SemiBoost: Boosting for Semi-supervised Learning

Pavan Kumar Mallapragada, *Student Member, IEEE*, Rong Jin, *Member, IEEE*, Anil K. Jain, *Fellow, IEEE*, and Yi Liu, *Student Member, IEEE*,

***Abstract*—Semi-supervised learning has attracted a significant amount of attention in pattern recognition and machine learning. Most previous studies have focused on designing special algorithms to effectively exploit the unlabeled data in conjunction with labeled data. Our goal is to improve the classification accuracy of *any* given supervised learning algorithm by using the available unlabeled examples. We call this as the *Semi-supervised improvement* problem, to distinguish the proposed approach from the existing approaches. We design a meta-semi-supervised learning algorithm that wraps around the underlying supervised algorithm, and improves its performance using unlabeled data. This problem is particularly important when we need to train a supervised learning algorithm with a limited number of labeled examples and a multitude of unlabeled examples. We present a *boosting* framework for semi-supervised learning, termed as SemiBoost. The key advantages of the proposed semi-supervised learning approach are: (a) performance improvement of any supervised learning algorithm with a multitude of unlabeled data, (b) efficient computation by the iterative boosting algorithm, and (c) exploiting both manifold and cluster assumption in training classification models. An empirical study on 16 different datasets, and on text categorization demonstrates that the proposed framework improves the performance of several commonly used supervised learning algorithms, given a large number of unlabeled examples. We also show that the performance of the proposed algorithm, SemiBoost, is comparable to the state-of-the-art semi-supervised learning algorithms.**

***Index Terms*—Machine learning, Semi-supervised learning, Semi-supervised improvement, Manifold assumption, Cluster assumption, Boosting**

I. Introduction

Semi-supervised learning has received a significant interest in pattern recognition and machine learning. While semi-supervised classification is a relatively new field, the idea of using unlabeled samples for prediction was conceived several decades ago. The initial work in semi-supervised learning is attributed to H. J. Scudders for his work on "self-learning" in 1965 [1]. An earlier work by Robbins and Monro [2] on sequential learning can also be viewed as related to semi-supervised learning. The key idea of semi-supervised learning, specifically semi-supervised classification, is to exploit both labeled and unlabeled data to learn a classification model. Enormous amount of data is being generated everyday in the form of news articles, documents, images and email to name a few. Most of the generated data is uncategorized or unlabeled, thereby making it difficult to use supervised approaches to automate applications like personal news filtering, email spam filtering, and document and image classification. Typically, there is only a small amount of labeled data available, for example, based on which articles a user marks interesting, or which email he marks as spam, but there is a huge amount of data that has not been marked. As a result, there is an immense need for algorithms that can utilize the small amount of labeled data, combined with the large amount of unlabeled data to build efficient classification systems.

Existing semi-supervised classification algorithms may be classified into two categories based on their underlying assumptions. An algorithm is said to satisfy the *manifold assumption* if it utilizes the fact that the data lie on a low-dimensional manifold in the input space. Usually, the underlying geometry of the data is captured by representing the data as a graph, with samples as the vertices, and the pairwise similarities between the samples as edge-weights. Several graph based algorithms such as Label propagation [3], [4], Markov random walks [5], Graph cut algorithms [6], Spectral graph transducer [7], and Low density separation [8] proposed in the literature are based on this assumption.

Several algorithms have been proposed for semi-supervised learning which are naturally inductive. Usually, they are based on an assumption, called the *cluster assumption* [9]. It states that the data samples with high similarity between them, must share the same label. This may be equivalently expressed as a condition that the decision boundary between the classes must pass through low density regions. This assumption allows the unlabeled data to regularize the decision boundary, which in turn influences the choice of classification models. Many successful semi-supervised algorithms like TSVM [10] and Semi-supervised SVM [11] follow this approach. These algorithms assume a model for the decision boundary, resulting in an inductive classifier.

Manifold regularization [12] is another inductive approach, that is built on the manifold assumption. It attempts to build a maximum-margin classifier on the data, while minimizing the corresponding inconsistency with the similarity matrix. This is achieved by adding a graph-based regularization term to an SVM based objective function. A related approach called LIAM [13] regularizes the SVM decision boundary using a priori metric information encoded into the Graph Laplacian, and has a fast optimization algorithm.

Most semi-supervised learning approaches design specialized learning algorithms to effectively utilize both labeled and unlabeled data. However, it is often the case that a user already has a favorite (well-suited) supervised learning algorithm for his application, and would like to improve its performance by utilizing the available unlabeled data. In this light, a more practical approach is to design a technique to utilize the unlabeled samples, regardless of the underlying learning algorithm. Such an approach would accommodate for the task-based selection of a classifier, while providing it with an ability to utilize unlabeled data effectively. We refer to this problem of improving the performance of *any* supervised learning algorithm using unlabeled data as *Semi-supervised Improvement*, to distinguish our work from the standard semi-

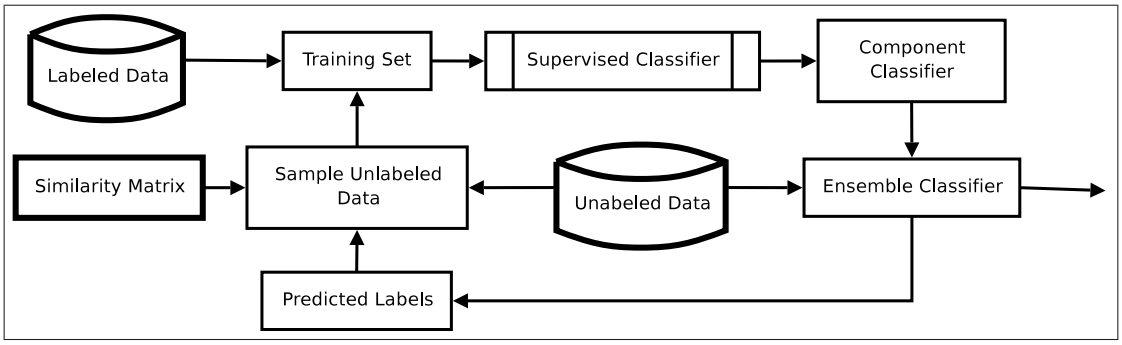


Fig. 1. Block diagram of the proposed algorithm, SemiBoost. The inputs to SemiBoost are: labeled data, unlabeled data and the similarity matrix.

supervised learning problems.

To address the semi-supervised improvement, we propose a boosting framework, termed *SemiBoost*, for improving a given supervised learning algorithm with unlabeled data. Similar to most boosting algorithms [14], SemiBoost improves the classification accuracy iteratively. At each iteration, a number of unlabeled examples will be selected and used to train a new classification model using the given supervised learning algorithm. The trained classification models from each iteration are combined linearly to form a final classification model. An overview of the SemiBoost is presented in Figure 1. The key difficulties in designing SemiBoost are: (1) how to sample the unlabeled examples for training a new classification model at each iteration, and (2) what class labels should be assigned to the selected unlabeled examples. It is important to note that unlike supervised boosting algorithms where we select labeled examples that are difficult to classify, SemiBoost needs to select unlabeled examples, at each iteration.

One way to address the above questions is to exploit both the clustering assumption and the large margin criterion. One can improve the classification margin by selecting the unlabeled examples with the highest classification confidence, and assign them the class labels that are predicted by the current classifier. The assigned labels are hereafter referred to as the *pseudo-labels*. The labeled data, along with the selected pseudo-labeled data are utilized in the next iteration for training a second classifier. This is broadly the strategy adopted by approaches like Self-training [15], ASSEMBLE [16] and Semi-supervised MarginBoost [17]. However, a problem with this strategy is that the introduction of examples with predicted class labels may only help to increase the classification margin, without actually providing any novel information to the classifier. Since the selected unlabeled examples are the ones that can be classified confidently, they often are far away from the decision boundary. As a result, the classifier trained by the selected unlabeled examples is likely to share the same decision boundary with the original classifier that was trained only by the labeled examples. This is because by adjusting the decision boundary, the examples with high classification confidence will gain even higher confidence. This implies that we may need additional guidance for improving the base classifier, along with the maximum margin criterion.

To overcome the above problem, we propose to use the pairwise similarity measurements to guide the selection of unlabeled examples at each iteration, as well as for assigning class labels to them. For each unlabeled example $\mathbf{x}_i$, we compute the confidence of assigning the example $\mathbf{x}_i$ to the positive class as well as the confidence of assigning it to the negative class. These two confidences are computed based on the prediction made by the boosted classifier and the similarity among different examples. We then select the examples with the highest classification confidence together with the labeled examples to train a new classification model at each iteration. The new classification model will be combined linearly with the existing classification models to make improved predictions. Note that the proposed approach is closely related to graph-based semi-supervised learning approaches that exploit the manifold assumption. The following section discusses the existing semi-supervised learning methods, and their relationship with SemiBoost.

II. Related work

Table I presents a brief summary of the existing semi-supervised learning methods and the underlying assumptions. The first column shows the assumptions on the data used in the algorithm. The second column gives the name of the approach with its reference, followed by a brief description of the method in column 3. Column 4 specifies if the algorithm is naturally inductive (I) or transductive (T). An inductive algorithm can be used to predict the labels of samples that are unseen during training (irrespective of it being labeled or unlabeled), on the other hand, transductive algorithms are limited to predicting only the labels of the unlabeled samples seen during training.

Graph-based approaches represent both the labeled and the unlabeled examples by a connected graph, in which each example is represented by a vertex, and pairs of vertices are connected by an edge if the corresponding examples have large similarity. The well known approaches in this category include Harmonic Function based approach [18], Spectral Graph Transducer (SGT) [7], Gaussian process based approach [23], Manifold Regularization [12] and Label Propagation approach [3], [4]. The optimal class labels for the unlabeled examples are found by minimizing their inconsistency with both the supervised class labels and the graph structure.

A popular way to define the inconsistency between the labels $\mathbf{y} = \{y_i\}_{i=1}^n$ of the samples $\{\mathbf{x}_i\}_{i=1}^n$, and the pairwise

TABLE I
A SUMMARY OF SEMI-SUPERVISED CLASSIFICATION ALGORITHMS. T AND I IN THE LAST COLUMN DENOTE TRANSDUCTIVE AND INDUCTIVE PROPERTY OF THE ALGORITHM, RESPECTIVELY.

Group	Approach	Summary	T/I
Manifold Assumption	Label Propagation [3], [4]	Graph-based; Maximize label consistency using Graph Laplacian	T
	Min-cuts [6]	Edge-weight based graph-partitioning algorithm constraining nodes with same label to be in same partition	T
	MRFs [5], GRFs [18]	Markov random field and Gaussian random field models	T
	LDS [19]	TSVM trained on a dimensionality reduced data using graph-based kernel	T
	SGT [7]	Classification cost minimized with a Laplacian regularizer	T
	LapSVM [12]	SVM with Laplacian regularization	I
Cluster Assumption	Co-training [20]	Maximizes predictor consistency among two distinct feature views	I
	Self-training [15]	Assumes pseudo-labels as true labels and retrains the model	I
	SSMB [17]	Maximizes pseudo-margin using boosting	I
	ASSEMBLE [16]	Maximizes pseudo-margin using boosting	I
	Mixture of Experts [21]	EM based model-fitting of mixture models	I
	EM-Naive Bayes [22]	EM based model-fitting of Naive Bayes	I
	TSVM [10], S3VM [11]	Margin maximization using density of unlabeled data	I
	Gaussian processes [23]	Bayesian discriminative model	I
Manifold & Cluster Assumptions	SemiBoost (Proposed)	Boosting with a graph Laplacian inspired regularization	I

similarities $S_{i,j}$ is the quadratic criterion,

$$F(\mathbf{y}) = \sum_{i=1}^{n}\sum_{j=1}^{n} S_{i,j}(y_i - y_j)^2 = \mathbf{y}^T L \mathbf{y}$$

where L is the combinatorial graph Laplacian. Given a semi-supervised setting, only a few labels in the above consistency measure are assumed to be known, and the rest are considered unknown. The task is to assign values to the unknown labels in such a way that the overall inconsistency is minimized. The approach presented in [6] considers the case when $y_i \in \{\pm 1\}$, thereby formulating it as a discrete optimization problem and solve it using a min-cut approach. Min-cuts are however prone to degenerate solutions, and hence the objective was minimized using a mixed integer programming approach in [24], which is computationally prohibitive [11]. A continuous relaxation of this objective function, where $y_i \in [0, 1]$ has been considered in several approaches, which is solved using Markov random fields [5], Gaussian random fields and harmonic functions [18].

The proposed framework is closely related to the graph-based approaches in the sense that it utilizes the pairwise similarities for semi-supervised learning. The inconsistency measure used in the proposed approach follows a similar definition, except that an exponential cost function is used instead of a quadratic cost for violating the labels. Unlike most graph-based approaches, we create a specific classification model by learning from both the labeled and the unlabeled examples. This is particularly important for semi-supervised improvement, whose goal is to improve a given supervised learning algorithm with massive amounts of unlabeled data.

The approaches built on cluster assumption utilize the unlabeled data to regularize the decision boundary. In particular, the decision boundary that passes through the region with low density of unlabeled examples is preferred to the one that is densely surrounded with unlabeled examples. These methods specifically extend SVM or related maximum margin classifiers, and are not easily extensible to non-margin based classifiers like decision trees. Approaches in this category include transductive support vector machine (TSVM) [10], Semi-supervised Support Vector Machine (S3VM) [11], and Gaussian processes with null category noise model [23]. The proposed algorithm, on the other hand, is a general approach which allows the choice of a base classifier well-suited to the specific task.

Finally, we note that the proposed approach is closely related to the family of ensemble approaches for semi-supervised learning. Ensemble methods have gained significant popularity under the realm of supervised classification, with the availability of algorithms such as AdaBoost [25]. The semi-supervised counter parts of ensemble algorithms rely on the cluster assumption, and prime examples include ASSEMBLE [16] and Semi-supervised MarginBoost (SSMB) [17]. Both these algorithms work by assigning a pseudo-label to the unlabeled samples, and then sampling them for training a new supervised classifier. SSMB and ASSEMBLE are margin-based boosting algorithms which minimize a cost function of the form

$$J(H) = C(y_i H(x_i)) + C(|H(x_i)|),$$

where H is the ensemble classifier under construction, and C is a monotonically decreasing cost function. The term $y_i H(x_i)$ corresponds to the margin definition for labeled samples. A margin definition involves the true label y_i, which is not available for the unlabeled samples. A pseudo-margin definition is used such as $|H(x_i)|$ in ASSEMBLE, or $H(x_i)^2$ in SSMB,

- Start with an empty ensemble.
- At each iteration,
  - Compute the peusdolabel (and its confidence) for each unlabeled example (using existing ensemble, and the pairwise similarity).
  - Sample most confident pseudolabeled examples; combine them with the labeled samples and train a component classifier using the supervised learning algorithm $\mathcal{A}$.
  - Update the ensemble by including the component classifier with an appropriate weight.

Fig. 2. An outline of the SemiBoost algorithm for semi-supervised improvement.

thereby getting rid of the y_i term in the objective function using the fact that $y_i \in \{\pm 1\}$. However, the algorithm relies on the prediction of pseudo-labels using the existing ensemble classifier at each iteration. In contrast, the proposed algorithm combines the similarity information along with the classifier predictions to obtain more reliable pseudo-labels, which is notably different from the existing approaches. SSMB on the other hand requires the base learner to be a semi-supervised algorithm in itself [17], [16]. Therefore, it is solving a different problem of boosting semi-supervised algorithms, in contrast with the proposed algorithm.

In essence, the SemiBoost algorithm combines the advantages of graph based and ensemble methods, resulting in a more general and powerful approach for semi-supervised learning.

III. Semi-supervised boosting

We first describe the semi-supervised improvement problem formally, and then present the SemiBoost algorithm.

A. Semi-supervised improvement

Let $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ denote the entire dataset, including both the labeled and the unlabeled examples. Suppose that the first n_l examples are labeled, given by $\mathbf{y}_l = (y_1^l, y_2^l, \ldots, y_{n_l}^l)$, where each class label y_i^l is either $+1$ or -1. We denote by $\mathbf{y}_u = (y_1^u, y_2^u, \ldots, y_{n_u}^u)$, the imputed class labels of unlabeled examples, where $n_u = n - n_l$. Let the labels for the entire dataset be denoted as $\mathbf{y} = [\mathbf{y}_l; \mathbf{y}_u]$. Let $S = [S_{i,j}]_{n \times n}$ denote the symmetric similarity matrix, where $S_{i,j} \geq 0$ represents the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$. Let $\mathcal{A}$ denote the given supervised learning algorithm. The goal of semi-supervised improvement is to improve the performance of $\mathcal{A}$ iteratively by treating $\mathcal{A}$ like a black box, using the unlabeled examples and the pairwise similarity S. A brief outline of the SemiBoost algorithm for semi-supervised improvement is presented in Figure 2.

It is important to distinguish the problem of semi-supervised improvement from the existing semi-supervised classification approaches. As discussed in section 2, any ensemble based algorithm must rely on the pseudo-labels for building the next classifier in the ensemble. On the other hand, graph based algorithms use the pairwise similarities between the samples, and assign the labels to unlabeled samples such that they are consistent with the similarity. In the semi-supervised improvement problem, we aim to build an ensemble classifier which utilizes the unlabeled samples in the way a graph based approach would utilize.

B. SemiBoost

To improve the given learning algorithm $\mathcal{A}$, we follow the idea of boosting by running the algorithm $\mathcal{A}$ iteratively. A new classification model will be learned at each iteration using the algorithm $\mathcal{A}$, and the learned classification models at different iterations will be linearly combined to form the final classification model.

1) Objective function: The unlabeled samples must be assigned labels following two main criteria: (a) the points with high similarity among unlabeled samples must share the same label, (b) those unlabeled samples which are highly similar to a labeled sample must share its label. Our objective function $F(\mathbf{y}, S)$ is a combination of two terms, one measuring the inconsistency between labeled and unlabeled examples $F_l(\mathbf{y}, S)$, and the other measuring the inconsistency among the unlabeled examples $F_u(\mathbf{y}_u, S)$.

Inspired by the harmonic function approach, we define $F_u(\mathbf{y}, S)$, the inconsistency between class labels $\mathbf{y}$ and the similarity measurement S, as

$$F_u(\mathbf{y}_u, S) \quad = \quad \sum_{i,j=1}^{n_u} S_{i,j} \exp(y_i^u - y_j^u). \tag{1}$$

Many objective functions using similarity or kernel matrices, require the kernel to be positive semi-definite to maintain the convexity of the objective function (e.g., SVM). However, since $\exp(x)$ is a convex function[1], and we assume that $S_{i,j}$ is non-negative $\forall i, j$, the function $F_u(\mathbf{y}_u, S)$ is convex irrespective of the positive definiteness of the similarity matrix. This allows similarity matrices which are asymmetric (e.g., similarity computed using KL-divergence) without changing the convexity of the objective function. Asymmetric similarity matrices arise when using directed graphs for modeling classification problems, and are shown to perform better in certain applications related to text categorization [26].

Though our approach can work for general similarity matrices, we assume that the similarity matrix provided is symmetric. Note that Eq (1) can be expanded as $F_u(\mathbf{y}_u, S) = \frac{1}{2}\sum S_{j,i} \exp(y_j^u - y_i^u) + \frac{1}{2}\sum S_{i,j} \exp(y_i^u - y_j^u)$, and due to the symmetry of S, we have

$$F_u(\mathbf{y}_u, S) \quad = \quad \sum_{i,j=1}^{n_u} S_{i,j} \cosh(y_i^u - y_j^u), \tag{2}$$

where $\cosh(y_i - y_j) = (\exp(-y_i + y_j) + \exp(y_i - y_j))/2$ is the hyperbolic cosine function. Note that $\cosh(x)$ is a convex function with its minimum at $x = 0$. Rewriting Eq (1) using the $\cosh(.)$ function reveals the connection

[1]Our choice of $F(y, S)$ is a mixture of exponential loss functions, and is motivated by the traditional exponential loss used in Boosting and the resulting large margin classifier. However, Any convex (monotonic) loss function should work with the current framework.

between the quadratic penalty used in the graph Laplacian based approaches, and the exponential penalty used in the current approach. Using a $\cosh(.)$ penalty not only facilitates the derivation of boosting based algorithms but also increases the classification margin. The utility of an exponential cost for boosting algorithms is well known [27].

The inconsistency between labeled and unlabeled examples $F_l(\mathbf{y}, S)$ is defined as

$$F_l(\mathbf{y}, S) = \sum_{i=1}^{n_l} \sum_{j=1}^{n_u} S_{i,j} \exp(-2y_i^l y_j^u). \quad (3)$$

Combining Eqs (1) and (3) leads to the objective function,

$$F(\mathbf{y}, S) = F_l(\mathbf{y}, S) + CF_u(\mathbf{y_u}, S). \quad (4)$$

The constant C is introduced to weight the importance between the labeled and the unlabeled data. Given the objective function in (4), the optimal class label $\mathbf{y}_u$ is found by minimizing F.

Let $\hat{y}_i^l, i = 1, \cdots, n_l$ denote the labels predicted by the learning algorithm over the labeled examples in the training data. Note that in Eq (4), there is no term corresponding to the inconsistency between predicted labels of the labeled samples and their true labels, which would be $F_{ll} = \sum_{i=1}^{n_l} \exp(y_i^l, \hat{y}_i^l)$. Adding this term would make the algorithm specialize to AdaBoost when no unlabeled samples are present. Since in practice, there is a limited amount of labeled data available, the F_{ll} term is usually significantly smaller than F_l and F_u, and therefore is omitted in the formulation in Eq (4).

selecting an even smaller subset of samples to train the classifier may not be effective. The current approach therefore, includes the prediction on the labeled data in the form of constraints, thereby utilizing all the available labeled data at each iteration of training a classifier for the ensemble. The problem can now be formally expressed as,

$$\begin{aligned} \min \quad & F(\mathbf{y}, S) \\ \text{s.t.} \quad & \hat{y}_i^l = y_i^l, i = 1, \cdots, n_l. \end{aligned} \quad (5)$$

This is a convex optimization problem, and therefore can be solved effectively by numerical methods. However, since our goal is to improve the given learning algorithm $\mathcal{A}$ by the unlabeled data and the similarity matrix S, we present a boosting algorithm that can efficiently minimize the objective function F. The following procedure is adopted to derive the boosting algorithm.

- The labels for the unlabeled samples y_i^u are replaced by the ensemble predictions over the corresponding data sample.
- A bound optimization based approach is then used to find the ensemble classifier minimizing the objective function.
- The bounds are simplified further to obtain the sampling scheme, and other required parameters.

The above objective function is strongly related to several graph based approaches, manifold regularization and ensemble methods. A discussion on the relationship between SemiBoost and several commonly used semi-supervised algorithms is presented in the Appendices F and G of the longer version of this paper [28].

C. Algorithm

We derive the boosting algorithm using the bound optimization approach. An alternate, conventional way to derive the boosting algorithm using the Function Gradient method is presented in [29]. This method may also be viewed as a relaxation that approximates the original objective function by a linear function. Such an approach however, involves specification of a parametric step size. In our derivation, the step size is automatically determined thus overcoming the difficulty in determining the step-size. SemiBoost algorithm is briefly summarized in Figure 3.

Let $h^{(t)}(\mathbf{x}) : \mathcal{X} \to \{-1, +1\}$ denote the 2-class classification model that is learned at the t-th iteration by the algorithm $\mathcal{A}$. Let $H(\mathbf{x}) : \mathcal{X} \to \mathbb{R}$ denote the combined classification model learned after the first T iterations. It is computed as a linear combination of the first T classification models, i.e.,

$$H(\mathbf{x}) \quad = \quad \sum_{t=1}^{T} \alpha_t h^{(t)}(\mathbf{x}),$$

where α_t is the combination weight. At the $(T+1)$-st iteration, our goal is to find a new classifier $h(\mathbf{x})$ and the combination weight α that can efficiently minimize the objective function F.

This leads to the following optimization problem:

$$\begin{aligned} \underset{h(\mathbf{x}),\alpha}{\arg\min} \quad & \sum_{i=1}^{n_l} \sum_{j=1}^{n_u} S_{i,j} \exp(-2y_i^l (H_j + \alpha h_j)) \\ + \quad & C \sum_{i,j=1}^{n_u} S_{i,j} \exp(H_i - H_j) \exp(\alpha(h_i - h_j)) \quad (6) \\ \text{s.t.} \quad & h(\mathbf{x}_i) = y_i^l, i = 1, \cdots, n_l, \quad (7) \end{aligned}$$

where $H_i \equiv H(\mathbf{x}_i)$ and $h_i \equiv h(\mathbf{x}_i)$.

This expression involves products of variables α and h_i, making it non-linear and hence difficult to optimize. The constraints, however, can be easily satisfied by including all the labeled samples in the training set of each component classifier. To simplify the computation, we construct the upper bound of the objective function, described in Proposition 1.

Proposition 1: Minimizing Eq (7) is equivalent to minimizing the function

$$\overline{F}_1 \quad = \quad \sum_{i=1}^{n_u} \exp(-2\alpha h_i) p_i + \exp(2\alpha h_i) q_i \quad (8)$$

where

$$p_i \quad = \quad \sum_{j=1}^{n_l} S_{i,j} e^{-2H_i} \delta(y_j, 1) + \frac{C}{2} \sum_{j=1}^{n_u} S_{i,j} e^{H_j - H_i} \quad (9)$$

$$q_i \quad = \quad \sum_{j=1}^{n_l} S_{i,j} e^{2H_i} \delta(y_j, -1) + \frac{C}{2} \sum_{j=1}^{n_u} S_{i,j} e^{H_i - H_j} \quad (10)$$

and $\delta(x, y) = 1$ when $x = y$ and 0 otherwise.

Proof Sketch: By substituting $H_i \leftarrow H_i + \alpha h_i$ into $F(\mathbf{y}, S)$ and regrouping the terms, we obtain the desired result. □

The quantities p_i and q_i can be interpreted as the confidence in classifying the unlabeled example $\mathbf{x}_i$ into the positive class and the negative class, respectively.

- Compute the pairwise similarity $S_{i,j}$ between any two examples.
- Initialize $H(\mathbf{x}) = 0$
- For $t = 1, 2, \ldots, T$
  - Compute p_i and q_i for every example using Equations (9) and (10)
  - Compute the class label $z_i = \text{sign}(p_i - q_i)$ for each example
  - Sample example $\mathbf{x}_i$ by the weight $|p_i - q_i|$
  - Apply the algorithm $\mathcal{A}$ to train a binary classifier $h_t(\mathbf{x})$ using the sampled examples and their class labels z_i
  - Compute α_t using Equation (11)
  - Update the classification function as $H(\mathbf{x}) \leftarrow H(\mathbf{x}) + \alpha_t h_t(\mathbf{x})$

Fig. 3. The SemiBoost algorithm

The expression in Eq (8) is difficult to optimize since the weight α and the classifier $h(x)$ are coupled together. We simplify the problem using the upper bound stated in the following proposition.

Proposition 2: Minimizing Eq (8) is equivalent to minimizing

$$\overline{F}_1 \quad \leq \quad \sum_{i=1}^{n_u}(p_i + q_i)(e^{2\alpha} + e^{-2\alpha} - 1) - \sum_{i=1}^{n_u} 2\alpha h_i(p_i - q_i).$$

Proof: See [28]. □

We denote the upper bound in the above equation by $\overline{F}_2$.

Proposition 3: To minimize $\overline{F}_2$, the optimal class label z_i for the example $\mathbf{x}_i$ is $z_i = \text{sign}(p_i - q_i)$, and the weight for sampling example $\mathbf{x}_i$ is $|p_i - q_i|$. The optimal α that minimizes $\overline{F}_1$ is

$$\alpha = \frac{1}{4} \ln \frac{\sum_{i=1}^{n_u} p_i \delta(h_i, 1) + \sum_{i=1}^{n_u} q_i \delta(h_i, -1)}{\sum_{i=1}^{n_u} p_i \delta(h_i, -1) + \sum_{i=1}^{n_u} q_i \delta(h_i, 1)}. \qquad (11)$$

Proof Sketch: Expression in Eq 11 can be obtained by differentiating $\bar{F}_2$ w.r.t α and setting it equal to 0. Observe that the above function is linear in $h_i(p_i - q_i)$ and is minimized when we choose $h_i = sign(p_i - q_i)$, for maximum values of $|p_i - q_i|$. □

Propositions 1-3 justify the relaxations made in the derivation of the SemiBoost. At each relaxation, the "touch-point" is maintained between the objective function and the upper bound. As a result, the procedure guarantees: (a) the objective function always decreases through iterations and (b) the final solution converges to a local minimum. For more details, see [30]. Proposition 3 establishes the key ingredients required for a boosting algorithm. Using these, the SemiBoost algorithm is presented in Figure 3.

Let ϵ_t be the weighted error made by the classifier, where

$$\epsilon_t = \frac{\sum_{i=1}^{n_u} p_i \delta(h_i, -1) + \sum_{i=1}^{n_u} q_i \delta(h_i, 1)}{\sum_i (p_i + q_i)}.$$

As in the case of AdaBoost [29], α can be expressed as

$$\alpha_t = \frac{1}{4} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right) \qquad (12)$$

which is very similar to the weighting factor of AdaBoost, differing only by a constant factor of $\frac{1}{2}$. Also, if AdaBoost encounters a situation where the base classifier has an error rate more than random, i.e. $\epsilon_{t+1} \geq \frac{1}{2}$, it returns the current classifier H_t. This situation has a direct correspondence with the condition in SemiBoost where the algorithm stops when $\alpha \leq 0$. From Eq (11) (or rather directly from Eq (12)), we can see that this happens only when the denominator exceeds the numerator, which means $\epsilon_{t+1} \geq \frac{1}{2}$ is equivalent to the condition $\alpha \leq 0$. However, since this condition may not be satisfied until a large number of classifiers are trained, usually there is a parameter specifying the number of classifiers to be used. It has been empirically determined that using a fixed number (usually 20) of classifiers for AdaBoost gives good performance [27].

The sampling scheme used in SemiBoost is significantly different from that of AdaBoost. AdaBoost knows the true labels of the data, and hence can proceed to increase/decrease the weights assigned to samples based on the previous iteration. In SemiBoost we do not have the true class labels for the unlabeled data, which makes it challenging to estimate the difficulty of classification. However, Proposition 2 gives us the result that selecting the most confident unlabeled data samples is optimal for reducing the objective function. Intuitively, using the samples with highly confident labeling is a good choice because they are consistent with the pairwise similarity information along with their classifications. The values of p_i and q_i tend to be large if (i) $\mathbf{x}_i$ can't be classified confidently, i.e., $|H_i|$ is small, and one of its close neighbors is labeled. This corresponds to the first term in Eq. (9) and Eq. (10). (ii) the example $\mathbf{x}_i$ is highly similar to some unlabeled examples that are already confidently classified, i.e., large $s_{i,j}$ and $|H_j|$ for unlabeled example x_j. This corresponds to the second term in Eq. (9) and Eq. (10). This indicates that the similarity information plays an important role in guiding the sample selection, in contrast with the previous approaches like ASSEMBLE and SSMB, where the samples are selected to increase value of $|H_i|$ alone.

Similar to most boosting algorithms, we can show that the proposed semi-supervised boosting algorithm reduces the original objective function F exponentially. This result is summarized in the following Theorem.

Theorem 1: Let $\alpha_1, ..., \alpha_t$ be the combination weights that are computed by running the SemiBoost algorithm (Fig 1). Then, the objective function at $(t+1)$st iteration, i.e., F_{t+1}, is bounded as follows:

$$F_{t+1} \leq \kappa_S \exp\left(-\sum_{i=1}^{t} \gamma_i \right),$$

where $\kappa_S = \left[\sum_{i=1}^{n_u} \left(\sum_{j=1}^{n_l} S_{i,j} + C \sum_{j=1}^{n_u} S_{i,j} \right) \right]$ and $\gamma_i = \log(\cosh(\alpha_i))$.

Proof: See [28]. □

The above theorem shows that the objective function follows an exponential decay, despite the relaxations made in the above propositions.

Corollary 1: The objective function at $(t+1)$st iteration is bounded in terms of the error ϵ_t as $F_{t+1} \leq$

$\kappa_S \prod_{i=1}^{t} \left(\frac{1-\epsilon_t}{\epsilon_t}\right)^{1/4}$.

Proof Sketch: Can be verified by substituting Eq. (12) for α_i in Theorem 1. □

In the above derivation, we constrained the objective function such that the prediction of the classifier on the labeled samples must match the true labels provided. However, if the true labels are noisy, the resulting semi-supervised classifier might not perform its best. An algorithm similar to SemiBoost may be derived in such a case by including a term penalizing the solution, if the predicted labels of the labeled samples are different from the true labels. In this paper, we assume that the given labels are correct. This is reasonable given the fact that there are very few labeled samples, one can ensure their correctness without much difficulty.

D. Implementation

1) Sampling: Sampling forms the most important step for SemiBoost, just like any other boosting algorithm. The criterion for sampling usually considers the following issues: (a) How many samples must be selected from the unlabeled samples available for training? and (b) What is the distribution according to which the sampling must be done?

Supervised boosting algorithms like AdaBoost have the true labels available, which makes it easy to determine which samples to choose or not to choose. On the other hand, the labels assigned during the SemiBoost iteration are pseudo labels, and may be prone to errors. This suggests that we should choose only the top few most confident data points for SemiBoost. But selecting a small number of samples might make the convergence slow, and selecting too large a sample might include non-informative or even poor samples into the training set. The choice currently is made empirically; selecting top 10% of the samples seem to work well in practice. From Proposition 3, to reduce $\bar{F}_1$, it is preferable to select the samples with a large value of $|p_i - q_i|$. This selection provides highly reliable pseudo-labeled samples to the classifier. The sampling is probabilistically done according to the distribution,

$$P_s(\mathbf{x}_i) = \frac{|p_i - q_i|}{\sum_{i=1}^{n_l} |p_i - q_i|},$$

where $P_s(\mathbf{x}_i)$ is the probability that the data point $\mathbf{x}_i$ is sampled from the transduction set.

2) Stopping Criterion: According to the optimization procedure, SemiBoost stops when $\alpha \leq 0$, indicating that addition of that classifier would increase the objective function instead of decreasing it. However, the value of α decreases very fast in the beginning, and eventually the rate of decrease falls down, taking a large number of iterations to actually make it negative. We currently use an empirically chosen fixed number of classifiers in the ensemble, specified as a parameter T. We set the value of $T = 20$.

3) Similarity Matrix: We use the Radial Basis Function similarity inspired from its success in graph based approaches. For any two samples $\mathbf{x}_i$ and $\mathbf{x}_j$, the similarity $S_{i,j}$ is computed as, $S_{i,j} = \exp(\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/\sigma^2)$, where σ is the scale parameter controlling the spread of the radial basis function. It is well known that the choice of σ has a large impact on the performance of the algorithm [18]. We set the scale parameter to the similarity values at the 10-th percentile to the 100-th percentile, varied in steps of 10, where μ_s is the average value of the similarity matrix S. Experimentation revealed that the transductive and inductive performances are stable for the chosen range of σ. This is a desirable property given the fact that choosing the right scale parameter is a difficult problem.

IV. Results and discussion

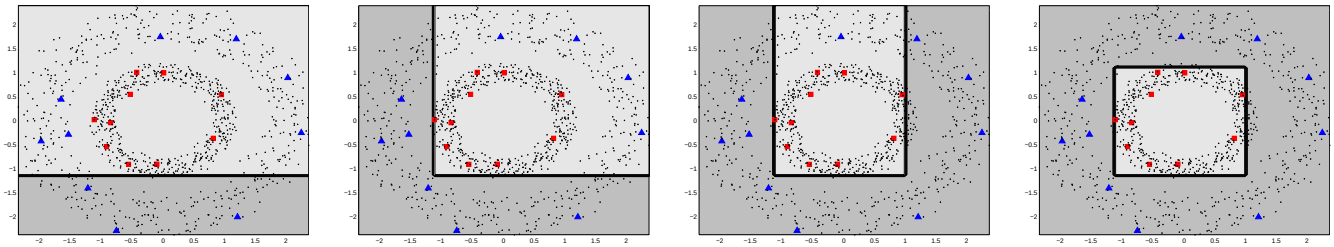
The focus of SemiBoost is to improve any given (supervised) classifier using the unlabeled data. Therefore, our primary aim is to evaluate SemiBoost based on the improvement achieved in the inductive performance of base classifiers.

An illustration of improvement in the performance of a supervised learner (Decision Stump) using SemiBoost on the "ring-norm" dataset is shown in Figure 4. The dataset has 2 classes, with 500 samples each. There are 10 labeled samples per class, indicated using (■, ▲). The solid line shows the decision boundary and the dark and light regions indicate the two class regions. The performance of SemiBoost at each iteration is given in parentheses below each of the plots shown. Figures 4(a)-(c) show the classifier obtained by SemiBoost at the first three iterations, and Figure 4(d) shows the final classifier obtained at the 12 iteration.

A. Datasets

SemiBoost was evaluated on 16 different datasets: 4 benchmark datasets provided in [9], 10 UCI datasets and 2 datasets from ethnicity classification from face images [31] and texture classification [32]. Since SemiBoost is applicable for two-class problems, we chose the two-class datasets from these benchmark datasets. The multiclass datasets in UCI are converted into two-class datasets by choosing the two most populated classes. The name of the dataset used, the classes chosen, the number of samples present in the selected classes n, and the dimensionality of the dataset d are summarized in the first column in Table II. In addition to this, we evaluated the proposed approach on text categorization problems, whose results are presented in the next section.

The transductive performance of semi-supervised learning algorithms is well-studied [9, Chapter 21]. However, semi-supervised learning is not limited to transductive learning, and out-of-sample extensions have attracted significant attention. In fact, inductive learning is important given that only a portion of the unlabeled samples are seen during the training phase. The real utility of learning in such cases lies in the ability to classify unseen test samples. With this motivation, we compare the performance of SemiBoost with three state-of-the-art inductive semi-supervised algorithms: Transductive SVM [7], an inductive version of Low Density Separation (LDS) [8] and Laplacian-SVM from the Manifold Regularization approach [12]. LDS is not an inductive algorithm as it involves a graph-based dimensionality reduction step. We use the labels predicted by the LDS on the transduction set to train an inductive classifier on the original data.



(a) Iter. 1, (65.0%) (b) Iter. 2, (76.5%) (c) Iter. 3, (86.4%) (d) Iter. 12, (94.6%)

Fig. 4. Decision boundary obtained by SemiBoost at iterations 1, 2, 3 and 12, on the two concentric rings dataset, using Decision Stump as the base classifier. There are 10 labeled samples per class (■,▲). The transductive performance (i.e., performance on the unlabeled data used for training) of SemiBoost is given at each iteration in parentheses.

B. Experimental setup

The experimental setup aims to study the improvement in performance obtained on a supervised learner, by using unlabeled data and the performance of the SemiBoost algorithm with three state of the art semi-supervised learning algorithms.

We use classification accuracy as the evaluation measure. The mean and standard deviation of % accuracy are reported over 20 runs of each experiment, with different subsets of training and testing data. To measure the inductive performance, we randomly split the dataset into two halves. We call them the training and test sets. The training set has 10 labeled points and the rest unlabeled. The ensemble classifier learnt by SemiBoost on the training set is evaluated by its performance on predicting the labels of the test set.

SemiBoost samples the unlabeled data, labels them at each iteration of boosting and builds a classifier $h_t(\mathbf{x})$. The number of such classifiers built will depend on the number of iterations T in boosting. T was set to 10 and we stop the boosting when weights α_t computed from Eq (11) become negative. We set the value of C in the objective function Eq (4) to be the ratio of number of labeled samples to the number of unlabeled samples $C = n_l/n_u$.

The first experiment studies the improvement in the performance of three different base classifiers ($h_t(\mathbf{x})$) after applying SemiBoost: Decision Stump (DS), the J48 decision tree algorithm (J48), and the Support Vector Machine with the sequential minimal optimization (SVM) algorithm. Software WEKA [33] was used to implement all the three classifiers. All the algorithms are run with their default parameters (e.g. default C and a linear kernel was used for SVM algorithm). We chose decision trees (DS and J48) and SVM as the base classifiers because they are shown to be successful in the supervised learning literature, for varied learning tasks.

C. Results

1) Choice of base classifier: Table II compares the supervised and the three benchmark semi-supervised algorithms to the SemiBoost algorithm. The columns DS, J48 and SVM give the performances of the base classifiers on the induction set. The columns SB-X give the inductive performances of SemiBoost with base classifier X. The last three columns in Table II correspond to the inductive performances of benchmark semi-supervised algorithms TSVM, LDS and LapSVM. Note that the idea is not to build the best classifier for individual classification problem, but to show the possible improvement in the performance of supervised classifiers using SemiBoost on all the classification problems. Results indicate that SemiBoost significantly improves the performance of all the three base classifiers for nearly all the datasets. Using an independent sample paired t-test, we observed that SemiBoost significantly improved the performance of Decision Stump on 12 out of 16 datasets. The performance of J48 is improved significantly on 13 out of 16 datasets, and there is a significant degradation on the house dataset. For SVM, there is a significant improvement in 7 out of 16 datasets, while a significant degradation in 3 cases. The cases where SVM degraded, the benchmark algorithms performed poor compared to the supervised classifiers, suggesting that unlabeled data is not helpful in these cases. The ensemble classifier obtained using SemiBoost is relatively more stable, as its classification accuracy has lower standard deviation when compared to the base classifier.

2) Performance comparison of SemiBoost with Benchmark Algorithms: Performance of SemiBoost is compared with three different algorithms, namely TSVM, LapSVM and ILDS (inductive version of the LDS algorithm). SemiBoost achieves performance comparable to that of the benchmark algorithms. SemiBoost performs better than ILDS on almost all the datasets, and significantly better on 4 of the datasets, 2 using Decision Stump and 2 using SVM as the base classifier. SemiBoost significantly outperforms TSVM on 10 out of 16 datasets using SVM, and 8 out of 16 datasets using Decision Stump. Also, TSVM had difficulty converging on three datasets in a reasonable amount of time (20 hours). SemiBoost performs comparably to LapSVM; SB-DS outperformed LapSVM significantly on 2 datasets, and performed worse than LapSVM on 1. Similarly, SB-SVM and LapSVM significantly outperform each other on 3 out of the 8 cases. There are datasets where one of the base classifiers outperforms SemiBoost. But in these cases, one of the base classifiers outperforms all the semi-supervised algorithms (e.g., SVM outperforms all the algorithms on COIL2, vehicle, sat and house datasets). This indicates that the unlabeled data do not always improve the base classifier, or in general, are not guaranteed to help in the learning process. When a base classifier outperforms the semi-supervised learning algorithms, we observed that the SemiBoost tends to perform close to the baseline compared to the others in most cases.

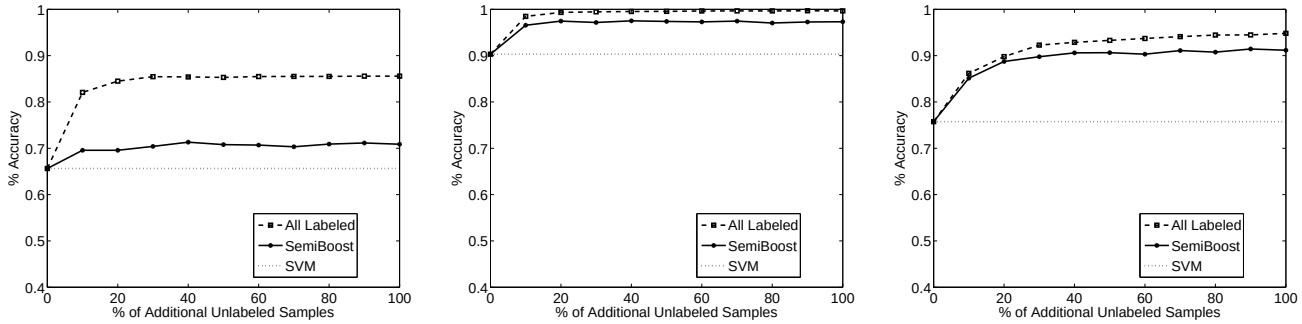
TABLE II

INDUCTIVE PERFORMANCE OF SEMIBOOST AND THE THREE BENCHMARK ALGORITHMS. THE FIRST COLUMN SHOWS THE DATASET AND THE TWO CLASSES CHOSEN. THE NUMBER OF SAMPLES n AND THE DIMENSIONALITY d ARE SHOWN BELOW THE NAME OF EACH DATASET. THE ALGORITHMS CHOSEN AS BASE CLASSIFIERS FOR BOOSTING ARE DECISION STUMP (DS), DECISION TREE (J48) AND SUPPORT VECTOR MACHINE (SVM). FOR EACH ALGORITHM, THE SB- PREFIXED COLUMN INDICATES USING THE SEMIBOOST ALGORITHM ON THE BASE CLASSIFIER. THE COLUMNS TSVM, ILDS AND LAPSVM SHOW THE INDUCTIVE PERFORMANCE OF THE THREE BENCHMARK ALGORITHMS. A '-' INDICATES THAT WE COULD NOT FINISH RUNNING THE ALGORITHM IN A REASONABLE TIME (20 HOURS) DUE TO CONVERGENCE ISSUES. EACH ENTRY SHOWS THE MEAN CLASSIFICATION ACCURACY AND STANDARD DEVIATION (IN PARENTHESES) OVER 20 TRIALS.

Dataset (classes) (n, d)	DS	SB-DS	J48	SB-J48	SVM	SB-SVM	TSVM	ILDS	LapSVM
Digit1 (1,2)	57.15	78.09	57.21	74.97	74.81	77.89	79.52	79.53	74.06
(1500, 241)	(7.0)	(3.6)	(7.1)	(4.3)	(6.2)	(4.6)	(5.0)	(7.0)	(4.1)
COIL2 (1,2)	55.14	55.84	54.81	55.27	59.75	55.42	50.23	54.62	55.64
(1500, 241)	(3.1)	(4.0)	(3.4)	(2.9)	(3.3)	(4.3)	(4.9)	(4.0)	(5.6)
BCI(1,2)	51.27	49.38	51.42	50.67	52.45	52.02	50.50	50.73	54.37
(400, 117)	(4.2)	(2.9)	(4.1)	(3.8)	(3.1)	(4.1)	(3.6)	(2.4)	(3.6)
g241n (1,2)	50.73	54.54	50.57	54.71	57.55	57.93	51.14	50.25	53.65
(1500, 241)	(3.1)	(2.8)	(2.9)	(2.5)	(2.6)	(3.4)	(3.5)	(1.5)	(3.1)
austra (1,2)	60.39	73.46	60.12	73.36	65.64	71.36	73.38	66.00	74.38
(690, 15)	(13.0)	(7.9)	(12.7)	(7.4)	(8.2)	(8.8)	(12.6)	(14.5)	(8.7)
ethn (1,2)	65.72	66.42	64.98	63.98	67.04	67.57	-	67.16	74.60
(2630, 30)	(8.6)	(6.4)	(7.9)	(5.3)	(4.8)	(5.7)		(16.7)	(5.8)
heart (1,2)	68.26	79.48	67.67	78.78	70.59	79.00	77.63	77.11	77.96
(270, 9)	(14.3)	(3.6)	(15.0)	(3.8)	(7.9)	(4.1)	(6.6)	(9.6)	(4.8)
wdbc (1,2)	79.47	88.98	75.95	89.82	75.74	88.82	86.40	85.07	91.07
(569, 14)	(16.3)	(6.5)	(17.1)	(4.0)	(9.7)	(9.9)	(8.6)	(8.7)	(3.4)
vehicle (2,3)	60.48	69.31	60.89	70.25	78.28	72.29	63.62	66.28	71.38
(435, 26)	(7.6)	(6.7)	(8.1)	(7.7)	(6.2)	(9.4)	(8.6)	(8.5)	(6.7)
texture (2,3)	95.67	98.90	89.46	98.50	98.44	99.91	-	98.38	99.11
(2026, 19)	(5.6)	(0.6)	(6.7)	(0.9)	(1.4)	(0.1)		(7.2)	(0.92)
uci image (1,2)	89.64	100.00	87.03	99.79	99.92	100.00	91.91	100	99.95
(660, 18)	(11.2)	(0.0)	(9.3)	(0.3)	(0.2)	(0.0)	(8.2)	(0.0)	(0.2)
isolet (1,2)	64.23	91.92	64.48	90.20	89.58	95.12	90.38	92.07	93.93
(600, 51)	(12.7)	(2.1)	(12.8)	(3.4)	(5.3)	(2.3)	(8.0)	(11.4)	(3.4)
mfeat fou (1,2)	82.25	96.25	85.90	96.00	98.78	99.85	95.32	96.5	100.00
(400, 76)	(2.6)	(2.0)	(12.9)	(1.8)	(1.1)	(0.3)	(7.5)	10.8	(0.0)
optdigits (2,4)	65.91	93.22	65.59	93.33	90.31	96.35	92.34	96.40	98.34
(1143, 42)	(13.4)	(3.0)	(13.1)	(2.6)	(3.6)	(2.4)	(9.0)	(11.1)	(2.4)
sat (1,6)	82.77	85.99	83.80	86.55	99.13	87.71	-	94.20	99.12
(3041, 36)	(5.5)	(3.7)	(6.1)	(3.0)	(0.7)	(2.9)		(14.2)	(0.5)
house (1,2)	94.83	91.92	94.83	91.34	91.16	90.65	86.55	89.35	89.95
(232, 16)	(6.0)	(3.9)	(6.0)	(3.3)	(3.8)	(2.8)	(4.4)	(2.2)	(4.4)

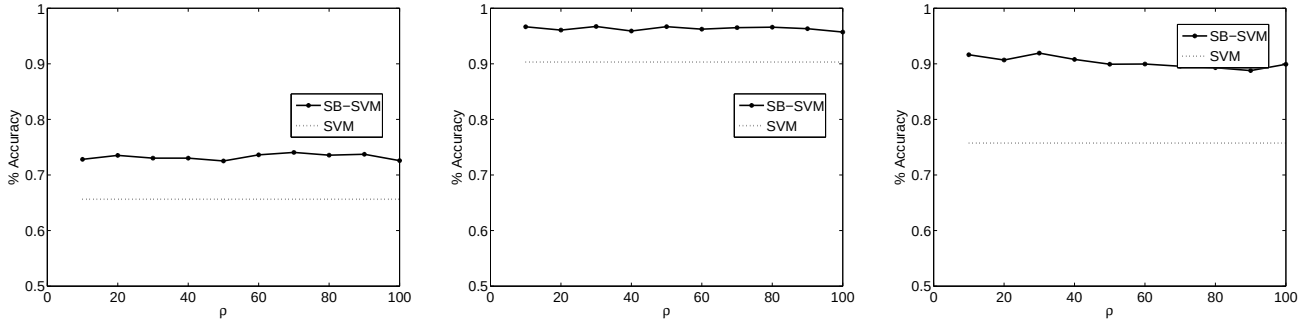
3) Performance with unlabeled data increment: Fig. 5 shows the performance of SemiBoost on three of the UCI datasets (Figs. 5 (a)-(c)) we used. Each dataset is split into two equal parts, one for training and one for inductive testing. Ten samples in the training set are labeled. The performance of SVM, trained on the labeled data and with default parameters, on the test set is shown with a dotted line in each plot. The unlabeled examples in the training set are incrementally added to the labeled examples in units of 10%. The solid line shows the performance of the SemiBoost algorithm with addition of unlabeled data. The dashed line shows the performance obtained by the SVM when all these added samples are labeled using their ground truth. It is observed that the performance of SemiBoost improves with the addition of more and more unlabeled data, whenever such an improvement is possible.

4) Sensitivity to parameter σ: Fig. 6 shows the performance of the SemiBoost-SVM, with varying value of the parameter σ. Sigma was chosen to be the ρ-th percentile of the distribution of similarities, with ρ varying between 10-th percentile to 100-th percentile. Selecting the value of σ is one of the most difficult aspects of graph construction, and several heuristics have been proposed to determine its value. On most of the datasets shown, SemiBoost is relatively stable with respect to the scale parameter. However, a choice of σ between 10-th percentile to 20-th percentile of the pairwise distances is recommended, based on empirical observations.



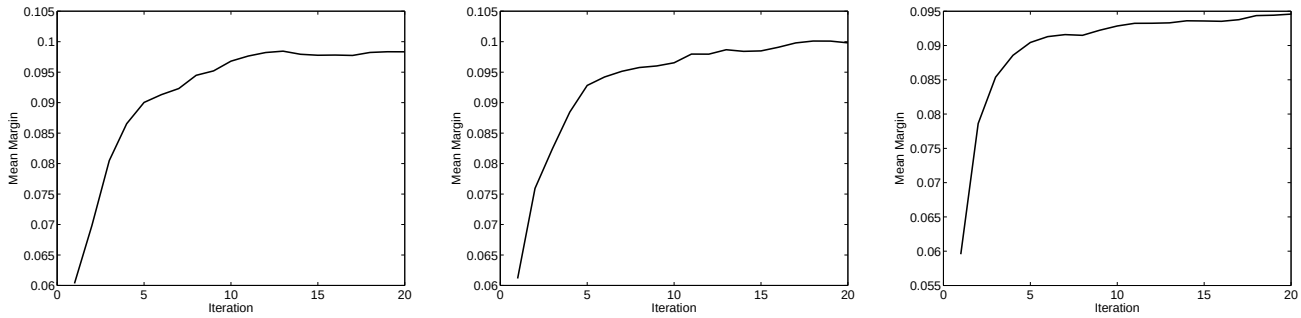
(a) austra (UCI) (b) optdigits (UCI) (c) wdbc (UCI)

Fig. 5. Performance of baseline algorithm SVM with 10 labeled samples, with increasing number of unlabeled samples added to the labeled set (solid line), and with increasing number of labeled samples added to the training set (dashed line).



(a) austra (UCI) (b) optdigits (UCI) (c) wdbc (UCI)

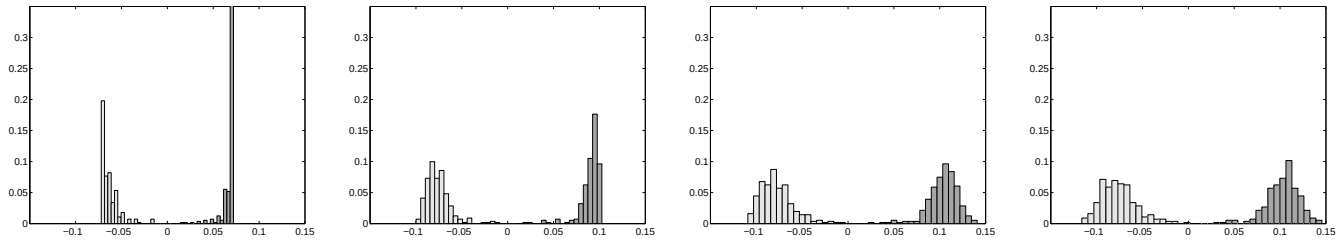
Fig. 6. Performance of baseline algorithm SVM with 10 labeled samples, with increasing value of the parameter σ. The increments in σ are made by choosing the ρ-th percentile of the similarities, where ρ is represented on the horizontal axis.



(a) Decision Stump (b) J48 (c) SVM

Fig. 7. The mean-margins over the iterations, on a single run of SemiBoost on optdigits dataset (classes 2,4), using different base classifiers.

5) Margin and Confidence: In this experiment we empirically demonstrate that SemiBoost has a tendency to maximize the mean-margin. For unlabeled data, a popular definition of margin is $|H(\mathbf{x}_i)|$ [16], [17]. The mean margin is the empirical average of $|H(\mathbf{x}_i)|$ over the unlabeled data used for training. Figs. 7((a)-(c)) show the mean-margin value on optdigits dataset (classes 2,4) over the iterations using Decision Stump, J48 and SVM as the base classifiers, respectively. The value of the mean-margin increases over the iterations, irrespective of the choice of the base classifier. However, it is important to note that the minimum margin may not increase at each iteration, although the test error decreases. When the training data consists of a small number of labeled samples which can be perfectly classified, the margin is largely decided by the unlabeled data. Considering the margin over the unlabeled data, the classifier at iteration 1 has a margin of α_1 for all the unlabeled data, wheres at the second iteration, the minimum margin is $\min_{\mathbf{x}_i} |H^{(2)}(\mathbf{x}_i)| = |\alpha_1 - \alpha_2| \leq \alpha_1 = \min_{\mathbf{x}_i} |H^{(1)}(\mathbf{x}_i)|$. In fact, over the iterations, the value of minimum margin may be traded off to obtain a gain in the performance, i.e. being in agreement with the similarity matrix. Recently, it was shown in [34] that maximizing the value of minimum margin does not necessarily translate to a better performance by the classifier. It is argued in the context of boosting that the approaches maximizing the mean-margin greedily are preferable to those that maximize the



(a) Iteration 1 (b) Iteration 2 (c) Iteration 10 (d) Iteration 20

Fig. 8. Distribution of the ensemble predictions $H_t(\mathbf{x}_i)$, over the unlabeled samples in the training data from optdigits dataset (classes 2,4) at the iteration t, where $t \in \{1, 2, 10, 20\}$. SVM is used as the base classifier. The light and dark bars in the histogram correspond to the two classes 2 and 4 respectively.

minimum margin. Fig. 8 shows the distribution of the value of $H(\mathbf{x}_i)$ over the iterations. The light and dark bars in the histogram represent the classes 2 and 4, in the optdigits dataset respectively. Note that as iterations progress, the classes get more and more separated.

D. Convergence

According to Theorem 1, SemiBoost converges exponentially. The following section demonstrates the convergence of SemiBoost on an example dataset. To illustrate the convergence, we chose the two most populous classes in the optdigits dataset, namely digits 2 and 4. The change in the objective function as new classifiers are added over iterations is demonstrated in Fig. 9(a). which follows an exponential reduction. Fig. 9(b) shows the value of α over the iterations. Initially, the value of α falls rapidly, and after around 20 iterations, the value is insignificantly small relative to that of initial classifiers. This suggests that although SemiBoost still needs more iterations to converge, the new classifiers added in boosting will not significantly change the decision value. Fig. 9(c) shows the accuracy of the SemiBoost with Decision Stump as the base classifier, over the iterations.

E. Comparison with AdaBoost

To evaluate the contribution of unlabeled data in improving the performance of a base classifier, we compared the performance of SemiBoost with that of AdaBoost on the same base classifier (or weak learner) and using a similar experimental procedure as in Section IV-B. Table III shows the performance of three base classifiers Decision Stump, J48 and SVM (shown in column 1) on 6 datasets shown in the top row. For each classifier, the first two rows show the inductive performance of the classifier and its boosted version (using AdaBoost) trained on 10 labeled samples. The third row shows the performance of SemiBoost when unlabeled data is added to the same set of labeled samples. The fourth and fifth rows, labeled *large* and *AB-large* show the performance of the classifier and its boosted version trained after labeling the unlabeled data used in SemiBoost.

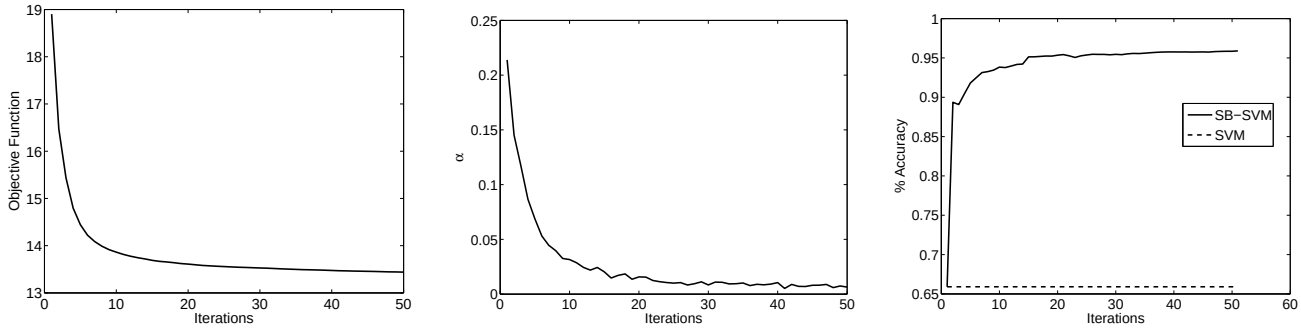
From Table III, we can see that the performance of SemiBoosted versions of the classifiers (SB-DS, SB-J48, and SB-SVM) is significantly better than classifiers trained using only labeled data, boosted (using AdaBoost) or unboosted (rows 1 and 2 for each classifier section). Naturally, when all the unlabeled data are labeled, the performance of the classifiers and their boosted versions are significantly better than SemiBoost (rows 4 and 5). The reduction in the inductive performance of AB-small compared to the base classifier on several datasets may be attributed to the overfitting due to small number of training samples. The addition of unlabeled data as a regularizing mechanism in SemiBoost avoids the overfitting, thereby achieving an improved classifier.

V. Performance on Text-Categorization

We further evaluate the SemiBoost algorithm on the Text Categorization problem using the popular 20-newsgroups dataset[2]. We performed the evaluation of SemiBoost algorithm with Decision Stump, J48 and SVM as the base classifiers on binary problems created using 10 most popular classes of the 20-newsgroups dataset. Note that this experimental setup is different from some other studies of semi-supervised learning in which the one-vs-rest approach is used for evaluation. Compared to one-vs-rest, the one-vs-one evaluation has the following advantages:

- There is a large variation in the best performing supervised classifier for the binary tasks. This enables us to show that when SVM is not the best classifier for a problem, then the methods that improve SVM using unlabeled data may not be the best semi-supervised algorithms to use.
- Semi-supervised learning algorithms rely on certain assumptions about the structure of the data and the classes. In one-vs-rest approaches, these assumptions are likely to be violated. For instance, many semi-supervised algorithms assume a large cluster gap between two classes. By aggregating multiple classes into one negative class, we expect to see a large cluster gap amongst the negative class itself. Violation of the manifold assumption can be explained similarily.
- There is a high imbalance in the data when we do a one-vs-rest classification. While a knowledge of priors may be used to incorporate this imbalance into semi-supervised learning to achieve high performance, we assume that nothing is known about the data other than the similarity information and a few training examples.

[2]http://people.csail.mit.edu/jrennie/20Newsgroups/



(a) (b) (c)

Fig. 9. Plots of (a) objective function, (b) α value and (c) Accuracy of SemiBoost when run over two classes (2,4) of the optdigits dataset using Decision Stump as the base classifier. The accuracy of the base classifier is 65.9%.

TABLE III

PERFORMANCE OF DIFFERENT CLASSIFIERS AND THEIR BOOSTED VERSIONS ON 6 UCI DATASETS. X-SMALL STANDS FOR THE CLASSIFIER TRAINED ON A SET OF 10 LABELED SAMPLES CHOSEN FROM THE DATA. THE PREFIX AB-X STANDS FOR ADABOOST WITH BASE CLASSIFIER X. SB-X STANDS FOR SEMIBOOST WITH BASE CLASSIFIER X. X-LARGE STANDS FOR THE CLASSIFIER TRAINED BY LABELING ALL THE UNLABELED DATA USED IN SB-X.

Classifier		austra	bupa	wdbc	optdigits	mfeat-fou	isolet
Decision Stump	small	60.39	54.94	79.47	65.91	82.25	64.23
	AB-small	62.55	56.45	70.02	63.20	77.62	64.82
	SemiBoost	73.46	55.78	88.98	93.22	96.25	91.92
	large	79.36	57.71	90.42	90.26	99.72	92.57
	AB-large	81.70	68.44	94.44	99.98	99.72	97.68
J48	small	60.12	54.97	75.95	65.59	85.90	64.48
	AB-small	60.68	55.09	68.86	61.40	75.80	65.33
	SemiBoost	73.36	54.74	89.82	93.33	96.00	90.20
	large	79.97	62.49	92.68	97.18	99.12	92.90
	AB-large	82.42	66.21	94.96	98.97	99.12	96.68
SVM	small	65.64	52.05	75.74	90.31	98.78	89.58
	AB-small	63.29	53.50	73.53	87.11	93.80	88.48
	SemiBoost	71.36	54.02	88.82	96.35	99.85	95.12
	large	85.57	58.15	94.81	99.66	100.00	99.72
	AB-large	84.29	65.64	95.89	99.65	100.00	99.72

- One-vs-one had been a popular approach for creating multiclass classifiers. The testing time can be significantly reduced in a one-vs-one setting by using a DAG based architecture [35].

We generate all the 45 possible binary problems of the 10 classes. Due to the limited space, we include only results on 10 binary problems created from 5 of the classes in the 20 newsgroups, summarized in Table IV, These results are similar to the results on the 45 binary problems.The first column in Table IV shows the classes chosen for creating the binary problems. Each classification task contains a dataset of approximately 2000 documents. We use the popular tf-idf features computed over the words which occur at least 10 times in total, in all the 2000 documents. The tf-idf features are later normalized per document. The dimensionality of the each dataset is shown in column 2 of Table IV. We follow the same inductive evaluation procedure as in Section IV-B. We use 2 labeled samples per class in training the classifier. We use the linear kernel (dot product between the feature vectors) as similarity, popular for the text classification tasks. The inductive performance of the different algorithms Decision Stump, J48, SVM and their SemiBoosted versions, Transductive SVM, Inductive LDS, Laplacian SVM are shown in Table IV. To allow a fair comparison, the parameter value C for all SVMs is set to 1. The mean and standard deviation of the performance over 20 runs of the experiment is reported.

Table IV shows that in the case of Decision Stump and J48, SemiBoost significantly (at a 95% confidence level, measured using independent sample paired t-test) improves the performance on all the pairs of classes. The performance of SVM is improved significantly on 5 out of 10 class pairs. Also, we notice that SemiBoosted Decision stump performs signficantly better than SemiBoosted SVM on all the pairs of classes. Comparing the SVM based methods, SB-SVM significantly outperforms LapSVM on 7 class pairs and ILDS on all the 10 pairs. TSVM outperforms SB-SVM on 5 out of 10 class pairs. Overall, SB-SVM performs comparable to TSVM and significantly better than LapSVM and ILDS. SB-DS outperforms TSVM significantly on 5 out of 10 class pairs, and LapSVM and ILDS on all the class pairs. The

TABLE IV
COMPARISON OF THE INDUCTIVE PERFORMANCE (MEASURED AS % ACCURACY) OF SEMIBOOST, TSVM, ILDS AND LAPSVM ON PAIRWISE BINARY PROBLEMS CREATED FROM 5 CLASSES OF THE 20-NEWSGROUPS DATASET.

Classes (d)	DS	SB DS	J48	SB J48	SVM	SB SVM	TSVM	ILDS	LapSVM
1, 2	55.82	82.30	56.93	72.86	71.02	70.74	75.44	55.10	68.23
(3736)	(14.0)	(12.3)	(15.5)	(8.0)	(8.7)	(5.6)	(13.2)	(16.6)	(3.9)
1, 3	54.61	85.95	56.24	77.05	72.17	74.83	89.34	58.88	71.34
(3757)	(10.6)	(9.6)	(10.7)	(8.0)	(8.2)	(5.4)	(5.9)	(20.2)	(4.8)
1, 4	51.35	87.36	54.71	80.65	77.22	78.47	88.71	61.72	74.67
(3736)	(7.1)	(11.4)	(13.4)	(5.7)	(9.0)	(3.6)	(6.8)	(9.4)	(3.1)
1, 5	55.72	91.37	57.55	86.65	74.84	82.64	92.35	66.45	78.01
(3979)	(11.4)	(7.8)	(12.6)	(6.7)	(9.8)	(4.3)	(5.5)	(16.7)	(4.1)
2, 3	48.94	73.33	49.43	64.52	63.12	64.06	66.05	50.76	61.68
(4154)	(2.3)	(11.6)	(2.1)	(7.8)	(5.2)	(4.5)	(10.6)	(1.8)	(3.8)
2, 4	49.60	88.43	49.40	78.07	69.47	74.85	81.50	50.32	70.95
(4143)	(4.0)	(9.5)	(3.8)	(5.0)	(7.0)	(3.2)	(13.5)	(2.1)	(3.2)
2, 5	49.38	94.65	49.24	83.87	71.62	80.12	84.94	53.94	74.79
(4406)	(1.9)	(6.5)	(1.6)	(6.0)	(6.6)	(4.8)	(12.4)	(7.3)	(3.4)
3, 4	51.16	90.22	51.46	77.34	72.22	75.26	81.98	50.08	71.45
(4130)	(3.1)	(8.6)	(3.7)	(7.2)	(5.4)	(5.1)	(12.7)	(2.7)	(3.8)
3, 5	51.67	92.93	51.71	81.16	73.65	78.31	77.38	53.83	74.91
(4426)	(4.0)	(6.5)	(4.9)	(7.0)	(8.3)	(3.9)	(16.2)	(8.1)	(4.2)
4, 5	51.68	79.51	51.53	68.27	62.08	68.07	67.54	52.39	65.05
(4212)	(4.1)	(11.5)	(3.9)	(8.4)	(5.8)	(5.3)	(12.7)	(6.5)	(5.0)

poor performance of ILDS may be ascribed to the kernel used by ILDS. ILDS uses a graph distance based kernel, which may not be as suitable for text classification based tasks as a linear kernel. From our experiments, we see that SemiBoosting Decision Stumps is a viable alternative to the SVM based semi-supervised learning approaches.

VI. CONCLUSIONS AND FUTURE WORK

We have proposed an algorithm for semi-supervised learning using a boosting framework. The strength of SemiBoost lies in its ability to improve the performance of any given base classifier in the presence of unlabeled samples. Overall, the results on both UCI datasets and the text categorization using 20-newsgroups dataset demonstrate the feasibility of this approach. The performance of SemiBoost is comparable to the state-of-the-art semi-supervised learning algorithms. The observed stability of SemiBoost suggests that it can be quite useful in practice. SemiBoost, like almost all other semi-supervised classification algorithms, is currently a two-class algorithm. We are exploring the multiclass extension by redefining the consistency measures to handle multiple classes. We are working towards obtaining theoretical results that will guarantee the performance of SemiBoost, when the similarity matrix reveals the underlying structure of data (e.g., the probability that two points may share the same class).

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their valuable comments. The research was partially supported by ONR grant no. N000140710225 and NSF grant no. IIS-0643494.

REFERENCES

[1] H. J. Scudder, "Probability of error of some adaptive pattern-recognition machines," *IEEE Trans. on Inf. Theory*, vol. 11, pp. 363–371, 1965.

[2] H. Robbins and S. Monro, "A stochastic approximation method," *Annals of Math. Stat.*, vol. 22, pp. 400–407, 1951.

[3] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002.

[4] Y. Bengio, O. B. Alleau, and N. Le Roux, "Label propagation and quadratic criterion," in *Semi-Supervised Learning* (O. Chapelle, B. Schölkopf, and A. Zien, eds.), pp. 193–216, MIT Press, 2006.

[5] M. Szummer and T. Jaakkola, "Partially labeled classification with Markov random walks," in *NIPS 14*, pp. 945–952, 2001.

[6] A. Blum and S. Chawla, "Learning from labeled and unlabeled data using graph mincuts," in *Proc. 18th International Conference on Machine Learning*, pp. 19–26, 2001.

[7] T. Joachims, "Transductive learning via spectral graph partitioning," in *Proc. 20th International Conference on Machine Learning*, pp. 290–297, 2003.

[8] O. Chapelle and A. Zien, "Semi-supervised classification by low density separation," in *Proc. 10th International Workshop on Artificial Intelligence and Statistics*, pp. 57–64, 2005.

[9] O. Chapelle, B. Scholkopf, and A. Zien, eds., *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

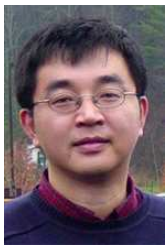
[10] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proc. 16th International Conference on Machine Learning*, pp. 200–209, 1999.

[11] G. Fung and O. Mangasarian, "Semi-supervised support vector machines for unlabeled data classification," *Optimization Methods and Software*, vol. 15, pp. 29–44, 2001.

[12] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: a geometric framework for learning from examples," Technical Report TR-2004-06, University of Chicago, Dept of Computer Science, 2004.

[13] V. Vural, G. Fung, J. G. Dy, and B. Rao, "Semi-supervised classifiers using a-priori metric information," *Optimization Methods and Software Journal, Special Issue on Machine Learning and Data Mining*, to appear.
[14] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. 13th International Conference on Machine Learning*, pp. 148–156, 1996.
[15] C. Rosenberg, M. Hebert, and H. Schneiderman, "Semi-supervised self-training of object detection models," in *Proc. 7th Workshopn on Applications of Computer Vision*, vol. 1, pp. 29–36, January 2005.
[16] K. P. Bennett, A. Demiriz, and R. Maclin, "Exploiting unlabeled data in ensemble methods," in *Proc. 8th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 289–296, 2002.
[17] F. d'Alche Buc, Y. Grandvalet, and C. Ambroise, "Semi-supervised marginboost," in *NIPS 14*, pp. 553–560, 2002.
[18] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proc. 20th International Conference on Machine Learning*, pp. 912–919, 2003.
[19] O. Chapelle and A. Zien, "Semi-supervised classification by low density separation," in *Proc. 10th International Workshop on Artificial Intelligence and Statistics*, pp. 57–64, 2005.
[20] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *COLT: Proceedings of the Workshop on Computational Learning Theory*, pp. 92–100, Morgan Kaufmann, San Francisco, CA, 1998.
[21] D. Miller and H. Uyar, "A mixture of experts classifier with learning based on both labeled and unlabeled data," in *NIPS 9*, pp. 571–577, 1996.
[22] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using EM," *Machine Learning*, vol. 39, pp. 103–134, 2000.
[23] N. D. Lawrence and M. I. Jordan, "Semi-supervised learning via gaussian processes," in *NIPS 17*, pp. 753–760, 2005.
[24] K. Bennett and A. Demiriz, "Semi-supervised support vector machines," in *NIPS 11*, pp. 368–374, 1998.
[25] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, pp. 119 – 139, August 1997.
[26] D. Zhou, J. Huang, and B. Scholkopf, "Learning from labeled and unlabeled data on a directed graph," in *Proc. 22nd International Conference on Machine Learning*, pp. 1036–1043, 2005.
[27] J. Friedman, T. Hastie, and R. Tibshirani, "Special invited paper. additive logistic regression: a statistical view of boosting," *The Annals of Statistics*, vol. 28, pp. 337–374, April 2000.
[28] P. K. Mallapragada, R. Jin, A. K. Jain, and Y. Liu, "Semiboost: Boosting for semi-supervised learning," Tech. Rep. MSU-CSE-07-197, Michigan State University, 2007.
[29] L. Mason, J. Baxter, P. Bartlett, and M. Frean, "Boosting algorithms as gradient descent in function space," in *NIPS 12*, pp. 512–518, 1999.
[30] T. Minka, "Expectation-maximization as lower bound maximization," *Tutorial published on the web at http://www-white.media.mit.edu/ tpminka/papers/em.html*, 1998.
[31] A. Jain and X. Lu, "Ethnicity identification from face images," in *Proceedings of the SPIE., Defense and Security Symposium*, vol. 5404, pp. 114–123, 2004.
[32] A. K. Jain and F. Farrokhina, "Unsupervised texture segmentation using gabor filters," *Pattern Recognition*, vol. 24, pp. 1167–1186, 1991.
[33] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, CA, 2nd ed., 2005.
[34] L. Reyzin and R. E. Schapire, "How boosting the margin can also boost classifier complexity," in *Proc. 22nd International Conference on Machine Learning*, pp. 753–760, 2006.
[35] J. Platt, N. Cristianini, and J. Shawe-Taylor, "Large margin dags for multiclass classification," in *NIPS 12*, 2000.

Pavan Kumar Mallapragada received the B.Tech (Hons.) and M.S. degrees from the Department of Computer Science and Engineering, IIIT Hyderabad in 2003 and 2005 respectively. He is currently a Ph.D student at the Department of Computer Science & Engineering at Michigan State University. He worked as a summer research intern at Yahoo Inc. in 2006. His research interests include machine learning and pattern recognition, specifically semi-supervised and unsupervised learning. He is a student member of the IEEE.



Rong Jin is an Associate Professor in the Computer and Science Engineering Department at Michigan State University. He is working in the areas of statistical machine learning and its application to information retrieval. He has published over eighty conference and journal articles on the related topics. Dr. Jin holds a B.A. in Engineering from Tianjin University, an M.S. in Physics from Beijing University, and an M.S. and Ph.D. in Computer Science from Carnegie Mellon University. He received the NSF Career Award in 2006.

Anil K. Jain is a University Distinguished Professor in the Department of Computer Science & Engineering at Michigan State University. His research interests include pattern recognition and biometric authentication. He received the 1996 IEEE Trans. Neural Networks Outstanding Paper Award and the Pattern Recognition Society best paper awards in 1987 and 1991. He served as the Editor-in-Chief of the IEEE Trans. PAMI (1991-94). He is a fellow of the AAAS, ACM, IEEE, IAPR and SPIE. He has received Fulbright, Guggenheim, Alexander von Humboldt, IEEE Computer Society Technical Achievement, IEEE Wallace McDowell and IAPR King-Sun Fu awards. Holder of six patents in the area of fingerprints, he is the author of a number of books, including Handbook of Biometrics (2007), Handbook of Multibiometrics (2006), Handbook of Face Recognition (2005), Handbook of Fingerprint Recognition (2003), BIOMETRICS: Personal Identication in Networked Society (1999) and Algorithms For Clustering Data (1988). ISI has designated him as a highly cited researcher. According to Citeseer, his book Algorithms for Clustering Data (Prentice-Hall, 1988) is ranked #93 in Most Cited Articles in Computer Science. He is currently serving as an Associate Editor of the IEEE Trans. Information Forensics and Security and ACM Trans. Knowledge Discovery in Data. He is a member of the Defense Science Board and The National Academies committees on Whither Biometrics and Improvised Explosive Devices.

Yi Liu received his B.S. and M.S. degrees in Automation from Tsinghua University, China in 2000 and 2003 respectively, and a Ph.D. degree in Computer Science from Michigan State University in 2007. He is now a research engineer at Google Inc. His research interests include machine learning, information retrieval and natural language processing.