Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2008, Article ID 674974, 21 pages
doi:10.1155/2008/674974

Research Article

Decision Aggregation in Distributed Classification by a Transductive Extension of Maximum Entropy/Improved Iterative Scaling

David J. Miller, Yanxin Zhang, and George Kesidis

Department of Electrical Engineering, The Pennsylvania State University, University Park, PA 16802, USA

Correspondence should be addressed to David J. Miller, djmiller@engr.psu.edu

Received 28 September 2007; Revised 28 January 2008; Accepted 4 March 2008

Recommended by Sergios Theodoridis

In many ensemble classification paradigms, the function which combines local/base classifier decisions is learned in a supervised fashion. Such methods require common labeled training examples across the classifier ensemble. However, in some scenarios, where an ensemble solution is *necessitated*, common labeled data may not exist: (i) legacy/proprietary classifiers, and (ii) spatially distributed and/or multiple modality sensors. In such cases, it is standard to apply fixed (*untrained*) decision aggregation such as voting, averaging, or naive Bayes rules. In recent work, an alternative *transductive* learning strategy was proposed. There, decisions on test samples were chosen aiming to satisfy constraints measured by each local classifier. This approach was shown to reliably correct for class prior mismatch and to robustly account for classifier dependencies. Significant gains in accuracy over fixed aggregation rules were demonstrated. There are two main limitations of that work. First, feasibility of the constraints was not guaranteed. Second, heuristic learning was applied. Here, we overcome these problems via a transductive extension of *maximum entropy/improved iterative scaling* for aggregation in distributed classification. This method is shown to achieve improved decision accuracy over the earlier transductive approach and fixed rules on a number of UC Irvine datasets.

1. INTRODUCTION

There has been a great deal of research on techniques for building ensemble classification systems, (e.g., [1–10]). Ensemble systems form ultimate decisions by aggregating (hard or soft) decisions made by individual classifiers. These systems are usually motivated by biases associated with various choices in classifier design [11]: the features, statistical feature models, the classifier's (parametric) structure, the training set, the training objective function, parameter initialization, and the learning algorithm for minimizing this objective. Poor choices for any subset of these design elements can degrade classification accuracy. Ensemble techniques introduce diversity in these choices and thus mitigate biases in the design. Ensemble systems have been theoretically justified from several standpoints, including, under the assumption of statistical independence [12], variance and bias reduction [9, 10], and margin maximization [8]. In most prior research, an ensemble solution has been chosen at the designer's *discretion* so as to improve performance. In paradigms such as *boosting* [5], all the classifiers are generated using the same training set. This training set could have simply been used to build a single (high complexity) classifier. However, boosted ensembles have been shown in some prior works to yield better generalization accuracy than single (standalone) classifiers [13].

In this work, we alternatively consider scenarios where, rather than discretionary, a multiple classifier architecture is *necessitated* by the "distributed" nature of the feature measurements (and associated training data) for building the recognition system [1, 14, 15]. Such applications include: (1) *classification over sensor networks*, where multiple sensors separately obtain measurements from the same object or phenomenon to be classified, (2) *legacy or proprietary systems*, where multiple proprietary systems are leveraged to build an ensemble classifier and (3) *classification based on multiple sensing modalities*, for example, vowel recognition using acoustic signals and video of the mouth [16] with separate classifiers for each modality, or disease classification based on separate microarray and clinical classifiers. In each of these scenarios, it is necessary to build an ensemble solution. However, unlike the standard ensemble setting, in the scenarios above, each classifier may only have its own *separate* training resources, that is, there may be no common labeled training examples across all (or even any subset of) the classifiers. Each classifier/sensor may in fact not have any training resources at all—each sensor could simply use an a priori known class-conditional density model for its feature measurements, with a "plug-in" Bayes classification rule applied. We will refer to this case, of central interest in this paper, as the *distributed classification problem*.

This problem has been addressed before, both in its general form (e.g., [1]) and for classification over sensor networks (e.g., [14]). Both [1, 14] developed fixed combining rule techniques. In [1], Bayes rule decision aggregation was derived accounting for redundancies in the features used by the different classifiers. This approach requires communication between local classifiers to identify the features they hold in common. In [14], fixed combining was derived under the assumption that feature vectors of the local classifiers are jointly Gaussian, with known correlation structure over the joint feature space (i.e., across the local classifiers). Neither these methods nor other past methods for distributed classification have considered *learning* the aggregation function. The novel contribution of [15] was the application and development of suitable *transductive learning* techniques [17–19], with learning based on the unlabeled test data, for optimized decision aggregation in distributed classification. In this work, we extend and improve upon the transductive learning framework from [15].

Common labeled training examples across local classifiers are needed if one is to jointly train the local classifiers in a supervised fashion, as done, for example, in boosting [5] and mixture of experts [20]. Common labeled training data is also needed if one is to learn, in a supervised fashion, the function which aggregates classifier decisions [7, 21–23]. These approaches treat local classifier hard/soft decisions as the input features to a second-stage classifier (the ensemble's aggregation function). Learning this second stage in a supervised fashion can *only* be achieved if there is a pool of common labeled training examples where, for each labeled instance, there is a realization of each local classifier's input feature vector (based upon which each local classifier can produce a hard/soft decision).

Consider legacy/proprietary systems. Multiple organizations may build separate recognition systems using "in-house" data and proprietary designs. The government or some other entity would like to leverage all the resulting systems (i.e., fuse decisions) to achieve best accuracy. Thus an ensemble solution is needed, but unless organizations are willing to share data, there will be no common labeled data for learning how to best aggregate decisions. Alternatively, if organization A shares its design method (features used, classifier structure, and learning method) with organization B, then B can build a version of A's classifier using B's data and then further use this data as a common labeled resource for supervised learning of an aggregation function.

As a second example, consider diagnosis for a much-studied disease. Different institutions may publish studies, each evaluating their own test biomarkers for predicting disease presence. Each study will have its own (labeled) patient pool, from which a classifier could be built (working on the study's biomarker features). If each study measured different features, for different patient populations, it is not possible to pool the datasets to create a common pool of labeled examples. Now, suppose there is a clinic with a population of new patients to classify. The clinic would like to leverage the biomarkers (and associated classifiers) from each of the studies in making decisions for its patients. This again amounts to distributed classification without common labeled training examples.

In all of these cases, without common labeled training data, the conventional wisdom is that one *must* apply a fixed (untrained) mathematical rule such as voting [12], voting with abstention mechanisms [24], fixed arithmetic averaging [25], or geometric averaging; Bayes rule [26]; a Bayesian sum rule [27]; or other fixed rules [3] in fusing individual classifier decisions. Fixed (untrained) decision aggregation also includes methods that weight the local classifier decisions [28] or even select a single classifier to rely on [29] in an input-dependent fashion, based on each classifier's local error rate estimate or local confidence. Such approaches do give input-dependent weights on classifier combination. However, the weights are heuristically chosen, separately by each local classifier. They are not jointly trained/learned to minimize a common mathematical objective function. In this sense, we still consider [28, 29] as fixed (untrained) forms of decision aggregation. Alternatively, in [15], it was shown that one can still beneficially *learn* a decision aggregation function, that is, one can jointly optimize test sample-dependent weights of classifier combination to minimize a *well-chosen* cost function and significantly outperform fixed aggregation rules. A type of *transductive learning* strategy [17–19] was proposed [15], wherein optimization of a *well-chosen* objective function measured over test samples directly *yields* the decisions on these samples. This work built on [18], which applied transductive learning to adapt class priors while making decisions in the case of a single classifier. While there is substantial separate literature on transductive/semisupervised learning and on ensemble/distributed classification, the novel contribution in [15] was the bridging of these areas via the application of transductive learning to decision aggregation in distributed classification.

There are two fundamental deficiencies of fixed combining which motivated the approach in [15]. First, local classifiers might assume incorrect class prior probabilities [15], relative to the priors reflected in the test data [18]. There are a number of reasons for this prior mismatch, for example, it may be difficult or expensive to obtain training examples from certain classes, (e.g., rare classes); also, classes that are highly confusable are not easily labeled and, thus, may not be adequately represented in a local training set. Prior mismatch can greatly affect fused decision accuracy. Second, there may be statistical dependencies between the decisions produced by individual classifiers. Fixed voting and averaging both give biased decisions in this case [30] and may yield very poor accuracy. This was demonstrated in [15] considering the case where some classifiers are *perfectly redundant*, that is, identical copies of each other. Suppose that in the ensemble there are a large number of identical copies of an inaccurate classifier and only a single highly accurate classifier. Clearly, the weak classifiers will dominate a single accurate classifier in a voting or averaging scheme, yielding biased, inaccurate ensemble decisions. Standard distributed detection techniques—which make the *naive Bayes* assumption that measurements at different sensors are independent given the class [31]—will also fare poorly when there is sensor dependency/redundancy. More localized schemes (e.g., [29]) can mitigate "dominance of the majority" in an ensemble, giving the most relevant classifiers (even if a small minority) primary influence on the ensemble decision making in a local region of the feature space. However, these methods are still vulnerable to the first-mentioned problem of class prior mismatch. In [2, 4], ensemble construction methods were also proposed that reduce correlation within the ensemble while still achieving good accuracy for the individual local classifiers. However, these methods require availability of a common labeled training set and/or common features for building the local classifiers.

Alternatively, [15] proposed a transductive, *constraint-based* (CB) method that optimizes decision aggregation without common labeled training data. CB resolves both afore-mentioned difficulties with fixed combining: in making fused decisions, it effectively *corrects for* inaccurate local class priors; moreover, it accounts for dependencies between classifiers and does so without *any* communication between local classifiers. In CB, each local classifier contributes statistical constraints that the aggregation function must satisfy through the decisions it makes on test samples. The constraints amount to local classifier "confusion matrix" information—the probability that a local classifier chooses class k given that the true class is c. The aggregation function is learned so that the confusion statistic between the aggregation function's predicted class c and a local classifier's predicted class k matches the confusion statistic between the *true* class c and the local classifier's predicted class k. Constraint-based learning is quite robust in the presence of classifier dependency/redundancy—if local classifiers A and B are perfectly redundant (i.e., if B yields an identical classification rule as A), then *so are their constraints*. Thus, if the aggregation function is learned to satisfy A's constraints, B's are *automatically* met as well—B's constraints will not alter the aggregation function solution, and the method is thus invariant to (perfectly) redundant classifiers in the ensemble. More generally, CB well handles statistical dependencies between classifiers, giving greater decision accuracy than fixed rule (and several alternative) methods [15].

Some of the key properties of CB are as follows [15]: (1) it is effective whether classifiers produce soft or *hard* decisions—the method (implicitly) compensates local classifier posteriors for inaccurate priors even when the local classifiers only produce hard decisions (to *explicitly* correct a local classifier for incorrect class priors, one must have access to the local class posteriors, not just to the hard decision output by the local classifier; e.g., the method in [18] performs explicit prior correction and thus requires access to soft classifier decisions); (2) CB works when local classifiers are weak (simple sensors) or strong (sophisticated classifiers, such as support vector machines); (3) CB gives superior results to fixed combining methods in the presence of classifier dependencies; (4) CB robustly and accurately handles the case where some classes are *missing* in the test data, whereas fixed combining methods perform poorly in this case; (5) CB is easily extended to encode auxiliary sensor/feature information, nonredundant with local classifier decisions, to improve the accuracy of the aggregation [32]. The original method required making decisions *jointly* on a *batch* of test samples. In some applications, sample-by-sample decisions are needed. In particular, if decisions are time-critical (e.g., target detection) and in applications where decisions require a simple explanation (e.g., credit card approval). Recently, a CB extension was developed that makes (sequential) decisions, sample by sample [33].

There are, however, limitations of the heuristic learning applied in [15]. First, in [15], there was no assurance of feasibility of the constraints because the local classifier training set support (on which constraints are measured) and the test set support (on which constraints are *met* by the aggregation function) are different. In the experiments in [15], constraints were found to be closely approximated. However, infeasibility of constraints could still be a problem in practice. Second, constraint satisfaction in [15] was practically effected by minimizing a particular nonnegative cost function (a sum of cross entropies). When, and only when, the cost is *zeroed*, the constraints are met. However, the cost function in [15] is nonconvex in the variables being optimized, with, thus, potential for finding positive (nonzero) local minima, for which the constraints are necessarily not met. Moreover, even in the feasible case, there is no *unique* feasible (zero cost) solution—feasible solutions found by [15] are not guaranteed to possess any special properties or good test set accuracy. In this paper, we address these problems by proposing a transductive extension of *maximum entropy/improved iterative scaling* (ME/IIS) [34–36] for aggregation in distributed classification. This approach ensures both feasibility of constraints and uniqueness of the solution. Moreover, the maximum entropy (ME) solution has been justified from a number of theoretical standpoints—in a well-defined statistical sense [37], ME is the "least-biased" solution, given measured constraints. We have found that this approach achieves greater accuracy than both the previous CB method [15] and fixed aggregation rules.

The rest of the paper is organized as follows. In Section 2, we give a concise description of the distributed classification problem. In Section 3, we review the previous work in [15]. In Section 4, we develop our transductive extension of ME/IIS for decision fusion in distributed classification. In Section 5, we present experimental results. The paper concludes with a discussion and pointer to future work.

2. DISTRIBUTED CLASSIFICATION PROBLEM

A system diagram for the distributed classification problem is shown in Figure 1. Each classifier produces either hard decisions or a posteriori class probabilities $P_j[\hat{C}_j = \hat{c} \mid \underline{x}^{(j)}] \in [0,1]$, $\hat{c} = 1,\dots,N_c$, $j = 1,\dots,M_e$, where N_c is the number of classes, M_e the number of classifiers, and $\underline{x}^{(j)} \in \mathcal{R}_{k^{(j)}}$ the feature vector for the jth classifier. Each local classifier is designed based on its own (separate) training set $\widetilde{\mathcal{X}}_j = \{(\underline{\tilde{x}}_i^{(j)}, \tilde{c}_i^{(j)}),\ i = 1,\dots,N_j\}$, where $\underline{\tilde{x}}_i^{(j)} \in \mathcal{R}_{k^{(j)}}$ and $\tilde{c}_i^{(j)}$ is the class label. We also denote the training set excluding the class labels by $\mathcal{X}_j = \{\underline{\tilde{x}}_i^{(j)}\}$. The local class priors, as reflected in each local training set, may differ from each other. More importantly, they will in general differ from the true (test set) priors. While there is no common labeled training data, during the operational (use) phase of the system, common data *is* observed across the ensemble, that is, for each new object to classify, a feature vector is measured by each classifier. If this were not the case, decision fusion across the ensemble, in *any* form, would not be possible. We do not consider the problem of missing features in this work, wherein *some* local feature vectors and associated classifier decisions are unavailable for certain test instances. However, we believe our framework can be readily extended to address the missing features case. Thus, during use/testing, the input to the ensemble system is effectively the concatenated vector $\underline{\mathbf{x}} = (\underline{x}^{(1)}, \underline{x}^{(2)}, \dots, \underline{x}^{(M_e)})$, but with classifier j only observing $\underline{x}^{(j)}$.

A key aspect is that we learn on a *batch* of test samples, $\mathcal{X}_{\text{test}} = \{\underline{\mathbf{x}}_1, \underline{\mathbf{x}}_2, \dots, \underline{\mathbf{x}}_{N_{\text{test}}}\}$—since we are learning solely from *unlabeled* data, we at least need a reasonably sizeable batch of such data, if we are to learn more accurate decisions than a fixed combining strategy. The transductive learning in [15] required *joint* decision making on all samples in the batch. In some applications, sequential decision making is instead required. To accomodate this, [33] developed a sequential extension wherein, at time t, a batch of size N is defined by a causal sliding window, containing the samples $\{\underline{\mathbf{x}}_{t-N+1}, \underline{\mathbf{x}}_{t-N+2}, \dots, \underline{\mathbf{x}}_{t-1}, \underline{\mathbf{x}}_t\}$. While the transductive learning produces decisions on *all* samples in the current batch, only the decision on $\underline{\mathbf{x}}_t$ is actually used since decisions on the past samples have already been made [33].

Before performing transductive learning, the aggregation function collects batches of soft (or hard) decisions conveyed by each classifier, for example, in the batch decision making case $\{\{P_j[\hat{C}_j = \hat{c} \mid \underline{x}_i^{(j)}]\ \forall \hat{c}\},\ j = 1,\dots,M_e,\ i = 1,\dots,N_{\text{test}}\}$. We ignore communication bandwidth considerations, assuming each classifier directly conveys posteriors (if, instead of hard decisions, they are produced), without quantization.

3. PRIOR WORK ON TRANSDUCTIVE LEARNING FOR DISTRIBUTED CLASSIFICATION

3.1. Transductive maximum likelihood methods

In [15], methods were first proposed that *explicitly* correct for mismatched class priors in several well-known ensemble combining rules. These methods extended [18], which addressed prior correction for a *single* classifier. These methods are transductive *maximum likelihood estimation* (MLE) algorithms that learn on $\mathcal{X}_{\text{test}}$ and treat the class priors as the *sole* model parameters to be estimated. There are three tasks that need to be performed in explicitly correcting for mismatched class priors: (1) estimating new (test batch) class priors $P_e[C = c]$, $c = 1,\dots,N_c$, (2) correcting local posteriors $P_j[\hat{C}_j = c \mid \underline{x}_i^{(j)}]\ \forall c$, $j = 1,\dots,M_e$, $i = 1,\dots,N_{\text{test}}$ to reflect the new class priors, and (3) aggregating the corrected posteriors to yield ensemble posteriors $P_e[C = c \mid \underline{\mathbf{x}}_i]\ \forall i, c$.

In [15], expectation maximization (EM) algorithms [38] were developed that *naturally* accomplish these tasks for several well-known aggregation rules when particular statistical assumptions are made. The M-step re-estimates class priors. Interestingly, the E-step *directly* accomplishes local classifier aggregation, yielding the ensemble posteriors and, *internal to this step*, correcting local posteriors. As shown in [15], these algorithms are *globally convergent*, to the unique MLE solution. At convergence, the ensemble posteriors produced in the E-step are used for maximum a posteriori (MAP) decision making.

For the naive Bayes (NB) case where local classifiers' feature vectors are assumed to be independent conditioned on the class, the following EM algorithm was derived [15]:

E-step(NB):

$$P_e^{(t)}[C = c \mid \underline{\mathbf{x}}_i] = \frac{P_e^{(t-1)}[C = c]\,p'}{\sum_{l=1}^{N_c} P_e^{(t-1)}[C = l]\,p''} \quad \forall c,\ \forall i, \tag{1}$$

where p' denotes $\prod_{j=1}^{M_e}(P_j[\hat{C}_j = c \mid \underline{x}_i^{(j)}]/P_j[\hat{C}_j = c])$, and p'' denotes $\prod_{j=1}^{M_e}(P_j[\hat{C}_j = l \mid \underline{x}_i^{(j)}]/P_j[\hat{C}_j = l])$,

$$\text{M-step: } P_e^{(t)}[C = c] = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} P_e^{(t)}[C = c \mid \underline{\mathbf{x}}_i] \quad \forall c. \tag{2}$$

The form of the ensemble posterior in (1) is the standard *naive Bayes* form, albeit with built-in prior correction.

In [15], it was also shown that aggregation based on *arithmetic averaging* (AA), again with built-in prior correction, is achieved via transductive MLE under different statistical assumptions. For this model, the M-step is the same as in (2), but the E-step now takes the (arithmetic averaging) form:

E-step(AA):

$$\begin{aligned} &P_e^{(t)}[C = c \mid \underline{\mathbf{x}}_i] \\ &\quad = \frac{P_e^{(t-1)}[C = c](1/M_e)\,q'}{\sum_{l=1}^{N_c} P_e^{(t-1)}[C = l](1/M_e)\,q''} \quad \forall c,\ \forall i, \end{aligned} \tag{3}$$

where q' denotes $\sum_{j=1}^{M_e}(P_j[\hat{C}_j = c \mid \underline{x}_i^{(j)}]/P_j[\hat{C}_j = c])$, and q'' denotes $\sum_{j=1}^{M_e}(P_j[\hat{C}_j = l \mid \underline{x}_i^{(j)}]/P_j[\hat{C}_j = l])$.

These two algorithms do adapt the decision rule to new class priors (as reflected in a test data batch). However,

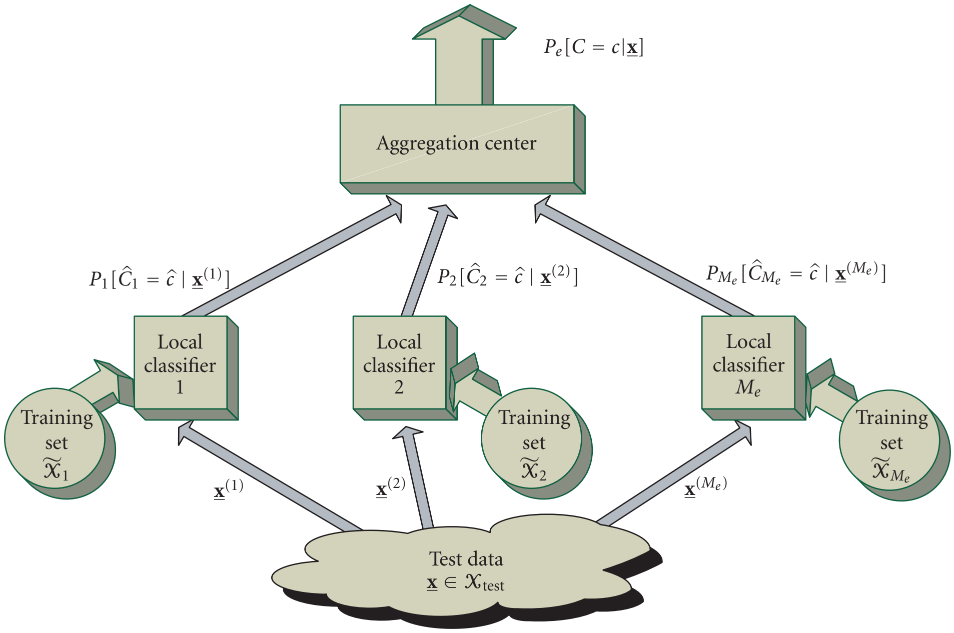

$P_e[C = c|\underline{\mathbf{x}}]$
Aggregation center
$P_1[\hat{C}_1 = \hat{c} \mid \underline{\mathbf{x}}^{(1)}]$
$P_2[\hat{C}_2 = \hat{c} \mid \underline{\mathbf{x}}^{(2)}]$
$P_{M_e}[\hat{C}_{M_e} = \hat{c} \mid \underline{\mathbf{x}}^{(M_e)}]$
Local classifier 1
Local classifier 2
Local classifier M_e
Training set $\widetilde{\mathcal{X}}_1$
Training set $\widetilde{\mathcal{X}}_2$
Training set $\widetilde{\mathcal{X}}_{M_e}$
$\underline{\mathbf{x}}^{(1)}$
$\underline{\mathbf{x}}^{(2)}$
$\underline{\mathbf{x}}^{(M_e)}$
Test data
$\underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}$

FIGURE 1: Distributed ensemble classification system.

unlike CB, they cannot be applied if classifiers solely produce hard decisions. Correction of local posteriors for mismatched priors can only be achieved if there is *access* to local posteriors—if each classifier is a "black box" solely producing hard decisions, the transductive MLE methods cannot be used for prior correction. More importantly, the ML methods are limited by their statistical assumptions, for example, conditional independence. When there are statistical dependencies between local classifiers, failing to account for them will lead to suboptimal aggregation. In [15], the following extreme example was given: suppose there are $M_e - 1$ identical copies of a weak (inaccurate) classifier, with the M_e-th classifier an accurate one. Clearly, if M_e is large, the weak classifiers will *dominate* (1) and (3) and yield poor accuracy [15]. Thus, CB was proposed to properly account for classifier redundancies, both for this extreme example and more generally.

3.2. Transductive constraint-based learning

CB differs in important aspects from the ML methods. First, CB effectively corrects mismatched class priors even if each local classifier only produces hard decisions. Second, unlike the transductive ML methods, CB is spare in its underlying statistical assumptions—the sole premise is that certain statistics measured on each local classifier's training set should be preserved (via the aggregation function's decisions) on the test set. As noted earlier, learning via constraint encoding is inherently robust to classifier redundancy. In the case of the degenerate example from the last section, the $M_e - 1$ identical weak classifiers all have the same constraints. Thus, as far as CB is concerned, the ensemble will *effectively* consist of only *two* classifiers—one strong, and one weak. The $M_e - 2$ redundant copies of the weak classifier do not bias CB's decision aggregation [15].

3.2.1. Choice of constraints

In principle, we would like to encode as constraints joint statistics that reduce uncertainty about the class variable as much as possible. For example, the joint probability $P[C = 1, \hat{C}_1 = 1, \hat{C}_2 = 0] = 0$ is quite informative about the true class variable (C). However, in our distributed setting, *with no common labeled training data*, it is not possible to measure joint statistics involving two or more classifiers and C. Thus, we are limited to encoding *pairwise* statistics involving C and individual decisions ($\hat{C}_j$). Each classifier j, using its local training data, can measure the pairwise pmf $P_g^{(j)}[C, \hat{C}_j]$ with "g" indicating "ground truth". This (naively) suggests choosing these probabilities as constraints. However, $P_g^{(j)}[C, \hat{C}_j]$ determines the *marginal* pmfs $P_g^{(j)}[C]$ and $P_g^{(j)}[\hat{C}_j]$. Via the superscript (j), we emphasize that these marginal pmfs are based on $\widetilde{\mathcal{X}}_j$ and are thus specific to local classifier j. Thus, choosing test set decisions to agree with $P_g^{(j)}[C, \hat{C}_j]$ forces agreement with the local class and class decision priors. Recall that these may differ from the true (test) priors. The local class priors $P_g^{(j)}[C]$, $j = 1, \dots, M_e$, also may be inconsistent with each other. Thus, encoding $\{P_g^{(j)}[C, \hat{C}_j]\}$ is ill-advised. Instead, it was suggested in [15] to encode the *conditional* pmfs (confusion matrices) $\{P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c]\ \forall c\}$. Confusion matrix information has been applied previously (e.g., in [39]) where it was used to define class ranks within a decision aggregation scheme and in [18] where it was used to help transductively estimate class prior probabilities for the case of a single classifier. In

[15], alternatively, confusion matrices were used to specify the constraints in our CB framework. These pmfs specify the pairwise pmfs $\{P_g^{(j)}[C, \hat{C}_j]\}$ *except for* the class priors.

The constraint probabilities are (locally) measured by

$$P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c] = \frac{\sum_{i=1:\tilde{c}_i^{(j)}=c}^{N_j} P_j[\hat{C}_j = \hat{c} \mid \underline{\tilde{x}}_i^{(j)}]}{\sum_{i=1:\tilde{c}_i^{(j)}=c}^{N_j} 1} \quad \forall j,\ \forall c,\ \forall \hat{c}. \tag{4}$$

The aggregation function's transductive (ensemble) *estimates* are

$$P_e[\hat{C}_j = \hat{c} \mid C = c] = \frac{\sum_{i=1}^{N_{\text{test}}} P_e[C=c \mid \underline{\mathbf{x}}_i] P_j[\hat{C}_j=\hat{c} \mid \underline{x}_i^{(j)}]}{\sum_{i=1}^{N_{\text{test}}} P_e[C = c \mid \underline{\mathbf{x}}_i]} \quad \forall j,\ \forall c,\ \forall \hat{c}. \tag{5}$$

In principle, then, the objective should be to choose the ensemble posteriors $\{\{P_e[C = c \mid \underline{\mathbf{x}}_i]\} \forall i\}$ so that the transductive estimates match the constraints, that is,

$$P_e[\hat{C}_j = \hat{c} \mid C = c] \doteq P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c] \quad \forall j, c, \hat{c}. \tag{6}$$

However, there is one additional complication. Suppose there is a class $\tilde{c}$ that does not *occur* in the test batch. Both the particular class and the *fact* that a class is missing from the test batch are of course unknown. It is inappropriate to impose the constraints $P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = \tilde{c}]\ \forall \hat{c},\ \forall j$— in doing so, one will assign test samples to class $\tilde{c}$, which will lead to gross inaccuracy in the solution [15]. What is thus desired is a simple way to *avoid* encoding these constraints, even as it is actually *unknown* that $\tilde{c}$ is void in the test set. A solution to this problem was practically effected by multiplying (6), both sides, by $P_e[C = c] = (1/N_{\text{test}})\sum_{i=1}^{N_{\text{test}}} P_e[C = c \mid \underline{\mathbf{x}}_i]$, that is, by expressing the *relaxed* constraints

$$\begin{aligned} &P_e[\hat{C}_j = \hat{c} \mid C = c] \cdot P_e[C = c] \\ &\quad \doteq P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c] \cdot P_e[C = c] \quad \forall j, c, \hat{c}. \end{aligned} \tag{7}$$

Note that (7) is equivalent to (6) for c such that $P_e[C = c] > 0$, but with *no* constraint imposed when $P_e[C = c] = 0$. Thus, if the learning successfully estimates that $\tilde{c}$ is missing from the test batch, encoding the pmf $\{P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = \tilde{c}]\ \forall \hat{c}\}$ will be avoided. In [15], it was found that this approach worked quite well in handling missing classes.

3.2.2. CB learning approach

In [15], the constraints (7) were practically met by choosing the ensemble posterior pmfs on $\mathcal{X}_{\text{test}}$, $\{P_e[C = c \mid \underline{\mathbf{x}}_i]\}$, to minimize a nonnegative cost consisting of the sum of relative entropies:

$$\begin{aligned} R &= \sum_{j=1}^{M_e} D\Big(P_e[\hat{C}_j \mid C]P_e[C] \| P_g^{(j)}[\hat{C}_j \mid C]P_e[C]\Big) \\ &= \sum_{j=1}^{M_e}\sum_{c=1}^{N_c} P_e[C = c] D\Big(P_e[\hat{C}_j \mid C = c] \| P_g^{(j)}[\hat{C}_j \mid C = c]\Big), \end{aligned} \tag{8}$$

based on the left- and right-hand sides of (7). Here, relative entropy is defined as $D(P[X]\|Q[X]) \equiv \sum_{x\in\mathcal{S}} P[x]\log(P[x]/Q[x])$, $\mathcal{S}$ the (common) support. Note that if R is driven to zero such that $P_e[C = c] > 0\ \forall c$, the constraints are all met. Thus, minimizing (driving to zero) R can be used to effect satisfaction of the constraints. To ensure that $P_e[C = c \mid \underline{\mathbf{x}}_i]$ is preserved as a pmf throughout the optimization, the posterior was parameterized using a softmax function, $P_e[C = c \mid \underline{\mathbf{x}}_i] = e^{\gamma_{c,i}}/\sum_{c'} e^{\gamma_{c',i}}$, with $\{\gamma_{c,i},\ \forall c,\ i = 1,\ldots,N_{\text{test}}\}$ the scalar parameters to be optimized. Minimization of R with respect to these parameters was performed by gradient descent.

This CB learning was found to give greater decision accuracy than both fixed naive Bayes, arithmetic averaging and their transductive ML extensions (1) and (3). However, there are three important limitations. First, the given constraints (6) may be infeasible. Second, even when these constraints are feasible, there is no assurance that the gradient descent learning will *find* a feasible solution (there may be local minima of the (nonconvex) cost, R). Finally, when the problem is feasible, there is a feasible solution *set*. Minimizing R assures neither a unique solution nor one possessing good properties (accuracy). We next address these shortcomings.

4. TRANSDUCTIVE CB BY MAXIMUM ENTROPY

The standard approach to finding a unique distribution satisfying given constraints is to invoke the *principle of maximum entropy* [37]. In our distributed classification setting, given the constraints (6) and the goal of *transductively* satisfying them in choosing test set posteriors, application of this principle leads to the learning objective:

Problem 1.

$$\begin{aligned} \max_{\{P_e[c|\underline{\mathbf{x}}]\ \forall c, \underline{\mathbf{x}}\in\mathcal{X}_{\text{test}}\}} H(C \mid \underline{\mathbf{X}}) = -\sum_{\underline{\mathbf{x}}\in\mathcal{X}_{\text{test}}} \frac{1}{N_{\text{test}}} \\ \cdot \sum_c P_e[c \mid \underline{\mathbf{x}}] \log P_e[c \mid \underline{\mathbf{x}}] \end{aligned} \tag{9}$$

subject to

$$\begin{aligned} &\sum_c P_e[c \mid \underline{\mathbf{x}}] = 1 \quad \forall \underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}, \\ &P_e[\hat{C}_j = \hat{c} \mid C = c] = P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c] \quad \forall j,\ \forall c,\ \forall \hat{c}, \end{aligned} \tag{10}$$

where $P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c]$ and $P_e[\hat{C}_j = \hat{c} \mid C = c]$ are measured by (4) and (5), respectively. In (9), we have assumed uniform support on the test set, that is,

$$P_e[\underline{\mathbf{x}}] = \begin{cases} \dfrac{1}{N_{\text{test}}}, & \underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}, \\ 0, & \text{else}, \end{cases} \tag{11}$$

which constrains the joint pmf to the form

$$P_e[c, \underline{\mathbf{x}}] = \begin{cases} \dfrac{1}{N_{\text{test}}} P_e[c \mid \underline{\mathbf{x}}], & \forall c, \underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}, \\ 0, & \text{else}. \end{cases} \tag{12}$$

A serious difficulty with Problem 1 is that the constraints may be infeasible. An example was given in [15]. The difficulty arises because the constraints are *measured* using each local classifier's training support $\widetilde{\mathcal{X}}_j$, but we are attempting to satisfy them using different (test) support. To overcome this, we propose to *augment* the test support to *ensure* feasibility. We next introduce three different support augmentations. In Section 4.1, we augment the test support using the local training set supports. In Section 4.2, we construct a more compact support augmentation derived from the constraints measured on the training supports. Both these augmentations ensure constraint feasibility. In Section 4.3, we discuss maximizing entropy on the *full* (discrete) support.

4.1. Augmentation with local classifier supports

The most natural augmentation is to add points from the *training set* supports ($\widetilde{\mathcal{X}}_j\ \forall j$) to the test set support. Since the constraints were *measured* on each local classifier's support, augmenting the test set support to include local training points should allow constraint feasibility. Note that this will require local classifiers to communicate both their constraints and the support points used in measuring them to the aggregation function. Consider separately the cases of (i) continuous-valued features $\underline{x}^{(j)} \in \mathcal{R}_{k(j)}\ \forall j$ and (ii) *discrete-valued* features $\underline{x}^{(j)} \in \mathcal{A}_j$, $\mathcal{A}_j$ a finite set. In the former case, there is zero probability that a training point $\tilde{\underline{x}}^{(j)}$ occurs as a component vector $\underline{x}^{(j)}$ of a test point $\underline{\mathbf{x}}$. Thus, in this case, we will augment the test support with each local classifier's full training support set $\widetilde{\mathcal{X}}_j$, and in doing so we are *exclusively* adding unique support points to the existing (test) support, that is, we assign nonzero probability to the joint events $\{C = c, \underline{\mathbf{X}} = \underline{\mathbf{x}}\}\ \forall \underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}$ and $\{C = c, \{\underline{\mathbf{x}} : \underline{x}^{(j)} = \tilde{\underline{x}}^{(j)}\}\}\ \forall \tilde{\underline{x}}^{(j)} \in \widetilde{\mathcal{X}}_j,\ \forall j$. Note that each test point is a distinct joint event, with the other joint events consisting of collections of the joint feature vectors sharing a common component vector that belongs to a local training set. Even if different local classifiers observe the same set of features, unless these classifiers measure precisely the same values for these features for some training examples (which should occur with probability zero in the continuous-valued case, assuming training sets are randomly generated, independently, for each local classifier), these classifiers will supply mutually exclusive additional support points.

Now consider the latter (discrete) case. Here, it is *quite* possible ¡?bhlt?¿that a training point $\tilde{\underline{x}}^{(j)}$ will appear as a component vector of a test point. In this case, it is *redundant* to add such points to the test support. Let $\mathcal{X}_{\text{test}}^{(j)}$ denote the set of component vectors for classifier j that occurred in the test set and $\overline{\mathcal{X}}_{\text{test}}^{(j)}$ the complement set. Then, in the discrete-valued case, we will add $\bigcup_j (\widetilde{\mathcal{X}}_j \bigcap \overline{\mathcal{X}}_{\text{test}}^{(j)})$ to the test support. In the following, our discussion is based on the continuous case.

We further note that some care must be taken to ensure that *sufficient* probability mass is allocated to the training supports to ensure constraint feasibility, for example, a *uniform* (equal) mass assignment to all support points, both test and training, will *not* in general ensure feasibility. Thus, we allow *flexible* allocation of probability mass to the training supports (both the total mass allocated to the training supports and how it is distributed across the different training support points are flexibly chosen), choosing the joint pmf to have the form

$$P_e[c, \{\underline{\mathbf{x}}\}] = \begin{cases} \dfrac{P_u}{N_{\text{test}}} P_e[c \mid \underline{\mathbf{x}}], & \forall c, \underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}, \\ P[\tilde{\underline{x}}^{(j)}, \tilde{c}^{(j)}], & \{\underline{\mathbf{x}} : \underline{x}^{(j)} = \tilde{\underline{x}}^{(j)}\}, \\ & \forall (\tilde{\underline{x}}^{(j)}, \tilde{c}^{(j)}) \in \widetilde{\mathcal{X}}_j,\ \forall j, \\ 0, & \text{else}. \end{cases} \tag{13}$$

Here, the total mass allocated to $\mathcal{X}_{\text{test}}$ is $P_u = \sum_{\underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}} \sum_c P_e[c, \underline{\mathbf{x}}]$, each *test* sample is assigned equal mass P_u/N_{test} (we allow flexible allocation of mass to the *training* support points in order to ensure feasibility of the constraints: some points are pivotal to this purpose and will need to be assigned relatively large masses, while other points are extraneous and hence may be assigned small mass; for the test set support, on the other hand, unless there are outliers, these points should contribute "equally" to constraint satisfaction (just as each sample was given equal mass in *measuring* the constraints $P_g^{(j)}[\cdot]$); accordingly, we give equal mass to each test support point) and for $\tilde{\underline{x}}^{(j)} \in \mathcal{X}_j$, $P[\tilde{\underline{x}}^{(j)}, c] = P[\tilde{\underline{x}}^{(j)}] P[c \mid \tilde{\underline{x}}^{(j)}]$, where

$$P[c \mid \tilde{\underline{x}}^{(j)}] = \begin{cases} 1, & (\tilde{\underline{x}}^{(j)}, c) \in \widetilde{\mathcal{X}}_j, \\ 0, & \text{else}, \end{cases} \tag{14}$$

that is, we exploit knowledge of the training labels in making *exclusive* posterior assignments on the training support. Here, P_u, $\{P_e[c \mid \underline{\mathbf{x}}]\}$, and $\{P[\tilde{\underline{x}}^{(j)}, \tilde{c}^{(j)}]\}$ are all parameters whose values will be learned.

For $P_e[c, \{\underline{\mathbf{x}}\}]$ defined by (13), we would like to satisfy the constraints (6). Accordingly, we need to compute the transductive estimate $P_e[\hat{C}_j = \hat{c} \mid C = c] = P_e[\hat{C}_j = \hat{c}, C = c] / P_e[C = c]$, using the joint pmf (13). However, a difficulty here is that (13) is defined on the support set $\mathcal{X}_{\text{test}} \bigcup_k \mathcal{X}_k$, but the posterior $P_j[\hat{C}_j = \hat{c} \mid \underline{x}^{(j)}]$ can *only* be evaluated on the support subset where classifier j's feature vector is observed, that is, over $\mathcal{X}_{\text{test}} \bigcup \mathcal{X}_j$. For $\mathcal{X}_k$, $k \neq j$, we only have instances of classifier k's feature vector, not j's. This means we cannot use the full support to measure $P_e[\hat{C}_j = \hat{c}, C = c]$. Formally, we resolve this issue by *conditioning*. Let $\mathcal{X}_r^{(j)} = \{\{\underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}\} \bigcup \{\underline{\mathbf{x}} : \underline{x}^{(j)} \in \mathcal{X}_j\}\}$. Then, we measure

$$P_e\Big[\hat{C}_j = \hat{c} \mid C = c, \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}\Big] = \frac{P_e\Big[\hat{C}_j = \hat{c}, C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}\Big]}{P_e\Big[C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}\Big]}. \tag{15}$$

We have

$$\begin{aligned}
&P_e\Big[\hat{C}_j = \hat{c}, C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}\Big] \\
&\quad = K_0^{-1}\left(\frac{P_u}{N_{\text{test}}}\sum_{\underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}} P_j\Big[\hat{C}_j = \hat{c} \mid \underline{x}^{(j)}\Big] P_e[c \mid \underline{\mathbf{x}}] \right.\\
&\qquad\qquad \left. + \sum_{\underline{\tilde{x}}^{(j)} \in \mathcal{X}_j, \tilde{c}^{(j)} = c} P_j\Big[\hat{C}_j = \hat{c} \mid \underline{\tilde{x}}^{(j)}\Big] P[\underline{\tilde{x}}^{(j)}, \tilde{c}^{(j)}]\right), \\
&P_e\Big[C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}\Big] \\
&\quad = K_0^{-1}\left(\frac{P_u}{N_{\text{test}}}\sum_{\underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}} P_e[c \mid \underline{\mathbf{x}}] + \sum_{\underline{\tilde{x}}^{(j)} \in \mathcal{X}_j, \tilde{c}^{(j)} = c} P[\underline{\tilde{x}}^{(j)}, \tilde{c}^{(j)}]\right),
\end{aligned} \tag{16}$$

where

$$\begin{aligned}
K_0 &= \sum_c \sum_{\hat{c}} \left(\frac{P_u}{N_{\text{test}}}\sum_{\underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}} P_j\Big[\hat{C}_j = \hat{c} \mid \underline{x}^{(j)}\Big] P_e[c \mid \underline{\mathbf{x}}] \right.\\
&\qquad\qquad \left. + \sum_{\underline{\tilde{x}}^{(j)} \in \mathcal{X}_j, \tilde{c}^{(j)} = c} P\Big[\hat{C}_j = \hat{c} \mid \underline{\tilde{x}}^{(j)}\Big] P_j[\underline{\tilde{x}}^{(j)}, \tilde{c}^{(j)}]\right) \\
&= \sum_c \left(\frac{P_u}{N_{\text{test}}}\sum_{\underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}} P_e[c \mid \underline{\mathbf{x}}] + \sum_{\underline{\tilde{x}}^{(j)} \in \mathcal{X}_j, \tilde{c}^{(j)} = c} P[\underline{\tilde{x}}^{(j)}, \tilde{c}^{(j)}]\right).
\end{aligned} \tag{17}$$

Note that from (17), we see that the same normalization constant K_0 appears in both pmfs given in (16), ensuring that these pmfs both sum to 1.

Letting $N_e[\hat{C}_j = \hat{c}, C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}] \equiv K_0 P_e[\hat{C}_j = \hat{c}, C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}]$, and $N_e[C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}] \equiv K_0 P_e[C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}]$, we have

$$P_e\Big[\hat{C}_j = \hat{c} \mid C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}\Big] = \frac{N_e\Big[\hat{C}_j = \hat{c}, C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}\Big]}{N_e\Big[C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}\Big]}. \tag{18}$$

The notation $N_e[\cdot]$ reflects the fact that this quantity represents the expected number of occurrences of the event given $\underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}$. We can thus now define the following problem.

Problem 2.

$$\begin{aligned}
&\max_{\{P_u, \{P_e[c \mid \underline{\mathbf{x}}],\, \underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}\}, \{P[\underline{\tilde{x}}^{(j)}, \tilde{c}^{(j)}]\}\}} H(C, \underline{\mathbf{X}}) \\
&\quad = -\sum_{\underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}} \frac{P_u}{N_{\text{test}}} \sum_c P_e[c \mid \underline{\mathbf{x}}] \log\left(\frac{P_u}{N_{\text{test}}} P_e[c \mid \underline{\mathbf{x}}]\right) \\
&\qquad - \sum_j \sum_{(\underline{\tilde{x}}^{(j)}, \tilde{c}^{(j)}) \in \widetilde{\mathcal{X}}_j} P[\underline{\tilde{x}}^{(j)}, \tilde{c}^{(j)}] \log P[\underline{\tilde{x}}^{(j)}, \tilde{c}^{(j)}]
\end{aligned} \tag{19}$$

subject to

$$\begin{aligned}
&P_e\Big[\hat{C}_j = \hat{c} \mid C = c, \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}\Big] = P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c] \\
&\qquad\qquad\qquad\qquad \forall j,\ \forall c,\ \forall \hat{c}, \\
&P_u + \sum_j \sum_{(\underline{\tilde{x}}^{(j)}, \tilde{c}^{(j)}) \in \widetilde{\mathcal{X}}_j} P[\underline{\tilde{x}}^{(j)}, \tilde{c}^{(j)}] = 1, \\
&\sum_c P_e[C = c \mid \underline{\mathbf{x}}] = 1, \quad \underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}.
\end{aligned} \tag{20}$$

We emphasize that in this problem the constraints are guaranteed to be feasible. In particular, they can always be satisfied via an exclusive assignment of all probability mass to the labeled supports (i.e., via the choice $P_u = 0$). A proof is provided in Appendix A.1.

4.2. *Augmentation with support derived from constraints*

The previous augmentation seems to imply that the local training set supports $\{\widetilde{\mathcal{X}}_j\}$ need to be made available to the decision aggregation function. Actually, only the posteriors (soft decisions) made on $\widetilde{\mathcal{X}}_j\ \forall j$, and the associated class labels are needed by the aggregation function. However, even this may not be realistic in distributed contexts involving proprietary classifiers or distributed multisensor classification. Suppose instead that the *only* local classifier information communicated to the aggregation function is the set of constraints $P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c]\ \forall j,\ \forall c,\ \forall \hat{c}$. We would *still* like to augment the test support to ensure a feasible solution. This can be achieved as follows.

First, note that $\underline{x}^{(j)}$ determines the local posterior $\{P_j[\hat{C}_j = \hat{c} \mid \underline{x}^{(j)}],\ \forall \hat{c}\}$ and that the joint probability $P[\underline{\tilde{x}}^{(j)}, \tilde{c}^{(j)}]$ can thus be equivalently written as $P[(\{P_j[\hat{C}_j = \hat{c} \mid \underline{\tilde{x}}^{(j)}],\ \forall \hat{c}\}, \tilde{c}^{(j)})]$. In other words, the method in the last subsection assigns nonzero joint probability *only* to the posterior pmfs and conjoined class labels $\{(\{P_j[\hat{C}_j = \hat{c} \mid \underline{\tilde{x}}^{(j)}],\ \forall \hat{c}\}, \tilde{c}^{(j)}), (\underline{\tilde{x}}^{(j)}, \tilde{c}^{(j)}) \in \widetilde{\mathcal{X}}_j\}$ that are induced by the local training set $\widetilde{\mathcal{X}}_j$. An alternative support augmentation ensuring feasibility is thus specified as follows.

Consider all pairs $(\hat{c}, c)$ such that $P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c] > 0$. For each such pair, introduce a new support point

$([0,\ldots,0,1,0,\ldots,0],c)$ with “1” in the $\hat{c}$-th entry, that is, the joint event that $C = c$ and

$$P_j\left[\hat{C}_j = \tilde{c} \mid \underset{\sim}{x}^{(j)}\right] = \begin{cases} 1, & \tilde{c} = \hat{c}, \\ 0, & \text{else.} \end{cases} \tag{21}$$

If $P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c\,] > 0\ \forall(c,\hat{c})$, there are N_c^2 such support points added for each local classifier $\hat{C}_j$. We assert that adding these support points, as an alternative to the training set supports, ensures feasibility of the ME constrained problem. A proof sketch is given in Appendix A.2. In Section 5, we will demonstrate experimentally that there are only small performance differences in practice between the use of these two support augmentations.

4.3. *Full support in the hard decision case*

Suppose each local classifier makes a hard decision, that is, $\underline{\mathbf{x}}$ determines a discrete-valued joint decision vector $\underline{\hat{c}}(\underline{\mathbf{x}}) = (\hat{c}_1,\ldots,\hat{c}_{M_e})$. In this case, we wish to transductively learn the joint pmf $P_e[C = c, \underline{\hat{C}} = \underline{\hat{c}}(\underline{\mathbf{x}})] = P_e[C = c \mid \underline{\hat{C}} = \underline{\hat{c}}(\underline{\mathbf{x}})]P_e[\underline{\hat{C}} = \underline{\hat{c}}(\underline{\mathbf{x}})]$ to meet the local classifier constraints $\{P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c]\}$. It is instructive to consider what happens if we meet constraints on the *full* space $\mathcal{C}\times\hat{\mathcal{C}}_1\times\cdots\times\hat{\mathcal{C}}_{M_e}$, that is, if we allow assigning positive values to $P_e[C = c, \underline{\hat{C}} = \underline{\hat{c}}]\ \forall(c,\underline{\hat{c}})$, rather than restricting nonzero probability to the test set via

$$P_e[C = c, \underline{\hat{C}} = \underline{\hat{c}}(\underline{\mathbf{x}})] = \begin{cases} \dfrac{1}{N_{\text{test}}} P_e[C = c \mid \underline{\hat{C}} = \underline{\hat{c}}(\underline{\mathbf{x}})], & \underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}, \\ 0, & \underline{\hat{c}} \notin \{\underline{\hat{c}}(\underline{\mathbf{x}}) : \underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}\}. \end{cases} \tag{22}$$

We have the following proposition.

Proposition 1. *The ME joint pmf* $P_e[C = c, \underline{\hat{C}} = \underline{\hat{c}}]$ *consistent with the specified constraints* $P_e[\hat{C}_j = \hat{c} \mid C = c] = P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c]\ \forall j, c, \hat{c}$ *has the naive Bayes joint pmf form*

$$P_e[C = c, \underline{\hat{C}} = \underline{\hat{c}}] = \frac{1}{N_c}\prod_{j=1}^{M_e} P_g^{(j)}[\hat{C}_j = \hat{c}_j \mid C = c] \tag{23}$$

with associated posterior

$$P_e[C = c \mid \underline{\hat{C}} = \underline{\hat{c}}] = \frac{\prod_{j=1}^{M_e} P_g^{(j)}[\hat{C}_j = \hat{c}_j \mid C = c]}{\sum_{c'=1}^{N_c}\prod_{j=1}^{M_e} P_g^{(j)}[\hat{C}_j = \hat{c}_j \mid C = c']}. \tag{24}$$

The proof follows the proof given in [40] that the ME solution given conditional probability constraints has the naive Bayes joint pmf form and, further, uses the fact that, given only conditional probability constraints, a uniform class prior pmf $P_e[C = c] = 1/N_c$ maximizes entropy. Thus, satisfying the constraints on the *full discrete* support leads to the naive Bayes solution and to (assumed) conditional independence of local classifier decisions. This is clearly undesirable, as the local classifier decisions may in fact be strongly *dependent*.

As a simple example, just consider the case where the classifiers in the ensemble are *perfectly dependent*, that is, *identical copies*. In this case, there is only nonzero support $P_e[\underline{\hat{c}}]$ on $\underline{\hat{c}} \in \mathcal{C}_{\text{ident}} = \{(1,1,\ldots,1),(2,2,\ldots,2),\ldots,(N_c,N_c,\ldots,N_c)\}$. It can be shown that the ME posterior satisfying the constraints $P_e[\hat{C}_j \mid C = c] = P_g[\hat{C}_j \mid C = c]\ \forall j$ using *only* the nonzero support set $\mathcal{C}_{\text{ident}}$ is the posterior

$$P_e[C = c \mid \underline{\hat{C}} = \underline{\hat{c}}] = \frac{P_g[\hat{C}_j = \hat{c}_j \mid C = c]}{\sum_{c'} P_g[\hat{C}_j = \hat{c}_j \mid C = c']}, \quad \text{any } j. \tag{25}$$

This is in fact the *true* posterior in the perfectly dependent case and correctly captures the fact that there is effectively only a single classifier in the ensemble. This solution is wholly different from that obtained by plugging $\underline{\hat{c}} \in \mathcal{C}_{\text{ident}}$ in (24), which yields

$$P_e[C = c \mid \underline{\hat{C}} = \underline{\hat{c}}] = \frac{\left(P_g[\hat{C}_j = \hat{c}_j \mid C = c]\right)^{M_e}}{\sum_{c'}\left(P_g[\hat{C}_j = \hat{c}_j \mid C = c']\right)^{M_e}}, \quad \underline{\hat{c}} \in \mathcal{C}_{\text{ident}}. \tag{26}$$

Note that (26) is *highly biased*, treating classifier decisions as conditional-independent when they are in fact perfectly *dependent*. A related point of view on the solution (24) is that it will have higher *entropy* $H(C,\underline{\mathbf{X}})$ than a solution that maximizes entropy on a reduced support set. Lower entropy is, in fact, desirable because although we choose distributions to *maximize* entropy while satisfying constraints, we should choose our constraints to make this maximum entropy as small as possible, that is, a *min-max entropy principle* [41]. Restricting support to the test set imposes additional constraints on the solution, which *reduces* the (maximum) entropy.

The previous discussion instructs that the test set support contains vital information, and the *only* information, that we possess about statistical dependencies between local classifiers. Satisfying constraints on the full support set discards this information, and will increase entropy. Even augmenting the test set support less dramatically, for example, by adding the training set supports, could affect accuracy of the posterior—(*over*)use of the training set support augmentation (high $(1 - P_u)$) may allow satisfying the constraints essentially *only* using the training set supports. Since the objective is to maximize $H(C,\underline{\mathbf{X}})$, the optimization would, in this case, choose posteriors on the test set to be as uniform as possible (while still satisfying the constraints). These test set posteriors could be quite inaccurate. In other words, too much reliance on training supports makes it less imperative to “get things right” on the test set. To make test set posteriors as accurate as possible, we believe they should contribute as much as possible to constraint satisfaction, for example, we have the following loosely stated learning principle: seek the *minimal* use of the extra

```
1. Initialize
   $\beta_a \leftarrow 0, \beta_b \leftarrow$ a large negative number.
2. Do
   (a) $\beta = (1/2)(\beta_a + \beta_b)$.
   (b) Run TIS algorithm at the given $\beta$ until $\Delta D$, the difference in the Deviation $D$ (defined in Section 4.8) between successive
         iterations is less than $\varepsilon_1$. Measure $P_u$.
   (c) If $(1 - P_u > \delta$ AND $D < \varepsilon_2)$, then $\beta_a \leftarrow \beta$, otherwise $\beta_b \leftarrow \beta$.
   while $(| \beta_a - \beta_b | < \varepsilon_3)$.
3. Output
   $c(\underline{\mathbf{x}}) = \arg\max_c P_e[c \mid \underline{\mathbf{x}}], \ \underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}$
```

Algorithm 1: ETIS algorithm pseudocode.

support necessary to achieve the constraints. To capture this learning principle mathematically, we propose the following extension of Problem 2.

Problem 3.

$$\max_{\{P_u,\{P_e[c|\underline{\mathbf{x}}],\ \underline{\mathbf{x}}\in\mathcal{X}_{\text{test}}\},\ \{P[\underline{\tilde{x}}^{(j)},\tilde{c}^{(j)}],\ (\underline{\tilde{x}}^{(j)},\tilde{c}^{(j)})\in\widetilde{\mathcal{X}}_j\}\}} H(C,\underline{\mathbf{X}})$$

$$= -\sum_{\underline{\mathbf{x}}\in\mathcal{X}_{\text{test}}} \frac{P_u}{N_{\text{test}}}\sum_c P_e[c \mid \underline{\mathbf{x}}] \log\left(\frac{P_u}{N_{\text{test}}}P_e[c \mid \underline{\mathbf{x}}]\right) \quad (27)$$

$$-\sum_j \sum_{(\underline{\tilde{x}}^{(j)},\tilde{c}^{(j)})\in\widetilde{\mathcal{X}}_j} P[\underline{\tilde{x}}^{(j)},\tilde{c}^{(j)}] \log P[\underline{\tilde{x}}^{(j)},\tilde{c}^{(j)}]$$

subject to

$$P_e\left[\hat{C}_j = \hat{c} \mid C = c,\ \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}\right] = P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c] \quad \forall j,\ \forall c,\ \forall \hat{c}, \quad (28)$$

$$P_u + \sum_j \sum_{(\underline{\tilde{x}}^{(j)},\tilde{c}^{(j)})\in\widetilde{\mathcal{X}}_j} P[(\underline{x}^{(j)},\tilde{c}^{(j)})] = 1, \quad (29)$$

$$\sum_c P_e[C = c \mid \underline{\mathbf{x}}] = 1, \quad \underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}, \quad (30)$$

and subject to

$$1 - P_u = P_o. \quad (31)$$

In this objective, we modify Problem 2 to also constrain the total probability allocated to the labeled training supports to some specified value P_o. In Section 4.7, we will develop an algorithm seeking to find the *minimum* value $P_o = P_o^*$ such that the constraints are still feasible. When the test set support is sufficient by itself to meet the constraints, $P_o^* = 0$, otherwise, $P_o^* > 0$. In the sequel, we will invoke the method of Lagrange multipliers and introduce a Lagrange multiplier β associated with (31) to set the level $1 - P_u$. Thus, for the algorithm in Section 4.7, the search for P_o^* will be realized by varying β. In our experimental results, we will demonstrate that as $1 - P_u$ is reduced, the entropy $H(C,\underline{\mathbf{X}})$ decreases, and moreover, the test set classification accuracy tends to increase.

4.4. Constraint relaxation

In the sequel, we develop a transductive extension of iterative scaling (IS) techniques [35, 36] for solving the ME constrained problem for fixed $1 - P_u$, that is, for fixed β. To apply IS, the ME problem must be convex in all parameters and the constraints must be linear in the probabilities $P_e[c, \underline{\mathbf{x}}]$ [34, 36]. The function $H(C,\underline{\mathbf{X}})$ is convex; however, the constraints (28) are nonlinear in the parameters since $P_e[C = c \mid \underline{\mathbf{x}}_i]$ appears in both the numerator and denominator in (15). However, it is possible to *relax* the constraints (28) to linear ones. In particular, assuming $N_e[C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}] > 0$, if we plug the right-hand side of (18) into (28) and multiply through by $N_e[C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}]$, we then have the equivalent linear constraints

$$N_e\left[\hat{C}_j = \hat{c}, C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}\right] = P_g^{(j)}[\hat{C}_j \mid C = c] N_e\left[C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}\right] \quad \forall \hat{c}. \quad (32)$$

Further, note that

$$N_e\left[\hat{C}_j = \hat{c}, C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}\right] = P_e[\hat{C}_j = \hat{c} \mid C = c] N_e\left[C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}\right]. \quad (33)$$

Thus, comparing (32) and (33), we see that whenever $N_e[C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}] > 0$, the relaxed constraints (32) are equivalent to the original constraints (28), as desired. This is reminiscent of the constraint relaxation built into [15]. However, in some cases, if it is not possible to satisfy the original constraints at the given value $1 - P_u$, the constraints (32) can *in principle* still be satisfied by choosing $N_e[C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}] = 0$ for some c and j. This would amount to *removing* the associated pmf constraint $\{P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c]\ \forall \hat{c}\}$. Thus, setting $N_e[C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}] = 0$ would preserve feasibility of the linearized constraints (32) and amount to satisfying only a subset of the original constraints (28), those being jointly feasible. It is quite conceivable that this type of constraint relaxation would be undesirable—it amounts to encoding less constraint information, which could have a deleterious effect on decision fusion accuracy. However, we emphasize that while in principle this "constraint relaxation" appears possible, this phenomenon has *never* occurred in *any* of our experiments, even those in which some classes were missing completely from the test batch, where we might have expected to observe this relaxation phenomenon. One explanation for why constraint relaxation never occurred in our experiments is that a solution with $N_e[C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}] = 0$ is intrinsically a *low entropy* solution, which will always be rejected in favor of solutions with higher entropy, so long as such solutions exist. Since, for example, in the missing class case, there will always be augmented support points from class $C = c$ (the missing test class), it is plausible that constraints involving $C = c$ can still be met through almost exclusive use of these augmented support points (i.e., even if very little ensemble posterior probability in class $C = c$ is assigned to any of the unlabeled test set points). In fact, this is what we have experimentally observed in the missing class case—on the test set, very little probability is assigned to the missing class, but positive mass is allocated to the augmented support points that are labeled by the missing class, and constraints involving the missing class are always met. Thus, at least in all of our experiments, the "linearized" constraints (32) were always found to be equivalent to the original constraints (28), as is desired.

4.5. Transductive iterative scaling (TIS) algorithm

The Lagrangian cost function corresponding to Problem 3 with the "relaxed constraints" is

$$\begin{aligned}
L(\beta) = &-\sum_{\underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}} \frac{P_u}{N_{\text{test}}} \sum_c P_e[c \mid \underline{\mathbf{x}}] \log P_e[c \mid \underline{\mathbf{x}}] - P_u \log \frac{P_u}{N_{\text{test}}} \\
&- \sum_j \sum_{(\underline{\tilde{x}}^{(j)}, \tilde{c}^{(j)}) \in \widetilde{\mathcal{X}}_j} P[\underline{\tilde{x}}^{(j)}, \tilde{c}^{(j)}] \log P[\underline{x}^{(j)}, \tilde{c}^{(j)}] \\
&+ \sum_{j,\hat{c},c} \gamma(\hat{C}_j = \hat{c}, C = c) \Big(N_e\Big[\hat{C}_j = \hat{c}, C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}\Big] \\
&\qquad - P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c] N_e\Big[C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}\Big]\Big) \\
&+ \alpha\Bigg(P_u + \sum_j \sum_{(\underline{\tilde{x}}^{(j)}, \tilde{c}^{(j)}) \in \widetilde{\mathcal{X}}_j} P[\underline{\tilde{x}}^{(j)}, \tilde{c}^{(j)}] - 1 \Bigg) \\
&+ \sum_{\underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}} \lambda(\underline{\mathbf{x}}) \Bigg(\sum_c P[c \mid \underline{\mathbf{x}}] - 1 \Bigg) + \beta(1 - P_u).
\end{aligned} \tag{34}$$

Here, $\{\gamma(\hat{C}_j = \hat{c}, C = c), \forall j, c, \hat{c}\}$ are the Lagrange multipliers associated with the local classifier constraints, which need to be learned. The Lagrange multipliers α and $\lambda(\underline{\mathbf{x}})$ $\forall \underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}$ will be automatically chosen to satisfy the pmf sum constraints. The Lagrange multiplier β is treated as an "external" parameter and chosen to set the value of $1 - P_u$ as previously discussed. The transductive iterative scaling (TIS) algorithm for minimizing $L(\beta)$ consists of alternating (i) optimization of P_u, $\{P_e[c \mid \underline{\mathbf{x}}]\}$, and $\{P[\underline{\tilde{x}}^{(j)}, \tilde{c}^{(j)}]\}$ given $\{\gamma(\hat{C}_j = \hat{c}, C = c)\}$ held fixed, followed by (ii) update of $\{\gamma(\hat{C}_j = \hat{c}, C = c)\}$ given the other parameters fixed. In step (ii), we update a single Lagrange multiplier (with the Lagrange multipliers selected in a fixed, cyclical order), that is, we have implemented a "sequential" TIS algorithm, akin to [42], rather than a batch algorithm where all Lagrange multipliers are updated in parallel. We believe there are no difficulties inherent to a batch version—we simply chose a sequential algorithm.

For fixed $\{\gamma(\hat{C}_j = \hat{c}, C = c)\}$, the globally optimal values for the remaining parameters, determined by taking partial derivatives of $L(\beta)$ and setting to zero, are found to be

$$P_e[c \mid \underline{\mathbf{x}}] = \frac{a'}{\sum_{c'=1}^{N_c} a''} \quad \forall c, \underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}, \tag{35}$$

where a' denotes $\exp(\sum_{j=1}^{M_e} \sum_{\hat{c}=1}^{N_c} \gamma(\hat{C}_j = \hat{c}, C = c)(P_j[\hat{C}_j = \hat{c} \mid \underline{x}^{(j)}] - P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c]))$, and a'' denotes $\exp(\sum_{j=1}^{M_e} \sum_{\hat{c}=1}^{N_c} \gamma(\hat{C}_j = \hat{c}, C = c')(P_j[\hat{C}_j = \hat{c} \mid \underline{x}^{(j)}] - P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c']))$,

$$\begin{aligned}
P[(\underline{\tilde{x}}^{(j)}, \tilde{c}^{(j)})] = \frac{1}{Z(\underline{\gamma})} \exp\Bigg(&\sum_{\hat{c}=1}^{N_c} \gamma\Big(\hat{C}_j = \hat{c}, C = \tilde{c}^{(j)}\Big) \\
&\cdot \Big(P_j\Big[\hat{C}_j = \hat{c} \mid \underline{\tilde{x}}^{(j)}\Big] \\
&\qquad - P_g^{(j)}\Big[\hat{C}_j = \hat{c} \mid C = \tilde{c}^{(j)}\Big]\Big)\Bigg), \\
&(\underline{\tilde{x}}^{(j)}, \tilde{c}^{(j)}) \in \widetilde{\mathcal{X}}_j, \ j = 1, \ldots, M_e,
\end{aligned} \tag{36}$$

$$\begin{aligned}
P_u = \frac{1}{Z(\underline{\gamma})} &N_{\text{test}} \exp(-\beta) \\
&\cdot \exp\Bigg(-\frac{1}{N_{\text{test}}} \sum_{\underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}} \sum_c P_e[c \mid \underline{\mathbf{x}}] \log P_e[c \mid \underline{\mathbf{x}}] \Bigg) \\
&\cdot \exp\Bigg(\sum_{j,c,\hat{c}} \gamma(\hat{C}_j = \hat{c}, C = c) \frac{1}{N_{\text{test}}} \sum_{\underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}} P_e[C = c \mid \underline{\mathbf{x}}] \\
&\qquad \cdot \Big(P_j\Big[\hat{C}_j = \hat{c} \mid \underline{x}^{(j)}\Big] - P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c]\Big)\Bigg),
\end{aligned} \tag{37}$$

where

$$\begin{aligned}
Z(\underline{\gamma}) = N_{\text{test}} &\exp(-\beta) \\
&\cdot \exp\Bigg(-\frac{1}{N_{\text{test}}} \sum_{\underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}} \sum_c P_e[c \mid \underline{\mathbf{x}}] \log P_e[c \mid \underline{\mathbf{x}}] \Bigg) \\
&\cdot \exp\Bigg(\sum_{j,c,\hat{c}} \gamma(\hat{C}_j = \hat{c}, C = c) \frac{1}{N_{\text{test}}} \sum_{\underline{\mathbf{x}} \in \mathcal{X}_{\text{test}}} P_e[C = c \mid \underline{\mathbf{x}}] \\
&\qquad \cdot \Big(P_j\Big[\hat{C}_j = \hat{c} \mid \underline{x}^{(j)}\Big] - P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c]\Big)\Bigg)
\end{aligned}$$

$$
\begin{aligned}
&+\sum_{j}\sum_{(\underline{\tilde{x}}^{(j)},\tilde{c}^{(j)})\in\tilde{\mathcal{X}}_j} \\
&\cdot \exp\left(\sum_{\hat{c}=1}^{N_c}\gamma\left[\hat{C}_j=\hat{c},C=\tilde{c}^{(j)}\right]\right. \\
&\qquad\left.\times\left(P_j\left[\hat{C}_j=\hat{c}\mid\underline{\tilde{x}}^{(j)}\right]-P_g^{(j)}\left[\hat{C}_j=\hat{c}\mid C=\tilde{c}^{(j)}\right]\right)\right).
\end{aligned}
\tag{38}
$$

Given these parameters held fixed, we optimize a single Lagrange Multiplier, for example, $\gamma(\hat{C}_j=\hat{c},C=c)$, via the TIS update:

$$
\begin{aligned}
\gamma(\hat{C}_j=\hat{c},C=c) &\longleftarrow \gamma(\hat{C}_j=\hat{c},C=c) \\
&+\log\left(\frac{P_g^{(j)}[\hat{C}_j=\hat{c}\mid C=c]N_e[C=c\mid\underline{\mathbf{x}}\in\mathcal{X}_r^{(j)}]}{N_e[\hat{C}_j=\hat{c},C=c\mid\underline{\mathbf{x}}\in\mathcal{X}_r^{(j)}]}\right).
\end{aligned}
\tag{39}
$$

In Appendix A.3, we give a simple proof that the TIS updates (39) are descent steps in the Lagrangian cost. Thus, both the alternating steps (35)–(37) (which are globally optimal) as well as (39) are descent steps in the Lagrangian. These alternating steps are performed until the constraints are well met (a convergence criterion is introduced in Section 4.8). Note that while descent in the Lagrangian is assured, we do not provide a proof in this paper that the TIS algorithm described above converges to the ME solution satisfying the constraints. This requires a more detailed technical argument. However, we do believe that such a proof can be developed, following along the lines of the technical approaches taken in [22, 36, 42].

4.6. Comments on TIS algorithm

(1) The algorithm solves a convex optimization problem with linear constraints, with, therefore, a unique solution at each β for which the constraints are feasible [43].

(2) The TIS algorithm descends in the Lagrangian cost function for a given β.

(3) If the problem is infeasible at a given β, convergence of the algorithm is not guaranteed.

(4) When $\beta\to-\infty$, $P_u=1$, and we are seeking a solution that *only* relies on the test set support.

(5) When $\beta\to\infty$, $P_u=0$, and we are not using the test support at all.

(6) When $\beta=0$, there is *no* constraint on P_u—this solution thus achieves highest entropy $H(C,\underline{\mathbf{X}})$, compared to solutions at other β values.

4.7. Extended TIS (ETIS) algorithm

Problem 3, but with the constraints (28) replaced by the relaxed constraints (32) and seeking $1-P_u=P_o^*$, is addressed by Algorithm 1 on page 10. This method solves the ME problem for fixed β using the TIS algorithm and embeds TIS within a search for P_o^* by varying β via a bisection search.

4.8. Comments on ETIS algorithm

At fixed β, the TIS algorithm is run until practical constraint satisfaction is achieved. We measure the squared deviation $D\equiv\sum_{j,c,\hat{c}}(N_e[\hat{C}_j=\hat{c},C=c\mid\underline{\mathbf{x}}\in\mathcal{X}_r^{(j)}]-P_g^{(j)}[\hat{C}_j=\hat{c}\mid C=c]N_e[C=c\mid\underline{\mathbf{x}}\in\mathcal{X}_r^{(j)}])^2$, and stop when the change in deviation (ΔD) is less than a threshold value. The *overall* algorithm terminates when one of the following two conditions is satisfied. (1) At the current β, constraint satisfaction is not achieved. This indicates that P_o^* is greater than the value $1-P_u$ associated with the current β. (2) At the current β, at termination of TIS, $1-P_u<\delta$, where δ is a small number. In this case, we have essentially found that there is an ME solution satisfying the constraints using only the test support, (i.e., $P_o^*\approx 0$).

5. RESULTS

We next present both illustrative results and experiments comparing the TIS algorithm (Section 4.5, with the choice $\beta=0$) and ETIS algorithm with a variety of alternative transductive and fixed combining schemes.

5.1. An infeasible constraint example

There are 2 classes ("+" and "○") in two dimensions, with local classifiers 1 and 2, shown in Figure 2. Suppose each local classifier makes hard decisions, that is, $P_g^{(j)}[\hat{C}_j=\hat{c}\mid C=c]\in\{0,1\}\ \forall\hat{c},c$.

Let local classifier 1 achieve perfect training accuracy, that is,

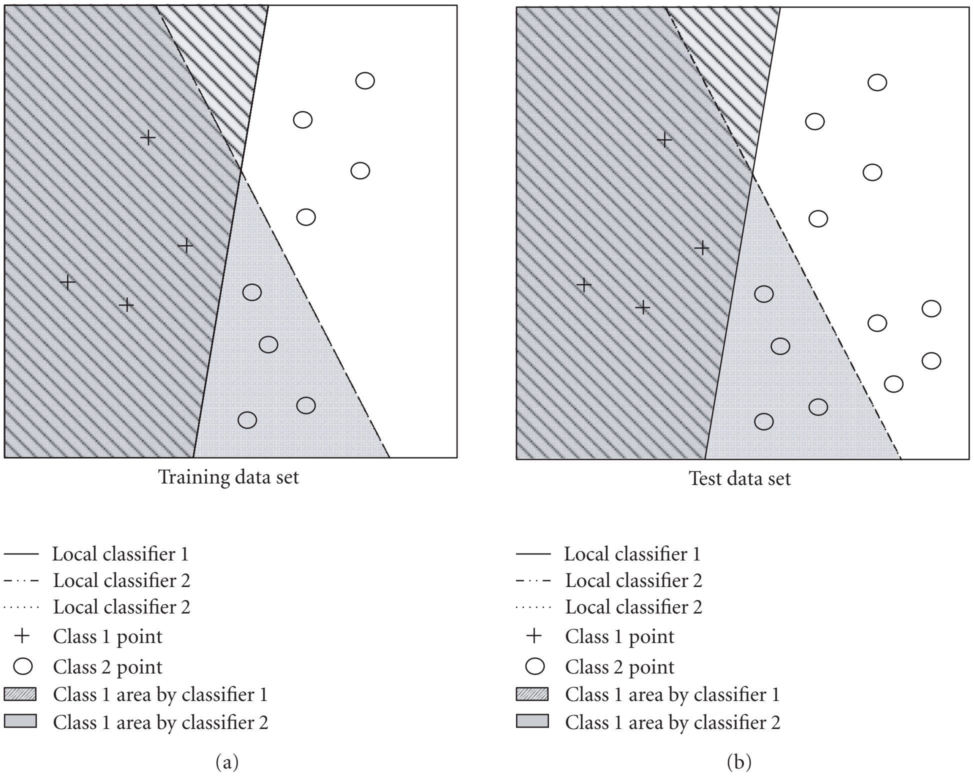
$$
\begin{aligned}
&P_g^{(1)}[\hat{C}_1=1\mid C=1]=1, \qquad P_g^{(1)}[\hat{C}_1=2\mid C=1]=0,\\
&P_g^{(1)}[\hat{C}_1=1\mid C=2]=0, \qquad P_g^{(1)}[\hat{C}_1=2\mid C=2]=1.
\end{aligned}
\tag{40}
$$

Let local classifier 2 have

$$
\begin{aligned}
&P_g^{(2)}[\hat{C}_2=1\mid C=1]=1, \qquad P_g^{(2)}[\hat{C}_2=2\mid C=1]=0,\\
&P_g^{(2)}[\hat{C}_2=1\mid C=2]=\frac{1}{2}, \qquad P_g^{(2)}[\hat{C}_2=2\mid C=2]=\frac{1}{2}.
\end{aligned}
\tag{41}
$$

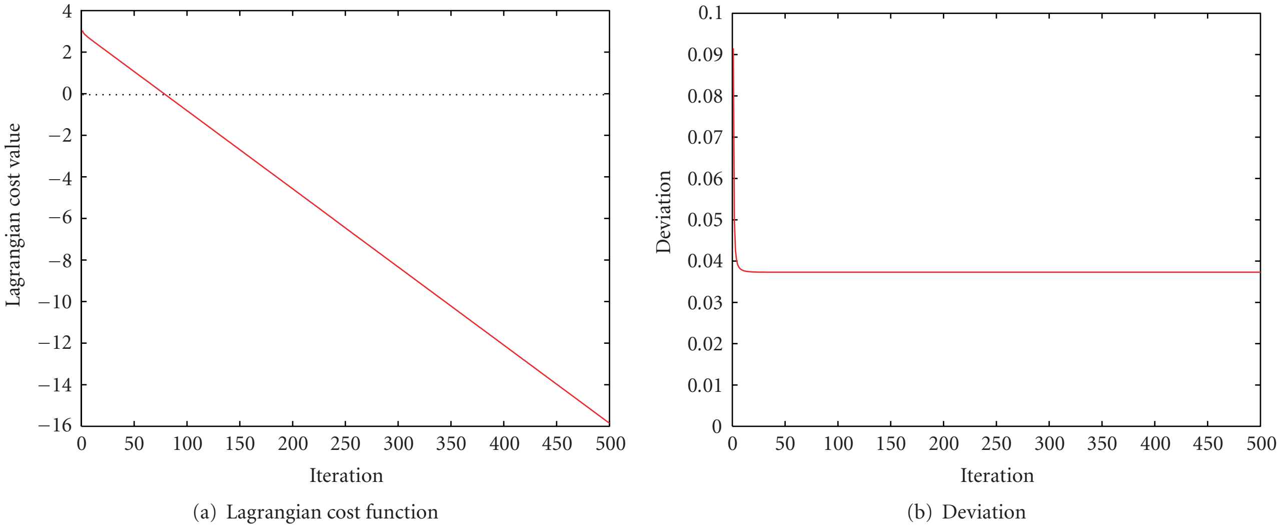
As Figure 2 shows, we let the test set be the same as the training set *except* that it contains four additional points from class 2 that are all correctly classified by *both* classifiers. In this case, it is not possible to choose $\{\{P_e[C=c\mid\underline{\mathbf{x}}]\},\ \underline{\mathbf{x}}\in\mathcal{X}_{\text{test}}\}$ to satisfy all the conditional constraints (6). Thus, Problem 1 is infeasible. We first evaluated the TIS algorithm *specialized* to the case where *no* training support augmentation is used, that is, seeking to solve Problem 1. We measured the Lagrangian cost function (34) and the deviation. In Figure 3, we can see the Lagrangian cost function monotonically decreases without convergence and the deviation approaches a fixed value above zero.

We next applied the TIS method ($\beta=0$) with local training set support augmentation. As Figure 4 demonstrates, a feasible solution is achieved in this case.



(a) (b)

Figure 2: An Example of constraint nonsatisfiability.



(a) Lagrangian cost function

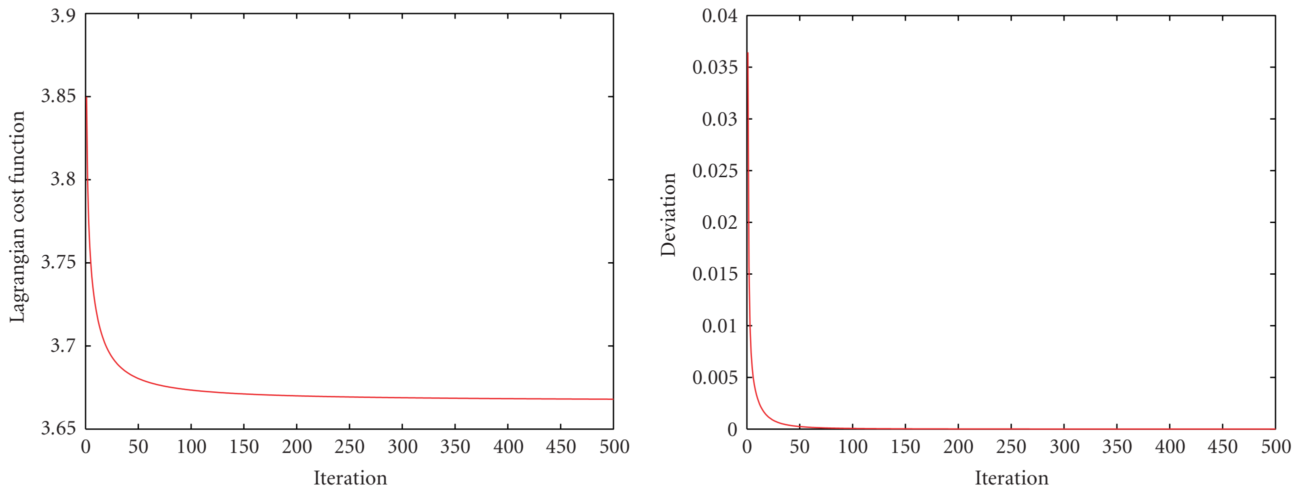
(b) Deviation

Figure 3: Infeasibility of TIS for Figure 2 example when no support augmentation is used.

5.2. *Real data experimental results*

We evaluated on datasets from the UC Irvine machine learning repository in Table 1 and we followed the experimental protocol from [15].

To simulate a distributed classification environment, we used five local classifiers, each a naive Bayes classifier working on a randomly selected subset of features. Thus, for the conditional independence model of Section 3.1, both the local classifiers and the combining rule are based on naive Bayes. For example, *Segment* has 7 classes and 19 continuous features. The first local classifier used 15 randomly selected features while the second classifier used 16. Since all the classifiers use a significant fraction of all the features, this



(a) Lagrangian cost function (b) Deviation

FIGURE 4: Feasible TIS solution for **Figure 2** example when support augmentation is used.

TABLE 1: UC irvine machine learning datasets that were evaluated.

Dataset	No. of class	No. of feature	Feature subspace size
Vehicle	4	18	[15, 14, 15, 16, 17]
Segment	7	19	[15, 16, 15, 17, 18]
Liver disorder	2	6	[5, 4, 5, 4, 5]
Diabetes	2	7	[6, 3, 4, 5, 6]
Sonar	2	60	[52, 54, 56, 58, 59]
Ionosphere	2	32	[28, 27, 29, 30, 31]

TABLE 2: Average test set class priors and mismatch values for different datasets.

Dataset	Class priors on the total dataset	Mismatch	Preset class priors on the test set
Vehicle	[0.24 0.26 0.25 0.25]	0.98	[0.1 0.4 0.11 0.39]
		2.73	[0.04 0.46 0.05 0.45]
		Missing	[0 0.5 0.05 0.45]
Segment	[0.14 0.14 0.14 0.14 0.14 0.14 0.14]	0.066	[0.14 0.14 0.13 0.17 0.16 0.16 0.1]
		1.41	[0.05 0.08 0.27 0.05 0.24 0.26 0.05]
		2.40	[0.03 0.24 0.03 0.04 0.3 0.05 0.31]
		Missing	[0 0.24 0.03 0.04 0.33 0.05 0.31]
Liver disorder	[0.42 0.58]	0.34	[0.6 0.4]
		1.79	[0.8 0.2]
		3.56	[0.9 0.1]
		Missing	[1 0]
Diabetes	[0.67 0.33]	0.74	[0.4 0.6]
		1.43	[0.3 0.7]
		2.57	[0.2 0.8]
		Missing	[1 0]
Sonar	[0.47 0.53]	0.32	[0.3 0.7]
		0.89	[0.2 0.8]
		2.28	[0.1 0.9]
		Missing	[0 1]
Ionosphere	[0.36 0.64]	0.21	[0.5 0.5]
		1.24	[0.1 0.9]
		2.25	[0.05 0.95]
		Missing	[0 1]

introduces statistical dependencies between local classifier feature vectors and, thus, between the local classifiers. All features in the datasets are continuous-valued and were modeled by (class-conditional) Gaussian densities. We performed five replications of two-fold cross-validation for all datasets. For each replication, the dataset was randomly divided into two equal-sized sets. In the first fold, one set was used as training and a subset of the other was used for testing, with these roles reversed in the second fold. We averaged test error rates over all replications and folds. For a given fold, only a subset was used for testing in order to introduce a controlled level of prior mismatch between the training and testing sets. This was accomplished as follows. The original test set priors and training set priors are very similar since they are based on a random, equal-sized split of the whole dataset. To introduce more prior mismatch and, thus, to test the robustness of the various combining schemes to incorrect local class priors, we performed resampling on the test data. For example, suppose in the original test set that there are two classes with priors $[0.5, 0.5]$. To increase the prior mismatch, we set the new test set priors as $[\alpha, 1-\alpha]$. The new test set is then obtained by sampling with replacement from the original test set (to create a dataset of size equal to the original test set size), based on these new priors. Note that sampling with replacement does introduce significant sample duplication in some cases. For example, on *diabetes*, for test class prior distribution [0.2 0.8] (used in Table 2) with 92 unique samples from class 2 and 266 test samples to be drawn, all samples from class 2 are likely to have a duplicate in the resampled test data. Sampling with replacement was used because of the small sample sizes of some of the datasets—sampling without replacement would have severely constrained the test set prior distributions that could have been achieved on these small datasets. Table 2 shows the class priors chosen for the test set, for different datasets, averaged over the ten experimental trials. We quantified the mismatch between test set priors ($P_{\text{test}}[C = c]$) and local training set priors ($P_{\text{trn}}^{(j)}[C = c]$) by the sum of cross-entropies:

$$M = \sum_{j=1}^{M_e} \sum_{c=1}^{N_c} P_{\text{trn}}^{(j)}[C = c] \log \left(\frac{P_{\text{trn}}^{(j)}[C = c]}{P_{\text{test}}[C = c]} \right). \tag{42}$$

These are also shown in Table 2. An extreme case of prior mismatch is when there are *no* examples from one of the classes in the test batch—this class has a zero test set prior probability. We refer to this case as the *missing class* case, also indicated in Table 2. In "missing class" experiments, we randomly selected a single class to be missing from the test batch. The missing class case is interesting for several reasons. First, it represents an extreme case of prior mismatch and thus a severe test of the robustness of a method to prior mismatch. Second, in an application scenario, it may be important to recognize that one of the classes is wholly not present in the testing data. This may provide valuable actionable information beyond mere class decisions. We also note that missing classes during testing is wholly different from the *anomaly detection* problem, wherein a class is present in test data but missing in the *training* data. The anomaly detection problem is not within the scope of this work.

Finally, we note that some datasets used in our experiments are relatively small, which makes it difficult to obtain accurate estimates of generalization accuracy. In particular, for *liver disorder*, *sonar*, and *ionosphere*, the test set size N_{test} does not meet the requirement from [44] that $N_{\text{test}} \geq 100/P_e$, P_e the optimal error rate for the problem. Even when accurate error rate estimates are difficult to obtain, we would like to have statistical confidence on the *relative* performance of the combining schemes. Accordingly, to test the statistical significance of head-to-head comparison of two schemes, we performed the 5×2 cv F test, proposed in [45]. According to [45], we can reject the hypothesis that two algorithms have the same error rate with 95% confidence if the F statistic is greater than 4.74. As seen in the sequel, this testing indicates that there are statistically significant gains in accuracy of our transductive CB method over fixed combining and over the previous CB methods [15] in many of the mismatch experiments.

5.3. Augmentation with support derived from constraints

We first compared the results obtained for TIS using local training support and training support derived from the constraints. We measured classification error rate on the test set and the joint entropy on the test and training supports. The TIS algorithm allocated probability mass between the training sets and test set without any constraint, that is, by letting $\beta = 0$.

From the results of Table 3, we see that the classification performance of TIS with derived training support is very similar to that of TIS with local training support, but with lower joint entropy. The mass allocation on the test set for TIS with local training support is much less than that with derived training support. The reason, we believe, is that since the derived training support is much smaller than the local training support and test support, entropy maximization in the derived support case requires heavy use of the test support, whereas in the local training support case, entropy is maximized making much more use of the augmented (local) training supports. In spite of this disparity in P_u for these two cases, both approaches give very similar classification accuracy.

Note also that on some datasets, for example, *diabetes* and *liver disorder*, the error rate in high mismatch and missing class cases is significantly lower than for low mismatch values. This is explained by recognizing that altering the class priors changes the Bayes-optimal error rate of the classification problem. In some case, the Bayes error rate is made much lower. In the missing class case, it can be further recognized that if the transductive method is successful in identifying that a class is missing (or, at any rate, has very few test instances), the classification problem is effectively made "easier", with fewer classes to discriminate between in classifying the test samples.

Table 3: Average classification performance (test error rate) and joint entropy using local training support and support derived from constraints.

Dataset	Mismatch	Local training support			Derived training support		
		Err	J-Ent	P_u	Err	J-Ent	P_u
Vehicle	0.98	0.531	8.065	0.311	0.532	6.854	0.857
	2.73	0.490	8.035	0.285	0.494	6.669	0.813
	Missing	0.478	8.022	0.273	0.480	8.945	0.773
Segment	0.066	0.176	8.934	0.219	0.172	7.413	0.884
	1.41	0.212	8.940	0.221	0.218	7.370	0.860
	2.40	0.107	8.908	0.200	0.105	7.307	0.867
	Missing	0.129	8.906	0.198	0.101	7.280	0.857
Liver disorder	0.34	0.424	7.067	0.254	0.478	5.711	0.873
	1.79	0.318	7.063	0.249	0.373	5.700	0.868
	3.56	0.248	7.055	0.239	0.290	5.653	0.848
	Missing	0.159	7.050	0.233	0.180	5.621	0.835
Diabetes	0.74	0.269	7.478	0.239	0.269	5.994	0.896
	1.43	0.262	7.472	0.232	0.265	5.934	0.882
	2.57	0.211	7.461	0.221	0.241	5.856	0.872
	Missing	0.027	7.426	0.187	0.010	5.680	0.856

Table 4: Average classification performance (test error rate) and joint entropy for ETIS and naive Bayes in the hard decision case.

Dataset	Mismatch	ETIS		Naive Bayes	
		Err	J-Ent	Err	J-Ent
Vehicle	0.98	0.580	2.782	0.588	5.882
	2.73	0.534	2.527	0.612	5.882
	Missing	0.542	2.381	0.643	5.882
Segment	0.066	0.190	2.885	0.190	4.001
	1.41	0.270	2.844	0.275	4.001
	2.40	0.118	2.482	0.209	4.001
	Missing	0.093	2.763	0.229	4.001
Liver disorder	0.34	0.430	2.191	0.415	3.641
	1.79	0.337	2.050	0.377	3.641
	3.56	0.278	1.907	0.375	3.641
Diabetes	0.74	0.306	2.365	0.282	3.440
	1.43	0.298	2.380	0.297	3.440
	2.57	0.239	2.331	0.306	3.440
	Missing	0.006	2.123	0.189	3.440
Sonar	0.32	0.340	1.360	0.353	3.246
	0.89	0.313	1.315	0.372	3.246
	2.28	0.209	1.259	0.378	3.246
	Missing	0.148	1.960	0.402	3.246
Ionosphere	0.21	0.168	1.394	0.164	2.806
	1.24	0.151	1.187	0.205	2.806
	2.25	0.055	1.075	0.201	2.806
	Missing	0.006	1.449	0.198	2.806

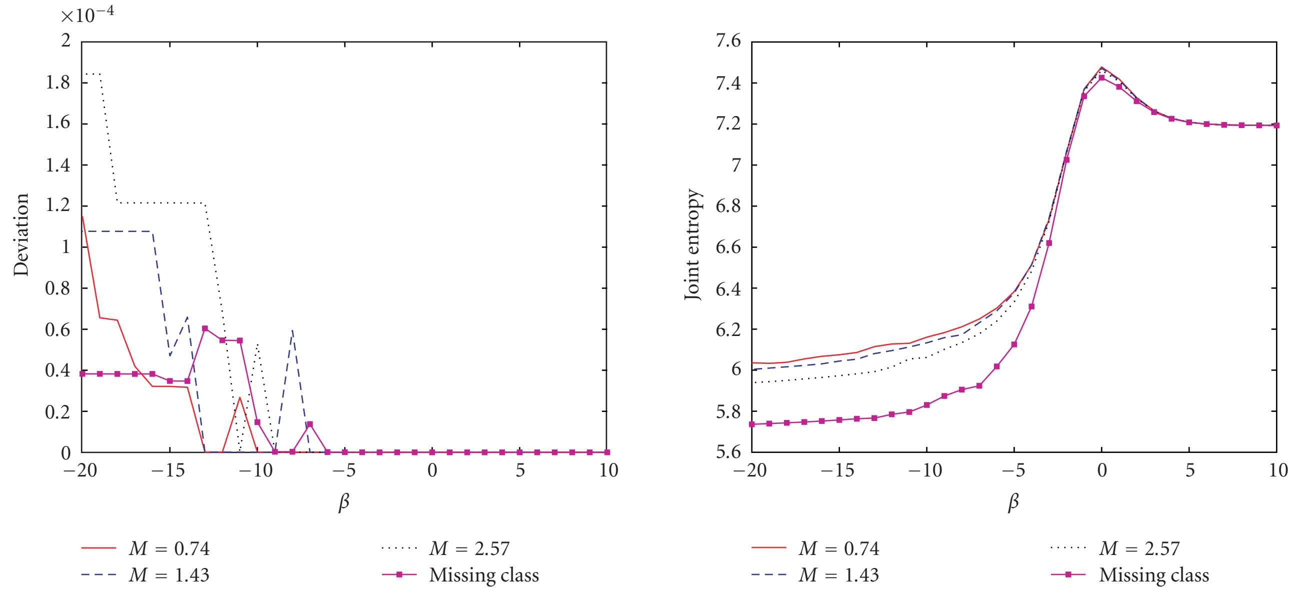
5.4. Full support in the hard decision case

We next considered the case where local classifiers produce hard decisions. We compared ETIS and the naive Bayes solution (see Section 4.3) with respect to test error rate and joint entropy. For the naive Bayes method, since the test set class priors are unknown, a uniform class prior was assumed. From Table 4, we can see the classification performance of ETIS is better than naive Bayes and with much lower joint entropy, as expected based on the discussion in Section 4.3. Note also that we again see the phenomenon of lower error rates in the high mismatch cases on some datasets, especially for the ETIS method.

5.5. Influence of β

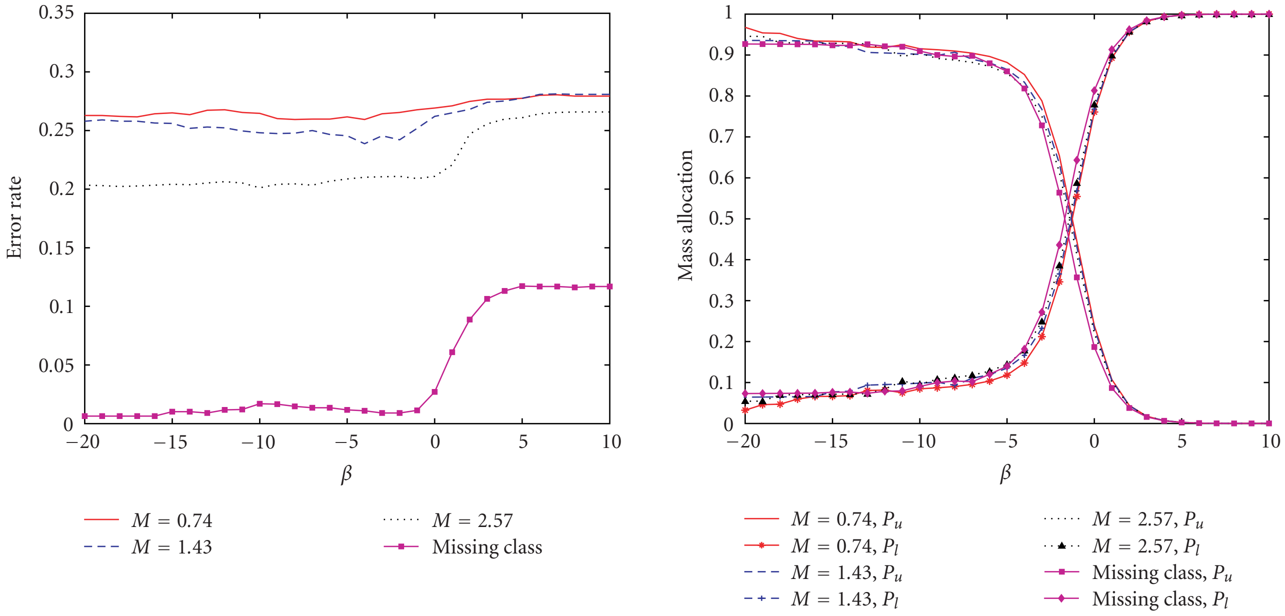
Figure 5 shows the influence of β on the TIS algorithm for *diabetes*. We measured four different values: deviation, joint entropy on the test and training set support, classification error rate on the test dataset, and probability mass allocation (P_u on the test set support and $P_l = 1 - P_u$ on the training set support). We plotted curves for four different mismatch cases. Each curve is based on an average of five replications of two-fold cross-validation. We changed β from -20 to 10 with $\Delta\beta = 1$ and, for each β, we ran the TIS algorithm until the stop criterion $\Delta D < 10^{-9}$.

We can see that when $\beta < -6$, the deviation does not approach 0 when TIS terminates. This demonstrates constraint infeasibility at finite β. As β becomes more negative, the test set support gets more probability mass allocation P_u. However, the rate of P_u increase decreases as β becomes more negative; this occurs because it is difficult to satisfy the local constraints if the TIS algorithm assigns too much probability mass to the test set. The TIS algorithm achieves peak entropy at $\beta = 0$ (as must be the case). When β goes either negative or positive, the joint entropy decreases—there is no constraint on entropy maximization in this case. Note also that for all mismatch values, the best test set error occurs for negative β and the smallest β satisfying the constraints achieves accuracy close to this best error rate.



(a) Deviation

(b) Joint entropy



(c) Classification error rate

(d) Probability mass allocation

Figure 5: The influence of β on TIS algorithm (Diabetes).

As noted in Section 4.7, we used an ETIS bisection search to find the smallest β for which the constraints are satisfied. For this β, the least mass is allocated to the extra support points. The initial search range is from $\beta_b = -100$ (a nonsatisfiable β) to $\beta_a = 0$ (a satisfiable β). For all the datasets we experimented on, Problem 1 was in fact infeasible. Moreover, for each dataset, $\beta = -100$ was negative enough such that the TIS algorithm could not satisfy constraints. We used the deviation as the criterion to determine whether the constraints are satisfied or not; here we chose $D/N_c^2 < 10^{-7}$.

Table 5 gives a comprehensive performance comparison between TIS ($\beta = 0$) and the ETIS algorithm (which stopped at the most negative β at which the constraints were met). ETIS guarantees more mass allocation on the test set support and smaller joint entropy. It has competitive classification performance with TIS and performs especially well in the missing class case. (We again emphasize that, in the missing class case, constraint relaxation never occurred in any of

Table 5: Average classification performance (test error rate), joint entropy, and mass allocation comparison between TIS and ETIS algorithms.

Dataset	Mismatch	TIS ($\beta = 0$)			ETIS (negative β)			
		Err	J-Ent	P_u	β	Err	J-Ent	P_u
Vehicle	0.98	0.531	8.065	0.311	−2.746	0.529	7.536	0.752
	2.73	0.490	8.035	0.285	−2.868	0.490	7.503	0.712
	Missing	0.478	8.022	0.273	−3.114	0.470	7.512	0.664
Segment	0.066	0.176	8.934	0.219	−1.516	0.171	8.691	0.519
	1.41	0.212	8.940	0.221	−1.354	0.217	8.742	0.474
	2.40	0.107	8.908	0.200	−1.386	0.0978	8.685	0.466
	Missing	0.128	8.906	0.198	−1.415	0.0970	8.672	0.461
Liver disorder	0.34	0.424	7.067	0.254	−8.629	0.453	5.940	0.879
	1.79	0.318	7.063	0.249	−9.092	0.360	6.010	0.836
	3.56	0.248	7.055	0.239	−13.593	0.271	5.819	0.848
	Missing	0.159	7.050	0.233	−16.821	0.138	5.683	0.847
Diabetes	0.74	0.269	7.478	0.239	−9.790	0.264	6.191	0.909
	1.43	0.262	7.472	0.232	−11.185	0.253	6.155	0.895
	2.57	0.211	7.461	0.221	−10.668	0.206	6.127	0.881
	Missing	0.027	7.426	0.187	−11.946	0.015	5.915	0.886
Sonar	0.32	0.325	6.529	0.233	−7.916	0.324	5.211	0.942
	0.89	0.305	6.520	0.224	−8.666	0.276	5.201	0.926
	2.28	0.226	6.509	0.212	−13.305	0.200	5.169	0.895
	Missing	0.168	6.501	0.204	−15.334	0.157	5.167	0.865
Ionosphere	0.21	0.157	7.038	0.226	−11.103	0.153	5.670	0.953
	1.24	0.130	7.007	0.200	−6.312	0.115	5.589	0.944
	2.25	0.066	6.999	0.191	−8.860	0.070	5.491	0.949
	Missing	0	6.987	0.179	−14.942	0.009	5.495	0.896

our experiments.) However, neither method dominates the other.

5.6. Comparisons between ETIS, TGD, TML, and fixed rules

In Table 6, we compare transductive ME (ETIS) with the transductive gradient descent (TGD) CB method and the transductive maximum likelihood (TML) algorithms proposed in [15] and also with the fixed rules Sum, Product, Max, and Min [3]. Note that Tables 5 and 6 have some common columns because ETIS results appear in both tables. We can see ETIS achieves overall better error rates than the previous transductive methods and is uniformly better than the fixed rules. In Table 7, we evaluate statistical significance of the gains of ETIS compared head-to-head against the fixed combining methods and the previous CB method. Note that the F-statistic is much larger than 4.74 in many of these comparisons, indicating that gains of ETIS are statistically significant in most mismatch cases compared with fixed methods, and in high mismatch cases compared with TGD and the TML methods, on most of the datasets.

Instead of measuring joint entropy (27), we measured conditional entropy (9) to make a fair entropy comparison between ETIS and TGD. Because TGD only uses the test support set to satisfy constraints, we only measured entropy on the test set as in (9). Note that ETIS has higher conditional entropy than TGD, as we would expect, since TGD does not perform entropy maximization while satisfying constraints.

6. CONCLUSIONS

In this work, we have proposed a new ME framework for transductive learning of decision aggregation rules when there is no common labeled data across local classifiers. The new approach overcomes constraint infeasibility and nonuniqueness of the solution and achieves better results than [15]. While our approach can be directly applied also to the single classifier case, which amounts to a *semisupervised* learning problem, it remains to be seen whether our transductive ME approach offers benefits in classification accuracy over standard supervised learning in this case. Future work should also develop a proof that TIS converges to the ME solution satisfying the constraints, as well as investigate application contexts for the distributed classification problem studied here.

APPENDIX

A. PROOFS

A.1. Proof of feasibility of Problem 2

First, assign all probability mass to the training set supports, that is, choose $P_u = 0$. Further, assign this mass to the

Table 6: Classification performance (test error rate) and conditional entropy comparison between ETIS, TGD, TML, and Sum, Prod, Max, and Min fixed rules.

Dataset	Mismatch	ETIS		TGD		TML Err		Fix Err			
		Err	C-Ent	Err	C-Ent	NB	AA	Sum	Prod	Max	Min
Vehicle	0.98	0.529	0.861	0.587	0.083	0.579	0.589	0.577	0.579	0.575	0.587
	2.73	0.490	0.765	0.603	0.105	0.597	0.600	0.614	0.614	0.617	0.619
	Missing	0.47	0.728	0.623	0.091	0.62	0.631	0.639	0.639	0.638	0.640
Segment	0.066	0.171	0.409	0.187	0.135	0.204	0.218	0.210	0.203	0.201	0.205
	1.41	0.217	0.455	0.239	0.105	0.320	0.307	0.347	0.342	0.365	0.338
	2.40	0.098	0.315	0.173	0.101	0.148	0.133	0.171	0.164	0.188	0.159
	Missing	0.097	0.312	0.169	0.107	0.161	0.127	0.195	0.186	0.216	0.179
Liver disorder	0.34	0.453	0.498	0.447	0.431	0.382	0.388	0.361	0.389	0.379	0.377
	1.79	0.360	0.474	0.357	0.321	0.289	0.285	0.355	0.356	0.363	0.342
	3.56	0.271	0.391	0.288	0.249	0.242	0.219	0.377	0.373	0.392	0.355
	Missing	0.138	0.344	0.170	0.226	0.190	0.138	0.380	0.380	0.403	0.370
Diabetes	0.74	0.264	0.429	0.276	0.025	0.266	0.268	0.285	0.286	0.280	0.299
	1.43	0.253	0.386	0.282	0.002	0.276	0.238	0.312	0.316	0.300	0.336
	2.57	0.206	0.303	0.271	0.01	0.271	0.196	0.349	0.357	0.349	0.367
	Missing	0.015	0.112	0.177	0.014	0.183	0.061	0.323	0.302	0.414	0.251
Sonar	0.32	0.324	0.375	0.341	0.027	0.351	0.350	0.358	0.358	0.358	0.357
	0.89	0.276	0.329	0.316	0.001	0.371	0.368	0.382	0.382	0.382	0.381
	2.28	0.2	0.251	0.333	10^{-6}	0.375	0.371	0.398	0.397	0.407	0.397
	Missing	0.157	0.212	0.341	10^{-6}	0.400	0.381	0.439	0.732	0.448	0.437
Ionosphere	0.21	0.153	0.345	0.16	10^{-4}	0.169	0.172	0.159	0.159	0.161	0.164
	1.24	0.115	0.204	0.200	10^{-6}	0.209	0.171	0.239	0.237	0.255	0.231
	2.25	0.07	0.148	0.193	10^{-6}	0.203	0.162	0.245	0.238	0.261	0.231
	Missing	0.009	0.069	0.189	10^{-5}	0.199	0.146	0.253	0.245	0.271	0.232

Table 7: 5 × 2 cv F test comparing ETIS head-to-head versus TGD, TML, and 4 fixed rules (Sum, Prod, Max, and Min).

Dataset	Mismatch	TGD	TML-NB	TML-AA	Sum	Prod	Max	Min
Vehicle	0.98	2.96	3.63	5.02	5.95	6.88	3.99	5.28
	2.73	5.16	4.94	5.19	23.15	22.20	33.84	15.33
Segment	0.066	1.51	3.63	9.27	5.50	2.99	3.13	4.24
	1.41	1.95	11.68	7.83	19.49	17.68	51.34	22.22
	2.40	1.11	0.74	0.61	1.10	0.96	1.74	0.89
	Missing	7.85	5.90	1.75	6.62	7.77	7.96	6.16
Liver disorder	0.34	0.61	1.00	0.94	1.01	0.93	1.01	0.94
	1.79	0.61	0.66	0.65	0.60	0.60	0.60	0.60
	3.56	0.65	0.67	0.68	0.68	0.67	0.74	0.64
	Missing	0.62	0.63	0.61	1.17	1.16	1.56	1.08
Diabetes	0.74	1.91	0.81	1.15	4.48	10.59	1.31	4.91
	1.43	10.05	6.21	1.09	47.99	62.03	10.62	23.28
	2.57	6.07	5.46	0.63	29.55	36.44	8.17	28.94
	Missing	23.14	40.47	2.27	150.60	89.70	111.60	168.84
Sonar	0.32	0.85	0.88	0.89	1.04	1.01	0.92	1.07
	0.89	1.05	1.88	1.87	1.83	1.83	1.77	2.06
	2.28	2.21	3.52	3.66	3.79	3.84	3.92	3.93
	Missing	2.14	3.27	3.68	3.76	3.67	3.65	3.81
Ionosphere	0.21	0.99	1.35	1.42	0.82	0.82	1.11	1.00
	2.25	7.35	7.39	1.78	10.31	10.37	12.16	11.09
	Missing	9.51	9.75	2.40	18.88	15.02	20.27	14.68

training supports in a uniform manner, that is, by the choice $P[\underline{\tilde{\mathbf{x}}}^{(j)}, c^{(j)}] = (1/N_j)\cdot(1/M_e)\ \forall(\underline{\tilde{\mathbf{x}}}^{(j)}, c^{(j)}) \in \widetilde{\mathcal{X}}^{(j)}$. With this choice, we have

$$\begin{aligned}
P_e\Big[\hat{C}_j = \hat{c} \mid C = c,\ \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}\Big]
&= \frac{f' P_j\Big[\hat{C}_j = \hat{c} \mid \underline{\tilde{\mathbf{x}}}^{(j)}\Big]\cdot(1/N_j)\cdot(1/M_e)}{f'(1/N_j)\cdot(1/M_e)} \\
&= \frac{f' P_j\Big[\hat{C}_j = \hat{c} \mid \underline{\tilde{\mathbf{x}}}^{(j)}\Big]}{f' 1} \\
&= P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c],
\end{aligned} \tag{A.1}$$

where f' denotes $\sum_{(\underline{\tilde{\mathbf{x}}}^{(j)}, c^{(j)}) \in \widetilde{\mathcal{X}}^{(j)}}$. Thus, Problem 2 is feasible.

A.2. Proof sketch of feasibility of the problem with support augmentation derived from the constraints

This proof follows the same strategy as Appendix A.1. We again first let $P_u = 0$, assigning all probability to the support points derived from the constraints. On these support points, we make the probability assignments $P[[0, 0, \ldots, 0, 1, 0, \ldots 0], c] = (1/N_c)P_g[\hat{C}_j = \hat{c} \mid C = c]$, where the "1" is in the $\hat{c}$-th entry $\forall(c, \hat{c})$. With these assignments, one again finds, similar to Appendix A.1, that $P_e[\hat{C}_j = \hat{c} \mid C = c,\ \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}] = P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c]$.

A.3. Proof that TIS updates descend in the Lagrangian

For fixed $P_e[c \mid \underline{\mathbf{x}}]$, $P[\underline{\tilde{\mathbf{x}}}^{(j)}, \tilde{c}^{(j)}]$, P_u, and β, when updating any single Lagrange multiplier $\gamma(\hat{C}_j = \hat{c}, C = c)$ using the TIS update rule (39) given the remaining Lagrange multipliers held fixed, the change in the Lagrangian (34) can be written as

$$\begin{aligned}
&L[\gamma(\hat{C}_j = \hat{c}, C = c) + \Delta\gamma(\hat{C}_j = \hat{c}, C = c)] \\
&\qquad - L[\gamma(\hat{C}_j = \hat{c}, C = c)] \\
&\quad = \Delta\gamma(\hat{C}_j = \hat{c}, C = c) \\
&\quad \cdot\Big(N_e\Big[\hat{C}_j = \hat{c}, C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}\Big] \\
&\qquad\qquad - P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c] N_e[C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}]\Big) \\
&\quad = h'\cdot\Big(N_e\Big[\hat{C}_j = \hat{c}, C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}\Big] \\
&\qquad\qquad - P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c] N_e[C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}]\Big) \\
&\quad \leq 0,
\end{aligned} \tag{A.2}$$

where h' denotes $\log(P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c] N_e[C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}] / N_e[\hat{C}_j = \hat{c}, C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}])$.

The final inequality is verified as follows. First, note that $N_e[C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}]$ and $N_e[\hat{C}_j = \hat{c}, C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}]$ are both nonnegative and $P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c] \in [0, 1]$. Thus, the logarithm is well defined. Next, observe that if $N_e[\hat{C}_j = \hat{c}, C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}] < P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c] N_e[C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}]$, then $\log(P_g^{(j)}[\hat{C}_j = \hat{c} \mid C = c] N_e[C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}] / N_e[\hat{C}_j = \hat{c}, C = c \mid \underline{\mathbf{x}} \in \mathcal{X}_r^{(j)}])$ is positive, otherwise, it is negative. Thus, in either case, $L[\gamma(\hat{C}_j = \hat{c}, C = c) + \Delta\gamma(\hat{C}_j = \hat{c}, C = c)] - L[\gamma(\hat{C}_j = \hat{c}, C = c)]$ is negative, and the Lagrangian is thus decreasing with TIS updates.

REFERENCES

[1] J. Basak and R. Kothari, "A classification paradigm for distributed vertically partitioned data," *Neural Computation*, vol. 16, no. 7, pp. 1525–1544, 2004.

[2] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[3] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas, "On combining classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 226–239, 1998.

[4] J. J. Rodríguez, L. I. Kuncheva, and C. J. Alonso, "Rotation forest: a new classifier ensemble method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1619–1630, 2006.

[5] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proceedings of the 13th International Conference on Machine Learning (ICML '96)*, pp. 148–156, Bari, Italy, July 1996.

[6] R. Maclin, "Boosting classifiers regionally," in *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI '98)*, pp. 700–705, Madison, Wis, USA, July 1998.

[7] X. Nguyen, M. J. Wainwright, and M. I. Jordan, "Nonparametric decentralized detection using kernel methods," *IEEE Transactions on Signal Processing*, vol. 53, no. 11, pp. 4053–4066, 2005.

[8] R. E. Schapire, "The boosting approach to machine learning: an overview," in *MSRI Workshop on Nonlinear Estimation and Classification*, Springer, Berkeley, Calif, USA, March 2002.

[9] K. Tumer and J. Ghosh, "Error correlation and error reduction in ensemble classifiers," *Connection Science*, vol. 8, no. 3-4, pp. 385–404, 1996.

[10] G. Valentini and T. G. Dietterich, "Bias-variance analysis of support vector machines for the development of SVM-based ensemble methods," *Journal of Machine Learning Research*, vol. 5, pp. 725–775, 2004.

[11] T. G. Dietterich, "Machine-learning research: four current directions," *AI Magazine*, vol. 18, no. 4, pp. 97–136, 1997.

[12] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[13] R. P. W. Duin and D. M. J. Tax, "Experiments with classifier combining rules," in *Proceedings of the 1st International Workshop on Multiple Classifier Systems (MCS '00)*, J. Kittler and F. Roli, Eds., pp. 16–29, Cagliari, Italy, June 2000.

[14] A. D'Costa, V. Ramachandran, and A. M. Sayeed, "Distributed classification of Gaussian space-time sources in wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 6, pp. 1026–1036, 2004.

[15] D. J. Miller and S. Pal, "Transductive methods for the distributed ensemble classification problem," *Neural Computation*, vol. 19, no. 3, pp. 856–884, 2007.

[16] S.-Y. Kung and J.-N. Hwang, "Neural networks for intelligent multimedia processing," *Proceedings of the IEEE*, vol. 86, no. 6, pp. 1244–1271, 1998.

[17] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proceedings of the 16th International Conference on Machine Learning (ICML '99)*, pp. 200–209, San Francisco, Calif, USA, June 1999.

[18] M. Saerens, P. Latinne, and C. Decaestecker, "Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure," *Neural Computation*, vol. 14, no. 1, pp. 21–41, 2002.

[19] D. J. Miller and H. S. Uyar, "A mixture of experts classifier with learning based on both labelled and unlabelled data," in *Advances in Neural Information Processing Systems 9*, pp. 571–577, MIT Press, Cambridge, Mass, USA, 1997.

[20] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the EM algorithm," *Neural Computation*, vol. 6, no. 2, pp. 181–214, 1994.

[21] H.-J. Kang, K. Kim, and J. H. Kim, "Optimal approximation of discrete probability distribution with *k*th-order dependency and its application to combining multiple classifiers," *Pattern Recognition Letters*, vol. 18, no. 6, pp. 515–523, 1997.

[22] S. Pal and D. J. Miller, "An extension of iterative scaling for decision and data aggregation in ensemble classification," *Journal of VLSI Signal Processing*, vol. 48, no. 1-2, pp. 21–37, 2007.

[23] N. Ueda and R. Nakano, "Combining discriminant-based classifiers using the minimum classification error discriminant," in *Proceedings of the 7th IEEE Workshop on Neural Networks for Signal Processing (NNSP '97)*, pp. 365–374, Amelia Island, Fla, USA, September 1997.

[24] C. Ferri and J. Hernández-Orallo, "Cautious classifiers," in *Proceedings of the 1st International Workshop on ROC Analysis in Artificial Intelligence (ROCAI '04)*, pp. 27–36, Valencia, Spain, August 2004.

[25] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[26] 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*, vol. 22, no. 3, pp. 418–435, 1992.

[27] D. Tao, X. Tang, X. Li, and X. Wu, "Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 7, pp. 1088–1099, 2006.

[28] D. J. Miller and L. Yan, "Critic-driven ensemble classification," *IEEE Transactions on Signal Processing*, vol. 47, no. 10, pp. 2833–2844, 1999.

[29] K. Woods, W. P. Kegelmeyer Jr., and K. Bowyer, "Combination of multiple classifiers using local accuracy estimates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 405–410, 1997.

[30] S. Berg, "Condorcet's Jury Theorem and the reliability of majority voting," *Group Decision and Negotiation*, vol. 5, no. 3, pp. 229–238, 1996.

[31] P. K. Varshney, *Distributed Detection and Data Fusion*, Springer, New York, NY, USA, 1997.

[32] D. J. Miller, S. Pal, and Y. Wang, "Constraint-based, transductive learning for distributed ensemble classification," in *Proceedings of the 16th IEEE Signal Processing Society Workshop on Machine Learning for Signal Processing (MLSP '06)*, pp. 15–20, Maynooth, Ireland, September 2006.

[33] D. J. Miller, S. Pal, and Y. Wang, "Extensions of transductive learning for distributed ensemble classification and application to biometric authentication," to appear in *Neurocomputing*.

[34] A. L. Berger, S. A. Della Pietra, and V. J. Della Pietra, "A maximum entropy approach to natural language processing," *Computational Linguistics*, vol. 22, no. 1, pp. 39–71, 1996.

[35] J. N. Darroch and D. Ratcliff, "Generalized iterative scaling for log-linear models," *The Annals of Mathematical Statistics*, vol. 43, no. 5, pp. 1470–1480, 1972.

[36] S. A. Della Pietra, V. J. Della Pietra, and J. Lafferty, "Inducing features of random fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 380–393, 1997.

[37] E. T. Jaynes, *Papers on Probability, Statistics, and Statistical Physics*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1989.

[38] A. Dempster, N. Laird, and D. Rubin, "Maximum-likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society B*, vol. 39, no. 1, pp. 1–38, 1977.

[39] J. R. Parker, "Rank and response combination from confusion matrix data," *Information Fusion*, vol. 2, no. 2, pp. 113–120, 2001.

[40] D. J. Miller and L. Yan, "An approximate maximum entropy method for feature inference: relation to other maximum entropy methods and to naive Bayes," in *Proceedings of the 34th Annual Conference on Information Sciences and Systems (CISS '00)*, pp. 1–6, Princeton, NJ, USA, March 2000.

[41] S. C. Zhu, Y. N. Wu, and D. Mumford, "Minimax entropy principle and its application to texture modeling," *Neural Computation*, vol. 9, no. 8, pp. 1627–1660, 1997.

[42] M. Collins, R. E. Schapire, and Y. Singer, "Logistic regression, AdaBoost and Bregman distances," in *Proceedings of the 13th Annual Conference on Learning Theory (COLT '00)*, pp. 158–169, Morgan Kaufmann, Palo Alto, Calif, USA, June-July 2000.

[43] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, New York, NY, USA, 2004.

[44] I. Guyon, J. Makhoul, R. Schwartz, and V. Vapnik, "What size test set gives good error rate estimates?" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 52–64, 1998.

[45] E. Alpaydin, "Combined 5 × 2 cv F test for comparing supervised classification learning algorithms," *Neural Computation*, vol. 11, no. 8, pp. 1885–1892, 1999.