SemiBoost: Boosting for Semi-supervised Learning

Pavan Kumar Mallapragada, Rong Jin, Anil K. Jain, and Yi Liu

Department of Computer Science and Engineering,

Michigan State University, East Lansing, MI-48823

{pavanm,rongjin,jain,liuyi3}@cse.msu.edu

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. 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) any supervised learning algorithm can be improved with a multitude of unlabeled data, (b) efficient computational algorithm by the iterative boosting algorithm, and (c) exploiting both manifold and cluster assumption in training classification models. Our empirical study on 16 different datasets 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. 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.

While semi-supervised classification is a relatively new field, the idea of using unlabeled samples for prediction was conceived several decades ago [1], in the form of *transduction*. Given a set of labeled samples (labeled set), and a set of unlabeled samples (transduction set), the goal of a transductive learner is to predict the labels of the samples in the transduction set. On the other hand, inductive algorithms build a general decision rule over the input feature space, using the samples from both labeled and transduction sets. This rule can then be applied later on unlabeled data points unseen during the training phase. To differentiate from the unlabeled data used for training, we refer to this set of unlabeled data as the induction set.

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 [2], [3], Markov random walks [4], Graph cuts [5], Spectral graph transducer [6], and Low density separation [7] proposed in the literature are based on this assumption. Most of these algorithms tend to be inherently transductive. While transductive learning has some useful applications (e.g., Content Based Image Retrieval [8]), many pattern recognition and machine learning tasks involve prediction of unseen data, which requires inductive algorithms. Some approaches have been developed to extend the transductive algorithm to be inductive [9]. These are called the out-of-sample extensions, which are applicable only to a few graph based algorithms. An intuitive way to convert a transductive algorithm to an inductive algorithm is by training a suitable classifier on the data, and then assuming the predicted labels on the transduction set as true labels. However, the predictions made by the transductive classifier are independent of the supervised classifier that will be trained later on this data, which may lead to a suboptimal performance.

Several algorithms have been proposed for semi-supervised learning which are naturally inductive. Usually, they are based on an assumption, called the *cluster assumption* [10]. 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 [8] 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.

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-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

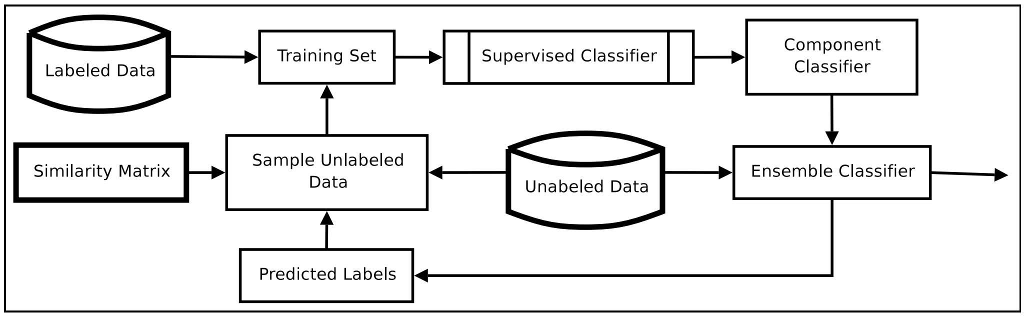


Fig. 1. Block diagram of the proposed algorithm, SemiBoost. The inputs to SemiBoost are: labeled data, unlabeled data and the similarity matrix.

to most boosting algorithms [13], 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 Fig 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 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 [14], ASSEMBLE [15] and Semi-supervised MarginBoost [16]. 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 are far away from the decision boundary. As a result, the classifier trained by the selected unlabeled examples is very 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. It is to be noted that margin maximization by itself does not provide any additional information to the semi-supervised learner.

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 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 in each iteration. The new classification model will be combined linearly with the existing classification models to make improved predictions. The following section discusses the existing semi-supervised learning methods, and their relationship with SemiBoost.

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 [2], [3]	Graph-based; Maximize label consistency using Graph Laplacian	T
	Min-cuts [5]	Edge-weight based graph-partitioning algorithm constraining nodes with same label to be in same partition	T
	MRFs [4], GRFs [9]	Markov random field and Gaussian random field models	T
	LDS [17]	TSVM trained on a dimensionality reduced data using graph-based kernel	T
	SGT [6]	Classification cost minimized with a Laplacian regularizer	T
	LapSVM [12]	SVM with Laplacian regularization	I
Cluster Assumption	Co-training [18]	Maximizes predictor consistency among two distinct feature views	I
	Self-training [14]	Assumes pseudo-labels as true labels and retrains the model	I
	SSMB [16]	Maximizes pseudo-margin using boosting	I
	ASSEMBLE [15]	Maximizes pseudo-margin using boosting	I
	Mixture of Experts [19]	EM based model-fitting of mixture models	I
	EM-Naive Bayes [20]	EM based model-fitting of Naive Bayes	I
	TSVM [8], S3VM [11]	Margin maximization using density of unlabeled data	I
	Gaussian processes [21]	Bayesian discriminative model	I
Manifold & Cluster Assumptions	SemiBoost (Proposed)	Boosting with a graph Laplacian inspired regularization	I

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). From Table I, one can see that almost all the algorithms based on manifold assumption are usually transductive, Laplacian SVM being an exception. On the other hand, almost all the approaches that are based on cluster assumption tend to be inductive.

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 [9], Spectral Graph Transducer (SGT) [6], Gaussian process based approach [21], Manifold Regularization [12] and Label Propagation approach [2], [3]. 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 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 [5] 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 [22], 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 [4], Gaussian random fields and harmonic functions [9]. The label propagation and harmonic function approach assume that the given labels are exact, and split the objective function in such a way that only the labels of unknown samples are estimated. Spectral graph transducer on the other hand, predicts the labels of even the labeled samples, however, with a penalty for violating the given labels.

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) [8], Semi-supervised Support Vector Machine (S3VM) [11], and Gaussian processes with null category noise model [21]. 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.

Kernel learning methods compute the optimal kernel matrices using both the labeled and the unlabeled examples. Cluster kernel [23], kernel alignment method [24], semi-definite programming approach [25], and graph kernel approach [26] take this approach. SemiBoost is built on the assumption that the kernel matrix is known. The kernels obtained using the kernel learning methods can be directly incorporated into SemiBoost, thereby deriving the advantages specific to optimal kernel matrices.

Ensemble methods have gained significant popularity under the realm of supervised classification, with the availability of algorithms such as AdaBoost [27]. The semi-supervised counter parts of ensemble algorithms rely on the cluster assumption, and prime examples include ASSEMBLE [15] and Semi-supervised MarginBoost (SSMB) [16]. 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, 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 [16], [15]. Therefore, it is solving a different problem of boosting semi-supervised algorithms, in contrast with the proposed algorithm.

Self-training [14] is an intuitive wrapper-based approach for improving a supervised algorithm using unlabeled samples. It first labels the unlabeled samples by training a supervised classifier on labeled samples. Those unlabeled samples which have high confidence of prediction are included in the training set along with the labeled samples, and the classifier is retrained. Because of its capability to utilize existing specialized supervised algorithms, self-training is popular in several applications, e.g text categorization, when compared to graph based approaches which usually are computationally complex (matrix inversion [9], eigen value computation [6]). Also graph based approaches tend to replace the existing supervised algorithms, thereby ignoring any special advantages supervised learners may present.

However, self-training solely relies on the labels predicted by a classifier on the unlabeled data for training a new supervised classifier. Given the small size of the available labeled training sample in many applications, there is a high probability of making an error on labeling the unlabeled samples. Any such error in the learning of base classifier gets magnified over the self-training iterations. Similar observations were made in [28]. The proposed algorithm on the other hand, gives more reliable label predictions as it combines the similarity information effectively with classifier predictions. While retaining the advantage of being wrapper based, the proposed algorithm SemiBoost has the advantages of graph based classifiers embedded into the objective function by enforcing the consistency between the similarity measure and the predicted labels.

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 $\mathcal{A}$ using the unlabeled examples and the pairwise similarity S.

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

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

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, 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 [29].

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

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

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 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 [30].

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

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

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

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

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 limited amount of labeled data available, 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,

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

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.

C. Algorithm

We derive the boosting algorithm using the approach presented in [30]. An alternate, conventional way to derive the boosting algorithm to use the Function Gradient approach presented in [31]. This approach may also be viewed as a relaxation approach to approximate 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.

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.,

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

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:

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

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

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

where

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

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

and $\delta(x,y)=1$ when $x=y$ and 0 otherwise. *Proof:* In appendix. ■

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.

The upper bound in Eq (8) is not very useful for a boosting algorithm because it is difficult to compute the weighting scheme of examples directly from Eq (8). The expression in Eq (8) is further simplified in the following proposition.

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

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

Proof: In appendix. ■

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

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

Proof: In appendix. ■

In proposition 3, the key ingredients required for a boosting algorithm are established. Fig 1 summarizes the SemiBoost algorithm. 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 [31], α can be expressed as

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

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. $\epsilon_{t+1} > \frac{1}{2}$, it returns the current classifier H_t. This situation has a direct correspondence with the condition in SemiBoost where the algorithm stops when $\alpha < 0$. From Eq (11) (or rather directly from Eq (12)), we can see that this happens only when the denominator exceeds the numerator, which means $\epsilon_{t+1} > \frac{1}{2}$ is equivalent to the condition $\alpha < 0$. 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 [30].

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 of samples based on the previous iteration. In SemiBoost, since we are predicting the labels for unlabeled samples, there is no way 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 high confidence labelings 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 is playing an important role in guiding the sample selection. This is in contrast with the previous approaches like ASSEMBLE and SSMB, where the sample selection is performed 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

```
• 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 $\alpha_t$ using Equation (11)
    – Update the classification function as $H(\mathbf{x}) \leftarrow H(\mathbf{x}) + \alpha_t h_t(\mathbf{x})$
```

Fig. 2. The SemiBoost algorithm

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: In appendix. ■

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:* In appendix. ■

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 and if they are given too much importance, 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 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 high 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 a 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 [9]. 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 it is a difficult problem to choose the right scale parameter.

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.

A. Datasets

SemiBoost was evaluated on 16 different datasets: 4 benchmark datasets provided in [10], 10 UCI datasets and 2 datasets from ethnicity classification from face images [32] and texture classification [33]. 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.

The transductive performance of semi-supervised learning algorithms is well-studied [10, 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 [6], an inductive version of Low Density Separation (LDS) [7] 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.

B. *Experimental setup*

The experimental setup aims to study the following important aspects of the performance of the SemiBoost algorithm:

- Does SemiBoost improve the inductive performance of *any* base classifier?
- How does SemiBoost algorithm compare with other inductive semi-supervised algorithms?
- How stable is SemiBoost with respect to the scale parameter σ?
- How does the number of unlabeled samples randomly selected from the data at each iteration, affect the performance of SemiBoost?

- What is the computational cost of the SemiBoost and the baseline 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 $C = n_l/n_u$.

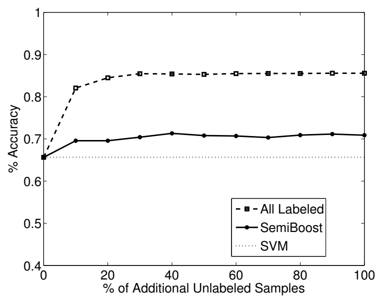
The first experiment studies the improvement in the performance of three different base classifiers ($h_t(\mathbf{x})$) after applying SemiBoost: Decision Stump (DS), the J48 decision tree algorithm (J48), and the Support Vector Machine with the sequential minimal optimization (SVM) algorithm. Software WEKA [34] 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

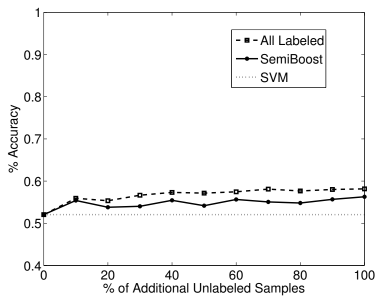
1) Choice of base classifier: Table II compares the supervised and the three benchmark semi-supervised algorithms to the SemiBoost algorithm. The columns DS, J48 and SVM give the performances of the base classifiers on the induction set. The columns SB-X give the inductive performances of SemiBoost with 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. 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. On almost all the datasets we see that SemiBoost performs significantly better than ILDS. This shows that an approach that is inductive by nature outperforms the the idea of converting a transductive algorithm like LDS into an inductive algorithm. SemiBoost outperforms TSVM on 14 out of 16 datasets. Also, TSVM had difficulty converging on three datasets in a reasonable amount of time (20 hours). SemiBoost performs comparably to LapSVM; on 8 datasets LapSVM outperformed SemiBoost, while SemiBoost outperformed LapSVM on the remaining 8.

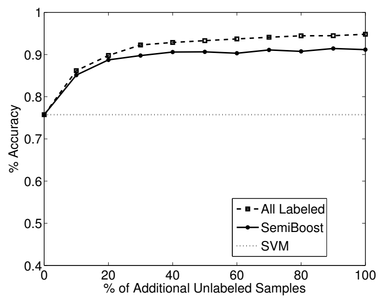
There are datasets where one of the base classifiers outperforms SemiBoost. But in these cases, one of the base classifiers outperforms all the semi-supervised algorithms (e.g., SVM outperforms all the algorithms on COIL2, vehicle, sat and house datasets). This indicates that the unlabeled data do not 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



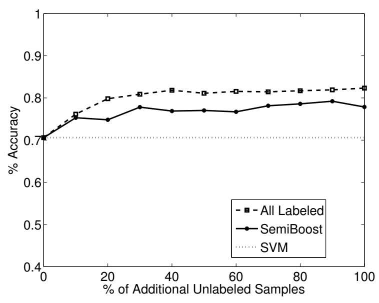
(a) austra (UCI)



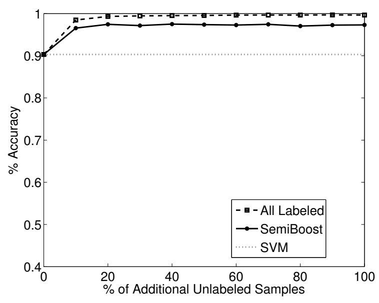
(b) bupa (UCI)



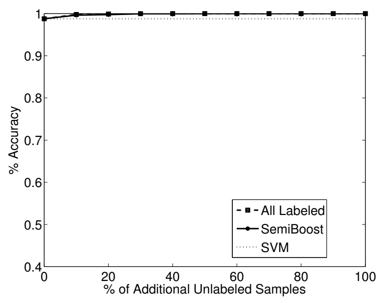
(c) wdbc (UCI)



(d) optdigits (UCI)



(e) mfeat-fou (UCI)



(f) isolet (UCI)

Fig. 3. 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).

algorithms, we observed that the SemiBoost tends to perform close to the baseline compared to the others in most cases.

3) Performance with unlabeled data increment: Fig. 3 shows the performance of SemiBoost on six of the UCI datasets (Figs. 3 (a)-(f)) 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 always labeled, and the performance obtained on the inductive test set by training an SVM with default parameters is shown with a dotted line in the plot. The rest of the (unlabeled) samples in the training set are added to the training set in units of 10%. The dashed line shows the performance obtained by the SVM classifier when all these added samples are labeled using their ground truth. The solid line shows the performance of the SemiBoost algorithm, treating the added samples as

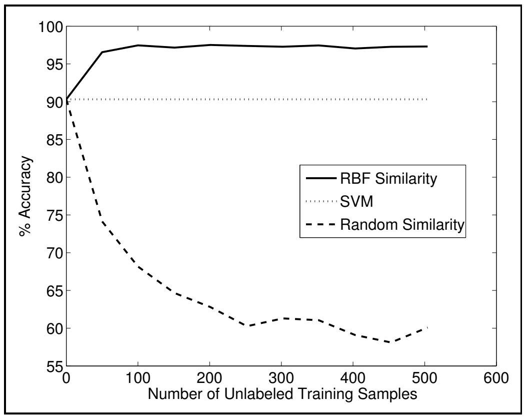
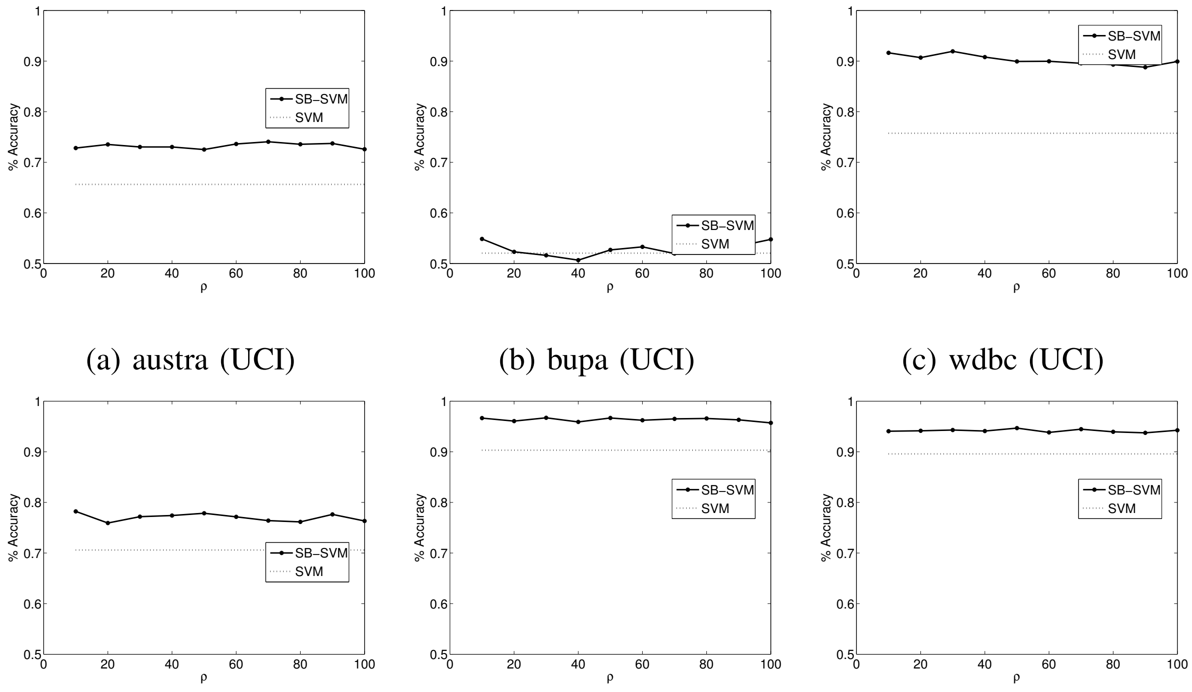


Fig. 4. Comparing the performance of SemiBoost with the RBF similarity matrix (shown in bold), and a random similarity matrix (dashed line). Accuracy of SVM is shown as a dotted line for reference.

unlabeled. 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) Choice of Similarity function: Similarity matrix (kernel matrix) encodes the geometry of the manifold on which the data lies. Therefore, it is crucial that the choice of similarity matches with the underlying geometry of the data. The performance of SemiBoost on optdigits dataset (classes 2,4) with two different similarity matrices is shown in Fig. 4. For each choice of similarity matrix, unlabeled samples are added incrementally to the training set and the performance is evaluated on the induction set. The solid line shows the performance of SemiBoost with RBF similarity and the dashed line shows the performance of SemiBoost using a random symmetric similarity matrix. This establishes that when the similarity matrix does not reflect the underlying geometry, addition of unlabeled data deteriorates the performance. The same reason may be attributed to the datasets shown in the Table II where semi-supervised algorithms perform worse than the baseline algorithm (e.g. COIL2 and vehicle datasets).

5) Sensitivity to parameter σ: Fig. 5 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 similarity values is recommended, based on empirical observations.

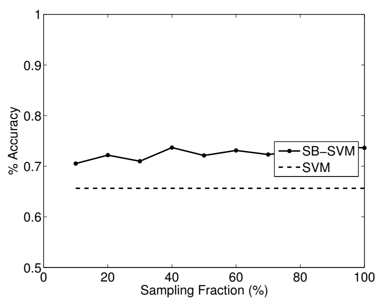


(a) austra (UCI) (b) bupa (UCI) (c) wdbc (UCI)

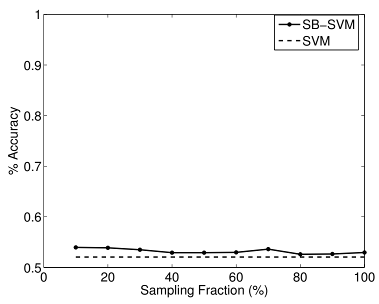
(d) optdigits (UCI) (e) mfeat-fou (UCI) (f) isolet (UCI)

Fig. 5. 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.

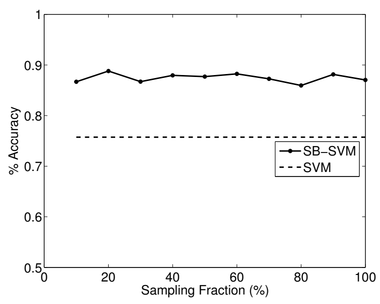
6) Sensitivity to selected sample size at each iteration: The importance of choosing an appropriate sample size in semi-supervised ensemble algorithms was explained in III-D.1. We study the performance of SemiBoost as the size of unlabeled data sampled from the training set



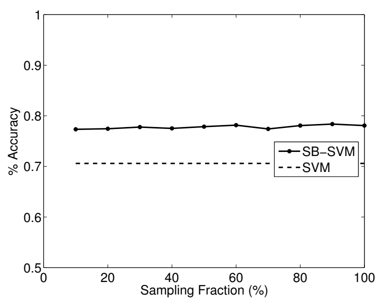
(a) austra (UCI)



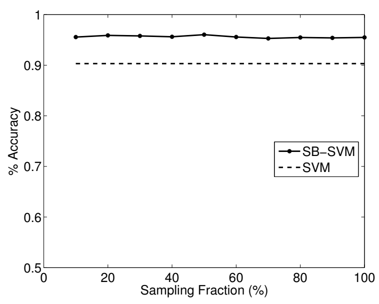
(b) bupa (UCI)



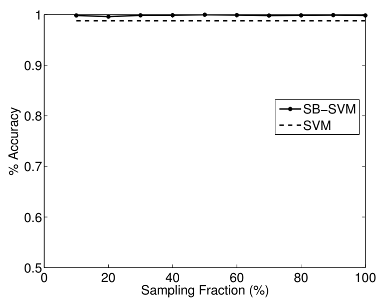
(c) wdbc (UCI)



(d) optdigits (UCI)



(e) mfeat-fou (UCI)

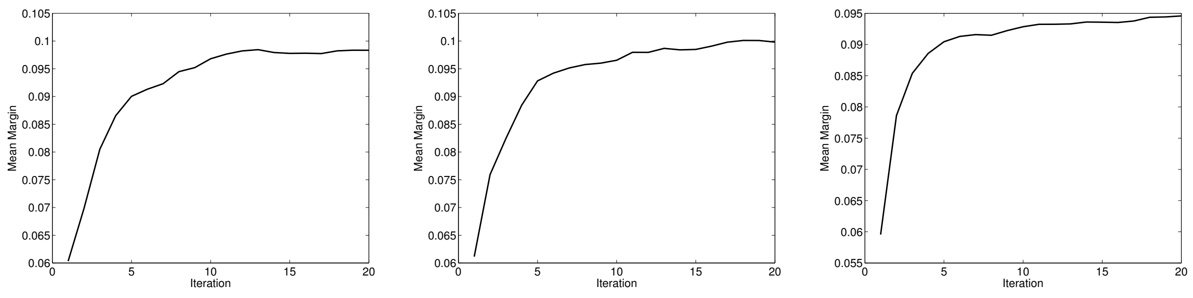


(f) isolet (UCI)

Fig. 6. Performance of baseline algorithm SVM with 10 labeled samples, with increasing number of unlabeled data points sampled. The horizontal axis shows the percentage of unlabeled samples selected, ranging from 10% to 100%. The vertical axis shows the classification accuracy.

varies from 10% to 100% of the total number of unlabeled samples available in the training set. The plots in Fig. 6 show the performance of SemiBoost vs the sampling rate. As a general rule, sampling 10%-30% of the unlabeled training samples is a safe choice, and it usually results in an improved performance. For several datasets (Figs. 6(a)-(f)), the performance is stable with respect to the number of unlabeled samples selected at each iteration. This stability is attributed to the usage of pairwise similarity information to guide the sample selection.

7) 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 $|H(\mathbf{x}_i)|$ [15], [16]. The mean margin is the empirical average of $|H(\mathbf{x}_i)|$ over the unlabeled



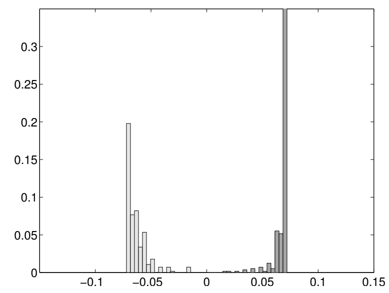
(a) Decision Stump (b) J48 (c) SVM

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

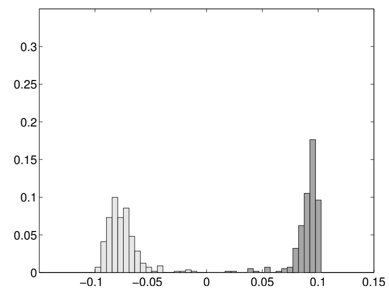
data used for training. Figs. 7((a)-(c)) show the mean-margin value on optdigits dataset (classes 2,4) over the iterations using Decision Stump, J48 and SVM as the base classifiers, respectively. The value of the mean-margin increases over the iterations, irrespective of the choice of the base classifier. Recently, it was shown in [35] 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 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 and computation

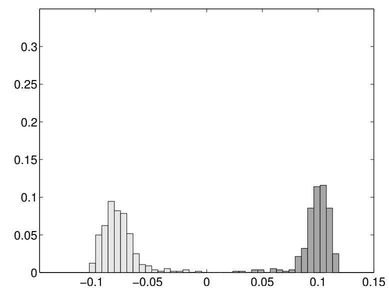
According to Theorem 1, SemiBoost converges exponentially. The following section demonstrates the convergence of SemiBoost on an example dataset. To illustrate the convergence, we chose the two most populous classes in the optdigits dataset, namely digits 2 and 4. The change in the objective function 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.



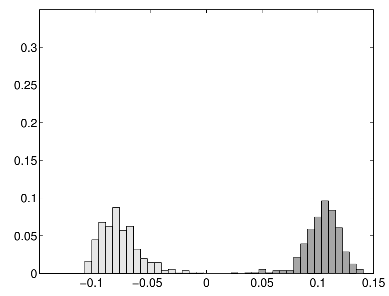
(a) Iteration 1



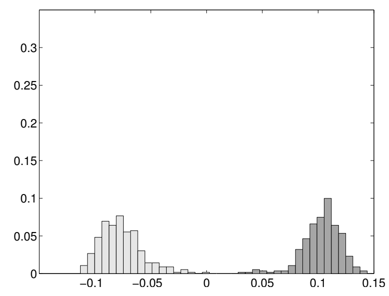
(b) Iteration 2



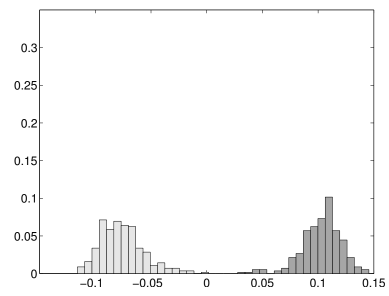
(c) Iteration 3



(d) Iteration 10



(e) Iteration 15



(f) 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 $t \in \{1, 2, 3, 10, 15, 20\}$. SVM is used as the base classifier. The light and dark bars in the histogram correspond to the two classes 2 and 4 respectively.

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.

SemiBoost is a meta-classification approach, which builds on a given base classifier. The complexity of SemiBoost, thus depends on the complexity of training the base classifier. Let $M_{tr}(n)$ be the complexity of training the base classifier over n samples, and $M_{te}(n)$ be the testing complexity. The similarity matrix is already assumed to be given, or it can be computed

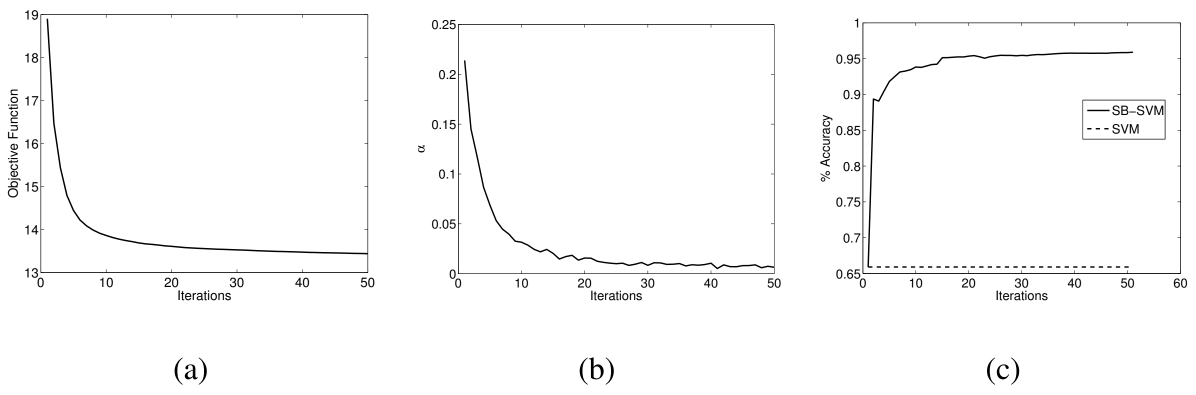


Fig. 9. Plots of (a) objective function, (b) α value and (c) Accuracy of SemiBoost when run over two classes (2,4) of the optdigits dataset using Decision Stump as the base classifier. The accuracy of the base classifier is 65.9%.

in $O(n^2)$ time, for n samples. Using SemiBoost, there are three steps to train the component classification model at each iteration, (i) to assign labels to the unlabeled data, $M_te(n_u)$, (ii) compute the confidences p_i and q_i. This step takes $O(n_u^2)$ time for a dense graph and $O(n_u)$ for a sparse graph, and (iii) to train a new classifier on n_s samples, $M_tr(n_s)$. If we consider that n_s is significantly smaller than n_u, and $M_{te}(n_u)$ is roughly linear, then the highest order is $O(n_u^2)$ for a dense graph and $O(n_u)$ for a sparse graph. The above analysis assumes that the number of iterations is upper bounded. Given the worst case time for training a classification model with n_u labeled examples with SVM is $O(n_u^3)$, SemiBoost is substantially faster. Indeed, semi-supervised learning algorithm such as Harmonic function based approach has complexity of $O(n_u^3)$ due to the computation of the inverse of the Laplacian matrix. Fig. 10 shows the increase in computational time due to addition of unlabeled samples. Testing involves evaluation of the T classifiers, which is the case with any of the ensemble classifiers.

V. 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

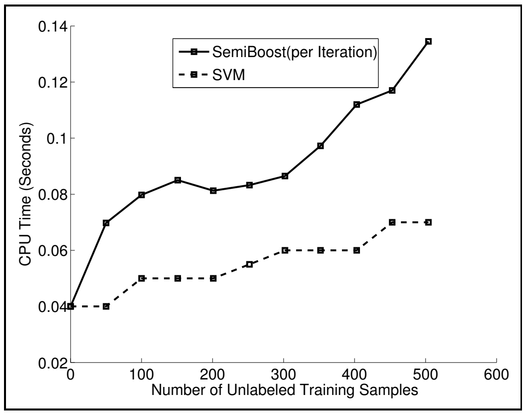


Fig. 10. The plot comparing the CPU times of SemiBoost per iteration and training an SVM with labeled data of same size.

in the presence of unlabeled samples. Overall, the results 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).

APPENDIX

PROOFS OF PROPOSITIONS AND THEOREM

A. *Proposition 1*

Proof: We first bound the quantity $\exp(\alpha(h_i - h_j))$ as

$$\exp(\alpha(h_i - h_j)) \leq \frac{1}{2}\left(\exp(2\alpha h_i) + \exp(-2\alpha h_j)\right),$$

and then upper bound the objective function F by the following expression:

$$F \leq \sum_{i=1}^{n_l}\sum_{j=1}^{n_u} S_{i,j}\exp(-2y_i^l H_j)\exp(-2\alpha y_i^l h_j) + C\sum_{i,j=1}^{n_u}\frac{S_{i,j}}{2}\exp(H_i - H_j)\left(\exp(2\alpha h_i) + \exp(-2\alpha h_j)\right).$$

We denote the above upper bound by $\overline{F}_1$, which can be rewritten as:

$$\overline{F}_1 \;=\; \sum_{i=1}^{n_u}\exp(-2\alpha h_i)p_i + \exp(2\alpha h_i)q_i, \tag{13}$$

where

$$p_i \;=\; \sum_{j=1}^{n_l} S_{i,j}e^{-2H_i}\delta(y_j,1) + \frac{C}{2}\sum_{j=1}^{n_u} S_{i,j}e^{H_j - H_i} \tag{14}$$

$$q_i \;=\; \sum_{j=1}^{n_l} S_{i,j}e^{2H_i}\delta(y_j,-1) + \frac{C}{2}\sum_{j=1}^{n_u} S_{i,j}e^{H_i - H_j}. \tag{15}$$

■

B. Proposition 2

Proof: The expression in Eq (8) is simplified using the following bound:

$$\exp(\gamma x) \leq \exp(\gamma) + \exp(-\gamma) - 1 + \gamma x,\ \forall x \in [-1,+1],$$

which leads to an upper bound for $\overline{F}_1$:

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

Let the above bound be represented using $\overline{F}_2$. The bound is composed of two terms, of which the first is independent of the classifier predictions h_i. Therefore, to minimize the objective function using the classifier at each iteration, samples with high value of $|p_i - q_i|$ need to be selected. ■

C. Proposition 3

Proof: From Proposition 1 we have,

$$\begin{aligned}\bar{F}_1 &= \sum_{i=1}^{n_u} p_i \exp(-2\alpha h_i) + q_i \exp(2\alpha h_i) \\ &= \sum_{i=1}^{n_u} \exp(-2\alpha)(p_i\delta(h_i,1) + q_i\delta(h_i,-1)) + \sum_{i=1}^{n_u} \exp(2\alpha)(p_i\delta(h_i,1) + q_i\delta(h_i,-1)) \\ &= \exp(-2\alpha)\eta(1-\epsilon) + \exp(2\alpha)\eta\epsilon \end{aligned} \tag{16}$$

where $\eta = \sum_{i=1}^{n_u}(p_i + q_i)$ and $\epsilon = \left(\sum_{i=1}^{n_u} p_i\delta(h_i,-1) + \sum_{i=1}^{n_u} q_i\delta(h_i,1)\right)/\eta$. Differentiating the above equation w.r.t α and equating it to zero, we have

$$-2\exp(-2\alpha)\eta(1-\epsilon) + 2\exp(2\alpha)\eta\epsilon = 0. \tag{17}$$

From Eq (17) we obtain

$$\alpha = \frac{1}{4}\ln\left(\frac{1-\epsilon}{\epsilon}\right). \tag{18}$$

■

D. Theorem 1 : Convergence

Proof: Similar to the Boosting algorithm, we can show that the proposed semi-supervised boosting algorithm is able to reduce the objective function exponentially. Let F_t denote the objective function at the t-th iteration. Let $\alpha_t > 0$ denote the combination weight of the tth iteration. We first show that

$$F_{t+1} = F_t \left(\frac{2}{\exp(2\alpha_t) + \exp(-2\alpha_t)}\right). \tag{19}$$

The above equality indicates that the reduction factor is $2/(\exp(2\alpha_t) + \exp(-2\alpha_t))$, which is guaranteed to be less than 1 when α_t is positive. The proof of the equality in Eq (19) is straightforward. According to the derivation in the previous section,

$$F_{t+1} \quad = \quad \sum_{i=1}^{n_u} \exp(-2\alpha_t h_i)p_i + \exp(2\alpha_t h_i)q_i.$$

By replacing α_t with the expression

$$\alpha_t = \frac{1}{4} \ln \frac{\sum_{i=1}^{n_u} p_i \delta(h_i, 1) + \sum_{i=1}^{n_u} q_i \delta(h_i, -1)}{\sum_{i=1}^{n_u} p_i \delta(h_i, -1) + \sum_{i=1}^{n_u} q_i \delta(h_i, 1)} \tag{20}$$

and by defining the quantities $\epsilon_t = \sum_{i=1}^{n_u} p_i \delta(h_i, -1) + \sum_{i=1}^{n_u} q_i \delta(h_i, 1)$, and $\eta_t = \sum_i p_i + q_i$, we have

$$F_{t+1} = 2\sqrt{(\eta_t - \epsilon_t)\epsilon_t} \ . \tag{21}$$

Using the fact that $F_t = \eta_t$, we have

$$F_{t+1} = 2\frac{F_t}{\eta_t}\sqrt{(\eta_t - \epsilon_t)\epsilon_t} = 2F_t \frac{\epsilon_t}{\eta_t}\sqrt{\left(\frac{\eta_t - \epsilon_t}{\epsilon_t}\right)}. \tag{22}$$

Using the relation $\alpha_t = \frac{1}{4}\log(\eta_t - \epsilon_t/\epsilon_t)$, we prove the equality in Eq (19) as follows:

$$F_{t+1} = 2F_t \frac{1}{\exp(4\alpha t) + 1} \exp(2\alpha_t) = \frac{F_t}{\cosh(2\alpha_t)}. \tag{23}$$

Extending this equality to F_0, we have the final expression for F_{t+1} in terms of the initial value of the objective function F_0, i.e.,

$$F_{t+1} \quad = \quad F_0 \exp\left(-\sum_{i=1}^{t} \gamma_i\right), \tag{24}$$

where $F_0 = \sum_{i=1}^{n_u} \sum_{j=1}^{n_l} S_{i,j}$ and $\gamma_i = \log(\cosh(2\alpha_t))$. As indicated in Eq (24), the objective function F is reduced exponentially iteratively. ■

E. Corollary 1

Proof: The connection between α_i and the error ϵ_t may be used to bound the objective function in terms of classification error at each iteration. From the theorem, we have $\gamma_t =$

$\log(\cosh(\alpha_t))$. Note that,

$$\gamma_t = \log(\cosh(\alpha_t)) \geq \log(\exp(\alpha_t)) = \alpha_t.$$

Using the definition of α_t from Eq (12), we have $\exp(-\gamma_t) \leq \left(\frac{1-\epsilon_t}{\epsilon_t}\right)^{1/4}$. Using this inequality with the bound in the Theorem 1 results in

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

■

F. Relation to Graph based approaches

Graph based approaches use the properties of the Graph Laplacian matrix to predict the labels of the unlabeled samples. SemiBoost minimizes the prediction error of the model, retaining the regularization properties of the Laplacian used in the graph based methods. The relation between Graph based approaches and SemiBoost can be established by exploring the connection between the graph Laplacian and the $\cosh(.)$ cost function used in SemiBoost.

Consider the power-series expansion of $\cosh(x)$, truncated after two terms,

$$\cosh(x) \approx 1 + \frac{x^2}{2}.$$

Therefore, the inconsistency term can be written as

$$\sum_{i,j} S_{i,j} \cosh(y_i - y_j) \approx \sum_{i,j} S_{i,j} + \sum_{i,j} S_{i,j}(y_i - y_j)^2. \tag{25}$$

Since the first term is a constant, we can see that by minimizing our consistency, we are also minimizing the graph Laplacian.

G. Relation to Manifold Regularization

Laplacian Support Vector Machines (LapSVM) [12], uses the following objective function:

$$\min_f \frac{1}{n_l} \sum_{i=1}^{n_l} (1 - y_i^l f(x_i))_+ + \gamma_A ||f||_K^2 + \frac{\gamma_I}{(n_u + n_l)^2} \mathbf{f}^T L \mathbf{f},$$

where $\mathbf{f}$ is the vector of length $n_l + n_u$, with i-th element $f_i = f(\mathbf{x}_i)$. This is equivalent to adding a Laplacian based regularizer to the SVM objective function. The regularizer can be split into three parts as

$$\mathbf{f}^T L \mathbf{f} = \mathbf{f}_l^T L_{ll} \mathbf{f}_l + 2\mathbf{f}_l^T L_{ul} \mathbf{f}_u + \mathbf{f}_u^T L_{uu} \mathbf{f}_\mathbf{u} \tag{26}$$

where $\mathbf{f} = [\mathbf{f}_l; \mathbf{f}_u]$, where $\mathbf{f}_l$ corresponds to the labeled samples, and $\mathbf{f}_u$ corresponds to unlabeled samples. Similarly, the Laplacian matrix is split into blocks based on the labeled and unlabeled samples, with L_{ll} as the block indexed corresponding to labeled samples, L_{uu} corresponding to the unlabeled samples, and L_{ul} corresponding to both labeled and unlabeled samples. Using the connection between the Laplacian and the $\cosh(.)$ penalty used in SemiBoost, the second and third terms in Eq (26) correspond to the inconsistencies F_l and F_u in the objective function, respectively. The SVM part of the objective function of LapSVM is replaced by the base classifier chosen trained on the labeled samples.

References

[1] V. Vapnik, *Estimation of Dependences Based on Empirical Data: Springer Series in Statistics*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1982.

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

[3] 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.

[4] M. Szummer and T. Jaakkola, "Partially labeled classification with Markov random walks," in *Advances in Neural Information Processing Systems 14*, pp. 945–952, 2001.

[5] 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.

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

[7] 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.

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

[9] 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.

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

[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] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. 13th International Conference on Machine Learning*, pp. 148–156, 1996.

[14] 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.

[15] 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.

[16] F. d'Alche Buc, Y. Grandvalet, and C. Ambroise, "Semi-supervised marginboost," in *Advances in Neural Information Processing Systems 14*, pp. 553–560, 2002.

[17] 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.

[18] 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.

[19] D. Miller and H. Uyar, "A mixture of experts classifier with learning based on both labeled and unlabeled data," in *Advances in Neural Information Processing Systems 9*, pp. 571–577, 1996.

[20] 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.

[21] N. D. Lawrence and M. I. Jordan, "Semi-supervised learning via gaussian processes," in *Advances in Neural Information Processing Systems 17*, pp. 753–760, 2005.

[22] K. Bennett and A. Demiriz, "Semi-supervised support vector machines," in *Advances in Neural Information Processing Systems 11*, pp. 368–374, 1998.

[23] O. Chapelle, J. Weston, and B. Scholkopf, "Cluster kernels for semi-supervised learning," in *Advances in Neural Information Processing Systems 15*, pp. 585–592, 2002.

[24] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. S. Kandola, "On kernel-target alignment.," in *Advances in Neural Information Processing Systems 14*, pp. 367–373, 2001.

[25] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *Journal of Machine Learning Research*, vol. 5, pp. 27–72, 2004.

[26] X. Zhu, J. Kandola, Z. Ghahramani, and J. Lafferty, "Nonparametric transforms of graph kernels for semi-supervised learning," in *Advances in Neural Information Processing Systems 17*, pp. 1641–1648, 2005.

[27] Y. Freund, "Boosting a weak learning algorithm by majority," *Information and Computation*, vol. 121(2), pp. 256–285, 1995.

[28] R. K. Ando and T. Zhang, "A framework for learning predictive structures from multiple tasks and unlabeled data," *Journal of Machine Learning Research*, pp. 1817–1853, 2005.

[29] 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.

[30] 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.

[31] L. Mason, J. Baxter, P. Bartlett, and M. Frean, "Boosting algorithms as gradient descent in function space," in *Advances in Neural Information Processing Systems 12*, pp. 512–518, 1999.

[32] 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.

[33] A. K. Jain and F. Farrokhina, "Unsupervised texture segmentation using gabor filters," *Pattern Recognition*, vol. 24, pp. 1167–1186, 1991.

[34] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, CA, 2nd ed., 2005.

[35] 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.

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