Semi-Supervised Learning via Regularized Boosting Working on Multiple Semi-Supervised Assumptions

Ke Chen, *Senior Member*, *IEEE*, and Shihai Wang

Abstract—Semi-supervised learning concerns the problem of learning in the presence of labeled and unlabeled data. Several boosting algorithms have been extended to semi-supervised learning with various strategies. To our knowledge, however, none of them takes all three semi-supervised assumptions, i.e., smoothness, cluster, and manifold assumptions, together into account during boosting learning. In this paper, we propose a novel cost functional consisting of the margin cost on labeled data and the regularization penalty on unlabeled data based on three fundamental semi-supervised assumptions. Thus, minimizing our proposed cost functional with a greedy yet stagewise functional optimization procedure leads to a generic boosting framework for semi-supervised learning. Extensive experiments demonstrate that our algorithm yields favorite results for benchmark and real-world classification tasks in comparison to state-of-the-art semi-supervised learning algorithms, including newly developed boosting algorithms. Finally, we discuss relevant issues and relate our algorithm to the previous work.

Index Terms—Semi-supervised learning, boosting framework, smoothness assumption, cluster assumption, manifold assumption, regularization.

1 Introduction

Traditionally, machine learning is categorized as two paradigms, i.e., supervised versus unsupervised learning. *Supervised learning* (SL) finds out a rule for the predictive relationship between input and output from a set of finite examples in the format of input-output pairs, while *unsupervised learning* seeks a structure of interests underlying a data set. In general, SL requires many training examples to establish a learner of the satisfactory generalization capability. The acquisition of training examples is nontrivial for SL, which needs to annotate input data with appropriate labels. In many practical applications, ranging from data mining to machine perception, however, the annotation of input data is often difficult, expensive, and time-consuming, especially when it has to be done manually by experts. On the other hand, there is often a massive amount of unannotated data available. In order to exploit unannotated data, *semi-supervised learning* (SSL) has become a novel paradigm by using a large number of unannotated points together with a small number of annotated examples to build a better learner [31], [40], [9]. Since SSL needs less human effort but could offer higher accuracy, exploiting unannotated data to help SL has received a great deal of attention from the machine learning community.

In SSL, especially semi-supervised classification, the ultimate goal is to find out a classifier which not only minimizes classification errors on the labeled examples, but also must be compatible with the input distribution by monitoring their values on unlabeled points [31], [40], [9]. To work toward the goal, unlabeled data can be exploited in various ways to discover how data are distributed in the input space, and then the information acquired from the unlabeled data is used to find out a good classifier. For different problem settings, SSL is classified as two categories; i.e., *transductive learning* (TL) and *semi-supervised inductive learning* (SSIL). TL [37] concerns only the problem of predicting the labels of test data given in advance based on a labeled data set by taking both labeled and unlabeled data together into account. In contrast, SSIL [18] is the problem of learning a decision rule automatically from a training set consisting of labeled and unlabeled data for other unseen data. In this paper, we focus mainly on SSIL as is demanded by many machine learning and pattern recognition tasks.

Recent studies have revealed that the success of SSL is attributed to the fact that certain *semi-supervised assumptions* (SSAs) hold for the data distribution [9]. As summarized in [9], there are three fundamental SSAs: *semi-supervised smoothness*, *cluster*, and *manifold* assumptions. The semi-supervised smoothing assumption states that if two points in a high-density region are close, then their corresponding labels should be the same or consistent. The cluster assumption is described as follows: If points are located in the same cluster, they are likely to belong to the same class. In other words, the decision boundary is likely to lie in a low data density region, which is also referred to as the *low-density separation* assumption. The manifold assumption states that the high-dimensional data lies on a low-dimensional manifold whose properties ensure more accurate density estimate and/or more appropriate similarity measures.

- *The authors are with the School of Computer Science, The University of Manchester, Kilburn Building, Oxford Road, Manchester M13 9PL, UK. E-mail: chen@cs.manchester.ac.uk, shihai.wang@postgrad.manchester.ac.uk.*

Manuscript received 19 Feb. 2009; revised 14 Aug. 2009; accepted 15 Feb. 2010; published online 16 Apr. 2010.
Recommended for acceptance by M. Yang.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2009-02-0123.
Digital Object Identifier no. 10.1109/TPAMI.2010.92.

To work on the aforementioned SSAs, *regularization* has been employed in SSL to exploit unlabeled data [18]. A number of regularization methods have been proposed based on a cluster or smoothness assumption, which exploits unlabeled data to regularize the decision boundary and therefore affects the selection of learning hypotheses [20], [7], [34], [5], [8], [17]. Working on a cluster or smoothness assumption, most of the regularization methods are naturally inductive. On the other hand, the manifold assumption has also been applied for regularization where the geometric structure behind labeled and unlabeled data is explored with a graph-based representation. In such a representation, examples are expressed as the vertices and the pairwise similarity between examples is described as a weighted edge. Thus, graph-based algorithms make good use of the manifold structure to propagate the known label information over the graph for labeling all nodes [3], [33], [21], [39], [6]. In nature, most of such graph-based regularization algorithms are transductive, although they can be converted into inductive algorithms with the out-of-sample extension [42]. Recently, manifold regularization for inductive learning has also been proposed by constructing a maximum-margin classifier, along with penalizing the corresponding inconsistency with the similarity matrix [1]. To our knowledge, there are only a few algorithms with regularization working on three semi-supervised assumptions simultaneously in literature [5], [8].

As a generic ensemble learning framework [27], boosting works via sequentially constructing a linear combination of base learners, which appears remarkably successful for SL [16]. Boosting has been extended to SSL with different strategies. Semi-supervised MarginBoost [14] and ASSEMBLE [2] were proposed by introducing the "pseudoclass" or the "pseudolabel" concepts to an unlabeled point so that unlabeled points can be treated as same as labeled examples in the boosting procedure. In essence, such extensions work in a self-training-like style; the unlabeled points are assigned pseudoclass labels based on the constructed ensemble learner so far, and, in turn, these pseudoclass labels will be used to find a new learner to be added to the ensemble. As pointed out in [19], such algorithms attempt to minimize both labeled and unlabeled margin cost only. Thus, a hypothesis can be very certain about the classification of unlabeled points with very low margin cost even though these unlabeled points are not classified correctly. The cotraining idea [4] was also extended to boosting, e.g., CoBoost [13] and the Agreement Boost [23]. To our knowledge, none of the aforementioned semi-supervised boosting algorithms takes fundamental SSAs into account explicitly.

Recently, SSAs have been adopted to develop novel boosting algorithms for SL and SSL. In [22], the graph Laplacian regularizer was introduced into the marginal AdaBoost for acquiring the manifold information to favor base learners that are smoothing in a certain sense during ensemble learning. This algorithm was originally proposed for SL but can be extended to SSL [22]. In our previous work [12], we proposed a generic regularizer working on semi-supervised smoothness and manifold assumptions and applicable to several semi-supervised boosting algorithms [13], [14], [2], [23]. However, this regularizer is independent of the boosting margin cost functional, and thus leads to a suboptimal boosting procedure for SSL. In addition, the low-density separation assumption had yet to be investigated, although the possibility of integrating it into the regularizer was discussed [12]. More recently, novel semi-supervised boosting algorithms have been developed based on semi-supervised smoothness and manifold assumptions for binary classification [24], [25] and multiclass classification [36]. However, none of the aforementioned algorithms has yet to take the low-density separation assumption, another form of the cluster assumption, into account. Alternatively, the expectation regularization principle has recently been applied for developing regularized boosting algorithms for SSL [29], [30].

In this paper, we extend our previous work [12] to a semi-supervised boosting framework with regularization working on semi-supervised smoothness, low-density separation, and manifold assumptions [9]. As a result, we first propose a novel cost functional consisting of the margin cost on labeled data and the regularization penalty on unlabeled data based on three fundamental SSAs. Then, we develop the boosting algorithm within the generic margin cost functional framework for boosting [27]. In this framework [27], boosting is treated as a greedy yet stagewise functional minimization procedure where each stage seeks a function from a given subspace so that combining it with those functions already found in the same way can lead to the greatest reduction in terms of a cost functional defined based on training examples. Since our algorithm is within the generic margin cost functional framework developed for generic yet abstract boosting algorithms, it allows a range of various margin cost functions to be applied. To facilitate our boosting learning, we also come up with an initialization setting based on clustering analysis. It is worth stating that our algorithm is developed for binary classification tasks, but easily extended to cope with multiclass classification tasks via the one-against-rest scheme, although this treatment might be less efficient than those methods developed very recently for multiclass boosting without the use of binary decomposition [43], [36], [30]. Extensive experiments demonstrate that our algorithm yields favorite results for benchmark and real-world classification tasks in comparison to many state-of-the-art SSL algorithms [9], including semi-supervised boosting algorithms [2], [24], [25].

In the reminder of this paper, Section 2 briefly reviews the generic margin cost functional framework for boosting. Section 3 presents our regularized semi-supervised boosting algorithm. Section 4 describes the experimental methodology and reports experimental results. Section 5 discusses relevant issues and relates our algorithm to previous work. The last section draws conclusions. Due to the limited space, Appendices A-D are left out of main text but can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2010.92.

2 Margin Cost Functional Framework and Semi-Supervised Boosting

In the section, we briefly review the generic margin cost functional framework for abstract boosting, including AdaBoost [16], and its application to semi-supervised boosting, e.g., ASSEMBLE [2], to make them self-contained. Later on, we shall develop our regularized boosting algorithm within this framework and employ AdaBoost and ASSEMBLE for comparison.

2.1 Margin Cost Functional Framework for Boosting

The generic form of an ensemble learner constructed by boosting is the voted combination of base learners, $\text{sign}[F(\mathbf{x})]$, where $F(\mathbf{x})$ is the linear combination of base learners as follows:

$$F(\mathbf{x}) = \sum_t w_t f_t(\mathbf{x}). \tag{1}$$

For binary classification, $f_t : X \to \{+1, -1\}$ are base classifiers and $w_t \in \mathbb{R}$ are weights for linear combination.

Given a training set of $|L|$ labeled examples, $L = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}, y_{|L|})\}$, generated according to a distribution, boosting finds out $F(\mathbf{x})$ so that $P(F(\mathbf{x}) \neq y)$ on this distribution is minimized. In reality, the distribution is unknown and a training set L is available only. Thus, boosting would find $F(\mathbf{x})$ by minimizing a margin cost functional defined on the training set L:

$$\mathcal{C}(F) = \frac{1}{|L|}\sum_{i \in L} C\big[y_i F(\mathbf{x}_i)\big], \tag{2}$$

where $C : \mathbb{R} \to \mathbb{R}$ is a nonnegative and monotonically decreasing cost function. In (2), $y_i F(\mathbf{x}_i)$ is the *margin* of an example, $i \in L$, with respect to $F(\mathbf{x})$.

At an abstract level, the base learners $f \in \mathcal{F}$ and their combinations F are viewed as elements of an inner produce space $(\mathcal{X}, <, >)$, where $\mathcal{X}$ is a linear space of functions containing $\text{lin}(\mathcal{F})$, the set of all linear combination of functions in $\mathcal{F}$. As a result, boosting at this abstract level is interpreted as finding a new $f \in \mathcal{F}$ to add it to $F \in \text{lin}(\mathcal{F})$ so that the cost functional $\mathcal{C}(F + \epsilon f)$ decreases for some small value of ϵ. Based on the Taylor expansion approximation on $\mathcal{C}(F + \epsilon f)$ to the first order, it would be desirable to choose $f = -\nabla\mathcal{C}(F)$, which always causes $\mathcal{C}(F + \epsilon f)$ to decrease most rapidly. Given the fact that $\nabla\mathcal{C}(F)$ may not be in $\mathcal{F}$ and f is restricted in $\mathcal{F}$, it is unrealistic to choose $f = -\nabla\mathcal{C}(F)$ in general. Instead, f can be sought with the greatest inner product with $-\nabla\mathcal{C}(F)$, i.e., f should be chosen to maximize $-<\nabla\mathcal{C}(F), f>$.

In order to maximize $-<\nabla\mathcal{C}(F), f>$, we need to approximate the functional derivative of $\mathcal{C}(F)$ in (2) for $i = 1, \ldots, |L|$ as follows:

$$\nabla\mathcal{C}(F)(\mathbf{x}) = \begin{cases} \dfrac{y_i C'\big[y_i F(\mathbf{x}_i)\big]}{|L|}, & \text{if } \mathbf{x} = \mathbf{x}_i, \\ 0, & \text{otherwise}, \end{cases} \tag{3}$$

where $C'(z)$ is the derivative of the margin cost function with respect to z. Therefore, the use of (3) in the inner product leads to

$$-<\nabla\mathcal{C}(F), f> = -\frac{1}{|L|^2}\sum_{i \in L} y_i f(\mathbf{x}_i) C'\big[y_i F(\mathbf{x}_i)\big]. \tag{4}$$

As a cost function $C(z)$ is required to be monotonically decreasing, the term $C'[y_i F(\mathbf{x}_i)]$ will always be negative. Normalizing $C'[y_i F(\mathbf{x}_i)]$ in (4), we see that finding a function f to maximize $-<\nabla\mathcal{C}(F), f>$ is equivalent to finding an f to minimize

$$-\sum_{i \in L} y_i f(\mathbf{x}_i) \frac{C'\big[y_i F(\mathbf{x}_i)\big]}{\sum_{k \in L} C'\big[y_k F(\mathbf{x}_k)\big]}. \tag{5}$$

For $i \in L$, we define their empirical distribution as $D(i) = C'[y_i F(\mathbf{x}_i)] / \sum_{k \in L} C'[y_k F(\mathbf{x}_k)]$. Then, (5) can be rewritten as $\sum_{i: f(\mathbf{x}_i) \neq y_i} D(i) - \sum_{i: f(\mathbf{x}_i) = y_i} D(i) = 2\sum_{i: f(\mathbf{x}_i) \neq y_i} D(i) - 1$ since $y_i f(\mathbf{x}_i) = +1$ if $f(\mathbf{x}_i) = y_i$ and -1 otherwise for binary classification and $\sum_{i \in L} D(i) = 1$. As a consequence, finding f to maximize $-<\nabla\mathcal{C}(F), f>$ can be done by finding f to minimize the weighted error $\sum_{i: f(\mathbf{x}_i) \neq y_i} D(i)$, which results in a generic boosting procedure.

Once f is determined with the procedure described above, the weight w for combination is chosen so that $\mathcal{C}(F + wf)$ defined in (2) is minimized or decreases as much as possible.

2.2 Semi-Supervised Boosting Learning

In SSL setting, a training set $S = L \cup U$ of $|L|$ labeled examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{|L|}, y_{|L|})\}$ in L and $|U|$ unlabeled points $\{\mathbf{x}_{|L|+1}, \ldots, \mathbf{x}_{|L|+|U|}\}$ in U is given. Since there exists no label information for unlabeled points, the critical idea behind semi-supervised boosting algorithms like ASSEMBLE [2] is introducing a pseudoclass [14] or a pseudomargin [2] concept to unlabeled points within the margin cost functional framework [27]. The pseudoclass label of an unlabeled point $\mathbf{x}$ is typically defined as $\tilde{y} = \text{sign}[F(\mathbf{x})]$ and its corresponding pseudomargin is $\tilde{y}F(\mathbf{x}) = |F(\mathbf{x})|$ [14], [2].

Within the generic margin cost functional framework [27], the semi-supervised boosting learning is to find F such that the cost of functional

$$\mathcal{C}(F) = \frac{1}{|S|}\sum_{i \in S}\{I_{i,L}\alpha_i C[y_i F(\mathbf{x}_i)] + I_{i,U}\alpha_i C[|F(\mathbf{x}_i)|]\} \tag{6}$$

is minimized for some nonnegative and monotonically decreasing cost function $C : \mathbb{R} \to \mathbb{R}$. Here, we define $I_{z,\Psi} = 1$ if $z \in \Psi$ and 0 otherwise. $\alpha_i \in \mathbb{R}^+$ in (6) are used to highlight some training data if the prior knowledge is available or differentiate between labeled and unlabeled data if labeled examples are treated more importantly than unlabeled points. Note that $\mathcal{C}(F)$ in (6) is no longer convex due to the pseudomargin cost $C[|F(\mathbf{x}_i)|]$, and hence, the use of the greedy yet stagewise learning strategy does not guarantee to find the global optimum.

As reviewed in Section 2.1, constructing an ensemble learner needs to choose a base learner, $f(\mathbf{x})$, to maximize the inner product $-\langle\nabla\mathcal{C}(F), f\rangle$. For unlabeled points $\mathbf{x}_{|L|+1}, \ldots, \mathbf{x}_{|U|}$, a subgradient of $\mathcal{C}(F)$ in (6) has been introduced in [2] to tackle its nondifferentiable problem as follows: $\nabla\mathcal{C}(F)(\mathbf{x}) = \alpha_i \tilde{y}_i C'[\tilde{y}_i F(\mathbf{x}_i)]/|S|$ if $\mathbf{x} = \mathbf{x}_i, \mathbf{x}_i \in U$ and 0 otherwise. Thus, unlabeled points of pseudoclass labels can be treated in the same way as labeled examples in the optimization problem. As a result, finding a proper $f(\mathbf{x})$ amounts to maximizing

$$\begin{aligned} -\langle\nabla\mathcal{C}(F), f\rangle = \frac{1}{|S|^2}\Bigg\{ &\sum_{i: f(\mathbf{x}_i) \neq y_i} I_{i,L}\alpha_i C'[y_i F(\mathbf{x}_i)] \\ &- \sum_{i: f(\mathbf{x}_i) = y_i} I_{i,L}\alpha_i C'[y_i F(\mathbf{x}_i)] \\ &+ \sum_{i: f(\mathbf{x}_i) \neq \tilde{y}_i} I_{i,U}\alpha_i C'[|F(\mathbf{x}_i)|] \\ &- \sum_{i: f(\mathbf{x}_i) = \tilde{y}_i} I_{i,U}\alpha_i C'[|F(\mathbf{x}_i)|] \Bigg\}. \end{aligned} \tag{7}$$

With the same treatment described in (4), finding $f(\mathbf{x})$ to maximize $-\langle \nabla \mathcal{C}(F), f \rangle$ is equivalent to searching for $f(\mathbf{x})$ to minimize

$$\sum_{i:f(\mathbf{x}_i)\neq\hat{y}_i} \hat{D}(i) - \sum_{i:f(\mathbf{x}_i)=\hat{y}_i} \hat{D}(i) = 2 \underbrace{\sum_{i:f(\mathbf{x}_i)\neq\hat{y}_i} \hat{D}(i)}_{\textbf{misclassification errors}} -1. \quad (8)$$

Here, $\hat{y}$ is a collective notation of the true label and the pseudoclass label and defined as $\hat{y}_i = y_i$ if $I_{i,L} = 1$ and $\hat{y}_i = \tilde{y}_i$ if $I_{i,U} = 1$. For $1 \leq i \leq |L| + |U|$, $\hat{D}(i)$ is the empirical data distribution defined as

$$\hat{D}(i) = \frac{I_{i,L}\alpha_i C'[\hat{y}_i F(\mathbf{x}_i)] + I_{i,U}\alpha_i C'[|F(\mathbf{x}_i)|]}{\sum_{k\in S}\{I_{k,L}\alpha_k C'[\hat{y}_k F(\mathbf{x}_k)] + I_{k,U}\alpha_k C'[|F(\mathbf{x}_k)|]\}},$$

and $\sum_{i\in S} \hat{D}(i) = 1$. From (8), $f(\mathbf{x})$ can be found by minimizing weighted errors $\sum_{i:f(\mathbf{x}_i)\neq\hat{y}_i} \hat{D}(i)$. Similarly, the weight w for combination is chosen so that $\mathcal{C}(F + wf)$ defined in (6) is minimized or decreases as much as possible. Thus, any boosting algorithms specified for SL [27] are now applicable to SSL with the aforementioned treatment for unlabeled points [2].

3 Regularized Semi-Supervised Boost

In this section, we first present a novel cost functional for our regularized semi-supervised boosting learning and then develop a solution to the optimization problem arising from this cost functional. Finally, we describe a derived boosting algorithm for SSL.

3.1 Regularized Margin Cost Functional

Given a training set $S = L \cup U$ of $|L|$ labeled examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{|L|}, y_{|L|})\}$ in L and $|U|$ unlabeled points $\{\mathbf{x}_{|L|+1}, \ldots, \mathbf{x}_{|L|+|U|}\}$ in U, we propose a cost functional for our regularized semi-supervised boosting learning as follows:

$$\mathcal{C}(F) = \sum_{i\in S}\left\{\frac{1}{|L|} I_{i,L}\alpha_i C\big[\hat{y}_i F(\mathbf{x}_i)\big] + \frac{1}{|U|} I_{i,U}\beta_i |N(i)|^{-1} \sum_{j\in N(i)} \omega_{ij} C\big[\hat{y}_j F(\mathbf{x}_i)\big]\right\}. \quad (9)$$

Here, $C : \mathbb{R} \rightarrow \mathbb{R}$ is a nonnegative and monotonically decreasing cost function and $\alpha_i \in \mathbb{R}^+$ are parameters to weight labeled examples based on prior knowledge and/or emphasize the importance of labeled examples. $\beta_i \in \mathbb{R}^+$ are parameters for unlabeled points in U:

$$\beta_i = \lambda\big[p(\mathbf{x}_i)\big], \quad (10)$$

where $p(\mathbf{x}_i)$ is the density of point $\mathbf{x}_i$ and $\lambda : \mathbb{R} \rightarrow \mathbb{R}$ is a nonnegative and monotonically increasing function. ω_{ij} is the affinity measure for any two points i and j in the input space:

$$\omega_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right), \quad (11)$$

where σ is the bandwidth parameter controlling the spread of this function. $N(i)$ is a neighborhood of size $|N(i)|$ for unlabeled point $i \in U$ without including itself. $I_{z,\Psi}$ is the same as defined in Section 2.2. Equation (9) defines a regularized margin cost functional that reflects not only the cost incurred by misclassification errors on labeled examples, but also the inconsistency among labeled and unlabeled data caused by violating the fundamental SSAs [9]. We have several remarks on our cost functional in (9) in terms of the fundamental SSAs [9] and the cost margin functional framework for boosting [27].

Remark 1. Based on the low-density separation assumption, β_i in (10) tend to ensure that points of higher density, more likely in the same cluster, play a more important role for regularization so that inconsistence in such a region would be penalized more severely. In other words, points of low density are likely to be located between cluster boundaries and therefore need less regularization.

Remark 2. Based on the semi-supervised smoothness assumption, a point should have the same label as that of its neighbors in $N(i)$. Thus, the local smoothness is measured by $\hat{y}_j F(\mathbf{x}_i)$, $j \in N(i)$, where $\hat{y}_j$ is the true label y_j if point $j \in N(i)$ is a labeled example and the pseudoclass label $\tilde{y}_j$ otherwise. $F(\mathbf{x}_i)$ is the output of the current ensemble learner for unlabeled point i. Apparently, the compatibility between unlabeled point i and its neighbor $j \in N(i)$ is high only if $F(\mathbf{x}_i)$ has the same sign as $\hat{y}_j$. Applying a monotonically decreasing function to the compatibility, i.e., $C[\hat{y}_j F(\mathbf{x}_i)]$, would take an effect of penalizing an unlabeled point of low label compatibility with its neighbors severely.

Remark 3. Based on the manifold assumption, the similarity between two points should be measured by an appropriate distance reflecting the manifold structure underlying the input space. Motivated by the graph-based regularization to measure pairwise similarities [41], [39], we employ the Gaussian kernel to define the affinity ω_{ij} in (11) so that the strength of a label incompatibility penalty would also be determined by the affinity of unlabeled point i to point j in $N(i)$, i.e., as the incompatibility between two points is fixed, the closer they are the severer the penalty is. Whenever we have prior knowledge on the intrinsic manifold structure of a specific input space, we would use its geodesic distance derived from the manifold structure to define affinity ω_{ij} instead of its current definition in (11).

Remark 4. Within the cost margin functional framework for boosting, the regularization penalty term $|N(i)|^{-1}\sum_{j\in N(i)} \omega_{ij} C[\hat{y}_j F(\mathbf{x}_i)]$ can be viewed as a novel approximation of the margin cost for unlabeled point i via the use of all the labels/pseudolabels of its neighbors in $N(i)$ and its current ensemble estimate $F(\mathbf{x}_i)$. Note that this margin cost approximation for unlabeled points readily distinguishes from the pseudomargin cost $C[|F(\mathbf{x}_i)|]$ in ASSEMBLE [2], which merely uses its own information of unlabeled point i without taking its neighbors into account, as reviewed in Section 2.2.

3.2 Optimization of Our Cost Functional

In order to construct an ensemble learner by boosting, we need to find a base learner $f_t(\mathbf{x})$ and a combination weight w_t to minimize the regularized margin cost functional in (9) at each boosting round. Within the margin cost functional framework briefly reviewed in Section 2.1, this optimization problem can be converted into maximizing $-<\nabla \mathcal{C}(F), f>$.

Since the regularization term has been introduced in the margin cost functional in (9), maximization of $-<\nabla\mathcal{C}(F), f>$ will no longer be straightforward as described in Sections 2.1 and 2.2. Thus, we develop a solution to this optimization problem, as described in Proposition 1

Proposition 1. *For the regularized margin cost functional defined in (9), finding $f(\mathbf{x})$ to maximize $-<\nabla\mathcal{C}(F), f>$ is equivalent to minimizing*

$$\underbrace{\sum_{i:f(\mathbf{x}_i)\neq\hat{y}_i}\hat{D}(i)}_{\textbf{misclassification errors}} + \underbrace{\sum_{i:f(\mathbf{x}_i)=\hat{y}_i}\left\{-\frac{I_{i,U}R(i)}{Z|U|}\right\}}_{\textbf{class label inconsistency}}, \tag{12}$$

where

$$R(i) = \beta_i|N(i)|^{-1}\left|\sum_{j\in N(i)}\omega_{ij}\hat{y}_jC'[\hat{y}_jF(\mathbf{x}_i)]\right|, \tag{13a}$$

$$R_U(i) = \beta_i|N(i)|^{-1}\sum_{j\in N(i)}\omega_{ij}C'\big[\hat{y}_jF(\mathbf{x}_i)\big], \tag{13b}$$

$$\hat{D}(i) = \frac{\frac{1}{|L|}I_{i,L}\alpha_iC'[\hat{y}_iF(\mathbf{x}_i)] + \frac{1}{|U|}I_{i,U}R_U(i)}{Z}, \tag{13c}$$

$$Z = \sum_{k\in S}\left\{\frac{1}{|L|}I_{k,L}\alpha_kC'[\hat{y}_kF(\mathbf{x}_k)] + \frac{1}{|U|}I_{k,U}R_U(k)\right\}. \tag{13d}$$

The proof of Proposition 1 is in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2010.92. Intuitively, finding a proper $f(\mathbf{x})$ needs to minimize not only the weighted classification errors for labeled examples and unlabeled points, but also the class-label inconsistency caused by the violation of SSAs for unlabeled points, even though their pseudoclass labels are consistent with the output of $f(\mathbf{x})$.

$\hat{D}(i)$ in (13c) is the empirical data distribution used for sampling training examples during boosting learning, and Z in (13d) is the normalization term such that $\sum_{i\in S}\hat{D}(i) = 1$. Note that Z is always negative since $C'(\cdot)$ is always negative, as $C(\cdot)$ is a monotonically decreasing function, which guarantees $\hat{D}(i)$ and the second term expressing class-label inconsistency in (12) are always positive.

$R(i)$ in (13a) is the actual penalty awarded to unlabeled point i via regularization. In the context of binary classification, $\hat{y}_j \in \{-1, +1\}$, we rewrite the regularizer in (13a) as follows:

$$R(i) = \beta_i|N(i)|^{-1}\left|\sum_{j\in N(i)}\{\delta(\hat{y}_j, +1)\omega_{ij}C'[F(\mathbf{x}_i)] - \delta(\hat{y}_j, -1)\omega_{ij}C'[-F(\mathbf{x}_i)]\}\right|, \tag{14}$$

where $\delta(u, v) = 1$ if $u = v$ and 0 otherwise. By ignoring β_i in (14), we observe that $R(i)$ tends to be small only if the pseudoclass label of point i is consistent with labels/pseudoclass labels of its neighbors, i.e., $\text{sign}[F(\mathbf{x}_i)] = \hat{y}_j$, $\forall j \in N(i)$. In other words, $R(i)$ tends to be large as point i has a noisy or inhomogeneous neighborhood. By taking β_i defined in (10) into account, $R(i)$ will be scaled up by β_i if point i lies in a high data density region, and vice versa. Thus, $R(i)$ forms a density-dependent class-label inconsistency measure for unlabeled points. Furthermore, the regularizer $R(i)$ also suggests a reliable unlabeled point labeling method based on the class-label inconsistency during our boosting learning. In (14), two terms $\sum_{j\in N(i)}\delta(\hat{y}_j, +1)\omega_{ij}C'[F(\mathbf{x}_i)]$ and $\sum_{j\in N(i)}\delta(\hat{y}_j, -1)\omega_{ij}C'[-F(\mathbf{x}_i)]$ correspond to the penalty when unlabeled point i is predicted to be ∓1 but the label/pseudoclass labels of its neighbors are ±1, respectively. Therefore, the affinity-based penalty competition leads to a rule for labeling unlabeled point i: For $i \in U$, $\tilde{y}_i = \pm1$ if $\sum_{j\in N(i)}\delta(\hat{y}_j, +1)\omega_{ij}C'[F(\mathbf{x}_i)] \lessgtr \sum_{j\in N(i)}\delta(\hat{y}_j, -1)\omega_{ij}C'[-F(\mathbf{x}_i)]$. Note that $C'(\cdot)$ is always negative since $C(\cdot)$ is a monotonically decreasing function. Based on (13a), the rule for labeling unlabeled points during our boosting learning is rewritten as

$$\tilde{y}_i = \text{sign}\left[-\sum_{j\in N(i)}\omega_{ij}\hat{y}_jC'[\hat{y}_jF(\mathbf{x}_i)]\right].$$

After a proper function f is found, a combination weight w needs to be chosen by minimizing the regularized cost functional $\mathcal{C}(F + wf)$ in order to construct optimal F. In general, w is a step size for linear search and needs to be chosen based on a specific cost function [27]. It is possible for some cost functions to find a closed-form solution to the line search with an optimal step size, while there is no closed-form solution to many cost functions. Thus, a proper w should be chosen according to the cost function $C(z)$.

Within the margin cost functional framework [27], a boosting algorithm terminates when $-<\nabla\mathcal{C}(F), f> \leq 0$, due to the fact that the function f no longer found points in the downhill direction of the margin cost functional $\mathcal{C}(F)$. In other words, the boosting algorithm should terminate as no f can be found to reduce $\mathcal{C}(F)$. For our cost functional defined in (9), the termination condition of its derived boosting algorithms is described in Proposition 2.

Proposition 2. *Semi-supervised boosting algorithms with the regularized margin cost functional in (9) terminate when*

$$\sum_{i:f(\mathbf{x}_i)\neq\hat{y}_i}\hat{D}(i) + \sum_{i:f(\mathbf{x}_i)=\hat{y}_i}\left\{-\frac{I_{i,U}R(i)}{Z|U|}\right\} > \frac{1}{2}, \tag{15}$$

where $\hat{D}(i)$, Z, and $R(i)$ are defined in (13).

The proof of Proposition 2 is in Appendix B, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI. 2010.92.

In summary, we achieve a generic solution to the optimization problem arising from the regularized margin cost functional in (9). A derived boosting algorithm from the cost functional needs to choose a proper cost function used in (9) and addresses other details. We will present a derived boosting algorithm in Section 3.3 for exemplification and employ it in our simulations.

3.3 Algorithm Description

Based on the generic solution obtained in Section 3.2, we develop a boosting algorithm by addressing all technical details required. We first present an initialization setting including initial unlabeled data labeling, training the first classifier and a nonlinear function to generate β_i from the density estimation. Then, we describe a regularized semi-supervised boosting algorithm for binary classification and exemplify this algorithm with the exponential cost function that has been widely used in AdaBoost [16], ASSEMBLE [2], and many other boosting algorithms.

3.3.1 Initialization Setting

For a semi-supervised boosting algorithm, it is necessary to label unlabeled points during initialization. Prior to boosting learning, there is no ensemble learner, and hence, $F(\mathbf{x}_i)=0$ for $i \in S$. As a result, the initial value of our regularizer in (14) is rewritten as

$$R_0(i) \propto \left| \sum_{j \in N(i)} \{\delta(\hat{y}_j, +1)\omega_{ij} - \delta(\hat{y}_j, -1)\omega_{ij}\} \right|. \quad (16)$$

In this situation, we need to take only labeled points in $N(i)$ into account to label unlabeled point i. Thus, a pseudoclass label should be assigned to unlabeled point i with the rule derived from (16) by following the winner-take-all principle described in Section 3.2.

For SSL, an implicit assumption is the labeled example sparsity; i.e., there are few labeled examples but many unlabeled points in a training set. Hence, it is highly likely that many unlabeled points have no labeled point in their neighborhoods. In (16), moreover, β_i defined in (10) suggests that we should consider the input data distribution during labeling, i.e., labeling unlabeled point i in a cluster tends to be more reliable than within the neighborhood of unlabeled point i especially as unlabeled point i is located in a low data density region. As a result, any density-based clustering algorithms, e.g., [38], can be employed to group training data into clusters, and then, the affinity-based competition would take place within each cluster other than their neighborhoods for labeling unlabeled points. To a great extent, doing so also remedies the labeled example sparsity problem since all unlabeled points in a cluster can be labeled as long as there is one labeled point in the cluster.

For a given training set $S = L \cup U$, we first employ a density-based clustering algorithm to partition S into K_S clusters: $c_1, \ldots, c_{K_S}$. For unlabeled point $i \in c_k, k \in \{1, \ldots, K_S\}$, its initial pseudoclass label is assigned with

$$\tilde{y}_i = \mathrm{argmax}_{y \in Y} \left\{ \sum_{j \in c_k} I_{j,L} \delta(y_j, y) \omega_{ij} \right\}, \quad (17)$$

where $Y = \{+1, -1\}$ for binary classification and $Y = \{1, 2, \ldots, M\}$ for multiclass classification of M classes. In general, there might be some clusters where no labeled example is available. In this circumstance, we stipulate that the initial pseudoclass label of unlabeled point i in such a cluster is $\tilde{y}_i = 0$ to indicate that no label has been assigned to unlabeled point i.

The competition-based initial unlabeled data labeling setting also provides a metric to measure the confidence of a pseudoclass-label assignment as follows:

$$B(i) = \max_{y \in Y} \left\{ \sum_{j \in c_k} I_{j,L} \delta(y_j, y) \omega_{ij} \right\} - \max_{y \in Y, y \neq \tilde{y}_i} \left\{ \sum_{j \in c_k} I_{j,L} \delta(y_j, y) \omega_{ij} \right\}, \quad (18)$$

where $\tilde{y}_i$ is the initial pseudoclass label of unlabeled point $i \in c_k, k \in \{1, \ldots, K_S\}$, assigned with (17). Also, we stipulate that $B(i) = 0$ if $\tilde{y}_i = 0$. Once the initial unlabeled data labeling is completed, we train a chosen base learner on all of the labeled examples in L and unlabeled points of the high confidence $B(i)$ in the ρth percentile of those unlabeled points of $B(i) > 0$ in U to obtain the first classifier. We anticipate that doing so would considerably lessen the adverse effect brought about by incorrectly initial unlabeled point labeling.

During the initialization, we need to estimate the density for computing β_i in (10) for a given training set $S = L \cup U$. In our experiments, we employ the kernel density estimation [15] for density estimation. For a given training set S, the probability density function is defined as

$$p(\mathbf{x}) = \frac{1}{|S| h^n} \sum_{i=1}^{|S|} K_h\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right),$$

where $K_h : \mathbb{R}^n \rightarrow \mathbb{R}$ is a positive kernel, n is the dimension of input space, and h is the bandwidth. After obtaining $p(\mathbf{x})$, we employ the following nonlinear function to compute β_i for unlabeled point i:

$$\beta_i = \sin\left(\frac{\pi}{2} \left[\bar{p}(\mathbf{x}_i)\right]^{\kappa}\right), \quad (19)$$

where $\kappa \in \mathbb{Z}^+$ is used to control the steepness of this nonlinear curve and $\bar{p}(\mathbf{x}_i)$ is the normalized version of $p(\mathbf{x}_i)$ in the following form:

$$\bar{p}(\mathbf{x}_i) = \frac{p(\mathbf{x}_i) - p_{min}}{p_{max} - p_{min}},$$

where p_{max} and p_{min} are the maximum and the minimum of density values across the whole training set S. In general, this nonlinear function allows regularization to be exerted severely on those unlabeled points in a region of high density but lessens the regularization effect on unlabeled points located in a low data density region via a proper choice of κ.

3.3.2 Algorithm

Based on the solution developed in Section 3.2 and the initialization setting in Section 3.3.1, we describe our regularized semi-supervised boosting algorithm for binary classification as follows:

1) Initialization

  1.1 Input training set $S = L \cup U$. For $i \in L$, set α_i and choose a cost function, $C(\cdot)$, used in (9). Set $F_0(\mathbf{x}) = 0$ and a maximum boosting round, T_{max}.

  1.2 Estimate the density function on S. For $i \in U$, choose κ and compute β_i with (19) and set a neighborhood $N(i)$ via either K *nearest neighbor* (NN) or ϵNN method for unlabeled point i.

  1.3 Choose σ and calculate the affinity, ω_{ij}, among all training data in S with (11). Fulfill the clustering analysis on S with a density-based clustering

```
        algorithm and assign the pseudoclass label $\tilde{y}_i$ for
        unlabeled point $i \in U$ with (17) to form $Y_0^U$, where
        $Y_t^U = \{\tilde{y}_i | i \in U\}$ is a collective notation of
        pseudoclass labels of unlabeled points at round $t$.
    1.4 Calculate the confidence, $B(i)$, for $i \in U$ with (18)
        and construct the initial training subset $S_0 = \{L, U'\}$
        based on the confidence of unlabeled points, where
        $U' = \{i | i \in U, B(i) \text{ in the } \rho\text{th percentile for } B(i) > 0\}$.
        Choose a learning algorithm $\mathcal{L}(\cdot,\cdot)$.
2) Repeat steps 2.1-2.4 for $1 \le t \le T_{max}$
    2.1 $f_t(\mathbf{x}) = \mathcal{L}(S_{t-1}, Y_{t-1}^U)$ to obtain a new base learner.
    2.2 Examine the termination condition with (15). If the
        condition is met, stop training and return the
        ensemble learner, $F_{t-1}(\mathbf{x})$, with the decision rule:
        $y = \mathrm{sign}[F_{t-1}(\mathbf{x})]$. Otherwise, choose a proper
        step-size $w_t$ according to $C(\cdot)$ for constructing the
        ensemble learner $F_t(\mathbf{x}) = F_{t-1}(\mathbf{x}) + w_t f_t(\mathbf{x})$.
    2.3 Reset $\tilde{y}_i = \mathrm{sign}[-\sum_{j \in N(i)} \omega_{ij} \hat{y}_j C'[\hat{y}_j F_t(\mathbf{x}_i)]]$ for
        unlabeled point $i \in U$ to form $Y_t^U$.
    2.4 Update $\hat{D}_t(i)$ with (13b) and (13c) for $i \in S$ and then
        obtain a new training set $S_t$ by resampling from $S$
        according to $\hat{D}_t(i)$; i.e., $S_t = \mathrm{Sample}(S, \hat{D}_t)$.
3) Return the ensemble learner, $F_{T_{max}}$, with the decision
   rule: $y = \mathrm{sign}[F_{T_{max}}(\mathbf{x})]$.
```

Note that this algorithm can be easily extended to multiclass classification with the one-against-rest scheme, as described in Appendix D, which is in the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2010.92.

3.3.3 Exemplification with $C(z) = e^{-z}$

The cost function $C(z) = e^{-z}$ was used in AnyBoost [27] to derive a boosting algorithm equivalent to AdaBoost [16] for SL (see also Section 2.1), ASSEMBLE [2] for SSL (see also Section 2.2), and many other boosting algorithms. Now we exemplify our algorithm described in Section 3.3.2 with this specific cost function. Later on, we shall use this derived algorithm in our experiments for comparison with AdaBoost [16] and ASSEMBLE [2] reviewed in Section 2.

Inserting the cost function $C(z) = e^{-z}$ into (9) leads to a specific regularized margin cost functional:

$$\mathcal{C}(F) = \sum_{i \in S} \left\{ \frac{1}{|L|} I_{i,L} \alpha_i e^{-\hat{y}_i F(\mathbf{x}_i)} + \frac{1}{|U|} I_{i,U} \beta_i |N(i)|^{-1} \sum_{j \in N(i)} \omega_{ij} e^{-\hat{y}_j F(\mathbf{x}_i)} \right\}.$$

Accordingly, the main components of our algorithm in (13) become

$$R_t(i) = \beta_i |N(i)|^{-1} \left| \sum_{j \in N(i)} \omega_{ij} \hat{y}_j e^{-\hat{y}_j F_t(\mathbf{x}_i)} \right|, \tag{20a}$$

$$R_{U,t}(i) = \beta_i |N(i)|^{-1} \sum_{j \in N(i)} \omega_{i,j} e^{-\hat{y}_j F_t(\mathbf{x}_i)}, \tag{20b}$$

$$\hat{D}_t(i) = -\frac{\frac{1}{|L|} I_{i,L} \alpha_i e^{-\hat{y}_i F_t(\mathbf{x}_i)} + \frac{1}{|U|} I_{i,U} R_{U,t}(i)}{Z_t}, \tag{20c}$$

$$Z_t = -\sum_{k \in S} \left\{ \frac{1}{|L|} I_{k,L} \alpha_k e^{-\hat{y}_k F_t(\mathbf{x}_k)} + \frac{1}{|U|} I_{k,U} R_{U,t}(k) \right\}, \tag{20d}$$

where $\hat{y}_i = y_i$ if $i \in L$ and $\tilde{y}_i \in Y_{t-1}^U$ if $i \in U$. In the algorithm described in Section 3.3.2, the termination condition in Step 2.2 is achieved by inserting (20) into (15). With this cost function, $C'[-\hat{y}_j F_t(\mathbf{x}_i)]$ in Step 2.3 and (13) in Step 2.4 are now instantiated with $-e^{-\hat{y}_j F_t(\mathbf{x}_i)}$ and (20), respectively. For the exponential cost function, there is an optimal step size w_t used in Step 2.2, as described in Proposition 3.

Proposition 3. *For the cost function $C(z) = e^{-z}$ used in (9), the regularized margin cost functional $\mathcal{C}(F_{t-1} + w_t f_t)$ is minimized by choosing the step size*

$$w_t = \frac{1}{2} \ln \left(\frac{\frac{1}{|L|} \sum_{i: f_t(\mathbf{x}_i) = \hat{y}_i} I_{i,L} \alpha_i e^{-\hat{y}_i F_{t-1}(\mathbf{x}_i)} + P(i)}{\frac{1}{|L|} \sum_{i: f_t(\mathbf{x}_i) \neq \hat{y}_i} I_{i,L} \alpha_i e^{-\hat{y}_i F_{t-1}(\mathbf{x}_i)} + Q(i)} \right), \tag{21}$$

where

$$P(i) = \frac{1}{|U|} \sum_{i \in S} \left\{ I_{i,U} \beta_i |N(i)|^{-1} \delta[f_t(\mathbf{x}_i), -1] \sum_{i \in N(i)} \delta(\hat{y}_j, -1) \omega_{ij} e^{F_{t-1}(\mathbf{x}_i)} + I_{i,U} \beta_i |N(i)|^{-1} \delta[f_t(\mathbf{x}_i), +1] \sum_{i \in N(i)} \delta(\hat{y}_j, +1) \omega_{ij} e^{-F_{t-1}(\mathbf{x}_i)} \right\},$$

$$Q(i) = \frac{1}{|U|} \sum_{i \in S} \left\{ I_{i,U} \beta_i |N(i)|^{-1} \delta[f_t(\mathbf{x}_i), -1] \sum_{i \in N(i)} \delta(\hat{y}_j, +1) \omega_{ij} e^{-F_{t-1}(\mathbf{x}_i)} + I_{i,U} \beta_i |N(i)|^{-1} \delta[f_t(\mathbf{x}_i), +1] \sum_{i \in N(i)} \delta(\hat{y}_j, -1) \omega_{ij} e^{F_{t-1}(\mathbf{x}_i)} \right\}.$$

The proof of Proposition 3 is in Appendix C, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI. 2010.92.

4 Experiments

In this section, we first use a synthetic data set to demonstrate the role that different SSAs play in terms of our cost functional. Then we thoroughly evaluate our boosting algorithm, hereinafter named as *RegBoost*, with a variety of SSL tasks, including SSL-book [9], UCI [35], and facial expression recognition [26] benchmarks. By using all of the aforementioned SSL tasks, we compare RegBoost with ASSEMBLE [2], a winning algorithm of the NIPS 2001 unlabeled data competition [2] and the third-party-independent assessment across various SSL techniques and problem domains [11], and AdaBoost [16] trained on only labeled examples as a baseline. Furthermore, we compare RegBoost with two state-of-the-art semi-supervised boosting algorithms [24], [25] on SSL-book and UCI benchmarks, respectively.

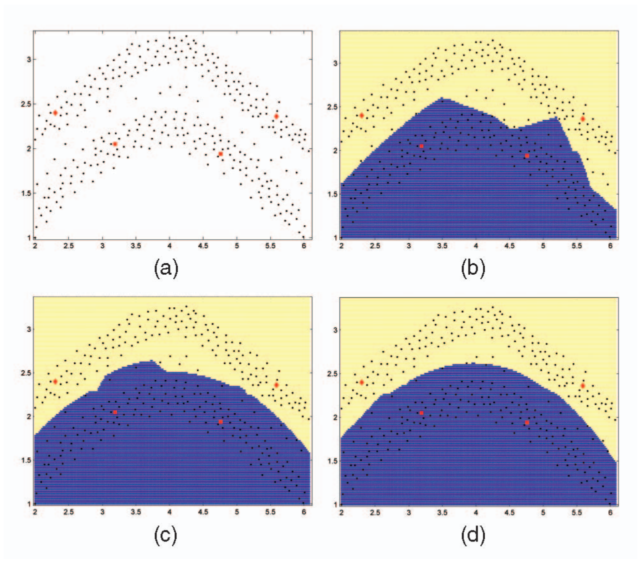


Fig. 1. Synthetic data set. (a) Training data set of four labeled examples. (b) The decision boundary produced by RegBoost-S. (c) The decision boundary produced by RegBoost-SM. (d) The decision boundary produced by RegBoost.

4.1 Synthetic Data Set

In order to facilitate the understanding how an SSL algorithm works on SSAs, we design a synthetic data set of two classes, named *noisy two half-moons*, where three fundamental SSAs [9] hold. The training data set shown in Fig. 1a consists of four labeled examples marked with ◆ and ■ and 322 unlabeled points. As depicted in Fig. 1a, this data set contains two half-moon-like manifold structures separated by a low data density region. Although the low data density region is noisy, the semi-supervised smoothness assumption holds within high data density clusters corresponding to two classes. With this data set, we can closely observe the behavior of an SSL algorithm working on SSAs.

In order to observe how our algorithm works on different SSAs, we modify our cost functional in (9) by first considering only the smoothness assumption, and then taking other SSAs into account. By setting $\omega_{ij}=1$ and $\beta_i=1/|U|$ in (9), we switch off manifold and low-density separation assumptions. To facilitate our presentation, the boosting algorithm derived from the modified cost functional is named *RegBoost-S*. Next, we keep $\beta_i=1/|U|$ but apply ω_{ij} in (11) to (9), which results in another boosting algorithm working on smoothness and manifold assumptions, named *RegBoost-SM*. As described in Section 3, RegBoost works on three SSAs. As a result, the usefulness of individual SSAs and their combinations would be exhibited via behaviors of three boosting algorithms. In our experiment, we employ an SVM with the RBF kernel to be the base learner and use the same initialization and parameters as discussed in Section 5 for three algorithms. Results after 20 boosting iterations are reported as more iterations do not change decision boundaries.

Figs. 1b, 1c, and 1d depict decision boundaries established by three boosting algorithms. In Fig. 1b, we observe that RegBoost-S results in an inaccurate zigzag decision boundary that traverses through the top of the lower half-moon cluster. In contrast, RegBoost-SM and RegBoost successfully identify manifold structures so that their decision boundaries separate two clusters well. As shown in Fig. 1c, RegBoost-SM produces a rough decision boundary corrupted by noise. In contrast, RegBoost yields a smoothing decision boundary that traverses right through the middle of the low data density region between two half-moon clusters, as illustrated in Fig. 1d, since less regularization is exerted on those noisy points due to the low-density separation assumption. Apparently, RegBoost is superior to RegBoost-SM in terms of generalization. The experimental results shown in Figs. 1b, 1c, and 1d vividly demonstrate the role that different SSAs play in our cost functional.

TABLE 1
Test Errors (in Percent) with 10 Labeled Training Points

Algorithm	g241c	g241d	Digit1	USPS	COIL	BCI	Text
1NN	44.05	43.22	23.47	19.82	65.91	48.74	39.44
SVM	47.32	46.66	30.60	20.03	68.36	49.85	45.37
MVU+1NN	48.68	47.28	11.92	14.88	65.72	50.24	39.40
LEM+1NN	47.47	45.34	12.04	19.14	67.96	49.94	40.84
QC+CMN	39.96	46.55	9.80	13.61	59.63	50.36	40.79
Discrete Reg.	49.59	49.05	12.64	16.07	63.38	49.51	40.37
TSVM	24.71	50.08	17.77	25.20	67.50	49.15	31.21
SGT	22.76	18.64	8.92	25.36	n/a	49.59	29.02
Cluster-Kernel	48.28	42.05	18.73	19.41	67.32	48.31	42.72
Data-Rep. Reg.	41.25	45.89	12.49	17.96	63.65	50.21	n/a
LDS	28.85	50.63	15.63	15.57	61.90	49.27	27.15
Laplacian RLS	43.95	45.68	5.44	18.99	54.54	48.97	33.68
CHM (normed)	39.03	43.01	14.86	20.53	n/a	46.90	n/a
AdaBoost	40.12	43.05	28.92	25.57	71.16	47.08	47.42
ASSEMBLE	40.62	44.41	23.49	21.77	65.49	48.96	49.13
ManifoldBoost	42.17	42.80	19.42	29.97	n/a	47.17	n/a
RegBoost	38.22	42.90	17.94	17.41	65.39	46.73	34.96

4.2 Experiments on Transductive Learning

TL is a special kind of SSL where the test set coincides with the set of unlabeled data used for training [37]. For TL, several benchmarks have been elaborately designed and already used for evaluating many SSL algorithms (see [9, Chapter 21] for details). We apply our algorithm, along with AdaBoost and ASSEMBLE, to seven benchmark tasks [9] and compare them with 14 state-of-the-art SSL algorithms [9] and ManifoldBoost [24].

The benchmarks used in our experiments are three artificial data sets, `g241c`, `g241d`, and `Digit1`, and four real data sets, `USPS` (imbalanced), `COIL`, `BCI`, and `Text` (sparse discrete). All data sets contain 1,500 points, except for `BCI` of 400 points, and the input data dimension is 241, apart from `BCI` and `Text` where their input data dimensions are 114 and 11,960, respectively. In addition, all benchmarks are for binary classification except for `COIL`, a six-class classification task. In our experiments, we strictly follow the instructions given by designers and use the exactly same experimental setup [9]. That is, we conduct experiments on 12 subsets of each data set with the number of 10 or 100 labeled points stipulated in the benchmark. As suggested in [9], we report the best mean test error on 12 subsets of each benchmark achieved by three boosting algorithms with three base learners, i.e., three nearest neighbor (3NN) classifier with the euclidean distance, a three-layered MLP of 10 hidden neurons, and an SVM with the linear kernel. In our experiments, we always fix parameters α_i, σ, κ, ρ, and T_{max}, but tune the remaining parameters in RegBoost for different data sets, as discussed in Section 5.

Tables 1 and 2 tabulate experimental results achieved by AdaBoost, ASSEMBLE, and RegBoost along with those of ManifoldBoost reported in [24] in comparison to those of 14 SSL algorithms [9]. The details of ManifoldBoost and

TABLE 2
Test Errors (in Percent) with 100 Labeled Training Points

Algorithm	g241c	g241d	Digit1	USPS	COIL	BCI	Text
1NN	40.28	37.49	6.12	7.64	23.27	44.83	30.77
SVM	23.11	24.64	5.53	9.75	22.93	34.31	26.45
MVU+1NN	44.05	43.21	3.99	6.09	32.27	47.42	30.74
LEM+1NN	42.14	39.43	2.52	6.09	36.49	48.64	30.92
QC+CMN	22.05	28.20	3.15	6.36	10.03	46.22	25.71
Discrete Reg.	43.65	41.65	2.77	4.68	9.61	47.67	24.00
TSVM	18.46	22.42	6.15	9.77	25.80	33.25	24.52
SGT	17.41	9.11	2.61	6.80	n/a	45.03	23.09
Cluster-Kernel	13.49	4.95	3.79	9.68	21.99	35.17	24.38
Data-Rep. Reg.	20.31	32.82	2.44	5.10	11.46	47.47	n/a
LDS	18.04	28.74	3.46	4.96	13.72	43.97	23.15
Laplacian RLS	24.36	26.46	2.92	4.68	11.92	31.36	23.57
CHM (normed)	24.82	25.67	3.79	7.65	n/a	36.03	n/a
AdaBoost	24.82	26.97	9.09	9.68	22.96	24.02	26.31
ASSEMBLE	27.19	27.42	6.71	8.12	21.84	28.75	27.77
ManifoldBoost	22.87	25.00	4.29	6.65	n/a	32.17	n/a
RegBoost	20.54	23.56	4.58	6.31	21.78	23.69	23.25

other SSL algorithms are omitted here due to the limited space but can be found from [24] and [9, Chapter 21].

From Tables 1 and 2, it is evident that RegBoost outperforms ASSEMBLE and AdaBoost on all seven benchmarks, regardless of the number of labeled examples used. It is observed that RegBoost is significantly better than ASSEMBLE on six binary classification tasks, while their performance is similar on `COIL`, a multiclass classification task. In comparison with the baseline system, the use of unlabeled points in RegBoost always leads to improvement, while it is not true for ASSEMBLE, as its performance is inferior to that of AdaBoost on four benchmarks. ManifoldBoost, working on the manifold assumption [24], was applied to five out of seven benchmarks. In comparison to ASSEMBLE and AdaBoost, ManifoldBoost generally performs better on those benchmarks whenever the manifold assumption holds and sufficient labeled points are used for training. It is observed from Tables 1 and 2 that RegBoost generally outperforms ManifoldBoost for all five data sets, although its performance is marginally worse than that of ManifoldBoost on `g241d` with 10 labeled examples for training and on `Digit1` with 100 labeled examples for training. According to [9], `g241d` was generated so that the cluster assumption holds but the cluster structure is misleading; the correct decision boundary traverses through high data density regions, which violates the low-density separation assumption. As 10 labeled examples are used for training, labeled examples provide too little information to detect the violation of the SSA, which causes RegBoost to yield a less satisfactory result. As labeled points are increased to 100, RegBoost copes with the misleading situation well, as shown in Table 2, since the sufficient supervision information can cancel out the adverse effect made by the violation of an SSA. On the other hand, `Digit1` was designed to have points close to a low-dimensional manifold embedded into a high-dimensional space but not to show a cluster structure [9]. In comparison to ASSEMBLE and AdaBoost, ManifoldBoost and RegBoost perform well on `Digit1` due to SSAs. Results on `g241d` and `Digit1` manifest that the use of hybrid SSAs in RegBoost leads to the favorite performance even though some SSAs do not hold for a data set. All of the above comparison results for different boosting algorithms suggest that the exploitation of unlabeled data with regularization working on SSAs plays a crucial role in improving boosting algorithms for SSL.

TABLE 3
UCI Binary Classification Benchmarks

Data Set	Size	Attribute
Australian (AUS)	690	14
BUPA (BUPA)	345	6
German Credit (GC)	1000	24
Harber Man's Survival (HMS)	306	3
Heart Disease Cleveland (HDC)	303	13
Hepatitis (HEP)	155	19
Horse Colic (HC)	368	27
Ionosphere (ION)	351	34
Kr-vs-Kp (KVK)	3196	36
Mammographic Mass (MM)	961	5
Pima Indians Diabetes (PID)	768	8
Vote (VOTE)	435	16
WDBC (WDBC)	569	30

In comparison to various SSL algorithms in [9], RegBoost yields satisfactory performance in general. Although RegBoost is developed for SSIL, it is comparable with those established SSL algorithms on all benchmarks except `Digit1`. In particular, RegBoost yields the best performance among 17 algorithms on the `BCI` benchmark, as shown in Tables 1 and 2. Since `BCI` is collected from a real-world task [9], the intrinsic structures underlying the data set are generally unknown, although it seems plausible that the signals recorded by an EEG have rather few degrees of freedom. Results achieved by RegBoost suggest that this data set may be of both manifold-like and cluster-like structures since the regularization working on hybrid SSAs yields the best performance regardless of experimental setup. Given the fact that no algorithm dominates the performance on all the benchmarks, we conclude that RegBoost would be highly competitive with the existing SSL algorithms for some specific tasks.

4.3 Experiments on Inductive Learning

The UCI machine learning repository [35], originally designed for assessing SL algorithms, has been extensively used for assessing SSL algorithms. As a result, we adopt UCI benchmarks to evaluate the generalization performance of RegBoost in comparison to AdaBoost [16], ASSEMBLE [2], and SemiBoost [25]. We first describe our experimental setting, and then report results achieved by four boosting algorithms on 13 UCI binary classification tasks under the same conditions. Due to the limited space, we report experimental results on the other 13 UCI multiclass classification tasks separately in Appendix D, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2010.92.

4.3.1 Experimental Setting

There are various benchmark data sets in the UCI machine learning repository [35]. As a result, we select a number of data sets to reflect the nature of various data sets, e.g., attribute types (numerical versus cardinal and complete versus missing attributes), the number of classes (binary versus multiclass), and balanced versus unbalanced data points in different classes. For binary classification, we employ 13 UCI data sets for performance evaluation. Table 3 tabulates the information on 13 UCI data sets, including the number of data points and attributes.

To produce training and test data sets, we first divide each UCI data set randomly into two subsets: S and T with the ratio 4:1 for training and test. Then, we further partition

TABLE 4
Test Errors (Mean±Std) Percent on the UCI Binary Classification Data Sets ($\mathrm{LDR} = 5\%$)

Data	3 Nearest Neighbors				Naïve Bayes				C4.5 Decision Tree			
Set	ADAB	ASMB	SEMIB	REGB	ADAB	ASMB	SEMIB	REGB	ADAB	ASMB	SEMIB	REGB
AUS	42.7±4.0	39.9±4.0	*36.8±4.9*	38.9±4.0	24.0±7.6	23.6±3.5	22.6±4.4	*21.4±5.5*	19.9±6.4	24.6±10.7	24.9±6.7	**17.8±5.3**
BUPA	47.0±6.8	45.2±9.3	45.5±7.1	*43.5±7.9*	46.4±7.9	51.0±6.0	44.6±7.6	*42.2±5.7*	41.9±7.2	46.7±7.3	45.5±7.6	**39.1±6.3**
GC	*37.4±3.7*	37.1±2.7	*31.0±4.9*	33.0±3.1	27.7±3.0	27.0±4.5	25.8±3.3	**24.9±3.5**	28.3±2.8	32.3±4.0	29.2±3.6	*27.1±2.7*
HMS	39.2±12.1	36.6±8.0	*31.3±6.4*	33.8±5.8	40.0±17.9	34.3±12.1	33.0±10.0	*31.6±5.9*	38.5±9.1	34.9±8.6	**30.4±4.6**	32.3±7.7
HDC	52.5±4.2	46.2±11.1	49.2±9.3	*44.3±11.5*	26.5±8.1	24.7±8.7	32.8±7.9	*24.2±8.2*	24.5±8.8	30.0±8.4	43.7±8.0	**24.0±7.1**
HEP	37.1±10.3	36.5±11.2	33.6±3.5	*30.0±7.3*	21.6±8.2	22.9±11.2	**19.4±5.9**	19.7±6.5	30.0±13.4	29.7±12.0	21.3±8.4	*20.9±8.2*
HC	33.3±13.4	32.2±8.1	34.3±6.1	*29.2±7.3*	29.6±11.1	31.5±12.3	30.3±13.7	*26.9±8.7*	27.8±12.4	31.8±11.4	31.5±7.8	**25.9±9.5**
ION	29.6±10.1	27.6±7.3	35.0±4.1	*26.9±8.7*	21.0±9.1	26.1±7.2	21.7±4.9	**18.0±7.0**	23.7±8.2	27.3±9.2	23.4±6.6	*21.0±9.2*
KVK	32.5±3.1	35.8±2.7	35.7±2.9	*32.4±2.1*	*19.6±2.4*	22.4±2.1	20.2±2.7	*19.6±1.7*	17.9±2.1	19.8±2.8	19.1±2.2	**17.6±1.2**
MM	28.4±4.3	27.0±2.6	26.4±4.7	*26.2±3.3*	19.8±3.6	24.4±5.5	20.7±3.0	**19.7±3.6**	24.8±3.0	22.2±2.9	24.0±3.2	*20.0±2.8*
PID	36.1±4.2	35.6±6.0	35.4±3.6	*33.9±3.5*	32.5±4.5	29.9±4.6	33.4±3.3	**29.5±3.4**	32.0±4.1	34.1±4.8	31.7±4.0	*29.9±4.2*
VOTE	12.9±4.6	12.1±3.9	13.6±5.7	*11.4±4.0*	13.2±3.4	13.3±4.3	14.6±4.3	*12.6±4.1*	8.2±4.3	9.4±4.8	11.7±7.4	**7.9±4.4**
WDBC	12.9±7.9	12.3±7.0	19.9±11.3	*11.6±7.0*	8.1±4.4	6.9±2.3	8.5±2.4	**6.8±3.3**	10.8±4.5	11.8±4.7	9.3±2.9	*8.8±4.5*

the training set S randomly into two subsets: L and U with a *labeled data rate* (LDR), $|L|/|S|$, which determines how many labeled examples would be used in the training set. In the training set S, the labels of all the examples in U are removed to generate unlabeled points, while labeled examples in L retain the information on their attributes and labels. We conduct experiments at the LDR of 5, 10, and 20 percent, respectively, for all 13 UCI binary classification benchmarks. For the robust performance evaluation, we conduct 10 trials, i.e., the use of 10 different training and test subset combinations for each benchmark, and use the test error, misclassification rate on the test set T, to assess the generalization performance of boosting algorithms.

In a boosting algorithm, we employ three typical weak learners, i.e., 3NN classifier with euclidean distance, naive Bayes (NB) classifier, and C4.5 decision tree (DT) as a base learner, respectively, although other base learners, e.g., SVM or neural networks, may achieve better performance according to our empirical studies not reported here. Thus, no parameter tuning is needed for three base learners once C4.5 DTs are restricted to two levels in depth for all the benchmarks. Finally, a boosting algorithm is terminated by meeting its stopping condition before T_{max} rounds or running $T_{max} = 50$ rounds given the fact that running more rounds does not considerably alter the performance in general. Parameters used in AdaBoost and ASSEMBLE are set as suggested in [16], [2].

For parameters in SemiBoost [25], our experiments reveal that the default parameters suggested in their paper seldom lead to satisfactory performance for the selected UCI benchmarks, although their default setting might be appropriate to their experiments where they always use 10 labeled examples for training. For fair comparison, we search for the best value of three essential parameters in a broad range. The default sampling parameter was set to 10 percent unlabeled points of top confidence at each boosting iteration [25]. But, we find that the default sampling rate often results in the poor performance, and hence, exhaustedly search for the range between 1 and 15 percent to find out the most appropriate sampling rate for each benchmark. Similarly, their empirical studies suggested that the scale parameter used to determine the similarity matrix was quite stable and the performance was insensitive to the choice of this parameter [25]. However, our empirical studies show that the choice of this parameter may affect the performance. As a result, we seek the best value for this parameter by looking at the similarity values from the *10th* to *90th* percentiles, varied in steps of 10, as suggested in [25]. Finally, the default stopping criterion of SemiBoost was set to $T_{max} = 20$ boosting rounds [25]. Nevertheless, we find that more boosting iterations may lead their algorithm to a better result. Thus, we always run SemiBoost for $T_{max} = 50$ boosting rounds if their stopping condition is not met and also record its performance after 20 rounds for each data set. Then, we report the better result achieved by either 20 or 50 rounds.

For RegBoost, we fix most of the parameters in our experiments as discussed in Section 5, although further fine tuning might yield better performance. Here, we describe only the setup of two parameters tuned for different UCI data sets. In our experiments, we use the KNN method to define the neighborhood of each unlabeled point, i.e., $|N(i)| = K$. In our experiments, K is chosen to be 4 or 5. A number of points between 6 and 25 percent of $|S|$ in proportion to LDRs are randomly chosen by sampling with $\hat{D}(i)$, defined in (20), at each boosting iteration for training a base learner.

4.3.2 Experimental Results

Corresponding to different LDRs, Tables 4, 5, and 6 show the generalization performance of boosting algorithms with three base learners on the 13 UCI data sets listed in Table 3, respectively. To facilitate the presentation, we abbreviate AdaBoost, ASSEMBLE, SemiBoost, and RegBoost to ADAB, ASMB, SEMIB, and REGB in Tables 4, 5, and 6. Below, we present results in terms of individual base learners, and then discuss results across different base learners.

Table 4 shows the generalization performance of four boosting algorithms as a training set is generated with $\mathrm{LDR} = 5\%$. As 3NN is used as the base learner, RegBoost improves the AdaBoost baseline performance on all 13 data sets, while ASSEMBLE and SemiBoost improve the baseline performance on 12 and 8 data sets, respectively. Moreover, RegBoost outperforms ASSEMBLE on all 13 data sets, while it yields the better performance than SemiBoost on 10 data sets. We use the italic font to mark the best performance of four boosting algorithms using the same base learner. It is observed from Table 4 that RegBoost wins on 10 out of 13 data sets and SemiBoost yields the lowest test errors on `AUS`, `GC`, and `HMS` with 3NN. As NB is used, RegBoost

TABLE 5
Test Errors (Mean±Std) Percent on the UCI Binary Classification Data Sets ($\mathrm{LDR} = 10\%$)

Data	3 Nearest Neighbors				Naïve Bayes				C4.5 Decision Tree			
Set	ADAB	ASMB	SEMIB	REGB	ADAB	ASMB	SEMIB	REGB	ADAB	ASMB	SEMIB	REGB
AUS	40.7±2.4	39.4±4.2	*36.2±4.6*	38.5±2.6	22.8±4.1	23.5±4.6	21.6±3.3	*19.6±4.5*	18.8±2.7	22.7±5.8	22.3±6.3	**17.0±3.5**
BUPA	46.8±5.6	42.5±5.3	43.3±6.1	*41.5±5.5*	45.9±7.7	50.1±5.4	43.8±7.4	**38.8±6.8**	40.7±5.4	44.9±7.1	45.1±6.6	*39.0±4.6*
GC	36.7±3.1	35.1±3.7	*30.5±3.9*	33.0±4.7	26.7±3.7	26.0±3.1	25.3±4.0	**24.8±3.3**	28.0±3.0	28.9±3.0	28.7±3.8	*26.9±3.8*
HMS	36.9±4.1	35.3±9.0	*31.0±5.6*	33.3±4.4	36.4±4.6	33.3±7.8	30.8±5.7	**30.0±4.4**	38.0±6.3	34.6±6.9	*30.2±4.7*	31.2±5.2
HDC	44.2±11.5	45.5±8.4	44.3±10.5	*43.0±9.7*	25.2±6.5	24.5±5.4	30.5±7.1	**19.8±3.9**	23.5±6.1	29.5±5.9	36.3±7.2	*22.7±7.3*
HEP	34.2±8.5	33.6±9.0	29.7±6.9	29.0±4.6	20.0±6.8	20.0±5.2	**19.0±4.9**	19.3±5.3	22.6±11.8	28.4±14.1	20.4±4.6	*20.0±5.8*
HC	32.1±10.7	29.2±9.0	34.0±6.2	*27.4±6.1*	26.4±7.1	26.7±5.8	27.3±7.6	*25.9±6.3*	21.1±5.8	25.8±9.8	24.5±5.2	**19.5±5.4**
ION	27.7±9.8	26.6±9.5	29.9±8.3	*23.6±8.7*	16.1±8.6	22.1±5.4	20.0±4.8	**13.1±6.2**	16.3±4.3	17.6±6.9	18.9±6.9	*16.1±6.5*
KVK	27.5±2.1	29.8±2.2	29.6±1.9	*26.8±1.5*	17.6±1.4	18.9±2.4	19.6±2.8	*17.4±1.6*	**15.9±1.4**	18.4±2.2	18.1±2.5	**15.9±1.4**
MM	28.3±4.5	26.5±3.4	*25.6±3.5*	26.1±4.1	19.7±3.8	20.5±3.8	19.9±3.7	**19.5±3.2**	24.2±2.9	21.0±4.9	21.1±2.5	*19.7±3.6*
PID	36.0±4.1	35.4±4.3	33.9±3.5	*31.9±4.4*	30.3±3.2	29.7±5.8	31.4±2.9	**26.5±2.4**	28.7±3.2	31.1±5.0	30.6±4.3	*27.7±3.6*
VOTE	11.5±2.1	11.4±2.9	13.1±5.3	*10.3±2.8*	9.4±4.6	11.0±4.6	11.4±3.9	*9.0±3.9*	6.9±2.4	8.5±3.1	8.1±2.2	**6.3±2.3**
WDBC	12.7±5.3	11.6±3.1	14.3±2.9	*11.1±3.9*	6.1±2.3	6.8±2.1	6.7±3.1	**5.8±1.8**	8.1±2.2	11.3±5.7	8.6±2.4	*7.0±1.3*

improves the baseline performance on 12 data sets and yields nearly the same error rate as the baseline performance on ION, while ASSEMBLE and SemiBoost outperform AdaBoost on seven and five data sets only, respectively. Again, RegBoost outperforms ASSEMBLE on all 13 data sets and SemiBoost on all data sets except HEP. As marked by either the italic or the bold font that stands for the best performance regardless of base learners, RegBoost and SemiBoost win on 11 data sets and HEP only, respectively, while none of the three semi-supervised boosting algorithms improves the baseline performance for KVK. By using the C4.5 DT, RegBoost improves the baseline performance on all 13 data sets, while ASSEMBLE and SemiBoost yield lower test errors than AdaBoost on only three and six data sets, respectively. By comparison, RegBoost outperforms ASSEMBLE and SemiBoost on 13 and 12 data sets, respectively. With the C4.5 DT, RegBoost wins on 12 data sets, while SemiBoost wins on HMS, as marked with the italic or the bold font. As highlighted in Table 4 with the bold font, RegBoost wins on 11 data sets overall, while SemiBoost wins on HMS and HEP, regardless of base learners used in experiments.

Table 5 shows the generalization performance of four boosting algorithms as a training set is generated with $\mathrm{LDR} = 10\%$. With 3NN, RegBoost improves the baseline performance on all 13 data sets, while ASSEMBLE and SemiBoost yield lower test error rates than AdaBoost on 11 and 7 data sets, respectively. RegBoost outperforms ASSEMBLE and SemiBoost on 13 and 10 data sets, respectively. By comparing four algorithms with 3NN, RegBoost wins on nine data sets and SemiBoost yields the lowest test errors on AUS, GC, HMS, and MM. As NB is used, RegBoost yields lower error rates than AdaBoost on all 13 data sets, while ASSEMBLE and SemiBoost improve the baseline performance on only six and five data sets, respectively. Moreover, RegBoost outperforms ASSEMBLE and SemiBoost on 13 and 12 data sets, respectively. By comparing four algorithms with NB, RegBoost wins on 12 data sets and SemiBoost wins on HEP. With the C4.5 DT, RegBoost improves the baseline performance on 12 data sets and yields the same performance on KVK, while ASSEMBLE and SemiBoost outperform AdaBoost on only two and three data sets, respectively. Moreover RegBoost outperforms ASSEMBLE and SemiBoost on 13 and 12 data sets, respectively. With the C4.5 DT, RegBoost wins on 12 data sets, while SemiBoost wins on HMS. As boldfaced in Table 5, RegBoost wins on 12 data sets overall, while SemiBoost wins on HEP only, regardless of base learners.

Table 6 shows the generalization performance of four boosting algorithms as a training set is generated with $\mathrm{LDR} = 20\%$. As 3NN is used, RegBoost outperforms AdaBoost on all 13 data sets, while ASSEMBLE and SemiBoost yield lower test errors than AdaBoost on 10 and eight data sets, respectively. Furthermore, RegBoost

TABLE 6
Test Errors (Mean±Std) Percent on the UCI Binary Classification Data Sets ($\mathrm{LDR} = 20\%$)

Data	3 Nearest Neighbors				Naïve Bayes				C4.5 Decision Tree			
Set	ADAB	ASMB	SEMIB	REGB	ADAB	ASMB	SEMIB	REGB	ADAB	ASMB	SEMIB	REGB
AUS	39.6±5.3	38.8±4.1	*35.1±3.8*	38.0±4.1	20.4±3.2	23.4±6.0	21.3±2.6	*18.3±3.0*	16.9±3.7	21.7±4.3	17.8±5.1	**15.5±3.4**
BUPA	45.9±6.7	42.0±7.4	41.7±4.5	*40.9±5.9*	44.2±9.4	50.0±4.1	43.4±7.7	*38.4±6.5*	38.7±6.5	39.1±4.1	44.3±4.5	**38.3±5.4**
GC	35.7±2.3	33.1±4.4	*29.8±3.1*	32.4±3.6	25.9±3.4	25.4±4.1	25.1±3.6	**24.4±3.3**	27.4±3.4	29.1±3.7	28.0±2.7	*26.8±3.4*
HMS	34.1±4.8	31.8±7.5	30.8±6.1	*30.7±4.4*	29.5±7.8	27.9±5.2	29.3±4.4	**27.4±6.4**	35.6±3.8	32.3±8.3	*29.8±4.6*	31.0±8.0
HDC	41.7±5.3	43.8±8.0	42.7±6.9	*40.0±4.4*	23.3±5.9	20.7±3.3	26.0±7.3	**19.5±3.5**	22.3±4.0	24.5±7.3	28.9±6.7	*21.0±6.9*
HEP	31.6±6.4	29.4±9.6	*24.2±5.1*	28.7±6.2	19.4±4.8	19.7±8.4	**18.4±6.6**	19.0±8.7	20.7±10.1	26.1±7.3	20.0±5.9	*19.7±6.9*
HC	30.7±4.5	28.0±7.7	27.5±6.6	*27.3±5.7*	26.2±6.7	28.5±5.4	*24.9±4.4*	25.8±4.4	20.7±9.3	22.9±11.7	19.0±5.5	**18.5±4.8**
ION	23.1±6.9	21.6±9.5	25.4±7.4	*19.9±6.8*	10.0±4.8	18.0±6.0	17.7±3.8	**9.9±4.7**	*12.0±4.6*	15.1±5.4	17.7±6.9	14.3±4.8
KVK	22.0±1.1	26.2±1.9	26.2±1.6	*21.9±1.4*	16.1±1.2	15.9±1.4	16.0±1.2	*15.8±1.4*	12.8±0.9	15.5±1.0	14.8±1.2	**12.2±0.8**
MM	26.7±5.2	26.4±2.6	*22.4±3.3*	25.9±3.2	19.5±3.5	20.2±3.3	19.7±3.6	*19.4±3.5*	23.5±3.5	20.6±3.1	**19.3±1.3**	19.5±3.2
PID	33.7±3.8	33.3±4.9	*29.7±4.1*	30.4±4.1	27.4±2.2	26.0±4.1	27.8±3.0	**25.8±4.4**	27.5±4.0	28.5±3.1	27.5±3.9	*27.1±4.1*
VOTE	11.2±4.1	10.7±3.8	12.9±4.1	*10.2±3.8*	*6.4±1.6*	8.7±2.6	10.1±2.8	7.4±2.0	6.2±1.1	8.2±4.0	6.6±2.7	**6.1±2.2**
WDBC	10.0±2.7	10.5±3.1	10.4±3.6	*9.9±2.2*	*5.0±2.9*	5.8±2.7	5.7±2.1	5.6±2.9	4.8±1.9	8.9±1.7	7.9±2.9	**4.7±1.8**

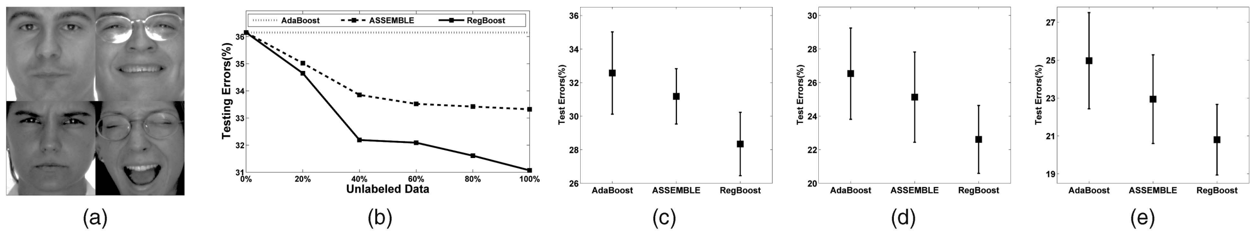


Fig. 2. Facial expression recognition on the AR face database. (a) Exemplar pictures corresponding to four facial expressions. (b) Evolution of averaging test errors as different numbers of unlabeled instances are used at $\mathrm{LDR} = 20\%$. (c), (d), and (e) Test errors at $\mathrm{LDR} = 30$, 40, and 50%, respectively.

yields better performance than ASSEMBLE and SemiBoost on 13 and eight data sets, respectively. With 3NN, RegBoost wins on eight data sets and SemiBoost wins on the remaining five data sets, as italicized in Table 6. By using NB, RegBoost improves the baseline performance on 11 but fails on two unbalanced data sets, VOTE and WDBC. In contrast, ASSEMBLE and SemiBoost outperform AdaBoost on only five data sets. With NB, RegBoost yields better performance than ASSEMBLE and SemiBoost on 13 and 11 data sets, respectively. Using NB, RegBoost and SemiBoost win on nine and two data sets, respectively, while AdaBoost wins VOTE and WDBC. As C4.5 DT is used, RegBoost improves the baseline performance on 12 data sets but fails on ION, while ASSEMBLE is inferior to the baseline on all data sets except HMS, and SemiBoost outperforms AdaBoost on only five data sets. With C4.5 DT, RegBoost wins on 11 data sets, while AdaBoost wins on ION, and SemiBoost wins on HMS. Regardless of base learners, overall, SemiBoost wins on HEP and MM and RegBoost wins on the remaining 11 data sets, as boldfaced in Table 6.

In summary, the use of NB or C4.5 DT as a base learner generally leads to better performance than 3NN for four boosting algorithms, regardless of LDRs. In contrast to the baseline performance, RegBoost with unlabeled points constantly makes improvements for all 13 data sets, regardless of LDRs, while ASSEMBLE and SemiBoost yield lower test errors on only a few data sets at different LDRs. Moreover, RegBoost outperforms ASSEMBLE on all 13 data sets at every LDR with different base learners, while SemiBoost performs better than ASSEMBLE on only some data sets at different LDRs. In addition, greater improvement is achieved by RegBoost on most of the data sets as fewer labeled examples are available for training or the LDR is smaller, as shown in Table 4. For a large LDR, there are already sufficient labeled examples to train AdaBoost for some data sets, e.g., KVK. Even in this circumstance, RegBoost with unlabeled points still improves the baseline performance, while ASSEMBLE and SemiBoost fail on the same condition, as shown in Table 6. Overall, RegBoost wins on 11 data sets at every LDR used in our experiments and holds AdaBoost to a performance draw on KVK at $\mathrm{LDR} = 10\%$, while SemiBoost wins on two, one, and two data sets at $\mathrm{LDR} = 5$, 10, and 20%, respectively, as boldfaced in Tables 4, 5, and 6. Based on experimental results on UCI benchmarks, including those on multiclass classification tasks reported in Appendix D, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2010.92, we conclude that the exploitation of multiple SSAs within the boosting framework paves a promising way for SSL.

4.4 Facial Expression Recognition

Automatic facial expression recognition is a task to use facial images of an individual to identify his/her emotional state at the moment when those pictures were taken. While a huge number of facial images are easily available, the annotation of a facial image with a proper emotional state is often extremely difficult and time-consuming since individuals express their emotion in rather diversified ways and facial expression is also affected by a number of factors, e.g., culture, habits, and so on. Thus, facial expression recognition becomes a typical SSL task and we would further assess our RegBoost with such a real-world multiclass classification task.

We evaluate RegBoost with the one-against-rest multiclass classification strategy by a facial expression benchmark, the AR face database [26], where there are 56 female and 70 male expressers who posed for two examples in different sessions for each of four facial expressions, *neutral*, *smile*, *anger*, and *scream*, as exemplified in Fig. 2a. In total, 1,008 pictures of 768×576 pixels were collected with different illumination conditions, background, and occlusions, e.g., glasses and scarf [26]. In our experiments, we apply the Gabor filter of three scales and eight orientations and PCA to each image for feature extraction. As a result, we use the top 100 PCA coefficients of a filtered image to form its feature vector.

For SSL simulations, we first randomly pick 20 percent of the images (balanced to four classes) as test data and the rest of the images constitute a training set (S) that is further randomly split into labeled data (L) and unlabeled data (U) subsets with different LDRs ranging from 20 to 50 percent. At $\mathrm{LDR} = 20\%$, we investigate the role of unlabeled data in detail by using a portion of U each time. At other LDRs, we report the performance with all unlabeled points in U only. We use a three-layered MLP of 80 hidden neurons as the base learner and set 70 boosting iterations to stop the algorithms if their termination conditions are not met. Parameter setting in RegBoost is discussed in Section 5. For reliability, 10 trials are conducted for each experiment at every LDR.

Fig. 2b shows the evolution of averaging test errors on 10 trials at $\mathrm{LDR} = 20\%$ as unlabeled points increase from 20 to 100 percent of $|U|$ against the baseline performance of AdaBoost trained on only labeled examples in L. It is observed that increasing unlabeled points for training leads to lower test errors in general. In particular, RegBoost always outperforms ASSEMBLE and yields greater improvement as more unlabeled points are used. From Figs. 2c, 2d, and 2e, it is evident that both ASSEMBLE and RegBoost improve the baseline performance, and RegBoost outperforms ASSEMBLE at all of the LDRs.

AdaBoost with more sophisticated feature extraction and selection techniques was previously applied to AR for SL, where a subset of 960 images was used by excluding some abnormal ones [32]. The five-fold cross validation was used to evaluate the performance of AdaBoost with different multiclass classification strategies [32], i.e., at each trial, 80 percent of labeled images in the AR subset of 960 images were used for training and the remaining 20 percent of images were employed for test. AdaBoost with the one-against-rest and the exhaustive strategies achieved 22.53 and 20.73 percent averaging error rates [32], respectively. In contrast, RegBoost trained on 40 percent labeled and 40 percent unlabeled AR images yields a test error rate of 20.80 percent, as shown in Fig. 2e. Our results suggest that the proper use of unlabeled data in SSL algorithms is promising for facial expression recognition.

5 Discussions

In this section, we discuss issues concerning our approach and relate it to the previous work and the latest developments in the context of regularization and boosting for SSL.

As elucidated in Section 3.1, the regularization term exerted on unlabeled points in (9) is based on three SSAs [9] and also on a novel margin cost approximation for unlabeled points within the margin cost functional framework [27], especially for semi-supervised classification. This margin cost approximation uses the pseudoclass labels of those unlabeled points in the neighborhood of unlabeled point i, defined as $\mathrm{sign}[F(\mathbf{x}_j)]$ and $j \in N(i)$, which causes the regularized margin cost functional defined in (9) to be nonconvex. Thus, the gradient-based local search procedure presented in Section 3.2 does not guarantee to find the global optimum. Nevertheless, boosting is a relaxation process that approximates the original target function F with a linear combination of functions. We need to find a function f from a given functional subspace and a step size w at each iteration to minimize a cost functional $\mathcal{C}(F+wf)$. At each iteration, the local F is fixed, and hence, $\mathcal{C}(F+wf)$ in (9) is convex with respect to f and w as long as the cost function $C(\cdot)$ is convex. To avoid getting stuck in an unwanted local optimum, we propose an initialization method based on clustering analysis in Section 3.3.1 to generate the first classifier for our RegBoost learning. Our empirical studies manifest that the initialization method works well, and hence, ensure that RegBoost yields the satisfactory performance. In our ongoing research, we shall be investigating alternative optimization techniques, e.g., those described in [10], to tackle the local optimum problem.

In our RegBoost, the regularization works on the low density separation assumption along with other SSAs. However, such an assumption does not always hold. The violation of this assumption may lead to two different situations, i.e., there is no cluster structure underlying a data set or there is a cluster structure but such structural information is misleading. The former situation does not significantly affect the performance of RegBoost without turning off the low-density separation assumption, as shown in our experiments on the UCI data sets of high Bayes errors that implicitly suggest no underlying cluster structures, e.g., BUPA [35]. However, the latter situation may degrade the performance of RegBoost without sufficient labeled examples used for training. As demonstrated on g241d, an SSL-book benchmark [9], RegBoost fails to yield better performance than those not working on the low-density separation assumption, e.g., ManifoldBoost [24], since the data set is designed to have a misleading cluster structure. To tackle this problem, cross-validation techniques may be applied to detect such a situation in general. Once the violation situation is identified, we can switch off the low-density separation assumption by setting all β_i equal to $1/|U|$ uniformly.

As the density of input data is explicitly employed to control regularization via β_i, $i \in U$, in (9), the accuracy of density estimation is also another factor to affect the performance of RegBoost. For the sake of computational efficiency, we employ a density-based clustering algorithm that fulfills density estimation for regularization and clustering analysis for the initialization together. In general, none of the existing clustering algorithms are perfect; a clustering algorithm may make a mistake by either overdividing an intrinsic cluster into two or more clusters or merging two or more natural clusters into one single cluster. If such a mistake occurs, an uneven labeled data distribution may degrade the performance of RegBoost. For instance, it could be problematic if there are only labeled examples of the same class available for a single cluster produced by wrongly merging two intrinsic clusters corresponding to two different classes. Nevertheless, our empirical studies indicate that as labeled examples are distributed properly, our cluster-based initialization works well even in the presence of incorrect clustering analysis.

There are several parameters in RegBoost. Our empirical studies reveal that the performance of RegBoost is insensitive to the choice of most parameters, while some of them need to be tuned for a given data set. First of all, α_i in (9) are always fixed to $\min(|U|/|L|, 5)$, and σ in (11) is set to the standard deviation of all distances between training examples. For all data sets used in our experiments, including those not reported here, the performance of RegBoost is insensitive to κ used for calculating β_i in (10) when κ is chosen within the range between four and six. As a result, κ is set to five for all experiments reported in this paper. Next, we find that satisfactory performance is achieved by setting ρ used in our initialization to a value in the range between 10 and 15 percent. Although the neighborhood of an unlabeled point i, $N(i)$, can be defined by either ϵNN specified with a distance threshold ϵ or KNN specified with the number of its nearest neighbors K, our empirical studies indicate that the performance of RegBoost is more sensitive to choice of ϵ than K. Therefore, we suggest the use of KNN to define the neighborhood and K is chosen in the range between 3 and 10 in proportion to the number of all training examples. Finally, $(\mathrm{LDR} + \tau)|S|$ training examples, where $0.01 \leq \tau \leq 0.1$ and $|S|$ is the number of all the training examples, are randomly chosen by sampling with $\hat{D}(i)$ defined in (20) to train a base learner at each boosting iteration.

In comparison with the existing regularization techniques used in SSL, our RegBoost is closely related to graph-based methods, e.g., [39]. In general, a graph-based method wants to find a function to satisfy two conditions simultaneously [40]: 1) It should be close to given labels on the labeled nodes and 2) it should be smooth on the whole graph. In particular, the work in [39] developed a regularization framework to carry out the above idea by defining the global and local consistency terms in their cost function. Likewise, our cost function for RegBoost in (12) has two terms explicitly corresponding to the global and the local consistency, which resembles theirs [39]. Nevertheless, a graph-based algorithm is naturally applicable to TL only, although it can be combined with other methods, e.g., a mixture model [42], for inductive learning. In contrast, our RegBoost is developed for SSIL.

Very recently, several semi-supervised boosting algorithms have been developed with regularization working on SSAs or information-theoretic constraints for binary classification [24], [25], [29] and multiclass classification [36], [30]. SemiBoost [25], ManifoldBoost [24], and MCSSB [36] were proposed based on manifold and smoothness assumptions without considering the low-density separation assumption, while SERBoost [29] and RMSBoost [30] were developed with the expectation regularization principle.

SemiBoost and MCSSB proposed an alternative type of regularization consisting of two terms in different forms to penalize the inconsistency between labeled and unlabeled points as well as the inconsistency between unlabeled points themselves separately [25], [36]. In order to avoid the nonconvex problem, they used the ensemble prediction $F(\mathbf{x}_i)$ for unlabeled point i other than its pseudoclass label $\mathrm{sign}[F(\mathbf{x}_i)]$ in their cost functionals [25], [36]. Similarly to ours, ManifoldBoost [24] was also developed within the margin cost functional framework [27], but directly employed the Laplace functional smoothness penalty suggested in [1] to be their regularizer. As the Laplace regularization is generally used for regression [1], the ensemble prediction $F(\mathbf{x}_i)$ for unlabeled point i is required in the regularization term. Hence, ManifoldBoost can be used in various tasks by different settings [24]. For classification, however, ensemble predictions for those unlabeled points near a decision boundary but belonging to two different classes may have a very small difference, which results in a low cost. In contrast, the use of pseudoclass labels in this circumstance leads to a high cost to penalize the label inconsistency for classification. Although the use of pseudoclass labels incurs a nonconvex cost functional, we firmly believe that our cost functional with a novel margin cost approximation for unlabeled points is more appropriate for semi-supervised classification.

For optimization, SemiBoost and MCSSB used a rather different procedure to derive their boosting algorithms [25], [36]. Unlike ManifoldBoost and our RegBoost, which apply the gradient-based local search to a functional subspace, they approximate their cost functionals with several bounds and the optimum of those bounds is used as their solutions [25], [36]. It is well known that the optimum of a cost functional may be different from that of its bounds. Thus, the tightness or the quality of those bounds would critically determine the performance of SemiBoost and MCSSB even though their cost functional is convex.

More recently, SERBoost and RMSBoost [29], [30] used the cross entropy between the prior probability and the optimized model for regularization on unlabeled points instead of SSAs. The gradient-based local search was used for optimization to derive SERBoost and RMSBoost algorithms. Unlike our initialization, which uses hard labels of unlabeled points, label priors used in SERBoost and RMSBoost lead to a probabilistic way of finding priors for unlabeled points based on underlying cluster structures, which could deal with the uncertainty of unlabeled data in a better way. As argued in [29], [30], the use of expectation regularization enables SERBoost and RMSBoost to utilize prior knowledge easily and tackle a large-scale problem efficiently. Thus, we believe that developing regularization techniques within the boosting framework from different perspectives would be helpful to understand SSAs and other useful constraints for semi-supervised ensemble learning.

6 Conclusions

We have proposed a semi-supervised boosting framework by introducing the regularization working on three fundamental SSAs. Experiments on different benchmark and real-world tasks demonstrate the constant improvement made by our algorithm with unlabeled data in comparison to the baseline and state-of-the-art SSL algorithms. In our ongoing work, we work toward seeking an alternative optimization strategy for our cost functional to tackle the local optimum problem in a more effective way and exploring potential real applications. Furthermore, we shall develop an effective approach to dealing with unlabeled data where SSAs do not hold.

Acknowledgments

The authors are grateful to M.H. Yang, the associate editor, and five anonymous reviewers for their comments, as well as X.J. Zeng for discussion on functional analysis and optimization. They would also like to thank P.K. Mallapragada and A.M. Yip for providing their MATLAB code of the SemiBoost algorithm [25] and the density-based clustering algorithm [38] used in our experiments. A preliminary version of this manuscript [12] was presented at NIPS 2007.

References

[1] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold Regularization: A Geometric Framework for Learning from Examples," *J. Machine Learning Research*, vol. 7, pp. 2399-2434, 2006.

[2] K. Bennett, A. Demiriz, and R. Maclin, "Expoliting Unlabeled Data in Ensemble Methods," *Proc. ACM Int'l Conf. Knowledge Discovery and Data Mining*, pp. 289-296, 2002.

[3] A. Blum and S. Chawla, "Combining Labeled and Unlabeled Data Using Graph Mincuts," *Proc. 10th Ann. Conf. Computational Learning Theory*, pp. 92-100, 1998.

[4] A. Blum and S. Chawla, "Learning from Labeled and Unlabeled Data with Co-Training," *Proc. Int'l Conf. Machine Learning*, pp. 19-26, 2001.

[5] O. Bousquet, O. Chapelle, and M. Hein, "Measure Based Regularization," *Advances in Neural Information Processing Systems*, vol. 16, MIT Press, 2004.

[6] Y. Bengio, O.B. Alleau, and N. Le Roux, "Label Propagation and Quadratic Criterion," *Semi-Supervised Learning*, pp. 193-207, MIT Press, 2006.

[7] O. Chapelle, J. Weston, and B. Schölkopf, "Cluster Kernels for Semi-Supervised Learning," *Advances in Neural Information Processing Systems*, vol. 15, MIT Press, 2003.

[8] O. Chapelle and A. Zien, "Semi-Supervised Classification by Low Density Separation," *Proc. 10th Int'l Workshop Artificial Intelligence and Statistics,* pp. 57-64, 2005.

[9] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-Supervised Learning.* MIT Press 2006.

[10] O. Chapelle, V. Sindhwani, and S. Keerthi, "Optimization Techniques for Semi-Supervised Support Vector Machines," *J. Machine Learning Research,* vol. 9, pp. 203-223, 2008.

[11] N.V. Chawla and G. Karakoulas, "Learning from Labeled and Unlabeled Data: An Empirical Study across Techniques and Domains," *J. Artificial Intelligence Research,* vol. 23, pp. 331-366, 2005.

[12] K. Chen and S. Wang, "Regularized Boost for Semi-Supervised Learning," *Advances in Neural Information Processing Systems,* vol. 20, MIT Press, 2007.

[13] M. Collins and Y. Singer, "Unsupervised Models for the Named Entity Classification," *Proc. SIGDAT Conf. Empirical Methods in Natural Language Processing and Very Large Corpora,* pp. 100-110, 1999.

[14] F. d'Alché-Buc, Y. Grandvalet, and C. Ambroise, "Semi-Supervised MarginBoost," *Advances in Neural Information Processing Systems,* vol. 14, MIT Press, 2002.

[15] R. Duda, P. Hart, and D. Stork, *Pattern Classification,* second ed. Wiley-Interscience, 2001.

[16] Y. Freund and R.E. Schapire, "Experiments with a New Boosting Algorithm," *Proc. Int'l Conf. Machine Learning,* pp. 148-156, 1996.

[17] Y. Grandvalet and Y. Begio, "Semi-Supervised Learning by Entropy Minimization," *Advances in Neural Information Processing Systems,* vol. 17, MIT Press, 2005.

[18] G. Haffari, "A Survey on Inductive Semi-Supervised Learning," technical report, Dept. of Computer Science, Simon Fraser Univ., 2006.

[19] T. Hertz, A. Bar-Hillel, and D. Weinshall, "Boosting Margin Based Distance Functions for Clustering," *Proc. Int'l Conf. Machine Learning,* 2004.

[20] T. Joachims, "Transductive Inference for Text Classification Using Support Vector Machines," *Proc. Int'l Conf. Machine Learning,* pp. 200-209, 1999.

[21] T. Joachims, "Transductive Learning via Spectral Graph Partitioning," *Proc. Int'l Conf. Machine Learning,* pp. 290-297, 2003.

[22] B. Kégl and L. Wang, "Boosting on Manifolds: Adaptive Regularization of Base Classifier," *Advances in Neural Information Processing Systems,* vol. 16, MIT Press, 2005.

[23] B. Leskes, "The Value of Agreement, a New Boosting Algorithm," *Proc. Int'l Conf. Computational Learning Theory,* pp. 95-110, 2005.

[24] N. Loeff, D. Forsyth, and D. Ramachandran, "ManifoldBoost: Stagewise Function Approximation for Fully-, Semi- and Un-Supervised Learning," *Proc. Int'l Conf. Machine Learning,* pp. 600-607, 2008.

[25] P. Mallapragada, R. Jin, A. Jain, and Y. Liu, "SemiBoost: Boosting for Semi-Supervised Learning," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 31, no. 11, pp. 2000-2014, Nov. 2009.

[26] A. Martinez and R. Benavente, "The AR Face Database," CVC Technical Report 24, Purdue Univ., 1998.

[27] L. Mason, P. Bartlett, J. Baxter, and M. Frean, "Functional Gradient Techniques for Combining Hypotheses," *Advances in Large Margin Classifiers,* MIT Press, 2000.

[28] K. Nigam, A. McCallum, S. Thrum, and T. Mitchell, "Using EM to Classify Text from Labeled and Unlabeled Documents," *Machine Learning,* vol. 39, pp. 103-134, 2000.

[29] A. Saffari, H. Grabner, and H. Bischof, "SERBoost: Semi-Supervised Boosting with Expectation Regularization," *Proc. European Conf. Computer Vision,* pp. III:588-601, 2008.

[30] A. Saffari, C. Leistner, and H. Bischof, "Regularized Multi-Class Semi-Supervised Boosting," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* 2009.

[31] M. Seeger, "Learning with Labeled and Unlabeled Data," technical report, School of Informatics, The Univ. of Edinburgh, 2000.

[32] P. Silapachote, D. Karuppiah, and A.R. Hanson, "Feature Selection Using Adaboost for Face Expression Recognition," *Proc. IASTED Int'l Conf. Visualization, Image, and Image Processing,* 2004.

[33] M. Szummer and T. Jaakkola, "Partially Labeled Classification with Markov Random Walks," *Advances in Neural Information Processing Systems,* vol. 15, MIT Press, 2001.

[34] M. Szummer and T. Jaakkola, "Information Regularization with Partially Labeled Data," *Advances in Neural Information Processing Systems,* vol. 15, MIT Press, 2003.

[35] *UCI Machine Learning Repository,* http://www.ics.uci.edu/mlearn/MLRepository.html, 2007.

[36] H. Valizadegan, R. Jin, and A. Jain, "Semi-Supervised Boosting for Multi-Class Classification," *Proc. European Conf. Machine Learning and Knowledge Discovery in Databases,* pp. 588-601, 2008.

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

[38] A.M. Yip, C. Ding, and T.F. Chan, "Dynamic Cluster Formation Using Level Set Methods," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 28, no. 6, pp. 877-889, June 2006.

[39] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, and B. Scholkopf, "Learning with Local and Global Consistency," *Advances in Neural Information Processing Systems,* vol. 16, MIT Press, 2004.

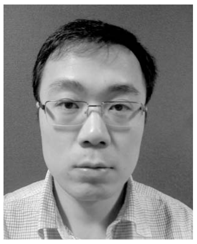
[40] X. Zhu, "Semi-Supervised Learning Literature Survey," Technical Report TR-1530, Dept. of Computer Science, Univ. of Wisconsin, 2005.

[41] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions," *Proc. Int'l Conf. Machine Learning,* pp. 912-919, 2003.

[42] X. Zhu and J. Lafferty, "Harmonic Mixtures: Combining Mixture Models and Graph-Based Methods for Inductive and Scalable Semi-Supervised Learning," *Proc. Int'l Conf. Machine Learning,* pp. 1052-1059, 2005.

[43] H. Zou, J. Zhu, and T. Hastie, "New Multicategory Boosting Algorithms Based on Multicategory Fisher-Consistent Losses," *Annals of Applied Statistics,* vol. 2, pp. 1290-1306, 2008.

Ke Chen received the BSc, MSc, and PhD degrees in computer science in 1984, 1987 and 1990, respectively. He has been with The University of Manchester since 2003. He was with The University of Birmingham, Peking University, The Ohio State University, Kyushu Institute of Technology, and Tsinghua University. He was a visiting professor at Microsoft Research Asia in 2000 and Hong Kong Polytechnic University in 2001. He has been on the editorial boards of several academic journals, including the *IEEE Transactions on Neural Networks*, and serves as the category editor of *Machine Learning and Pattern Recognition* in Scholarpedia. He was the program chair of the first International Conference on Natural Computation and has been a member of the technical program committee of numerous international conferences, including CogSci and IJCNN. He chairs the Intelligent Systems Applications Technical Committee (ISATC) and University Curricula Subcommittee of the IEEE Computational Intelligence Society. He also served as the task force chair and a member of NNTC, ETTC, and DMTC in the IEEE CIS. He was a recipient of several academic award, including the NSFC Distinguished Principal Young Investigator Award and JSPS Research Award. He has published more than 100 academic papers in refereed journals and conferences. His current research interests include machine learning, pattern recognition, machine perception, and computational cognitive systems. He is a senior member of the IEEE.



Shihai Wang received the BSc degree in computer science from Harbin Institute of Technology, China, and the MSc degree in computer science from The University of Sheffield, United Kingdom. He is currently working toward the PhD degree in the School of Computer Science at The University of Manchester, United Kingdom. His research interests lie in pattern recognition, machine learning, and their applications to automatic facial expression analysis and recognition.