The Pennsylvania State University

The Graduate School

Department of Electrical Engineering

# STATISTICAL MODELS FOR DATA MINING: GENERAL INFERENCES AND CLASS DISCOVERY IN LARGE DATABASES

A Thesis in

Electrical Engineering

by

John D. Browning



Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

May 2003

We approve the thesis of John D. Browning.

Date of Signature

| | |
|---|---|
| David J. Miller<br>Associate Professor of Electrical Engineering<br>Thesis Adviser<br>Chair of Committee | |
| Nirmal Bose<br>HRB Systems Professor of Electrical Engineering | |
| John Doherty<br>Associate Professor of Electrical Engineering | |
| George Kesidis<br>Associate Professor of Electrical Engineering | |
| C. Lee Giles<br>Professor of Information Science and Technology | |
| W. Kenneth Jenkins<br>Professor of Electrical Engineering<br>Head of the Department of Electrical Engineering | |

# Abstract

This thesis is about the application of statistical models to data mining. Data mining involves searching for patterns in large data sets. With the introduction of cheaper storage devices with high capacity, faster communication and increasing computer power, large databases can be searched, or 'mined' for correlations in the data. These databases can be created by business applications, biological applications, from work in astronomy, weather forecasting, natural language applications, speech recognition and many other areas. Typically these databases are much larger than traditional pattern recognition databases so that algorithms used on these databases must be able to scale with the data. A second identifying trait of data mining applications is missing and erroneous data. When this data is collected errors can occur during data entry or data can be missing, either randomly or deterministically. One advantage of statistical models is that they are based on a mathematical theory that enables a principled approach to missing/erroneous data. We investigate application of statistical models to two data mining tasks that have a lot of missing data. The first is collaborative filtering, which involves inference when most of the data is missing. The second application is a new problem, where some of the data comes from unknown classes that we have to discover. This problem is related to data clustering.

# Table of Contents

# List of Tables

# List of Figures

# Acknowledgments

I am most grateful and indebted to my thesis advisor, David J. Miller, for the large doses of guidance, patience, and encouragement he has shown me during my time here at Penn State. I am especially indebted for the financial support which he has provided to me over the years. I thank my other committee members, Dr. Bose, Dr. Doherty, Dr. Kesidis and Dr. Giles for taking the time to serve on my committee and for their insightful commentary on my work.

I would like to dedicate this thesis to my parents without whose support and encouragement I might not be here.

# Chapter 1

# Introduction

In this work we develop tools to be used for data mining. Data mining techniques are methodologies for revealing patterns in data, especially very large databases. Some common factors of such databases are 1) large size in both the number of observations and the number of variables per observation, 2) missing data, complete observations being rare, and 3) the data may contain gross inaccuracies. For example a fundamental data mining task is *collaborative filtering* (CF), wherein database records are used as examples to support automated decision making/predictions for new examples using a recommendation engine. In this case a collaborative filtering database can contain tens of thousands of users' ratings on items. In addition the items being rated could be movies, products in a store, or newspaper articles where the number of objects being rated is very large (over a thousand). It is difficult to get ratings from a user on so many objects and so most such databases have many missing ratings. Missing data can also be caused because the data has been collected automatically with no further processing. Data collected from the Internet is typical. Collecting some types of data is almost free, but labeling all of it for a pattern recognition algorithm may not be possible. The labels may not be known or there may simply be too much data to label.

The data mining tasks we examine are examples of pattern recognition or classification. The object of classification is to identify items as belonging to known categories

by examining attributes of the item. These attributes are also known as *features*. The known category an item belongs to is its *label*. An example is to classify a patient's ailment based on a set of symptoms he exhibits. Here the items are patients, the labels assigned by the pattern recognition algorithm are diseases this person could be suffering from, and the input features are symptoms i.e. temperature, pulse rate, blood pressure, diagnostic measurements, nausea. Typically a classifier is created by using a *training set*. This is a database of items with each items feature values specified and with each item labeled by the category to which it belongs. If some of the items in the training set have no labels they are *unlabeled* data.

## 1.1 Collaborative Filtering

Collaborative Filtering (CF) is one data mining task that we address here. CF differs from standard classification in two important respects. First, both the database records and the new examples will typically have many missing attribute values. Second, the attributes to be inferred or predicted for the new examples can be *any* (user-selected) items, given *any* available set of partial knowledge on the user's other preferences. In other words, the model must be designed to solve any of a huge (combinatoric) set of possible inference tasks.

Both of these aspects make the CF objective a particularly challenging one. CF has applications to Web browsing, marketing, product recommendation, candidate selection (for employment or admissions), information retrieval (e.g. text article retrieval) from databases, and to rule aggregation in expert and knowledge-based systems.

First we describe the Collaborative Filtering problem. Consider a random feature vector $\underline{F} = (F_1, F_2, \ldots, F_N)$, with $F_i \in \mathcal{A}_i$, and $\mathcal{A}_i$ the finite set $\{1, 2, 3, \ldots, |\mathcal{A}_i|\}$. In the CF context, a realization $\underline{f} \equiv (f_1, f_2, \ldots, f_N)$ represents one record (example) in the database. For concreteness, if we consider an application to predicting which Web sites a user will visit, then $N$ represents the number of Web sites, $F_i$ a particular site, $\mathcal{A}_i \in \{0, 1\}\ \forall i$, with the discrete values representing 'no visit', and 'visit', respectively. Likewise, in a product recommendation context, $F_i$ represents a particular product, again taking on a discrete set of values, e.g. $\{0, 1, 2\}$ representing 'no information', 'dislike' and 'like', respectively, with $N$ the number of products. In this case, $\underline{f}$ represents one example consumer. In CF applications, one is assumed to be given a database of examples $\mathcal{F}_d \equiv \{\underline{f_1^{(d)}}, \underline{f_2^{(d)}}, \ldots, \underline{f_T^{(d)}}\}$, where $\underline{f_t^{(d)}} \equiv (f_{t1}^{(d)}, f_{t2}^{(d)}, \ldots, f_{tN}^{(d)})$, $f_{ti}^{(d)} \in \mathcal{A}_i$. The CF inference objective is then stated as follows:

**Given**:

1) A target attribute $i^*$ to be predicted, with $i^* \in \{1, 2, \ldots, N\}$.

2) A set of $L(< N)$ *known* attributes $i_1, i_2, \ldots, i_L$.

3) A set of $K$ new (reduced dimension) examples $\mathcal{F}_n \equiv \{\underline{f_1^{(n)}}, \underline{f_2^{(n)}}, \ldots, \underline{f_K^{(n)}}\}$, where $\underline{f_k^{(n)}} \equiv (f_{ki_1}^{(n)}, f_{ki_2}^{(n)}, \ldots, f_{ki_L}^{(n)})$, $f_{ki_m}^{(n)} \in \mathcal{A}_{i_m}$.

**Goal**: Predict the values $f_{ki^*}^{(n)},\ k = 1, 2, \ldots, K$.

Example: Given 4 users $(\underline{f_1^{(d)}}, \underline{f_2^{(d)}}, \underline{f_3^{(d)}}, \underline{f_4^{(d)}})$ and their ratings on 6 movies. The possible ratings are 0-5 and an empty box indicates that the user has not seen that movie. Predict how new user $\underline{f_1^{(n)}}$ will rate Movie 6.

| | Movie 1 | Movie 2 | Movie 3 | Movie 4 | Movie 5 | Movie 6 |
|---|---|---|---|---|---|---|
| $f_4^{(d)}$ | 2 | 3 | 2 | | 2 | 3 |
| $f_3^{(d)}$ | 1 | 4 | | 3 | 0 | 2 |
| $f_2^{(d)}$ | 3 | | 1 | 2 | 4 | |
| $f_1^{(d)}$ | | 3 | 2 | | 1 | 5 |
| $f_1^{(n)}$ | | | 1 | | 2 | ? |

Several important observations can be made about this problem in general:

1) The target feature is essentially a *class* feature. Thus, specifying $i^*$ and the known attributes amounts to specifying a classification problem that needs to be solved. Also, although we have indicated the goal of inferring a *single* target attribute value for each new example, this is easily extended to the prediction of a *collection* of attribute values.

2) The number of such classification problems is *huge* (combinatoric), as one can choose both the target feature(s) and the subset of known features. Thus, off-line learning and storage of one classifier for each possible inference task is utterly infeasible[1]. Practical methods for CF must thus be versatile approaches, capable of producing inferences for *any* posed CF task *without* undue delay, i.e. by either a direct (in some sense) 'on-line' inference procedure or by 'on-line' model learning based on $\mathcal{F}_d$ *followed* by inference.

[1] Classification-based methods, discussed in the sequel, do use off-line learning to build $N$ classifiers, with each dedicated to the prediction of a single feature, given knowledge of *all* the remaining feature values. However, this approach does not build a classifier dedicated to each *inference* task, i.e. it does not account for the (many) missing features that will occur in the new (test) examples.

## 1.2 Databases with Labeled and Unlabeled Data

The second problem we address is the analysis of large data sets where much or most of the data items do not have their classes specified. These items are known as *unlabeled data*. An unlabeled data item may come from one of the classes specified in the labeled subset of the data or it may in fact come from an unknown class. One reason a class may be unknown is that an expert only labeled a random subset of the data and this random set contained no instances of the unknown class. In this scenario, the class is actually known to the expert and if a different random subset of the training set had been selected for labeling, an instance of the unknown class might have been included. Another cause for unknown classes is that the class may not have even been discovered yet. An example is from astronomy. A database of galaxies could be compiled in which some of the examples may not fit into any known types. This could constitute a totally new type of galaxy not yet discovered by astronomers.

The ability to model classes outside those known to exist in the data set confers several capabilities for knowledge discovery from the data set.

1) Robust Classification - If the data set contains outliers or data from unknown classes, this data usually has a detrimental effect on a classifier's training/design. When using unlabeled data, it is assumed that this data comes from one of the known classes. If that is not the case and the data is used to estimate model parameters for a class from which it did not originate, the accuracy of these model parameters will suffer. Our approach lessens the effect of such data by clustering the noisy data so it does not contaminate the good data. It also allows prediction for the two class problem of

determining whether an item comes from: 1) one of the a predetermined classes; or 2) one of the unknown classes or is an outlier. This can be used in conjunction with a classifier to yield more accurate predictions. If the sample is an outlier or belongs to a class that is unknown to the classifier, it can be rejected before classification is attempted.

2) Class Discovery - If some of the data comes from unknown classes, it is desirable to model the unknown classes to allow further analysis and the possibility of identifying these new classes. This is another objective of our model. Class Discovery is a combination of classification and clustering which we look at in chapter 4.

## 1.3 Statistical Models and Mixture Models

We define a statistical model as a mathematical function that is derived according to the rules of probability and that generates joint or conditional probabilities for class membership based on feature values that are input to it. In some cases the form of the model is specified; in other cases the model results from the application of a general methodology. All of the models in this thesis assume that each of the data samples is generated independently of the rest of the samples in the data set. This means that the generation of each sample can be considered separately from all of the other samples and vastly simplifies the task of statistical modeling. One particular type of model used in several places in this work is a *mixture model.* This statistical model assumes that first an index $\omega_l$ is randomly chosen with probability $P[\omega_l]$. Next the selected process emits the sample $\underline{\mathrm{x}}$ with probability $P[\underline{x}|\omega_l, \underline{\Theta_l}]$ for discrete random variables (or likelihood $f[\underline{x}|\omega_l, \underline{\Theta_l}]$ in the case of continuous random variables). The variable $\underline{\Theta_l}$ is the

vector of the parameters for the lth probability distribution. Therefore the probability of generating an element $\underline{x}$ is

$$P[\underline{x}|\underline{\Theta}] = \sum_{l=1}^{L} P[\omega_l]P[\underline{x}|\omega_l, \underline{\theta_l}] \tag{1.1}$$

The values $P[\omega_l]$ are called mixing parameters and L is the total number of mixture components. These models can be trained by maximum likelihood estimation using the Expectation Maximization (EM) algorithm [20]. Normally mixture models assume that classes and mixture components are synonymous i.e. classes deterministically own one or multiple mixture components. Once the mixture component that produced an item has been identified, its class is also known. Building on mixture models, the method in [47] (subsequently called the MU model) creates a model differing from the standard mixture model in the following ways:

1. The MU model treats $\underline{x}$, the feature vector *and* c, the class label in each training set example as random quantities that are chosen conditioned on the mixture component. With a normal mixture model $\underline{x}$ is created by a mixture component that is deterministically identified with c.

2. The MU model learns using labeled *and* unlabeled data by maximizing the joint data likelihood using the EM algorithm.

The MU model was used in chapter 4 as the starting point for development of a new mixture model used for class discovery and also as a method to compare with this new

model. Because the MU model is important in several areas of this thesis, we give a brief explanation here.

The log likelihood of the data for the MU model is

$$\log \mathcal{L} = \sum_{\underline{x_i} \in \chi_u} \log \sum_{l=1}^{L} \alpha_l f(\underline{x_i}|\underline{\theta_l}) + \sum_{\underline{x_i} \in \chi_l} \log \sum_{l=1}^{L} \alpha_l \beta_{class(\underline{x_i})|l} f(\underline{x_i}|\underline{\theta_l}). \tag{1.2}$$

Here $\alpha_l$ are the mixing parameters, $\underline{\theta_l}$ are parameters for individual probability components $f(\underline{x_i}|\underline{\theta_l})$ and $\beta_{k|l}$ is the conditional probability of the class label k given the particular mixture component indexed by l. The class label of feature vector $\underline{x_i}$ is denoted $class(\underline{x_i})$. The labeled and unlabeled data sets are $\chi_l$ and $\chi_u$, respectively.

Related to the log likelihood equation is the complete data likelihood equation

$$\mathcal{L}_c = \prod_{\underline{x_i} \in \chi_u} \prod_{l=1}^{L} (\alpha_l f(\underline{x_i}|\underline{\theta_l}))^{V_{\underline{x_i}l}} \prod_{\underline{x_i} \in \chi_l} \prod_{l=1}^{L} (\alpha_l \beta_{class(\underline{x_i})|l} f(\underline{x_i}|\underline{\theta_l}))^{V_{\underline{x_i}l}}. \tag{1.3}$$

The hidden data is the variable $V_{\underline{x_i}l}$ and the class label for the unlabeled data, $c_i$. $V_{\underline{x_i}l}$ is a 1 if feature vector $\underline{x_i}$ was created by component l and a 0 otherwise. Since the hidden data is linear in the complete data log likelihood equation and because $V_{\underline{x_i}l}$ is a Boolean value, the expectation step only needs to find $E[V_{\underline{x_i}l}|\underline{\Theta}] = P[V_{\underline{x_i}l}|\underline{\Theta}]$ which by Bayes rule is

$$P[V_{\underline{x_i}j} = 1|\underline{x_i}, c_i, \underline{\Theta}^{(t)}] = \frac{\alpha_j^{(t)} \beta_{c_i|j} f(\underline{x_i}|\underline{\theta_j}^{(t)})}{\sum_{l=1}^{L} \alpha_l^{(t)} \beta_{c_i|l} f(\underline{x_i}|\underline{\theta_l}^{(t)})}, \tag{1.4}$$

$$P[V_{\underline{x_i}j} = 1|\underline{x_i}, \underline{\Theta}^{(t)}] = \frac{\alpha_j^{(t)} f(\underline{x_i}|\underline{\theta_j}^{(t)})}{\sum_{l=1}^{L} \alpha_l^{(t)} f(\underline{x_i}|\underline{\theta_l}^{(t)})}, \tag{1.5}$$

and

$$P[V_{\underline{x_i}j} = 1, c_i = k|\underline{x_i}, \underline{\Theta}^{(t)}] = \frac{\alpha_j^{(t)} \beta_{c_i|j} f(\underline{x_i}|\underline{\theta_j}^{(t)})}{\sum_{l=1}^{L} \alpha_l^{(t)} f(\underline{x_i}|\underline{\theta_l}^{(t)})}. \tag{1.6}$$

Assume that $f(\underline{x_i}|\underline{\theta_l})$ is a Gaussian density with mean vector $\underline{\mu_i}$ and covariance matrix $\Sigma_i$. The Maximization step requires the value of the model parameters which will maximize the log likelihood of the complete data. So the update equations for the mean and covariance parameters are obtained by differentiating the complete data log likelihood equations and solving for the parameters. The results are as follows:

$$\underline{\mu_j}^{(t+1)} = \frac{1}{N\alpha_j^{(t)}} \left( \sum_{\underline{x_i} \in \chi_l} \underline{x_i} P[V_{\underline{x_i}j} = 1|\underline{x_i}, c_i, \underline{\Theta}^{(t)}] + \sum_{\underline{x_i} \in \chi_u} \underline{x_i} P[V_{\underline{x_i}j} = 1|\underline{x_i}, \underline{\Theta}^{(t)}] \right) \tag{1.7}$$

$$\Sigma_j^{(t+1)} = \frac{1}{N\alpha_j^{(t)}} \left( \sum_{\underline{x_i} \in \chi_l} S_{ij}^{(t)} P[V_{\underline{x_i}j} = 1|\underline{x_i}, c_i, \underline{\Theta}^{(t)}] + \sum_{\underline{x_i} \in \chi_u} S_{ij}^{(t)} P[V_{\underline{x_i}j} = 1|\underline{x_i}, \underline{\Theta}^{(t)}] \right). \tag{1.8}$$

where $S_{ij}^{(t)} \equiv (\underline{x_i} - \underline{\mu_j}^{(t+1)})(\underline{x_i} - \underline{\mu_j}^{(t+1)})^T$. The update equation for the mixing parameters is

$$\alpha_j^{(t+1)} = \frac{1}{N} \left( \sum_{\underline{x_i} \in \chi_l} P[V_{\underline{x_i}j} = 1|\underline{x_i}, c_i, \underline{\Theta}^{(t)}] + \sum_{\underline{x_i} \in \chi_u} P[V_{\underline{x_i}j} = 1|\underline{x_i}, \underline{\Theta}^{(t)}] \right) \tag{1.9}$$

and to recalculate $\beta_{k|j}$

$$\beta_{k|j}^{(t+1)} = \frac{1}{N\alpha_j^{(t)}}\Big(\sum_{\underline{x_i}\in\chi_l \bigcap c_i=k} P[V_{\underline{x_i}j} = 1|\underline{x_i}, c_i, \underline{\Theta}^{(t)}] + \sum_{\underline{x_i}\in\chi_u} P[V_{\underline{x_i}j} = 1, c_i = k|\underline{x_i}, \underline{\Theta}^{(t)}]\Big). \tag{1.10}$$

This model was used in [47] and compared to a deterministic annealing method for Radial Basis Function Neural Networks. It was found to give better results, especially when the amount of labeled data was small compared to the unlabeled data. We attribute the better performance to more accurate estimation of the model parameters by the use of the unlabeled data. This shows the worth of this model, and it also shows the value of extra unlabeled data in the design of a classifier.

## 1.4 Contributions of This Thesis

One contribution concerns the application of statistical models to Collaborative Filtering. We show results for several statistical models on some benchmark CF databases and demonstrate their superiority over existing methods. In Chapter 4 we consider a totally new data mining problem not addressed before. Two new statistical models are developed in relation to this problem. Specifically:

1. In chapter 2, we show how to apply the Maximum Entropy Model and Iterative Scaling to a new domain, Collaborative Filtering. This involved overcoming the very long learning time associated with ME models on very large spaces by finding an appropriate learning algorithm and breaking down the problem into manageable pieces. In working on this problem we have also developed techniques for dealing

with unlabeled data that is always present in CF problems. One technique increases accuracy by 7.8 %.

2. In Chapter 3 the latent cluster model (LCM) is applied to CF. Previous applications of this model to large CF databases have been to simplified versions of the databases. We show how to learn an LCM for the whole database. This method can learn extremely large databases (much larger than EachMovie, which took 30 minutes to learn) in a reasonable amount of time. The accuracy of the resulting model gives the best results for this database that we have seen.

3. Finally in chapter 4, both the problem and the proposed solutions are new. As outlined in section 1.2, this chapter shows how to create a statistical model for a database where some of the unlabeled data can come from unknown classes. Being as this is the first time this problem has been posed, all the techniques we describe in this chapter are newly formulated for this problem. We formulate the learning algorithms for these models, and show how to use them for two classification tasks germane to this type of database. We also show a way of doing class discovery to identify the unknown classes.

# Chapter 2

# Maximum Entropy Model and other Methods for Collaborative Filtering

Many methods have been tried for CF. Some are neural networks [7], dimensionality reduction techniques using Principal Component Analysis [28] or Singular Value Decomposition [55], SVMs [23], correlation [53],rule based methods [27], and statistical models [32]. These methods can be divided into three categories: 1) memory based methods e.g. correlation; 2) classification methods like neural networks and SVMs; and 3) statistical models. In order to judge the worth of statistical models, we compare them to representatives of the other two types, namely correlation and SVMs.

## 2.1 Memory Based Methods - Correlation

Presently, the most popular algorithms for doing collaborative filtering are the memory based methods. This name comes from the fact that the whole database resides in the computer's memory and is used for creating predictions. These are the oldest methods for CF, dating to the early nineties and the Pearson correlation method is the benchmark against which other methods are compared. An alternative to memory based methods is classification based methods, one of which, Support Vector Machines (SVMs), is examined in this work. Finally the methods that are the thrust of this thesis, statistical models, can also be applied to the problem.

Memory based methods essentially perform weighted voting using the examples in the database, with the weight for each database example based on the degree of similarity between the example and the new instance $\underline{f^{(n)}}$. Here, we have considered similarity based on Pearson correlation as used in [9], [36]. Using the same notation as in the introduction , T is the number of user realizations in the training set (the size of the training set). The features in the model are random variables, denoted by $F_{ti_j}$ and the values they assume are $f^{(d)}_{ti_j}$, where $i_j$, $j = 1 \ldots L$, ranges in the reduced set of features. The feature we are trying to predict is $F_{ti*}$. Out of the total number of features in the problem, N, only a subset of L features are rated by the user, and these are the ones used to create the prediction for the new feature. Superscript (d) denotes a member of the training set and superscript (n) means a member of the test set. The weights are given by

$$W_t = \frac{\sum_{j=1}^{L}(f^{(d)}_{ti_j} - \overline{f^{(d)}_{ti}})(f^{(n)}_{ki_j} - \overline{f^{(n)}_{ki}})}{\sqrt{\sum_{j=1}^{L}(f^{(d)}_{ti_j} - \overline{f^{(d)}_{ti}})^2}\sqrt{\sum_{j=1}^{L}(f^{(n)}_{ki_j} - \overline{f^{(n)}_{ki}})^2}}, t = 1, \ldots, T, \tag{2.1}$$

where $\overline{f^{(d)}_{ti}}$ is the average vote

$$\overline{f^{(d)}_{ti}} = \frac{\sum_{j=1}^{L} f^{(d)}_{ti_j}}{L}, t = 1, \ldots, T, \tag{2.2}$$

for user t and $\overline{f^{(n)}_{ki}}$ is the average vote

$$\overline{f_{ki}^{(n)}} = \frac{\sum_{j=1}^{L} f_{ki_j}^{(n)}}{L}, k = 1, \ldots, K. \tag{2.3}$$

for test vector k.

Once the weights for each member of the training set are calculated, a weighted average of the votes in the database is calculated to give the predictions, i.e.

$$\widehat{f_{ki^*}^{(n)}} = \frac{\sum_{t=1}^{T} f_{ki_t}^{(d)} W_t}{\sum_{t=1}^{T} |W_t|} \tag{2.4}$$

There are a number of variations on this memory-based approach:

1) Often, before voting is done, the similarity values are discarded if they fall below a threshold, in order to to reduce the size of the voting ensemble. This is somewhat akin to the K-nearest neighbor method in classification [29].

2) Use of the Cosine-based similarity measure [51] is nearly the same as the Pearson similarity measure. The difference is that the mean of the users' votes are not subtracted from the ratings.

3) The middle rating, $\frac{1}{|A_{i_j}|} \sum_{l=1}^{|A_{i_j}|} l$, can be subtracted from each vote in the correlation of (2.1), instead of the average user vote, $\overline{f_{ti}^{(d)}}$ or $\overline{f_{ki}^{(d)}}$, [56].

4) In [1], instead of using a correlation to create the weight, an exponential function was used. This method achieved a 12% improvement in Mean Absolute error compared with the use of the Pearson correlation.

5) Users can be clustered together via the K-means algorithm. This leads to decreased prediction times and increased accuracy [25].

6) An interesting alternative to computing similarity measures (correlations) between users is to compute similarity measures between *items*. In order to predict items that a new user would select, items that have a strong affinity for items the user has already selected in the past are predicted [1].

Some practical concerns for memory-based methods include: 1) the reliability of the similarity measure, which is based on only the known feature subset (of size $L \ll N$); 2) the choice of the threshold, which controls the number of voters (and hence which can affect performance [1]). However, the most serious issue is how complexity scales with increasing database size – since the similarity is evaluated for each database example, inference complexity grows *linearly* with $T$. For MSWEB with 32,711 users in the database, inference takes about 6 seconds on a Sun Ultrasparc 2 machine. For a database with millions of records, complexity will become a serious obstacle. Clustering is one way to mitigate this problem.

## 2.2 Classification-based Methods - Support Vector Machines

One example of this approach is the application of support vector machines (SVMs) to CF [23] (although other classifiers can also be used). The SVM implements a linear decision boundary in the feature space, as shown in figure 2.1. This figure shows a two class problem in which one class is represented by solid dots and the other class by empty dots. In this figure there are many hyperplanes that will separate the two classes, but the position shown gives the greatest distance from the hyperplane to the closest point from either class. This is called the optimal separating hyperplane and is

found by solving a quadratic optimization problem. This figure shows a linearly separable case. It is also possible that the data is not linearly separable, in which case no separating hyperplane exists. In this case an optimal hyperplane that minimizes a measure of the training error is found [61]. The result of this optimization is a vector $\underline{w}$ and a discriminant function

$$g(\underline{x}) = \underline{w^T x} + w_0. \tag{2.5}$$

If $g(\underline{x}) < 0$ then $\underline{x}$ is on one side of the separating plane and if $g(\underline{x}) > 0$ it is on the other side. This corresponds to $\underline{x}$ being predicted as belonging to one class or the other.

In the case of MSWEB where the prediction task is to predict 1 or 0, (visit or non-visit), one builds N SVMs and thresholds the discriminant function (2.5) to get a binary classification. In the case of EachMovie [1], ratings are needed, not binary classifications. For EachMovie, $|A|$ SVMs are built, where $|A|$ is the number of possible ratings for each movie (in the case of ratings of 1..6, six SVMs per movie are created). Each SVM corresponds to one rating value and is trained to perform the the binary classification task of predicting whether or not that movie received that rating value. To predict a rating for one movie, all six SVMs are evaluated, and the rating's SVM with the highest output is the predicted rating for this movie. Here the assumption is that the higher the output of a rating's discriminant function, the more likely that rating is the correct one for this movie. For N movies, $N|A|$ SVMs are created. Since the SVM learning complexity grows with the data set size $T$, the learning must be done off-line.

[1] This database is available from Compaq Computer Systems Research Center at http://research.compaq.com/SRC/eachmovie/.

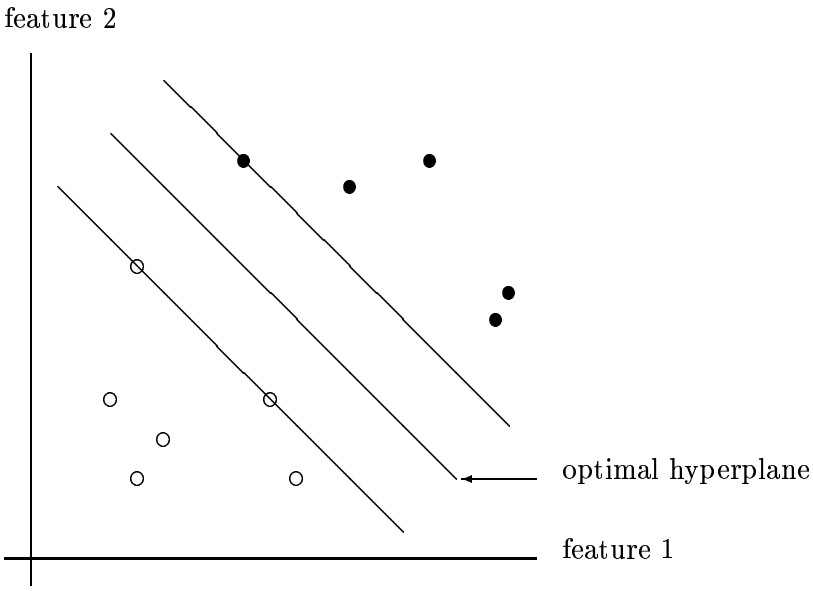


Fig. 2.1. A 2 class, 2 feature SVM problem

Moreover, since it is infeasible to learn and store one classifier for each possible inference task (i.e. each combination of target feature $i^*$ and known feature subset), each SVM must be built for the *full* space of dimension $N$, i.e. assuming an $(N-1)$-dimensional input vector with all values known. One difficulty this creates is that the database will have many missing feature values. Thus, a heuristic imputation strategy must be used to 'fill in' these values prior to SVM training. More serious is the fact that the new (test) examples will *also* have many missing features. Thus, heuristic imputations must also be made for many attributes in the new (test) example before SVM-based inference can proceed. These imputations can seriously affect inference accuracy. We have found that for the MSWEB database, SVMs give poorer results than correlation. SVMs for EachMovie were implemented by [23] where it was also found that the results were worse than correlation. Another issue is (even) the *off-line* learning complexity, since this grows at least with the square of $N$[2].

## 2.3 Statistical Modeling Approaches

These methods are based on a joint probability model over the feature space consisting of the known feature subset plus the feature to be predicted. First, given a specified CF inference task, the joint probability model is formed (in some cases, learned). The predictions are then based on the *a posteriori* probabilities $P[F_{i^*} = f_{i^*}|f_{i_1}, f_{i_2}, \ldots, f_{i_L}]$, which are computed (via Bayes rule) consistent with the joint probability model. This approach is grounded in the well-known theoretical result that the

[2]There are $N$ SVMs to build, with learning complexity for each one growing at least linearly with $N$.

optimal (Bayes) classifier in the presence of missing features uses the *a posteriori* probabilities that condition *only on the set of known features.* We state the result as follows Theorem: Given a probability space $(\underline{F^N}) = (F_1, F_2, \ldots, F_N)$, and an *incomplete* pattern $(\underline{f^K}) = (f_1, f_2, \ldots, f_K)$, where $K < N$, the Bayes classifier constructed on the marginal distribution $P[f_{i*}|\underline{F^K}]$ is optimal in minimizing the probability of classification error [63] [21].

In particular it is suboptimal to perform missing feature inference if one can instead evaluate the proper *a posteriori* probabilities, which condition only on the known features.

We believe that this class of CF methods is the most promising one. Statistical modeling approaches do not suffer from the missing feature difficulties of the classification-based methods[3], and their complexity has very limited dependence on the database size $T$. Moreover, these methods produce probabilities as inferences, which are attractive in some applications. We next briefly describe two such methods.

### 2.3.1 The Naive Bayes Model (NBM)

This model assumes all known features are conditionally independent given $F_{i*}$. The joint probability is thus $P[F_{i_1} = f_{i_1}, F_{i_2} = f_{i_2}, \ldots, F_{i_L} = f_{i_L}, F_{i*} = f_{i*}] = P[F_{i*} = f_{i*}] \prod_{j=1}^{L} P[F_{i_j} = f_{i_j}|F_{i*} = f_{i*}]$ and the associated *a posteriori* probabilities are

[3]No missing feature imputations are required for test examples. The unknown features are marginalized out. For model-building, low-order statistics (probabilities) must be estimated via frequency counts over the database. Here, missing features are encountered. We deal with this in the sequel.

(via Bayes rule)

$$P[F_{i^*} = f_{i^*} | f_{i_1}, f_{i_2}, \ldots, f_{i_L}] = \frac{P[F_{i_1} = f_{i_1}, F_{i_2} = f_{i_2}, \ldots, F_{i_L} = f_{i_L}, F_{i^*} = f_{i^*}]}{\sum_{k=1}^{|\mathcal{A}_{i^*}|} P[F_{i_1} = f_{i_1}, F_{i_2} = f_{i_2}, \ldots, F_{i_L} = f_{i_L}, F_{i^*} = k]}. \tag{2.6}$$

The conditional probabilities involved in (2.6) are estimated via frequency counts over the database, i.e. $P[F_{i_j} = f_{i_j} | F_{i^*} = f_{i^*}] = \frac{N(f_{i_j}, f_{i^*})}{N(f_{i^*})}$, where $N(\cdot)$ gives the number of times an event occurred.

### 2.3.2 The Maximum Entropy Model (MEM)

The ME concept is to choose the simplest explanation for the data possible, essentially a restatement of Occam's razor (do not multiply causes beyond necessity). For the case of classification this translates into creating a probability mass function (PMF) that agrees with low order probabilities measured from the data (these are the ones most accurately measured) and otherwise has maximum entropy. In [12], it was proposed to find the ME joint pmf consistent with *arbitrary* lower order probability constraints. This suggestion is in principle powerful, as it allows the joint pmf to express general dependencies, without any explicit assumptions of conditional independence (required e.g. by Bayesian nets or Naive Bayes). Unfortunately, there is a difficult learning problem for estimating the ME joint pmf. Several techniques have been proposed e.g. [43, 26]. However, these methods have intractable complexity for feature dimensionalities ($N$) that commonly arise [4]. To overcome learning difficulties, [30] proposed *Kutato*, which applies

[4] Ku and Kullback only considered $N = 4$, with two values per feature. By contrast, in [50] models with $N = 23$ and over 40 billion joint pmf cells were considered. [43] and other ME

the ME principle, but for the construction of Bayesian nets (BNs) rather than general ME models[5]. In recent work [50], the general ME learning problem was reconsidered, with the authors developing an approximate method for learning discrete space classifiers and then naturally extending this to general inference models [62]. In this thesis one of our aims is to develop methods that allow the application of ME to large-scale inference tasks, including CF. In the next section, we review the classifier formulation from [50].

#### 2.3.2.1 A review of ME for classification, the ME Joint PMF

Let $\underline{F}$ be a discrete feature space i.e. $\underline{F} = (F_1, F_2, \ldots, F_N)$, with $F_i \in \mathcal{A}_i$, and $\mathcal{A}_i$ the finite set $\{1, 2, 3, \ldots, |\mathcal{A}_i|\}$. Denote the full space by $\mathcal{G} \equiv \mathcal{A}_1 \times \mathcal{A}_2 \ldots \times \mathcal{A}_N$. The ME problem is to choose $P[\underline{F}]$ to maximize joint entropy while satisfying equality constraints on lower order probabilities. To illustrate the principal difficulties, it suffices to impose second order (pairwise) constraints – similar difficulties will arise for higher order ones. Thus, suppose we are given knowledge of *all* $\frac{N(N-1)}{2}$ *pairwise* pmfs $\{P[F_m, F_n],\ \forall m, \forall n > m\}$ [6]. The ME joint pmf consistent with these pairwise pmfs has

techniques cannot be applied to problems of this size, due to the intractability of the ME joint pmf's normalization and the pmf marginalizations required for satisfying constraints.

[5] BNs require explicit assertions of conditional independence. This limits the choice of constraints that can be encoded into the model.

[6] In practice, these probabilities are *estimated*, often from training set co-occurrence counts. Since the accuracy of estimates based on co-occurrence counts decreases with increasing *order* of the probabilities, it is reasonable to encode as constraints *all* probabilities at the same order (all such probabilities will be known to essentially the same degree of accuracy, assuming $|\mathcal{A}_i| \simeq |\mathcal{A}|, \forall i$ ). This also explains why in practice we limit the constraint order to two (pairwise pmfs) – higher order probabilities would require larger (perhaps huge) amounts of data to achieve reasonable accuracy for co-occurrence based estimates.

the Gibbs form:

$$P[\underline{F} = \underline{f}] = \frac{\exp\left(\sum_{m=1}^{N-1}\sum_{n>m}^{N}\gamma_{(F_m=f_m,F_n=f_n)}\right)}{\sum_{\underline{f'}\in\mathcal{G}}\exp\left(\sum_{m=1}^{N-1}\sum_{n>m}^{N}\gamma_{(F_m=f'_m,F_n=f'_n)}\right)}, \tag{2.7}$$

with $\underline{f} \equiv (f_1, \ldots, f_N) \in \mathcal{G}$. Here, $\gamma_{(F_m=f_m,F_n=f_n)}$ is the Lagrange multiplier associated with an equality constraint on $P[F_m = f_m, F_n = f_n]$. The joint pmf is specified by the *set* of Lagrange multipliers $\Gamma \equiv \{\gamma_{(F_m=f_m,F_n=f_n)}\ \forall m, \forall n > m,\ f_m \in \mathcal{A}_m\ f_n \in \mathcal{A}_n\}$. While the form of the joint pmf is known, direct evaluation of (2.7) is intractable, due to the denominator which requires computing and summing $f'$ over all possible combinations of features, for a total of $\prod_{i=1}^{N} |\mathcal{A}_i|$ terms. This may not be problematic during model use since one is generally not interested in $P[\underline{f}]$ directly, but rather in *a posteriori* probabilities of single features or feature groups. While these probabilities also depend on $\Gamma$, they can often be tractably computed.

However, this assumes $\Gamma$ is known. The real difficulties lie in *learning* $\Gamma$. We have already indicated the intractability of previous methods. Since the form of the joint pmf is known, one is tempted to try a "direct" approach: i) write down the ME constrained problem; ii) use the Lagrangian method to convert the constrained problem into an unconstrained one. The Lagrangian cost will have the form $D - TH$, $H$ the joint entropy, $D$ a cost function encoding the equality constraints, $T$ a Lagrange multiplier; iii) express $H$ and $D$ *explicitly* as functions of the joint pmf with optimal form (2.7); iv) minimize $D - TH$ over $\Gamma$. Some search for the value of $T$ satisfying constraints with maximum entropy may be required.

There are two major difficulties here. First, the optimization requires calculating $P[\underline{f}]$, intractable as already discussed. Second, $D$ will require (intractable) marginalizations over the joint pmf. While the method just outlined is infeasible, an *approximate* ME method, inspired by the principle of this "direct" approach, was recently introduced specifically for the design of statistical classifiers [50]. We next review this method.

#### 2.3.2.2 An Approximate ME Classifier

Consider $\underline{F_a} = (\underline{F}, C)$, with $\underline{F}$ augmented by a *class* feature $C \in \{1, 2, \ldots, K\}$. The ME joint pmf for $\underline{F_a}$ consistent with all pairwise feature constraints still has the intractable form (2.7). However, classification does not require computing $\underline{F_a}$ directly, but rather just the *a posteriori* probabilities *associated with* $P[\underline{F_a}]$. These probabilities, which only depend on the Lagrange multipliers involving the class feature, have the form:

$$P[C = c|\underline{F} = \underline{f}] = \frac{\exp(\sum_{i=1}^{N} \gamma_{(F_i=f_i, C=c)})}{\sum_{c'=1}^{K} \exp(\sum_{i=1}^{N} \gamma_{(F_i=f_i, C=c')})}, \quad c = 1, \ldots, K. \tag{2.8}$$

Note that unlike (2.7), (2.8) is easily computed. Now, given we are only interested in class inference, we only truly need to learn the *subset* of Lagrange multipliers required for (2.8), i.e. $\Gamma_c \equiv \{\gamma_{(F_i=f_i, C=c)}\}$. What we thus suggest is a tractable learning approach that maximizes entropy of the joint pmf subject to *all* the given constraints, while producing at least, but perhaps *no more* than the reduced set of parameters $\Gamma_c$[7]. The 'direct

[7] Even though the learning only needs to produce $\Gamma_c$, we should still encode as much information as possible when forming the ME joint pmf. Thus, we still suggest here to encode all pairwise pmf constraints. However, later we will also discuss alternative methodologies for choosing the constraints to encode.

approach' for obtaining the joint pmf, described in the previous section, learns more parameters (all of $\Gamma$) than those required for classification, but is not feasible. Here we review a tractable, approximate method.

*Joint PMF Form*

We start by representing the augmented joint pmf *using only the required Lagrange multipliers*. This can be achieved via the Bayes rule decomposition $P[\underline{F} = \underline{f}, C = c] = P[C = c|\underline{F} = \underline{f}]P[\underline{F} = \underline{f}], \forall \underline{f}, \forall c$, where we express $P[c|\underline{f}]$ in its ME form (2.8) but, *crucially*, where $\{P[\underline{f}], \underline{f} \in \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_N\}$ is instead specified by an exhaustive table of joint pmf values. Clearly, this introduces no suboptimality, as there are choices for the 'parameter set' $\{\{\gamma_{(F_i=f_i,C=c)}\}, \{P[\underline{f}]\}\}$ which achieve the ME joint pmf for the augmented feature vector $\underline{F_a}$. However, this representation does not directly help achieve tractable learning. In fact, this form vastly *expands* the number of parameters, replacing $\Gamma - \Gamma_c$ with the huge table $\{P[\underline{f}], \underline{f} \in \mathcal{G}\}$.

*Support Approximation*

While optimizing over $\{\{\gamma_{(F_i=f_i,C=c)}\}, \{P[\underline{f}]\}\}$ is not practical, a related *approximation* will indeed yield tractable, effective (approximate) learning of the desired parameters $\{\gamma_{(F_i=f_i,C=c)}\}$. The key is to view the values $\{P[\underline{f}]\}$ as *auxiliary* variables which are useful for supporting learning, but which play *no* role during inference – they are discardable following learning. Thus, we might represent $\{P[\underline{f}]\}$ in an approximate fashion if this helps make learning tractable. The approximation may have some effect on accuracy of the learned model $\{\gamma_{(F_i=f_i,C=c)}\}$, but will not sacrifice our capability for doing inference using (2.8). Thus, instead of optimizing $P[\underline{f}]$ on the full space $\mathcal{G}$, suppose that, during training, we restrict non-zero support of $P[\underline{f}]$ to a *subset* $\mathcal{G}_s$ of $\mathcal{G}$, with $|\mathcal{G}_s| \ll |\mathcal{G}|$.

Essentially, while optimizing over $|\mathcal{G}|$ values would entail enormous complexity, optimizing over $|\mathcal{G}_s| \ll |\mathcal{G}|$ values will be seen to be feasible. A good candidate for a reduced support set is the *empirical support expressed by the training data set itself* – i.e., the training feature vectors (excluding class labels). Some theoretical justification for this choice will emerge later when we identify the connection between this ME approach and iterative scaling[8]. Moreover, to further reduce learning complexity, it is also possible to pick the support set as a random *subset* of the available training set. Often, so long as $|\mathcal{G}_s|$ is made reasonably large, there is little effect on the classification performance [50].

*Lagrangian Formulation*

Given the support restriction, we choose $\{\{\gamma_{(F_i=f_i,C=c)}\}, \{P[\underline{f}], \underline{f} \in \mathcal{G}_s\}\}$ to maximize joint entropy while satisfying the given (pairwise) constraints. Following the 'direct approach', we form $H$ and $D$. Let us denote the joint pmf support by an enumerated set, $\mathcal{G}_s \equiv \{\underline{f_m} \equiv (f_1^{(m)}, f_2^{(m)}, \ldots, f_N^{(m)}), m = 1, \ldots, |\mathcal{G}_s|\}$. Then, given the support restriction, the joint entropy for $\underline{F_a}$ can be written

$$H = -\sum_{m=1}^{|\mathcal{G}_s|} P[\underline{f_m}] \log(P[\underline{f_m}]) - \sum_{m=1}^{|\mathcal{G}_s|} P[\underline{f_m}] \sum_{c=1}^{K} P[c|\underline{f_m}] \log(P[c|\underline{f_m}]). \tag{2.9}$$

Now let us compose $D$. For satisfying constraints on lower order probabilities, there are several criteria that could be used, including squared distance and cross entropy. We suggest the cross entropy criterion. Our *model's* pairwise pmf $P_M[F_k, F_l]$ can be

[8]Specifically, for this choice of support, with the additional constraint of a uniform pmf over the support set, the learning problem will be seen to be equivalent to a particular maximum likelihood problem.

calculated as

$$P_M[F_k = i, F_l = j] = \sum_{\underline{f_m} \in \mathcal{G}_s | f_k^{(m)} = i, f_l^{(m)} = j} P[\underline{f_m}], \quad i = 1, \ldots, |\mathcal{A}_k|, j = 1, \ldots, |\mathcal{A}_l|. \tag{2.10}$$

Note that this is an efficient *marginalization* of the joint pmf. Based on the properties of cross entropy, equality constraints on *all* probabilities involving $(F_k, F_l)$ will be satisfied if $P_M[\cdot]$ is chosen so that the cross entropy/Kullback distance

$$D_c(P[F_k, F_l] || P_M[F_k, F_l]) \equiv \sum_{i=1}^{|\mathcal{A}_k|} \sum_{j=1}^{|\mathcal{A}_l|} P[F_k = i, F_l = j] \log \frac{P[F_k = i, F_l = j]}{P_M[F_k = i, F_l = j]} \tag{2.11}$$

is minimized, i.e. *zeroed*. For pairwise constraints involving the class label, $P[F_k, C]$, we similarly have

$$P_M[F_k = i, C = c] = \sum_{\underline{f_m} \in \mathcal{G}_s | f_k^{(m)} = i} P[c | \underline{f_m}] P[\underline{f_m}], \quad i = 1, \ldots, |\mathcal{A}_k|, c = 1, \ldots, K, \tag{2.12}$$

and an associated cross entropy cost. To satisfy *all* pairwise constraints, our *overall constraint cost* $D$ is formed as a sum of all the individual pairwise costs, i.e.

$$D = \sum_{k=1}^{N-1} \sum_{l>k} D_c(P[F_k, F_l] || P_M[F_k, F_l]) + \sum_{k=1}^{N} D_c(P[F_k, C] || P_M[F_k, C]). \tag{2.13}$$

Given $D$ and $H$, we form the *Lagrangian cost* $L = D - TH$, with $T$ a Lagrange multiplier, and seek to minimize this over $\{\{\gamma_{(F_i = f_i, C = c)}\}, \{P[\underline{f}], \underline{f} \in \mathcal{G}_s\}\}$. Our ultimate goal is to minimize $L$ such that we find the solution with maximum $H$ under the constraint $D = 0$.

We tackle this by a *continuation* method, minimizing $L$ for a sequence of decreasing $T$ values, tracking parameter values from one $T$ to the next, and stopping when $D$ drops below a suitable threshold or stops changing, with the minimization at each $T$ achieved by gradient descent. With this approach, we control satisfaction of all constraints by variation of a *single* parameter, $T$. In practice, we have found that little variation of $T$ (annealing) is required – we can start from small $T$ (e.g. $T = 0.1$) and run the continuation approach until our stopping condition is reached. Starting from higher $T$ does not typically yield better results.

While this ME method was initially formulated for classification problems involving up to 30 features, the main point of this research was to look at methods appropriate for the collaborative filtering problem and other large-scale inference tasks. While theoretically the ME approach can be applied to CF, because the collaborative filtering problem typically is very large (involving hundreds or thousands of features) the ME approach is not practical as is. In this thesis we demonstrate that by capitalizing on some simple observations about the CF problem, and by using iterative scaling, ME learning can be sped up to the point of being practical for CF.

#### 2.3.2.3 Relation to Iterative Scaling Techniques

The ME method [50] chooses the parameters $\{\gamma_{(F_i=f_i,F_{i*}=f_{i*})}\}, \{P[\underline{f}]\}\}$ to maximize joint entropy while satisfying the given (pairwise) constraints. Note that we now refer to the class label as $F_{i*}$ instead of C. When classification is done in the CF framework, there is no fixed class label, the class label can be any of the features. This is reflected in the use of $F_{i*}$. One way of reducing the complexity of this approach is by

instead *fixing* the distribution $\{P[\underline{f}], \underline{f} \in \mathcal{G}_s\}$ and optimizing only over the $\gamma(\cdot)$ parameters. Consider the special case where $\mathcal{G}_s$ is the full training set support (of size denoted $N_t$), with $P[\underline{f}]$ fixed to be uniform, i.e. $P[\underline{f}] = \frac{1}{N_t}, \underline{f} \in \mathcal{G}_s$, and where the pairwise constraints $\{P[F_k = f_k, C = c]\}$, $\{P[F_k = f_k, F_l = f_l]\}$ are obtained from frequency count estimates over the training set, e.g. $P[F_k = f_k, F_l = f_l] = \frac{N(F_k=f_k, F_l=f_l)}{N_t}$. Here, $N(\cdot, \cdot)$ is the number of times a given event occurs in the training set. The pairwise constraints that have to be met when $P[\underline{f}] = \frac{1}{N_t}$ are

$$\sum_{\underline{f_t}^{(d)} \in \mathcal{F}_d : f_{ti_j}^{(d)} = f_{i_j}} \frac{1}{T} P_{\mathrm{ME}}[F_{i^*} = f_{i^*} | \tilde{\underline{f}}_t^{(d)}] = \frac{N(F_{i^*} = f_{i^*}, F_{i_j} = f_{i_j})}{T}. \tag{2.14}$$

For this particular case, we next show that the ME problem *specializes* to that addressed by iterative scaling techniques [4, 52, 5] – i.e., we show that for the given case, the ME problem is equivalent to a particular *maximum likelihood* problem associated with the conditional probability model (2.8).

We start by making several observations about the choice $P[\underline{f}] = \frac{1}{N_t}, \underline{f} \in \mathcal{G}_s$. First, note that with the constraints based on frequency counts, the uniform distribution choice for $\{P[\underline{f}]\}$ *automatically* satisfies all the pairwise constraints that do not involve the class label, i.e. for $P_M[\cdot]$ specified by (2.10), we have $P_M[F_k = m, F_l = n] = P[F_k = m, F_l = n] = \frac{N(F_k=m, F_l=n)}{N_t}$ $\forall k, l, m, n$. Second, we emphasize that, although automatically satisfying these constraints, the uniform distribution for $\{P[\underline{f}]\}$ is in general *not* the choice consistent with maximizing joint entropy subject to *all* pairwise

constraints[9]. However, in practice, as will be seen shortly, we have found that, while not the ME choice, solutions based on a uniform $\{P[\underline{f}]\}$ over the training set support have joint entropy only slightly less than that based on optimizing over $\{P[\underline{f}]\}$. Thus, since fixing $P[\underline{f}]$ to be uniform substantially reduces both the number of parameters to be optimized *and* the number of constraints to be explicitly encoded, and since it is nearly ME-optimal in practice, this strategy appears to be a quite reasonable one that should significantly reduce learning complexity. Finally, note that with $P[\underline{f}]$ fixed, the joint entropy $H = H(\underline{F}) + H(C|\underline{F})$ only depends on $\{\gamma(\cdot,\cdot)\}$ thru the conditional entropy $H(C|\underline{F})$. Thus, the ME learning problem can be restated as maximizing $H(C|\underline{F})$ subject to the constraints $\{P[F_1, C], P[F_2, C], \ldots, P[F_N, C]\}$. Accordingly, following [4], the ME Lagrangian cost function can be written as

$$L = -\sum_{\underline{f}\in\mathcal{G}_s}\sum_{k=1}^{K}\frac{1}{N_t}P[F_{i*}=k|\underline{f}]\log P[C=k|\underline{f}]$$

$$+\sum_{i=1}^{N}\sum_{j=1}^{|\mathcal{A}_i|}\sum_{l=1}^{K}\gamma_{(F_i=j,F_{i*}=l)}(P_M[F_i=j,F_{i*}=l]-\frac{N(F_i=j,F_{i*}=l)}{N_t}), \qquad (2.15)$$

with $P_M[\cdot,\cdot]$ given by (2.12) and with $P[C=k|\underline{f}]$ given by (2.8). The second term in $L$ embodies the ME constraints[10]. After several steps of simplification, it can be shown

[9]Since $H = H(\underline{F}) + H(C|\underline{F})$, the *unconstrained* joint entropy is maximized by the choice of the uniform distribution for $\{P[\underline{f}]\}$. However, when the constraints $\{P[F_1, C], P[F_2, C], \ldots, P[F_N, C]\}$ must be satisfied, the uniform distribution need not be, and often is not the ME choice for $P[\underline{f}]$.

[10]Note that constraints are encoded differently in the iterative scaling formulation than in [50], where cross entropy costs are used. The Lagrangian form (2.15) used in iterative scaling is required to have an equivalence to the maximum likelihood problem.

that this Lagrangian reduces to

$$L = \frac{1}{N_t} \sum_{(\underline{f},c)\in\tilde{\mathcal{G}}_s} \log P[C = c|\underline{f}], \tag{2.16}$$

where $\tilde{\mathcal{G}}_s$ is the training set support for the *augmented* feature space $(\underline{F}, C)$. In other words, the Lagrangian cost function associated with the ME problem under consideration is precisely the log-likelihood for the class-conditional model $P[C = k|\underline{f}]$, based on the training set $\{(\underline{f}, c) \in \tilde{\mathcal{G}}_s\}$. This log-likelihood is the quantity maximized by iterative scaling techniques [4].

Taking the partial derivative with respect to any of the Lagrange multipliers can be shown to give the following (satisfying) result:

$$\frac{\partial L}{\partial \gamma_{(F_{i^*}=f_{i^*},F_{i_j}=f_{i_j})}} = \frac{N(F_{i_j}=f_{i_j},F_{i^*}=f_{i^*})}{T} - \sum_{\underline{f_t}^{(d)}\in\mathcal{F}_d:f_{ti_j}^{(d)}=f_{i_j}} \frac{1}{T} P_{\mathrm{ME}}[F_{i^*} = f_{i^*}|\underline{\tilde{f}_t^{(d)}}], \tag{2.17}$$

i.e., setting the partial derivatives to zero, a necessary optimality condition is simply the constraint condition (2.14), which says that constraint probabilities measured with respect to the model's joint pmf must equal those measured with respect to the empirical (uniform) distribution over the database. Thus, ME learning amounts to simply choosing the parameters of the exponential model to satisfy the constraints. Moreover, since the constraints are linear, the solution is unique [4]. Thus, ME learning seeks the (unique) exponential model satisfying the given constraints (2.14). There are several techniques that can be used to perform this learning of the $\gamma(\cdot,\cdot)$ parameters, including gradient descent and improved iterative scaling [4]. Both of these methods will require computing

the sums

$$\sum_{\underline{f_t}^{(d)} \in \mathcal{F}_d : f_{ti_j}^{(d)} = f_{i_j}} \frac{1}{T} P_{\mathrm{ME}}[F_{i*} = f_{i*} | \underline{\tilde{f}_t^{(d)}}]. \tag{2.18}$$

The efficiency of this computation can in some cases be greatly increased with the following observation: since $L \ll N$, it may also be true that $|\mathcal{A}_{i_1} \times \mathcal{A}_{i_2} \times \ldots \times \mathcal{A}_{i_L}| \ll T$. When this is true, the number of terms in the sum can be drastically reduced. In particular the sum can be rewritten as

$$\sum_{\underline{f} \in \mathcal{A}_{i_1} \times \mathcal{A}_{i_2} \times \cdots \times \mathcal{A}_{i_L} \times \mathcal{A}_{i*} : F_{i_j} = f_{i_j}} \frac{1}{T} N(\underline{f}) P_{\mathrm{ME}}[F_{i*} = f_{i*} | \underline{\tilde{f}}]. \tag{2.19}$$

For the MSWEB database, this simplification leads to substantial savings in learning time. For EachMovie, this approach does not help since $|\mathcal{A}_{i_1} \times \mathcal{A}_{i_2} \times \ldots \times \mathcal{A}_{i_L}| > T$ for the choices of $L$ that we will consider. Below we describe the iterative scaling approach we have taken for ME learning.

#### 2.3.2.4 Iterative Scaling Algorithm

Iterative scaling (IS) is a method of learning a joint probability distribution that agrees with given marginal constraints. This method is outlined in many places, e.g. [15]. For a joint probability distribution with two variables $P[I = i, J = j]$ and known marginal constraints $P_{MC}[I = i]$ and $P_{MC}[J = j]$ that are to be met by $P[I = i, J = j]$, the following equations are iterated until convergence

$$P^{(t+1)}[I = i, J = j] = P^{(t)}[I = i, J = j] \frac{P_{MC}[I = i]}{\sum_j P^{(t)}[I = i, J = j]}, i = 1..|I|, \tag{2.20}$$

$$P^{(t+1)}[I=i,J=j] = P^{(t)}[I=i,J=j]\frac{P_{MC}[J=j]}{\sum_i P^{(t)}[I=i,J=j]}, j = 1..|J|. \qquad (2.21)$$

This will converge to the joint probability distribution closest in the cross-entropy/Kullback sense to the initial distribution of $P[I=i,J=j]$. If $P[I=i,J=j]$ is initialized to a uniform probability distribution (the maximum entropy distribution if there are no other constraints), the resulting distribution after running IS to convergence, will be the one that agrees with the given constraints and has maximum entropy. Because the conditional exponential model has a different structure from a simple table of joint probabilities, Lagrange multipliers have to be learned instead of joint probability table entries. The algorithm for learning the Lagrange multipliers is known as improved iterative scaling (IIS) and is the subject of [52]. Instead of equations (2.20) and (2.21) to update table entries, each Lagrange multiplier $\gamma_{(F_{i*}=f_{i*},F_{i_j}=k)}$ is updated by calculating

$$\Delta\gamma_{(F_{i*}=f_{i*},F_{i_j}=k)} = \frac{1}{L}\log\frac{P_{MC}[F_{i^*}=f_{i^*},f_{i_j}=k]}{\sum_{n|f_{i_j}=k}P[F_{i^*}=f_{i^*}|\underline{f^n}]P[\underline{f^n}]} \qquad (2.22)$$

and then the LM is updated

$$\gamma_{(F_{i*}=f_{i*},F_{i_j}=k)} = \gamma_{(F_{i*}=f_{i*},F_{i_j}=k)} + \Delta\gamma_{(F_{i*}=f_{i*},F_{i_j}=k)}. \qquad (2.23)$$

Here L is a constant that is equal to the number of features being conditioned on.

There are several important implications, which we next identify:

1. Iterative scaling methods essentially amount to a special case of our ME approach, where the support pmf is fixed to be uniform over the full training set and where constraints are measured via frequency counts over the training set[11].

2. By reducing the numbers of parameters to be optimized and constraints to be explicitly encoded, iterative scaling significantly reduces learning complexity compared with the ME approach in [50]. Moreover, we have observed experimentally that the 'improved iterative scaling' technique [5], which uses a closed form expression for updating one Lagrange multiplier at a time, appears to converge quite quickly, and much faster than gradient descent. We have capitalized on this efficient iterative scaling approach to develop an *on-line* ME model building and inference approach for collaborative filtering.

   Finally, by assuming $\{P[\underline{f}]\}$ to be uniform on the training set, iterative scaling *implicitly* encodes not only all pairwise pmfs $\{P[F_k, F_l]\}$, but all *higher* order ones as well. This is because the measured constraints $P[F_k = f_k, F_l = f_l] = \frac{N(F_k=f_k, F_l=f_l)}{N_t}$, and the model's estimate of pairwise constraints,

$$P[F_k = f_k, F_l = f_l] = \sum_{n|F_k^n=f_k, F_l^n=f_l} P[\underline{f^n}] \tag{2.24}$$

   for other pairs of constraints will automatically be the same for the choice $P[\underline{f^n}] = \frac{1}{T}$. As a matter of fact all constraints of all orders measured from the training set

[11] The two methods also differ in the translation from constraints to Lagrangian cost function. However, our emphasis here is on the support pmf and the way constraint probabilities are measured.

are satisfied by the choice $P[\underline{f^n}] = \frac{1}{T}$. By contrast, in [50] $P[\underline{f^n}]$ was not held constant but was a value that could be optimized to maximize the entropy of the joint pmf while just encoding pairwise constraints between all features.

Direct comparisons of the IIS approach for ME model learning to the method in [50] leads to very similar results when inference results are compared for several UC Irvine data sets, see table (2.1).

| | Maxent | | Iterative Scaling | |
|---|---|---|---|---|
| Data Sets | Error Rate | Entropy | Error Rate | Entropy |
| Car | 0.249 | 7.07 | 0.249 | 7.04 |
| Balance | 0.200 | 6.31 | 0.236 | 6.29 |
| Nursery | 0.236 | 8.93 | 0.230 | 8.91 |
| Congress | 0.059 | 5.78 | 0.052 | 5.74 |
| Credit | 0.119 | 6.24 | 0.119 | 6.24 |
| Hepatitis | 0.255 | 4.69 | 0.255 | 4.64 |

Table 2.1. Test set misclassification rates and final joint entropy values for our approach and iterative scaling.

We also found a very substantial speed advantage for IIS vs. [50], which uses a gradient search. For a model that takes 10 minutes for [50] to learn, IIS takes less than five seconds on a Sun Ultrasparc 2. This is a speed advantage of 200 to 300 times! Without this great increase of speed, learning for larger databases would be impossible. We will capitalize on the speed advantage of IIS in proposing an ME approach for CF.

#### 2.3.2.5 Improved Iterative Scaling

The improved iterative scaling procedure was first developed in [52] as a method of finding the Lagrange multipliers to maximize the log likelihood (2.16) when $P_{\mathrm{ME}}[F_{i^*} = f_{i^*}|\underline{\tilde{f}_t^{(d)}}]$ has an exponential form, as in (2.8). It has been shown e.g. in [64] and [52] that this log-likelihood function is convex in the Lagrange multipliers and thus has a unique maximum.

#### 2.3.2.6 Algorithm

In the following, when we wish to refer to the vector composed from the full set of Lagrange multipliers or to a perturbation of this vector, we will use the notation $\underline{\gamma}$ or $\underline{\Delta\gamma}$, respectively. The improved iterative scaling algorithm consists of the following basic steps:

1. Initialize the Lagrange multipliers, $\underline{\gamma}$. This can be done randomly; $k \leftarrow 0$.

2. Calculate an increment for the Lagrange multipliers $\underline{\Delta\gamma^{(k)}}$ that is guaranteed to increase the log likelihood function.

3. Add the calculated increments to the Lagrange multipliers, i.e.

$$\underline{\gamma^{(k+1)}} \leftarrow \underline{\gamma^{(k)}} + \underline{\Delta\gamma^{(k)}};\ k \leftarrow k+1 \tag{2.25}$$

Steps 2 and 3 are repeated until a convergence criterion is met.

Step 2 is implemented through the use of an auxiliary function $A(\underline{\Delta\gamma}, \underline{\gamma})$. This function is chosen so that:

1) For any $\underline{\Delta\gamma}$

$$\mathcal{L}(\underline{\gamma}+\underline{\Delta\gamma})-\mathcal{L}(\underline{\gamma})\geq A(\underline{\Delta\gamma},\underline{\gamma}); \tag{2.26}$$

2) $A(\underline{\Delta\gamma},\underline{\gamma})$ is a convex function in $\underline{\Delta\gamma}$, i.e. it has a unique maximum, also $\max_{\underline{\Delta\gamma}} A(\underline{\Delta\gamma},\underline{\gamma})\geq 0$.

Let $\underline{\Delta\gamma^{(k)}}$ be the value that maximizes $A(\underline{\Delta\gamma},\underline{\gamma})$. This increment to the Lagrange multipliers will guarantee an increase in the log likelihood function. As the algorithm iterates, $\mathcal{L}(\underline{\gamma^{(k)}})$ will thus increase monotonically. It is proved in [52] (and the proof is outlined later) that, in fact, the likelihood converges to $\max\mathcal{L}(\underline{\gamma^{(k)}})$ and $\underline{\gamma}\rightarrow\arg\max_{\underline{\gamma}}\mathcal{L}(\underline{\gamma})$.

For our ME problem the auxiliary function is given by

$$A(\underline{\Delta\gamma},\underline{\gamma})=\sum_{j=1}^{N_c}\sum_{m=1}^{|A_{i*}|}\sum_{f_{i_j}=1}^{|A_{i_j}|}\Delta\gamma_{(F_{i*}=m,F_{i_j}=f_{i_j})}\frac{N(F_{i^*}=f_{i^*},F_{i_j}=f_{i_j})}{T}+1- \tag{2.27}$$

$$\sum_{j=1}^{N_c}\sum_{m=1}^{|A_{i*}|}\sum_{f_{i_j}=1}^{|A_{i_j}|}\frac{1}{LT}\sum_{\underline{f_t}^{(d)}\in\mathcal{F}_d:f_{ti_j}^{(d)}=f_{i_j}}P_{\mathrm{ME}}[F_{i^*}=f_{i^*}|\underline{\tilde{f}_t^{(d)}}]e^{L\Delta\gamma_{(F_{i*}=m,F_{i_j}=f_{i_j})}},$$

where L is the number of known features. It can be shown that $A(\underline{\Delta\gamma},\underline{\gamma})$ satisfies (2.26), see [52]. The convexity of this function can be shown by computing the Hessian matrix and confirming that it is indeed negative definite.

To maximize $A(\underline{\Delta\gamma},\underline{\gamma})$, we take the partial derivative, set it to zero, and solve for $\Delta\gamma_{(F_{i*}=m,F_{i_j}=f_{i_j})}\forall i_j,j=1,\ldots,L\ \forall m\forall f_{i_j}$. We then find that, for each Lagrange

multiplier the update is:

$$\Delta\gamma^{(k+1)}_{(F_{i*}=m,F_{i_j}=f_{i_j})} = \frac{1}{L}\log\frac{\frac{N(F_{i*}=f_{i*},F_{i_j}=f_{i_j})}{T}}{\frac{1}{T}\sum_{\underline{f_t}^{(d)}\in\mathcal{F}_d:f^{(d)}_{ti_j}=f_{i_j}} P_{\mathrm{ME}}[F_{i*}=f_{i*}|\underline{\tilde{f}_t^{(d)}}]}. \tag{2.28}$$

This is the update used in step 2 of the algorithm. Note that when a pairwise constraint is satisfied, the expression inside the log() is one, and the associated component of $\underline{\Delta\gamma}$ is thus zero. The auxiliary function $A(\underline{\Delta\gamma},\underline{\gamma})$ has several important properties [52]:

1. $A(\underline{0},\underline{\gamma}) = 0$, which implies that the maximum of $A(\underline{\Delta\gamma},\underline{\gamma})$ is at least 0. This condition is necessary because if $\underline{\Delta\gamma} = \underline{0}$ there is no change in the log likelihood.

2. The directional derivatives of $\mathcal{L}(\underline{\gamma}+\underline{\Delta\gamma})$ and $A(\underline{\Delta\gamma},\underline{\gamma})$ are equal at $\underline{\Delta\gamma} = \underline{0}$, i.e.

$$\frac{d}{dt}\mathcal{L}(\underline{\gamma}+t\underline{\Delta\gamma})|_{t=0} = \frac{d}{dt}A(t\underline{\Delta\gamma},\underline{\gamma})|_{t=0}. \tag{2.29}$$

Since $\frac{d}{dt}A(t\underline{\Delta\gamma},\underline{\gamma})|_{t=0} = \nabla A(\underline{0},\underline{\gamma})\cdot\underline{\Delta\gamma}$ this implies that at $\underline{\Delta\gamma} = \underline{0}$ the gradients of $A(\underline{\Delta\gamma},\underline{\gamma})$ and $\mathcal{L}(\underline{\gamma})$ are equal. Specifically if $\nabla A(\underline{0},\underline{\gamma}) = \underline{0}$, i.e. if the maximum of $A(\underline{\Delta\gamma},\underline{\gamma})$ is found when $\underline{\Delta\gamma} = \underline{0}$, then the maximum of $\mathcal{L}(\underline{\gamma})$ has also been found.

#### 2.3.2.7 Convergence

A convergence proof is given in [52]. We provide a brief summary of the argument, as follows. We can see from the algorithm described above that every iteration step increases the log likelihood. Since the sequence steps are probability distributions which

are members of a compact set, there is a convergent subsequence and at least one cluster point. It is shown in [52] that all cluster points of the algorithm will be maxima of the log likelihood. Further, since the log likelihood is convex it has only one maximum and hence only one cluster point. Therefore the algorithm converges to the unique point that satisfies $\nabla A(\underline{0}, \underline{\gamma}) = \nabla \mathcal{L}(\underline{\gamma}) = \underline{0}$. This also satisfies the constraints in 2.14.

#### 2.3.2.8 Application of IIS to Collaborative Filtering

In order to apply the IIS algorithm to CF, we must specify the inputs to the model and which constraints the model will learn. For EachMovie with 1622 features, a user may have only rated five features $f_{i_1}, \ldots, f_{i_5}$ (L=5), and is requesting a prediction for a sixth, $f_{i*}$. To make this problem tractable, we only want to learn a conditional probability model based on the L features the user has rated and the 1 feature to be predicted. Only the five constraints (pairwise probabilities) $P(F_{i*}, F_{i_1}), \ldots, P(F_{i*}, F_{i_5})$ are learned. Since the number of constraints is few, a new model can be learned on-line for each inference task. [12] This is the way we mitigate the combinatoric problem of section 1.1. A further problem in implementation is that of missing data. This is addressed in the next section.

#### 2.3.2.9 Implicit Versus Explicit Databases and Missing Votes

We have tested our ME inference method on two different CF databases, EachMovie and MSWEB. The EachMovie database consists of movie ratings from 61,264 users. There are a total of 1622 rated movies. Each user rates movies he or she has seen.

[12] It is possible to learn a model based on the whole space, but not in 'real-time', on-line.

The ratings are "1" to "6" with a value of "0" for all movies not seen. The MSWEB database lists Microsoft web sites visited by 32,711 users. There are 285 web sites that have been visited by at least one user. The database values are "0" and "1" for "not visited" and "visited", respectively. The CF objective is to predict whether a user will visit a given web site or not. The first database, EachMovie, is an example of an *explicit* database while the second one is an example of an *implicit* database. The difference is that the records represent users' preferences based on either their 1) explicit voting or rating of objects or 2) their past behavior. Explicit voting involves explicit rating of items. For example, rating movies or newspaper articles. In this case there may be many missing features, i.e. a person will not usually see every single movie that Hollywood has released, and so many movies are unrated in the database. On the other hand, implicit votes are based on the interpretation of a user's behavior. If a person accesses an item, for instance buying a product or visiting a web site, that is counted as a "yes" vote. All items not accessed are counted as "no" votes. Implicit voting is interpreted as having no missing information, i.e. by definition any items not accessed are assigned "no" votes[13]. For the case of explicit voting, one needs to address the problem of missing feature values in the database records when building the model. Specifically the missing votes for $F_{i*}$, the feature that is predicted, needs to be addressed. There can also be missing votes in the L known features, but the Lagrange multipliers associated with conditioning features equal to unrated $\gamma_{(F_{i*}=f_{i*},F_{i_j}=\text{"unrated"})}$ are only used in learning, and not used

[13]This representation will sometimes falsely interpret the user's behavior, e.g. a user may have had very limited time to explore a Web database. Thus, the sites indicated as visited may not accurately reflect the sites the user would be *inclined* to visit in general. However, there is no additional data available for gauging users' intentions.

in prediction. This is because the L features being conditioned on are always rated by the user so these features always have values. Therefore we regard this as a less serious problem. We next consider the problem of missing votes for $F_{i*}$, proposing several novel learning approaches amenable to the ME statistical modeling approach to account for these missing feature values.

We have the following strategies for handling missing votes:

1. One can learn a model where the rating $F_{i*}$ = "unrated", is a valid choice. In this case, for EachMovie, the possible ratings would be the six values "1" to "6" and another value representing "unrated", making a total of seven possible ratings. If the maximum *a posteriori* rule $f_{i*} = \arg\max_l P_{\text{ME}}[F_{i*} = l|\underline{f}]$ is applied, this will result in many "unrated" predictions because the database has many unrated instances. Since our objective is to predict *ratings* for the movies, these unrated predictions would be unacceptable. An alternative, after model learning, is to apply a rule that disallows unrated predictions, i.e. to use the rule $f_{i*} = \arg\max_l P_{\text{ME}}[F_{i*} = l|\underline{f}, l \in \{\text{ratings choices}\}]$, where $P_{\text{ME}}[F_{i*} = l|\underline{f}, l \in \{\text{ratings choices}\}] = \frac{P_{\text{ME}}[F_{i*}=l|\underline{f}]}{(1-P_{\text{ME}}[F_{i*}=\text{“}unrated\text{”}|\underline{f}])}, l \in \{\text{ratings choices}\}$. This is a MAP rule that *excludes* "unrated" as a valid choice.

2. A second method is to *learn* a model that restricts the values taken on by $F_{i*}$ to be ratings values. To achieve this, one must discard all database examples for which $F_{i*}$ = "unrated" when forming frequency count estimates of the pairwise statistics $\{P[F_{i*}, F_j]\}$. This will result in the ratings-restricted learned pmf $P_{\text{ME}}[F_{i*} =$

$l|\underline{f}], l \in$ {ratings choices}. This method, called the 'throw away data' method, achieves better results than the first method.

3. A third method we propose for on-line ME is a two stage learning method: First, learn an ME model using method 2. Next, evaluate $P_{\text{ME}}[F_{i*} = l|\underline{f}^{(n)}], l \in$ {ratings choices} for all database examples for which $F_{i*}$ = "unrated". Effectively, this produces probabilistic ratings for all unrated database examples. Thus, all database examples will now have ratings, either deterministic or probabilistic ones. We can thus form new constraint probabilities via

$$P[F_{i*} = k, F_{i_j} = f_{i_j}] = \frac{N(F_{i*}=k, F_{i_j}=f_{i_j}) + \sum_{t: f_{ti*}^{(d)} = \text{"unrated"}} P_{\text{ME}}[F_{i*}=k|\underline{\tilde{f}_t^{(d)}}]}{T}, k \in \{\text{ratings choices}\}. \tag{2.30}$$

In this way, unlike method 2, all the data is tapped for estimating the constraint probabilities. Finally, we relearn the ME model $P_{\text{ME}}[F_{i*} = l|\underline{f}^{(n)}], l \in$ {ratings choices} and use it in a MAP rule. This method, which we call 'relearn model', is the best missing feature method we have found for on-line ME classification.

## 2.4 Results

Two databases, EachMovie and MSWEB, were used for evaluation. For the MSWEB database, the following constitutes one trial for a CF algorithm. Either $L$ =2, 5 or 10 of the 285 visited web sites were randomly chosen as known, with values for the remaining sites to be inferred. Of the 5000 users in the test set, only those with at least

one of the selected sites visited were included in the test set for this particular trial. That means that the number of test vectors differ for each trial. A model was learned for each inference task and prediction was then done on all of the test vectors in the trial. Because the results varied from one trial to another, a total of fifteen trials were done. The reported results were obtained by micro-averaging over all of the trials. The methods we compare include Correlation, SVMs, naive Bayes, a brute-force frequency-count estimator (FC), and ME. For Correlation, we used the Pearson coefficient as the basis for the weights, as in [9]. The weights were formed for all the training examples and then weighted voting was performed. Weight amplification with a factor of 2 was also used. [14] The FC method estimates *a posteriori* probabilities directly via $\frac{N(F_i{*}{=}f_i{*},\underline{F}{=}\underline{f})}{N(\underline{F}{=}\underline{f})}$. For SVMs, the *SVM light* program of Thorsten Joachims [39] was used. We created 285 different linear SVMs. To do inference, the known values were set to 1 or 0 and all unknown values were set to 0.5. The dot products for all sites to be inferred were formed in the usual way and the values were compared to a threshold for prediction. While the threshold is typically chosen to be zero, we found that a non-zero value gave better performance. Results are reported using the recall/precision metric. Recall (r) is the percentage of visited web sites that we predicted as visited. Precision (p) is the percentage of predicted visits that really were visited. For MSWEB we report the $F_1$ measure which combines precision and recall. The definition of $F_1$ is:

$$F_1 = \frac{2pr}{p+r}. \tag{2.31}$$

[14] Weight amplification is described in [9]. It is merely raising the weights resulting from eq. (2.1) to the designated power, in this case 2, while preserving the sign of the weight. This is one of several heuristic techniques described in [9] to improve correlation classification.

For each method the threshold was found that gave the point where precision equals recall. As aforementioned, since results can vary considerably for different groups of known features, we averaged the results over 15 (randomly selected) groups.

| | Number Features Present | | |
|---|---|---|---|
| Data sets | Given 2 | Given 5 | Given 10 |
| MEM | 0.331 | 0.329 | 0.332 |
| NBM | 0.317 | 0.303 | 0.326 |
| FC | 0.329 | 0.325 | 0.331 |
| CORR | 0.198 | 0.195 | 0.278 |
| SVM | 0.197 | 0.204 | 0.273 |

Table 2.2. MSWEB prediction results for various methods given 2,5 and 10 known features. The performance is based on the F1 measure.

As seen in Table 2.2, the three statistical modeling methods clearly outperform SVMs and Correlation. The poor performance of SVMs is apparently due to missing features, since SVM performance was observed to improve as we continued to increase $L$ (beyond 10). The FC method does surprisingly well. We attribute its success to the fact that $L$ is relatively small and the amount of data, $T$, is large. We also report on execution times on a Sunsparc10 workstation. For naive Bayes and FC, it took less than one second to compile the necessary statistics and less than 1 second to do one CF inference task with $K = 237$ (K is the number of test vectors in 1 trial). For Correlation, it took about 6 seconds to do 280 predictions on a *single* example (i.e., $K = 1$). For ME, with $L = 5$, it takes $< 1$ second to create the model, after which it takes $< 1$ second to

do one CF inference task (with $K = 237$). SVMs requires anywhere from 15 minutes to 2 hours to build each model. Each prediction takes on the order of milliseconds. Given the modest learning time for ME for small $L$, it does appear feasible to use this approach in an on-line, i.e. real-time setting.

For the EachMovie Database, only records with 40 or more rated movies were used for testing. There are 20,322 such records out of 61,264 total. Of these, 600 were randomly selected for the test set and 15,322 records were used as the training set. For each of the users in the test set, five sites were chosen at random for the known sites and 35 of the other sites were randomly chosen as sites to be predicted. A model was learned for each inference problem and then prediction was performed. Because MSWEB is an implicit database with no missing values, a model could be built and applied to all of the test vectors. The EachMovie database is an explicit database and every test vector has different missing features. The model assumes that the L known features are present and none are missing so one model that conditions on a given set of features could not be applied to the whole EachMovie test set. Therefore a different model had to be built for each of the 35 prediction tasks performed for each test vector. For this database, the possible ratings are "1",...,"6". We chose the rating with the highest probability as our prediction (an alternative is to use the expected value – these two methods give comparable results). All the results in Table 2.3 are for $L = 5$. We report the average absolute value of the prediction error, over K=21,000 predictions. The average absolute

value of the prediction error is defined as

$$\text{error}_{\text{av}} = \frac{1}{\text{K}} \sum_{\text{k=1}}^{\text{K}} |\text{f}_{\text{ki}*}^{(\text{n})} - \widehat{\text{f}_{\text{ki}*}^{(\text{n})}}| \tag{2.32}$$

where $\widehat{f_{ki*}^{(n)}}$ is the estimated value of $f_{ki*}^{(n)}$. A higher average absolute value of prediction error indicates greater deviation from the correct answer and poorer performance.

| Method | Result |
|---|---|
| CORR | 1.18 |
| NBM | 0.975 |
| MEM second best | 0.923 |
| MEM throw away data | 0.861 |
| MEM relearn model | 0.851 |

Table 2.3. EachMovie prediction results for various methods given 5 known features. The performance is based on the average absolute value of the prediction error

For the EachMovie data set, as for MSWEB, the Maximum Entropy method is seen to give appreciably better performance than the correlation method. By way of comparison, [9] gets an average absolute prediction of 1.139 for correlation as compared to our value of 1.18. A possible reason for this difference is that [9] uses a slightly different training/test split for EachMovie than we did. The two methods for handling missing features in ME models, throw away data and relearn model, further improve the accuracy of the predictions. The correlation method requires about 4 seconds to do

inference for one test vector consisting of 35 movies to be inferred. ME requires about 10 seconds to do learning and inference for one inference task.

## 2.5 Conclusion

Our new (ME) method falls into the class of 'statistical modeling' approaches to CF. As such, it has an inherent advantage over classification-based approaches (e.g. SVMs) in handling missing features in the training set. Even more importantly, unlike SVMs the new method does not require (heuristic) missing feature imputations when forming its predictions. Another advantage of the new method and other modeling approaches to CF lies in their scalability, both with increasing size of the feature space, but especially with increasing size of the database ($T$). Classification-based approaches will typically require a linear increase in *learning* complexity with increasing database size. As a more serious obstacle, memory-based methods will experience a linear increase in the complexity of the *prediction* phase. By contrast, the complexity of model-based approaches may have very little dependence on $T$. The complexity of estimating constraint probabilities via frequency counts does grow with $T$, but this takes very little time even for $T$ on the order of millions. More importantly, the complexity of ME learning of the *a posteriori* model is *independent* of $T$ if the sums required in iterative scaling are computed by summing over all elements of the known feature space, as in (2.19), rather than by summing directly over the training set, as in (2.18). ME model-building complexity thus may only grow with the number of *known* feature values and/or the number of ME constraints that are encoded. Thus, we expect that the new method will be more practically viable than memory-based methods for domains with huge databases

(e.g. millions of data records) and limited known attribute information (e.g. 5, 10, or 20 known feature values). Moreover, whereas the complexity of classification-based methods typically grows (at least) quadratically with the size of the feature space ($N$), complexity for the new method (and for other model-based approaches) grows (for each inference task) only with the size of the *known* feature subset ($L$). Even for spaces with thousands of features (e.g. the *EachMovie* domain), the size of the known feature subset may only be as large as ten or twenty features. While dimensionality reduction techniques can be used with classification-based methods, e.g. [31], one may need to sacrifice important information to achieve a practical dimensionality for learning and inference. Finally, compared with alternative statistical approaches, the advantage of the ME method lies in its improved inference accuracy. The new method was observed to achieve gains in the F1-measure and in absolute deviations over naive Bayes, as well as over SVMs and the Pearson correlation method. The gains over the other statistical models are attributable to the ME method's encoding of more constraints, as well as to its inherent avoidance of unnecessary statistical assumptions (e.g. independence).

# Chapter 3

## Latent Cluster Model for Collaborative Filtering

In the last chapter we looked at two statistical models suitable for collaborative filtering, the Naive Bayes Model and the Maximum Entropy Model. In this chapter we investigate a third model, the Latent Cluster model (LCM). This is a more powerful (mixture) model than naive Bayes, one that assumes features are conditionally independent given an unobserved (latent) 'cluster' variable, $M$. The joint probability (based on the known feature subset) is given by

$$\begin{aligned} &P[F_{i_1} = f_{i_1}, F_{i_2} = f_{i_2}, \ldots, F_{i_L} = f_{i_L}, F_{i^*} = f_{i^*}] = \\ &\sum_{m=1}^{N_c} P[M = m] P[F_{i^*} = f_{i^*} | M = m] \prod_{j=1}^{L} P[F_{i_j} = f_{i_j} | M = m]. \end{aligned} \tag{3.1}$$

It should be noted that Naive Bayes is a special case of this model. This is recognized by setting $M = F_{i^*}$ in (3.1). The associated *a posteriori* probabilities are again easily formed via Bayes rule.

$$P[F_{i^*} = f_{i^*} | f_{i_1}, f_{i_2}, \ldots, f_{i_L}] = \frac{P[F_{i_1} = f_{i_1}, F_{i_2} = f_{i_2}, \ldots, F_{i_L} = f_{i_L}, F_{i^*} = f_{i^*}]}{\sum_{k=1}^{|\mathcal{A}_{i^*}|} P[F_{i_1} = f_{i_1}, F_{i_2} = f_{i_2}, \ldots, F_{i_L} = f_{i_L}, F_{i^*} = k]}. \tag{3.2}$$

The different values of M can be thought of as representing typical preference patterns that people can exhibit. For instance, some people prefer action movies, some people like

romance, and other people like musicals. The use of the cluster variable M is an attempt to model these different tastes people have. The conditional independence assumption means that given a preference pattern, ratings of different movies are independent. The L movies the user has already rated give an indication as to which of the typical preference patterns this user has.

This model can be learned for each new CF task via maximum likelihood estimation (MLE). Since the model is not very large ($L \ll N$), MLE can often be practically accomplished 'on-line' [1]. Alternatively, a single cluster model can be learned, *off-line*, for the full $N$-dimensional space (or for a reduced-space determined by a feature selection strategy). We have implemented the learning via the EM algorithm [19]. Note also that one must choose the number of (latent) clusters, $N_c$. We have found that an improper choice can lead either to overfitting or to underfitting the data. Cluster models for CF were proposed in several previous works [32],[9].

## 3.1 EM Algorithm

A standard approach for solving MLE problems is the Expectation-Maximization (EM) algorithm [19]. 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 additionally postulate unobserved/*hidden* data associated with the problem at hand. The observed and hidden data together

[1] For L=5, it takes less than one second to do the learning and inference to perform one prediction task for either the EachMovie or MSWEB database on a SUN Ultrasparc 2 machine.

constitute the *complete data*, with an associated likelihood. The EM approach amounts to an iteration consisting of i) evaluation of the expected complete data likelihood (E-step) followed by ii) its maximization with respect to the model parameters (M-step). The log likelihood of the observed data increases monotonically at each iteration of the above steps. The EM algorithm finds a maximum for the log likelihood of the observed data, although this may only be a local maximum. Using EM, the parameters to be learned are the cluster probabilities $\{P[M = m]\}$ and the cluster-conditional feature probabilities, $\{P[F_i = f_i|M = m]\}$. The description below is for learning the full space model. Learning for a partial model is similar.

To develop the EM algorithm, we first write down the complete data likelihood based on the latent data variables M, and the observed data variables $(F_1, F_2, \ldots, F_N)$. Denote the complete data realization for database example n by $\underline{c^{(n)}} \equiv (m^{(n)}, f_1^{(n)}, f_2^{(n)}, \ldots, f_N^{(n)})$. Then the complete data likelihood is

$$\mathcal{L} \equiv P[\underline{c^{(1)}}, \underline{c^{(2)}}, \ldots, \underline{c^{(T)}}] = \prod_{n=1}^{T} (P[M^n = m] \prod_{j=1}^{N} P[F_i^n = f_i|M^n = m]) \tag{3.3}$$

and the expected value of the log of $\mathcal{L}$ is

$$E[\log \mathcal{L}] = \sum_{m=1}^{N_c} \{ \sum_{n=1}^{T} (\log P[M^n = m] + \sum_{i=1}^{N} \log P[F_i^n = f_i|M = m]) P[M = m|\underline{F^n}]\} \tag{3.4}$$

where by Bayes rule

$$P[M=m|\underline{F^n}] = \frac{P[M=m]\prod_{i=1}^{N} P[F_i^n = f_i|M=m]}{\sum_{m'=1}^{N_c} P[M=m']\prod_{i=1}^{N} P[F_i^n = f_i|M=m']}. \tag{3.5}$$

To maximize (3.4 ) over $P[F_i = f_i|M=m]$, differentiate (3.4) with respect to $P[F_i = f_i|M=m]$. Use a Lagrange Multiplier $\lambda$ to enforce the constraint that $\sum_{k=1}^{|A_i|} P[F_i = k|M=m] = 1$. We obtain the update equation

$$P^{(t+1)}[F_i = f_i|M=m] = \frac{\sum_{n=1|F_i^n=f_i}^{T} P^{(t)}[M=m|\underline{F^n}]}{\sum_{n=1}^{T} P^{(t)}[M=m|\underline{F^n}]}. \tag{3.6}$$

Similarly for P[M=m] the update equation is

$$P^{(t+1)}[M=m] = \frac{1}{T}\sum_{n=1}^{T} P^{(t)}[M=m|\underline{F^n}]. \tag{3.7}$$

## 3.2 Reduced Feature Space

For the EachMovie database there are N=1622 rated movies. Since this full space model would require a substantial amount of learning time, it is desirable to reduce the model size, and thus the learning time, while still providing the capability for rating (any) of the 1622 movies. This is accomplished as follows. First we can use the EM learning approach to build a model for a feature space of reduced dimensionality $K < N$. For example we can reduce the space by selecting the top (the ones most often rated) 300 movies (K=300). Below $F_{i_j}$ denotes the subset (subsequence) of K movies used. The

resulting model cannot be (directly) used to predict any of the remaining 1322 movies. However, we can provide ratings for the remaining movies in the following way. Suppose we wished to augment our learned, reduced-space model to include one additional feature $F_{new}$ (from the remaining 1322). The complete data likelihood function for the augmented feature space that includes $F_{new}$ is

$$\mathcal{L} = \prod_{n=1}^{T} P[M^n = m] P[F^n_{new} = f_{new} | M = m] \prod_{j=1}^{K} P[F^n_{i_j} = f_{i_j} | M = m]. \tag{3.8}$$

To find $P[F_{new} = k | M = m]$, we first find $P[M = m | \underline{F^n}]$ from the existing model for each training set vector $F^n$. From Bayes rule this is

$$P[M = m | \underline{F^n}] = \frac{P[M = m] \prod_{j=1}^{K} P[F_{i_j} = f^n_{i_j} | M = m]}{\sum_{m'=1}^{N_c} P[M = m'] \prod_{j'=1}^{K} P[F^n_{i_j} = f_{i_j} | M = m']}. \tag{3.9}$$

Now the maximum likelihood value for $P[F_{new} = k | M = m]$ is derived in a manner similar to (3.6), and the result is

$$P[F_{new} = k | M = m] = \frac{\sum_{n=1 | F^n_{new} = k}^{T} P[M = m | \underline{F^n}]}{\sum_{n'=1}^{T} P[M = m | \underline{F^{n'}}]}. \tag{3.10}$$

Because $P[F_{new} = k | M = m]$ does not appear in the right side of (3.9) no iteration is needed in computing these values. In this way we can augment the original reduced-space model to find parameter values for the remaining 1322 movies.

## 3.3 Machine Learning Issues in CF

### 3.3.1 On-line local vs. off-line global models

One way of dealing with the large problem space is to learn a model on-line specifically suited to solving the inference task at hand. Regarding features missing during inference, the theorem first mentioned in section 2.3 indicates that when features are missing in a classification system, the marginal distributions over just the presented features should be used. Any efforts to estimate or assume values for absent features are unnecessary. Such attempts will not statistically improve the accuracy of the classification, compared to the use of marginal distribution functions . This theorem says that if we knew the *true* marginal distribution we will get optimal results. Unfortunately in practice we have to estimate the distribution from a training set. There are two possible learning approaches suggested below.

1) Learn a model only over the presented space. Assume the training set only includes the target attribute and the features being conditioned upon. To learn the parameters, perform maximum likelihood estimation over just this subset of the training data. Because of the greatly reduced dimensionality, learning will often be possible in an on-line/real-time setting, i.e. the model can be learned in seconds. This model is tailored to the inference task at hand and can be discarded following its use.

2) Learn a model over a larger (e.g. the full feature) space, and then marginalize out the unknown features for the given inference task at hand.

The Naive Bayes and cluster models both learn a joint probability mass function which can be easily marginalized to get the joint pmf required for an individual task.

Therefore the whole model can be learned off-line and then easily marginalized to solve any particular inference problem. Instead of learning the whole model, one can also apply our method of learning a partial model over a subset of the features and then extending this to the remaining features via eq (3.10). Both of these approaches, learning a partial model for each query, and learning the whole model and 'marginalizing out' the features not in the query are investigated in the sequel.

### 3.3.2 Implicit vs. Explicit databases and Missing votes

As mentioned in section 2.3.2.9 missing votes in explicit databases can cause a mismatch between the model learned and the inference task to be performed, and this mismatch can adversely affect classification performance. One learning approach for handling missing feature values is the following:

#### 3.3.2.1 Missing as a feature value

One possibility is to treat missing as another feature value. For EachMovie, the possible ratings would be the 6 possible values, 1..6 and another for "unrated", for a total of seven possible ratings. In this database, more than half the movies are unrated and therefore, "unrated" is the most likely classification. However our objective is to provide accurate ratings, not to predict whether or not a movie has been rated. Thus with this approach, if unrated is the most likely prediction, we must instead choose the second most likely rating. This method is used in [9] and in the next section.

## 3.4 Basic Experimental Comparison of CF Approaches

The Latent Cluster Model was tested on the same two databases as the Maximum Entropy Model of the last chapter and in the same manner. The results are given in tables 3.1 and 3.2. The results for the LCM were very close to the MEM. When learning the cluster model for the MSWEB database, the whole feature space was used. The method of section 3.2 was not needed for MSWEB because the feature space is not very large, K=285 as opposed to K=1622 for EachMovie. Once again, in table 3.1 the results are reported using the recall/precision metric. For this metric a higher value means better performance.

| | Number Features Present | | |
|---|---|---|---|
| Method | Given 2 | Given 5 | Given 10 |
| CORR | 0.198 | 0.195 | 0.278 |
| SVM | 0.197 | 0.204 | 0.273 |
| NBM | 0.317 | 0.303 | 0.326 |
| LCM | 0.316 | 0.319 | 0.319 |
| MEM | 0.331 | 0.329 | 0.332 |

Table 3.1. Results for MSWEB, higher values are better (MEM included for comparison)

For the EachMovie database the mean absolute error is reported. For this table a lower result is better. The results for EachMovie show the same trends as for MSWEB. In both cases statistical modeling methods (latent clustering, maximum entropy and naive bayes) outperformed the other two methods (SVMs and correlation). In both cases

| Method | Number Features Present | | | |
|---|---|---|---|---|
| | Given 2 | Given 5 | Given 10 | Given 20 |
| CORR | 2.18 | 1.18 | 1.057 | - |
| SVM | - | - | - | 1.46 |
| NBM (p) | 0.945 | 1.099 | 1.342 | - |
| NBM (s) | 0.900 | 0.886 | 0.886 | - |
| LCM | 0.855 | 0.839 | 0.830 | - |
| MEM relearn model | - | 0.851 | - | - |

Table 3.2. Results for EachMovie, lower values are better (MEM included for comparison, this method was only done for Given 5)

correlation and SVMs improved as more features were added. We attribute the poor performance of SVMs to the many missing features during inference, as the performance improved as more features were known. The SVM results for EachMovie were copied directly from [25]. For the latent clustering method, the learning outlined in section 2.5 requires that K, the number of movies in the model initially learned by EM, be chosen. We experimented with different values from K=30 to K=300. We obtained the best results (lowest absolute mean error) for K=50. For higher and lower values of K the performance on the test set suffered. This may be due to the Hughes effect [34] which states that for finite training sets, there is an optimal size for the model being learned. All our tests were done with K=50, where the 50 chosen movies were ones most often rated of the 1622 total movies.

In contrast to the other classification methods we tested, the results for Naive Bayes worsened as we conditioned on more features. These results are listed as NBM (p) in table 3.2 where the p denotes the product rule. To explain this note that the Naive

Bayes joint pmf is given as

$$P[F_1, F_2, \ldots, F_K, F_{i*}] = P[F_{i*}] \prod_{j=1}^{K} P[F_j|F_{i*}] \tag{3.11}$$

which will be referred to as the product rule for combining conditional probabilities. An alternative means of combining the conditional probabilities is

$$f(F_1, F_2, \ldots, F_K, F_{i*}) = P[F_{i*}] + \sum_{j=1}^{K} P[F_j|F_{i*}], \tag{3.12}$$

known as the sum rule. While the sum rule does not correspond to a probability function, its consideration facilitates understanding the problem with the Naive Bayes Model, as we now explicate. First the measured conditional probabilities are modeled by their true (and unknown) value plus a noise term. Following the development in [41], if $P[F_{i*}|F_j]$ is the true value of the conditional probability and our measured value is $P[F_{i*}|F_j] + \epsilon_{i*j}$, then we can approximate the product term in (3.11) by

$$\prod_{j=1}^{K} (P[F_j|F_{i*}] + \epsilon_{i*j}) = \prod_{j=1}^{K} P[F_j|F_{i*}]) \prod_{j=1}^{K} (1 + \frac{\epsilon_{i*j}}{P[F_j|F_{i*}]}). \tag{3.13}$$

Dropping all terms of $\epsilon_{i*j}$ of power greater than 1 gives

$$\prod_{j=1}^{K} (P[F_j|F_{i*}] + \epsilon_{i*j}) \simeq \prod_{j=1}^{K} P[F_j|F_{i*}](1 + \sum_{j=1}^{K} \frac{\epsilon_{i*j}}{P[F_j|F_{i*}]}), \tag{3.14}$$

which leads to the following approximation,

$$P[F_{i*}]\prod_{j=1}^{K}(P[F_j|F_{i*}]+\epsilon_{i*j}) \simeq (P[F_{i*}]\prod_{j=1}^{K}P[F_j|F_{i*}])(1+\sum_{j=1}^{K}\frac{\epsilon_{i*j}}{P[F_j|F_{i*}]}). \quad (3.15)$$

The sum rule can be rewritten as

$$P[F_{i*}]+\sum_{j=1}^{K}(P[F_j|F_{i*}]+\epsilon_{i*j}) = P[F_{i*}]+(\sum_{j=1}^{K}P[F_j|F_{i*}])(1+\frac{\sum_{j=1}^{K}\epsilon_{i*j}}{\sum_{j=1}^{K}P[F_j|F_{i*}]}). \quad (3.16)$$

$$= P[F_{i*}]+(\sum_{j=1}^{K}P[F_j|F_{i*}])(1+\sum_{j=1}^{K}\frac{\epsilon_{i*j}}{\sum_{j'=1}^{K}P[F_{j'}|F_{i*}]}). \quad (3.17)$$

Comparison of the terms $(1+\sum_{j=1}^{K}\frac{\epsilon_{i*j}}{P[F_j|F_{i*}]})$ for the product rule and $(1+\sum_{j=1}^{K}\frac{\epsilon_{ij}}{\sum_{j'=1}^{K}P[F_{j'}|F_{i*}]})$ for the sum rule show the difference of the effect of noise on the two rules. Specifically the product rule has a greater noise effect because the noise is divided by a probability where the sum rule divides the noise by a sum of probabilities.

## 3.5 Further Possibilities for Missing Features

In this section we investigate some other ways of handling the missing feature problem, aside from picking the second most likely rating as was done in the last section.

### 3.5.1 Threshold the ratings

Another method, used in [1], [32] and [7], is to threshold the data to binary values, 0 and 1. This effectively changes the database from an explicit database to an implicit

one. This eliminates missing features altogether but it also eliminates the ability to predict ratings of 1..6. We did not try this method.

### 3.5.2 Labeled, Unlabeled data for learning on-line models

We propose here a method that can be used for learning models tailored for a single inference task. This approach is based on a classification method first proposed in [47] (the MU model), and since found quite useful in several other PR contexts, [40]. In utilizing this procedure, we view $f_{i*}$ as the class label for a given classification problem. Missing values for $f_{i*}$ (the fact that some users have not rated this particular movie) can be viewed as unlabeled data in the context of a classification problem. It is possible to build a model using only the labeled data in this context and so learn a model that predicts only valid ratings of 1-6. In this case all the unlabeled data (which can be more than half of the training set) is wasted. Using *all* of the data should result in more accurate estimation of model probabilities.

To use this approach, we hypothesize that the unlabeled data (corresponding to users that have not rated the target attribute) simply have no labels, and change the complete data likelihood equation accordingly where $\chi_l$ is the labeled data and $\chi_u$ is the unlabeled data.

$$\mathcal{L} = \prod_{n \in \chi_l} [P[M^n = m] P[F^n_{i*} = f_{i*} | M^n = m] \prod_{j=1}^{L} P[F^n_{i_j} = f_{i_j} | M^n = m]]$$

$$\prod_{n \in \chi_u} (P[M^n = m] \prod_{j=1}^{L} P[F^n_{i_j} = f_{i_j} | M^n = m]).$$

Note that the class label parameters only appear in the terms belonging to $\chi_l$. We again use the EM algorithm to maximize the log likelihood of the data. The Expectation step for the cluster variables is

$$P^{(t+1)}[M=m|\underline{F^n}] = \frac{P^{(t)}[M=m]P^{(t)}[F_{i^*}=f_{i^*}|M=m]\prod_{j=1}^{L}P^{(t)}[F_{i_j}=f_{i_j}|M=m]}{\sum_{m'=1}^{N_c}P^{(t)}[M=m']P^{(t)}[F_{i^*}|M=m']\prod_{j=1}^{L}P[f_{i_j}|M=m']} \tag{3.18}$$

if the class label is rated for this user n and

$$P^{(t+1)}[M=m|\underline{F^n}] = \frac{P^{(t)}[M=m]\prod_{j=1}^{L}P^{(t)}[F_{i_j}=f_{i_j}|M=m]}{\sum_{m'}P^{(t)}[M=m']\prod_{j=1}^{L}P^{(t)}[F_{i_j}=f_{i_j}|M=m']} \tag{3.19}$$

if it is not. To learn the class probabilities use only the labeled data.

$$P^{(t+1)}[F_{i^*}=k|M=m] = \frac{\sum_{n\in\chi_l|F_{i^*}=k}P^{(t)}[M=m|\underline{F^n}]}{\sum_{n\in\chi_l}P^{(t)}[M=m|\underline{F^n}]} \tag{3.20}$$

To learn the other feature probabilities use both the labeled and unlabeled (all of the) data.

$$P^{(t+1)}[F_{i_j}=k|M=m] = \frac{\sum_{n=1|f_{i_j}=k}^{T}P^{(t)}[M=m|\underline{F^n}]}{\sum_{n=1}^{T}P^{(t)}[M=m|\underline{F^n}]} \tag{3.21}$$

The update equation for $P[M = m]$ is

$$P^{(t+1)}[M = m] = \frac{1}{T} \sum_{n=1}^{T} P^{(t)}[M = m|\underline{F^n}] \tag{3.22}$$

This method gives improved performance over on-line latent cause clustering using the second best method as shown in table 3.3. However comparison of table 3.3 and table 3.4 shows that (off-line) learning a larger part of the model is superior to on-line learning of a very small part. Although this seems a reasonable result, at present we have no good explanation for this. The obvious question is then, do we have a method for dealing with missing class features when we learn the whole model?

### 3.5.3 Missing Features for the Whole Model

When learning the whole model, there is no single class label that can be used to separate the data into labeled and unlabeled subsets, therefore the method in [47] cannot be applied to whole model learning. This is because we know before we perform on-line learning which feature is the class variable, and so we can divide the data set into labeled and unlabeled subsets for learning. To learn a whole model to be used for inference on any possible feature, we do not know which feature is the class label before learning and so no such division is possible for learning. One way that missing data can be accounted for is to change the likelihood equation so that it only includes features that are rated. In the expectation step the product is only taken over the features that are rated, leading to a different number of features in the product terms for each user. In this case the likelihood of the data is

$$L = \prod_{n=1}^{T} [P[M^n = m] \prod_{j=1|F_j \neq \text{unrated}}^{N} P^n(F_j = f_j|M = m)]$$

For this method the expectation step is a product only over the rated items for each user,

$$P[M = m|\underline{F^n}] = \frac{P[M = m] \prod_{j=1|F_j \neq \text{unrated}}^{N} P[F_j = f_j|M = m]}{\sum_{m'=1}^{N_c} P[M = m'] \prod_{j=1|F_j \neq \text{unrated}}^{N} P[f_j|M = m']} \tag{3.23}$$

and values for $P[F_j = f_j|M]$ just for $f_j \neq$ unrated are calculated in the maximization step.

$$P[F_i = f_i|M = m] = \frac{\sum_{n=1|F_i=f_i}^{T} P[M = m|\underline{F^n}]}{\sum_{j=1}^{T} P[M = m|\underline{F^n}]} \tag{3.24}$$

We have seen modest improved performance over the second-best method using this approach, as shown in table 3.4.

## 3.6 Conclusion

The latent cluster model worked well on the two CF databases to which it was applied. Its performance was better than the Maximum Entropy Model of the last chapter. Examination of table 3.1 for MSWEB and table 3.2 for EachMovie show this.

| Methods | Mean Absolute Error |
|---|---|
| Second-best | 0.91 |
| Miller-Uyar | 0.88 |

Table 3.3. On-line learning results for EachMovie

| Methods | EachMovie |
|---|---|
| Second-best | 0.839 |
| Only Rated | 0.836 |

Table 3.4. Whole model results forEachMovie

Applied to EachMovie the Latent Cluster Model took about 1 hour to learn, and once it was learned, it took on the order of milliseconds to do a prediction. The Latent Cluster model has been applied to CF by other people [9], [32], [51]. In [9] it appears that they used the same model that was used here, but the results obtained were not the same as ours. They got about the same results for correlation, (1.139) as for LCM (1.14). Our results here show a 25% decrease in the mean absolute error for LCM over correlation. Since the details in [9] for evaluating the Latent Cluster model are very sparse, it is not possible to determine why they obtained such poor results for this method. In their paper they state that they learned a model for the 300 most popular movies and then tested the model on all 1622 movies. What they did to produce inferences for the 1322 movies not in their model is unclear. Another approach to learning this model is reported in [13]. A latent cluster model utilizing a Markov Chain Monte Carlo learning scheme

was used. The average absolute error was reported there to be 1.26. We have seen no results comparable to ours. The combination of good prediction performance and very fast predictions make this a very capable inference method.

# Chapter 4

## Labeled and Unlabeled Databases and Class Discovery

As mentioned in the introduction, the MU mixture model is able to learn from a combination of labeled and unlabeled data. That model assumes that all of the unlabeled data comes from the same set of classes as the labeled data. In this context the unlabeled data serves to improve the parameter estimates over what would have been achieved using just the labeled data. Here, we consider the (reasonable) scenario in which unlabeled points may belong either to known/predefined *or* to heretofore *undiscovered* classes. We propose a novel statistical mixture model which views as observed data not only the feature vector and the class label, but also the *fact* of label presence/absence for each point. Two types of mixture components are posited to explain label presence/absence. 'Predefined' components generate both labeled and unlabeled points, and they assume missing class labels are *missing at random*. 'Non-predefined' components only generate unlabeled points – thus, in localized regions, they capture data subsets that are *exclusively* unlabeled. Such subsets may represent an outlier distribution, i.e. new classes. The predefined/non-predefined character of each component is *data-driven*, learned along with the other parameters via an expectation-maximization (EM) algorithm. The resulting model yields *a posteriori* inferences over the predefined classes, assuming the feature vector originates from a known class. However, it also yields inferences on whether *or not* a feature vector originates from any of the known classes. There are three natural

applications: 1) robust classifier design, given a mixed labeled/unlabeled training set with outliers; 2) classification with rejections, where our model provides both a built-in rule and a good (unknown class) explanation for rejecting samples; 3) the identification of the unlabeled points (and their representative components) that do not originate from known classes. 3) is a step towards new class discovery. Experiments are reported for each of the indicated applications, including topic discovery for the *Reuters* domain.

## 4.1 Introduction

Consider the two-dimensional example data set shown in Figure 1a. The points are represented by symbols which reflect both the underlying data classes and the data set owner's *knowledge* of these classes. There are three *predefined* classes, characterized by being known *a priori* to the user. This class set is represented by the symbol set $\mathcal{P}_c = \{1, 2, 3\}$. Data points that come with a known labeling are accordingly indicated in the figure. There are also unlabeled points. Some of these points originate from one of the predefined classes and are accordingly represented in the figure by $U_1$, $U_2$, or $U_3$[1]. Other unlabeled points actually belong to classes that are wholly *unknown* to the data owner. In this example there is one such class, i.e. $\mathcal{P}_u = \{4\}$, with these points represented in the figure by $U_4$. Note that, as indicated in the figure, classes may be composed of more than one compact group of data, i.e. more than one 'cluster'.

The two key attributes of this data set are 1) its mixed labeled/unlabeled character and 2) the fact that some classes are exclusively unlabeled. When all points in the data

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

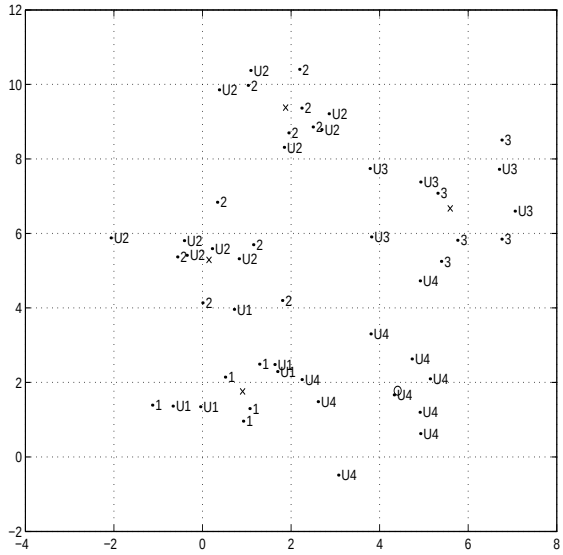


Fig. 4.1. Illustrative Example

set that originate from a particular ground truth class are unlabeled, we will say that *the data set contains an unknown class.* Note that this definition of an 'unknown' class covers two different scenarios that may be of practical interest:

i) where the classification problem at hand involves some classes that are defined for the domain, but for which only unlabeled data has been made available. For example, consider a character recognition domain where all training 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 data labeler randomly selected a fraction of the points to label, and happened not to select any instances of "z";

ii) where the training 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 data set of galaxy instances which includes unlabeled examples that originate from a galaxy type that has not yet been discovered by astronomers. In this case, we would say the new galaxy type is an unknown class, both with respect to the training set *and* more generally.

*Reasons for Missing Labels*

There are several good reasons why, in practice, some (often many) data may be missing labels. Providing supervising labels is a time-consuming activity that in some domains may require substantial expertise and hence expense (Consider, for example, a radiologist's hourly wage.). Moreover, even if expertise is not required, it may simply be too tedious a process for a human to ascertain supervising labels for even a significant subsample from a large database, e.g. a text article archive with 100,000 documents. Also,

if the data set contains outliers, the labeler may be unable to provide appropriate labels for these samples and may thus choose to leave them unlabeled.

Finally, consider a *transductive inference* approach [39],[48]. This approach is motivated by methods for learning classifiers based on both labeled and unlabeled data, e.g. [57],[47],[40]. In [47],[48], it was observed that the ability to effectively incorporate unlabeled data within classifier training means that one can in fact incorporate in this fashion *any* new unlabeled data, including the (test/use) samples that need to be classified. Thus, a *combined learning and use* paradigm was proposed. For each new batch of data to classify, this method augments the existing training set with the new (unlabeled) data batch and then relearns the classifier. The resulting model is then used to classify the new batch. In addition to increasing the size of the training set, this approach can adapt the classifier to a (possibly) changing data distribution. This type of approach has also been referred to as *transductive inference* and applied to the training/use of support vector machines [39]. Since a labeled training set and an unlabeled test/use batch to be classified is a *standard* situation, one can argue that mixed labeled/unlabeled training data is ubiquitous, encountered whenever one chooses a transductive approach. This important scenario will be addressed in the sequel.

*Reasons for Unknown Classes*

As suggested before, the fact that many points have not been labeled may also account for the presence of unknown classes – if a significant fraction of a large database is unlabeled (with, in the case of random selection by the labeler, each training instance equally likely to have the label missing), it is quite possible that, simply by chance, all

instances from some of the classes may be unlabeled [2]. Unknown classes 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. As one example, the land use classes associated with multispectral data may depend on whether the land use is agrarian, urban, industrial, or military. As a second example, data/text/images elicited from an Internet search are likely to provide examples from heretofore unseen topics/classes. Unknown classes may also arise in domains where there is uncertainty or disagreement 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. [58]. Another example is topic categorization for text documents [22], or other domains where there is some degree of subjectivity in the choice of classes. Finally, as suggested earlier, unknown classes may arise in scientific domains when their existence is inconsistent with current theory.

Data sets of this type, with mixed labeled/unlabeled character and with some unknown classes, are germane to a number of fundamental pattern recognition tasks, including robust classifier learning, outlier detection, sample rejection and, prominently, new class discovery. In this chapter, we propose novel statistical models, along with associated maximum likelihood estimation (MLE) learning algorithms and model-based statistical inference, tailored for these types of data sets and directly applicable to all of the aforementioned tasks. In particular, our models address each of the problems next identified and described.

[2] By the same token, the fact that some classes are in general unknown may likewise explain why some points are missing labels (it may reflect the inability of the labeler to provide an appropriate label for points from these classes).

### 4.1.1 Robust Classifier Design, Given a Mixed Training Set

Several authors have recently proposed use of unlabeled data, along with labeled training samples, when learning a statistical classifier [57],[18],[60],[47],[40],[8]. The problem has also been studied theoretically in [10]. For classifiers based on an underlying statistical model and MLE for model learning, it has been found that augmenting a small pool of labeled data with a substantially larger pool of unlabeled data can improve accuracy in the estimation of the class-conditional densities, and hence in the classification performance[57],[47],[40] [3]. Unlabeled data has also been used effectively in other classifier learning approaches, e.g. [8]. These works essentially assume that all unlabeled samples originate from one of the *known* classes, defined for the given classification domain. Moreover, they effectively assume no points are outliers. However, as discussed before, the presence of unlabeled samples may suggest that one or both of these assumptions is faulty – in either case, the labeler may have had difficulty in labeling some of the data. It is well-known [33] that even a few outliers can have a dramatic, deleterious effect on estimation. Thus, there is strong motivation for *robust classifier learning for mixed labeled/unlabeled data sets.* Here, for a statistical modeling approach to classification (over the known classes), we seek to minimize the contribution of outliers or points from unknown classes, when learning the joint density over features and known classes.

This problem is related to the *unsupervised* learning problem of robust clustering, e.g. [17]. In this context, our work is perhaps closest to [16] where a 'noise' cluster was

[3]The use of unlabeled data does not strictly lead to improvements in performance. If the labeled sample size is sufficiently large or if the ratio of unlabeled to labeled samples is inadequate, use of unlabeled data has been found in some cases to degrade performance, e.g. [35]. Some explanation for this is given in [14].

introduced to capture outliers, to ensure they do not contaminate true clusters. Our approach in some sense introduces *multiple* such clusters, within a mixed labeled/unlabeled framework. However, unlike [16], where the outlier distribution is not accurately modeled, our approach aims to simultaneously accurately model both the known class distribution and the outlier distribution. This allows our method to address modeling and inference tasks both for the known classes and for the unknown classes.

In the semi-supervised learning context, [40] gave a different (constant) weight to unlabeled samples than to labeled ones. However, this approach treats all unlabeled samples in the same way, i.e. it does not identify outliers within the unlabeled subset and diminish their impact on estimation. A method closer in spirit to ours is [59]. This is a mixture modeling approach for mixed data similar to [57] 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 [59] is that, unlike [59], we treat the fact that a class label is missing *as observed data*. We provide both random and deterministic (new class) explanations for missing labels. Our approach is well-matched to a data set that contains some unknown classes. Thus, if the data truly contains unknown classes, our approach should yield more accurate classifiers than [59]. A perhaps more significant difference is that, unlike [59], we explicitly model the new classes/outliers. Thus, our approach can also be used to tackle new class discovery.

### 4.1.2 Classification With a Special Sample Rejection Mechanism

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. Here, a simple classifier is first applied. If a reliable decision cannot be reached, the simple classifier rejects the sample, triggering a more complex second stage classifier. Likewise, rejection at the second (or a subsequent) stage will trigger an even more complex next stage classifier. If rejections are infrequent, then the average computational requirements of the system are modest. In this work, we suggest rejecting samples for a different reason, i.e. *under the hypothesis that they are either outliers or belong to new, undiscovered classes.* Our model directly provides *a posteriori* probabilities for the binary choice of a 'predefined class' or 'new class' origin for a given data point. Thus, it naturally provides a mechanism for rejecting samples, along with an 'unknown class' explanation for this decision.

### 4.1.3 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, with applications in scientific domains such as molecular biology [58],[3], as well as in text categorization and remote sensing [38]. We can also envision other applications, e.g. in image and video-based object classification, where the plethora of possible objects and scenes makes unknown objects and classes quite likely. Several different class discovery problems and techniques have been previously proposed. [11] proposed to overcome uncertainty in the class definitions by

allowing non-mutually exclusive classes, so that points can be owned by unions of defined classes (effectively forming new classes). [3] considered gene classification based on expression profiles. The authors defined a problem that is similar to standard clustering but which allows *multiple* clusterings of a data set. The idea here is that different genes 'code' for different sets of classes. Somewhat closer to our work is [58], 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 representative class 'codeword'. Since the data points are assumed to all be labeled, each cluster (a potential new class) owns points from at least one, and quite possibly several predefined classes. A cluster's codeword is the average of the predefined class codewords that it owns [5]. A model complexity term is used to control the number of clusters. This method effectively 'distrusts' the labels given to some (perhaps many) data samples. Moreover, it postulates that the existing class definitions are inadequate for representing these samples. Thus, new clusters (effectively, new classes) are introduced to better represent the data. One main difference between our work and [58] is that we consider a semi-supervised framework, with new classes corresponding to purely unsupervised data subsets, rather than to mislabeled ones. Our statistical model could, however, be suitably modified for the scenario in [58], to also possibly account for mislabeled data and suboptimal class definitions.

[4] For example, with three predefined classes, the predefined 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 points from the first two predefined classes (and the same number from each).

Finally, class discovery was considered in a semi-supervised setting in [38]. This approach assumes there is only a *single* predefined class and that its density function is known, *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 [38] are: 1) we treat the fact of label presence/absence as observed data; 2) we do not restrict to a single predefined class; moreover, we *learn* the predefined class densities, rather than assuming them known *a priori*; 3) while [38] is a heuristic approach, we develop a unified statistical framework that models both the predefined and new classes. Moreover, we use a single (EM) learning approach which performs hillclimbing in the model's likelihood function. The method we propose requires that the data set be of mixed labeled/unlabeled character, with the data from unknown classes purely unlabeled. One might question whether this scenario is commonly encountered in practice. However, earlier we described several situations where such data may arise, including transductive inference [6].

We view class discovery as consisting of two (stratified) goals. The first is to identify the subset of points which do not belong to predefined classes. For our model, this objective is addressed using the same inference rule as for sample rejection, described in the previous subsection. A more ambitious objective is to *validate* the components which own these 'unknown class' points. Finding a set of valid components for these points will not wholly solve new class discovery. However, it will provide good candidates

[6] The assumption that the unknown class data is purely unlabeled will not always hold – some labeled training points may be unknown class points that were improperly labeled. To address this situation, our approach may require extension along the lines of [58].

for new classes that can be forwarded to a domain expert, for further consideration. Both of these goals will be addressed in the sequel. In Section 2, we develop our generative mixture models for mixed labeled/unlabeled data sets. In Section 3, we introduce a method based on the expectation-maximization (EM) algorithm [19] 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 on our experimental results. We conclude with some directions for future work.

## 4.2 A Mixture Model for Labeled/Unlabeled 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[7]. This mixed data scenario was considered for classification in [57], where a class-partitioned mixture model was learned, via MLE, to fit the mixed data, $\mathcal{X}_m$. In [49], [47], a mixture of experts model was proposed. Unlike [57], this model does not assign mixture components exclusively to individual classes, but rather allows components to be *probabilistically* associated with each of the classes. These probabilistic assignments are learned, along with the other model parameters, via an EM algorithm. This approach was found to achieve somewhat better results [49] and to be less sensitive to initialization than [57]. The mixture of

[7]For concreteness, in the subsequent development we will consider feature vectors $\underline{x} \in \mathcal{R}^n$. However, our methodology can also be applied e.g. to categorical features, as in [40].

experts model and EM approach [48] was also used for document classification in [40]. Other recent works have applied this type of semi-supervised learning approach to face recognition [2] and to time series data [35], based on a mixture of hidden Markov models.

### 4.2.1 Formulation

All these prior works assume unlabeled samples originate from one of the predefined classes. As in [38], we allow points 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 $\mathcal{X}_l = \{(\underline{x_1}, \text{“l”}, c_1), (\underline{x_2}, \text{“l”}, c_2), \ldots, (\underline{x_{N_l}}, \text{“l”}, c_{N_l})\}$ and $\mathcal{X}_u = \{(\underline{x_{N_l+1}}, \text{“m”}), \ldots, (\underline{x_N}, \text{“m”})\}$. Here we have introduced, for each sample, the *new* random observation $\mathcal{L} \in \{\text{“l”}, \text{“m”}\}$, taking on one of two categorical values indicating the sample is either *labeled* or *missing* the label. We will propose mixture-based statistical models that explain *all* the observed data, including the fact of label presence/absence for each sample. For each of our (two) proposed 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 points, and they assume missing class labels are *missing at random*. “Non-predefined” components *deterministically* generate unlabeled points – thus, in localized regions, they capture data subsets that are *exclusively* unlabeled. Such subsets may represent an outlier distribution, i.e. new classes.

*Notation*

Let $\mathcal{M}_k$, $k = 1, \ldots, M$ denote the $k$th mixture component, with $M$ the number of components. Let $\mathcal{C}_{\text{pre}}$ denote the subset of 'predefined' components, with the remaining subset denoted $\bar{\mathcal{C}}_{\text{pre}}$. Let $C \in \mathcal{P}_c$ be a random variable defined over the predefined classes, with $c(\underline{x}) \in \mathcal{P}_c$ the class label paired with data sample $\underline{x}$. As aforementioned, let $\mathcal{L} \in$ ("l", "m") be a random variable indicating, respectively, whether or not a given sample is labeled. Let $\alpha_k$ denote the prior probability for component $k$ with $\sum_{k=1}^{M} \alpha_k = 1$, $\theta_k$ 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 function family for all mixture components.

### 4.2.2 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. Prob[$\mathcal{L}$ = "l"|$\mathcal{M}_g \in C_{\text{pre}}$], 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.

2) The probability that the class label is $C = c$ given that the sample is generated by component $k$, a predefined component, and given that a class label is produced, i.e. Prob[$C = c|\mathcal{M}_k \in \mathcal{C}_{\text{pre}}, \mathcal{L} =$ "$l$"].

In summary, our first statistical model is based on the *composite parameter set* $\Lambda_1 = \{\{\alpha_k\}, \{\theta_k\}, \mathcal{C}_{\text{pre}}, \{\text{Prob}[\mathcal{L}|\mathcal{M}_g \in C_{\text{pre}}]\}, \{\text{Prob}[C|\mathcal{M}_k \in \mathcal{C}_{\text{pre}}, \mathcal{L} =$ "$l$"$]\}\}$. Note that $\mathcal{C}_{\text{pre}}$ is included in $\Lambda_1$. This reflects the fact that $\mathcal{C}_{\text{pre}}$, along with the rest of $\Lambda_1$, must be chosen/learned.

*Hypothesis for Random Generation of the Data*

Our first statistical model hypothesizes that each sample from $\mathcal{X}_m$ is generated independently, based on $\Lambda_1$, according to the following stochastic generation process:

i) Randomly select a mixture component $\mathcal{M}_j$ according to $\{\alpha_k\}$.

ii) Randomly select a feature vector $\underline{x}$ according to $f(\underline{x}|\theta_j)$.

iii) If $\mathcal{M}_j \in \mathcal{C}_{\text{pre}}$, then randomly select the fact of label presence/absence $v \in \{\text{“l”}, \text{“m”}\}$ according to the pmf $\text{Prob}[\mathcal{L} = v|\mathcal{M}_g \in \mathcal{C}_{\text{pre}}]$.

iv) If $\mathcal{M}_j \in \mathcal{C}_{\text{pre}}$ and $v =$ “l”, then randomly select a predefined class label according to the pmf $\text{Prob}[C|\mathcal{M}_j \in \mathcal{C}_{\text{pre}}, \mathcal{L} = \text{“l”}]$.

v) If $\mathcal{M}_j \in \bar{\mathcal{C}}_{\text{pre}}$, set $v =$ “m”.

Note that with this model, we have $\text{Prob}[\mathcal{L} = \text{“m”}|\mathcal{M}_g \in \bar{\mathcal{C}}_{\text{pre}}] = 1$, i.e. non-predefined components *deterministically* explain missing labels, whereas predefined components hypothesize labels missing at random.

*Joint Data Likelihood Function*

Let

$$v_k = \begin{cases} 1 & \text{if } \mathcal{M}_k \in \mathcal{C}_{\text{pre}} \\ 0 & \text{if } \mathcal{M}_k \in \bar{\mathcal{C}}_{\text{pre}} \end{cases}$$

Then, the log of the joint data likelihood associated with our first mixture model is

$$\log L_m = \sum_{\underline{x} \in \mathcal{X}_u} \log \left( \sum_{k=1}^{M} \alpha_k f(\underline{x}|\theta_k)((1 - v_k) + v_k \text{P}[\mathcal{L} = \text{“m”}|\mathcal{M}_g \in \mathcal{C}_{\text{pre}}]) \right) + \sum_{\underline{x} \in \mathcal{X}_l} \log \left( \sum_{k=1}^{M} v_k \alpha_k f(\underline{x}|\theta_k) \text{P}[\mathcal{L} = \text{“l”}|\mathcal{M}_g \in \mathcal{C}_{\text{pre}}] \text{P}[C = c(\underline{x})|\mathcal{M}_k \in \mathcal{C}_{\text{pre}}, \mathcal{L} = \text{“l”}] \right). \tag{4.1}$$

### 4.2.3 Model II: Class-conditional Missing Mechanism

The first model assumes a common missing label pmf $\text{Prob}[\mathcal{L}|\mathcal{M}_{\text{g}} \in \mathcal{C}_{\text{pre}}]$ for *all* predefined components. This assumption is consistent with the 'large database' scenario, where, from the pool of samples that originate from predefined classes, the labeler randomly selects a subpool, with each sample equally likely to be labeled. However, an alternative approach is to allow a *different* random missing label model for each *class*, i.e. to condition the random missing label mechanism on the underlying predefined class for the sample, $\text{Prob}[\mathcal{L}|\mathcal{M}_{\text{g}} \in \mathcal{C}_{\text{pre}}, \text{C} = \text{c}]$, $\forall c$ [8]. In addition to explaining missing labels for the 'large database' scenario[9], this model also addresses the (quite 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 $\Lambda_2 = \{\{\alpha_k\}, \{\theta_k\}, \mathcal{C}_{\text{pre}}, \{\text{Prob}[\text{C}|\mathcal{M}_{\text{k}} \in \mathcal{C}_{\text{pre}}]\}\}, \{\text{Prob}[\mathcal{L}|\mathcal{M}_{\text{g}} \in \text{C}_{\text{pre}}, \text{C} = \text{c}]\}\}$. Note that in this case the random class label generator does not condition on $\mathcal{L} =$ "l". This will be explained, shortly.

*Hypothesis for Random Generation of the Data*

[8] One can instead imagine conditioning on the mixture component, i.e. forming $\text{Prob}[\mathcal{L}|\mathcal{M}_{\text{k}} \in \mathcal{C}_{\text{pre}}]$. However, this model may provide too much flexibility in explaining the presence of missing labels. In particular, we introduce two types of components with different random missing label mechanisms in order both to allow and to 'encourage' non-predefined components to capture purely unlabeled subsets, representing possible unknown classes. However, if predefined components are allowed to choose $\text{Prob}[\mathcal{L} = \text{"l"}|\mathcal{M}_{\text{k}} \in \mathcal{C}_{\text{pre}}]$ arbitrarily, i.e. to make this probability very small, these components may capture unlabeled data that would otherwise have been captured by nonpredefined components. Thus, the ability to discern purely unlabeled subsets, and hence potential new classes, might be compromised.

[9] This can be achieved by choosing $\text{Prob}[\mathcal{L}|\mathcal{M}_{\text{g}} \in \mathcal{C}_{\text{pre}}, \text{C} = \text{c}] = \text{Prob}[\mathcal{L}|\mathcal{M}_{\text{g}} \in \mathcal{C}_{\text{pre}}]$ $\forall$c.

With this alternative model, we hypothesize that each sample from $\mathcal{X}_m$ is generated independently, based on $\Lambda_2$, according to the following process:

i) Randomly select a mixture component $\mathcal{M}_j$ according to $\{\alpha_k\}$.

ii) Randomly select a feature vector $\underline{x}$ according to $f(\underline{x}|\theta_j)$.

iii) If $\mathcal{M}_j \in \mathcal{C}_{\text{pre}}$, then randomly select the class label $c$ according to Prob[C|$\mathcal{M}_{\text{j}} \in \mathcal{C}_{\text{pre}}$].

iv) If $\mathcal{M}_j \in \mathcal{C}_{\text{pre}}$, then randomly select the fact of label presence/absence $v \in \{$"l", "m"$\}$ according to $\{\text{Prob}[\mathcal{L} = v|\mathcal{M}_g \in \mathcal{C}_{\text{pre}}, C = c]\}$. If $v =$ "l", then the class label $c$ is *revealed and taken to be observed*; if $v =$ "m", then the class label is *withheld and taken to be missing.*

v) If $\mathcal{M}_j \in \bar{\mathcal{C}}_{\text{pre}}$, set $v =$ "m".

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

*Joint Data Likelihood Function*

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

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

## 4.3 Expectation-Maximization Learning Algorithm

### 4.3.1 Introduction

A standard approach for solving MLE problems (as discussed in previous chapters) is the Expectation-Maximization (EM) algorithm [19]. 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 computationally simple iteration. Specifically, we treat the (unknown) component of origin for each data sample as the missing data in the EM framework. Accordingly, we define the quantities

$$V_{\underline{x}k} = \begin{cases} 1 & \text{if } \underline{x} \in \mathcal{M}_k \\ 0 & \text{else} \end{cases}$$

where $\underline{x} \in \mathcal{M}_k$ denotes that sample $\underline{x}$ was generated by $\mathcal{M}_k$, and where the $\{V_{\underline{x}k}\}$ are constrained by $\sum_k V_{\underline{x}k} = 1$. It is then seen that the associated *complete* data log-likelihood, based on knowledge of the hidden data, can be written (e.g. for model I) in the form

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

Before further developing the associated EM algorithm, it is informative to give a brief aside justifying our model and learning approach. In particular, consider the

$\{v_k\}$ variables, specifying $\mathcal{C}_{\text{pre}}$, which were introduced in Section 2. We will treat these hard 0-1 variables as *parameters*, to be learned. However, there are two alternatives: i) we could allow components to be predefined/nonpredefined *in probability* and, thus, learn the parameters $\{\text{Prob}[\mathcal{M}_k \in \mathcal{C}_{\text{pre}}]\}$; ii) we could treat the $\{v_k\}$ as *missing* data and estimate their expected values within the EM framework. As it turns out, neither of these approaches is computationally feasible in practice. To illustrate the (common) difficulty, consider the stochastic data generation consistent with i). Essentially, i) suggests that one *first* randomly generates 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, described in sections 2.2 and 2.3. Now, consider the likelihood associated with this model. For each of the $2^M$ possible configurations, there is an associated configuration-specific likelihood, evaluated based on (4.1) or (4.2). The likelihood for the *model* is obtained by *averaging* over these $2^M$ configuration-dependent likelihoods, based on the joint configuration pmf $\{P[V_1 = v_1, V_2 = v_2, \ldots, V_M = v_M] = \prod_{m=1}^{M} \text{Prob}[\mathcal{M}_k \in \mathcal{C}_{\text{pre}}]^{v_k}(1 - \text{Prob}[\mathcal{M}_k \in \mathcal{C}_{\text{pre}}])^{(1-v_k)}\}$. The model's likelihood is thus *explicitly* based on a weighted sum of $2^M$ terms, with each term specified by (4.1) or (4.2). 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. If instead, as in ii), the $\{v_k\}$ are treated as *missing* variables within the EM framework, then careful scrutiny of (4.1) or (4.2) will lead to the same conclusion – the E-step will require an explicit average over all configuration-specific (complete data) log-likelihoods, with M-step complexity

then exponential in $M$. The computational difficulties with both these approaches are circumvented by treating the $\{v_k \in \{0,1\}\}$ as *parameters to be learned*, as seen next.

### 4.3.2 Formulation

Our learning strategy is to perform an iterative optimization with each iteration consisting of two steps: i) maximize $\log L_m$ over the component natures $\{v_k\}$ given the remaining parameters in $\Lambda$ held fixed; ii) using the EM algorithm, maximize over the remaining parameters in $\Lambda$, given the $\{v_k\}$ held fixed. Each step is nondecreasing in $\log L_m$. We next develop this optimization approach in detail for model I.

*Optimization over Component Natures*

Consider $\log L_m$ given in (4.1) or (4.2). Notice that the dependence on $\{v_k\}$ is somewhat complicated. There are several possible approaches to maximizing $\log L_m$ over these variables. One possibility, if $M$ is not too large, is simply 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 does not guarantee convergence to a global, or even a local optimum. However, it does guarantee ascent in $\log L_m$.

*EM Algorithm for the Remaining Parameters*

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

*E-step:*

In the Expectation step of the EM algorithm [19], 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 $E[V_{\underline{x}k}|\underline{x} \in \mathcal{X}_l; \Lambda^{(t)}]$ and $E[V_{\underline{x}k}|\underline{x} \in \mathcal{X}_u; \Lambda^{(t)}]$. Note that, since $V_{\underline{x}k} \in \{0,1\}$ with the constraint $\sum_k V_{\underline{x}k} = 1$, these expected quantities are simply probabilistic assignments of points to components, i.e. $E[V_{\underline{x}k}|\underline{x} \in \mathcal{X}_l; \Lambda^{(t)}] \equiv \text{Prob}[\mathcal{M}_k|\underline{x} \in \mathcal{X}_l; \Lambda^{(t)}]$. These probabilities, derived via Bayes rule, are given by

$$\text{Prob}[\mathcal{M}_k|\underline{x} \in \mathcal{X}_l; \Lambda_1^{(t)}] = \begin{cases} \dfrac{\alpha_k^{(t)} f(\underline{x}|\theta_k^{(t)}) \text{Prob}[C=c(\underline{x})|\mathcal{M}_k \in \mathcal{C}_{\text{pre}}, \mathcal{L}=\text{“l”}]^{(t)}}{\sum\limits_{n \in \mathcal{C}_{\text{pre}}} \alpha_n^{(t)} f(\underline{x}|\theta_n^{(t)}) \text{Prob}[C=c(\underline{x})|\mathcal{M}_n \in \mathcal{C}_{\text{pre}}, \mathcal{L}=\text{“l”}]^{(t)}} & k \in \mathcal{C}_{\text{pre}} \\ 0 & \text{else} \end{cases} \tag{4.4}$$

$$\text{Prob}[\mathcal{M}_k|\underline{x} \in \mathcal{X}_u; \Lambda_1^{(t)}] = \begin{cases} \dfrac{\alpha_k^{(t)} f(\underline{x}|\theta_k^{(t)}) \text{Prob}[\mathcal{L}=\text{“m”}|\mathcal{M}_g \in \mathcal{C}_{\text{pre}}]^{(t)}}{\sum\limits_{n \in \mathcal{C}_{\text{pre}}} \alpha_n^{(t)} f(\underline{x}|\theta_n^{(t)}) \text{Prob}[\mathcal{L}=\text{“m”}|\mathcal{M}_g \in \mathcal{C}_{\text{pre}}]^{(t)} + \sum\limits_{n \in \bar{\mathcal{C}}_{\text{pre}}} \alpha_n^{(t)} f(\underline{x}|\theta_n^{(t)})} & \\ & k \in \mathcal{C}_{\text{pre}} \\ \dfrac{\alpha_k^{(t)} f(\underline{x}|\theta_k^{(t)})}{\sum\limits_{n \in \mathcal{C}_{\text{pre}}} \alpha_n^{(t)} f(\underline{x}|\theta_n^{(t)}) \text{Prob}[\mathcal{L}=\text{“m”}|\mathcal{M}_g \in \mathcal{C}_{\text{pre}}]^{(t)} + \sum\limits_{n \in \bar{\mathcal{C}}_{\text{pre}}} \alpha_n^{(t)} f(\underline{x}|\theta_n^{(t)})} & \\ & \text{else} \end{cases} \tag{4.5}$$

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

$$
\begin{aligned}
E[\log L_c|\Lambda_1^{(t)}] \;=\; & \sum_{\underline{x}\in\mathcal{X}_l}\sum_{k\in\mathcal{C}_{\text{pre}}} \text{Prob}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_l;\Lambda_1^{(t)}]\cdot \\
& \log\Big(\alpha_k f(\underline{x}|\theta_k)\text{Prob}[\mathcal{L}=\text{“l”}|\mathcal{M}_g\in\mathcal{C}_{\text{pre}}]\text{Prob}[C=c(\underline{x})|\mathcal{M}_k\in\mathcal{C}_{\text{pre}},\mathcal{L}=\text{“l”}]\Big) \\
+ & \sum_{\underline{x}\in\mathcal{X}_u}\sum_{k\in\mathcal{C}_{\text{pre}}} \text{Prob}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_u;\Lambda_1^{(t)}]\log\Big(\alpha_k f(\underline{x}|\theta_k)\text{Prob}[\mathcal{L}=\text{“m”}|\mathcal{M}_g\in\mathcal{C}_{\text{pre}}]\Big) \\
+ & \sum_{\underline{x}\in\mathcal{X}_u}\sum_{k\in\bar{\mathcal{C}}_{\text{pre}}} \text{Prob}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_u;\Lambda_1^{(t)}]\log\left(\alpha_k f(\underline{x}|\theta_k)\right). \qquad (4.6)
\end{aligned}
$$

*M-step:*

For concreteness, suppose that $f(\cdot)$ is a joint Gaussian density function, with parameter set given by the mean vector and covariance matrix, i.e. $\theta_k = \{\underline{m}_k, \Sigma_k\}$ [10]. In the M-step for model I, $E[\log \mathcal{L}_c|\Lambda_1^{(t)}]$ is maximized over the $\Gamma$ 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 $\Gamma$ 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. Finally, the global Hessian matrix, defined over all of $\Gamma$, is negative definite when evaluated at this solution. Thus, the M-step globally maximizes $E[\log \mathcal{L}_c|\Lambda_1^{(t)}]$

[10] 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. The choice of multivariate Gaussians is merely for illustration.

with respect to $\Gamma$. This M-step is given by the (decoupled) parameter estimates:

$$\alpha_k^{(t+1)} = \frac{\sum_{\underline{x}\in\mathcal{X}_l} \text{Prob}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_l;\Lambda_1^{(t)}] + \sum_{\underline{x}\in\mathcal{X}_u} \text{Prob}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_u;\Lambda_1^{(t)}]}{N} \qquad \forall k \qquad (4.7)$$

$$\underline{m}_k^{(t+1)} = \frac{\sum_{\underline{x}\in\mathcal{X}_l} \underline{x}\text{Prob}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_l;\Lambda_1^{(t)}] + \sum_{\underline{x}\in\mathcal{X}_u} \underline{x}\text{Prob}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_u;\Lambda_1^{(t)}]}{\sum_{\underline{x}\in\mathcal{X}_l} \text{Prob}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_l;\Lambda_1^{(t)}] + \sum_{\underline{x}\in\mathcal{X}_u} \text{Prob}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_u;\Lambda_1^{(t)}]} \qquad \forall k \quad (4.8)$$

$$\Sigma_k{}^{(t+1)} = \left(\frac{1}{\sum_{\underline{x}\in\mathcal{X}_l} \text{Prob}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_l;\Lambda_1^{(t)}] + \sum_{\underline{x}\in\mathcal{X}_u} \text{Prob}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_u;\Lambda_1^{(t)}]}\right) \qquad (4.9)$$

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

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

$$\text{P}[C=c|\mathcal{M}_k\in\mathcal{C}_{\text{pre}},\mathcal{L}=\text{“l”}]^{(t+1)} = \frac{\sum_{\underline{x}\in\mathcal{X}_l:c(\underline{x})=c} \text{P}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_l;\Lambda_1^{(t)}]}{\sum_{\underline{x}\in\mathcal{X}_l} \text{P}[\mathcal{M}_k|\underline{x}\in\mathcal{X}_l;\Lambda_1^{(t)}]} \quad \forall c,k:\mathcal{M}_k\in\mathcal{C}_{\text{pre}}$$

$$\mathrm{P}[\mathcal{L} = \text{“l”} | \mathcal{M}_g \in \mathcal{C}_{\mathrm{pre}}]^{(t+1)} = \frac{\sum_{\underline{x} \in \mathcal{X}_l} \sum_{k \in \mathcal{C}_{\mathrm{pre}}} \mathrm{P}[\mathcal{M}_k | \underline{x} \in \mathcal{X}_l; \Lambda_1^{(t)}]}{\sum_{\underline{x} \in \mathcal{X}_l} \sum_{k \in \mathcal{C}_{\mathrm{pre}}} \mathrm{P}[\mathcal{M}_k | \underline{x} \in \mathcal{X}_l; \Lambda_1^{(t)}] + \sum_{\underline{x} \in \mathcal{X}_u} \sum_{k \in \mathcal{C}_{\mathrm{pre}}} \mathrm{P}[\mathcal{M}_k | \underline{x} \in \mathcal{X}_u; \Lambda_1^{(t)}]}. \tag{4.10}$$

*Pseudocode for Overall Iterative Optimization*

Our iterative optimization over the full parameter set $\Lambda_1$, which performs hillclimbing in the data likelihood, is summarized in pseudocode as shown below:

**Initialize** $v_k = 1 \ \forall k$ and initialize $\Gamma$ 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 $\Lambda_{(n)} = \{\Gamma_{(n)}, \{v_k\}_{(n+1)}\}$. Convergence is defined as a change in the log likelihood of less than $10^{-5}$. 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 'non-predefined'.

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 $\{\text{Prob}[C = c|\mathcal{M}_k \in \mathcal{C}_{\text{pre}}, \mathcal{L} = \text{“l”}]\}$ 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.

### 4.3.3 Learning for Model II

We will not present in full detail the learning approach for model II. However, the main points are as follows:

1. The learning strategy is the same as for model I, achieving hill climbing in (4.2), based on alternating $\{v_k\}$ and EM optimizations.

2. The E-step equations for model II are given by:

$$\mathrm{P}[\mathcal{M}_k|\underline{x} \in \mathcal{X}_l; \Lambda_2^{(t)}] = \begin{cases} \dfrac{\alpha_k^{(t)} f(\underline{x}|\theta_k^{(t)}) \mathrm{P}[C{=}c(\underline{x})|\mathcal{M}_k{\in}\mathcal{C}_{\mathrm{pre}}]^{(t)}}{\sum\limits_{n \in \mathcal{C}_{\mathrm{pre}}} \alpha_n^{(t)} f(\underline{x}|\theta_n^{(t)}) \mathrm{P}[C{=}c(\underline{x})|\mathcal{M}_n{\in}\mathcal{C}_{\mathrm{pre}}]^{(t)}} & \mathcal{M}_k \in \mathcal{C}_{\mathrm{pre}} \\ 0 & \text{else} \end{cases} \tag{4.11}$$

$$\mathrm{P}[\mathcal{M}_k|\underline{x} \in \mathcal{X}_u; \Lambda_2^{(t)}] = \begin{cases} \dfrac{\alpha_k^{(t)} f(\underline{x}|\theta_k^{(t)}) \sum\limits_c \mathrm{P}[C{=}c|\mathcal{M}_k{\in}\mathcal{C}_{\mathrm{pre}}]^{(t)} \mathrm{P}[\mathcal{L}{=}\text{“m”}|\mathcal{M}_g{\in}\mathcal{C}_{\mathrm{pre}}, C{=}c]^{(t)}}{\sum\limits_n \alpha_n^{(t)} f(\underline{x}|\theta_n^{(t)})(1{+}v_n \sum\limits_c \mathrm{P}[C{=}c|\mathcal{M}_n{\in}\mathcal{C}_{\mathrm{pre}}]^{(t)} \mathrm{P}[\mathcal{L}{=}\text{“m”}|\mathcal{M}_g{\in}\mathcal{C}_{\mathrm{pre}}, C{=}c]^{(t)})} & \mathcal{M}_k \in \mathcal{C}_{\mathrm{pre}} \\ \dfrac{\alpha_k^{(t)} f(\underline{x}|\theta_k^{(t)})}{\sum\limits_n \alpha_n^{(t)} f(\underline{x}|\theta_n^{(t)})(1{+}v_n \sum\limits_c \mathrm{P}[C{=}c|\mathcal{M}_n{\in}\mathcal{C}_{\mathrm{pre}}]^{(t)} \mathrm{P}[\mathcal{L}{=}\text{“m”}|\mathcal{M}_g{\in}\mathcal{C}_{\mathrm{pre}}, C{=}c]^{(t)})} & \text{else} \end{cases} \tag{4.12}$$

3. The M-step equations for $\{\alpha_k\}$, $\{\underline{m}_k\}$, and $\{\Sigma_k\}$ are the same as for model I, given in equations (4.7),(4.8),(4.9), but now based on the probabilities given in (4.11),(4.12).

4. However, the partial derivatives for $\{\mathrm{Prob}[C = c|\mathcal{M}_k \in \mathcal{C}_{\mathrm{pre}}]\}$ and $\{\mathrm{Prob}[\mathcal{L} = \text{“m”}|\mathcal{M}_g \in \mathcal{C}_{\mathrm{pre}}, C = c]\}$ are now coupled [11]. What this means is that 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

[11] For concision, we omit inclusion of these equations here.

soft-max functions, in order to ensure they remain probabilities throughout the local optimization. In particular, we let $\text{Prob}[C = c|\mathcal{M}_k \in \mathcal{C}_{\text{pre}}] = \frac{e^{\lambda_{c,k}}}{\sum_{c'} e^{\lambda_{c',k}}}$ 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 $\{\gamma_c\}$ [12]. The overall M-step for model II thus consists of i) closed form updates for $\{\alpha_k\}$, $\{\underline{m}_k\}$, and $\{\Sigma_k\}$, followed by ii) 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.

## 4.4 Statistical Inferences from the Models

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

*Classification to a Predefined Class*

[12] The gradient descent equations are straightforward to develop and are omitted for concision.

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

$$\text{Prob}[C = c|\underline{x}; \Lambda_1] = \frac{\sum\limits_{k \in \mathcal{C}_{\text{pre}}} \alpha_k f(\underline{x}|\theta_k) \text{Prob}[C = c|\mathcal{M}_k \in \mathcal{C}_{\text{pre}}, \mathcal{L} = \text{“}l\text{”}]}{\sum\limits_{k \in \mathcal{C}_{\text{pre}}} \alpha_k f(\underline{x}|\theta_k)}, \ \forall c \in \mathcal{P}_c. \tag{4.13}$$

Likewise, for model II, we have:

$$\text{Prob}[C = c|\underline{x}; \Lambda_2] = \frac{\sum\limits_{k \in \mathcal{C}_{\text{pre}}} \alpha_k f(\underline{x}|\theta_k) \text{Prob}[C = c|\mathcal{M}_k \in \mathcal{C}_{\text{pre}}]}{\sum\limits_{k \in \mathcal{C}_{\text{pre}}} \alpha_k f(\underline{x}|\theta_k)}, \ \forall c \in \mathcal{P}_c. \tag{4.14}$$

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

*Identifying Unknown Class Data*

For task ii), we need the *a posteriori* probability that a given unlabeled feature vector is generated by a non-predefined component. For both models I and II, this is simply

$$\text{Prob}[\mathcal{M}_g \in \bar{\mathcal{C}}_{\text{pre}}|\underline{x} \in \mathcal{X}_u] = 1 - \sum_{k \in \mathcal{C}_{\text{pre}}} \text{Prob}[\mathcal{M}_k|\underline{x} \in \mathcal{X}_u; \Lambda], \tag{4.15}$$

where $\text{Prob}[\mathcal{M}_k|\underline{x} \in \mathcal{X}_u; \Lambda]$ is given in (4.5) and (4.12), respectively. In addition to identifying unknown class data, (4.15) also forms the basis for sample rejection when the goal is classification to one of the predefined classes.

*Transductive Inference*

In this case, the test set/use samples $\mathcal{X}_{\text{te}} = \{\underline{x}_i\}$ that augment the existing training set $\mathcal{X}_{\text{tr}}$ are *purely* unlabeled. The learning approach thus taken should depend upon whether $\mathcal{X}_{\text{tr}}$ is a) purely labeled or b) mixed labeled/unlabeled. In particular, our models assume the data set is mixed labeled/unlabeled and treat label presence/absence as observed data, explained by an underlying random model. However, in $\mathcal{X}_{\text{te}}$, labels are *deterministically* missing. Thus, unless it is necessary to do so, we should not treat them as missing at random.

*Case a): $\mathcal{X}_{\text{tr}}$ Purely Labeled:*

In this important 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 $\mathcal{X}_{\text{te}}$ (those that are hypothesized to originate from predefined classes) are *treated* as missing at random, even though they are actually deterministically missing[13] . 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 2.1 and 2.2). Consider, for example, learning model I. The learning will produce the probability $\text{Prob}[\mathcal{L} = \text{“}m\text{”} | \mathcal{M}_g \in \mathcal{C}_{\text{pre}}] \in [0, \frac{|\mathcal{X}_{\text{te}}|}{|\mathcal{X}_{\text{te}}|+|\mathcal{X}_{\text{tr}}|}]$. At the two extremes, the probability will be zero if all of $\mathcal{X}_{\text{te}}$ is owned by non-predefined 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, (4.13),(4.14), and (4.15) are used to address inference tasks involving $\mathcal{X}_{\text{te}}$. Note that, as aforementioned, this approach has *very general* applicability, since it covers a standard case, where the training set is purely labeled, with the unlabeled data coming solely

[13] More precisely, class label presence/absence is treated as a random variable for each sample in the combined set $\mathcal{X}_m = \{\mathcal{X}_{\text{tr}}, \mathcal{X}_{\text{te}}\}$.

from the (always unlabeled) test set. This approach will be experimentally evaluated in section 6.

*Case b): $\mathcal{X}_{\text{tr}}$ Mixed Labeled/Unlabeled:*

In this case, it is not necessary to treat label presence for $\mathcal{X}_{\text{te}}$ as observed data, explained by an underlying (random) model. Thus, a (small) modification is required, both in the learning and in the inference rules. For MLE learning, the data set now consists of $\mathcal{X} = \{\mathcal{X}_m, \mathcal{X}_{te}\}$, with $\mathcal{X}_m = \mathcal{X}_{\text{tr}}$. Accordingly, the new joint data likelihood takes the form

$$\log L_{\text{trans}} = \log L_m + \sum_{\underline{x} \in \mathcal{X}_{\text{te}}} \log \sum_k \alpha_k f(\underline{x}|\theta_k), \tag{4.16}$$

where $\log L_m$ is given in (4.1) and (4.2) for models I and II, respectively. (4.16) reflects the fact that label presence is no longer treated as observed data for $\mathcal{X}_{\text{te}}$. $\log L_{\text{trans}}$ can be maximized using an algorithm identical in structure to the pseudocode given earlier, only with different E and M steps, consistent with (4.16). These steps are omitted for concision. Given the learned model, we predict whether or not a sample $\underline{x} \in \mathcal{X}_{\text{te}}$ originates from a predefined class based on the *a posteriori* probabilities

$$\text{Prob}[\mathcal{M}_g \in \text{C}_{\text{pre}} | \underline{x} \in \mathcal{X}_{\text{te}}] = \frac{\sum\limits_{k \in \mathcal{C}_{\text{pre}}} \alpha_k f(\underline{x}|\theta_k)}{\sum\limits_k \alpha_k f(\underline{x}|\theta_k)}. \tag{4.17}$$

Sample rejection, consistent with the hypothesis that the sample comes from an unknown class, is based on thresholding (4.17) (a MAP rule). If the sample is not rejected, then it is classified using a MAP rule based on (4.13) or (4.14).

## 4.5 Model Selection

While the EM learning in section 3 assumed that the number of mixture components $M$ is fixed and known, in practice this size must be estimated. Model order selection is a difficult and pervasive problem. Several different statistical techniques have been proposed to estimate the number of components in a mixture, e.g. [46],[6],[24]. 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. [37]. In this work, our aim is not to propose an optimal technique for model selection in mixtures. 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, the EM algorithm is run, based on a size $M$ chosen either to *overestimate* the true number of components[14] 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 re-running EM 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

[14] 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 $\mathcal{X}_m$.

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. This model is used for all of the pattern recognition tasks identified earlier, including class discovery. 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 [54]. This criterion function is given by:

$$\mathcal{L}_{\text{MDL}} = \frac{1}{2}(\sum_{m=1}^{M} P_m) \log N - \log L_m, \tag{4.18}$$

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).

## 4.6 Experimental Results

### 4.6.1 Description of Methods

For purpose of comparison, we implemented three other methods, in addition to the two models we have developed. One of the methods to be compared we dub 'Supervised Clustering'. This approach is loosely based on the work in [38]. In this method, 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 mass. For classification, the components are then used to form

a nearest-prototype classifier, e.g. [42]. For predefined/non-predefined discrimination, an unlabeled sample is declared to be from an unknown class if it is deemed to be 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: i) a point that originates from one of the components classified as being an outlier with respect to every component; ii) a point from an unknown class identified as originating from a component. The threshold is chosen to make these two error types equally likely.

The second method used for comparison is based on [47] and is named the MU method. 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 [47] for predefined/non-predefined discrimination, each mixture component is deemed "predefined" if it 'owns' any labeled samples; else, the component is deemed "non-predefined"[15]. Predefined/non-predefined discrimination is thus performed similar to our method, based on (4.15), but with $P[\mathcal{M}_k|\underline{x} \in \mathcal{X}_u]$ as given in [47], rather than based on (4.5).

*Model Size Selection for above models*

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. For the

[15] A mixture component is said to "own" a sample if the membership probability of the sample with the component is greater than 50%.

methods of comparison introduced above, the MDL cost is

$$MDL = \frac{MP}{2} \log N - \log L, \tag{4.19}$$

with $P$ the number of parameters per component.

*Unlabeled Class MU (UCMU) Method*

The third method is also based on [47] like the last one. This model treats all of the unlabeled data as belonging to a new class, the unlabeled class denoted here by U. If the original problem had $N_c$ classes, adding the U class will make it an $N_c + 1$ class problem. This means all of the data is now labeled. We refer to this method as Unlabeled Class MU (UCMU). The log likelihood equation in this case is

$$\log L = \sum_{x_i \in \chi_m} \log[\sum_{l=1}^{L} \alpha_l \beta_{k|l} f(x_i|\alpha_l)]. \tag{4.20}$$

The difference between the above equation and the original MU log likelihood, eq. (1.2), is that in this model there is no unlabeled data term. The update equations for $\mu, \sigma, \alpha$ and $\beta$ parameters are the same as those given in Section 1.3 for the MU model, except that in each case the term relating to the unlabeled data is missing. Once the model is learned, the value of $\beta_{\text{class=U}|l}$ for each cluster l is examined and if it is very close to 1 ($> 0.98$), this particular cluster is identified as a 'non-predefined' cluster; other clusters are identified as 'predefined' clusters. So all the clusters that are predominately populated by the unlabeled class are 'non-predefined' clusters.

The equation to calculate $\text{Prob}[\mathcal{M}_g \in \bar{\mathcal{C}}_{\text{pre}} | \underline{x} \in \mathcal{X}_u]$ is

$$\text{Prob}[\mathcal{M}_g \in \bar{\mathcal{C}}_{\text{pre}} | \underline{x} \in \mathcal{X}_u] = 1 - \sum_{k \in \mathcal{C}_{\text{pre}}} \text{Prob}[\mathcal{M}_k | \underline{x} \in \mathcal{X}_u; \Lambda], \tag{4.21}$$

where

$$\text{Prob}[\mathcal{M}_k | \underline{x} \in \mathcal{X}_u; \Lambda_1] = \frac{\alpha_k f(\underline{x}|\theta_k)\text{Prob}[C{=}U|\mathcal{M}_k]}{\sum_n \alpha_n f(\underline{x}|\theta_n)\text{Prob}[C{=}U|\mathcal{M}_n]} \tag{4.22}$$

With this third UCMU model, a direction application of the MDL criterion used above gave very poor results and we have not found a model selection procedure that works. For this reason, we have not performed model size selection on this model. Instead we just present result for fixed size models.

For our methods, Model 1 and Model 2, iterative cycling was used to choose the $\{v_k\}$ for all the data sets that we considered. Since the EM algorithm is used by all the methods, a common EM stopping condition was used – termination when the change in likelihood from one iteration to the next fell below $10^{-5}$. For our methods, the (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.

### 4.6.2 Results

**Illustrative Example:** In Figure 4.2 there are 50 points, representing 5 ground truth components. Four components represent known classes, labeled "1" , "2", and "3".

Class 1 has one component centered at (1,2), class 2 has two components centered at (0,5) and (1,8) and class 3 has one component at (5,6). Each of these components has 5 labeled points and 5 unlabeled points, with the unlabeled points from class $i$ labeled $Ui$. The other component, representing unknown class data, has a mean at (4,2). These data points are labeled by $U4$. The search described previously was conducted to determine the best model I solution. Starting from $M = M_{\max} = 9$, the minimum (albeit an indistinct one) occurred at five components (see Figure 4.3). In this solution, also shown in Fig. 1a, there are four predefined components, with centers denoted by "x"s, and one nonpredefined component, with center shown by an "o". The oval around each component mean indicates an equiprobable contour at one standard deviation from the mean. When the unlabeled points were classified as predefined/nonpredefined, all were correctly classified with the exception of the three U4 points indicated by triangles. By comparison, the solution obtained using [47] did not find a non-predefined component. Thus, for this method all the $U4$ points were misclassified.

**Experimental Comparisons:** We used both synthetic and real-world data sets for evaluation. First, we consider tests using the synthetic data. We randomly generated thirty 2D sets, each with 560 samples. The data was randomly drawn from seven isotropic, equal-mass Gaussian components, each with a random mean and variance one in each dimension. Three of the components represent predefined classes and four represent unknown class data. Trials were done with 25%, 50% and 75 % of the data from the predefined classes labeled, and all the data (320 points) from the non-predefined components unlabeled. In the initial experiments we assumed the random missing label mechanism to be independent of the class label. For all the methods, we chose $M_{\max} = 9$.

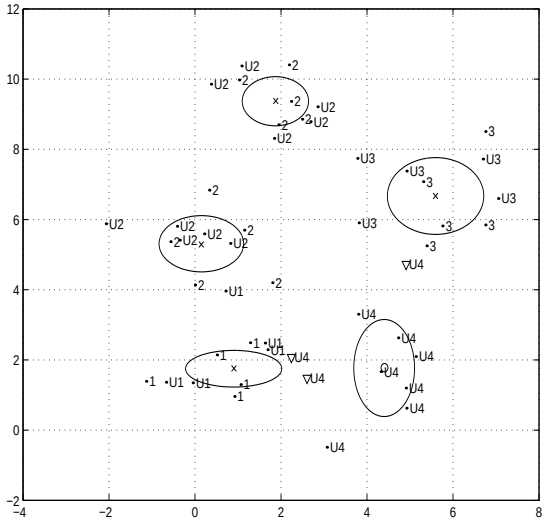

Fig. 4.2. Illustrative Example

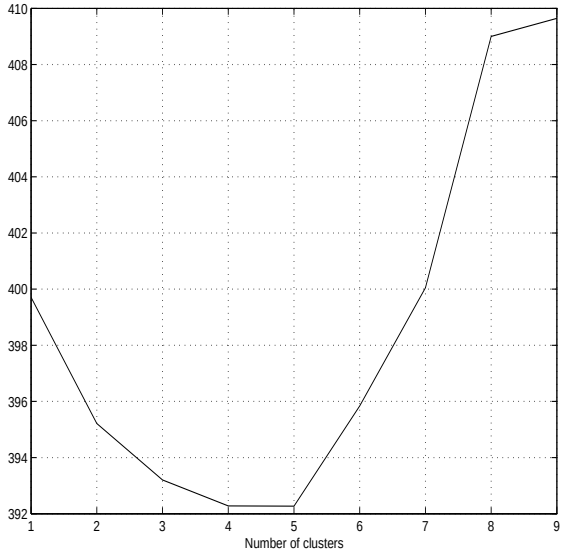


Fig. 4.3. MDL score for model size selection

The results for predefined/non-predefined discrimination are shown in Table 4.1, with classification results (over the predefined classes) shown in Table 4.2 . In both cases, the performance was measured over the unlabeled data subset (effectively, this amounts to a transductive inference setting). For the classification experiments, a two-step process was invoked, with predefined/nonpredefined discrimination first applied, and then, for those samples deemed predefined, classification to one of the known classes. If either step is incorrect, an error is counted. All the results represent averages over the thirty data sets.

| Method | Percent missing | | |
|---|---|---|---|
| | 25% | 50% | 75% |
| New EM method Model I | .049 | .044 | .033 |
| New EM method Model II | .061 | .057 | .047 |
| Supervised clustering method | .16 | .148 | .144 |
| Method from [47] | .32 | .440 | .543 |

Table 4.1. Fraction of incorrect predefined/unknown class decisions for synthetic data, evaluated for 25, 50, and 75 % of the known class data labeled.

In all cases the new methods fared the best, with the Supervised Clustering method next in performance. In many of the experiments, we found that model selection for our models found the true size ($M = 7$). The poor performance of the method based on [47] 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 are likely to

| Method | Percent missing | | |
|---|---|---|---|
| | 25% | 50% | 75% |
| New EM method Model I | .063 | .052 | .037 |
| New EM method Model II | .075 | .066 | .052 |
| Supervised clustering method | .17 | .155 | .147 |
| Method from [47] | .343 | .451 | .549 |

Table 4.2. Fraction of incorrect classifications on the synthetic data, evaluated for 25, 50, and 75 % of the known class data labeled.

end up owning some labeled data. Thus, there may not be any non-predefined components, in which case all the data will be identified as coming from predefined classes. The likelihood of this event may increase with the percentage of labeled data. This might explain the (contrary) behavior of the method from [47] as the percentage of labeled data is increased.

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 6 as known classes and the last 5 as unknown classes. Again for this data set we tested with 25%, 50% and 75% of the data from the predefined classes labeled. In this case, we chose $M_{\max} = 40$. Predefined/non-predefined discrimination and classification results for this data set are shown in Tables 4.3 and 4.4. In this case, Supervised Clustering fared the poorest. The method based on [47] performed (relatively) better on this data set, due to the larger model size, which allowed this method to find non-predefined components.

| Method | Percent missing | | |
|---|---|---|---|
| | 25% | 50% | 75% |
| New EM method Model I | .224 | .155 | .158 |
| New EM method Model II | .364 | .205 | .117 |
| Supervised clustering method | .467 | .451 | .408 |
| Method from [22] | .329 | .393 | .330 |

Table 4.3. Fraction of incorrect predefined/unknown class decisions for the *vowel* data, evaluated for 25, 50, and 75 % of the known class data labeled.

*Performance of Model II*

In Tables 4.1 - 4.4 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*, and therefore has more parameters to learn than Model I. This makes it easier for Model II to exhibit overlearning than Model I. Since the missing label mechanism was independent of the class for these experiments, Model II will have more parameters than it needs to model this data. To demonstrate the potential advantage of model II, we thus created new data sets with the missing label mechanism now class-dependent. For the synthetic data, we chose 25% labeled for the first class, 50 % labeled for the second class, and 75 % labeled for the third class. Additionally there were 4 components representing unknown class data[16]. For the *vowel* data, we chose 25% labeled for the first two classes, 50 % labeled for the second two classes, and 75 % labeled for the third two classes. The results in Tables 4.5 and 4.6 demonstrate an overall advantage of model II for these data sets, as expected.

[16]The (thirty) synthetic data sets used in these experiments were the same as those used in the previous experiments. The only difference was the random missing label mechanism.

| Method | Percent missing | | |
|---|---|---|---|
| | 25% | 50% | 75% |
| New EM method Model I | .369 | .327 | .227 |
| New EM method Model II | .469 | .358 | .188 |
| Supervised clustering method | .611 | .586 | .496 |
| Method from [22] | .554 | .551 | .421 |

Table 4.4. Fraction of incorrect classifications on the *vowel* data, evaluated for 25, 50, and 75 % of the known class data labeled.

*Performance of UCMU method*

Because we could not find a good way of doing model selection for this model, we only tested this model on one model size. We compared this method only with Model I, and report the results for 100 trials. Each trial consists of three predefined clusters and four non-predefined clusters with random centers, each consisting of 80 data points, as described in section 4.6.2. For each trial, both models were initialized identically. Referring to tables 4.7 and 4.8, for predefined/nonpredefined classification, Model I averaged 24 % better than the UCMU method and 14 % better for robust classification. On the vowel data in tables 4.9 and 4.10, Model I was 59 % better for predefined/nonpredefined classification and 36 % better for robust classification. During testing of this UCMU model, we noticed two problems that account for the difference in performance with Model I. The first problem was that the UCMU model utilized a threshold to determine whether or not a cluster was predefined. If a predefined cluster is located next to a non-predefined cluster some of the labeled points of the predefined cluster can easily be

| Method | Data Set | |
|---|---|---|
| | Synthetic | Vowel |
| New EM method Model I | .142 | .193 |
| New EM method Model II | .051 | .181 |
| Supervised clustering method | .153 | .416 |
| Method from [22] | .413 | .481 |

Table 4.5. Fraction of incorrect predefined/unknown decisions for the synthetic and *vowel* data with a class-dependent missing mechanism.

claimed by the non-predefined cluster. If only 1 or 2 labeled points are owned by a cluster that is otherwise wholly unlabeled, the cluster will be deemed a predefined cluster with accompanying poor classification for that trial. The second problem is one that we also saw with Model II, i.e. each cluster has its own parameter for the percentage of missing data in the cluster. Thus two adjacent clusters can be labeled as one big cluster with a small amount of labeled data. When this happens, the probability of a data point being unlabeled, $\mathrm{P}[\mathcal{L} = \text{“m”} | \mathcal{M}_g \in \mathcal{C}_{\mathrm{pre}}, C = c]$, increases markedly. This is unlikely to happen with Model I with a global parameter for $\mathrm{P}[\mathcal{L} = \text{“m”} | \mathcal{M}_g \in \mathcal{C}_{\mathrm{pre}}]$ that applies to all of the clusters because the other classes will continue to have the same lower value for this parameter.

*Global optimization vs. local optimization*

In order to test the effectiveness of our method for cycling over component natures, we compared with global optimization on the *vowel* data for the case of 50% of the known class data missing labels. We learned model I with 19 mixture components, using both global optimization and iterative cycling. As seen in Table 4.11, the results were

| Method | Data Set | |
|---|---|---|
| | Synthetic | Vowel |
| New EM method Model I | .142 | .343 |
| New EM method Model II | .059 | .344 |
| Supervised clustering method | .159 | .498 |
| Method from [22] | .425 | .698 |

Table 4.6. Fraction of incorrect classifications on the synthetic and *vowel* data with a class-dependent missing mechanism.

identical. Moreover, for synthetic data sets with 7 components, we have found that global optimization and iterative cycling often yield precisely the same results.

*Performance With No Unknown Class Data*

We also investigated whether our model would declare the existence of unknown class data/components in situations where all the data originates from predefined classes. This was tested on 30 synthetic data sets randomly generated as before, with 7 components and with the fraction of labeled samples taken to be 30 % in one experiment, 25 % in another, and 20 % in a third. In the case of 30 %, no non-predefined components were found, for any of the 30 data sets. For 25 % labeled, a single non-predefined component was found in one of the thirty data sets. For 20 % , 5 non-predefined components were found over the thirty data sets. These results are suggestive that our method can be effectively used even when there are no unknown classes in the data. In fact, when our method fails to find non-predefined components, the model precisely, aptly *specializes* to the mixture of experts model from [47].

| Method | Percent missing | | | |
|---|---|---|---|---|
| | 5% | 25% | 50% | 75% |
| New EM method Model I | .146 | .120 | .100 | .090 |
| UCMU | .157 | .138 | .115 | .129 |

Table 4.7. Fraction of incorrect predefined/unknown decisions for the synthetic data.

## 4.7 New Class Discovery on *Reuters*

As an initial investigation of the new class discovery problem, 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 1200 articles. The first 12 topics were treated as predefined classes, with the remaining topics unknown classes. Specifically, the known topics were: *acq*, *alum*, *bop*, *cocoa*, *coffee*, *ipi*, *crude*, *earn*, *gnp*, *gold*, *grain* and *interest*. The unknown topics 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 Figures 4.4-4.7. We labeled 50% of the articles from known topics, with all the unknown topic articles unlabeled.

Instead of a Gaussian model, the Bernoulli form of a naive Bayes model was used [45]. The word list for this model 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 for decreasing $M$). This was

| Method | Percent missing | | | |
|---|---|---|---|---|
| | 5% | 25% | 50% | 75% |
| New EM method Model I | .164 | .134 | .108 | .101 |
| UCMU | .173 | .15 | .124 | .134 |

Table 4.8. Fraction of incorrect classifications on the synthetic data.

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 1200 articles). MLE learning via EM was then performed. This learning took about 30 minutes on a Sun Ultrasparc 10. Once the model was learned, predefined/unknown class discrimination performance was measured on the unlabeled data subset, as done previously. The error rate was 27.9 %.

As a second performance measure, we counted the number of unknown topics 'discovered' by the model. This was done as follows. For each non-predefined component we determined the probability mass in each of the 24 topics. This was measured by summing up the probabilities (4.5) associated with all articles from the same topic. We then *labeled* each component by the topic that possessed the largest probability mass in that component. Finally, we counted up the number of distinct unknown topics 'covered' by the labels of these non-predefined 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 4 of the components in Fig. 4.4-4.7. The x-axis

| Method | Percent missing | | | |
|---|---|---|---|---|
| | 5% | 25% | 50% | 75% |
| New EM method Model I | .232 | .174 | .134 | .17 |
| UCMU | .258 | .241 | .287 | .314 |

Table 4.9. Fraction of incorrect predefined/unknown decisions for the *vowel* data.

represents the 24 topics. Each figure shows the probability mass of each of the topics for a given component. It is clear that each of these four components has found a primary (unknown) topic. This primary topic is indicated in the figure caption.

## 4.8 Future Work

In future, we will aim to address other applications that are well-matched to our learning and inference scenarios. Some of these include:

i) *Knowledge Discovery from an Internet/Database Search*, e.g. [44]: 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 associated with the query, while other data may either be spurious *or* may actually correspond to *a priori unknown* categories relevant to the query. While none of the queried data 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 sets may reasonably be treated as being of mixed labeled/unlabeled

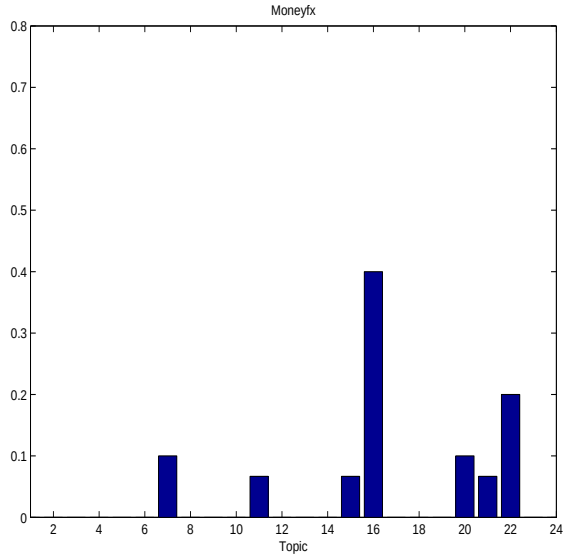


Fig. 4.4. Topic representation in *Moneyfx* cluster.

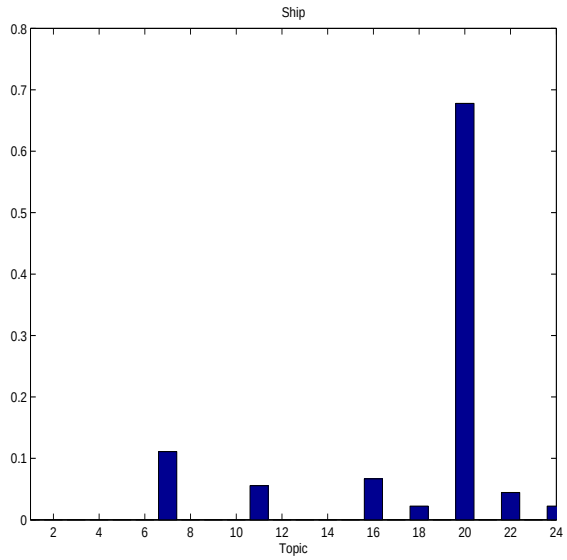


Fig. 4.5. Topic representation in *Ship* cluster.

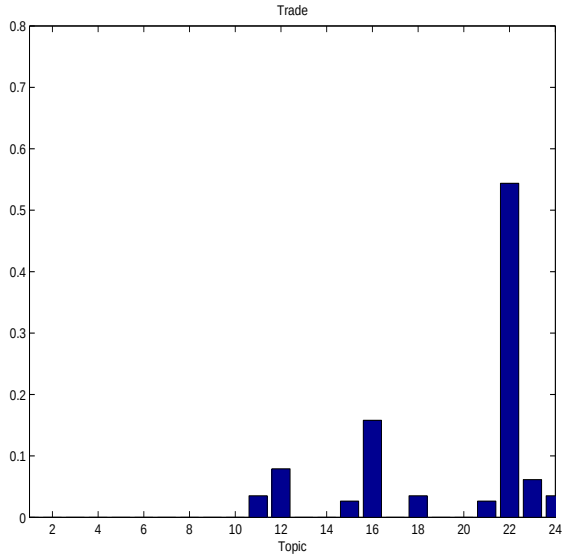


Fig. 4.6. Topic representation in *Trade* cluster.

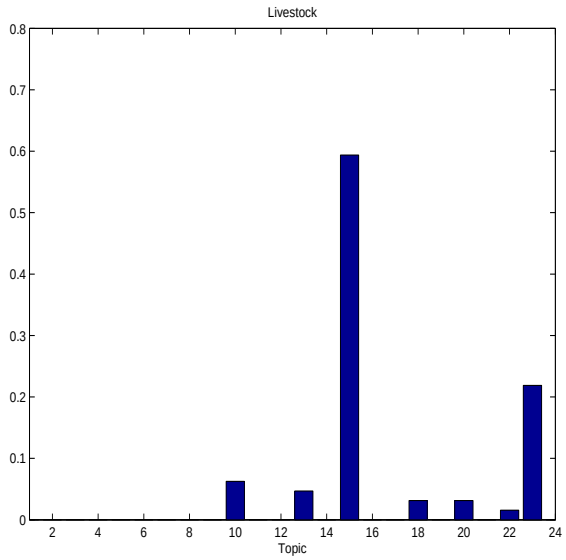


Fig. 4.7. Topic representation in *Livestock* cluster.

| Method | Percent missing | | | |
|---|---|---|---|---|
| | 5% | 25% | 50% | 75% |
| New EM method Model I | .453 | .375 | .285 | .260 |
| UCMU | .454 | .444 | .432 | .400 |

Table 4.10. Fraction of incorrect classifications on the *vowel* data.

character. Moreover, robust classifier design, sample rejection, and new class discovery are all relevant tasks for this domain.

ii) *Scientific Applications*: The main goal here is scientific classification/taxonomy, where 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.

iii) *Object Classification in Video Databases*: Image and video databases contain numerous objects, some of which fit predefined categories, but many of which defy existing categories. If a purely labeled object training set is combined with a purely unlabeled video database, our learning approach could be used to learn a model for both existing and new classes. We can then infer whether or not an object fits existing categories.

Finally, our learning approach assumes that data from unknown classes is purely unlabeled. Alternatively, we might consider a scenario more like [58], where existing class definitions may be inadequate and where data may have been mislabeled. Our predefined/non-predefined component dichotomy could be useful in this context as well,

| Classification Task | Type of Optimization | |
|---|---|---|
| | Global | Local |
| Predefined/Nonpredefined | .044 | .044 |
| Classification | .052 | .052 |

Table 4.11. Fraction of incorrect decisions for global/local optimization of component natures.

with 'predefined' components owning points that possess valid labels, and with 'non-predefined' components owning mislabeled points. Unlabeled data may also be present in this scenario.

## 4.9 Conclusion

The problem outlined here, learning a classifier from a mixed labeled and unlabeled data set, where some of the unlabeled data may derive from an unknown class is a new problem. Therefore all comparisons are with techniques we have devised. UCMU method seems to be the best competitor we have found, although at present there is no good way to determine the model size. When the new models were tested on the synthetic data set that included data from unknown classes, they routinely found the missing classes accurately (an example of which is given in Figure 4.2). This is verified by visual inspection of the resulting models learned for the 2-dimensional data sets. For the 10-dimensional vowel data set where visualization is impossible, with 5 of the 11 classes taken as missing, we got classification results commensurate with other classifiers operating on the original 11-class problem. While these results may not be the best possible

for this task, they do indicate that these two models are effective at creating classifiers in the presence of unlabeled data and unknown classes. We also tested the class discovery aspect of this model on the Reuters data set. With 12 of the topics unknown, we were able to learn a model where 11 of these unknown topics were the majority topic in a non-predefined cluster. These 11 non-predefined clusters have 'found' these 11 unknown topics. We believe this is indicative of successful class discovery.

# Chapter 5

# Conclusion

Data mining is a new field made possible by faster computers, very high capacity storage and the collection of large databases. In this work we have developed statistical models for data mining tasks. Applied to the Collaborative filtering problem, our models have achieved very good results on two standard CF databases, MSWEB and EachMovie. The results are better than any other published results that we know of, including results in [51], [13] and [9]. Additionally, statistical models have an advantage in the amount of time needed to create a prediction. While the time to do a prediction with correlation, currently the most popular method for CF, increases linearly with the size of the training set, the time it takes to answer a query with statistical models increases very little with increasing training set size. For the latent clustering model we created a way of learning a model for the whole space in a tractable amount of time. Our technique of learning a small part of the model and extending it to the whole space specified in Section 3.2 is new and will work with databases much larger than EachMovie. We also devised ways of dealing with missing data in Collaborative Filtering tasks. In the case of the Maximum Entropy Model, this increased the accuracy of the classifier by about 7.8 %.

We also considered a new problem, that of creating a model of a large database with unlabeled data, where some of the unlabeled data may originate from unknown classes. We demonstrated two models that can do inference and discover unknown

classes. These new models were compared to other possible methods of achieving these aims and found to provide more accurate inference. It is not yet clear whether any other method can do class discovery.

# References

[1] B. Arwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *Proc. of the 2nd ACM Conf. on Electronic Commerce*, pages 158–167, 2000.

[2] S. Baluja. Using labeled and unlabeled data for probabilistic modeling of face orientation. *Intl. Journal of Pattern Recognition and Artificial Intelligence*, 14:1097–1107, 2000.

[3] A. Ben-Dor, N. Friedman, and Z. Yakhini. Class discovery in gene expression data. In *Proc. of the Fifth Annual Int. Conf. on Computational Molecular Biology:RECOMB 2001*, pages 31–38, 2001.

[4] A. Berger, S. Della Pietra, and V. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22:39–68, 1996.

[5] Adam Berger. The Improved Iterative Scaling Algorithm: A Gentle Introduction. This is available online at http://www-2.cs.cmu.edu/ aberger/, 1997.

[6] C. Biernacki, G. Celeux, and G. Govaert. Assessing a mixture model for clustering with the integrated classification likelihood. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 22:719–725, 2000.

[7] Daniel Billsus and Michael J. Pazzani. Learning collaborative information filters. In *Proc. of the International Conference on Machine Learning*, pages 46–54. Morgan Kaufman Publishers, 1998.

[8] A. Blum and T. Mitchell. Combined labeled and unlabeled data with co-training. In *Proc. of the 11th Annual Conf. on Computational Learning Theory*, pages 92–100, 1998.

[9] J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. of the Fourteenth Conf. on Uncertainty in Art. Intell.*, pages 43–52, 1998.

[10] V. Castelli and T. M. Cover. On the exponential value of labeled samples. *Pattern Recognition Letters*, 16:105–111, 1995.

[11] I. Chang and M. Loew. Pattern recognition with new class discovery. In *Proc. of CVPR 1991*, pages 438–443, 1991.

[12] P. Cheeseman. A method of computing generalized Bayesian probability values for expert systems. In *Proc. of the Eighth Intl. Joint Conf. on AI*, volume 1, pages 198–202, 1983.

[13] Y. Chen and E. George. A Bayesian Model for Collaborative Filtering. In *Proc. of the Seventh Int'l. Workshop on Artificial Intelligence and Statistics*, pages 187–192, 1999.

[14] F.G. Cozman and I. Cohen. Unlabeled data can degrade classification performance of generative classifiers. Hewlett Packard Technical Report, HPL-2001-234.

[15] J.N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43:1470–1480, 1972.

[16] R. Dave. Characterization and detection of noise in clustering. *Pattern Recognition Letters*, 12:657–664, 1997.

[17] R. Dave and R Krishnapuram. Robust clustering methods:a unified view. *IEEE Trans. on Fuzzy Systems*, 5:2:270–293, 1997.

[18] V. de Sa. Learning classification with unlabeled data. In *Advances in Neural Information Processing Systems, 6*, pages 112–119, 1994.

[19] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Roy. Stat. Soc., Ser. B*, 39:1–38, 1977.

[20] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* Wiley-Interscience, New York, NY, 1974.

[21] R. O. Duda, P.E. Hart, and D. Stork. *Pattern Recognition.* Wiley-Interscience, 2001.

[22] S. Dumais, J. Platt, and D. Heckerman. Inductive learning algorithms and representations for text categorization. In *Proc. of the Seventh Int. Conf. on Inf. and Knowledge Management*, pages 148–155, 1998.

[23] D. Fisher et al. Swami: A framework for collaborative filtering. Class project report at UC Berkeley, website address http://guir.cs.berkeley.edu/projects/swami/.

[24] M. A. T. Figueiredo and A. K. Jain. Unsupervised learning of finite mixture models. *IEEE Trans. on Pattern Anal. and Machine Intell.*, pages 381–396, 2002.

[25] D. Fisher, K. Hildrum, J. Hong, M. Newman, M. Thomas, and R. Vuduc. Swami: a framework for collaborative filtering algorithm development and evaluation. In *Proc. of the 23rd annual Int. ACM SIGIR Conf. on Research and Development in Info. Retrieval*, pages 366–368. ACM Press, 2000.

[26] W. B. Gevarter. Automatic probabilistic knowledge acquisition from data. Technical Report NASA technical memorandum 88224, NASA, 1986.

[27] D. Goldberg, D. Nichols, B.M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35:12:61–70, 1992.

[28] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A Constant Time Collaborative Filtering Algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[29] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* Springer Series in Statistics, 2001.

[30] E. Herskovits and G. Cooper. Kutato: an entropy-driven system for construction of probabilistic expert systems from databases. In *Uncertainty in AI*, volume 6, pages 117–125, 1991.

[31] T. Hofmann. Probabilistic latent semantic indexing. In *Proc. of the 22nd Annual Int. ACM Conf. on Research and Development in Info. Retrieval*, pages 50–57, 1999.

[32] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *Proc. of the IJCAI '99*, pages 688–693, 1999.

[33] P.J. Huber. *Robust Statistics*. Wiley, New York, 1981.

[34] G. Hughes. On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, 14:1:55–63, 1968.

[35] M. Inoue and N. Ueda. HMMs for both labeled and unlabeled time series data. In *Proc. of the 2001 IEEE Workshop on Neural Networks for Signal Processing*, pages 93–102, 2001.

[36] A. Borchers J. Herlocker, J. Konstan and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd Annual International ACM SIGIR conference*, pages 230–237. ACM Press, 1999.

[37] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, NJ, 1988.

[38] B. Jeong and D. Landgrebe. Partially supervised classification using weighted unsupervised clustering. *IEEE Trans. on Geoscience and Remote Sensing*, 37:1073–1079, 1999.

[39] T. Joachims. *Advances in Kernel Methods-Support Vector Learning*. MIT Press, 1999.

[40] S. Thrun K. Nigam, A. McCallum and T. Mitchell. Learning to classify text from labeled and unlabeled documents. In *AAAI-98 Fifteenth National Conference on Artificial Intelligence*, pages 792–799. AAAI Press / The MIT Press, 1998.

[41] J. Kittler, M. Hatef, R. Duin, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:226–239, 1998.

[42] T. Kohonen. *Self organization and associative memory.* Springer-Verlag, Berlin, 1984.

[43] H. H. Ku and S. Kullback. Approximating discrete probability distributions. *IEEE Trans. on Inform. Theory*, 15(4):444–447, 1969.

[44] J. Larsen, L. K. Hansen, T. Christiansen, and T. Kolenda. Webmining:learning from the World Wide Web. *Computational Statistics and Data Analysis*, 38:517–532, 2002.

[45] A. McCallum and K. Nigam. A Comparison of Event Models for Naive Bayes Text Classification. In *In Proc. of the AAAI-98 Workshop on Learning for Text Categorization, AAAI Press*, pages 41–48, 1998.

[46] G. McLachlan and D. Peel. *Finite Mixture Models.* John Wiley and Sons, 2000.

[47] D. Miller and H. S. Uyar. A mixture of experts classifier with learning based on both labelled and unlabelled data. In *Advances in Neural Information Processing Systems 9*, pages 571–577, 1997.

[48] D. Miller and H. S. Uyar. Combined learning and use for a mixture model equivalent to the RBF classifier. *Neural Computation*, 10:281–293, 1998.

[49] D. J. Miller and H. Uyar. A generalized Gaussian mixture classifier with learning based on both labelled and unlabelled data. In *Princeton Conf. on Info. Science and Systems*, pages 783–787, 1996.

[50] D. J. Miller and L. Yan. Approximate maximum entropy joint feature inference consistent with arbitrary lower-order probability constraints: Application to statistical classification. *Neural Computation*, 12:2175–2207, 2000.

[51] D.M. Pennock, E. Horvitz, S. Lawrence, and C.L. Giles. Collaborative filtering by personality diagnosis: A hybrid memory-and model-based approach. In *Proc. of the Sixteenth Conf. on Uncertainty in Art. Intell. (UAI-2000)*, pages 473–480, 2000.

[52] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 19:380–393, 1996.

[53] P. Resnick, N. Iacovor, M. Suchak, P. Bergstrom, and J Riedl. Grouplens:An Open Architecture for Collaborative Filtering of Netnews. In *Proc. of the 1994 Computer Supported Cooperative Work Conf., ACM*, pages 178–186, 1994.

[54] J. Rissanen. Stochastic complexity and modeling. *Annals of Statistics*, 14:1080–1100, 1986.

[55] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Dimensionality Reduction in Recommender System- a Case Study. Technical Report Technical Report CS-TR 00-043, University of Minnesota, 2000.

[56] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating word of mouth. In *Proceedings of the CHI-95 Conference*, pages 210–217. ACM Press, 1995.

[57] B. Shashahani and D. Landgrebe. The effect of unlabeled samples in reducing the small sample size problem and mitigating the Hughes phenomenon. *IEEE Trans. on Geoscience and Remote Sensing*, 32:1087–1095, 1994.

[58] A. Sierra and F. Corbacho. Reclassification as supervised clustering. *Neural Computation*, 12:2537–2546, 2000.

[59] S. Tajudin and D. Landgrebe. Robust parameter estimation for mixture model. *IEEE Trans. on Geoscience and Remote Sensing*, 38:439–445, 2000.

[60] G. Towell. Using unlabeled data for supervised learning. In *Advances in Neural Information Processing Systems, 8*, pages 647–653, 1996.

[61] V. N. Vapnik. *Statistical Learning Theory.* Wiley, 1998.

[62] L. Yan and D. J. Miller. General statistical inference for discrete and mixed spaces by an approximate application of the maximum entropy principle. *IEEE Trans. on Neural Networks*, 11:558–573, 2000.

[63] Q. Zhu. On the minimum probability of error of classification with incomplete patterns. *Pattern Recognition*, 23:1281–1290, 1990.

[64] S.C. Zhu, Y.N. Wu, and D. Mumford. Minimax entropy principle and its application to texture modeling. *Neural Computation*, 9:1627–1660, 1997.

# Vita

John Browning

Education

Ph.D. in E.E. The Pennsylvania State University 8/97 - 12/02

M.S. in E.E. Drexel University 9/91 - 5/96

B.A. in Math Rutgers University 9/77 - 7/82

Employment

Penn State University 8/97 - Present

Research Assistant, Teaching Assistant

General Atronics Corporation, Wyndmoor Pa. 2/83 - 8/97, Software Engineer

Publications

Journal Papers

A Mixture Model and EM Algorithm for Class Discovery, Robust Classification, and Outlier Rejection in Mixed Labeled/Unlabeled Data Sets, submitted to Pattern Analysis and Machine Intelligence (with David Miller).

A Maximum Entropy Approach for Collaborative Filtering, to appear in Journal of VLSI Signal Processing Systems (with David Miller).

Approximate Maximum Entropy Learning for Classification: Comparison with Other Methods, submitted to Pattern Recognition (with David Miller).

Conference Papers - There was one conference paper.