution $P(\Theta)$.

2.3.3 Complete Data Pseudo-Log-Likelihood

Based on our model's stochastic data generation, it is natural to choose the mixture components $\{V_{j\,|\,i},\ \forall j \in [L],\ \forall i \in [N_u{+}N_l]\}$ as the hidden (latent) variables within the EM framework [48], [111]. The *complete data* likelihood of our model can be written as

$$\begin{aligned} P(\overline{\mathcal{X}}, \mathcal{C}_l, \mathcal{V} \,|\, \Theta) &= P(\mathcal{C}_l \,|\, \overline{\mathcal{X}}, \mathcal{V}, \Theta)\ P(\overline{\mathcal{X}}, \mathcal{V} \,|\, \Theta) \\ &= P(\mathcal{C}_l \,|\, \overline{\mathcal{X}}, \mathcal{V}_l, \Theta) \prod_{i=1}^{N_u+N_l} \prod_{j=1}^{L} (\alpha_j\ f_{\underline{X}\,|\,j}(\underline{x}_i \,|\, \theta_j))^{V_{j\,|\,i}}. \end{aligned}$$

Invoking the pseudo-likelihood approximation for $P[\mathcal{C}_l \,|\, \overline{\mathcal{X}}, \mathcal{V}_l, \Theta]$ and taking logarithm, we get the complete data pseudo-log-likelihood for our model:

$$\begin{aligned} \log \widetilde{P}(\overline{\mathcal{X}}, \mathcal{C}_l, \mathcal{V} \,|\, \Theta) \ &= \ \sum_{i=1}^{N_u+N_l} \sum_{j=1}^{L} V_{j\,|\,i}\ \log[\alpha_j f_{\underline{X}\,|\,j}(\underline{x}_i \,|\, \theta_j)] \\ + \ \sum_{i=N_u+1}^{N_u+N_l} &\log P(C_i = c \,|\, \{c_m : m \in \mathcal{I}_l \backslash \{i\}\}, \overline{\mathcal{X}}, \mathcal{V}_l, \Theta) \\ = \ \sum_{i=1}^{N_u+N_l} \sum_{j=1}^{L} V_{j\,|\,i}\ &\log[\alpha_j f_{\underline{X}\,|\,j}(\underline{x}_i \,|\, \theta_j)] \qquad (2.7) \\ + \ \sum_{i=N_u+1}^{N_u+N_l} \sum_{j=1}^{L} V_{j\,|\,i}\ &\log\left[\, \delta\left(M_{j,c_i}(i) > 0\right)\ R_{j,i}(c_i \,|\, \mathcal{C}_l, \overline{\mathcal{X}}, \mathcal{V}_l)\ +\ \delta\left(M_{j,c_i}(i) = 0\right)\ \beta_{c_i\,|\,j}\,\right]. \end{aligned}$$

2.3.4 Generalized EM Algorithm

For our model, the joint posterior distribution of the latent variables conditioned on the observed data is given by [5]

$$P(\mathcal{V} \,|\, \overline{\mathcal{X}}, \mathcal{C}_l, \Theta) \ = \ \frac{P(\mathcal{C}_l \,|\, \overline{\mathcal{X}}, \mathcal{V}_l, \Theta)\ P(\overline{\mathcal{X}}, \mathcal{V} \,|\, \Theta)}{\sum_{\mathcal{V}} P(\mathcal{C}_l \,|\, \overline{\mathcal{X}}, \mathcal{V}_l, \Theta)\ P(\overline{\mathcal{X}}, \mathcal{V} \,|\, \Theta)}.$$

Even if we apply the pseudo-likelihood approximation to $P(\mathcal{C}_l \,|\, \overline{\mathcal{X}}, \mathcal{V}_l, \Theta)$, the normalization term in the denominator and the expectations with respect to this distribution will not be tractable to compute in general (unless the labeled samples are very sparse) because of the non-linear de-

[5] We use $\sum_{\mathcal{V}}$ as a shorthand notation for $\sum_{\underline{V}_1 \in \Delta} \sum_{\underline{V}_2 \in \Delta} \cdots \sum_{\underline{V}_{N_u+N_l} \in \Delta}$, where Δ is the set of all $\{0,1\}$-valued tuples of size L such that in each tuple exactly one of the values is 1 and the rest are 0, and $\underline{V}_i = [V_{1\,|\,i}, \ldots, V_{L\,|\,i}]^T$ is the vector of latent variables of sample i.

pendence of the RNN terms $R_{j,i}(c_i \,|\, \mathcal{C}_l, \overline{\mathcal{X}}, \mathcal{V}_l)$ on $\mathcal{V}_l$ – the latent variables associated with the labeled samples. One way to circumvent this problem is to treat the latent variables associated with the labeled samples $\mathcal{V}_l$ as unknown *deterministic* variables (parameters) rather than as random variables. There is precedence for such treatment, both in proposals to maximize a mixture model's complete data log-likelihood [85], [14], as well as from K-means clustering, which also effectively treats data assignments to clusters as binary (0-1) variables to optimize[6]. In our case, such treatment greatly assists development of a tractable estimation and class inference procedure. Based on this assumption, and making use of the fact that the labeled samples are independent of $\mathcal{V}_u$, the posterior distribution over $\mathcal{V}_u$ conditioned on the observed data is given by

$$P(\mathcal{V}_u \,|\, \overline{\mathcal{X}}, \mathcal{C}_l, \Theta) = P(\mathcal{V}_u \,|\, \mathcal{X}_u, \mathcal{X}_l, \Theta) = \frac{P(\mathcal{X}_u, \mathcal{V}_u \,|\, \Theta)\, P(\mathcal{X}_l \,|\, \Theta)}{\sum\limits_{\mathcal{V}_u} P(\mathcal{X}_u, \mathcal{V}_u \,|\, \Theta)\, P(\mathcal{X}_l \,|\, \Theta)} = \frac{P(\mathcal{X}_u, \mathcal{V}_u \,|\, \Theta)}{\sum\limits_{\mathcal{V}_u} P(\mathcal{X}_u, \mathcal{V}_u \,|\, \Theta)}$$

$$= \frac{\prod\limits_{i=1}^{N_u} \prod\limits_{j=1}^{L} (\alpha_j \, f_{\underline{X} \,|\, j}(\underline{x}_i \,|\, \theta_j))^{V_{j\,|\,i}}}{\prod\limits_{i=1}^{N_u} \sum\limits_{\underline{V}_i \in \Delta} \prod\limits_{j=1}^{L} (\alpha_j \, f_{\underline{X} \,|\, j}(\underline{x}_i \,|\, \theta_j))^{V_{j\,|\,i}}} = \prod_{i=1}^{N_u} \prod_{j=1}^{L} P[V_{j\,|\,i} = 1 \,|\, \underline{x}_i, \Theta]^{V_{j\,|\,i}}, \tag{2.8}$$

where

$$P[V_{j\,|\,i} = 1 \,|\, \underline{x}_i, \Theta] \;=\; \frac{\alpha_j \, f_{\underline{X} \,|\, j}(\underline{x}_i \,|\, \theta_j)}{\sum\limits_{k=1}^{L} \alpha_k \, f_{\underline{X} \,|\, k}(\underline{x}_i \,|\, \theta_k)}, \;\; \forall j \in [L], \; \forall i \in [N_u] \tag{2.9}$$

is the standard component posterior of a mixture model.

The *incomplete data pseudo-log-likelihood* of our model, corresponding to the complete data pseudo-log-likelihood (2.7), is given by

$$\log \widetilde{P}(\overline{\mathcal{X}}, \mathcal{C}_l \,|\, \Theta) \;=\; \log \sum_{\mathcal{V}_u} \widetilde{P}(\overline{\mathcal{X}}, \mathcal{C}_l, \mathcal{V} \,|\, \Theta)$$

$$= \log \sum_{\mathcal{V}_u} \left[\widetilde{P}(\mathcal{C}_l \,|\, \overline{\mathcal{X}}, \mathcal{V}_l, \Theta) \prod_{i=1}^{N_u} \prod_{j=1}^{L} (\alpha_j f_{\underline{X} \,|\, j}(\underline{x}_i \,|\, \theta_j))^{V_{j\,|\,i}} \prod_{i=N_u+1}^{N_u+N_l} \prod_{j=1}^{L} (\alpha_j f_{\underline{X} \,|\, j}(\underline{x}_i \,|\, \theta_j))^{v_{j\,|\,i}} \right]$$

$$= \log \left[\widetilde{P}(\mathcal{C}_l \,|\, \overline{\mathcal{X}}, \mathcal{V}_l, \Theta) \prod_{i=1}^{N_u} \sum_{\underline{V}_i \in \Delta} \prod_{j=1}^{L} (\alpha_j f_{\underline{X} \,|\, j}(\underline{x}_i \,|\, \theta_j))^{V_{j\,|\,i}} \prod_{i=N_u+1}^{N_u+N_l} \prod_{j=1}^{L} (\alpha_j f_{\underline{X} \,|\, j}(\underline{x}_i \,|\, \theta_j))^{v_{j\,|\,i}} \right]$$

[6] In fact, it can be shown that, for the squared Euclidean distance measure, the K-means algorithm maximizes the complete data log-likelihood for a Gaussian mixture model with an isotropic component covariance matrix, $\sigma^2\mathbf{I}$, shared by all components.

$$= \sum_{i=N_u+1}^{N_u+N_l} \sum_{j=1}^{L} v_{j\,|\,i} \log \left[\delta\left(M_{j,c_i}(i) > 0\right) \; R_{j,i}(c_i \,|\, \mathcal{C}_l, \overline{\mathcal{X}}, \mathcal{V}_l) \; + \; \delta\left(M_{j,c_i}(i) = 0\right) \; \beta_{c_i\,|\,j} \right]$$

$$+ \sum_{i=1}^{N_u} \log \sum_{j=1}^{L} \alpha_j \, f_{\underline{X}\,|\,j}(\underline{x}_i \,|\, \theta_j) \; + \sum_{i=N_u+1}^{N_u+N_l} \sum_{j=1}^{L} v_{j\,|\,i} \; \log[\alpha_j \; f_{\underline{X}\,|\,j}(\underline{x}_i \,|\, \theta_j)]. \tag{2.10}$$

We observe that this pseudo-log-likelihood function possesses a *data exchangeability* property, *i.e.*, it does not depend on the order in which samples are generated – both with respect to the samples in $\overline{\mathcal{X}}$ and, separately, with respect to the class labels in $\mathcal{C}_l$. The model only requires that the class labels be generated after first generating $\overline{\mathcal{X}}$. To find the maximum pseudo-likelihood estimates of the parameters, we will use the generalized EM (GEM) algorithm, a variant of the standard EM algorithm [52], [119], [59], which iteratively finds parameter estimates that monotonically ascend in $\log \widetilde{P}(\overline{\mathcal{X}}, \mathcal{C}_l \,|\, \Theta)$ until a local maximum solution is found. The parameter updates at each iteration of the GEM algorithm are found by maximizing (or increasing) a lower bound of the incomplete data pseudo-log-likelihood, which is often easier to maximize. We next present the E-step and the generalized M-step for our model. Note that since $\mathcal{V}_l$ are also unknown parameters to be estimated, we augment Θ to include $\mathcal{V}_l$.

2.3.4.1 E-step

Suppose $\Theta^{(t)}$ are the parameter estimates at iteration t of the EM algorithm, the E-step first computes the component posteriors for the unlabeled samples given by (2.9) (evaluated at $\Theta = \Theta^{(t)}$), and then finds an auxiliary lower bound of the incomplete data pseudo-log-likelihood given by

$$Q(\Theta, \Theta^{(t)}) \; = \; E[\log \widetilde{P}(\overline{\mathcal{X}}, \mathcal{C}_l, \mathcal{V} \,|\, \Theta) \,|\, \overline{\mathcal{X}}, \mathcal{C}_l, \Theta^{(t)}] \; + \; H[P(\mathcal{V}_u \,|\, \overline{\mathcal{X}}, \mathcal{C}_l, \Theta^{(t)})] \tag{2.11}$$

$$= \sum_{\mathcal{V}_u} P(\mathcal{V}_u \,|\, \overline{\mathcal{X}}, \mathcal{C}_l, \Theta^{(t)}) \; \log \widetilde{P}(\overline{\mathcal{X}}, \mathcal{C}_l, \mathcal{V} \,|\, \Theta) - \sum_{\mathcal{V}_u} P(\mathcal{V}_u \,|\, \overline{\mathcal{X}}, \mathcal{C}_l, \Theta^{(t)}) \; \log P(\mathcal{V}_u \,|\, \overline{\mathcal{X}}, \mathcal{C}_l, \Theta^{(t)}),$$

where the first term is the expectation of the complete data pseudo-log-likelihood with respect to the distribution $P(\mathcal{V}_u \,|\, \overline{\mathcal{X}}, \mathcal{C}_l, \Theta^{(t)})$, and the second term is the entropy of the distribution $P(\mathcal{V}_u \,|\, \overline{\mathcal{X}}, \mathcal{C}_l, \Theta^{(t)})$. Since the distribution $P(\mathcal{V}_u \,|\, \overline{\mathcal{X}}, \mathcal{C}_l, \Theta^{(t)})$ has the factorized form (2.8), it is straightforward to show that the final expression for these two terms is given by

$$E[\log \widetilde{P}(\overline{\mathcal{X}}, \mathcal{C}_l, \mathcal{V} \,|\, \Theta) \,|\, \overline{\mathcal{X}}, \mathcal{C}_l, \Theta^{(t)}] \; = \sum_{i=N_u+1}^{N_u+N_l} \sum_{j=1}^{L} v_{j\,|\,i} \; \log[\alpha_j \; f_{\underline{X}\,|\,j}(\underline{x}_i \,|\, \theta_j)]$$

$$+ \sum_{i=1}^{N_u} \sum_{j=1}^{L} P[V_{j\,|\,i} = 1 \,|\, \underline{x}_i, \Theta^{(t)}] \; \log[\alpha_j \; f_{\underline{X}\,|\,j}(\underline{x}_i \,|\, \theta_j)]$$

$$+ \sum_{i=N_u+1}^{N_u+N_l} \sum_{j=1}^{L} v_{j\,|\,i} \log \left[\delta\left(M_{j,c_i}(i) > 0\right) \; R_{j,i}(c_i \,|\, \mathcal{C}_l, \overline{\mathcal{X}}, \mathcal{V}_l) \; + \; \delta\left(M_{j,c_i}(i) = 0\right) \; \beta_{c_i\,|\,j} \right],$$

and

$$H[P(\mathcal{V}_u \,|\, \overline{\mathcal{X}}, \mathcal{C}_l, \Theta^{(t)})] \; = \; -\sum_{i=1}^{N_u} \sum_{j=1}^{L} P[V_{j\,|\,i} = 1 \,|\, \underline{x}_i, \Theta^{(t)}] \log P[V_{j\,|\,i} = 1 \,|\, \underline{x}_i, \Theta^{(t)}].$$

When evaluated at $\Theta^{(t)}$, the lower bound will be exactly equal to the incomplete data pseudo-log-likelihood, *i.e.*, $Q(\Theta^{(t)}, \Theta^{(t)}) = \log \widetilde{P}(\overline{\mathcal{X}}, \mathcal{C}_l \,|\, \Theta^{(t)})$.

2.3.4.2 Generalized M-step

The standard M-step of the EM algorithm at iteration t would involve solving the problem $\Theta^{(t+1)} = \arg\max_{\Theta} Q(\Theta, \Theta^{(t)})$. However, finding a joint closed form maximum with respect to all the parameters is not tractable because of the complex dependence of $Q(\Theta, \Theta^{(t)})$ on the parameters $\mathcal{V}_l$ and $a_j, \;\; j \in [L]$. The GEM algorithm handles this problem using a generalized M-step instead of the standard M-step. The generalized M-step considers, in turn, different parameter subsets, and optimizes these given all other parameters in Θ held fixed. Each such optimization, and hence the sequence of such optimizations is non-decreasing in $Q(\Theta, \Theta^{(t)})$, and hence also non-decreasing in its upper bound $\log \widetilde{P}(\overline{\mathcal{X}}, \mathcal{C}_l \,|\, \Theta)$ [111], [119]. Below, we specify the generalized M-step for multivariate Gaussian component densities, *i.e.*, $f_{\underline{X}\,|\,j}(\underline{x} \,|\, \theta_j) \; = \; \mathcal{N}(\underline{x}\,; \underline{\mu}_j, \mathbf{\Sigma}_j)$, which is the model we will use in the experimental evaluation of our method.

Update of $\{\alpha_j\}$ *and* $\{\theta_j\}$*:*
It is straightforward to derive closed-form M-step updates for the mixture model parameters $\{\alpha_j, \; \forall j \in [L]\}$ and $\{\theta_j = (\underline{\mu}_j, \mathbf{\Sigma}_j), \; \forall j \in [L]\}$ which globally maximize $Q(\Theta, \Theta^{(t)})$, given all other parameters held fixed. These updates are given by:

$$\alpha_j^{(t+1)} \; = \; \frac{\sum_{i=1}^{N_u} P[V_{j\,|\,i} = 1 \,|\, \underline{x}_i, \Theta^{(t)}] \; + \; \sum_{i=N_u+1}^{N_u+N_l} v_{j\,|\,i}^{(t)}}{N_u + N_l}, \;\; \forall j \in [L], \tag{2.12}$$

$$\underline{\mu}_j^{(t+1)} \; = \; \frac{\sum_{i=1}^{N_u} P[V_{j\,|\,i} = 1 \,|\, \underline{x}_i, \Theta^{(t)}] \; \underline{x}_i \; + \; \sum_{i=N_u+1}^{N_u+N_l} v_{j\,|\,i}^{(t)} \; \underline{x}_i}{(N_u + N_l) \; \alpha_j^{(t+1)}}, \;\; \forall j \in [L], \tag{2.13}$$

$$\boldsymbol{\Sigma}_j^{(t+1)} = \frac{\sum_{i=1}^{N_u} P[V_{j\,|\,i} = 1 \,|\, \underline{x}_i, \Theta^{(t)}]\ (\underline{x}_i - \underline{\mu}_j^{(t+1)})(\underline{x}_i - \underline{\mu}_j^{(t+1)})^T}{(N_u + N_l)\ \alpha_j^{(t+1)}} + \frac{\sum_{i=N_u+1}^{N_u+N_l} v_{j\,|\,i}^{(t)}\ (\underline{x}_i - \underline{\mu}_j^{(t+1)})(\underline{x}_i - \underline{\mu}_j^{(t+1)})^T}{(N_u + N_l)\ \alpha_j^{(t+1)}},\ \ \forall j \in [L]. \tag{2.14}$$

Update of scale parameters, $\{a_j\}$:

Next, we need to maximize $Q(\Theta, \Theta^{(t)})$ over the $\{a_j\}$ parameters, given all other parameters fixed at $\{\{\alpha_j^{(t+1)}\}, \{\theta_j^{(t+1)}\}, \{v_{j\,|\,i}^{(t)}\}\}$. Since it is not possible to find closed form M-step updates for these parameters, we use gradient ascent to find updated values $a_j^{(t+1)},\ \forall j \in [L]$ which increase the function $Q(\Theta, \Theta^{(t)})$. Note that the term which depends on the scale parameters is the same in both $Q(\Theta, \Theta^{(t)})$ and the incomplete data pseudo-log-likelihood $\log \widetilde{P}(\overline{\mathcal{X}}, \mathcal{C}_l \,|\, \Theta)$. Hence, maximizing one with respect to the scale parameters is equivalent to maximizing the other.

Update of the $\{v_{j\,|\,i}\}$ variables:

Next, given the fixed set of parameters, $\{\{\alpha_j^{(t+1)}\}, \{\theta_j^{(t+1)}\}, \{a_j{}^{(t+1)}\}\}$, we maximize the function $Q(\Theta, \Theta^{(t)})$ over the assignments of labeled samples to components, $\{v_{j\,|\,i},\ \forall j \in [L], \forall i \in \mathcal{I}_l\}$. Note that for this parameter subset also, maximizing $Q(\Theta, \Theta^{(t)})$ is equivalent to maximizing $\log \widetilde{P}(\overline{\mathcal{X}}, \mathcal{C}_l \,|\, \Theta)$. If $v_{j\,|\,i}$ are treated as binary variables taking values $\{0, 1\}$, with the constraint that $\sum_{j=1}^{L} v_{j\,|\,i} = 1,\ \forall i \in \mathcal{I}_l$, then joint maximization would require a search over L^{N_l} values, which is typically intractable. A tractable approach to solve this problem would be to maximize $Q(\Theta, \Theta^{(t)})$ (or equivalently $\log \widetilde{P}(\overline{\mathcal{X}}, \mathcal{C}_l \,|\, \Theta)$) cyclically, visiting one labeled sample at a step and choosing its optimal component assignment, while the component assignments of the other labeled samples are held fixed. Each such step requires only a search over L values. The component assignments of labeled samples are iteratively optimized in this fashion (with some randomization in the order) until a complete sweep over all the labeled samples leads to no change of component assignments (*i.e.*, a local maximum of $Q(\Theta, \Theta^{(t)})$ is obtained). This cyclical optimization is akin to discrete optimization approaches previously taken, for example, in [13] and [59]. Another tractable approach for optimizing these variables is as follows. We can treat the variables $v_{j\,|\,i}$ as probabilities and without loss of representation power parameterize them using *softmax functions* [21], [120], *i.e.*, $v_{j\,|\,i} = \frac{e^{w_{j,i}}}{\sum_{k=1}^{L} e^{w_{k,i}}}$. We can jointly optimize the real-valued variables $\{w_{j,i},\ \forall j \in [L],\ \forall i \in \mathcal{I}_l\}$ via gradient ascent on $Q(\Theta, \Theta^{(t)})$, again with all other parameters in Θ held fixed. In our experiments, we only evaluated the former discrete

optimization approach.

Having fully described the parameter estimation method for our model given a data set with labeled and unlabeled samples, we next look at class inference based on our model.

2.3.5 Class inference

Denote a new test sample that needs to be classified according to our model as $\underline{x}_n$, where $n = N_u + N_l + 1$ (for notational convenience). To perform inductive inference on $\underline{x}_n$, we would like to evaluate the posterior probability $P(C_n = c_n \,|\, \overline{\mathcal{X}}, \mathcal{C}_l, \underline{x}_n, \Theta)$, which conditions on all available information, and choose the class which maximizes this posterior. Typically, there is a *unique* maximum *a posteriori* decision rule. However, for our NFGL model we can derive two distinct class inference rules. Fundamentally, we derive two different rules because there are two different subsets of data generated by the NFGL model – the labeled and unlabeled subsets – with a *different* stochastic data generation mechanism for each of these types of data. Accordingly, there is also a different class posterior probability for samples belonging to each of these two types. Thus, the inference rule depends on whether we interpret a test sample as a sample belonging to the labeled data subset or to the unlabeled data subset.

Test sample is part of labeled data subset:
First, suppose that the test sample is from the *labeled* subset, albeit with its class label C_n and mixture component of origin $V_{j\,|\,n},\ \forall j$ unknown (missing). Essentially, in this case, we consider C_n to be part of the undirected graph (MRF) on $\mathcal{C}_l' = \mathcal{C}_l \cup \{C_n\}$. The exact class posterior, *i.e.*, the probability of C_n given $\underline{x}_n$ and all other observed data is given by:

$$P(C_n = c_n \,|\, \overline{\mathcal{X}}, \underline{x}_n, \mathcal{C}_l, \Theta) \;=\; \frac{P\left(\mathcal{C}_l, C_n = c_n \,|\, \overline{\mathcal{X}}, \underline{x}_n, \Theta\right)}{\sum_{k=1}^{N_c} P\left(\mathcal{C}_l, C_n = k \,|\, \overline{\mathcal{X}}, \underline{x}_n, \Theta\right)}. \tag{2.15}$$

Note that although the mixture components of all the labeled samples, $\mathcal{V}_l$, are not explicit in the conditioning information of the class posterior, they are available as part of the estimated parameter set Θ. Since the mixture component for the test sample $v_{j\,|\,n},\ \forall j$ is not known, it has to be marginalized out. Defining $\overline{\mathcal{X}}' = \overline{\mathcal{X}} \cup \{\underline{x}_n\},\ \mathcal{V}_l' = \mathcal{V}_l \cup \{v_{j\,|\,n},\ \ \forall j \in [L]\}$, and observing that for our stochastic data generation method the mixture component of the test sample is independent of $\overline{\mathcal{X}}$, we get

$$P\left(\mathcal{C}_l, C_n = c_n \,|\, \overline{\mathcal{X}}, \underline{x}_n, \Theta\right) \;=\; \sum_{j=1}^{L} P\left(\mathcal{C}_l, C_n = c_n \,|\, v_{j\,|\,n} = 1, \overline{\mathcal{X}}, \underline{x}_n, \Theta\right) P(v_{j\,|\,n} = 1 \,|\, \overline{\mathcal{X}}, \underline{x}_n, \Theta)$$

$$= \sum_{j=1}^{L} P\left(\mathcal{C}_l, C_n = c_n \,|\, v_{j\,|\,n} = 1, \overline{\mathcal{X}}', \Theta\right) P(v_{j\,|\,n} = 1 \,|\, \underline{x}_n, \Theta),$$

where $P(v_{j\,|\,n} = 1 \,|\, \underline{x}_n, \Theta)$ is the standard mixture component posterior given by (2.9). Exact computation of these probabilities is not possible because it involves the joint distribution of the MRF on $\mathcal{C}'_l$. However, we can derive an approximate posterior probability, again based on the pseudo-likelihood [13], [103], as follows:

$$\widetilde{P}\left(\mathcal{C}_l, C_n = c_n \,|\, \overline{\mathcal{X}}, \underline{x}_n, \Theta\right) =$$
$$\sum_{j=1}^{L} P(v_{j\,|\,n} = 1 \,|\, \underline{x}_n, \Theta) \prod_{i=N_u+1}^{N_u+N_l+1} P(C_i = c_i \,|\, \mathcal{C}'_l \backslash \{c_i\}, \overline{\mathcal{X}}', v_{j\,|\,n} = 1, \Theta).$$

Comparing with (2.4) and reconciling the notational differences, it should be clear that

$$P(C_i = c_i \,|\, \mathcal{C}'_l \backslash \{c_i\}, \overline{\mathcal{X}}', v_{j\,|\,n} = 1, \Theta)$$
$$= \prod_{k=1}^{L} \left[\delta\left(M_{k,c_i}(i) > 0\right) \; R_{k,i}(c_i \,|\, \mathcal{C}'_l, \, \overline{\mathcal{X}}', \, \mathcal{V}'_l) \; + \; \delta\left(M_{k,c_i}(i) = 0\right) \; \beta_{c_i\,|\,k} \right]^{v_{k\,|\,i}}.$$

In this case the resulting class posterior and inference is approximate because we are using the pseudo-likelihood approximation in place of the joint probability of the MRF on $\mathcal{C}'_l$.

Test sample is part of unlabeled data subset:

Next, consider the case where we treat the test sample as belonging to the unlabeled data subset $\mathcal{X}_u$. What this means is that, we are assuming that the test sample is generated in the *same* way as any other sample from the unlabeled data subset (*i.e.*, i.i.d., according to the mixture density). Also, C_n is not a part of the MRF on the class labels of labeled samples. Now, it is of course true that the NFGL model does not hypothesize class label generation for samples from the unlabeled subset. However, under the assumption that the class labels for $\mathcal{X}_u$ are generated after $\mathcal{C}_l$, consistent with the conditional MRF rule (2.4), an exact class posterior probability exists for the unlabeled samples, and is given by

$$P(C_n = c_n \,|\, \overline{\mathcal{X}}, \underline{x}_n, \mathcal{C}_l, \Theta)$$
$$= \sum_{j=1}^{L} P(C_n = c_n \,|\, \overline{\mathcal{X}}', \mathcal{C}_l, v_{j\,|\,n} = 1, \Theta) \, P(v_{j\,|\,n} = 1 \,|\, \overline{\mathcal{X}}, \underline{x}_n, \mathcal{C}_l, \Theta)$$
$$= \sum_{j=1}^{L} P(v_{j\,|\,n} = 1 \,|\, \underline{x}_n, \Theta) \, P(C_n = c_n \,|\, \overline{\mathcal{X}}', \mathcal{C}_l, v_{j\,|\,n} = 1, \Theta), \tag{2.16}$$

where $P(v_{j\,|\,n} = 1\,|\,\underline{x}_n, \Theta)$ is given by (2.9) and

$$\begin{aligned} & P(C_n = c_n\,|\,\overline{\mathcal{X}}', \mathcal{C}_l, v_{j\,|\,n} = 1, \Theta) \\ = \quad & \delta\left(M_{j,c_n}(n) > 0\right) \frac{\sum\limits_{\substack{i=N_u+1: \\ c_i = c_n}}^{N_u+N_l} v_{j\,|\,i}\, e^{-a_j \|\underline{x}_i - \underline{x}_n\|^2}}{\sum\limits_{i=N_u+1}^{N_u+N_l} v_{j\,|\,i}\, e^{-a_j \|\underline{x}_i - \underline{x}_n\|^2}} + \delta\left(M_{j,c_n}(n) = 0\right) \beta_{c_n\,|\,j}. \end{aligned}$$

In summary, fundamentally there are two natural class inference rules for NFGL because one can interpret a test sample as being generated either according to the labeled subset or the unlabeled subset. As will be seen in the sequel, although it may be counter-intuitive, there are reasons for preferring the latter posterior (2.16) in practice.

2.4 A Parametric Fine-Grained Labeling Model (PFGL)

2.4.1 Problems with NFGL at Very Low Labeled Fractions

The NFGL model proposed in the previous section is a sound fine-grained framework, with some desirable properties like being non-parametrically capable of varying the complexity of its class posterior probability within mixture components, based on the number and variety of class labels (of labeled samples) that are generated (or explained) by the components. However, as we had foreshadowed earlier in section 2.3.1, when the number of labeled samples is very small the RNN class posterior (2.5) can take a value 0 or become undefined. We used the fixed component conditional class probabilities of the MOE model [122] as a "fall-back" to handle such cases. We now consider, in more detail, such scenarios where the number of labeled samples is very small, to bring out certain limitations of the NFGL method. Specifically, first, consider the case where there is a single labeled sample $(\underline{x}_{i_1}, c_{i_1})$ generated by a given component j. Now, by inspecting (2.5), recognize that in evaluating the conditional probability for c_{i_1}, the sums in both the numerator and the denominator on the right hand side necessarily *exclude* i_1. The implication is that $R_{j,i_1}(c\,|\,\mathcal{C}_l, \overline{\mathcal{X}}, \mathcal{V}_l)$ evaluates to zero divided by zero, for all classes, including the true class c_{i_1}. Next consider a related but distinct scenario wherein, within component j, there are *only* two labeled samples $(\underline{x}_{i_1}, 1)$ and $(\underline{x}_{i_2}, 2)$ from different classes denoted by 1 and 2 here. Again by inspecting (2.5), we find that $R_{j,i_1}(1\,|\,\mathcal{C}_l, \overline{\mathcal{X}}, \mathcal{V}_l) = 0$ and, likewise, $R_{j,i_2}(2\,|\,\mathcal{C}_l, \overline{\mathcal{X}}, \mathcal{V}_l) = 0$. In this case, if both $\underline{x}_1$ and $\underline{x}_2$ are well explained by component j, then we would expect the MOE model of [122] to estimate values of $\beta_{1\,|\,j} \approx 0.5$ and $\beta_{2\,|\,j} \approx 0.5$. Although this is a well-

defined solution for the class probabilities within component j, it is not a fine-grained solution – one where points from component j in the neighborhood of $\underline{x}_{i_1}$ have a high probability for class 1 and points from component j in the neighborhood of $\underline{x}_{i_2}$ have a high probability for class 2. Moreover, similar results will occur in scenarios where, within a given component, there are single labeled sample *representatives* from multiple classes. While it may be argued that these are extreme cases, this in fact will be a fairly common scenario when the fraction of labeled data is very small and there are multiple classes in the data. Moreover, such scenarios *are* the reason why semi-supervised learning (rather than supervised learning) may be needed in the first place.

Accordingly, in this section we propose an alternative to NFGL's (non-parametric) RNN model (2.4) that does not suffer from these limitations when there is labeled sample sparsity. These limitations are avoided by invoking a *parametric* within-component posterior. As discussed in section 2.3.1, there are many possible choices for parametric posterior models. For concreteness and to make explicit some connection to the NFGL method from the previous section, we will focus on a particular posterior model – a randomized *nearest prototype* (RNP) posterior:

$$P(C = c \,|\, V_{j\,|\,i} = 1, \underline{x}_i, \phi_j) \;=\; \frac{\sum\limits_{\substack{l=1: \\ m_l^{(j)}=c}}^{N^{(j)}} e^{-a_j \,||\underline{x}_i - \underline{s}_l^{(j)}||^2}}{\sum\limits_{l=1}^{N^{(j)}} e^{-a_j \,||\underline{x}_i - \underline{s}_l^{(j)}||^2}}, \;\; \forall c \in \mathcal{C}, \; \forall j \in [L] \tag{2.17}$$

where $N^{(j)}$ is the number of prototypes and ϕ_j is the parameter set of the RNP posterior for component j. As before, $\underline{V}_i = [V_{1\,|\,i}, \ldots, V_{L\,|\,i}]^T$ specifies the mixture component of sample $\underline{x}_i$, with $V_{j\,|\,i}$ taking a value 1 (0) according to whether to sample $\underline{x}_i$ was generated (not generated) by component j. The RNP posterior of each component $j \in [L]$ has a scale parameter $a_j > 0$, $N^{(j)}$ prototype vectors $\underline{s}_l^{(j)} \in \mathbb{R}^d, \; l = 1, \ldots, N_j$, and an index mapping of prototype vectors to classes $m_l^{(j)} \in \mathcal{C}, \; l = 1, \ldots, N^{(j)}$, where $m_l^{(j)}$ is the class to which the prototype $\underline{s}_l^{(j)}$ belongs. Accordingly, the set of adjustable parameters of the RNP posterior for component j is $\phi_j = \{a_j, \{m_l^{(j)}, \underline{s}_l^{(j)} \;\, | \; l \in [N^{(j)}]\}\}$. Note that as $a_j \to \infty$, (2.17) converges to a nearest-prototype decision rule [91], [157] within component j. Also, note that unlike (2.5), (2.17) is always well-defined so long as $N^{(j)} \geq 1$, and (2.17) will not assign zero probability to a class that possesses only a single prototype within a given component. However, unlike (2.5), (2.17) generally *does* require model order selection to choose the number of prototypes, $N^{(j)}$ in the class posterior for each component. In section 2.4.5, we will describe a methodology for judiciously selecting the model order complexity.

2.4.2 Stochastic Data Generation

Associated with the component conditional class posterior model (2.17), there is a natural, simple stochastic data generation mechanism, as follows:

1) Independently, for each unlabeled sample $\underline{x}_i \in \mathcal{X}_u$:

i) randomly select a mixture component j (*i.e.*, $V_{j\,|\,i} = 1$) according to $\{\alpha_k,\ \forall k \in [L]\}$.

ii) randomly generate $\underline{x}_i$ according to $f_{\underline{X}\,|\,j}(\underline{x}\,|\,\theta_j)$.

2) Independently, for each labeled sample $(\underline{x}_i, c_i) \in \mathcal{X}_l$:

i) randomly select a mixture component j (*i.e.*, $V_{j\,|\,i} = 1$) according to $\{\alpha_k,\ \forall k \in [L]\}$.

ii) randomly generate $\underline{x}_i$ according to $f_{\underline{X}\,|\,j}(\underline{x}\,|\,\theta_j)$.

iii) randomly select the class label c_i according to the component-conditional RNP posterior $\{P(C = c\,|\,V_{j\,|\,i} = 1, \underline{x}_i, \phi_j),\ \forall c \in \mathcal{C}\}$ specified in (2.17).

Clearly, this is a fine-grained extension of the stochastic data generation in [122].

2.4.3 Incomplete and Complete Data Log-likelihood

Denote the set of all model parameters by $\Theta = \{\alpha_j, \theta_j,\ \phi_j \mid\ j \in [L]\}$. The latent variable associated with each sample (indexed) i is its mixture component of origin $\underline{V}_i$. Denote the set of all latent variables by $\mathcal{V}\}$, the set of latent variables associated with unlabeled samples by $\mathcal{V}_u$, and the set of latent variables associated with labeled samples by $\mathcal{V}_l$. Based on the stochastic data generation of our model, the incomplete data log-likelihood and the complete data log-likelihood [48] are given respectively by

$$
\begin{aligned}
& \log P(\mathcal{X}_l, \mathcal{X}_u\,|\,\Theta) \ =\ \log P(\mathcal{X}_u\,|\,\Theta) \ +\ \log P(\mathcal{X}_l\,|\,\Theta) \\
= \ & \log \prod_{i=1}^{N_u} \sum_{j=1}^{L} (\alpha_j\ f_{\underline{X}\,|\,j}(\underline{x}_i\,|\,\theta_j)) \\
+ \ & \log \prod_{i=N_u+1}^{N_u+N_l} \sum_{j=1}^{L} (\alpha_j\ f_{\underline{X}\,|\,j}(\underline{x}_i\,|\,\theta_j)\ P(C = c_i\,|\,V_{j\,|\,i} = 1, \underline{x}_i, \phi_j)) \\
= \ & \sum_{i=1}^{N_u} \log \sum_{j=1}^{L} (\alpha_j\ f_{\underline{X}\,|\,j}(\underline{x}_i\,|\,\theta_j)) \\
+ \ & \sum_{i=N_u+1}^{N_u+N_l} \log \sum_{j=1}^{L} (\alpha_j\ f_{\underline{X}\,|\,j}(\underline{x}_i\,|\,\theta_j)\ P(C = c_i\,|\,V_{j\,|\,i} = 1, \underline{x}_i, \phi_j))
\end{aligned}
\tag{2.18}
$$

and

$$
\log P(\mathcal{X}_l, \mathcal{X}_u, \mathcal{V}\,|\,\Theta) \ =\ \log P(\mathcal{X}_u, \mathcal{V}_u\,|\,\Theta) \ +\ \log P(\mathcal{X}_l, \mathcal{V}_l\,|\,\Theta)
$$

$$
\begin{aligned}
&= \log \prod_{i=1}^{N_u} \prod_{j=1}^{L} (\alpha_j \, f_{\underline{X}\,|\,j}(\underline{x}_i \,|\, \theta_j))^{V_{j\,|\,i}} \\
&+ \log \prod_{i=N_u+1}^{N_u+N_l} \prod_{j=1}^{L} (\alpha_j \, f_{\underline{X}\,|\,j}(\underline{x}_i \,|\, \theta_j) \, P(C = c_i \,|\, V_{j\,|\,i} = 1, \underline{x}_i, \phi_j))^{V_{j\,|\,i}} \\
&= \sum_{i=1}^{N_u} \sum_{j=1}^{L} V_{j\,|\,i} \, \log(\alpha_j \, f_{\underline{X}\,|\,j}(\underline{x}_i \,|\, \theta_j)) \\
&+ \sum_{i=N_u+1}^{N_u+N_l} \sum_{j=1}^{L} V_{j\,|\,i} \, \log(\alpha_j \, f_{\underline{X}\,|\,j}(\underline{x}_i \,|\, \theta_j) \, P(C = c_i \,|\, V_{j\,|\,i} = 1, \underline{x}_i, \phi_j)). \qquad (2.19)
\end{aligned}
$$

Note that this model possesses full sample exchangeability, *i.e.*, data samples can be generated independently in any order. Given a model of fixed size (number of components and number of prototypes per component), we seek to estimate its parameters by maximum likelihood estimation. Unlike the NFGL model, where we estimate the parameters by maximizing the incomplete data pseudo-log-likelihood, here it is possible to estimate the parameters by directly maximizing the incomplete data log-likelihood (2.18).

2.4.4 Generalized EM Algorithm

In order to maximize the incomplete data log-likelihood, we will apply the GEM algorithm [52], [119], [59] (as we did for the NFGL model), which iteratively finds parameter estimates that monotonically ascend in the incomplete data log-likelihood by maximizing (or increasing) an auxiliary lower bound function, until a local maximum solution is found. Before specifying the E-step and the generalized M-step for our model, we find a expression for the joint posterior distribution of the latent variables $\mathcal{V}$ conditioned on the data and the parameters. Using (2.18) and (2.19), it can be shown that

$$
\begin{aligned}
P(\mathcal{V} \,|\, \mathcal{X}_u, \mathcal{X}_l, \Theta) \;&=\; \frac{P(\mathcal{X}_l, \mathcal{X}_u, \mathcal{V} \,|\, \Theta)}{P(\mathcal{X}_l, \mathcal{X}_u \,|\, \Theta)} \;=\; \frac{e^{\log P(\mathcal{X}_l, \mathcal{X}_u, \mathcal{V} \,|\, \Theta)}}{e^{\log P(\mathcal{X}_l, \mathcal{X}_u \,|\, \Theta)}} \\
= \prod_{i=1}^{N_u} \prod_{j=1}^{L} P(V_{j\,|\,i} = 1 \,|\, \underline{x}_i, \Theta)^{V_{j\,|\,i}} &\prod_{i=N_u+1}^{N_u+N_l} \prod_{j=1}^{L} P(V_{j\,|\,i} = 1 \,|\, \underline{x}_i, c_i, \Theta)^{V_{j\,|\,i}}, \qquad (2.20)
\end{aligned}
$$

where $\forall j \in [L],\ i = 1, \ldots, N_u$

$$
P[V_{j\,|\,i} = 1 \,|\, \underline{x}_i, \Theta] \;=\; \frac{\alpha_j \, f_{\underline{X}\,|\,j}(\underline{x}_i \,|\, \theta_j)}{\sum_{k=1}^{L} \alpha_k \, f_{\underline{X}\,|\,k}(\underline{x}_i \,|\, \theta_k)} \qquad (2.21)
$$

and $\forall j \in [L],\ i = N_u + 1, \ldots, N_u + N_l$

$$P[V_{j\,|\,i} = 1 \,|\, \underline{x}_i, c_i, \Theta] \;=\; \frac{\alpha_j \; f_{\underline{X}\,|\,j}(\underline{x}_i \,|\, \theta_j) \; P(C = c_i \,|\, V_{j\,|\,i} = 1, \underline{x}_i, \phi_j)}{\sum_{k=1}^{L} \alpha_k \; f_{\underline{X}\,|\,k}(\underline{x}_i \,|\, \theta_k) \; P(C = c_i \,|\, V_{k\,|\,i} = 1, \underline{x}_i, \phi_k)}. \tag{2.22}$$

2.4.4.1 E-step

Suppose $\Theta^{(t)}$ are the parameter estimates at iteration t of the EM algorithm, the E-step first computes the component posteriors for the unlabeled and labeled samples given by (2.21) and (2.22) (evaluated at $\Theta = \Theta^{(t)}$), and then finds an auxiliary lower bound of the incomplete data log-likelihood given by

$$\begin{aligned} Q(\Theta, \Theta^{(t)}) \;&=\; E[\log P(\mathcal{X}_l, \mathcal{X}_u, \mathcal{V} \,|\, \Theta) \,|\, \mathcal{X}_l, \mathcal{X}_u, \Theta^{(t)}] \;+\; H[P(\mathcal{V} \,|\, \mathcal{X}_u, \mathcal{X}_l, \Theta^{(t)})] \qquad (2.23) \\ =\; \sum_{\mathcal{V}} P(\mathcal{V} \,|\, \mathcal{X}_u, \mathcal{X}_l, \Theta^{(t)}) \; &\log P(\mathcal{X}_l, \mathcal{X}_u, \mathcal{V} \,|\, \Theta) - \sum_{\mathcal{V}} P(\mathcal{V} \,|\, \mathcal{X}_u, \mathcal{X}_l, \Theta^{(t)}) \; \log P(\mathcal{V} \,|\, \mathcal{X}_u, \mathcal{X}_l, \Theta^{(t)}), \end{aligned}$$

where the first term is the expectation of the complete data log-likelihood with respect to the distribution $P(\mathcal{V} \,|\, \mathcal{X}_u, \mathcal{X}_l, \Theta^{(t)})$, and the second term is the entropy of the distribution $P(\mathcal{V} \,|\, \mathcal{X}_u, \mathcal{X}_l, \Theta^{(t)})$. Since $P(\mathcal{V} \,|\, \mathcal{X}_u, \mathcal{X}_l, \Theta^{(t)})$ has the factorized form (2.20), it is straightforward to show that the final expression for these two terms is given by

$$\begin{aligned} E[\log P(\mathcal{X}_l, \mathcal{X}_u, \mathcal{V} \,|\, \Theta) \,|\, \mathcal{X}_l, \mathcal{X}_u, \Theta^{(t)}] \;&=\; \sum_{i=1}^{N_u} \sum_{j=1}^{L} P(V_{j\,|\,i} = 1 \,|\, \underline{x}_i, \Theta^{(t)}) \; \log[\alpha_j \; f_{\underline{X}\,|\,j}(\underline{x}_i \,|\, \theta_j)] \\ +\; \sum_{i=N_u+1}^{N_u+N_l} \sum_{j=1}^{L} &P(V_{j\,|\,i} = 1 \,|\, \underline{x}_i, c_i, \Theta^{(t)}) \; \log(\alpha_j \; f_{\underline{X}\,|\,j}(\underline{x}_i \,|\, \theta_j) \; P(C = c_i \,|\, V_{j\,|\,i} = 1, \underline{x}_i, \phi_j)) \end{aligned}$$

and

$$\begin{aligned} H[P(\mathcal{V} \,|\, \mathcal{X}_u, \mathcal{X}_l, \Theta^{(t)})] \;&=\; -\sum_{i=1}^{N_u} \sum_{j=1}^{L} P(V_{j\,|\,i} = 1 \,|\, \underline{x}_i, \Theta^{(t)}) \; \log P(V_{j\,|\,i} = 1 \,|\, \underline{x}_i, \Theta^{(t)}) \\ -\; \sum_{i=N_u+1}^{N_u+N_l} \sum_{j=1}^{L} &P(V_{j\,|\,i} = 1 \,|\, \underline{x}_i, c_i, \Theta^{(t)}) \; \log P(V_{j\,|\,i} = 1 \,|\, \underline{x}_i, c_i, \Theta^{(t)}). \end{aligned}$$

When evaluated at $\Theta^{(t)}$, the lower bound will be exactly equal to the incomplete data log-likelihood, *i.e.*, $Q(\Theta^{(t)}, \Theta^{(t)}) = \log P(\mathcal{X}_l, \mathcal{X}_u \,|\, \Theta^{(t)})$.

2.4.4.1 Generalized M-step

In the generalized M-step, we update the parameters to $\Theta^{(t+1)}$ such that $Q(\Theta^{(t+1)}, \Theta^{(t)}) \geq Q(\Theta^{(t)}, \Theta^{(t)}) = \log P(\mathcal{X}_l, \mathcal{X}_u \,|\, \Theta^{(t)})$. A generalized M-step, rather than a single M-step, is required because of the complicated dependence of $Q(\Theta, \Theta^{(t)})$ on the prototype vectors and scale parameters. As mentioned earlier, the generalized M-step considers, in turn, different parameter subsets, and optimizes these given all other parameters in Θ held fixed. Each such optimization, and hence the sequence of such optimizations will be non-decreasing in $Q(\Theta, \Theta^{(t)})$, and hence also non-decreasing in $\log P(\mathcal{X}_l, \mathcal{X}_u \,|\, \Theta)$. Below, we specify the generalized M-step for Gaussian component densities, *i.e.*, $f_{\underline{X} \,|\, j}(\underline{x} \,|\, \theta_j) = \mathcal{N}(\underline{x}\,; \underline{\mu}_j, \mathbf{\Sigma}_j)$

Update of $\{\alpha_j\}$ *and* $\{\theta_j\}$*:*
It is straightforward to derive closed-form M-step updates for the mixture model parameters $\{\alpha_j, \forall j \in [L]\}$ and $\{\theta_j = (\underline{\mu}_j, \mathbf{\Sigma}_j),\ \forall j \in [L]\}$ which globally maximize $Q(\Theta, \Theta^{(t)})$, given all other parameters held fixed. These updates are given by:

$$\alpha_j^{(t+1)} = \frac{\sum_{i=1}^{N_u} P(V_{j\,|\,i} = 1 \,|\, \underline{x}_i, \Theta^{(t)}) + \sum_{i=N_u+1}^{N_u+N_l} P(V_{j\,|\,i} = 1 \,|\, \underline{x}_i, c_i, \Theta^{(t)})}{N_u + N_l}, \tag{2.24}$$

$$\underline{\mu}_j^{(t+1)} = \frac{\sum_{i=1}^{N_u} P(V_{j\,|\,i} = 1 \,|\, \underline{x}_i, \Theta^{(t)})\, \underline{x}_i + \sum_{i=N_u+1}^{N_u+N_l} P(V_{j\,|\,i} = 1 \,|\, \underline{x}_i, c_i, \Theta^{(t)})\, \underline{x}_i}{(N_u + N_l)\, \alpha_j^{(t+1)}}, \tag{2.25}$$

$$\begin{aligned} \mathbf{\Sigma}_j^{(t+1)} &= \frac{\sum_{i=1}^{N_u} P(V_{j\,|\,i} = 1 \,|\, \underline{x}_i, \Theta^{(t)})\, (\underline{x}_i - \underline{\mu}_j^{(t+1)})(\underline{x}_i - \underline{\mu}_j^{(t+1)})^T}{(N_u + N_l)\, \alpha_j^{(t+1)}} \\ &+ \frac{\sum_{i=N_u+1}^{N_u+N_l} P(V_{j\,|\,i} = 1 \,|\, \underline{x}_i, c_i, \Theta^{(t)})\, (\underline{x}_i - \underline{\mu}_j^{(t+1)})(\underline{x}_i - \underline{\mu}_j^{(t+1)})^T}{(N_u + N_l)\, \alpha_j^{(t+1)}}. \end{aligned} \tag{2.26}$$

Update of prototype vectors and scale parameters:
It is not possible to find a closed form solution for the prototype vectors $\{\underline{s}_l^{(j)},\ \forall l \in [N^{(j)}]\}$, $\forall j \in [L]$ which maximizes $Q(\Theta, \Theta^{(t)})$. This is also the case for the scale parameters $a_j,\ \forall j \in [L]$. Hence, we optimize these parameters via gradient ascent applied directly on the data log-likelihood $\log P(\mathcal{X}_l, \mathcal{X}_u \,|\, \Theta)$. The class affiliations of the prototype vectors $m_l^{(j)},\ \forall l \in [N^{(j)}]$,

$\forall j \in [L]$ are carefully initialized as discussed in section 2.4.5, but we do not perform a discrete optimization of these parameters in the generalized M-step. First, we optimize the prototype vectors with all other parameters (including the scale parameters) kept fixed at their current values, and subsequently optimize the scale parameters with all the other parameters (including the prototype vectors) fixed at their current values. Although the generalized M-step only requires that the updates on the parameter subsets be non-decreasing in the auxiliary function $Q(\Theta, \Theta^{(t)})$, we can also find parameter updates which are non-decreasing in the original data log-likelihood objective. Such an extension of the EM algorithm, called the *Expectation/Conditional Maximisation Either (ECME)* algorithm, has been shown to have faster convergence properties [106]. In order to ensure that the M-step updates of ECME monotonically increase the data log-likelihood, the only requirement is that the parameter updates which are based on the data log-likelihood be performed at the end, after all the parameter updates which are based on the auxiliary lower bound have been performed first [106], [145]. Indeed, the sequence of parameter updates proposed here satisfy this requirement.

2.4.5 Overall Learning Strategy

The GEM algorithm just specified assumes that both the number of prototypes and their class affiliations have already been chosen. Here, we describe an overall learning framework that integrates the EM steps specified above with a heuristic strategy for incrementally "growing" a suitable level of class posterior model complexity (neither too few, nor too many prototypes) within each mixture component. This framework consists of the following steps:

Step 1: Learn an MOE model for the data

We first learn a standard Gaussian mixture model, using the Bayesian Information Criterion (BIC) [152] to select the number of components. This is used to initialize the parameters for the EM algorithm in [122], applied to learn an MOE model.

Step 2: Single prototype per-class assignment

For each mixture component $j \in [L]$ we first allocate one prototype for every class $k \in \mathcal{C}$ that satisfies $\beta_{k\,|\,j} > \epsilon$ [7], *i.e.*, for every class at least partially represented by one or more labeled samples within the component. The prototype for class k is initialized with the weighted average of the feature vectors of all labeled samples with class label k, where the weight of a labeled sample $(\underline{x}_i, c_i)$ is its component posterior $P(V_{j\,|\,i} = 1 \,|\, \underline{x}_i, c_i, \Theta)$ based on the MOE model. Note that for a class-pure component ($\beta_{k^*\,|\,j} = 1$), this reduces to a single prototype being assigned, with this component essentially modeled according to hard-MOE [135], *i.e.*, $m_1^{(j)} =$

[7] We choose a small positive constant $\epsilon \ll 1$ that sets a noise level to determine the value of $\beta_{k\,|\,j}$ below which a prototype for class k (associated with component j) is not allocated.

k^*. We then optimize all the (just initialized) prototype vectors via gradient ascent applied to the incomplete data log-likelihood (2.18), while keeping all *other* model parameters fixed at their current values.

Step 3: Sequentially introduce additional prototypes

We would like to introduce additional prototypes where they are needed, within highly class-heterogeneous components. Accordingly, we propose a sequential model growing algorithm for adding prototypes and optimizing their locations. At each step, the algorithm considers, as new candidate (initialized) prototypes, the full set of labeled samples $\mathcal{X}_l$, and couples each such candidate $(\underline{x}_i, c_i),\ i = N_u + 1, \ldots, N_u + N_l$ with all mixture components "within its vicinity", *i.e.*, with all components $j \in [L]$ such that $P[V_{j\,|\,i} = 1 \,|\, \underline{x}_i, c_i, \Theta] > \tau$ [8]. In other words, we trial-create additional candidate prototypes $\underline{s}^{(j)}_{N^{(j)}+1} = \underline{x}_i,\ \ i = N_u + 1, \ldots, N_u + N_l$, where $m^{(j)}_{N^{(j)}+1} = c_i$, for all components $j \in [L]$ "in the vicinity" of $\underline{x}_i$ as specified above. For each candidate index pair (i, j), we evaluate the data log-likelihood (2.18) that results when the candidate prototype is included in the model, and we identify the best candidate index pair (i^*, j^*). To best evaluate this pair, we further perform trial-optimization of *all* the prototypes within component (j^*) (including this chosen best candidate initialized at the labeled sample), along with adjustment of the scale parameter a_{j^*} (which we discuss subsequently). We then evaluate the resulting candidate new model's BIC cost [152], which is the sum of the negative data log-likelihood (2.18) and the codelength (or model description length) required to specify all free parameters in the model, Θ [9]. The trial optimized candidate prototype is only accepted into the model if the new BIC cost (based on inclusion of the candidate prototype) is lower than the *current* model's BIC cost (without the inclusion of the candidate prototype). We specify the following codelength (or model description length) for the PFGL model consistent with unique decodability:

$$\begin{aligned} CC(L, \{N^{(j)}\}, \Theta) \;\; &= \;\; \frac{(L-1)}{2}\, \log(N_u + N_l) \;+\; \frac{1}{2} \sum_{j=1}^{L} \Gamma(\theta_j)\, \log(N_u + N_l) \;+\; \frac{L}{2}\, \log N_l \\ &+ \;\; \left(\frac{d}{2}\, \log N_l \;+\; \log |\mathcal{C}| + \log L \right) \sum_{j=1}^{L} N^{(j)}\,, \end{aligned}$$

[8] The value of $\tau \in (0, 1)$ determines how strongly a labeled sample $(\underline{x}_i, c_i)$ has to be associated with a component j, so that $\underline{x}_i$ may be considered as a candidate prototype initialization for component j (with class affiliation c_i).

[9] BIC can be interpreted using the Minimum Description Length (MDL) principle [63], [5] as a two part code-length, with the first part being the codelength required to specify the model parameters, and the second part being the codelength required to specify the data given the model. The codelength required to specify the model parameters should be efficient and consistent with unique decodability [63]. The codelength for specifying the data given the model is simply the negative log-likelihood of the data under the model. Similar approaches for specifying the model complexity can also been found in [59] and [124].

where the first term is the codelength for the component masses, the second term is the codelength for the free parameters in the component densities ($\Gamma(\theta_j)$ is the number of free parameters in θ_j [10]), and the third term is the codelength for the scale parameters $\{a_j\}$ in the RNP class posterior. In the last term, $\frac{d}{2}\ \log N_l$ is the codelength for each prototype vector (of dimension d), $\log|\mathcal{C}|$ is the codelength needed to specify the class of the prototype (assuming each class is equally likely with probability $1/|\mathcal{C}|$), and $\log L$ is the codelength needed to specify the mixture component with which the prototype is associated (assuming each component is equally likely with probability $1/L$). Note that the codelength of a single continuous valued parameter in the BIC framework is $\frac{1}{2}\ \log n$, where $n = N_l$ for the parameters of the RNP class posterior which explain only the labeled data, and $n = N_u + N_l$ for the mixture model parameters which explain both the labeled and the unlabeled data.

The trial optimization of the prototype vectors $\{\underline{s}_l^{(j^*)},\ l = 1, \ldots, N^{(j^*)} + 1\}$ is done via gradient ascent applied to the incomplete data log-likelihood (2.18), while keeping all *other* model parameters, including a_{j^*}, held fixed at their current values. Note that fixing a_{j^*} ensures non-zero gradients, so that the prototypes can be adjusted to new (locally optimal) locations, accommodating $\underline{s}_{N_{j^*}+1}^{(j^*)}$. The motivation for fixing the parameters $\{\alpha_j, \theta_j,\ \forall j \in [L]\}$ during this step is so that the learned components are not biased by the sequential fashion in which individual components acquire enhanced class posterior models. In particular, one can imagine two neighboring mixture components (A and B), which "own" labeled samples from the same class. If a prototype for this class is added to component A (before component B), a re-optimization of component A's parameters at this juncture might allow component A to "steal" from component B *both* labeled samples from this class as well as unlabeled samples in their vicinity, *i.e.*, samples which should "rightfully" belong to component B). Fixing all the component parameters in this step prevents this potential source of sequential learning bias.

As mentioned above, we first hold a_{j^*} fixed while optimizing the prototypes. Subsequently, we would like to trial "adjust" a_{j^*} to reflect the current level of uncertainty associated with the assignment of samples to prototypes within this component. As one reasonable way to measure this uncertainty, we suggest the following procedure. First, we model all the samples (probabilistically) associated with component j^* using an isotropic Gaussian mixture model, consisting of $N^{(j^*)}$ (sub-)components (assuming $N^{(j^*)}$ has already been incremented by one to reflect the addition of the candidate prototype), with the prototypes $\{\underline{s}_l^{(j^*)},\ \forall l \in [N^{j^*}]\}$ fixed as the sub-component *means*, and with unknown shared covariance matrix $\sigma_{j^*}^2\mathbf{I}$. We apply a simple EM algorithm on the (weighted) samples belonging to component j^* in order to estimate

[10] For a Gaussian mixture model with unrestricted component covariance matrices, $\Gamma(\theta_j) = d + d(d+1)/2$, which is the number of free parameters in the mean vector and the covariance matrix.

the variance $\sigma^2_{j^*}$, where the sample weights w_{ij^*} are given by the mixture component posteriors (2.21) and (2.22). Based on this mixture model within component j^*, we can calculate the weighted entropy of the (Gaussian mixture) sub-component posterior $\{P^{j^*}_{l\,|\,i},\ \forall l \in [N^{(j^*)}],\ \forall i \in [N_u + N_l]\}$, over all data points, via:

$$\overline{H} \ = \ - \sum_{i=1}^{N_u+N_l} w_{ij^*} \sum_{l=1}^{N^{(j^*)}} P^{j^*}_{l\,|\,i} \log P^{j^*}_{l\,|\,i}.$$

We can also calculate the weighted entropy of the within-component prototype posterior, over all data points, via:

$$H(a_{j^*}) \ = \ - \sum_{i=1}^{N_u+N_l} w_{ij^*} \sum_{l=1}^{N^{(j^*)}} \widehat{P}^{j^*}_{l\,|\,i} \log \widehat{P}^{j^*}_{l\,|\,i},$$

where

$$\widehat{P}^{j^*}_{l\,|\,i} \ = \ \frac{e^{-a_{j^*}\,||\underline{x}_i - \underline{s}^{(j^*)}_l||^2}}{\sum_{m=1}^{N^{(j^*)}} e^{-a_{j^*}\,||\underline{x}_i - \underline{s}^{(j)}_m||^2}}, \ \ \forall l \in [N^{(j^*)}],\ \forall i \in [N_u + N_l].$$

Finally, we use Newton's method to solve for a positive a_{j^*} such that $H(a_{j^*}) = \overline{H}$. Each adjustment of the scale parameter a_{j^*}, following the addition of a new prototype, tends to further increase a_{j^*}, which (properly) reflects the reduction in sample-to-prototype association uncertainty, as each new prototype is added to j^*'s class posterior model.

New prototypes are sequentially added, in this fashion, until the best candidate no longer reduces the BIC cost.

Step 4: Jointly optimize Θ:

Finally, after sequential within-component class posterior model growing is complete, we re-optimize the entire set of model parameters, Θ, based on the Generalized EM algorithm specified earlier. As we mentioned earlier, this optimization should (in principle) include a discrete optimization of the prototype-to-class assignments $\{m^{(j)}_l,\ \forall l \in [N^{(j)}]\}$, starting from the initial assignments specified in Steps 2 and 3. However, we did not perform this discrete optimization in this work.

2.4.6 Class inference

Class inference for the PFGL model is very similar to the second inference rule for NFGL, given in (2.16). Specifically, the class posterior for a new sample $\underline{x}_n$ is:

$$P(C_n = c \,|\, \underline{x}_n, \Theta) \;=\; \sum_{j=1}^{L} P(V_{n\,|\,j} = 1 \,|\, \underline{x}_n, \Theta)\; P(C_n = c \,|\, V_{n\,|\,j} = 1, \underline{x}_n, \phi_j), \qquad (2.27)$$

where $P(V_{n\,|\,j} = 1 \,|\, \underline{x}_n, \Theta)$ is the component posterior for the unlabeled samples given by (2.21). The class value which maximizes $P(C_n = c \,|\, \underline{x}_n, \Theta)$ is the class prediction for $\underline{x}_n$.

2.5 Using regularized covariance matrix estimates

When the dimension of the data (number of features) d is large relative to the number of samples $N_u + N_l$, the maximum likelihood estimates of the covariance matrices of the component Gaussian densities may not be well-conditioned (*i.e.*, close to singular) [11]. One way to handle this problem is to introduce weak data-dependent conjugate priors on the component covariance matrices, and find the maximum-a-posteriori (MAP) estimate of the component covariance matrices [128]. We will see that this effectively regularizes the covariance matrix estimates. For the covariance matrix of a multivariate Gaussian density, the Inverse Wishart density is a conjugate prior [136], which makes it analytically tractable to find the posterior distribution and the MAP estimates. We will only introduce conjugate priors for the component covariance matrices, and will assume that the other parameters have non-informative prior distributions, which ensure that the MAP estimates are equal to the maximum likelihood estimates.

Suppose the Inverse Wishart prior distribution for the covariance matrix of component $j \in [L]$ is given by

$$P(\mathbf{\Sigma}_j \,; \mathbf{S}_{j0}, \nu_{j0}) \;=\; \kappa_j \; |\mathbf{\Sigma}_j|^{-\frac{1}{2}\,(\nu_{j0}+d+1)} \; e^{-\frac{1}{2}\,tr(\mathbf{S}_{j0}\,\mathbf{\Sigma}_j^{-1})}, \;\; \forall j \in [L], \qquad (2.28)$$

where κ_j is the normalizing constant of the Inverse Wishart density and $tr(\cdot)$ denotes the trace of a matrix. The hyper-parameter $\mathbf{S}_{j0}$ is a $d \times d$ positive definite matrix, and the hyper-parameter $\nu_{j0} \geq d$ is usually called the degree of freedom of the Inverse Wishart density. We will discuss how to choose these hyper-parameters shortly. First, we find an expression for the MAP estimate of $\mathbf{\Sigma}_j$ for both the NFGL and the PFGL models. For the NFGL model, the objective function

[11] We may use a rule of thumb such as $d/(N_u + N_l) > 0.1$ to say that the dimension of the data is large relative to the number of samples. But, the problem of singular covariance matrices may also manifest for smaller values of this ratio.

for MAP estimation is $\log \widetilde{P}(\overline{\mathcal{X}}, \mathcal{C}_l \,|\, \Theta) + \sum_{j=1}^{L} \log P(\mathbf{\Sigma}_j \,; \mathbf{S}_{j0}, \nu_{j0})$, and for the PFGL model it is $\log P(\mathcal{X}_l, \mathcal{X}_u \,|\, \Theta) + \sum_{j=1}^{L} \log P(\mathbf{\Sigma}_j \,; \mathbf{S}_{j0}, \nu_{j0})$. To find the generalized M-step updates, the objective function is the lower bound $Q(\Theta, \Theta^{(t)}) + \sum_{j=1}^{L} \log P(\mathbf{\Sigma}_j \,; \mathbf{S}_{j0}, \nu_{j0})$, where $Q(\Theta, \Theta^{(t)})$ for the NFGL and PFGL models are given respectively by (2.11) and (2.23). It can be shown that the portion of this objective function which depends on $\mathbf{\Sigma}_j, \ \forall j \in [L]$ can be expressed as

$$g(\mathbf{\Sigma}_1, \ldots, \mathbf{\Sigma}_L) \ = \ -\frac{1}{2} \sum_{j=1}^{L} \log |\mathbf{\Sigma}_j| \, (\mathbb{N}_j + \mathbb{N}_{j0}) \ - \ \frac{1}{2} \sum_{j=1}^{L} tr((\bar{\mathbf{S}}_j + \mathbf{S}_{j0}) \, \mathbf{\Sigma}_j^{-1}), \quad (2.29)$$

where $\mathbb{N}_{j0} = \nu_{j0} + d + 1$; $\mathbb{N}_j$ is given by the following expressions for the NFGL and PFGL model respectively

$$\mathbb{N}_j \ = \ \sum_{i=1}^{N_u} P(V_{j\,|\,i} = 1 \,|\, \underline{x}_i, \Theta^{(t)}) \ + \ \sum_{i=N_u+1}^{N_u+N_l} v_{j\,|\,i}^{(t)}$$

and

$$\mathbb{N}_j \ = \ \sum_{i=1}^{N_u} P(V_{j\,|\,i} = 1 \,|\, \underline{x}_i, \Theta^{(t)}) \ + \ \sum_{i=N_u+1}^{N_u+N_l} P(V_{j\,|\,i} = 1 \,|\, \underline{x}_i, c_i, \Theta^{(t)}) \, ;$$

and $\bar{\mathbf{S}}_j$ is given by the following expression for the NFGL and PFGL model respectively

$$\begin{aligned} \bar{\mathbf{S}}_j \ &= \ \sum_{i=1}^{N_u} P(V_{j\,|\,i} = 1 \,|\, \underline{x}_i, \Theta^{(t)}) \, (\underline{x}_i - \underline{\mu}_j^{(t+1)})(\underline{x}_i - \underline{\mu}_j^{(t+1)})^T \\ + \ &\sum_{i=N_u+1}^{N_u+N_l} v_{j\,|\,i}^{(t)} \, (\underline{x}_i - \underline{\mu}_j^{(t+1)})(\underline{x}_i - \underline{\mu}_j^{(t+1)})^T \end{aligned}$$

and

$$\begin{aligned} \bar{\mathbf{S}}_j \ &= \ \sum_{i=1}^{N_u} P(V_{j\,|\,i} = 1 \,|\, \underline{x}_i, \Theta^{(t)}) \, (\underline{x}_i - \underline{\mu}_j^{(t+1)})(\underline{x}_i - \underline{\mu}_j^{(t+1)})^T \\ + \ &\sum_{i=N_u+1}^{N_u+N_l} P(V_{j\,|\,i} = 1 \,|\, \underline{x}_i, c_i, \Theta^{(t)}) \, (\underline{x}_i - \underline{\mu}_j^{(t+1)})(\underline{x}_i - \underline{\mu}_j^{(t+1)})^T . \end{aligned}$$

Here $\mathbb{N}_j$ is the probabilistic count of the number of samples belonging to component j, $\mathbb{N}_{j0}$ can

be taken to represent the prior count of samples belonging to component j, $\bar{\mathbf{S}}_j$ is the weighted scatter matrix of the samples belonging to component j, and $\mathbf{S}_{j0}$ can be taken to represent the prior scatter matrix for component j.

By equating the matrix derivative of $g(\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_L)$ with respect to $\boldsymbol{\Sigma}_j,\ j \in [L]$ to $\mathbf{0}$, we can find its stationary point, which is also its global maximum. This gives the modified M-step update for the covariance matrices (at iteration t) as

$$\begin{aligned} \boldsymbol{\Sigma}_j^{(t+1)} &= \frac{\bar{\mathbf{S}}_j + \mathbf{S}_{j0}}{\mathbb{N}_j + \mathbb{N}_{j0}} &= \frac{\mathbb{N}_j}{\mathbb{N}_j + \mathbb{N}_{j0}} \bar{\boldsymbol{\Sigma}}_j + \frac{\mathbb{N}_{j0}}{\mathbb{N}_j + \mathbb{N}_{j0}} \boldsymbol{\Sigma}_{j0} \\ &= (1 - \gamma_{j0})\, \bar{\boldsymbol{\Sigma}}_j + \gamma_{j0}\, \boldsymbol{\Sigma}_{j0}\,, \ \ \forall j \in [L]. \end{aligned} \tag{2.30}$$

In the above expression, $\bar{\boldsymbol{\Sigma}}_j = \frac{1}{\mathbb{N}_j} \bar{\mathbf{S}}_j$ is exactly equal to the M-step update of the covariance matrix in the maximum likelihood setting (given by (2.14) for the NFGL model and (2.26) for the PFGL model), $\boldsymbol{\Sigma}_{j0} = \frac{1}{\mathbb{N}_{j0}} \mathbf{S}_{j0}$ acts as a prior estimate of the covariance matrix, and $\gamma_{j0} = \frac{\mathbb{N}_{j0}}{\mathbb{N}_j + \mathbb{N}_{j0}} \in (0, 1)$. Thus, we see that $\boldsymbol{\Sigma}_{j0}$ effectively regularizes the covariance matrix estimate found by MLE, and the value of γ_{j0} controls the extent or strength of the regularization [128].

Having specified the regularized M-step updates for the component covariances, the next issue to be addressed is the choice of the hyper-parameters $\boldsymbol{\Sigma}_{j0}$ and ν_{j0}. Instead of specifying the value of ν_{j0}, we can equivalently specify $\mathbb{N}_{j0}$ or γ_{j0}. One approach is to use cross-validation, choosing the value of $\gamma_{j0} \in (0, 1)$ that maximizes the average data log-likelihood calculated over the held out test folds. An alternative approach which requires less computation is the Ledoit-Wolf shrinkage estimation [101], where a closed-form expression for γ_{j0} is found by minimizing the squared loss function. It is also possible to use Bayesian approaches to set the hyper-parameters, but we will not discuss these approaches here. For the prior covariance matrix, it is common to use the data dependent choice $\boldsymbol{\Sigma}_{j0} = diag(\boldsymbol{\Sigma}_j^{(\mathrm{ml})})$ [128] (or equivalently $\mathbf{S}_{j0} = \mathbb{N}_{j0}\, diag(\boldsymbol{\Sigma}_j^{(\mathrm{ml})})$), where $\boldsymbol{\Sigma}_j^{(\mathrm{ml})}$ is the maximum likelihood estimate of $\boldsymbol{\Sigma}_j$. For problems where d is large relative to $N_u + N_l$, the MLE itself may be close to singular (which is the reason why regularization is required in the first place) if the covariance matrix is unrestricted. For the purpose of choosing $\mathbf{S}_{j0}$, we can impose some restrictions on $\boldsymbol{\Sigma}_j$, *e.g.*, restrict it to be diagonal or use a shared (but otherwise unrestricted) covariance matrix for all the components, and then find the MLE based on just the unlabeled data using a standard EM algorithm for the mixture of Gaussians.

2.6 Relationship to Some Previous Methods

Our fine-grained methods "propagate" class label information from labeled to unlabeled samples for the case where a mixture model distribution assumption for the feature vector is reasonable, but where it may be unreasonable to assume that the mixture components (clusters) are class-pure. In the sense of performing label propagation, our methods are loosely related to methods that perform label propagation on graphs. Early such methods include [161] and [183]. Many subsequent graph-based semi-supervised learning methods are discussed in [182]. Most of these methods are particularly motivated by problems where the data points from the individual classes lie on an underlying low dimensional manifold (which can be well-captured by a global graph defined on all the data samples). By contrast, the proposed fine-grained modeling methods are motivated by domains where the data points are well-modeled by a mixture density, and where the distribution of classes within components can be heterogeneous (a non-trivial function of the feature vector). The NFGL approach effectively defines an MRF with seperate, fully connected sub-graphs (cliques) over the class labels of labeled samples that are generated from the same mixture component. The method in [186] also combines graph-based semi-supervised learning with mixture modeling. However, their method is quite different from ours. First, their model is not generative, but rather discriminative – their mixture model is mainly used to reduce the complexity of class inference on the (global) graph defined on all the data points. Second, they method defines a global graph, while the NFGL method effectively defines separate sub-graphs for each component/cluster. Third, while their approach is motivated by (and best suited for) domains where the data points have an underlying manifold structure, our methods are best suited for domains well-modeled by mixture distributions. The approach in [179] is also related to the current work in that the inclusion of must-link and cannot-link constraints into the mixture modeling entails optimization of a Markov random field potential function. Also, as discussed in section 2.2, the current work derives motivation from and builds on the limitations of the previous methods [122], [135], [154], extending them to achieve fine-grained within-component class labeling.

2.7 Experimental Results

2.7.1 An Illustrative Example for NFGL

Figure 2.5 shows an illustrative two-dimensional example for NFGL, involving two ground-truth Gaussian components and two classes, but with an XOR-like class structure embedded within these two components. Note that a two-component mixture solution based on [135] will be

necessarily poor since this method is restricted to learning class-pure components. The method in [122] can learn the ground-truth mixture components, but only a crude class proportions model within each mixture component. Figure 2.5 shows the BIC-selected 2-component NFGL solution, which accurately captures both the ground-truth component structure and the within-component class structure present in the data. Labeled samples are denoted by their class (1 or 2), and the shaded part is used to delineate the class decision boundary learned by NFGL.

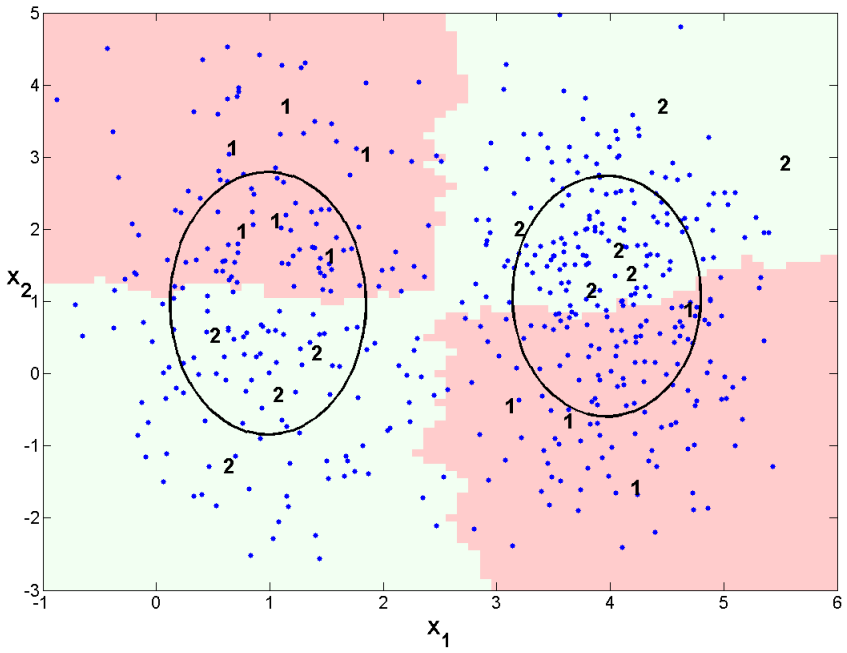


Figure 2.5. A two-dimensional semisupervised example with two ground-truth Gaussian components and an XOR-like class structure. Note that the NFGL solution captures both the ground-truth mixture components and the XOR class structure in the data.

2.7.2 Experimental Protocol

We evaluated the classification accuracies of our nonparametric fine-grained labeling method (NFGL, section 2.3) and parametric fine-grained labeling method (PFGL, section 2.4), in comparison with a number of conventional semi-supervised and supervised learning methods. For the NFGL method, we have found (unsurprisingly) that the simpler, second class inference rule (2.16), which treats a test sample as part of the unlabeled data subset, has better classification performance. A possible reason why the first class inference rule does not have comparable performance is because it uses the pseudo-likelihood approximation to find a tractable expression for the class posterior. While the pseudo-likelihood is a commonly used framework for param-

eter estimation, it is not commonly used for inference tasks. Also, the first class inference rule (2.15) involves a product of terms over all labeled samples, which may lead to probabilities close to zero at low labeled fractions (as discussed in section 2.4). Accordingly, we have used the class inference based on (2.16) in all of our experiments.

The methods we have compared with include the following semi-supervised learning methods: the mixture of experts classifier (MOE) [122], the specialization of MOE with hard component-to-class assignments (MOE-hard) [135], and within-component nearest neighbor (WC-NN) (section 2.2). We also evaluated a variant of MOE-hard, which we call *MOE-hard-plus*. This method is essentially the same as the MOE-hard method, but with the number of components increased until the BIC cost of MOE-hard first exceeds the BIC cost of PFGL [12]. Since the PFGL method has a parametric within-component class posterior which allows it a greater model complexity, in this way we allow for a roughly fair model complexity comparison between MOE-hard-plus and PFGL. We also compared with the following purely supervised learning methods: linear and Gaussian kernel support vector machines (SVMs) and K-nearest-neighber (KNN) classification.

We used eight real-world data sets from the UC Irvine (UCI) machine learning repository [1], as summarized in Table 3.1. Moreover, the general trends in the results we will present here (on these eight data sets) are representative of what we have seen in experimentation on a larger set of data sets from the UC Irvine machine learning repository. For some of the data sets, there are a few categorical-valued features. We did not use these in the experiments in order to make the mixture modeling less complicated.

Table 2.1. Summary of the data sets used in experiments.

Data set	Number of instances	Number of attributes used	Number of classes
Yeast	1484	6	10
Waveform database generator	5000	21	3
Image Segmentation	2310	16	7
Liver Disorders	345	6	2
Pima Indians Diabetes	768	8	2
Ecoli	336	5	8
Breast Cancer Wisconsin	569	30	2
Page Blocks Classification	5473	10	5

[12] For MOE-hard-plus, the codelength needed to specify the model parameters, as part of the BIC cost, includes the specification of the class to which each component is hard-assigned. This codelength is $\log N_c$ for each component, assuming that each class is equally likely.

We learned the models using a training set and evaluated performance on a separate test set. Except for the *Image Segmentation* data set, for which a training-test split is already specified in its UCI data description, for all the other UCI data sets we performed a random 50% split into training and test subsets. The same split was used for all the methods. In each experiment with training/test sets fixed, we varied the size of the labeled training subset (N_l) from 2% (*i.e.*, $N_l = 0.02\,(N_u + N_l)$) up to 24% with 2% steps. For some data sets, where the number of data instances is small, we chose to start from a 4% labeled training subset size. For a given labeled training subset size (N_l), we performed ten random selections of labeled/unlabeled training subsets, with the same labeled/unlabeled training subsets used for all the methods. The semi-supervised learning methods used all the data (including the unlabeled training and test samples) for learning, while the supervised learning methods (SVMs and KNN) only used the labeled training subset for learning. All the methods then made decisions on the test split. Results were averaged over the ten random "trials", yielding an average test set classification error rate for a given labeled training subset size.

Parameter Initialization and implementation details:
We first estimated the parameters of a Gaussian mixture model (GMM) by standard unsupervised (excluding the class labels) maximum likelihood estimation using the EM algorithm. This EM algorithm is initialized using the hard clustering solution found by the K-means algorithm (with cluster centroids randomly initialized using points from the training set). The GMM parameters estimated in this way by the EM algorithm are used to initialize the mixture model based semi-supervised learning methods. We considered using full covariance matrices for the Gaussian components. However, on a number of the data sets we observed that the problem of singular covariances occurred during EM learning. Moreover, we also noticed that, in some cases, quite low GMM model orders were chosen (model order selection is discussed further in the sequel) when full covariances were used. In order to avoid these problems, we restricted to using diagonal covariances, on all the data sets. We acknowledge that overall better accuracy may be achievable if use of diagonal vs. full covariances is decided separately for each given data set/domain.

For the NFGL method, the assignment of labeled samples to components $\{v_{j\,|\,i},\ \forall j \in [L];\ i = N_u + 1, \ldots, N_u + N_l\}$ are also model parameters. In order to initialize them, we performed a hard clustering on the labeled sample feature vectors using the initial GMM parameters. If a labeled sample feature vector $\underline{x}_i$ is assigned to component j, then $v_{j\,|\,i}$ is initialized to 1 and $v_{k\,|\,i},\ \forall k \neq j$ are initialized to 0. The scale parameters $\{a_j,\ \forall j \in [L]\}$ in NFGL were all initialized to the reciprocal of the average-squared-distance between all pairs of labeled samples. For the MOE method, the component-conditional class PMF was initialized as uniform over all

classes, *i.e.*, $\beta_{c\,|\,j} = 1/|\mathcal{C}|,\ \forall c \in \mathcal{C}, \forall j \in [L]$.

For the NFGL method, an undesirable phenomenon that can occur during unconstrained cyclical optimization of the $\{v_{j\,|\,i}\}$ parameters is as follows. As discussed in section 2.4.1, when a mixture component has singleton labeled samples from one or more classes, the randomized nearest neighbor class posterior (2.5) evaluates to zero probability for those classes. During the cyclical optimization of the $\{v_{j\,|\,i}\}$ parameters, such components could try to "steal" labeled samples from other components in order to have non-zero probabilities and increase the pseudo-log-likelihood. This undesirable phenomenon is avoided by preventing labeled samples from being assigned to such components during the cyclical optimization. Also, labeled samples belonging to such components are not allowed to switch to other components. For PFGL, the constants ϵ and τ (discussed in steps 2 and 3 of section 2.4.5) were both set to a small value (0.001) in our experiments. In practice, more generally, one should choose ϵ in a component-specific fashion. Specifically, it is reasonable to impose that every class c which is assigned a prototype within component j should have $\beta_{c\,|\,j}\,\hat{M}_j > \frac{1}{2}$, where $\hat{M}_j$ is a probabilistic (soft) count of the number of labeled samples belonging to component j (based on the mixture model component posterior of samples) – *i.e.*, at least "half a labeled sample" from class c should belong to component j if class c is to be allocated a prototype in component j. Regarding τ, we have observed that, so long as it is chosen sufficiently small that viable (prototype, component) associations are not rejected, it does not have much impact on the model "growing" process.

We investigated the classification performance of SVMs using both linear and Radial Basis Function (RBF/Gaussian) kernels. The support vector classification was performed using the integrated software LIBSVM (Version 3.1) [31]. A linear SVM model has a hyperparameter, C, that controls the degree of margin slackness. An RBF kernel SVM has both C and a hyperparameter γ, the scale of the Gaussian kernel. For each labeled subset size and training set realization, we chose these hyperparameters via a cross validation procedure, as detailed next. For the linear SVM, we varied C over the interval $[0.1, 10000]$ and performed 10-fold cross validation on the labeled training subset (to be precise the number of folds is $\min\{10, N_l\}$), selecting the C value from a piecewise uniform grid of 334 candidate values. According to the suggestions in the LIBSVM manual [31], the search step size s was set as follows: for $0.1 \leq C < 1$: $s = 0.1$, for $1 \leq C < 10$: $s = 1$, for $10 \leq C < 100$: $s = 2$, for $100 \leq C < 1000$: $s = 10$, and for $1000 \leq C \leq 10000$: $s = 50$. For the RBF kernel SVM, we performed 10-fold cross validation via a 2D grid search to select the parameters (C and γ). The range of search for C is the same as that for a linear SVM. The second parameter γ was given range $[0.001, 10]$ with search step size s_γ set as follows: for $0.001 \leq \gamma < 0.01$: $s_\gamma = 0.002$, for $0.01 \leq \gamma < 0.1$: $s_\gamma = 0.02$, and for $0.1 \leq \gamma \leq 10$: $s_\gamma = 0.2$. We picked the grid point (C, γ) that gave the highest cross validation

accuracy and used these hyperparameters within the final SVM training. For data sets that have more than 2 classes, we applied "one-against-one" multi-class classification as implemented in the LIBSVM software.

The hyperparameter K in the KNN classifier was chosen using 10-fold cross-validation ($\min\{10, N_l\}$, to be precise) applied to the training set. The range of K was set to $\{2, \ldots, 8\}$ when $N_l > 400$, and $\{1, \ldots, 5\}$ for smaller values of N_l. The WC-NN classifier was implemented as described in section 2.2.

Model Order Selection:

On a given data set, for all the mixture model based methods (excepting MOE-hard-plus), the same model order (number of components) was used. We chose this number of components upfront using the Bayesian Information Criterion (BIC) [152] applied to the initial maximum likelihood solution of the GMM, which is based on the combined set of training and test data feature vectors [13]. For *MOE-hard*, however, the number of components cannot be chosen smaller than the number of classes in the data set (since, otherwise, some classes will not be represented at all by the model). Therefore, for this method we chose the model order to be either the BIC-selected order or the number of classes, whichever is larger. Table 2.2 gives the number of components selected by BIC, and also the range of the number of components used by the method *MOE-hard-plus* for all the data sets. We acknowledge that the model complexities of some of the mixture-based methods are not *precisely* the same (as the different methods apply different models for the within-component class posterior). For example, PFGL uses a more complex class posterior model than MOE, which in turn uses a more complex class posterior model than MOE-hard. Furthermore, one must distinguish *model complexity* from, *e.g.* computational complexity of class inference. The NFGL model may form a complex within-component decision boundary when the labeled fraction is relatively high, yet without any increase in the number of model parameters. PFGL, on the other hand, must increase model complexity (add prototypes) to achieve more "fine-grained" class modeling within the components. We are specifically evaluating the method MOE-hard-plus because this allows a complexity-wise fair comparison between one of our methods (PFGL) and a conventional method (MOE-hard) [14].

[13] Note that using the test data feature vectors in this way for selecting the number of components does not provide any unfair advantage for the semi-supervised mixture model based methods over the other methods used for comparison.

[14] Note, though, that MOE-hard-plus has a degree of freedom to optimize (the number of components), which was not exercised for PFGL.

Table 2.2. Summary of number of components used for all semisupervised methods

Data set	Number of components	Component range for MOE-hard-plus
Yeast	5 (10 for MOE-hard)	7 - 14
Waveform database	9	11 - 13
Image Segmentation	13	15 - 18
Liver Disorders	4	5 - 6
Pima Indians	6	7 - 10
Ecoli	3 (8 for MOE-hard)	3 - 6
Breast Cancer	10	12 - 14
Page Blocks	10	13 - 18

2.7.3 Classification Accuracy Evaluation

The average test set error rates of our proposed methods methods and the methods compared, on the UC Irvine data sets (listed in Table 3.1), are shown as a function of the labeled training subset size (expressed as a percentage of the total number of training samples) in Fig. 2.6 to Fig. 2.13. From the error rate curves, we can make the following observations:

- Although there is some variability in relative performance of methods across the data sets, PFGL and NFGL both achieve overall performance improvement over MOE, MOE-hard, and WC-NN classification. Moreover, while KNN performs better on a few data sets, the PFGL method (which generally outperformed NFGL) performed overall better than KNN.

- NFGL and PFGL performed better than both the supervised linear and RBF SVMs at low labeled fractions on most data sets. Note that, unlike the SVMs (and KNN), NFGL and PFGL do not use any hyperparameter tuning focused on improving classification accuracy – presumably, their performance could be further improved through such tuning (*e.g.*, by selecting the number of components via cross validation). Note also that MOE performance is comparable to PFGL and NFGL at the lowest labeled fraction on some data sets. This is not surprising if, at this low fraction, there is essentially one labeled sample per component – in such a case, "fine-grained labeling" and MOE both specialize to representing a component using a single class.

- The linear SVM gives comparable results to the RBF kernel SVM on these data sets.

- MOE performs overall better than MOE-hard. MOE-hard performance is best on *Ecoli*, where it used 8 components (one per class), whereas the other mixture-based methods (excluding MOE-hard-plus) used only 3. The MOE-hard-plus method overall performs

better than both MOE and MOE-hard. Moreover, notably on the Waveform data (and for certain labeled subset sizes on the Page Blocks data), the use of a significant number of extra components allows it to outperform the fine-grained labeling methods.

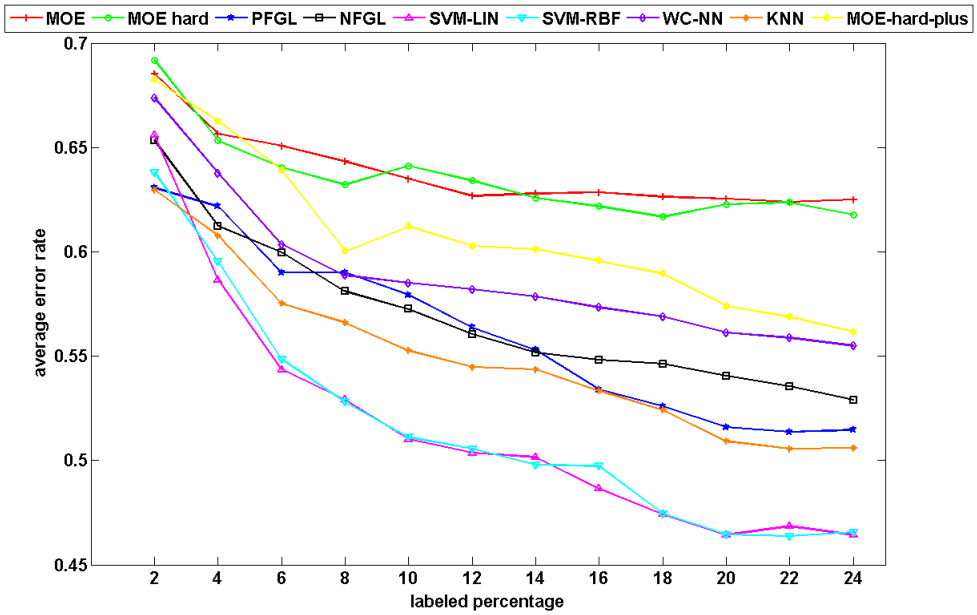


Figure 2.6. Average test error rate on the Yeast data set.

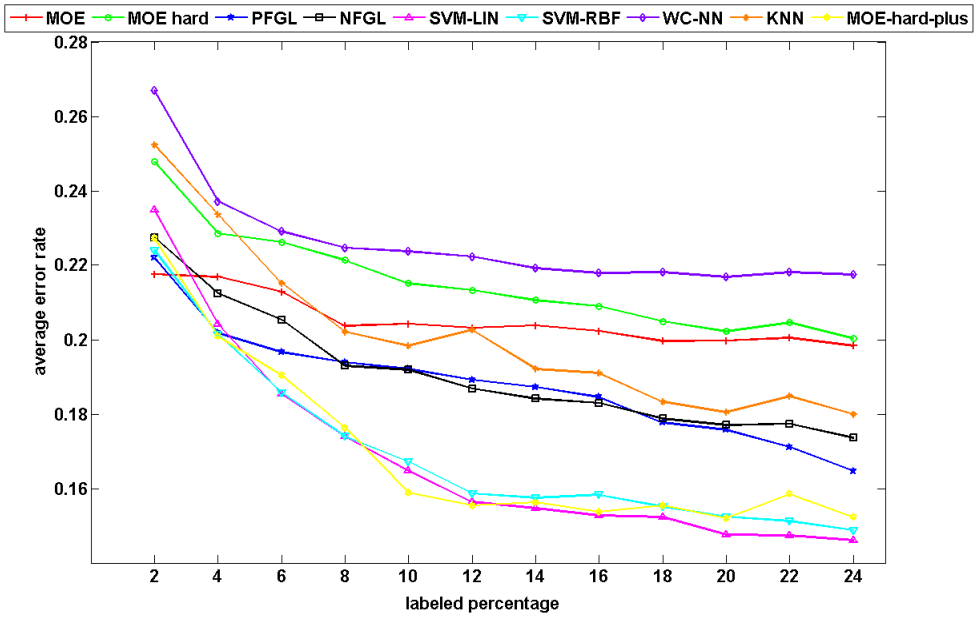


Figure 2.7. Average test error rate on the Waveform database generator data set.

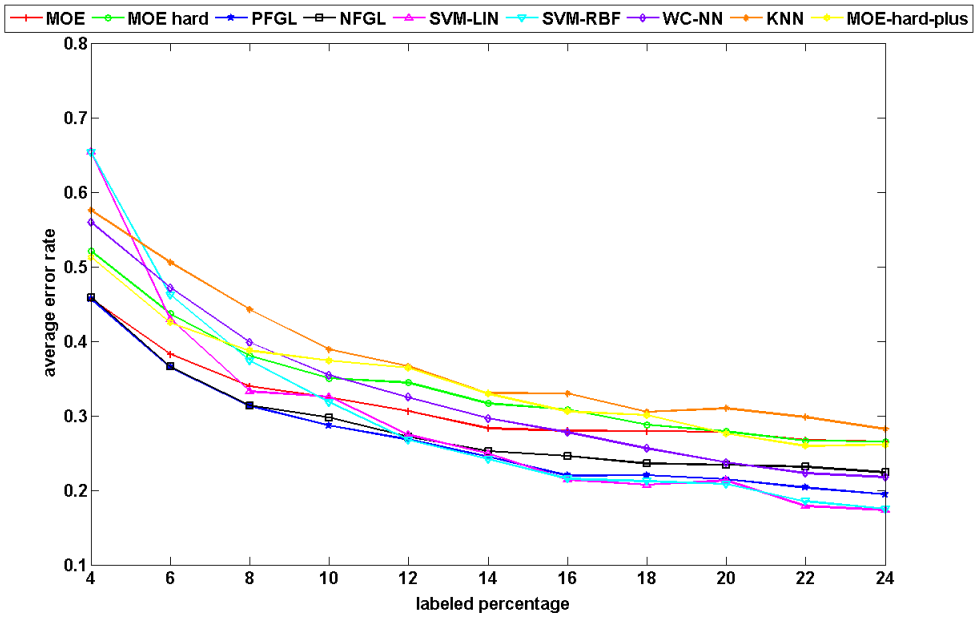


Figure 2.8. Average test error rate on the Image segmentation data set.

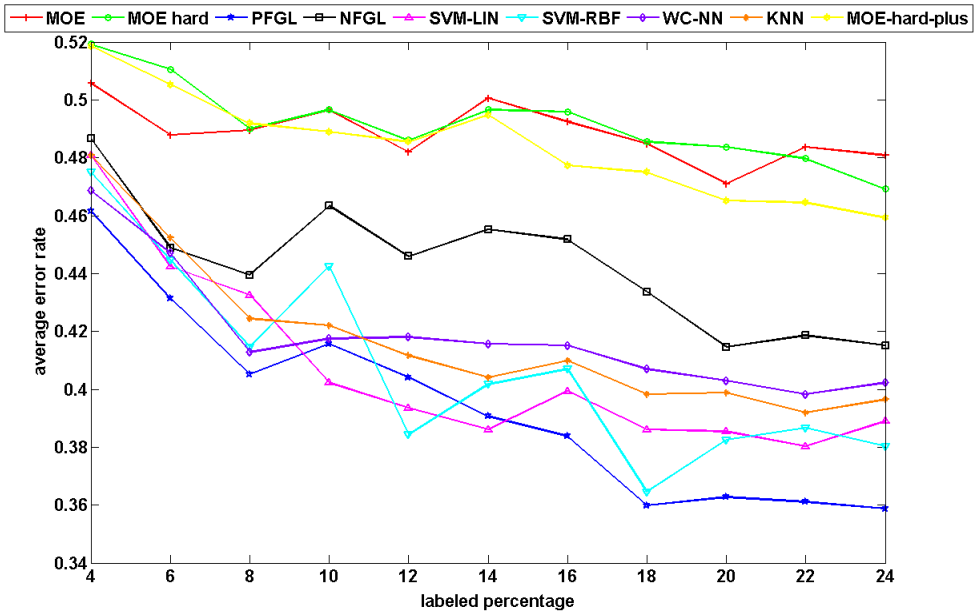


Figure 2.9. Average test error rate on the Liver disorders data set.

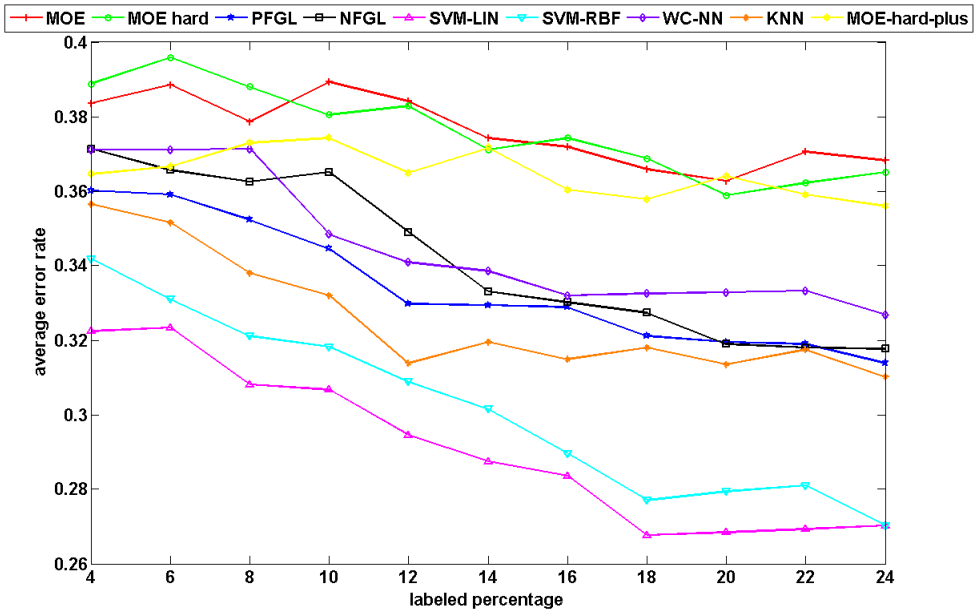


Figure 2.10. Average test error rate on the Pima Indians data set.

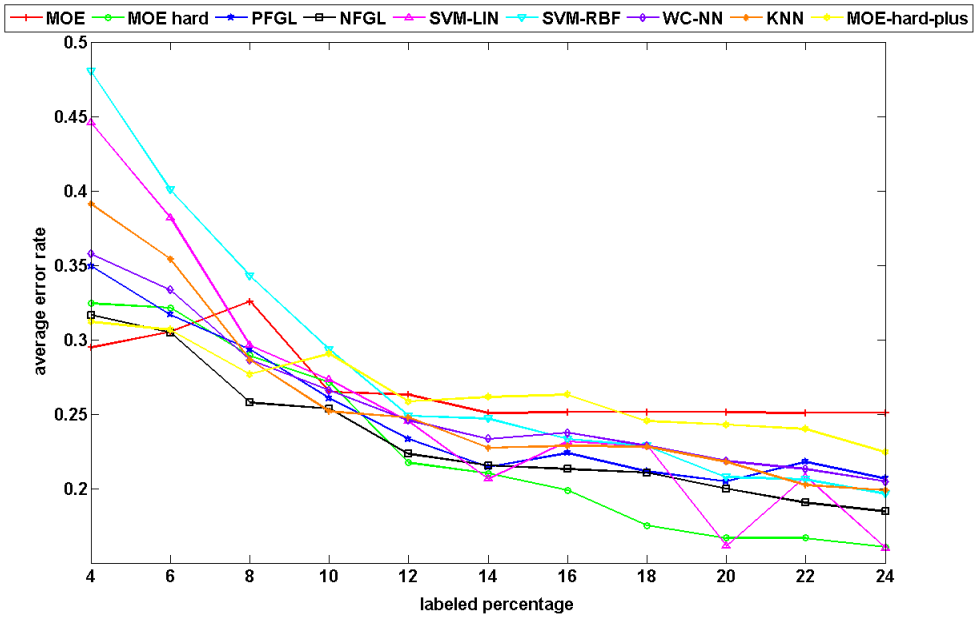


Figure 2.11. Average test error rate on the Ecoli data set.

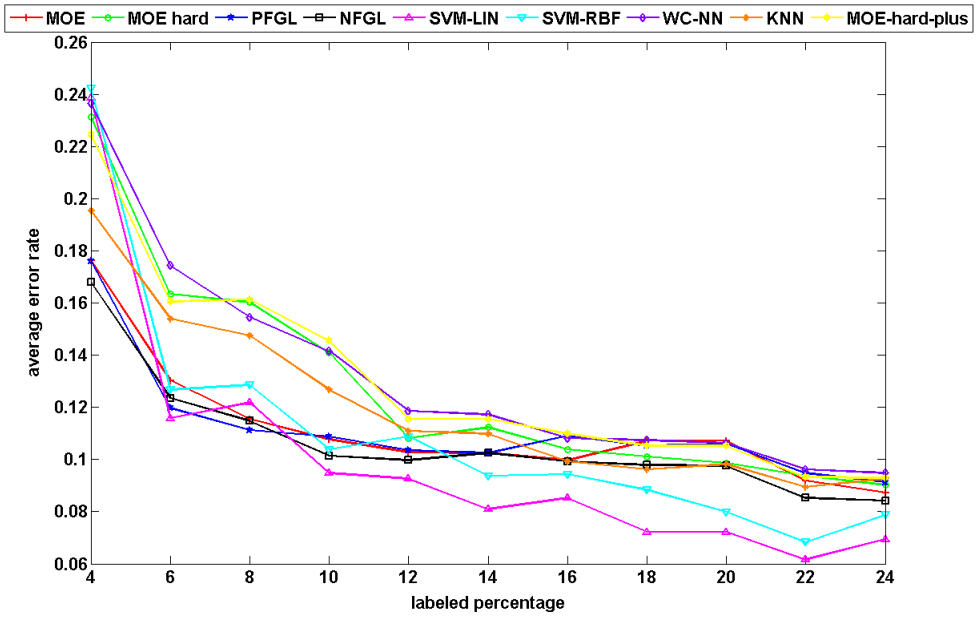


Figure 2.12. Average test error rate on the Breast cancer data set.

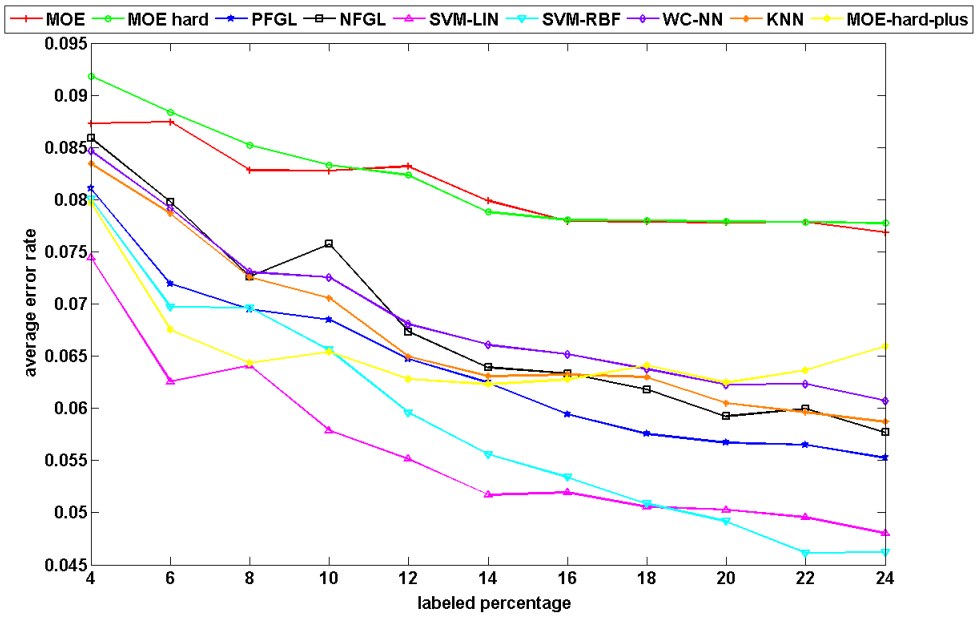


Figure 2.13. Average test error rate on the Page blocks data set.

Chapter 3

Semi-supervised mixture model based learning from pairwise-sample constraints with imposed space-partitioning

In this chapter, we present a new method for semi-supervised learning from pairwise sample (must-link and cannot-link) constraints. It addresses an important limitation of many existing methods, whose solutions do not achieve effective propagation of the constraint information to *unconstrained* samples. We overcome this limitation by constraining the solution to comport with a smooth (soft) class partition of the feature space, which *necessarily* entails constraint propagation and generalization to unconstrained samples. This is achieved via a parameterized mean-field (variational) approximation to the posterior distribution over component assignments (given the data and parameters), with the parametrization chosen to match the representation power of the chosen generative mixture density family. Unlike many existing methods, our method flexibly models classes using a variable number of components, which allows it to learn complex class boundaries. Also, unlike most methods, our method estimates the number of latent classes present in the data. Experiments on a number of synthetic data and real datasets show that, overall, our method achieves significant improvements in classification performance compared to a number of existing methods.

3.1 Introduction

As discussed in chapter 1, the supervision information in semi-supervised learning may come either in the form of class labels or instance-level constraints. *Pairwise sample constraints* are instance-level constraints, wherein a pair of samples is either known to belong to the same class (must-links (MLs)), or known *not* to belong to the same class (cannot-links (CLs)) [167], [90], [46]. Such constraints do not explicitly specify the class membership of any of the samples. Nor do they necessarily explicitly determine even the *number* of classes actually represented in the data. We assume MLs satisfy transitive closure, *i.e.*, if pairs of points (a, b) and (b, c) are must-linked, then (a, c) is also must-linked. Applying transitive closure, we can find groups of points, called *chunklets* [155], all belonging to the same class. Note that distinct chunklets A and B *could* belong to the same class. However, without MLs or CLs between a pair (a, b), $a \in A$ and $b \in B$, the constraints neither ensure nor exclude this. Finally, if there is a CL between points $a \in A$ and $b \in B$, then CLs are entailed between *all* pairs (a', b'), $a' \in A$ and $b' \in B$.

It can be argued that pairwise sample constraints may be more readily available than labels in practice, since they do not require any prior knowledge of the underlying classes, and since they may require less expertise to elicit than class labels. In an interactive learning setting, users can provide feedback on whether a pair of samples should or should not belong to the same group, without having actual knowledge of the category space [39], [46], [174], [155]. In a distributed learning scenario, the task of annotating a large database can be divided among a set of uncoordinated teachers, where each teacher annotates a subset of the database [4]. The class definitions/label conventions used by different teachers may not be coordinated, which precludes the subsets from being directly pooled into a single *labeled* database. However, labels on samples imply ML and CL constraints on all labeled sample pairs within each subset. This does allow the subsets to be pooled into a single database with supervision in the form of pairwise sample constraints.

There are several different practical objectives in learning from a data set that possesses instance-level constraints:

1) Statistical classification: One may wish to learn a statistical model for the latent groups present in the dataset, viewing the given data and constraints as a training set. The model can then be used to classify new (test) samples.

2) Clustering: One may view the problem ostensibly as an *unsupervised* clustering problem, but one in which there are additional constraints (side information) available, to guide the search for clustering structure in the data set.

3) Class and cluster number estimation: Based on the supplied data and constraints, one may wish to estimate the number of classes present in the data (and, if multiple clusters or components

are needed to represent each class, the number of clusters present).
The modeling framework developed here is applicable to all of the above objectives, but with special focus on 1).

3.1.1 Approaches for semi-supervised learning with instance-level constraints

In this section, we briefly review several general frameworks that have been proposed for semi-supervised learning from instance-level constraints. A more detailed literature survey, focusing on the relationship between prior works and our proposed work, will be given later in section 3.3. One vein of research modifies hard clustering (*e.g.*, K-means) or "soft clustering" (mixture modeling) methods to incorporate pairwise sample constraint information. In the case of hard clustering, this is done, for instance, by modifying the cluster assignment step of the K-means algorithm [167], or by adding a constraint violation penalty term to the clustering distortion objective (in some cases also combined with learning a distance metric) [15], [6]. In the case of mixture modeling, this is done by modifying the stochastic data generation mechanism of the mixture model (and hence the data log-likelihood objective) by conditioning on the pairwise constraint information [155], by adding a constraint violation penalty term to the mixture model negative complete data log-likelihood [179], or by using a prior distribution on the mixture model component assignment of data points such that the prior penalizes violation of the pairwise constraints [107]. A weakness of all of these methods (elaborated in detail in the sequel) is that they generally require *many* constraints in order for the constraints to affect the clustering solution, i.e. in order to achieve *constraint propagation* in the solution. Also, strict constraint satisfaction [167], [155] may not be desirable because i) the assumed cluster shape may not allow the satisfaction of all the constraints and ii) constraints may be noisy or inconsistent.

A second commonly applied framework is *metric learning* [90], [170], [4], wherein a non-isotropic ("distorted") distance metric is learned from the pairwise sample constraints, such that the distances between must-linked points are decreased, and between cannot-linked points are increased. The learned distance metric is subsequently used by an unsupervised clustering algorithm. In [15] and [6] both the metric learning and the clustering steps are integrated by minimizing a single objective function. While a majority of these methods focus on learning a linear Mahalanobis distance metric, some recent methods learn non-linear metrics using feature transformations and the kernel trick [158], [173], [47]. Metric learning methods also suffer from some limitations. First, the heuristic two-stage learning procedure (common to most metric learning methods) does not make use of the unconstrained data samples for learning the metric. Also, there is no further adaptation of the learned metric in light of the quality of the learned clusters, *i.e.*, there is no *joint* optimization of the metric and the clusters. Two-stage learning is

thus susceptible to finding poor solutions not even locally optimal in a well-defined sense. Moreover, the learned distance metric is typically *global*. This means that all the clusters are restricted to have the same shape, which may restrict the flexibility of these methods in well satisfying the constraints. Learning global non-linear metrics can mitigate this problem to some extent.

There are some methods which cannot be categorized into one of the aforementioned frameworks. For instance, [108] addresses the problem in a discriminative setting using Gaussian processes, and [105] learns a kernel matrix by solving a semi-definite programming problem, followed by the kernel K-means algorithm.

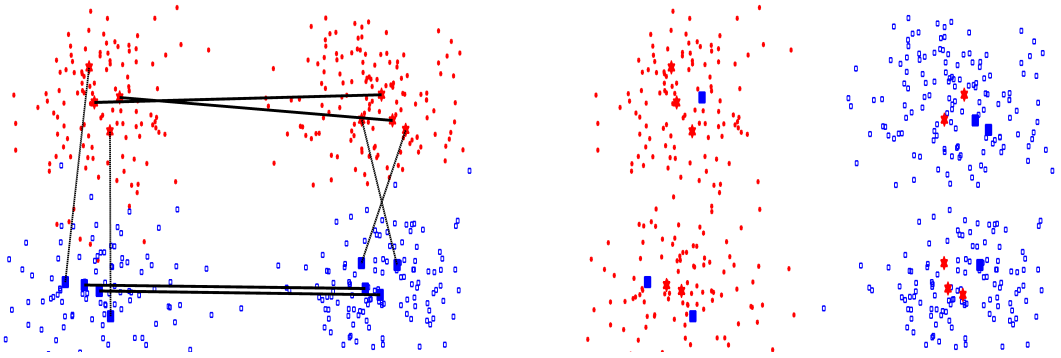
3.1.2 General limitations of prior works

We next elaborate on fundamental limitations common to many prior works.

3.1.2.1 Constraint propagation, solutions smoothness, and generalization

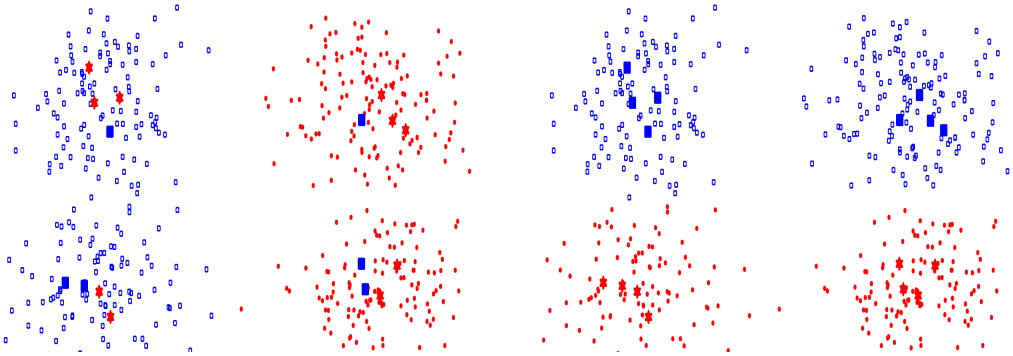
Perhaps the most important limitation of nearly all previous (hard or soft) clustering-based works is that they do not efficiently "propagate" constraint information over the given feature space. To be concrete, suppose that there is an ML constraint between a pair of data samples $\underline{x}_a$ and $\underline{x}_b$. Consider a third *unconstrained* point $\underline{x}_c$ that is very close to $\underline{x}_b$ ($\underline{x}_c$ and $\underline{x}_b$ could even be co-located). An ML constraint between $\underline{x}_a$ and $\underline{x}_b$ means these two samples should be assigned to the *same* group (class). Moreover, while there is no constraint provided, we would certainly expect that if $\underline{x}_c$ is very close to $\underline{x}_b$ or, in the most defining case, if $\underline{x}_c = \underline{x}_b$, these samples should all be assigned to the same group. However, in many of the existing instance-level constraint-based approaches [167], [155], [107], [179], [140], there is neither necessity nor *any* surety that these samples will be assigned to the same group. Moreover, this is *not* due to the methods finding poor local optima of their proposed training objective functions. Solutions that do *not* assign $\underline{x}_c$ to the same group as $\underline{x}_a$ and $\underline{x}_b$ may in fact be *globally optimal* with respect to the learning objective functions proposed in these works. In particular, as will be fully developed in the sequel (*cf.*, section 3.2.2.1), for these methods, the optimal (objective function minimizing) class posterior probability for unconstrained data point $\underline{x}_c$ has *no dependence* on the constraint information $\mathcal{I}_{\mathcal{C}}$, *i.e.*, $P[C = c \,|\, \underline{x}_c, \mathcal{I}_{\mathcal{C}}] = P[C = c \,|\, \underline{x}_c]$. This means that the class memberships of neighboring or even *co-located* constrained samples are ignored in assigning class membership probabilities to $\underline{x}_c$.

To appreciate the consequences for the learned model solution, consider the 2-dimensional data set with two classes shown in Fig. 3.1. Eight pairwise constraints (four MLs and four CLs) and the (latent) true class labels of all samples are shown in Fig. 3.1(a). The group assignments of samples obtained using the methods in [179] (with a single component per class) and [155]



(a) Data with true class labels and pairwise constraints

(b) Solution of method in [179]



(c) Solution of method in [155]

(d) Solution of our space-partitioning method

Figure 3.1. Two class synthetic data set with four ML constraints and four CL constraints, with the group assignments of samples obtained using the methods in [179] (with single component per class), [155], and our method shown in Fig .3.1(a) through Fig .3.1(d) respectively. The true class labels of samples are shown in Fig .3.1(a) with different symbols and colors. The ML constraints are shown with a solid line and the CL constraints are shown with a dotted line. The samples involved in constraints are shown with larger symbols for clarity.

are shown in Fig. 3.1(b) and Fig. 3.1(c) respectively. The group assignments obtained using our space-partitioning method are shown in Fig. 3.1(d). Although the solutions of [179] and [155] satisfy all the constraints, they are inaccurate (as seen by comparing with 3.1(a)). This is due to the fact that there is essentially no propagation of constraint information (or generalization) to the unconstrained samples, which can be observed from the discrepancy in group assignments between some of the constrained samples and their neighboring unconstrained samples in Fig. 3.1(b) and Fig. 3.1(c). These can also be described as group membership *discontinuities* at some of the constrained sample locations. On the other hand, the solution in Fig. 3.1(d) not only satisfies the constraints, but finds an accurate and smooth class partitioning of the whole data set. One way to summarize this example is by stating that the solutions in Fig. 3.1(b) and Fig. 3.1(c)

violate the semi-supervised smoothness assumption [33][1]. Another is to say that the solutions in Fig. 3.1(b) and Fig. 3.1(c) do not generalize from the constraints.

In addition to [179] and [155], other hard clustering and mixture modeling based methods, such as [167], [107], and [140] also suffer similar failure on the example in Fig. 3.1(a). We emphasize that the failure of these methods is *not* due to their finding poor, locally optimal solutions. The *globally* optimal solution, with respect to the objective functions proposed in these works, will not achieve constraint propagation and generalization on this (fundamental) example. For instance, the solution in Fig. 3.1(b) will have the largest data log-likelihood and the least number of constraint violations for the method in [179] with a single component per class. Since the global optimum itself may not achieve constraint propagation, it is the *choice* of the objective function that is the source of failure in these methods. This lack of constraint propagation will be further demonstrated theoretically in section 3.2 and experimentally in section 3.4.3.

There are several potential ways to overcome this limitation and achieve constraint propagation, including metric learning methods [170], [90], [158], and discriminative semi-supervised learning [108]. The approach developed here constrains the class (group) posterior on the feature space using a softmax function [21] applied to a parameterized class discriminant function of given representation power (*e.g.*, linear, quadratic, nonlinear kernel *etc.*). Parametrically constraining the class memberships ensures that neighboring points will have similar class assignments, and that co-located points will have *identical* class assignments. We refer to this approach as imposing a *space-partitioning* on the solution. The solution in Fig. 3.1(d) was obtained using this approach. Also, for this approach, solutions with group membership discontinuities (such as in Fig. 3.1(b)) are precluded.

3.1.2.2 Representation power of classes and number of classes

Many existing methods based on clustering algorithms such as K-means or hierarchical clustering [167], [90], [6], mixture modeling [155], [107], [98], or metric learning [170], [4], [158], [47] assume *both* (i) that the number of classes is known ([90] is an exception which estimates the number of clusters), and (ii) that each learned cluster is a distinct class. Both of these assumptions are restrictive and potential sources of sub-optimality. In particular, for the clustering distortion measure or mixture component density function family (*e.g.*, Gaussian) assumed by a given method, a complicated class (with a multimodal class-conditional density) may only be well-modeled if represented by *multiple* clusters (components) [179]. In fact, Fig. 3.1(a) is also illustrative of *this* point since, in this example, each of the classes consists of multiple (two)

[1] This essentially states that points in a small neighborhood, within a high density region of the feature space, are likely to have "similar" outputs.

multivariate Gaussian mixture components. Accordingly, in general, more complex class modeling than a "single cluster per class" will be required in order to satisfy the given constraints. In fact, it is *a priori* unknown how many (e.g. Gaussian) components may be required in order to well-model a given class.

A second significant limitation is the assumption that the number of classes represented in the given data set is known. This is a basic assumption made by nearly all prior works ([179] is an exception). However, since the partial supervision is in the form of pairwise constraints, and not class labels, the number of classes will *not* be known in general, in which case a core assumption of many prior works does not hold.

3.1.3 Overview of our approach

We cast semi-supervised learning with pairwise sample constraints as a mixture modeling problem, building on the prior work [179]. Accordingly, we start with the penalized negative complete data log-likelihood (also called the potential), and propose to minimize the expectation of this potential taken with respect to the posterior distribution of component assignments. Invoking a mean-field approximation to this posterior distribution (to avoid intractability), and performing an unconstrained minimization with respect to the distribution factors (as in [179]) can result in a lack of constraint propagation and group membership discontinuities, as illustrated in Fig. 3.1(b). To avoid this problem, we use a parameterized mean-field (or variational) approximation [109], [68] to the posterior distribution, with the parametrization chosen to be a smooth function of the feature vector having the same representation power as the parametric density function of the mixture model. The expectation maximization (EM) algorithm is used to minimize our objective function over the variational parameters of the factorized approximation in the E-step, and over the parameters of the mixture model in the M-step. By imposing space-partitioning in the solution, this approach avoids the group membership discontinuity problem and can achieve constraint propagation from a very limited number of constraints, as discussed in section 3.2.2.1 and demonstrated by experimental results.

In order to allow modeling of complex class boundaries and automatic estimation of the number of classes in the data, we introduce binary parameters which jointly indicate the class with which each mixture component is affiliated. These parameters are estimated along with the mixture model parameters in the M-step. In learning these binary parameters, we achieve allocation of a variable number of mixture components to each class. Furthermore, given a fixed number of mixture components, K, and, thus, a *maximum* class cardinality $L_{\max} \leq K$, the actual number of classes estimated by the model is the number of unique class indices $c \in \{1, 2, \ldots, K\}$ such that at least one mixture component is assigned to that class. Clearly, based on this description,

both the (variable) allocation of mixture components to classes and the number of estimated classes are functions of K. Thus, effective component allocation and accurate class number estimation critically hinge on good estimation of the number of mixture components, K. We determine this number by applying a model order estimation "wrapper" around our minimization problem. The optimal model order is chosen by jointly considering the Bayesian Information Criterion (BIC) [152] and the degree of constraint satisfaction in the solution.

The rest of the chapter is organized as follows. In the next section, the problem formulation and solution approach of our method are developed. In section 3.1.2, we compare and contrast some of the prior work in the literature that are related to our proposed work. Detailed experimental evaluations including comparisons to prior work are given in section 3.4.

3.2 Method formulation and solution approach

Consider a data set $\mathcal{X} = \{\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_N\}$, where each $\underline{x}_i \in \mathbb{R}^d$. The must-link constraints are specified by the set of index pairs of samples $\mathcal{I}_m$, such that if $(i,j) \in \mathcal{I}_m$, then $\underline{x}_i$ and $\underline{x}_j$ are known to have the same class label. The cannot-link constraints are specified by the set of index pairs of samples $\mathcal{I}_c$, such that if $(i,j) \in \mathcal{I}_c$, then $\underline{x}_i$ and $\underline{x}_j$ are known to have different class labels. For a positive integer n, we denote the set $\{1, 2, \ldots, n\}$ by $[n]$. The index set of samples not involved in any pairwise constraints is defined as $\mathcal{I}_u = \{i \in [N] \mid \forall j \in [N], (i,j) \notin \mathcal{I}_m \cup \mathcal{I}_c \text{ and } (j,i) \notin \mathcal{I}_m \cup \mathcal{I}_c\}$. We assume that there exists a confidence (or relative weight) $C_{ij} \in [0,1]$ for every constraint $(i,j) \in \mathcal{I}_m \cup \mathcal{I}_c$. If these values are not specified, we set them to a default value $C_{ij} = 1, \ \forall (i,j) \in \mathcal{I}_m \cup \mathcal{I}_c$. For convenience, we define $\mathcal{C} = \{C_{ij}, \ \forall i,j \in [N]\}$, where $C_{ij} = 0, \ \forall (i,j) \notin \mathcal{I}_m \cup \mathcal{I}_c$. Consider a mixture model with K components and parameter set $\Theta = \{(\alpha_k, \theta_k) \mid k \in [K]\}$, where $\alpha_k \geq 0$ is the mixing proportion (or prior probability) of component k satisfying $\sum_k \alpha_k = 1$, and θ_k is the parameter set of component k specifying its component-conditional density $f(\underline{x} \,|\, \theta_k)$. For each sample $\underline{x}_i$, the unknown mixture component of origin is defined by the vector $\underline{M}_i = [M_{i1}, \ldots, M_{iK}]$, where $M_{ij} \in \{0,1\}$ and $\sum_{j=1}^{K} M_{ij} = 1$. Define $\mathcal{M} = \{\underline{M}_i \mid i \in [N]\}$, the set of latent variables (mixture component of origin) associated with all the samples, and also $\mathcal{M}_u = \{\underline{M}_i \mid i \in \mathcal{I}_u\}$, and $\mathcal{M}_c = \mathcal{M} \backslash \mathcal{M}_u$.

3.2.1 Penalized complete data negative log-likelihood

The complete data log-likelihood [48] for a standard mixture model, which assumes independent generation of each sample, is given by

$$\log P(\mathcal{X}, \mathcal{M} \mid \Theta) \;=\; \sum_{i=1}^{N} \sum_{k=1}^{K} M_{ik} \;\log[\alpha_k \, f(\underline{x}_i \mid \theta_k)].$$

In our problem, the presence of constraints between certain pairs of samples must be accounted for. We will do so by adding a penalty term that tallies constraint violations.

In section 3.1.2.2 we discussed having sufficient model representation power to learn complex decision boundaries between classes. Here we allow each class to be flexibly modeled by multiple mixture components, and also learn the class-to-component associations. This will allow us to estimate the *number* of classes present in the data, covering situations where the number of classes present may be unknown. Assuming that the total number of mixture components for accurately modeling the data (K) is known [2], an upper bound on the number of classes present is $L_{\max} \leq K$, where in the case of equality each component represents a distinct class. Let $\mathcal{V} = \{V_{kl} \mid k \in [K], l \in [L_{\max}]\}$ denote the set of binary component assignments to classes, with $V_{kl} = 1$ if component k is assigned to class l; else $V_{kl} = 0$. Each component is assigned to only one of the classes, *i.e.*, $\sum_l V_{kl} = 1, \;\; \forall k \in [K]$. The number of components assigned to class l is given by $\sum_k V_{kl}$. As will be seen later, solutions for our method may have some classes for which this sum is 0, in which case the estimated number of classes is given by $L = |\{l \in [L_{\max}] \;\mid \sum_k V_{kl} \;\; \geq \;\; 1\}|$. We treat the class assignments to components as model parameters to be estimated, as in some past works [122], [135], [120]. Accordingly, we extend Θ to include $\mathcal{V}$, *i.e.*, $\Theta = \{(\alpha_k, \theta_k, \{V_{kl}, \mid l \in [L_{\max}]\}) \mid k \in [K]\}$.

In order to discourage or penalize solutions in which component assignments of samples are not consistent with the given ML and CL constraints, we add a penalty term to the complete data negative log-likelihood. This penalized complete data negative log-likelihood, also referred to as the *potential* [179], is defined as:

$$U(\mathcal{M}, \Theta) \;=\; -\log P(\mathcal{X}, \mathcal{M} \mid \Theta) \;+\; \beta \, h(\mathcal{M}_c, \mathcal{V}, \mathcal{C}), \tag{3.1}$$

where

$$h(\mathcal{M}_c, \mathcal{V}, \mathcal{C}) \;=\; \sum_{(i,j) \in \mathcal{I}_m} C_{ij} \left(1 - \sum_{l=1}^{L_{\max}} W_{il} \, W_{jl}\right) \;+\; \sum_{(i,j) \in \mathcal{I}_c} C_{ij} \sum_{l=1}^{L_{\max}} W_{il} \, W_{jl}$$

[2] We address estimation of K in the sequel.

Here the *class* memberships of samples are defined by $W_{il} = \sum_k M_{ik} V_{kl}, \ \forall i \in [N], \ l \in [L_{\max}], \ h(\cdot)$ is the constraint violation penalty, and β is a positive constant which controls the possible tradeoff between modeling the data density well and meeting the constraints. Consider the first term of $h(\mathcal{M}_c, \mathcal{V}, \mathcal{C})$. If a must-linked pair (i, j) is such that $\underline{x}_i$ and $\underline{x}_j$ have different class memberships, *i.e.*, $W_{il} \neq W_{jl}, \ \forall l \in [L_{\max}]$, then there is a positive contribution C_{ij} to the penalty term. Consider the second term of $h(\mathcal{M}_c, \mathcal{V}, \mathcal{C})$. If a cannot-linked pair (i, j) is such that $\underline{x}_i$ and $\underline{x}_j$ have the same class membership, *i.e.*, $W_{il} = W_{jl}, \ \forall l \in [L_{\max}]$, then there is a positive contribution C_{ij} to the penalty term. The distinction between classes and components (clusters) should be clear - components are involved in data modeling (the complete data log-likelihood term), while classes, which consist of (possibly) multiple components, are involved in the constraint penalty term. Two samples satisfy an ML constraint even if they belong to different components, so long as both the components belong to the same class, and they satisfy a CL constraint only if they belong to components from different classes.

Note that the constraint penalty $h(\mathcal{M}_c, \mathcal{V}, \mathcal{C}) \geq 0$, with equality only if all the constraints are satisfied. This form of the penalty is an improvement over the formulation in [179], where both CL and ML constraint violations have the same contribution $C_{ij} \sum_l W_{il} W_{jl}$, but with C_{ij} positive for CL constraints and negative for ML constraints. To appreciate this distinction, consider a simple scenario with one ML and one CL constraint. Suppose the ML constraint is satisfied, but the CL constraint is not satisfied for solutions minimizing (3.1), over a range of values of β. For the approach in [179], the term involving the ML constraint has a greater negative contribution to the potential as β is increased (although the ML constraint was satisfied in the solution for a smaller values of β itself). Thus, a smaller value of potential (obtained by increasing β) will *not*, in this case, reflect greater constraint satisfaction. By contrast, the potential proposed in (3.1) can *only* decrease for increasing β if *more* constraints are satisfied.

We also note that a "single component per class" is a special case of (3.1) where $L_{\max} = K$ and $V_{kk} = 1, \ \forall k \in [K]$. Both "single component per class" and "multiple components per class" variants will be experimentally evaluated in the sequel.

3.2.2 Objective function and optimization method

Our learning objective function is the Helmholtz free energy [175], [130], [146] associated with the potential (3.1), given by [3]

$$F(\Theta) \;=\; -\log \sum_{\mathcal{M}} e^{-U(\mathcal{M},\Theta)}. \tag{3.2}$$

To motivate this, consider the free energy corresponding to the potential (3.1) when the constraint penalty term is *absent*, *i.e.*, $U(\mathcal{M},\Theta) \;=\; -\ln P(\mathcal{X},\mathcal{M}\,|\,\Theta)$. In this case, the free energy is the incomplete data negative log-likelihood [48], *i.e.*,

$$-\log P(\mathcal{X}\,|\,\Theta) \;=\; -\log \sum_{\mathcal{M}} P(\mathcal{X},\mathcal{M}\,|\,\Theta) \;=\; -\log \sum_{\mathcal{M}} e^{-U(\mathcal{M},\Theta)}.$$

We add the term $h(\mathcal{M}_c,\mathcal{V},\mathcal{C})$ to dissuade latent variable assignments ($\mathcal{M}_c$) which violate the given ML and CL constraints. Consider the joint posterior distribution of the latent variables conditioned on the data and parameters, which has the form of a Gibbs distribution, *i.e.*,

$$P(\mathcal{M}\,|\,\mathcal{X},\Theta) \;=\; \frac{P(\mathcal{X},\mathcal{M}\,|\,\Theta)}{P(\mathcal{X}\,|\,\Theta)} \;=\; \frac{e^{-U(\mathcal{M},\Theta)}}{\sum_{\mathcal{M}} e^{-U(\mathcal{M},\Theta)}}. \tag{3.3}$$

Also, consider another *freely chosen* probability distribution, $P^0(\mathcal{M}\,|\,\mathcal{X})$. The free energy (3.2) can be rewritten as

$$\begin{aligned}
F(\Theta) \;&=\; \sum_{\mathcal{M}} P^0(\mathcal{M}\,|\,\mathcal{X}) \;\log\left[\frac{P(\mathcal{M}\,|\,\mathcal{X},\Theta)}{e^{-U(\mathcal{M},\Theta)}}\right] \\
&=\; \sum_{\mathcal{M}} P^0(\mathcal{M}\,|\,\mathcal{X}) \;\log\left[\frac{P(\mathcal{M}\,|\,\mathcal{X},\Theta)}{e^{-U(\mathcal{M},\Theta)}}\;\frac{P^0(\mathcal{M}\,|\,\mathcal{X})}{P^0(\mathcal{M}\,|\,\mathcal{X})}\right] \\
&=\; \sum_{\mathcal{M}} P^0(\mathcal{M}\,|\,\mathcal{X})\, U(\mathcal{M},\Theta) \;+\; \sum_{\mathcal{M}} P^0(\mathcal{M}\,|\,\mathcal{X}) \;\log\left[P^0(\mathcal{M}\,|\,\mathcal{X})\right] \\
&-\; \sum_{\mathcal{M}} P^0(\mathcal{M}\,|\,\mathcal{X}) \;\log\left[\frac{P^0(\mathcal{M}\,|\,\mathcal{X})}{P(\mathcal{M}\,|\,\mathcal{X},\Theta)}\right] \\
&=\; E_{P^0}\left[U(\mathcal{M},\Theta)\right] \;-\; H\left[P^0\right] \;-\; D_{KL}[P^0,P],
\end{aligned} \tag{3.4}$$

[3] We use $\sum_{\mathcal{M}}$ as a shorthand notation for $\sum_{\underline{M}_1\in\Delta}\sum_{\underline{M}_2\in\Delta}\cdots\sum_{\underline{M}_N\in\Delta}$, where Δ is the set of all $\{0,1\}$-valued tuples of size K such that in each tuple exactly one of the values is 1 and the rest are 0.

where $H\left[P^0\right]$ is the entropy of the distribution $P^0(\mathcal{M} \mid \mathcal{X})$, and $D_{KL}[P^0, P]$ is the Kullback-Leibler (KL) distance [42] between the distributions. Defining

$$\widetilde{F}(\Theta, P^0) \;=\; E_{P^0}\left[U(\mathcal{M}, \Theta)\right] \;-\; H\left[P^0\right], \tag{3.5}$$

and observing that the KL distance is always non-negative, we have

$$\widetilde{F}(\Theta, P^0) \;=\; F(\Theta) \;+\; D_{KL}[P^0, P] \;\geq\; F(\Theta).$$

The upper bound quantity $\widetilde{F}(\Theta, P^0)$ is called the *variational free energy* [109], and it is equal to $F(\Theta)$ when $P^0(\mathcal{M} \mid \mathcal{X}) = P(\mathcal{M} \mid \mathcal{X}, \Theta)$. The free energy $F(\Theta)$ can, at least *in principle*, be minimized using the EM algorithm, with the E-step minimizing the upper bound over all valid distributions $P^0(\mathcal{M} \mid \mathcal{X})$ for fixed Θ (by choosing $P^0 = P$, which achieves $\widetilde{F}(\Theta, P^0) = F(\Theta)$), and the M-step minimizing the upper bound over Θ for fixed $P^0(\mathcal{M} \mid \mathcal{X})$ [130], [175]. These two steps are iterated, with each iteration descending in $F(\Theta)$ until a local minimum is reached.

However, in some problems, it may not be tractable to analytically compute the summation (or integration) involved in the normalization term of the posterior distribution (3.3) and also in the expectations taken with respect to the posterior distribution. One approach to handle this intractability is called *variational approximation* [109], [78], [129], wherein the distribution $P^{(0)}$ is restricted to have a simpler form relative to the true posterior distribution, and the E-step optimization is performed over the restricted space of distributions. For example, $P^{(0)}$ can be a factorized product of simpler distributions over disjoint subsets of the full set of random variables $\mathcal{M}$, and the E-step performs an optimization over the factors of the posterior distribution. This approach is commonly known as the *mean-field approximation* [68], [138]. Sometimes the distribution $P^{(0)}$ or its component factors are chosen to be parametric functions such that the summations (or integrations) become tractable to compute analytically. Then the E-step performs an optimization over the parameters of $P^{(0)}$, which are also referred to as variational parameters.

3.2.2.1 Lack of constraint propagation in the optimal posterior distribution

From (3.3) and (3.1), the optimizing distribution in the E-step can be written as

$$P(\mathcal{M} \mid \mathcal{X}, \Theta) \;=\; \frac{e^{\log P(\mathcal{X}, \mathcal{M} \mid \Theta)}\; e^{-\beta\, h(\mathcal{M}_c, \mathcal{V}, \mathcal{C})}}{\displaystyle\sum_{\mathcal{M}} e^{\log P(\mathcal{X}, \mathcal{M} \mid \Theta)}\; e^{-\beta\, h(\mathcal{M}_c, \mathcal{V}, \mathcal{C})}}$$

$$= \frac{\prod_{i\in\mathcal{I}_u}\prod_{k=1}^{K}(\alpha_k\, f(\underline{x}_i\,|\,\theta_k))^{M_{ik}}}{\sum_{\mathcal{M}_u}\prod_{i\in\mathcal{I}_u}\prod_{k=1}^{K}(\alpha_k\, f(\underline{x}_i\,|\,\theta_k))^{M_{ik}}} \; \frac{e^{-\beta\, h(\mathcal{M}_c,\mathcal{V},\mathcal{C})}\prod_{i\in[N]\setminus\mathcal{I}_u}\prod_{k=1}^{K}(\alpha_k\, f(\underline{x}_i\,|\,\theta_k))^{M_{ik}}}{\sum_{\mathcal{M}_c} e^{-\beta\, h(\mathcal{M}_c,\mathcal{V},\mathcal{C})}\prod_{i\in[N]\setminus\mathcal{I}_u}\prod_{k=1}^{K}(\alpha_k\, f(\underline{x}_i\,|\,\theta_k))^{M_{ik}}} \tag{3.6}$$

The normalization term over $\mathcal{M}_u$ can be simplified as

$$\sum_{\mathcal{M}_u}\prod_{i\in\mathcal{I}_u}\prod_{k=1}^{K}(\alpha_k\, f(\underline{x}_i\,|\,\theta_k))^{M_{ik}} = \prod_{i\in\mathcal{I}_u}\sum_{\underline{M}_i}\prod_{k=1}^{K}(\alpha_k\, f(\underline{x}_i\,|\,\theta_k))^{M_{ik}} = \prod_{i\in\mathcal{I}_u}\sum_{k=1}^{K}\alpha_k\, f(\underline{x}_i\,|\,\theta_k).$$

However, the normalization term

$$B(\Theta) \;=\; \sum_{\mathcal{M}_c} e^{-\beta\, h(\mathcal{M}_c,\mathcal{V},\mathcal{C})}\prod_{i\in[N]\setminus\mathcal{I}_u}\prod_{k=1}^{K}(\alpha_k\, f(\underline{x}_i\,|\,\theta_k))^{M_{ik}}$$

cannot be simplified in this way (as a product of summations over the component assignments of individual samples $\underline{M}_i$) because the term $e^{-\beta\, h(\mathcal{M}_c,\mathcal{V},\mathcal{C})}$ involves products of component assignments over pairs of must-linked and cannot-linked samples. Hence, $B(\Theta)$ will be intractable to compute in practice. From (3.6), it should be evident that $\mathcal{M}_u$ and $\mathcal{M}_c$ are conditionally independent given $\mathcal{X}$ and Θ, and the posterior distribution over latent variables can be written as $P(\mathcal{M}\,|\,\mathcal{X},\Theta) \;=\; P(\mathcal{M}_u\,|\,\mathcal{X},\Theta)\;\; P(\mathcal{M}_c\,|\,\mathcal{X},\Theta)$, where

$$\begin{aligned} P(\mathcal{M}_u\,|\,\mathcal{X},\Theta) \;&=\; \prod_{i\in\mathcal{I}_u}\prod_{k\in[K]}\Big(\frac{\alpha_k\, f(\underline{x}_i|\theta_k)}{\sum_{k'\in[K]}\alpha_{k'}\, f(\underline{x}_i|\theta_{k'})}\Big)^{M_{ik}} \\ &\triangleq\; \prod_{i\in\mathcal{I}_u}\prod_{k\in[K]} P(M_{ik}=1\,|\,\underline{x}_i,\Theta)^{M_{ik}}, \end{aligned} \tag{3.7}$$

and

$$P(\mathcal{M}_c\,|\,\mathcal{X},\Theta) \;=\; \frac{e^{-\beta\, h(\mathcal{M}_c,\mathcal{V},\mathcal{C})}}{B(\Theta)}\prod_{i\in[N]\setminus\mathcal{I}_u}\prod_{k\in[K]}[\alpha_k\, f(\underline{x}_i|\theta_k)]^{M_{ik}}, \tag{3.8}$$

We will now argue that even if exact computation of this distribution *were* possible, such a solution is in fact not *desirable* from the standpoint of constraint propagation. Consider the marginal component posterior of an unconstrained sample $\underline{x}_i$ given by $P(M_{ik}=1\,|\,\underline{x}_i,\Theta)$ in (3.7). This has *no dependence* on the constraint penalty term. But for a constrained sample $\underline{x}_j$, the marginal component posterior $P(M_{jk}=1\,|\,\mathcal{X},\Theta)$ (obtained by marginalizing $P(\mathcal{M}_c\,|\,\mathcal{X},\Theta)$) depends on the constraint penalty term and on β. To appreciate that this solution in general achieves no propagation of constraint information even to neighboring points in the feature

space, consider the following stark example. Suppose that 1) $\underline{x}_i = \underline{x}_j$, *i.e.*, they are co-located, 2) $P(M_{il} = 1 \,|\, \underline{x}_i, \Theta) \approx 1$, *i.e.*, $\underline{x}_i$ strongly belongs to component l based on the current mixture model parameters, and 3) suppose that $\underline{x}_k$ is a data sample that is must-linked with $\underline{x}_j$, but which, if based solely on its feature vector, would strongly belong to a *different* component $m \neq l$, *i.e.*, $\frac{\alpha_m f(\underline{x}_k|\theta_m)}{\sum_{n\in[K]} \alpha_n f(\underline{x}_k|\theta_n)} \approx 1$, with components l and m affiliated to different classes. When $P(\mathcal{M}_c \,|\, \mathcal{X}, \Theta)$ has the unconstrained form (3.8), one can make β sufficiently large such that the penalty term $e^{-\beta\, h(\mathcal{M}_c, \mathcal{V}, \mathcal{C})}$ dominates the data likelihood term. For the example at hand, this means that $P(M_{kl} = 1 \,|\, \mathcal{X}, \Theta) = P(M_{jl} = 1 \,|\, \mathcal{X}, \Theta) \approx 1$ can be achieved without *any* adjustment of the mixture component parameters. This scenario precisely corresponds to the example in Fig. 3.1(b), with $\underline{x}_k$ playing the role of a constrained sample whose class (and component) membership based on (3.8) is a "hole" discontinuity relative to the unconstrained samples in its neighborhood. On the other hand, it is *also* possible that, based on (3.8) both $\underline{x}_j$ and $\underline{x}_k$ will be strongly associated with component m. In this case, the co-located points $\underline{x}_i$ and $\underline{x}_j$ will have different class (and component) memberships. Both solutions are poor.

Clearly, from this example, we conclude that (3.7) and (3.8) provide no real mechanism for constraint propagation – for the given example, we would expect that the only way to satisfy the constraints and at the same time fit the data well should be by adjusting the mixture component parameters, *i.e.*, by *moving* the components in the feature space and/or changing their class affiliations. We next propose an effective way of doing so.

3.2.2.2 Constrained mean field approximation through parametrization

There are two objectives in our choice of approximate E-step: 1) to make the E-step tractable; 2) unlike the methods in [167], [155], [107], [179], [140], to achieve *constraint propagation* via the E-step. To make the E-step tractable, we invoke a factorized mean field approximation [109], [68], *i.e.*,

$$P^0(\mathcal{M} \,|\, \mathcal{X}) \;=\; \prod_{i=1}^{N} \prod_{k=1}^{K} q(k \,|\, \underline{x}_i)^{M_{ik}}.$$

If the factors are unconstrained (other than the requirement that they be valid PMFs) as proposed in [179], the preceding argument for the joint posterior distribution should also convince the reader that no constraint propagation and generalization will be achieved with the factorized, unconstrained mean field approximation. To overcome this deficiency, we propose to constrain all the factors in the mean field approximation to be smooth functions (*i.e.*, functions having continuous higher order derivatives) of the feature vector. In particular, we choose the factors to

be

$$q(k \,|\, \underline{x}_i) \;=\; \frac{e^{\psi_k(\underline{x}_i)}}{\sum\limits_{m\in[K]} e^{\psi_m(\underline{x}_i)}}, \quad \forall k \in [K],\; \forall i \in [N] \tag{3.9}$$

where $\psi_k(\underline{x}),\ k \in [K]$ are smooth functions, to be optimized in the E-step. In (3.9) we are effectively invoking parameterized discriminant functions $\psi_k(\underline{x})$, one per mixture component, and then "softmaxing" these functions [21] to create a probabilistic space-partitioning of the feature space. Note that, while for *tractability*, only $P(\mathcal{M}_c \,|\, \mathcal{X}, \Theta)$ needs to be approximated by a factorized distribution (which could even be *unconstrained*), the lack of constraint propagation can be avoided only if we impose that *all* the factors (both for constrained and unconstrained samples) have the same form $q(k \,|\, \underline{x})$ specified in (3.9). Since the mixture component posterior is now *exclusively* a parameterized function of the feature vector, the only way to achieve constraint satisfaction in the previous example (and in the example in Fig. 3.1) will be by adjusting the discriminant function parameters and the component to class assignments. Moreover, regardless of the choice of these parameter values, they induce a smooth component (and class) posterior, without discontinuities. We next propose two different approaches for choosing and optimizing $\psi_k(\underline{x}),\ k \in [K]$, in the E-step.

One approach is to constrain the functions $\psi_k(\underline{x}),\ k \in [K]$ to belong to a smooth parametric family, *i.e.*, a set of functions which have the same dependence on $\underline{x}$, but different parameters. In this case, $\psi_k(\underline{x}) = \psi(\underline{x}\,; \phi_k),\ \ k \in [K]$, where ϕ_k is the set of adjustable parameters for component k, and $\Phi = \{\phi_1, \cdots, \phi_K\}$. For a particular component density, we can choose $\psi(\underline{x}\,; \phi_k)$ to have the same dependence on $\underline{x}$ as the function $\log[\alpha_k\, f(\underline{x} \,|\, \theta_k)]$. For example, for a Gaussian mixture model (GMM), $\log[\alpha_k\, f(\underline{x} \,|\, \theta_k)]$ is a quadratic function of $\underline{x}$. So, we can choose $\psi(\underline{x}\,; \phi_k) = \underline{x}^T \mathbf{W}_k\, \underline{x} + \underline{x}^T \underline{v}_k + b_k$, where $\phi_k = \{\mathbf{W}_k,\, \underline{v}_k,\, b_k\}$ with $\mathbf{W}_k \in \mathbb{R}^{d\times d}$ and symmetric, $\underline{v}_k \in \mathbb{R}^d$, and $b_k \in \mathbb{R}$. The motivation for this choice is that, in the *absence* of any constraint information, if $\psi(\underline{x}\,; \phi_k) = \log[\alpha_k\, f(\underline{x} \,|\, \theta_k)]$, then $q(k \,|\, \underline{x}, \Phi)$ [4] in (3.9) "specializes" to the standard mixture component posterior (for i.i.d. data generation). Also, this choice is consistent with our inductive bias that the data is well modeled by the mixture density family. The parameters Φ are also referred to as *variational* parameters. In this case, the minimization problem in the E-step is given by $\min_\Phi \widetilde{F}(\Theta, \Phi)$ [5], subject to constraints on the members of Φ, if any.

To illustrate the generality of our space-partitioning approach, we note that another valid choice is to restrict the functions $\psi_k(\underline{x}),\ \ k \in [K]$ to the Reproducing Kernel Hilbert Space

[4] $q(k \,|\, \underline{x}_i)$ is rewritten as $q(k \,|\, \underline{x}_i, \Phi)$ in order to bring out its dependence on parameters Φ.

[5] $\widetilde{F}(\Theta, P^0)$ in (3.5) is rewritten as $\widetilde{F}(\Theta, \Phi)$ in order to bring out its dependence on Φ.

(RKHS) $\mathcal{H}$ of a kernel $g(\cdot\,,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ [164], [69]. The smoothness (regularity) of the functions can be controlled by adding a kernel based regularizer to the variational free energy $\widetilde{F}(\Theta, \psi_1, \ldots, \psi_K)$ [6] defined in (3.5). In this case, the functional minimization problem in the E-step is

$$\min_{\{\psi_k \in \mathcal{H},\, k \in [K]\}} \widetilde{F}(\Theta, \psi_1, \ldots, \psi_K) \;+\; \lambda \sum_{k=1}^{K} \|\psi_k\|^2_{\mathcal{H}}, \tag{3.10}$$

where λ is a positive regularization coefficient. This seemingly difficult problem can be simplified by invoking the Representer theorem [164], which states that the solution of (3.10) allows a representation of the form

$$\psi_k(\underline{x}) \;=\; \sum_{i=1}^{N} a_{ki}\; g(\underline{x}, \underline{x}_i), \;\; \forall k \in [K],$$

i.e., the solution lies in the finite dimensional subspace $\mathcal{H}_{\mathcal{X}} = \text{span}\{g(\cdot\,, \underline{x}_1), \ldots, g(\cdot\,, \underline{x}_N)\}$ of the RKHS $\mathcal{H}$. Therefore, (3.10) reduces to a nonlinear optimization problem over the NK variables $\Phi = \{\underline{a}_k = (a_{k1}, \ldots, a_{kN})^T \in \mathbb{R}^N \;\mid k \in [K]\}$, which are the variational parameters for this solution approach. Note that λ is a hyperparameter, which has to be set to a sufficiently large value to ensure that the functions $\psi_k(\underline{x})$ are smooth.

While the kernel based parametrization allows more flexibility in the choice of discriminant functions than the parametric approach, it also requires more computation in the E-step because the hyperparameter λ, which controls the smoothness of the functions $\psi_k(\underline{x})$, has to be set to a sufficiently large value by searching over some candidate values and using some objective criterion. Also, the need for computing and storing the $N \times N$ Gram matrix of the kernel, and optimizing over NK variables will make it less suitable for datasets with a large number of samples N. Therefore, we will focus on the parametric approach in this work, and indeed choose the parametrization to give the same representation power as a standard mixture component posterior. Although the form of this parametrization is fixed, allowing a variable number of components per class (as we do) provides additional flexibility in the modeling.

3.2.2.3 EM Algorithm

We have discussed how the parameterized mean field (or variational) approximation can be used both to find a tractable approximation to the optimal posterior distribution, *and* to impose space-partitioning (smoothness) in the solution. The variational free energy, given by $\widetilde{F}(\Theta, \Phi) \;= E_{P^0}\left[U(\mathcal{M}, \Theta)\right] - H\left[P^0\right]$, is straightforward to compute since the distribution P^0 has a factor-

[6] $\widetilde{F}(\Theta, P^0)$ is rewritten as $\widetilde{F}(\Theta, \psi_1, \ldots, \psi_K)$ in order to bring out its dependence on the functions $\psi_k(\underline{x}),\; k \in [K]$.

ized form. The expectation of the potential with respect to P^0 is given by

$$E_{P^0}\left[U(\mathcal{M},\Theta)\right] \;=\; E_{P^0}\left[-\log P(\mathcal{X},\mathcal{M}\,|\,\Theta)\right] \;+\; \beta\, E_{P^0}\left[h(\mathcal{M}_c,\mathcal{V},\mathcal{C})\right] \tag{3.11}$$

$$= \;-\sum_{i=1}^{N}\sum_{k=1}^{K} q(k\,|\,\underline{x}_i,\Phi)\;\log[\alpha_k\, f(\underline{x}_i\,|\,\theta_k)] \;+\; \beta \sum_{(i,j)\in\mathcal{I}_c} C_{ij}\, A_{ij} \;+\; \beta \sum_{(i,j)\in\mathcal{I}_m} C_{ij}\,(1-A_{ij}),$$

where

$$A_{ij} \;=\; \sum_{k=1}^{K}\sum_{k'=1}^{K} q(k\,|\,\underline{x}_i,\Phi)\, q(k'\,|\,\underline{x}_j,\Phi) \sum_{l=1}^{L_{max}} V_{kl}\, V_{k'l}.$$

The entropy of the distribution $P^0(\mathcal{M}\,|\,\mathcal{X},\Phi)$ is given by

$$H\left[P^0\right] \;=\; -\sum_{i=1}^{N}\sum_{k=1}^{K} q(k\,|\,\underline{x}_i,\Phi)\;\log q(k\,|\,\underline{x}_i,\Phi). \tag{3.12}$$

As before, the objective is to minimize $\widetilde{F}(\Theta,\Phi)$ with respect to both the model parameters Θ and the variational parameters Φ, which can be done via an EM algorithm as follows. Given fixed model parameters $\Theta^{(t)}$, at iteration t the E-step solves $\Phi^{(t)} = \arg\min_{\Phi} \widetilde{F}(\Theta^{(t)},\Phi)$, thus finding the best parameterized mean field approximation to the distribution $P(\mathcal{M}\,|\,\mathcal{X},\Theta^{(t)})$. Given fixed variational parameters $\Phi^{(t)}$, at iteration t the M-step solves $\Theta^{(t+1)} = \arg\min_{\Theta} \widetilde{F}(\Theta,\Phi^{(t)})$. These two steps are iterated until a local minimum of $\widetilde{F}(\Theta,\Phi)$ is found. Note that this EM algorithm does not find a local minimum of the free energy $F(\Theta)$ (3.2) because of our use of a factorized, *parameterized* approximation of $P(\mathcal{M}\,|\,\mathcal{X},\Theta)$. However, this is by design, because the optimal distribution $P(\mathcal{M}\,|\,\mathcal{X},\Theta)$ (or a factorized, but unconstrained approximation of it) will not achieve the desired space-partitioning and constraint propagation. We next present the details of the E-step and M-step for a mixture of Gaussian densities, but the ideas also apply more generally to distributions from the exponential family.

E-step

The minimization of $\widetilde{F}(\Theta^{(t)},\Phi)$ with respect to $\Phi = \{(\mathbf{W}_k, \underline{v}_k, b_k) \;|\; k \in [K]\}$ does not have a closed form solution. We applied the gradient descent method on the vector of parameters in Φ to convergence to find $\Phi^{(t)}$, a local minimum of $\widetilde{F}(\Theta^{(t)},\Phi)$.

M-step

The global minimum of $\widetilde{F}(\Theta,\Phi^{(t)})$ with respect to the component masses, component means, and component covariances $(\alpha_k, \underline{\mu}_k, \mathbf{\Sigma}_k),\;\; k \in [K]$ can be found analytically. These M-step

equations at iteration t are given by

$$\alpha_k^{(t+1)} \;=\; \frac{1}{N}\sum_{i=1}^{N} q(k \,|\, \underline{x}_i, \Phi^{(t)}), \tag{3.13}$$

$$\underline{\mu}_k^{(t+1)} \;=\; \frac{1}{N\,\alpha_k^{(t+1)}}\sum_{i=1}^{N} q(k \,|\, \underline{x}_i, \Phi^{(t)})\;\underline{x}_i, \tag{3.14}$$

and

$$\mathbf{\Sigma}_k^{(t+1)} \;=\; \frac{1}{N\,\alpha_k^{(t+1)}}\sum_{i=1}^{N} q(k \,|\, \underline{x}_i, \Phi^{(t)})\;(\underline{x}_i - \underline{\mu}_k^{(t+1)})(\underline{x}_i - \underline{\mu}_k^{(t+1)})^T. \tag{3.15}$$

The class assignments to components $\mathcal{V}$ are binary variables, which satisfy $\sum_l V_{kl} = 1 \;\forall k \in [K]$. Jointly optimizing these variables requires an exhaustive search over L^K possible states, which is computationally prohibitive. Alternatively, we use the *iterated conditional modes* algorithm [59], which optimizes the class assignments to components sequentially, one component at a time. This cycle is repeated until the class assignment does not change for any of the components, which means a local minimum configuration has been reached. If the number of classes L is known, we ensure that at least one component is assigned to each of the classes during the optimization.

3.2.3 Overall learning algorithm

In section 3.2.2, we presented our learning objective and solution approach for a fixed value of β and fixed number of mixture components K. We now discuss a principled approach to select these model hyper-parameters, which is important from the standpoint of applying our method to real data sets.

3.2.3.1 Setting hyperparameter β

Consider again the variational free energy objective

$$\widetilde{F}(\Theta, \Phi) \;=\; E_{P^0}[-\ln P(\mathcal{X}, \mathcal{M} \,|\, \Theta)] \;-\; H\left[P^0\right] \;+\; \beta\, E_{P^0}\left[h(\mathcal{M}_c, \mathcal{V}, \mathcal{C})\right],$$

where the individual terms are given by (3.11) and (3.12). For a fixed value of β, the EM algorithm discussed in section 3.2.2.3 is applied to minimize $\widetilde{F}(\Theta, \Phi)$. If the learning is repeated for increasing values of β, starting from the same initial Θ and Φ in each case, in theory (excepting local minima of the EM optimization), the value of the constraint penalty $E_{P^0}\left[h(\mathcal{M}_c, \mathcal{V}, \mathcal{C})\right]$ in the solution should be non-increasing. In this work, our primary objective is to utilize the pair-

wise constraints to learn a solution with accurate class boundaries, rather than one which best models the data density at the expense of not satisfying some constraints. Hence, we choose β as follows. Starting from a small value, β is increased in predetermined steps until the constraint penalty does not decrease significantly with further increase in β. Let $h_i,\ i = 1, \ldots, N^{(s)}$ denote the final constraint penalty values[7] in the solutions corresponding to different candidate β values $\beta_i,\ i = 1, \ldots, N^{(s)}$. Let $h^* = \min_{i \in [N^{(s)}]} h_i$. First, we find a smaller set of candidate solutions whose constraint penalties are not significantly different from the minimum, according to the criterion $h_i - h^* < \delta$, where $0 < \delta \ll 1$ (a small value such as 0.01). Among these candidate solutions, we select the one which has the smallest value for the data log-likelihood component of the variational free energy, $E_{P^0}[-\ln P(\mathcal{X}, \mathcal{M} \,|\, \Theta)] - H\left[P^0\right]$. This is done in order to potentially avoid choosing a solution corresponding to a large β value, which may have good constraint satisfaction, but a poor likelihood fit to the data.

3.2.3.2 Estimating the number of classes and the number of components

A novel feature of our method is that the number of classes can be directly inferred from the class assignments to components, $\mathcal{V}$. The solution of the EM algorithm may have some class values which are not assigned to any components, *i.e.*, $l \in [L_{\max}]$ for which $\sum_k V_{kl} = 0$. Such classes are effectively unused. Accordingly, the number of estimated classes is given by $L = |\{l \in [L_{\max}] \ | \ \sum_k V_{kl} \ \geq \ 1\}|$. This class number estimate will be experimentally evaluated in section 3.4.6.

Class number estimation depends on *component number* estimation. The number of components K is chosen as follows. The number of components is increased sequentially starting from an initial number K_0. If the number of classes in the data is unknown, we set $K_0 = 2$. If the number of classes in the data is known to be M, we set $K_0 = M$. For each model order, the optimization problem is solved for a range of β values, and a single β is selected using the approach discussed in section 3.2.3.1. For the solution corresponding to this selected β, the BIC/MDL model selection criterion [152], [63] is calculated. For our model with K mixture components, the BIC is given by

$$\mathrm{BIC}(\Theta, K) \ = \ \mathrm{CC}(\Theta, K) \ - \ \sum_{i=1}^{N} \log \sum_{j=1}^{K} (\alpha_j \, f(\underline{x}_i \,|\, \theta_j)), \tag{3.16}$$

where the codelength (description length) of the model, assuming Gaussian component densities

[7] The constraint penalty is normalized to be in the interval $[0, 1]$ for convenience. This is done by dividing by the maximum possible constraint penalty $\sum_{(i,j) \in \mathcal{I}_c \cup \mathcal{I}_m} C_{ij}$.

with unrestricted covariance matrix, is given by

$$\mathrm{CC}(\Theta, K) \;=\; \frac{1}{2}\,(K - 1 + d\,K + \frac{d\,(d+1)}{2}\,K)\,\log N \;+\; K\,\log L\,.$$

The term $K\,\log L$ is the codelength required to specify the discrete class assignments to components $\mathcal{V}$, assuming that each of their L^K configurations of values are equally likely. We stop increasing the number of components at the order K_{ceil} at which the BIC attains a minimum value (before increasing monotonically). If the primary objective is to utilize the pairwise constraints to learn an accurate mixture model for the data, then choosing the solution with minimum BIC would be suitable. However, since we are more interested in learning an accurate grouping of the data points which satisfies most of the constraints, we select, from among the solutions with $K \leq K_{\mathrm{ceil}}$, the one with the least constraint penalty as the final solution.

3.2.3.3 Parameter initialization and component splitting procedure

In this section, we discuss details like parameter initialization and addition of new mixture components, with focus on a mixture of Gaussian densities. For the starting number of components (model order), K_0, maximum likelihood estimation (MLE) is first used to learn a mixture model for the data without using the pairwise constraints. A certain number of trial initializations are then created by adding small perturbations to the mean vectors of the component Gaussian densities. For each such trial initialization, the constraint penalty term is calculated, and the trial initialization with the smallest constraint penalty is selected. Given the initial model parameters Θ, the variational parameters Φ are initialized by equating the quadratic, linear, and constant terms in the equation $\underline{x}^T \mathbf{W}_k\, \underline{x} + \underline{x}^T \underline{v}_k + b_k = \ln[\alpha_k\ f(\underline{x}|\theta_k)]$. When the number of classes in the data is not known, we set $L_{\max} = K_0$, and the parameters $\mathcal{V}$ are initialized by randomly assigning half of the components to one class, and the rest to a second class [8]. When the number of classes in the data is known to be M, we set $K_0 = L_{\max} = M$, and the parameters of $\mathcal{V}$ are initialized by assigning one component to each of the classes.

For subsequent (increasing) model orders, instead of initializing the parameters independently at random, we take the solution of the previous model order, and create a new component by "splitting" one of its existing components into two. The parameters of the remaining components (including their class assignments) are retained in the initialization for the next model order. For a mixture of Gaussians, in order to split a particular component, we first find the eigendecomposition of its covariance matrix, and then choose one of the eigenvectors as the direction

[8] Although the maximum number of classes will in general be greater than two, the optimization of $\mathcal{V}$ will assign additional classes only if required in order to satisfy the constraints. Starting, in this way, from below the true number of classes present can potentially avoid overestimating the number of classes.

to split. The mean vectors of the two split components are obtained by adding to the original component mean vector, a positive and a negative perturbation along the chosen eigenvector direction. The covariance matrices of the split components are each obtained by decreasing the corresponding eigenvalue of the original covariance matrix by a factor of 4, leaving the other eigenvalues unchanged. The mixing proportions of the split components are each chosen to be half the mixing proportion of the component before splitting. The initial class assignment of the two split components are selected in order to have the best constraint satisfaction. In order to go from K components to $K+1$ components, we split each of the K components along the direction of each of the d eigenvectors of its covariance matrix. For each such split, the initial class assignments of the two split components are varied to create a set of candidate initializations. The constraint penalty and data log-likelihood of the candidate initializations are calculated, and among the candidates whose penalties are close to the minimum penalty (using a small δ as before), the one with the largest log-likelihood is chosen as the initialization for the model with $K+1$ components.

3.2.3.4 Class prediction

In our model, since the class (group) assignment is independent of the feature vector given the component assignment, the class posterior for a sample $\underline{x}$ is given by

$$P(C = c \,|\, \underline{x}, \Theta, \Phi) \;=\; \sum_{k=1}^{K} V_{kc}\; q(k \,|\, \underline{x}, \Phi), \quad \forall c \in [L_{\max}]. \tag{3.17}$$

The class prediction for $\underline{x}$ is then the class with the maximum class posterior.

3.3 Related Work

Our method is closely related to that of [179], which also allows variable number of mixture components per class, and automatic class number estimation. However, this method has very limited ability to achieve constraint propagation and generalization as discussed in section 3.2.2.1, and demonstrated by the experimental results.

The problem of constraint propagation has been recognized and addressed in prior works such as [90], [98], [108], and [105]. We briefly discuss these approaches. In [90], constraint propagation is addressed using complete-linkage hierarchical agglomerative clustering. Their method is a type of metric learning, where the pairwise sample distance matrix is adjusted to be consistent with the constraints and their spatial implications. Complete-linkage clustering enables the constraint information to be propagated to unconstrained points. However, their

approach is not very robust because even a single constraint can dramatically change the distance matrix entries. In [98] (for a detailed technical report see [99]), the objective function is a weighted linear combination of the posterior likelihood over constraints and the data log-likelihood under a mixture model. By assigning more weight to the constraint posterior term, more emphasis can be given to learning the class boundaries consistent with the constraint information. This method requires significant user input that may be unavailable in practice. It assumes the number of classes is known and models each class with a single mixture component. Also, the method is developed only for must-links in [98]. In [108], the Gaussian process classifier is extended to handle pairwise constraints instead of class labels, and unconstrained samples are introduced into the learning through the covariance function of the Gaussian process prior, using data-dependent semi-supervised kernels. Since the method learns the class boundaries discriminatively, the solution has a smooth class posterior and achieves constraint propagation. One limitation of their method is the heuristic approximation of the expectation of a function involving products as a product of expectations. Although their method can be extended to multi-class problems, the formulation and experimentation only address two class problems. In [105], a constrained optimization problem is formulated in order to find a smooth non-linear transformation such that all the samples map to the surface of a unit hypersphere, samples involved in ML constraints map to the same point, and samples involved in CL constraints map to orthogonal points in the transformed feature space. Although their method can achieve constraint propagation by virtue of minimizing the smoothness objective, it has some fundamental limitations. First, their method learns a kernel matrix on the training data support, and hence cannot be applied to unobserved data. Second, their method requires hard constraint satisfaction because of the equality constraints in the optimization problem. The optimized kernel matrix is used in a kernel K-means clustering algorithm, which gives a hard clustering solution. Also, a common limitation shared by all the methods discussed here is that the number of classes needs to be specified.

3.4 Experiments

3.4.1 Performance measures

We evaluated the performance of the constraint-based semi-supervised learning methods in our experiments using two metrics: F-score [155], [179], [15] and Normalized mutual information (NMI) [166], [6], [105]. F-score, a combined measure of purity P and accuracy A, is defined as $F = \frac{2PA}{P+A}$. To define P and A, suppose that there are M true classes in the data, and that a method finds L groups (estimated classes) in the solution. Let n_{ij} be the number of samples

belonging to true class i that are predicted to belong to group j, where $i = 1, \ldots, M$ and $j = 1, \ldots, L$. Also, let $n_i = \sum_{j=1}^{L} n_{ij}$ be the number of samples in class i, $\hat{n}_j = \sum_{i=1}^{M} n_{ij}$ be the number of samples in group j, and n be the total number of samples. Purity and accuracy are then defined as

$$P = \sum_{j=1}^{L} \frac{\hat{n}_j}{n} \max_{i \in [M]} \frac{n_{ij}}{\hat{n}_j} = \frac{1}{n} \sum_{j=1}^{L} \max_{i \in [M]} n_{ij}. \tag{3.18}$$

$$A = \sum_{i=1}^{M} \frac{n_i}{n} \max_{j \in [L]} \frac{n_{ij}}{n_i} = \frac{1}{n} \sum_{i=1}^{M} \max_{j \in [L]} n_{ij}. \tag{3.19}$$

The normalized mutual information between the true class random variable C and the predicted class (group) random variable $\hat{C}$ is defined as

$$\mathrm{NMI}(\mathrm{C}, \hat{\mathrm{C}}) = \frac{\mathrm{I}(\mathrm{C}, \hat{\mathrm{C}})}{\sqrt{\mathrm{H}(\mathrm{C})\, \mathrm{H}(\hat{\mathrm{C}})}}, \tag{3.20}$$

where $I(\cdot, \cdot)$ denotes the mutual information between two random variables, and $H(\cdot)$ denotes the Shannon entropy of a random variable. We calculate both these metrics only over the unconstrained samples (defined by index set $\mathcal{I}_u$).

3.4.2 Methods used for comparison

We compared the performance of our method with the mixture model based methods of [155] (referred to as SCGMM) and [179] (referred to as MCGMM), the constrained K-means clustering method of [167] (referred to as COP_kmeans), the pairwise constraint propagation method (referred to as PCP) using semi-definite programming (SDP) of [105], and the non-linear metric learning method (referred to as NLML) of [158]. We implemented the multiple component per class version of [179], with the model order (number of components) selection and parameter initialization done the same way as for our method. The implementations for the methods in [155] and [167] by their respective authors were obtained from the Web [9]. For [167], learning was repeated for 10 random initializations, and the average values of the performance metrics are reported.

We implemented the methods PCP and NLML based on their algorithm descriptions given in [105] and [158]. We evaluated only the nonlinear version of NLML, which requires specifying the regularization parameter α, the reduced dimensionality d', and the window parameter

[9] Code for [155] was obtained from `http://www.openu.ac.il/home/shental/`, and code for [167] was obtained from `http://www.cs.ucdavis.edu/~davidson/constrained-clustering/CAREER/CAREER.html`

w. In [158], the authors suggest a heuristic choice of values for these hyperparameters which does not require any search. However, in our experiments, this choice typically resulted in poor performance for NLML compared to the other methods. In an attempt to evaluate the "best" case performance for NLML, we repeated the learning for different values of the three hyperparameters[10] and report the *best* performance, obtained by choosing the hyperparameters corresponding to the largest normalized mutual information value. For PCP, the open source SDP solver CSDP 6.0.1 [11] was used in the implementation. For this method, the authors suggest a range of values from which to choose the hyperparameter σ, but they do give an objective criterion for choosing σ. Therefore, we varied σ in the suggested range, and again report the *best* performance of this method by choosing σ corresponding to the largest normalized mutual information value. For both [105] and [158], the K-means (or kernel K-means) clustering was repeated from 20 random initializations, and the average performance measures are reported. Note that this method of choosing hyperparameters for NLML and PCP would not be possible in a realistic scenario where test set class labels are not known. However, this essentially gives an upper bound on the performance of these methods.

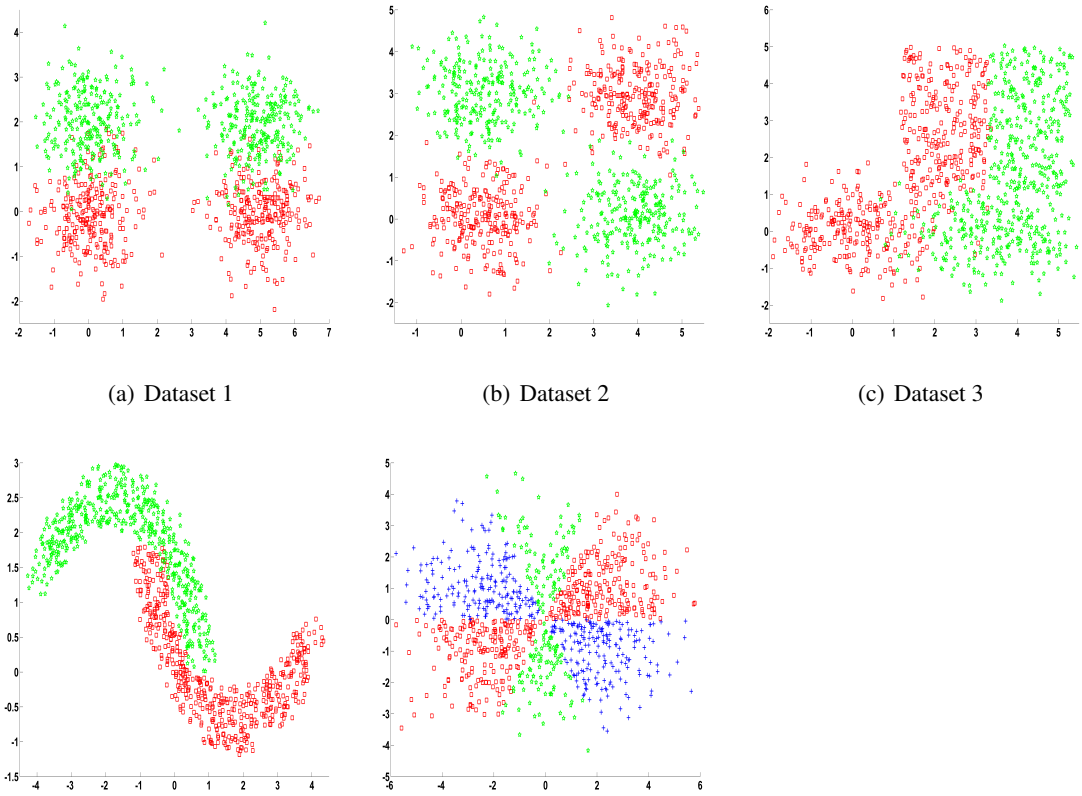
For our method, we evaluate both the single component per class (referred to as SCP (single component parameterized)) and the multiple components per class (referred to as MCP (multiple components parameterized)) models. This will be useful to illustrate that in some cases, just by imposing space-partitioning, even when the classes are *n*ot flexibly modeled with multiple components, one can obtain better solutions compared to methods which do not achieve constraint propagation. Whereas the other methods all require specification of the number of classes in the dataset, MCP and MCGMM can automatically estimate the number of classes. However, for fair comparison, we also provide the number of classes as input to MCP and MCGMM. We separately evaluate the accuracy of class number estimation for our method in section 3.4.6. For the mixture model based methods, the type of component covariance matrix (full or diagonal), and whether all components use distinct covariance matrices or tied ones (across all components) are decided based on (i) the number of samples and feature dimension of the data and (ii) the covariance type that leads to the solution with minimum BIC when MLE is performed on the data without any constraints.

In all experiments, the set of constraints is created by randomly selecting pairs of points, without replacement, from the entire dataset, and using the true class labels to determine whether it is an ML or a CL constraint [12]. Note that while this method can lead to more CL constraints

[10] We used a search set of nine values in $(0.001, 0.8)$ for α, a search set of nine values in $\{1, 2, \ldots, M\}$ (where M is the number of samples involved in ML constraints) for d', and a search set of nine values in $(\bar{d}/16, 5\,\bar{d})$ (where $\bar{d}$ is the average pairwise distance between the data samples) for w.

[11] https://projects.coin-or.org/Csdp/

[12] The class labels are used only for the purpose of generating the constraints and calculating the performance



(a) Dataset 1 (b) Dataset 2 (c) Dataset 3

(d) Dataset 4 (e) Dataset 5

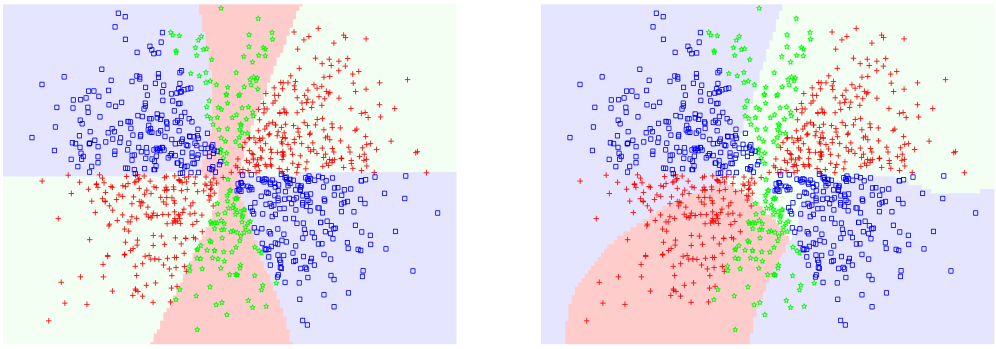
Figure 3.2. Plots of synthetically generated data used in experiments. Points from different classes are denoted by different colors and symbols.

than ML constraints for multi-class datasets, it is also a realistic way of creating constraint sets. From a given set of constraints, the larger set of entailed ML and CL constraints was then determined. In our experiments, the same entailed constraint set is provided as input to all the methods, and the F-score and NMI are reported as a function of the number of constraints used (number of constraint pairs divided by N). For each constraint set size, 10 randomly selected constraint sets are created, and the F-score and NMI values are averaged over these constraint sets.

3.4.3 Experiments on synthetic data

We first studied the performance of the methods on five synthetically generated 2D datasets, whose plots are shown in Fig. 3.2 (The dataset in Fig. 3.2(a) was also used in the illustrative

measure.



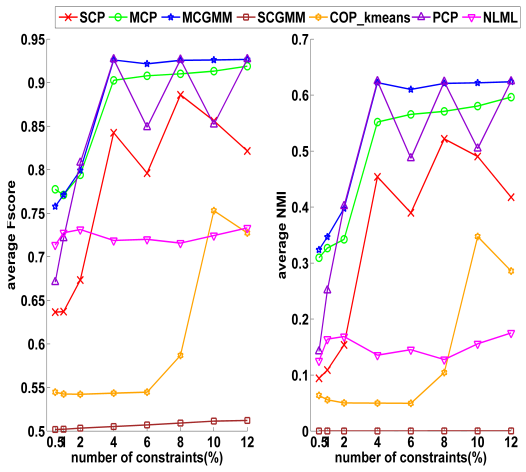
(a) Solution learned by the MCP method (b) Solution learned by the MCGMM method

Figure 3.3. Solutions learned by the MCP and MCGMM methods on the dataset in Fig. 3.2(e), for a particular set of constraints. The shaded regions show the learned class boundaries, and the samples are shown with their true class labels using different colors and symbols. For MCGMM, the class membership discontinuities at the location of the constrained samples are not shown.

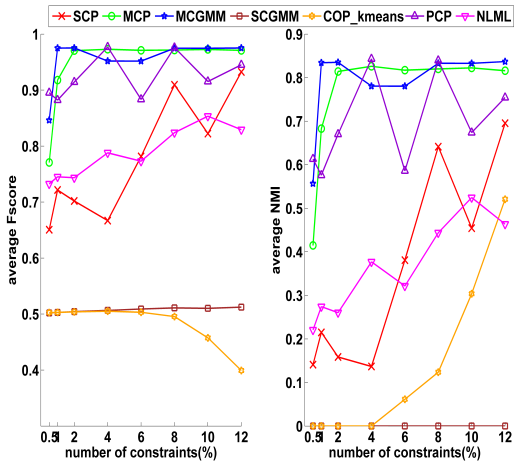
example in Fig. 3.1). The average F-score and average NMI, as a function of the constraint set size are shown in Fig. 3.4. Among the methods which use a single component (cluster) per class – COP_kmeans, SCGMM, and SCP, the first two learn poor or suboptimal solutions on all the five datasets. The method SCP, however, learns better solutions on datasets 1, 2, 3, and 5 (Fig. 3.4(a), 3.4(b), 3.4(c), and 3.4(e)). The methods MCGMM and MCP, both allow flexible modeling of classes with multiple components, but MCGMM does not achieve constraint propagation in its solution. This limitation can be clearly observed on datasets 4 and 5 (Fig. 3.4(d) and 3.4(e)), especially for dataset 5, where MCGMM models the data well but satisfies many constraints by simply introducing class membership discontinuities. The solutions learned by MCP and MCGMM on dataset 5, for one of the constraint sets is shown in Fig. 3.3. In Fig. 3.3(b), we observe that a majority of the samples from the red (plus) and green (star) classes are grouped incorrectly. Overall, on these synthetically generated datasets our method MCP has a significant performance improvement compared to the other methods, including the non-parametric kernel based method PCP, and the nonlinear metric learning method NLML.

3.4.4 Experiments on real data

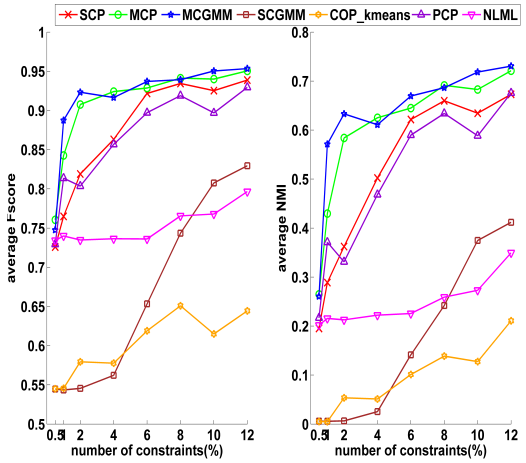
In this section, we evaluate the performance of the methods on ten datasets from the UC Irvine machine learning repository [1] and a real image segmentation dataset. A summary of the UCI datasets is given in Table 3.1. We applied principal components analysis (PCA) as a preprocess-



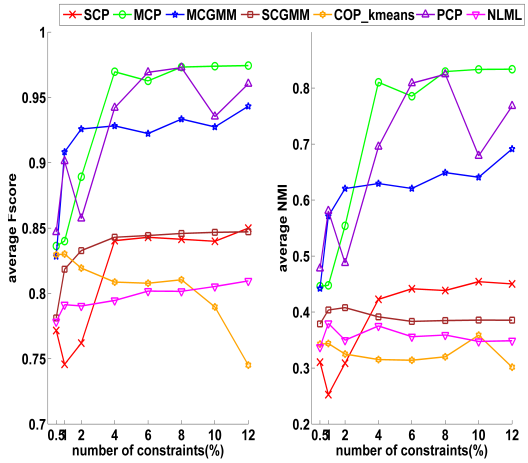
(a) Dataset 1



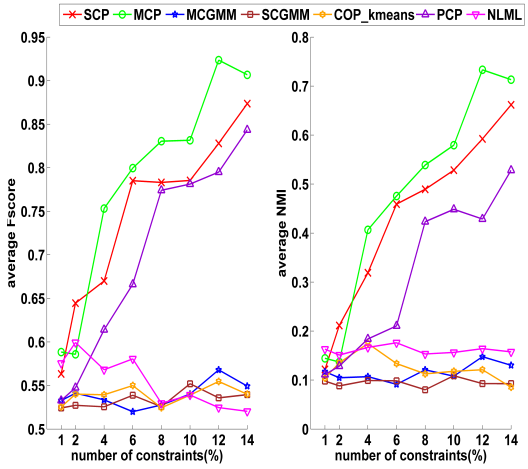
(b) Dataset 2



(c) Dataset 3



(d) Dataset 4



(e) Dataset 5

Figure 3.4. Average F-score and average NMI vs. number of constraints for all methods on the synthetically generated datasets shown in Fig. 3.2.

ing step, retaining only the dimensions which account for 99.5% of the total variance [13]. This was done in order to allow the mixture model based methods to learn better parameter estimates, and also to avoid the possibility of rank deficient covariance matrices. On the datasets for which PCA does not result in any dimension reduction, we used the original set of features. The av-

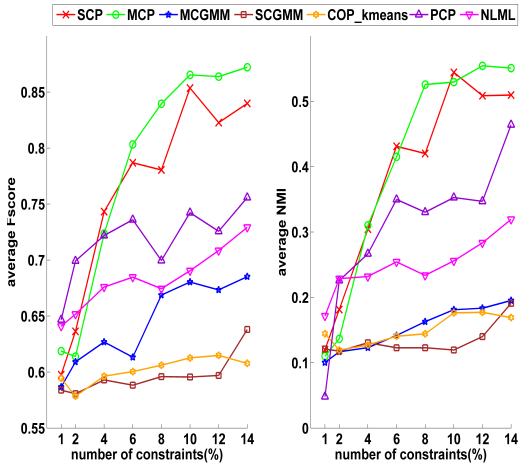
Table 3.1. Summary of the data sets used in experiments. N – number of samples, d' – original number of features, d – number of features after applying PCA, L^* – number of classes

Dataset	N	d'	d	L^*
Balance Scale	625	4	4	3
Breast Cancer (diagnostic)	569	30	7	2
Cardiotocography	2126	21	7	3
Image Segmentation	2098	19	6	7
Ionosphere	351	34	24	2
Pen-based Recognition	5499	16	14	10
Pima Indians	733	8	6	2
Vehicle Silhouettes	846	18	8	4
Wall Following Robot	5456	4	4	4
Waveform Database	5000	21	21	3

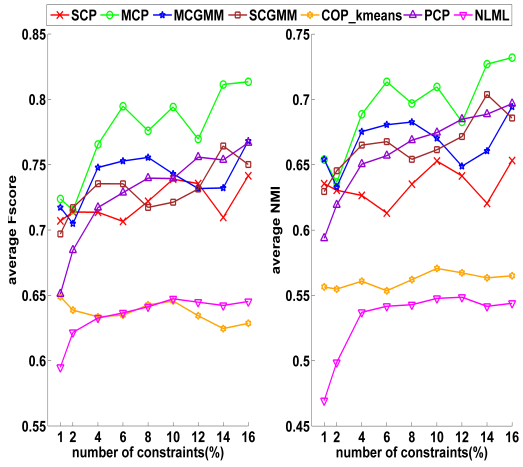
erage F-score and average NMI curves for the different methods, as a function of the constraint set size, are shown in Fig. 3.6. For the PCP method, we were unable to obtain results for the datasets Pen-based recognition, Wall following robot, and Waveform database (which have more than 5000 samples), because of its very long execution time (more than 24 hours for solving the SDP problem for a single σ value).

The methods MCP and SCP have a large performance improvement over others on the datasets Balance scale (Fig. 3.5(a)), Cardiotocography (Fig. 3.5(c)), and Wall following robot (Fig. 3.5(e)), and a relatively smaller improvement on Image segmentation (Fig. 3.5(b)). It is interesting to note that on the Cardiotocography dataset, the MCP solution does not satisfy about 10% of the constraints, but still has good generalization performance compared to other methods, which either impose hard constraint satisfaction (SCGMM, COP_kmeans, PCP, and NLML) or satisfy most of the constraints (MCGMM). Both MCP and MCGMM, which allow multiple components per class, have a large performance improvement on the Pen-based recognition dataset (Fig. 3.6(b)). On the Ionosphere dataset (Fig. 3.5(d)), the method PCP performs a lot better than the rest of the methods. But note that we are reporting the *best* performance

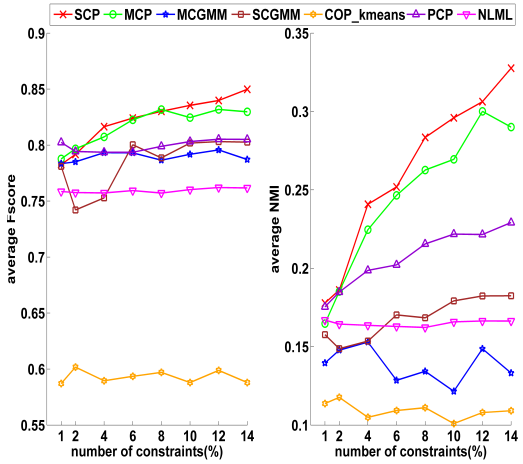
[13] Despite this conservatively large threshold, we found that the dimensionality could be significantly reduced on some of the datasets.



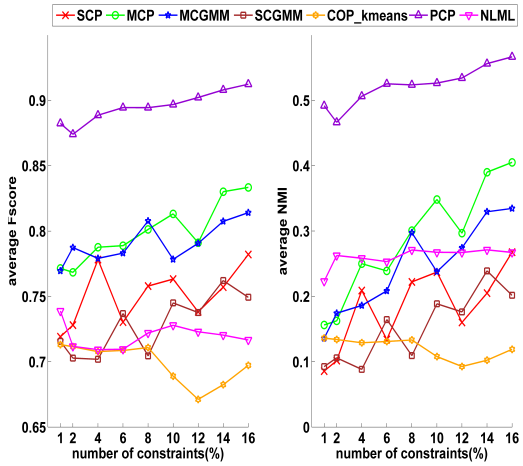
(a) Balance scale



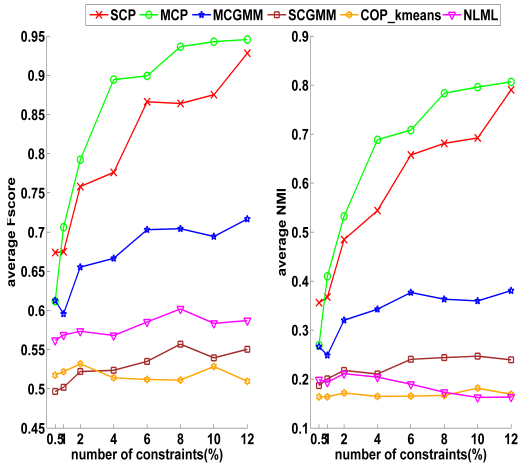
(b) Image segmentation



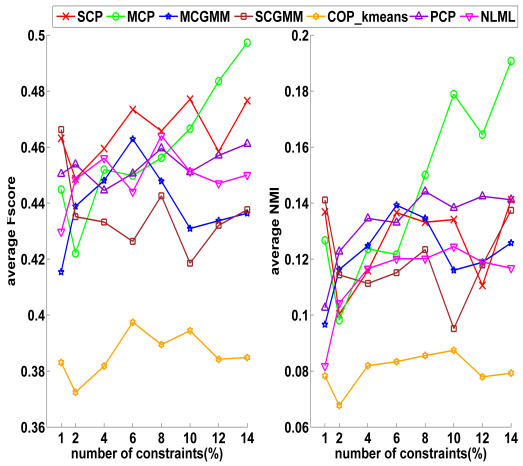
(c) Cardiotocography



(d) Ionosphere



(e) Wall following robot



(f) Vehicle silhouettes

Figure 3.5. Average F-score and average NMI vs. number of constraints for all methods on the UC Irvine datasets.

of PCP, using the hyperparameter σ that gives the largest NMI value. Also, on this dataset, we observed that the large F-score and NMI values of PCP were not obtained for any other σ value in the candidate set. The performance of NLML is only comparable and in some cases worse than the other methods used for comparison, even with its hyperparameter tuning.

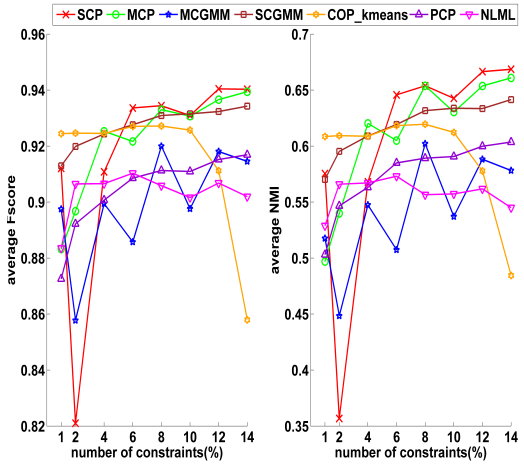
Additionally, we evaluated the methods on a texture image segmentation task from the USC-SIPI image database [14]. Similar to the approach in [107], we chose four Brodatz texture images to represent four underlying classes. The data samples were created by dividing the images into 16 x 16 blocks, and rearranging them into 256 dimensional feature vectors, following which PCA was applied to reduce the dimensionality to 16. ML and CL constraint sets of different sizes were created as described in section 3.4.2, and the average performance of the methods as a function of the number of constraints is shown in Fig. 3.6(e). The F-score and NMI of MCP, SCP, and MCGMM are all similar, and better compared to the other methods, but there is not much improvement as the number of constraints is increased.

3.4.5 Discussion of results

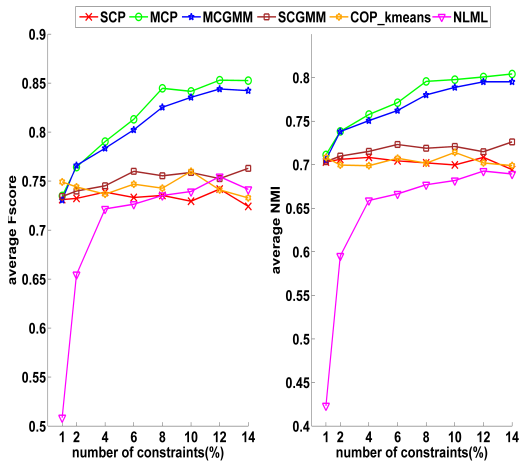
3.4.5.1 When is space-partitioning useful?

It is interesting to note that when the data patterns from each class form distinctive clusters which can be well modeled by the assumed parametric mixture model, the solutions found by the mixture model based methods which do not impose space-partitioning are usually accurate. For example, the solution learned by MCGMM on synthetic datasets 1 and 2 (see Fig. 3.4(a) and 3.4(b)). In these solutions, space-partitioning is obtained as a result of the *cluster* structure in the data – it is not explicitly imposed by the method in order to satisfy the constraints. Hence, the constraints are mainly informative only about which clusters belong to the same class, and which ones do not. This idea is also supported on the Waveform dataset from the UC Irvine machine learning repository. From the performance plot in Fig. 3.6(c), we see that the methods MCGMM, SCGMM, and MCP have almost no improvement in F-score as the number of constraints is increased. This suggests that the constraints are not very informative on this dataset. Indeed, we found that unsupervised (without the constraints) MLE learning of a GMM, with each component modeling a class, gets nearly the same average F-score and NMI as the other methods. On the other hand, consider synthetic dataset 5 (Fig. 3.2(e)). Here, a method which models the data well, but only satisfies the constraints locally (without propagating them to the unconstrained samples) does not learn an accurate solution. From this standpoint, the constraints are more informative on this dataset, and imposing space-partitioning is beneficial. Such a sce-

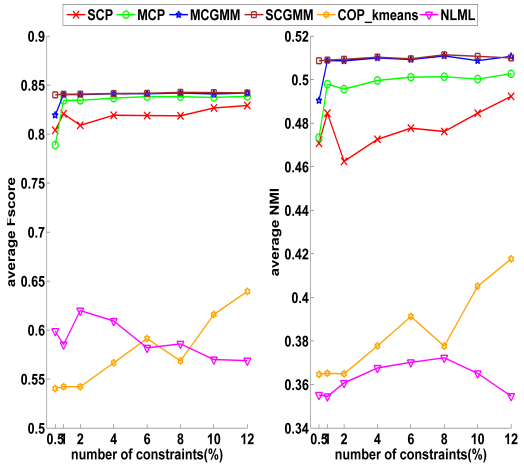
[14] This dataset was downloaded from `http://sipi.usc.edu/database/`.



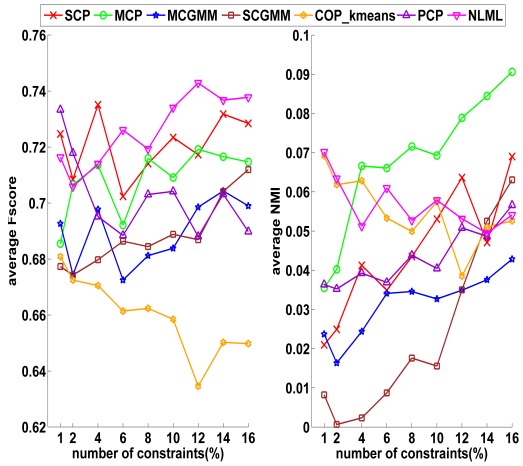
(a) Breast cancer



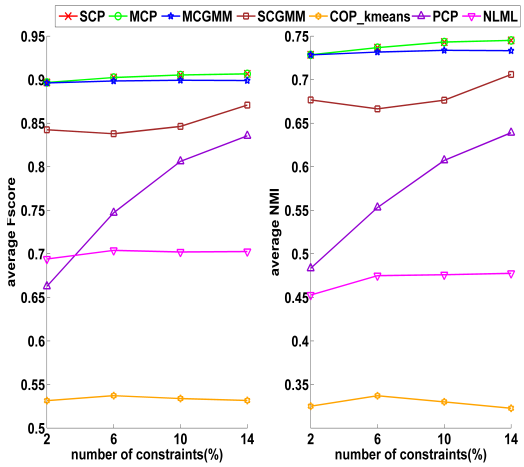
(b) Pen-based recognition



(c) Waveform database

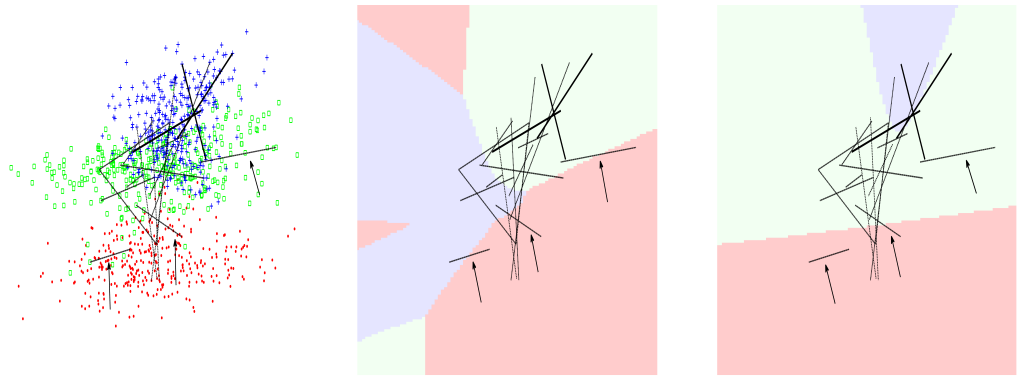


(d) Pima indians



(e) Texture image segmentation

Figure 3.6. Average F-score and average NMI vs. number of constraints for all methods on the UC Irvine and the texture image segmentation datasets.



(a) Dataset with true class labels and pairwise constraints

(b) Class boundaries learned by the MCP method

(c) Class boundaries learned by the MCGMM method

Figure 3.7. Example illustrating the effect of class overlap on the solutions learned by the MCP and MCGMM methods on a dataset with three classes, randomly generated from a Gaussian mixture. The true class labels of samples (markers with different symbols and colors), must-links (solid lines), and cannot-links (disconnected lines) are shown in Fig. 3.7(a). The class boundaries learned by MCP and MCGMM are shown in Fig. 3.7(b) and Fig. 3.7(c) respectively.

nario appears to play out on real datasets too, as evidenced by the performance plots in Fig. 3.5(e), Fig. 3.5(c), and 3.5(a).

3.4.5.2 Class overlap and inadequate number of constraints

While space-partitioning is a desirable property in the solution, in some situations where the classes have significant overlap, and/or when the number of constraints is very small, imposing space partitioning can actually mis-specify the class boundaries. The methods which do not impose space-partitioning, however, will not be affected much in such a scenario because they satisfy the constraints sample-wise. We studied the performance of the methods MCP and MCGMM under such a scenario by creating datasets by sampling from randomly generated Gaussian mixtures, with varying amounts of class overlap and a small number of constraints. One such illustrative example is shown in Fig. 3.7. The dataset has three classes. The true class labels of all samples, and the pairwise constraints are shown in Fig. 3.7(a). The class boundaries learned by MCP and MCGMM are shown in Fig. 3.7(b) and Fig. 3.7(c) respectively. The three CL constraints pointed out with arrows fall in regions where there is class overlap, and hence these constraints mislead the MCP method to learn incorrect class boundaries as shown in Fig. 3.7(b). On the other hand the solution of the MCGMM method in Fig. 3.7(c) satisfies these constraints and still has many CL pairs contained in regions that belongs to the same predicted class (group). Though the MCP method may learn suboptimal decision boundaries in such scenarios, it should be noted that the learned boundaries are completely consistent with the provided

constraint information. When the underlying class conditional distributions of the data can be captured by the model assumptions of the MCP method, it should be able to approximate the true class boundaries as more informative constraints are provided.

3.4.6 Evaluating the class number estimate

Here, we evaluate how close our class number estimate is to the true number of classes on three multi-class datasets from the UC Irvine repository: Image segmentation, Pen-based recognition, and Wall following robot. The constraint set size is varied over a small (2%), intermediate (6%), and large value (12%), and for each constraint set size, the average and standard deviation of L over ten randomly chosen constraint sets is calculated. The results are summarized in Table 3.2. We observe that the estimated number of classes is close to the true number of classes.

Table 3.2. Average class number estimate and its standard deviation (in paranthesis) for the MCP method on multi-class UC Irvine datasets.

Dataset	L^*	L		
		2%	6%	12%
Image Segmentation	7	8 (0.89)	6.9 (0.7)	6.8 (0.6)
Pen-based recognition	10	8.9 (0.7)	10.3 (0.46)	10.6 (0.49)
Wall following robot	4	4.2 (0.6)	3.8 (0.4)	3.8 (0.6)

3.4.7 Evaluating the choice of number of components

Our method of selecting the number of components was described in section 3.2.3.2. Fig. 3.8 shows the average F-score and average NMI of the MCP method as a function of the number of components on three UCI data sets (Ionosphere, Vehicle silhouettes, and Pen-based recognition), for a fixed constraint set size of 10%. The values are averaged over 10 randomly selected constraint sets. For a given number of components, β is chosen as described in section 3.2.3.1. Also, the average number of components chosen by our method, and the average number of components at the BIC minimum are indicated in the figure as K_1 and K_2 respectively. We observe that K_1 and K_2 are not very different on these datasets. Also, the F-score and NMI values of the MCP method do not have a lot of variance once the number of components is larger than a certain value. Similar results were observed on other UCI datasets. Accordingly, our model selection approach is reasonably supported by the classification results.

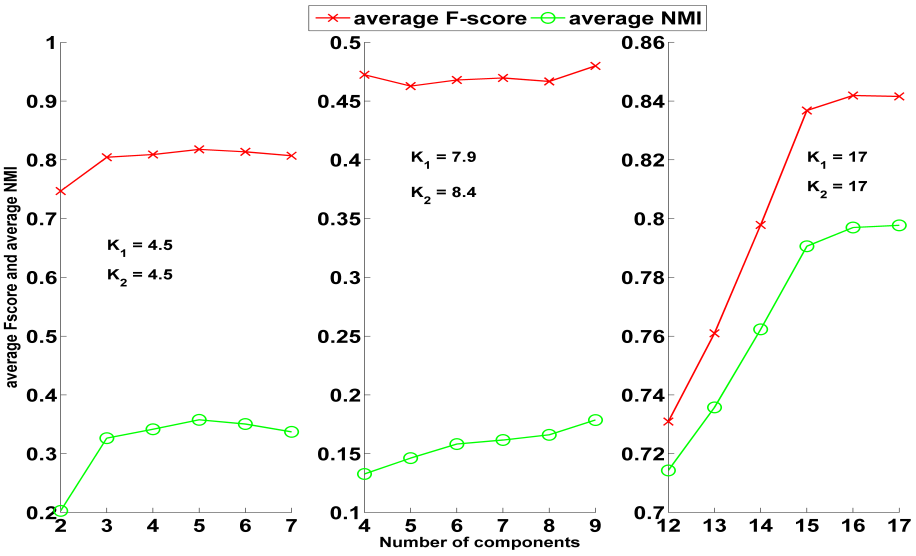


Figure 3.8. Plot of the average F-score and average NMI of the MCP method as a function of the number of mixture components for the UCI datasets (i) Ionosphere, (ii) Vehicle silhouettes, and (iii) Pen-based recognition (from left to right) for a constraint set size 10%. The average number of components selected by the method described in section 3.2.3.2 K_1, and the average number of components at the BIC minimum K_2 are also shown.

3.4.8 Run-time analysis of our method

The asymptotic run-time complexity of our method is given by $\mathcal{O}((K_{\mathrm{f}} - K_{\mathrm{i}})\, N_\beta\, N_{\mathrm{EM}}\, C_{\mathrm{EM}})$, where

$$C_{\mathrm{EM}} = T_{\max}\left(N\, K_{\mathrm{f}}\, d^2 + (|\mathcal{I}_c| + |\mathcal{I}_m|)\, K_{\mathrm{f}}^2\, L_{\max}\, d^2\right). \tag{3.21}$$

In the above expression, $[K_{\mathrm{i}}, K_{\mathrm{f}}]$ specifies the range of the number of components, N_β is the number of β values, N_{EM} is a bound on the number of EM cycles, C_{EM} (given by (3.21)) is a bound on the number of computations in each EM step, and $T_{\max}$ is a bound on the number of gradient descent steps in each E-step optimization. The average running times of the methods MCP, MCGMM, PCP, SCGMM, and NLML (with hyperparameter search) on three UCI datasets for a constraint set size 10% are reported in Table 3.3. All the methods were run on a computer with a 3.0 GHZ Intel Xeon 3160 Dual core processor and 8 GB memory. The methods MCP and PCP have relatively large running times compared to the other methods. In the case of our method MCP, this is caused by the nontrivial E-step, and the search over hyperparameter β and the number of components. Note that MCP achieves significantly better results than PCP on most data sets, and thus better exploits its computational cycles. Also, our method is amenable to parallelization since the learning for different β values and different number of components are independent of each other. Since the number of model parameters and the running time of

our method have squared dependence on the number of features d, for high dimensional data these can be reduced by using flexible models such as mixture of factor analysis [58].

Table 3.3. Average run-times (in seconds) of the methods on some UCI datasets for a constraint set size 10%

Dataset	Average run-time (seconds)				
	MCP	MCGMM	PCP	SCGMM	NLML
Cardiotocography	7805	86	22066	5	4882
Balance scale	813	38	1514	4	246
Vehicle silhouettes	2151	85	3237	6	210

Chapter 4

Semi-supervised domain adaptation of mixture model based classifiers with imposed space-partitioning

4.1 Introduction

In the previous chapter, we presented a method for semi-supervised learning from pairwise-sample constraints which imposes space-partitioning and smoothness in the solution, thus ensuring that there is propagation of constraint information to all the samples. Here, we apply the solution approach developed there to the problem of semi-supervised domain adaptation of a mixture model based classifier. As discussed in Chapter 1, the semi-supervised domain adaptation problem [45], [16], [24], [25], [82] considers a scenario where the target domain of interest has a small labeled data set and a relatively large unlabeled data set, while a related source domain has a fully labeled data set available - one that is large enough to reliably learn a supervised classifier. In order to learn a classifier specific to the target domain data, it may not be reliable to directly utilize the labeled data set from the source domain since the underlying joint distribution of the feature vector $\underline{X}$ and the class C may be different in the two domains. However, in order to leverage the existing trained classifier and/or the fully labeled data set in the source domain in a useful way and in fact, to also avoid possible degradation in performance of the target domain classifier during adaptation, one would have to assume that the joint distributions of $\underline{X}$ and C are not *very* different in the source and target domains. Under such an assumption, we consider

the problem of adapting the parameters of a mixture model based classifier from the source to the target domain using the combined labeled and unlabeled target domain data sets.

Recall that in section 1.4 of chapter 1, we discussed some plausible realistic problem scenarios where classifier domain adaptation will be useful. One such scenario is when there is a contextual difference in the way the data is generated/recorded in the two domains. For example, the data in the source and target domains may be obtained at different times, or at different locations. Also, there could be situations where the class-conditional distribution of the features may change conditioned on the value of a latent variable (which "signifies" the domain). For example, in a computer networking scenario, the demand profile for a certain application (packet-flow) could be different at two sites, such that the training data was captured when the demand (total bytes or packets) was low and the test data was captured when the demand was high. Such a scenario also commonly occurs in the problem of automatic speech recognition [50], [49], where the speech models (usually continuous density hidden Markov models) learned using training data from a given set of speakers may not be directly suitable for recognizing speech from a different set of speakers, especially when there are differences like native versus non-native speakers of a language.

In this work, we focus on adaptation of a generative classifier, where the conditional distribution of the feature vector given each class is modeled using a mixture of parametric distributions (typically Gaussian), and the maximum *a posteriori* (MAP) rule is applied to the plug-in Bayes class posterior (using the estimated model) to make class predictions. This approach, also known as mixture discriminant analysis (MDA) [65], [53], [62], is capable of learning complex decision boundaries given sufficient labeled training data and with suitably chosen model complexity. We assume that the target domain has a small set of labeled samples, and a relatively large number of unlabeled samples just like in semi-supervised learning [33], [182], but here we additionally exploit the classifier model trained using the labeled source domain data. Specifically, starting from the source domain classifier model, our method maximizes the likelihood of target domain data, while constraining the solution to agree as closely as possible with the available class label information in the target domain. This is achieved via an expectation maximization (EM) algorithm, where the joint posterior distribution of the latent variables given the data is constrained using a smooth parametric mean-field (variational) approximation in the E-step, in order to ensure that space-partitioning implications are gleaned from the labeled target domain samples. The motivation for using a parametric mean-field approximation in the E-step is similar to that of the method developed in chapter 3, with the difference being that, here we attempt to achieve *label propagation* to neighboring unlabeled samples, whereas the method in chapter 3 achieves constraint propagation in the solution.

The rest of the chapter is organized as follows. In the next section, we formulate our problem, discuss two different solution approaches, and motivate through an example why one approach is preferred over the other. In section 4.3, we discuss the overall model learning strategy, and also propose a validation score used to eliminate potentially *poor* solutions (in terms of target domain classification performance) from the set of candidate solutions. A discussion of related domain adaptation approaches in the literature is given in section 4.4. Finally, in section 4.5, we discuss the methods used for performance comparison, and present experimental results both on synthetically generated data sets (with changes introduced in the distributions generating the source and target domain data), and on publicly available Internet packet-flow traffic data from different temporal and spatial domains.

4.2 Method formulation

Consider a labeled source domain data set $\mathcal{X}_l^{(s)} = \{(\underline{x}_i^{(s)}, c_i^{(s)}),\ i = 1, \ldots, N_l^{(s)}\}$, where $\underline{x}_i^{(s)} \in \mathbb{R}^d$ is the feature vector and $c_i^{(s)} \in \mathcal{C} \equiv \{1, \ldots, K\}$ is the class label, from a set K classes. We will develop the method for continuous-valued features, and outline ideas for extending the method to mixed continuous and categorical valued features in chapter 5. The target domain data consists of a labeled subset $\mathcal{X}_l^{(t)} = \{(\underline{x}_i^{(t)}, c_i^{(t)}),\ i = 1, \ldots, N_l^{(t)}\}$, and an unlabeled subset $\mathcal{X}_u^{(t)} = \{\underline{x}_i^{(t)},\ i = N_l^{(t)}+1, \ldots, N_l^{(t)}+N_u^{(t)}\}$, where $\underline{x}_i^{(t)} \in \mathbb{R}^d$. It is useful to define the following: the index set of target domain labeled samples $\mathcal{I}_l^{(t)} = \{1, 2, \ldots, N_l^{(t)}\}$, the index set of all target domain samples $\mathcal{I}^{(t)} = \{1, 2, \ldots, N_l^{(t)} + N_u^{(t)}\}$, and the set of all feature vectors in the target domain data set $\mathcal{X}^{(t)} = \{\underline{x}_i^{(t)}\ \ \forall i \in \mathcal{I}^{(t)}\}$. It is assumed that the features in the source and target domain are measured in the same way, and have the same semantic meaning. It is also assumed the same set of classes are present in the two domains, albeit perhaps with very different class prior probabilities on the two domains [1].

4.2.1 Classification model

An MDA classifier for the source domain is learned by modeling the conditional density of the feature vector $\underline{X}$ given the class C as a finite mixture of Gaussian densities, *i.e.*,

$$f_{\underline{X}\,|\,C}(\underline{x}\,|\,c, \omega_c) \;=\; \sum_{l \in \mathcal{M}_c} \alpha_{cl}\, \mathcal{N}(\underline{x}\,; \underline{\mu}_{cl}, \mathbf{\Sigma}_{cl}),\ \ \forall c \in \mathcal{C},$$

[1] Although classes may be missing at random in the target domain labeled subset, in this work we do not consider "new class discovery" [179] in the target domain.

where $\mathcal{N}(\underline{x}\,;\underline{\mu},\boldsymbol{\Sigma})$ is the multivariate Gaussian density with mean vector $\underline{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, $\mathcal{M}_c$ is the index set of components in the mixture model for class c, and $\omega_c = \{(\alpha_{cl}, \underline{\mu}_{cl}, \boldsymbol{\Sigma}_{cl}),\ l \in \mathcal{M}_c\}$ is the set of parameters of the mixture model for class c. Here α_{cl} is the prior probability of component l given class c which has to satisfy the non-negativity and sum-to-one (over $l \in \mathcal{M}_c$) probability constraint, $\underline{\mu}_{cl}$ is the component mean vector, and $\boldsymbol{\Sigma}_{cl}$ is the component covariance matrix which has to satisfy the symmetry and positive definiteness constraint. Let $\pi_c,\ \forall c \in \mathcal{C}$ denote the class prior probabilities which are non-negative and sum to 1 over all the classes. The set of all parameters, $\Theta = \{\Theta_c = (\pi_c, \omega_c),\ \forall c \in \mathcal{C}\}$, can be estimated using maximum-likelihood (ML) or maximum-a-posterior (MAP) estimation (in a Bayesian setting with conjugate prior distributions) on the labeled source domain data. This is essentially a straightforward application of the EM algorithm [48], [111]. The complexity of the mixture model for each class increases with increase in the number of components, and also with the choice of the covariance matrix (unrestricted or diagonal, distinct or shared across the components). In order to make these choices, we use the Bayesian information criterion (BIC) [152] which is a theoretically grounded criterion to select the complexity of the model. For a fixed model size chosen by the BIC, suppose the set of parameters estimated based on the labeled source domain data is denoted by $\Theta^{(s)}$, then the plug-in Bayes classification rule is given by

$$\hat{C}(\underline{x}) \;=\; \arg\max_{c\in\mathcal{C}}\; P(C=c\,|\,\underline{x},\Theta^{(s)}) \;=\; \arg\max_{c\in\mathcal{C}}\; \pi_c^{(s)}\, f_{\underline{X}\,|\,C}(\underline{x}\,|\,c,\omega_c^{(s)}). \tag{4.1}$$

The goal of this problem is to adapt the classifier parameters $\Theta^{(s)}$ using the combined labeled and unlabeled data samples in the target domain, such that the MDA classifier based on the adapted parameters has improved classification performance for the target domain data, ideally, better than that of a classifier that is based purely on supervised or semi-supervised learning from the target domain data.

4.2.2 Review of variational approximation

In order to set the tone for our learning objective and optimization approach, we first review the method of variational approximation applied to the E-step of the Expectation-Maximization (EM) algorithm. Consider the (incomplete data) log-likelihood [48] of the set of all feature vectors in the target domain data ($\mathcal{X}^{(t)}$)

$$\mathcal{L}(\Theta) = \;\log P(\mathcal{X}^{(t)}\,|\,\Theta) \;=\; \sum_{i\in\mathcal{I}^{(t)}} \log\left[\sum_{c\in\mathcal{C}}\sum_{l\in\mathcal{M}_c} \pi_c\,\alpha_{cl}\,\mathcal{N}(\underline{x}_i^{(t)}\,;\underline{\mu}_{cl},\boldsymbol{\Sigma}_{cl})\right].$$

We identify the set of binary, latent random variables $\mathcal{Z} = \{Z_{icl},\ \forall i \in \mathcal{I}^{(t)}, \forall c \in \mathcal{C}, \forall l \in \mathcal{M}_c\}$, where $Z_{icl} = 1$ if the (possibly unknown) class label of sample $\underline{x}_i^{(t)}$ is c and if $\underline{x}_i^{(t)}$ is generated from the conditional density of component $l \in \mathcal{M}_c$; else $Z_{icl} = 0$. The complete data log-likelihood is then given by

$$\mathcal{L}_c(\Theta) = \log P(\mathcal{X}^{(t)}, \mathcal{Z} \,|\, \Theta) = \sum_{i \in \mathcal{I}^{(t)}} \sum_{c \in \mathcal{C}} \sum_{l \in \mathcal{M}_c} Z_{icl} \log \left[\pi_c \, \alpha_{cl} \, \mathcal{N}(\underline{x}_i^{(t)}; \underline{\mu}_{cl}, \mathbf{\Sigma}_{cl}) \right]. \quad (4.2)$$

Let $q(c, l \,|\, \underline{x}_i) \equiv P(C = c, M_c = l \,|\, \underline{x}_i)$ denote an arbitrary probability mass function (pmf) on the class C and mixture component M_c (for class c) given feature vector $\underline{x}_i$, satisfying $\sum_{c \in \mathcal{C}} \sum_{l \in \mathcal{M}_c} q(c, l \,|\, \underline{x}_i) = 1$. Consider the joint-posterior distribution of the latent variables $\mathcal{Z}$ given $\mathcal{X}^{(t)}$ that has the factorized form

$$P^{(0)}\left(\mathcal{Z} \,|\, \mathcal{X}^{(t)}\right) = \prod_{i \in \mathcal{I}^{(t)}} \prod_{c \in \mathcal{C}} \prod_{l \in \mathcal{M}_c} q(c, l \,|\, \underline{x}_i^{(t)})^{\, Z_{icl}}. \quad (4.3)$$

Observing that $\mathbb{E}_{P^{(0)}}[Z_{icl}] = q(c, l \,|\, \underline{x}_i^{(t)})$, the expectation of the complete data log-likelihood with respect to the distribution $P^{(0)}(\mathcal{Z} \,|\, \mathcal{X}^{(t)})$ is given by [2]

$$\begin{aligned} \mathbb{E}_{P^{(0)}}\left[\log P(\mathcal{X}^{(t)}, \mathcal{Z} \,|\, \Theta)\right] &= \sum_{\mathcal{Z}} P^{(0)}(\mathcal{Z} \,|\, \mathcal{X}^{(t)}) \, \log P(\mathcal{X}^{(t)}, \mathcal{Z} \,|\, \Theta) \\ = \sum_{i \in \mathcal{I}^{(t)}} \sum_{c \in \mathcal{C}} \sum_{l \in \mathcal{M}_c} & q(c, l \,|\, \underline{x}_i^{(t)}) \log \left[\pi_c \, \alpha_{cl} \, \mathcal{N}\left(\underline{x}_i^{(t)}; \underline{\mu}_{cl}, \mathbf{\Sigma}_{cl}\right) \right]. \end{aligned} \quad (4.4)$$

Also, the entropy of the distribution $P^{(0)}(\mathcal{Z} \,|\, \mathcal{X}^{(t)})$ is given by

$$\begin{aligned} H\left[P^{(0)}(\mathcal{Z} \,|\, \mathcal{X}^{(t)})\right] &= -\sum_{\mathcal{Z}} P^{(0)}(\mathcal{Z} \,|\, \mathcal{X}^{(t)}) \, \log P^{(0)}(\mathcal{Z} \,|\, \mathcal{X}^{(t)}) \\ &= -\sum_{i \in \mathcal{I}^{(t)}} \sum_{c \in \mathcal{C}} \sum_{l \in \mathcal{M}_c} q(c, l \,|\, \underline{x}_i^{(t)}) \, \log q(c, l \,|\, \underline{x}_i^{(t)}). \end{aligned} \quad (4.5)$$

Suppose the true posterior distribution over the latent variables conditioned on the data and parameters is given by

$$P(\mathcal{Z} \,|\, \mathcal{X}^{(t)}, \Theta) = \frac{P(\mathcal{X}^{(t)}, \mathcal{Z} \,|\, \Theta)}{\sum_{\mathcal{Z}} P(\mathcal{X}^{(t)}, \mathcal{Z} \,|\, \Theta)}, \quad (4.6)$$

[2] We use $\sum_{\mathcal{Z}}$ as a shorthand notation for $\sum_{\underline{Z}_1 \in \Delta} \sum_{\underline{Z}_2 \in \Delta} \cdots \sum_{\underline{Z}_{N_u + N_l} \in \Delta}$, where $\underline{Z}_i$ is a vector consisting of the latent variables $Z_{icl},\ \forall c \in \mathcal{C},\ \forall l \in \mathcal{M}_c$, and Δ is the set of all $\{0, 1\}$-valued tuples of size $\sum_{c \in \mathcal{C}} |\mathcal{M}_c|$ such that in each tuple exactly one of the values is 1 and the rest are 0.

then we can derive a lower bound on the incomplete data log-likelihood as follows:

$$\begin{aligned}
\mathcal{L}(\Theta) \;&=\; \log P(\mathcal{X}^{(t)} \,|\, \Theta) \;=\; \sum_{i \in \mathcal{I}^{(t)}} \log \sum_{\mathcal{Z}} P(\mathcal{X}^{(t)}, \mathcal{Z} \,|\, \Theta) \\
&= \sum_{i \in \mathcal{I}^{(t)}} \sum_{\mathcal{Z}} P^{(0)}(\mathcal{Z} \,|\, \mathcal{X}^{(t)}) \;\log \frac{P(\mathcal{X}^{(t)}, \mathcal{Z} \,|\, \Theta)}{P(\mathcal{Z} \,|\, \mathcal{X}^{(t)}, \Theta)} \\
&= \sum_{i \in \mathcal{I}^{(t)}} \sum_{\mathcal{Z}} P^{(0)}(\mathcal{Z} \,|\, \mathcal{X}^{(t)}) \;\log \left[\frac{P(\mathcal{X}^{(t)}, \mathcal{Z} \,|\, \Theta)\; P^{(0)}(\mathcal{Z} \,|\, \mathcal{X}^{(t)})}{P(\mathcal{Z} \,|\, \mathcal{X}^{(t)}, \Theta)\; P^{(0)}(\mathcal{Z} \,|\, \mathcal{X}^{(t)})} \right] \\
&= \sum_{i \in \mathcal{I}^{(t)}} \sum_{\mathcal{Z}} P^{(0)}(\mathcal{Z} \,|\, \mathcal{X}^{(t)}) \;\log P(\mathcal{X}^{(t)}, \mathcal{Z} \,|\, \Theta) \;-\; \sum_{i \in \mathcal{I}^{(t)}} \sum_{\mathcal{Z}} P^{(0)}(\mathcal{Z} \,|\, \mathcal{X}^{(t)}) \;\log P^{(0)}(\mathcal{Z} \,|\, \mathcal{X}^{(t)}) \\
&+ \sum_{i \in \mathcal{I}^{(t)}} \sum_{\mathcal{Z}} P^{(0)}(\mathcal{Z} \,|\, \mathcal{X}^{(t)}) \;\log \frac{P^{(0)}(\mathcal{Z} \,|\, \mathcal{X}^{(t)})}{P(\mathcal{Z} \,|\, \mathcal{X}^{(t)}, \Theta)}. \\
&= \mathbb{E}_{P^{(0)}} \left[\log P(\mathcal{X}^{(t)}, \mathcal{Z} \,|\, \Theta) \right] \;+\; H \left[P^{(0)}(\mathcal{Z} \,|\, \mathcal{X}^{(t)}) \right] \;+\; D_{\mathrm{KL}}(P^{(0)}, P) \qquad (4.7)
\end{aligned}$$

where $D_{\mathrm{KL}}(P^{(0)}, P)$ is the Kullback-Leibler distance between the probability distributions $P^{(0)}(\mathcal{Z} \,|\, \mathcal{X}^{(t)})$ and $P(\mathcal{Z} \,|\, \mathcal{X}^{(t)}, \Theta)$, which is always non-negative and takes a value 0 if and only if $P^{(0)}(\mathcal{Z} \,|\, \mathcal{X}^{(t)}) = P(\mathcal{Z} \,|\, \mathcal{X}^{(t)}, \Theta)$. Therefore, we have a lower bound on the log-likelihood given by

$$Q(\Theta, P^{(0)}) = \mathbb{E}_{P^{(0)}} \left[\log P(\mathcal{X}^{(t)}, \mathcal{Z} \,|\, \Theta) \right] + H \left[P^{(0)}(\mathcal{Z} \,|\, \mathcal{X}^{(t)}) \right] \;\leq\; \mathcal{L}(\Theta). \qquad (4.8)$$

The above formulation lends an alternate view of the EM algorithm, as an iterative alternating maximization over the (latent variable) posterior distribution and the parameters [130]. Given fixed parameters Θ, in the E-step the lower bound $Q(\Theta, P^{(0)})$ is maximized over the space of all probability distributions. The formulation conveniently gives the solution to this problem as $P^{(0)}(\mathcal{Z} \,|\, \mathcal{X}^{(t)}) = P(\mathcal{Z} \,|\, \mathcal{X}^{(t)}, \Theta)$. This makes the lower bound equal to the incomplete data log-likelihood in the E-step. In the M-step, given the fixed distribution $P^{(0)}$, the lower bound $Q(\Theta, P^{(0)})$ is maximized over the parameters Θ. These two steps are iterated, with each iteration ascending monotonically in both $Q(\Theta, P^{(0)})$ and $\mathcal{L}(\Theta)$, until a local maximum of $\mathcal{L}(\Theta)$ is reached.

As we discussed in Chapter 3, when computation of the posterior distribution (over latent variables given the data) and the expectations relative to this distribution in the E-step are not analytically tractable, it is common to use a factorized mean-field approximation or a variational approximation [109], [78], [68], [138]. In the case of mean-field approximation, the E-step optimization is directly performed over the factors of the approximating distribution, while constraining them to be valid probability mass functions. In the case of variational approximation,

the factors of the approximating distribution are constrained to be smooth (not necessary in general), parametric functions, and the E-step optimization is performed over the parameters of the factors, which are referred to as *variational parameters*.

4.2.3 Optimization problem for classifier adaptation

For the semi-supervised setting in the target domain, our learning objective will be to maximize the lower bound $Q(\Theta, P^{(0)})$ (or equivalently minimize its negative) subject to a constraint on the expected error-rate on the labeled target domain samples (where the expectation is with respect to the distribution $P^{(0)}$), *i.e.*,

$$\begin{aligned} &\min_{\Theta,\, \{q(c,l|\underline{x}_i^{(t)})\}} -\mathbb{E}_{P^{(0)}}\left[\log P(\mathcal{X}^{(t)}, \mathcal{Z} \,|\, \Theta)\right] \;-\; H\left[P^{(0)}(\mathcal{Z} \,|\, \mathcal{X}^{(t)})\right] \\ &\text{such that } \frac{1}{N_l^{(t)}} \sum_{i \in \mathcal{I}_l^{(t)}} \sum_{\substack{c \in \mathcal{C} \\ c \neq c_i^{(t)}}} \sum_{l \in \mathcal{M}_c} q(c, l \,|\, \underline{x}_i^{(t)}) \;\leq\; \rho, \end{aligned} \tag{4.9}$$

where the expected complete data log-likelihood is given by (4.4) and the entropy is given by (4.5). Note that the constraint term can be expressed as

$$\frac{1}{N_l^{(t)}} \sum_{i \in \mathcal{I}_l^{(t)}} \sum_{\substack{c \in \mathcal{C} \\ c \neq c_i^{(t)}}} \sum_{l \in \mathcal{M}_c} q(c, l \,|\, \underline{x}_i^{(t)}) \;=\; \mathbb{E}_{P^{(0)}}[\, \frac{1}{N_l^{(t)}} \sum_{i \in \mathcal{I}_l^{(t)}} \sum_{\substack{c \in \mathcal{C} \\ c \neq c_i^{(t)}}} \sum_{l \in \mathcal{M}_c} Z_{icl}\,]. \tag{4.10}$$

Intuitively, the constraint term is a soft count of the assignments of labeled target domain samples to components that are not affiliated with the class label, and thus tries to discourage such assignments. The Lagrangian for the constrained optimization problem (4.9) (not explicitly including the term which would ensure that $q(c, l \,|\, \underline{x}_i)$ is a valid probability mass function) is given by

$$\begin{aligned} &\mathcal{F}(\Theta,\, \{q(c,l|\underline{x}_i^{(t)})\}) = -\mathbb{E}_{P^{(0)}}\left[\log P(\mathcal{X}^{(t)}, \mathcal{Z} \,|\, \Theta)\right] \;-\; H\left[P^{(0)}(\mathcal{Z} \,|\, \mathcal{X}^{(t)})\right] \\ +\;& \beta \sum_{i \in \mathcal{I}_l^{(t)}} \sum_{\substack{c \in \mathcal{C} \\ c \neq c_i^{(t)}}} \sum_{l \in \mathcal{M}_c} q(c, l \,|\, \underline{x}_i^{(t)}), \\ =\;& -\mathbb{E}_{P^{(0)}}[\log P(\mathcal{X}^{(t)}, \mathcal{Z} \,|\, \Theta) \;-\; \beta \sum_{i \in \mathcal{I}_l^{(t)}} \sum_{\substack{c \in \mathcal{C} \\ c \neq c_i^{(t)}}} \sum_{l \in \mathcal{M}_c} Z_{icl}\,] \;-\; H\left[P^{(0)}(\mathcal{Z} \,|\, \mathcal{X}^{(t)})\right] \end{aligned} \tag{4.11}$$

where $\beta \geq 0$ is the Lagrange multiplier which has absorbed the constant factor $N_l^{(t)}$. Different values of β in the Lagrangian correspond to different values of ρ in (4.9), and larger values of β

will constrain the error rate on the labeled target domain samples to be smaller. This optimization problem can also be solved using the framework of the EM algorithm that we just discussed in section 4.2.2. However, in this case, the optimal posterior distribution (over the latent variables) which minimizes (4.11) over *all* valid distributions (not just those that have the factorized form (4.3)) in the E-step is given, in its un-normalized form, by

$$\begin{aligned} P(\mathcal{Z} \,|\, \mathcal{X}^{(t)}, \Theta) \;\propto\; & \exp[\, \log P(\mathcal{X}^{(t)}, \mathcal{Z} \,|\, \Theta) \;-\; \beta \sum_{i \in \mathcal{I}_l^{(t)}} \sum_{\substack{c \in \mathcal{C} \\ c \neq c_i^{(t)}}} \sum_{l \in \mathcal{M}_c} Z_{icl} \,] \\ = \;& \exp[\, \sum_{i \in \mathcal{I}^{(t)}} \sum_{c \in \mathcal{C}} \sum_{l \in \mathcal{M}_c} Z_{icl} \, (\log[\pi_c \, \alpha_{cl} \, \mathcal{N}(\underline{x}_i^{(t)}; \underline{\mu}_{cl}, \mathbf{\Sigma}_{cl})] \;-\; \beta \, \delta(c \neq c_i^{(t)}) \, \delta(i \in \mathcal{I}_l^{(t)})) \,] \\ = \;& \prod_{i \in \mathcal{I}^{(t)}} \prod_{c \in \mathcal{C}} \prod_{l \in \mathcal{M}_c} \Big(\pi_c \, \alpha_{cl} \, \mathcal{N}(\underline{x}_i^{(t)}; \underline{\mu}_{cl}, \mathbf{\Sigma}_{cl}) \; \exp[\, -\beta \, \delta(c \neq c_i^{(t)}) \, \delta(i \in \mathcal{I}_l^{(t)})] \Big)^{Z_{icl}} \;, \end{aligned} \tag{4.12}$$

where we have made use of (4.2) in going from the first to the second step, and $\delta(\cdot)$ is a binary indicator which takes a value 1 (0) if the condition in its argument is satisfied (not satisfied). Since each term in the product (4.12) depends only on the latent variable of a single sample Z_{icl}, the normalization term is tractable to compute analytically, and given by

$$\prod_{i \in \mathcal{I}^{(t)}} \sum_{c \in \mathcal{C}} \sum_{l \in \mathcal{M}_c} \pi_c \, \alpha_{cl} \, \mathcal{N}(\underline{x}_i^{(t)}; \underline{\mu}_{cl}, \mathbf{\Sigma}_{cl}) \; \exp[\, -\beta \, \delta(c \neq c_i^{(t)}) \, \delta(i \in \mathcal{I}_l^{(t)})]. \tag{4.13}$$

From (4.12) and (4.13), the normalized form of the posterior distribution over the latent variables is given by

$$\begin{aligned} & P(\mathcal{Z} \,|\, \mathcal{X}^{(t)}, \Theta) \;= \\ & \prod_{i \in \mathcal{I}^{(t)}} \prod_{c \in \mathcal{C}} \prod_{l \in \mathcal{M}_c} \left(\frac{\pi_c \, \alpha_{cl} \, \mathcal{N}(\underline{x}_i^{(t)}; \underline{\mu}_{cl}, \mathbf{\Sigma}_{cl}) \; \exp[\, -\beta \, \delta(c \neq c_i^{(t)}) \, \delta(i \in \mathcal{I}_l^{(t)})]}{\sum_{k \in \mathcal{C}} \sum_{m \in \mathcal{M}_c} \pi_k \, \alpha_{km} \, \mathcal{N}(\underline{x}_i^{(t)}; \underline{\mu}_{km}, \mathbf{\Sigma}_{km}) \; \exp[\, -\beta \, \delta(k \neq c_i^{(t)}) \, \delta(i \in \mathcal{I}_l^{(t)})]} \right)^{Z_{icl}} . \end{aligned} \tag{4.14}$$

Note that this is different from the scenario in chapter 3, where the normalization term in the optimal posterior distribution over latent variables, (3.6), was not tractable to compute, and hence the factorized mean-field approximation had to be invoked. However, in this case, $P(\mathcal{Z} \,|\, \mathcal{X}^{(t)}, \Theta)$ is tractable to compute, and has the factorized form (4.14). In the sequel, we will show that using this optimal posterior distribution in the E-step of the EM algorithm is not desirable from the standpoint of achieving space-partitioning and smoothness in the solution, which are essential in order to achieve label propagation.

The goal of the classifier adaptation problem is to minimize the objective function (4.11). In

order to utilize the model solution learned on the labeled source domain data, we can initialize the EM algorithm with the parameters $\Theta = \Theta^{(s)}$. Depending on the difference between the underlying joint distributions which generated the source and target domain data, this initialization for the classifier parameters may or may not be the best choice. One way to handle this problem is to create a set of random initializations based on small *perturbations* to the source domain parameters $\Theta^{(s)}$. In the case of Gaussian mixtures, we will apply small, random perturbations to the component mean vectors and create a set of random initializations. This is further discussed in section 4.3. We note that another valid approach to control the extent to which the initialization (based on the source domain parameters) affects the classifier adaptation is to impose a Bayesian prior on the parameters, and add a log-prior term to the objective function (4.11). For example, we can use the Normal-Inverse-Wishart (NIW) distribution [136], given by $\text{NIW}(\underline{\mu}_{cl}, \mathbf{\Sigma}_{cl}\,;\, \underline{m}_{cl}^{(0)}, k_{cl}^{(0)}, \mathbf{S}_{cl}^{(0)}, \nu_{cl}^{(0)}) = \mathcal{N}(\underline{\mu}_{cl}\,;\, \underline{m}_{cl}^{(0)}, \frac{1}{k_{cl}^{(0)}}\mathbf{\Sigma}_{cl})\ \text{IW}(\mathbf{\Sigma}_{cl}\,;\, \mathbf{S}_{cl}^{(0)}, \nu_{cl}^{(0)})$, as a joint conjugate prior on the mean vector $\underline{\mu}_{cl}$ and covariance matrix $\mathbf{\Sigma}_{cl}$ of component l of class c. Based on the estimated source domain classifier parameters, we can set two of the hyperparameters as $\underline{m}_{cl}^{(0)} = \underline{\mu}_{cl}^{(s)}$ and $\mathbf{S}_{cl}^{(0)} = M_{cl}^{(s)}\,\mathbf{\Sigma}_{cl}^{(s)}$, where $M_{cl}^{(s)}$ is a probabilistic count of the number of labeled source domain samples belonging to component l of class c in the source domain classifier solution. The other two hyperparameters $k_{cl}^{(0)} \geq 1$ and $\nu_{cl}^{(0)} \geq d$ control the strength of the prior distribution (sometimes referred to as the degrees of freedom of the NIW distribution). When they are set to a small value, the effect of the prior distribution on the posterior distribution and the MAP estimates is weak, and as they are set to larger values the prior distribution has a stronger effect. We can set $\nu_{cl}^{(0)} + d + 1 = M_{cl}^{(s)},\ \forall c \in \mathcal{C}, \forall l \in \mathcal{M}_c$ [3], and in order to create random initializations, the hyperparameter controlling the mean vector $k_{cl}^{(0)} = k$, $\forall c \in \mathcal{C}, \forall l \in \mathcal{M}_c$ can be varied. In this work, we will focus only on the former approach for creating multiple classifier adaptation solutions, and leave evaluation of the Bayesian approach as future work.

4.2.3.1 Optimal nonparametric E-Step

Since the posterior distribution of the latent variables given the data, (4.14), already has a factorized form that is tractable to compute, it is not necessary to invoke the mean-field approximation, as we did for the method in chapter 3. From (4.14), the factors of this distribution for the labeled

[3] For the Inverse Wishart distribution $\nu_{cl}^{(0)} + d + 1$ is the effective number of samples corresponding to the prior scatter matrix hyperparameter $\mathbf{S}_{cl}^{(0)}$.

and unlabeled samples are respectively given by

$$q(c,l\,|\,\underline{x}_i^{(t)}) \;=\; \frac{\pi_c\,\alpha_{cl}\,\mathcal{N}(\underline{x}_i^{(t)}\,;\underline{\mu}_{cl},\boldsymbol{\Sigma}_{cl})\;e^{-\beta\,\delta[c\neq c_i^{(t)}]}}{\sum\limits_{k\in\mathcal{C}}\sum\limits_{m\in\mathcal{M}_k}\pi_k\,\alpha_{km}\,\mathcal{N}(\underline{x}_i^{(t)}\,;\underline{\mu}_{km},\boldsymbol{\Sigma}_{km})\;e^{-\beta\,\delta[k\neq c_i^{(t)}]}},\quad \begin{smallmatrix}\forall c\in\mathcal{C}\\ \forall l\in\mathcal{M}_c\\ \forall i\in\mathcal{I}_l^{(t)}\end{smallmatrix} \tag{4.15}$$

and

$$q(c,l\,|\,\underline{x}_i^{(t)}) \;=\; \frac{\pi_c\,\alpha_{cl}\,\mathcal{N}(\underline{x}_i^{(t)}\,|\,\underline{\mu}_{cl},\boldsymbol{\Sigma}_{cl})}{\sum\limits_{k\in\mathcal{C}}\sum\limits_{m\in\mathcal{M}_k}\pi_k\,\alpha_{km}\,\mathcal{N}(\underline{x}_i^{(t)};\underline{\mu}_{km},\boldsymbol{\Sigma}_{km})},\quad \begin{smallmatrix}\forall c\in\mathcal{C}\\ \forall l\in\mathcal{M}_c\\ \forall i\in\mathcal{I}^{(t)}\backslash\mathcal{I}_l^{(t)}\end{smallmatrix} \tag{4.16}$$

For the unlabeled samples, $q(c,l\,|\,\underline{x}_i^{(t)})$ is simply the standard Gaussian mixture posterior, while for the labeled sample $q(c,l\,|\,\underline{x}_i^{(t)})$ has an additional factor $e^{-\beta}$ multiplying terms in the numerator and denominator for which the class is not equal to the class label. We will show in the sequel that a solution based on this E-step is not desirable because, though it may achieve a small error rate on the given target domain labeled samples (for a sufficiently large β), it may do so without actually achieving any propagation of the label information to the neighboring unlabeled samples, by simply introducing discontinuities in $q(c,l\,|\,\underline{x}_i^{(t)})$.

4.2.3.2 Parametric E-Step

In this approach, we will constrain the factors corresponding to all the samples in the posterior distribution (4.3) to be a parametric function consistent with the type of component conditional distribution used in the mixture model for each class (in this case multivariate Gaussian). As discussed in section 4.2.2, this approach is also commonly known as the variational approximation. When the class conditional densities are modeled by mixtures of Gaussians, we constrain the factors to have the parametric form

$$q(c,l\,|\,\underline{x},\Phi) = \frac{\exp\left[\underline{x}^T\mathbf{W}_{cl}\underline{x}+\underline{x}^T\underline{w}_{cl}+b_{cl}\right]}{\sum\limits_{k\in\mathcal{C}}\sum\limits_{m\in\mathcal{M}_c}\exp\left[\underline{x}^T\mathbf{W}_{km}\underline{x}+\underline{x}^T\underline{w}_{km}+b_{km}\right]},\quad \begin{smallmatrix}\forall c\in\mathcal{C}\\ \forall l\in\mathcal{M}_c\\ \forall i\in\mathcal{I}^{(t)}\end{smallmatrix} \tag{4.17}$$

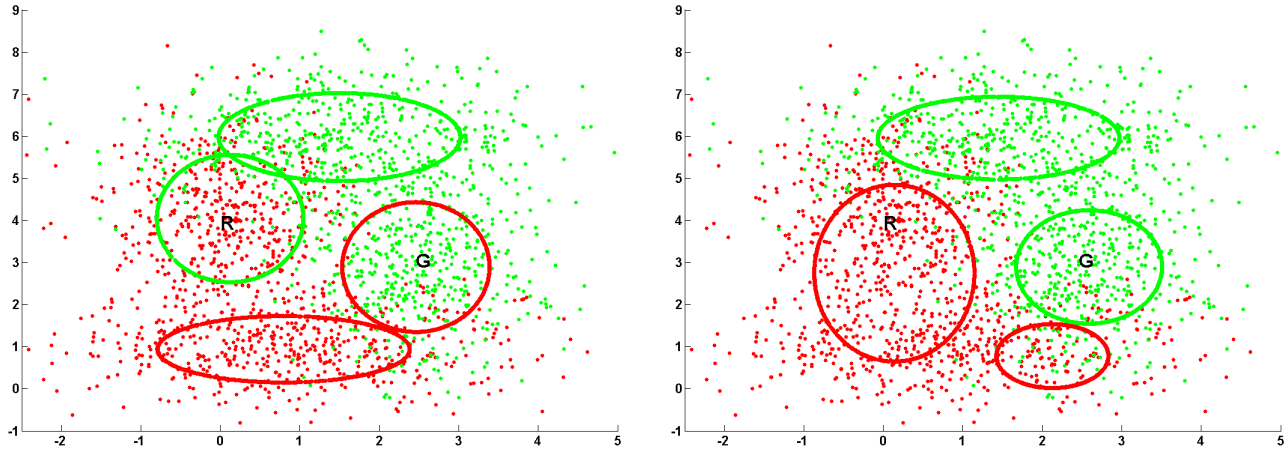
where the $d\times d$ symmetric matrices $\mathbf{W}_{cl}$, $\forall c\in\mathcal{C}$, $\forall l\in\mathcal{M}_c$, the d dimensional vectors $\underline{w}_{cl}$, $\forall c\in\mathcal{C}$, $\forall l\in\mathcal{M}_c$, and the real valued scalars b_{cl}, $\forall c\in\mathcal{C}$, $\forall l\in\mathcal{M}_c$ are called the variational parameters. Similar to the solution approach in chapter 3, we constrain *all* the factors (corresponding to both the labeled and unlabeled samples) to have the same parametric form in order to ensure that space-partitioning implications can be gleaned from a limited number of target domain labeled samples. The E-Step now involves minimizing $\mathcal{F}$ (given by (4.11)) over the variational parameters $\Phi=\{(\mathbf{W}_{cl},\underline{w}_{cl},b_{cl}),\ \forall c\in\mathcal{C},\forall l\in\mathcal{M}_c\}$. Since this optimization

does not have a closed form solution, we apply the gradient descent method on the vector of all parameters in Φ in order to find a local minimum of the objective function $\mathcal{F}$.

4.2.3.3 Weakness of the nonparametric E-Step

In the nonparametric posterior distribution (4.15), by making β sufficiently large, one can ensure that a labeled sample from the target domain $(\underline{x}_i^{(t)}, c_i^{(t)})$ will have high probability of association with the subset of components affiliated to class $c_i^{(t)}$, even if these components are not the *nearest* ones (spatially) to $\underline{x}_i^{(t)}$. An unlabeled sample from the target domain $\underline{x}_j^{(t)}$, highly proximal to $\underline{x}_i^{(t)}$, has *no* β-dependence in $q(c, l \mid \underline{x}_j^{(t)})$, and thus will end up with strong association to its nearest components, regardless of the class affiliation of the component, *i.e.*, $\underline{x}_i^{(t)}$ and $\underline{x}_j^{(t)}$ may have *very different* (class, component) associations. In other words, this approach takes the "easy way out", satisfying constraints (on the target domain labeled samples) sample-wise, without there being any *space-partitioning* implications [90]. On the other hand, in the parametric E-step, since $q(c, l \mid \underline{x}, \Phi)$ is a smooth function of $\underline{x}$, it must be the case that $q(c, l \mid \underline{x}_i^{(t)}, \Phi)$ and $q(c, l \mid \underline{x}_j^{(t)}, \Phi)$ are close in value when $\underline{x}_i^{(t)}$ and $\underline{x}_i^{(t)}$ are highly proximal in the feature space. In this case, the *only* way to satisfy the constraints on the labeled samples is by *moving* the components, ensuring that the neighboring unlabeled samples also have similar class-component associations as the labeled samples. Thus the solution will have the desired space-partitioning implications.

We illustrate this idea in Fig.4.1 using a synthetic data set with two features and two classes (distinguished by the colors red and green). Data in the source domain (not shown in the figure) was generated from a Gaussian mixture distribution with four components, whose means and covariance matrices were suitable chosen. Data points that were generated by two particular components were assigned a class label "red", while data points that were generated from the other two components were assigned a class label "green". The parameters of this four component mixture model (used to generate the source domain data) are directly used to specify an *optimal* MDA classifier for the source domain data. The target domain data was generated using a Gaussian mixture distribution whose parameters are the same as that of the source domain Gaussian mixture, but with the class affiliation of two of the components interchanged (swapped). The target domain data samples are shown both in Fig.4.1(a) and Fig.4.1(b). There are only two labeled samples, one from each class, at the locations specified by the symbols "R" (corresponding to the class red) and "G" (corresponding to the class green). The rest of samples shown with the small symbols are unlabeled, but whose unknown class labels are revealed by the symbol colors (red or green). It is obvious that the MDA classifier for the source domain will make many classification errors on the target domain data, unless its parameters are adapted such that two of its components are actually "moved" in the feature space (by adapting the component



(a) Target domain data and the solution based on the non-parametric E-Step

(b) Target domain data and the solution based on the parametric E-Step

Figure 4.1. A synthetic data example illustrating the domain adapted classifier solution found by using (i)the nonparametric E-step and (ii)the parametric E-step.

means).

Two MDA classifier solutions for the target domain were obtained by applying the EM based adaptation method presented in section 4.2, one based on the nonparametric E-step, and the other based on the parametric E-step. These adapted MDA classifiers are illustrated by the constant density contours of their Gaussian components, with the contour color specifying the class affiliation of the components (red or green). The solution in Fig.4.1(a) corresponds to the nonparametric E-step and the solution in Fig.4.1(b) corresponds to the parametric E-step. The non-parametric E-step based method simply drives the error based penalty close to zero without actually moving the components, which results in a poor classification solution as evident from Fig.4.1(a). On the other hand, the parametric E-step based method can only drive the error based penalty close to zero by actually moving the required components, which results in a good classification solution (that also imposes a smooth partition of the feature space into two classes) as evident from Fig.4.1(b). This example highlights the need for using the parametric E-step, despite its greater computational requirement compared to the nonparametric E-step.

4.3 Overall learning strategy

For a fixed value of β, the objective function $\mathcal{F}$ is minimized by alternating the E and M steps to convergence. At iteration n, suppose the model parameters are $\Theta^{(n)}$, then the parametric E-

step finds $\Phi^{(n)}$, a local minimum of $\mathcal{F}(\Theta^{(n)}, \Phi)$, using gradient descent. The M-step finds the model parameter updates $\Theta^{(n+1)}$ which minimize $\mathcal{F}(\Theta, \Phi^{(n)})$. These parameter updates can be computed in closed form according to following equations:

$$\pi_c^{(n+1)} = \frac{\sum_{\underline{x} \in \mathcal{X}^{(t)}} \sum_{l \in \mathcal{M}_c} q(c, l \,|\, \underline{x}, \Phi^{(n)})}{N_u + N_l}, \quad \forall c \in \mathcal{C},$$

$$\alpha_{cl}^{(n+1)} = \frac{\sum_{\underline{x} \in \mathcal{X}^{(t)}} q(c, l \,|\, \underline{x}, \Phi^{(n)})}{\sum_{\underline{x} \in \mathcal{X}^{(t)}} \sum_{l \in \mathcal{M}_c} q(c, l \,|\, \underline{x}, \Phi^{(n)})}, \quad \forall c \in \mathcal{C}, \ \forall l \in \mathcal{M}_c,$$

$$\underline{\mu}_{cl}^{(n+1)} = \frac{\sum_{\underline{x} \in \mathcal{X}^{(t)}} q(c, l \,|\, \underline{x}, \Phi^{(n)}) \, \underline{x}}{\sum_{\underline{x} \in \mathcal{X}^{(t)}} q(c, l \,|\, \underline{x}, \Phi^{(n)})}, \quad \forall c \in \mathcal{C}, \ \forall l \in \mathcal{M}_c,$$

$$\mathbf{\Sigma}_{cl}^{(n+1)} = \frac{\sum_{\underline{x} \in \mathcal{X}^{(t)}} q(c, l \,|\, \underline{x}, \Phi^{(n)}) \, (\underline{x} - \underline{\mu}_{cl}^{(n+1)})(\underline{x} - \underline{\mu}_{cl}^{(n+1)})^T}{\sum_{\underline{x} \in \mathcal{X}^{(t)}} q(c, l \,|\, \underline{x}, \Phi^{(n)})}, \quad \forall c \in \mathcal{C}, \ \forall l \in \mathcal{M}_c.$$

Suppose the EM algorithm is initialized with model parameters $\Theta^{(0)}$, then the initial variational parameters $\Phi^{(0)}$ are found by equating the quadratic, linear, and constant terms in the equation $\underline{x}^T \mathbf{W}_{cl}^{(0)} \, \underline{x} + \underline{x}^T \underline{w}_{cl}^{(0)} + b_{cl}^{(0)} = \ln[\, \pi_c^{(0)} \, \alpha_{cl}^{(0)} \, \mathcal{N}(\underline{x} \,|\, \underline{\mu}_{cl}^{(0)}, \mathbf{\Sigma}_{cl}^{(0)}) \,]$. This equation is obtained from a comparison of (4.16) and (4.17). Also, we denote the model parameters and the variational parameters upon convergence of the EM algorithm by $\Theta^{(t)}$ and $\Phi^{(t)}$.

The penalty coefficient β is a hyper-parameter which controls the possible trade-off in the solution between achieving a small error rate on the target domain labeled samples and finding a good mixture model fit to the marginal density of the feature vector in the target domain. Starting from a small value, β is increased in steps, and for each value of β the EM algorithm is used to find the adapted classifier parameters, starting from the same initialization $\Theta^{(0)}$. The value of the constraint penalty term (or soft error count on the labeled target domain samples), given by $\sum_{i \in \mathcal{I}_l^{(t)}} \sum_{\substack{c \in \mathcal{C} \\ c \neq c_i^{(t)}}} \sum_{l \in \mathcal{M}_c} q(c, l \,|\, \underline{x}_i^{(t)}, \Phi^{(t)})$, is observed upon convergence of the EM algorithm for successive increasing values of β. If the difference between successive values of the penalty term falls below a threshold, we stop increasing β.

From experiments, we have observed that sometimes initializing exactly at the source domain classifier parameter values can lead to solutions which do not have a sufficiently small error rate on the target domain labeled samples. This may be due to the adaptation finding local minima and/or due to significant difference between the underlying distributions which generate the source and target domain data. In order to mitigate this problem, we create a set of candidate initializations based on small perturbations to the source domain classifier parameters $\Theta^{(s)}$ (as discussed in section 4.2.3). In the case of a Gaussian mixture based MDA classifier, we create a set of random initializations by uniformly sampling the mean vector of each component $l \in \mathcal{M}_c$ of each class $c \in \mathcal{C}$ from the interval $(\underline{\mu}_{cl}^{(s)} - 2\,\mathrm{diag}(\mathbf{\Sigma}_{cl}^{(s)}),\, \underline{\mu}_{cl}^{(s)} + 2\,\mathrm{diag}(\mathbf{\Sigma}_{cl}^{(s)}))$, where $\mathrm{diag}(\cdot)$ refers to the column vector of diagonal elements of a matrix. All the other model parameters are set equal to their corresponding values in the source domain classifier.

4.3.1 Choosing a solution using a novel validation criterion

To select between different solutions (from random restarts, and different β values), we define a validation criterion called *transformed source domain accuracy*. In order to do so, we first learn the parameters of two Gaussian mixture models to approximate the marginal density of the feature vector ($\underline{X}$) separately on the source and target domain feature vector data (*i.e.*, excluding the class labels) by maximum likelihood estimation using the EM algorithm. Suppose these estimated models are given, for the source and target domain respectively, by

$$P_s(\underline{x}) \;=\; \sum_{i \in \mathcal{M}^{(s)}} \gamma_i^{(s)}\, \mathcal{N}(\underline{x}\,; \underline{m}_i^{(s)}, \mathbf{V}_i^{(s)})$$

and

$$P_t(\underline{x}) \;=\; \sum_{i \in \mathcal{M}^{(t)}} \gamma_i^{(t)}\, \mathcal{N}(\underline{x}\,; \underline{m}_i^{(t)}, \mathbf{V}_i^{(t)})$$

where $\mathcal{M}^{(s)}$ and $\mathcal{M}^{(t)}$ are the index sets of components in the source and target domain respectively, and the rest of the parameters are the usual component priors, mean vectors, and covariance matrices of a Gaussian mixture (superscript *s* and *t* are used to denote the source and target domains). The number of components in the mixture models are selected according to the BIC criterion [152] applied to the maximum likelihood solution.

Each source domain component is then mapped to a particular target domain component such that the overall cost of mapping given by

$$G(\{v_{ij}\}) = \sum_{i \in \mathcal{M}^{(s)}} \sum_{j \in \mathcal{M}^{(t)}} v_{ij}\, c_{ij} \tag{4.18}$$

is minimized, where $v_{ij} \in \{0, 1\}$ takes a value 1 (0) if component i is mapped (not mapped) to component j, and $c_{ij} > 0$ is the cost of mapping component i to component j. We chose $c_{ij} = \|\underline{m}_i^{(s)} - \underline{m}_j^{(t)}\|^2$, the Euclidean distance between the component Gaussian densities [4]. Minimization of (4.18) is the well known Linear Assignment problem, which can be solved using the *Hungarian algorithm* [26]. If the number of components in the source domain mixture ($|\mathcal{M}^{(s)}|$) is smaller than the number of components in the target domain mixture ($|\mathcal{M}^{(t)}|$), then some of the target domain components will be left unmapped. On the other hand, if $|\mathcal{M}^{(s)}| > |\mathcal{M}^{(t)}|$, then we find a mapping of only $|\mathcal{M}^{(t)}|$ components of the source domain mixture, and the rest of them are left unmapped. The unmapped source domain components $i \in \mathcal{M}^{(s)}$ in the solution will have $\sum_{j \in \mathcal{M}^{(t)}} v_{ij} = 0$.

Once we have the mapping from source to target domain components (*i.e.*, from $\mathcal{M}^{(s)}$ to $\mathcal{M}^{(t)}$), for Gaussian component densities, we can find the unique affine transformation $T_{ij}(\underline{x})$ that takes a feature vector $\underline{x}$ belonging to source component i to target component j. This is given by $T_{ij}(\underline{x}) = \mathbf{A}_j^{(t)} (\mathbf{A}_i^{(s)})^{-1} (\underline{x} - \underline{m}_i^{(s)}) + \underline{m}_j^{(t)}$, where $\mathbf{A}_i^{(s)}$ and $\mathbf{A}_j^{(t)}$ are the factors obtained by Cholesky decomposition of $\mathbf{V}_i^{(s)}$ and $\mathbf{V}_j^{(t)}$ respectively. Also, let $T_i(\underline{x}) = \sum_{j \in \mathcal{M}^{(t)}} v_{ij}\, T_{ij}(\underline{x})$, $\forall i \in \mathcal{M}^{(s)}$. Based on these component conditional transformation of points from the source domain to the target domain, we define a measure called the transformed source domain accuracy, which is the classification accuracy of the labeled source domain data, transformed according to the component conditional affine transformations, under the adapted target domain classifier. We define the transformed source domain accuracy as

$$\frac{1}{N_l^{(s)}} \sum_{(\underline{x},c) \in \mathcal{X}_l^{(s)}} \frac{\pi_c^{(t)}}{\pi_c^{(s)}} \sum_{i \in \mathcal{M}^{(s)}} \Big(\sum_{j \in \mathcal{M}^{(t)}} v_{ij} \Big) \frac{\gamma_i^{(s)} \mathcal{N}(\underline{x}; \underline{m}_i^{(s)}, \mathbf{V}_i^{(s)})}{\sum_{j \in \mathcal{M}^{(s)}} \gamma_j^{(s)} \mathcal{N}(\underline{x}; \underline{m}_j^{(s)}, \mathbf{V}_j^{(s)})} \, \delta\left(\hat{C}_t(T_i(\underline{x})) = c\right), \tag{4.19}$$

where $\hat{C}_t(\underline{y})$ is the MAP rule of the target domain classifier based on the parametric class posterior obtained upon convergence of the EM algorithm for adaptation. Specifically,

$$\hat{C}_t(\underline{y}) = \arg\max_{c \in \mathcal{C}} \sum_{l \in \mathcal{M}_c} q(c, l \,|\, \underline{y}, \Phi^{(t)}) \,. \tag{4.20}$$

Note that, in (4.19) when $|\mathcal{M}^{(s)}| > |\mathcal{M}^{(t)}|$, only the components for which $\sum_{j \in \mathcal{M}^{(t)}} v_{ij} = 1$ will contribute to the transformed source domain accuracy. Also, the ratio $\frac{\pi_c^{(t)}}{\pi_c^{(s)}}$ is an importance

[4] We evaluated other types of costs c_{ij} such as the symmetrized Kullback-Leibler distance, Hellinger distance, and Bhattacharya distance [30] between the Gaussian densities of components i and j on some synthetically generated data sets. Surprisingly, from these experiments we found the simple Euclidean distance between the component means to be more suitable for mapping two sets of Gaussian components.

sampling factor to account for the difference in class priors between the source and target domains. In this way, the classification performance of an adapted classifier for the target domain can be evaluated using the available labeled data from source domain. However, this can be a reliable measure *only* if the component conditional affine transformations (found based on the mapping of components) reasonably capture the actual (unknown) underlying transformation that relates the source and target domain feature vector distributions. Otherwise the transformed source domain accuracy values can be misleading.

We have to choose a solution from the set of candidate solutions obtained from different random initialization of parameters and from different values of β. We will use the transformed source domain accuracy (4.19) and the error rate on the labeled target domain samples (4.10) jointly as criteria to eliminate potentially poor solutions (for target domain classification) as follows. Based on a histogram of the transformed source domain accuracies (from all the candidate solutions), we choose a value $\eta \in (0, 1)$ close to the maximum where the density of solutions is relatively high. Similarly, based on a histogram of the error rate on the labeled target domain samples, we choose a value $\epsilon \in (0, 1)$ close to the minimum where the density of solutions is relatively high. We choose among all the candidate solutions, the ones which have transformed source domain accuracy greater than η and error rate on the target domain labeled samples smaller than ϵ. From the solutions satisfying the above criteria, we choose the one with the smallest value of the objective function $\mathcal{F}(\Theta^{(t)}, \Phi^{(t)})$ (at convergence) as the final classifier solution for the target domain. We acknowledge that this is not a fully objective way of choosing a unique solution. However, an alternative for choosing a solution from the candidates such as cross-validation on the labeled target domain samples would be computationally intensive. Moreover, cross-validation may not be reliable if there are very few labeled samples in the target domain. In our approach, we address this problem by leveraging the available labeled data from the source domain.

4.4 Related work

There is prior work on domain adaptation for discriminative and generative classifier models, both in the unsupervised and the semi-supervised setting. Our work falls under "generative and semi-supervised" domain adaptation. Early work, in a speech recognition context, is [50] and [49]. In [50], parameters of Gaussian mixture models used to model the state conditional observation densities of a Hidden Markov model are adapted in a constrained fashion to maximize the likelihood of a limited amount of adaptation data from a new speaker. This method is extended in [49] to handle large amounts of adaptation data in a better way using Bayesian techniques. In

[25], an unsupervised domain adaptation method for generative classifiers was proposed. Specifically, labeled data in the source domain is used to learn a mixture-based generative classifier, which serves as the *initialization* for a model trained in a purely unsupervised fashion via EM to maximize the log-likelihood of the target domain data. To overcome these difficulties, we propose a semi-supervised extension in section 4.5.1, which avoids poor local maxima problems through the use of random restarts, coupled with a novel model validation strategy. We first review some work on discriminative classifier models. In [82], an instance weighting framework for semi-supervised domain adaptation is developed, where several strategies are proposed to remove misleading labeled samples from the source domain, to assign more weight to labeled target domain samples than labeled source domain samples, and to augment the training set with predicted target domain samples. In [24], an unsupervised domain adaptation method for support vector machines called DASVM is proposed. Also, a circular validation strategy for identifying reliable solutions in the target domain, using only the labeled source domain data, is proposed. In [16] and [11], Structural Correspondence Learning (SCL) for domain adaptation is proposed. The key idea is to identify correspondences among features from the different domains by modeling their correlations with what are called *pivot* features, which behave in the same way for discriminative learning in the two domains. A detailed literature survey of domain adaptation methods can be found in [81].

4.5 Experimental Results

In this section, we evaluate the classification performance of the proposed semi-supervised domain adaptation method.

4.5.1 Methods used for performance comparison

For performance comparison, we use three baseline methods. One is a direct porting of the source domain classifier (without adapting its parameters) to make predictions on the target domain data. The second is the Mixture of Experts (MOE) semi-supervised learning method [122] based solely on the target domain data. As a third method, we compare with the following semi-supervised extension of the unsupervised domain adaptation method of [25]. In order to condition on the available labeled data in the target domain, we modify their log-likelihood objective, which is based only on unlabeled data, to include the log-likelihood of the labeled data.

The modified objective function is given by

$$\begin{aligned}\mathcal{P}(\mathcal{X}_u^{(t)}, \mathcal{X}_l^{(t)} \,|\, \Theta) &= \sum_{(\underline{x},c)\in\mathcal{X}_l^{(t)}} \log \sum_{l\in\mathcal{M}_c} [\pi_c \, \alpha_{cl} \, \mathcal{N}(\underline{x}\,; \underline{\mu}_{cl}, \mathbf{\Sigma}_{cl})] \\ &+ \sum_{\underline{x}\in\mathcal{X}_u^{(t)}} \log \sum_{c\in\mathcal{C}} \sum_{l\in\mathcal{M}_c} [\pi_c \, \alpha_{cl} \, \mathcal{N}(\underline{x}\,; \underline{\mu}_{cl}, \mathbf{\Sigma}_{cl})].\end{aligned}$$

There is a closed form EM algorithm for maximizing this objective, and the algorithm is repeated from different random initializations of the parameters as we do for our method. Also, as described in section 4.3.1, the transformed source domain accuracy and the error rate on the labeled target domain samples are used to find a reduced set of candidate solutions, from which the one having the largest data log-likelihood is selected. In order to have an upper bound on the classification performance (of an MDA classifier) on the target domain, we learn an MDA classifier with Gaussian mixtures in a supervised setting, *i.e.*, with class labels provided for all the target domain samples (including the unlabeled samples). Specifically, we calculate and report the ten-fold average cross validation accuracy of the classifier using all the target domain data with class labels.

4.5.2 Results on Internet traffic classification datasets

We evaluated the performance of the methods on the problem of Internet traffic (packet-flow) classification [133], [126]. We used four publicly available network packet traces recorded at the University of Cambridge, UK [27]. These data sets were collected over a period of four years at two different sites called *Site A* and *Site B*. From Site A, three data sets called *Day1*, *Day2*, and *Day3*, and from Site B one data set (called *Site-B*) are made available. Detailed information about the data sets, the features, and application classes can be found in [104]. Also, in [104] and [187] it was demonstrated that there is a drop in classification performance when a classifier trained on one of these data sets is directly ported to make predictions on a different one. In our experiments, we selected the following three (out of the six) continuous valued features which have good discriminative capability: *min_seg_size_clnt*, *avg_seg_size_serv*, and *IP_bytes_med_clnt*, whose descriptions are given in Table 3 of [104]. We focused on four application classes *WWW*, *MAIL*, *BULK*, and *P2P*, which have a sufficient number of samples in all the data sets. Since the data sets are very large (around 200000 to 400000 samples), for computational reasons, we randomly sampled 10% from whole data set while preserving the class proportions present in the original data set.

In Table 4.1, the average 10 fold cross validation performance of an MDA classifier (representing an *upper* bound on performance of domain adaptation) on the four data sets is given.

Table 4.1. Ten fold cross-validation performance (in percentage) of an MDA classifier. **Class**: A - WWW, B - MAIL, C - BULK, D - P2P

Data	Accuracy	Recall				Precision			
		A	B	C	D	A	B	C	D
Day1	94.28	95.27	90.44	91.53	31.59	99.16	96.53	46.33	42.08
Day2	93.75	97.18	77.87	78.24	73.89	98.69	87.69	64.71	60.47
Day3	97.45	98.51	70.52	72.97	97.71	99.27	53.35	71.08	99.01
Site-B	96.23	99.09	81.15	64.67	72.40	97.89	87.22	59.13	82.12

Since these data sets have a skewed class prior (large proportion of the samples are from the WWW class), we also report the precision and recall measures of the classifier for each class. We performed three domain adaptation experiments: (i) Day3 as the source and Site-B as the target, (ii) Day1 as the source and Day3 as the target, and (iii) Day2 as the source and Site-B as the target. In all cases, the target domain labeled subset was created by random sampling, and the size was varied from 10, in steps of 5, up to 25. The results for all the methods were averaged over three different labeled subsets (size varied from 10 to 25 in each case), with the same labeled and unlabeled subsets used for all the methods. We used 10 random restarts for our method, and 100 random restarts for the semi-supervised extension of [25] (because this method is less computationally intensive).

The results for the domain adaptation experiments from Day3 to Site-B, Day1 to Day3, and Day2 to Site-B are given in Table 4.2, Table 4.3, and Table 4.4 respectively. The tables compare the average classification performance of the methods (in terms of overall accuracy, and precision and recall for each class) for different number of target domain labeled samples. We observe that the proposed method (I) outperforms (or is at least as good as) the other methods in all cases, and from comparison with Table 4.1, that the accuracy (in some cases), also the precision and recall of our method with only a few labeled samples approaches that of supervised learning on the target domain. We also evaluated the effectiveness of the transformed source domain accuracy in retaining solutions which have good classification performance on the target domain. We found it to be quite effective on these data sets, perhaps suggesting that the distribution of the feature vector is not very different on these data domains. For instance, for the Day1 to Day3 experiment with 20 labeled target-domain samples, the average target domain accuracy of the solutions eliminated based on the transformed source domain accuracy is 94.90%, while that of the retained solutions is 97.25%.

Table 4.2. Domain adaptation from Day3 to Site-B. Average classification performance (in percentage) on the target domain for different labeled subset sizes. **Method**: I - Proposed method, II - Extension of [25], III - Semisupervised learning [122], IV - Direct porting of the source domain classifier. Entry * denotes an undefined value.

$N_l^{(t)}$	Method	Accuracy	Recall				Precision			
			WWW	MAIL	BULK	P2P	WWW	MAIL	BULK	P2P
10	I	94.46	98.97	80.30	75.76	50.06	97.01	78.62	25.99	87.63
	II	89.81	98.07	74.25	67.88	1.61	95.65	42.41	16.38	11.48
	III	84.68	90.94	60.07	60.61	25.84	97.66	*	4.31	*
15	I	94.51	98.88	70.27	75.15	57.92	96.91	72.27	24.52	89.08
	II	89.87	98.12	74.66	70.30	1.59	95.77	51.93	9.70	15.03
	III	87.45	93.41	32.04	43.03	51.53	97.53	*	16.84	*
20	I	95.88	99.13	85.20	74.54	64.25	97.49	81.42	31.73	89.44
	II	90.05	98.15	78.04	67.88	1.61	95.80	53.01	12.51	11.51
	III	90.15	96.68	30.72	29.09	50.45	97.10	*	3.17	*
25	I	95.08	99.02	80.28	75.15	57.80	96.87	80.76	29.50	89.45
	II	90.73	98.70	78.00	73.33	4.06	96.08	56.67	12.47	33.52
	III	89.44	95.82	31.73	29.09	50.46	97.16	45.94	3.17	*
N/A	IV	91.62	99.50	88.42	76.79	0.28	92.48	84.46	37.07	8.20

Table 4.3. Domain adaptation from Day1 to Day3. Average classification performance (in percentage) on the target domain for different labeled subset sizes. **Method**: I - Proposed method, II - Extension of [25], III - Semisupervised learning [122], IV - Direct porting of the source domain classifier. Entry * denotes an undefined value.

$N_l^{(t)}$	Method	Accuracy	Recall				Precision			
			WWW	MAIL	BULK	P2P	WWW	MAIL	BULK	P2P
10	I	96.88	98.02	54.96	77.87	97.80	99.44	61.57	46.98	97.66
	II	88.12	97.98	52.69	78.68	0	99.43	65.72	13.31	*
	III	86.41	92.88	16.50	59.36	41.96	98.48	*	31.98	*
15	I	97.30	98.49	55.56	77.35	97.83	99.45	62.15	53.56	97.25
	II	90.54	97.42	52.69	78.85	32.58	99.44	65.69	19.35	*
	III	83.83	91.31	16.71	42.60	32.24	98.37	*	29.56	*
20	I	97.28	98.47	55.01	77.53	97.81	99.44	61.64	53.81	97.09
	II	90.54	97.41	52.65	78.84	32.58	99.44	65.65	19.34	*
	III	83.88	91.28	16.71	47.82	31.95	98.29	*	20.47	*
25	I	97.29	98.50	55.01	77.21	97.81	99.43	61.58	54.17	97.09
	II	95.35	96.29	52.65	78.65	97.72	99.43	65.65	31.31	99.51
	III	92.36	98.20	0	42.63	63.45	98.24	*	29.56	56.09
N/A	IV	88.23	98.06	55.02	78.17	0.18	91.43	59.84	37.78	3.92

Table 4.4. Domain adaptation from Day2 to Site-B. Average classification performance (in percentage) on the target domain for different labeled subset sizes. **Method**: I - Proposed method, II - Extension of [25], III - Semisupervised learning [122], IV - Direct porting of the source domain classifier. Entry * denotes an undefined value.

$N_l^{(t)}$	Method	Accuracy	Recall				Precision			
			WWW	MAIL	BULK	P2P	WWW	MAIL	BULK	P2P
10	I	93.77	98.29	87.66	72.73	44.31	95.73	85.30	14.43	85.33
	II	91.92	96.53	85.03	49.09	42.52	95.84	79.19	09.76	57.46
	III	84.68	90.94	60.07	60.61	25.84	97.66	*	04.30	*
15	I	93.82	98.14	87.54	72.73	46.98	95.95	85.36	13.96	84.47
	II	92.58	96.92	83.70	49.09	47.80	96.18	78.97	11.81	61.78
	III	87.45	93.41	32.04	43.03	51.53	97.52	*	16.84	*
20	I	93.91	98.19	85.90	72.12	48.54	96.09	85.10	11.97	84.81
	II	92.77	96.91	86.76	71.51	47.75	96.26	79.27	15.14	64.78
	III	90.15	96.67	30.72	29.09	50.45	97.10	*	03.17	*
25	I	93.80	98.02	85.74	73.33	49.06	96.08	85.75	13.03	82.11
	II	92.63	97.10	82.30	55.76	46.82	96.45	73.27	10.15	65.35
	III	89.44	95.82	31.73	29.09	50.46	97.16	45.94	03.17	*
N/A	IV	89.03	96.60	83.55	28.57	04.25	92.58	83.09	02.03	60.32

Chapter 5

Conclusions and future directions

5.1 Summary of contributions

In chapter 2, we developed novel semi-supervised mixture model based classification methods that introduce fine-grained within-component class modeling, within statistically sound generative modeling frameworks. One of our methods (NFGL) uses non-parametric within-component class modeling, which is seen to entail an underlying Markov random field on the class labels of labeled samples within each mixture component. Our second method (PFGL) uses a parametric within-component class posterior model that is motivated by randomized nearest prototype classification, and which also addresses a limitation of NFGL that manifests at very low labeled fractions. While NFGL automatically specifies the right level of intra-component class modeling complexity based on the labeled data, PFGL requires optimization and careful model order selection in order to set a suitable level of intra-component class modeling complexity. Both the methods were demonstrated to outperform previous semi-supervised mixture model based classifiers, and also supervised classifiers such as K-nearest neighbors, within-component nearest neighbor, and at low labeled fractions linear and nonlinear kernel based SVMs on a number of data sets from the UC Irvine machine learning repository.

In chapter 3, we developed a semi-supervised learning method based on pairwise sample constraints which can effectively achieve constraint propagation and generalization to unconstrained samples from a small number of constraints. Our approach imposes a smooth space-partitioning on the solution via a parametric mean-field approximation of the posterior distribution over component assignments in the E-step of the EM algorithm (which minimizes a variational free energy objective). This precludes both the possibility and tendency (exhibited by many existing methods) to satisfy constraints trivially by introducing class membership discon-

tinuities. Another feature of the method is that it allows classes to be flexibly modeled with multiple mixture components, thus capturing both the class and sub-class (cluster) structure in the data. This also allows the method to estimate the number of latent classes present in the data – a novel feature compared to most methods which require the number of classes to be specified. Experiments on synthetic datasets, UC Irvine datasets, and a real texture based image segmentation dataset demonstrate that our method can utilize pairwise sample constraints to learn (overall) significantly better classification solutions compared to existing methods.

In chapter 4, we developed a semi-supervised domain adaptation method for mixture model-based classifiers, which formulates the problem as a data log-likelihood maximization subject to a constraint on the (probabilistic) error count measured on the labeled target domain samples. In the EM algorithm based solution to the optimization problem, instead of performing a standard E-step by computing the true posterior distribution over the latent variables, the method uses a modified E-step by constraining the factors of the posterior distribution over the latent variables to be smooth parametric functions. Similar to the observations made in chapter 3, this allows the method to ensure that, in the adapted classifier solution, space-partitioning implications are gleaned from a limited number of target domain labeled samples. We also proposed a validation criterion called the transformed source domain accuracy, which is used to eliminate potentially poor classifier solutions (for the target domain) from a number of candidate adapted classifier solutions. Experiments on Internet packet-flow traffic data from different temporal and spatial domains demonstrate the validity and usefulness of the domain adaptation method.

5.2 Directions for future research

We next identify some directions for extending the scope and applicability of the methods developed, some of which have already been highlighted in the chapters. Some of the ideas which are common to all the methods are discussed upfront, and then the ideas specific to each method are discussed.

High dimensional feature spaces

Since the methods presented in chapters 3, 4, and 5 are all based on generative, joint modeling of the feature vector and the class label using finite mixture models, there is a need to address issues of modeling data with high feature dimensionality. These include model bias and insufficient number of data samples for reliable estimation of the model parameters and probabilities, well known as the curse of dimensionality [51] in pattern recognition literature. Since generative methods model the joint distribution of the features and the class $P(\underline{X}, C)$, the curse of dimen-

sionality problem directly affects them. Note that the problem of high feature dimensionality can also affect discriminative learning methods because pairwise distances (based on metrics in the Euclidean space) between points tend to become more similar and hence less discriminative for the purpose of classification.

A number of methods have been proposed in the literature for addressing the issue of density modeling and clustering in high dimensional feature spaces [59], [66], [101], [115], [116], [148], [151]. A method for regularizing the component covariance matrix estimates of a Gaussian mixture based on Ledoit-Wolf shrinkage estimation [101] was given in section 2.5. While this method ensures that the covariance matrix estimates are well-conditioned, it does not reduce the number of parameters to estimate (which is still $\mathrm{O}\left(d^2\right)$ for a d dimensional vector). A widely used method for addressing this issue is the mixture of factor analyzers (MFA) [116], [66], which allows flexibility in the number of free parameters of the component covariance matrices. By modeling the generation of the feature vector $\underline{X} \in \mathbb{R}^d$ in terms of a latent mixture component $M \in \{1, \ldots, K\}$ and a latent factor $\underline{Z} \in \mathbb{R}^q$, where $q \ll d$, MFA models the density of $\underline{X}$ as the mixture

$$P(\underline{x}) \;=\; \sum_{j=1}^{K} \alpha_j \, \mathcal{N}(\underline{x}\,; \underline{\mu}_j, \mathbf{B}_j \, \mathbf{B}_j^T + \mathbf{D}_j),$$

where $\mathbf{B}_j \in \mathbb{R}^{d \times q}$ is called the factor loading matrix, and $\mathbf{D}_j$ is a $d \times d$ diagonal matrix. The factor loading matrix of each component has only qd free parameters compared to $d\,(d+1)/2$ in the case of unrestricted covariance matrices in a standard Gaussian mixture. Note that a related dimensionality reduction method, mixture of probabilistic principal component analyzers [162] is actually a specialization (restriction) of the MFA model, with $\mathbf{D}_j = d_j \, \mathbf{I}, \; \forall j$.

The methods presented in chapters 3, 4, and 5 are all amenable to modeling using MFA instead of a standard Gaussian mixture, which would make them more suitable for modeling high dimensional data. For the methods in chapters 3 and 4, the choice of parametrization for the factors of the approximating distribution in the E-step also changes according to the density model of MFA. This reduces the number of variational parameters, making the E-step optimization more tractable for high dimensional data.

Mixed continuous and categorical valued features

In certain practical applications like modeling text documents in natural language processing, all the features are categorical (or discrete) valued. For such data sets, generative modeling approaches usually employ models like a mixture of Multinomial distributions (also known as the "bag of words" model) [110], with conditional independence assumed between the features given the component of origin. The methods in chapter 3 and chapter 4 can certainly be applied with

such models, by choosing an appropriate parametric function (based on *e.g.*, the Multinomial distribution) for the factors of the approximating distribution in the E-step. For the fine-grained classification models in chapter 2, the distance metric used in the within component RNN and RNP class posterior (given by (2.5) and (2.17)) needs to be modified from the Euclidean distance to a distance metric more appropriate for a categorical valued feature vector, such as the Hamming distance, Jaccard index, or simple matching distance.

It is also common for data sets collected from clinical analysis and social sciences to have a mixture of continuous and categorical valued features. A number of works in the past have addressed the modeling of mixed feature data such as [92], [93], [75], [132], [131], [76]. While the most straightforward mixture modeling approaches assume that the categorical features are conditionally independent of each other and conditionally independent of the continuous valued features given the component of origin, methods like the *location modeling* approach allow for some dependence between these two sets of features [92], [131], [113]. Recently, an extension of the mixture of factor analysis method for mixed continuous and categorical valued features in high dimensions has been developed based on the variational approximation method [88], [89]. Extending or generalizing the methods developed in this dissertation to handle mixed continuous and categorical valued features based on some of the aforementioned works is a direction for future research.

Future directions for the domain adaptation work

We can identify several extensions for the semi-supervised domain adaptation method developed in chapter 4. One of them is handling high dimensional feature spaces and mixed continuous and categorical valued features, which we have already discussed.

Before applying the domain adaptation method, it is useful to test whether there is any significant difference (deviation) between the data distributions in the source and target domains. Since the distribution $P(\underline{X}, C)$ is unknown in the target domain, this test will have to be based either on comparing the joint distribution of all features $P(\underline{X})$, or the marginal distribution of the individual features $P(X_j),\ \forall j \in [d]$ in the source and target domains. The distance between the distributions in the two domains can be quantified using statistical measures such as the Kullback-Leibler distance, Total variation distance, or Hellinger distance [30]. Also, the two sample Kolmogorov-Smirnov (K-S) test [160] can be used to test whether there is a statistically significant difference between two empirical cumulative distribution functions in one dimension (such as the marginal distribution of individual features). In practical domain adaptation settings, we may find that only a subset of all features have statistically significant difference based on the K-S test. In such a scenario, it may be a good idea to *only* adapt those parameters that are

specific to these features, and "freeze" the remaining parameters to their values in the source domain classifier. Note that this test can only detect the presence of covariate-shift, but as we discussed in chapter 1, there could still be class prior imbalance (mismatch) between the source and target domain data. This is another possible direction for future work.

In section 4.3.1 of chapter 4, we proposed a measure called the transformed source domain accuracy in order to filter out candidate domain adapted solutions which may not have good classification performance on the target domain. This measure is reliable only if the joint distribution of the features, $P(\underline{X})$, in the target domain can be well-approximated by applying controlled, locally affine transformations to the source domain feature vector. Secondly, there is a possibility of mismatch between (i) the number of components in the mixtures modeling $P(\underline{X})$ in the source and target domains and (ii) the mapping between the components found using the Hungarian algorithm. If this is the case, then the set of affine transformations may not capture the actual underlying transformation that relates the joint distribution of features in the two domains. Future work can try to address these challenges in order to come up with a more reliable method for validating the domain adapted classifier solutions.

The method in chapter 4 assumes that the same set of classes are present in the source and target domain data. However, in real world problems, it is possible that some classes present in the source domain data may not be present in the target domain data and vice-verse. Some existing works [120], [179] have addressed the problem of new class discovery for semi-supervised learning. Addressing new class discovery and handling missing classes in the domain adaptation problem is a valid direction for future research.

For the method in chapter 4, the parameters of the source domain classifier were utilized to create a set of random parameter initializations for the EM algorithm based classifier adaptation. We also discussed an alternative Bayesian approach where the parameters of the source domain classifier are utilized to define a prior distribution on the parameters of the target domain classifier, and certain hyper-parameters of the prior distribution are varied in order to create multiple adapted classifier solutions for the target domain. The effectiveness of this approach can be compared to the non-bayesian approach that we evaluated in chapter 4. In both cases, the method finds multiple candidate domain adapted solutions (for different constant β values and different random parameter initializations). Instead of selecting one solution from the candidates based on the method described in section 4.3.1, it may be useful to create an ensemble classifier solution for the target domain by selectively combining the candidate adapted classifiers based on an aggregation method such as voting or boosting [176], [94], [181], [169].

Bibliography

[1] A. Asuncion and D. Newman. UCI machine learning repository, University of California, Irvine, School of Information and Computer Science. `http://www.ics.uci.edu/~mlearn/`, 2007.

[2] M.-F. Balcan and A. Blum. A pac-style model for learning from labeled and unlabeled data. In *Learning Theory*, pages 111–126. Springer, 2005.

[3] M.-F. Balcan and A. Blum. A discriminative model for semi-supervised learning. *Journal of the ACM (JACM)*, 57(3):19, 2010.

[4] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning a Mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, 6(1):937, 2006.

[5] A. Barron, J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling. *Information Theory, IEEE Transactions on*, 44(6):2743–2760, 1998.

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

[7] S. Basu, M. Bilenko, and R. J. Mooney. Comparing and unifying search-based and similarity-based approaches to semi-supervised clustering. In *Proceedings of the ICML-2003 Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, pages 42–49, 2003.

[8] M. Belkin and P. Adviser-Niyogi. Problems of learning on manifolds. 2003.

[9] M. Belkin and P. Niyogi. Semi-supervised learning on riemannian manifolds. *Machine learning*, 56(1-3):209–239, 2004.

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

[11] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19:137, 2007.

[12] J. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 192–236, 1974.

[13] J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society*, 48(3):259–302, 1986.

[14] C. Biernacki, G. Celeux, and G. Govaert. Assessing a mixture model for clustering with the integrated completed likelihood. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(7):719–725, 2000.

[15] M. Bilenko, S. Basu, and R. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the Twenty-First International Conference on Machine Learning*, page 11. ACM, 2004.

[16] J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128. Association for Computational Linguistics, 2006.

[17] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the 18th international conference on Machine learning*, 2001.

[18] A. Blum, J. Lafferty, M. R. Rwebangira, and R. Reddy. Semi-supervised learning using randomized mincuts. In *Proceedings of the twenty-first international conference on Machine learning*, page 13. ACM, 2004.

[19] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

[20] U. Brefeld, C. Büscher, and T. Scheffer. Multi-view discriminative sequential learning. In *Machine Learning: ECML 2005*, pages 60–71. Springer, 2005.

[21] J. S. Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing*, pages 227–236. Springer, 1990.

[22] L. D. Brown. Fundamentals of statistical exponential families with applications in statistical decision theory. *Lecture Notes-monograph series*, pages i–279, 1986.

[23] L. Bruzzone and D. Fernandez Prieto. A partially unsupervised cascade classifier for the analysis of multitemporal remote-sensing images. *Pattern recognition letters*, 23(9):1063–1071, 2002.

[24] L. Bruzzone and M. Marconcini. Domain adaptation problems: A DASVM classification technique and a circular validation strategy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(5):770–787, 2010.

[25] L. Bruzzone and D. Prieto. Unsupervised retraining of a maximum likelihood classifier for the analysis of multitemporal remote sensing images. *Geoscience and Remote Sensing, IEEE Transactions on*, 39(2):456–460, 2001.

[26] R. Burkard, M. Dell'Amico, and S. Martello. *Assignment problems*. Society for Industrial Mathematics, 2009.

[27] Cambridge Brasil Group. Characterizing network-based applications. `http://www.cl.cam.ac.uk/research/srg/netos/brasil/data/index.html`.

[28] V. Castelli and T. Cover. The relative value of labeled and unlabeled samples in pattern recognition. In *Information Theory, 1993. Proceedings. 1993 IEEE International Symposium on*, pages 355–355. IEEE, 1993.

[29] V. Castelli and T. M. Cover. The relative value of labeled and unlabeled samples in pattern recognition with an unknown mixing parameter. *Information Theory, IEEE Transactions on*, 42(6):2102–2117, 1996.

[30] S.-H. Cha. Comprehensive survey on distance/similarity measures between probability density functions. *City*, 1(2):1, 2007.

[31] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[32] O. Chapelle, M. Chi, and A. Zien. A continuation method for semi-supervised svms. In *Proceedings of the 23rd international conference on Machine learning*, pages 185–192. ACM, 2006.

[33] O. Chapelle, B. Schölkopf, A. Zien, et al. *Semi-supervised learning*, volume 2. MIT press Cambridge, MA:, 2006.

[34] O. Chapelle, V. Sindhwani, and S. Keerthi. Branch and bound for semi-supervised support vector machines. 2007.

[35] O. Chapelle and A. Zien. Semi-supervised classification by low density separation. 2004.

[36] N. V. Chawla and G. I. Karakoulas. Learning from labeled and unlabeled data: An empirical study across techniques and domains. *J. Artif. Intell. Res.(JAIR)*, 23:331–366, 2005.

[37] C. Chelba and A. Acero. Adaptation of maximum entropy classifier: Little data can help a lot. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Barcelona, Spain*, 2004.

[38] 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. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(12):1553–1566, 2004.

[39] D. Cohn, R. Caruana, and A. McCallum. Semi-supervised clustering with user feedback. *Constrained Clustering: Advances in Algorithms, Theory, and Applications*, 4(1):17, 2003.

[40] R. Collobert, F. Sinz, J. Weston, and L. Bottou. Trading convexity for scalability. In *Proceedings of the 23rd international conference on Machine learning*, pages 201–208. ACM, 2006.

[41] A. Corduneanu and T. Jaakkola. On information regularization. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 151–158. Morgan Kaufmann Publishers Inc., 2002.

[42] T. Cover and J. Thomas. *Elements of Information Theory*, pages 19–20. J. Wiley & Sons, 2006.

[43] F. G. Cozman, I. Cohen, and M. Cirelo. Unlabeled data can degrade classification performance of generative classifiers. In *FLAIRS Conference*, pages 327–331, 2002.

[44] F. G. Cozman, I. Cohen, M. C. Cirelo, et al. Semi-supervised learning of mixture models. In *ICML*, pages 99–106, 2003.

[45] H. Daumé III and D. Marcu. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26(1):101–126, 2006.

[46] I. Davidson and S. Basu. A survey of clustering with instance level constraints. *ACM Transactions on Knowledge Discovery from Data*, pages 1–41, 2007.

[47] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216. ACM, 2007.

[48] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[49] V. Digalakis and L. Neumeyer. Speaker adaptation using combined transformation and bayesian methods. *Speech and Audio Processing, IEEE Transactions on*, 4(4):294–300, 1996.

[50] V. Digalakis, D. Rtischev, and L. Neumeyer. Speaker adaptation using constrained estimation of gaussian mixtures. *Speech and Audio Processing, IEEE Transactions on*, 3(5):357–366, 1995.

[51] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification*. John Wiley & Sons, 2012.

[52] J. A. Fessler and A. O. Hero. Space-alternating generalized expectation-maximization algorithm. *Signal Processing, IEEE Transactions on*, 42(10):2664–2677, 1994.

[53] C. Fraley and A. Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97(458):611–631, 2002.

[54] S. Fralick. Learning to recognize patterns without a teacher. *Information Theory, IEEE Transactions on*, 13(1):57–64, 1967.

[55] S. J. Frame and S. R. Jammalamadaka. Generalized mixture models, semi-supervised learning, and unknown class inference. *Advances in Data Analysis and Classification*, 1(1):23–38, 2007.

[56] A. Fujino, N. Ueda, and K. Saito. A hybrid generative/discriminative approach to semi-supervised classifier design. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 764. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

[57] D. G. Gavin and F. S. Hu. Bioclimatic modelling using gaussian mixture distributions and multiscale segmentation. *Global Ecology and Biogeography*, 14(5):491–501, 2005.

[58] Z. Ghahramani, G. E. Hinton, et al. The em algorithm for mixtures of factor analyzers. Technical report, Technical Report CRG-TR-96-1, University of Toronto, 1996.

[59] M. Graham and D. Miller. Unsupervised learning of parsimonious mixtures on large spaces with integrated feature and component selection. *Signal Processing, IEEE Transactions on*, 54(4):1289–1303, 2006.

[60] Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems*, volume 17, pages 529–536. MIT Press, 2005.

[61] J. Gui, D.-S. Huang, and Z. You. An improvement on learning with local and global consistency. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE, 2008.

[62] Z. Halbe and M. Aladjem. Regularized mixture discriminant analysis. *Pattern Recognition Letters*, 28(15):2104–2115, 2007.

[63] M. H. Hansen and B. Yu. Model selection and the principle of minimum description length. *Journal of the American Statistical Association*, 96(454):746–774, 2001.

[64] J. Hao, B. P. Koester, T. A. Mckay, E. S. Rykoff, E. Rozo, A. Evrard, J. Annis, M. Becker, M. Busha, D. Gerdes, et al. Precision measurements of the cluster red sequence using an error-corrected gaussian mixture model. *The Astrophysical Journal*, 702(1):745, 2009.

[65] T. Hastie and R. Tibshirani. Discriminant analysis by Gaussian mixtures. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 155–176, 1996.

[66] G. E. Hinton, P. Dayan, and M. Revow. Modeling the manifolds of images of handwritten digits. *Neural Networks, IEEE Transactions on*, 8(1):65–74, 1997.

[67] H. Höfling and R. Tibshirani. Estimation of sparse binary pairwise markov networks using pseudo-likelihoods. *The Journal of Machine Learning Research*, 10:883–906, 2009.

[68] T. Hofmann and J. Buhmann. Pairwise data clustering by deterministic annealing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(1):1–14, 1997.

[69] T. Hofmann, B. Schölkopf, and A. Smola. Kernel methods in machine learning. *The Annals of Statistics*, pages 1171–1220, 2008.

[70] A. Holub, M. Welling, and P. Perona. Exploiting unlabelled data for hybrid object classification. In *Proc. Neural Information Processing Systems, Workshop Inter-Class Transfer*, volume 7, 2005.

[71] A. D. Holub, M. Welling, and P. Perona. Combining generative models and fisher kernels for object recognition. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 136–143. IEEE, 2005.

[72] D. W. Hosmer Jr. A comparison of iterative maximum likelihood estimates of the parameters of a mixture of two normal distributions under three different types of sample. *Biometrics*, pages 761–770, 1973.

[73] J.-T. Huang and M. Hasegawa-Johnson. On semi-supervised learning of gaussian mixture models for phonetic classification. In *Proceedings of the NAACL HLT 2009 Workshop on Semi-Supervised Learning for Natural Language Processing*, pages 75–83. Association for Computational Linguistics, 2009.

[74] J.-T. Huang and M. Hasegawa-Johnson. Semi-supervised training of gaussian mixture models by conditional entropy minimization. *Optimization*, 4:5, 2010.

[75] L. Hunt and M. Jorgensen. Theory & methods: Mixture model clustering using the multimix program. *Australian & New Zealand Journal of Statistics*, 41(2):154–171, 1999.

[76] L. Hunt and M. Jorgensen. Mixture model clustering for mixed data with missing information. *Computational statistics & data analysis*, 41(3):429–440, 2003.

[77] T. Jaakkola, D. Haussler, et al. Exploiting generative models in discriminative classifiers. *Advances in neural information processing systems*, pages 487–493, 1999.

[78] T. S. Jaakkola. Tutorial on variational approximation methods. *Advanced mean field methods: theory and practice*, pages 129–160, 2001.

[79] T. Jebara. *Discriminative, generative and imitative learning*. PhD thesis, Massachusetts Institute of Technology, 2001.

[80] T. Jebara and A. Pentland. Maximum conditional likelihood via bound maximization and the cem algorithm. In *NIPS*, volume 1, page 7, 1998.

[81] J. Jiang. A literature survey on domain adaptation of statistical classifiers. *URL: http://sifaka.cs.uiuc.edu/jiang4/domainadaptation/survey*, 2008.

[82] J. Jiang and C. Zhai. Instance weighting for domain adaptation in NLP. In *Annual Meeting-Association For Computational Linguistics*, volume 45, page 264, 2007.

[83] J. Jiang and C. Zhai. A two-stage approach to domain adaptation for statistical classifiers. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 401–410. ACM, 2007.

[84] T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 200–209. Morgan Kaufmann Publishers Inc., 1999.

[85] S. Johns. On identifying the population of origin of each observation in a mixture of observations from two normal populations. *Technometrics*, 12(3):553–563, 1970.

[86] A. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in neural information processing systems*, 14:841, 2002.

[87] G. Karakoulas and R. Salakhutdinov. Semi-supervised mixture-of-experts classification. In *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*, pages 138–145. IEEE, 2004.

[88] M. E. Khan, B. M. Marlin, G. Bouchard, and K. P. Murphy. Variational bounds for mixed-data factor analysis. In *NIPS*, pages 1108–1116, 2010.

[89] M. E. Khan, S. Mohamed, B. M. Marlin, and K. P. Murphy. A stick-breaking likelihood for categorical data analysis with latent gaussian models. In *International conference on Artificial Intelligence and Statistics*, pages 610–618, 2012.

[90] D. Klein, S. Kamvar, and C. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proc. ICML*, July 2002.

[91] T. Kohonen, G. Barna, and R. Chrisley. Statistical pattern recognition with neural networks: Benchmarking studies. In *Neural Networks, 1988., IEEE International Conference on*, pages 61–68. IEEE, 1988.

[92] W. Krzanowski. The location model for mixtures of categorical and continuous variables. *Journal of Classification*, 10(1):25–49, 1993.

[93] W. J. Krzanowski. Stepwise location model choice in mixed-variable discrimination. *Applied statistics*, 32(3):260–266, 1983.

[94] P. Kumar Mallapragada, R. Jin, A. K. Jain, and Y. Liu. Semiboost: Boosting for semi-supervised learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(11):2000–2014, 2009.

[95] J. Larsen, L. K. Hansen, A. S. Have, T. Christiansen, and T. Kolenda. Webmining: learning from the world wide web. *Computational statistics & data analysis*, 38(4):517–532, 2002.

[96] J. A. Lasserre, C. M. Bishop, and T. P. Minka. Principled hybrids of generative and discriminative models. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 87–94. IEEE, 2006.

[97] M. Law, A. Topchy, and A. Jain. Clustering with soft and group constraints. *Structural, Syntactic, and Statistical Pattern Recognition*, pages 662–670, 2004.

[98] M. Law, A. Topchy, and A. Jain. Model-based clustering with probabilistic constraints. In *Proceedings of SIAM data mining*, pages 641–645, 2005.

[99] M. Law, A. Topchy, and A. Jain. Model-based clustering with soft and probabilistic constraints. In *Technical report, Dept. of Comp. Sci. and Eng., Michigan State University*, 2005.

[100] N. D. Lawrence and M. I. Jordan. Semi-supervised learning via gaussian processes. In *NIPS*, volume 17, pages 753–760, 2004.

[101] O. Ledoit and M. Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of multivariate analysis*, 88(2):365–411, 2004.

[102] B. Leskes. The value of agreement, a new boosting algorithm. In *Learning Theory*, pages 95–110. Springer, 2005.

[103] S. Z. Li and S. Singh. *Markov random field modeling in image analysis*, volume 3. Springer, 2009.

[104] W. Li, M. Canini, A. Moore, and R. Bolla. Efficient application identification and the temporal and spatial stability of classification schema. *Computer Networks*, 53(6):790–809, 2009.

[105] Z. Li, J. Liu, and X. Tang. Pairwise constraint propagation by semidefinite programming for semi-supervised classification. In *Proceedings of the 25th international conference on Machine learning*, pages 576–583. ACM, 2008.

[106] C. Liu and D. B. Rubin. The ECME algorithm: a simple extension of EM and ECM with faster monotone convergence. *Biometrika*, 81(4):633–648, 1994.

[107] Z. Lu and T. Leen. Penalized probabilistic clustering. *Neural Computation*, 19(6):1528–1567, 2007.

[108] Z. Lu and T. Leen. Semi-supervised clustering with pairwise constraints: A discriminative approach. In *Eleventh International Conference on Artificial Intelligence and Statistics*, 2007.

[109] D. MacKay. *Information theory, Inference, and Learning algorithms*, pages 422–433. Cambridge Univ Press, 2003.

[110] A. McCallum, K. Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.

[111] G. McLachlan and T. Krishnan. *The EM algorithm and extensions*, volume 382. John Wiley & Sons, 2007.

[112] G. McLachlan and D. Peel. *Finite mixture models*, volume 299. Wiley-Interscience, 2000.

[113] G. McLachlan and D. Peel. *Finite mixture models*, pages 136–141. Wiley-Interscience, 2000.

[114] G. J. McLachlan. Estimating the linear discriminant function from initial samples containing a small number of unclassified observations. *Journal of the American Statistical Association*, 72(358):403–406, 1977.

[115] G. J. McLachlan, R. Bean, and D. Peel. A mixture model-based approach to the clustering of microarray expression data. *Bioinformatics*, 18(3):413–422, 2002.

[116] G. J. McLachlan, D. Peel, and R. Bean. Modelling high-dimensional data by mixtures of factor analyzers. *Computational Statistics & Data Analysis*, 41(3):379–388, 2003.

[117] P. D. McNicholas and T. B. Murphy. Model-based clustering of microarray expression data via latent gaussian mixture models. *Bioinformatics*, 26(21):2705–2712, 2010.

[118] X.-L. Meng and D. B. Rubin. Maximum likelihood estimation via the ecm algorithm: A general framework. *Biometrika*, 80(2):267–278, 1993.

[119] X.-L. Meng and D. Van Dyk. The EM algorithman old folk-song sung to a fast new tune. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 59(3):511–567, 1997.

[120] D. Miller and J. Browning. A mixture model and EM-based algorithm for class discovery, robust classification, and outlier rejection in mixed labeled/unlabeled data sets. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(11):1468–1483, 2003.

[121] D. Miller and S. Pal. Transductive methods for the distributed ensemble classification problem. *Neural computation*, 19(3):856–884, 2007.

[122] D. Miller and H. Uyar. A mixture of experts classifier with learning based on both labelled and unlabelled data. *Advances in Neural Information Processing Systems*, pages 571–577, 1997.

[123] D. J. Miller, Y. Zhang, and G. Kesidis. Decision aggregation in distributed classification by a transductive extension of maximum entropy/improved iterative scaling. *EURASIP Journal on Advances in Signal Processing*, 21, 2008.

[124] D. J. Miller, Y. Zhang, G. Yu, Y. Liu, L. Chen, C. D. Langefeld, D. Herrington, and Y. Wang. An algorithm for learning maximum entropy probability models of disease risk that efficiently searches and sparingly encodes multilocus genomic interactions. *Bioinformatics*, 25(19):2478–2485, 2009.

[125] T. K. Moon. The expectation-maximization algorithm. *Signal processing magazine, IEEE*, 13(6):47–60, 1996.

[126] A. Moore and D. Zuev. Internet traffic classification using Bayesian analysis techniques. In *ACM SIGMETRICS Performance Evaluation Review*, volume 33, pages 50–60. ACM, 2005.

[127] K. P. Murphy. *Machine learning: A probabilistic perspective*. MIT Press, 2012.

[128] K. P. Murphy. *Machine learning: A probabilistic perspective*, pages 129–130. MIT Press, 2012.

[129] K. P. Murphy. *Machine learning: A probabilistic perspective*, pages 732–763. MIT Press, 2012.

[130] R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. *NATO ASI Series D Behavioural And Social Sciences*, 89:355–370, 1998.

[131] S.-K. Ng and G. McLachlan. Expert networks with mixed continuous and categorical feature variables: a location modeling approach. *Machine Learning Research Progress, Nova Science, Hauppauge, New York*, pages 1–14, 2008.

[132] S.-K. Ng and G. J. McLachlan. Normalized gaussian networks with mixed feature data. In *AI 2005: Advances in Artificial Intelligence*, pages 879–882. Springer, 2005.

[133] T. Nguyen and G. Armitage. A survey of techniques for Internet traffic classification using machine learning. *Communications Surveys & Tutorials, IEEE*, 10(4):56–76, 2008.

[134] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 86–93. ACM, 2000.

[135] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2):103–134, 2000.

[136] A. O'Hagan, J. Forster, and M. G. Kendall. *Bayesian inference*. Arnold London, 2004.

[137] T. J. O'neill. Normal discrimination with unclassified observations. *Journal of the American Statistical Association*, 73(364):821–826, 1978.

[138] M. Opper and D. Saad. *Advanced mean field methods: Theory and practice*. MIT press, 2001.

[139] S. Parise and M. Welling. Learning in markov random fields: An empirical study. In *Joint Statistical Meeting*, volume 4, page 7. Citeseer, 2005.

[140] D. Pelleg and D. Baras. K-means with large and noisy constraint sets. *Machine Learning: ECML 2007*, pages 674–682, 2007.

[141] H. Permuter, J. Francos, and I. Jermyn. A study of gaussian mixture models of color and texture features for image classification and segmentation. *Pattern Recognition*, 39(4):695–706, 2006.

[142] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[143] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn. Speaker verification using adapted gaussian mixture models. *Digital signal processing*, 10(1):19–41, 2000.

[144] P. Rigollet. Generalization error bounds in semi-supervised classification under the cluster assumption. *arXiv preprint math/0604233*, 2006.

[145] A. Roche. EM algorithm and variants: An informal tutorial. Technical report, 2011.

[146] K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proceedings of the IEEE*, 86(11):2210–2239, 1998.

[147] C. Rosenberg, M. Hebert, and H. Schneiderman. Semi-supervised self-training of object detection models. 2005.

[148] L. Ruan, M. Yuan, and H. Zou. Regularized parameter estimation in high-dimensional gaussian mixture models. *Neural computation*, 23(6):1605–1622, 2011.

[149] M. Saerens, P. Latinne, and C. Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure. *Neural Computation*, 14(1):21–41, 2002.

[150] S. Satpal and S. Sarawagi. Domain adaptation of conditional probability models via feature subsetting. *Knowledge Discovery in Databases: PKDD 2007*, pages 224–235, 2007.

[151] L. K. Saul and S. T. Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *The Journal of Machine Learning Research*, 4:119–155, 2003.

[152] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[153] M. Seeger et al. Learning with labeled and unlabeled data. Technical report, technical report, University of Edinburgh, 2001.

[154] B. M. Shahshahani and D. A. Landgrebe. The effect of unlabeled samples in reducing the small sample size problem and mitigating the hughes phenomenon. *Geoscience and Remote Sensing, IEEE Transactions on*, 32(5):1087–1095, 1994.

[155] N. Shental, A. Bar-Hillel, T. Hertz, and D. Weinshall. Computing Gaussian mixture models with EM using equivalence constraints. *Advances in Neural Information Processing Systems*, 16:465–472, 2004.

[156] V. Sindhwani and D. S. Rosenberg. An rkhs for multi-view learning and manifold co-regularization. In *Proceedings of the 25th international conference on Machine learning*, pages 976–983. ACM, 2008.

[157] J. Sinkkonen, S. Kaski, and J. Nikkilä. Discriminative clustering: Optimal contingency tables by learning metrics. In *Machine Learning: ECML 2002*, pages 418–430. Springer, 2002.

[158] M. Soleymani Baghshah and S. Bagheri Shouraki. Non-linear metric learning using pairwise similarity and dissimilarity constraints and the geometrical structure of data. *Pattern Recognition*, 43(8):2982–2992, 2010.

[159] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2. IEEE, 1999.

[160] A. Stuart, K. Ord, and S. Arnold. Classical inference and the linear model, volume 2a of kendalls advanced theory of statistics. *London: Arnold*, 1999.

[161] M. Szummer and T. Jaakkola. Information regularization with partially labeled data. In *nips*, pages 1025–1032, 2002.

[162] M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analyzers. *Neural computation*, 11(2):443–482, 1999.

[163] V. Vapnik. *The nature of statistical learning theory*. springer, 2000.

[164] J. Vert, K. Tsuda, and B. Schölkopf. A primer on kernel methods. *Kernel Methods in Computational Biology*, pages 35–70, 2004.

[165] D. Ververidis and C. Kotropoulos. Emotional speech classification using gaussian mixture models. In *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, pages 2871–2874. IEEE, 2005.

[166] N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research*, 9999:2837–2854, 2010.

[167] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained k-means clustering with background knowledge. In *Proc. of the Eighteenth International Conference on Machine Learning*, pages 577–584, 2001.

[168] J. Wang, T. Jebara, and S.-F. Chang. Semi-supervised learning using greedy max-cut. *The Journal of Machine Learning Research*, 14(1):771–800, 2013.

[169] J. Wang and S.-W. Luo. Exploiting ensemble method in semi-supervised learning. In *Machine Learning and Cybernetics, 2006 International Conference on*, pages 1104–1107. IEEE, 2006.

[170] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. *Advances in Neural Information Processing Systems*, 15:505–512, 2002.

[171] T. Yang and C. E. Priebe. The effect of model misspecification on semi-supervised classification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(10):2093–2103, 2011.

[172] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196. Association for Computational Linguistics, 1995.

[173] D.-Y. Yeung and H. Chang. A kernel approach for semisupervised metric learning. *Neural Networks, IEEE Transactions on*, 18(1):141–149, 2007.

[174] S. Yu and J. Shi. Segmentation given partial grouping constraints. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):173–183, 2004.

[175] A. Yuille, P. Stolorz, and J. Utans. Statistical physics, mixtures of distributions, and the EM algorithm. *Neural Computation*, 6(2):334–340, 1994.

[176] M. Zanda and G. Brown. A study of semi-supervised generative ensembles. In *Multiple Classifier Systems*, pages 242–251. Springer, 2009.

[177] T. Zhang and F. Oles. The value of unlabeled data for classification problems. In *Proceedings of the Seventeenth International Conference on Machine Learning,(Langley, P., ed.)*, pages 1191–1198. Citeseer, 2000.

[178] J.-H. Zhao and P. L. Yu. Fast ml estimation for the mixture of factor analyzers via an ecm algorithm. *Neural Networks, IEEE Transactions on*, 19(11):1956–1961, 2008.

[179] Q. Zhao and D. Miller. Mixture modeling with pairwise, instance-level class constraints. *Neural Computation*, 17(11):2482–2507, 2005.

[180] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. *Advances in neural information processing systems*, 16(16):321–328, 2004.

[181] Z.-H. Zhou. When semi-supervised learning meets ensemble learning. *Frontiers of Electrical and Electronic Engineering in China*, 6(1):6–16, 2011.

[182] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530.

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

[184] X. Zhu, Z. Ghahramani, J. Lafferty, et al. Semi-supervised learning using Gaussian fields and harmonic functions. In *ICML*, volume 3, pages 912–919, 2003.

[185] X. Zhu and A. B. Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.

[186] X. Zhu and J. Lafferty. Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 1052–1059. ACM, 2005.

[187] G. Zou, G. Kesidis, and D. Miller. A flow classifier with tamper-resistant features and an evaluation of its portability to new domains. *Selected Areas in Communications, IEEE Journal on*, 29(7):1449–1460, 2011.

Vita

Jayaram Raghuram

Education:

- 2008 – 2014: The Pennsylvania State University, Electrical Engineering
  PhD, supervised by Prof. David J. Miller and Prof. George Kesidis
- 2004 – 2008: Anna University, Electronics and Communications Eng., Chennai, India
  B.E, senior project supervised by Prof. Nelson Iruthayanathan

Publications:

- Raghuram, J., Miller, D.J., Kesidis, G., "Instance-Level Constraint Based Semi-supervised Learning With Imposed Space-Partitioning", IEEE Transactions in Neural Networks and Learning Systems, Vol 25(8), 2014. DOI: 10.1109/TNNLS.2013.2294459.
- Raghuram, J., Miller, D.J., Kesidis, G., "Unsupervised, low latency anomaly detection of algorithmically generated domain names by generative probabilistic modeling", Journal of Advanced Research, Vol 5(4), pp. 423–433, 2014. DOI: 10.1016/j.jare.2014.01.001.
- Raghuram, J., Kesidis, G., Miller, D.J., Levitt, K., Rowe, J., Scaglione, A., "Generation bidding game with flexible demand", 9th International workshop on feedback computing, 2014.
- Raghuram, J., Miller, D.J., Kesidis, G., "Semisupervised domain adaptation for mixture model based classifiers", Proceedings of the 46th annual Conference on Information Science and Systems (CISS), 2012. DOI: 10.1109/CISS.2012.6310708.
- Miller, D.J., Raghuram, J., Kesidis, G., and Collins, C.M., "Improved generative semisupervised learning based on finely grained component-conditional class labeling", Neural Computation, Vol 24(7), 1926–1966, 2012. DOI: 10.1162/NECO_a_00284.
- Celik, Z.B., Raghuram, J., Kesidis, G., and Miller, D.J., "Salting public traces with attack traffic to test flow classifiers", Proceedings of the 4th conference on Cyber Security Experimentation and Test (CSET), 2011.
- Chen, Li, et.al., "Comparative analysis of methods for detecting interacting loci", BMC Genomics, Vol 12(1), 2011. DOI: 10.1186/1471-2164-12-344.