Creating Diverse Ensemble Classifiers

Prem Melville

The University of Texas at Austin, 2003

Supervisor: Raymond J. Mooney

Ensemble methods like Bagging and Boosting which combine the decisions of multiple hypotheses are some of the strongest existing machine learning methods. The diversity of the members of an ensemble is known to be an important factor in determining its generalization error. We present a new method for generating ensembles, DECORATE (Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples), that directly constructs diverse hypotheses using additional artificially-constructed training examples. The technique is a simple, general meta-learner that can use any strong learner as a base classifier to build diverse committees. Experimental results using decision-tree induction as a base learner demonstrate that this approach consistently achieves higher predictive accuracy than both the base classifier and Bagging. DECORATE also obtains higher accuracy than Boosting early in the learning curve when training data is limited.

We propose to show that DECORATE can also be effectively used for (1) *active learning*, to reduce the number of training examples required to achieve high accuracy; (2) exploiting unlabeled data to improve accuracy in a *semi-supervised* learning setting; (3) combining active learning with semi-supervision for improved results; (4) obtaining better class membership probability estimates; (5) reducing the error of regressors; and (6) improving the accuracy of relational learners.

Contents

Chapter 1

Introduction

One of the major advances in inductive learning in the past decade was the development of *ensemble* or *committee* approaches that learn and retain multiple hypotheses and combine their decisions during classification (Dietterich, 2000). For example, *Boosting* (Freund & Schapire, 1996) is an ensemble method that learns a series of "weak" classifiers each one focusing on correcting the errors made by the previous one; and it is currently one of the best generic inductive classification methods (Hastie, Tibshirani, & Friedman, 2001).

Constructing a *diverse* committee in which each hypothesis is as different as possible, while still maintaining consistency with the training data, is known to be a theoretically important property of a good ensemble method (Krogh & Vedelsby, 1995). Although all successful ensemble methods encourage diversity to some extent, few have focused directly on the goal of maximizing diversity. Existing methods that focus on achieving diversity (Opitz & Shavlik, 1996; Rosen, 1996) are fairly complex and are not general *meta-learners* like Bagging (Breiman, 1996b) and Boosting which can be applied to any base learner to produce an effective committee (Witten & Frank, 1999).

We present a new meta-learner DECORATE (Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples), that uses an existing "strong" learner (one that provides high accuracy on the training data) to build an effective diverse commit-

tee in a simple, straightforward manner. This is accomplished by adding different randomly constructed examples to the training set when building new committee members. These artificially constructed examples are given category labels that *disagree* with the current decision of the committee, thereby easily and directly increasing diversity when a new classifier is trained on the augmented data and added to the committee.

Boosting and Bagging provide diversity by sub-sampling or re-weighting the existing training examples. If the training set is small, this limits the amount of ensemble diversity that these methods can obtain. DECORATE ensures diversity on an arbitrarily large set of additional artificial examples. Therefore, one hypothesis is that it will result in higher generalization accuracy when the training set is small. In our preliminary work, we present experimental results on a wide range of UCI data sets comparing Boosting, Bagging, and DECORATE, all using J48 decision-tree induction as a base learner. J48 is a Java implementation of C4.5 (Quinlan, 1993) introduced in (Witten & Frank, 1999). Cross-validated learning curves support the hypothesis that "DECORATEd trees" generally result in greater classification accuracy for small training sets. In fact, even given large training sets, DECORATE outperforms Bagging and is competitive with AdaBoost.

We claim that DECORATE's success is due to its focus on *diversity* while constructing ensembles. We support this claim with additional experiments that show a strong correlation between *diversity* and *error reduction*.

This proposal explains how DECORATE works and demonstrates its effectiveness through a wide range of experiments. Our proposed work aims to show that DECORATE can be useful in many ways other than improving *classification* accuracy in a *purely* supervised setting. We propose to show that DECORATE can also be effectively used for the following:

- *Active learning*, to reduce the number of training examples required to learn an accurate model;
- Exploiting unlabeled data to improve accuracy in a *semi-supervised* learning setting;
- Combining both *active* and *semi-supervised* learning for improved results;

- Obtaining improved class membership probability estimates, to assist in cost-sensitive decision making;
- Reducing the error of regression methods; and
- Improving the accuracy of relational learners.

Finally, we plan to investigate the theoretical properties of DECORATE.

Chapter 2

Background and Related Work

In this chapter we provide some background on ensemble methods and introduce the notion of ensemble *diversity*. We then motivate our approach and discuss related work. The focus of our preliminary work has been on methods for building ensembles for *classification* using purely supervised learning. We propose several extensions to our preliminary work; and an introduction and related work for each of our extensions is provided separately in the chapter on proposed work (Chapter 5).

2.1 Ensembles of Classifiers

We begin by introducing some notation and defining the supervised learning task. We attempt to adhere to the notation and definitions in (Dietterich, 1997).

Y is a set of classes.

T is a set of training examples, i.e. description-classification pairs.

C is a classifier, a function from objects to classes.

C^* is an ensemble of classifiers.

C_i is the i^{th} classifier in ensemble C^*.

w_i is the weight given to the vote of C_i.

n is the number of classifiers in ensemble C^*.

x_i is the description of the i^{th} example/instance.

y_i is the correct classification of the i^{th} example.

m is the number of instances to be classified.

L is a learner, a function from training sets to classifiers.

In supervised learning, a learning algorithm is given a set of training examples of the form $\{(x_1, y_1), ..., (x_m, y_m)\}$ for some unknown function $y = f(x)$. The description x_i is usually a vector of the form $< x_{i,1}, x_{i,2}, ..., x_{i,k} >$ whose components are real or discrete (nominal) values, such as height, weight, age, eye-color, and so on. These components of the description are often referred to as the features or attributes of an example. The values of y are typically drawn from a discrete set of classes Y in the case of *classification* or from the real line in the case of *regression*. Our work is primarily focused on the classification task. A learning algorithm L, is trained on a set of training examples T, to produce a *classifier* C. The classifier is a hypothesis about the true (target) function f. Given a new example x, the classifier predicts the corresponding y value. The aim of the classification task is to learn a classifier that minimizes the error in predictions on an independent test set of examples (generalization error). For classification, the most common measure for error is the 0/1 loss function, given by:

$$error_{C,f}(x) = \begin{cases} 0: \text{ if } C(x) = f(x) \\ 1: \text{otherwise} \end{cases} \tag{2.1}$$

An *ensemble (committee)* of classifiers is a set of classifiers whose individual decisions are combined in some way (typically by weighted or unweighted voting) to classify new examples. One of the most active areas of research in supervised learning has been to study methods for constructing good ensembles of classifiers. This area is referred to by different names in the literature — committees of learners, mixtures of experts, classifier ensembles, multiple classifier systems, consensus theory, etc. (Kuncheva & Whitaker, 2003). In general, an ensemble method is used to improve on the accuracy of a given learn-

ing algorithm. We will refer to this learning algorithm as the *base learner*. The base learner trained on the given set of training examples is referred to as the *base classifier*. It has been found that in most cases combining the predictions of an ensemble of classifiers produces more accurate predictions than the base classifier (Dietterich, 1997).

There have been many methods developed for the construction of ensembles. Some of these methods, such as Bagging and Boosting are *meta-learners* i.e. they can be applied to *any* learning algorithm. Other methods are specific to particular learners. For example, Negative Correlation Learning (Liu & Yao, 1999) is used specifically to build committees of Neural Networks. We focus primarily on ensemble methods that are *meta-learners*. This is because, some learning algorithms are often better suited for a particular domain than others. Therefore a *general* ensemble approach that is independent of the particular base learner is preferred.

2.2 Ensemble Diversity

In an ensemble, the combination of the output of several classifiers is only useful if they disagree on some inputs (Hansen & Salamon, 1990; Tumer & Ghosh, 1996). We refer to the measure of disagreement as the *diversity/ambiguity* of the ensemble. For regression problems, *mean squared error* is generally used to measure accuracy, and *variance* is used to measure diversity. In this setting, Krogh and Vedelsby (1995) show that the generalization error, E, of the ensemble can be expressed as $E = \bar{E} - \bar{D}$; where $\bar{E}$ and $\bar{D}$ are the mean error and diversity of the ensemble respectively. This result implies that increasing ensemble diversity while maintaining the average error of ensemble members, should lead to a decrease in ensemble error. Unlike regression, for the classification task the above simple linear relationship does not hold between E, $\bar{E}$ and $\bar{D}$. But there is still strong reason to believe that increasing diversity should decrease ensemble error (Zenobi & Cunningham, 2001).

There have been several measures of diversity for classifier ensembles proposed in

the literature. In a recent study, Kuncheva and Whitaker (2003) compared ten different measures of diversity. They found that most of these measures are highly correlated. However, to the best of our knowledge, there has not been a conclusive study showing which measure of diversity is the best to use for constructing and evaluating ensembles.

2.3 Our Basic Approach

For our work, we use the disagreement of an ensemble member with the ensemble's prediction as a measure of diversity. More precisely, if $C_i(x)$ is the prediction of the i-th classifier for the label of x; $C^*(x)$ is the prediction of the entire ensemble, then the diversity of the i-th classifier on example x is given by

$$d_i(x) = \begin{cases} 0: \text{ if } C_i(x) = C^*(x) \\ 1: \text{otherwise} \end{cases} \tag{2.2}$$

To compute the diversity of an ensemble of size n, on a training set of size m, we average the above term:

$$\frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} d_i(x_j) \tag{2.3}$$

This measure estimates the probability that a classifier in an ensemble will disagree with the prediction of the ensemble as a whole. Our approach is to build ensembles that are consistent with the training data and that attempt to maximize this diversity term.

In (Melville & Mooney, 2003) we introduced a new meta-learner DECORATE (Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples) that uses an existing learner to build an effective diverse committee in a simple, straightforward manner. This is accomplished by adding different randomly constructed examples to the training set when building new committee members. These artificially constructed examples are given category labels that *disagree* with the current decision of the committee, thereby easily and directly increasing diversity when a new classifier is trained on the augmented data and added to the committee.

2.4 Related Work

There have been many ensemble methods studied in the literature. We only present the approaches that are relevant to this study; for an excellent survey on ensemble methods see (Dietterich, 2000). Bagging (Breiman, 1996b) and AdaBoost (Freund & Schapire, 1996) are the most popular methods, and so we compare our approach to them.

2.4.1 Bagging

In a Bagging ensemble, each classifier is trained on a set of m training examples, drawn randomly with replacement from the original training set of size m. Such a training set is called a *bootstrap replicate* of the original set. Each bootstrap replicate contains, on average, 63.2% of the original training set, with many examples appearing multiple times.

Predictions on new examples are made by taking the majority vote of the ensemble. Since each ensemble member is not exposed to the same set of examples, they are different from each other. By voting the predictions of each of these classifiers, Bagging seeks to reduce the error due to variance of the base classifier.

2.4.2 Boosting

There are several variations of Boosting that appear in the literature. When we talk about Boosting or AdaBoost, we refer to the AdaBoost.M1 algorithm described in (Freund & Schapire, 1996) (see Algorithm 1). This algorithm assumes that the base learner can handle weighted examples. AdaBoost maintains a set of weights over the training examples. In each iteration i, the classifier C_i is trained to minimize the weighted error on the training set. The weighted error of C_i is computed and used to update the distribution of weights on the training examples. The weights of misclassified examples are increased and the weights on correctly classified examples are decreased. The next classifier is trained on the examples with this updated distribution and the process is repeated.

After training, the ensemble's predictions are made using a weighted vote of the individual classifiers: $\sum_i w_i C_i(x)$. The weight of each classifier, w_i, is computed according to its accuracy on the weighted example set it was trained on.

Algorithm 1 The ADABOOST.M1 algorithm

Input:
$BaseLearn$ - base learning algorithm
T - set of m training examples $< (x_1, y_1), ..., (x_m, y_m) >$ with labels $y_j \in Y$
I - number of Boosting iterations
Initialize Distribution of weights on examples, $D_1(x_j) = 1/m$ for all $x_j \in T$

1. For i = 1 to I
2. Train base learner given the distribution D_i, C_i = $BaseLearn(T, D_i)$
3. Calculate error of C_i, $\epsilon_i = \sum_{\substack{x_j \in T, \\ C_i(x_j) \neq y_j}} D_i(x_j)$
4. If $\epsilon_i > 1/2$ then set $I = i - 1$ and abort loop
5. Set $\beta_i = \epsilon_i / (1 - \epsilon_i)$
6. Update weights, $D_{i+1}(x_j) = D_i(x_j) \times \begin{cases} \beta_t : \text{if } C_i(x_j) = y_j \\ 1 \ : \text{otherwise} \end{cases}$
7. Normalize weights, $D_{i+1}(x_j) = \dfrac{D_{i+1}(x_j)}{\sum_{x_j \in T} D_{i+1}(x_j)}$

Output: The final hypothesis, $C^*(x) = \arg\max_{y \in Y} \sum_{i: C_i(x) = y} log \dfrac{1}{\beta_i}$

AdaBoost is a very effective ensemble method that has been tested extensively by many researchers (Bauer & Kohavi, 1999; Dietterich, 2000; Quinlan, 1996a; Maclin & Opitz, 1997). Applying AdaBoost to decision trees has been particularly successful; to the extent that Breiman, a leading statistician, referred to AdaBoost with trees as the "best off-the-shelf classifier in the world" (Breiman, 1996a). The success of AdaBoost has lead to its use in a host of different applications, including text categorization (Schapire & Singer,

2000), online auctions (Schapire, Stone, McAllester, Littman, & Csirik, 2002), document routing (Iyer, Lewis, Schapire, Singer, & Singhal, 2000), part-of-speech tagging (Abney, Schapire, & Singer, 1999), recommender systems (Freund, Iyer, Schapire, & Singer, 1998), first-order learning (Quinlan, 1996b) and named-entity extraction (Collins, 2002).

Despite its popularity, Boosting does suffer from some drawbacks. In particular, Boosting can fail to perform well given insufficient data (Schapire, 1999). This observation is consistent with the Boosting theory. Boosting also does not perform well when there is a large amount of classification noise (i.e. training and test examples with incorrect class labels) (Dietterich, 2000).

2.4.3 Explicit Diversity-Based Approaches

DECORATE differs from ensemble methods, such as Bagging, in that it *explicitly* tries to foster ensemble diversity. There have been some other attempts at building ensembles that focus on the issue of diversity.

Liu and Yao (1999) and Rosen (1996) simultaneously train neural networks in an ensemble using a correlation penalty term in their error functions. Brown and Wyatt (2003) provide a good theoretical analysis of these methods, commonly referred to as Negative Correlation Learning.

Opitz and Shavlik (1996) and Opitz (1999) use a genetic algorithm to search for a good ensemble of networks. To guide the search they use an objective function that incorporates both an accuracy and diversity term.

Tumer and Ghosh (1996) reduce the correlation between classifiers in an ensemble by exposing them to different feature subsets. They train m classifiers, one corresponding to each class in a m-class problem. For each class, a subset of features that have a low correlation to that class is eliminated. The degree of correlation between classifiers can be controlled by the amount features that are eliminated. This method, called *input decimation*, has been further explored in (Tumer & Oza, 1999).

Zenobi and Cunningham (2001) also build ensembles based on different feature subsets. In their approach, feature selection is done using a hill-climbing strategy based on classifier error and diversity. A classifier is rejected if the improvement of one of the metrics leads to a "substantial" deterioration of the other; where "substantial" is defined by a pre-set threshold.

All these approaches attempt to simultaneously optimize diversity and error of *individual* ensemble members. On the other hand, DECORATE focuses on reducing the error of the *entire* ensemble by increasing diversity. At no point does the training accuracy of the ensemble go below that of the base classifier; however, this is a possibility with previous methods. Furthermore, apart from (Opitz, 1999), none of the previous studies compared their methods with the standard ensemble approaches such as Boosting and Bagging.

2.4.4 Use of Artificial Examples

To the best of our knowledge, DECORATE is the only method that uses artificially constructed examples to improve generalization accuracy.

One ensemble approach that also utilizes artificial training data is the active learning method introduced in (Cohn, Atlas, & Ladner, 1994). Rather than to improve accuracy, the goal of the committee here is to select good new training examples using the existing training data. The labels of the artificial examples are selected to produce hypotheses that more faithfully represent the entire version space rather than to produce diversity. Cohn's approach labels artificial data either all positive or all negative to encourage, respectively, the learning of more general or more specific hypotheses.

Another application of artificial examples for ensembles is Combined Multiple Models (CMMs) (Domingos, 1997). The aim of CMMs is to improve the comprehensibility of an ensemble of classifiers, by approximating it by a single classifier. Artificial examples are generated and labeled by a voted ensemble. They are then added to the original training set. The base learner is trained on this augmented training set to produce an approximation

of the ensemble. The role of artificial examples here is to create less complex models, *not* to improve classification accuracy.

Chapter 3

Our Approach

In this chapter we explain our algorithm DECORATE in detail, and present arguments for its effectiveness.

3.1 DECORATE: Algorithm Definition

In DECORATE (see Algorithm 2), an ensemble is generated iteratively, first learning a classifier and then adding it to the current ensemble. We initialize the ensemble to contain the classifier trained on the given training data. The classifiers in each successive iteration are trained on the original training data combined with some artificial data. In each iteration, artificial training examples are generated from the data distribution; where the number of examples to be generated is specified as a fraction, R_{size}, of the training set size. The labels for these artificially generated training examples are chosen so as to differ maximally from the current ensemble's predictions. The construction of the artificial data is explained in greater detail in the following section. We refer to the labeled artificially generated training set as the *diversity data*. We train a new classifier on the union of the original training data and the diversity data, thereby forcing it to differ from the current ensemble. Therefore adding this classifier to the ensemble will increase its diversity. While forcing diversity we

still want to maintain training accuracy. We do this by rejecting a new classifier if adding it to the existing ensemble decreases its accuracy. This process is repeated until we reach the desired committee size or exceed the maximum number of iterations.

To classify an unlabeled example, x, we employ the following method. Each base classifier, C_i, in the ensemble C^* provides probabilities for the class membership of x. If $P_{C_i,y}(x)$ is the probability of example x belonging to class y according to the classifier C_i, then we compute the class membership probabilities for the entire ensemble as:

$$P_y(x) = \frac{\sum_{C_i \in C^*} P_{C_i,y}(x)}{|C^*|}$$

where $P_y(x)$ is the probability of x belonging to class y. We then select the most probable class as the label for x i.e. $C^*(x) = \arg\max_{y \in Y} P_y(x)$

3.2 Construction of Artificial Data

We generate artificial training data by randomly picking data points from an approximation of the training-data distribution. For a numeric attribute, we compute the mean and standard deviation from the training set and generate values from the Gaussian distribution defined by these. For a nominal attribute, we compute the probability of occurrence of each distinct value in its domain and generate values based on this distribution. We use Laplace smoothing so that nominal attribute values not represented in the training set still have a non-zero probability of occurrence. In constructing artificial data points, we make the simplifying assumption that the attributes are independent. It is possible to more accurately estimate the joint probability distribution of the attributes; but this would be time consuming and require a lot of data. Furthermore, the results seem to indicate that we can achieve good performance even with the crude approximation we use.

In each iteration, the artificially generated examples are labeled based on the current ensemble. Given an example, we first find the class membership probabilities predicted by

Algorithm 2 The DECORATE algorithm

Input:
$BaseLearn$ - base learning algorithm
T - set of m training examples $< (x_1, y_1), ..., (x_m, y_m) >$ with labels $y_j \in Y$
C_{size} - desired ensemble size
I_{max} - maximum number of iterations to build an ensemble
R_{size} - factor that determines number of artificial examples to generate

1. $i = 1$
2. $trials = 1$
3. $C_i = BaseLearn(T)$
4. Initialize ensemble, $C^* = \{C_i\}$
5. Compute ensemble error, $\epsilon = \frac{\sum_{x_j \in T, C^*(x_j) \neq y_j} 1}{m}$
6. While $i < C_{size}$ and $trials < I_{max}$
7. Generate $R_{size} \times |T|$ training examples, R, based on distribution of training data
8. Label examples in R with probability of class labels inversely proportional to predictions of C^*
9. $T = T \bigcup R$
10. $C' = BaseLearn(T)$
11. $C^* = C^* \bigcup \{C'\}$
12. $T = T - R$, remove the artificial data
13. Compute training error, ϵ', of C^* as in step 5
14. If $\epsilon' \leq \epsilon$
15. $i = i + 1$
16. $\epsilon = \epsilon'$
17. otherwise,
18. $C^* = C^* - \{C'\}$
19. $trials = trials + 1$

the ensemble. We replace zero probabilities with a small non-zero value and normalize the probabilities to make it a distribution. Labels are then selected, such that the probability of selection is inversely proportional to the current ensemble's predictions. So if the current ensemble predicts the class membership probabilities $P_y(x)$, then a new label is selected based on the distribution

$$P_y^{-1}(x) = \frac{1/P_y(x)}{\sum_y 1/P_y(x)}$$

3.3 Why DECORATE Should Work

Ensembles of classifiers are often more accurate than its component classifiers if the errors made by the ensemble members are uncorrelated (Hansen & Salamon, 1990). By training classifiers on oppositely labeled artificial examples, DECORATE reduces the correlation between ensemble members. Furthermore, the algorithm ensures that the *training* error of the ensemble is always less than or equal to the error of the base classifier; which usually results in a reduction of *generalization* error. This leads us to our first hypothesis:

Hypothesis 1: On average, combining the predictions of DECORATE ensembles will improve on the accuracy of the base classifier.

We believe that diversity is the key to constructing good ensembles, and is thus the basis of our approach. Other ensemble methods also encourage diversity, but in different ways. Bagging implicitly creates ensemble diversity, by training classifiers on different subsets of the data. Boosting fosters diversity, by explicitly modifying the distributions of the training data given to subsequent classifiers. We note, that both these methods rely solely on the *training* data for encouraging diversity. So when the size of the training set is small, they are limited in the amount of diversity they can produce. On the other hand, DECORATE ensures diversity on an arbitrarily large set of additional artificial examples, while still exploiting all the available training data. This leads us to our next hypothesis:

Hypothesis 2: DECORATE will outperform Bagging and AdaBoost low on the learning curve i.e. when training sets are small.

Chapter 4

Preliminary Experiments

In this chapter we present experiments comparing DECORATE with the leading ensemble methods, Bagging and AdaBoost. We also discuss some additional experiments that we ran to better understand DECORATE's performance.

4.1 Experimental Methodology

To evaluate the performance of DECORATE we ran experiments on 15 representative data sets from the UCI repository (Blake & Merz, 1998) that were used in similar studies (Webb, 2000; Quinlan, 1996a). The data sets are summarized in Table 4.1. Note that the datasets vary in the numbers of training examples, classes, numeric and nominal attributes; thus providing a diverse testbed.

We compared the performance of DECORATE to that of AdaBoost, Bagging and J48, using J48 as the base learner for the ensemble methods and using the Weka implementations of these methods (Witten & Frank, 1999). For the ensemble methods, we set the ensemble size to 15. Note that in the case of DECORATE we can only specify a *desired* ensemble size; the algorithm terminates if the number of iterations exceeds the maximum limit set even if the desired ensemble size is not reached. For our experiments, we set

Table 4.1: Summary of Data Sets

Name	Cases	Classes	Attributes	
			Numeric	Nominal
anneal	898	6	9	29
audio	226	6	–	69
autos	205	6	15	10
breast-w	699	2	9	–
credit-a	690	2	6	9
glass	214	6	9	–
heart-c	303	2	8	5
hepatitis	155	2	6	13
colic	368	2	10	12
iris	150	3	4	–
labor	57	2	8	8
lymph	148	4	–	18
segment	2310	7	19	–
soybean	683	19	–	35
splice	3190	3	–	62

the maximum number of iterations in DECORATE to 50. We ran experiments varying the amount of artificially generated data, R_{size}; and found that the results do not vary much for the range 0.5 to 1. However, R_{size} values lower than 0.5 do adversely affect DECORATE, because there is insufficient artificial data to give rise to high diversity. The results we report are for R_{size} set to 1, i.e. the number of artificially generated examples is equal to the training set size.

The performance of each learning algorithm was evaluated using 10 complete runs of 10-fold cross-validation. In each 10-fold cross-validation, each data set is randomly split into 10 equal-size segments and results are averaged over 10 trials. For each trial, one segment is set aside for testing, while the remaining data is available for training. To test performance on varying amounts of training data, learning curves were generated by testing the system after training on increasing subsets of the overall training data. Since we would like to summarize results over several data sets of different sizes, we select different *percentages* of the total training-set size as the points on the learning curve.

To compare two learning algorithms across all domains we employ the statistics used in (Webb, 2000), namely the win/draw/loss record and the geometric mean error ratio. The win/draw/loss record presents three values, the number of data sets for which algorithm A obtained better, equal, or worse performance than algorithm B with respect to classification accuracy. We also report the *statistically significant* win/draw/loss record; where a win or loss is only counted if the difference in values is determined to be significant at the 0.05 level by a paired t-test.

The geometric mean error ratio is defined as $\sqrt[n]{\prod_{i=1}^{n} \frac{E_A}{E_B}}$, where E_A and E_B are the mean errors of algorithm A and B on the same domain. If the geometric mean error ratio is less than one it implies that algorithm A performs better than B, and vice versa. We compute error ratios to capture the degree to which algorithms out-perform each other in win or loss outcomes.

4.1.1 Results

Our results are summarized in Tables 4.2-4.4. Each cell in the tables presents the accuracy of DECORATE versus another algorithm. If the difference is statistically significant, then the larger of the two is shown in bold. We varied the training set sizes from 1-100% of the total available data, with more points lower on the learning curve since this is where we expect to see the most difference between algorithms. The bottom of the tables provide summary statistics, as discussed above, for each of the points on the learning curve.

The results in Table 4.2 confirm our hypothesis that combining the predictions of DECORATE ensembles will, on average, improve the accuracy of the base classifier. DECORATE almost always does better than $J48$, producing considerable reduction in error throughout the learning curve.

DECORATE has more *significant* wins to losses over Bagging for all points along the learning curve (see Table 4.3). DECORATE also outperforms Bagging on the geometric mean ratio. This suggests that even in cases where Bagging beats DECORATE the improve-

ment is less than DECORATE's improvement on Bagging on the rest of the cases.

DECORATE outperforms AdaBoost early on the learning curve both on significant wins/draw/loss record and geometric mean ratio; however, the trend is reversed when given 75% or more of the data. Note that even with large amounts of training data, DECORATE's performance is quite competitive with AdaBoost - given 100% of the training data, DECORATE produces higher accuracies on 6 out of 15 data sets.

It has been observed in previous studies (Webb, 2000; Bauer & Kohavi, 1999) that while AdaBoost usually significantly reduces the error of the base learner, it occasionally increases it, often to a large extent. DECORATE does not have this problem as is clear from Table 4.2.

On many data sets, DECORATE achieves the same or higher accuracy as Bagging and AdaBoost with far fewer training examples. Figures 4.1 and 4.2 show learning curves that clearly demonstrate this point. Hence, in domains where little data is available or acquiring labels is expensive, DECORATE has an advantage over other ensemble methods.

4.2 Diversity versus Error Reduction

Our approach is based on the claim that ensemble diversity is critical to error reduction. We attempt to validate this claim by measuring the correlation between diversity and error reduction. We ran DECORATE at 10 different settings of R_{size} ranging from 0.1 to 1.0, thus varying the diversity of ensembles produced. We then compared the diversity of ensembles with the reduction in generalization error, by computing Spearman's rank correlation between the two. Diversity of an ensemble is computed as the mean diversity of the ensemble members (as given by Eq. 2.3). We compared ensemble diversity with the *ensemble error reduction*, i.e. the difference between the average error of the ensemble members and the error of the entire ensemble (as in (Cunningham & Carney, 2000)). We found that the correlation coefficient between diversity and ensemble error reduction is 0.6602 ($p \ll 10^{-50}$),

which is fairly strong.[1] Furthermore, we compared diversity with the *base error reduction*, i.e. the difference between the error of the base classifier and the ensemble error. The base error reduction gives a better indication of the improvement in performance of an ensemble over the base classifier. The correlation of diversity versus the base error reduction is 0.1607 ($p \ll 10^{-50}$). We note that even though this correlation is weak, it is still a *statistically significant* positive correlation. These results reinforce our belief that increasing ensemble diversity is a good approach to reducing generalization error.

4.3 Influence of Ensemble Size

To determine how the performance of DECORATE changes with ensemble size, we ran experiments with increasing sizes. We compared results for training on 20% of available data since the advantage of DECORATE is most noticeable low on the learning curve. The results were produced using 10-fold cross-validation. We present graphs of *accuracy* versus *ensemble size* for five representative datasets (see Figure 4.3). The performance on other datasets is similar. We note, in general, that the accuracy of DECORATE increases with ensemble size; though on most datasets, the performance levels out with an ensemble size of 10 to 25.

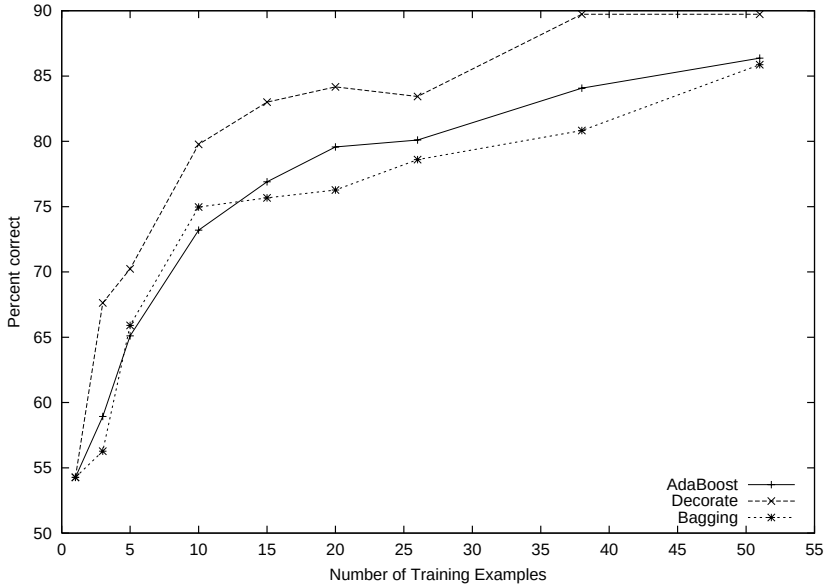
[1] The p-value is the probability of getting a correlation as large as the observed value by random chance, when the true correlation is zero.

Table 4.2: DECORATE vs J48

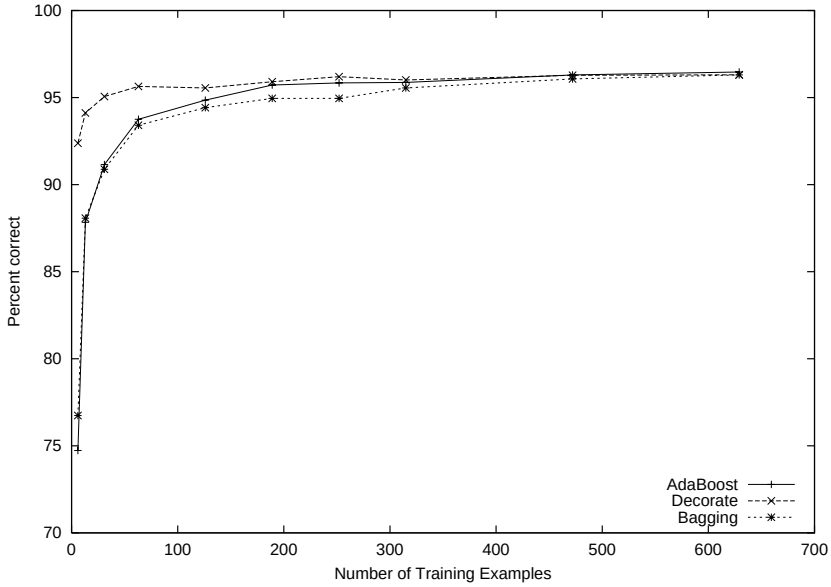
Dataset	1%	2%	5%	10%	20%	30%	40%	50%	75%	100%
anneal	**75.29**/72.49	**78.14**/75.31	**85.24**/82.08	**92.26**/89.28	**96.48**/95.57	**97.36**/96.47	**97.73**/97.3	**98.16**/97.93	98.39/98.35	98.71/98.55
audio	16.66/16.66	23.73/23.07	41.72/41.17	**55.42**/51.67	**64.09**/60.59	**67.62**/64.84	**70.46**/68.11	**72.82**/70.77	**77.8**/75.15	**82.1**/77.22
autos	24.33/24.33	29.6/29.01	**36.73**/34.37	42.89/41.22	52.2/50.53	**59.86**/53.92	**64.77**/59.68	**68.6**/65.24	**78**/73.15	**83.64**/81.72
breast-w	**92.38**/74.73	**94.12**/87.34	**95.06**/89.42	**95.64**/92.21	**95.55**/93.09	**95.91**/93.36	**96.2**/93.85	**96.01**/94.24	**96.28**/94.65	**96.31**/95.01
credit-a	71.78/69.54	74.83/**77.46**	80.61/81.57	83.09/82.35	84.38/84.29	84.68/84.59	**85.22**/84.41	**85.57**/84.78	85.61/85.43	85.93/85.57
Glass	31.69/31.69	**35.86**/32.96	**44.5**/38.34	**55.4**/46.62	**61.77**/54.16	**66.01**/60.63	**68.07**/61.38	**68.85**/63.69	**72.73**/67.53	**72.77**/67.77
heart-c	**58.66**/49.57	**65.11**/58.03	**73.55**/67.71	**75.05**/70.15	**77.66**/73.44	**78.34**/74.61	**79.09**/74.78	**79.46**/75.62	**78.74**/76.7	**78.48**/77.17
hepatitis	52.33/52.33	**71.95**/65.93	**76.59**/72.75	78.85/78.25	80.28/78.61	**81.14**/78.63	**81.53**/79.35	**81.68**/79.57	**82.37**/79.04	**82.43**/79.22
colic	**59.85**/52.85	**68.19**/65.31	74.91/74.37	78.45/79.94	81.81/82.71	82.47/83.41	82.74/83.55	83.5/**84.66**	83.93/**85.18**	85.24/85.16
iris	33.33/33.33	**50.87**/33.33	**80.67**/59.33	**91.27**/84.33	**93.07**/91.33	**94.4**/92.73	**95.07**/93	94.07/93.33	94.67/94.07	94.93/94.73
labor	54.27/54.27	54.27/54.27	**67.7**/58.93	**71.47**/64.77	**78.6**/70.07	**81.67**/73.7	**85.67**/75.17	**84.2**/75.8	**87.53**/77.4	**89.5**/78.8
lymph	48.39/48.39	**53.49**/46.64	**65.73**/60.39	**72.79**/68.21	**74.57**/70.79	**78.84**/73.58	**78.37**/74.53	**78.31**/73.34	**78.06**/75.63	**78.74**/76.06
segment	**67.94**/52.43	**80.75**/73.26	**89.52**/85.41	**92.87**/89.34	**94.99**/92.22	**95.82**/93.37	**96.54**/94.34	**96.93**/94.77	**97.56**/95.94	**98.02**/96.79
soybean	**19.37**/13.69	**32.12**/22.32	**55.55**/42.94	**73.51**/59.04	**84.63**/74.49	**88.52**/81.59	**90.37**/84.78	**91.35**/86.89	**92.85**/89.44	**93.81**/91.76
splice	**63.48**/59.92	67.56/**68.69**	77.34/77.49	82.62/82.58	**88.2**/87.98	90.46/90.44	91.82/91.77	92.5/92.4	93.41/93.47	93.92/**94.03**
Win/Draw/Loss	15/0/0	13/0/2	13/0/2	14/0/1	14/0/1	14/0/1	14/0/1	14/0/1	13/0/2	14/0/1
Sig. W/D/L	7/8/0	10/3/2	11/4/0	10/5/0	11/4/0	12/3/0	13/2/0	12/2/1	10/4/1	10/4/1
GM error ratio	0.858	0.8649	0.8116	0.8098	0.8269	0.8103	0.7983	0.8305	0.8317	0.8293

Table 4.3: DECORATE vs Bagging

Dataset	1%	2%	5%	10%	20%	30%	40%	50%	75%	100%
anneal	75.29/74.57	**78.14**/76.42	**85.24**/82.88	**92.26**/89.87	**96.48**/95.67	**97.36**/96.89	**97.73**/97.34	**98.16**/97.78	98.39/98.53	98.71/98.83
audio	**16.66**/12.98	23.73/23.68	**41.72**/38.55	**55.42**/51.34	**64.09**/61.76	67.62/66.9	70.46/70.29	72.82/73.07	77.8/77.32	**82.1**/80.71
autos	**24.33**/22.16	29.6/28	36.73/35.88	42.89/44.65	52.2/54.32	59.86/59.67	64.77/65.6	68.6/69.88	78/77.97	83.64/83.12
breast-w	**92.38**/76.74	**94.12**/88.07	**95.06**/90.88	**95.64**/93.41	**95.55**/94.42	**95.91**/94.95	**96.2**/94.95	**96.01**/95.55	96.28/96.07	96.31/96.3
credit-a	71.78/69.54	74.83/**77.99**	80.61/**82.58**	83.09/83.9	84.38/**85.13**	84.68/**85.78**	85.22/85.59	85.57/85.64	85.61/**86.12**	85.93/85.96
Glass	**31.69**/24.85	**35.86**/31.47	**44.5**/40.87	**55.4**/49.6	**61.77**/58.9	**66.01**/64.35	68.07/66.3	68.85/68.44	72.73/72	72.77/**74.67**
heart-c	**58.66**/50.56	**65.11**/55.67	**73.55**/68.77	**75.05**/73.17	**77.66**/76.12	78.34/77.9	79.09/78.44	79.46/79.11	78.74/79.05	78.48/78.68
hepatitis	52.33/52.33	**72.14**/63.18	76.8/75.2	79.48/78.64	80.7/80.42	81.81/81.07	81.65/81.22	**83.19**/81.06	**82.99**/80.87	82.62/81.34
colic	**58.37**/53.14	**66.58**/63.83	75.85/76.44	79.54/80.06	81.33/**83.04**	82.47/**83.58**	83.02/**83.98**	83.1/**84.47**	84.02/**85.4**	84.69/85.34
iris	33.33/33.33	**50.27**/33.33	**80.67**/60.47	**91.53**/81.4	**93.2**/90.67	**94.2**/92.33	**94.73**/92.87	**94.4**/93.6	94.53/94.47	94.67/94.73
labor	54.27/54.27	54.27/54.27	**67.63**/56.27	**70.23**/65.9	**79.77**/74.97	**83**/75.67	**84.17**/76.27	**83.43**/78.6	**89.73**/80.83	**89.73**/85.87
lymph	48.39/48.39	**53.62**/47.11	**65.06**/60.12	71.2/69.68	**76.74**/73.6	**78.84**/76.58	78.17/77.68	**78.99**/76.98	**79.14**/76.8	79.08/77.97
segment	**67.03**/55.88	**81.16**/76.36	**89.61**/87.42	**92.83**/91.01	**94.88**/93.4	**95.94**/94.65	**96.47**/95.26	**96.93**/95.82	**97.58**/96.78	**98.03**/97.41
soybean	**19.51**/14.56	**32.4**/24.58	**55.36**/47.46	**73.06**/65.45	**85.14**/79.29	**88.27**/85.05	**90.22**/87.89	**91.4**/89.22	**92.75**/91.56	**93.89**/92.71
splice	62.77/62.52	67.8/**72.36**	77.37/**80.5**	82.55/**85.44**	88.24/**89.5**	90.47/**91.44**	91.84/**92.4**	92.41/**93.07**	93.44/**94.06**	93.92/**94.53**
Win/Draw/Loss	15/0/0	13/0/2	12/0/3	11/0/4	11/0/4	12/0/3	11/0/4	10/0/5	10/0/5	8/0/7
Sig. W/D/L	8/7/0	10/3/2	10/3/2	9/5/1	10/2/3	8/4/3	6/7/2	8/5/2	5/7/3	4/9/2
GM error ratio	0.8727	0.8785	0.8552	0.8655	0.8995	0.9036	0.8979	0.9214	0.9312	0.9570

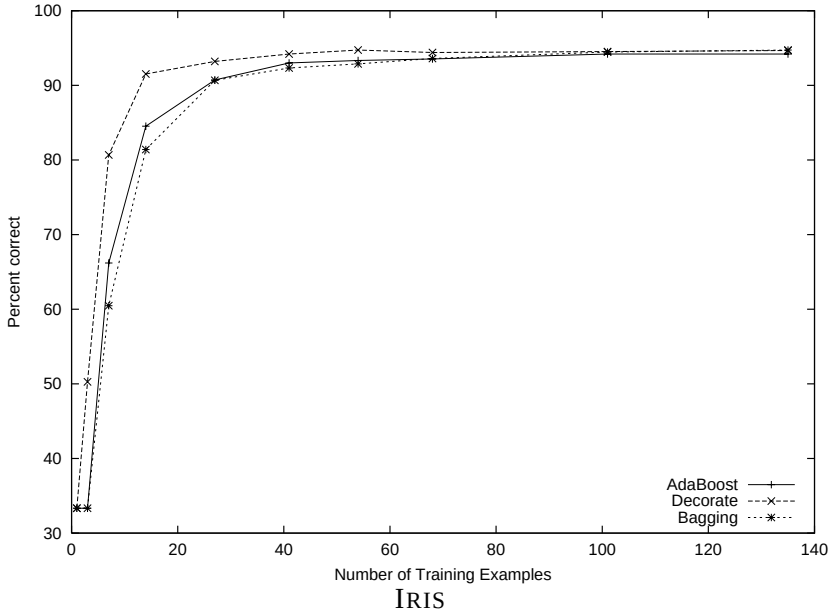


LABOR

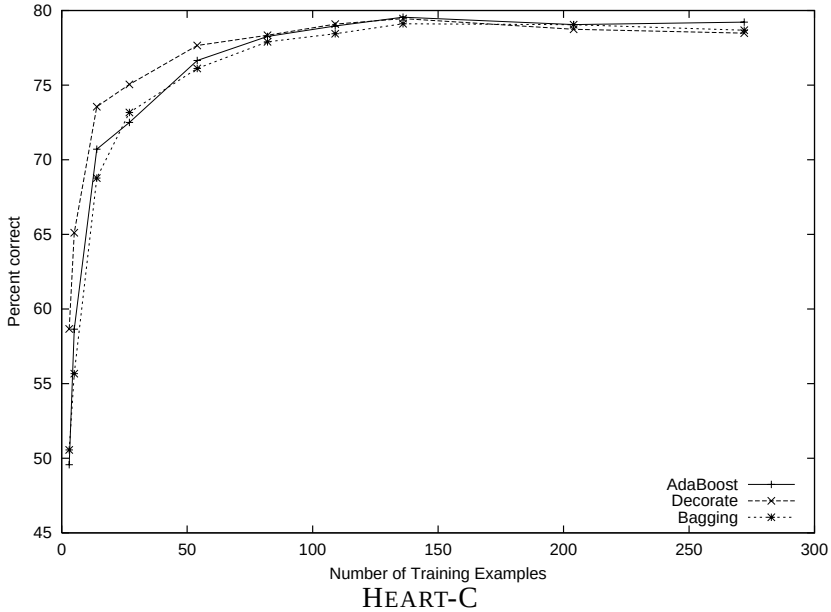


BREAST-W

Figure 4.1: DECORATE compared to AdaBoost and Bagging



IRIS



HEART-C

Figure 4.2: DECORATE compared to AdaBoost and Bagging

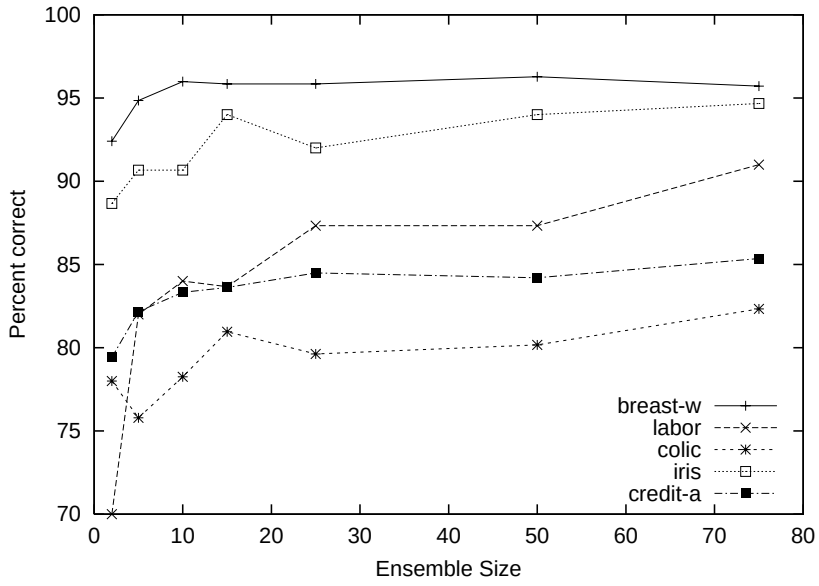


Figure 4.3: DECORATE at different ensemble sizes

Table 4.4: DECORATE vs AdaBoost

Dataset	1%	2%	5%	10%	20%	30%	40%	50%	75%	100%
anneal	**75.29**/73.02	78.14/77.12	85.24/**87.51**	92.26/**94.16**	96.48/**97.13**	97.36/**97.95**	97.73/**98.54**	98.16/**98.8**	98.39/**99.23**	98.71/**99.68**
audio	16.66/16.66	23.73/23.41	**41.72**/40.24	**55.42**/52.7	64.09/64.15	67.62/68.91	70.46/**73.07**	72.82/**75.92**	77.8/**81.74**	82.1/**84.52**
autos	24.33/24.33	29.6/29.71	36.73/34.2	42.89/43.28	52.2/**56.13**	59.86/**62.2**	64.77/**69.14**	68.6/**72.03**	78/**80.28**	83.64/**85.28**
breast-w	**92.38**/74.73	**94.12**/87.84	**95.06**/91.15	**95.64**/93.75	**95.55**/94.85	95.91/95.72	96.2/95.84	96.01/95.87	96.28/96.3	96.31/96.47
credit-a	**71.78**/68.8	74.83/75.3	80.61/79.68	**83.09**/81.14	**84.38**/83.04	84.68/84.22	**85.22**/84.13	**85.57**/84.58	85.61/84.93	85.93/85.42
Glass	31.69/31.69	**35.86**/32.93	**44.5**/40.71	**55.4**/49.78	**61.77**/58.03	66.01/64.33	68.07/66.93	68.85/68.69	72.73/**74.69**	72.77/**76.06**
heart-c	**58.66**/49.57	**65.11**/58.65	**73.55**/70.71	**75.05**/72.5	77.66/76.65	78.34/78.26	79.09/78.96	79.46/79.55	78.74/79.06	78.48/79.22
hepatitis	52.33/52.33	**72.14**/65.93	**76.8**/73.01	**79.48**/76.95	80.7/79.44	**81.81**/79.22	81.65/81.27	83.19/82.63	82.99/83.24	82.62/82.71
colic	**58.37**/52.85	66.58/67.18	**75.85**/72.85	**79.54**/77.17	**81.33**/79.36	**82.47**/79.24	**83.02**/79.51	**83.1**/80.22	**84.02**/80.59	**84.69**/81.93
iris	33.33/33.33	**50.27**/33.33	**80.67**/66.2	**91.53**/84.53	**93.2**/90.73	**94.2**/93	**94.73**/93.33	**94.4**/93.53	94.53/94.2	94.67/94.2
labor	54.27/54.27	54.27/54.27	**67.63**/58.93	**70.23**/65.1	**79.77**/73.2	**83**/76.9	**84.17**/79.57	**83.43**/80.1	**89.73**/84.07	**89.73**/86.37
lymph	48.39/48.39	**53.62**/46.64	**65.06**/60.54	71.2/69.57	**76.74**/74.16	78.84/78.62	78.17/**80.35**	78.99/79.88	79.14/**80.96**	79.08/**81.75**
segment	**67.03**/60.22	**81.16**/77.38	**89.61**/88.5	92.83/92.71	94.88/95.01	95.94/96.03	96.47/**96.9**	96.93/**97.23**	97.58/**98**	98.03/**98.34**
soybean	**19.51**/14.26	**32.4**/23.36	**55.36**/49.37	**73.06**/69.49	85.14/85.01	88.27/88.37	90.22/90.04	91.4/90.89	92.75/92.57	**93.89**/92.88
splice	62.77/**65.11**	67.8/**73.9**	77.37/**82.22**	82.55/**86.13**	88.24/88.27	90.47/89.82	**91.84**/90.8	**92.41**/90.78	**93.44**/92.63	93.92/93.59
Win/Draw/Loss	14/0/1	11/0/4	13/0/2	12/0/3	10/0/5	10/0/5	10/0/5	9/0/6	6/0/9	6/0/9
Sig. W/D/L	7/7/1	8/6/1	11/2/2	10/3/2	7/6/2	4/9/2	5/5/5	5/6/4	3/6/6	3/6/6
GM error ratio	0.8812	0.8937	0.8829	0.9104	0.9407	0.9598	0.9908	0.9957	1.0377	1.0964

Chapter 5

Proposed Work

Our preliminary experiments have demonstrated that applying DECORATE to decision trees generally results in greater classification accuracy than Bagging or Boosting (when trained on few labeled examples). We have also shown that the error reduction brought about by the use of DECORATE is due to the *diversity* of the committee that it generates. There are several other machine learning tasks that can benefit from using diverse ensembles. We will present each of these in the following sections, and discuss how we can use DECORATE to perform these tasks better.

5.1 Active Learning

5.1.1 Introduction

Most research in inductive learning has focused on learning from training examples that are randomly selected from the data distribution. On the other hand, in *active learning* (Cohn, Ghahramani, & Jordan, 1996) the learning algorithm exerts some control over which examples it is trained on. The most common form of active learning is *selective sampling*, in which, given a set of unlabeled examples, the system uses what it has already learned to select the most informative new examples to be labeled. Examples are considered infor-

mative if knowing their class label would reduce classification error over the distribution of examples.

Most real-world classification tasks require a large number of training examples, and often acquiring labels for examples is expensive or time-consuming. In general, *selective sampling* methods alleviate this problem, by reducing the number of training examples required to accurately learn a concept.

5.1.2 Related Work

Various active learning methods have been successfully applied to different classification learning schemes, such as neural networks (Davis & Hwang, 1992; Cohn et al., 1994), decision tree induction (Lewis & Catlett, 1994), Hidden Markov Models (Dagan & Engelson, 1995), SVMs (Tong & Koller, 2000; Brinker, 2003) and nearest neighbor classifiers (Lindenbaum, Markovitch, & Rusakov, 1999). There are several methods to perform active learning. The most popular among these methods are:

- *Uncertainty sampling* (Lewis & Gale, 1994), which selects examples on which the current learner has the greatest uncertainty in its predicted label;
- *Query-by-Committee* (QBC) (Seung, Opper, & Sompolinsky, 1992; Freund, Seung, Shamir, & Tishby, 1997), which selects examples on which a committee of learners most disagree; and
- *Estimation of error reduction* (Roy & McCallum, 2001; Lindenbaum et al., 1999), which selects examples, that once labeled and added to the training set, are expected to result in the lowest error on future test examples.

5.1.3 Future Work

We are most interested in *Query-by-Committee* methods for active learning. Freund et al. (1997) have shown that QBC converges to an optimal classifier more quickly than *uncer-*

tainty sampling. QBC attempts to reduce the error of the learner, by choosing an example to be labeled that will minimize the size of the *version space* (Mitchell, 1997) consistent with the training data. Instead of explicitly computing the size of the version space, QBC attempts to divide the version space into two parts of comparable size. To achieve this, it uses a committee of hypotheses sampled from the version space to predict labels of unlabeled examples; and then selects the example on which the committee disagrees most to be labeled by the user.

QBC provides a theoretical guarantee of a logarithmic reduction in the number of labeled training examples needed. However, for QBC to be successful it needs to maintain all possible hypotheses consistent with the training data (version space) in some form (Liere & Tadepalli, 1997). For most interesting hypothesis classes this is computationally intractable. As an alternative to QBC, Abe and Mamitsuka (1998) introduce *Query by Bagging* and *Query by Boosting*, where the committee of hypotheses consistent with the training data are created by Bagging and AdaBoost respectively. For both methods, the next query example is selected by picking a point on which the (weighted) majority voting by the committee of hypotheses has the least *margin*. Here the *margin* for an example is defined as the difference between the weight assigned to the label predicted by the ensemble and the maximal weight assigned to any single incorrect label.

Abe and Mamitsuka show that Query by Bagging and Query by Boosting achieves higher accuracies than both C4.5 and boosted C4.5 with random selection of examples. Given their success with combining active learning with ensemble methods, we plan to apply active learning to DECORATE as well. Extending DECORATE to incorporate active learning is quite straightforward (see Algorithm 3). We can define the *disagreement* of an ensemble C^*, on an example x, in terms of diversity:

$$dis(C^*, x) = \sum_{i=1}^{|C^*|} d_i(x)$$

Recalling the definition of diversity,

$$d_i(x) = \begin{cases} 0: \text{ if } C_i(x) = C^*(x) \\ 1: \text{otherwise} \end{cases}$$

We propose to implement *active*-DECORATE and compare it to DECORATE applied to J4.8 (with random selection). Given the success of other ensemble active learning approaches, we expect:

Hypothesis 5.1.1: *Active*-DECORATE will produce classifiers with the same accuracy as DECORATE using fewer training examples.

To do a comparative study, we will also implement *Query by Bagging* and *Query by Boosting*. Our preliminary experiments have demonstrated that DECORATE performs better than Bagging and Boosting, when training sets are small; which is when active learning is most useful. Therefore, we expect the following:

Hypothesis 5.1.2: *Active*-DECORATE will achieve steeper learning curves than *Query by Bagging* and *Query by Boosting*.

Algorithm 3 Active-DECORATE algorithm

Given:
T - set of training examples
U - set of unlabeled training examples
n - number of selective sampling iterations
$Params$ - set of DECORATE parameters, ($BaseLearn$, C_{size}, I_{max}, R_{size})

1. Repeat n times
2. $C^* = Decorate(T, Params)$, produce DECORATE ensemble
3. $x_j = \arg\max_{x \in U} dis(C^*, x)$, select example with the maximum disagreement
4. Label x_j
5. Remove x_j from U and add it to T
6. Return $Decorate(T, Params)$

5.2 Semi-supervised Learning

5.2.1 Introduction

In semi-supervised learning one tries to improve the accuracy of a supervised learner by exploiting the availability of a set of unlabeled examples (Muslea, 2002b). This is similar to active learning, in that, it also reduces the need for labeled training examples. Several semi-supervised algorithms work in the following way: first, they use the base learner L and a small set of labeled training examples T to learn an initial hypothesis h. Then h is applied to the examples in the unlabeled set U. Examples for which h can confidently generate labels are added to T. This process is repeated for a number of iterations.

Semi-supervised learning is based on the following intuition: even though the initial hypothesis h is learned based on a small training set, its highest confidence predictions are likely to be correct. Hence by adding these "high confidence" examples from U to the training set, we get more training data, and we may be able to learn a more accurate hypothesis. In turn, the more accurate hypothesis can be used to label more examples from U, and so on. A more detailed study of why unlabeled examples can be used to improve accuracy can be found in (Blum & Mitchell, 1998; Mitchell, 1999).

Algorithm 4 presents the outline of a typical semi-supervised learner. This framework captures the essence of most semi-supervised algorithms (Muslea, 2002b).

5.2.2 Related Work

There have been few approaches to exploiting unlabeled data for improving the accuracy of ensemble learners. Co-training (Blum & Mitchell, 1998) can be thought of a special case of semi-supervised ensemble learning, where the ensemble size is always two. Co-training requires that the examples in the domain can be described by two disjoint sets of features (*views*). The initial training set is used to learn a classifier in both views. Then each classifier is applied to the unlabeled set, and the examples on which each classifier makes

Algorithm 4 Semi-supervised Learner

Input:
$BaseLearn$ - base learning algorithm
T - set of labeled training examples
U - set of unlabeled examples
k - number of iterations to be performed
n - number of examples to be added at each iteration

1. Repeat k times
2. Train classifier $C = BaseLearn(T)$
3. Let MCP be the n examples in U for which C makes the most confident predictions
4. For each $x \in MCP$
5. Remove x from U
6. Add $< x, C(x) >$ to T

the most confident predictions are identified. These self-labeled examples are added to the training set and the process is repeated for a number of iterations. In the end, a hypothesis is constructed by voting the predictions made by the classifiers in each view. The drawback of co-training is that it relies on the assumption that the two views are *compatible* and *uncorrelated* (i.e. every example is identically labeled by the target concepts in each view; and, given the label of any example, its descriptions in each view are independent) (Muslea, 2002a; Nigam & Ghani, 2000).

Unlike co-training, ASSEMBLE (Bennett, Demiriz, & Maclin, 2002) can build semi-supervised ensembles of any size, and does not require the domain to have multiple *views*. ASSEMBLE incorporates *self-labeled* examples in a Boosting framework. In each iteration of ASSEMBLE examples from the unlabeled set are labeled by the current ensemble and added to the training set. Apart from labeling the unlabeled examples, the algorithm is similar to AdaBoost.

There have been other approaches to semi-supervised learning that do not use ensemble methods. Of note, are semi-supervised EM (Nigam, McCallum, Thrun, & Mitchell, 2000), transductive SVMs (Joachims, 1999), and semi-supervised SVMs (Bennett & Demiriz, 1999).

5.2.3 Future Work

Bennett et al. (2002) show how unlabeled data can be effectively combined in ensemble learning. We plan on implementing an approach similar to theirs; but while they use Boosting as the underlying ensemble learner, we will use DECORATE. Incorporating unlabeled data in DECORATE is quite straightforward: in each iteration of semi-supervised DECORATE we label examples from the unlabeled set, and add the n examples with the most confident predictions to the training set. We add the remaining unlabeled examples to the set of artificial training examples. The rest of the DECORATE algorithm remains unchanged (see Algorithm 5).

Given that exploiting unlabeled data in ASSEMBLE improves the classification accuracy of Boosting, we expect to see a similar improvement in results for DECORATE. Furthermore, since DECORATE generally outperforms Boosting given few training examples, we would expect to see similar results for the semi-supervised versions of both algorithms. These hypothesis can be summarized as follows:

Hypothesis 5.2.1: Using unlabeled data can increase the accuracy of classifiers produced by DECORATE.

Hypothesis 5.2.2: When labeled examples are scarce, semi-supervised DECORATE will produce more accurate classifiers than ASSEMBLE.

To verify the above hypotheses we will need to implement ASSEMBLE and semi-supervised DECORATE. We also propose to implement co-training to use as a baseline semi-supervised algorithm for comparison. Since most available datasets do not have multiple views, we will construct the two views required for co-training by randomly splitting

Algorithm 5 Semi-supervised DECORATE algorithm

Input:
$BaseLearn$ - base learning algorithm
T - set of m training examples $< (x_1, y_1), ..., (x_m, y_m) >$ with labels $y_j \in Y$
U - set of unlabeled training examples
n - number of unlabeled examples to use
C_{size} - desired ensemble size
I_{max} - maximum number of iterations to build an ensemble
R_{size} - factor that determines number of artificial examples to generate

1. $i = 1$
2. $trials = 1$
3. $C_i = BaseLearn(T)$
4. Initialize ensemble, $C^* = \{C_i\}$
5. Compute ensemble error, $\epsilon = \frac{\sum_{x_j \in T: C^*(x_j) \neq y_j} 1}{m}$
6. While $i < C_{size}$ and $trials < I_{max}$
7. Generate $R_{size} \times |T|$ artificial training examples, R
8. Label examples in U using C^*
9. Select the n most confident predictions from U and add them to T
10. Add the remaining unlabeled examples to R
11. Label examples in R with probability of class labels inversely proportional to C^*'s predictions
12. $T = T \bigcup R$
13. $C' = BaseLearn(T)$
14. $C^* = C^* \bigcup \{C'\}$
15. $T = T - R$, remove the artificial data
16. Compute training error, ϵ', of C^* as in step 5
17. If $\epsilon' \leq \epsilon$
18. $i = i + 1$
19. $\epsilon = \epsilon'$
20. otherwise,
21. $C^* = C^* - \{C'\}$
22. $trials = trials + 1$

feature sets. This approach to co-training has been used in (Nigam & Ghani, 2000). Since co-training's assumptions are violated in most domains, we expect that semi-supervised DECORATE should be able to outperform it. This leads us to our next hypothesis:

Hypothesis 5.2.3: Semi-supervised DECORATE will produce more accurate classifiers than co-training using random feature splits.

The use of unlabeled examples in DECORATE is based on the same intuition as in other semi-supervised learners; namely, that the most confident predictions of the base learner are likely to be right and adding these *high confidence* examples from the unlabeled set to the training set may increase accuracy. Apart from using the high confidence examples from the unlabeled set, DECORATE can readily exploit the remaining unlabeled data, by using this data the same way it uses artificial data to foster diversity in the ensemble. Hence having many unlabeled examples can also reduce the amount of artificial examples that need to be generated in each DECORATE iteration; thus reducing the training time. We propose to run experiments with semi-supervised DECORATE, varying the amount of unlabeled and artificial examples, to verify the following hypothesis:

Hypothesis 5.2.4: Using unlabeled data can reduce the amount of artificial training examples required by DECORATE to achieve a given accuracy.

5.3 Combining Active Learning with Semi-Supervision

Both *active learning* and *semi-supervised learning* can be viewed as methods that reduce the amount of labeled data required for supervised learning. Given a large set of unlabeled examples, we can combine both these frameworks for improved results.

In each iteration of active-DECORATE we use the current ensemble to select the most informative example to be labeled from the unlabeled data. However, in the active-DECORATE algorithm we described above, we only use the *labeled* data to produce the current ensemble. But, if possible, we should exploit the unlabeled data to produce more accurate ensembles at each iteration. This algorithm is a straightforward extension to active-

DECORATE (see Algorithm 6).

Algorithm 6 Active Semi-supervised DECORATE algorithm

Given:
T - set of training examples
U - set of unlabeled training examples
n - number of selective sampling iterations
$Params$ - set of DECORATE parameters, ($BaseLearn$, C_{size}, I_{max}, R_{size})

1. Repeat n times
2. $C^* = SemiSupDecorate(T, U, Params)$, produce semi-supervised DECORATE ensemble
3. $x_j = \arg\max_{x \in U} dis(C^*, x)$, select example with the maximum disagreement
4. Label x_j
5. Remove x_j from U and add it to T
6. Return $SemiSupDecorate(T, U, Params)$

We expect semi-supervised DECORATE and active-DECORATE to reinforce each other in the following way. By exploiting unlabeled data, semi-supervised DECORATE will improve the accuracy of the classifiers learned by active-DECORATE. And at the same time, active-DECORATE will improve semi-supervised DECORATE's accuracy by providing highly informative labeled examples. Depending on the success of both components, we conjecture the following:

Hypothesis 5.3.1: Active semi-supervised DECORATE will produce higher accuracies than both active-DECORATE and semi-supervised DECORATE, using the same number of training examples.

Similar combinations of semi-supervised and active learning have shown success in the past. McCallum and Nigam (1998) show that combining semi-supervised Expectation-Maximization (EM) and Query-by-Committee (QBC) performs better than both EM and QBC. Muslea (2002a) introduces Co-EMT, a combination of Co-Testing (Muslea, Minton,

& Knoblock, 2000) and Co-EM (Nigam & Ghani, 2000), which performs better than its components.

5.4 Improved Class Membership Probability Estimates

5.4.1 Introduction

In many supervised learning applications, it is not sufficient to predict the most likely class label for a test example. Quite often, what is needed is an accurate estimate of the probability of a test example belonging to each class.

Cost-sensitive learning is one area that stands to benefit from having accurate class probability estimates. Most learning algorithms for classification are designed to minimize *zero-one loss* i.e. the number of misclassified examples. This implicitly assumes that all errors are equally costly. However, in many domains, such as fraud detection, medical diagnosis, or intrusion detection, the cost of making a false positive error may be significantly different from making a false negative error (Margineantu & Dietterich, 2002). The area of *cost-sensitive learning* focuses on building classifiers that take into account these different misclassification costs. In general, misclassification costs may be represented as a cost matrix C, where $C(i, j)$ is the cost of predicting that an example belongs to class i when in fact it belongs to class j. According to decision theory, we should select a class to minimize the cost:

$$y = \arg\min_{i \in Y} \sum_{j} P(j|x)C(i, j)$$

To make an optimal decision we need to have accurate estimates of the class membership probabilities $P(j|x)$. Cost-sensitive learning is a well studied area; and a good bibliography can be found in (Turney, 1997).

5.4.2 Future Work

We believe that DECORATE not only improves classifier accuracy; but by averaging across an *accurate* and *diverse* ensemble, it also provides better class probability estimates than the base classifier. This leads us to the following:

Hypothesis 5.4.1: DECORATE improves the class membership probability estimates of the base classifier.

We plan on testing our hypothesis on the *donor-solicitation* task that was used in the data mining contest in KDD in 1998 (Bay, 2000). In this task we are given that the cost of requesting a donation from an individual x is \$0.68, and that the best estimate of the amount that x will donate, is $g(x)$. The goal of the task is to learn when to solicit x for a donation, such that our net revenue is maximized. The optimal decision-making strategy, would be to solicit x for a donation if and only if, $P(donor|x)g(x) > 0.68$, where $P(donor|x)$ is the estimate of the probability that x belongs to the class of donors. Clearly having good estimates of the class membership probabilities is critical for this task. We will compare the effectiveness of using class membership probabilities from J48 decision trees versus DECORATEd trees.

There have been several other studies that have focused on improving class probability estimates (Provost & Domingos, 2000; Zadrozny & Elkan, 2001, 2002; Margineantu & Dietterich, 2002; Saar-Tsechansky & Provost, 2001; Bennett, 2003). Based on the improvements we observe, we will consider comparing with some of these methods.

5.5 Regression

5.5.1 Introduction

So far we have focused on the *classification* task of learning the function $y = f(x)$, where the y values are drawn from a discrete set of classes $\{1, ..., K\}$. However, quite often domains require *regression*, where the y values are drawn from the real line. The typical

goal of regression is to learn a function that minimizes the *mean squared error* (MSE), defined as:

$$\sum_{x} (f_T(x) - f(x))^2$$

where $f(x)$ is the target function and $f_T(x)$ is the function learned based on the training set T. There have been many well-studied learning algorithms developed for regression (Hastie et al., 2001); but we are chiefly interested in methods that decrease error by using ensembles of regressors.

5.5.2 Related Work

There have been several approaches to applying ensemble methods to regression; most of which have focused on ensembles of neural networks (Liu & Yao, 1999; Rosen, 1996; Opitz & Shavlik, 1996; Krogh & Vedelsby, 1995; Brown & Wyatt, 2003). These ensemble approaches are tied to using neural networks as the base learner; whereas we are interested in developing methods that can be applied to improve the generalization accuracy of any learning algorithm. Drucker (1997) compared Bagging and Boosting techniques for regression. He applied standard Bagging (Breiman, 1996b) and a variation of *AdaBoost.R* (Freund & Schapire, 1996) to Breiman's Classification and Regression Trees (CART)(Breiman et al., 1984). From his experiments, he concluded that in most cases, Boosting produces improvements and is never statistically worse than Bagging.

5.5.3 Future Work

So far we have focused on using DECORATE for the classification task; however, it can easily be modified to solve regression problems. The algorithm will largely remain the same as Algorithm 2. The only step that requires modification is the labeling of artificial examples (step 8). This is because in regression, our target function maps to numbers on the real line, as opposed to a discrete set of class labels. But we still want to label artificial examples, such that training on them will increase the ensemble's diversity. In the case of

regression, we use the ensemble's variance around the mean, as our measure of *diversity*. So to give rise to increased diversity, we label each artificial example with $\mu \pm \alpha \times \sigma$; where μ and σ are the mean and standard deviation of the predictions of the current ensemble's members, and α is a factor that determines how much we want to increase the variance (values of $\alpha > 1$ will increase the variance).

As mentioned earlier, the diversity decomposition from (Krogh & Vedelsby, 1995), shows that the generalization error, E, of the ensemble can be expressed as $E = \bar{E} - \bar{D}$, where $\bar{E}$ and $\bar{D}$ are the mean error and diversity (variance) of the ensemble respectively. Therefore, by increasing the variance as described above we can ensure a drop in the ensemble's error.

To evaluate the effectiveness of DECORATE for regression, we will apply the approach described above to neural networks. We expect the following:

Hypothesis 5.5.1: A DECORATE ensemble of neural networks will produce significantly lower error than a single neural network.

To further evaluate the amount of error-reduction that DECORATE can bring about, we will compare our approach to Bagging and Boosting neural networks. We will run comparisons on the sets of synthetic data used in (Friedman, 1991) as well as a representative collection of real datasets from the UCI repository (Blake & Merz, 1998). Given the promising results in classification error-reduction, we expect to see similar improvements here:

Hypothesis 5.5.2: DECORATE will produces lower errors than Bagging and AdaBoost.R1 applied to neural networks; specially when training data is scarce.

Apart from neural networks, we would also like to test DECORATE's performance on other regressors, such as regression trees and SVMs. Time-permitting we will run these experiments too.

5.6 Relational Learning

5.6.1 Introduction

Most current machine learning techniques assume a "propositional" (a.k.a "feature vector" or "attribute value") representation of examples (Witten & Frank, 1999). *Relational learning* (RL) (Džeroski & Lavrač, 2001b), on the other hand, concerns learning from multiple relational tables that are richly connected. The most widely studied methods for inducing relational patterns are those in *inductive logic programming* (ILP)(Muggleton, 1992; Lavrac & Dzeroski, 1994). ILP concerns the induction of Horn-clause rules in first-order logic (i.e., logic programs) from data in first-order logic.

ILP is the study of learning methods for data and rules that are represented in first-order predicate logic. Predicate logic allows for quantified variables and relations and can represent concepts that are not expressible using examples described as feature vectors. A relational database can be easily translated into first-order logic and be used as a source of data for ILP (Wrobel, 2001). As an example, consider the following rules, written in Prolog syntax (where the conclusion appears first), that define the uncle relation:

```
uncle(X,Y) :- brother(X,Z),parent(Z,Y).
uncle(X,Y) :- husband(X,Z),sister(Z,W),
              parent(W,Y).
```

The goal of *inductive logic programming* (ILP) is to infer rules of this sort given a database of background facts and logical definitions of other relations (Muggleton, 1992; Lavrac & Dzeroski, 1994). For example, an ILP system can learn the above rules for uncle (the *target predicate*) given a set of positive and negative examples of uncle relationships and a set of facts for the relations parent, brother, sister, and husband (the *background predicates*) for the members of a given extended family, such as:

```
uncle(tom,frank), uncle(bob,john),
¬uncle(tom,cindy), ¬uncle(bob,tom)
```

```
parent(bob,frank), parent(cindy,frank),
parent(alice,john), parent(tom,john),
brother(tom,cindy), sister(cindy,tom),
husband(tom,alice), husband(bob,cindy).
```

Alternatively, rules that logically define the brother and sister relations could be supplied and these relationships inferred from a more complete set of facts about only the "basic" predicates: `parent`, `spouse`, and `gender`.

If-then rules in first-order logic are formally referred to as *Horn clauses*. A more formal definition of the ILP problem follows:

- **Given:**
  - Background knowledge, B, a set of Horn clauses.
  - Positive examples, P, a set of Horn clauses (typically ground literals).
  - Negative examples, N, a set of Horn clauses (typically ground literals).
- **Find:** A hypothesis, H, a set of Horn clauses such that:
  - $\forall p \in P : H \cup B \models p$ (completeness)
  - $\forall n \in N : H \cup B \not\models n$ (consistency)

Many algorithms for the ILP problem have been developed (Džeroski & Lavrač, 2001a) and applied to a variety of important data-mining problems (Džeroski, 2001). The algorithms that we will primarily work with are ALEPH (Srinivasan, 2001) and BETH (Tang, Mooney, & Melville, 2003).

5.6.2 Related Work

There have been few attempts at applying ensemble methods to first-order learning. Quinlan (1996b) applied Boosting to FFOIL, a first-order system that constructs definitions of functional relations. He found that Boosting produces modest improvements in accuracy

on 4 out of 5 datasets. Bagging has been applied to ALEPH and has been shown to produce sizeable improvements in results (Dutra, Page, Costa, & Shavlik, 2002; Mooney, Melville, Tang, Shavlik, de Castro Dutra, Page, & Costa, 2002). Hoche and Wrobel (2001) apply a variation of Boosting, *constrained confidence-rated Boosting*, to a weak ILP learner that they constructed. They claim that their weak learner can be boosted to perform comparably with more powerful ILP learners. Their focus is comprehensibility of results and efficiency of learning, and so their work is not directly relevant.

5.6.3 Future Work

Prior work has shown that using ensembles for first-order learning can improve accuracy of the base learner. So it would be interesting to see to what extent results may be improved by applying DECORATE to first-order learning. However, generating artificial training examples is not as straightforward for relational data as it is for feature-vector data. But, in some domains it is possible to acquire unlabeled training examples, e.g. the Contract-Killing domain (Mooney et al., 2002). We can exploit this unlabeled data to build DECORATE ensembles in an approach similar to that described in section 5.2.

We plan on implementing Bagging, Boosting and DECORATE using both ALEPH and BETH as base learners. We will compare the results of applying each of these ensemble methods on the Contract-Killing and Mutagenicity dataset (Srinivasan, Muggleton, Sternberg, & King, 1996). To evaluate each system we will run standard 10-fold cross-validation and generate learning curves. The promising results of applying DECORATE to propositional learning lead us to the following hypotheses:

Hypothesis 5.6.1: DECORATE applied to ALEPH and BETH will produce more accurate classifiers than the base classifiers at all points on the learning curve.

Hypothesis 5.6.2: When there is a paucity of training data DECORATE will achieve a higher accuracy than Boosting or Bagging applied to ALEPH and BETH.

5.7 Theoretical Issues

The empirical success of DECORATE in the classification task, raises the issue of the need for a sound theoretical understanding of its effectiveness. Here are a few of the theoretical questions that we would like to address:

- Can we prove that the DECORATE algorithm improves the bound on generalization error?
- How does ensemble *diversity* relate to bias and variance error decomposition (Domingos, 2000)?
- How does DECORATE relate to methods that attempt to maximize the margins on the training sample, such as AdaBoost (Schapire, Freund, Bartlett, & Lee, 1998)?

To answer these questions, we need to develop a strong theoretical framework for DECORATE.

Chapter 6

Conclusion

DECORATE provides strong evidence that diversity is a critical factor in constructing effective ensemble classifiers. By manipulating artificial training examples, DECORATE is able to use a strong base learner to produce an accurate, diverse ensemble. Preliminary experimental results demonstrate that our approach produces highly accurate ensembles that outperform both Bagging and Boosting low on the learning curve. Moreover, given large training sets, DECORATE outperforms Bagging and is competitive with Boosting. The goal of our proposed research is to show that DECORATE can also be effectively used for (1) active learning, (2) semi-supervised learning, (3) combining active learning with semi-supervision, (4) regression, (5) improving class membership probability estimates and (6) relational learning.

Bibliography

Abe, N., & Mamitsuka, H. (1998). Query learning strategies using boosting and bagging. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-98)*, pp. 1–10.

Abney, S., Schapire, R. E., & Singer, Y. (1999). Boosting applied to tagging and PP attachment. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning, 36*.

Bay, S. (2000). UCI KDD archive. Department of Information and Computer Sciences, University of California, Irvine. http://kdd.ics.uci.edu/.

Bennett, K., & Demiriz, A. (1999). Semi-supervised support vector machines. *Advances in Neural Information Processing Systems, 11*, 368–374.

Bennett, K., Demiriz, A., & Maclin, R. (2002). Exploiting unlabeled data in ensemble methods. In *Proceedings of the Eighth ACM SIGKDD Internation Conference on Knowledge Discovery and Data Mining*, pp. 289–296, Edmonton, Alberta, Canada.

Bennett, P. N. (2003). Using asymmetric distributions to improve text classifier probability estimates. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Toronto, Canada. ACM Press.

Blake, C. L., & Merz, C. J. (1998). UCI repository of machine learning databases. http://www.ics.uci.edu/˜mlearn/MLRepository.html.

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, Madison, WI.

Breiman, L. (1996a). Neural Information Processing Systems Workshop.

Breiman, L. (1996b). Bagging predictors. *Machine Learning*, *24*(2), 123–140.

Breiman, L., Friedman, J. H., Olshen, R., & Stone, C. (1984). *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA.

Brinker, K. (2003). Incorporating diversity in active learning with support vector machines. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pp. 59–66. AAAI Press.

Brown, G., & Wyatt, J. L. (2003). The Use of the Ambiguity Decomposition in Neural Network Ensemble Learning Methods. In Fawcett, T., & Mishra, N. (Eds.), *20th International Conference on Machine Learning (ICML'03)*, Washington DC, USA.

Cohn, D., Atlas, L., & Ladner, R. (1994). Improving generalization with active learning. *Machine Learning*, *15*(2), 201–221.

Cohn, D. A., Ghahramani, Z., & Jordan, M. I. (1996). Active learning with statistical models. *Journal of Artificial Intelligence Research*, *4*, 129–145.

Collins, M. (2002). Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL-02)*.

Cunningham, P., & Carney, J. (2000). Diversity versus quality in classification ensembles based on feature selection. In *11th European Conference on Machine Learning*, pp. 109–116.

Dagan, I., & Engelson, S. P. (1995). Committee-based sampling for training probabilistic classifiers. In *Proceedings of the Twelfth International Conference on Machine Learning (ICML-95)*, pp. 150–157, San Francisco, CA. Morgan Kaufmann.

Davis, D. T., & Hwang, J. N. (1992). Attentional focus training by boundary region data selection. In *International Joint Conference on Neural Networks*, Vol. 1, pp. 676–81. IEEE.

Dietterich, T. (2000). Ensemble methods in machine learning. In Kittler, J., & Roli, F. (Eds.), *First International Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science*, pp. 1–15. Springer-Verlag.

Dietterich, T. G. (1997). Machine learning research: Four current directions. *AI Magazine*, *18*(4), 97–136.

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, *40*(2), 139–157.

Domingos, P. (1997). Knowledge acquisition from examples via multiple models. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 98–106, Nashville, TN. Morgan Kaufmann.

Domingos, P. (2000). A unified bias-variance decomposition and its applications. In *Proc. 17th International Conf. on Machine Learning*, pp. 231–238. Morgan Kaufmann, San Francisco, CA.

Drucker, H. (1997). Improving regressors using boosting techniques. In *Proc. 14th International Conference on Machine Learning*, pp. 107–115. Morgan Kaufmann.

Dutra, I., Page, D., Costa, V. S., & Shavlik, J. (2002). An empirical evaluation of bagging in inductive logic programming. In *Proceedings of the Twelfth International Conference on Inductive Logic Programming*, pp. 48–65, Sydney, Australia.

Džeroski, S. (2001). Relational data mining applications: An overview. In Džeroski, S., & Lavrač, N. (Eds.), *Relational Data Mining*. Springer Verlag, Berlin.

Džeroski, S., & Lavrač, N. (2001a). An introduction to inductive logic programming. In Džeroski, S., & Lavrač, N. (Eds.), *Relational Data Mining*. Springer Verlag, Berlin.

Džeroski, S., & Lavrač, N. (Eds.). (2001b). *Relational Data Mining*. Springer Verlag, Berlin.

Freund, Y., Iyer, R., Schapire, R. E., & Singer, Y. (1998). An efficient boosting algorithm for combining preferences. In Shavlik, J. W. (Ed.), *Proceedings of ICML-98, 15th International Conference on Machine Learning*, pp. 170–178, Madison, US. Morgan Kaufmann Publishers, San Francisco, US.

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. In Saitta, L. (Ed.), *Proceedings of the Thirteenth International Conference on Machine Learning (ICML-96)*. Morgan Kaufmann.

Freund, Y., Seung, H. S., Shamir, E., & Tishby, N. (1997). Selective sampling using the query by committee algorithm. *Machine Learning*, *28*, 133–168.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics*, *19*, 1–141.

Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*(10), 993–1001.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The Elements of Statistical Learning*. Springer Verlag, New York.

Hoche, S., & Wrobel, S. (2001). Relational learning using constrained confidence-rated boosting. In *Proceedings of the 11th International Conference on Inductive Logic Programming*, pp. 51–64.

Iyer, R. D., Lewis, D. D., Schapire, R. E., Singer, Y., & Singhal, A. (2000). Boosting for document routing. In Agah, A., Callan, J., & Rundensteiner, E. (Eds.), *Proceedings of CIKM-00, 9th ACM International Conference on Information and Knowledge Management*, pp. 70–77, McLean, US. ACM Press, New York, US.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML-99)*, Bled, Slovenia.

Krogh, A., & Vedelsby, J. (1995). Neural network ensembles, cross validation and active learning. In *Advances in Neural Information Processing Systems 7*.

Kuncheva, L., & Whitaker, C. (2003). Measures of diversity in classifier ensembles and their relationship with ensemble accuracy. *Machine Learning*, pp. 181–207.

Lavrac, N., & Dzeroski, S. (1994). *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood.

Lewis, D., & Gale, W. (1994). A sequential algorithm for training text classifiers. In *Proc. of 17th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*.

Lewis, D. D., & Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the Eleventh International Conference on Machine Learning (ICML-94)*, pp. 148–156, San Francisco, CA. Morgan Kaufmann.

Liere, R., & Tadepalli, P. (1997). Active learning with committees for text categorization. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pp. 591–596, Providence, RI.

Lindenbaum, M., Markovitch, S., & Rusakov, D. (1999). Selective sampling for nearest neighbor classifiers. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pp. 366–371.

Liu, Y., & Yao, X. (1999). Ensemble learning via negative correlation. *Neural Networks*, *12*.

Maclin, R., & Opitz, D. (1997). An empirical evaluation of bagging and boosting. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, Providence, RI. AAAI Press.

Margineantu, D. D., & Dietterich, T. G. (2002). Improved class probability estimates from decision tree models. In Denison, D. D., Hansen, M. H., Holmes, C. C., Mallick, B., & Yu, B. (Eds.), *Nonlinear Estimation and Classification: Lecture Notes in Statistics*, 171, pp. 169–184. Springer-Verlag.

McCallum, A., & Nigam, K. (1998). Employing EM and pool-based active learning for text classification. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-98)*, Madison, WI. Morgan Kaufmann.

Melville, P., & Mooney, R. (2003). Constructing diverse classifier ensembles using artificial training examples. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pp. 505–510, Mexico.

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill, New York, NY.

Mitchell, T. (1999). The role of unlabeled data in supervised learning. In *Proceedings of the Sixth International Colloquium on Cognitive Science*.

Mooney, R. J., Melville, P., Tang, L. R., Shavlik, J., de Castro Dutra, I., Page, D., & Costa, V. S. (2002). Relational data mining with inductive logic programming for link discovery. In *Proceedings of the National Science Foundation Workshop on Next Generation Data Mining*.

Muggleton, S. H. (Ed.). (1992). *Inductive Logic Programming*. Academic Press, New York, NY.

Muslea, I., Minton, S., & Knoblock, C. A. (2000). Selective sampling with redundant views. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pp. 621–626.

Muslea, I. (2002a). Active + semi-supervised learning = robust multi-view learning. In *Proceedings of the 19th International Conference on Machine Learning (ICML 2002)*, pp. 435–442.

Muslea, I. (2002b). *Active learning with multiple views*. Ph.D. thesis, University of Southern California.

Nigam, K., McCallum, A. K., Thrun, S., & Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning*, *39*, 103–134.

Nigam, K., & Ghani, R. (2000). Analyzing the effectiveness and applicability of co-training. In *Proceedings of the Ninth International Conference on Information and Knowledge (CIKM-2000)*, pp. 86–93.

Opitz, D., & Shavlik, J. (1996). Actively searching for an effective neural-network ensemble. *Connection Science*, *8*.

Opitz, D. (1999). Feature selection for ensembles. In *Proceedings of 16th National Conference on Artificial Intelligence (AAAI)*, pp. 379–384.

Provost, F., & Domingos, P. (2000). Well-trained pets: Improving probability estimation trees. Tech. rep. IS-00-04, Stern School of Business, New York University.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo,CA.

Quinlan, J. R. (1996a). Bagging, boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pp. 725–730, Portland, OR.

Quinlan, R. (1996b). Boosting first-order learning. In *Proceedings of 7th International Workshop on Algorithmic Learning Theory*, pp. 143–155.

Rosen, B. (1996). Ensemble learning using decorrelated neural networks. *Connection Science*, *8*.

Roy, N., & McCallum, A. (2001). Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of 18th International Conference on Machine Learning (ICML-2001)*.

Saar-Tsechansky, M., & Provost, F. J. (2001). Active learning for class probability estimation and ranking. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-2001)*, pp. 911–920.

Schapire, R. E. (1999). Theoretical views of boosting and applications. In *Proceedings of the Tenth International Conference on Algorithmic Learning Theory*, pp. 13–25.

Schapire, R. E., Freund, Y., Bartlett, P., & Lee, W. S. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, *26*(5), 1651–1686.

Schapire, R. E., & Singer, Y. (2000). Boostexter: A boosting-based system for text categorization. *Machine Learning*, *39*(2/3), 135–168.

Schapire, R. E., Stone, P., McAllester, D., Littman, M. L., & Csirik, J. A. (2002). Modeling auction price uncertainty using boosting-based conditional density estimation. In *Proceedings of 19th International Conference on Machine Learning (ICML-2002)*.

Seung, H. S., Opper, M., & Sompolinsky, H. (1992). Query by committee. In *Proceedings of the ACM Workshop on Computational Learning Theory*, Pittsburgh, PA.

Srinivasan, A. (2001). *The Aleph manual*.

Srinivasan, A., Muggleton, S., Sternberg, M. J. E., & King, R. D. (1996). Theories for mutagenicity: A study in first-order and feature-based induction. *Artificial Intelligence*, *85*(1-2), 277–299.

Tang, L. R., Mooney, R. J., & Melville, P. (2003). Scaling up ILP to large examples: Results on link discovery for counter-terrorism. In *KDD-03 Workshop on Multi-Relational Data Mining*.

Tong, S., & Koller, D. (2000). Support vector machine active learning with applications to text classification. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000)*.

Tumer, K., & Oza, N. (1999). Decimated input ensembles for improved generalization. In *International Joint Conference on Neural Networks*.

Tumer, K., & Ghosh, J. (1996). Error correlation and error reduction in ensemble classifiers. *Connection Science*, *8*(3-4), 385–403.

Turney, P. (1997). Cost-sensitive learning bibliography. Online bibliograpy, Institute for Information Technology of the National Research Council of Canada, Ottawa, Canada. http://ai.iit.nrc.ca/bibliographies/cost-sensitive.html.

Webb, G. (2000). Multiboosting: A technique for combining boosting and wagging. *Machine Learning*, *40*.

Witten, I. H., & Frank, E. (1999). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco.

Wrobel, S. (2001). Inductive logic programming for knowledge discovery in databases. In Džeroski, S., & Lavrač, N. (Eds.), *Relational Data Mining*. Springer Verlag, Berlin.

Zadrozny, B., & Elkan, C. (2001). Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Proceedings of 18th International Conference on Machine Learning (ICML-2001)*, Williamstown, MA.

Zadrozny, B., & Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002)*, Edmonton, Alberta.

Zenobi, G., & Cunningham, P. (2001). Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error. In *Proceedings of the European Conference on Machine Learning*.