A Mixture Model and EM-Based Algorithm for Class Discovery, Robust Classification, and Outlier Rejection in Mixed Labeled/Unlabeled Data Sets

David J. Miller, *Member*, *IEEE*, and John Browning

Abstract—Several authors have shown that, when labeled data are scarce, improved classifiers can be built by augmenting the training set with a large set of unlabeled examples and then performing suitable learning. These works assume each unlabeled sample originates from one of the (known) classes. Here, we assume each unlabeled sample comes from either a known *or* from a heretofore *undiscovered* class. We propose a novel mixture model which treats as observed data not only the feature vector and the class label, but also the *fact* of label presence/absence for each sample. Two types of mixture components are posited. "Predefined" components generate data from known classes and assume class labels are *missing at random*. "Nonpredefined" components only generate unlabeled data—i.e., they capture *exclusively unlabeled* subsets, consistent with an outlier distribution or new classes. The predefined/nonpredefined natures are *data-driven*, learned along with the other parameters via an extension of the EM algorithm. Our modeling framework addresses problems involving both the known and unknown classes: 1) robust classifier design, 2) classification with rejections, and 3) identification of the unlabeled samples (and their components) from unknown classes. Case 3 is a step toward new class discovery. Experiments are reported for each application, including topic discovery for the *Reuters* domain. Experiments also demonstrate the value of label presence/absence data in learning accurate mixtures.

Index Terms—Class discovery, labeled and unlabeled data, outlier detection, sample rejection, mixture models, EM algorithm, text categorization.

1 Introduction

Consider the two-dimensional data set shown in Fig. 1. The data are represented by symbols which reflect both the underlying classes and the data owner's *knowledge* of these classes. There are three *predefined* (or *known*) classes, characterized by being known a priori to the owner/user. These classes are represented by the symbol set $\mathcal{P}_c = \{1, 2, 3\}$. Data that come with a known labeling are accordingly indicated in the figure. There are also unlabeled data. Some of these originate from one of the known classes and are accordingly represented in the figure by U_1, U_2, or U_3.[1] Other unlabeled data actually belong to classes that are wholly *unknown* to the owner, i.e., these classes are such that *all* data originating from them are unlabeled. We will refer to such classes as *unknown classes*. In this example, there is one such class, i.e., $\mathcal{P}_u = \{4\}$, with these data represented by the symbol U_4. For our purposes, the four key attributes of this data set are:

1. its mixed labeled/unlabeled character;
2. the fact that some classes (perhaps quite a few) may be unknown;
3. the fact that the data set can be well-modeled by a mixture distribution. The data in Fig. 1 were generated according to a five-component Gaussian mixture, with all data originating from the same component coming from the same class, i.e., classes "own" one or multiple mixture components.[2] This is only one possibility, chosen for illustrative purposes. More generally, there may be a *distribution* (probability mass function) over the class labels, conditioned on the mixture component of origin [25], [17]. Since mixture distributions are widely used in various applications [23], [30], it is reasonable to take a mixture as the stochastic generator for the data. We only emphasize this because an underlying mixture is a *vital* aspect of our modeling framework, developed in the sequel.
4. Finally, we note that, for the known classes, it appears in Fig. 1 (and, in fact, is the case) that labels are missing *at random*, while for the unknown classes, labels are *deterministically* (always) missing. This is a crucial fact, one which drives the modeling approach taken in this paper.

We envision four natural tasks associated with the data scenario depicted in Fig. 1.

1. **Robust Classifier Training Given a Mixed Data Set.** First, suppose the data in Fig. 1 is a *training set*, to be used for learning a classifier on the known

1. The data owner/user is not privy to the labelings U_1, U_2, and U_3. They are merely used in the figure to indicate the ground truth to the reader.

2. Class 2 consists of two components in this example.

• *The authors are with the Electrical Engineering Department, Pennsylvania State University, Rm. 227-C EE West, University Park, PA 16802. E-mail: {millerdj, jdb263}@ee.psu.edu.*

Manuscript received 29 Aug. 2002; revised 6 Mar. 2003; accepted 9 May 2003. Recommended for acceptance by Y. Amit.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 117212.

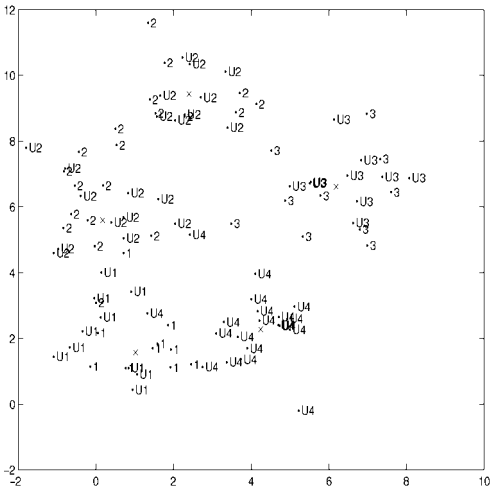

Fig. 1. An example involving partially labeled data and an unknown class. The xs denote mixture component centers.

classes, $\mathcal{P}_c$. Recently, several authors have proposed building classifiers given mixed labeled/unlabeled data, including [33], [25], [28], [3], [16]. The motivation behind much of this work is to use additional (unlabeled) data to help improve the accuracy of estimated class-conditional densities. Most of these methods assume *all* the data (including the unlabeled examples) originate from known classes and none of the data represent outliers. For the case in Fig. 1, however, the points from class 4 *are* effectively outliers with respect to $\mathcal{P}_c$. These samples will corrupt the parameter estimates and, hence, degrade the classification performance, obtained by methods such as [33], [25], [28]. It is well-known [15] that even a few outliers can have a dramatic, deleterious effect on estimation. Thus, we can identify the problem of *robust classifier learning for mixed data sets*, aiming to mitigate the effects of (unlabeled) outliers.

2. **Classification with Outlier Rejection.** As a related task, suppose the data in Fig. 1 is not the training set, but rather a new batch of data that needs to be classified (assume all points are now received *sans* labels). In this case, in addition to correctly classifying data from known classes, we would like to detect outliers and declare a *rejection* of the hypothesis that these samples belong to known classes. Rejection in classification is sometimes used to indicate that new feature information must be obtained before making a reliable decision. It is also sometimes used in a multistage classifier [29]. Here, a simple classifier is first applied. If a reliable decision cannot be reached, the simple classifier rejects the sample, triggering a more complex next stage classifier. If rejections are infrequent, the average computational requirements of the system are modest. Here, we suggest sample rejection for a different reason, i.e., *under the hypothesis that the sample is either an outlier or belongs to a new, undiscovered class.*
3. **New Class Discovery.** The third problem, and perhaps the most interesting one, is defined by shifting the emphasis (that of the first two tasks) from the *known* class domain to the *unknown* class domain. In particular, we now treat the data in Fig. 1 as a *partially labeled database*, one which is suspected to contain, within the unlabeled subset, some novel content relative to that in the labeled subset. The objective is to "mine" the database for novel content, i.e., *for unknown classes*. There are two valid interpretations of "unknown classes:" 1) where the problem involves some classes that are defined for the domain, but for which only unlabeled data has been made available. For example, consider character recognition where all instances of "2" and "z" are left unlabeled due to uncertainty on the part of the data labeler. Likewise, all instances of "z" may be unlabeled simply because the labeler (e.g., randomly) selected a fraction of the points to label, and happened not to select any instances of "z." 2) Where the data set includes examples which originate from classes that have not even been *defined* for the given domain. As one example, in astronomy we may have a set of galaxy instances which includes unlabeled samples originating from a galaxy type that has not yet been discovered by astronomers. In this case, we would say the new galaxy type is unknown, both with respect to the training set *and* more generally.

   We view class discovery as consisting of two (stratified) goals. The first is to identify the subset of samples which do not belong to known classes. This objective is really no different than that of tasks 1 and 2—identifying the unlabeled samples that are outliers with respect to the known classes. A more ambitious objective is, in addition to the first goal, to identify the *true* homogeneous subgroups, i.e., *true mixture components*, within this unknown class subset of the data. Even supposing that an algorithm is able to capture true underlying component/group structure within the outlier data, one cannot simply assert that the found components are new classes of interest.[3] However, they are at any rate good *candidates* for new classes, ones that can be put forward to a domain expert for further consideration.
4. **Mixture-based Density Estimation.** Finally, one may be more interested in the feature vector (two-dimensional in Fig. 1) rather than the class label, i.e., the objective may be mixture-based joint feature density estimation. In this case, the class labels are only of value to the extent that they help to learn a more accurate mixture for the features. That this is possible is clearly true in certain cases—suppose *all* the data were labeled, with each class consisting of a single component. In this case, the class labels allow one to break the (complex) problem of learning the mixture density into the far simpler one of separately estimating the component density for each class. Moreover, even when only a subset of the data is labeled, there is some work suggestive that unlabeled data, in conjunction with the known labels, can help improve accuracy of parameter estimates for the mixture-based density [13]. The

3. The discovered mixture components, though valid, may not be very meaningful—they could, e.g., be outlier groups caused by measurement equipment failure or human error.

methods in [33], [25] make use of labels in this way, even though the focus was on learning classifiers, rather than on density estimation. However, for the data scenario in Fig. 1, there is *additional* categorical data, beyond the known class labels. This data is the *fact* of label presence/absence for each sample. Unlike the class labels, many of which will be missing, label presence/absence data is *fully* (always) observed. Moreover, although we are not aware of prior work suggesting the use of this data for density estimation, it is plausible to expect that this data may help to learn a more accurate mixture density. One can visualize this new data as a third dimension in Fig. 1 (z), coming out of the paper, with possible values $\{0, 1\}$. Looking from above the data set, the perfect coherence of the z-values (e.g., $z = 1$) for components from unknown classes and the lack of coherence in z for components from known classes should be useful for distinguishing neighboring known and unknown class components. Moreover, if neighboring known and unknown class components are more reliably estimated, this can potentially create a "domino effect" during learning, with all the mixture components adjusted to a more accurate configuration. The value of label presence/absence for the density estimation problem will be empirically explored in Section 6.

The four tasks identified above are only compelling insofar as the assumed data scenario is realistic. Thus, we next further discuss mixed data and unknown classes.

Mixed Labeled/Unlabeled Data and Unknown Classes. In many domains, there are either huge data repositories directly available or there are resources (e.g., Internet search engines) for building them. Such domains include text [10], medical image, and remote sensing (land use) data, e.g., [33]. Unfortunately, much of this data is unlabeled and, hence, largely useless for standard supervised methods, which require a supervising label for each training set example. Providing labels for even a significant subsample of a large database may be impractical—labeling may require expensive human expertise (consider a radiologist's hourly wage). It may also simply be too arduous a task for humans, e.g., consider a text archive with 100,000 documents. The pervasity of label scarcity has, in recent years, inspired *semisupervised* methods, which utilize both labeled *and* unlabeled examples in building a classifier [33], [25], [28], [3]. As aforementioned, the premise behind much of this work is that unlabeled samples can help improve estimation of the class-conditional feature densities. There is some theoretical analysis of this approach which suggests unlabeled data may be helpful in classification [5]. Moreover, while unlabeled data is not always helpful (and can degrade performance in some cases [7], [16], [28]), studies do suggest a large pool of unlabeled data, augmenting a small labeled corpus, can significantly enhance classification accuracy in certain domains [33], [25], [28].

While the unlabeled data may come from the same repository as the labeled data, test data or new data to be classified is *another* source of unlabeled data. This suggests a *combined learning and use* paradigm [26], [20]. Here, for each new batch of data to classify, one augments the existing training set with the new (unlabeled) batch and then relearns the classifier. The resulting classifier is then used to classify the new batch. In [20], this type of approach was referred to as *transductive inference* and applied to the training/use of support vector machines.

Irrespective of the source of the unlabeled data, a basic assumption in most work on semisupervised learning is that each unlabeled sample originates from one of the *known* classes, defined on the given domain. However, if one considers a huge database, with only a small fraction (e.g., $< 20\%$) of the samples labeled, there are two other possibilities. One is that there may be statistical outliers in the unlabeled data. The mere fact of missing labels may suggest outlier presence—this might explain why an expert was unable to label some samples. A second possibility is that some subset of the unlabeled corpus may actually correspond to *unknown* classes. Suppose that the (huge) data set was collected automatically, e.g., by "intelligent agents" searching the Internet or by continuous sensing of the environment. It is quite plausible that a data set collected in this fashion will initially be *purely* unlabeled. Thus, an expert/labeler may first need to *choose* the set of classes (based, e.g., on prior knowledge of the domain) and then label a subset of the data. Given the size of the database, it is quite possible the expert might overlook some meaningful groups when defining the classes. Moreover, with caching mechanisms, the database content may change over time. These scenarios suggest the plausibility of unknown classes. We can also identify several application domains where they may arise.

Domains with Unknown Classes. Unknown classes may occur in scientific domains when their existence is inconsistent with current theory, e.g., [34]. They may also arise if there is uncertainty as to the origin of the data set or to the environment from which the data was obtained [19]. As one example, land use classes associated with multispectral data may depend on whether the land is agrarian, urban, industrial, or military. As a second example, data/text/images elicited from an Internet search are likely to provide examples from unanticipated topics/classes, e.g., [21]. Unknown classes may also occur when there is uncertainty, even in the definition of a *core* set of classes. One example is the choice of function classes for genes in molecular biology, e.g., [34]. Another is topic categorization for text documents [10] or other domains where there is subjectivity in the choice of classes.

Before developing our modeling framework to address each of the four aforementioned pattern recognition problems, we first review relevant prior work.

1.1 Robust Classifier Design, Given a Mixed Training Set

Robust classifier design is related to the *unsupervised* problem of robust clustering. In this context, our work is perhaps closest to [8] where a "noise" cluster was introduced to capture outliers, to ensure they do not contaminate true clusters. Our approach introduces *multiple* such clusters, within a mixed labeled/unlabeled framework. However, unlike [8], where the outlier distribution is not accurately modeled, we model both the known classes and the outliers. This allows our method to address modeling and inference tasks both for known and unknown classes.

In the semisupervised learning context, [28] gave a different (constant) weight to unlabeled samples than to labeled ones. However, this treats all unlabeled samples in the same way, i.e., it does not identify outliers and diminish

their impact on estimation. A method closer in spirit to ours is [35]. This is a mixture modeling approach for mixed data similar to [33], except that a robust variant of the Expectation-Maximization (EM) algorithm is employed, based on a variable weight given to each unlabeled sample. The weights are based on a soft estimate of whether a point represents an outlier. The principal *modeling* difference between our approach and [35] is that, unlike [35], we treat the fact that a class label is missing *as observed data*. The value of these observations will be demonstrated by our experimental results. We provide both random and deterministic (new class) explanations for missing labels. Our approach is well-matched to a data set that contains unknown classes. Thus, for such data, our approach should yield more accurate classifiers than [35]. A perhaps more significant difference is that, unlike [35], we explicitly model new classes/outliers. Thus, our approach can also be used to tackle new class discovery.

1.2 Identifying Unknown Class Data and Class Discovery

As suggested earlier, data sets may contain unknown classes for a variety of reasons. New class discovery is an emergent problem in scientific domains such as molecular biology [34], [1], as well as in text categorization and remote sensing [19]. We can also envision applications, e.g., in video-based object classification, where the plethora of possible objects and scenes may make unknown objects and classes quite likely. Several different class discovery problems and techniques have been previously proposed. Chang and Loew [6] overcame uncertainty in the class definitions by allowing nonmutually exclusive classes. Ben-Dor et al. [1] considered gene classification based on expression profiles. The authors defined a problem similar to standard clustering, but which allows *multiple* clusterings of a data set. The idea is that different genes "code" for different sets of classes. Somewhat closer to our work is [34], where (purely) supervised clustering is performed in the augmented feature space that includes the class label (actually, a codeword representing the label[4]). Here, each cluster representative consists of a representative feature vector *and* a class "codeword." Since the data are assumed to all be labeled, each cluster (a potential new class) owns points from at least one, and quite possibly several known classes. A cluster's codeword is the average of the known class codewords that it owns.[5] A model complexity term is used to control the number of clusters. This method "distrusts" the class labels that have been provided. Moreover, it postulates that the existing class definitions are inadequate for representing the data. Thus, new clusters (effectively, new classes) are introduced. One main difference between our work and [34] is that we consider a semisupervised framework, with new classes corresponding to purely unlabeled subsets, rather than to mislabeled ones. Our model could, however, be modified for the scenario in [34] to also possibly account for mislabeled data and suboptimal class definitions.

Finally, class discovery was considered in a semisupervised setting in [19]. This approach assumes there is only a *single* known class and that its density function is given, a priori. Additional classes are then built up via a weighted clustering algorithm. This method requires a heuristic nonparametric technique for (initially) estimating the density function defined over the unknown classes. The main differences between our work and [19] are:

1. we treat label presence/absence as observed data;
2. we do not restrict to a single known class; moreover, we *learn* the known class densities, rather than assuming them given a priori; and
3. while [19] is a heuristic approach, we develop a unified statistical framework that models both the known and unknown classes, with associated learning that performs hillclimbing in the model's likelihood function.

1.3 Organization

The rest of the paper is organized as follows: In Section 2, we develop our generative models for mixed labeled/unlabeled data. In Section 3, we introduce a method based on the expectation-maximization (EM) algorithm [9] for finding model solutions. In Section 4, we identify two different types of a posteriori probabilities that can be calculated from the learned models and used to address various inference tasks. In Section 5, we propose an integrated model learning and model selection framework for addressing new class discovery. In Section 6, we report our experimental results. The paper concludes with directions for future work.

2 A Model for Mixed Data with Unknown Classes

Consider a data set $\mathcal{X}_m$ consisting of two subsets, $\mathcal{X}_m = \{\mathcal{X}_l, \mathcal{X}_u\}$, where $\mathcal{X}_l = \{(\underline{x}_1, c_1), (\underline{x}_2, c_2), \ldots, (\underline{x}_{N_l}, c_{N_l})\}$ is the *labeled* subset, with $\underline{x}_i \in \mathcal{R}^n$ and $c_i \in \mathcal{P}_c$, and where $\mathcal{X}_u = \{\underline{x}_{N_l+1}, \ldots, \underline{x}_N\}$ is the *unlabeled* subset.[6] This mixed data scenario was considered in [33], where a class-partitioned mixture model was learned, via maximum likelihood estimation (MLE), to fit the mixed data, $\mathcal{X}_m$. In [27], [25], a mixture of experts (MOE) model [17], [38] was proposed. Unlike [33], the MOE model does not constrain mixture components to be assigned exclusively to individual classes, but rather allows components to be *probabilistically* associated with each of the classes. The probabilistic assignments are learned, along with the other model parameters, via an EM algorithm [25]. These learned assignments reflect the proportion of points from each class "owned" by a component. Such associations will provide a better fit to the data than a hard-partitioned model when the learned mixture components own data from several classes. Even if the true components do belong exclusively to individual classes, learned components will still end up owning data from multiple classes if the true component distributions (those from different classes) are overlapping. Moreover, as mentioned in Section 1, the true components may, *in fact*, generate data from more than one class. Thus, the MOE model provides a more flexible representation than the

4. For example, with three known classes, the known class codewords could be the unit vectors $(1,0,0)$, $(0,1,0)$, $(0,0,1)$.

5. For the example from footnote 4, a cluster representing a new class could, e.g., have a codeword $(\frac{1}{2}, \frac{1}{2}, 0)$ indicating it owns samples from the first two known classes (and the same number from each).

6. For concreteness, in the subsequent development, we will consider feature vectors $\underline{x} \in \mathcal{R}^n$. However, our methodology can also be applied to categorical features as, e.g., in [28].

hard-partitioned model.[7] In [27], this model was found to achieve somewhat better results and to be less sensitive to initialization than [33]. This model's classification rule has also been shown to be equivalent to that of a radial basis function classifier [26]. Since MOE is widely used, e.g., [14], and since it has some advantages over other models, e.g., [33], we choose to develop our approach for unknown classes by extending the MOE model from [25]. This entails a particular hypothesis for stochastic generation of the data. While our approach will be developed specifically by extending [25], we emphasize that the key ideas are general and, thus, could be applied to extend other mixture models as well.

2.1 Formulation

Prior semisupervised works assume all unlabeled samples originate from one of the known classes. As in [19], we allow data to also possibly originate from new, i.e., *unknown* classes. A key element in our approach is that we treat the fact of "missingness" for unlabeled samples *as observed data*. Accordingly, we redefine the data set $\mathcal{X}_m = \{\mathcal{X}_l, \mathcal{X}_u\}$, where now, $\mathcal{X}_l = \{(\underline{x}_1, "\mathrm{l}", c_1), (\underline{x}_2, "\mathrm{l}", c_2), \ldots, (\underline{x}_{N_l}, "\mathrm{l}", c_{N_l})\}$ and $\mathcal{X}_u = \{(\underline{x}_{N_l+1}, "\mathrm{m}"), \ldots, (\underline{x}_N, "\mathrm{m}")\}$. Here, we have introduced, for each sample, the *new* random observation $\mathcal{L} \in \{"\mathrm{l}", "\mathrm{m}"\}$, taking on values indicating the sample is either *labeled* or *missing* the label.[8] We propose mixture-based statistical models that are extensions of [25] and which explain *all* the observed data, including the fact of label presence/absence for each sample. For each of our (two) models, two *types* of mixture components are posited, differing in the mechanism they use for generating label presence/absence. "Predefined" components generate both labeled and unlabeled samples, and assume labels are *missing at random*. These components represent the known classes. "Nonpredefined" components *deterministically* generate unlabeled data—thus, in localized regions, they capture subsets that are *exclusively* unlabeled. Such subsets may represent an outlier distribution or *unknown* classes.

2.1.1 Notation

Let $\mathcal{M}_k$, $k = 1, \ldots, M$ denote the kth mixture component, M the number of components. Let $\mathcal{C}_{\mathrm{pre}}$ denote the subset of "predefined" components, with the remaining subset denoted $\bar{\mathcal{C}}_{\mathrm{pre}}$. Let $C \in \mathcal{P}_c$ be a random variable defined over the known classes, with $c(\underline{x}) \in \mathcal{P}_c$ the class label paired with $\underline{x}$. Let α_k denote the prior probability for component k, θ_k the parameter set specifying component k's feature density, and let $f(\underline{x}|\theta_k)$ denote this density. Here, we assume the same parametric density family for all components.

2.2 Model I: Common Missing Label Mechanism

We also define two other pmfs that are parameters specific to our first model:

1. The probability that a class label is produced, given that the sample-generating component is "predefined," i.e., $\mathrm{Prob}[\mathcal{L} = "\mathrm{l}" | \mathcal{M}_g \in C_{\mathrm{pre}}]$, where $\mathcal{M}_g$ generically denotes the generating component. Note that this probability is the same, regardless of which predefined component generates a given sample (i.e., it is tied across components). This is consistent with the scenario of a large database where, from the pool of samples that originate from known classes, the labeler randomly selects a subpool, with each sample equally likely to be labeled. In this case, the missing label probability should be the same for all $\mathcal{M}_k \in \mathcal{C}_{\mathrm{pre}}$.
2. The probability that $C = c$ given that the sample is generated by $\mathcal{M}_k$, a predefined component, and given that a label is present, i.e., $\mathrm{Prob}[C = c | \mathcal{M}_k, \mathcal{L} = "l"]$, $\mathcal{M}_k \in C_{\mathrm{pre}}$.

In summary, our first model is based on the *composite parameter set*

$$\Lambda_1 = \{\{\alpha_k\}, \{\theta_k\}, \mathcal{C}_{\mathrm{pre}}, \{\mathrm{Prob}[\mathcal{L} | \mathcal{M}_g \in C_{\mathrm{pre}}]\}, \{\mathrm{Prob}[C | \mathcal{M}_k, \mathcal{L} = "l"], \mathcal{M}_k \in \mathcal{C}_{\mathrm{pre}}\}\}.$$

Note that $\mathcal{C}_{\mathrm{pre}}$ is included in Λ_1. This reflects the fact that $\mathcal{C}_{\mathrm{pre}}$, along with the rest of Λ_1, must be chosen/learned.

2.2.1 Hypothesis for Random Generation of the Data

Our first model hypothesizes that each sample is generated independently, based on Λ_1, according to the following process:

1. Randomly select a mixture component $\mathcal{M}_j$ according to $\{\alpha_k\}$.
2. Randomly select a feature vector $\underline{x}$ according to $f(\underline{x}|\theta_j)$.
3. If $\mathcal{M}_j \in \mathcal{C}_{\mathrm{pre}}$, then select $v \in \{"\mathrm{l}", "m"\}$ according to $\mathrm{Prob}[\mathcal{L} = v | \mathcal{M}_g \in \mathcal{C}_{\mathrm{pre}}]$.
4. If $\mathcal{M}_j \in \mathcal{C}_{\mathrm{pre}}$ and $v = "\mathrm{l}"$, then select a known label according to $\mathrm{Prob}[C | \mathcal{M}_j, \mathcal{L} = "\mathrm{l}"]$.
5. If $\mathcal{M}_j \in \bar{\mathcal{C}}_{\mathrm{pre}}$, set $v = "\mathrm{m}"$.

2.2.2 Joint Data Likelihood Function

Let $v_k = 1$ if $\mathcal{M}_k \in \mathcal{C}_{\mathrm{pre}}$ and 0 if $\mathcal{M}_k \in \bar{\mathcal{C}}_{\mathrm{pre}}$. Then, the log of the joint data likelihood associated with our first model is

$$\begin{aligned}\log L_m = &\sum_{\underline{x} \in \mathcal{X}_l} \log \left(\sum_{k=1}^{M} v_k \alpha_k f(\underline{x}|\theta_k) \mathrm{Prob}[\mathcal{L} = "\mathrm{l}" | \mathcal{M}_g \in \mathcal{C}_{\mathrm{pre}}] \right. \\ &\left. \mathrm{Prob}[C = c(\underline{x}) | \mathcal{M}_k, \mathcal{L} = "\mathrm{l}"] \right) \\ &+ \sum_{\underline{x} \in \mathcal{X}_u} \log \left(\sum_{k=1}^{M} \alpha_k f(\underline{x}|\theta_k)((1 - v_k) \right. \\ &\left. + v_k \mathrm{Prob}[\mathcal{L} = "\mathrm{m}" | \mathcal{M}_g \in \mathcal{C}_{\mathrm{pre}}]) \right). \end{aligned} \quad (1)$$

Note that with this model, we have $\mathrm{Prob}[\mathcal{L} = "\mathrm{m}" | \mathcal{M}_g \in \bar{\mathcal{C}}_{\mathrm{pre}}] = 1$, i.e., nonpredefined components *deterministically* explain missing labels, whereas predefined components

7. Moreover, in practice, some learned probabilistic assignments will turn out to be hard (0-1). Thus, the model [25] both allows and typically exhibits exclusive ownership of (some) components by classes, yet without constraining such assignments.

8. Other notational choices are possible, e.g., one could indicate label presence/absence by assuming the data set is ordered, with the labeled samples coming first. Our explicit introduction of the random variable $\mathcal{L}$ is pedagogically motivated, to emphasize that label presence/absence is being treated as observed data that must be explained by the model.

hypothesize labels missing at random. The probability $\mathrm{Prob}[\mathcal{L} ='' \mathrm{m}''|\mathcal{M}_g \in \mathcal{C}_{\mathrm{pre}}]$ reflects the fraction (< 1) of data from known classes that is unlabeled. Thus, since $\mathrm{Prob}[\mathcal{L} ='' \mathrm{m}''|\mathcal{M}_g \in \bar{\mathcal{C}}_{\mathrm{pre}}] = 1 > \mathrm{Prob}[\mathcal{L} ='' \mathrm{m}''|\mathcal{M}_g \in \mathcal{C}_{\mathrm{pre}}]$, it is propitious, in the sense of the likelihood (1), to hypothesize that purely unlabeled data subsets are generated by nonpredefined components.

2.3 Model II: Class-Conditional Missing Mechanism

While our first model assumes a common missing label pmf $\mathrm{Prob}[\mathcal{L}|\mathcal{M}_{\mathrm{g}} \in \mathcal{C}_{\mathrm{pre}}]$ for *all* predefined components, one can instead imagine conditioning on the mixture component, i.e., forming $\mathrm{Prob}[\mathcal{L}|\mathcal{M}_{\mathrm{k}}], \mathcal{M}_{\mathrm{k}} \in \mathcal{C}_{\mathrm{pre}}$. However, this alternative model may provide too much flexibility in explaining missing labels. In particular, we introduce two types of components with different missing label mechanisms in order to both allow and to "encourage" nonpredefined components to capture purely unlabeled subsets, representing possible unknown classes. If predefined components are allowed to choose $\mathrm{Prob}[\mathcal{L} ='' \mathrm{l}''|\mathcal{M}_{\mathrm{k}}]$ *freely*, individual components may reduce $\mathrm{Prob}[\mathcal{L} ='' \mathrm{l}''|\mathcal{M}_{\mathrm{k}}]$ in order to better explain (and, thus, capture) unlabeled data subsets that would otherwise have only been adequately explained by nonpredefined components. Thus, the ability to discern purely unlabeled subsets and, hence, potential new classes, might be compromised.

While having a different missing label mechanism for each *component* does not appear to be promising, a valid alternative is to allow a different missing label mechanism for each *class*, i.e., to form $\mathrm{Prob}[\mathcal{L}|\mathcal{M}_{\mathrm{g}} \in \mathcal{C}_{\mathrm{pre}}, \mathrm{C} = \mathrm{c}]$, $\forall c$. In addition to explaining missing labels for the "large database" scenario,[9] this model also addresses the practical scenarios in which some classes are more confusable or less commonly encountered than others—such classes may be less likely to have labels assigned. Given this more flexible missing label mechanism, our second model's parameter set is

$$\Lambda_2 = \{\{\alpha_k\}, \{\theta_k\}, \mathcal{C}_{\mathrm{pre}}, \{\mathrm{Prob}[\mathrm{C}|\mathcal{M}_{\mathrm{k}}], \mathcal{M}_{\mathrm{k}} \in \mathrm{C}_{\mathrm{pre}}\}, \{\mathrm{Prob}[\mathcal{L}|\mathcal{M}_g \in C_{\mathrm{pre}}, C = c]\}\}.$$

Note that, in this case, the random class generator does not condition on $\mathcal{L} ='' \mathrm{l}''$. This will be explained, shortly.

2.3.1 Hypothesis for Random Generation of the Data

With this alternative model, we hypothesize that each sample is generated independently, based on Λ_2, according to the following process:

1. Select a mixture component $\mathcal{M}_j$ according to $\{\alpha_k\}$.
2. Select a feature vector $\underline{x}$ according to $f(\underline{x}|\theta_j)$.
3. If $\mathcal{M}_j \in \mathcal{C}_{\mathrm{pre}}$, then select the class label c according to $\{\mathrm{Prob}[\mathrm{C}|\mathcal{M}_{\mathrm{j}}]\}$.
4. If $\mathcal{M}_j \in \mathcal{C}_{\mathrm{pre}}$, select $v \in \{''\mathrm{l}'', ''m''\}$ according to $\{\mathrm{Prob}[\mathcal{L} = v|\mathcal{M}_g \in \mathcal{C}_{\mathrm{pre}}, C = c]\}$. If $v =''\mathrm{l}''$, then the class label c is *revealed and taken to be observed*; if $v =''\mathrm{m}''$, then the class label is *withheld and taken to be missing*.
5. If $\mathcal{M}_j \in \bar{\mathcal{C}}_{\mathrm{pre}}$, set $v =''\mathrm{m}''$.

9. This is achieved by choosing $\mathrm{Prob}\,[\mathcal{L} \mid \mathcal{M}_{\mathrm{g}} \in \mathcal{C}_{\mathrm{pre}}, \mathrm{C} = \mathrm{c}] = \mathrm{Prob}[\mathcal{L} \mid \mathcal{M}_{\mathrm{g}} \in \mathcal{C}_{\mathrm{pre}}]\ \forall_c$, i.e., by specializing to model I.

Thus, according to this model, a predefined component *always* produces a known label accompanying the feature vector—it is simply that sometimes the label is (randomly) withheld from observation.

2.3.2 Joint Data Likelihood Function

For this model, the log joint data likelihood takes the form:

$$\log L_m = \sum_{\underline{x} \in \mathcal{X}_l} \log \left(\sum_{k=1}^{M} v_k \alpha_k f(\underline{x}|\theta_k) \mathrm{Prob}[\mathrm{C} = \mathrm{c}(\underline{\mathrm{x}})|\mathcal{M}_{\mathrm{k}}] \right.$$
$$\left. \mathrm{Prob}[\mathcal{L} =''\mathrm{l}''|\mathcal{M}_g \in \mathcal{C}_{\mathrm{pre}}, C = c(\underline{x})] \right) +$$
$$\sum_{\underline{x} \in \mathcal{X}_u} \log \left(\sum_{k=1}^{M} \alpha_k f(\underline{x}|\theta_k)((1 - v_k) + v_k \sum_{c} \mathrm{Prob}[\mathrm{C} = \mathrm{c}| \right.$$
$$\left. \mathcal{M}_k] \mathrm{Prob}[\mathcal{L} =''\mathrm{m}''|\mathcal{M}_{\mathrm{g}} \in \mathcal{C}_{\mathrm{pre}}, \mathrm{C} = \mathrm{c}]) \right). \tag{2}$$

Before developing our learning algorithms for maximizing (1) and (2), it is informative to give a brief aside further justifying our models. In particular, consider the $\{v_k\}$ variables which specify $\mathcal{C}_{\mathrm{pre}}$. We are treating these hard 0-1 variables as *parameters*, to be learned. An alternative approach would be to allow components to be predefined/nonpredefined *in probability* and, thus, learn the parameters $\{\mathrm{Prob}[\mathcal{M}_k \in \mathcal{C}_{\mathrm{pre}}]\}$. However, this choice will not lead to a model that can be learned in a computationally feasible manner. To illustrate, consider the stochastic data generation associated with this model. One must *first* randomly select the predefined/nonpredefined nature for each component—there are 2^M such component configurations. Then, for the chosen configuration of natures, the data set is stochastically generated in the usual way, as previously described for models I and II. Now, consider the likelihood function associated with this model. For each of the 2^M configurations, there is an associated configuration-specific likelihood, evaluated based on (1) or (2). The likelihood for the *model* is obtained by *averaging* over these 2^M configuration-dependent likelihoods, based on the joint configuration pmf

$$\{P[V_1 = v_1, V_2 = v_2, \ldots, V_M = v_M] =$$
$$\prod_{k=1}^{M} \mathrm{Prob}[\mathcal{M}_k \in \mathcal{C}_{\mathrm{pre}}]^{v_k} (1 - \mathrm{Prob}[\mathcal{M}_k \in \mathcal{C}_{\mathrm{pre}}])^{(1 - v_k)}\}.$$

The model's likelihood is thus *explicitly* based on a weighted sum of 2^M terms, each specified by (1) or (2). It can be further seen that the concomitant complexity of learning the model grows exponentially with the number of components. Moreover, so long as the component natures are probabilistic, this cannot be simplified, even if the EM algorithm is used to maximize the likelihood. Computational difficulties are circumvented by treating the $\{v_k \in \{0, 1\}\}$ as *parameters to be learned*, as seen next.

3 Learning Based on an Extension of EM

3.1 Formulation for Model I

Our learning strategy is to perform an iterative optimization with each iteration consisting of two steps: 1) maximize $\log L_m$ over the component natures $\{v_k\}$ given the remaining parameters in Λ held fixed; 2) using the EM algorithm, maximize over the remaining parameters in Λ, given the $\{v_k\}$ held fixed. Each step is nondecreasing in $\log L_m$. This approach effectively partitions the parameter set into two subsets, with alternating optimizations performed over each subset given the remaining subset held fixed.[10] This is a type of cyclic coordinate ascent and is related to model reduction generalizations of the EM algorithm [24]. In particular, our learning algorithm is a simple instance of the general approach in [11], one in which EM is used for optimizing one of the parameter subsets, with a discrete optimization technique used for the other subset. We next develop this learning algorithm in detail for model I.

3.1.1 Optimization over Component Natures

Consider $\log L_m$ given in (1) (or (2)). Notice that the dependence on $\{v_k\}$ is somewhat complicated. There are several possible approaches to maximizing $\log L_m$ over these variables. If M is not too large, one can apply an exhaustive search over all 2^M configurations. A second method is to use a global optimization technique such as simulated annealing. As a computationally simple alternative, when exhaustive search is infeasible, we propose to iteratively select the $\{v_k\}$ one component at a time, keeping the remaining ones held fixed. Each v_k is chosen simply by evaluating $\log L_m$ for the two cases, $v_k = 0, 1$ and selecting the value yielding the greater likelihood. Cycling over the v_k choices continues until there are no further changes. This method is really an instance of *iterated conditional modes* [2]. It does not guarantee convergence to a global, or even a local optimum. However, it does guarantee ascent in $\log L_m$.

3.1.2 EM Algorithm for the Remaining Parameters

A standard approach for MLE is the EM algorithm [9]. This framework is attractive in part because of its flexibility of application and also because it yields an iterative solution that guarantees an increase in the data likelihood with each iteration. The EM framework treats the observed data as being *incomplete* and, thus, requires one to postulate unobserved/*hidden* data associated with the problem at hand. The observed and hidden data together constitute the *complete data*, with an associated likelihood. The EM approach amounts to an iteration consisting of 1) evaluation of the expected complete data likelihood (E-step), followed by 2) its maximization with respect to the model parameters (M-step). While for certain problems there are several candidate hidden data choices, in the case of mixture models, there is a natural choice that also often leads to a simple iteration. Specifically, we treat the (unknown) component of origin for each data sample as the hidden data in the EM framework. Accordingly, we define the quantities $V_{\underline{x}k}$, with $V_{\underline{x}k} = 1$ if $\underline{x} \in \mathcal{M}_k$, and 0 otherwise. Here, $\underline{x} \in \mathcal{M}_k$ denotes that sample $\underline{x}$ was (exclusively) generated by $\mathcal{M}_k$. It is then seen that the associated *complete* data log-likelihood, based on knowledge of the hidden data, can be written (for model I) in the form

$$\begin{aligned}\log L_c = &\sum_{\underline{x}\in\mathcal{X}_l}\sum_{k=1}^{M} v_k V_{\underline{x}k} \log\Big(\alpha_k f(\underline{x}|\theta_k)\mathrm{Prob}[\mathcal{L}=''1''|\mathcal{M}_g\in\mathcal{C}_{\mathrm{pre}}]\\ &\mathrm{Prob}[C=c(\underline{x})|\mathcal{M}_k,\mathcal{L}=''1'']\Big)\\ &+\sum_{\underline{x}\in\mathcal{X}_u}\sum_{k=1}^{M} v_k V_{\underline{x}k}\log\Big(\alpha_k f(\underline{x}|\theta_k)\mathrm{Prob}[\mathcal{L}=''\mathrm{m}''|\mathcal{M}_g\in\mathcal{C}_{\mathrm{pre}}]\Big)\\ &+\sum_{\underline{x}\in\mathcal{X}_u}\sum_{k=1}^{M}(1-v_k)V_{\underline{x}k}\log\Big(\alpha_k f(\underline{x}|\theta_k)\Big).\end{aligned} \tag{3}$$

Let us denote the parameter set Λ, excluding the $\{v_k\}$ parameters by Γ. Further, let us denote the parameter estimates after the tth EM iteration by $\Gamma^{(t)}$. Our EM algorithm optimizes over Γ given fixed $\{v_k\}$.

3.1.3 E-Step

In the Expectation step of the EM algorithm [9], we compute the expected complete data log-likelihood given the current parameter estimates, $\Lambda^{(t)} \equiv \{\Gamma^{(t)}, \{v_k\}\}$. This quantity is based on the expectations $E[V_{\underline{x}k} \mid \underline{x} \in \mathcal{X}_l;\ \Lambda^{(t)}]$ and $E[V_{\underline{x}k}|\underline{x}\in\mathcal{X}_u;\Lambda^{(t)}]$. Note that, based on the above definition for $V_{\underline{x}k}$, these expected quantities are simply probabilistic assignments of samples to components, i.e., $E[V_{\underline{x}k}|\underline{x}\in\mathcal{X}_l;\Lambda^{(t)}] \equiv \mathrm{Prob}[\mathcal{M}_k \mid \underline{x}\in\mathcal{X}_l;\Lambda^{(t)}]$, where, via Bayes rule:

$$\mathrm{Prob}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_l;\Lambda_1^{(t)}] = \begin{cases}\frac{\alpha_k^{(t)} f(\underline{x}|\theta_k^{(t)})\mathrm{Prob}[C=c(\underline{x})|\mathcal{M}_k,\mathcal{L}=''1'']^{(t)}}{\sum_{n\in\mathcal{C}_{\mathrm{pre}}}\alpha_n^{(t)} f(\underline{x}|\theta_n^{(t)})\mathrm{Prob}[C=c(\underline{x})|\mathcal{M}_n,\mathcal{L}=''1'']^{(t)}} & \mathcal{M}_k\in\mathcal{C}_{\mathrm{pre}}\\ 0 & \text{else,}\end{cases} \tag{4}$$

$$\mathrm{Prob}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_u;\Lambda_1^{(t)}] = \begin{cases}\frac{\alpha_k^{(t)} f(\underline{x}|\theta_k^{(t)})\mathrm{Prob}[\mathcal{L}=''\mathrm{m}''|\mathcal{M}_g\in\mathcal{C}_{\mathrm{pre}}]^{(t)}}{\sum_{n\in\mathcal{C}_{\mathrm{pre}}}\alpha_n^{(t)} f(\underline{x}|\theta_n^{(t)})\mathrm{Prob}[\mathcal{L}=''\mathrm{m}''|\mathcal{M}_g\in\mathcal{C}_{\mathrm{pre}}]^{(t)}+\sum_{n\in\bar{\mathcal{C}}_{\mathrm{pre}}}\alpha_n^{(t)} f(\underline{x}|\theta_n^{(t)})} & \mathcal{M}_k\in\mathcal{C}_{\mathrm{pre}}\\ \frac{\alpha_k^{(t)} f(\underline{x}|\theta_k^{(t)})}{\sum_{n\in\mathcal{C}_{\mathrm{pre}}}\alpha_n^{(t)} f(\underline{x}|\theta_n^{(t)})\mathrm{Prob}[\mathcal{L}=''\mathrm{m}''|\mathcal{M}_g\in\mathcal{C}_{\mathrm{pre}}]^{(t)}+\sum_{n\in\bar{\mathcal{C}}_{\mathrm{pre}}}\alpha_n^{(t)} f(\underline{x}|\theta_n^{(t)})} & \text{else.}\end{cases} \tag{5}$$

Based on these quantities, the expected complete data log likelihood is

$$\begin{aligned}E[\log L_c|\Lambda_1^{(t)}] = &\sum_{\underline{x}\in\mathcal{X}_l}\sum_{k\in\mathcal{C}_{\mathrm{pre}}}\mathrm{Prob}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_l;\Lambda_1^{(t)}]\cdot\\ &\log\Big(\alpha_k f(\underline{x}|\theta_k)\mathrm{Prob}[\mathcal{L}=''1''|\mathcal{M}_g\in\mathcal{C}_{\mathrm{pre}}]\\ &\mathrm{Prob}[C=c(\underline{x})|\mathcal{M}_k,\mathcal{L}=''1'']\Big)\\ &+\sum_{\underline{x}\in\mathcal{X}_u}\sum_{k\in\mathcal{C}_{\mathrm{pre}}}\mathrm{Prob}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_u;\Lambda_1^{(t)}]\log\\ &\Big(\alpha_k f(\underline{x}|\theta_k)\mathrm{Prob}[\mathcal{L}=''\mathrm{m}''|\mathcal{M}_g\in\mathcal{C}_{\mathrm{pre}}]\Big)\\ &+\sum_{\underline{x}\in\mathcal{X}_u}\sum_{k\in\bar{\mathcal{C}}_{\mathrm{pre}}}\mathrm{Prob}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_u;\Lambda_1^{(t)}]\log\Big(\alpha_k f(\underline{x}|\theta_k)\Big).\end{aligned} \tag{6}$$

10. For model II, we will later describe an alternative approach that involves a *three-way* partition, with three associated optimization steps per iteration.

3.1.4 M-Step

For concreteness, suppose $f(\cdot)$ is a joint Gaussian density function, with parameter set given by the mean vector and covariance matrix, i.e., $\theta_k = \{\underline{m}_k, \Sigma_k\}$.[11] In the M-step for model I, $E[\log \mathcal{L}_c | \Lambda_1^{(t)}]$ is maximized over the Γ parameters, yielding $\Lambda_1^{(t+1)} = \{\Gamma^{(t+1)}, \{v_k\}\}$. Taking partial derivatives with respect to each parameter, it is found that, except for the mean and covariance parameters, the Γ parameters are totally decoupled from each other in the respective partial derivatives. Moreover, setting each partial derivative to zero, one finds that there is a unique closed form solution (M-step) for each parameter, one which satisfies the first order necessary optimality condition. This M-step, which maximizes $E[\log \mathcal{L}_c | \Lambda_1^{(t)}]$ with respect to Γ, is given by the parameter estimates:

$$\alpha_k^{(t+1)} = \frac{1}{N}\left(\sum_{\underline{x}\in\mathcal{X}_l} \text{Prob}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_l;\Lambda_1^{(t)}] + \sum_{\underline{x}\in\mathcal{X}_u} \text{Prob}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_u;\Lambda_1^{(t)}]\right) \quad \forall k. \tag{7}$$

$$\underline{m}_k^{(t+1)} = \frac{\sum_{\underline{x}\in\mathcal{X}_l} \underline{x}\,\text{Prob}\,[\mathcal{M}_k \mid \underline{x}\in\mathcal{X}_l;\Lambda_1^{(t)}] + \sum_{\underline{x}\in\mathcal{X}_u} \underline{x}\,\text{Prob}\,[\mathcal{M}_k \mid \underline{x}\in\mathcal{X}_u;\Lambda_1^{(t)}]}{\sum_{\underline{x}\in\mathcal{X}_l} \text{Prob}\,[\mathcal{M}_k \mid \underline{x}\in\mathcal{X}_l;\Lambda_1^{(t)}] + \sum_{\underline{x}\in\mathcal{X}_u} \text{Prob}\,[\mathcal{M}_k \mid \underline{x}\in\mathcal{X}_u;\Lambda_1^{(t)}]} \quad \forall k. \tag{8}$$

$$\Sigma_k^{(t+1)} = \left(\frac{1}{\sum_{\underline{x}\in\mathcal{X}_l} \text{Prob}\,[\mathcal{M}_k \mid \underline{x}\in\mathcal{X}_l;\Lambda_1^{(t)}] + \sum_{\underline{x}\in\mathcal{X}_u} \text{Prob}\,[\mathcal{M}_k \mid \underline{x}\in\mathcal{X}_u;\Lambda_1^{(t)}]}\right) \cdot \tag{9}$$

$$\left(\sum_{\underline{x}\in\mathcal{X}_l} (\underline{x}-\underline{m}_k^{(t+1)})(\underline{x}-\underline{m}_k^{(t+1)})^T \text{Prob}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_l;\Lambda_1^{(t)}] + \sum_{\underline{x}\in\mathcal{X}_u} (\underline{x}-\underline{m}_k^{(t+1)})(\underline{x}-\underline{m}_k^{(t+1)})^T \text{Prob}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_u;\Lambda_1^{(t)}]\right).$$

$$\text{Prob}[C=c|\mathcal{M}_k,\mathcal{L}=''\,\text{l}''\,]^{(t+1)} = \frac{\sum_{\underline{x}\in\mathcal{X}_l:c(\underline{x})=c} \text{Prob}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_l;\Lambda_1^{(t)}]}{\sum_{\underline{x}\in\mathcal{X}_l} \text{Prob}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_l;\Lambda_1^{(t)}]} \quad \forall c,k : \mathcal{M}_k \in \mathcal{C}_{\text{pre}}. \tag{10}$$

$$\text{Prob}[\mathcal{L}=''\,\text{l}''|\mathcal{M}_g\in\mathcal{C}_{\text{pre}}]^{(t+1)} = \frac{\sum_{\underline{x}\in\mathcal{X}_l}\sum_{k\in\mathcal{C}_{\text{pre}}} \text{Prob}\,[\mathcal{M}_k \mid \underline{x}\in\mathcal{X}_l;\Lambda_1^{(t)}]}{\sum_{\underline{x}\in\mathcal{X}_l}\sum_{k\in\mathcal{C}_{\text{pre}}} \text{Prob}[\mathcal{M}_k \mid \underline{x}\in\mathcal{X}_l;\Lambda_1^{(t)}] + \sum_{\underline{x}\in\mathcal{X}_u}\sum_{k\in\mathcal{C}_{\text{pre}}} \text{Prob}\,[\mathcal{M}_k \mid \underline{x}\in\mathcal{X}_u;\Lambda_1^{(t)}]}.$$

3.1.5 Pseudocode for Overall Iterative Optimization

Our iterative optimization over the full parameter set Λ_1, which performs hillclimbing in the data likelihood, is summarized in pseudocode as shown below:

11. It is straightforward to modify the M-step development here for other continuous feature models, as well as for categorical feature models, e.g., a naive Bayes model [28]. The choice of multivariate Gaussians is mainly for illustration.

Initialize $v_k = 1\ \forall k$ and initialize Γ to $\Gamma_{(0)}$; $n \leftarrow 0$.
Do

Cycle over the $\{v_k\}$, one by one, selecting each variable in turn to maximize $\log L_m$, given the remaining ones held fixed. Stop when there are no further changes. The resulting configuration is denoted $\{v_k\}_{(n+1)}$.

Perform the EM algorithm (alternating E-steps and M-steps), starting from $\Lambda_{(n)} = \{\Gamma_{(n)}, \{v_k\}_{(n+1)}\}$. Stop when a convergence criterion is met. The new parameter estimates are stored in $\Lambda_{(n+1)} = \{\Gamma_{(n+1)}, \{v_k\}_{(n+1)}\}$.

$n \leftarrow n+1$
While not converged

Comments:

1) The subscript n counts the number of iterations. This counter is distinct from the superscript (t), which counts the number of (inner) EM steps taken for each execution of the EM algorithm.
2) Note that initially we choose $v_k = 1 \forall k$. This choice is in some sense least biased since it is difficult to have any a priori knowledge of which components are "nonpredefined."
3) Two convergence criteria are used, one for terminating the (inner) EM algorithm, with the other terminating the overall algorithm. Our choice of criteria, along with our choice for $\Gamma_{(0)}$, will be indicated in Section 6.
4) If a component switches from $v_k = 1$ to $v_k = 0$, its predefined parameters $\{\text{Prob}[C=c|\mathcal{M}_k,\mathcal{L}=''\,\text{l}''\,]\}$ are held static and saved for use in the optimization of component natures. If the component later switches back to $v_k = 1$, the EM algorithm will update these parameters starting from their current, saved values.
5) If all components are constrained to be predefined, both the model and learning (now pure EM) specialize to those in [25].

3.2 Learning for Model II

There are two learning approaches that we have investigated for model II. One of these, consistent with the learning for model I, involves a two-way partitioning of the parameter space, with the other based on a three-way partitioning. We first describe the approach based on a two-way partition. We will not present the learning approach in full detail. However, the main points are as follows:

1. The learning strategy is the same as for model I, achieving hillclimbing in (2), based on alternating $\{v_k\}$ and EM optimizations.
2. The E-step equations for model II are given by:

$$\text{Prob}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_l;\Lambda_2^{(t)}] = \begin{cases} \frac{\alpha_k^{(t)} f(\underline{x}|\theta_k^{(t)})\text{Prob}[C=c(\underline{x})|\mathcal{M}_k]^{(t)}}{\sum_{n\in\mathcal{C}_{\text{pre}}} \alpha_n^{(t)} f(\underline{x}|\theta_n^{(t)})\text{Prob}[C=c(\underline{x})|\mathcal{M}_n]^{(t)}} & \mathcal{M}_k \in \mathcal{C}_{\text{pre}} \\ 0 & \text{else.} \end{cases} \tag{12}$$

$$\text{Prob}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_u;\Lambda_2^{(t)}]=$$
$$\begin{cases}\dfrac{\alpha_k^{(t)} f(\underline{x}\mid\theta_k^{(t)})\sum_c \text{Prob}\,[C=c\mid\mathcal{M}_k]^{(t)}\,\text{Prob}\,[\mathcal{L}=''\text{m}''\mid\mathcal{M}_g\in\mathcal{C}_{\text{pre}},C=c]^{(t)}}{\sum_n \alpha_n^{(t)} f(\underline{x}\mid\theta_n^{(t)})(1+v_n\sum_c \text{Prob}\,[C=c\mid\mathcal{M}_n]^{(t)}\,\text{Prob}\,[\mathcal{L}=''\text{m}''\mid\mathcal{M}_g\in\mathcal{C}_{\text{pre}},C=c]^{(t)})} & \mathcal{M}_k\in\mathcal{C}_{\text{pre}}\\ \dfrac{\alpha_k^{(t)} f(\underline{x}\mid\theta_k^{(t)})}{\sum_n \alpha_n^{(t)} f(\underline{x}\mid\theta_n^{(t)})(1+v_n\sum_c \text{Prob}\,[C=c\mid\mathcal{M}_n]^{(t)}\,\text{Prob}\,[\mathcal{L}=''\text{m}''\mid\mathcal{M}_g\in\mathcal{C}_{\text{pre}},C=c]^{(t)})} & \text{else.}\end{cases} \tag{13}$$

3. The M-step equations for $\{\alpha_k\}$, $\{\underline{m}_k\}$, and $\{\Sigma_k\}$ are the same as for model I, given in (7), (8), and (9), but now based on the probabilities given in (12) and (13).
4. However, the partial derivatives for $\{\text{Prob}[C=c|\mathcal{M}_k]\}$ and $\{\text{Prob}[\mathcal{L}=''\text{m}''|\mathcal{M}_g\in\mathcal{C}_{\text{pre}},C=c]\}$ are now coupled.[12] A closed form M-step for these parameters is not possible. Instead, a local optimization technique, such as gradient descent, must be used. The approach we have taken involves first parameterizing these quantities (without sacrificing representational power) using soft-max functions [4], in order to ensure they remain probabilities throughout the local optimization. In particular, we let $\text{Prob}[C=c|\mathcal{M}_k]=\frac{e^{\lambda_{c,k}}}{\sum_{c'}e^{\lambda_{c',k}}}$ and
$$\text{Prob}[\mathcal{L}=''\text{m}''|\mathcal{M}_g\in\mathcal{C}_{\text{pre}},C=c]=\frac{e^{\gamma_c}}{1+e^{\gamma_c}},$$
where $\{\lambda_{c,k}\}$ and $\{\gamma_c\}$ are real-valued parameters. We then perform gradient descent on the expected complete data likelihood surface with respect to $\{\lambda_{c,k}\}$ and $\{\gamma_c\}$.[13] The overall M-step for model II thus consists of 1) closed form updates for $\{\alpha_k\}$, $\{\underline{m}_k\}$, and $\{\sum_k\}$, followed by 2) gradient descent to *convergence* in $\{\lambda_{c,k}\}$ and $\{\gamma_c\}$. The need for local optimization within the M-step can be viewed as the "price" associated with model II's more complex stochastic generative model.

While Cases 1-4 outline a learning procedure based on a partitioning of the parameter set into two subsets, an alternative is to base the learning on a *three-way* partition: 1) $\{v_k\}$, 2) $\{\{\lambda_{c,k}\},\{\gamma_c\}\}$, and 3) $\Lambda_2-\{v_k\}-\{\{\lambda_{c,k}\},\{\gamma_c\}\}$. In this case, an iteration consists of three successive steps: 1) $\{v_k\}$ optimization as before; 2) EM learning of Λ_2 - $\{v_k\}$ - $\{\{\lambda_{c,k}\},\{\gamma_c\}\}$;3) Gradient descent on the *incomplete* data likelihood (2) with respect to $\{\{\lambda_{c,k}\},\{\gamma_c\}\}$. The potential advantage here is that since step 3) optimizes (with respect to parameter subset 2) *directly* over (2) rather than over the auxiliary function used by EM, convergence is expected to be faster, see e.g., [24]. We will discuss the convergence and performance of both learning procedures for model II in the results section.

4 Statistical Inferences from the Models

Given the learned models, we can form a posteriori probabilities to directly address the inference tasks of 1) classification to one of the known classes and 2) identification of samples from unknown classes/sample rejection. We also consider transductive inference in this section.

12. For concision, we do not include these equations here.

13. The gradient descent equations are straightforward to develop and are omitted for concision.

4.1 Classification to a Known Class

For task 1), we require evaluation of the a posteriori known class probabilities. For model I, these probabilities are given (via Bayes rule) by:

$$\text{Prob}[C=c|\underline{x};\Lambda_1]=\frac{\sum_{k\in\mathcal{C}_{\text{pre}}}\alpha_k f(\underline{x}|\theta_k)\text{Prob}[C=c|\mathcal{M}_k,\mathcal{L}=''l'']}{\sum_{k\in\mathcal{C}_{\text{pre}}}\alpha_k f(\underline{x}|\theta_k)},\quad\forall c\in\mathcal{P}_c. \tag{14}$$

Likewise, for model II, we have:

$$\text{Prob}[C=c|\underline{x};\Lambda_2]=\frac{\sum_{k\in\mathcal{C}_{\text{pre}}}\alpha_k f(\underline{x}|\theta_k)\text{Prob}[C=c|\mathcal{M}_k]}{\sum_{k\in\mathcal{C}_{\text{pre}}}\alpha_k f(\underline{x}|\theta_k)},\quad\forall c\in\mathcal{P}_c. \tag{15}$$

These probabilities form the basis for maximum a posteriori (MAP) classification rules.

4.2 Identifying Unknown Class Data

For task 2), we need the a posteriori probability that a given unlabeled feature vector is generated by a nonpredefined component. For both models I and II, this is simply

$$\text{Prob}[\mathcal{M}_g\in\bar{\mathcal{C}}_{\text{pre}}|\underline{x}\in\mathcal{X}_u]=1-\sum_{k\in\mathcal{C}_{\text{pre}}}\text{Prob}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_u;\Lambda], \tag{16}$$

where $\text{Prob}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_u;\Lambda]$ is given in (5) and (13), respectively. In addition to helping identify unknown class data, (16) also forms the basis for sample rejection when the goal is classification to one of the known classes.

4.3 Transductive Inference

In this case [26], [20], the test set/new samples to classify $\mathcal{X}_{\text{te}}=\{\underline{x}_i\}$ that augment the existing training set $\mathcal{X}_{\text{tr}}$ are *purely* unlabeled. Suppose the training set $\mathcal{X}_{\text{tr}}$ is purely labeled.[14] In this case, the *only* unlabeled data comes from $\mathcal{X}_{\text{te}}$. Thus, our (mixed data) modeling approach can only be used if the missing labels from $\mathcal{X}_{\text{te}}$ are *treated* as missing at random (under the hypothesis of a predefined generating component), even though they are actually deterministically missing.[15] To do so, we let $\mathcal{X}_u=\mathcal{X}_{\text{te}}$, form the mixed data set $\mathcal{X}_m=\{\mathcal{X}_{\text{tr}},\mathcal{X}_{\text{te}}\}$, and learn our mixture model in the usual way (Sections 3.1 and 3.2). Consider, for example, learning model I. The learning will produce the probability $\text{Prob}[\mathcal{L}=''m''|\mathcal{M}_g\in\mathcal{C}_{\text{pre}}]\in[0,\frac{|\mathcal{X}_{\text{te}}|}{|\mathcal{X}_{\text{te}}|+|\mathcal{X}_{\text{tr}}|}]$. At the two extremes, the probability will be zero if all of $\mathcal{X}_{\text{te}}$ is owned by nonpredefined components and will be $\frac{|\mathcal{X}_{\text{te}}|}{|\mathcal{X}_{\text{te}}|+|\mathcal{X}_{\text{tr}}|}$ if all of $\mathcal{X}_{\text{te}}$ belongs to predefined components. Given the learned model, (14), (15), and (16) are used to address inference tasks involving $\mathcal{X}_{\text{te}}$.

14. We have also considered the case where $\mathcal{X}_{\text{tr}}$ is *mixed* labeled/unlabeled. This leads to somewhat different learning and inference than for the case where $\mathcal{X}_{\text{tr}}$ is purely labeled. We omit the learning and inference description for this alternate case, for the sake of brevity.

15. More precisely, under the hypothesis of a predefined generating component, label presence/absence is treated as a random variable for each sample in the combined set $\mathcal{X}_m=\{\mathcal{X}_{\text{tr}},\mathcal{X}_{\text{te}}\}$.

5 Model Selection and Class Discovery

While the learning in Section 3 assumed the number of components M is fixed and known, in practice, M must be estimated. Model order selection is a difficult and pervasive problem. Several techniques have been proposed to estimate the number of components in a mixture, e.g., [23], [12]. There is also a vast literature on the related problems of estimating the number of clusters in data and of choosing between clustering solutions (cluster validation), e.g., [18]. Here, our aim is not to propose an optimal technique. Rather, we simply propose a practical (heuristic) strategy in order to demonstrate that effective model order selection, yielding good mixture solutions, can be achieved for our mixed data framework. Determination of an optimal or near-optimal strategy could be the subject of future work.

The procedure we suggest is as follows: First, our learning algorithm is run, based on a size M chosen either to *overestimate* the true number of components,[16] or based on computational resource limits (associated with model learning and inference). Next, we implement a *model order reduction procedure*, with the number of components reduced one-by-one. The reduced component is the one whose removal (followed by rerunning the learning to fine-tune the model) yields the greatest decrease/smallest increase in a performance function consisting of the negative data log likelihood plus a model complexity penalty function. The model is reduced in this way, all the way down to a single component. The model in this sequence with the smallest value of the performance function is retained as the "validated" model. The nonpredefined components (and the samples they own) in this validated solution can be taken as candidate new classes. There are several possible choices for the performance function. We have chosen the *minimum description length* (MDL) criterion from [31]. This criterion function is given by:

$$\mathcal{L}_{\text{MDL}} = \frac{1}{2}\left(\sum_{m=1}^{M} P_m\right)\log N - \log L_m, \tag{17}$$

where N is the data set size and P_m is the number of parameters specifying component m (which differs for predefined and nonpredefined components).

6 Experimental Results

6.1 Description of Methods

For purpose of comparison, we have implemented three other methods in addition to the two models we have developed. One of these three methods we dub "Supervised Clustering." This approach is loosely based on the work in [19]. Here, we first perform standard mixture modeling *using only the labeled data*. Each mixture component is then deterministically associated with one of the classes. This class is chosen by first measuring the fraction of the component's probability mass that comes from each of the known classes. The component is then hard-assigned to the class with maximum probability. For classification to a known class, the components are then used to form a nearest-prototype classifier. For determining samples that do not belong to known classes (known versus unknown class discrimination), an unlabeled sample is declared to be from an unknown class if it is deemed an outlier with respect to *each* of these (predefined) components. This is determined by thresholding the component's density, evaluated at the sample. The threshold is chosen as follows: There are two types of errors: 1) a point that originates from a known class declared to be an outlier with respect to every predefined component; 2) a point from an unknown class not identified as being an outlier with respect to every predefined component. The threshold is chosen to make these two error types equally likely.

The second method used for comparison is based on [25]. This method is similar to ours in that it incorporates unlabeled data when learning a classifier; however, there is no explicit modeling of (some) unlabeled data as coming from unknown classes. In order to use [25] for known versus unknown class discrimination, each mixture component is deemed "predefined" if it "owns" any labeled samples; else, the component is deemed "nonpredefined."[17] Known versus unknown class discrimination is thus performed similar to our method, based on (16), but with $P[\mathcal{M}_k|\underline{x} \in \mathcal{X}_u]$ as given in [25], rather than based on (5). Finally, in some experiments, we compare against *standard* finite normal mixtures (SFNMs), also trained via EM. This approach only models the feature vector, i.e., the class label information is discarded. Note that these three baseline methods represent three fundamental alternatives to our approach: 1) use only the labeled data, 2) use all the data ([25]), and 3) *ignore* the labels in learning a mixture.

Since EM is used by all the methods, a common EM stopping condition was applied—termination when the change in likelihood from one iteration to the next fell below 10^{-5}. For our methods, iterative cycling was used to choose the $\{v_k\}$. Also, our (outer) algorithm was terminated when no component natures changed from one iteration to the next. For all the methods, assuming Gaussian density functions, component means were randomly initialized within the data, component masses were initialized uniformly, main diagonal covariance matrix elements were initialized with a common value, and off-diagonal entries were constrained to be zero. Since all the methods are based on mixture models, we used the same (MDL-based) model selection approach, as described in Section 5, for each of the methods.

6.2 Results

6.2.1 Mixture-Based Density Estimation

The objective of our initial experiments was to empirically explore two fundamental questions: 1) Is there information in label presence/absence that helps to improve mixture density estimation accuracy (problem 4 from Section 1)? 2) Since our model and its learning (based on local optimization) are more complex than those for SFNMs and [25], is our method more sensitive to initialization, i.e., to finding poor local optima? *Illustrative Example:* Fig. 2 displays solutions for model I (Fig. 2a), the method from [25] (Fig. 2b), and SFNMs (Fig. 2c), for a seven-component data set, 80 samples per (isotropic) component, with three

16. One approach is to set a minimum average component mass (number of points owned), N_c, and then let $M = \lfloor \frac{N}{N_c} \rfloor$. The initialization of the component means prior to running EM can be obtained by randomly sampling from the data set $\mathcal{X}_m$.

17. A mixture component is said to "own" a sample if the membership probability of the sample with the component is greater than 50 percent.

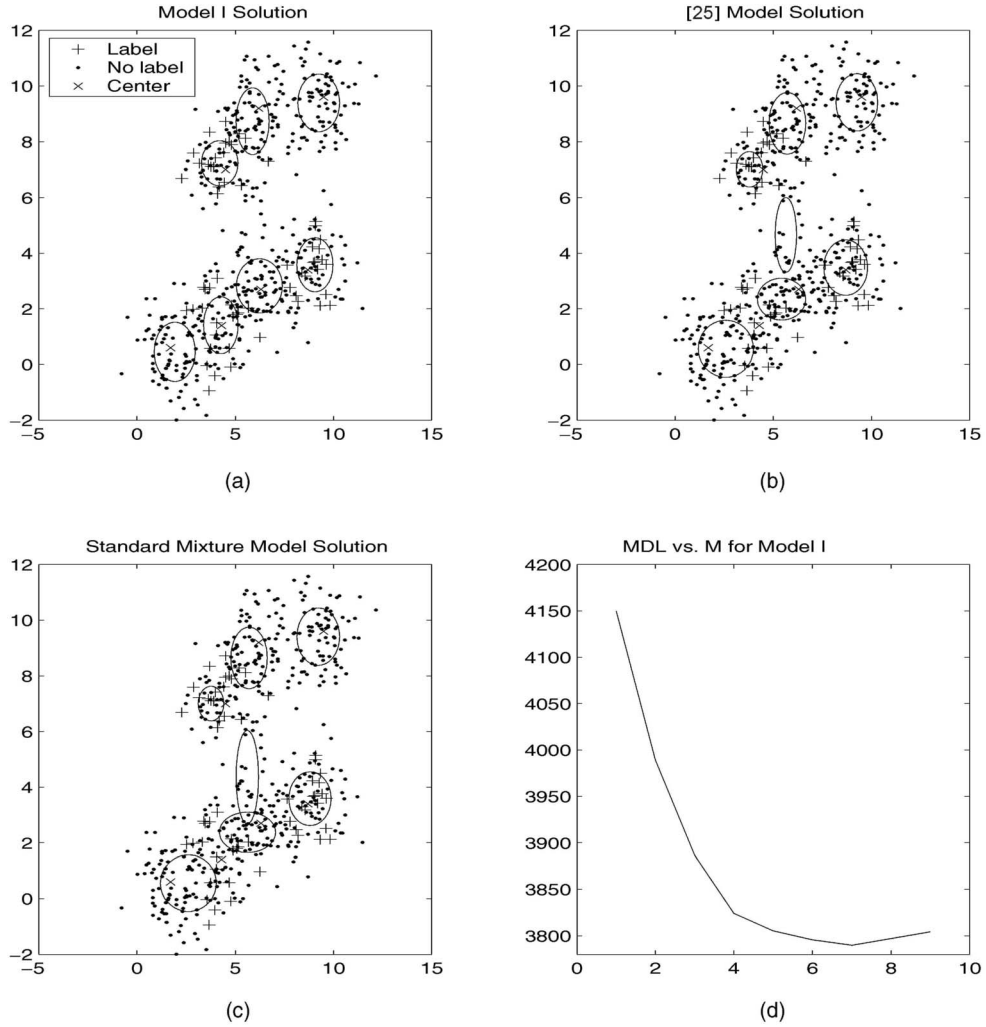

Fig. 2. (a), (b), and (c) Mixture solutions for a two-dimensional synthetic data set with seven components. Xs indicate true means. The ellipses are drawn at one standard deviation from the learned component means. (d) MDL cost versus model order for model I.

known classes (each represented by an individual component) and with four components representing unknown classes. Twenty-five percent of the data from known classes was labeled. For all the methods, we fixed the number of components to the true number, seven, and initialized means, variances, and masses in an identical fashion (variances and masses initialized to the true values). The initialization was not necessarily a very good one, with one component mean chosen near (6, 5), distant from *all* of the true means. For [25] and SFNMs, this initialization resulted in the poor solutions indicated in the figure, with the component initialized near (6, 5) effectively "trapped" at this position. By contrast, our method found an excellent solution, one that accurately discerns all the underlying components. Fig. 2d further indicates that, starting from a particular random initialization with nine components and using our model selection strategy, the MDL cost associated with our model does have a minimum at the true order (with the solution comparable to that in Fig. 2a).

This example is *not* intended to suggest our method does not suffer from sensitivity to initialization. However, it does suggest that our method does not always suffer from the *same* local optimum traps as other methods. Further, it is suggestive that label presence/absence data may have value for density estimation. This raises the question of whether this effect is reproducible. To explore this, we performed experiments involving both numerous random initializations for the data in Fig. 2, as well as for a variety of other randomly generated data sets. To compare the three methods quantitatively (keeping in mind that SFNMs ignore the class labels), we used a "component label error" measure, computed as follows: First, after learning all three models, we only *retain* the parameters (for our method and [25]) needed to specify an SFNM, i.e., parameters associated with the class label information are discarded. Thus, we have three SFNMs, each learned by a different method. Then, for each model, for each learned component, we identify the nearest *true* component (based on Euclidean distance between component means) and assign to the learned component the index of this true component. Thus, each learned component is effectively "owned" by a true component. Each data sample is then classified, via an SFNM MAP rule, to one of the learned components and, thus, via the index assignment, to one of the true components. If the predicted true component is not the one which actually generated the sample, an error is counted. Using this measure, the model I error rate in Fig. 2 was .127, while the error rates were .241 and .248 for

TABLE 1
Fraction of Incorrect Known versus Unknown Class Decisions for Synthetic Data, Evaluated for 5 Percent, 25 Percent, 50 Percent, and 75 Percent of the Known Class Data Labeled

Method	5	25	50	75
Model I	.095	.049	.044	.033
Model II	.143	.061	.057	.047
Supervised clustering method	.212	.16	.148	.144
Method from [25]	.221	.32	.440	.543

TABLE 2
Fraction of Incorrect Classifications on the Synthetic Data, Evaluated for 5 Percent, 25 Percent, 50 Percent, and 75 Percent of the Known Class Data Labeled

Method	5	25	50	75
Model I	.121	.063	.052	.037
Model II	.219	.075	.066	.052
Supervised clustering method	.231	.17	.155	.147
Method from [25]	.250	.343	.451	.549

[25] and SFNMs, respectively. To explore reproducibility of these results for the data in Fig. 2, we computed *average* error rates and standard deviations based on 100 random initializations. The (mean, standard deviation) error rate pairs were (.192, .055), (.253, .050), and (.246, .049) for model I, [25], and SFNMs, respectively. The performance difference between model I and the other methods is substantial. We then considered the performance when averaging over both 100 initializations *and* 100 data sets.[18] The (mean, standard deviation) error rate pairs were (.253, .058), (.264, .054), and (.268, .054) for model I, [25], and SFNMs, respectively. We draw two main conclusions from these experiments. First, both class labels *and* label presence/absence appear to be helpful in mixture density estimation—[25], which uses the available class labels, performed somewhat better than SFNMs, while model I, which uses *all* the available categorical information, outperformed both SFNMs and [25]. Moreover, the relative gain (drop in the error rate from 0.264 to 0.253) by using label presence/absence is more substantial than the gain from using the class labels (a drop from 0.268 to 0.264). One explanation for this is that label presence/absence is always observed, while class labels were observed only 11 percent of the time.[19] On some data sets (e.g., Fig. 2), the average performance gain of model I was quite substantial (on some data sets, model I was also, on average, worse). The second conclusion is that model I *does* appear to be more sensitive to initialization than the other methods. This is indicated by the fact that the model I error rate has the largest standard deviation. However, the average performance is better, which recommends label presence/absence and our model for density estimation when given a mixed data set and unknown classes. Moreover, there are several methods for avoiding local optima in MLE (e.g., simulated annealing, [36], [37]) that could be applied to better learn our mixture models.

6.2.2 Classification and Class Discovery Experiments

We used both synthetic and real-world data sets for evaluation. First, we consider tests using the synthetic data. These data sets had precisely the same characteristics as those used in the previous experiments, with seven components in two dimensions. Three of the components represent known classes and four represent unknown class data. We randomly generated 30 such data sets. Trials were done with 5 percent, 25 percent, 50 percent, and 75 percent of the data from known classes labeled, with all the data (320 points) from unknown classes unlabeled. For these initial experiments, the random missing label mechanism was independent of the class label. For all the methods, we used model selection and chose $M_{\max} = 9$. The results for known versus unknown class discrimination are shown in Table 1, with classification results (over the known classes) shown in Table 2. In both cases, the performance was measured over the unlabeled data subset (effectively, this amounts to transductive inference). For the classification experiments, a two-step process was invoked, with known versus unknown discrimination first applied and, then, for those samples deemed known, classification to one of the known classes. If either step is incorrect, an error is counted. All the results represent averages over the 30 data sets.

In all cases, the new methods fared best, with Supervised Clustering next in performance. In most of the experiments, we found that model selection for our models found the true size ($M = 7$) or underestimated the order by one. We have also observed that model selection generally finds better solutions at the selected size than those obtained by *directly* learning a model of this size (i.e., without going through model pruning to reach this size). This is not surprising, since the model selection procedure prunes poor components (those that contribute least to the data likelihood), thus providing some robustness to (poor) initializations. With the exception of [25], the performance of all the methods worsens as the labeled fraction of known class data is decreased. This can be understood from the standpoint that predefined and nonpredefined components become more similar (less discriminable) as this labeled fraction is decreased. The poor performance of [25] can be explained as follows: With a small number of true components in the data and, hence, with a small model size, most (or all) of the mixture components learned by [25] are likely to end up owning *some* labeled data. Thus, [25] may not find any nonpredefined components, in which case all the data will be identified as coming from known classes. The likelihood of this event should increase with the percentage of labeled data. This explains the (contrary) behavior of this method as the percentage of labeled data is increased. The performance of [25] could potentially be improved by altering the rule it uses to define nonpredefined components—"nonpredefined" could be declared if the number of labeled samples owned by a component is less than a given (*optimized*) threshold. However, a tedious trial and error optimization would be required to properly choose this threshold (and the threshold would need to change as a function of the labeled data fraction). We did not perform this optimization for [25] in our experiments.

18. Each 2D data set had seven components, three from known classes and four from unknown classes, with variances equal to one and equal masses. The component means were randomly selected from within a 10×10 box. Twenty-five percent of the data from known classes was labeled.

19. The known class data comprises 43 percent of the data, with 25 percent of these samples labeled.

TABLE 3
Fraction of Incorrect Known versus Unknown Class Decisions for the *Vowel* Data, Evaluated for 5 Percent, 25 Percent, 50 Percent, and 75 Percent of the Known Class Data Labeled

Method	5	25	50	75
Model I	.247	.142	.155	.158
Model II	.439	.255	.205	.117
Supervised clustering method	.385	.467	.451	.408
Method from [25]	.223	.330	.393	.330

TABLE 4
Fraction of Incorrect Classifications on the *Vowel* Data, Evaluated for 5 Percent, 25 Percent, 50 Percent, and 75 Percent of the Known Class Data Labeled

Method	5	25	50	75
Model I	.474	.353	.327	.227
Model II	.538	.432	.358	.188
Supervised clustering method	.592	.611	.586	.496
Method from [25]	.472	.554	.551	.421

We also note that the performance gap between methods does not really decrease with an increasing fraction of labeled data from known classes. This difference *would* shrink if the amount of *unknown* class data were decreased, but this amount was held fixed in these experiments. Finally, we note that the results for model II were obtained using learning based on the two-way parameter partition (Section 3.2). The second method, based on a three-way partition, was found to be faster—on a Sparc10 machine, 3 minutes, 25 seconds versus 1 minute, 27 seconds to learn models for 20 synthetic data sets (20 iterations for each model). However, the learned models achieved quite comparable classification accuracy.[20]

The second data set is Deterding's 10-dimensional *vowel* set, with features consisting of log area ratios, derived from linear predictive coding coefficients. Again, the features were modeled by Gaussian densities. This set has 990 samples, consisting of 11 vowels, where we used the first six as known classes and the last five as unknown classes. In this case, we chose $M_{\max} = 40$. Known versus unknown discrimination and classification results for this data set are shown in Tables 3 and 4. In this case, Supervised Clustering fared the poorest. The method based on [25] performed (relatively) better on this data set. We believe this is due to the larger model size, which allowed this method to find nonpredefined components. Note that [25] is even slightly better than model I when only 5 percent of the data is labeled. This may be attributable to model I's strong ability to find nonpredefined components coupled with the fact that, for *vowel*, with each of the 11 classes possibly consisting of multiple components, some true underlying components from *known* classes may be purely unlabeled when only 5 percent of the known class data is labeled.[21] Finally, note that model II performance is quite poor for the 5 percent case—the amount of labeled data in this case appears to be inadequate to well-support a class-conditional missing mechanism.

Performance of Model II: In Tables 1, 2, 3, and 4 it is seen that model II usually performs somewhat worse than model I. This is not surprising since model II is tailored for the case where the missing label mechanism is *class-dependent*, whereas this mechanism was independent of the class for the experiments considered until now. To demonstrate the potential advantage of model II, we created new data sets with the missing label mechanism now class-dependent. For the synthetic case, we used the same 30 data sets as before, but with 25 percent labeled for the first class, 50 percent labeled for the second class, and 75 percent labeled for the third class. For the *vowel* data, we chose 25 percent labeled for the first two classes, 50 percent labeled for the second two classes, and 75 percent labeled for the third two classes. The results in Tables 5 and 6 demonstrate an overall advantage of model II for these data sets, as expected. The performance of [25] is particularly poor. Again, we expect this performance would improve if the threshold used to declare nonpredefined components were optimized for these experiments.

Global optimization versus local optimization of $\{v_k\}$*:* In order to test the effectiveness of our method for cycling over component natures, we compared with global optimization (exhaustive search) on the *vowel* data for the case of 50 percent of the known class data missing labels. We learned model I with 19 mixture components, using both global optimization and iterative cycling. We found that the results were identical. Moreover, for synthetic data sets with seven components, we have found that global optimization and iterative cycling often yield the same results.

Varying the Number of Unknown Class Components: Table 7 gives classification results, averaged over 30 synthetic data sets, each with three known class components, as the number of unknown class components is reduced from four down to one. Compared with [25], as one would expect, the performance advantage of our method shrinks as the number of unknown class components is reduced. For both model I and [25], the trend is a decreasing error rate as the number of components is reduced, with an opposite trend for Supervised Clustering. We believe these trends can be understood as follows: Our method and [25] both model the unknown class data. The difficulty of this modeling problem decreases as the number of unknown class components is reduced. Thus, the error rate decreases. Supervised Clustering, on the other hand, does not model the unknown class data. Moreover, the threshold used by Supervised Clustering to determine whether a sample is an

TABLE 5
Fraction of Incorrect Known versus Unknown Class Decisions for the Synthetic and *Vowel* Data with a Class-Dependent Missing Mechanism

Method	Synthetic	Vowel
Model I	.142	.193
Model II	.051	.181
Supervised clustering method	.153	.416
Method from [25]	.413	.481

20. The second method was inspired by a reviewer's comment, made on our initial paper submission.

21. If these underlying components were (ground truth) known, and if they were treated for performance evaluation purposes as *unknown* classes (which is reasonable, since they are purely unlabeled), model I might perhaps outperform [25] for the 5 percent case. Unfortunately, this hypothesis cannot be tested since the true underlying components are unknown for the *vowel* data.

TABLE 6
Fraction of Incorrect Classifications on the Synthetic and *Vowel* Data with a Class-Dependent Missing Mechanism

Method	Synthetic	Vowel
Model I	.142	.343
Model II	.059	.344
Supervised clustering method	.159	.498
Method from [25]	.425	.698

TABLE 7
Fraction of Incorrect Classifications on Data Sets with Varying Numbers of Unknown Class Components

Method	1	2	3	4
Model I	.0475	.044	.056	.063
Supervised clustering method	.244	.205	.1875	.170
Method from [25]	.164	.261	.332	.340

outlier was held fixed for these experiments. Based on this threshold, we observed that the error rate for samples from unknown classes was lower than for samples from known classes. As the number of unknown class components is reduced, the fraction of the unlabeled data that comes from known classes (with the higher error rate) increases. Thus, the average error rate (measured over the unlabeled data from *both* known and unknown classes) increases.

Performance with Data Exclusively from Known Classes: We also investigated whether our model would declare the existence of unknown class data/components in situations where all the data originates from known classes. This was tested on 30 synthetic data sets randomly generated as before, with the fraction of labeled samples taken to be 25 percent. We investigated two cases, one where the model size was fixed to the true size ($M = 7$), and the other where the model selection strategy was used. With a fixed model size, our method did exhibit a tendency to falsely declare nonpredefined components—on average, 1.7 out of seven components were declared nonpredefined. By contrast, [25] did not falsely declare any nonpredefined components. However, this tendency for our method was found to be almost entirely suppressed by the use of model selection. For this case, taken over all 30 data sets, our method found only one nonpredefined component. For [25], again, no nonpredefined components were found. One explanation for the model I results is that the model selection procedure did, in these experiments, tend to underestimate the order (to either six or five components). This makes it less likely to find nonpredefined components. For the model selection case, the average classification error results for model I and [25] were nearly identical.[22]

New Class Discovery on Reuters: As an initial investigation of new class discovery, we considered the *Reuters* news articles database, with 21,578 articles, labeled by 135 possible topics. We considered the 24 most popular topics. For each such topic, we took the 50 longest articles, forming a data set of 1,200 articles. The first 12 topics were treated as known classes, with the remaining topics unknown classes. Specifically, the known topics, labeled 1-12, were: *acq*, *alum*, *bop*, *cocoa*, *coffee*, *ipi*, *crude*, *earn*, *interest*, *gnp*, *gold*, and *grain*. The unknown topics, labeled 13-24, were: *ironsteel*, *jobs*, *livestock*, *moneyfx*, *moneysupply*, *oilseed*, *reserves*, *ship*, *sugar*, *trade*, *veg oil*, and *wheat*. These 24 topics correspond, respectively, to the 24 marked positions on the x-axis in the four histograms in Fig. 3. We labeled 50 percent of the articles from known topics, with all the unknown topic articles unlabeled.

Instead of a Gaussian model, the Bernoulli form of naive Bayes [22] was used as the probability model for the features (which were presence/absence indications for each word in a word list). The word list was based on 10,000 *Reuters* articles. Using a stop list of 293 common words and stemming, 10,302 words were found. We attempted to use the MDL-based model selection for this experiment but found that it did not yield meaningful results (the MDL cost *strictly* decreased with M). This was expected since the feature space (over 10,000) and, hence, the penalty on the model size, was very large. Instead, we simply chose a model with 50 components (roughly two per topic). Each of the 50 components was initialized with word probabilities based on random perturbation of the global word probabilities (measured over all 1,200 articles). Learning for our model I was then performed. This learning was based on the approach in Section 3, but modified in a simple fashion to optimize the naive Bayes word probabilities, rather than Gaussian parameters. This learning took about 30 minutes on a Sun Ultrasparc 10. Once the model was learned, known versus unknown class discrimination performance was measured on the unlabeled data subset, as done previously. The error rate was 27.9 percent.

As a second performance measure, we counted up the number of unknown topics "discovered" by the model. This was done as follows: For each nonpredefined component, we determined the probability mass in each of the 24 topics. This was measured by summing up the probabilities (5) associated with all articles from the same topic. We then *labeled* each component by the topic with maximum probability. Finally, we counted up the number of distinct unknown topics "covered" by the labels of these nonpredefined components. In our experiment, 11 out of the 12 unknown topics were "discovered" in this fashion. Only the topic *reserves* was not found. We give representative histograms for four of the components in Fig. 3. The x-axis represents the 24 topic labels. Each graph shows the probability mass in each of the topics for a given component. It is clear that each of these four components 1) primarily owns articles from the unknown topics (13-24) and 2) has a strong distinct peak at a single unknown topic, i.e., it primarily owns articles from a *single* unknown topic. This primary "discovered" topic is indicated in the figure title.

7 Future Work

In future, we will aim to address other applications well-matched to our learning and inference scenarios. These include:

22. As mentioned earlier, the two models and their learning are identical when our method fails to find any nonpredefined components during learning.

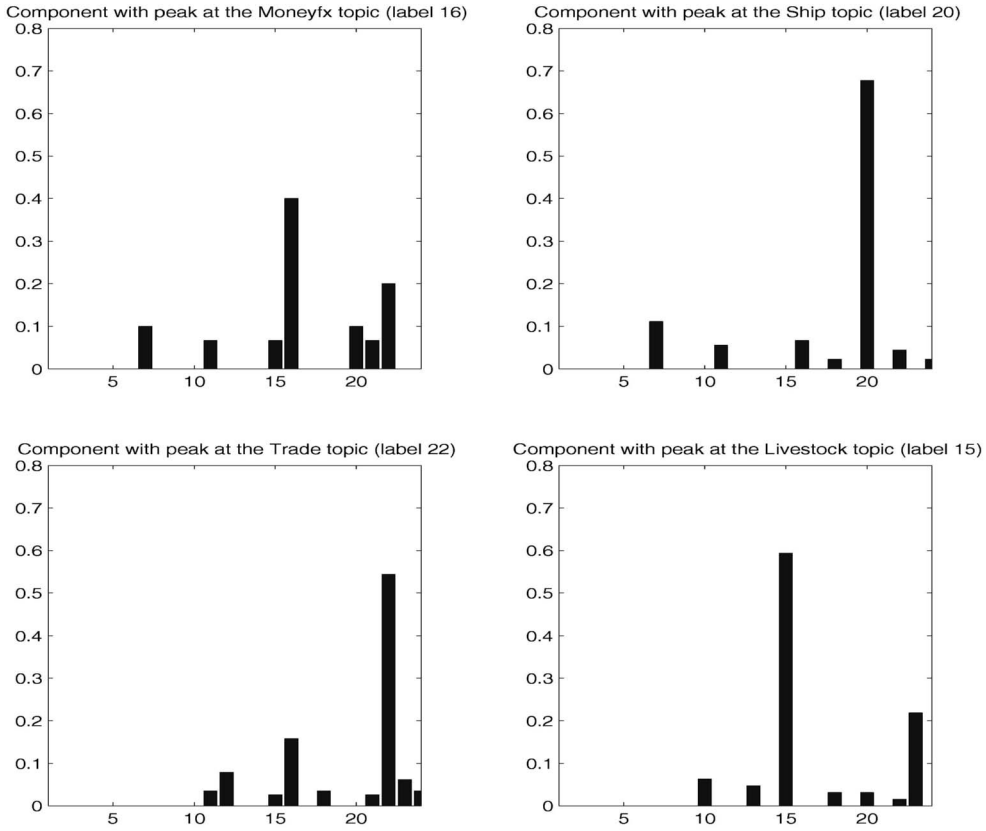

Fig. 3. Topic histograms for components with strong peaks at unknown topics. This primary (unknown) topic is indicated in the heading for each graph.

1. *Knowledge Discovery from an Internet/Database Search*, e.g., [21]: Web sites and associated data files gleaned from an Internet search often need to be categorized/organized. Moreover, data extracted in this way may be taken as a "training set" and used to build an automated classifier on some domain. Often, a search will yield some data that pertains to known categories, while other data may either be spurious *or* may correspond to a priori *unknown* categories relevant to the query. While none of the queried examples may be *explicitly* labeled, key words within URL addresses and within text may *implicitly* categorize some of the examples/sites. Thus, query-driven data may reasonably be treated as mixed labeled/unlabeled. Moreover, robust classifier design, sample rejection, and class discovery are all relevant tasks.
2. *Scientific Applications*: Here, one is interested both in assigning objects (e.g., species, natural phenomena) to known categories, as well as in discovering new ones. Progress in this area will require interaction with domain experts.

We may also pursue several extensions of our framework. One is for *active learning* e.g., [32] wherein the labeled subset is no longer fixed, but rather *grows*(as new samples are labeled). We believe a good choice of new samples to label are those from (current) *nonpredefined* components. After labeling new samples (which will effect conversion of some nonpredefined components to predefined ones), the model can then be relearned. In this active learning extension of our approach, the number of known *classes* will grow as more data is labeled. Another extension of our approach is for a scenario more like [34], where existing class definitions may be inadequate and where data may have been mislabeled. Our predefined/nonpredefined component dichotomy could be useful in this context as well, with "predefined" components owning samples that possess valid labels and with "nonpredefined" components owning mislabeled samples. Unlabeled data may also be present in this scenario. Another research avenue is to provide theoretical support to our empirical observation that label presence/absence does have value in density estimation. A final direction is to address model selection for huge feature spaces. Effective class discovery requires accurate model order selection, but this is a difficult, largely unresolved problem for huge feature spaces such as the text domain.

ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation Award IIS-0082214.

REFERENCES

[1] A. Ben-Dor, N. Friedman, and Z. Yakhini, "Class Discovery in Gene Expression Data," *Proc. Fifth Ann. Int'l Conf. Computational Biology*, pp. 31-38, 2001.
[2] J. Besag, "On the Statistical Analysis of Dirty Pictures," *J. Royal Statistical Soc. B.*, vol. 48, pp. 259-302, 1986.

[3] A. Blum and T. Mitchell, "Combined Labeled and Unlabeled Data with Co-Training," *Proc. Conf. Computational Learning Theory*, pp. 92-100, 1998.
[4] J.S. Bridle, "Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition," *Neurocomputing: Algorithms, Architectures, and Applications*, New York: Springer-Verlag, pp. 227-236, 1990.
[5] V. Castelli and T. Cover, "On the Exponential Value of Labeled Samples," *Pattern Recognition Letters*, vol. 16, pp. 105-111, 1995.
[6] I. Chang and M. Loew, "Pattern Recognition with New Class Discovery," *Proc. Conf. Computer Vision and Pattern Recognition*, pp. 438-443, 1991.
[7] F.G. Cozman and I. Cohen, "Unlabeled Data can Degrade Classification Performance of Generative Classifiers," Hewlett Packard technical report, pp. 1-16, 2001.
[8] R. Dave, "Characterization and Detection of Noise in Clustering," *Pattern Recognition Letters*, vol. 12, pp. 657-664, 1991.
[9] A. Dempster, N. Laird, and D. Rubin, "Maximum-Likelihood from Incomplete Data via the EM Algorithm," *J. Royal Statistical Soc. B*, vol. 39, pp. 1-38, 1977.
[10] S. Dumais, J. Platt, D. Heckerman, and M. Sahami, "Inductive Learning Algorithms and Representations for Text Categorization," *Proc. Conf. Information and Knowledge Management*, 1998.
[11] J.A. Fessler and A.O. Hero, "Space-Alternating Generalized Expectation-Maximization Algorithm," *IEEE Trans. Signal Processing*, vol. 42, pp. 2664-2677, 1994.
[12] M.A.T. Figueiredo and A.K. Jain, "Unsupervised Learning of Finite Mixture Models," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, pp. 381-396, 2002.
[13] P. Hall and D.M. Titterington, "The Use of Uncategorized Data to Improve the Performance of a Nonparametric Estimator of a Mixture Density," *J. Royal Statistical Soc. B*, vol. 47, pp. 155-163, 1985.
[14] S. Haykin, *Neural Networks: A Comprehensive Foundation*. New York: Macmillan, 1999.
[15] P.J. Huber, *Robust Statistics*. New York: Wiley, 1981.
[16] M. Inoue and N. Ueda, "HMMs for Both Labeled and Unlabeled Time Series Data," *Proc. IEEE Workshop Neural Networks for Signal Processing*, pp. 93-102, 2001.
[17] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, and G.E. Hinton, "Adaptive Mixtures of Local Experts," *Neural Computation*, pp. 79-87, 1991.
[18] A.K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, N.J.: Prentice Hall, 1988.
[19] B. Jeong and D. Landgrebe, "Partially Supervised Classification Using Weighted Unsupervised Clustering," *IEEE Trans. Geoscience and Remote Sensing*, pp. 1073-1079, 1999.
[20] T. Joachims, "Transductive Inference for Text Classification Using Support Vector Machines," *Proc. 14th Conf. Uncertainty in Artificial Intelligence*, pp. 200-209, 1999.
[21] J. Larsen, L.K. Hansen, T. Christiansen, and T. Kolenda, "Webmining: Learning from the World Wide Web," *Computational Statistics and Data Analysis*, vol. 38, pp. 517-532, 2002.
[22] A. McCallum and K. Nigam, "A Comparison of Event Models for Naive Bayes Text Classification," *Proc. AAAI Workshop Learning for Text Categorization*, pp. 41-48, 1998.
[23] G. McLachlan and D. Peel, *Finite Mixture Models*. New York: John Wiley and Sons, 2000.
[24] X.-L. Meng and D. van Dyk, "The EM Algorithm—An Old Folk-Song Sung to a Fast New Tune," *J. Royal Statistical Soc. B*, vol. 59, no. 3, pp. 511-567, 1997.
[25] D.J. Miller and H. Uyar, "A Mixture of Experts Classifier with Learning Based on Both Labelled and Unlabelled Data," *Proc. Neural Information Processing Systems Conf.*, vol. 9, pp. 571-577, 1997.
[26] D.J. Miller and H. Uyar, "Combined Learning and Use for a Mixture Model Equivalent to the RBF Classifier," *Neural Computation*, vol. 10, pp. 281-294, 1998.
[27] D.J. Miller and H. Uyar, "A Generalized Gaussian Mixture Classifier with Learning Based on Both Labelled and Unlabelled Data," *Proc. Int'l Conf. Information Sciences and Systems*, pp. 783-787, 1996.
[28] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell, "Text Classification from Labeled and Unlabeled Documents Using EM," *Machine Learning*, pp. 1-34, 2000.
[29] P. Pudil, J. Novovicova, S. Blaha, and J. Kittler, "Multistage Pattern Recognition with Reject Option," *Proc. Int'l Conf. Pattern Recognition Methodology and Systems*, pp. 92-95, 1992.
[30] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*. Englewood Cliffs, N.J.: Prentice Hall, 1993.
[31] J. Rissanen, "Modelling by Shortest Data Description," *Automatica*, vol. 14, pp. 465-471, 1978.
[32] N. Roy and A. McCallum, "Toward Optimal Active Learning through Sampling Estimation of Error Reduction," *Proc. Int'l Conf. Machine Learning*, pp. 441-448, 2001.
[33] B. Shashahani and D. Landgrebe, "The Effect of Unlabeled Samples in Reducing the Small Sample Size Problem and Mitigating the Hughes Phenomenon," *IEEE Trans. Geoscience and Remote Sensing*, vol. 32, pp. 1087-1095, 1994.
[34] A. Sierra and F. Corbacho, "Reclassification as Supervised Clustering," *Neural Computation*, vol. 12, pp. 2537-2546, 2000.
[35] S. Tajudin and D. Landgrebe, "Robust Parameter Estimation for Mixture Model," *IEEE Trans. Geoscience and Remote Sensing*, vol. 38, pp. 439-445, 2000.
[36] N. Ueda and R. Nakano, "Deterministic Annealing EM Algorithm," *Neural Networks*, vol. 11, pp. 271-282, 1998.
[37] N. Ueda, R. Nakano, Z. Ghahramani, and G. Hinton, "Split and Merge EM Algorithm for Improving Gaussian Mixture Density Estimates," *Proc. IEEE Workshop Neural Networks for Signal Processing*, pp. 274-283, 1998.
[38] L. Xu, M.I. Jordan, and G.E. Hinton, "An Alternative Model for Mixtures of Experts," *Proc. Neural Information Processing Systems Conf.*, vol. 7, pp. 633-640, 1995.

David J. Miller received the BSE degree from Princeton University, Princeton, NJ, in 1987, the MSE degree from the University of Pennsylvania, Philadelphia, in 1990, and the PhD degree from the University of California, Santa Barbara, in 1995, all in electrical engineering. From January 1988 through January 1990, he was with General Atronics Corp., Wyndmoor, Pennsylvania. From September 1995 to July 2001, he was an assistant professor of electrical engineering at the Pennsylvania State University, University Park campus. He is now a tenured associate professor of electrical engineering at the Pennsylvania State University. His research interests include statistical pattern recognition, machine learning, source coding and coding over noisy channels, and image and video coding. Dr. Miller received the US National Science Foundation CAREER Award in 1996. Since 1997, he has been a member of the Neural Networks for Signal Processing Technical Committee within the IEEE Signal Processing Society. He was general cochair for the 2001 IEEE Workshop on Neural Networks for Signal Processing. He is a member of the IEEE.

John Browning received the BA degree in math from Rutgers University, New Brunswick, New Jersey, in 1982, the MSEE degree in electrical engineering from Drexel University, Philadelphia, Pennsylvania, in 1996, and the PhD degree in electrical engineering from the Pennsylvania State University in 2003. From 1982 to 1997, he was with General Atronics Corp., Wyndmoor, Pennsylvania, and since then, he has been a research and teaching assistant at Penn State. His research interests include statistical pattern recognition and signal processing.