The Pennsylvania State University

The Graduate School

Department of Electrical Engineering

MAXIMUM ENTROPY AND IMPROVED ITERATIVE SCALING FOR CLASSIFICATION ON MIXED SPACES, ENSEMBLE CLASSIFICATION AND EXTENSIONS.

A Thesis in

Electrical Engineering

by

Siddharth Pal

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

December 2006

The thesis of Siddharth Pal was read and approved[1] by the following:

David J. Miller
Associate Professor of Electrical Engineering
Thesis Adviser
Co-Chair of Committee

W. Kenneth Jenkins
Professor of Electrical Engineering
Head of Department of Electrical Engineering
Co-Chair of Committee

Kwang Y. Lee
Professor of Electrical Engineering

Hongyuan Zha
Professor of Computer Science and Engineering

[1]Signatures on file in the Graduate School.

Abstract

Improved iterative scaling (IIS) is an algorithm for learning maximum entropy (ME) joint and conditional probability models, consistent with specified constraints, that has found great utility in natural language processing and related applications. In most IIS work on classification, discrete-valued "feature functions" are considered, depending on the data observations and class label, with constraints measured based on frequency counts, taken over *hard* (0-1) training set instances. In this thesis, we consider the case where the training (and test) set consist of instances of *probability mass functions* on the features, rather than hard feature values. IIS extends in a natural way for this case. This has applications 1) to ME classification on mixed discrete-continuous feature spaces and 2) to ME aggregation of soft classifier decisions in ensemble classification. Moreover, we combine these methods, yielding a method, with proved learning convergence, that jointly performs (soft) decision-level and feature-level fusion in making ensemble decisions.

In this thesis we also consider ensemble classification for the case where there is no common labeled training data for jointly designing the individual classifiers and the function which aggregates their decisions. This problem, which we call *distributed ensemble classification*, applies when individual classifiers operate (perhaps remotely) on different sensing modalities and when combining proprietary or legacy classifiers. The conventional wisdom in this case is to apply fixed rules of combination such as voting methods or rules for aggregating probabilities. Alternatively, we take a *transductive*

approach, optimizing the combining rule for an objective function measured on the unlabeled batch of test data. We propose maximum likelihood (ML) objectives that are shown to yield well-known forms of probabilistic aggregation, albeit with iterative, Expectation Maximization(EM) based adjustment to account for mismatch between class priors used by individual classifiers and those reflected in the new data batch. We also propose an information-theoretic method which generally outperforms the ML methods, better handles classifier redundancies, and which addresses some scenarios where the ML methods are not applicable. This method also well-handles the case of missing classes in the test batch. On UC Irvine benchmark data, all our methods give improvements in classification accuracy over the use of fixed rules when there is prior mismatch.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I am most grateful and indebted to my thesis advisor, Dr. 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 am also grateful and indebted to Dr. W. K. Jenkins, for inspiration and enlightening discussions on a wide variety of topics. I thank my other committee members, Dr. Kwang Y. Lee and Dr. Hongyuan Zha, for their insightful commentary on my work.

Special thanks go out to my friends for their friendship and support.

Finally, I am most grateful to my family, especially my parents for their continued love and support.

Chapter 1

Introduction

Statistical classification problems span over wide domains, varying from speech recognition to gene expression, from handwriting recognition to fault detection in machinery or from stock market predictions to clinical diagnosis. In recent years, Maximum Entropy (ME) techniques for statistical classification and more general inference tasks have found increasing use in natural language processing [1],[2], in scientific applications [3], as well as other domains, e.g. [4].

In this dissertation we extend the maximum entropy(ME) learning to handle mixed discrete-continuous feature spaces and also extend the formulation to novel latent variable ME extension to ensemble classification. Besides the ME-IIS learning, we also consider ensemble classification for the case where there is no common labeled training data for jointly designing the individual classifiers and the function which aggregates their decisions. This problem, which we call *distributed ensemble classification*, applies when individual classifiers operate (perhaps remotely) on different sensing modalities and when combining proprietary or legacy classifiers. The conventional wisdom in this case is to apply fixed rules of combination such as voting methods or rules for aggregating probabilities. Alternatively, we take a *transductive* approach, optimizing the combining rule for an objective function measured on the unlabeled batch of test data.

In this chapter, we start with relevant background. Then we introduce ME based learning and subsequently describe previous work done. We conclude the chapter by summarizing the key contributions and then describing the organization of this dissertation.

1.1 Background

1.1.1 Learning Methods

Statistical machine learning methods can be grouped into the following categories

- Supervised learning (or learning with a teacher): Given a labeled set of data $(\underline{X}, C)$, where C are scalar class labels of d-dimensional vectors $\underline{X}$. The objective is to train a function g so that it can produce for unseen $\underline{X}_u$ (generated from the same distribution as $\underline{X}$) outputs $\hat{C}_u$ which are close as possible to the desired outputs (C_u). Supervised learning methods can be divided into two categories depending on the type of output.

  - *Classification:* The output vector $\hat{C}_u$ represents the class of the an object to recognize. The examples are assigned to a discrete number of classes.
  - *Regression :* This is also known as *function approximation.* Here, the training/testing examples are samples from a learning function $C = f(\underline{X})$. The output $(\hat{C}_u = g(\underline{X}_u))$ of the learned function is continuous.

Classification is a special case of function approximation.

- Unsupervised learning (or learning without teacher): In this type of learning there is no labeled training data. The algorithm has access to data only. Even when the labels are not available one can still try to discover the underlying structure of the data. The useful representation of the data can be in the form of 1) associating the data samples to different clusters (based on distance metrics or on other information criteria) or 2) modeling the data density. Unsupervised classification is much more difficult problem as sometimes the total number of classes is not known. Even the definition of the class is not available. Hence the objective is to cluster of pattern in the data which are similar, thus identifying the potential classes.

- Reinforcement learning: In this kind of learning the learner self learns through trial-and-error interactions with a dynamic environment. The most important difference between reinforcement and supervised learning is that there is no presentation of input/output pairs. Instead, after choosing an action the agent is told the immediate reward and the subsequent state, but is not told which action would have been in its best long-term interests.

1.1.2 Maximum Entropy

The concept of maximum entropy can be best undertsood through an example. Suppose we want to model an instructor's grading policy in a particular course. Let us assume for the purpose of building model (M) that we have access to grades awarded by the instructor from the past years. We might observe that the instructor always chooses the grades from the following set $\{A, A-, B+, B, B-\}$. Using this information we can

add first constraint on the model M that

$$p(A) + p(A-) + p(B+) + p(B) + P(B-) = 1 \tag{1.1}$$

This is first constraint and we can now try to find suitable models which will satisfy this. A few models which satisfy the equation (1.1) are

- $p(A) = 0.2, p(A-) = 0.8, p(B+) = p(B) = p(B-) = 0.$
- $p(A) = 0.4, p(A-) = 0.4, p(B+) = 0.2, p(B) = p(B-) = 0$
- $p(A) = 1$

and so on. As we can imagine there are an infinite number of models which satisfy constraints given by equation (1.1). But all these models make more assumptions than we actually are given. Based on the information (the constraint) we have the more appealing model given by

$$p(A) = p(A-) = p(B+) = p(B) = p(B-) = 0.2. \tag{1.2}$$

This model gives uniform probability to all the grades. Also, this model has maximum entropy among all the models listed above.

Now lets assume we have some more information about the grading policy, for example the instructor gives A, B+ or B grades 72% of the time. This information can be written in constraint form as

$$p(A) + p(B+) + p(B) = 0.72 \tag{1.3}$$

Now we can update our model in equation (1.2) by imposing the constraint given by equations (1.1) and (1.3). Again we can find many distributions which are consistent with above constraints but the least biased and reasonable choice will be

$$p(A) = p(B+) = p(B) = 0.24 \tag{1.4}$$

$$p(A-) = p(B-) = 0.15 \tag{1.5}$$

In this fashion we can add more constraints and update the model accordingly but then we again look for the most uniform model satisfying these constraints. In other words we chose a model which is consistent with all constraints but is as uniform as it can be.

1.1.3 Mixture Modeling

Mixture modeling is a useful tool for density estimation. Here we assume that the density function is formed by a linear combination of basis function. The most commonly used basis function is the normal distribution. The following equation gives the form of the estimated density

$$p(x) = \sum_{j=1}^{M} \alpha_j \Phi(x; \mu_j; \Sigma_j) \tag{1.6}$$

where α_j are called mixing parameters or component masses and $\Phi(x; \mu_j; \Sigma_j)$ are component densities (normal) with means μ_j and covariance(Σ_j). The mixing parameters are chosen to satisfy $\sum_{j=1}^{M} \alpha_j = 1$ and $0 \leq \alpha_j \leq 1$. To generate the data from the distribution given by equation (1.6), one of the components is selected first at random with probability α_j. Then using the component density $\Phi(x; \mu_j; \Sigma_j)$ a data point is generated. The parameters α_j,μ_j and Σ_j are learned using maximum likelihood via Expectation

Maximization (EM) procedure. With mixture models can approximate any continuous density to arbitrary accuracy provided the model has a sufficiently large number of components.

1.2 Maximum Entropy Learning

In [5], the maximum entropy principle is stated as "when we make inference based on incomplete information, we should draw them from that probability distribution that has maximum entropy permitted by the information we do have." The ME approach treats the problem of learning joint or conditional probability models as one of choosing the distribution in order to agree with specified statistical constraints. In a classification context, e.g., to determine whether or not to approve a credit card, the probabilities $P[C = \text{`good risk}', \text{`no college loans}']$ and $P[C = \text{`bad risk}', \text{`no job}']$ may be meaningful statistics to encode. Such probabilities would first be estimated from frequency counts measured over a (labeled training) pool of current credit card holders, each having been hand-labeled (in hindsight) as a good or bad risk. One would then learn the ME class posterior model to agree with specified constraints. Without *any* constraints, the ME posterior is a uniform probability mass function (pmf) over all classes. Each encoded constraint (further) reduces the (maximum) entropy, yielding a sharper class posterior model. The accuracy with which constraints are measured (or known) determines accuracy of the resulting model – if the constraints are accurate, the model should exhibit little or no overfitting.

The ME approach represents a significant departure from more commonly applied approaches to classification, including neural network (multilayer perceptron (MLP),

radial basis function, and learning vector quantizer) classifiers. Consider MLPs trained via backpropagation to minimize a sum of squared errors to class target values. Suppose the training drives the sum-of-squared errors over the training set to *zero*. Effectively, this achieves an equality constraint *sample-wise*, *i.e.* an equality constraint is satisfied for every sample in the training set. However, if many examples are noisy/unreliable, it may be unwise to encode sample-wise constraints. By contrast, ME encodes expected value constraints, measured by *averaging* over the training samples. Such constraints will in general be more reliable than the sample-wise ones implicitly encoded by an MLP.

In addition to the above, there are several attractive features of ME-based learning and inference, compared with other approaches. First, by maximizing entropy while encoding constraints, one is being least-biased/maximally uncertain "with respect to everything else" [6]. Other approaches may implicitly (e.g. via the parametric model structure) make unjustified assumptions which can lead either to biased solutions [6] or to overfitting. Second, since ME frames model learning as constraint encoding, there is flexibility in the information that can be brought to bear – one can accommodate statistics on unconventional features, e.g. [7],[1]. A third advantage is that there are simple, elegant approaches for learning the model – *generalized iterative scaling* [8] and *improved iterative scaling* (IIS) [1] – guaranteed to converge to the (unique) ME solution. By contrast, sensitivity to initialization and, hence, convergence to poor local optima, are issues for some other approaches. Moreover, IIS can often be based on simple parameter updates which do not require the (sometimes difficult) choice of a learning rate/step size, required for gradient-based approaches. Finally, and perhaps less well known, by satisfying constraints using *only* the support of the training data, IIS captures statistical

dependencies between features implicitly expressed through the training data. That is, whereas the parametric *form* of an ME classifier is in some cases equivalent to that of a Bayesian network (BN) classifier [9], unlike the BN the ME approach does not in general make conditional independence assumptions – via the learned parameters, the ME model will capture statistical dependencies between features that are implicitly embodied in the training set. In [9], for an ME model equivalent in form to a Naive Bayes classifier (but without the conditional independence assumptions of Naive Bayes), it was demonstrated that ME achieved better classification results. In [10], ME yielded more accurate models than BNs for more general inference tasks.

1.3 Prior Work

There are four areas of related work: 1) mixed feature space classification; 2) input-dependent boosting; 3) ME-IIS methods and 4) Distributed ensemble classification. Ref. [11] developed an extension of Bayesian network classifiers tailored for mixed continuous-discrete feature spaces. Ref. [12] applied iterative scaling to phonetic classification, based on (continuous-valued) acoustic features. The authors did model individual continuous features using Gaussian mixtures. However, they did not impose constraints involving the (latent) mixture component indices. Instead, their approach encodes constraints on the mean value of the mixture density, conditioned on each class. We believe these constraints are not as informative as the component label constraints we encode. There is also work on a *latent maximum entropy principle* (LME) [13]. There, a latent variable structure is imposed on the probability model, with the best model of this structure sought, to maximize entropy while agreeing with constraints based on the observed

data. In [13], the latent variables are completely hidden/unobserved, with their expected values learned within the ME learning framework. Accordingly, the primary application for the approach in [13] is an ME alternative to the maximum likelihood criterion for learning models with latent variables such as mixture models and hidden Markov models. By contrast, our method assumes the latent variables are observed, albeit imprecisely,*i.e.* we have "probabilistic instances" of them. Unlike [13], our approach encodes measured constraints involving these latent variables. The main thrusts of our method are to extend the types of constraints encoded for continuous-valued features, as well as to extend the types of data from which ME constraints are measured to include probabilistic information sources. The main applications are mixed space classification and learning ensemble combining rules. There are several prior works on input-dependent boosting [14],[15], with the latter method compared against ours in Chapter 3. Finally, there is [16] which unifies standard boosting and iterative scaling techniques. Our work applies the auxiliary function machinery developed there and in [1]. The method we develop extends standard boosting to achieve a particular type of input-dependent combining.

We also focus on the case, which we refer to as *distributed ensemble classification*. This type of problem has been recognized before, either in its general form [17], [18], or as seen in the context of classification over sensor networks [19]. [17] derive a fixed combining rule by applying Bayes theorem and taking proper account of redundancies in the feature spaces used by the local classifiers. This approach requires communication between all pairs of local classifiers to identify the features in common. [19] derive a fixed combining rule for the case where the local classifiers' feature vectors are jointly Gaussian. Ref. [20] developed a transductive maximum likelihood (ML) method to

correct class priors for the case of a single classifier. We extend their approach to address distributed ensembles and demonstrate performance gains over fixed combining when there is mismatch between the true priors and those used by individual classifiers. Moreover, we observe that, for the likelihood objectives considered in this work and in [20], the EM learning is *globally convergent*, i.e. it finds the (unique) maximum likelihood estimate. This was not recognized in [20].

1.4 Contribution

The major contributions of this thesis are as follows:

- **A latent variable extension of ME/IIS for classification on mixed spaces:**
  - We have developed a more accurate, as well as a principled way to handle mixed continuous-discrete feature spaces in building ME conditional probability models, based on the introduction of discrete *latent* variables. Probabilities for the latent variables, learned via mixture modeling, are treated as *data* and used to measure ME constraints involving the latent variables. This method introduces discrete variables *without* feature quantization, i.e. without information loss. The learning approach is based on improved iterative scaling (IIS) [2],[1] and represents a novel extension of existing IIS techniques, which are generally only used for purely discrete spaces and which are typically applied to natural language processing tasks [2].
  - Our method achieves performance comparable to the *best* results, reported for a variety of standard classification algorithms, on nine different UC Irvine

machine learning repository benchmark data sets; also, performance comparable to "best-case" results for support vector machines (SVMS), i.e. where the SVM parameters were tuned to maximize *test set* performance, for each individual data set. These favorable comparisons were obtained *without* significant tuning of the ME model choices (i.e., the choice of ME constraints).

  - This work has been published

- **ME for general inference (GI)**: In earlier work [10], Dr. Miller had proposed methods for learning *all* of the Lagrange multiplier parameters required to specify an ME *joint* density function/ probability mass function. Given these learned parameters, one can address a more general set of inference tasks than standard classification, including 1) treating *any* individual feature as the class feature, to be predicted based on values for the remaining features (generalized classification inference, e.g. collaborative filtering [21]); 2) *paired* inference tasks, i.e. doing inference on any *pair* (or, more generally, any low order collection) of features, given values for the remaining features. A drawback of [10] was the high complexity of the learning method. We have developed a quite fast alternative method for learning ME GI models which is again an extension of improved iterative scaling. This approach achieves essentially identical inference accuracy as [10] (which was found to substantially outperform both Bayesian networks and multilayer perceptrons for GI tasks) while reducing the learning time considerably.

- **Supervised Ensemble Classification:**

- We have identified that there is a natural application and extension of this new ME approach for *combined data and decision fusion* and for *ensemble classification*.
- We propose a simple ME-IIS extension to *boosting*. This method encodes the pairwise pmfs between the *true class* and *decisions* from local classifiers. This method is shown to perform slightly better than Adaboost. [22] gives a connection between Adaboost and logistic regression (i.e., maximum likelihood for exponential models). We also demonstrate how our method differs to that in [22].
- We propose a novel input-dependent ME-IIS ensemble learner. As described in previous contribution, we proposed a novel method based on mixture modeling to encode constraints on continuous features. We devised an approach that assimilates the class posterior probabilities and probabilities associated with latent variables (i.e. the input space information of local classifiers) obtained from local classifiers making the resulting ensemble combiner input dependent. This method is shown to outperform Adaboost, input-dependent boosting [15] and other supervised ensemble aggregation rules.
- This work has been published
- We also describe a Hidden Markov model based approach to ensemble classification. The approach does not fare well experimentally in comparison to our ME-IIS extensions. But still this is a novel approach and is mathematically well defined.

- **Transductive Ensemble Classification:**
  - We developed novel methods to address ensemble classification for the case where there is no common labeled training data for jointly designing the individual classifiers and the function which aggregates their decisions. This problem, which we call *distributed ensemble classification*, applies when individual classifiers operate (perhaps remotely) on different sensing modalities and when combining proprietary or legacy classifiers.
  - We propose maximum likelihood (ML) objectives that are shown to yield well-known forms of probabilistic aggregation, albeit with iterative, EM-based adjustment to account for mismatch between class priors used by individual classifiers and those reflected in the new data batch. These methods are extensions, for the ensemble case, of the work of [20].
  - We also propose an information-theoretic method which generally outperforms the ML methods, better handles classifier redundancies, and which addresses some scenarios where the ML methods are not applicable. This method also well-handles the case of missing classes in the test batch.
  - This has been published

1.5 Organization

This dissertation is divided into seven chapters, including this introductory chapter. ME-IIS classification method for both continuous and mixed discrete-continuous spaces is described in Chapter 2. In Chapter 3, we give our two extensions of ME-IIS

learning for ensemble classification. We also describe a novel Hidden Markov Model based ensemble method. Chapter 4 discusses a *distributed* ensemble classification problem. There we propose novel constraint based transductive learning methods for ensemble classification when no training data is available to supervise the combining rule. In Chapter 5 we apply our transductive IT method for Biometric fusion. In Chapter 6 we propose two extensions of transductive IT methods described in Chapter 4. We finally conclude this thesis by summarizing the dissertation and discussing future work in Chapter 7.

Chapter 2

ME-IIS Classification on Continuous and Mixed Spaces

In most prior work on ME-IIS for classification, a *discrete-valued* feature space is considered, i.e. $(\underline{F}, C)$, $\underline{F} = (F_1, F_2, \ldots, F_{N_d})$, $F_i \in \mathcal{A}_i, |\mathcal{A}_i|$ finite, $C \in \mathcal{C}$, $|\mathcal{C}|$ finite, with the constraints chosen as probabilities on lower order feature collections, conjoined with the class label C, e.g. [2]. As a particular example, one can choose to encode the pairwise probability mass functions (pmfs) $\{P[F_i = j, C = c], i = 1, \ldots N_d, j \in \mathcal{A}_i, c \in \mathcal{C}\}$. One can of course encode higher order pmfs, e.g. third order pmfs $P[F_i, F_j, C]$ or pmfs based on more complicated *functions* of the feature vector $\underline{F}$ [1]. One can also encode select probabilities, i.e. $P[F_i = j, C = c]$, rather than the full pmf $P[F_i, C]$ [1]. Regardless, what is common in prior ME-IIS work is that the constraint probabilities are obtained/estimated based on *frequency counts*, measured over a training data set of hard (feature vector, class label) instances. That is, assuming a discrete-valued feature space, the training set usually consists of $\mathcal{T} = \{(f_1^{(t)}, f_2^{(t)}, \ldots, f_{N_d}^{(t)}, c^{(t)}), t = 1, \ldots, T\}$, $f_i(t) \in \mathcal{A}_i$, T the training set length, with the constraint probability $P[F_i = j, C = c]$ typically estimated as $\frac{1}{T} \sum_{t=1}^{T} \delta(f_i^{(t)} = j, c^{(t)} = c)$, $\delta(\cdot)$ the Kronecker delta function, taking one when its argument is true [1]. Alternatively, we consider the case of "probabilistic instances", wherein, for some (or all) features, we do not have *hard* (0-1) occurrences

[1] Smoothed estimates are also used [23]. However, these are still based on "hard" frequency counts.

$F_i = f_i^{(t)} \in \mathcal{A}_i$, but rather (*soft*) probability mass functions (pmfs), *i.e.* $\{P[F_i = j|t],\ j \in \mathcal{A}_i\}$[2]. Note that "hard" instances is just the special case where the pmf is constrained to have the form $(0, 0, \ldots, 1, 0, \ldots, 0)$, for each t. It should be noted that the frameworks in [8],[1] do not *require* hard feature instances. However, in the prior applications of which we are aware, when the features F_i are discrete-valued, it has been assumed there are hard occurrences $\{f_i^{(t)} \in \mathcal{A}_i,\ \forall i, t = 1, \ldots, T\}$, e.g. [2], [1],[4]. We address ME-IIS for the more general case of "probabilistic instances" and refer to this as a "probabilistic extension of IIS".

One can imagine that "probabilistic instances" could be used to capture uncertainty in the measurement of (discrete-valued) features. However, this is not our primary aim. Our motivation in developing ME-IIS for this case is to well-address two important specialized classification problems: i) classification on mixed continuous-discrete feature spaces and ii) decision fusion in ensemble classification. For mixed spaces $(\underline{F}, \underline{A})$, $\underline{A} \equiv (A_1, A_2, \ldots, A_{N_c}), A_i \in \mathbb{R}$, a key question is: what type of ME constraints should be encoded on a continuous-valued feature A_i ? One possibility is to encode *natural* constraints on continuous-valued features such as moments and covariates or other related quantities, conditioned on the class label [10],[3]. However, as we will later argue, such constraints are in general fairly weak, *i.e.* they are not strongly class-discriminating. Alternatively, one can *quantize* a continuous feature A_i, creating a discrete-valued one, $Q_i = Q(A_i)$. The advantage is that this allows encoding the usual (probabilistic) constraints, *e.g.* pairwise pmfs $\{P[Q_i = q_i, C = c]\}$. However, quantization introduces

[2]We still assume there are hard instances for the class label. However, it is also possible to consider probabilistic (soft) class labels.

information loss. We will propose an alternative to hard quantization, based on our "probabilistic IIS" extension.

2.1 Maximum Entropy Classification

In this section we consider a purely discrete-valued space $\underline{F} = (F_1, F_2, \ldots, F_{N_d})$, $F_i \in \mathcal{A}_i$ and augmented space $(\underline{F}, C)$. As noted in the previous section, in most prior ME-IIS applications, the training data, to be used for learning the probability model, consists of hard (feature vector, class label) pairs, *i.e.* $\mathcal{T} \equiv \{(\underline{f}^{(t)}, c^{(t)}), t = 1, \ldots, T\}$. In this case, the goal is to learn the class posterior model $P[C = c|\underline{F} = \underline{f}]$. However, here, instead, we suppose that in both the training and test data, the feature information consists of the *probabilistic instances* $P_{\underline{f}^{(t)}} = \{P[F_i = j|t] \forall i, j\}$, with the training set given as $\mathcal{T}_p = \{(\{P[F_i = j|t],\ i = 1, \ldots, N_d\ j \in \mathcal{A}_i\}, c^{(t)})\ t = 1, \ldots, T\}$. In this case, our goal is to learn the posterior $P[C = c|P_{\underline{f}^{(n)}}]$, where 'n' generically denotes any instance of a probabilistic feature vector.

For clarity's sake, and without loss of generality, we will develop our ME-IIS approach assuming a particular set (albeit a reasonable choice) of constraints – the set of *pairwise pmfs*, $\{P[C, F_i], i = 1, \ldots, N_d\}$. The constraint probabilities are measured/estimated based on the probabilistic counts in $\mathcal{T}_p$, *i.e.*

$$P_g[C = c, F_i = j] = \frac{1}{T} \sum_{t=1:c^{(t)}=c}^{T} P[F_i = j|t]. \tag{2.1}$$

Here, the superscript 'g' denotes "ground truth". The model's *estimate* of these probabilities is computed as:

$$P_m[C = c, F_i = j] = \frac{1}{T} \sum_{t=1}^{T} P[F_i = j|t] P[C = c|\underline{f}^{(t)}]. \tag{2.2}$$

The Shannon conditional entropy of the model, $H(C|\mathcal{N})$, is expressed as:

$$H(C|\mathcal{N}) = -\sum_{t=1}^{T} \frac{1}{T} \sum_{k=1}^{|\mathcal{C}|} P[C = k|P_{\underline{f}^{(t)}}] \log P[C = k|P_{\underline{f}^{(t)}}], \tag{2.3}$$

based on an assumed uniform pmf over the probabilistic feature vector instance support: $P[\mathcal{N} = t] = \frac{1}{T},\ t = 1, \ldots, T$. The ME problem is to choose the posterior to maximize the conditional entropy while satisfying the constraints:

$$P_m[C = c, F_i = j] = P_g[C = c, F_i = j]\ \forall i, j, c \tag{2.4}$$

and

$$\sum_{k=1}^{|\mathcal{C}|} P[C = k|P_{\underline{f}^{(t)}}] = 1\ \forall t. \tag{2.5}$$

We first convert the constrained problem into an (unconstrained) Lagrangian dual objective:

$$L = H(C|\mathcal{N}) + \sum_{i,j,c} \gamma(F_i = j, C = c) \left(P_m[C = c, F_i = j] - P_g[C = c, F_i = j] \right) + \sum_{t=1}^{T} \lambda_0^{(t)} \left(\left[\sum_{k=1}^{K} P[C = k | P_{\underline{f}^{(t)}}] \right] - 1 \right), \quad (2.6)$$

with $\gamma(F_i = j, C = c)$ the Lagrange multiplier associated with constraint $P_g[C = c, F_i = j]$. The dual problem is then [1],[3]:

$$\min_{\{\gamma(F_i=j,C=c)\},\{\lambda_0^{(t)}\}} \max_{\{P[C=k|P_{\underline{f}^{(t)}}]\}} L. \quad (2.7)$$

The inner maximization with respect to $\{P[C = k|P_{\underline{f}^{(t)}}]\}$ is a concave problem [1]. Supposing all the Lagrange multipliers $\underline{\gamma} \equiv \{\gamma(C = c, F_i = j)\}$ are held fixed and choosing $\lambda_0^{(t)}$ to satisfy the pmf constraints, the optimal pmf, achieving the inner maximization of L, takes the Gibbs/conditional exponential form:

$$P[C = c|P_{\underline{f}^{(t)}}] = \frac{\exp\left(\sum_{i=1}^{N_d} \sum_{j \in \mathcal{A}_i} P[F_i = j|t] \gamma(C = c, F_i = j) \right)}{\sum_{c'=1}^{K} \exp\left(\sum_{i=1}^{N_d} \sum_{j \in \mathcal{A}_i} P[F_i = j|t] \gamma(C = c', F_i = j) \right)}. \quad (2.8)$$

As a precursor to finding the values for the Lagrange multipliers which satisfy the constraints(2.4), we first plug this ME form (2.8) into (2.2) and (2.3), which we then plug back into the Lagrangian expression (2.6). After a bit of simplification, this yields

the negative *posterior log-likelihood*, i.e.

$$L = -\log \prod_{t=1}^{T} \left(\frac{\exp\left(\sum_{i=1}^{N_d} \sum_{j \in \mathcal{A}_i} P[F_i = j|t] \gamma(C = c, F_i = j) \right)}{\sum_{c'=1}^{K} \exp\left(\sum_{i=1}^{N_d} \sum_{j \in \mathcal{A}_i} P[F_i = j|t] \gamma(C = c', F_i = j) \right)} \right). \tag{2.9}$$

The outer minimization with respect to the Lagrange multipliers $\{\gamma(F_i = j, C = c)\}$ in (2.7) is thus equivalent to maximization of the posterior log-likelihood. We thus see a well-known correspondence between ME constrained optimization and maximum likelihood – the Lagrangian dual of the ME constrained problem is the (negative) posterior log-likelihood for the conditional exponential model [2]. Furthermore, from the theory of constrained optimization for a concave objective over a convex set [24], the dual (Lagrangian) objective function is convex in the dual (Lagrange multiplier) parameters, $\underline{\gamma}$, *i.e.* there is a single (global) minimum. Finally, in [1], it is shown that, assuming the constraints are feasible, the (unique) maximum likelihood conditional exponential model is *also* the unique solution to the ME constrained problem. In section 2.3, we develop an IIS algorithm for finding this solution.

2.2 Relation to ME for Hard Feature Instances

The usual case of "hard" feature instances $F_i = f_i^{(t)} \in \mathcal{A}_i$ is the special case of probabilistic instances wherein $P[F_i = j|t] = \delta(f_i^{(t)} = j)$. Specializing to this, the ME posterior is:

$$P[C = c|\underline{F} = \underline{f}] = \frac{\exp(\sum_{i=1}^{N_d} \gamma(C = c, F_i = j))}{\sum_{c'=1}^{K} \exp(\sum_{i=1}^{N_d} \gamma(C = c', F_i = j))}, \tag{2.10}$$

with ME learning again equivalent to posterior log-likelihood maximization, now measured with respect to the "hard" training set $\mathcal{T}$.

2.3 An IIS Algorithm for Probabilistic Instances

A simple, algorithm for maximizing L is *improved iterative scaling* [1]. IIS is based on the derivation of an auxiliary function, one easy to maximize, whose optimization lower bounds (above zero) the gain in log-likelihood associated with replacing the current parameter vector $\underline{\gamma}^{(n)}$ (at iteration n) by the updated vector $\underline{\gamma}^{(n+1)} = \underline{\gamma}^{(n)} + \underline{\Delta\gamma}^{(n+1)}$ [1],[16]. This auxiliary function decouples *all* of the Lagrangian increments $\Delta\gamma(C = c, F_i = j)$, allowing a simple determination of the "optimal" increments at each iteration, those which maximize the lower bound and ensure a gain in the original log-likelihood objective. Following the definition in [16], we define an auxiliary function $A(\underline{\Delta\gamma}|\underline{\gamma})$ for our ME problem as any function satisfying the following two properties:

A1. $L(\underline{\gamma} + \underline{\Delta\gamma^*}) - L(\underline{\gamma}) \geq A(\underline{\Delta\gamma^*}|\underline{\gamma}) \geq 0$.

A2. $A(\underline{\Delta\gamma^*}|\underline{\gamma}) = 0 \Rightarrow$ the constraints in equation (2.4) are all met.

Here, $\underline{\gamma}$ is the "current" parameter vector and $\underline{\Delta\gamma^*} = \arg\max_{\underline{\Delta\gamma}} A(\underline{\Delta\gamma}|\underline{\gamma})$, *i.e.* it is the algorithm's update, given the current vector. The first property states that, with updates chosen to maximize the auxiliary function, the log-likelihood is nondecreasing with each iteration. Since the log-likelihood is upper-bounded (by zero), the first property essentially implies that the log-likelihood difference, and hence also the maximum of the auxiliary function, must converge to zero. The second property ensures that, when this occurs, the constraints are necessarily met. Finally, since [1] proves there is a *unique* conditional exponential model that satisfies the constraints and which is the solution to

the ME (and maximum likelihood) problems, the parameter vector $\underline{\gamma} + \underline{\Delta\gamma^*}$ achieving $A(\underline{\Delta\gamma^*}|\underline{\gamma}) = 0$ must be the ME-optimal vector.

For constrained Bregman-distance optimization, a generalization of the ME problem, a formal proof of convergence, based on use of the auxiliary function properties as outlined in the proof sketch above, is given in [16]. This convergence proof applies to our ME problem if we can cast our ME learning algorithm as an iterative optimization of an auxiliary function (satisfying A1 and A2). In Appendix A, using the approach in [25], we derive a function that satisfies A1. Appendix B proves this function also satisfy A2 and is thus a valid auxiliary function [16] for our problem.

This auxiliary function is:

$$A(\underline{\Delta\gamma}|\underline{\gamma}) = T + \sum_{t=1}^{T}\sum_{i=1}^{N_d}\sum_{j\in\mathcal{A}_i} P[F_i = j|t]\Delta\gamma(C = c^{(t)}, F_i = j) - \frac{1}{N_d}\sum_{t=1}^{T}\sum_{k=1}^{N_c}\sum_{i=1}^{N_d}\sum_{j\in\mathcal{A}_i} P[C = k|P_{\underline{f}^{(t)}}]P[F_i = j|t]e^{N_d\Delta\gamma(C=k,F_i=j)}. \quad (2.11)$$

The IIS updates, which maximize this function over $\underline{\Delta\gamma}$ at each iteration, are easily found, by setting $\nabla A(\underline{\Delta\gamma}|\underline{\gamma}) = \underline{0}$, which yields:

$$\Delta\gamma^{(n+1)}(C = c, F_i = j) = \frac{1}{N_d}\log\frac{P_g[C = c, F_i = j]}{P_m^{(n)}[C = c, F_i = j]},\ \forall c, i, j \quad (2.12)$$

where the superscript on $P_m[\cdot]$ indicates this probability is measured (based on (2.2) and (2.8)) using the current parameter estimates, $\underline{\gamma}^{(n)}$. Note that the parameter increments are evaluated separately, i.e. "in parallel". Note also that, from (2.12), there is no update for $\gamma(C = c, F_i = j)$ whenever $P_m^{(n)}[c, j] = P_g[c, j]$.

2.4 Classification on Mixed Feature Spaces

The previous section considered discrete spaces. Here we address the case of mixed discrete-continuous spaces $(\underline{F}, \underline{A})$. In previous work, e.g. [10], [3], *natural* constraints on continuous features were encoded in learning ME models, i.e. constraints related to class-conditional means and second moments (and possibly covariates).

Unfortunately, these *natural* constraints are in general *weak*, i.e., they are not strongly discriminating between the different classes. This can be understood by considering the example in Figure C.1. Here, we show two class-conditional densities, each consisting of a mixture of three Gaussians. Assume the classes are equally likely. The two distributions are clearly different. However, the class-conditional means and variances are the same, i.e., $E[A|C=1] = 0.02 = E[A|C=2]$, $\sigma^2_{A|C=1} = 55.30 = \sigma^2_{A|C=2}$. For this example, consider maximizing the entropy $H(C|A)$ with respect to the posterior $\{P[C=c|a] \; \forall a\}$ while satisfying the mean and variance constraints. The *unconstrained* maximum of $H(C|A)$ is achieved by the uniform pmf $P[C=1|a] = P[C=2|a] = 0.5$, any a. But this solution is *also* consistent with satisfaction of the constraints. In particular,

$$P[C=c|a] = \frac{P[C=c] f_{A|C}(a|C=c)}{\sum\limits_{c'=1,2} P[C=c'] f_{A|C}(a|C=c')}, \quad c = 1, 2 \tag{2.13}$$

with $P[C=1] = P[C=2] = 0.5$, and if we choose $f_{A|C}(a|C=1) = f_{A|C}(a|C=2)$ to be Gaussian with mean and variance equal to the common values, i.e., $E[A|C=*]$ and $\sigma^2_{A|C=*} (* = 1\,,\, 2)$, we have a uniform posterior that maximizes $H(C|A)$ *and* satisfies the constraints. Unfortunately, the maximum *a posteriori* classification error rate for

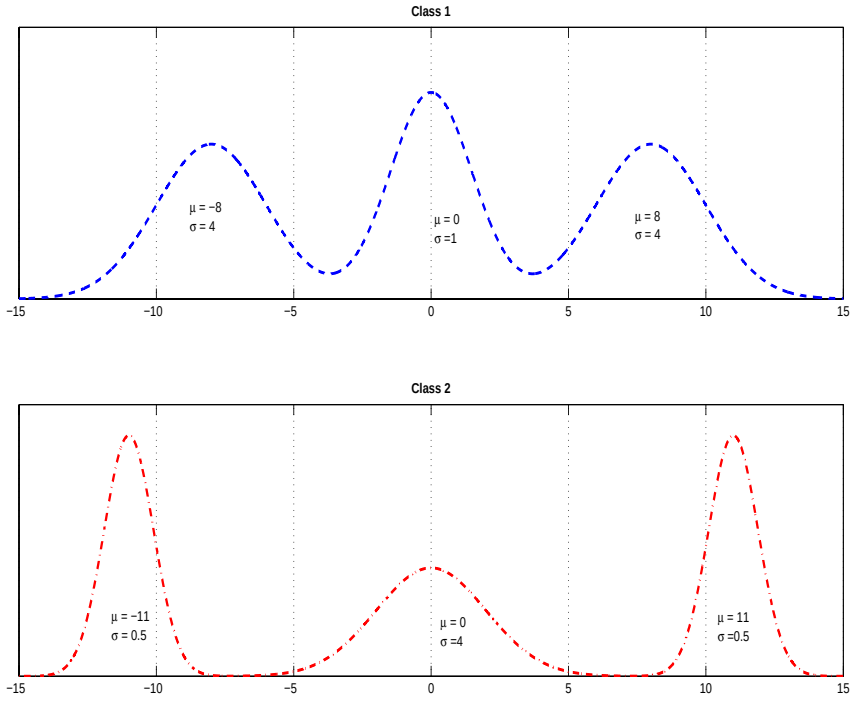

Fig. 2.1. An example two-class mixture for which moment constraints are not class discriminating

this model is 50%. In summary, the natural (moment) constraints for continuous-valued features capture "global" data characteristics but fail, e.g. to capture multimodality of the (class – conditional)distribution.

A standard way to glean "local" (sub-interval-based) information from continuous-valued features, for use in Bayesian networks and ME models, is by *forcing* a finite-valued set, i.e. by feature quantization, $Q_i = Q(A_i), Q_i \in \{1, 2, \ldots, N_i\}$, N_i the number of levels [26]. Given these (derived) discrete features, one can then encode the pairwise pmfs $P[Q_i, C]$. However, quantization is information-lossy. Moreover, in general, heuristic techniques are needed to determine both the number of quantization levels and the quantization thresholds. These choices can impact model accuracy. Alternatively, we propose to create discrete-valued features and encode "local" statistics into ME-IIS learning without resorting to hard quantization. We accomplish this by introducing discrete *latent* variables associated with the continuous features and by supposing that we are given (or can estimate) probabilities on the latent values, conditioned on the continuous feature values – these are our "probabilistic feature instances". We obtain these latent probabilities via *mixture modeling* – we apply mixture-based density estimation to fit the data along either a single continuous-feature dimension or along a subspace of the continuous feature space. There are many possibilities here. At one extreme, we could introduce a single latent variable associated with the full vector $\underline{A} \in \mathbb{R}^{N_c}$ – this will entail learning a single N_c-dimensional mixture. At the other extreme, we can introduce one latent variable for *each* individual continuous feature – this entails learning a mixture for each feature A_i. While mixture modeling in high dimensions may be subject to poor local optima of learning and to inaccuracy in model order (number of components) estimation, these

problems are less severe for low-dimensional spaces. Thus, the choice of the particular latent variable approach may reasonably depend on how large is N_c. The number of mixture components can be estimated in a principled way, e.g. either via the*Bayesian information criterion*(BIC) [27], as considered here, or via cross validation [28]. Encoding probabilistic constraints involving the latent variables yields a simple IIS extension with enhanced modeling capabilities for mixed spaces.

The basic procedure we propose, assuming a mixture is learned for each A_i:

1. Given the training set, $(\underline{f}^{(t)}, \underline{a}^{(t)}, t = 1, 2, ..., T)$ associated with a mixed feature space $(\underline{F}, \underline{A})$: learn a Gaussian mixture for A_i, using BIC [27] to select the model order, L_i. The latent variable $M_i \in \{1, 2, ...L_i\}$ is the component index for the L_i-component mixture.

2. Based on the learned model, compute *a posteriori* probabilities via Bayes rule on the latent values for each training sample: $P[M_i = j|{a_i}^{(t)}; \Lambda_i] \ \forall i \ \forall j, \ t = 1, ..., T$, Λ_i the set of mixture parameters.

3. Learn an ME-IIS model, encoding e.g. the pairwise pmf constraints $P[F_i, C]$ $i = 1, ..., N_d$ and $P[M_i, C]$ $i = 1, ..., N_c$. This is accomplished via the "probabilistic extension of IIS" developed in the previous section, which accommodates the "probabilistic feature instances" for the latent variables, i.e., $\{P[M_i = j|t], t = 1, ..., T\}$.

For a latent variable, the constraint probability is :

$$P_g[C = c, M_i = j] = \frac{1}{T} \sum_{t=1}^{T} \delta(c^{(t)} = c) P[M_i = j|a_i^{(t)}; \Lambda_i] \tag{2.14}$$

and the model's *estimate* is:

$$P_m[C=c, M_i=j] = \frac{1}{T}\sum_{t=1}^{T} P[C=c|\underline{f}^{(t)}, \underline{a}^{(t)}]P[M_i=j|a_i^{(t)};\Lambda_i]. \tag{2.15}$$

The constraints $P[F_i, C]$ were considered previously. Proceeding as in the last section, the ME posterior is derived and shown to have the form

$$P[c|\underline{f}, \underline{a}] = \frac{e^{\left(\sum_{i=1}^{N_d}\gamma(c,f_i)+\sum_{i=1}^{N_c}\sum_{j=1}^{L_i}P[M_i=j|a_i;\Lambda_i]\gamma(c,M_i=j)\right)}}{\sum_{c'=1}^{K} e^{\left(\sum_{i=1}^{N_d}\gamma(c',f_i)+\sum_{i=1}^{N_c}\sum_{j=1}^{L_i}P[M_i=j|a_i;\Lambda_i]\gamma(c',M_i=j)\right)}}. \tag{2.16}$$

As in standard IIS, solving the ME problem boils down to conditional log-likelihood maximization. Moreover, a generalization of the standard IIS auxiliary function can be derived, whose maximization forms the basis for convergent ME learning. Carrying this derivation through, we find the updates are given by

$$\Delta\gamma^{(n+1)}(C=c, F_i=j) = \frac{1}{N_d+N_c}\log\frac{P_g[C=c, F_i=j]}{P_m^{(n)}[C=c, F_i=j]},\ \forall c,i,j \tag{2.17}$$

and

$$\Delta\gamma^{(n+1)}(C=c, M_i=j) = \frac{1}{N_d+N_c}\log\frac{P_g[C=c, M_i=j]}{P_m^{(n)}[C=c, M_i=j]},\ \forall c,i,j \tag{2.18}$$

This procedure easily specializes for the case of a purely continuous space $\underline{A}$.

Evaluation on the Figure C.1 example: We tested this new "latent variable" ME-IIS approach for the data from Figure C.1, based on 3000 training samples and 3000 test

samples, independently drawn from the distribution. This is a purely continuous space with $N_c = 1$. We learned the mixture and BIC–selected the model order(correctly) as 6. We then learned the associated latent variable ME model. We measured training and test set classification error rates of 19.20% and 19.83%, respectively. This is substantially better than the 50% rate achieved by encoding "natural continuous constraints". This ME-IIS extension for mixed spaces is further evaluated in section 2.6.

2.5 ME General Inference Models on Discrete Spaces and Extensions on Mixed Spaces

While our primary research objective and results relate to extending ME classification (*a posteriori* conditional probability models for the class variable) to well-handled mixed features, we also made significant strides in improving the practicality of ME *general inference* models. Recall that a *general inference (GI) engine* [10] is effectively a model for the joint distribution on the full feature space. Given this joint distribution (or, in our case, the full complement of Lagrange multipliers that specify it), we can compute the *a posteriori* probability of *any* individual feature or small collection of features, given values for the remaining features. I.e., this method builds a model that can "on the fly" be used to treat any feature as the class feature, to be predicted given the remaining feature values. We have developed an extension of improved iterative scaling (IIS) [2] that affords very efficient learning of this model.

The IIS approach for learning ME *a posteriori* models $P[C|\underline{F}]$ (for a class feature C, given a discrete feature vector $\underline{F} = (F_1, F_2, \ldots, F_N)$) [2] is obtained by first recognizing that the ME problem is *equivalent* to that of maximizing the conditional

log-likelihood

$$L(\Lambda) \equiv \sum_{t=1}^{T} \log P[C = c^{(t)} | \underline{F} = \underline{f}^{(t)}], \tag{2.19}$$

measured with respect to the training set examples $\{(c^{(t)}, \underline{f}^{(t)}), t = 1, \ldots, T\}$. Here, $P[C = c^{(t)} | \underline{F} = \underline{f}^{(t)}]$ is given in its ME (Gibbs) form [10], based on the Lagrange multipliers associated with all the pmf constraints. In our case, we assume all pairwise pmfs as constraints, i.e. $\Lambda = \{\gamma(F_n = f_n, F_m = f_m)\}$. A very efficient way of maximizing this log-likelihood is based on the introduction of an *auxiliary function* $B(\Delta|\Lambda)$, with Λ the (current) set of Lagrange multipliers and with $\Delta \equiv \{\delta(F_n = f_n, F_m = f_m)\}$ a set of *updates* for these multipliers, i.e., at iteration k, $\Lambda^{(k+1)} = \Lambda^{(k)} + \Delta_k$ [2]. The key property of this auxiliary function is that $L(\Lambda + \Delta) - L(\Lambda) \geq B(\Delta|\Lambda)$. Thus, by maximizing $B(\Delta|\Lambda)$ with respect to Δ (assured to achieve a positive result) and then updating Λ, we are taking a step that is assured of increasing the log likelihood. The computational efficiency of this approach derives from the fact that there is a *closed form update* (maximization) of $B(\cdot)$ with respect to Δ, given the current Λ held fixed. In [29], we have found this approach to be quite fast for building conditional probability models for single feature inference (classification). The algorithm described in [2] can in fact also be applied to learn the *joint* pmf (for GI tasks). However, we believe this approach does not scale well with increasing dimensionality – it requires computationally intensive Monte Carlo simulations. Alternatively, similar to [10], we maximize the *sum* of the conditional log-likelihoods in learning the ME joint pmf parameters, i.e., for the

discrete feature case:

$$L(\Lambda) = \sum_{i=1}^{N} \sum_{t=1}^{T} \log P[F_i = f_i^{(t)} | \underline{F_{(-i)}} = \underline{f_{(-i)}}^{(t)}]. \tag{2.20}$$

Following the methodology in [2], an auxiliary function $B(\Delta|\Lambda)$ is derived, again with closed form updates for the Δ parameters (updated together in parallel, given the current Λ). The climax of the derivation is a parallel update of the Lagrange multipliers (all at once). For the case of pairwise pmf constraints, the updates are given by:

$$\delta(F_m = p, F_n = q) =$$

$$\frac{1}{N} \log \left(\frac{2N(F_m = p, F_n = q)}{\sum\limits_{t=1:f_n^{(t)}=q}^{T} P[F_m = p | \underline{F}_{(-m)} = \underline{f}_{(-m)}^{(t)}] + \sum\limits_{t=1:f_m^{(t)}=p}^{T} P[F_n = q | \underline{F}_{(-n)} = \underline{f}_{(-n)}^{(t)}]} \right)$$

$$\forall m, n, p, q \tag{2.21}$$

These updates are non-decreasing in the objective function (the sum of log-likelihoods). This algorithm learns the *full* complement of Lagrange multipliers, which specify the *joint* probability mass function.

Based on our latent variable extension from the last section, an IIS approach with updates very *similar* to (2.21) can also be derived for the mixed continuous-discrete feature space case. We believe this new learning approach for ME GI models will allow scaling up these methods to address large feature spaces.

We performed some experiments on the UC Irvine machine learning repository to evaluate the inference accuracy of this GI model and to assess the computational efficiency of the learning procedure (preliminary to evaluating how well the method scales to very large feature spaces). Figure 2.2 shows a learning curve for our ME GI method for discrete feature spaces, applied to the UC Irvine *Mushroom* database. This data set has 22 discrete features with various mushroom characteristics and a binary class label (indicating whether the mushroom is edible or poisonous). As the figure 2.2 indicates, the sum of log-likelihoods objective $L(\Lambda)$ increases monotonically during learning (as is theoretically guaranteed). The learning for the ME GI model (with the capability of predicting each of the 23 Mushroom features (or any pair) given values for the remaining features) is very fast – for this data set, with 23 dimensions and 4000 examples, the model is learned in roughly 20 seconds on a laptop running at 1 GHz.

2.6 Experiments

In this section, we compare our ME-IIS method with other supervised classification methods. We used 14 datasets from the UC Irvine machine learning repository. The ME-IIS learning used a fixed stopping criteria. This criteria was chosen empirically. Figure 2.3 gives the example curve illustrating that test set performance is not insensitive to the choice of stopping rule. The dataset *waveform* consist of three classes and 21 continuous features. The dataset *cmc* (contraceptive method choice) has three classes, two continuous features and seven discrete features. For both continuous and mixed feature spaces we stop the ME-IIS learning when the relative change in log-likelihood drops below 1e-4.

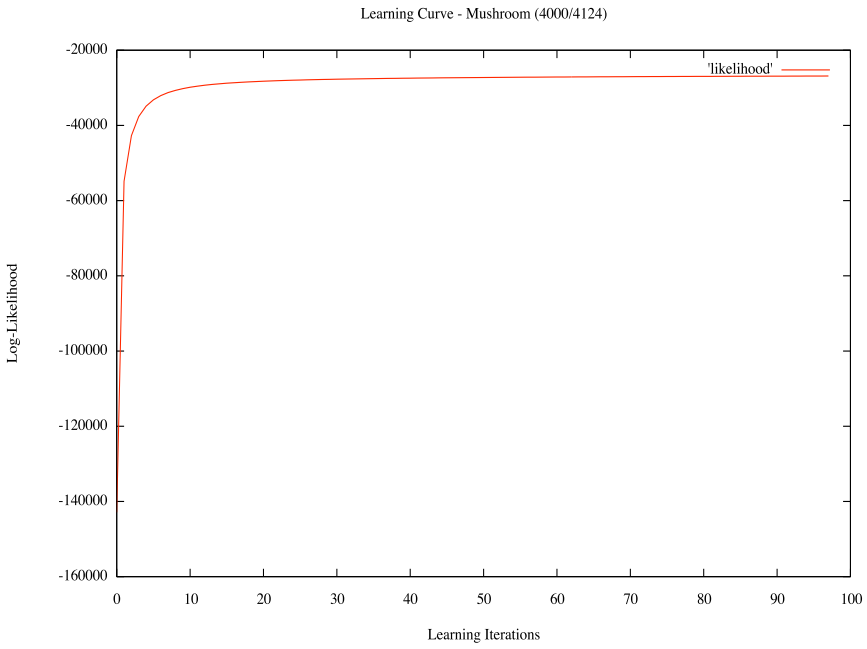

Fig. 2.2. Learning curve for discrete space ME approach applied to the UC Irvine *Mushroom* data set.

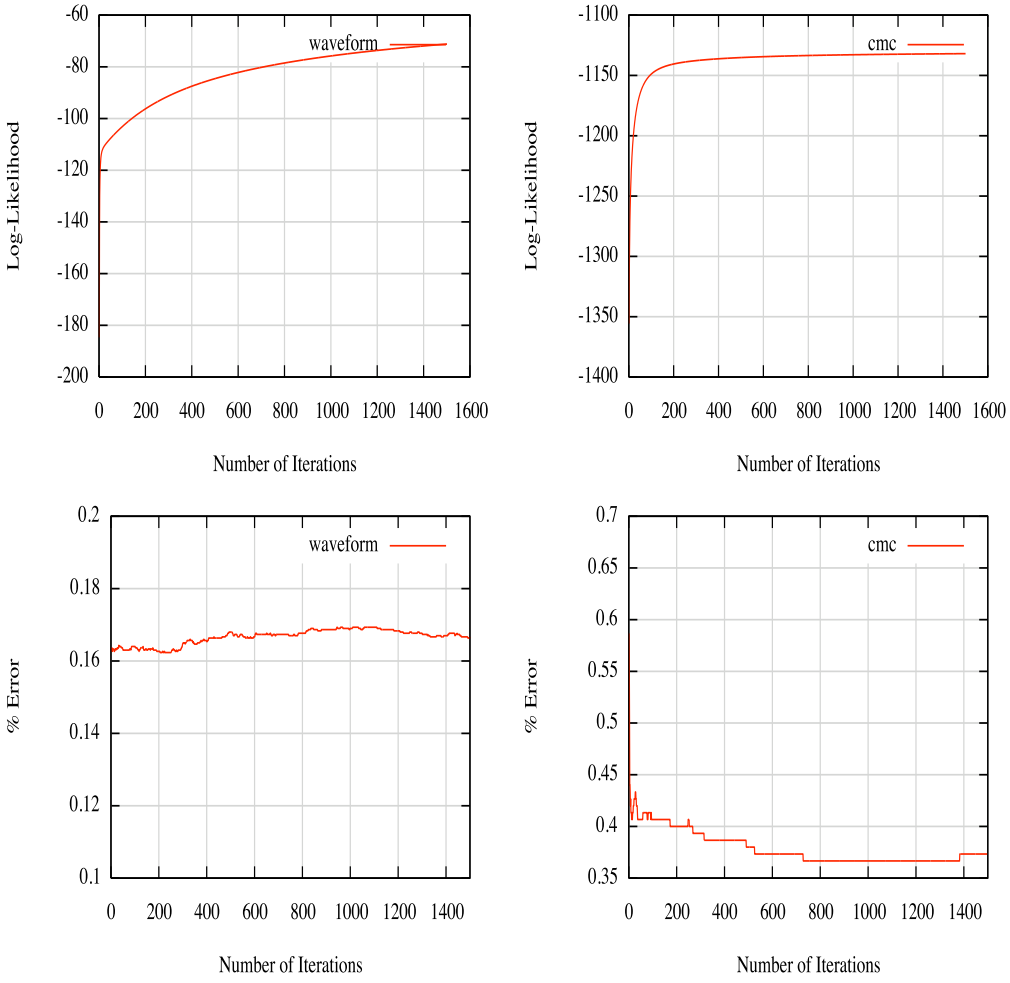

Fig. 2.3. Learning curve and generalization error as function of learning iterations for ME-IIS classification on UC Irvine *waveform* and *contraceptive choice method* data set.

2.6.1 Classification on Continuous and Mixed Feature Spaces

To evaluate classification accuracy, we performed 10-fold cross validation except on *satellite*, where the training/test (4435/2000) split was given.

Continuous Spaces

For the ME-IIS approach, we tried learning a class–dependent mixture model for each continuous feature, yielding class conditional latent variables and associated probabilities, i.e. $P[M_i = j|a_i^{(t)}; \Lambda_i; c]$. These probabilities replace $P[M_i = j|a_i^{(t)}; \Lambda_i]$ in equations (2.1), (2.2), and (2.8). For *diabetes* using a class–dependent mixture in ME-IIS learning yielded a generalization error rate of 21.6% while using a class–independent mixture, the error rate was 30.83%. Similarly on *liver disorder* we get generalization error rates of 30.6% and 43.43%, respectively. Clearly, the class–dependent mixture encodes more information and was thus chosen for all experiments. We assumed Gaussian component densities. For model order selection we used BIC. We compared the performance of our method with the results of the *best-performing* algorithm from [30], for each tested data set. We also compared with a nonlinear support vector machine (SVM) whose parameters were tuned for each tested data set. In [30] twenty-two decision trees algorithms, nine classical and modern statistical algorithms, and two neural networks were evaluated. For SVMs, we used the RBF kernel and tried various choices of the width parameter (W) and cost parameter (C), selecting the choice with the best test error, for each data set. Table 2.1 gives a comparison between the methods. Our approach is competitive with the optimized SVM and the best results from [30], which are achieved by different algorithms on the different data sets. For example, in [30] a decision tree variant gave

the best result on *liver disorder* while quadratic discriminant analysis (QDA) did not perform well. On the other hand, on *vehicle*, QDA gave the best result and decision trees did not fare well. However, our method gave competitive results on all the data sets. On *vehicle*, *satellite* and *segment* we found that some of the features are highly correlated and that learning multivariate Gaussian mixtures on small collections of features gave better performance than using a separate (1-D) mixture for each continuous feature. In Table 2.1 the result in parentheses indicates the error rate when the former (multivariate) mixture modeling is applied to some features, while the result outside the parentheses corresponds to unidimensional feature modeling.

Table 2.1. Classification results for continuous feature spaces.

UC Irvine Dataset	**ME-IIS Error Rate %**	**Best Result from [30] Error rate %**	**Nonlinear SVM Error rate %**
Diabetes	21.6	22	23.8
Liver Disorder	30.6(27.8)	28	28.7
Vehicle	23.6(18.1)	15	17.3
Satellite	12.4(11.4)	10	8.5
Segment	1.2(0.95)	-	1.8
Waveform	16.57	15	-

Mixed Discrete-Continuous Spaces

To evaluate performance on mixed spaces we used *hepatitis*, *horse-colic*, *credit-screening* and *cmc* from UCI machine learning repository. The dataset *Smoking restriction* is taken from http://lib.stat.cmu.edu/dataset/csb. This is a three class dataset with five discrete and three continuous features. For comparison purposes we had same data as used by

[30]. We performed 10-fold cross validation on *hepatitis*, *credit-screening* and *cmc*. The training/test split for *horse-colic* was (300/68). We compared our latent variable ME approach to the standard discrete space ME-IIS (section 2.2), based on quantizing all the continuous features. We performed uniform quantization of continuous features using number of levels the same as the number of BIC–selected components for each continuous feature. Table 2.2 gives classification error rates. Our method performs better than the standard approach based on feature quantization.

Table 2.2. Classification results on mixed discrete-continuous feature spaces.

UCI Dataset	**ME-IIS Error Rate %**	**Quantizing continuous features Error rate %**	**Best Result from [30] Error rate %**
Hepatitis	14.1	15.4	-
Credit Screening	11.8	13.4	-
Horse Colic	16.1	16.1	-
Contraceptive method choice (cmc)	43.5	-	43.0
Smoking restriction	31.0	-	30.0

2.6.2 General Inference on Discrete Spaces:

Table 2.3 gives a comparison against a naive Bayes model on several UC Irvine machine learning data sets. The naive Bayes results are only shown here for the standard classification task (criterion 1). The ME model is evaluated for this measure and also for two GI tasks: criterion 2 is the inference accuracy, averaged over treating *each* discrete feature as the class feature, to be predicted based on knowledge of all other features; criterion 3 is the inference accuracy, averaged over predicting *every pair* of features,

given knowledge of all the remaining feature values. While we do not report results for the naive Bayes model for these latter two (very demanding) GI tasks here, we have found that just as for the standard classification criterion (shown in the table), the performance of the ME model is superior on these tasks. It is important to note that the naive Bayes model has nearly *identical* model complexity as the ME approach [9] – the *form* of the decision rule is essentially the same, but the parameter values are not. The advantage of the ME model lies solely in not making unnecessary statistical assumptions (conditional feature independence), which are required for the Naive Bayes model – recall that by satisfying constraints using *only* the support of the training data, the ME approach captures dependencies implicit in the training data. The fact that our ME approach achieves superior accuracy indicates that ME is capturing significant statistical dependencies, expressed by the data.

Table 2.3. General inference results for our ME method, in comparison with a naive Bayes model.

Dataset	**ME**	**NB**
Mushroom		
Criteria 1	00.17	02.14
Criteria 2	18.07	-
Criteria 3	42.66	-
Zoo		
Criteria 1	0.00	02.12
Criteria 2	06.27	-
Criteria 3	17.92	-
Balance		
Criteria 1	07.63	11.67
Criteria 2	56.21	-
Criteria 3	87.89	-

Chapter 3

Supervised Ensemble Classification

Ensemble classification systems form ultimate decisions by aggregating the (hard or soft) decisions made by classifiers in a collection or ensemble. This paradigm is often motivated by considerations of accuracy, taking into account factors that affect (standalone) classifier performance such as limited training data, local optima of the training objective, a lack of prior knowledge of the class-discriminating features, and of their underlying distributions. Diversity of classifiers in an ensemble – in the feature spaces they use, in their structures, in their parameter initializations, and in their training data – can all help to reduce inductive bias associated with a single classifier. Ensembles have been justified e.g. under the assumption of statistically independent decisionmaking [31] and from the standpoint of variance reduction [32].

We can broadly categorize ensemble techniques according to their method of training. For *jointly optimized ensembles*, the base classifiers and the aggregation function which combines their decisions are jointly designed, using a common set of training (and possibly validation) data. Such methods include boosting [33], mixture of experts [34], as well as other techniques, e.g. [35]. A second category focuses on training the aggregation rule, given fixed classifiers. An example is [36]. Finally, there are techniques that do not involve *any* training of the aggregation rule. These methods apply fixed mathematical rules to combine decisions. Such techniques include hard decision voting

[31], applications of Bayes rule [37], arithmetic averaging [38], variants of averaging [18], robust averaging [39], and Dempster-Shafer theory [40].

3.1 Existing Methods

There are a variety of ensemble aggregation techniques [41],[42].In this section, we briefly discuss a few of these methods.

3.1.1 Bagging

Bagging [38] is a "bootstrap" ensemble method that generates base classifiers for its ensemble by training each classifier on a random redistribution of the training set.

Table 3.1. Hypothetical runs of Bagging.

Original Training set	1,2,3,4,5,6,7,8,9,10
Bagging	
	Resampled training data
Base classifier 1	3,2,8,1,5,9,7,2,4,3
Base classifier 2	3,2,3,1,5,9,7,2,9,3
Base classifier 3	6,3,1,8,2,6,2,8,2,1
Base classifier 4	5,1,10,3,4,9,7,4,1,10
Base classifier 5	8,3,5,4,2,6,1,2,3,9

Each classifier's training set is generated by random sampling (with replacement) from original training set, so each of original examples may appear once, more than once or not at all. Table 3.1[1]gives an example of how the new training data for base classifier is generated by resampling the original training data. The basic flowchart for bagging is

[1]motivated from Figure 2 in [43].

given below.

Input: Training set S, Base classifier C, Ensemble size N_e

1. For m = 1 to N_e
   - resample with replacement from the training set to obtain training set $\hat{S}_m$
   - $C_m = C_m(\hat{S}_m)$
2. $C^*(x) = \text{argmax}_{c \in N_c} \sum_{C_m(x)=c} 1$

Output: classifier C^*

To obtain the final decision the individual base classifier decisions are averaged over the ensemble size, thereby reducing its variance. An important thing to remember for bagging is that the training set of one base classifier is not dependent on the performance or training set of another base classifier.

3.1.2 Boosting

One of the most popular ones is the general paradigm of *boosting* [33], which *jointly* trains the individual classifiers and the function which combines their decisions. The main motivation of boosting is to combine the predictions of multiple weak classifiers to produce a superior decision. Boosting trains a sequence of classifiers, with each focused on the "most difficult" training examples seen until now, i.e. those with the highest error rate based on the current ensemble. This focusing is done by applying

adaptive weights to the training samples, effecting a training set "resampling" for the design of each stage classifier. The boosting variant Adaboost.M1, specified below for a two–class problem.

Adaboost.M1 training:

1. Initialize the weights, $w_t^{(1)} = \frac{1}{T}$, $t = 1, ..., T$
2. For m = 1 to M
   - Learn classifier $G_m(\underline{a})$ based on the training set with weights $\left\{ w_t^{(m)} \right\}$
   - compute $e_m = \dfrac{\sum_{t=1}^{T} w_t^{(m)} \delta(y_t \neq G_m(\underline{a}^{(t)}))}{\sum_{t=1}^{T} w_t^{(m)}}$
   - compute $\alpha_m = \ln\left(\dfrac{1 - e_m}{e_m}\right)$
   - update weights via $w_t^{(m+1)} = w_t^{(m)} \exp(\alpha_m \delta(y_t \neq G_m(\underline{a}^{(t)}))),\ t = 1, ..., T$

Inference:

1. $G(\underline{a}) = \text{sign}\ (\sum_{m=1}^{M} \alpha_m G_m(\underline{a}))$.

In boosting also, the original training data is resampled to generate the training data for successive stage(base) classifiers. But the difference between sampling for bagging and boosting is that, in case of boosting the training set for the next stage depends on the performance of the current stage classifiers. The sampling procedure is biased towards

the training samples which were misclassified by the current stage classifier. Again this is explained in Table 3.2

Table 3.2. Hypothetical runs of Boosting.

Original Training set	1,2,3,4,5,6,7,8,9,10
Boosting	
	Resampled training data
Base classifier 1	3,2,8,1,5,9,7,2,4,3
Base classifier 2	1,10,5,1,5,9,7,5,6,1
Base classifier 3	1,8,1,4,2,5,1,8,3,1
Base classifier 4	1,10,1,3,1,2,1,4,1,10
Base classifier 5	1,3,1,1,2,1,4,2,1,1

3.2 Application to Ensemble classification (Boosting):

Here, we apply the new ME-IIS approach to learn a function that aggregates the decisions of individual classifiers in an ensemble. The method we develop improves upon the standard boosting approach.

As seen from the inference step (in Adaboost.M1 flowchart), the "confidence weights" applied to each classifier's output in the classifier combination are *constants* – each classifier is given decision weight commensurate with its *overall* reliability (average error rate). However, some classifiers may be trained to focus on (and hence may only be very accurate for) a small region of the feature space. One can imagine that these classifiers should in principle only be given large weight in the region on which their training was focused, not everywhere. Several approaches have been proposed to remedy this perceived problem and make boosting input-dependent [14],[15]. We here

propose two approaches, one is a simple ME aggregation method and another is a novel approach which is input dependent. These are experimentally evaluated in section 3.4.

3.2.1 ME-IIS for Simple Classifier Combining

Suppose the ith classifier , $i = 1, \ldots, N_e$, takes in $\underline{A}_i$ [2] and produces a *decision* random variable $\hat{C}_i \in \mathcal{C}$. We suppose the classifier produces soft decisions, i.e., $\left\{P[\hat{C}_i = c|\underline{A}_i = a_i], c \in \mathcal{C}\right\}$. We propose to encode the pairwise pmfs $P[C, \hat{C}_i]$, $i = 1, .., N_e$ in learning the *ensemble* posterior $P[C = c|\{a_i\}]$. In prior work [44], pairwise pmfs involving the class label and individual discrete features were encoded into the learning of ME classifiers and this was found to achieve good accuracy. In this case, the constraint probability is measured by:

$$P_g[C = c, \hat{C}_i = \hat{c}] = \sum_{t=1}^{T} \delta(c^{(t)} = c) \times P[\hat{C}_i = \hat{c}|\underline{a}_i^{(t)}]$$

and the model's estimate is:

$$P_m[C = c, \hat{C}_i = \hat{c}] = \sum_{t=1}^{T} P[C = c|\{\underline{a}_i^{(t)}\}] \times P[\hat{C}_i = \hat{c}|\underline{a}_i^{(t)}],$$

where $P[C = c|\{\underline{a}_i^{(t)}\}]$ is the ME *a posteriori* pmf. Following our ME–IIS formulation, one finds that the ME combining rule now takes the form:

$$P[C = c|\{\underline{a}_i\}] = \frac{e^{\sum_{i=1}^{N_e} \sum_{\hat{c}=1}^{K} P[\hat{C}_i = \hat{c}|\underline{A}_i = \underline{a}_i]\gamma(C=c, \hat{C}_i = \hat{c})}}{Z}, \tag{3.1}$$

[2] Discrete and mixed discrete-continuous feature spaces can also be handled. Restriction here to purely continuous features is simply for clarity, without loss of generality.

where Z is the proper normalization, ensuring a valid pmf. The Lagrange multipliers are again learned via the IIS approach, using the following update rule:

$$\gamma(C = c, \hat{C}_i = \hat{c}, M_i = n) = \frac{1}{N_d} \ln \left[\frac{P_g[C = c, \hat{C}_i = \hat{c}]}{P_m[C = c, \hat{C}_i = \hat{c}]} \right] \tag{3.2}$$

We call this method as simple ME (SiME).

3.2.2 ME-IIS for Input-Dependent Classifier Combining

In previous subsection we encoded the pairwise pmfs $P[C, \hat{C}_i],\ i = 1, .., N_e$. However, we suggest to go further. As in the last chapter, we can learn a mixture model for each classifier's *input space* $\underline{A}_i$, yielding the latent variables $M_1, M_2, \ldots, M_{N_e}$ and associated probabilities $P[M_i = j|\underline{A}_i = \underline{a}_i^{(t)}; \Lambda_i]\ \forall i\ \forall j,\ t = 1, \ldots, T$ [3]. We can devise an approach that assimilates *both* of these probabilistic information sources while learning an ME model. In particular, for each classifier in the ensemble, suppose we encode the *ternary* pmfs[4], i.e. $P[C, \hat{C}_i, M_i],\ i = 1, \ldots, N_e$. The constraint probabilities and model's estimates depend on both probabilistic information sources, i.e. $\{P[\hat{C}_i = c|\cdot]\}$ and $\{P[M_i = n|\cdot]\}$. In this case, the constraint probability is measured by:

$$P_g[C = c, \hat{C}_i = \hat{c}, M_i = j] = \sum_{t=1}^{T} \delta(c^{(t)} = c) \times P[\hat{C}_i = \hat{c}|\underline{a}_i^{(t)}] P[M_i = j|\underline{a}_i^{(t)}]$$

[3] Here, as one example, we are considering the case of a (single) mixture model for each vector $\underline{A}_i$.

[4] Higher order constraints, which encode dependencies between base classifiers, are also possible. However they would entail greater complexity, and a large training set to accurately measure the constraints .

and the model's estimate is:

$$P_m[C=c,\hat{C}_i=\hat{c},M_i=j]=\sum_{t=1}^{T}P[C=c|\{\underline{a}_i^{(t)}\}]\times P[\hat{C}_i=\hat{c}|\underline{a}_i^{(t)}]P[M_i=j|\underline{a}_i^{(t)}],$$

where $P[C=c|\{\underline{a}_i^{(t)}\}]$ is the ME *a posteriori* pmf. Following our ME–IIS formulation, one finds that the ME combining rule now takes the form:

$$P[C=c|\{\underline{a}_i\}]=\frac{e^{\sum_{i=1}^{N_e}\sum_{\hat{c}=1}^{K}\sum_{n=1}^{L_i}P[\hat{C}_i=\hat{c}|\underline{A}_i=\underline{a}_i]P[M_i=n|\underline{A}_i=\underline{a}_i]\gamma(C=c,\hat{C}_i=\hat{c},M_i=n)}}{Z}, \tag{3.3}$$

where Z is the proper normalization, ensuring a valid pmf. The Lagrange multipliers are again learned via the IIS approach, using the following update rule:

$$\gamma(C=c,\hat{C}_i=\hat{c},M_i=n)=\frac{1}{N_d}\ln\left[\frac{P_g[C=c,\hat{C}_i=\hat{c},M_i=j]}{P_m[C=c,\hat{C}_i=\hat{c},M_i=n]}\right] \tag{3.4}$$

This ME approach combines both *decision-level* ($\{P[\hat{C}_i=c|\cdot]\}$) and (soft-quantized) *feature-level* ($\{P[M_i=n|\underline{A}_i]\}$ fusion within the same rule of combination and does so in a "discriminative" fashion, to maximize the class *a posteriori* data likelihood. This approach should have advantages over standard boosting methods [33], which do not have feature-selective confidence weights on base classifier decisions. Moreover, while more recent boosting methods do provide an input-dependent capability, the results in the next section will demonstrate a performance advantage for the method proposed here. This approach is referred to as Latent variable ME (LvME).

Relation to Adaboost and Logistic Regression:

[22] gives a connection between Adaboost and logistic regression (i.e., maximum likelihood for exponential models) – the latter requires the model to be normalized to form a pmf. [22] shows that these two approaches optimize the same KL divergence, subject to the same feature constraints. Our approach (SiME) can be related to [22] as follows. The model in [22] takes the form

$$P[C=1|\underline{a}^{(t)}] = \frac{e^{\sum_{j=1}^{N} \gamma(C=1,C_j=1)P[C_j=1|\underline{a}^{(t)}]}}{e^{\sum_{j=1}^{N} \gamma(C=1,C_j=1)P[C_j=1|\underline{a}^{(t)}]} + e^{\sum_{j=1}^{N} \gamma(C=0,C_j=0)P[C_j=0|\underline{a}^{(t)}]}}. \tag{3.5}$$

Note that equation (3.5) is consistent with *solely* encoding the constraint probabilities $P[C = c, C_j = c] \ \forall c$. By contrast, in our approach (SiME), in addition to encoding $P[C = c, C_j = c]$, we *additionally* encode $P[C = c, C_j = \hat{c}] \ \forall c \neq \hat{c}$, which we believe also contains valuable information for classification. The performance gain from encoding these additional constraints is demonstrated in section 3.4.

3.3 Hidden Markov Model for Ensemble Classification (Boosting):

Hidden Markov Model(HMM) is a powerful stochastic tool of modeling and identifying sequential signals. They are used in almost all speech recognition systems and in other areas of pattern recognition. The basic elements of HMM[5] are

1. The number of states in the model (N_e).

2. The number of distinct observation symbol per state (N_c).

[5]The following formulation is adapted from [45]

3. The state transition probabilities, $(A = \{a_{i/j}\})$. Where $a_{i/j} = P[q_{t+1} = S_j | q_t = S_i]$ i.e. probability of being in state S_j at time q_{t+1} given that at time q_t it was in state S_i.

4. The observation symbol probability distribution in state j, $\left\{P[\hat{C}_i = j|\underline{f}_t, C = c]\right\}$. These are obtained from local classifiers.

5. Initial state distribution, $\pi = \{\pi_i\}$, $\pi_i = P[q_i = S_i]$ $i = 1, ..., N_e$. For our method we take initial state distribution to be uniform.

The Adaboost algorithm successively builds stage classifiers. These can be visualized as the stages of a HMM. Each of these classifiers make hard/soft predictions. This can be treated as observation for each state. Figure 3.1 shows a class conditional HMM for ensemble combining. To build a ensemble classifier using HMM we propose to learn a HMM for each class. On the basis of training data we learn the parameters $(\lambda_i = (A, P, \pi)$ $i = 1, ..., N_c)$ for the model. In our simulations we considered probabilistic outputs, but this method could be well extended to include hard decisions from the base classifiers. A observation O is therefore represented by a sequence of probabilistic outputs,

$$O = \mathrm{P}_1, \mathrm{P}_2, \cdots, \mathrm{P}_{N_e}$$

where $\mathrm{P}_i = \left\{P[C_i = j|\underline{f}_t, C = c']\right\}$, $j = 1, ..., N_c$. The basic procedure for implementing a hidden markov model for ensemble combining is as follows.

1. Initialization

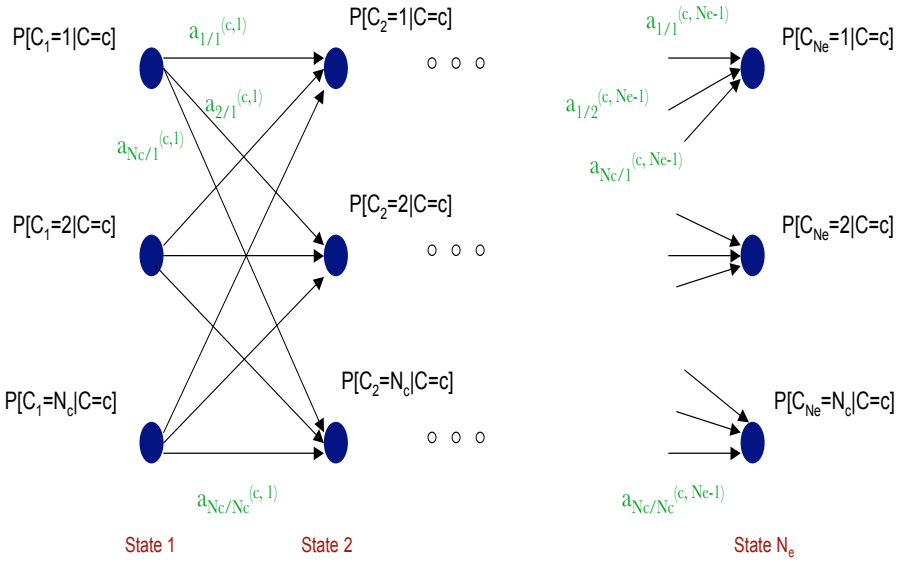

Fig. 3.1. Hidden Markov model for ensemble combining.

- The base classifiers are built sequentially using Adaboost algorithm. We used the Naive Bayes as base classifier to obtain probabilistic output.
- Describe the elements of HMM.

2. Training the model for each class
   - Learn model parameters $\lambda_i = (A, P, \pi)$ $i = 1, ..., N_c$ which maximize $P_i[O|\lambda_i]$.
3. Evaluating generalization performance
   - Given an observation sequence (posterior probability obtained on test sample from base classifiers), compute $P_i[O|\lambda_i]$, $i = 1, ..., N_c$. Select the class with maximum value $P_i[O|\lambda_i]$.

The question that arises now is how do we compute steps 2 and step 3 ? [45] gives an efficient method to solve this question. $P_i[O|\lambda_i]$ are efficiently computed by forward and backward procedure. These procedure involve computing forward and backward variables. For our implementation these procedures are

Forward procedure :

$$\alpha_1^{(c,t)}(i) = \frac{1}{N_c} P[\hat{C}_1 = i|\underline{f}_t], \quad i = 1, \cdots, N_c$$

$$\alpha_{k+1}^{(c,t)}(i) = (\sum_{i=1}^{N_c} \alpha_k^{(c,t)}(i) \cdot a_{j/i}^{(c,t)}) \cdot P[\hat{C}_{k+1} = j|\underline{f}_t], \quad \begin{cases} k = 1, \cdots, N_e - 1 \\ j = 1, \cdots, N_c \\ t = 1, \cdots, N_{train} : c^{(t)} = c \end{cases} \tag{3.6}$$

Backward procedure :

$$\beta_1^{(c,t)}(i) = 1, \quad i = 1, \cdots, N_c.$$

$$\beta_{k+1}^{(c,t)}(i) = (\sum_{j=1}^{N_c} a_{j/i}^{(c,k)} \cdot P[\hat{C}_{k+1} = j|\underline{f}_t] \cdot \beta_{k+1}^{(c,t)}(j)), \quad \begin{cases} i = 1, \cdots, N_c \\ k = N_e - 1, \cdots, 1 \\ t = 1, \cdots, N_{train} : c^{(t)} = c \end{cases} \tag{3.7}$$

To train the model we used Baum-Welch method (also known as EM method). The details of this method can be looked in [45]. This is an iterative procedure which selects the parameters which locally maximize $P_i[O|\lambda_i]$. In the *expectation* step, we compute $\xi_k^{(c,t)}(i,j)$ (the probability of being in state i at time t and in state j at time $t+1$) and $\gamma_k^{(c,t)}(i)$ (the probability of being in state i at time t, given the entire observation). These are computed as

$$\xi_k^{(c,t)}(i,j) = \frac{\tilde{\alpha}_k^{(c,t)}(i) \cdot a_{j/i}^{(c,k)} \cdot P[\hat{C}_{k+1} = j|\underline{f}_t] \cdot \beta_{k+1}^{(c,t)}(j)}{\sum_{i'=1}^{N_c} \sum_{j'=1}^{N_c} \tilde{\alpha}_k^{(c,t)}(i') \cdot a_{j'/i'}^{(c,k)} \cdot P[\hat{C}_{k+1} = j'|\underline{f}_t] \cdot \beta_{k+1}^{(c,t)}(j')} \tag{3.8}$$

$$\gamma_k^{(c,t)}(i) = \frac{\tilde{\alpha}_k^{(c,t)}(i) \cdot \tilde{\beta}_k^{(c,t)}(i)}{\sum_{i'} \tilde{\alpha}_k^{(c,t)}(i') \cdot \tilde{\beta}_k^{(c,t)}(i')} \tag{3.9}$$

The *maximization* step for the algorithm updates the $\{a_{i/j}\}$ parameters. This is done as

$$a_{j/i}^{(c,k)} = \frac{\sum_{t=1:c^{(t)}=c}^{N_{train}} \xi_k^{(c,t)}(i,j)}{\sum_{t=1:c^{(t)}=c}^{N_{train}} \gamma_k^{(c,t)}(i)} \tag{3.10}$$

To stop the learning procedure we implemented a fixed stopping criteria. This criteria was chosen empirically. We stopped learning when the relative improvement in the learning objective function drops below 1e-4. This novel approach is experimentally evaluated with Simple ME boosting in Section 3.4.2.

3.4 Experiments

In this section, we compare our ME–IIS method with other input dependent methods and with other (MLP & SVM) supervised combining rules. We used 11 datasets from the UC Irvine machine learning repository.

3.4.1 Comparison of Boosting with ME-IIS Ensemble Classification

In this subsection we compare the ME-IIS methods with Adaboost.M1, input dependent boosting with a regularizer[15], MLPs, and SVMs. For these experiments we performed 10 trials of 10 fold cross validation on all datasets except *satellite*, where the training/test (4435/2000) split was given. The latent variables and the associated probabilities for LvME were obtained by learning a class–dependent mixture.

ME-IIS Ensemble Combining Compared with Adaboost, MLPs, and Linear SVMs

Table 3.3. Ensemble classification performance of supervised ensemble learning, SiME and LvME for varying ensemble size.

Dataset	**Method**	10	20	50	70	100
Diabetes	Ada.	24.04	23.28	23.59	24.04	23.59
	MLP	23.41	23.01	23.79	24.05	24.43
	SVM	22.08	22.68	22.42	23.17	22.56
	SiME	22.42	22.75	22.38	22.83	22.45
	LvME	21.99	22.10	22.18	22.29	22.21
Liver Disorder	Ada.	37.33	34.63	32.28	32.10	32.55
	MLP	31.63	29.69	29.50	30.27	29.96
	SVM	33.94	29.30	28.47	29.55	28.46
	SiME	31.55	30.15	30.61	30.10	29.65
	LvME	31.04	29.61	29.29	29.76	29.42
Sonar	Ada.	21.25	19.91	17.49	17.69	16.80
	MLP	19.32	18.39	17.44	16.74	15.80
	SVM	19.73	18.44	16.67	16.39	16.02
	SiME	19.18	18.12	16.46	16.26	15.31
	LvME	18.61	17.17	15.31	15.38	14.85
Ionosphere	Ada.	8.03	8.01	7.16	6.87	7.27
	MLP	6.76	7.39	7.60	7.55	7.15
	SVM	6.48	7.05	7.10	7.04	7.56
	SiME	7.56	7.04	7.16	6.82	6.72
	LvME	6.36	6.13	5.22	5.05	5.27
Satellite	Ada.	20.45	20.25	20.45	20.60	20.20
	MLP	16.05	15.85	16.10	15.60	15.65
	SiME	15.50	15.75	15.25	15.85	15.70
	LvME	11.70	11.55	11.25	11.50	11.40
Wine	Ada.	2.78	2.22	3.33	2.78	2.22
	MLP	3.33	1.67	2.78	2.22	2.22
	SiME	2.90	2.78	2.22	2.22	2.78
	LvME	1.78	1.67	1.11	1.67	1.67
Segment	Ada.	23.42	23.46	20.63	19.86	19.96
	MLP	11.20	10.17	9.37	10.22	9.95
	SiME	11.11	10.50	9.20	10.45	9.45
	LvME	5.15	5.28	5.35	4.85	4.98

Here we present the comparison of ME–IIS combining with Adaboost, MLPs, and linear SVMs. Naive Bayes multivariate Gaussian classifiers were used as base classifiers. These base classifiers (for all the methods) were built using the Adaboost.M1 algorithm. Each stage classifier outputs *a posteriori* class probabilities, based on the naive Bayes model. The class *a posteriori* probabilities on the training set, thus obtained, were used to learn the ME-IIS models. The models were then evaluated making ensemble maximum *a posteriori* decisions on the generalization set. [42] describes a method for linearly combining discriminant–based classifiers. Instead of combining the discriminant functions, the SVM and MLP methods we implemented combine the *a posteriori* probabilities $\left\{P[\hat{C}_i = \hat{c}|\underline{a}^{(t)}] \;\; c = 1, ..., N_c \;\; i = 1, ..., N_e\right\}$. The input space dimensionality for MLPs and linear SVMs is thus $N_c \times N_e$(N_c is the number of classes and N_e is the ensemble size); moreover each feature value (a probability) $\in [0, 1]$. The number of output units in the MLP was N_c. For each dataset the number of hidden units was varied from 8 to 25 and the number which gave us best test set classification accuracy was selected as the number of hidden units[6]. Linear SVMs were also tuned for best classification accuracy on each dataset. The MLPs were trained via backpropagation to minimize the squared error to the binary vector [00..100..0], with a 1 in the position of the true class for the training sample. SVMs were trained using the SVM toolbox [46]. Table 3.3 gives the generalization error rates for Adaboost.M1, MLPs, linear SVMs, SiME and LvME on UC Irvine datasets for varying ensemble sizes. Both ME-IIS methods outperform Adaboost.M1.

[6]This search was done with N_e fixed at 10. The selected number of hidden units was then used for all ensemble sizes.

The latent variable ME approach gives best classification results for all datasets for varying population sizes except on *liver disorder* where linear SVMs gave better accuracy. Simple ME boosting is competitive with MLPs and linear SVMs, most of the time giving lower generalization error. The performance gain of SiME over standard Boosting method demonstrates that encoding the full set of pairwise pmfs($\{P[C, C_i]\}$) instead of just the probabilities ($\{P[C = c, C_i = c]\}$) does improve classification accuracy.

Table 3.4. Ensemble classification performance of input dependent boosting, SiME and LvME for varying ensemble size.

Dataset	10	20	50	70	100
Diabetes					
IdB	23.03	22.65	23.20	22.65	22.80
SiMe	22.81	22.76	22.79	22.75	22.71
LvMe	22.49	22.22	22.86	22.48	21.90
Liver Disorder					
IdB	36.79	34.13	32.41	32.41	31.91
SiMe	31.15	31.58	31.49	32.06	31.70
LvMe	30.39	30.56	29.99	30.02	29.93
Sonar					
IdB	21.09	17.95	16.98	16.38	15.52
SiMe	19.46	19.36	16.48	14.41	15.80
LvMe	18.18	16.66	14.93	13.94	14.60
Ionosphere					
IdB	7.22	7.20	7.65	7.50	6.64
SiMe	6.82	6.59	6.25	6.87	6.53
LvMe	6.08	5.33	5.97	5.33	5.85
Satellite					
IdB	20.30	20.20	20.20	20.15	19.80
SiMe	16.55	15.85	15.70	15.30	15.25
LvMe	11.30	11.75	11.55	11.60	11.35
Segment					
IdB	22.85	20.43	18.14	19.75	18.94
SiMe	13.87	11.97	12.55	10.74	9.56
LvMe	6.00	5.74	5.89	5.32	4.81

ME-IIS Ensemble Combining Compared with Input Dependent Boosting

Boosting using an input dependent regularizer(IdB)[15] is similar to Adaboost except for the weight updating function. Here an additional term("input-dependent factor"),β,

is introduced in the weight update equation. This additional term gives special importance to the misclassified samples that are close to the decision boundary. ME-IIS ensemble learning does not impose any constraints on how the stage/base classifiers are constructed. It optimizes the combining rule based on the set of given constraints. Thus, to focus the comparison on the combining method, we used IdB boosting to construct the stage classifiers that were used *both* in IdB combining and in our ME–IIS combining. β was chosen experimentally to give the best performance for IdB, for each dataset. Table 3.4 gives the test set classification error rates for boosting using IdB and ME-IIS ensemble combining methods. Both the ME-IIS methods outperform boosting using the input dependent regularizer.

Statistical Tests

To assess statistical significance of the comparisons, we used the 5x2 cv F test [47]. The hypothesis that two algorithms have the same error rate can be rejected with 95% confidence if the statistic f is greater than 4.74. For this test we set the number of base classifiers to 100. In Table 3.5, columns 2 to 5 give values of f for comparisons between SiME and 1) adaboost, 2) IdB, 3) MLP and 4) SVMs. Similarly, the columns 6 to 9

Table 3.5. 5x2 F cross validation test for supervised ensemble learning.

	5x2 F cv test							
	Simple ME				**Latent Variable ME**			
Dataset	**Adaboost**	**Idb**	**MLP**	**SVM**	**Adaboost**	**Idb**	**MLP**	**SVM**
Diabetes	3.43	2.91	3.87	0.61	5.63	4.24	10.80	1.65
Sonar	2.53	0.36	0.85	0.87	4.84	3.84	2.69	3.40
Ionosphere	2.91	0.58	0.76	1.51	5.10	4.93	1.58	5.27
Segment	6.66	5.96	0.75	-	9.11	8.56	3.37	-

give the values of f for comparison between LvME and 1) adaboost, 2) IdB, 3) MLP and 4) SVMs. Latent variable ME gives statistically significant improvement over Adaboost for all datasets. It also gives statistically significant improvement over the IdB on all datasets except *sonar*. The simple ME boosting does not give statistically significant improvement over Adaboost or the IdB except on *segment*.

Computational Time

The experiments were run on an Intel Xeon 2.0 Ghz processor with 1 Gb of RAM. Table 3.6 gives the time taken by each algorithm to finish one trial of 10 fold cross validation. The time taken to learn a Gaussian mixture model in each fold (for the latent variable ME approach) is also included in the total time.

Table 3.6. Time in minutes for single trial.

	Ensemble classifiers				
Dataset	Adaboost	MLP	SVM	SiMe	LvME
Liver Disorder	0.24	6.15	0.03	1.40	10.13
Diabetes	0.36	9.58	0.07	1.11	6.3

3.4.2 Comparison of HMMboost with ME-IIS Ensemble Classification

In this section we present the comparison between HMM for ensemble classification and Simple ME boosting. For this purpose, we used 4 datasets from UCI machine learning repository. All the datasets contain two classes. The HMM based ensemble classification system does not fare well in comparison with SiME boosting. A possible reason could be lack of sufficient training data.

Table 3.7. Comparison between HMMboost and SiME boosting.

Dataset	**Simple ME Boosting**	**HMM boosting**
Diabetes	22.71	26.57
Sonar	29.86	38.10
Ionosphere	7.10	7.68
Liver Disorder	18.71	20.73

Chapter 4

Transductive Methods for Ensemble Classification

In this chapter we focus on ensemble techniques that do not involve *any* training of the decision aggregation rule. This category, without training optimization of the combiner, is often applied *of necessity*. There are two primary scenarios for which such training appears to be precluded:

1. *Legacy or proprietary classifiers*: different organizations may have built classifiers, each using "in-house" data. Even if organizations are willing to share data, each classifier may use a different feature space[1]. In this case, from company A's data (which was collected with a particular feature space in mind), it may not be possible to extract instances of the input features used by company B's classifier. Thus, in general it will not be possible to form a common pool of labeled data, as required for supervised combiner learning.

2. *Distributed multisensor/multimedia classification*: spatially remote sensors may collect measurements at different times, using either the same or different sensing modalities [48]. Each sensor may form its own local training set, without any known correspondence e.g. between labeled examples in sensor A's database and those in sensor B's. In fact, the two sensors, while taking measurements of objects from the same set of classes, may not have sampled *any* common examples. Even if there are common examples, solving

[1]There may be *some* features in common. Even in this case, we would say the classifiers use different feature spaces.

the correspondence problem to identify these examples appears daunting unless the data comes with time stamps. A related application is e.g. vowel recognition based on speech and mouth images, with separate training sets.

In the above examples there is no common labeled data across the ensemble, needed for supervised learning of the combining rule. This problem is dubbed as *distributed ensemble classification*. There is very little prior work done in this area [17], [19]. We emphasize that [17] and [19] apply *fixed* combining rules. The conventional wisdom is that, without common labeled training data, one must apply a fixed combining strategy. Alternatively, we will show it is both possible and, in general, desirable to *learn* the combining rule. We propose a *transductive learning* strategy [49], [50], [20] wherein an objective measured over the test data batch is optimized with respect to the batch of class decisions [2]. Several such objective functions will be proposed.

There are two main reasons why we expect transductive learning may outperform fixed combining. First, consider the very practical case where the class priors seen (in training data) by individual classifiers do not reflect the true class priors, *i.e.*, those seen in the test batch. There are a number of possible reasons for such mismatch:

- For the distributed multisensor domain, the class distribution may be spatially varying, e.g. each local region (and hence each local classifier) may only "see" examples from a subset of the classes.

[2]Batch transductive learning jointly yields decisions for all samples in the test batch. Alternatively, to avoid sample (but not processing) delays in decisionmaking, we can instead apply transductive learning *sample by sample*. In this case, the decision on the current sample $\underline{\mathbf{x}}_t$ is made (at least in principle) by transductive learning using the entire batch of test samples seen until now, i.e. $\{\underline{\mathbf{x}}_1, \underline{\mathbf{x}}_2, \ldots, \underline{\mathbf{x}}_{t-1}\}$. While this learning will yield decisions on *all* samples in the current batch, only the decision on $\underline{\mathbf{x}}_t$ is retained, since decisions on past samples have already been made in this (sample by sample) case.

- Rare classes may not be represented proportionately in a training data set.
- Class priors may be environment-dependent.
- Groups of classes that are highly confusable are difficult to hand-label. Thus, ironically, although it is important to have sufficient examples from them, these classes may be underrepresented in a labeled set.

A second motivation for a transductive approach is to avoid unnecessary statistical assumptions *implicitly* made by using a particular combining rule. We will show that arithmetic averaging and naive Bayes rules emerge as the transductive ML-optimal rules when particular assumptions – conditional independence, in the naive Bayes case – are made about the underlying stochastic model which generated the data. As will be seen in our results, conditional independence in particular is a poor assumption when there is statistical redundancy between the decisions produced by individual classifiers. To avoid making unnecessary assumptions, we also propose an *information-theoretic* transductive technique for aggregation of distributed ensembles. We have found that this method generally outperforms the ML techniques and, in particular, much better accounts for redundancy between classifiers than the ML methods. This method is also applicable for some scenarios where transductive ML techniques cannot be used.

4.1 Assumptions for Distributed Ensemble Classification

Consider an ensemble with N_e classifiers and an aggregation function for combining their decisions. The transductive ML techniques we develop assume each classifier estimates *a posteriori* probabilities $P_j[C = c|\underline{x}^{(j)}], c = 1, \ldots, N_c,\ j = 1, \ldots, N_e,\ N_c$ the

number of classes and $\underline{x}^{(j)}$ the feature vector for the jth classifier. Each classifier has its own labeled training set for building an *a posteriori* model. The priors reflected in this training set will in general be different for each local classifier and, of course, may differ from the true priors reflected in the test data batch, which are *unknown*. Consistent e.g. with the proprietary scenario, we assume there is no communication between the individual classifiers for discerning common features. As emphasized before, there is no common labeled training data that could be used for supervised learning of the aggregation function. However, during the operational (use) phase, common data *is* observed (in a synchronized fashion) across the ensemble, i.e. for each test object, a corresponding feature vector is measured by each classifier. Thus, the input to the ensemble system is the concatenated vector $\underline{\mathbf{x}} = (\underline{x}^{(1)}, \underline{x}^{(2)}, \ldots, \underline{x}^{(N_e)})$, but with classifier j only seeing $\underline{x}^{(j)}$. We suppose it is not feasible to directly convey the feature vectors $\underline{x}^{(j)}$ to the aggregation function – only the classifier decisions (hard or soft) are conveyed. This is consistent with the proprietary scenario and also with the case of high feature dimensionality (and hence high bandwidth transmission cost). Each classifier may either communicate soft decisions $\{P_j[C = c|\underline{x}^{(j)}], c = 1, \ldots, N_c\}$ or maximum *a posteriori* (MAP) hard decisions.

A key aspect of our approach is that, in making decisions, we perform transductive learning on a *batch* of test samples. As aforementioned, one possibility is to make joint decisions on all samples in a given test set batch, $\mathcal{X}_{\text{test}} = \{\underline{\mathbf{x}}_i, i = 1, \ldots, N_{\text{test}}\}$. This will entail collecting batches of soft (or hard) decisions from each of the individual classifiers, i.e. $\{\{P_j[c|\underline{x}_i^{(j)}]\forall c\ \forall j\}, i = 1, \ldots, N_{\text{test}}\}$. There will be delays in decisionmaking associated both with decision collection and with processing by the aggregation function to produce the ensemble decision batch $\{\{P_e[c|\underline{\mathbf{x}}_i]\}, i = 1, \ldots, N_{\text{test}}\}$. For our methods,

these delays will grow linearly with N_{test}. Alternatively, in the aforementioned *sample-by-sample* case (see Footnote 2), there is no delay associated with data (i.e. decision) collection.

Finally, we note an important distinction between the ML methods and the information-theoretic method to be developed, for the case where individual classifiers forward *hard* decisions. For the ML methods, each local classifier must explicitly produce *a posteriori* probabilities $\{P_j[c|\underline{x}^{(j)}]\}$, even though only a hard MAP decision will be conveyed. This is required because local MAP decisions cannot be explicitly corrected to account for new estimated priors unless there is access, at least at the local classifiers, if not at the aggregation function, to local *a posteriori* probabilities. By contrast, as will be seen, the information-theoretic method can be applied and can effectively correct priors even if each local classifier is an inaccessible "black box" that only produces hard decisions.

4.2 Transductive Maximum Likelihood Ensembles

[20] considered the case of a single *a posteriori* model $P[c|\underline{x}]$. They effectively assumed this model is reasonably accurate except possibly in the class priors implicitly used. They proposed to re-estimate class priors so as to maximize the data likelihood measured on the test set batch. Within the Expectation-Maximization (EM) learning framework [51], [52], the revised *a posteriori* class probabilities, based on the current estimates of the class priors, are computed in the E-step, with the class priors re-estimated in the M-step. Here, we extend their approach to address distributed ensembles. We

consider two underlying stochastic models, each of which will be shown to be consistent with a particular, well-known form of ensemble combining.

4.2.1 Class-conditional Feature Independence

We denote the class-conditional density (or probability mass function (pmf), in the discrete case) for the random feature vector $\underline{X}^{(j)}$, given $C = c$, by $p_j[\underline{x}^{(j)}|C = c]$. Since each local classifier is only required to produce an *a posteriori* model $P_j[c|\underline{x}^{(j)}]$, the density function will in general be unspecified[3]. Moreover, even if each classifier is explicitly using a class-conditional density for its feature vector, we are not proposing, as in e.g. [50], to re-estimate the parameters of these densities within the transductive learning framework. Regardless, the class-conditional densities do appear in our maximum likelihood formulation.

In this subsection, we assume the local feature vectors $\{\underline{X}^{(j)}, j = 1, \ldots, N_e\}$ are conditionally independent, given the class. This is a standard *naive Bayes* assumption that is often applied in distributed classification/detection contexts (Varshney, 1997). The associated stochastic data generation for each test sample $\underline{\mathbf{x}}_i$, assumed to be independently generated, is to first randomly select a class based on $\{P_e[C = k]\}$ and then to randomly generate each local feature vector, given the chosen class k', via the density $p_j[\underline{x}_i^{(j)}|C = k']\forall j$. This stochastic generation defines a special *mixture model* for generating $\underline{\mathbf{x}}$, with mixture component k representing class k, $k = 1, \ldots, N_c$. Thus, $P_e[C = k]$ is the prior for component(class) k, with $\{p_j[\underline{x}_i^{(j)}|C = k], j = 1, \ldots, N_e\}$ the

[3]It will be known if the *a posteriori* model is derived from an underlying stochastic model for the feature vector.

component model for component(class) k. For this mixture model, applying Bayes rule, the transductive incomplete data log likelihood for the unlabeled batch test set, $\mathcal{X}_{\text{test}}$, is

$$\begin{aligned}
\log L(\mathcal{X}_{\text{test}}) &= \log L(\underline{\mathbf{x}}_1, \ldots, \underline{\mathbf{x}}_{N_{test}}) \qquad (4.1)\\
&= \sum_{i=1}^{N_{\text{test}}} \log(\sum_{k=1}^{N_c} P_e[C=k] \prod_{j=1}^{N_e} p_j[\underline{x}_i^{(j)}|C=k])\\
&= \sum_{i=1}^{N_{\text{test}}} \log(\sum_{k=1}^{N_c} P_e[C=k] \prod_{j=1}^{N_e} P_j[C=k|\underline{x}_i^{(j)}] \cdot \frac{p_j[\underline{x}_i^{(j)}]}{P_j[C=k]})\\
&= \sum_{i=1}^{N_{\text{test}}} \log(\prod_{j=1}^{N_e} p_j[\underline{x}_i^{(j)}]) + \sum_{i=1}^{N_{\text{test}}} \log(\sum_{k=1}^{N_c} P_e[C=k] \prod_{j=1}^{N_e} \frac{P_j[C=k|\underline{x}_i^{(j)}]}{P_j[C=k]}),
\end{aligned}$$

with the final form obtained using the fact that $\prod_{j=1}^{N_e} p_j[\underline{x}_i^{(j)}]$ has no dependence on k. This is to be maximized over the *ensemble* class priors $\{P_e[C=k]\}$. Here, $P_j[C=k|\underline{x}_i^{(j)}]$ is local classifier j's posterior, $P_j[C=k]$ is its prior (measured based on its training set), and $p_j[\underline{x}_i^{(j)}]$ is its feature vector density. Note that $p_j[\underline{x}_i^{(j)}]$ is in general unknown since each classifier is only assumed to produce class posteriors. However, this is not problematic since $\sum_{i=1}^{N_{\text{test}}} \log(\prod_{j=1}^{N_e} p_j[\underline{x}_i^{(j)}])$ has no dependence on the parameters to be optimized. The optimization is accomplished via an EM algorithm whose defining formulation details are given in the Appendix B. The algorithm amounts to a sequence of iterations, each consisting of 1) re-estimation of the ensemble class posteriors on the test data, given fixed ensemble priors (E-step) followed by 2) re-estimation of the ensemble class priors, given fixed posteriors (M-step), applied until a convergence condition is met. Each step in the optimization is guaranteed to ascend in $\log L$. The E-step at iteration

t is:

$$P_e^{(t)}[C=k|\underline{\mathbf{x}}_i] = \frac{P_e^{(t-1)}[C=k]\prod_{j=1}^{N_e}(\frac{P_j[C=k|\underline{x}_i^{(j)}]}{P_j[C=k]})}{\sum_{l=1}^{N_c} P_e^{(t-1)}[C=l]\prod_{j=1}^{N_e}(\frac{P_j[C=l|\underline{x}_i^{(j)}]}{P_j[C=l]})} \tag{4.2}$$

and the M-step is:

$$P_e^{(t)}[C=k] = \frac{1}{N_{\text{test}}}\sum_{i=1}^{N_{\text{test}}} P_e^{(t)}[C=k|\underline{\mathbf{x}}_i], \tag{4.3}$$

with the initial prior probabilities given as $\{P_e^{(0)}[C=k]\}$. We emphasize that in the E-step, $P_j[C=k]$ and $P_j[C=k|\underline{x}_i^{(j)}]$, the prior and posterior for the j-th classifier, remain unchanged throughout the optimization. Note that the ratio $\frac{P_e^{(t-1)}[C=k]}{P_j[C=k]}$ is effecting correction of the posterior $P_j[C=k|\underline{x}_i^{(j)}]$ to reflect the current estimates of the ensemble class priors measured on the test batch. Note also that the ratio $\frac{P_j[C=k|\underline{x}_i^{(j)}]}{P_j[C=k]}$ is proportional to the class-conditional density $p_j[\underline{x}_i^{(j)}|C=k]$. Thus, as it must, (4.2) takes the *naive Bayes* posterior form, *i.e.* the form of a model that assumes class-conditional independence of the feature vectors $\{\underline{X}^{(j)}, j=1,\ldots,N_e\}$, with class priors given by the transductive ML-estimated ones. At convergence, the E-step is used to compute the ensemble class *a posteriori* probabilities. Note that the difference between our approach and *standard* naive Bayes aggregation is that our method learns and, at convergence, *uses* corrected priors in the naive Bayes rule, as given in (4.2). In fact, the solution after the *first* E-step (without any prior correction) *is* the standard naive Bayes solution, which

we will refer to as the "fixed product rule" and compare against in our experimental results.

It is important to observe that the objective function (4.1), as well as the objective in [20], is concave in the parameters $\{P_e[C = k]\}$, with, thus, a unique maximum. Moreover, convergence for this particular problem – ML estimation for mixture proportions, with the parameters of the component densities held fixed – is addressed in [53]. There it is shown that, under mild assumptions (nonzero initialization of mixing proportions for all components, positivity of the mixture density for each sample, linearly independent mixture components with positive support on the sample domain), the EM algorithm is globally convergent, to the unique maximum likelihood estimate. Only local optimality was recognized in [20].

Our method requires each local classifier to convey its test batch posterior pmfs to the aggregation function. Moreover, each classifier must also convey its class priors $\{P_j[C = k]\}$.

4.2.2 Censored Feature Vectors

A well-known drawback of the *product-based* rule (4.2) is its sensitivity to unreliable experts – a single classifier that gives low probability to the true class may cause an ensemble error. Arithmetic averaging is generally believed to be more robust to unreliable "experts". In deriving the rule (4.2) within the transductive ML framework, we modeled stochastic generation of $\underline{\mathbf{x}}$ in a standard way, using conditional independence. Alternatively, in this section we show that an arithmetic averaging rule emerges when a somewhat unconventional stochastic model is assumed. To wit, imagine that, for any

given test set object, only *one* of the individual classifiers receives measurements, with the other feature vectors essentially *censored*. Each classifier is assumed equally likely to be uncensored. More precisely, the stochastic data generation is as follows. For each test sample, one first randomly selects a class based on $\{P_e[C = k]\}$, then the uncensored classifier (based on a uniform pmf over the N_e classifiers) and finally the sample, given the chosen class (k') and classifier (j'), *i.e.* via $p_{j'}[\underline{x}_i^{(j')}|C = k']$. For this model, the transductive *censored* incomplete data log-likelihood, assuming independent test set samples, is

$$\log L_{\text{cens}}(\mathcal{X}_{\text{test}}) = \sum_{i=1}^{N_{\text{test}}} \log \sum_{k=1}^{N_c} P_e[C = k](\frac{1}{N_e} \sum_{j=1}^{N_e} p_j[\underline{x}_i^{(j)}|C = k]). \tag{4.4}$$

The Appendix casts this ML problem within the EM framework. The ensemble *a posteriori* class probabilities are again re-estimated in the E-step, which now takes the *arithmetic averaging* form:

$$P_e^{(t)}[C = k|\underline{\mathbf{x}}_i] = \frac{P_e^{(t-1)}[C = k]\frac{1}{N_e}\sum\limits_{j=1}^{N_e}(\frac{P_j[C=k|\underline{x}_i^{(j)}]}{P_j[C=k]})}{\sum\limits_{l=1}^{N_c} P_e^{(t-1)}[C = l]\frac{1}{N_e}\sum\limits_{j=1}^{N_e}(\frac{P_j[C=l|\underline{x}_i^{(j)}]}{P_j[C=l]})}. \tag{4.5}$$

The M-step again computes the new class priors using equation (4.3). Again, the ratio $\frac{P_e^{(t-1)}[C=k]}{P_j[C=k]}$ is effecting class prior correction. At convergence, (4.5) computes the ensemble *a posteriori* probabilities. The solution after the first E-step (before prior correction) in this case is the fixed "arithmetic averaging" rule, which we will compare against in our results. As in the previous subsection, this EM algorithm is globally convergent.

It is important to emphasize we are not especially advocating the censored model as a good statistical description that captures particular distributed ensemble scenarios[4]. We have simply identified the statistical modeling assumption (random censoring) needed for transductive ML to yield the (popular) arithmetic averaging form of combining. In section 4.4, we will experimentally evaluate this transductive technique, along with the "product rule" method and an alternative method, developed next.

4.3 An Information-Theoretic Transductive Ensemble

The aggregation technique developed in this section, which we dub the "information-theoretic" (IT) method, differs in important respects from the ML methods. First, the IT method does *not* require that individual classifiers produce *a posteriori* probabilities. Correction of class priors is effectively achieved even if each local classifier only produces hard decisions. Second, we saw in the last section that well-known aggregation rules emerge when specific statistical assumptions are made. On the one hand, these rules *generalize*, providing an ensemble decision given any set of local pmfs $\{\{P_j[c|\underline{\mathbf{x}}^{(j)}]\}, j = 1, \ldots, N_e\}$, associated with a new $\underline{\mathbf{x}}$. That is, suppose there are two test batches, $\mathcal{X}_1$ and $\mathcal{X}_2$. If transductive ML learning is applied to $\mathcal{X}_1$, assuming the class priors in $\mathcal{X}_2$ are the same as in $\mathcal{X}_1$, decisions can be made on the examples in $\mathcal{X}_2$ without performing additional learning[5]. On the other hand, the statistical assumptions

[4] In particular, in our experiments in section 4.4, there is no actual censoring of feature vectors.

[5] In principle $\mathcal{X}_1 \bigcup \mathcal{X}_2$ should be used for transductive learning, or, if the class priors are different in the two batches, separate transductive learning should be performed on each. However, learning complexity may preclude this.

may not be warranted – in particular, conditional independence, when there is significant redundancy between classifiers. As an extreme example, consider an ensemble with $N_e - 1 \gg 1$ *identical* weak classifiers (*i.e.*, ones that are perfect copies of each other) and one strong classifier, statistically independent of the rest. Clearly, the naive Bayes rule in (4.2) will bias in favor of the weak classifiers and give poor accuracy in this case. What is instead desired is that the aggregation function properly *account* for redundancies. For this example, it should ideally treat the ensemble as *effectively* consisting of two (independent) classifiers, one strong and one weak.

The IT method does not produce a parametric aggregation function – given a test batch, it simply yields a table of class *a posteriori* pmf values, with each row specifying the pmf for one sample in the test batch. On the one hand, this means there is no automatic rule of generalization for new samples – decisions on new samples can only be made via a new round of transductive learning. On the other hand, this approach does not make the statistical assumptions required for the ML techniques. The IT method can be applied for the case of a single classifier, as considered in [20] and in this case it will not require the statistical assumptions made there. However, IT is especially advantageous for the ensemble case focused on here, where there is need to account for classifier redundancies. We will demonstrate that IT *does* properly treat redundancy for the stark type of example described above and, more generally, outperforms the ML methods. Finally, we note that there is no *explicit* prior correction in the IT method because it optimizes directly over test batch decisions, rather than over prior parameters. Regardless, IT is experimentally observed to achieve good decision accuracy in cases of large prior mismatch, *i.e.*, it achieves *effective* prior correction.

The IT method is inspired by frameworks such as *maximum entropy (ME) statistical inference* [5], which treats learning of probability distributions as a problem of encoding equality constraints on a set of judiciously chosen statistics. ME has been justified on statistical and axiomatic grounds and the solution consistent with given satisfiable linear constraints is unique. Moreover, there are simple ME parameter learning algorithms with guaranteed convergence such as iterative scaling techniques, e.g. [1]. In other work [54] we have explicitly cast *standard* ensemble combining (where there is common training data across the ensemble) as an ME problem with an iterative scaling solution. Unfortunately, for the *transductive* setting addressed here, the "right" constraints are not guaranteed to be satisfied. Accordingly, we will take a heuristic optimization approach, albeit for a principled goal, choosing ensemble posteriors on the test batch to agree "as closely as possible" with constraints measured by the local classifiers, as reflected by an information-theoretic measure of "distance" between probability distributions.

Casting learning as a problem of constraint encoding is especially advantageous for achieving robustness to classifier redundancy, as can be understood from the following simple argument. Suppose we have already optimized the test set decisions so as to *precisely* meet the constraints measured by an ensemble of N_e local classifiers. If we increase the ensemble size by one, adding a classifier that is an identical copy of an existing one in the ensemble, this classifier's constraints are perfectly redundant and *already met* by the current solution. Thus, this new classifier will have no effect on the solution. As will be demonstrated, the IT method handles classifier redundancy in a fashion quite consistent with this description, achieving accuracy nearly invariant to explicit redundancy in the ensemble.

4.3.1 Choice of Constraints

In general, one would like to encode as much (accurate) statistical constraint information as possible, to learn the most accurate probability model [5], *i.e.* joint probabilities on collections of features, e.g. [1] and [44]. In our ensemble setting, the local decision $\hat{C}_j$ is a discrete-valued feature. Thus, *in principle*, we might try encoding joint statistics such as, e.g., fourth-order pmfs $P[C, \hat{C}_j, \hat{C}_k, \hat{C}_l]$. However, constraint probabilities are usually *estimated*, via training set frequency counts, with accuracy that decreases with increasing order of the probabilities. This limits the orders used in practice. Moreover, in our distributed setting, with no common labeled data, it is not possible to *measure* joint statistics involving two or more classifiers. Thus, we are limited to encoding pairwise statistics involving C and individual decisions, $\hat{C}_j$. In prior work [44], pairwise pmfs involving the class label and individual discrete features were encoded into the learning of ME classifiers and this was found to achieve good accuracy. In our ensemble setting, using its local training set, each classifier, j, can measure the pairwise pmf $P^{(j)}_{\text{con}}[C, \hat{C}_j]$[6], with 'con' indicating this pmf is a constraint. Thus, at first glance, it appears reasonable to loosely cast our transductive learning problem as choosing the ensemble posteriors $\{P_e[C|\underline{\mathbf{x}}_i], i = 1, \ldots, N_{\text{test}}\}$ to satisfy the pairwise pmf constraints contributed by each of the local classifiers, *i.e.* $\{P^{(j)}_{\text{con}}[C, \hat{C}_j], j = 1, ..., N_e\}$. However, note that the pairwise pmf $P^{(j)}_{\text{con}}[C, \hat{C}_j]$ *determines* the marginal pmfs $P^{(j)}_{\text{con}}[C]$ and $P^{(j)}_{\text{con}}[\hat{C}_j]$. That is, by seeking to agree with the constraint $P^{(j)}_{\text{con}}[C, \hat{C}_j]$, one is also trying to force the ensemble test batch posteriors $\{P_e[C|\underline{\mathbf{x}}_i]\}$ to agree with the (perhaps *erroneous*, as well as

[6]The superscript indicates that the constraint is measured by classifier j.

inconsistent) class priors and class decision priors measured by each local classifier, based on its respective training set[7]. Thus, rather than encoding the *joint* pmfs $\{P^{(j)}_{\text{con}}[C, \hat{C}_j]\}$, we instead suggest to encode the *conditional* pmfs $\{[P^{(j)}_{\text{con}}[\hat{C}_j|C = c]\forall c\}$. These constraint pmfs do *not* specify the class priors and they specify the pairwise pmf *except for* the prior. They are thus as close as we can get to encoding pairwise pmfs [8]. Note also that the pmfs $\{P^{(j)}_{\text{con}}[\hat{C}_j|C = c]\forall c\}$ define the *confusion matrix* for classifier j (measured based on its training set). We believe it is quite reasonable to expect these statistics should be preserved as measured on the test batch.

In [20] and previous works cited there, the confusion matrix was used in a procedure to re-estimate class priors for the test batch. These priors were then used to correct posteriors in essentially the same way they are corrected within our ML algorithms. While it is possible we could also use $\{[P^{(j)}_{\text{con}}[\hat{C}_j|C = c]\forall c\}$ in this way, such an approach would be similar to the methods suggested in the previous section and would *only* address prior correction. We instead use the confusion matrix information from each local classifier to define a set of *constraints* for transductive learning. This approach will be shown experimentally to be quite effective when there is prior mismatch. Beyond this, though, this method does not make unnecessary statistical assumptions – our (constraint-based)

[7] If $P^{(j)}_{\text{con}}[C]$ is mismatched to the true class prior, it is likely that $P^{(j)}_{\text{con}}[\hat{C}_j]$ is also mismatched to the true class decision prior (reflected in the test data batch).

[8] One might imagine that it is worthwhile to also encode $P^{(j)}_{\text{con}}[C|\hat{C}_j = \hat{c}]$. However, note that $P^{(j)}_{\text{con}}[C = c|\hat{C}_j = \hat{c}] = \frac{P^{(j)}_{\text{con}}[\hat{C}_j=\hat{c}|C=c]P^{(j)}_{\text{con}}[C=c]}{P^{(j)}_{\text{con}}[\hat{C}_j=\hat{c}]}$, *i.e.* the only information in $P^{(j)}_{\text{con}}[C = c|\hat{C}_j = \hat{c}]$ relative to $P^{(j)}_{\text{con}}[\hat{C}_j = \hat{c}|C = c]$ is a function of the (mismatched) quantities $P^{(j)}_{\text{con}}[C = c]$ and $P^{(j)}_{\text{con}}[\hat{C}_j = \hat{c}]$. Moreover, experimentally we have found that encoding $P^{(j)}_{\text{con}}[\hat{C}_j = \hat{c}|C = c]$ gives *much* better accuracy than encoding $P^{(j)}_{\text{con}}[C = c|\hat{C}_j = \hat{c}]$.

method will be demonstrated to handle classifier redundancy much better than the ML methods.

The constraint probabilities for classifier j are estimated as[9]:

$$P_{\text{con}}^{(j)}[\hat{C}_j = \hat{c}|C = c] = \frac{\sum\limits_{i=1:c_i^{(j)}=c}^{N_j} P_j[\hat{C}_j = \hat{c}|\underline{x}_i^{(j)}]}{\sum\limits_{i=1:c_i^{(j)}=c}^{N_j} 1}. \tag{4.6}$$

Here, $\mathcal{X}_j \equiv \{(\underline{x}_i^{(j)}, c_i^{(j)}), i = 1, ..., N_j\}$ is the labeled training set for classifier j. The transductive *estimate* of the constraint (4.6), produced by choosing the ensemble *a posterior* pmfs on the test batch $\mathcal{X}_{\text{test}}$, is

$$P_e[\hat{C}_j = \hat{c}|C = c] = \frac{\sum\limits_{i=1}^{N_{\text{test}}} P_e[C = c|\underline{\mathbf{x}}_i] P_j[\hat{C}_j = \hat{c}|\underline{x}_i^{(j)}]}{\sum\limits_{i=1}^{N_{\text{test}}} P_e[C = c|\underline{\mathbf{x}}_i]}. \tag{4.7}$$

4.3.2 Nonsatisfiability of Constraints

Consider the following 2-class example with 2 classifiers: suppose each classifier makes hard decisions and let one classifier achieve perfect training accuracy, i.e.,

$$P_{\text{con}}^{(1)}[\hat{C} = 1|C = 2] = 0, P_{\text{con}}^{(1)}[\hat{C} = 2|C = 1] = 0. \tag{4.8}$$

[9] It is also possible to improve this estimate via a cross validation procedure.

For the second classifier, let

$$P^{(2)}_{\text{con}}[\hat{C}=1|C=1]=1, P^{(2)}_{\text{con}}[\hat{C}=2|C=2]=1/2. \tag{4.9}$$

Suppose the same training set is seen by each local classifier. Let the test batch be the same as the training set, *except* that it contains additional samples from class 2 that are all correctly classified by *both* classifiers. In this case, it is easy to see it is not possible to choose $\{\{P[C=c|\underline{\mathbf{x}}_i]\}, i=1,...,N_{test}\}$ to satisfy all the specified conditional constraints. Thus, as one expects, one cannot in general jointly satisfy distributed constraints, gleaned from a collection of local classifiers, via transductive learning.

4.3.3 Information-theoretic Learning Approach

Although the given constraints cannot be *precisely* met in general, in practice we have found they typically can be quite well met. Thus, here we develop a learning approach that seeks to agree with them "as much as possible". Since our constraints are on pmfs, we quantify the discrepancy between constraint pmfs and ensemble estimates via the information-theoretic measure of *relative entropy* [55], *i.e.* $D(P[X]||Q(X)) = \sum_{x\in\mathcal{S}} P[x]\log\frac{P[x]}{Q[x]}$, $\mathcal{S}$ the (common) support set, with $P[\cdot]$ playing the role of "posterior" and $Q[\cdot]$ the "prior". In our case, the ensemble's estimate is the "posterior" and the constraint pmfs are the "priors". From the previous section, we would ideally like to achieve

$$P_e[\hat{C}_j|C=c] \doteq P^{(j)}_{\text{con}}[\hat{C}_j|C=c], \forall j, c. \tag{4.10}$$

It thus *appears* natural to choose the ensemble posteriors to minimize the sum of cross entropies:

$$\tilde{R} = \sum_{j=1}^{N_e} \sum_{\hat{c}=1}^{|C|} D(P_e[\hat{C}_j|C=c]||P_{con}^{(j)}[\hat{C}_j|C=c]). \tag{4.11}$$

However, consider the case where the true prior is *zero* for some class $\tilde{c}$, *i.e.* where this class does not occur in the test batch[10]. In this case, the posterior pmfs should be chosen such that $\sum_{i=1}^{N_{\text{test}}} P_e[C=\tilde{c}|\underline{\mathbf{x}}_i] = 0$. However, this choice would lead to an improper estimate of $P_e[\hat{C}_j|C=\tilde{c}]$ in (4.7). Choosing the test set posteriors to minimize (4.11) *forces* the choice of a proper $P_e[\hat{C}_j|C=\tilde{c}]\forall j$, which is achieved *only* if $\sum_{i=1}^{N_{\text{test}}} P_e[C=\tilde{c}|\underline{\mathbf{x}}_i] > 0$. Thus, for the case of a missing class, minimizing $\tilde{R}$ must yield soft decision inaccuracy (as reflected by $\sum_{i=1}^{N_{\text{test}}} P_e[C=\tilde{c}|\underline{\mathbf{x}}_i] > 0$).

The problem here is that (4.10) (and $\tilde{R}$) include constraints (involving $\tilde{c}$) that should not be imposed. What we desire is a simple way to *avoid* encoding constraints $\{P_{\text{con}}^{(j)}[\hat{C}_j|C=\tilde{c}], \forall j\}$, associated with classes $\tilde{c}$ that do not occur in the test batch (even as we do not *a priori* know whether there *are* such missing classes). This is effectively achieved by multiplying (4.10), both sides, by $P_e[C=c] = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} P_e[C=c|\underline{\mathbf{x}}_i]$, *i.e.* by expressing the constraints

$$P_e[\hat{C}_j|C=c] \cdot P_e[C=c] \doteq P_{\text{con}}^{(j)}[\hat{C}_j|C=c] \cdot P_e[C=c], \forall j, c. \tag{4.12}$$

[10]Both the particular class and the *fact* that a class is missing from the test batch are of course unknown.

It may look as though (4.12) is imposing the (undesirable) pairwise pmf constraints discussed in section 4.3.1, since (4.12) equates two pairwise pmfs. However, (4.12) does not impose the local classifier's class prior, $P^{(j)}_{\text{con}}[C = c]$. It only imposes the conditional pmf $P^{(j)}_{\text{con}}[\hat{C}_j|C = c]$. In particular note that (4.12) is *equivalent to* the conditional pmf constraints (4.10) for c such that $P_e[C = c] > 0$ (since in this case $P_e[C = c]$ can be canceled on both sides), but with *no* constraint imposed when $P_e[C = c] = 0$, i.e. (4.12) encodes conditional pmf constraints for all classes estimated to have non-zero priors in the test batch. The effect of multiplying both sides of (4.10) by $P_e[C = c]$ is thus to avoid encoding constraints for classes estimated not to be present in the test batch. We encode the objectives (4.12) by choosing the ensemble posterior pmfs on $\mathcal{X}_{\text{test}}$ to minimize the sum of relative entropies over all local classifiers:

$$\begin{aligned} R &= \sum_{j=1}^{N_e} \left[\sum_{c=1}^{|C|} \sum_{\hat{c}=1}^{|C|} P_e[\hat{C}_j = \hat{c}|C = c] P_e[C = c] \log \frac{P_e[\hat{C}_j = \hat{c}|C = c] P_e[C = c]}{P^{(j)}_{\text{con}}[\hat{C}_j = \hat{c}|C = c] P_e[C = c]} \right] \\ &= \sum_{j=1}^{N_e} \sum_{c=1}^{|C|} P_e[C = c] D(P_e[\hat{C}_j|C = c] || P^{(j)}_{con}[\hat{C}_j|C = c]). \end{aligned} \tag{4.13}$$

The latter form clearly shows the relationship between R and $\tilde{R}$. Experimentally, we have found that minimizing $\tilde{R}$ gives very similar classification accuracy to minimizing R when there are no missing classes in the test batch. However, minimizing $\tilde{R}$ *fails* when there are missing classes, as will be shown in section 4.4.

We perform the minimization of R via gradient descent. The partial derivative with respect to a given posterior probability can be shown to be:

$$\frac{\partial R}{\partial P_e[C=k|\underline{\mathbf{x}}_m]} = \sum_{j=1}^{N_e}\sum_{\hat{c}_j=1}^{N_c} \frac{1}{N_{test}} P_j[\hat{C}_j=\hat{c}_j|\underline{x}_m^{(j)}] \log(\frac{P_e[\hat{C}_j=\hat{c}_j|C=k]}{P_{\text{con}}^{(j)}[\hat{C}_j=\hat{c}_j|C=k]}) \forall k,m. \quad (4.14)$$

Note that one way of achieving zero gradient (and an optimal solution) is if the constraints are precisely met.

In practice, without loss of generality, but to ensure $P_e[C=k|\underline{\mathbf{x}}_m]$ is preserved as a pmf throughout the optimization, we parameterize it using a softmax function, i.e.,

$$P_e[C=k|\underline{\mathbf{x}}_m] = \frac{e^{\gamma_{k,m}}}{\sum\limits_{k'} e^{\gamma_{k',m}}}, \quad (4.15)$$

based on the real-valued parameters $\{\gamma_{k,m}\}$. Applying the chain rule, we have

$$\frac{\partial R}{\partial \gamma_{m,k}} = \sum_{k'=1}^{N_c} \frac{\partial R}{\partial P_e[C=k'|\underline{\mathbf{x}}_m]} \cdot \frac{\partial P_e[C=k'|\underline{\mathbf{x}}_m]}{\partial \gamma_{m,k}}, \quad (4.16)$$

where

$$\frac{\partial P_e[C=k'|\underline{\mathbf{x}}_m]}{\partial \gamma_{m,k}} = \begin{cases} P_e[C=k'|\underline{\mathbf{x}}_m] \cdot (1 - P_e[C=k'|\underline{\mathbf{x}}_m]), & \text{for } k=k' \\ -P_e[C=k'|\underline{\mathbf{x}}_m] \cdot P_e[C=k|\underline{\mathbf{x}}_m], & \text{for } k \neq k'. \end{cases} \quad (4.17)$$

We have not determined whether (4.13), based on the parameterization (4.15), is convex – there may be multiple local minima. Thus, unlike section 3 where the learning was

globally convergent, gradient descent may only find solutions locally optimal with respect to (4.13).

Finally, note that implementation of this technique requires that each classifier, j, in addition to conveying its test batch decisions, also conveys the constraint pmfs $\{P_{\text{con}}^{(j)}[\hat{C}_j|C=c]\forall c\}$ to the aggregation function.

4.4 Experimental Results

We have performed a number of experiments, involving several types of comparisons:

1. Transductive ML learning versus fixed combining (sum and product).

2. Transductive information-theoretic (IT) learning versus fixed combining.

3. Transductive ML versus transductive IT learning.

4. Specialized tests to study the effect of explicit classifier redundancy, the performance with strong classifiers (SVMs), the case of *missing* classes, and the effect of batch size.

The data sets, taken from the UC Irvine machine learning repository, are described in Table 4.1. To simulate a distributed classification environment we used 5 local classifiers, each (initially) a Naive Bayes classifier working on a randomly selected subset of features[11]. Continuous features were modeled by a Gaussian distribution and discrete features by a multinomial. For example, *Vehicle* has 4 classes and 18 continuous features. The first local classifier used 15 randomly selected features while the second classifier used 14. We performed five replications of two-fold cross-validation for all data

[11]Thus, for the conditional independence model of section 3.1, both the local classifiers and the combining rule are based on naive Bayes.

Table 4.1. UC Irvine ML data sets that were evaluated.

Data set	Type of Features	Number of classes	Number of features	Feature subspace size
Satellite	Continuous	5	33	[28, 29, 30, 31, 32]
Vehicle	Continuous	4	18	[15,14,15,16,17]
Segment	Continuous	7	19	[15, 16, 15, 17, 18]
Liver Disorder	Continuous	2	6	[5, 4, 5, 4, 5]
Diabetes	Continuous	2	7	[6, 3, 4, 5, 6]
Mushroom	Discrete	2	22	[13, 15, 17, 19, 21]
Sonar	Continuous	2	60	[52, 54, 56, 58, 59]

sets except *Satellite*. For each replication, the data set was divided into two equal-sized sets. In the first fold, one set was used as training and the other as test, with these roles reversed in the second fold.

Prior Mismatch : We evaluated classification accuracy for varying test set priors. A new test set with given priors was obtained by sampling with replacement from the original test set. For example, for the first fold of the first replication, denote the training set by S_a and the test set by S_b. Suppose we want to change the test set priors to $\{m_1, ..., m_{N_c}\}$ where $\sum_{j=1}^{N_c} m_j = 1$. Then the new test set ($\hat{S}_b$) is obtained by sampling with replacement from S_b, so as to achieve (or approximately achieve) $\{m_1, ..., m_{N_c}\}$. The size of the new test set is chosen the same as the original set, S_b. *Satellite* has separate training and test sets. For this set, rather than performing cross validation, we averaged the test error rate over 10 trials, based on different resamplings of the test data for the given mismatched priors. We quantify the mismatch between test set and local training set priors by the

sum of cross entropies:

$$M = \sum_{j=1}^{N_e} \sum_{k=1}^{N_c} P_j[C = k] \log \left(\frac{P_j[C = k]}{P_{test}[C = k]} \right). \tag{4.18}$$

We evaluated performance varying the value of this mismatch index.

Initialization : The transductive ML methods used initial priors estimated based on frequency counts taken over all the local training sets. These priors were also used as (fixed) priors by the fixed rule methods. The fixed rule methods are thus equivalent to the solutions obtained by the (corresponding) EM algorithms (sections 3.1 and 3.2) after the first E-step, before any prior correction is made. The IT method started from a uniform posterior pmf initialization for all test batch samples, i.e. via $\gamma_{k,m} = 0 \; \forall \; k, m$.

Stopping Criteria : In practice, one must either use a fixed stopping rule or one *automatically* determined, given each new data set. The desire is that learning should cease without substantial over or underfitting. In this work, we have applied fixed stopping rules empirically chosen for each method. Figure 4.1 gives example curves illustrating that test set performance is not very sensitive to the choice of stopping rule – we observed little overfitting for any of the methods. For the ML methods, we stop learning when the relative change in log-likelihood drops below 1e-3. Since most of the improvement in test error was observed to occur in the first few iterations, this was found to generally stop near the best test set performance. For IT, we used gradient descent starting from a nominal step size (with value 10), taking steps with this size so long as the learning objective decreases. When this is not the case, the step size is lowered by half. Thus, (4.13) monotonically decreases with each step taken. We have found that the general

trend for the test error rate is (also) non-increasing behavior with increasing number of steps. For IT, we stop learning when the relative change in (4.13) is less than 1e-4 or after 3000 learning steps. This ensures the algorithm does a sufficient amount of learning. For all the methods, there are cases where generalization error does increase during learning. However, we have observed these increases to be modest.

Transductive ML learning versus fixed combining (sum and product)

Comparison of generalization error rates for transductive ML and fixed combining is given in Table 4.3. The second column gives the mismatch between training and test set priors. It can be seen that, when the mismatch increases, ML methods outperform the fixed rules, with the difference generally growing with mismatch. For example, for *Diabetes* when the mismatch is 0.74, ML (sum/product) error rates are 27.29/26.58 which are less than the fixed combining values of 28.91/28.83. (Saerens et al., 2002) used a likelihood ratio test to decide if the *a priori* probabilities have significantly changed from training to test set. The solutions obtained by their EM learning were only retained if the likelihood ratios exceeded a confidence threshold. We have found it is not very necessary to apply this type of test for our method – only in one case of small mismatch value, for *Liver Disorder*, did we find that use of transductive ML gave a noticeably higher test set error rate. That is, use of our approach was typically found to either improve performance or have little effect (in the small mismatch case).

Transductive IT learning versus fixed combining (sum and product)

Table 4.3 compares error rates between IT and fixed combining. The IT method outperformed fixed rules for all data sets, with gains generally increasing with the prior mismatch. As one example of IT's prior correction, on *diabetes* with $M = 2.58$, the

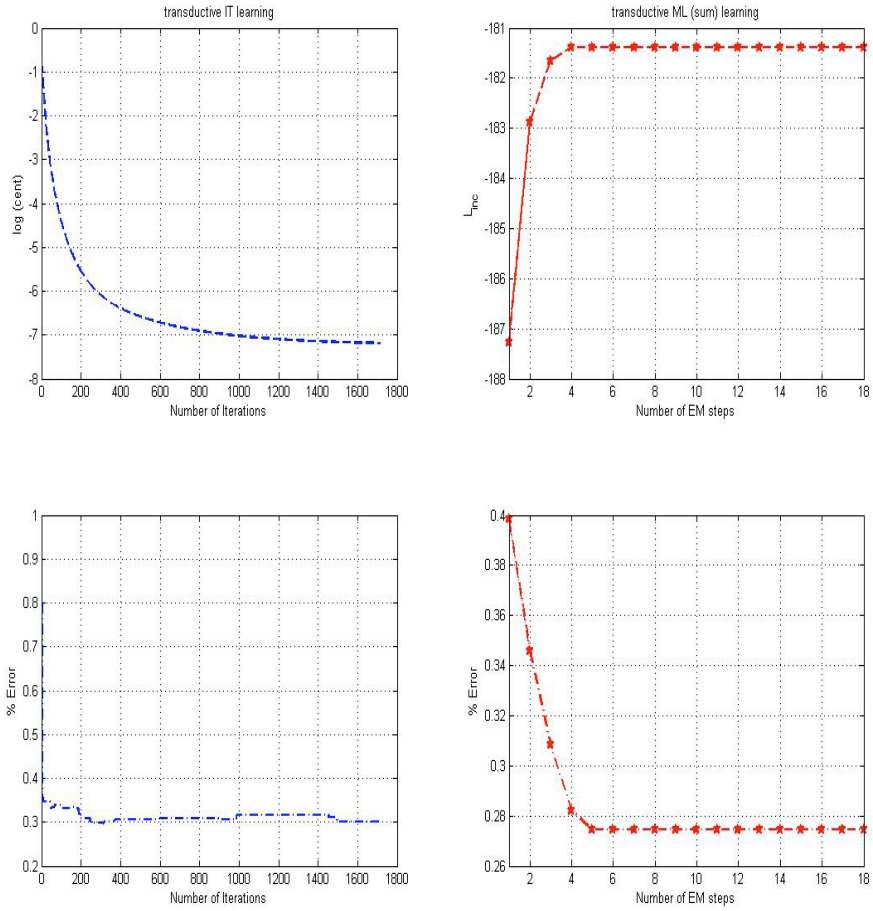

Fig. 4.1. Learning objective and the test error rate as learning proceeds for transductive methods on *Diabetes* with prior mismatch value of 2.55.

Table 4.2. Generalization error rates based on 5 replications of two-fold cross validation, for transductive ML and fixed methods.

Dataset	M	**Sum**		**Product**		**5x2 cv F Test**			
						ML Sum		**ML Product**	
		ML	Fixed	ML	Fixed	Fixed (S)	Fixed (P)	Fixed (S)	Fixed (P)
Satellite	0.31	22.56	22.51	22.65	22.56	-	-	-	-
	0.53	21.00	21.02	20.97	21.10	-	-	-	-
	0.97	21.16	21.43	21.11	21.53	-	-	-	-
	1.68	24.87	25.32	24.97	25.28	-	-	-	-
Segment	3.2e-2	20.88	20.94	20.96	20.96	0.56	0.60	0.70	0.77
	1.30	33.13	33.85	33.00	34.11	0.78	0.98	0.94	1.21
	1.73	25.78	27.04	26.10	27.16	2.35	2.87	2.71	3.81
	2.57	41.44	43.38	41.00	43.48	6.12	4.78	6.27	4.81
Liver D	5.1e-2	46.13	44.10	47.96	44.05	1.65	1.59	1.91	1.36
	0.55	36.24	38.03	35.20	38.38	1.31	1.26	1.57	1.55
	0.83	35.49	40.98	34.97	40.92	4.49	3.61	4.74	4.28
	2.43	21.04	31.21	17.92	31.27	5.89	5.13	4.12	4.12
Diabetes	3.1e-2	23.20	23.27	23.05	23.20	0.73	1.09	0.68	0.65
	0.74	27.29	28.91	26.58	28.83	1.93	1.95	2.82	2.67
	1.42	26.99	33.08	24.85	32.93	4.73	4.58	4.76	4.90
	2.58	23.50	33.46	21.77	33.08	8.15	7.87	7.18	7.50
Mushroom	1.2e-3	0.65	0.65	0.70	0.69	0.78	1.16	1.2	1.0
	0.36	0.49	0.93	0.49	0.87	6.04	3.9	4.5	3.9
	0.99	6.6e-2	0.44	5.7e-2	0.42	5.35	4.8	6.4	5.86
	2.36	0.078	0.16	0.12	0.25	0.66	1.37	0.552	0.96
Sonar	4.2e-2	31.83	32.12	32.02	32.02	0.76	0.84	0.82	1.0
	0.60	26.54	26.83	26.06	26.83	0.79	0.79	0.68	0.68
	1.35	25.58	26.54	25.67	26.92	0.86	1.34	0.79	1.23
	1.94	18.85	20.96	19.42	20.77	5.00	5.83	5.16	5.40

Table 4.3. Generalization error rates based on 5 replications of two-fold cross validation, for transductive IT and fixed combining methods.

			Fixed		**5x2 cv F Test**	
Dataset	M	IT	sum	product	sum	product
Satellite	0.96	20.88	21.46	21.40	-	-
	1.68	25.28	25.60	25.51	-	-
Vehicle	1.27	41.35	45.60	45.56	6.45	5.6
	2.73	34.53	41.18	41.11	1.26	1.25
Segment	3.4e-2	16.51	21.32	21.40	5.13	5.26
	1.31	27.75	37.23	37.57	34.01	26.34
	1.83	11.85	13.59	13.89	9.75	8.63
	2.57	32.12	43.66	43.71	9.98	9.65
Liver Disorder	5.6e-2	44.57	45.20	45.09	0.71	0.68
	0.55	43.24	44.51	44.68	0.77	0.74
	1.74	32.54	35.66	35.72	0.80	0.80
Diabetes	2.3e-2	23.08	23.72	23.91	1.30	1.45
	0.74	26.99	33.76	33.16	2.68	1.99
	1.43	24.36	30.68	30.90	15.77	15.45
	2.57	17.82	31.84	31.69	8.31	7.66
Mushroom	1.6e-3	0.24	0.33	0.57	4.0	8.11
	0.36	0.073	0.13	0.17	2.42	4.52
	0.99	0.12	0.83	0.81	4.86	5.0
	2.36	0.12	0.52	0.96	2.49	7.11
Sonar	0.32	31.83	35.29	35.38	2.56	2.58
	0.91	27.21	35.19	35.29	2.72	2.78
	1.37	23.08	37.50	37.79	7.74	8.36

true priors were $[0.2, 0.8]$ and the transductive estimates were $[P_e[C = 0], P_e[C = 1]] = [0.29, 0.71]$.

Transductive ML versus transductive IT learning

Table 4.4 compares all the transductive methods. For all data sets, IT performed better than the ML methods except on *Mushroom* for the mismatch values 0.36 and 2.36. The difference in error rates in both cases is $\leq 0.1\%$. We attribute much of the gains of the IT method to the nonparametric nature of its combining – in encoding the given constraint information, IT does not make the statistical assumptions (presumably unwarranted) of the ML methods. We will shed further light on this point, shortly, when we consider specialized tests involving explicit classifier redundancy.

Statistical tests

To assess statistical significance of the comparisons, we used the 5x2 cv F test (Alpaydin, 1999). Here, 5 replications of two-fold cross validation are performed. The difference between the generalization error (p_i^j) for both classifiers is computed where i is the replication index and j the fold index. The estimated variance (s_i^2) of this difference is computed, with the average difference on each replication given by $\overline{p_i} = 0.5 * (p_i^1 + p_i^2)$. The 5x2 cv F test statistic

$$f = \frac{\sum_{i=1}^{5} \sum_{j=1}^{2} \left(p_i^j\right)^2}{2 * \sum_{i=1}^{5} s_i^2} \tag{4.19}$$

is approximately F distributed (Alpaydin, 1999) with 10 and 5 degrees of freedom. We can reject the hypothesis that two algorithms have the same error rate with 95% confidence if f is greater than 4.74. In Table 4.2, columns 7 and 8 give values of f

Table 4.4. Generalization error rates based on 5 replications of two-fold cross validation, for transductive IT and ML methods.

			ML		**5x2 cv F Test**	
Dataset	M	IT	sum	product	sum	product
Satellite	0.97	20.85	20.94	20.85	-	-
	1.68	25.38	25.54	25.47	-	-
Vehicle	1.27	44.59	46.15	47.42	0.86	1.49
	2.73	34.45	34.86	34.83	0.52	0.51
Segment	3.3e-2	16.13	21.58	21.56	4.37	4.33
	1.30	25.34	34.98	35.42	3.06	3.01
	1.83	10.82	12.79	13.22	1.78	2.88
	2.57	31.93	43.85	43.20	1.65	1.75
Liver Disorder	4.2e-2	44.45	45.55	48.5	0.69	1.45
	2.44	21.33	33.87	37.23	2.04	2.12
	3.54	19.42	20.58	21.51	0.54	0.72
Diabetes	1.6e-2	23.05	23.83	24.06	0.69	1.09
	0.74	23.65	24.54	24.36	1.53	1.26
	1.43	23.91	26.84	26.39	2.48	1.56
	2.57	18.31	22.78	21.47	1.23	1.58
Mushroom	1.6e-3	0.24	0.34	0.56	10.4	8.6
	0.36	0.073	0.064	0.17	1	5.8
	0.99	0.12	0.25	0.30	1.07	1.98
	2.36	0.12	0.044	0.11	4.4	0.67
Sonar	2.5e-2	29.9	31.83	31.92	1.72	1.47
	0.18	30.29	31.35	31.35	1.29	1.39
	0.61	27.60	27.79	27.69	2.67	2.43
	1.34	25.96	27.02	26.73	0.68	0.66

Table 4.5. Effect of explicit classifier redundancy on transductive IT and ML test set classification accuracy.

Dataset	5	5+1	5	5+2	5	5+5	5	5+20
Diabetes								
mismatch	0.7331	0.8861	0.7363	1.0308	0.7278	1.4556	0.7374	2.94
IT	21.19	21.21	24.17	24.40	21.81	22.42	21.12	22.76
Tml(sum)	30.55	34.03	36.81	43.78	29.26	39.72	32.17	48.93
Tml(pro)	28.88	33.50	36.45	46.36	25.04	41.71	27.55	56.14
Ionosphere								
mismatch	0.2660	0.3192	0.2812	0.3937	0.2678	0.5356	0.2654	1.0617
IT	14.12	14.07	14.57	14.43	12.44	12.86	15.65	16.48
Tml(sum)	15.64	15.84	16.44	18.40	15.17	18.29	19.43	24.17
Tml(pro)	14.90	15.43	16.12	16.89	14.70	16.28	16.91	21.74

for comparisons between ML (sum) and fixed rules (sum/product). Columns 9 and 10 compare ML (product) and fixed rules (sum/product). Note that as prior mismatch increases the value of f generally increases. For *Sonar* when the mismatch is 1.94, $f \geq 4.74$ for all ML methods compared with fixed rules. In Table 4.3, columns 6 and 7 compare IT with the fixed combining rules. Except for *Liver Disorder*, all other data sets have at least one case where $f \geq 4.74$. Moreover, if we compare Tables 2 and 3, we see IT gives statistically significant improvement over the fixed methods both in more cases and for smaller mismatch values than the ML methods – IT gives significant improvement for $M < 1.5$ on *Vehicle*, *Segment*, *Diabetes*, and *Mushroom*. In Table 4.4, we compare IT with the ML methods. Although IT achieves smaller error rates, $f \leq 4.74$ except on *Mushroom*.

Redundant Classifiers

Table 5 gives results (for a single training/test split) comparing methods in the presence of *explicit* classifier redundancy. We selected the second weakest classifier from the five

Table 4.6. Generalization error based on five replications of two-fold cross validation for IT and fixed combining of linear, two-class SVMs.

Dataset	mismatch	SVM error (test)	IT	fixed (sum)	5x2 cv F test
	0.0563	25.08	19.78	20.52	0.69
Sonar	1.3801	27.20	23.10	23.89	0.82
	0.0165	17.20	14.18	15.42	0.968
Ionosphere	2.305	31.28	19.30	26.32	6.148
	0.0538	27.59	25.04	29.06	1.968
Diabetes	1.4301	41.43	28.80	45.02	8.148

in the ensemble (in terms of test error) and duplicated this classifier k times, k =1,2, 5, and 20. The column under '5' indicates the test error for the original ensemble, *i.e.* with $N_e = 5$, with column '5+k' giving the test error for the redundant ensemble, *i.e.* with $N_e = 5 + k$. Note that IT is almost *perfectly* robust to redundant classifiers, whereas for both ML methods the accuracy degrades as k increases. This experiment exhibits the weakness of the ML methods – and the strength of IT – in handling redundancy.

Strong Classifiers

In a distributed sensor context, each classifier may be inherently weak (e.g. with computation limited by battery power). However, when there is freedom to choose the local classifier, one might choose a strong one such as a support vector machine (SVM), also believed to be less sensitive to class prior mismatch than other classifiers. It is thus interesting to assess the value of transductive learning for this choice. Table 6 compares the IT method (the only transductive method that works with hard classifier decisions) with fixed averaging (sum) for linear SVMs on two-class tasks. The column 'SVM error (test)' gives the average test error of the *individual* SVM classifiers. For *Sonar*, SVM

performance is only weakly sensitive to increases in the mismatch value, with the error increasing from 25.1 to 27.2 %, while for the other two data sets SVM performance *is* quite sensitive to mismatch. From the table, the advantage of IT over fixed rules appears to depend on the degree of this sensitivity – the advantage of IT on *Sonar* is small (especially compared with the results for naive Bayes classifiers in Table 3), while the advantage for the high mismatch case is statistically significant on both *Ionosphere* and *Diabetes*.

Missing Classes in the Test Batch

We performed two experiments to validate our claim in section 4.4 that the IT method (minimizing R in (4.13)) handles missing classes in the test batch much better than minimizing $\tilde{R}$ in (4.11). On the two-class *Diabetes*, excluding class 1 from the test set, IT estimated $P_e[C = 1] = 0.05$ and achieved a test error rate of 2.9%. By contrast, minimizing $\tilde{R}$ gave $P_e[C = 1] = 0.51$ and a test error rate of 39.62%. On the seven class *Segment*, excluding class 1, IT estimated $P_e[C = 1] = 0.01$, with a test error rate of 14.87%. Minimizing $\tilde{R}$ yielded $P_e[C = 1] = 0.08$ and a test error rate of 30.47%.

Batch Size

Finally we assessed how batch size affects performance. Table 4.7 shows generalization error for varying test batch size, for IT compared with ML combining (shown in parenthesis). While IT improves with increasing batch size, IT performance is comparable to/better than ML when the batch size is at least 30% of the test set. Note also that the ML performance seems largely invariant to batch size.

Table 4.7. Effect of batch size on generalization error for transductive IT learning, compared with ML sum/product combining (shown in parentheses).

Dataset	M	batch size as % of test set				
		10%	30%	50%	80%	100%
L. Disorder	4.6e-2	47.54 (46.89/47.64)	44.20 (46.34/47.10)	41.21 (45.26/46.88)	42.64 (46.40/46.15)	43.00 (46.56/46.05)
Diabetes	2.2e-2	25.12 (22.56/22.10)	22.78 (23.44/23.48)	23.28 (22.12/22.52)	23.10 (23.52/23.40)	21.58 (22.10/21.78)

4.5 Conclusions

We identified the distributed ensemble classification problem, discussed applications in which it arises, and proposed several transductive methods for its solution. All our methods were found to improve on fixed combining rules, with relatively large gains in accuracy achieved in the case of large class prior mismatch. The IT method generally outperformed the ML methods and is the only method suitable when local classifiers solely produce hard decisions. This method gave performance benefits both when individual classifiers are relatively weak (naive Bayes) and strong (SVMs). This approach was also demonstrated to be much more robust to classifier redundancy than the ML methods. IT also well-handled the case where classes are *missing* from the test batch. There are several directions to pursue in future. First, we did not address any practical communications issues, such as quantization of soft decisions and variable delays from different classifiers, which could disrupt synchronization needed to aggregate decisions. We also did not consider possible information exchange between local classifiers. Second, a theoretical study of conditions under which the ensemble class priors converge to the true ones could be undertaken. Third, our approaches could be improved by conveying information beyond local decisions. In [54], we proposed a method based on

mixture modeling for encoding novel constraints involving continuous features, within a maximum entropy framework. This approach could be applied within our distributed ensemble setting, with information based on a local sensor's continuous features (i.e. *a posteriori* probabilities on mixture component indices) conveyed, along with local decisions, and with associated constraints encoded. Finally, other aggregation rules could be explored, e.g. it is straightforward to extend the ML methods to produce the alpha-trimmed mean in [39]. It would be interesting to determine whether other combining rules also arise as ML learning solutions.

Chapter 5

Application of Transductive Ensemble Learning

5.1 Introduction

In this information era, secure access to information is becoming consistently important. The traditional approaches of using password or personal identification numbers do not meet the requirements of identification of a person. The security of password based systems depends on strength and safekeeping of passwords. Biometrics on the other hand are much more powerful and we don't have to remember anything, we carry it all the time. Biometrics is used to recognize or verify an identity using a measurable physiological (including palmprint, fingerprint, retina scans, or facial features) or behavioral (gait, voice scans, or handwritten signatures) characteristics. Biometrics based identification systems are emerging as a reliable identification method.

The terms *identification* and *authentication* can be used interchangeably in normal day to day conversation but these are certainly different terms in Biometric recognition. In *identification* scenario the biometrics of the person(F_i) are compared with the biometrics of persons (or clients)($\{I_1, ..., I_N\}$) stored in the database. The identity (I_m) which is closet to the candidate set is assigned to the candidate(F_i). Where as in *authentication* (or *verification*), the candidate(F_i) claims an identity(I_j) and the system compares the candidate set (F_i) with only the claimed identity. If the candidate is less

than the threshold value then claim is accepted otherwise it is rejected. *Authentication* is therefore a two class problem i.e., either the candidate is a client or is an impostor.

Since the Biometrics involve measurement of different physiological and behavioral characteristics, the most pertinent question to ask would be how to build a recognition system that will take into account all these measurements and yield lower error rates. There are possible several solutions to this

- The most common approach is to combine all modal measurements and learn a single classifier. The main drawback using this approach is that the as feature dimensionality increases there is always the problem of obtaining sufficient training data and also the accuracy of the system takes the hit. Another potential problem would be that a single classifier might not be *optimal* for all the measurements. For example, it might turn out that DCT based classifier works well for facial data and a Gaussian mixture model based classifier may work well for speech data.

- Another approach is to divide the feature space and learn separate classifier for different modalities and later on *fuse* the decisions from these classifiers. This approach can also be applicable to different classifiers operating on same modality for example two cameras separated in placement and capturing different profiles of the face and feeding the data to two different facial classifiers (e.g. GMM or DCT based classifier). For example, if the biometric measurement is obtained for fingerprint, face and speech then the system can use different classifiers for each of these modalities like DCT for facial, GMM for speech and LDA for fingerprint. Then decisions from these can be fused together either by fixed aggregation rules

(sum, product, geometric mean, maximum, minimum) or supervised aggregation techniques. [56] showed the improved accuracy by combining voice and face based biometrics.

Mostly all of the previous work done in biometric fusion focuses on either the fixed aggregation rule or supervised aggregation rule (see for example [57], [58], [59], [60], [61], [62]). But there might be circumstances where the training of the aggregation rule is not possible and either fixed decision combining rule or transductive methods might be need. For example, consider a company which plans to install a biometrics based system to only allow employees inside the main premise. Due to lack of funds or any other logistic problems the company starts with only a simple system which works with the facial image and in due course of time the additional biometrics (either using a voice classifier or a fingerprint classifier or any other biometric feature classifier) might be installed. As these systems were installed at different period of time (or they could be proprietary systems), it would not be possible to retrain the aggregation rule. In such cases supervised aggregation rule with fail to incorporate the new decision making hardware. In Chapter 4, we describe two main reasons why we expect transductive learning may outperform fixed combining. The main necessity of our transductive learning is availability of batch test set. We now discuss some scenarios in biometric recognition where such batch data might be available.

- One of such situations could be a employee authentication before he/she is allowed access in the work place. When the employee arrive at work place they are required take a token and submit their biometrics. Before entering the main premise they

are required to submit the token. They will only be allowed the entry if the system has cleared a particular token number. Mostly employees enter the office in groups (either normal office hours or working in shifts).

- Nowadays with increased security risk to nations from terrorists, it has increasingly become important to stop potential miscreants from entering country. Another potential application of could be in immigration checkpoint at the airport. When the airplane lands, all the passengers are required to take a token and provide with biometric samples and then move onto the queue for immigration check counter. This scenarios presents us with the batch test set and by the time person reaches the immigration official the system would have already given a decision about him. Here we can have known law offenders, drug cartel operators, people on terrorist watch list or those required by interpol as *"clients"*

- An other possible scenario could be a secure meeting (either in a virtual world or in real world). All the participants are required to submit their biometrics to a secure system before being allowed entry to the secure meeting location. In case of virtual world the participants can be distance apart but supply their biometric credentials to the central server which on accepting the claim of all or atleast M of the participants initiates a secure video conferencing(between all or M participants[1]).

Another method of constructing a batch test-set (of size T) is by recursively using $(T-1)$ previous batch data and adding on current test sample. This *dummy* batch (of

[1]Here we assume that each participant would not in turn rebroadcast this meeting on the network and is dedicated to keep the meeting as secret

size $T-1$) may have some impostor accesses and is reflective of the security concerns of the organization[2].

5.2 Experiments

There are independent projects going on to get multimodal biometric databases a few of these are BANCA [63], XM2VTS [64], BIOMET [65], and MYCT. The transductive IT method aggregates the decisions of the baseline systems. The baseline system can either produce *hard* or *soft* decision. The objective of our method is on achieving an effective aggregation rule given the scores from the baselines systems. For this purpose we used the score files[3] from publicly available XM2VTS benchmark database.

Evaluation criteria:

The Biometric systems are evaluated on basis of false acceptance (FA) and false rejection (FR). The Normalized versions of FA and FR are called false acceptance rate (FAR) and false rejection rate (FRR), respectively. They are defined as:

$$\begin{aligned} FAR &= \frac{FA}{N_I} \\ FRR &= \frac{FR}{N_C} \end{aligned}$$

where N_I and N_C are the total number of impostor and client accesses, respectively. Sometime the systems are also evaluated on basis of Half Total Error Rate (HTER). It is defined as $\frac{1}{2}(FAR + FRR)$

[2]This means that if the intrusion rate is pretty low then the dummy test set is reflective of that or vice versa

[3]http://www.idiap.ch/ norman/fusion

Dataset and fusion protocol:

The score files from baseline systems are available in form of 32 fusion datasets. They consists of both multimodal and intramodal fusion problems. These fusion datasets have been used by [66] for biometric authentication problem. For each dataset there are two sets of scores (development and evaluation) available. The 32 datasets are divided into 2 protocols called as Lausanne Protocol 1(LP1) and Lausanne Protocol 2 (LP2). In LP1 there are 111800 impostor instances and 400 client in evaluation set. While in development set the number of impostor access are 40000 and client access are 600. In LP2 there are 111800 impostor instances and 400 client in evaluation set. While in development set the number of impostor access are 40000 and client access are 400. In both the protocols there are huge number of impostor access. Hence the criteria of HTER which gives equal weights to false acceptance rate and false rejection rate is not suitable.

Results:

To compare results with [66] we fused only 2 systems at a time. [66] takes the mean of the scores and then find optimal threshold for the combined score on evaluation set. The same thresholding is then applied on test set to evaluate the performance of the system. Our transductive IT method does not require any search for optimal threshold. The availability of evaluation set allowed us to also generate results from SVMs. The comparative results are given in Tables 5.1 - 5.5. In all the fusion cases, the transductive IT method beats the fixed rule (mean) used by [66] in terms of the total number of errors and in the number of false accepts. As expected the SVMs perform the best in terms of total number of errors and in number of false accepts. Figure 5.1 shows the receiver

operating characteristic (ROC) curve for our transductive IT method. These are shown for fusion datasets with different modalities for LP1. We perform better in 11 out of 15 fusion problems.

Table 5.1. Fusion with different modalities for LP1

Fusion candidate	**Method in [66]**			**Trans. IT Method**			**SVM**		
	FA (111800)	FR (400)	Total (112200)	FA	FR	Total	FA	FR	Total
(FH,MLP)-(LFCC,GMM)	825	3	828	12	32	44	0	29	29
(FH,MLP)-(PAC,GMM)	872	5	877	21	43	64	0	41	41
(FH,MLP)-(SSC,GMM)	1102	3	1105	18	37	55	0	33	33
(DCTs,GMM)-(LFCC,GMM)	369	2	371	98	27	125	0	27	27
(DCTs,GMM)-(PAC,GMM)	936	7	943	230	40	270	0	44	44
(DCTs,GMM)-(SSC,GMM)	603	6	609	177	36	213	0	38	38
(DCTb,GMM)-(LFCC,GMM)	414	1	415	13	20	33	0	14	14
(DCTb,GMM)-(PAC,GMM)	1092	3	1095	45	34	79	0	27	27
(DCTb,GMM)-(SSC,GMM)	228	4	232	47	28	75	0	21	21
(DCTs,MLP)-(LFCC,GMM)	1402	1	1403	22	38	60	0	34	34
(DCTs,MLP)-(PAC,GMM)	1650	2	1652	70	58	128	0	54	54
(DCTs,MLP)-(SSC,GMM)	2028	2	2030	46	69	115	0	40	40
(DCTb,MLP)-(LFCC,GMM)	2040	4	2044	148	47	195	0	42	42
(DCTb,MLP)-(PAC,GMM)	4182	9	4191	75	350	425	0	77	77
(DCTb,MLP)-(SSC,GMM)	2760	11	2760	221	66	287	0	65	65

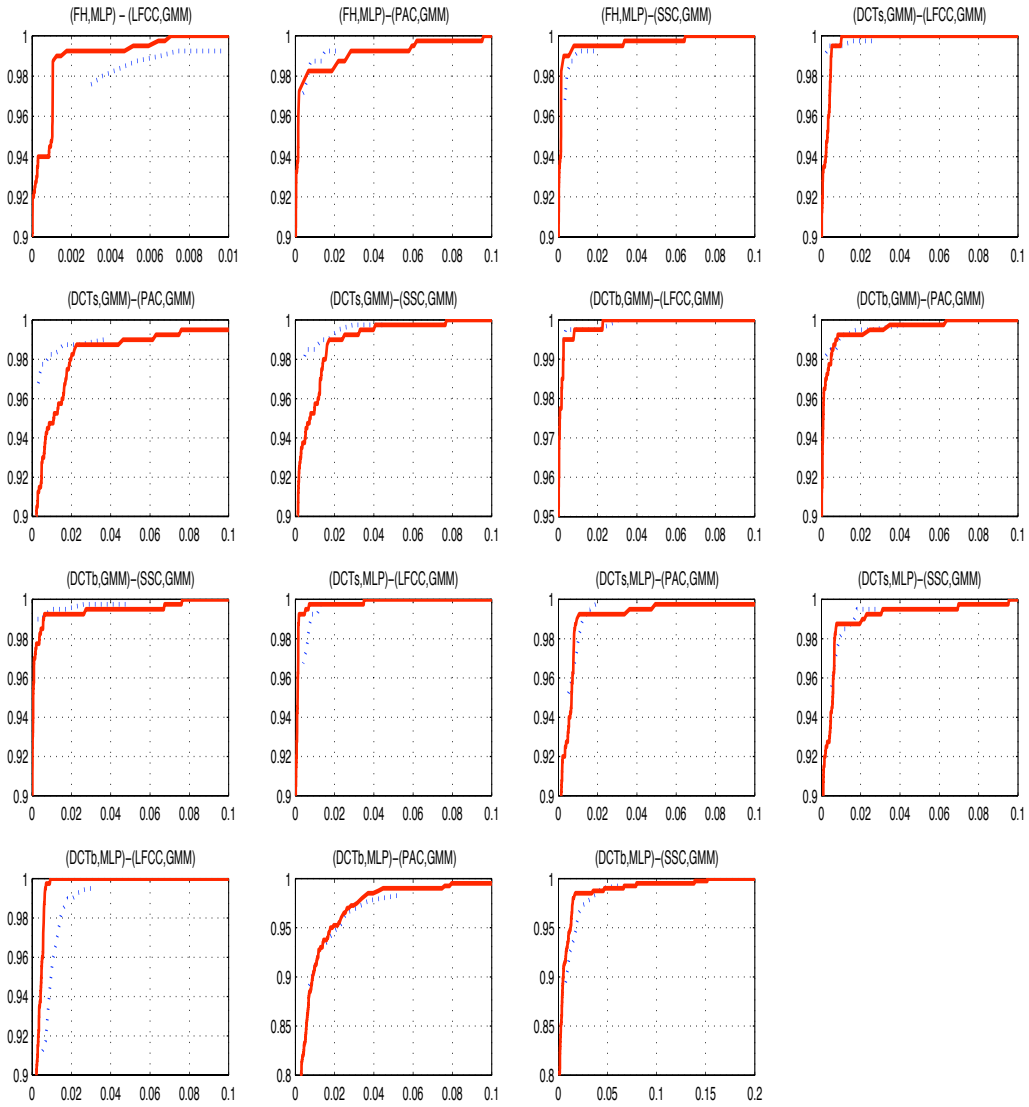

Fig. 5.1. ROC curve for transductive IT method for fusion with different modalities for LP1

Table 5.2. Fusion with different feature sets for LP1

Fusion candidate	**Method in [66]**			**Trans. IT Method**			**SVM**		
	FA (111800)	FR (400)	Total (112200)	FA	FR	Total	FA	FR	Total
(FH,MLP)-(DCTs,GMM)	1743	4	1747	26	44	70	0	30	30
(FH,MLP)-(DCTb,GMM)	1043	5	1048	7	38	45	0	37	37
(FH,MLP)-(DCTs,MLP)	1416	7	1409	20	50	70	0	31	31
(FH,MLP)-(DCTb,MLP)	1051	11	1062	14	55	69	0	30	30
(LFCC,GMM)-(SSC,GMM)	2047	3	2050	766	18	784	0	17	17
(PAC,GMM)-(SSC,GMM)	4408	18	4426	627	58	685	0	400	400

Table 5.3. Fusion with different classifiers for LP1

Fusion candidate	**Method in [66]**			**Trans. IT Method**			**SVM**		
	FA (111800)	FR (400)	Total (112200)	FA	FR	Total	FA	FR	Total
(DCTs,GMM)-(DCTs,MLP)	2911	7	2918	134	50	184	0	50	50
(DCTb,GMM)-(DCTb,MLP)	3338	8	3346	50	134	184	0	39	39

Table 5.4. Fusion with different modalities for LP2

Fusion candidate	**Method in [66]**			**Trans. IT Method**			**SVM**		
	FA (111800)	FR (400)	Total (112200)	FA	FR	Total	FA	FR	Total
(FH,MLP)-(LFCC,GMM)	798	1	799	5	24	29	0	24	24
(FH,MLP)-(PAC,GMM)	1567	0	1567	24	32	56	0	30	30
(FH,MLP)-(SSC,GMM)	842	4	846	3	34	37	0	32	32
(DCTb,GMM)-(LFCC,GMM)	18	1	19	1	12	13	0	1	1
(DCTb,GMM)-(PAC,GMM)	192	0	192	9	25	34	0	17	17
(DCTb,GMM)-(SSC,GMM)	219	0	219	2	25	27	0	17	17

Table 5.5. Fusion with different feature sets for LP2

Fusion candidate	**Method in [66]**			**Trans. IT Method**			**SVM**		
	FA (111800)	FR (400)	Total (112200)	FA	FR	Total	FA	FR	Total
(FH,MLP)-(DCTb,GMM)	1298	2	1300	6	32	38	0	14	14
(LFCC,GMM)-(SSC,GMM)	1573	1	1574	562	21	583	0	25	25
(PAC,GMM)-(SSC,GMM)	3100	12	3122	41	755	796	0	45	45

Chapter 6

Extensions of Transductive Ensemble Learning

The transductive ensemble learning method described in Chapter 4 encodes the *conditional* pmfs $\{[P^{(j)}_{\text{con}}[\hat{C}_j|C=c]\forall c\}$. We mention in concluding remarks of Chapter 4 that there can be additional information that can be encoded other than the local classifier decisions. We can use mixture modeling for encoding novel constraints involving continuous features. There can be possible two different ways to encode the *latent variable* from local sensors. We describe these in next section.

6.1 Extension I

There may be additional information, *non redundant* with $\hat{C}_j$, contained in $\underline{X}^{(j)}$. To extract this, in first extension, we propose to perform (e.g. multivariate Gaussian) mixture modeling on $\underline{X}^{(j)}$, yielding the *latent* mixture variable M_j and associated posteriors $\{P[M_j = l|\underline{x}^{(j)}_t], l = 1, \ldots, L_j, t = 1, \ldots, N_{\text{test}}\}$. The number of components, L_j, can be estimated via e.g. the Bayesian information criterion (BIC) or cross validation (We use BIC here.). Particular values for M_j (particular mixture components) may *e.g.* give indication of the degree of reliability of the associated posteriors $P[\hat{C}_j|\underline{x}^{(j)}]$. They may also simply possess statistical correlation with C that is not captured via $P[C, \hat{C}]$. We thus suggest the "more informative" constraints

$$P_e[\hat{C}_j, M_j, C = c] \doteq P^{(j)}_{\text{con}}[\hat{C}_j, M_j|C = c] \cdot P_e[C = c], \forall j, c. \tag{6.1}$$

Here,

$$P_{con}^{(j)}[\hat{C}_j = \hat{c}, M_j = l | C = c] = \frac{\sum_{t=1:C=c}^{N_{test}} P[\hat{C}_j = \hat{c} | \underline{x}_t] P[M_j = l | \underline{x}_t]}{N_{train}(C = c)} \tag{6.2}$$

and

$$P_e[C = c, \hat{C}_j = \hat{c}, M_j = l] = \frac{\sum_{t=1}^{N_{test}} P_e[C = c | \underline{x}_t] P[\hat{C}_j = \hat{c} | \underline{x}_t] P[M_j = l | \underline{x}_t]}{N_{\text{test}}}. \tag{6.3}$$

Again, consistent with satisfying (6.1), we form a sum of cross entropies learning objective and minimize over the test set posteriors via a gradient descent procedure.

The above described approach is the analogue, for the distributed ensemble problem, of the approach suggested in [54] for the *standard* ensemble problem (where there *is* a common pool of labeled training data across the ensemble).

6.2 Extension II

In previous section we proposed to encode ternary pmf. But instead of encoding the class conditional pmfs ($P_{\text{con}}^{(j)}[\hat{C}_j, M_j | C = c]$) we can encode constraints jointly conditioned on true class label and mixture component index ($P_{con}^{(j)}[\hat{C}_j | M_j = l, C = c]$). The constraint equation takes the form.

$$P_e[\hat{C}_j | M_j = l, C = c] \doteq P_{\text{con}}^{(j)}[\hat{C}_j | M_j = l, C = c], \forall j, l, c. \tag{6.4}$$

Here,

$$P_{con}^{(j)}[\hat{C}_j = \hat{c}|M_j = l, C = c] = \frac{\sum_{t=1:C=c}^{N_{train}} P[\hat{C}_j = \hat{c}|\underline{x}_t]P[M_j = l|\underline{x}_t]}{\sum_{t=1:C=c}^{N_{train}} P[M_j = l|\underline{x}_t]} \tag{6.5}$$

and

$$P_e[\hat{C}_j = \hat{c}|M_j = l, C = c] = \frac{\sum_{t=1}^{N_{test}} P_e[C = c|\underline{x}_t]P[\hat{C}_j = \hat{c}|\underline{x}_t]P[M_j = l|\underline{x}_t]}{\sum_{t=1}^{N_{test}} P[\hat{C}_j = \hat{c}|\underline{x}_t]P[M_j = l|\underline{x}_t]}. \tag{6.6}$$

Again, consistent with satisfying (6.4), we form a sum of cross entropies learning objective given by

$$L = \sum_{j=1}^{N_e} D(P_e[\hat{C}_j, M_j, C]||P_{con}^{(j)}[\hat{C}_j, M_j, C]) \tag{6.7}$$

and minimize over the test set posteriors via a gradient descent procedure.

6.3 Experimental Results

In this section we present the results comparative results among transductive IT method, its extension (Ext. I) and fixed combining rules (sum and product). We have not performed experiments with the second extension. Again, the datasets were taken from the UC Irvine repository. We chose $N_e = 5$ and used naive Bayes local classifiers (Gaussian, for continuous features and multinomial for discrete features). Diversity in the ensemble was created by randomly selecting a subset of features for each classifier. Prior mismatch was created by altering the test set – a new test set, with given priors,

was formed by sampling with replacement from the original test set. Prior mismatch is characterized by the sum of cross entropies measure

$$M = \sum_{j=1}^{N_e} \sum_{k=1}^{N_c} P_j[C = k] \log \left(\frac{P_j[C = k]}{P_{test}[C = k]} \right). \tag{6.8}$$

Here we focus on evaluating the IT extension. For Table 6.1, the results are based on 5 replications of two-fold cross validation. The IT extension I is compared with the original IT method and with arithmetic averaging (sum) and naive Bayes (product) combining (where there is no prior correction). Both IT methods give significant gains on some data sets for high prior mismatch ($M > 1.2$). Note also that the IT extension outperforms basic IT, with large gains in particular on *Segment* and *Ionosphere*.

Table 6.1. Generalization error rates based on 5 replications of two-fold cross validation, for IT, the IT extension(extension I), and for fixed sum and product methods.

Dataset	M	**IT**		**Fixed rules**	
		Ext. I	Orig.	Sum	Product
Diabetes	0.06	25.53	25.75	28.16	27.78
	0.36	24.04	26.09	28.27	28.61
	2.53	18.95	21.20	36.43	35.13
Liver Disorder	0.02	40.00	43.66	44.64	44.26
	0.62	33.47	34.10	39.02	39.13
	1.70	41.10	43.62	44.18	44.12
Sonar	0.05	26.06	27.69	30.38	28.46
	0.89	16.63	22.69	35.38	35.00
	1.35	24.90	26.35	27.40	28.46
Ionosphere	0.03	7.50	15.57	16.48	16.76
	0.26	7.78	14.03	16.17	16.06
	1.23	9.60	14.72	18.01	15.60
vehicle	0.01	23.29	37.86	46.40	47.88
	2.63	21.64	36.31	41.04	41.63
Segment	0.05	4.84	10.92	25.41	22.03
	2.57	9.37	25.69	41.61	42.45

Chapter 7

Conclusions

The focus of this dissertation is on three main areas 1) ME-IIS classification on continuous and mixed feature spaces, 2) ME-IIS extension to ensemble classification and 3) Distributed ensemble learning for the case where supervised learning is not applicable. Experiments have demonstrated that using the *soft quantization* information from continuous features, rather than hard quantizing, results in better classification accuracy. We successfully encoded the latent variables in ME-IIS framework for ensemble learning. Both our extension for supervised ensemble learning perform better than standard Adaboost. We also proposed transductive learning methods that outperform fixed combining rule when the training data is not available. The most of the material in this dissertation has been taken from our publications (journal and conference). There are some extensions of our work that can be considered in future work:

- We have considered ME-IIS for the case of probabilistic feature instances in order to address classification on mixed spaces and decision/data aggregation in ensemble classification. The experimental results have validated this technique in comparison with alternative approaches for both problems. Future work could better explore the space of possible constraints, e.g. using the algorithm in [1] to greedily build up more complex models, with higher order constraints.

- Another possible direction for ME-IIS learning would be to jointly perform feature selection and classification on continuous and mixed feature spaces. One method to address this will be starting with no features in the model and then evaluate N_f (number of features) model with each model consisting of only one of the features. Then select the model which achieves the best log-likelihood and then again repeat the procedure with the selected feature removed from the pool. This procedure can be continued till there is no more improvement. This is clearly a computationally intensive procedure. The challenge is to come up with a principled way which is fast and not too computationally intensive.

- Extending distributed ensemble classification problem to clustering and regression would also be a possible future direction. There is also the potential of applying the transductive techniques described here to applications in sensor networks and biomedical informatics. In both domains, there are multiple "sensing" modalities available and situations where only separate training data sets are available. In such cases, our methods should be exploited to good effect. Another possible way to go would be evaluating the proposed extensions of transductive learning and finding relevant application areas of such methods.

- Another interesting future research direction for distributed ensemble classification would be to come up with decision rule when there is some (very little) common labeled data.

Appendix A

Lagrangian Updates for Continuous and Mixed Spaces

A.1 Lagrange Multiplier Updates for Probabilistic Instances

For an arbitrary parameter vector $\underline{\gamma}$, the conditional log-likelihood is

$$L(\underline{\gamma}) = \sum_{t=1}^{T} \ln \left[\frac{e^{\sum_{i=1}^{N_d} \sum_{j \in \mathcal{A}_i} P[F_i=j|t]\gamma(C=c^{(t)},F_i=j)}}{\sum_{k=1}^{N_c} e^{\sum_{i=1}^{N_d} \sum_{j \in \mathcal{A}_i} P[F_i=j|t]\gamma(C=k,F_i=j)}} \right] \tag{A.1}$$

For a change in the parameter vector $\underline{\Delta\gamma}$, the change in log-likelihood is $L(\underline{\gamma}+\underline{\Delta\gamma})-L(\underline{\gamma})$. Using the identity $-\ln(\alpha) \geq 1-\alpha$, we obtain the lower bound

$$L(\underline{\gamma}+\underline{\Delta\gamma}) - L(\underline{\gamma}) \geq \sum_{t=1}^{T} \left[\sum_{i=1}^{N_d} \sum_{j \in \mathcal{A}_i} P[F_i=j|t]\Delta\gamma(C=c^{(t)},F_i=j) + 1 \right]$$
$$- \sum_{t=1}^{T} \sum_{k=1}^{N_c} P[C=k|P_{\underline{f}^{(t)}}] \times e^{\sum_{i=1}^{N_d} \sum_{j \in \mathcal{A}_i} P[F_i=j|t]\Delta\gamma(C=k,F_i=j)} \quad \widehat{=} \quad B(\underline{\Delta\gamma}|\underline{\gamma})$$

With $\underline{\Delta\gamma}$ chosen so that $B(\underline{\Delta\gamma}|\underline{\gamma}) \geq 0$ there is improvement in log-likelihood. The obvious approach then is to maximize $B(\underline{\Delta\gamma}|\underline{\gamma})$. However, there is coupled dependence in $B(\underline{\Delta\gamma}|\underline{\gamma})$ on the individual components $\{\Delta\gamma(C=c,F_i=j)\}$, which would necessitate

a complicated joint optimization. Thus, we seek an auxiliary function that decouples this dependence. We first rewrite

$$B(\underline{\Delta\gamma}|\underline{\gamma}) = \sum_{t=1}^{T}\left[\sum_{i=1}^{N_d}\sum_{j\in\mathcal{A}_i} P[F_i=j|t]\Delta\gamma(C=c^{(t)},F_i=j)+1\right] - \sum_{t=1}^{T}\sum_{k=1}^{N_c} P[C=k|P_{\underline{f}^{(t)}}] \times e^{\sum_{i=1}^{N_d}\sum_{j\in\mathcal{A}_i}\frac{P[F_i=j|t]}{N_d}[N_d\Delta\gamma(C=k,F_i=j)]} \quad \text{(A.2)}$$

Then, we note that $\sum_{i=1}^{N_d}\sum_{j\in\mathcal{A}_i}\frac{P[F_i=j|t]}{N_d} = 1$ i.e., $\left\{\frac{1}{N_d}P[F_i=j|t],\ i=1,...,N_d,\ j\in\mathcal{A}_i\right\}$ is an instance of the joint pmf $P[I,F_I]$, associated with first selecting a feature $i\in\{1,...,N_d\}$ and then a feature value $f_i\in\mathcal{A}_i$. Applying Jensen's inequality $e^{\sum_x p(x)q(x)} \leq \sum_x p(x)e^{q(x)}$ to the right hand side of (A.2), we have

$$B(\underline{\Delta\gamma}|\underline{\gamma}) \geq T + \sum_{t=1}^{T}\sum_{i=1}^{N_d}\sum_{j\in\mathcal{A}_i} P[F_i=j|t]\Delta\gamma(C=c^{(t)},F_i=j) - \frac{1}{N_d}\sum_{t=1}^{T}\sum_{k=1}^{N_c}\sum_{i=1}^{N_d}\sum_{j\in\mathcal{A}_i} P[C=k|P_{\underline{f}^{(t)}}]P[F_i=j|t]e^{N_d\Delta\gamma(C=k,F_i=j)} \quad \widehat{=} \quad A(\underline{\Delta\gamma}|\underline{\gamma}). \quad \text{(A.3)}$$

Thus $L(\underline{\gamma}+\underline{\Delta\gamma}) - L(\underline{\gamma}) \geq B(\underline{\Delta\gamma}|\underline{\gamma}) \geq A(\underline{\Delta\gamma}|\underline{\gamma})$, i.e., we have a new, not as tight, lower bound. Let $\underline{\Delta\gamma^*} = \arg\max_{\underline{\Delta\gamma}} A(\cdot)$. Since it is easy to verify that $A(\underline{0}|\underline{\gamma}) = 0$, it must be true that $A(\underline{\Delta\gamma^*}|\underline{\gamma}) \geq 0$, i.e., property A1 in section 2.3 is satisfied by this function. Moreover, $A(\underline{\Delta\gamma}|\underline{\gamma})$ can be additively decoupled into individual terms each depending on a single $\Delta\gamma(C=k,F_n=q)$. Differentiating $A(\underline{\Delta\gamma}|\underline{\gamma})$ with respect to $\Delta\gamma(C=k,F_n=q)$

and equating to 0 we get the choice of $\underline{\Delta\gamma}$ to maximize $A(\underline{\Delta\gamma}|\underline{\gamma})$, i.e.

$$\Delta\gamma^*(C=k,F_n=q) = \frac{1}{N_d}\ln\left[\frac{\sum\limits_{t=1;C=k}^{T} P[F_n=q|t]}{\sum\limits_{t=1}^{T} P[F_n=q|t]P\left[C=k|P_{\underline{f}^{(t)}}\right]}\right]$$
$$= \frac{1}{N_d}\ln\frac{P_g[C=k,F_n=q]}{P_m[C=k,F_n=q]},\ \forall k,\ n,\ q. \tag{A.4}$$

A.2 Proof of Constraint Satisfaction for Probabilistic Instances

Theorem 1. *Consider the function $A(\underline{\Delta\gamma}|\underline{\gamma})$ defined in (A.3). Let $\underline{\Delta\gamma^*} = \arg\max_{\underline{\Delta\gamma}} A(\underline{\Delta\gamma}|\underline{\gamma})$. Then, $A(\underline{\Delta\gamma^*}|\underline{\gamma}) = 0$ iff $\underline{\Delta\gamma^*} = \underline{0}$. When this occurs the constraints (2.4) are all met.*

Proof. Setting $\dfrac{\partial A(\underline{\Delta\gamma}|\underline{\gamma})}{\partial\Delta\gamma(F_i=j,C=c)} = 0$ gives the solution in equation (A.4). Moreover, this is a maximum since $\dfrac{\partial^2 A(\underline{\Delta\gamma}|\underline{\gamma})}{\partial\Delta\gamma^2(F_i=j,C=c)} < 0 \forall i,j,c.$

Thus, $\underline{\Delta\gamma^*} = \{\Delta\gamma^*(F_i=j,C=c)\ \forall i,\ \forall j\in\mathcal{A}_i,\ c\in C\}$.

Plugging the solution for $\underline{\Delta\gamma^*}$ back into $A(\underline{\Delta\gamma}|\underline{\gamma})$ in (A.3) and simplifying gives

$$A(\underline{\Delta\gamma^*}|\underline{\gamma}) = T + \sum_t^T\sum_i\sum_j\sum_c P[F_i=j|t]\frac{1}{N_d}\ln\left[\frac{\sum\limits_{t=1:c^{(t)}=c}^{T} P[F_i=j|t]}{\sum\limits_{t=1}^{T} P[F_i=j|t]P\left[C=c|P_{\underline{f}^{(t)}}\right]}\right]$$
$$- \frac{1}{N_d}\sum_i\sum_j\sum_{c=1}^{C}\sum_{t=1:c^{(t)}=c} P[F_i=j|t] \tag{A.5}$$

Noting that the third term equals T, we have

$$A(\underline{\Delta\gamma}^*|\underline{\gamma}) = \frac{1}{N_d}\sum_t^T\sum_i\sum_j\sum_c P[F_i=j|t]\ln\left[\frac{\sum\limits_{t=1:c^{(t)}=c}^{T} P[F_i=j|t]}{\sum\limits_{t=1}^{T} P[F_i=j|t]P\left[C=c|P_{\underline{f}^{(t)}}\right]}\right]$$

$$= -\frac{1}{N_d}\sum_t^T\sum_i\sum_j\sum_c P[F_i=j|t]\ln\left[\frac{\frac{1}{T}\sum\limits_{t=1}^{T} P[F_i=j|t]P\left[C=c|P_{\underline{f}^{(t)}}\right]}{\frac{1}{T}\sum\limits_{t=1:c^{(t)}=c}^{T} P[F_i=j|t]}\right]$$

Now, since $E(-\ln(\cdot)) \geq -\ln(E(\cdot))$ by Jensen's inequality, we have

$$A(\underline{\Delta\gamma}^*|\underline{\gamma}) \geq -\frac{T}{N_d}\sum_i \ln\left[\sum_{j,c}\frac{1}{T}\sum_{t=1:c^{(t)}=c}^{T} P[F_i=j|t]\left[\frac{\frac{1}{T}\sum\limits_{t'=1}^{T} P[F_i=j|t']P[C=c|P_{\underline{f}^{(t')}}]}{\frac{1}{T}\sum\limits_{t'=1:c^{(t')}=c}^{T} P[F_i=j|t']}\right]\right]$$

$$= -\frac{T}{N_d}\sum_i \ln\left[\sum_j\sum_c\frac{1}{T}\sum_{t'=1}^{T} P[F_i=j|t']P[C=c|P_{\underline{f}^{(t')}}]\right]$$

$$= -\frac{T}{N_d}\sum_i \ln(1) = 0$$

i.e., $A(\underline{\Delta\gamma}^*|\underline{\gamma}) \geq 0$. Finally by strict convexity of $\ln(\cdot)$, equality is achieved *iff* the

argument of the $\ln(\cdot)$ is 1. But this occurs *iff*

$$\sum_{t=1}^{T} P[F_i = j|t] P\left[C = c|P_{\underline{f}^{(t)}}\right] = \sum_{t=1;C=c}^{T} P[F_n = q|t] \ \forall i, j, c$$

i.e., by (A.4), *iff* $\underline{\Delta\gamma^*} = \underline{0}$. Clearly, when this occurs, the constraints are met. □

A.3 Lagrange Multiplier Updates for Mixed Spaces

The log-likelihood function for mixed spaces is given by

$$L(\Lambda) = \sum_{t=1}^{N} \log \left[\frac{e^{\sum_{i=1}^{N_d} \gamma(F_i=f_i{}^{(t)}, C=c^{(t)}) + \sum_{i=1}^{N_f}\sum_{j=1}^{L_i} P[M_i=j|a_i;\Lambda_i]\gamma(C=c^{(t)}, M_i=j)}}{\sum_{\hat{c}=1}^{N_c} e^{\sum_{i=1}^{N_d} \gamma(F_i=f_i{}^{(t)}, C=\hat{c}) + \sum_{i=1}^{N_f}\sum_{j=1}^{L_i} P[M_i=j|a_i;\Lambda_i]\gamma(C=\hat{c}, M_i=j)}} \right] \tag{A.6}$$

Where N_c denotes the number of classes, N_d is the number of discrete features, N_f is number of continuous features. Total number of training samples is given by T. L_i denotes number of mixture components for i^{th} continuous feature. For a change in the parameter vector $\underline{\Delta\gamma}$, the change in log-likelihood is

$L(\Lambda + \Delta) - L(\Lambda) =$

$$= \sum_{t=1}^{T}\sum_{i=1}^{N_d} \Delta\gamma(f_i{}^{(t)}, c^{(t)})) + \sum_{t=1}^{T}\sum_{i=1}^{N_f}\sum_{j=1}^{L_i} P[M_i = j|a_i;\Lambda_i]\Delta\gamma(c^{(t)}, M_i = j)$$

$$- \sum_{t=1}^{T} \log \left[\frac{\sum_{\hat{c}=1}^{N_c} e^{\sum_{i=1}^{N_d} (\gamma(f_i{}^{(t)}, \hat{c}) + \Delta\gamma(f_i{}^{(t)}, \hat{c})) + \sum_{i=1}^{N_f}\sum_{j=1}^{L_i} P[M_i=j|a_i;\Lambda_i](\gamma(f_i{}^{(t)}, \hat{c}) + \Delta\gamma(f_i{}^{(t)}, \hat{c}))}}{\sum_{\hat{c}=1}^{N_c} e^{\sum_{i=1}^{N_d} \gamma(f_i{}^{(t)}, \hat{c}) + \sum_{i=1}^{N_f}\sum_{j=1}^{L_i} P[M_i=j|a_i;\Lambda_i]\gamma(\hat{c}, M_i=j)}} \right] \tag{A.7}$$

Using the identity $-\log(\alpha) \geq 1-\alpha$, we find the lower bound on change in log-likelihood

$L(\Lambda+\Delta) - L(\Lambda) \geq$

$$= \sum_{t=1}^{T}\sum_{i=1}^{N_d} \Delta\gamma({f_i}^{(t)}, c^{(t)})) + \sum_{t=1}^{T}\sum_{i=1}^{N_f}\sum_{j=1}^{L_i} P[M_i = j|a_i;\Lambda_i]\Delta\gamma(c^{(t)}, M_i = j) + 1$$

$$- \sum_{t=1}^{T}\sum_{\hat{c}=1}^{N_c} \frac{e^{\sum_{i=1}^{N_d}(\gamma({f_i}^{(t)},\hat{c})+\Delta\gamma({f_i}^{(t)},\hat{c}))+\sum_{i=1}^{N_f}\sum_{j=1}^{L_i} P[M_i=j|a_i;\Lambda_i](\gamma({f_i}^{(t)},\hat{c})+\Delta\gamma({f_i}^{(t)},\hat{c}))}}{\sum_{\hat{c}=1}^{N_c} e^{\sum_{i=1}^{N_d}\gamma({f_i}^{(t)},\hat{c})+\sum_{i=1}^{N_f}\sum_{j=1}^{L_i} P[M_i=j|a_i;\Lambda_i]\gamma(\hat{c},M_i=j)}}$$

$$\widehat{=} \quad A(\underline{\Delta\gamma}|\underline{\gamma}) \quad \text{(A.8)}$$

i.e., $L(\Lambda+\Delta) - L(\Lambda) \geq A(\underline{\Delta\gamma}|\underline{\gamma})$

With $\Delta\gamma$ chosen so that $A(\underline{\Delta\gamma}|\underline{\gamma}) \geq 0$ there is improvement in log-likelihood. But the lagrange multipliers are coupled together. To decouple the lagrangians we rewrite the change in log-likelihood as

$$A(\underline{\Delta\gamma}|\underline{\gamma}) = \sum_{t=1}^{T}\sum_{i=1}^{N_d}\Delta\gamma(f_i{}^{(t)}, c^{(t)})) + \sum_{t=1}^{T}\sum_{i=1}^{N_f}\sum_{j=1}^{L_i} P[M_i = j|a_i;\Lambda_i]\Delta\gamma(c^{(t)}, M_i = j) + 1$$

$$- \sum_{t=1}^{T}\sum_{\hat{c}=1}^{N_c}\left[\frac{e^{\sum_{i=1}^{N_d}\gamma(f_i{}^{(t)},\hat{c})+\sum_{i=1}^{N_f}\sum_{j=1}^{L_i}P[M_i=j|a_i;\Lambda_i]\gamma(\hat{c},M_i=j)}}{\sum_{\hat{c}=1}^{N_c} e^{\sum_{i=1}^{N_d}\gamma(f_i{}^{(t)},\hat{c})+\sum_{i=1}^{N_f}\sum_{j=1}^{L_i}P[M_i=j|a_i;\Lambda_i]\gamma(\hat{c},M_i=j)}}\right] \cdot$$

$$e^{\sum_{i=1}^{N_d}\Delta\gamma(f_i{}^{(t)},\hat{c})+\sum_{i=1}^{N_f}\sum_{j=1}^{L_i}P[M_i=j|a_i;\Lambda_i]\Delta\gamma(\hat{c},M_i=j)} \quad \text{(A.9)}$$

The term between square brackets in equation (A.9) is $P\left[c|\underline{f},\underline{a}\right]$.

$$A(\underline{\Delta\gamma}|\underline{\gamma}) = \sum_{t=1}^{T}\sum_{i=1}^{N_d}\Delta\gamma(f_i{}^{(t)}, c^{(t)})) + \sum_{t=1}^{T}\sum_{i=1}^{N_f}\sum_{j=1}^{L_i} P[M_i = j|a_i;\Lambda_i]\Delta\gamma(c^{(t)}, M_i = j) + 1$$

$$- \sum_{t=1}^{T}\sum_{\hat{c}=1}^{N_c} P\left[c|\underline{f},\underline{a}\right] \cdot e^{\sum_{i=1}^{N_d}\Delta\gamma(f_i{}^{(t)},\hat{c})+\sum_{i=1}^{N_f}\sum_{j=1}^{L_i}P[M_i=j|a_i;\Lambda_i]\Delta\gamma(\hat{c},M_i=j)} \quad \text{(A.10)}$$

Rearranging the terms inside the exponent and applying the Jensen's inequality we get

$$A(\underline{\Delta\gamma}|\underline{\gamma}) \geq \sum_{t=1}^{T}\sum_{i=1}^{N_d}\Delta\gamma(f_i{}^{(t)}, c^{(t)})) + \sum_{t=1}^{T}\sum_{i=1}^{N_f}\sum_{j=1}^{L_i} P[M_i = j|a_i;\Lambda_i]\Delta\gamma(c^{(t)}, M_i = j) + 1$$

$$- \frac{1}{N_d+N_f}\sum_{t=1}^{T}\sum_{\hat{c}=1}^{N_c}\sum_{i=1}^{N_d} P\left[c|\underline{f},\underline{a}\right] \cdot e^{(N_d+N_f)\cdot\Delta\gamma(f_i{}^{(t)},\hat{c})}$$

$$- \frac{1}{N_d+N_f}\sum_{t=1}^{T}\sum_{\hat{c}=1}^{N_c}\sum_{i=1}^{N_f}\sum_{j=1}^{L_i} P[M_i = j|a_i;\Lambda_i] \cdot P\left[c|\underline{f},\underline{a}\right] \cdot e^{(N_d+N_f)\cdot\Delta\gamma(\hat{c},M_i=j)}$$

$$\widehat{=} \quad B(\underline{\Delta\gamma}|\underline{\gamma}) \quad \text{(A.11)}$$

$B(\underline{\Delta\gamma}|\underline{\gamma})$ is a new, not as tight, lower bound on the change of likelihood. That is, $L(\Lambda+\Delta) - L(\Lambda) \geq B(\underline{\Delta\gamma}|\underline{\gamma})$. Differentiating $B(\underline{\Delta\gamma}|\underline{\gamma})$ with respect to $\Delta\gamma({f_i}^{(t)}, \hat{c})$ and $\Delta\gamma(\hat{c}, M_i = j)$ we get

$$\frac{\partial B(\underline{\Delta\gamma}|\underline{\gamma})}{\partial \Delta\gamma({f_m}^{(t)} = p, C = k)} = \sum_{t=1; f_m^t = p, C=k}^{T} 1 - \sum_{t=1; F_m^t = p}^{T} P\left[k|\underline{f}, \underline{a}\right] \cdot e^{(N_d+N_f)\cdot\Delta\gamma(F_m=p, C=k)} \tag{A.12}$$

and

$$\frac{\partial B(\underline{\Delta\gamma}|\underline{\gamma})}{\partial \delta(C = k, M_n = q)} = \sum_{t=1; C=k}^{T} P[M_n = q|a_n; \Lambda_n] - \sum_{t=1}^{T} P[M_n = q|a_n; \Lambda_n] \cdot P\left[k|\underline{f}, \underline{a}\right] \cdot e^{(N_d+N_f)\cdot\Delta\gamma(C=k, M_n=q)}. \tag{A.13}$$

Equating equations (A.12) and (A.13) to 0, we obtain the updates as

$$\Delta\gamma({f_m}^{(t)} = p, C = k) = \frac{1}{(N_d+N_f)} \log\left[\frac{\sum_{t=1; f_m^t = p, C=k}^{T} 1}{\sum_{t=1; F_m^t = p}^{T} P\left[k|\underline{f}, \underline{a}\right]}\right] \tag{A.14}$$

and

$$\Delta\gamma(C = k, M_n = q) = \frac{1}{(N_d+N_f)} \log\left[\frac{\sum_{t=1; C=k}^{T} P[M_n = q|a_n; \Lambda_n]}{\sum_{t=1}^{T} P[M_n = q|a_n; \Lambda_n] \cdot P\left[k|\underline{f}, \underline{a}\right]}\right] \tag{A.15}$$

Appendix B

EM Framework for Transductive ML Techniques

For both transductive ML techniques, the data treated as missing within the EM framework is the class label for each new batch sample, i.e. let $Z_{ik} = 1$ if sample $\underline{\mathbf{x}}_i$ (exclusively) belongs to class k, and $Z_{ik} = 0$, otherwise. Then, for the independence case of section 3.1, the *complete data log-likelihood* is

$$\log L_c(\mathcal{X}_{\text{test}}) = \sum_{i=1}^{N_{\text{test}}} \sum_{k=1}^{N_c} Z_{ik} (\log P_e[C = k] + \sum_{j=1}^{N_c} \log p_j[\underline{x}_i^{(j)} | C = k; \Lambda_{jk}]). \tag{B.1}$$

Here, the parameters that specify the class-conditional models, Λ_{jk}, even though in general unknown, are introduced for completeness. In the E-step, we compute the expected value of $\log L_c(\mathcal{X}_{\text{test}})$ given the current parameter estimates, i.e. given $\Lambda \equiv \{\{P_e[C = k]\}, \{\Lambda_{jk}\}\}$. This is

$$E[\log L_c(\mathcal{X}_{\text{test}})|\Lambda] = \sum_{i=1}^{N_{\text{test}}} \sum_{k=1}^{N_c} E[Z_{ik}|\Lambda] (\log P_e[C = k] + \sum_{j=1}^{N_e} \log p_j[\underline{x}_i^{(j)} | C = k; \Lambda_{jk}]). \tag{B.2}$$

The expectations are just the ensemble class *a posteriori* probabilities, which can be derived under the feature vector independence assumption, i.e.

$$\begin{aligned} E[Z_{ik}|\Lambda] \equiv P_e[C=k|\underline{\mathbf{x}}_i] &= \frac{P_e[C=k]\prod_{j=1}^{N_e} p_j[\underline{x}_i^{(j)}|C=k]}{Z} \\ &= \frac{P_e[C=k]\prod_{j=1}^{N_e} \frac{P_j[C=k|\underline{x}_i^{(j)}]}{P_j[C=k]}}{Z'}, \end{aligned} \tag{B.3}$$

where Z and Z' are the proper normalizations, ensuring a valid probability mass function. The E-step is thus specified by (4.2). In the M-step we maximize (B.2) over the parameters to be adjusted, in this case $\{P_e[C=k]\}$, given the E-step probabilities held fixed. This gives the update in (4.3).

Likewise, for the ML problem in section 3.2, we can derive the expected complete data censored log likelihood

$$E[\log L_{\mathrm{c-cens}}(\mathcal{X}_{\mathrm{test}})|\Lambda] =$$

$$\sum_{i=1}^{N_{\mathrm{test}}}\sum_{k=1}^{N_c} E[Z_{ik}|\Lambda](\log P_e[C=k] + \log(\sum_{j=1}^{N_e}\frac{1}{N_e}p_j[\underline{x}_i^{(j)}|C=k;\Lambda_{jk}])). \tag{B.4}$$

In this case, the E-step probabilities are derived via

$$
\begin{aligned}
E[Z_{ik}|\Lambda] = P_e[C=k|\underline{\mathbf{x}}_i] &= \sum_{j=1}^{N_e} P_e[C=k|\underline{x}_i, j \text{ uncensored}](\frac{1}{N_e}) \qquad \text{(B.5)} \\
&= \frac{1}{Z}\sum_{j=1}^{N_e} p_j[\underline{x}_i^{(j)}|C=k]P_e[C=k] \\
&= \frac{1}{Z'}\sum_{j=1}^{N_e} \frac{P_j[C=k|\underline{x}_i^{(j)}]}{P_j[C=k]} P_e[C=k].
\end{aligned}
$$

The E-step is thus specified by (4.5). Maximizing (B.4) with respect to $\{P_e[C=k]\}$ given fixed E-step probabilities again gives (4.3).

Appendix C

An Example Showing the Data Form

In this appendix we give a example showing a form of data used by classifying algorithm[1]. This example is taken form handwritten recognition problem. The task involves identifying the digits from 0 to 9. These digits are in form of binary images of dimension 16x16. The feature extraction procedure is used to extract meaningful features which can be used by the classifying algorithm to predict the class label (in this case the class labels are digits 0 to 9). For our example, all the pixel values arranged in one dimensional array (of size 1x256) were used as feature vector. This array is then given as input to the classifier which in turn predicts the class label.

[1]The image for image data is taken from http://www.cedar.buffalo.edu/Databases/CDROM1/digits.html

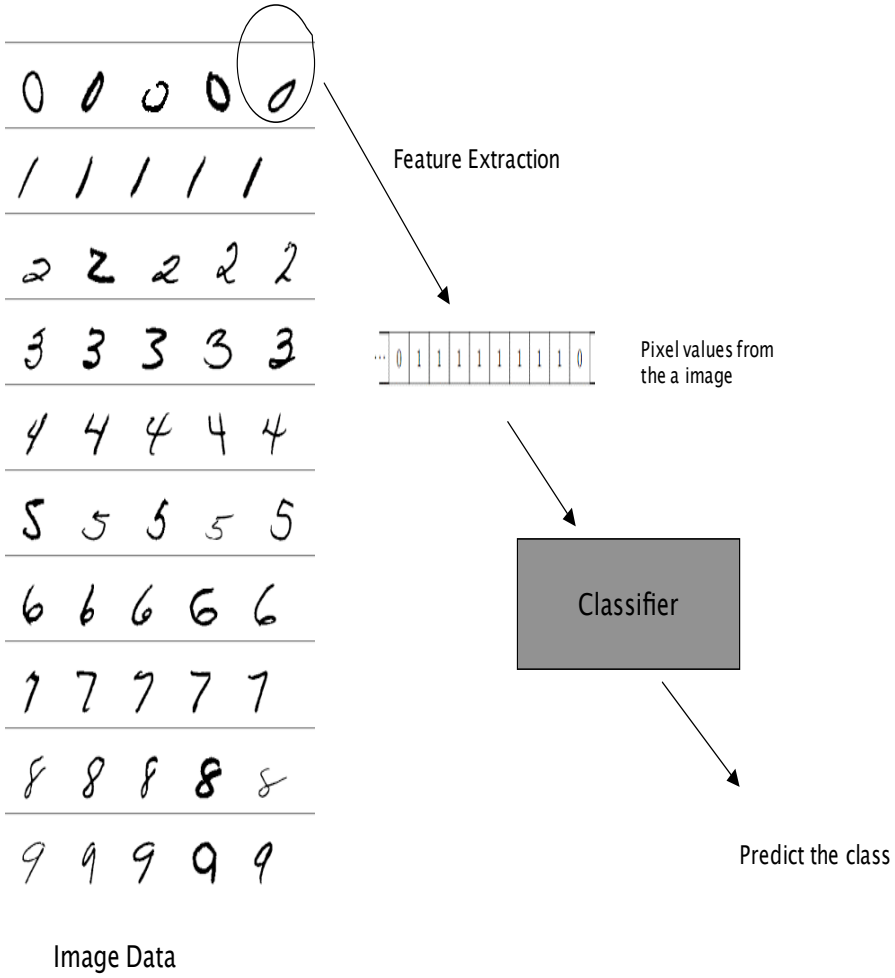

Fig. C.1. An example showing a form of data used for classification

Bibliography

[1] Stephen Della Pietra, Vincent J. Della Pietra, and John D. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

[2] Adam L. Berger, Stephen Della Pietra, and Vincent J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[3] Steven J. Phillips, Miroslav Dudìk, and Robert E. Schapire. A maximum entropy approach to species distribution modeling. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 83, 2004.

[4] Jiwoon Jeon and R. Manmatha. Using maximum entropy for automatic image annotation. In *CIVR*, pages 24–32, 2004.

[5] E. T. Jaynes. On the rationale of maximum-entropy methods. *Proc. of IEEE*, 70(9):939–952, 1982.

[6] E. T. Jaynes. *Papers on probability, statistics and statistical physics.* Reidel, Dordrecht, 1982.

[7] Raymond Lau, Ronald Rosenfeld, and Salim Roukos. Adaptive language modeling using the maximum entropy principle. In *HLT '93: Proceedings of the workshop on Human Language Technology*, pages 108–113, 1993.

[8] J.N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *Ann. Math. Stat.*, 43:1470–1480, 1972.

[9] D. J. Miller and L. Yan. An approximate maximum entropy method for classification and more general inference: relation to other maxent methods and to naive bayes. In *CISS*, 2000.

[10] L. Yan and D. J. Miller. General statistical inference for discrete and mixed spaces by an approximate application of the maximum entropy principle. *IEEE Transactions on Neural Networks*, 11(3):558–573, 2000.

[11] Nir Friedman, Moises Goldszmidt, and Thomas J. Lee. Bayesian network classification with continuous attributes: getting the best of both discretization and parametric fitting. In *Proc. 15th International Conf. on Machine Learning*, pages 179–187. Morgan Kaufmann, San Francisco, CA, 1998.

[12] Yasser Abdel-Haleem, Steve Renals, and Neil Lawrence. Acoustic space dimensionality selection and combination using the maximum entropy principle. In *Proc. IEEE ICASSP*, 2004.

[13] S. Wang, D. Schuurmans, and Y. Zhao. The latent maximum entropy principle, 2002.

[14] Ron Meir, Ran El-Yaniv, and Shai Ben-David. Localized boosting. In *Proc. 13th Annu. Conference on Comput. Learning Theory*, pages 190–199, 2000.

[15] R. Jin, Y. Liu, L. Si, J. Carbonell, and A. Hauptmann. A new boosting algorithm using input-dependent regularizer. In *Proc. of Twentieth International Conference on Machine Learning (ICML'03)*, 2003.

[16] Michael Collins, Robert E. Schapire, and Yoram Singer. Logistic regression, adaboost and bregman distances. In *Computational Learing Theory*, pages 158–169, 2000.

[17] Jayanta Basak and Ravi Kothari. A classification paradigm for distributed vertically partitioned data. *Neural Computation*, 16(7):1525–1544, 2004.

[18] D. J. Miller and L. Yan. Critic-driven ensemble classification. *IEEE Transactions Signal Processing*, 11(3):558–573, 2000.

[19] A. D'Costa, V. Ramachandran, and A. Sayeed. Distributed classification of gaussian space-time sources in wireless sensor networks. *IEEE Journal on Selected Areas in Communications*, 22(6):1026–1036, 2004.

[20] Marco Saerens, Patrice Latinne, and Christine Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. *Neural Computation*, 14(1):21–41, 2002.

[21] J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. *Uncertainty in Artificial Intelligence, Proceedings of the Fourteenth Conference, Morgan Kaufman*, pages 43–52, 1998.

[22] G. Lebanon and J. Lafferty. Boosting and maximum likelihood for exponential models. *NIPS*, 2002.

[23] S. Chen and R. Rosenfeld. A survey of smoothing techniques for me models. *IEEE Transactions on Speech and Audio Processing*, 8(1):37–50, 2000.

[24] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[25] A. Berger. The improved iterative scaling algorithm: A gentle introduction, 1997.

[26] Ron Kohavi and Mehran Sahami. Error-based and entropy-based discretization of continuous features. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 114–119, 1996.

[27] Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[28] Padhraic Smyth. Clustering using monte carlo cross-validation. In *Knowledge Discovery and Data Mining*, pages 126–133, 1996.

[29] John Browning and David J. Miller. A maximum entropy approach for collaborative filtering. *J. VLSI Signal Process. Syst.*, 37(2-3):199–209, 2004.

[30] Tjen-Sien Lim, Wei-Yin Loh, and Yu-Shan Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Mach. Learn.*, 40(3):203–228, 2000.

[31] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(10):993–1001, 1990.

[32] Kagan Tumer and Joydeep Ghosh. Error correlation and error reduction in ensemble classifiers. *Connection Science*, 8(3-4):385–403, 1996.

[33] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[34] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[35] Eun Bae Kong and Thomas G. Dietterich. Error-correcting output coding corrects bias and variance. In *International Conference on Machine Learning*, pages 313–321, 1995.

[36] I. Lisiecki, F. Billoudet, M. P. Pileni, H. Kang, K. Kim, and J. Kim. Optimal approximation of discrete probability distribution with $k^{th}-$order dependency and its application to combining multiple classifiers. *Pattern Recognition Letters*, 18, June 1997.

[37] L. Xu, A. Krzyzak, and C. Y. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 22(3):418–435, 1992.

[38] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[39] K. Tumer and J. Ghosh. Robust combining of disparate classifiers through order statistics. *Pattern Analysis and Applications*, 5(2):189–200, 2002.

[40] Ke Chen, Lan Wang, and Huisheng Chi. Methods of combining multiple classifiers with different features and their applications to text-independent speaker identification. *IJPRAI*, 11(3):417–445, 1997.

[41] Josef Kittler, Mohamad Hatef, Robert P.W. Duin, and Jiri Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[42] N. Ueda and R. Nakano. Combining discriminant-based classifers using the minimum classification error discriminant. *IEEE Conf. Neural Networks for Signal Processing*, pages 365–374, 1997.

[43] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

[44] 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(9):2175–2207, 2000.

[45] L. Rabiner and B. H. Juang. *Fundamentals of speech recognition*. Prentice Hall, 1993.

[46] A. Schwaighofer. Svm toolbox for matlab.

[47] Ethem Alpaydin. Combined 5 x 2 cv f test for comparing supervised classification learning algorithms. *Neural Computation*, 11(8):1885–1892, 1999.

[48] David Landgrebe. Multispectral land sensing; where from, where to? *IEEE Transactions on Geoscience and Remote Sensing*, 43(3), 2005.

[49] Thorsten Joachims. Transductive inference for text classification using support vector machines. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 200–209, Bled, SL, 1999. Morgan Kaufmann Publishers, San Francisco, US.

[50] D. J. Miller and H. S. Uyar. A mixture of experts classifier with learning based on both labeled and unlabeled data. *Neural Information Processing Systems Conf.*, 9:571–577, 1997.

[51] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[52] G. McLachlan and D. Peel. *Finite mixture models*. New York: Wiley, 2000.

[53] R. Redner and H. Walker. Mixture densities, maximum likelihood, and the em algorithm. *SIAM Review*, 26(2):195–239, 1984.

[54] D. J. Miller and S. Pal. An extension of iterative scaling for joint decision-level and feature-level fusion in ensemble classification. In *IEEE Intl. Workshop Machine Learning for Signal Processing*, 2005.

[55] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. New York: Wiley, 1991.

[56] Souheil Ben-Yacoub, Yousri Abdeljaoued, and Eddy Mayoraz. Fusion of face and speech data for person identity verification. *IEEE Transactions on Neural Networks*, 10(05):1065–1074, 1999.

[57] Patrick Verlinde, Gerard Chollet, and Marc Acheroy. Multi-modal identity verification using expert fusion. *Information Fusion*, 1(1):17–33, 2000.

[58] Stèphane Pigeon and Luc Vandendorpe. Image-based multimodal face authentication. *Signal Process.*, 69(1):59–79, 1998.

[59] Arun Ross and Anil Jain. Information fusion in biometrics. *Pattern Recogn. Lett.*, 24(13):2115–2125, 2003.

[60] Lin Hong and Anil Jain. Integrating faces and fingerprints for personal identification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(12):1295–1307, 1998.

[61] B. Duc, E. S. Bigun, J. Bigun, G. Maitre, and S. Fischer. Fusion of audio and video information for multi modal person authentication. *Pattern Recognition Letters*, 18:835–843, 1997.

[62] Roberto Brunelli and Daniele Falavigna. Person identification using multiple cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(10):955–966, 1995.

[63] V. Popovici, J. Thiran, E. Bailly-Bailliere, S. Bengio, F. Bimbot, M. Hamouz, J. Kittler, J. Mariethoz, J. Matas, K. Messer, B. Ruiz, and F. Poiree. The BANCA Database and Evaluation Protocol. In *4th International Conference on Audio- and Video-Based Biometric Person Authentication, Guildford, UK*, volume 2688, pages 625–638. SPIE, 2003.

[64] J. Luttin. Evaluation protocol for the xm2fdb database (lausanne protocol), 1998.

[65] Sonia Garcia-Salicetti, Charles Beumier, Gérard Chollet, Bernadette Dorizzi, Jean Leroux les Jardins, Jan Lunter, Yang Ni, and Dijana Petrovska-Delacrétaz. Biomet: A multimodal person authentication database including face, voice, fingerprint, hand and signature modalities. In *AVBPA*, pages 845–853, 2003.

[66] Norman Poh and Samy Bengio. Database, protocol and tools for evaluating score-level fusion algorithms in biometric authentication. In *Fifth Int'l. Conf. Audio- and Video-Based Biometric Person Authentication AVBPA*, 2005.

[67] Ahmed Al-Ani and M. Deriche. A new technique for combining multiple classifiers using the Dempster–Shafer theory of evidence. *Journal of Artificial Intelligence Research*, 17:333–361, 2002.

[68] D. Bahler and L. Navarro. Methods for combining heterogeneous sets of classifiers. In *17th Natl. Conf. on Artificial Intelligence (AAAI)*, 2000.

[69] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *COLT' 98: Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, 1998.

[70] R. S. Blum, S. A. Kassam, and H. V. Poor. Distributed detection with multiple sensors. *Proc. of the IEEE*, pages 64–79, 1997.

[71] Rich Caruana. Learning many related tasks at the same time with backpropagation. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 657–664. The MIT Press, 1995.

[72] Gregory F. Cooper and Edward Herskovits. A bayesian method for the induction of probabilistic networks from data. *Mach. Learn.*, 9(4):309–347, 1992.

[73] M.A.T. Figueiredo and A.K. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):381–396, 2002.

[74] Robert Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *COLING-02: proceeding of the 6th conference on Natural language learning*, pages 1–7, 2002.

[75] L. Yan and D. J. Miller. Critic-driven ensemble classification via a learning method akin to boosting. *Intelligent Engineering Systems Through Artificial Neural Networks 11*, pages 27–32, 2001.

Vita

EDUCATION

- Ph.D. in E.E., The Pennsylvania State University 08/03 - 12/06
- M.S. in E.E., The Pennsylvania State University 08/01 - 06/04
- B.E. in E.E., I.I.T Roorkee 08/96 - 05/00

PUBLICATION

Journal Papers

1. Transductive Methods for the Distributed Ensemble Classification Problem (D. J. Miller and S. Pal). Accepted for *Neural Computation*
2. An Extension of Iterative Scaling for Decision and Data Aggregation in Ensemble Classification (S. Pal and D. J. Miller). Accepted for *Journal of VLSI Signal Processing Systems*

Conference Papers

1. A Latent Variable Extension of Iterative Scaling for Classification on Continuous and Mixed Continuous-Discrete Feature Spaces (D. J. Miller and S. Pal). *CISS* 2005
2. An Extension of Iterative Scaling for Joint Decision-Level and Feature-Level Fusion in Ensemble Classification (D. J. Miller and S. Pal). *MLSP* , 2005
3. Transductive Methods for Distributed Ensemble Classification (D. J. Miller and S. Pal). *CISS*, 2006
4. Constraint-based, Transductive Learning for Distributed Ensemble Classification (D. J. Miller, S. Pal and Y. Wang). *MLSP*, 2006
5. Robust Optimization Strategies for Adaptive Filters Operating with Fixed and Transient Hardware Errors (S. Pal and W. K. Jenkins). *Asilomar Conference on Signals, Systems, and Computers*, 2005
6. Structured Stochastic Optimization Strategies for Problems with Ill-conditioned Error Surfaces (S. Pal, D. J. Krusienski and W. K. Jenkins). *ISCAS* 2005
7. Constrained Random Search for IIR Adaptive and Non-Linear Filters (S. Pal and W. K. Jenkins). *MWSCAS*, 2005