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

David J. Miller, Chu-Fang Lin, George Kesidis, and Christopher M. Collins

Department of Computer Science and Engineering
Technical Report
CSE-10-001

January 6, 2010
Improved Generative Semisupervised Learning
Based on Fine-Grained
Component-Conditional Class Labeling

David J. Miller, Chu-Fang Lin, George Kesidis, and Christopher M. Collins
Dept. of Electrical Engineering, The Pennsylvania State University.
Rm. 227-C EE West Bldg. University Park, PA 16802
djmiller@engr.psu.edu
Technical Report
CSE 10-001

Abstract

We introduce new generative semisupervised mixtures with more
fine-grained class label generation mechanisms than in previous works.
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. We propose several
two-stage stochastic data generation mechanisms which involve first
generating unlabeled data, then generating each labeled sample, not
directly according to a component density, but rather based on unla-
beled samples that were generated according to the component density.
Variants of our method differ by whether both the samples and labels,
or just the labels, are generated based on unlabeled samples. Our gen-
eration mechanisms entail more complicated (albeit still exact) E-step
evaluations than for standard mixtures. These form the basis for generalized EM algorithms for learning. Our models are advantageous when within-component class proportions are not constant over the feature space region “owned by” a component. Experiments on UC Irvine data sets demonstrate consistent gains in classification accuracy, compared with previous semisupervised mixtures. For clustering and density estimation, our methods outperform previous semisupervised approaches, but only outperform unsupervised mixture learning when there is both significant true component overlap and sufficient labeled examples to disambiguate components.

**Keywords:** semisupervised learning, generative models, transductive inference, mixture models, nearest neighbor classification, Expectation-Maximization algorithm

1 Introduction

In many practical machine learning and data mining applications where data needs to be classified or categorized, a large number of unlabeled training samples can be easily collected, while only a limited number of labeled samples is available. Such domains include image databases, text document databases, and Internet traffic traces. Since labeled training samples may be expensive and/or time-consuming to obtain, machine learning techniques that can make effective use of both labeled and unlabeled data have received considerable recent attention [28]. 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 solely making use of the available labeled data. Alternatively, and less commonly, an unlabeled data set may be pooled with labeled samples, with the objective of more accurately clustering the data or modeling the data density. The former objective is a supervised one
(statistical classification), while the latter is generally an unsupervised one. In this paper, we develop new models applicable to both objectives.

Semisupervised approaches can be categorized into two groups: i) generative methods, wherein a stochastic data generation mechanism is explicitly hypothesized, with the associated model learning aiming to well-fit the data (typically cast as a maximum likelihood or Bayesian learning problem), e.g. [20, 21]; ii) discriminative methods, e.g. [5], wherein the sole objective is to learn a discriminant function that well-separates data from the different classes. Here our focus is on generative semisupervised learning and in particular on improving the model for class label generation within each mixture component. In the next section we review previous semisupervised mixture modeling methods [20, 21, 25] with a focus on their limitations. We identify that the class label generation mechanism in these models is fairly crude and may introduce severe model bias. In previous work [8], it was demonstrated that, if statistical modeling assumptions are incorrect, semisupervised learning may in fact degrade classification performance relative to supervised classifier learning, which relies solely on use of the labeled data. We put forward the crude label generation mechanisms in [20], [21], and [25] as, at any rate, one of the main potential sources of modeling error that may lead to such performance degradation. We also illustrate that when the goal is unsupervised mixture modeling/data clustering, the use of labeled data in [20], [21], and [25] 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 fine-grained class probability modeling within each mixture component/cluster than previous works. Our experimental results will show that this improved representation power yields improved classification accuracy over the models in [20], [21], [25]. We will also demonstrate that our models, with more fine-grained label generation, can also help to mitigate the tendency of past semisupervised methods to learn biased clustering solutions. Thus, the models we
develop have potential “dual use”, for both statistical classification and mixture modeling/clustering, and aim to improve accuracy for both these objectives. Section 5 presents experimental comparisons of our methods against alternative semisupervised and unsupervised methods, evaluating both with respect to classification and with respect to clustering and density estimation accuracies. Section 6 gives our paper conclusions.

2 Limitations of Previous Semisupervised Mixture Models

2.1 Notation

Consider a random feature vector $\mathbf{X} \in \mathbb{R}^d$, a random class label $C \in \mathcal{C} \equiv \{1, 2, \ldots, K\}$, and a random component label $M \in \mathcal{M} = \{1, 2, \ldots, L\}$. 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 \cite{16} with component densities $f_{\mathbf{X}|l}(\mathbf{x}|\theta_l), l = 1, \ldots, L$, where $\theta_l$ is the parameter set for the $l$-th density. The mixture density for $\mathbf{X}$ is thus $f_{\mathbf{X}}(\mathbf{x}) = \sum_{l=1}^{L} \alpha_l f_{\mathbf{X}|l}(\mathbf{x}|\theta_l)$. In semisupervised mixture modeling, one jointly models the feature vector $\mathbf{X}$ and the class label $C$, i.e. one learns the joint density $f_{\mathbf{X},C}(\mathbf{x}, c) = \sum_{l=1}^{L} \alpha_l f_{\mathbf{X},C|l}(\mathbf{x}, c|\theta_l)$, based on a pool of both labeled and unlabeled data samples. Denote the unlabeled data subset by $\mathcal{X}_u = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N_u}\}$, $\mathbf{x}_i \in \mathcal{R}^d$, the labeled subset by $\mathcal{X}_l = \{(\mathbf{x}_{N_u+1}, c_{N_u+1}), (\mathbf{x}_{N_u+2}, c_{N_u+2}), \ldots, (\mathbf{x}_{N_u+N_l}, c_{N_u+N_l})\}$, $\mathbf{x}_i \in \mathcal{R}^d$, $c_i \in \mathcal{C}$, the set of class labels considered by themselves, $\mathcal{C}_l = \{c_i, i = N_u+1, \ldots, N_u+N_l\}$, and the pooled data set by $\mathcal{X} = \{\mathcal{X}_u, \mathcal{X}_l\}$. 
### 2.2 Hard Versus Soft Class-to-Component Assignments

Early work on semisupervised mixture modeling includes [20, 21, 25]. There has also been a significant amount of followup interest in this area, e.g. [7, 13, 15, 22, 24, 26, 28]. An important distinction between [20] and [21, 25] is in how these methods hypothesize stochastic generation of a class label, given the mixture component of origin for a sample. We first briefly summarize [20] and how it differs from [21, 25]. We then identify two fundamental limitations common to all three of these methods, which will motivate development of our novel semisupervised mixtures in sections 3 and 4 of the paper.

In [20], the following stochastic data generation was assumed for the pooled data $\mathcal{X}$:

1) Independently, for each $x_i \in \mathcal{X}_u$:
   i) randomly select a mixture component $j$ according to $\{\alpha_l\}$; ii) randomly generate $x_i$ according to $f_{X|j}(x|\theta_j)$.

2) Independently, for each $(x_i, c_i) \in \mathcal{X}_l$:
   i) randomly select a mixture component $j$ according to $\{\alpha_l\}$; ii) randomly generate $x_i$ according to $f_{X|j}(x|\theta_j)$; iii) randomly select the class label $c_i$ according to the component-conditional probability mass function (pmf) $\{P[C = c|M = j] = \beta_{c|l}\}$.

Thus, it was assumed class labels are stochastically generated, conditioned on the sample’s component of origin and that $\mathcal{X}$ and $C$ are conditionally independent, given $M$. In this case, the joint density is $f_{X,C}(x,c) = \sum_{l=1}^{L} \alpha_l f_{X|l}(x|\theta_l)\beta_{c|l}$ and the associated class a posteriori probabilities, used for class decisionmaking, take the “mixture of experts” form:

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

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

$$
\mathcal{L}(\mathcal{X}) = \left( \prod_{i=1}^{N_u} \sum_{l=1}^{L} \alpha_l f_{X|l}(x_i|\theta_l) \right) \left( \prod_{i=N_u+1}^{N_u+N_l} \sum_{l=1}^{L} \alpha_l f_{X|l}(x_i|\theta_l) \beta_{ci|l} \right).
$$

(2)

This was achieved in a locally optimal fashion via an Expectation-Maximization (EM) algorithm [9], developed in [20].

The methods in [21] and [25] are closely related to [20]. Like [20], [21] learned the mixture model to maximize the joint data likelihood over $\mathcal{X}$ (whereas [25] maximized a class-conditional likelihood objective). Also like [20], [21] developed an EM algorithm for parameter learning. However, an important distinction between [20] and [21, 25] lies in how these methods hypothesize class label generation. Unlike [20], in [21, 25] it was assumed each mixture component is exclusively assigned to a single class, i.e. it only generates labeled samples from this (assigned) class. Thus, in [21] and [25], once the component of origin for a sample is selected, its class label is also determined. More specifically, let $z_{kl} \in \{0, 1\}$ be a binary parameter with $z_{kl} = 1$ indicating that component $l$ is assigned to class $k$. These parameters must be chosen to satisfy $\sum_{k=1}^{K} z_{kl} = 1$ (each component assigned to a single class). It is also highly desirable to satisfy $\sum_{l=1}^{L} z_{kl} \geq 1$ (every class assigned at least one component), because if there are classes not represented by components, a statistical classifier based on this model will completely misclassify such classes. Clearly, $\{z_{kl}\}$ is a special, restricted case of the pmf $\{\beta_{kl}\}$. Accordingly, the joint mixture density $f_{X,C}(x, c) = \sum_{l=1}^{L} \alpha_l f_{X|l}(x|\theta_l) z_{c|l}$, the associated class posterior probability $P[C = c|x] = \frac{f_{X,C}(x,c)}{f_X(x)}$, the joint data likelihood $\mathcal{L}(\mathcal{X})$, and the associated EM algorithm from [21] are specializations of [20], achieved by constraining $\beta_{c|l} \in \{0, 1\}$.

\footnote{Actually, in [21] the authors did not re-estimate $\{z_{kl}\}$ within their EM algorithm – component assignments to classes were simply initialized and then left unchanged during the optimization process.}
Figure 1: A 2-D mixture example with two components, A and B, one of which (A) generates labeled samples from two different classes. Also shown (dashed contour) is an (incorrect) learned component that exclusively generates samples from class 2.

Allowing stochastic (soft) label generation, given the mixture component of origin, i.e. $\beta_{kl} \in [0,1]$, $\sum_{k=1}^{K} \beta_{kl} = 1$, as in [20], is advantageous in some cases because the assumption of exclusive component-to-class assignments in [21, 25] may be too inflexible to accurately model the given semisupervised data. The M-step update $\beta_{kl} \rightarrow \beta_{kl}$ optimizing $\{z_{kl}\}$ for a given $l$, consistent with the constraint $\sum_{m} z_{kl} \geq 1 \forall k$, 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.
data and capture its underlying cluster structure. Consider the example shown in Fig. 1. Here, we depict a 2-D 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. 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 [21, 25] cannot accurately learn the ground-truth mixture in this case – one of the two learned components must capture all the labeled points from class 2, and the class 2 “outliers” (those from component A) would degrade parameter estimates for this learned component. A solution based on [21] is depicted via a shaded ellipse for the learned component that generates labeled samples from
class 2. The model in [20], 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 precisely reflect the proportion of labeled samples from each class within component 1 (4 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 2-D projection of the real-world 4-D *Iris* data set in Fig. 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 [23], and chose the best 4-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 shown for each sample. Note that component B clearly “owns” samples from two different classes (consistent with the Fig. 1 example). Again, [20], which allows components to generate samples from multiple classes, is more suitable than [21, 25] 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 [20] does give more flexible within-component label generation than [21, 25], this degree of flexibility is still insufficient, as we next elaborate by highlighting fundamental limitations that [20, 21, 25] all share in addressing both the supervised objective (building a statistical classifier) and the unsupervised one (discerning the underlying cluster structure).

### 2.3 Crude Within-Component Class Labeling

First, again consider the data in Fig. 1. Although [20] 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 $x$ originating from component A, [20] will
estimate $P[C = c | \mathbf{x}] = \frac{1}{2}$, $c = 1, 2$, even though it would clearly be far better to choose $P[C = 1 | \mathbf{x}, x_2 > 1] = 1$ and $P[C = 2 | \mathbf{x}, x_2 < 1] = 1$ for $\mathbf{x}$ from component A, i.e. more fine-grained within-component class labeling is needed to give accurate classification in this example. A crude, heuristic way to achieve this fine-graining would be as follows: 1) perform unsupervised clustering/mixture modeling, ignoring the available class labels; 2) make exclusive (0-1) assignments of each labeled sample to its best-fitting component; 3) for each component, directly form a nearest-prototype classifier \cite{14,27} by simply using as prototypes the labeled samples assigned to that component. An (accurate) classifier obtained in this way is depicted via an example in Fig. 3. In sections 3 and 4, we will show that this fine-grained labeling can be
achieved in a less heuristic fashion, via new generative, semisupervised mixture models with more involved stochastic data generation mechanisms than in previous proposals. Moreover, in section 5, we will show that our new generative models yield more accurate classifiers than the heuristic, fine-grained approach described above.

Figure 4: An example with two ground-truth components, each of which generates labeled samples from two classes. Assuming a two-component model, the method from [21], with exclusive class-to-component assignments, will fail to learn an accurate classifier on this example.

We also note that while the methods with exclusive class-to-component assignments [21, 25] would in fact yield a good classifier for the example shown in Fig. 1 (but not a good clustering shown, as discussed earlier), these methods will fail in other cases, e.g. for the example shown in Fig. 4. Such examples suggest more fine-grained within-component class labeling
is generally needed to achieve accurate classification. Moreover, while these examples are illustrative and hence somewhat artificial, the benefits of fine-grained labeling will be amply demonstrated on real-world data sets in our experimental results.

2.4 Accuracy for the Unsupervised (Clustering) Objective

Consider the data shown in Fig. 5, with ground-truth components depicted using solid elliptical contours. The methods in [21, 25] cannot capture the ground-truth components in this case – they must learn the (class-pure) components indicated by the dashed elliptical contours. Moreover, while the model in [20] does possess the it flexibility to represent the ground-truth mixture, for this example the ground-truth solution is not efficacious with respect to the joint data likelihood from [20] – the (class-pure) dashed contour solution in fact has greater joint data likelihood $L(X)$ (evaluated based on (2)) than the ground-truth component solution, due to the latter solution’s large class uncertainty within component B ($\beta_{1|B} = \beta_{2|B} = \frac{1}{2}$). Alternatively, a model with fine-grained within-component class labeling could achieve low class uncertainty “everywhere” within true component B (assigning class 2 for $x_1 < t$ and class 1 for $x_1 > t$). For this (until now, hypothetical) fine-grained model, the true mixture solution should be efficacious with respect to the likelihood on $(X, C)$.

In summary, from the above examples, we expect that a semisupervised mixture with fine-grained within-component class labeling may give improved accuracy for both statistical classification and for unsupervised learning/clustering objectives. We next develop generative mixture models that realize this capability.
Figure 5: A two-component ground-truth mixture (solid ellipses) and the (incorrect) model solution (dashed ellipses) that must be found by [21],[25], which is also likelihood favorable for [20]. Fine-grained within-component labeling is needed to discern the ground-truth clustering structure for this example.

3 Basic Semisupervised Mixture with Fine-Grained Labeling

As in [20], we treat class labels as data and model their stochastic generation, along with the feature data. Different from the previous methods [20, 21, 25], we hypothesize generation of $X$ in two stages, with $X_u$ first generated and then $X_l$ generated conditioned on both $X_u$ and on the (hidden) components
of origin that produced the individual samples $x_i \in X_u$, as follows:

1. Independently, for each sample $x_i$, $i = 1, \ldots, N_u$:
   (i) Select a mixture component index ($j$) according to the pmf $\{\alpha_m\}$.
   (ii) Generate $x_i$ according to the multivariate density $f_{X|j}(x|\theta_j)$.

2. Independently, for each sample $x_i$, $i = N_u + 1, \ldots, N_u + N_l$:
   (i) Select a mixture component index ($j$) according to $\{\alpha_m\}$.
   (ii) Consider the subset of samples from $X_u$ generated by component $j$ in step 1.(i) (of size $N_j$). Randomly select one of these sample indices ($i^*$) according to a uniform pmf $\frac{1}{N_j}$.
   (iii) Generate the feature vector $x_i$ according to a multivariate Gaussian density centered on $x_{i^*}$, with isotropic covariance $\Sigma = \sigma^2 I$, i.e. $x_{i^*} \sim N(x_{i^*}, \sigma^2 I)$. Also generate the class label according to the sample-conditional probability mass function $\{P[C = c|i^*] \equiv \beta_{c|i^*}, c \in C\}$.

Note that for $X_l$, the stochastic generation is more fine-grained than in [20] and two-stage hierarchical, rather than flat, with mixture components first selected, but then with feature vectors and labels generated not directly based on the component density, but rather based on samples that were previously generated based on the component density. Why we choose to first generate unlabeled samples and then labeled samples based on the unlabeled samples, rather than first generating the labeled samples and then unlabeled samples based on the labeled samples will be clarified in the sequel.

### 3.1 Incomplete, Complete, and Expected Complete Data Log-Likelihood

The model parameters associated with the above stochastic data generation consist of $\Theta = \{\{\theta_l\}, \{\alpha_l\}$,
\{\{\beta_{cj}\}, i = 1, \ldots, N_u\}, \sigma^2\}. The corresponding incomplete data log-likelihood [9], which we seek to maximize in learning these parameters, is:

\[
\log L_{inc} = \sum_{i=1}^{N_u} \log \sum_{j=1}^{L} \alpha_j f_{X|j}(x_i|\theta_j) + \\
\sum_{i=N_u+1}^{N_u+N_l} \log \sum_{j} \alpha_j \sum_{i'}^{N_u} \left( \frac{f_{X|j}(x_{i'}|\theta_j)}{\sum_{j'}^{N_u} f_{X|j'}(x_{i'}|\theta_{j'})} \right) f_X(x_i|x_{i'}; \sigma^2 I) \beta_{cj|i'}. \tag{3}
\]

Note that in the log-likelihood term for \(X_l\), the quantity in parentheses is the probability of selecting unlabeled sample \(x_{i'}\) as the model generator for labeled sample \((x_i, c_i)\), given that the mixture component of origin for both \(x_i\) and \(x_{i'}\) is \(j\). Direct maximization of \(L_{inc}\) is complicated by the denominator in this probability, \(e.g.,\) if \(f_{X|j}(\cdot)\) is a multivariate Gaussian density, gradient ascent on \(\log L_{inc}\) with respect to \(\Theta\) would need to carefully ensure that constraints on variances (positive) or covariances (positive definite) are strictly obeyed during the optimization.

A standard approach to maximizing the incomplete data log-likelihood is to develop an Expectation-Maximization (EM) algorithm [9] for the stochastic model under study, which achieves monotonic ascent in the incomplete data likelihood for successive algorithm iterations. Here we develop just such an approach for locally maximizing (3). A key step in formulating an EM algorithm is to identify natural hidden data [9] for the problem at hand. We can identify two sources of hidden data in the above stochastic data generation: i) the component of origin for each sample, both labeled and unlabeled, \(i.e., \{V_{ji} \in \{0, 1\}, i = 1, \ldots, N_u + N_l, j = 1, \ldots, L\},\) with \(\sum_{j=1}^{L} V_{ji} = 1 \forall i;\) ii) the unlabeled sample used to generate each labeled sample, given component \(j\) was selected for the labeled sample, \(i.e., Z_{i|i'j} \in \{0, 1\}, i = 1, \ldots, N_u, i' = N_u + 1, \ldots, N_u + N_l,\) satisfying \(\sum_{i=1}^{N_u} Z_{i|i'j} = 1 \forall j, i' = N_u + 1, \ldots, N_u + N_l.\)
Based on this hidden data, we can form the complete data log-likelihood \([9]\):

\[
\log \mathcal{L}_c = 
\sum_{i=1}^{N_u} \sum_j V_{ji} \log(\alpha_j f_{X|j}(x_i|\theta_j)) + 
\sum_{i=N_u+1}^{N_u+N_l} \sum_j V_{ji} \sum_{i'=1}^{N_u} Z_{i'|ij} \log(\alpha_j f_{X|j}(x_i|x_{i'}; \sigma^2 I) \beta_{c_{ij}}).
\]  

In the EM algorithm, we require evaluation of \(E[\log \mathcal{L}_c|\mathcal{X}; \Theta]\), where \(E[\cdot]\) is the expectation operator and where the hidden data are treated as random variables. We note that the expected hidden quantities in our case are probabilities: \(E[V_{ji}|\mathcal{X}; \Theta] = P[M_i = j|\mathcal{X}; \Theta]\) and \(E[V_{ji}Z_{i'|ij}|\mathcal{X}; \Theta] = P[M_i = j, I_i = i'|\mathcal{X}; \Theta]\), where we let random variable \(M_i\) denote the mixture component of origin for sample \(i\) and \(I_i\) denotes the index of the unlabeled sample that generates labeled sample \(i\). Finally, applying Bayes rule and the fact that a labeled sample and its unlabeled generator must originate from the same mixture component, we have:

\[
P[M_i = j, I_i = i'|\mathcal{X}; \Theta] = 
P[I_i = i'|M_i = j, \mathcal{X}; \Theta]P[M_i = j|\mathcal{X}; \Theta] = 
(\frac{P[M_{i'} = j|\mathcal{X}; \Theta]}{\sum_{i''=1}^{N_u} P[M_{i''} = j|\mathcal{X}; \Theta]})P[M_i = j|\mathcal{X}; \Theta], \ i' = 1, \ldots, N_u \ i = N_u + 1, \ldots, N_u + N_l.
\]  

Thus, we see that both expected hidden quantities can be written solely in terms of the mixture component posteriors for both the unlabeled and
labeled samples. The expected complete data log-likelihood is then:

\[
E[\log \mathcal{L}_c|\mathcal{X}; \Theta] = \\
\sum_{i=1}^{N_u} \sum_j P[M_i = j|\mathcal{X}; \Theta] \log(\alpha_j f_{X|j}(x_i|\theta_j)) + \\
\sum_{i=N_u+1}^{N_u+N_l} \sum_j \sum_{i'=1}^{N_u} P[M_i = j, I_i = i'|\mathcal{X}; \Theta] \cdot \\
\log(\alpha_j f_X(x_i|\mathcal{x}_{i'}; \sigma^2 I)\beta_{c_i|I'}),
\]

with (5) substituted in for \( P[M_i = j, I_i = i'|\mathcal{X}; \Theta] \).

3.2 Expectation-Maximization Algorithm

The EM algorithm consists of iterative alternation of maximization steps (choosing \( \Theta^{(t+1)} = \arg \max_\Theta E[\log \mathcal{L}_c(\Theta)|\mathcal{X}; \Theta^{(t)}] \)) and expectation steps (re-evaluating the expected log-likelihood: \( E[\log \mathcal{L}_c|\mathcal{X}; \Theta^{(t+1)}] \)). These steps ensure monotonic increase in the incomplete data log-likelihood [9]. We next develop these two steps.

3.2.1 E-Step

Based on the discussion above, the E-step amounts to recomputing the pmfs \( \{P[M_i = j|\mathcal{X}; \Theta^{(t)}]\}, i = 1, \ldots, N_u + N_l \), given the current model parameters \( \Theta^{(t)} = \Theta^2 \). For labeled samples, the E-step evaluation is in fact straightfor-
wardly derived to be:

\[
P[M_i = j | \mathcal{X}; \Theta] =
\]

\[
\frac{\alpha_j \sum_{i'=1}^{N_u} \left( \frac{f_{X|i}(x_{i'}|\theta_j)}{\sum_{i''=1}^{N_u} f_{X|i'}(x_{i''}|\theta_j)} \right) f_X(x_i|x_{i'}; \sigma^2 I) \beta_{c_i|i'}}{\sum_k \alpha_k \sum_{i'=1}^{N_u} \left( \frac{f_{X|k}(x_{i'}|\theta_k)}{\sum_{i''=1}^{N_u} f_{X|k}(x_{i''}|\theta_k)} \right) f_X(x_i|x_{i'}; \sigma^2 I) \beta_{c_i|i'}}
\]

\[
i = N_u + 1, \ldots, N_u + N_l, \forall j.
\]

(7)

For unlabeled samples, we first note that for a *standard* Gaussian mixture, \(P[M_i = j | \mathcal{X}; \Theta] = P[M_i = j | x_i; \Theta]\). However, this simplification is clearly invalid for our model since unlabeled samples are used to generate labeled samples originating from the same mixture component. However, we can still derive a closed form E-step. In particular, we start from the following equation:

\[
P[M_i = j | \mathcal{X}; \Theta^{(t)}] = \frac{\mathcal{L}(\mathcal{X}, M_i = j|\Theta^{(t)})}{\mathcal{L}(\mathcal{X}|\Theta^{(t)})}, i = 1, \ldots, N_u, \forall j.
\]

(8)

Here, recall \(\mathcal{L}(\cdot)\) denotes the data likelihood. We note that since all data samples are independently generated, both the numerator and denominator in (8) can be written as products over likelihoods for the individual samples. Moreover, many of these likelihoods are the same in numerator and denominator and cancel. The desired exact E-step quantity (mixture component posterior) is thus obtained after some careful simplification of (8), and found
to be:

\[
P[M_i = j | \mathcal{X}; \Theta] = \frac{\mathcal{L}(\mathcal{X}, M_i = j | \Theta)}{\mathcal{L}(\mathcal{X}|\Theta)}
\]

\[
= \frac{\mathcal{L}(\mathcal{X}_i, \mathcal{X}_u - x_i, x_i, M_i = j | \Theta)}{\mathcal{L}(\mathcal{X}_i, \mathcal{X}_u|\Theta)}
\]

\[
= \frac{\mathcal{L}(\mathcal{X}_i|\mathcal{X}_u, M_i = j, \Theta)\mathcal{L}(x_i, \mathcal{X}_u - x_i, M_i = j | \Theta)}{\mathcal{L}(\mathcal{X}_i, \mathcal{X}_u|\Theta)}
\]

\[
= \frac{\mathcal{L}(\mathcal{X}_i|\mathcal{X}_u, M_i = j, \Theta)\mathcal{L}(x_i|\mathcal{X}_u - x_i, M_i = j | \Theta)\mathcal{L}(\mathcal{X}_u - x_i|\Theta)}{\mathcal{L}(\mathcal{X}_i, \mathcal{X}_u|\Theta)}
\]

\[
= \frac{\mathcal{L}(\mathcal{X}_i|\mathcal{X}_u, M_i = j, \Theta)}{\mathcal{L}(\mathcal{X}_i|\mathcal{X}_u, \Theta)} \mathcal{L}(x_i|\mathcal{X}_u - x_i, M_i = j | \Theta),
\]

where \(\mathcal{L}(\mathcal{X}_i|\mathcal{X}_u, \Theta)\mathcal{L}(x_i|\mathcal{X}_u - x_i, \Theta)\)

\[
= \prod_{i=N_u+1}^{N_u+N_j} \sum_{j'=1}^{L} \alpha_{j'} \sum_{i=1}^{N_u} \left( \frac{f_{X|j'}(x_i|\theta_{j'})}{ \sum_{j''=1}^{L} f_{X|j''}(x_i|\theta_{j''})} \right) f_{X}(x_i|\mathcal{X}_i; \sigma^2 I) \beta_{c|i} \cdot \sum_{j'=1}^{L} \alpha_{j'} f_{X|j'}(x_i|\theta_{j'})
\]

(9)

and \(\mathcal{L}(\mathcal{X}_i|\mathcal{X}_u, M_i = j, \Theta)\mathcal{L}(x_i|\mathcal{X}_u - x_i, M_i = j | \Theta)\)

\[
= \prod_{i=N_u+1}^{N_u+N_j} \sum_{j'=1}^{L} \alpha_{j'} \sum_{i'=1}^{N_u} P[I_i = i'|M_i = j, M_i = j'; \mathcal{X}_u] f_{X}(x_i|\mathcal{X}_i; \sigma^2 I) \beta_{c|i'} \cdot \alpha_{j'} f_{X|j'}(x_i|\theta_{j}).
\]

(10)

Here, the probabilities \(P[I_i = i'|M_i = j, M_i = j'; \mathcal{X}_u]\) are computed for four different cases:

\[
P[I_i = i'|M_i = j', M_i = j; \mathcal{X}_u] = \begin{cases} 
1 + \frac{1}{\sum_{\mathcal{X}_u, n \neq i} P[M_n = j | \mathcal{X}_u]}, & i' = i; j' = j, \\
0, & i' = i; j' \neq j, \\
1 + \frac{1}{\sum_{\mathcal{X}_u, n \neq i} P[M_n = j | \mathcal{X}_u]}, & i' \neq i; j' = j, \\
\sum_{\mathcal{X}_u, n \neq i} P[M_n = j' | \mathcal{X}_u], & i' \neq i; j' \neq j,
\end{cases}
\]

19
where

\[ P[M_i = j | x_i] = \frac{\alpha_j f_{X|j}(x_i | \theta_j)}{\sum_{k=1}^L f_{X|k}(x_i | \theta_k)} \tag{11} \]

3.2.2 M-step

This step chooses \( \Theta^{(t+1)} = \arg \max_{\Theta} E[\log L_c(\Theta) | X; \Theta^{(t)}] \), subject to constraints that enforce the \( \{\alpha_j\} \) and \( \{\beta_{i|j}\} \) to be valid probability mass functions. For our mixture model, this (global) maximization step is achieved via the following closed form updates for all parameter types, assuming multivariate Gaussian mixture components, each with (mean vector, covariance matrix) given by \((\mu_j, \Sigma_j)\):

\[ \alpha^{(t+1)}_j = \frac{\sum_{i=1}^{N_u} P[M_i = j | X; \Theta^{(t)}] + \sum_{i'=N_u+1}^{N_u+N_l} \sum_{i'=1}^{N_u} P[M_{i'} = j, I_{i'} = i | X; \Theta^{(t)}]}{N_u + N_l}, \forall j. \tag{12} \]

\[ \mu^{(t+1)}_j = \frac{\sum_{i=1}^{N_u} x_i P[M_i = j | X; \Theta^{(t)}]}{\sum_{i=1}^{N_u} P[M_i = j | X; \Theta^{(t)}]}, \forall j. \tag{13} \]

\[ \Sigma^{(t+1)}_j = \frac{\sum_{i=1}^{N_u} (x_i - \mu^{(t+1)}_j)(x_i - \mu^{(t+1)}_j)^T P[M_i = j | X; \Theta^{(t)}]}{\sum_{i=1}^{N_u} P[M_i = j | X; \Theta^{(t)}]}, \forall j. \tag{14} \]
\[ \sigma^2(t+1) = \frac{1}{dN_l} \sum_{i=N_u+1}^{N_u+N_l} \sum_{j=1}^{L} \sum_{i'=1}^{N_u} P[M_i = j, I_i = i' | \mathcal{X}; \Theta^{(t)}] \left\| \mathbf{x}_i - \mathbf{x}_{i'} \right\|^2, \quad (15) \]

and

\[ \beta^{(t+1)}_{cl'i'} = \frac{\sum_{i=N_u+1; c_i=c}^{N_u+N_l} \sum_{j=1}^{L} P[M_i = j, I_i = i' | \mathcal{X}; \Theta]}{\sum_{i=N_u+1}^{N_u+N_l} \sum_{j=1}^{L} P[M_i = j, I_i = i' | \mathcal{X}; \Theta]}, \quad i' = 1, \ldots, N_u, \quad c \in \mathcal{C}. \]

Note that, as seen from our model’s stochastic data generation and (3), the mixture components (excepting their component masses) are not used to explain the labeled subset, \( \mathcal{X}_l \). This explains why the above parameter updates for \( \{\theta_j = (\mu_j, \Sigma_j)\} \) are solely based on \( \mathcal{X}_u \). However, since typically \( N_u \gg N_l \), \( \mathcal{X}_u \) is generally large enough in practice to expect to get good estimates of the mixture component model parameters, \( \{\theta_j\} \). On the hand, suppose we had exchanged roles in the stochastic data generation mechanism, with labeled samples first generated, and unlabeled samples then generated based on the labeled samples. In this case, the component model parameters would only have been estimated based on \( \mathcal{X}_l \) – with \( N_l \) typically small both in an absolute sense and relative to \( N_u \), there would likely be insufficient sample size to accurately estimate \( \{\theta_j\} \) in this case.

### 3.3 Transductive Class Inference/Decisionmaking

Given a model that has been learned via the EM algorithm described above, there are two types of class inference/decisionmaking that can be performed. One is batch transductive inference \([6],[20]\). Here, we are making class decisions on all samples belonging to the unlabeled subset \( \mathcal{X}_u \), i.e. these samples are first used for EM model learning and then the learned model is used to
classify this batch of (test) samples. For this (batch) case, we simply note that
the probability mass function \( \{ \beta_{ci} \} : x_i \in X_u \) is, in fact, the \textit{a posteriori} class
pmf, conditioned on (test) sample \( x_i \). Thus, maximum \textit{a posteriori} (MAP) class
inference is simply achieved via \( \hat{c}_i = \arg \max_c \beta_{ci}, i = 1, \ldots, N_u \). The
second (on-line) case is where the initial unlabeled data subset \( X_u \) is simply
used for model learning (\textit{i.e.}, in this case we do not care about class decisions
made on these samples), with decisionmaking subsequently required for \textit{new}
unlabeled (test) samples which may in fact even be sequentially observed. In
this case, we require decisionmaking for a given new unlabeled sample \( x_n \).
This will again be achieved by applying a MAP rule to a sample-conditional
class pmf \( \{ \beta_{ci} \} \). However, this new pmf must first be learned. To achieve
this, we can incrementally apply the above EM algorithm as each new (test)
sample is observed, either updating \textit{all} the model parameters starting from
their current values or, at a minimum, just learning the new pmf \( \{ \beta_{ci} \} \) while
keeping all other parameters fixed (which simply amounts to a specialization
of our EM algorithm); the initialization for \( \{ \beta_{ci} \} \) is reasonably taken to be
a uniform pmf over all classes. The converged values after running EM are
then used in MAP classification.

4 Model Extensions

4.1 Sample-dependent Variances

In the model developed in the last section, labels are “transmitted” to unlabeled samples based on proximity to labeled samples within the same cluster/component. How far labels will propagate depends on the variance, \( \sigma^2 \), common to all the unlabeled sample “generators”. However, some unlabeled samples may be very far from the nearest labeled sample, while others may be proximal to quite a few labeled samples, from several different classes. This suggests it may be suboptimal to use a common value \( \sigma^2 \) for all unlabeled sample generators. Much more flexibility is afforded by allowing each
unlabeled sample generator its own variance parameter, $\sigma_i^2$, \textit{i.e.} for $\mathbf{x}_i^\prime \in \mathcal{X}_i$, it is generated by $\mathbf{x}_i \in \mathcal{X}_u$ according to the density $f_{\mathbf{X}}(\mathbf{x}|(\mathbf{x}_i, \sigma_i^2 \mathbf{I}))^3$. The M-step update equation for these sample-dependent parameters is:

$$\sigma_i^{2(t+1)} = \frac{1}{d} \sum_{i'=N_u+1}^{N_u+N_i} \sum_{j=1}^{L} P[M_i = j, I_{i'} = i|\mathbf{X}; \Theta^{(t)}] ||\mathbf{x}_{i'} - \mathbf{x}_i||^2 \sum_{i'} \sum_{j} P[M_i = j, I_{i'} = i], \quad i = 1, \ldots, N_u$$

(16)

and should be implemented while restricting $\sigma_i^2 > \sigma_0^2$, $\forall i$, $\sigma_0^2$ a chosen minimum value, to prevent singularities from occurring. This variant of the fine-grained model will be experimentally evaluated in section 5.

### 4.2 Within-component Mixture of Class Proportions and Fine-Grained Labeling

Consider the example shown in Fig. 6. Here, use of the fine-grained labeling mechanism appears to make sense for propagating labels to unlabeled samples in the upper portion of the cluster. However, for samples in the lower portion, within-component class proportions will give a better model for the observed class labels. Ideally, different label extrapolation/propagation models could be used in different component sub-regions, or even by each sample within a component. This type of flexibility can be introduced via the following within-component mixture-based extension of the model from section 3:

\textit{An alternative solution of intermediate complexity is to allow a different variance parameter, not for each unlabeled sample, but rather for each mixture component. The idea here is that a mixture component $j$ with much greater scatter ($|\Sigma_j|$) than other components will tend to have larger inter-sample distances, and thus will require larger variance $\sigma_j^2$ for its unlabeled sample generators in order to propagate labels throughout this component/cluster. Even another variant is to allow nonisotropic covariances, \textit{i.e.} $\mathbf{X}_{i'} \sim f_{\mathbf{X}|\mathbf{x}}(\mathbf{x}|(\mathbf{x}_i, \Sigma_i)), \mathbf{x}_i \in \mathcal{X}_u$, where $\Sigma_i$ is diagonal, but possibly with different variances for each feature dimension.}
Figure 6: A mixture component with different label propagation/extrapolation models well-matched to different sub-regions – fine-grained propagation is most suitable in the upper part of the cluster, with the simpler class proportions best-matched to the lower part of the cluster.

\[
\log \mathcal{L}_{\text{inc}} = \sum_{i=1}^{N_u} \log \sum_{j=1}^{L} \alpha_j f_{X|j}(\bar{x}_i|\theta_j) + \\
\sum_{i=N_u+1}^{N_u+N_l} \log \sum_{j=1}^{L} \alpha_j \sum_{i'=1}^{N_u} \left( \frac{f_{X|j}(\bar{x}_{i'}|\theta_j)}{\sum_{j'=1}^{N_u} f_{X|j'}(\bar{x}_{i'}|\theta_j)} \right) [q_{i'} f_{X}(\bar{x}_i|\bar{x}_{i'}; \sigma^2 I) \beta_{c_i|i'} + (1 - q_{i'}) f_{X|j}(\bar{x}_i|\theta_j) \gamma_{c_i|j}].
\]

(17)
Here, the bracketed term gives a sample-dependent (for each $i'$), two-component mixture-based explanation for $(x_i, c_i)$, based on mixing proportions $(q_{i'}, 1 - q_{i'})$. If $q_{i'} = 1$, fine-grained modeling is exclusively invoked to explain $(x_i, c_i)$, whereas if $q_{i'} = 0$, the component density model $f_{X|j}(\cdot)$ and component-conditional class proportions (now denoted $\{\gamma_{c|j}\}$) are invoked. The probabilistic parameters $\{0 \leq q_i \leq 1, i = 1, \ldots, N_u\}$ and $\{\gamma_{c|j}\}$ have closed form M-step updates that are straightforward to derive. This variant of the fine-grained model is also evaluated in section 5.

4.3 An Improved Model Achieving Fine-Grained Labeling Without Transduction

One disadvantage of the previous models is that they require transduction in order to make classification decisions for new (test) samples, i.e. the learned model does not determine an explicit decision boundary for the feature space. Another disadvantage is that, to achieve fine-grained labeling, these models require use of a somewhat implausible stochastic generation mechanism – to propagate a label from labeled sample $x_i$ to unlabeled $x_{i'}$, one must suppose that $x_i \sim f_{X|x_{i'}}(x_{i'}; \sigma^2 I)$ even if there are unlabeled samples that are far closer to $x_i$ than $x_{i'}$ (and, thus, much more likely to have generated $x_i$). A related disadvantage is seen by using the model from section 3 to stochastically generate a synthetic data set. As seen in Fig. 7, the use of unlabeled samples for nested (labeled) data generation leads to an implausible data cluster, particularly the labeled samples produced at the cluster’s periphery.

In this section, we modify the model from section 3 to remedy both these disadvantages.

To overcome the second disadvantage, our new model does not hypothesize that the pair $(x_i, c_i)$ is generated based on an unlabeled sample (e.g., $x_{i'}$) from the same originating component ($j$); rather, it hypothesizes that only the label $c_i$ is generated based on $x_{i'}$, with the feature vector itself generated based on component $j$’s model, i.e. $x_i \sim f_{X|j}(x_j; \theta_j)$. In this way, the problem
Figure 7: Semisupervised data set realization for a component generated according to the fine-grained model from section 3, with labeled samples only displayed at the cluster’s periphery. Note that the data scatter is not consistent with a Gaussian cluster, in particular the labeled samples generated based on unlabeled samples at the cluster’s periphery.

at the cluster periphery, shown in Fig. 7, is avoided. To overcome the first disadvantage, we parameterize stochastic generation of label $c_i$ based on unlabeled sample $x_i'$. Suppose $x_i'$ is an unlabeled sample generated by component $j$ and consider the set of labeled samples originating from this same component (i.e., $\{i \in \{N_u + 1, N_u + 2, \ldots, N_u + N_l\} : V_{j|i} = 1\}$). These labeled samples can be used to form a hard, within-component nearest-prototype classifier [14] as we discussed in section 2. Alternatively, the following randomized version of the nearest prototype classifier gives a within-component,
sample-conditioned class generator:

\[
P[C = c|\mathbf{x}_i'; V_{ji'}^i = 1] = \frac{\sum_{i=N_u+1}^{N_u+N_l} V_{ji'}^i \exp(-a_j \|\mathbf{x}_i - \mathbf{x}_i'\|^2)}{\sum_{i=N_u+1}^{N_u+N_l} V_{ji'}^i \exp(-a_j \|\mathbf{x}_i - \mathbf{x}_i'\|^2)}, \forall c. \tag{18}
\]

Here, \(a_j\) is a component-specific scale parameter. Note that as \(a_j \to \infty\), \(P[C = c|\mathbf{x}_i'; V_{ji'}^i = 1] \to \{0, 1\}\), consistent with a hard nearest-prototype rule applied within component \(j\). This randomized classifier has been applied before within the context of (discriminative) supervised classifier design [19]. Here, we are invoking it within a generative, semisupervised learning framework.

**Stochastic Data Generation:**

Based on the above description, our stochastic data generation for \(\mathcal{X}_l\) (\(\mathcal{X}_u\) is generated first, following the same mechanism given in section 3) is as follows:

Independently, for each sample \(\mathbf{x}_i\), \(i = N_u + 1, \ldots, N_u + N_l\):

(i) Select a mixture component index \((j)\) according to \(\{\alpha_m\}\).

(ii) Randomly generate \(\mathbf{x}_i\) according to \(f_{\mathcal{X}_j}(\mathbf{x}|\theta_j)\).

(iii) Consider the subset of samples from \(\mathcal{X}_u\) generated by component \(j\) (of size \(N_j\)). Randomly select one of these sample indices \((i^*)\) according to a uniform pmf \(\frac{1}{N_j}\). Generate the class label \(c_i\) according to the sample-conditional probability mass function \(\{P[C = c|\mathbf{x}_{i^*}; V_{ji'} = 1]\}\) as given in (18).

The associated complete data log-likelihood, based on the same hidden data as before, \(\{V_{ji'}, \{Z_{i'|ji'}\}\}\) is:
log $L_c = \sum_{i=1}^{N_u+N_l} \sum_{j=1}^{L} V_{ji} \log(\alpha_j f_{\theta_j} (x_i | \theta_j)) +$

$$\sum_{i=N_u+1}^{N_u+N_l} \sum_{j=1}^{L} V_{ji} \sum_{i'=1}^{N_u} Z_{i'ij} \log(\frac{\sum_{m=N_u+1}^{N_u+N_l} V_{jm} \exp(-a_j \|z_m - z_{i'}\|^2)}{\sum_{m=N_u+1}^{N_u+N_l} V_{jm} \exp(-a_j \|z_m - x_i\|^2)}).$$  \hspace{1cm} (19)

**E-step:**

If we treat all the hidden data as random variables, it is unfortunately intractable to evaluate $E[\log L_c | X; \Theta]$ because the second sum in (19) is a complicated nonlinear function of all variables in the set $\{ \{V_{ji}\}, i = N_u + 1, \ldots, N_u + N_l \}$. However, it is possible to evaluate $E[\log L_c | X; \Theta]$ if we treat these hidden data variables as unknown deterministic parameters, i.e. $\{ \{v_{ji} \forall j\}, i = N_u + 1, \ldots, N_u + N_l \}$, rather than as random variables. There is precedence for such treatment, both in proposals to maximize a mixture model’s complete data log-likelihood [12],[3], as well as from K-means clustering, which also effectively treats data assignments to clusters as binary (0-1) parameters to optimize. In our case, such treatment greatly assists development of a tractable learning procedure. In particular, assuming these hidden variables are unknown deterministic parameters, $E[\log L_c | X; \Theta]$ can be precisely evaluated, based on the quantities $E[V_{ji} | X; \Theta] = P[M_i = j | X; \Theta], i = 1, \ldots, N_u$ and $E[Z_{i'ij} | X; \Theta] = P[I_i = i' | X; M_i = j; \Theta] = \sum_{i''=1}^{N_u} \frac{P[M_{i''} = j | X; \Theta]}{\sum_{i'=1}^{N_u} P[M_{i'} = j | X; \Theta]}$, $i' = 1, \ldots, N_u, i = N_u + 1, \ldots, N_u + N_l$. These quantities can be derived in closed form following the same approach taken in section 3.3. Furthermore, again by assuming the labeled data assignments to components are unknown parameters to estimate, the associated incomplete data

---

4In fact, it can be shown that, for the squared Euclidean distance distortion 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.
log-likelihood is:

$$\log \mathcal{L}_{inc} = \sum_{i=1}^{N_u} \log \left( \sum_{j=1}^{L} \alpha_j f_{X|j}(x_i|\theta_j) \right) + \sum_{i=N_u+1}^{N_u+N_l} \sum_{j} v_{ji}| \log(\alpha_j f_{X|j}(x_i|\theta_j)) + \sum_{i'=1}^{N_u} \frac{f_{X|j}(x_{i'}|\theta_j)}{\sum_{i''=1}^{N_u} f_{X|j}(x_{i''}|\theta_j)} \log(\frac{\sum_{m=N_u+1}^{N_u+N_l} v_{jm}\exp(-a_j||x_m - x_{i'}||^2)}{\sum_{m=N_u+1}^{N_u+N_l} v_{jm}\exp(-a_j||x_m - x_{i'}||^2)})$$

(20)

**Generalized M-step:**

Consider the new parameter set $\Theta = \{\{\theta_j\}, \{\alpha_j\}, \{a_j\}, \{v_{ji}\}\}$, which treats the labeled data sample assignments to components as unknown parameters to be estimated. Here, we specify a generalized EM (GEM) algorithm [10],[17],[11], for choosing these parameters to locally maximize the above incomplete data log-likelihood. A GEM algorithm, rather than a pure EM algorithm, is required for the current model because of the complicated dependence in both $\log \mathcal{L}_{inc}$ and $E[\log \mathcal{L}_c|\mathcal{X}; \Theta]$ on the parameters $\{v_{ji}\}$, which precludes a closed form M-step for these parameters. Our GEM algorithm consists of alternating E-steps and generalized M-steps, and gives parameter updates that are non-decreasing in $\log \mathcal{L}_{inc}$. The E-step was specified above. The generalized M-step considers, in turn, different parameter subsets, and optimizes these parameter subsets, given all other parameters in $\Theta$ held fixed. Each such optimization, and hence the sequence of such optimizations which comprises a generalized M-step, is non-decreasing in the incomplete data log-likelihood. Below, we specify the optimization steps comprising our 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 $\{\theta_j = (\mu_j, \Sigma_j)\}$ which globally maximize $E[\log \mathcal{L}_c|\mathcal{X}; \Theta^{(t)}]$ 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:

\[
\alpha_j^{(t+1)} = \frac{\sum_{i=1}^{N_u} P[M_i = j|\mathcal{X}; \Theta^{(t)}] + \sum_{i'=N_u+1}^{N_u+N_l} \sum_{i=1}^{N_u} v_{j|i'} P[I_{i'} = i|M_{i'} = j; \mathcal{X}; \Theta^{(t)}]}{N_u + N_l}, \forall j,
\]

(21)

\[
\mu_j^{(t+1)} = \frac{\sum_{i=1}^{N_u} \mathbb{E}_i P[M_i = j|\mathcal{X}; \Theta^{(t)}] + \sum_{i'=N_u+1}^{N_u+N_l} \sum_{i=1}^{N_u} \mathbb{E}_{i'} v_{j|i'} P[I_{i'} = i|M_{i'} = j; \mathcal{X}; \Theta^{(t)}]}{\sum_{i=1}^{N_u} P[M_i = j|\mathcal{X}; \Theta^{(t)}] + \sum_{i'=N_u+1}^{N_u+N_l} \sum_{i=1}^{N_u} v_{j|i'} P[I_{i'} = i|M_{i'} = j; \mathcal{X}; \Theta^{(t)}]}, \forall j,
\]

(22)

\[
\Sigma_j^{(t+1)} = \frac{\sum_{i=1}^{N_u} (\mathbb{E}_i - \mu_j^{(t+1)})(\mathbb{E}_i - \mu_j^{(t+1)})^T P[M_i = j|\mathcal{X}; \Theta^{(t)}]}{\sum_{i=1}^{N_u} P[M_i = j|\mathcal{X}; \Theta^{(t)}] + \sum_{i'=N_u+1}^{N_u+N_l} \sum_{i=1}^{N_u} v_{j|i'} P[I_{i'} = i|M_{i'} = j; \mathcal{X}; \Theta^{(t)}]} + \frac{\sum_{i'=N_u+1}^{N_u+N_l} \sum_{i=1}^{N_u} (\mathbb{E}_{i'} - \mu_j^{(t+1)})(\mathbb{E}_{i'} - \mu_j^{(t+1)})^T v_{j|i'} P[I_{i'} = i|M_{i'} = j; \mathcal{X}; \Theta^{(t)}]}{\sum_{i=1}^{N_u} P[M_i = j|\mathcal{X}; \Theta^{(t)}] + \sum_{i'=N_u+1}^{N_u+N_l} \sum_{i=1}^{N_u} v_{j|i'} P[I_{i'} = i|M_{i'} = j; \mathcal{X}; \Theta^{(t)}]}, \forall j.
\]

(23)

One final advantage over the model in section III (in addition to, as aforementioned, not needing transduction for decision making and being more suitable for elliptical cluster discovery) can be seen from the generalized M-step updates shown above. Note that for the model considered in this section, \{\theta_j\} is estimated based on both \mathcal{X}_u and \mathcal{X}_l, unlike the model in section III where these parameters were estimated based solely on \mathcal{X}_u. In some cases, where \lambda_l is comparable in size to \lambda_u, this increase in sample size may sig-
significantly improve parameter estimates.

*Update of Scale Parameters, \( \{ a_j \} \):*

Next, given all other parameters fixed (at \( \{ \alpha_j^{(t+1)} \}, \{ \theta_j^{(t+1)} \}, \{ v_j^{(t)} \} \)), 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 via gradient ascent, applied to the objective function \( \log L_{inc} \) in (20). 5

*Update of the \( \{ v_{ji} \} \) Variables:*

Next, 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 to components. For these variables, there are two possibilities. If \( \{ v_{ji}, i = N_u + 1, \ldots, N_u + N_l \} \) are treated as binary variables, each belonging to \( \{0, 1\} \), then they can be optimized cyclically, i.e. 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 \( \Theta \) held fixed, and then repeating for sample \( (i+1) \) modulo \( N \). 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 \( (x_i, c_i) \) to the component \( (L \) possible choices) such that the largest log-likelihood, \( \log L_{inc} \) in (20), is achieved.

This cyclical optimization is akin to the discrete optimization approaches previously taken in [2],[18],[11]. Alternatively, if we let \( v_{ji} \in [0, 1] \), we can without loss of representation power express \( v_{ji} = \frac{e^{v_{ji}}}{\sum_{k=1}^{L} e^{v_{ki}}} \), i.e. using softmax functions [4], [18]. In this case, we can jointly optimize the real-valued parameters \( \{ v_{ji}, \forall j, i = N_u + 1, \ldots, N_u + N_l \} \) via gradient ascent on (20), again

5One could choose \( \{ a_j \} \) to maximize either \( E[\log L_c | \mathcal{X}; \Theta] \) or \( \log L_{inc} \). Both choices will lead to monotonic increase in \( \log L_{inc} \), which is the objective of a GEM step. However, it is plausible that directly maximizing over the target objective function (\( \log L_{inc} \)), rather than the EM algorithm’s lower bound term \( (E[\log L_c | \mathcal{X}; \Theta]) \), should yield faster algorithm convergence to a local maximum of \( \log L_{inc} \).
with all other parameters in $\Theta$ held fixed. These two options for determining $\{v_{ji}^{(t+1)}\}$ will be experimentally explored in section 5.

In summary, our algorithm for learning this new semisupervised mixture model performs a sequence of iterations (monotonically ascending in (20)), consisting of: i) E-step, evaluating $P[M_i = j|X; \Theta]$, $j = 1, \ldots, L$, $i = 1, \ldots, N_u$; ii) generalized M-step updates for $\{\alpha_j\}$ and $\{\theta_j\}$, given $\{a_j\}$ and $\{v_{ji}\}$ held fixed; iii) gradient ascent update of $\{a_j\}$ given all other parameters in $\Theta$ held fixed; iv) either cyclical optimization of binary parameters $\{v_{ji}\}$ (until there are no further data assignment-to-component changes, from one complete sweep through the labeled data to the next) or gradient ascent update of the $\{\nu_{ti}\}$, given all other parameters in $\Theta$ held fixed. In practice, these iterations are applied until the gain (or relative gain) in $\log L$ from one iteration to the next falls below a preset convergence threshold.

**Class Decisionmaking:**

Based on the above stochastic model, the class a posteriori probabilities for a new (test) sample $\underline{x}$ can be directly evaluated, rather than requiring a transductive step. In particular, we have:

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

$$
= \sum_{j} P[C = c | M = j; \underline{x}; \Theta] \frac{\alpha_j f_{\underline{x}|j}(\underline{x}|\theta_j)}{\sum_{m} \alpha_m f_{\underline{x}|m}(\underline{x}|\theta_m)}, \quad (24)
$$

with $P[C = c | M = j; \underline{x}; \Theta]$ given by (18).

### 5 Experimental Results

We have evaluated both supervised (classification) and unsupervised (clustering and density estimation) accuracies, in comparison with three other methods, in order to comprehensively evaluate the performance of the proposed
methods. For supervised classification, we evaluated the “basic fine-grained labeling method” (BFGL), (section 3), the extension using a mixture of class proportions and fine-grained labeling (MCPFGL), (section 4.2), and the improved model (FGL-Imp), (section 4.3). For BFGL and MCPFGL, we used sample-dependent variances (section 4.A) since this was found to outperform the fixed variance version. For FGL-Imp, we used cyclical optimization to learn the \( \{v_{jik}\} \) parameters since this was found to give results comparable to the softmax optimization, and with less computational cost.

These methods were compared with the mixture of experts classifier (MOE) [20], the specialization of MOE with hard component-to-class assignments (MOE-hard) [21], with a linear support vector machine (SVM), and with within-component K-nearest neighbor (WC K-NN) classification \((K = 3)\). The latter method was implemented as follows. First, a standard GMM was learned on the training set, with each sample then assigned to a component via the MAP rule. Then, for classifying a test sample, one first MAP-assigns the sample to a component and then applies the K-NN rule using the labeled training samples that belong to the same component.

For unsupervised (density estimation) evaluation, we compared the following 5 methods: standard unsupervised GMMs learned via the EM algorithm, along with the following semisupervised methods: BFGL, MCPFGL, FGL-Imp, and MOE. All the semisupervised methods learn a GMM to model the feature vector \( X \), but they also learn additional parameters that explain class labels. For the density estimation experiments, these latter parameters were simply discarded following semisupervised learning. Thus, for density estimation accuracy, we simply evaluated the GMM learned by each of the methods.

For most of the experiments, we used real-world data sets from the UC Irvine machine learning repository [1]. Table 1 summarizes the data sets and indicates whether they were used for supervised classification (‘S’) or for unsupervised learning (‘U’) evaluation. For both types of experiments, we
Table 1: Summary of the data sets used in experiments.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Number of Instances</th>
<th>Number of Attributes</th>
<th>Number of classes</th>
<th>Exp. Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Haberman’s Survival</td>
<td>306</td>
<td>3</td>
<td>2</td>
<td>S</td>
</tr>
<tr>
<td>Pima Indians Diabetes</td>
<td>768</td>
<td>8</td>
<td>2</td>
<td>S</td>
</tr>
<tr>
<td>Ecoli</td>
<td>336</td>
<td>8</td>
<td>8</td>
<td>S</td>
</tr>
<tr>
<td>Breast Cancer</td>
<td>683</td>
<td>9</td>
<td>2</td>
<td>S and U</td>
</tr>
<tr>
<td>Iris</td>
<td>150</td>
<td>4</td>
<td>3</td>
<td>S and U</td>
</tr>
<tr>
<td>Image Segmentation</td>
<td>2310</td>
<td>19</td>
<td>7</td>
<td>S and U</td>
</tr>
<tr>
<td>Yeast</td>
<td>1484</td>
<td>8</td>
<td>10</td>
<td>S and U</td>
</tr>
<tr>
<td>Letter Recognition</td>
<td>20000</td>
<td>16</td>
<td>26</td>
<td>S and U</td>
</tr>
</tbody>
</table>

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 50/50% split into training and test subsets. The same split was used for all methods. Moreover, to observe the effect of the amount of labeled data on performance of the semisupervised methods, we varied the labeled training set percentage from 8 up to as high as 80%. For a given labeled percentage, we performed ten random selections of a labeled training subset, with the same labeled/unlabeled subsets used for all the semisupervised methods. For classification experiments, the semisupervised methods used all the data (including the unlabeled test samples) for learning. They then made test decisions on the test split – for BFGL and MCPFGL these decisions were made via the respective transductive inference rules, as described previously. For FGL-Imp and the MOE methods this was done via the MAP decision rules specified previously. Results were

---

6In fact, the unlabeled “training” samples could also have been included in the classification performance evaluation. However, for consistency with the density estimation experiments, we only evaluated classification accuracy on the test split. Note, further, that since the unlabeled test samples (representing 50% of all data) are used for semisupervised learning, the fraction of labeled samples in the total pool of samples used for semisupervised learning is in fact one half of the labeled training sample fraction.
then averaged over the ten “trials”, yielding an average test set classification error rate. For unsupervised density estimation evaluation on the UCI data sets, all the methods learned using only the training split and evaluated performance on the test split\footnote{For unsupervised evaluation, if the unlabeled test samples are used for both learning and test set evaluation, the test set performance would be biased.}. Since there is no ground-truth knowledge of clusters (only ground-truth classes) for the UCI data sets\footnote{For a given UCI data set, a mixture of many Gaussian components may be needed to adequately model even some single classes.}, we evaluated the average test set log-likelihood (over the ten trials) for the GMM learned by each of the methods as the measure of density estimation accuracy.

.Parameter Initialization:. For the standard 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 $\Theta = \{\{\theta_j\}, \{\alpha_j\}\}$ for each of the semisupervised methods. For the MOE method, the $\{\beta_{c|i} = P[c_i|M_i = j]\}$ were initialized as uniform over all classes. Likewise, for BFGL and MCPFGL, the $\{\beta_{c|i'}\}$ were initialized as uniform over all classes and for MCPFGL, so too were the $\{\gamma_{c|i}\}$. The mixing proportions $\{q_i\}$ were all initialized as 0.5. For BFGL and MCPFGL, sample-dependent variances $\{\sigma_i\}$ were all initialized as the global variance of the data set divided by the number of samples. For the FGL-Imp method, $\{v_{j|i}\}$ were randomly assigned as 0 or 1, and the scale parameters $\{a_j\}$ were all initialized to value 1. For model order selection, all the methods used the Bayesian Information Criterion (BIC) \[23\], based on the log-likelihood evaluated only on the samples, ignoring the labels, to select the number of mixture components.

5.1 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 8 to 15. As seen from the figures, we can observe the
following:

- Generally, BFGL, MCPFGL, and FGL-Imp all give distinct performance improvement over MOE, MOE-hard, and WC $K$-NN classification. Moreover, this advantage does not strongly depend on the labeled training fraction.
- FGL-Imp gives better performance than BFGL and MCPFGL, although these gains are more modest, compared to the gains over MOE, MOE-hard, and WC $K$-NN.
- Nonmonotonicity of curves is attributed to the fact that a different random subset of labeled samples is chosen for every trial, for every labeled fraction (if we averaged over more training/test splits and more trials, we would expect strictly monotonic characteristics).

![Figure 8: Average test error rate on Haberman’s Survival data set.](image1)

![Figure 9: Average test error rate on Pima Indians Diabetes data set.](image2)

We also evaluated the classification accuracy of a linear SVM on three two-class data sets (Haberman’s Survival, Pima Indian Diabetes, and Breast Cancer data sets) at a low labeled fraction (8%). We first tried both linear and radial basis function (RBF kernels) but found the RBF kernels did not
give improved results. We varied parameter C through a range of values from 100 to 900 with an increment of 50 and then selected the C value that gave the optimal performance on the first trial of each data set. The classification error rate for Haberman’s Survival data set was 24.2% using SVM and 18.0% using FGL-Imp method; for Pima Indian Diabetes data set was 11.5% using SVM and 5.8% using FGL-Imp; for Breast Cancer data set was 8.3% using SVM and 13.4% using FGL-Imp.

5.2 Density Estimation and Clustering Evaluation

We compared GMM-based density estimation performance of all the methods by carrying out a set of experiments using both real UCI data sets, given in Table 1, and some synthetic data sets. Figures 16 to 20 show average test set log-likelihood versus the labeled training set percentage on five UCI data sets. From the log-likelihood plots, we can observe that standard GMM mixture modeling based on the EM algorithm, which is purely unsupervised (i.e., it ignores the class label information) produces overall higher test set log-likelihoods than all the semisupervised methods on all 5 data sets. How-
ever, we also note that on Breast Cancer, Image Segmentation, and Yeast, when the labeled fraction is high, FGL-Imp gives log-likelihoods comparable to or greater than the unsupervised GMM approach. None of the other semisupervised methods approaches the density estimation performance of FGL-Imp and GMM.

There are several explanations for why FGL-Imp gives improved GMM-based density estimation, compared with the other semisupervised methods: i) As noted in section IV-C, FGL-Imp models all feature vectors (those from both $\mathcal{X}_u$ and $\mathcal{X}_l$) using a standard GMM model; by contrast, BFGL and MCPFGL use the nested data generation mechanism for $\mathcal{X}_l$, which may lead to implausible Gaussian components, as illustrated by the data realization example in Fig. 7; ii) FGL-Imp uses both $\mathcal{X}_u$ and $\mathcal{X}_l$ to estimate its GMM model, whereas for BFGL only $\mathcal{X}_u$ is used, i.e. FGL-Imp estimates GMM parameters using more samples, leading to a more accurate GMM model.

It is even more important to provide an explanation for the general trend shown in the density estimation results: semisupervised learning, which tries...
Figure 14: Average test error rate on Yeast data set.

Figure 15: Average test error rate on Letter Recognition data set.

to exploit class labels to improve clustering/modeling accuracy, generally degrades density estimation accuracy on the UCI data sets compared to unsupervised learning, with the performance loss decreasing as the labeled percentage increases. One explanation for this trend lies in the fact that the semisupervised methods maximize the joint log-likelihood of features and labels, which encourages learning mixture components with high labeled-sample class purity. However, as indicated by the Iris example in Fig. 2, the cluster structure in the data may not coincide with the class structure, i.e. the ground-truth mixture components (or those that maximize the GMM-based data likelihood) may not be class-pure. In such cases, sample labels may confound semisupervised learning of accurate GMMs. As a validation for this explanation, we evaluated class purity of the mixtures learned by FGL-Imp and GMM on Iris, for which GMM dominates FGL-Imp in density estimation accuracy at all labeled fractions (Fig. 17). To measure class...
purity of a GMM, we used the “1-class per component error rate” defined as:

\[
\frac{\sum_{j=1}^{L} \sum_{x_i: M_i = j} \delta(c_i - \hat{c}_{pl}(j))}{|X|}.
\]

(25)

Here, \(\hat{c}_{pl}(j)\) is the plurality (possibly the majority) class for component \(j\), considering all unlabeled samples that are GMM MAP-assigned to component \(j\). We fixed the labeled training percentage at 32 percent and evaluated the average of the above class purity measure over ten trials. For FGL-Imp the average 1-class per component error rate was 0.0894, while for GMM this error rate was 0.1210. Thus, the FGL-Imp components have higher class purity, coinciding with poorer density estimation accuracy. This is consistent with our above explanation.

We also evaluated the five methods on a simple synthetic mixture of Gaussian data set to identify under what conditions labels may in fact be helpful for clustering/density estimation. Since we know the ground-truth component labels for all synthetically-generated samples, unlike for the UCI
Figure 18: Average test set log-likelihood on Image Segmentation.

Figure 19: Average test set log-likelihood on Yeast.

Figure 20: Average test set log-likelihood on Letter Recognition.

data sets, we can evaluate a test set component label error rate measure (error rate associated with MAP assignment of samples to components), in addition to test set log-likelihood.

Fig. 21 shows a synthetic example with two overlapped components and
labeled samples specifically chosen in the overlap region between the components (20% of training samples were labeled and 50% of the labeled samples were chosen in the overlap region. For visual clarity, the figure does not display most of the labeled samples that are not in the overlap region.). In this example, both components are class-pure, i.e., the class labels (1 and 2) also serve as component labels. The misclassification error rate on this clustering example (for test data generated according to the ground-truth GMM and MAP-assigned to the learned components) is 0.095 using unsupervised GMM, 0.09675 using MOE, 0.093 using BFGL and MCPFGL, and 0.0825 using FGL-Imp. Thus, in this example, where class labels precisely specify component labels and where there are labeled samples in the region where component uncertainty is greatest, fine-grained semisupervised learning is seen to improve the learned GMM, i.e., the error rate associated with MAP-assigning samples to components.

We next study this example in a more systematic way. In Fig. 21, the shaded ellipse indicates the region where \( P[M_i = j | x] \in [0.4, 0.6], j \in \{1, 2\} \), i.e., where component uncertainty is the greatest. To ensure a substantial number of labeled samples in the overlap region while varying the overall fraction of labeled samples, we chose 25% of labeled samples from this region, with the other 75% coming from outside of this region. Fig. 22 shows the component error rate on this synthetic mixture example as a function of the labeled training set fraction.

As seen from Fig. 22, the clustering performance for the semisupervised learning methods is enhanced as the labeled fraction increases. Moreover, the fine-grained BFGL, MCPFGL, and FGL-Imp outperform standard GMM clustering when the labeled training set fraction is larger than 28%. MOE only outperforms GMM when the labeled fraction goes above 50%. Note also that FGL-IMP achieves the best results. Fig. 23 shows test set log-likelihoods for this synthetic example as a function of labeled training set fraction. Note that when the labeled fraction is above 32%, FGL-
Figure 21: Two overlapping mixture components. The shaded area indicates the region where the component *a posteriori probability* satisfies $P[M_i = j | z] \in [0.4, 0.6], j \in \{1, 2\}$. To improve visual clarity of this example, only 32 of the 80 labeled samples (mainly those in the overlap region) are displayed.

Imp, BFGL, and MCPFGL achieve better density estimation than standard GMMs. Again, FGL-Imp achieves the best results. Finally, to demonstrate the importance of selecting sufficient samples from the overlap region, we evaluated FGL-Imp for a 36% labeled fraction, but *without* constraining that 25% of these samples come from the overlap region (as a result, a much smaller percentage will come from this region). For this case, we found average test set error rate and log-likelihood values of 0.090 and -11.9765, respectively substantially worse than the FGL-Imp results at this labeled fraction in Figures 22 and 23. Thus, having “sufficient” labeled samples in the region of greatest component uncertainty has a large impact on accuracy.
Figure 22: Component labeling error rate, based on MAP assignments, for different learning methods, versus labeled fraction (with 25 % of labeled samples in the overlap region).

Figure 23: Test set log-likelihood versus the fraction of labeled training samples, with 25 % of labeled samples in the overlap region.

of the learned mixture.

6 Conclusion

We have developed new semisupervised mixture models that build something akin to “within-component nearest-prototype classification” within a statistically sound generative semisupervised mixture modeling framework. Our methods were demonstrated to outperform previous semisupervised mixture models, within-component K-NN classification and, at low labeled fractions, supervised SVMs, in semisupervised learning of statistical classifiers on UC Irvine data sets. For density estimation and clustering objectives, FGL-Imp was shown to give the clear best results among all the tested semisupervised methods. However, this method still frequently performed worse than
standard unsupervised learning of GMMs. From the density estimation experiments on both real and synthetic data sets, we observed that FGL-Imp only achieved better results than standard GMMs on some real data sets for high labeled fractions and for synthetic data sets with one component per class and overlapped components, where component labels in the overlapped region were demonstrated to help disambiguate the clusters. Future work may consider an extension of this work for semisupervised class discovery [18]. We may also evaluate our method against instance-level constraint-based semisupervised learning methods, e.g. [26], which via hard instance-level constraints, unlike our method, are essentially “intolerant” to learning clusters that own labeled samples from more than one class.

References


