Exploiting Unlabeled Data in Ensemble Methods

Kristin P. Bennett*
Department of Mathematical Sciences
Rensselaer Polytechnic Institute
Troy, NY 12180
bennek@rpi.edu

Ayhan Demiriz
E-Business Department
Verizon Inc.
919 Hidden Ridge
Irving, TX 75038
ayhan.demiriz@verizon.com

Richard Maclin
Department of Computer Science
University of Minnesota-Duluth
Duluth, MN 55812
rmaclin@d.umn.edu

ABSTRACT

An adaptive semi-supervised ensemble method, ASSEMBLE, is proposed that constructs classification ensembles based on both labeled and unlabeled data. ASSEMBLE alternates between assigning "pseudo-classes" to the unlabeled data using the existing ensemble and constructing the next base classifier using both the labeled and pseudo-labeled data. Mathematically, this intuitive algorithm corresponds to maximizing the classification margin in hypothesis space as measured on both the labeled and unlabeled data. Unlike alternative approaches, ASSEMBLE does not require a semi-supervised learning method for the base classifier. ASSEMBLE can be used in conjunction with any cost-sensitive classification algorithm for both two-class and multi-class problems. ASSEMBLE using decision trees won the NIPS 2001 Unlabeled Data Competition. In addition, strong results on several benchmark datasets using both decision trees and neural networks support the proposed method.

Keywords

Boosting, Semi-Supervised Learning, Ensemble Learning, Classification

1. INTRODUCTION

For many practical classification applications in areas such as image analysis, drug discovery, and web pages analysis, labeled data (with known class labels) can be in short supply but unlabeled data (with unknown class labels) is more readily available. Semi-supervised learning deals with methods for exploiting the unlabeled data in addition to the labeled data to improve performance on the classification task. Semi-supervised learning has been the topic of four different Neural Information Processing Workshops [5, 10, 14, 15]. Existing approaches include semi-supervised SVM [2, 22], co-training [4], and mixture models [17]. In this paper, we examine the problem from an ensemble perspective. Ensemble methods such as AdaBoost [8] work by iteratively using a base learning mechanism to construct a classifier to improve the ensemble classifier and then adding the classifier to the current ensemble with an appropriate scalar multiplier (the step-size). It is well known that such algorithms are performing gradient descent of an error function in function space [13, 16]. Depending on the measure of quality of the classifier, different criteria are produced for choosing the base classifier and assigning the step-size. D'Alché-Buc et al., [6] showed that these error measures can be extended to semi-supervised learning, but the resulting algorithm, SSMBoost, is of limited utility because it requires the base learner to be a semi-supervised algorithm and many base classifiers may be required for acceptable performance because SSMBoost uses a small fixed step-size. Our goal is to produce an adaptive semi-supervised ensemble method that can be used with any cost-sensitive base learner and that has a simple, adaptive step-size rule.

A very intuitive approach would be to choose the ensemble such that it works consistently on the unlabeled data, (i.e., such that the classification functions tended to vote for the same class on the unlabeled data). This is a form of regularization that could prevent overfitting on the training data. In this work, we will show how this intuitive idea is equivalent to maximizing the margin in function space of both the labeled and unlabeled data. It is now well known that for classification boosting can be regarded as maximizing a measure of the margin in functions space [16] and that margin measures can be adopted to semi-supervised learning [2, 6, 11]. As in SSMBoost [6], we adopt the MarginBoost [16] notation and strategy adapted to the margin measured on both the labeled and unlabeled data. But the ASSEMBLE analysis and resulting algorithms are very different from those of SSMBoost. The key difference is that we assign "pseudo-classes" to the unlabeled data. These pseudo-classes make ASSEMBLE a far more practical and powerful approach.

The advantages of ASSEMBLE are:

- Any weight-sensitive classification algorithm can be boosted using labeled and unlabeled data.
- Unlabeled data can be assimilated into margin-cost based ensemble algorithms for both two-class and multi-class problems.

*http://www.rpi.edu/∼bennek

SIGKDD '02 Edmonton, Alberta CA

- ASSEMBLE can efficiently exploit the adaptive step-sizes used to weight each base learner within existing supervised ensemble methods. SSMBoost is practically limited to fixed step-sizes.

- ASSEMBLE can exploit unlabeled data to reduced the number of classifiers needed in the ensemble therefore speeding up learning.

- ASSEMBLE works well in practice. This paper provides the first account of the winning ASSEMBLE algorithm used in the 2001 NIPS Unlabeled Data Competition.

- Computational results show the approach is effective on a number of test problems, producing more accurate ensembles than AdaBoost using the same number of base learners.

The remainder of this paper is organized as follows: Section 2 examines margins for labeled and unlabeled data and our semi-supervised version of the AnyBoost algorithm. Section 3 presents the winning ASSEMBLE.AdaBoost algorithm from the 2001 NIPS Unlabeled Data Competition and provides comparative results with supervised AdaBoost on the contest datasets. In Section 4, semi-supervised and supervised ensembles of neural networks and decision trees are studied for several benchmark datasets. We conclude with discussion and future work in Section 5.

2. MARGINS WITH UNLABELED DATA: SEMI-SUPERVISED BOOSTING

In order to apply boosting to datasets with both labeled and unlabeled data we must come up with a mechanism for defining the margin associated with unlabeled data points. While the strategy is in general applicable to many different margin cost functions, we focus on the one used in AdaBoost. Let the base classifiers be $f_j(x) : R^n \rightarrow [1, -1]$ where f_j is the jth classifier in the ensemble. Let the labeled training data, L, be the n-dimensional points $x_1, \ldots, x_\ell$ with know labels, $y_1, \ldots, y_\ell$. For now assume the problem has two classes $y_i = 1$ or -1. A multi-class extension is discussed in later sections. The ensemble classifier $F(x)$ is formed from a linear combination of the J base classifiers $F(x) = \sum_{j=1}^{J} w_j f_j(x)$ where w_j is the weighting term for the jth classifier. For labeled data points the margin is $y_i F(x_i)$. AdaBoost performs gradient descent in function space in order to minimize an exponential margin cost function

$$\frac{1}{\ell} \sum_{i \in labeled} e^{-y_i F(x_i)} \tag{1}$$

Suppose we are given a set of unlabeled data, U. To incorporate unlabeled data we must define the margin of an unlabeled data point. We do not know the class y_i for unlabeled data points. Note that for labeled data points, the margin, $y_i F(x_i)$, is positive if the point is correctly classified and negative if the point is wrongly classified. An unlabeled point is never right or wrong (in some sense you can think of it as always being correct). So as in [2, 6, 11, 22] we define the margin for an unlabeled data point x_i to be

$$|F(x_i)| \tag{2}$$

To allow the same margin to be used for both supervised and unsupervised data we can introduce the concept of a pseudo-class. The pseudo-class of an unlabeled data point x_i is defined as $y_i = sign(F(x_i))$. The margin then is

$$y_i F(x_i) \tag{3}$$

where y_i is the known class label if x_i is labeled or the pseudo-class if x_i is unlabeled. The introduction of the pseudo-class is the critical difference between our approach and the independently developed SSMBoost [6]. By introducing pseudo-classes, we can show that our Adaptive Semi-Supervised ensEMBLE method (ASSEMBLE), which corresponds to the intuitive semi-supervised ensemble algorithm, maximizes the margins of both the labeled and unlabeled points in function space. As noted above, given that the margin for labeled points is $y_i F(x_i)$, we can define the margin for unlabeled data points as $|F(x_i)|$. Using these values we can then define a margin cost function that incorporates both labeled and unlabeled data. The ASSEMBLE cost function for ADABoost is

$$C(F) = \sum_{i \in labeled} \alpha_i e^{-y_i F(x_i)} + \sum_{j \in unlabeled} \alpha_j e^{-|F(x_j)|} \tag{4}$$

In general for the supervised case with cost function $M : R \rightarrow R$, the cost function is

$$C(F) = \sum_i \alpha_i M(y_i F(x_i)) \tag{5}$$

(typically $\alpha_i = \frac{1}{\ell}$ where ℓ is number of points but we allow different weights). In general the ASSEMBLE cost function for any margin cost function, M, is

$$C(F) = \sum_{i \in labeled} \alpha_i M\left(-y_i F(x_i)\right) + \sum_{j \in unlabeled} \alpha_j \left(-|F(x_j)|\right) \tag{6}$$

The terms α_i and α_j are used to weight the labeled and unlabeled data so that we could, for example, choose to weight the margins associated with unlabeled data points as counting only 40% as much as the margins for labeled data points.

To create a practical descent-based algorithm we build on the AnyBoost approach. Recall the AnyBoost algorithm from [16]:

ALGORITHM 2.1. *Anyboost Algorithm*

1. *Let* $F_0(x) := 0$
2. *for* $t := 0$ *to* T *do*
3. *Let* $f_{t+1} := \mathcal{L}(F_t, -\nabla C(F_t))$
4. *if* $- < \nabla C(F_t), f_{t+1} > \leq 0$ *then*
5. *return* F_t
6. *end if*
7. *Choose* w_{t+1}
8. *Let* $F_{t+1} := F_t + w_{t+1} f_{t+1}$
9. *end for*
10. *return* F_{T+1}

Here, F_t represents the ensemble classifier after the tth component classifier has been added. T is the maximum number of classifiers we propose to include in our ensemble. $\mathcal{L}(F_t, -\nabla C(F_t))$ is a weak learning function applied to our existing ensemble classifier that produces a new possible classifier f_{t+1} that maximizes $- < \nabla C(F_t), f_{t+1} >$, the inner product of the new base classifier with the gradient of the cost function. If this inner product is negative, adding f_{t+1} cannot decrease the cost function, and the algorithm terminates without adding the new component classifier. To add the classifier to the ensemble, a weighting factor w_{t+1} is selected via an appropriate linesearch that guarantees decrease in the cost function.

But adding in unlabeled points our cost function now becomes

$$C(F) = \sum_{i \in labeled} \alpha_i M(y_i F(x_i)) + \sum_{j \in unlabeled} \alpha_j M(|F(x_i)|) \tag{7}$$

Note that this function is not differentiable since the absolute value function is not differentiable. By introduction of a pseudo-class y_i for the the unlabeled data, the subgradient[1] of C is

$$\nabla C(F)(x) = \left\{ \begin{array}{ll} 0 & if x \neq x_i, \\ & \quad i = \ell + 1 \ldots \ell + u \\ \alpha_i y_i M'(y_i F(x_i)) & if\ x = x_i \end{array} \right\} \tag{8}$$

where ℓ and u are the number of labeled and unlabeled data points respectively, y_i is the class (true for labeled or pseudo-class for unlabeled), and $M'(z)$ is the derivative of the margin cost function with respect to z. Then we get

$$- < \nabla C(F), f >= - \sum_i \alpha_i y_i f(x_i) M'(y_i F(x_i)) \tag{9}$$

We can safely assume the margin cost function is monotonically decreasing, thus $-M'(z)$ is always positive. Maximizing $- < \nabla C(F), f >$ is equivalent to finding f maximizing

$$- \sum_{y_i = f(x_i)} \alpha_i M'(y_i F(x_i)) + \sum_{y_i \neq f(x_i)} \alpha_i M'(y_i F(x_i)) \tag{10}$$

where for $j \in unlabeled$, $y_j = sign(F(x_j))$.

Equivalently we can define a vector D of misclassification costs, to allow a base learner to maximize $- < \nabla C(F), f >$ by constructing the minimum cost classifier. Let

$$z = - \sum_i \alpha_i M'(y_i F(x_i))$$

and define

$$D(i) := \frac{\alpha_i M'(y_i F(x_i))}{z}. \tag{11}$$

The base learner then can construct the base classifier f by minimizing (or approximately minimizing) the weighted error $\sum i : f(x_i) \neq y_i D(i)$. Note that the pseudo-classes are used for the unlabeled data, thus any cost sensitive base learning algorithm can be used.

The resulting Adaptive Semi-Supervised ensEMBLE algorithm is the following:

ALGORITHM 2.2. *ASSEMBLE*

1. *Select* $D_0(i)$
2. $y_i := [1, 0, -1]$ *for* $i \in unlabeled$.
3. *Let* $F_0(x) := 0$
4. *for* $t := 0$ *to* T *do*
5. *Let* $f_{t+1} := \mathcal{L}(S, Y, D_t)$
6. *if* $\sum D_t(i) y_i f_{t+1}(x_i) \leq 0$ *then*
7. *return* F_t
8. *end if*
9. *Choose* w_{t+1}
10. *Let* $F_{t+1} := F_t + w_{t+1} f_{t+1}$
11. *Let* $y_i = sign(F_{t+1}(x_i))$ *if* $i \in unlabeled$
12. *Let* $D_{t+1}(i) := \frac{\alpha_i M'(y_i F_{t+1}(x_i))}{\sum_j \alpha_j M'(y_j F_{t+1}(x_j))}$ *for all* i
13. *end for*
14. *return* F_{T+1}

$\mathcal{L}(S, Y, D_t)$ is our base learning algorithm that is applied to the distribution of data points S with current labels Y where the data points are weighted according to the current distribution D_t. We assume that this base learning algorithm $\mathcal{L}$ will optimize the misclassification costs, ignoring points i with either $y_i = 0$ or $D_t(i) = 0$. Now the exact algorithm that results depends on the choice of cost function. For example if assume we choose the cost function used in AdaBoost, $M(z) = e^{-z}$, then step 12 in the algorithm becomes

$$Let\ D_{t+1}(i) := \frac{\alpha_i e^{-y_i F_{t+1}(x_i)}}{\sum_j \alpha_j e^{-y_j F_{t+1}(x_j)}}\ for\ all\ i \tag{12}$$

2.1 Choice of Step-Size

More problematic is how to determine the step-size w_{t+1} in step 9. In the original AdaBoost algorithm, the step-size is performed using an exact linesearch since for the AdaBoost cost functional, the exact step-size has a closed form solution. When we add the unsupervised data we would like w_{t+1} to be chosen to minimize

$$\begin{array}{l} \sum_{i \in labeled} \alpha_i M(y_i(F_t(x_i) + w_{t+1} f_t(x_i))) \\ + \sum_{i \in unlabeled} \alpha_i M(|(F_t(x_i) + w_{t+1} f_t(x_i)|). \end{array} \tag{13}$$

Unfortunately, no closed form solution exists for w_{t+1}. One possibility is to pick a small fixed step-size (e.g., $w_{t+1} = 0.05$). If the step-size is sufficiently small, the algorithm will converge but it may converge very slowly. The algorithm then becomes inefficient in terms of training time, storage of the classification function, and prediction time because many base classifiers will be used. This was the approach used in SSMBoost [6].

An intuitive approach would be to just use the step-sizes used for supervised learning ensembles, and simply use the pseudo-classes for the unlabeled data. For example, we could choose w_{t+1} to minimize

$$\sum_i \alpha_i M(y_i(F_t(x_i) + w_{t+1} f_{t+1}(x_i)) \tag{14}$$

[1]To simplify presentation, we use ∇ to also represent the subgradient. Strictly speaking we are now using a subgradient method instead of a gradient descent method.

It is easy to show that mathematically, this leads to a decrease whenever a decrease is possible. For any unlabeled point x_i for which the sign of the pseudo-class $y_i = sign(F_t(x_i))$ is correctly predicted by the new base classifier in the ensemble (i.e., $sign(f_t(x_i)) = y_j$), the pseudo-class cost function is accurate since

$$M(|F_t(x_i) + w_{t+1}f_t(x_i)|) = M(y_i(F_t(x_i) + w_{t+1}f_t(x_i))) \tag{15}$$

If the unlabeled point is incorrectly classified (i.e., if $y_i \neq sign(f_t(x_i))$), then

$$\begin{array}{l} y_i(F_t(x_i) + w_{t+1}f_t(x_i)) \\ = |F_t(x_i)| + y_i(w_{t+1}f_t(x_i)) \\ = |F_t(x_i)| - |w_{t+1}f_t(x_i)| \\ \leq |F_t(x_i) + w_{t+1}f_t(x_i)| \end{array} \tag{16}$$

Since M is a monotonically decreasing function,

$$M(y_i(F_t(x_i) + w_{t+1}f_t(x_i))) \geq M(|(F_t(x_i) + w_{t+1}f_t(x_i)|) \tag{17}$$

So the the pseudo-class cost function provides an upper bound on the true cost function. So if we choose w_{t+1} to strictly decrease the pseudo-cost function then it must strictly decrease the true cost function. Furthermore it is always possible to strictly decrease the pseudo-cost function if the problem is not optimal. If α_i is a constant for every data point, then we can just use the step-sizes developed for supervised ensemble methods for various costs (see for example [16]). If α_i is variable, then minor modifications may be required.

2.2 ASSEMBLE Variations

It is possible to construct many variations of the basic ASSEMBLE algorithm. For example, to employ ASSEMBLE for multi-class problems we can extend the mechanism by assigning pseudo-classes for unlabeled points based on the weighted vote of the previous classifiers in the ensemble. This approach works well in our letter-recognition results (see Section 4.2).

As another example, we could employ other loss functions and associated step-sizes defined for gradient based classification algorithms such as exponential loss and logistic regression. All one has to do is add a step to estimate the pseudo-costs in each iteration. In the next sections we examine how AdaBoost has been adapted to semi-supervised learning.

3. ASSEMBLE IN NIPS COMPETITION

The following variation of the ASSEMBLE algorithm was used for the semi-supervised method competition at the NIPS'2001 workshop, *Competition: Unlabeled Data for Supervised Learning,* organized by Stefan C. Kremer and Deborah A. Stacey. ASSEMBLE (previously known as Semi-Supervised Boosting) was the best among 34 algorithms and over 100 participants utilizing unlabeled data [14].

ASSEMBLE was used to assimilate unlabeled data into a multiclass version of AdaBoost. AdaBoost was adopted to multiclass using a similar approach to [12]. Specifically, $f_t(x_i) = 1$ if an instance x_i is correctly classified and $f_t(x_i) = -1$ otherwise. This makes AdaBoost increase the weight of a misclassified point and decrease it otherwise. The predicted class is the one which achieves a majority in a vote weighted by the ensemble weights. As in the two-class case, the pseudo-classes of the unlabeled data are their predicted classes. To keep training times similar for AdaBoost and ASSEMBLE.AdaBoost, the unlabeled and labeled data were sampled at each iteration so that the size of the training set for the base learner equaled the size of labeled data. The usual adaptive step-size for AdaBoost was used with the class of unlabeled points based on the pseudo-classes.

Here is the Contest version of ASSEMBLE:

ALGORITHM 3.1. *ASSEMBLE.AdaBoost(L, U, T,α, β)*

1. *Let* $\ell := |L|$ *and* $u := |U|$
2. *Let* $D_1(i) := \left\{ \begin{array}{ll} \beta/\ell & if\ i \in L \\ (1-\beta)/u & if\ i \in U \end{array} \right\}$
3. *Let* $y_i := c$ *where* c *is the class of the nearest neighbor point in* L *for* $i \in U$.
4. *Let* $f_1 := \mathcal{L}(L + U, Y, D_1)$
5. *for* $t := 1$ *to* T *do*
6. *Let* $\hat{y}_i := f_t(x_i), \quad i = 1 \ldots \ell + u$
7. $\epsilon = \sum_i D_t[y_i \neq \hat{y}_i], \quad i = 1 \ldots \ell + u$
8. *If* $\epsilon > 0.5$ *then Stop*
9. $w_t = 0.5 * log(\frac{1-\epsilon}{\epsilon})$
10. *Let* $F_t := F_{t-1} + w_t f_t$
11. *Let* $y_i = F_t(x_i)$ *if* $i \in U$
12. *Let* D_{t+1} *as in AdaBoost (equation (12) with* α*)*
13. $S = Sample(L + U, \ell, D_{t+1})$
14. $f_{t+1} = \mathcal{L}(S, Y, D_{t+1})$
15. *end for*
16. *return* F_{T+1}

The ASSEMBLE.AdaBoost algorithm practically inherits all the good properties of AdaBoost. But AdaBoost is prone to overfitting. Overfitting in AdaBoost is prevented by maximizing the margin in function space based on both labeled and unlabeled data. Since we sample ℓ points from all the available data, ASSEMBLE has similar computational complexity with AdaBoost. Nearest neighbor assignment was used to assign the initial pseudo-class labels. While the above algorithm is potentially applicable to any supervised learning method, we used limited depth decision trees [21] in the competition. The depth of the decision trees was set to a level that minimizes the AdaBoost training error. The rationale behind using decision trees was to show that semi-supervised approach can work even with a simple classification method. The initial misclassification costs D_0 were set skewed to emphasize the labeled data, but after that misclassification costs for unlabeled and labeled data were assumed to have equal importance ($\alpha_i = 1$).

For the contest, ASSEMBLE.AdaBoost was implemented in SPLUS. The results from the competition are summarized in Table 1. There were originally 13 datasets in the competition for both classification and regression problems. We only considered the classification problems. Because of the limitations in SPLUS, we did not run ASSEMBLE boosting on some of the datasets. Separate labeled (L), unlabeled

Table 1: Results for ASSEMBLE.AdaBoost in NIPS 2001 Semi-Supervised Competition. Acc is the test set accuracy and Impr is the percentage improvement of ASSEMBLE.AdaBoost over AdaBoost.

Data	Dim.	No of Classes	No of Points			Acc	Impr
			L	U	T	(%)	(%)
P1	13	2	36	92	100	65.00	8.33
P4	192	9	108	108	216	78.70	21.43
P5	1000	2	50	3952	100	76.00	16.92
P6	12	2	530	2120	2710	75.72	0.10
P8	296	4	537	608	211	57.82	16.19
CP2	21	3	300	820	300	50.67	26.67
CP3	10	5	1000	1269	500	41.20	6.74

(U) and test data were provided in this competition. Accuracy results on the test data and the improvements compared to AdaBoost are reported in Table 1. We used the same number of iterations for both AdaBoost and ASSEMBLE.AdaBoost in general for the competition. The number of iterations (T) were determined based on the training error convergence in the supervised AdaBoost runs. The parameter β was set to 0.9 and α_i to 1 in all runs. Semi-supervised algorithm produced double-digit improvements relative to AdaBoost.

4. BENCHMARK EXPERIMENTS

In order to assess the performance of the algorithms outlined above, we performed additional experiments on a number of datasets drawn from the literature. To test the effectiveness of ASSEMBLE across different classifier methodologies, we performed experiments using decision trees and neural networks, methods that have proven very effective in previous experiments with ensembles [1, 18].

4.1 Results from DT Experiments

ASSEMBLE.AdaBoost used with decision trees was tested on three benchmark datasets obtained from boosting literature [19]. The benchmark datasets were used without any data transformation. Each dataset consists of 100 random instances of training and test dataset pairs. The detailed information about datasets can be found in [19]. To conform with previous approaches in empirical studies of semi-supervised learning methods [6, 7, 9, 11], we left some of the training data as unlabeled data and then evaluated the algorithm on test data. Training data were sampled 10 times at three different levels to form labeled and unlabeled data. Specifically, the size of the combined labeled and unlabeled data was held constant, but the proportion of data treated as unlabeled was varied from 20 to 60 percent. Thus 10 predictions are made for each test point.

The ASSEMBLE.AdaBoost code used in this comparison was identical to the one used in the contest described in Section 3. The code was implemented in SPLUS to allow the RPART [21] decision tree algorithm to be used as the base learner. We set the maximum depth of the decision trees generated by RPART at 4. Information gain was used as the splitting criterion. RPART can be used for regression problems but we examined only the classification case in this paper.

The number of boosting iterations was limited to a maximum of 25. Again we set the parameters β to 0.9 and α_i to 1 in our experiments. The results from experiments are reported in Table 2. We report the mean error rates of ASSEMBLE.Adaboost and Adaboost on 1000 different runs for each benchmark dataset. Unlabeled data were sampled as 60%, 40% and 20% of the total data used for training.

Results in Table 2 demonstrate that that semi-supervised boosting is better or comparable to AdaBoost despite the fact that in each trial ASSEMBLE.Adaboost received much less labeled data.

4.2 Results from NN Experiments

In our last set of experiments we examined the performance of ASSEMBLE on neural networks. The ASSEMBLE variant for the neural network experiments was slightly different than that used for the NIPS Contest. The algorithm differs in that step 13 from ASSEMBLE.AdaBoost is dropped (all of the available data is used), and in steps 2 and 3 the unlabeled points are initially set to have class 0 (they are not incorporated into the first classifier). These changes were used largely because of the significant cost for nearest neighbor labeling for the larger datasets we employed in our multiple tests. We also employ a parameter α_i that is different for the two sets of data, using a term that weights the margins of unlabeled points by less (fractions from 0.4 to 1.0 were employed) to prevent the networks from overly focusing on the unlabeled points.

In our neural network experiments we used simple multilayer perceptrons with a single layer of hidden units. The networks were trained using backpropagation with a learning rate of 0.15 and a momentum value of 0.90. The datasets for the experiments are breast-cancer-wisconsin, pima-indians diabetes, and letter-recognition drawn from the UCI Machine Learning repository [3]. The number of units in the hidden layer for the datasets was 5 for the breast-cancer and diabetes datasets and 40 in the letter-recognition dataset. The number of training epochs was set to 20 for breast-cancer, 30 for diabetes, and 30 for letter recognition, as done in previous ensemble experiments (see [18]). We explored a number of values ranging from 0.4 to 1.0 for the parameter used to weight the unlabeled data points, though there was very little difference among the values. The results we show use a value of 0.4 for this parameter.

To obtain our results we performed ten 10-fold cross validation experiments for each result. Unlabeled data was obtained by randomly marking a percentage of the data (in this case, 10, 25, and 50 percent) of the data as unlabeled points. Each semi-supervised and regular AdaBoost experiment was run on the same set of points (though the unlabeled points from each dataset are left out when applying AdaBoost). Each method was allowed to produce up to 30 members of the ensemble, although the breast-cancer and

Table 2: Experiments on Decision Trees

Dataset	Unlabeled Rate (%)	AdaBoost Test Set Error (Std Dev)	ASSEMBLE Test Set Error (Std Dev)
B. Cancer	60	32.51 (2.88)	32.07 (2.93)
B. Cancer	40	31.59 (3.00)	31.36 (3.01)
B. Cancer	20	30.95 (3.25)	30.26 (3.20)
Banana	60	15.96 (0.73)	16.41 (0.74)
Banana	40	14.50 (0.58)	14.94 (0.58)
Banana	20	13.71 (0.53)	14.17 (0.59)
Diabetes	60	27.91 (1.23)	27.52 (1.16)
Diabetes	40	27.55 (1.22)	27.21 (1.15)
Diabetes	20	27.39 (1.36)	26.92 (1.27)

Table 3: Experiments for ASSEMBLE.AdaBoost on Neural Networks

Dataset	Unlabeled Rate (%)	AdaBoost Test Set Error (Std Dev)	ASSEMBLE Test Set Error (Std Dev)
Wisconsin Breast Cancer	50	5.09 (1.17)	4.34 (1.21)
Wisconsin Breast Cancer	25	4.91 (0.62)	4.15 (0.45)
Wisconsin Breast Cancer	10	4.46 (0.41)	3.84 (0.43)
Pima Indians Diabetes	50	25.95 (1.44)	25.54 (1.24)
Pima Indians Diabetes	25	25.81 (1.25)	24.45 (1.04)
Pima Indians Diabetes	10	25.45 (1.29)	24.22 (1.19)
Letter Recognition	50	10.11 (0.50)	9.50 (0.49)
Letter Recognition	25	9.63 (0.30)	9.21 (0.31)
Letter Recognition	10	6.87 (0.21)	6.15 (0.16)

diabetes converged quickly and therefore stopped at many fewer classifiers.

Table 3 shows the results using the neural networks described above as component classifiers for this variation of the ASSEMBLE algorithm. Note that in every case, ASSEMBLE produced a small but measurable gain in the overall performance.

As a further test of the effectiveness of our algorithm we performed several experiments on the largest of these datasets (letter-recognition) to assess how useful unlabeled data would be in overcoming limitations imposed by the bias of the classifier. In this case, we greatly limited the capabilities of the classifier by decreasing the number of units in the hidden layer (we performed experiments using, 5, 10, and 20 hidden units). Since this problem has 26 output classes, the resulting neural networks not only have to perform generalization, but also an encoding/decoding problem. Thus, we would expect that additional data might help to overcome these problems.

Figure 1 shows the resulting error rate graphed as a function of the number of classifiers. Note that not only does ASSEMBLE outperform standard AdaBoost but that the resulting gains generally happen earlier, and it is only with a large number of classifiers in the ensemble that AdaBoost begins to catch up.

5. CONCLUSION

In this paper we introduce a novel semi-supervised learning technique ASSEMBLE (for Adaptive Semi-Supervised enSEMBLE) that is able to solve semi-supervised learning problems (i.e., problems where some of the data is not labeled with a class). ASSEMBLE can be used to make any cost sensitive classification method semi-supervised by incorporating it into a semi-supervised boosting algorithm. The key to making ASSEMBLE work is the introduction of pseudo-classes for the unlabeled data. We demonstrate that an appropriate margin can be derived by attaching pseudo-class y_i to an unlabeled data point that simply reflects the majority class picked for that point by the current ensemble. This choice of pseudo-class allows us to derive a class of boosting algorithms that can be applied in semi-supervised learning situations. By incorporating pseudo-classes, existing ensemble algorithms can be readily adapted to semi-supervised learning. In this work, we focus on ASSEMBLE.AdaBoost, a semi-supervised version of the popular AdaBoost algorithm based on exponential margin cost functions.

ASSEMBLE performs extremely well in empirical tests on both two-class and multi-class problems. Using decision trees as its component classifier, it placed first out of 34 algorithms in the NIPS 2001 competition on semi-supervised datasets. In further empirical tests using neural networks and decision trees on datasets where some of the data points were artificially marked as unlabeled, ASSEMBLE consistently performs as well as or better than AdaBoost. Overall, the ASSEMBLE algorithm appears to be a robust method for combining unlabeled data with labeled data.

We plan to apply ASSEMBLE to larger datasets in order to determine how well the algorithm scales for larger problems. ASSEMBLE should be readily adaptable to scalable boosting algorithms such as in [20]. An interesting open problem is how to exploit unlabeled data in regression problems. For classification ASSEMBLE favors ensembles that vote consistently on the unlabeled data. The analogous strategy for regression would be to favor ensembles that exhibit low variance on the unlabeled data. But we leave these issues to future work.

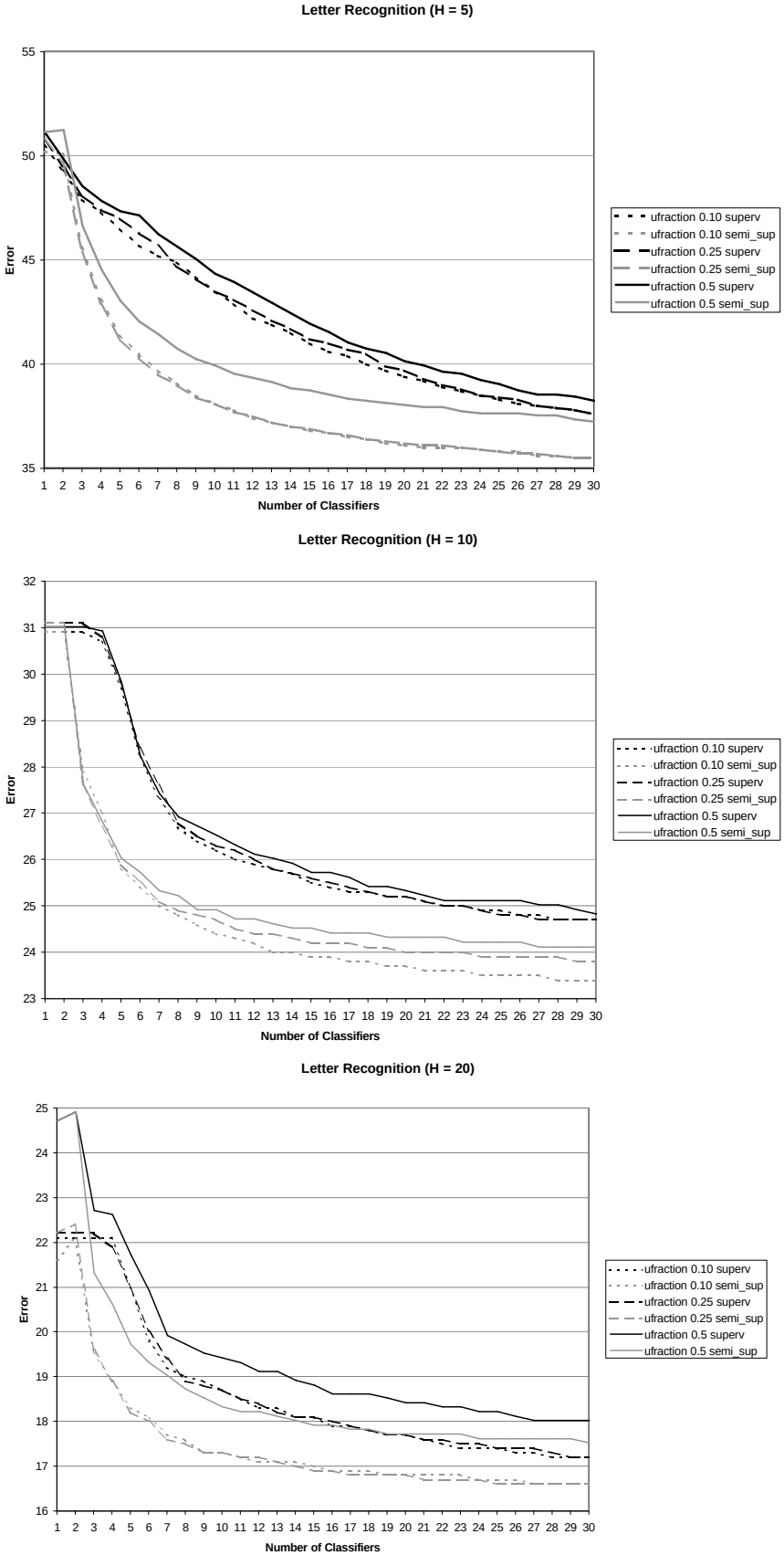

Figure 1: Neural network results for the Letter Recognition dataset using networks with 5, 10 and 20 hidden units. Results shown are for 10, 25, and 50 percent of the data marked as unlabeled (ufractions of 0.10, 0.25, and 0.5) for AdaBoost (superv) and ASSEMBLE (semi_sup).

6. ACKNOWLEDGMENTS

This work was partially supported by NSF IRI-9702306, NSF IIS-9979860 and NSF DMS-9872019. Many thanks to Stefan Kremer and Deborah Stacey for organizing the NIPS workshop.

7. REFERENCES

[1] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–139, 1999.

[2] K. P. Bennett and A. Demiriz. Semi-supervised support vector machines. In D. C. M. Kearns, S. Solla, editor, *Advances in Neural Information Processing Systems 11*, pages 368–374, Cambridge, MA, 1999. MIT Press.

[3] C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[4] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, 1998.

[5] R. Caruana, V. de Sa, M. Kearns, and A. McCallum. Integrating supervised and unsupervised learning, 1998. http://www.cs.cmu.edu/∼mccallum/supunsup/.

[6] F. d'Alché Buc, Y. Grandvalet, and C. Ambroise. Semi-supervised marginboost. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.

[7] A. Demiriz, K. P. Bennett, and M. J. Embrechts. A genetic algorithm approach for semi-supervised clustering. *Journal of Smart Engineering System Design*, 4:35–44, 2002. Taylor & Francis.

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

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

[10] T. Graepel, R. Herbrich, and K. Obermayer. Using unlabeled data for supervised learning, 1999. http://stat.cs.tu-berlin.de/nips99/.

[11] Y. Grandvalet, F. d'Alché Buc, and C. Ambroise. Boosting mixture models for semi-supervised learning. In G. Dorffner, H. Bischof, and K. Hornik, editors, *ICANN 2001*, pages 41–48. LNCS 2130, Springer-Verlag, 2001.

[12] A. J. Grove and D. Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *AAAI/IAAI*, pages 692–699, 1998.

[13] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, New York, 2001.

[14] S. C. Kremer and D. A. Stacey. Competition: Unlabeled data for supervised learning, 2001. http://q.cis.uoguelph.ca/∼skremer/NIPS2001/.

[15] S. C. Kremer, D. A. Stacey, and K. P. Bennett. Unlabeled data supervised learning competition, 2000. http://q.cis.uoguelph.ca/∼skremer/NIPS2000/.

[16] L. Mason, P. Bartlett, J. Baxter, and M. Frean. Functional gradient techniques for combining hypotheses. In B. Schölkopf, A. Smola, P. Bartlett, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press, 2000.

[17] K. Nigam, A. McCallum, S. Thrum, and T. Mitchell. Using EM to classify test from labeled and unlabeled documents. *Machine Learning*, 39:2:103–134, 2000.

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

[19] G. Rätsch. Benchmark datasets, 1998. http://ida.first.gmd.de/∼raetsch/data/benchmarks.htm.

[20] W. N. Street and Y. Kim. An ensemble method for large-scale classification. In *Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-01)*, 2001.

[21] T. Therneau and B. Atkinson. Rpart: Recursive partitioning software, February 2000. Available at http://www.mayo.edu/hsr/Sfunc.html.

[22] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.