LETTER **Communicated by Gert Van Dijck**

Improved Generative Semisupervised Learning Based on Finely Grained Component-Conditional Class Labeling

David J. Miller
djmiller@engr.psu.edu
Jayaram Raghuram
jzr148@psu.edu
Department of Electrical Engineering, Pennsylvania State University, University Park, PA 16802, U.S.A.

George Kesidis
Kesidis@engr.psu.edu
Department of Electrical Engineering, Pennsylvania State University, University Park, PA 16802, U.S.A., and Center for NMR Research, Radiology, Pennsylvania State University College of Medicine, Hersey, PA 17033, U.S.A.

Christopher M. Collins
cmcollins@psu.edu
Department of Electrical Engineering, Pennsylvania State University, University Park, PA 16802, U.S.A.

We introduce new inductive, generative semisupervised mixtures with more finely grained class label generation mechanisms than in previous work. Our models combine advantages of semisupervised 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 or prototypes. For our NN-based method, we propose a novel two-stage stochastic data generation, with all samples first generated using a standard finite mixture and then all class labels generated, conditioned on the samples and their components of origin. This mechanism entails an underlying Markov random field, specific to each mixture component or cluster. We invoke the pseudo-likelihood formulation, which forms the basis for an approximate generalized expectation-maximization model learning algorithm. Our, NP-based model overcomes a problem with the NN-based model that manifests at very low labeled fractions. Both models are advantageous when within-component class proportions are not constant over the feature space region "owned by" a component. The Practicality of this scenario is borne out by experiments on UC Irvine data sets, which demonstrate significant gains in classification accuracy over

previous semisupervised mixtures and also overall gains, over KNN classification. Moreover, for very small labeled fractions, our methods overall outperform supervised linear and nonlinear kernel support vector machines.

1 Introduction

In many practical applications of statistical classification, a large number of unlabeled training samples can be easily collected, while only a limited number of labeled samples are available. Such domains include image and text document databases and Internet traffic traces. Since labeled training samples may be expensive or time-consuming to obtain, machine learning techniques that can make effective use of both labeled and unlabeled data have received considerable recent attention (Zhu, 2008). In the most common objective for semisupervised learning, one augments a small labeled training set with a large set of unlabeled samples, with the aim of building a more accurate statistical classifier than that learned by making use solely of the available labeled data. In this way, semisupervised learning is extending the traditional supervised learning problem. Less commonly, an unlabeled data set may be augmented by labeled samples, with the objective of more accurately clustering the data or modeling the data density—in other words, semisupervised extension of the unsupervised learning problem. In this letter we primarily focus on the former (statistical classification) objective.

Semisupervised approaches can be categorized into two groups: generative and discriminative methods. In the former, (Miller & Uyar, 1997; Nigam, McCallum, Thrun, & Mitchell, 2000), a stochastic generation mechanism is explicitly hypothesized for all the observed data, with the associated model learning aiming to fit the data well. This is typically cast as either a maximum likelihood or a Bayesian learning problem (Duda, Hart & Stork, 2001). In discriminative methods (Chapelle, Schölkopf & Zien, 2006; Blum & Mitchell, 1998), the focus is on learning a discriminant function (or the class posterior distribution) that well separates data from the different classes. Zhu (2008) loosely summarizes the difference between these two approaches by saying that generative methods learn the joint distribution over the class C and feature vector $\underline{X}$, $P[C, \underline{X}]$, while discriminative methods exclusively learn the posterior $P[C|\underline{X}]$.

Over the past decade or so, there has been a substantial amount of work, particularly on discriminative approaches, including cotraining (Blum & Mitchell, 1998), transductive support vector machines (Joachims, 1999; Chapelle, Chi, & Zien, 2006), gaussian process approaches (Lawrence & Jordan, 2005), information regularization (Szummer & Jaakkola, 2003; Corduneanu & Jaakkola, 2003), and a number of graph-based methods (Belkin, Niyogi, & Sindhwani, 2006; Zhu & Lafferty, 2005). A detailed review of

semisupervised learning methods can be found in Zhu (2008). Here our focus is on generative semisupervised learning. Generative, mixture-based models (Miller & Uyar, 1997; Nigam et al., 2000; Shahshahani & Landgrebe, 1994) represent one of the earliest approaches to semisupervised learning. Moreover, this approach is suitable only if the feature data are well modeled by a finite mixture of the assumed parametric density form. If the mixture model assumption is incorrect, semisupervised mixtures may give poor performance (Cohen, Cozman, Sebe, Cirelo, & Huang, 2004). However, all semisupervised methods make underlying assumptions and have limits to their applicability. For example, as noted in Zhu (2008), cotraining requires a (natural) partitioning of the feature space into two subsets; transductive SVMs and the works in Lawrence and Jordan (2005) and Chapelle and Zien (2005) 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 follow an underlying manifold structure and also require transductive inference (Zhu, 2008), which may be of high complexity.

Moreover, mixture models such as gaussian mixtures are ubiquitous and appropriate, and they are used very effectively in a variety of application domains: modeling speech (Ververidis & Kotropoulos, 2005; Reynolds, Quatieri, & Dunn, 2000), image content (Stauffer & Grimson, 1999; Permuter, Francos, & Jermyn, 2006), in bioinformatics (McNicholas & Murphy, 2010), and in modeling many types of scientific data (Hao et al., 2009; Gavin & Hu, 2005). Mixtures of other distributions, such as the multinomial, naturally model text documents (Nigam et al., 2000). Accordingly, generative, mixture-based semisupervised learning is highly suitable for a large number of real-world application domains. Likewise, improvements to existing semisupervised mixtures should find significant application. Thus, in this letter, we focus on generative methods and in particular on improving the model for class label generation within semisupervised 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 or inference problem (which hypothesizes stochastic generation of the parameters as well as the data; Duda et al., 2001). Bayesian extension of the approaches developed here is identified as a good topic for future work.

In section 2, we review generative semisupervised learning methods, focusing particularly on semisupervised mixture modeling (Miller & Uyar, 1997; Nigam et al., 2000; Shahshahani & Landgrebe, 1994) 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 (Cohen et al., 2004), it was demonstrated that if statistical modeling assumptions are incorrect, semisupervised 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 Miller and Uyar (1997), Nigam et al. (2000), and Shahshahani and Landgrebe (1994) as 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 Miller and Uyar (1997), Nigam et al. (2000), and Shahshahani and Landgrebe (1994) may bias the learned solution away from the ground-truth mixture solution. In sections 3 and 4, we develop our new generative semisupervised approaches, which achieve more finely grained class probability modeling within each mixture component or cluster than previous works. In section 5, we discuss the relationship to previous work. In section 6, we present experimental comparisons against alternative semisupervised and supervised methods. The results will show that our models' finely grained class modeling yields significantly better classification accuracy than the previous semisupervised mixtures (Miller & Uyar, 1997; Nigam et al., 2000; Shahshahani & Landgrebe, 1994). 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. In section 7, we identify several possible extensions that may be considered in future work. The last section gives our conclusions.

2 Limitations of Previous Semisupervised Mixtures

2.1 Notation. Consider a random feature vector $\underline{X} \in \mathcal{R}^d$, 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 $\{P[M = l] \equiv \alpha_l, l = 1, \ldots, L\}$, $0 \leq \alpha_l \leq 1$, $\sum_{l=1}^{L} \alpha_l = 1$, be the masses (proportions) for an L-component mixture model (McLachlan & Peel, 2000) with component densities $f_{\underline{X}|l}(\underline{x}|\theta_l), l = 1, \ldots, L$, where θ_l is the parameter set for the lth 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 semisupervised mixture modeling, one jointly models the feature vector $\underline{X}$ and the class label C, that is, 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 $\mathcal{X}_u = \{\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_{N_u}\}$, $\underline{x}_i \in \mathcal{R}^d$; the labeled subset by $\mathcal{X}_l = \{(\underline{x}_{N_u+1}, c_{N_u+1}), (\underline{x}_{N_u+2}, c_{N_u+2}), \ldots, (\underline{x}_{N_u+N_l}, c_{N_u+N_l})\}$, $\underline{x}_i \in \mathcal{R}^d$, $c_i \in \mathcal{C}$; 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 $\tilde{\mathcal{X}} = \{\underline{x}_1, \ldots, \underline{x}_{N_u}, \underline{x}_{N_u+1}, \ldots, \underline{x}_{N_u+N_l}\}$; 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 $\mathcal{I}_l = \{N_u + 1, \ldots, N_u + N_l\}$.

2.2 Hard Versus Soft Class-to-Component Assignments. Early work on semisupervised mixture modeling includes (Miller & Uyar, 1997;

Nigam et al., 2000; Shahshahani & Landgrebe, 1994). There has also been a significant amount of follow-up interest in this area (Chawla & Karakoulas, 2005; Karakoulas & Salakhutdinov, 2004; Larsen, Hansen, Have, Christiansen, & Kolenda, 2002; Saerens, Latinne, & Decaestecker, 2002; Seeger, 2001; Shental & Weinshall, 2004; Zhu, 2008). There is also related work on semisupervised clustering (Wagstaff, Cardie, Rogers, & Schroedl, 2001; Basu & Mooney, 2003) and semisupervised mixture modeling (Shental & Weinshall, 2004; Law, Topchy, & Jain, 2004; Zhao & Miller, 2005) where, rather than labels, the supervision takes the form of class must-link and cannot-link constraints. Most of these approaches are similar to Nigam et al. (2000) and Shahshahani and Landgrebe (1994) in that they essentially make a one-class-per-component assumption: all clusters or mixture components are each assigned to a single class, with all data samples within the cluster or component assumed to belong to this (assigned) class. The model in Miller and Uyar (1997) differs from these approaches in not making this assumption. We next first briefly summarize Miller and Uyar (1997) and how it differs from Nigam et al. (2000) and Shahshahani and Landgrebe (1994). We then identify two fundamental limitations common to all three of these methods, which will motivate the development of our new semisupervised mixtures in sections 3 and 4.

Miller and Uyar (1997) assumed the following stochastic data generation for the pooled data $\mathcal{X}$:

1. Independently, for each $\underline{x}_i \in \mathcal{X}_u$: (a) randomly select a mixture component, $M = j$, according to $\{\alpha_l\}$; (b) randomly generate $\underline{x}_i$ according to $f_{\underline{X}|j}(\underline{x}|\theta_j)$.
2. Independently, for each $(\underline{x}_i, c_i) \in \mathcal{X}_l$: (a) randomly select a mixture component, $M = j$, according to $\{\alpha_l\}$; (b) randomly generate $\underline{x}_i$ according to $f_{\underline{X}|j}(\underline{x}|\theta_j)$; and (c) 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:

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

The model parameters $\Theta = \{\{\theta_l, \alpha_l, \{\beta_{c|l}\}\}, l = 1, \ldots, L\}$ were estimated by maximizing the joint data likelihood over $\mathcal{X}$:

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

This was achieved in a locally optimal fashion via an expectation-maximization (EM) algorithm (Dempster, Laird, & Rubin, 1977), developed in Miller and Uyar (1997).

The methods in Nigam et al. (2000) and Shahshahani and Landgrebe (1994) are closely related to those of Miller and Uyar (1997). Like Miller and Uyar (1997), Nigam et al. (2000) learned the mixture model to maximize the joint data likelihood over $\mathcal{X}$ (whereas Shahshahani & Landgrebe, 1994, maximized the class-conditional likelihood). Also like Miller and Uyar (1997), Nigam et al. (2000) developed an EM algorithm for parameter learning. However, a distinction among Miller and Uyar (1997) and Nigam et al. (2000) Shahshahani and Landgrebe (1994) lies in how these methods hypothesize class label generation. Unlike Miller and Uyar (1997), in Nigam et al. (2000) and Shahshahani and Landgrebe (1994) it was assumed that each mixture component is exclusively assigned to a single class, that is, it generates only labeled samples from this assigned class. Thus, in Nigam et al. (2000) and Shahshahani and Landgrebe (1994), 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 $\sum_{k=1}^{K} z_{k|l} = 1 \ \forall l$ (each component is assigned to a single class). It is also highly desirable to satisfy $\sum_{l=1}^{L} z_{k|l} \geq 1 \ \forall k$ (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 $\mathcal{L}(\mathcal{X})$, and the associated EM algorithm from Nigam et al. (2000) are specializations of Miller and Uyar (1997), achieved by constraining $\beta_{c|l} \in \{0, 1\}$.[1]

[1] Actually, Nigam et al. (2000) did not reestimate $\{z_{k|l}\}$ 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 (Fessler &

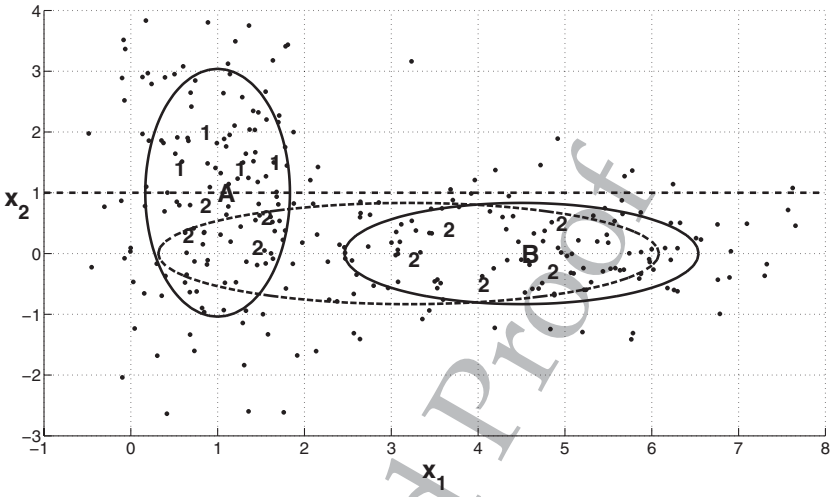

Figure 1: A 2D mixture example with two components, A and B, one of which (A) generates labeled samples from two different classes. Also shown (dashed contour) is an (incorrect) learned component that exclusively generates samples from class 2.

Allowing stochastic (soft) label generation, given the mixture component of origin, that is, $\beta_{k|l} \in [0, 1], \sum_{k=1}^{K} \beta_{k|l} = 1$, as in Miller and Uyar (1997), is advantageous in some cases because the assumption of exclusive component-to-class assignments in Nigam et al. (2000) and Shahshahani and Landgrebe (1994) may be too inflexible to accurately model the given semisupervised data and capture its underlying cluster structure. Consider the illustrative example shown in Figure 1. Here, we depict a 2D semisupervised 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 Nigam et al. (2000) and Shahshahani and Landgrebe (1994) 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

Hero, 1994; Meng & Dyk, 1997; Graham & Miller, 2006), 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.

[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.

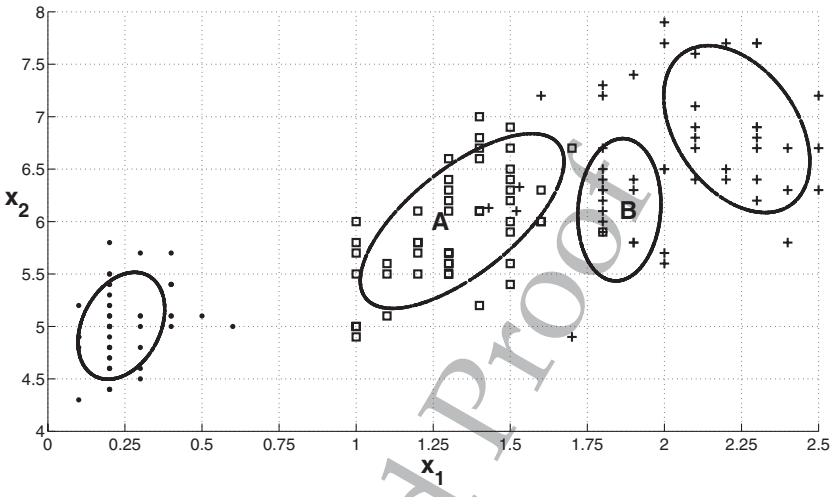

Figure 2: A 2D projection (onto the first and fourth features) of the three-class Iris data set (150 instances, 4 dimensions), along with a four-component mixture modeling solution. Note that components A and B own samples from two classes (squares and pluses).

outliers (those from component A) would degrade parameter estimates for this learned component. A solution based on Nigam et al. (2000) is depicted with a dashed ellipse for the learned component that generates labeled samples from class 2. The model in Miller and Uyar (1997), 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 reflect the proportion of labeled samples from each class within component 1—four labeled samples from each class).

Since the reader might consider the within-component class impurity in this example to be artificial, we also consider a 2D projection of the real-world 4D UC Irvine repository Iris data set in Figure 2. In this case, we modeled the data without using any labels, selected the number of components as four based on the Bayesian information criterion (BIC; Schwarz, 1978), and chose the best four-component gaussian mixture solution (log-likelihood sense) based on 50 EM runs starting from different random parameter initializations. We then plotted this mixture solution, with class labels now indicated for each sample by one of three different symbols. Note that components A and B clearly "own" samples from two different classes (consistent with the Figure 1 example). Again, Miller and Uyar (1997), which allows components to generate samples from multiple classes, is more suitable than Nigam et al. (2000) and Shahshahani and

Landgrebe(1994) for learning the underlying cluster structure (assuming the best-learned gaussian mixture solution well represents this unknown structure) from semisupervised data sets such as this, where ground-truth clusters may own data from more than one class. However, while Miller and Uyar (1997) give more flexible within-component class label generation than Nigam et al. (2000) and Shahshahani and Landgrebe (1994), this degree of flexibility is still insufficient, as we next elaborate by highlighting fundamental limitations that Miller and Uyar (1997), Nigam et al. (2000) and Shahshahani and Landgrebe (1994) all share in seeking to build a statistical classifier based on semisupervised data when the "one class per cluster" assumption is violated.

2.3 Crude Within-Component Class Labeling. First, again consider the data in Figure 1. Although Miller and Uyar (1997) 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, Miller and Uyar (1997) will estimate $P[C = c|\underline{x}] = \frac{1}{2}$, $c = 1, 2$, even though it would be far better to choose $P[C = 1|\underline{x}, x_2 > 1] = 1$ and $P[C = 2|\underline{x}, x_2 < 1] = 1$ for $\underline{x}$ from component A; more finely grained within-component class labeling is needed to give an accurate classification in this example.

There are several ways this can be achieved in principle. First, as suggested in Nigam et al. (2000), one can simply introduce more components into the model, splitting components with high class impurity into multiple smaller components that exhibit class purity. This can certainly improve classification accuracy. However, there are two potential problems with this remedy. First, in the Figure 1 example, there really are two gaussian components; it is only the class label, not the feature vector, that requires more complex modeling. So 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 suboptimal classification accuracy, especially if the within-cluster class structure is complicated. Consider the example in Figure 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 conform well 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 suboptimal class decision boundary within cluster B. In our experimental results, we will explore the classification benefit achieved simply by using a large number of

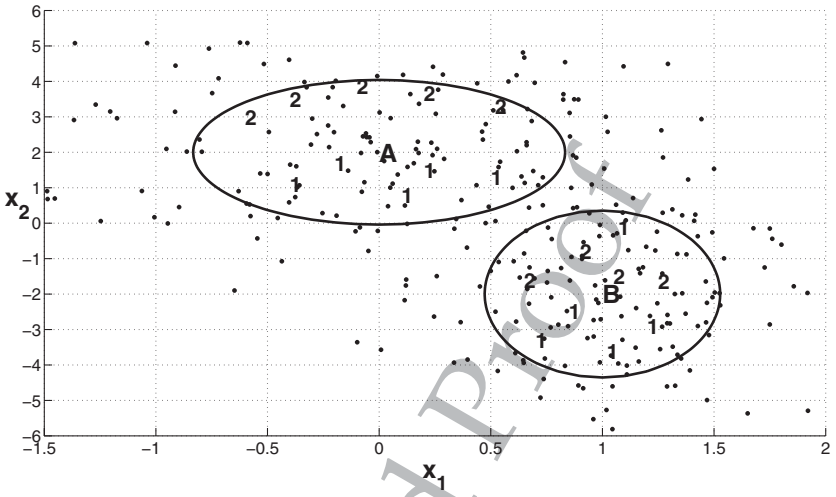

Figure 3: An illustrative example with two ground-truth components, each of which generates labeled samples from two classes. Assuming a two-component model, the method from Nigam et al. (2000), with exclusive class-to-component assignments, will fail to learn an accurate classifier on this example. Moreover, splitting each natural component into two class-pure components will also give suboptimal classification accuracy in this example.

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 finely grained, "discriminative" class modeling within each natural cluster or 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 or mixture modeling, ignoring the available class labels; (2) make exclusive (0-1) assignments of each labeled sample to its best-fitting component; and (3) for each component, directly form a nearest-neighbor classifier (Duda et al., 2001) based simply on the labeled samples assigned to that component. An (accurate) classifier obtained in this way is depicted via an illustrative example in Figure 4.

In summary, from the examples, we anticipate that a semisupervised mixture model with more finely grained ("discriminative") within-component class modeling may give improved classification accuracy compared with previous semisupervised mixtures. Moreover, we emphasize that while the examples shown in this section (excepting Figure 2) were synthetically generated and thus merely illustrative, the practical benefits

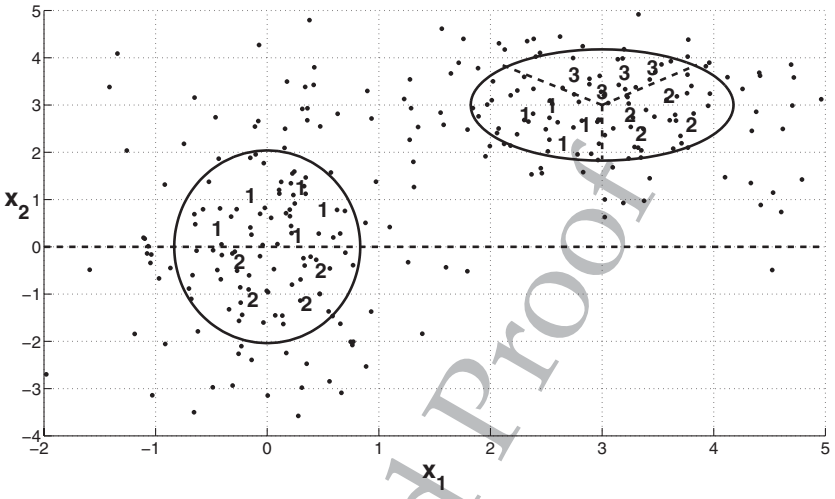

Figure 4: Nearest-neighbor classification applied within each (learned) cluster achieves accurate within-cluster classification in this 2D example.

of finely grained labeling will be demonstrated on real-world data sets in our experimental results.

In sections 3 and 4, we will show that finely 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 semisupervised mixtures with more involved stochastic data generation than previous proposals. The model in section 3 performs finely grained class modeling in a nonparametric fashion, with the resulting model based on Markov random fields (MRFs) (Li, 2009), and with the model learning based on maximization of the pseudo-likelihood function, associated with MRFs (Li, 2009). The method in section 4, alternatively, takes a parametric approach to finely grained class modeling and is specifically designed to overcome a problem experienced by the nonparametric method at very low labeled fractions. Moreover, the model learning maximizes a true likelihood function rather than the pseudo-likelihood considered in section 3. On the other hand, unlike the method in section 3, the parametric approach may require the integration of a model order selection procedure within the model learning, in order to introduce the "right" level of class modeling complexity within each mixture component. Both of our finely grained approaches model the joint distribution $P[C, \underline{X}]$ and in this sense are purely generative. However, since these models give more detailed within-component class posterior modeling than previously proposed semisupervised mixtures, one might also reasonably describe them as hybrid generative/discriminative.

In section 6, we 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 semisupervised 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.

3 A Nonparametric Finely Grained Labeling Model

3.1 Stochastic Data Generation for the New Model. As in Miller and Uyar(1997), we treat class labels as data and model their stochastic generation, along with the feature data. In a departure from previous methods, we hypothesize the generation of $\mathcal{X}$ in two (wholly distinct) stages. In the first stage, the feature vectors for all samples, both unlabeled and labeled ($\bar{\mathcal{X}}$), are generated, independent and identically distributed (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, conditioned on both $\bar{\mathcal{X}}$ and on $\mathcal{V} \equiv \{V_{j|i}, \forall i, j\}$, where $V_{j|i} \in \{0, 1\}$ and $\sum_{j=1}^{L} V_{j|i} = 1, \forall i$, that is, conditioned on knowledge of the component of origin for each sample.

In our approach, the class labels in $\mathcal{C}_l$ are generated according to a within-component class posterior model, $P[C = c|M = j, \underline{x}]$. This is what makes our approach finely grained. The modeling framework we develop in this section is in principle amenable to many different choices for this finely grained posterior. For example, we could model the class posterior, conditioned on $\underline{x}$ belonging to a given component, as a logistic function a randomized (soft) linear discriminant function), a nonlinear (e.g., kernelized) logistic function (i.e., a randomized generalized linear discriminant function), or even 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; for example, a single linear discriminant function does not have sufficient power to discriminate the three classes present in one cluster in Figure 4. Likewise, a kernelized logistic function may increase 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 Figure 4). 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 that 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

$\bar{\mathcal{X}}$ and determining a mixture component of origin for each sample, $\mathcal{V}$. Further, for the moment, suppose that we have already generated the labels for $N_l - 1$ of the N_l labeled samples, that is, it is, left only to generate the label for the remaining sample, i^*, which was generated by component j. Consider the set of labeled samples originating from this same mixture component: $\{i \in \mathcal{I}_l : V_{j|i} = 1, i \neq i^*\}$, of size N_j. These labeled samples could be used to form a hard, within-component, K-nearest-neighbor classifier (with $K = N_j$), similar to the method discussed in section 2. Alternatively, we define a randomized version of this classifier:

$$P[C_{i^*} = c | \{c_i, i \in \mathcal{I}_l - i^*\}, \bar{\mathcal{X}}, \mathcal{V}] = \frac{\sum_{i=N_u+1:c_i=c, i\neq i^*}^{N_u+N_l} V_{j|i} \exp(-a_j ||\underline{x}_i - \underline{x}_{i^*}||^2)}{\sum_{i=N_u+1, i\neq i^*}^{N_u+N_l} V_{j|i} \exp(-a_j ||\underline{x}_i - \underline{x}_{i^*}||^2)}, \quad \forall c, \text{ for } \mathcal{V} : V_{j|i^*} = 1. \tag{3.1}$$

Note that C_{i^*} in equation 3.1 is in fact independent of all labeled samples that do not originate from component j, conditioned on all other labeled samples that do originate from component j. Here, a_j is a component-specific scale parameter. As $a_j \to \infty$, $P[C_{i^*} = c | \{c_i, i \in \mathcal{I}_l - i^*\}, \bar{\mathcal{X}}, \mathcal{V}] \to \{0, 1\}$, consistent with a hard N_j-nearest-neighbor rule applied within component j. Note that the complexity of this within-component posterior (decision rule) is not determined by how many parameters we decide to include in the model; it is automatically determined by the number of labeled samples (and the class variety of same) falling in 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 randomized KNN rule is automatically very simple. If there are many labeled samples in a given component, the randomized KNN rule will be more complex. This is automatically determined or controlled based on the amount of available labeled data rather than requiring careful model selection to control the model complexity. Moreover, the posterior, equation 3.1, introduces very few additional parameters—just one, scaling parameter a_j for each component. Different soft nearest-neighbor rules, achieved, for example, by replacing the (implicit) kernel choice in equation 3.1, $k(\underline{x}, \underline{y}) = \exp(-a_j \|\underline{x} - \underline{y}\|^2)$ by alternative kernels, can also be considered.

As noted above, a key point about equation 3.1 is that it specifies that the probability for any label C_i, conditioned on $\{C_m, m \in \mathcal{I}_l - i\}$, 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 $P[\mathcal{C}_l | \bar{\mathcal{X}}, \mathcal{V}]$ factors as a product of mutually exclusive joint label subset distributions, one per component, each involving the labeled samples that originate from

that component, $P[\mathcal{C}_l|\bar{\mathcal{X}}, \mathcal{V}] = \prod_{j=1}^{L} P[\mathcal{C}_l^{(j)}|\bar{\mathcal{X}}, \mathcal{V}]$, where $\mathcal{C}_l^{(j)}$ is the subset of labels whose samples originate from component j. The second implication of equation 3.1 is that, conditioned on knowledge of the results from step 1 in the stochastic data generation ($\bar{\mathcal{X}}$ and $\mathcal{V}$), each component's label subset $\mathcal{C}_l^{(j)}$ forms a Markov random field (Li, 2009); it forms a conditional Markov random field, defined over the subset of classes that 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; if $V_{j|i} = 1$, the neighborhood for C_i is $\{C_{i'}, i' \in \mathcal{I}_l : V_{j|i'} = 1, i' \neq i\}$. Moreover, while it is not obvious in what precise form to express a component's MRF-based joint label subset distribution $P[\mathcal{C}_l^{(j)}|\bar{\mathcal{X}}, \mathcal{V}]$ consistent with the conditional distributions, equation 3.1, such a joint distribution is guaranteed to exist so long as a strict positivity condition holds on the conditionals, equation 3.1 (Besag, 1974). Inspecting equation 3.1, one can see strict positivity of the conditionals indeed holds for the subset of classes belonging to the given component (which is the same as the subset of classes over which the component's MRF is defined). Thus, the existence of $P[\mathcal{C}_l^{(j)}|\bar{\mathcal{X}}, \mathcal{V}]$ is guaranteed. Accordingly, so too the joint label distribution $P[\mathcal{C}_l|\bar{\mathcal{X}}, \mathcal{V}] = \prod_{j=1}^{L} P[\mathcal{C}_l^{(j)}|\bar{\mathcal{X}}, \mathcal{V}]$, is guaranteed to exist.

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

1. Generate $\bar{\mathcal{X}}$ (and, in the process, $\mathcal{V}$) by independently randomly generating each sample according to a standard finite mixture density. Note that the samples in $\bar{\mathcal{X}}$ can be generated in any order.
2. Given the results from step (1), generate the class labels $\{c_{N_u+1}, \ldots, c_{N_u+N_l}\}$ by sampling from the joint label distribution $P[C_{N_u+1}, \ldots, C_{N_u+N_l}|\bar{\mathcal{X}}, \mathcal{V}]$.

3.2 Pseudo-Likelihood Function. Ideally, we would like to develop a strict EM algorithm, choosing our model's parameters to maximize the joint data likelihood $\mathcal{L}(\bar{\mathcal{X}}, \mathcal{C}_l|\Theta) = \mathcal{L}(\mathcal{C}_l|\bar{\mathcal{X}}; \Theta)\mathcal{L}(\bar{\mathcal{X}}|\Theta)$. However, in section 3.1, we showed that even if we assume knowledge of the hidden data $\mathcal{V}$, $\mathcal{L}(\mathcal{C}_l|\bar{\mathcal{X}}, \mathcal{V}; \Theta)$ 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 the denominator, which sums over all possible joint realizations. In our case, the problem is even worse, because although we know it factorizes as a product over terms associated with each mixture component, the joint distribution $\mathcal{L}(\mathcal{C}_l|\bar{\mathcal{X}}, \mathcal{V}; \Theta)$ is not known; our MRFs are defined only by the conditional distributions 3.1. One standard strategy for handling intractabilities when performing learning or inference involving MRFs is to

work with the pseudo-likelihood function (Besag, 1986; Li, 2009) rather than the true joint likelihood. For a collection of statistically dependent random variables $\{A_i\}$, the pseudo-likelihood employs the approximation $\tilde{P}[A_1, \ldots, A_N] = \prod_{i=1}^{N} P[A_i | \{A_m, m \neq i\}]$. In our case, the pseudo-likelihood approximation is

$$\tilde{P}[\mathcal{C}_l | \bar{\mathcal{X}}, \mathcal{V}] = \prod_{i=N_u+1}^{N_u+N_l} P[\mathsf{C}_i = c_i | \{c_{i'}, i' \in \mathcal{I}_l - i\}, \bar{\mathcal{X}}, \mathcal{V}]. \tag{3.2}$$

Note, conveniently, that the required conditional probabilities in this equation are precisely those given in equation 3.1. In what follows, we develop a generalized EM algorithm (Fessler & Hero, 1994; Meng & Dyk, 1997; Graham & Miller, 2006), choosing our model's parameters to maximize the data pseudo-log-likelihood.

3.3 Complete Data Pseudo-Log-Likelihood. Based on our model's stochastic data generation, it is natural to choose the hidden data within the EM framework (Dempster et al., 1977) as $\{V_{j|i}\}$. Based on these quantities, the complete data pseudo-log-likelihood function for our model is

$$\begin{aligned} \log \tilde{\mathcal{L}}_c = & \sum_{i=1}^{N_u+N_l} \sum_{j=1}^{L} V_{j|i} \log(\alpha_j f_{\underline{X}|j}(\underline{x}_i | \theta_j)) \\ & + \sum_{i=N_u+1}^{N_u+N_l} \sum_{j=1}^{L} V_{j|i} \log \left[\frac{\sum_{m=N_u+1:c_m=c_i, m\neq i}^{N_u+N_l} V_{j|m} \exp(-a_j ||\underline{x}_m - \underline{x}_i||^2)}{\sum_{m=N_u+1, m\neq i}^{N_u+N_l} V_{j|m} \exp(-a_j ||\underline{x}_m - \underline{x}_i||^2)} \right]. \end{aligned} \tag{3.3}$$

3.4 E-Step. If we treat all the hidden data as random variables, it is unfortunately intractable to evaluate $E[\log \tilde{\mathcal{L}}_c | \mathcal{X}; \Theta]$ because the second sum in eqution 3.3 is a complicated nonlinear function of all variables in the set $\{\{V_{j|i} \quad \forall j\}, i = N_u + 1, \ldots, N_u + N_l\}$. However, it is possible to evaluate $E[\log \tilde{\mathcal{L}}_c | \mathcal{X}; \Theta]$ if we treat these hidden variables as unknown deterministic variables, $\{\{v_{j|i} \quad \forall j\}, i = N_u + 1, \ldots, N_u + N_l\}$, rather than as random variables. There is precedence for such treatment in proposals to maximize a mixture model's complete data log-likelihood (John, 1970; Biernacki, Celeux, & Govaert, 2008), as well as from K-means clustering, which also effectively treats data assignments to clusters as binary (0-1) variables to optimize.[3] In our case, such treatment greatly assists the development of

[3] 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 isotropic components and common covariance, $\sigma^2 I$, for all components.

a tractable learning procedure. In particular, assuming these hidden variables are unknown deterministic parameters, $E[\log \tilde{\mathcal{L}}_c|\mathcal{X};\Theta]$ can be directly evaluated based on the standard mixture component posteriors,

$$E[V_{j|i}|\mathcal{X};\Theta] = P[M_i = j|\mathcal{X};\Theta] = P[M_i = j|\underline{x}_i;\Theta] = \frac{\alpha_j f_{\underline{X}|j}(\underline{x}_i|\theta_j)}{\sum_{k=1}^{L}\alpha_k f_{\underline{X}|k}(\underline{x}_i|\theta_k)}, \quad i = 1,\ldots,N_u. \tag{3.4}$$

Furthermore, again by assuming the labeled data assignments to components are unknown deterministic variables (which need to be estimated), the associated incomplete data pseudo-log-likelihood is

$$\log \tilde{\mathcal{L}}_{\text{inc}} = \sum_{i=1}^{N_u}\log\Big(\sum_{j=1}^{L}\alpha_j f_{\underline{X}|j}(\underline{x}_i|\theta_j)\Big) + \sum_{i=N_u+1}^{N_u+N_l}\sum_{j=1}^{L} v_{j|i}\log(\alpha_j f_{\underline{X}|j}(\underline{x}_i|\theta_j)) + \sum_{i=N_u+1}^{N_u+N_l}\sum_{j=1}^{L} v_{j|i}\log\left[\frac{\sum_{m=N_u+1:c_m=c_i,m\neq i}^{N_u+N_l} v_{j|m}\exp(-a_j||\underline{x}_m-\underline{x}_i||^2)}{\sum_{m=N_u+1:m\neq i}^{N_u+N_l} v_{j|m}\exp(-a_j||\underline{x}_m-\underline{x}_i||^2)}\right]. \tag{3.5}$$

Note that what makes this likelihood "incomplete" (Dempster et al., 1977) is the fact that the $\{V_{j|i}, i = 1,\ldots,N_u\}$ are not known and hence do not appear in equation 3.5; they remain hidden data random variables. Equation 3.5 is the function over which our GEM algorithm (fully specified below) monotonically ascends. Note also that this pseudo-log-likelihood function possesses a data exchangeability property—the pseudo-log-likelihood does not depend on the order in which samples are generated—with respect to the samples in $\tilde{\mathcal{X}}$ and, separately, with respect to the class labels in $\mathcal{C}_l$. The model requires only that the labels be generated after first producing $\tilde{\mathcal{X}}$.

3.5 Generalized EM Algorithm. Consider the new parameter set $\Theta = \{\{\theta_j\},\{\alpha_j\},\{a_j\},\{\{v_{j|i}\}, i = N_u+1,\ldots,N_u+N_l\}\}$, which treats the labeled data sample assignments to components as additional unknown variables to be estimated. Here, we specify a generalized EM (GEM) algorithm (Fessler & Hero, 1994; Meng & Dyk, 1997; Graham & Miller, 2006) for choosing these parameters to locally maximize the above incomplete data pseudo-log-likelihood. A GEM algorithm, rather than a pure EM algorithm, is required for the current model because of the complicated dependence in both $\log \tilde{\mathcal{L}}_{\text{inc}}$ and $E[\log \tilde{\mathcal{L}}_c|\mathcal{X};\Theta]$ on the $\{v_{j|i}\}$ and $\{a_j\}$, which precludes a closed form M-step for these parameters. Our GEM algorithm consists of alternating E-steps, equation 3.4, and generalized M-steps, and gives parameter updates that are non decreasing in $\log \tilde{\mathcal{L}}_{\text{inc}}$. The E-step was

specified above. 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 that comprises a generalized M-step, is nondecreasing in the incomplete data pseudo-log-likelihood. Below, we specify the optimization steps comprising our generalized M-step, assuming gaussian components $\mathcal{N}(\underline{\mu}_j, \Sigma_j)$ for concreteness.

3.5.1 Generalized M-Step

Update of $\{\alpha_j\}$ *and* $\{\theta_j\}$. Based on the E-step quantities , it is straightforward to derive closed-form M-step updates for $\{\alpha_j\}$ and $\{\theta_j = (\underline{\mu}_j, \Sigma_j)\}$ which globally maximize $E[\log \tilde{\mathcal{L}}_c | \mathcal{X}; \Theta^{(t)}]$ in determining the new parameter values $\{\alpha_j^{(t+1)}\}$ and $\{\theta_j^{(t+1)}\}$, given all other parameters held fixed. These updates are given by

$$\alpha_j^{(t+1)} = \frac{\sum_{i=1}^{N_u} P\left[M_i = j | \underline{x}_i; \Theta^{(t)}\right] + \sum_{i=N_u+1}^{N_u+N_l} v_{j|i}^{(t)}}{N_u + N_l}, \quad \forall j, \tag{3.6}$$

$$\underline{\mu}_j^{(t+1)} = \frac{\sum_{i=1}^{N_u} P\left[M_i = j | \underline{x}_i; \Theta^{(t)}\right] \underline{x}_i + \sum_{i=N_u+1}^{N_u+N_l} v_{j|i}^{(t)} \underline{x}_i}{\alpha_j^{(t+1)}(N_u + N_l)}, \quad \forall j, \tag{3.7}$$

$$\begin{aligned}\sum_j^{(t+1)} = &\frac{\sum_{i=1}^{N_u} P\left[M_i = j | \underline{x}_i; \Theta^{(t)}\right] (\underline{x}_i - \underline{\mu}_j^{(t+1)})(\underline{x}_i - \underline{\mu}_j^{(t+1)})^T}{\alpha_j^{(t+1)}(N_u + N_l)} \\ &+ \frac{\sum_{i=N_u+1}^{N_u+N_l} v_{j|i}^{(t)} (\underline{x}_i - \underline{\mu}_j^{(t+1)})(\underline{x}_i - \underline{\mu}_j^{(t+1)})^T}{\alpha_j^{(t+1)}(N_u + N_l)}, \quad \forall j.\end{aligned} \tag{3.8}$$

Update of Scale Parameters, $\{a_j\}$. Next, given all other parameters fixed (at $\{\{\alpha_j^{(t+1)}\}, \{\theta_j^{(t+1)}\}, \{v_{j|i}^{(t)}\}\}$), we need to optimize over the $\{a_j\}$ parameters, yielding $\{a_j^{(t+1)}\}$. It is not possible to find closed- form M-step updates for these parameters. However, they can be optimized by gradient ascent, applied to the objective function $\log \tilde{\mathcal{L}}_{\text{inc}}$ in equation 3.5.[4]

Update of the $\{v_{j|i}\}$ *Variables.* Given the fixed set of parameters, $\{\{\alpha_j^{(t+1)}\}, \{\theta_j^{(t+1)}\}, \{a_j^{(t+1)}\}\}$, we optimize over the labeled data assignments

[4]One could choose $\{a_j\}$ to maximize either $E[\log \tilde{\mathcal{L}}_c | \mathcal{X}; \Theta]$ or $\log \tilde{\mathcal{L}}_{\text{inc}}$. Both choices will lead to a monotonic increase in $\log \tilde{\mathcal{L}}_{\text{inc}}$, which is the objective of a GEM step. However, it is plausible that directly maximizing over the target objective function ($\log \tilde{\mathcal{L}}_{\text{inc}}$), rather than the EM algorithm's lower bound ($E[\log \tilde{\mathcal{L}}_c | \mathcal{X}; \Theta]$), should yield faster algorithm convergence to a local maximum of $\log \tilde{\mathcal{L}}_{\text{inc}}$.

to components. For these variables, there are two possibilities. If $\{\{v_{j|i}\}, i = N_u + 1, \ldots, N_u + N_l\}$ are treated as binary variables, each belonging to $\{0, 1\}$, then they can be optimized cyclically, visiting one sample (i) at a step, optimizing sample i's component assignment given the assignments of all other labeled samples held fixed and given all other parameters in Θ held fixed, and then repeating for sample $(i + 1)$. The labeled sample assignments to components are iteratively reoptimized in this fashion until a complete sweep over the assignments for all labeled samples leads to no further changes. At this point, a local maximum has been achieved. At each step of this cyclical optimization, we assign the labeled sample $(\underline{x}_i, c_i)$ to the component such that the largest pseudo-log-likelihood, $\log \tilde{\mathcal{L}}_{\text{inc}}$, is achieved.

This cyclical optimization is akin to discrete optimization approaches previously taken in Besag (1986) and Graham & Miller (2006). Alternatively, if we let $v_{j|i} \in [0, 1]$, we can without loss of representation power express $v_{j|i} = \frac{e^{v_{j,i}}}{\sum_{k=1}^{L} e^{v_{k,i}}}$, that is, using *softmax functions* (Bridle, 1990; Miller & Browning, 2003). In this case, we can jointly optimize the real-valued parameters $\{v_{j,i}, \forall j, i = N_u + 1, \ldots, N_u + N_l\}$ via gradient ascent on equation 3.5, again with all other parameters in Θ held fixed. In this work, we consider only the discrete optimization strategy.

In summary, our algorithm for learning this new semisupervised mixture model performs a sequence of iterations (monotonically ascending in equation 3.5), consisting of: E-step, evaluating $P[M_i = j|\underline{x}_i; \Theta]$, $j = 1, \ldots, L$, $i = 1, \ldots, N_u$; generalized M-step updates for $\{\alpha_j\}$ and $\{\theta_j\}$, given $\{a_j\}$ and $\{v_{j|i}\}$ held fixed; gradient-ascent update of $\{a_j\}$ given all other parameters in Θ held fixed; and either cyclical optimization of binary variables $\{v_{j|i}\}$ or gradient-ascent update of the $\{v_{i,j}\}$, given all other parameters in Θ held fixed. In practice, these iterations are applied until the gain (or relative gain) in $\log \tilde{\mathcal{L}}_{\text{inc}}$ from one iteration to the next falls below a preset convergence threshold.

3.6 Class Decision Making. Consider a new sample, $\underline{x}$, that needs to be classified according to our model. It is useful to denote this sample by $\underline{x} \equiv \underline{x}_{N_u+N_l+1}$ and also to define $n \equiv N_u + N_l + 1$.[5] To perform inductive inference on $\underline{x}_n$, we would like to evaluate the posterior probability $P[C_n = c|\mathcal{X}, \underline{x}_n, \Theta]$, which conditions on all available information, and choose the class that maximizes this posterior. Typically, there is a unique maximum a posteriori decision rule. However, for our NFGL model, we will derive two distinct posterior rules. Fundamentally, we derive two different rules because there are two different subsets of data generated by our NFGL model—the labeled and unlabeled subsets—with a different stochastic data generation mechanism for each of these types of data. Accordingly, there is

[5]This allows the equations to be written in a compact form.

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 the unlabeled data subset.

First, suppose that the test sample is from the labeled subset, albeit with its label unknown. Essentially, in this case, $\underline{x}_n$ is treated as being part of the labeled sample graph, albeit with both the label and the mixture component of origin for this sample unknown. Exact inference is not possible because of intractabilities associated with the MRF defined on the labeled samples. However, we can define an approximate posterior probability, based on the pseudo-likelihood, as follows:

$$P[C_n = c|\mathcal{X}, \underline{x}_n; \Theta] = \frac{\tilde{\mathcal{L}}\left[\mathcal{C}_l, C_n = c|\bar{\mathcal{X}}, \underline{x}_n, \Theta\right]}{\sum_{k=1}^{K} \tilde{\mathcal{L}}\left[\mathcal{C}_l, C_n = k|\bar{\mathcal{X}}, \underline{x}_n; \Theta\right]}. \tag{3.9}$$

Furthermore, we can simply derive that

$$\tilde{\mathcal{L}}\left[\mathcal{C}_l, C_n = c|\bar{\mathcal{X}}, \underline{x}_n; \Theta\right] = \sum_{j=1}^{L} \tilde{\mathcal{L}}\left[\mathcal{C}_l, C_n = c, M_n = j|\bar{\mathcal{X}}, \underline{x}_n; \Theta\right]$$

$$= \sum_{j=1}^{L} \tilde{\mathcal{L}}\left[\mathcal{C}_l, C_n = c|M_n = j, \bar{\mathcal{X}}, \underline{x}_n; \Theta\right] P[M_n = j|\bar{\mathcal{X}}, \underline{x}_n; \Theta].$$

Now, given the stochastic data generation in section 3.1, where each $\underline{x} \in \mathcal{X}_u$ is independently generated, we have that $P[M_n = j|\bar{\mathcal{X}}, \underline{x}_n; \Theta] = P[M_n = j|\underline{x}_n; \Theta]$. Thus, we finally get

$$\tilde{\mathcal{L}}\left[\mathcal{C}_l, C_n = c|\bar{\mathcal{X}}, \underline{x}_n; \Theta\right]$$

$$= \sum_{j=1}^{L} P[M_n = j|\underline{x}_n; \Theta]$$

$$\times \prod_{i=N_u+1}^{N_u+N_l+1} \sum_{j'=1}^{L} v_{j'|i} \left(\frac{\sum_{m=N_u+1:c_m=c_i, m\neq i}^{N_u+N_l+1} v_{j'|m} \exp(-a_{j'}||\underline{x}_m - \underline{x}_i||^2)}{\sum_{m=N_u+1, m\neq i}^{N_u+N_l+1} v_{j'|m} \exp(-a_{j'}||\underline{x}_m - \underline{x}_i||^2)} \right). \tag{3.10}$$

Note that in the above final expression, we denote c by $c_{N_u+N_l+1}$ in order to write the posterior in a compact form. Note also that when $M_n = j$, the associated component indicator variable, as used in equation 3.10, is given by a discrete delta function, as follows: $v_{j'|n} = \delta[j' - j] \quad \forall j'$. Essentially this is equivalent to the statement $v_{j'|n} = \delta[j' - M_n]$. Finally, note that the above

derivation is exact, but the resulting posterior is approximate because we are defining it based on the pseudo-likelihood function rather than on the true likelihood function.

Next, consider the case where we treat the test sample as belonging to the unlabeled data subset. What this means is that in this case, we are assuming that the test sample is generated in the same way as any other sample from the unlabeled data subset (i.i.d., according to the mixture density). It is 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, equation 3.1, an exact class posterior probability exists for the unlabeled samples. Next we derive this true class posterior. Since $\mathcal{C}_l$ is statistically independent of $\mathcal{X}_u$ and M_n is statistically independent of $\mathcal{X}_l$, we have in this case that

$$\begin{aligned} P[C_n = c|\mathcal{X}, \underline{x}_n; \Theta] &= P[C_n = c|\mathcal{X}_l, \underline{x}_n; \Theta] \\ &= \sum_{j=1}^{L} P[C_n = c|M_n = j, \mathcal{X}_l, \underline{x}_n; \Theta]\ P[M_n = j|\mathcal{X}_l, \underline{x}_n; \Theta] \\ &= \sum_{j=1}^{L} P[C_n = c|M_n = j, \mathcal{X}_l, \underline{x}_n; \Theta]\ P[M_n = j|\underline{x}_n; \Theta] \\ &= \sum_{j=1}^{L} P[M_n = j|\underline{x}_n; \Theta] \frac{\sum_{m=N_u+1:c_m=c}^{N_u+N_l} v_{j|m} \exp\left(-a_j||\underline{x}_m - \underline{x}_n||^2\right)}{\sum_{m=N_u+1}^{N_u+N_l} v_{j|m} \exp\left(-a_j||\underline{x}_m - \underline{x}_n||^2\right)}. \end{aligned} \tag{3.11}$$

In summary, fundamentally there are two natural inference rules to introduce for NFGL because one can interpret a test sample $\underline{x}$ as being generated according to either the labeled subset or the unlabeled subset. As will be seen, although it may be counterintuitive, there are reasons for preferring the latter posterior, equation 3.11, in practice.

4 A Parametric Finely Grained Labeling Model

4.1 Problems with NFGL at Very Low Labeled Fractions. The NFGL model is a sound, finely grained framework when the labeled sample fraction is not "too low." However, a difficulty with this model can be seen by considering several scenarios that will in fact occur in practice when the amount of labeled training data available is quite spartan. First, consider the case where there is a single labeled sample generated by a given component j, denoted i_1. By inspecting equation 3.1, 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 $P[C_{i_1} = c|\{c_i, i \in \mathcal{I}_l - i_1\}, \bar{\mathcal{X}}, \mathcal{V}]$ evaluates to zero divided by zero, for all classes, including the true class $c = c_{i_1}$. That is, equation 3.1 is degenerate when there is only a single labeled sample generated by a given component. This degeneracy can be resolved in practice by using either the MOE model within this component or hard-assigning this component to one of the classes (specializing the component-specific posterior to a hard-MOE model (Nigam et al., 2000) within this component). However, next consider a related but distinct scenario wherein, within component j, there are two labeled samples, one from class 1 (whose index is denoted i_1) and one from class 2 (i_2). Again by inspecting equation 3.1, we find that $P[C_{i_1} = 1|\{c_i, i \in \mathcal{I}_l - i_1\}, \bar{\mathcal{X}}, \mathcal{V}] = 0$ and, likewise, $P[C_{i_2} = 2|\{c_i, i \in \mathcal{I}_l - i_2\}, \bar{\mathcal{X}}, \mathcal{V}] = 0$. While the posterior is well defined in this case, the solution is unsatisfactory—there should be a very *large* probability rather than zero probability of generating class 1 for labeled sample i_1 and generating class 2 for labeled sample i_2. Moreover, similar results will occur whenever, within a given component, there is a single labeled example from a given class. While one might argue that these are extreme cases, involving very few labeled samples within a given component, this in fact will be a fairly common scenario when the labeled fraction is very low. Moreover, a low labeled fraction is the reason that semisupervised learning (rather than supervised learning) may be needed in the first place.

In this section, we propose an alternative to NFGL's (nonparametric) randomized nearest-neighbor model, equation 3.1, that does not suffer from degeneracy/or poor solutions when dealing with labeled sample sparsity. These problems are avoided by invoking a parametric within-component posterior. There are many possible parametric posterior models that could be chosen. In this section, although we recognize the generality of our ideas, both 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 posterior:

$$P[C = c|M_i = j, \underline{x}_i; \Theta] = \frac{\sum_{l=1:m_l^{(j)}=c}^{N_j} \exp(-a_j||\underline{x}_i - \underline{s}_l^{(j)}||^2)}{\sum_{l=1}^{N_j} \exp(-a_j||\underline{x}_i - \underline{s}_l^{(j)}||^2)}, \ \forall c. \qquad (4.1)$$

Here, component j has N_j prototypes $\underline{s}_l^{(j)}, l = 1, \ldots, N_j$, where the index mapping $m_l^{(j)} \in \{1, 2, \ldots, N_c\}$ indicates the class to which prototype $\underline{s}_l^{(j)}$ belongs. Note that $\underline{s}_l^{(j)}$ and a_j are both (adjustable) model parameters. As $a_j \to \infty$, equation 4.1 converges to a nearest prototype decision rule (Kohonen, Barna, & Chrisley, 1988; Sinkkonen, Kaski, & Nikkilä, 2002)

within component j. Note that, unlike equation 3.1, equation 4.1 is always well defined so long as $N_j \geq 1$ and will not assign zero probability to a class that possesses (only) a single prototype within a given component. However, unlike equation 3.1, equation 4.1 does, in general, require model order selection, to choose the number of prototypes, N_j, their locations, and their class assignments $\{m_l^{(j)}, l = 1, \ldots, N_j\}$. In section 4.5, we devise a methodology for judiciously learning these choices.

4.2 Stochastic Data Generation. Associated with our proposed posterior, equation 4.1, there is a natural, simple stochastic data generation mechanism, as follows:

1. Independently, for each $\underline{x}_i \in \mathcal{X}_u$; randomly select a mixture component, $M_i = j$, according to $\{\alpha_l\}$, and then randomly generate $\underline{x}_i$ according to $f_{\underline{X}|j}(\underline{x}|\theta_j)$.
2. Independently, for each $(\underline{x}_i, c_i) \in \mathcal{X}_l$; randomly select a mixture component, $M_i = j$, according to $\{\alpha_l\}$, randomly generate $\underline{x}_i$ according to $f_{\underline{X}|j}(\underline{x}|\theta_j)$; and randomly select the class label c_i according to the component-conditional posterior $\{P[C = c|M_i = j, \underline{x}_i; \Theta]\}$ specified in equation 4.1. Clearly, this is a finely grained extension of the stochastic generation in Miller and Uyar (1997).

4.3 Incomplete Data Log Likelihood. The model parameters associated with the above stochastic data generation are $\Theta = \{\{\theta_j\}, \{\alpha_j\}, \{a_j\}, \{\underline{s}_l^{(j)}\}, \{m_l^{(j)}\}\}$. Given a model of fixed size, the incomplete data log likelihood, which we seek to maximize in learning Θ, is

$$\log \mathcal{L}_{\text{inc}} = \sum_{i=1}^{N_u} \log \left[\sum_{j=1}^{L} \alpha_j f_{\underline{X}|j}(\underline{x}_i|\theta_j) \right] + \sum_{i=N_u+1}^{N_u+N_l} \log \left[\sum_{j=1}^{L} \alpha_j f_{\underline{X}|j}(\underline{x}_i|\theta_j)\ P[C = c_i|M_i = j, \underline{x}_i; \Theta] \right]. \quad (4.2)$$

Note that this model possesses full sample exchangeability; data samples can be generated in any order.

4.4 Generalized EM Algorithm. Unlike the developments in section 3 where we maximize a pseudo-log-likelihood, one can derive a GEM algorithm for directly maximizing $\log \mathcal{L}_{\text{inc}}$ in equation 4.2. A GEM algorithm, rather than a pure EM algorithm, is required because of the complicated dependence of $\log \mathcal{L}_{\text{inc}}$ on the prototype vectors and scale parameters. The hidden datum that we will introduce is knowledge of the component of

origin for each sample $\{V_{j|i}, i = 1, \ldots, N_u + N_l\}$. One then obtains the following E-step and generalized M-step, alternately applied, to generate a sequence of parameter estimates that is increasing in $\log \mathcal{L}_{\text{inc}}$.

4.4.1 E-step. For unlabeled samples, we have

$$E[V_{j|i}|\mathcal{X}; \Theta] = P[M_i = j|\underline{x}_i; \Theta] = \frac{\alpha_j f_{\underline{X}|j}(\underline{x}_i|\theta_j)}{\sum_{k=1}^{L} \alpha_k f_{\underline{X}|k}(\underline{x}_i|\theta_k)}, \; i = 1, \ldots, N_u. \tag{4.3}$$

For labeled samples, we have

$$\begin{aligned} &E[V_{j|i}|\mathcal{X}; \Theta] = P[M_i = j|(\underline{x}_i, c_i); \Theta] \\ &= \frac{\alpha_j f_{\underline{X}|j}(\underline{x}_i|\theta_j)\, P[C = c_i|M_i = j, \underline{x}_i; \Theta]}{\sum_{k=1}^{L} \alpha_k f_{\underline{X}|k}(\underline{x}_i|\theta_k)\, P[C = c_i|M_i = k, \underline{x}_i; \Theta]}, \quad i = N_u + 1, \ldots, N_u + N_l. \end{aligned} \tag{4.4}$$

4.4.2 Generalized M-Step.
Update of $\{\alpha_j\}$ *and* $\{\theta_j\}$*:*

$$\alpha_j^{(t+1)} = \frac{\sum_{i=1}^{N_u} P[M_i = j|\underline{x}_i; \Theta^{(t)}] + \sum_{i=N_u+1}^{N_u+N_l} P[M_i = j|(\underline{x}_i, c_i); \Theta^{(t)}]}{N_u + N_l}, \forall j, \tag{4.5}$$

$$\underline{\mu}_j^{(t+1)} = \frac{\sum_{i=1}^{N_u} P[M_i = j|\underline{x}_i; \Theta^{(t)}]\underline{x}_i + \sum_{i=N_u+1}^{N_u+N_l} P[M_i = j|(\underline{x}_i, c_i); \Theta^{(t)}]\underline{x}_i}{\alpha_j^{(t+1)}(N_u + N_l)}, \forall j \tag{4.6}$$

$$\begin{aligned} \sum\nolimits_j^{(t+1)} = &\frac{\sum_{i=1}^{N_u} P[M_i = j|\underline{x}_i; \Theta^{(t)}](\underline{x}_i - \underline{\mu}_j^{(t+1)})(\underline{x}_i - \underline{\mu}_j^{(t+1)})^T}{\alpha_j^{(t+1)}(N_u + N_l)} \\ &+ \frac{\sum_{i=N_u+1}^{N_u+N_l} P[M_i = j|(\underline{x}_i, c_i); \Theta^{(t)}](\underline{x}_i - \underline{\mu}_j^{(t+1)})(\underline{x}_i - \underline{\mu}_j^{(t+1)})^T}{\alpha_j^{(t+1)}(N_u + N_l)}, \forall j \end{aligned} \tag{4.7}$$

Update of Prototype Vectors, $\{\{\underline{s}_l^{(j)}\} \; \forall j\}$. Next, given all the other parameters fixed, we optimize all the prototype vectors via gradient ascent applied

to $\log \mathcal{L}_{\text{inc}}$, since it is not possible to find closed-form M-step updates for these parameters.

Update of Scale Parameters, $\{a_j\}$. Finally, we optimize over the $\{a_j\}$ parameters, with all other parameters fixed at their updated values, to yield $\{a_j{}^{(t+1)}\}$. Similar to section 3.5, it is not possible to find closed-form M-step updates for these parameters. However, they can be optimized via gradient ascent, applied to $\log \mathcal{L}_{\text{inc}}$ in (15).

4.5 Overall Learning Strategy. The GEM algorithm 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 posterior model complexity (neither too few, nor too many prototypes) within each mixture component. This framework consists of the following steps:

4.5.1 Step 1: Learn an MOE Model for $\mathcal{X}$. We first learn a standard gaussian mixture model, using BIC to select the number of components. This is used as initialization for the EM algorithm in Miller and Uyar (1997), applied to learn an MOE model.

4.5.2 Step 2: Single Prototype Per Class Assignment. For each mixture component, j, we first allocate one prototype for every class k that satisfies $\beta_{k|j} > \epsilon$—for every class at least partially represented by at least one labeled sample within the component. The prototype for class k is intialized at the (MOE-posterior-weighted) centroid of all labeled samples from class k that fall within the component. 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 (Nigam et al., 2000): $m_1^{(j)} = k^*$. We then optimize all the (just initialized) prototype vectors via gradient ascent applied to the incomplete data log likelihood, equation 4.2, while keeping all other model parameters fixed at their current values.

4.5.3 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," that is, with all components j such that $P[M_i = j|(\underline{x}_i, c_i); \Theta] > \tau$. In other words, we trial-create an additional prototype $\underline{s}_{N_j+1}^{(j)} = \underline{x}_i, i = N_u + 1, \ldots, N_u + N_l$, where $m_{N_j+1}^{(j)} = c_i$, for all components j in the vicinity of $\underline{x}_i$ as specified above. For each candidate pair (i, j), we

evaluate the joint log likelihood, equation 4.2, that results when the candidate prototype is included in the model, and we identify the best candidate pair (i^*, j^*). To best evaluate this pair, we perform trial optimization of all the prototypes within component (j^*) (including this best candidate initialized at the labeled sample), along with adjustment of the scale parameter a_{j^*}. We then evaluate the resulting candidate new model's BIC cost (Schwarz, 1978), based on the joint log-likelihood, equation 4.2, and based on an accounting of the code length in bits required to specify all free parameters in the model, Θ (Schwarz, 1978). This code length is given by:

$$CC(L, \{N_j\}, \Theta) = \frac{(L-1)}{2} \log_2(N_u + N_l) + \frac{L|\theta_j|}{2} \log_2(N_u + N_l) + \frac{L}{2} \log_2 N_l + \left(\frac{d}{2} \log_2 N_l + \log_2 N_c + \log_2 L \right) \sum_{j=1}^{L} N_j.$$

Here, the first term specifies the component masses, the second term the parameters of all the mixture components, and the third term the scale parameters $\{a_j\}$ in the class posterior models. Also, $\frac{d}{2} \log_2 N_l$ bits are needed to specify each real-valued prototype vector, $\log_2 N_c$ bits are needed to specify the class of the prototype, and $\log_2 L$ bits are needed to specify the mixture component for which the prototype is used to model the within-component-class posterior. The trial-optimized candidate prototype is accepted into the model only if the new BIC cost (based on inclusion of the prototype) is lower than the current model's BIC cost.

The trial optimization of the prototype vectors $\{\underline{s}_l^{(j^*)}, l = 1, \ldots, N_{j^*} + 1\}$ is done by gradient ascent applied to the incomplete data log likelihood, equation 4.2, while keeping all other model parameters, including a_{j^*}, held fixed at their current values. Note that fixing a_{j^*} ensures soft (nonzero) 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 $\{\theta_j, \{\alpha_j\}\}$ 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) that "own" labeled samples from the same class. If a prototype for this class is added to component A (before component B), a reoptimization 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 (samples that should "rightfully" belong to component B). Fixing all the component parameters in this step prevents this sequential source of potential learning bias.

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 $\{\underline{s}_l^{(j^*)}\}$ fixed as the subcomponent means, and with unknown common covariance matrix $\sigma_{j^*}^2\mathbf{I}$. We perform the EM algorithm on the (weighted) samples belonging to component j^* in order to estimate the variance $\sigma_{j^*}^2$, where the sample weights w_{ij^*} are given by the mixture component posteriors, equations 4.3 and 4.4. Based on this mixture model within component j^*, we can calculate the weighted entropy of the (gaussian mixture) subcomponent posterior $\{P_{l|i}^{j^*}\}$, over all data points using

$$\bar{H} = \sum_{i=1}^{N_u+N_l} w_{ij^*} \sum_{l=1}^{N_{j^*}} P_{l|i}^{j^*} \log P_{l|i}^{j^*}.$$

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

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

where

$$\hat{P}_{l|i}^{j^*} = \frac{\exp(-a_{j^*}||\underline{x}_i - \underline{s}_l^{(j^*)}||^2)}{\sum_{l'=1}^{N_{j^*}} \exp(-a_{j^*}||\underline{x}_i - \underline{s}_{l'}^{(j)}||^2)}, l = 1, \ldots, N_{j^*}, i = 1, \ldots, N_u + N_l.$$

Finally, we use Newton's method to find a positive a_{j^*} such that $H(a_{j^*}) = \bar{H}$. 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.

4.5.4 Step 4: Jointly Optimize Θ. Finally, after sequential within-component class posterior model growing is complete, we reoptimize the

entire set of model parameters, Θ, based on the EM algorithm. In principle, this optimization includes (discrete) optimization of the prototype-to-class assignments $\{m_j^{(j)}\}$, starting from the initial assignments specified in steps 2 and 3. However, we did not perform this discrete optimization in this work.

4.6 Class Decision Making. Class inference for the PFGL model is very similar to the second inference rule for NFGL, given in equation 4.1. Specifically, the class posterior for a new sample $\underline{x}$ is

$$P[C = c|\underline{x}; \Theta] = \sum_{j=1}^{L} P[M = j|\underline{x}; \Theta]\ P[C = c|M = j, \underline{x}; \Theta], \tag{4.8}$$

where $P[C = c|M = j, \underline{x}; \Theta]$ is the randomized nearest-prototype rule given in equation 4.1.

5 Relationship to Some Previous Methods

Our methods "propagate" class label information from labeled to unlabeled data for the case where a mixture distribution assumption for the feature vector is reasonable, but where it is in fact unreasonable to assume that the mixture components 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 Szummer and Jaakkola (2002) and Zhu and Ghahramani (2002). Many subsequent graph-based semisupervised methods are discussed in Zhu (2008). Most of these methods are particularly motivated by problems where the classes lie on an underlying manifold (which can be well captured by a global graph defined on all the data samples). By contrast, our approaches are motivated by domains where the feature data are well modeled by a mixture model (but with within-component class heterogeneity). Our NFGL approach effectively defines a fully connected subgraph (a particular MRF) over the labels for samples that are generated by the same mixture component (i.e., there is a separate, fully connected graph for each mixture component, with the nodes corresponding to the labeled samples that belong to that component). The method in Zhu and Lafferty (2005) also combines graph-based semisupervised 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 inference (decision making) on the (global) graph defined on all the data samples. Second, they have a global graph, while we effectively have a separate graph for each component or cluster. Third, their approach is motivated (and best suited for) domains with data manifold structure, while ours is best suited for domains well modeled by mixture distributions. The

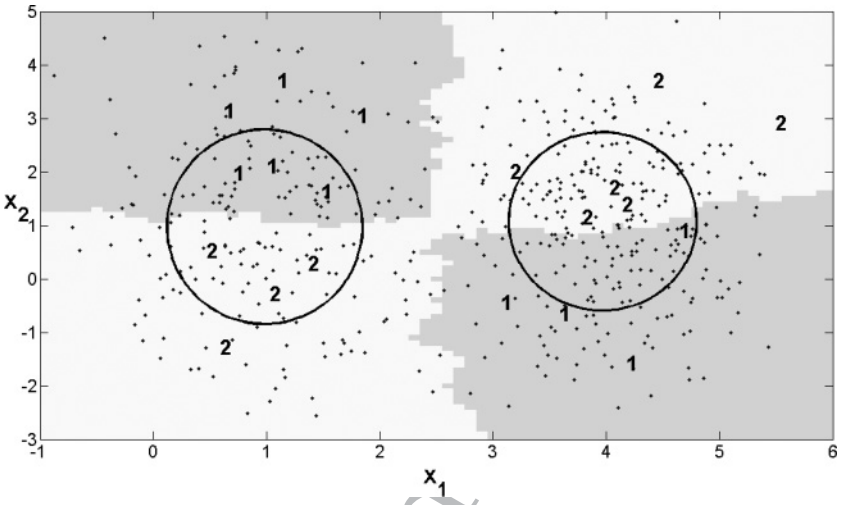

Figure 5: A two-dimensional semisupervised example with two ground-truth gaussian components and an XOR-like class structure. Note that the NFGL solution captures both the ground-truth mixture components and the XOR class structure in the data.

approach in Zhao and Miller (2005) 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. Of course, the current work builds on the previous methods (Miller & Uyar, 1997; Nigam et al., 2000; Shahshahani & Landgrebe, 1994) and extends them to achieve finely grained within-component class labeling.

6 Experimental Results

6.1 An Illustrative Example for NFGL. Figure 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 Nigam et al. (2000) will be necessarily poor since this method is restricted to learning class-pure components. The method in Miller and Uyar (1997) can learn the ground-truth mixture components, but only a crude class proportions model within each mixture component. Figure 5 shows the BIC-selected two-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 decision boundary learned by NFGL.

Table 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

6.2 Experimental Protocol. We evaluated the classification accuracies of our nonparametric finely grained labeling method (NFGL, section 3) and parametric finely grained labeling method (PFGL, section 4), in comparison with a number of conventional semisupervised and supervised methods. For the NFGL method, we have found (unsurprisingly) that the simpler, second inference rule, equation 3.11, which, unlike equation 3.9, does not involve products over all labeled samples, has much less difficulty with degeneracies and zero probabilities at low labeled fractions (as discussed in section 4) than equation 3.9. Accordingly, we have used equation 3.11 in all of our experiments. The methods we have compared with include the following semisupervised methods: the mixture of experts classifier (MOE) (Miller & Uyar, 1997), the specialization of MOE with hard component-to-class assignments (MOE-hard) (Nigam et al., 2000), and within-component nearest neighbor (WC-NN) (section 2). We also evaluated MOE-hard-plus, the MOE-hard method, but with the number of components increased until MOE-hard's BIC cost first exceeds PFGL's BIC cost.[6] Note that this choice gives a roughly fair model complexity comparison between MOE-hard-plus and PFGL.

We also compared with the following purely supervised methods: linear and gaussian kernel support vector machines (SVMs) and K-nearest-neighbor (KNN) classification.

We used eight real-world data sets from the UC Irvine machine learning repository (Asuncion & Newman, 2007), as summarized in Table 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 UCI data sets. For some of the data sets, there are categorical-

[6]For MOE-hard-plus, the code length 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 requires $\log_2 N_c$ bits for each component.

valued features. We did not use these in the experiments in order to make the mixture modeling less complicated.

We learned models using a training set and evaluated performance on a separate test set. For Image Segmentation, we used the training/test partition specified in the UCI data description. For all the other UCI sets, there is a single data set. For these, we performed a random 50% split into training and test subsets. The same split was used for all methods. In each experiment with training/test sets fixed, we varied the labeled training set percentage from 2% 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 fraction. For a given labeled percentage, we performed 10 random selections of labeled or unlabeled training subsets, with the same labeled or unlabeled training subsets used for all the methods. The semisupervised methods used all the data (including the unlabeled training and test samples) for learning, while the supervised methods (SVMs and KNN) used only the labeled training subset for learning. All the methods then made decisions on the test split. Results were averaged over the 10 "trials," yielding an average test set classification error rate for a given labeled training set fraction.

6.2.1 Parameter Initialization and Implementation Details. For a standard gaussian mixture model (GMM), we initialized parameters for the EM algorithm by first running K-means (with means randomly initialized to training samples). The GMM learned via EM was then used to initialize the GMM parameters $\{\{\theta_j\}, \{\alpha_j\}\}$ used by each of the semisupervised methods. We considered using full covariance matrices for the gaussian components. However, on a number of the data sets, we observed that 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 used only diagonal covariances, on all the data sets. We acknowledge that overall better accuracy may be achieved if use of diagonal versus full covariances is decided separately for each given data set or domain. The same set of initial GMM parameters was used for all semisupervised learning methods. For NFGL, we used cyclical optimization to learn the binary $\{v_{j|i}\}$ variables. We first hard-assigned each labeled sample to a mixture component using the maximum a posteriori (MAP) rule applied to the standard GMM component posterior with the initial set of parameters and set $v_{j|i}$ to 1 for the MAP component and to 0 for all other components. The scale parameters $\{a_j\}$ in NFGL were all initialized to the reciprocal of the average squared distance between all pairs of labeled samples. For the MOE method, the parameters $\{\beta_{c|j} = P[C = c|M = j]\}$ were initialized as uniform over all classes.

For NFGL, when a mixture component has either no labeled samples or just a single labeled sample assigned to it, $\sum_{i=N_u+1}^{N_u+N_l} v_{j|i} \leq 1$, we use the

MOE component-conditional class pmf in place of the randomized nearest-neighbor class posterior, equation 3.1, which is undefined (degenerate) in this case. Also, when zero probabilities occur in NFGL learning or inference, we replace them by a very small constant for numerical stability. For NFGL, an undesirable phenomenon that can occur during unconstrained cyclical optimization of the $\{v_{j|i}\}$ (component switch) variables is as follows. As discussed in section 4.1, when a mixture component has singleton-labeled samples from one or more classes, the randomized nearest-neighbor class posterior, equation 3.1, evaluates to zero probability for those classes. During the cyclical optimization of the component switch variables, such components could try to "steal" labeled samples from other components in order to have nonzero 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 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 k assigned a prototype within component j should have $\beta_{k|j} > \frac{1}{2\hat{N}_j}$, where $\hat{N}_j$ is an estimate (easily obtained) of the number of labeled samples belonging to the component; that is, at least "half a labeled sample" from class k should belong to component j if class k 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) (Chang & Lin, 2001). 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 fraction and training set realization, we chose these hyperparameters using a cross-validation procedure, as next detailed. For the linear SVM, we varied C over the range $[0.1, 10,000]$ and performed 10-fold cross-validation on the labeled training subset (actually the number of folds was chosen as $\min\{10, N_{tr}\}$, where N_{tr} is the size of the labeled training set), selecting the C value from a piecewise uniform grid of 334 candidate values. According to the suggestions in the LIBSVM manual (Chang & Lin, 2001), 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 $1000 \leq C \leq 10,000$: $s = 50$. 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 $0.001 \leq \gamma < 0.01$:

Table 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

$s_\gamma = 0.002$, 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 two classes, we applied one-against-one multiclass classification as implemented in the LIBSVM software.

The hyperparameter K in the KNN classifier was chosen using 10-fold cross-validation applied to the training set. The range of K was set to $\{2, \ldots, 8\}$ for data sets larger than 400 samples, and $\{1, \ldots, 5\}$ for smaller data sets. The WC-NN classifier was implemented as described in section 2.

6.2.2 Model Order Selection. On a given data set, for all the mixture-based methods (excepting MOE-hard-plus), the same model order (number of components) was used, chosen based on the Bayesian information criterion (BIC) (Schwarz, 1978), applied to the initially learned standard gaussian mixture model, which was learned based on the training and test sets combined. For MOE-hard, however, one should not allow BIC to select the number of components less than the number of classes (otherwise, some classes will not be represented). 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 gives the number of components selected by BIC and the range of the number of components used by MOE-hard-plus for all the data sets. Table 3 gives the parameter k for the KNN classifier, averaged over the 10 random trials for the different labeled fractions 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 class labels). For example, MOE uses a more complex class model than MOE-hard. Furthermore, one must distinguish model complexity from, say, the computational complexity of decision making. The NFGL model may form a complex within-component decision boundary when the labeled fraction is relatively high yet without

Table 3: Average K Value for Different Labeled Fractions for KNN Classifier.

Data Set	4%	6%	8%	10%	12%	14%	16%	18%	20%	22%	24%
Yeast	3.1	2.9	3.7	3.7	3.9	3.6	4.1	4.2	4.4	4.6	4.3
Waveform database	4.3	4.5	4.6	4.4	4.3	4.8	4.5	6.4	6.9	6.5	6.8
Image segmentation	1.9	1.6	1.7	1.6	1.6	1.8	2.2	1.4	2.2	2.1	1.9
Liver disorders	2.4	2.5	2.9	3.3	3.8	3.4	3.1	3.4	2.5	3.1	3.5
Pima Indians	2.6	3.4	3.6	3.7	3.5	4.0	4.3	3.9	4.4	3.8	4.6
Ecoli	1.2	2.8	2.1	2.8	3.6	3.8	3.6	3.4	2.7	3.3	3.8
Breast Cancer	1.8	2.8	2.6	2.2	3.5	3.0	3.2	3.2	3.2	2.7	2.3
Page Blocks	1.7	2.3	2.7	2.4	2.2	2.4	2.8	3.5	2.6	3.2	3.2

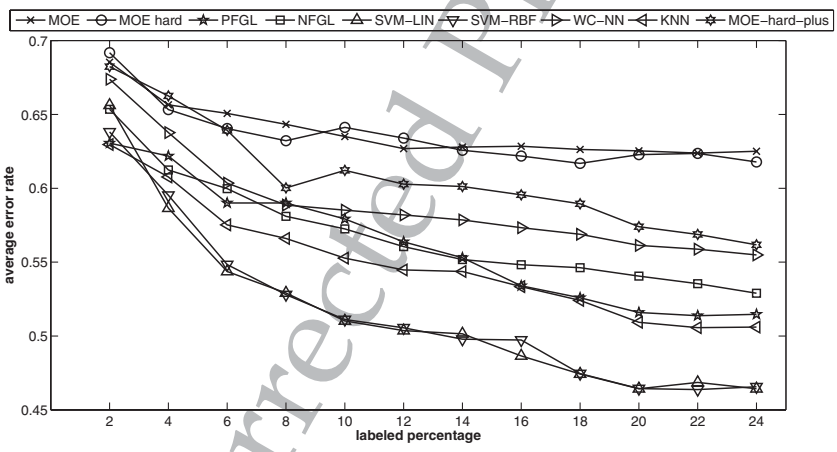

Figure 6: Average test error rate on the Yeast data set.

any increase in the number of model parameters. PFGL must increase model complexity (add prototypes) to achieve more finely grained class modeling. We are specifically evaluating MOE-hard-plus because this allows a complexity-wise fair comparison between one of our methods (PFGL) and a conventional method (MOE-hard).[7]

6.3 Classification Accuracy Evaluation. The average test error rates achieved by our methods, compared with the previous methods, are shown as a function of the labeled training set percentage in Figures 6 to 13.

As seen from the error rate curves, we can make the following observations:

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

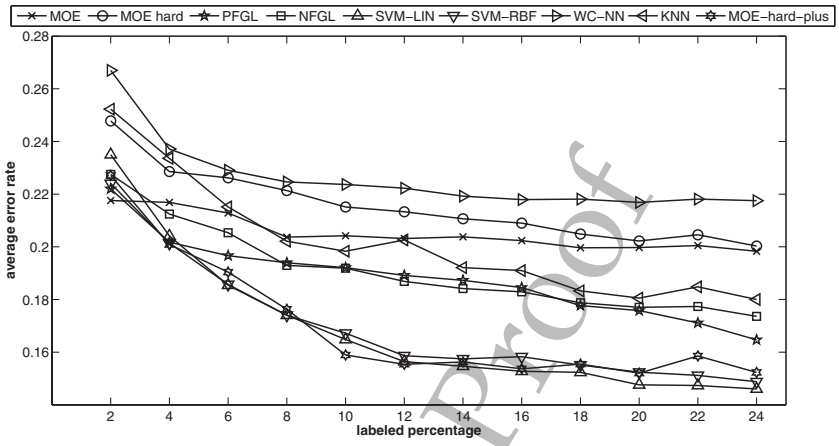

Figure 7: Average test error rate on the Waveform database generator data set.

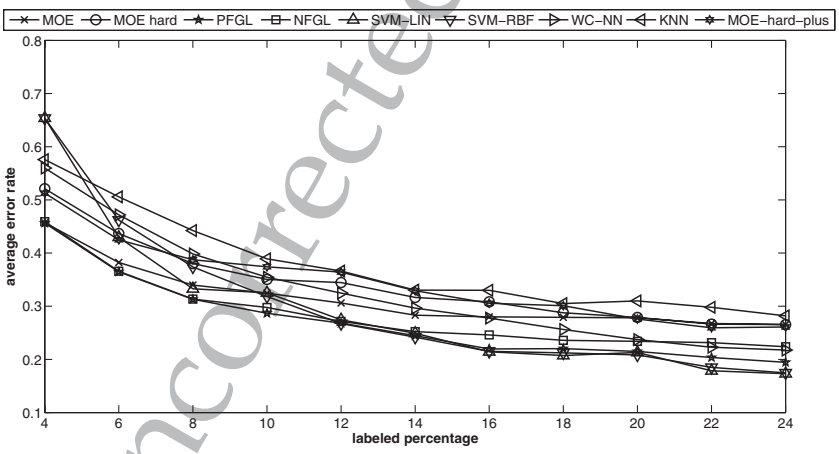

Figure 8: Average test error rate on the Image segmentation data set.

- Although there is some variability in relative performance across the data sets, both PFGL and NFGL 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 quite low labeled fractions on most data sets. Note that unlike the SVMs (and KNN), NFGL and PFGL do not use any

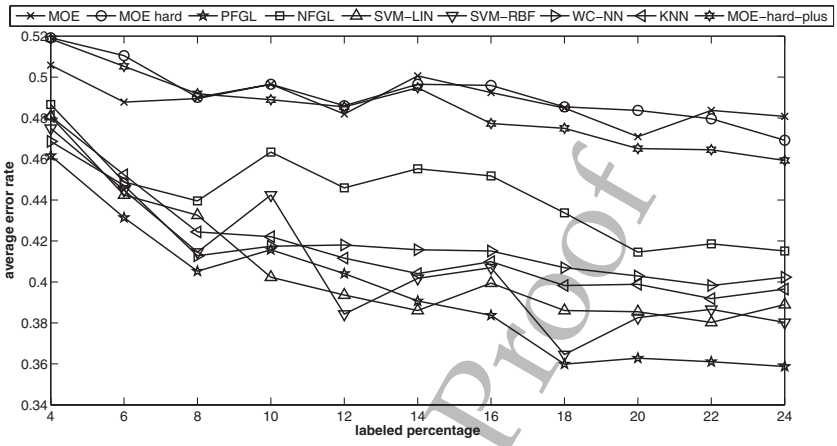

Figure 9: Average test error rate on the Liver disorders data set.

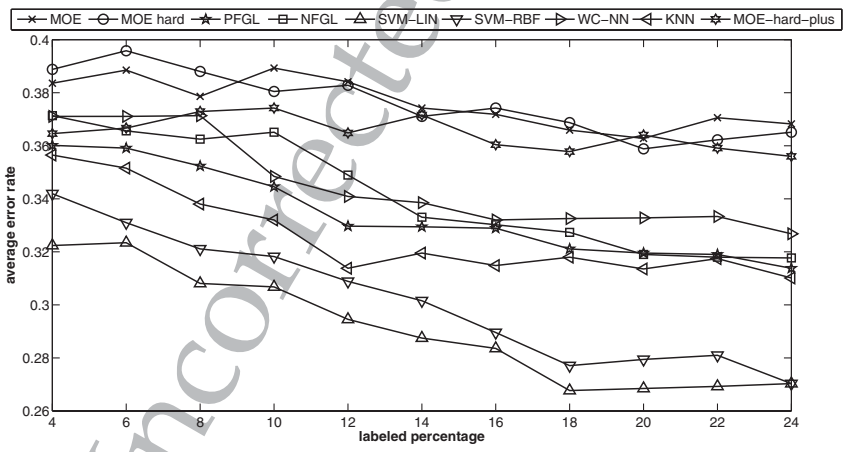

Figure 10: Average test error rate on the Pima Indians data set.

hyperparameter tuning focused on improving classification accuracy. Presumably, FGL 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, "finely grained labeling" and MOE both specialize in representing a component using a single class.

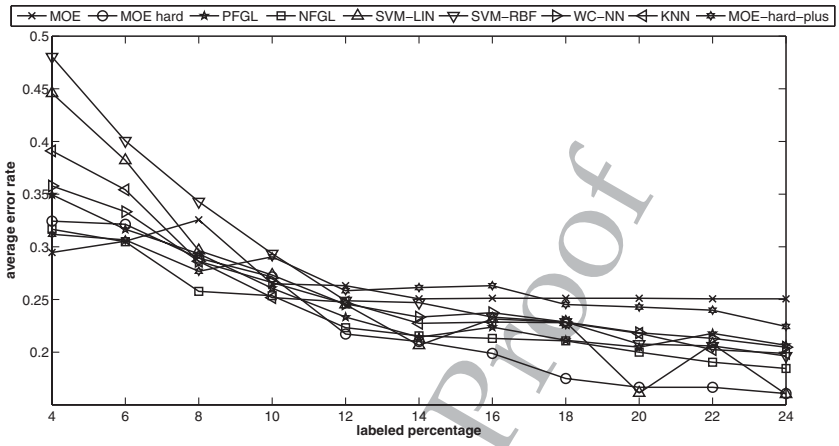

Figure 11: Average test error rate on the Ecoli data set.

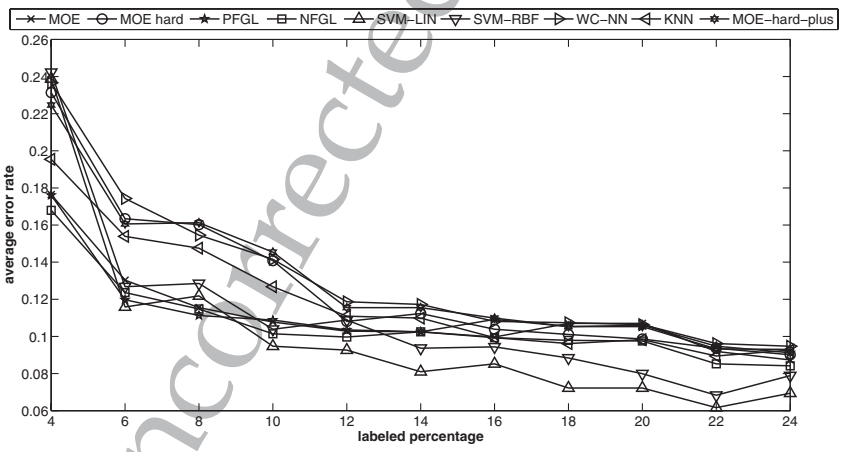

Figure 12: Average test error rate on the Breast cancer data set.

- 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 eight components (one per class), whereas the other mixture-based methods (excluding MOE-hard-plus) used only three. The MOE-hard-plus method performs somewhat better than both MOE and MOE-hard. Moreover, notably on the Waveform data (and for some labeled fractions on Page Blocks data), the use of a significant number of extra components allows it

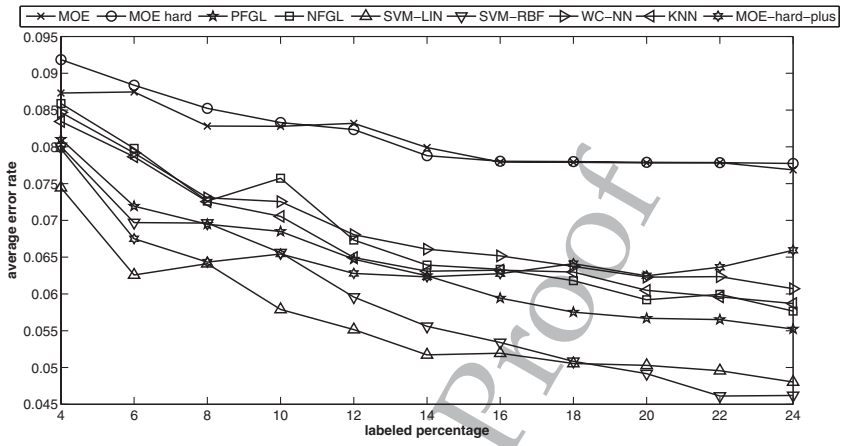

Figure 13: Average test error rate on the Page blocks data set.

to outperform the FGL methods. However, overall the FGL methods achieve higher accuracy.

7 Future Work

There are several potential directions for continuing this work. First, we may consider a Bayesian (a "fully generative") framework rather than the maximum likelihood learning framework considered here. Second, as identified in section 3, we can consider alternative within-component class posterior models to the randomized nearest-neighbor and nearest-prototype posteriors invoked here for NFGL and PFGL, respectively. Third, alternative mixture model order selection criteria may be used, such as minimum message length (Figueiredo & Jain, 2002). Fourth, while our frameworks should apply for any identifiable mixture density family, we have only fully developed and evaluated gaussian mixtures. Alternative mixture densities such as t-mixtures (Shoham, Fellows, & Normann, 2003), which are useful when dealing with outliers, could be considered in future. One caveat, though, is that at very low labeled fractions, labeled samples are precious. Thus, one should be very judicious in hypothesizing (and discarding) labeled samples as outliers. Fifth, we may evaluate our methods when the goal is semisupervised clustering or density estimation rather than classification. We may also consider alternatives to the pseudo-likelihood approximation for tractably handling our component-conditional Markov random field model within NFGL. Finally, we would like to investigate active learning extensions of our FGL methods, wherein active learning would be performed so as to directly minimize a model selection (data plus model code length) criterion. This will be investigated in our future work.

8 Conclusion

We have developed new semisupervised mixture models that introduce more finely grained intracomponent class modeling, within statistically sound generative modeling frameworks. One of our methods (NFGL) uses nonparametric within-component class modeling, which is seen to entail an underlying MRF model within each mixture component. Our second method (PFGL), which uses parameterized within-component posteriors, was motivated by a limitation of NFGL that manifests at very low labeled fractions. PFGL requires careful model order selection, to set the "right" level of intracomponent class modeling complexity. Both methods were demonstrated to outperform previous generative semisupervised mixtures, KNN, within-component NN and, at very low labeled fractions, supervised linear and nonlinear kernel SVMs, in learning statistical classifiers on UC Irvine data sets.

Acknowledgments

We gratefully acknowledge the critical comments of the anonymous reviewers, which helped us to achieve a greatly improved manuscript.

References

Asuncion, A., & Newman, D. J. (2007). *UCI Machine Learning Repository.* University of California, Irvine, School of Information and Computer Sciences. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Basu, S., & Mooney, R. J. (2003). Comparing and unifying search-based and similarity-based approaches to semi-supervised clustering. In *Proc. of the ICML-2003 Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining* (Vol. 2, pp. 42–49). Cambridge, MA: AAAI Press.

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

Besag, J. (1986). On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society, Series B*, *48*, 259–302.

Besag, J. (1974). Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society, Series B*, *36*, 192–236.

Biernacki, C., Celeux, G., & Govaert, G. (2000). Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE Trans. Pattern Anal. Mach. Intell.*, *22*, 719–725.

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proc. of the Eleventh Annual Conference on Computational Learning Theory* (Vol. 2, pp. 92–100).

Bridle, J. S. (1990). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In L. V. Fausett (ed.), *Neurocomputing: Algorithms, Architectures and Applications* (pp. 227–236). Upper Saddle River, NJ: Prentice Hall.

Chang, C. C., & Lin, C. J. (2001). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*. http://www.csie.ntu.edu.tw/cjlin/libsvm.

Chapelle, O., Chi, M., & Zien, A. (2006). A continuation method for semi-supervised SVMs. In *Proc. of the 23rd Int'l Conf. Machine Learning* (pp. 185–192). New York: ACM.

Chapelle, O., Schölkopf, B., & Zien, A. (2006). *Semi-supervised learning*. Cambridge, MA: MIT Press.

Chapelle, O., & Zien, A. (2005). Semi-supervised classification by low density separation. In *Proc. of the 10th Int'l Workshop on Artificial Intelligence and Statistics* (pp. 57–64). N.p.: Society for Artificial Intelligence and Statistics.

Chawla, N. V., & Karakoulas, G. (2005). Learning from labeled and unlabeled data: An empirical study across techniques and domains. *Journal of Artificial Intelligence Research*, *23*, 331–366.

Cohen, I., Cozman, F. G., Sebe, N., Cirelo, M. C., & Huang, T. S. (2004). Semisupervised learning of classifiers: Theory, algorithms, and their application to human-computer interaction. *IEEE Trans. Pattern Anal. Mach. Intell.*, *26*, 1553–1567.

Corduneanu, A., & Jaakkola, T. (2003). On information regularization. In *Proc. of the 19th Conf. on Uncertainty in Artificial Intelligence (UAI)* (pp. 151–158). San Francisco: Morgan Kaufmann.

Dempster, A. P., Laird, N. M., & Rudin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, *39*, 1–38.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern classification* (2nd ed.). New York: Wiley.

Fessler, J. A., & Hero, A. O. (1994). Space-alternating generalized expectation-maximization algorithm. *IEEE Trans. Signal Processing*, *42*, 2664–2677.

Figueiredo, M.A.T., & Jain, A. K. (2002). Unsupervised learning of finite mixture models. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, *24*, 381–396.

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

Graham, M. W., & Miller, D. J. (2006). Unsupervised learning of parsimonious mixtures on large spaces with integrated feature and component selection. *IEEE Trans. Signal Processing*, *54*, 1289–1303.

Hao, J., Koester, B. P., Mckay, T. A., Rykoff, E. S., Rozo, E., Evrard, A., et al. (2009). Precision measurements of the cluster red sequence using an error-corrected gaussian mixture model. *Astrophysical Journal*, *702*(1), 745.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Proc. of 16th Int'l Conf. on Machine Learning* (pp. 200–209). San Francisco: Morgan Kaufmann.

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

Karakoulas, G., & Salakhutdinov, R. (2004). Semi-supervised mixture-of-experts classification. In *Fourth IEEE Int'l Conf. on Data Mining (ICDM), 2004* (pp. 138–145). Piscataway, NJ: IEEE.

Kohonen, T., Barna, G., & Chrisley, R. (1988). Statistical pattern recognition with neural networks: Benchmarking studies. In *Proc. of IEEE Int'l Conf. on Neural Networks, 1988* (Vol. 1, pp. 61–68). Piscataway, NJ: IEEE.

Larsen, J., Hansen, L. K., Have, A. S., Christiansen, T., & Kolenda, T. (2002). Web-mining: Learning from the World Wide Web. *Computational Statistics and Data Analysis*, *38*, 517–532.

Law, M. H. C., Topchy, A., & Jain, A. K. (2004). Clustering with soft and group constraints. In *Proc. Joint IAPR Int'l Workshop on Structural and Syntactic Pattern Recognition (SSPR)* (pp. 662–670). New York: Springer.

Lawrence, N. D., & Jordan, M. I. (2005). Semi-supervised learning via gaussian processes. In L. Saul, Y. Weiss, & L. Bottou (Eds.), *Advances in neural information processing systems* (pp. 753–760). Cambridge, MA: MIT Press.

Li, S. Z. (2009). *Markov random field modeling in image analysis* (3rd ed.). New York: Springer.

McLachlan, G., & Peel, D. (2000). *Finite Mixture Models*. New York: Wiley.

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

Meng, X. L., & Dyk, D. V. (1997). The EM algorithm–An old folk-song sung to a fast new tune. *Journal of the Royal Statistical Society, Series B*, *59*, 511–567.

Miller, D. J., & Browning, J. (2003). A mixture model and EM-based algorithm for class discovery, robust classification, and outlier rejection in mixed labeled/unlabeled data sets. *IEEE Trans. Pattern Anal. Mach. Intell.*, *25*, 1468–1483.

Miller, D. J., & Uyar, H. S. (1997). A mixture of experts classifier with learning based on both labelled and unlabelled data. In M. I. Jordan, M. J. Kearns, & S. A. Solla (Eds.), *Advances in neural information processing systems, 9* (pp. 571–577). Cambridge, MA: MIT Press.

Nigam, K., McCallum, A. K., Thrün, S., & Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning*, *39*, 103–134.

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

Reynolds, D. A., Quatieri, T. F., & Dunn, R. B. (2000). Speaker verification using adapted gaussian mixture models. *Digital Signal Processing*, *10*, 19–41.

Saerens, M., Latinne, P., & Decaestecker, C. (2002). Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. *Neural Computation*, *14*, 21–41.

Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, *6*, 461–464.

Seeger, M. (2001). *Learning with labeled and unlabeled data.* (Technical Rep.). Edinburgh: University of Edinburgh

Shahshahani, B., & Landgrebe, D. (1994). The effect of unlabeled samples in reducing the small sample size problem and mitigating the Hughes phenomenon. *IEEE Trans. Geosci. Rem. Sens.*, *32*, 1087–1095.

Shental, N., & Weinshall, D. (2004). Computing gaussian mixture models with EM using equivalence constraints. In S. Thrün, L. K. Saul, & B. Schölkopf (Eds.), *Advances in neural information processing systems*, *16* (pp. 465–472). Cambridge, MA: MIT Press.

Shoham, S., Fellows, M., & Normann, R. (2003). Robust, automatic spike sorting using mixtures of multivariate t-distributions. *Journal of Neuroscience Methods*, *127*, 111–122.

Sinkkonen, J., Kaski, S., & Nikkilä, J. (2002). Discriminative clustering: Optimal contingency tables by learning metrics. In *Proc. of the 13th European Conf. on Machine Learning (ECML'02)* (Vol. 3, pp. 418–430). New York: Springer.

Stauffer, C., & Grimson, W. E. L. (1999). Adaptive background mixture models for real-time tracking. In *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition* (Vol. 2, pp. 246–252). Piscataway, NJ: IEEE.

Szummer, M., & Jaakkola, T. (2002). Partially labeled classification with Markov random walks. In T. G. Dietterich, S. Becher, & Z. Ghahramani (Eds.), *Advances in Neural Information Processing Systems, 14* (pp. 945–952). Cambridge, MA: MIT Press.

Szummer, M., & Jaakkola, T. (2003). Information regularization with partially labeled data. In S. Becker, S. Thrün, & K. Obermayer (Eds.), *Advances in neural information processing systems*, *15* (pp. 1025–1032). Cambridge, MA: MIT Press.

Ververidis, D., & Kotropoulos, C. (2005). Emotional speech classification using Gaussian mixture models. In *Proc. IEEE Int'l Symposium on Circuits and Systems* (Vol. 3, pp. 2871–2874). Piscataway, NJ: IEEE.

Wagstaff, K., Cardie, C., Rogers, S., & Schroedl, S. (2001). Constrained *K*-means clustering with background knowledge. In *Proc. of Int'l Conf. on Machine Learning* (pp. 577–584). San Francisco: Morgan Kaufmann.

Zhao, Q., & Miller, D. J. (2005). Mixture modeling with pairwise, instance-level class constraints. *Neural Computation*, *17*, 2482–2507.

Zhu, X. (2008). *Semi-supervised learning literature survey* (Tech. Rep.). Madison: University of Wisconsin at Madison.

Zhu, X., & Ghahramani, Z. (2002). *Learning from labeled and unlabeled data with label propagation* (Tech. Rep.). Pittsburgh, PA: School of Computer Science, Carnegie Mellon University.

Zhu, X., & Lafferty, J. (2005). Harmonic mixtures: Combining mixture models and graph-based methods for inductive and scalable semi-supervised learning. In *Proc. of 22th Int'l Conf. on Machine Learning* (pp. 1052–1059). New York: ACM.

Received December 23, 2010; accepted December 5, 2011.