The Dissertation Committee for Prem Noel Melville
certifies that this is the approved version of the following dissertation:

Creating Diverse Ensemble Classifiers to Reduce Supervision

Committee:

Raymond J. Mooney, Supervisor

Benjamin Kuipers

Peter Stone

Joydeep Ghosh

Jude Shavlik

Creating Diverse Ensemble Classifiers to Reduce Supervision

by

Prem Noel Melville, B.A.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

December 2005

To my loving parents.

Acknowledgments

First and foremost, I would like to thank my advisor, Ray Mooney. Over the years, Ray has been an excellent mentor and a good friend. I could not have asked for a better balance of direction and freedom from an advisor. I appreciate his patience and support during my random walk through different research topics. Not only did he teach me the importance of scientific rigor, but also the subtle art of effectively presenting ideas. Ray has had a tremendous influence on the way I think and write about research.

I would like to express my gratitude to the members of my thesis committee — Jude Shavlik, Joydeep Ghosh, Peter Stone and Ben Kuipers. I enjoyed all my discussions with them, and appreciate the insightful feedback I received.

I have been very fortunate to have a terrific set of collaborators, who have all contributed to the work presented in this thesis. In particular, I would like to thank Maytal Saar-Tsechansky, Foster Provost, Nishit Shah, Lily Mihalkova, Stewart Yang and Yuk Lai Suen. It was truly a pleasure working with all of them. I would like to single out Maytal, for not only being a great researcher to collaborate with, but also for being a caring and concerned friend.

I would like to express my appreciation to all the members of the Machine Learning group, for the constant camaraderie and intellectual stimulation. I am glad I had the opportunity to interact with such a stellar group of individuals. In particular, I would like to thank Misha Bilenko, Sugato Basu, Lily Mihalkova, John Wong and Razvan Bunescu for providing valuable feedback on several papers. A special thanks goes to Misha, for being a superb friend, a great officemate and a wonderful sounding board for many ideas. Misha was particularly instrumental in helping me integrate various ideas into a coherent thesis. I will miss the crazy all-nighters in the office doing *Science* together.

I would also like to acknowledge some exemplary members of the department staff, Stacy Miller, Gloria Ramirez and Katherine Utz, for being extremely resourceful and always ready to help.

I am indebted to Joseph Modayil and Amol Nayate for being invaluable friends and excellent roommates, and for putting up with my all eccentricities. Joseph was also tremendously helpful in the preparation of countless presentations and documents (including these acknowledgments). I am grateful to several lovely creatures who have helped maintain my sanity during the writing of this thesis — Kim Chen, Shirley Birman, Anna Yurko and Nalini Belaramani.

Lastly, I would like to thank my parents and my brother, Pravin, for inspiring me and encouraging me to pursue my dreams.

The research in this thesis was supported by DARPA grants F30602-01-2-0571 and HR0011-04-1-007, and in part by the National Science Foundation under CISE Research Infrastructure Grant EIA-0303609.

PREM NOEL MELVILLE

The University of Texas at Austin

December 2005

Creating Diverse Ensemble Classifiers to Reduce Supervision

Publication No. ____________________

Prem Noel Melville, Ph.D.
The University of Texas at Austin, 2005

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. In this thesis, 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-generated training examples. The technique is a simple, general meta-learner that can use any strong learner as a base classifier to build diverse committees. The diverse ensembles produced by DECORATE are very effective for reducing the amount of supervision required for building accurate models. The first task we demonstrate this on is classification given a fixed train-

ing set. Experimental results using decision-tree induction as a base learner demonstrate that our approach consistently achieves higher predictive accuracy than the base classifier, Bagging and Random Forests. Also, DECORATE attains higher accuracy than Boosting on small training sets, and achieves comparable performance on larger training sets. Additional experiments demonstrate DECORATE's resilience to imperfections in data, in the form of missing features, classification noise, and feature noise.

DECORATE ensembles can also be used to reduce supervision through *active learning*, in which the learner selects the most informative examples from a pool of unlabeled examples, such that acquiring their labels will increase the accuracy of the classifier. Query by Committee is one effective approach to active learning in which disagreement within the ensemble of hypotheses is used to select examples for labeling. Query by Bagging and Query by Boosting are two practical implementations of this approach that use Bagging and Boosting respectively, to build the committees. For efficient active learning it is critical that the committee be made up of consistent hypotheses that are very different from each other. Since DECORATE explicitly builds such committees, it is well-suited for this task. We introduce a new algorithm, ACTIVEDECORATE, which uses DECORATE committees to select good training examples. Experimental results demonstrate that ACTIVEDECORATE typically requires labeling fewer examples to achieve the same accuracy as Query by Bagging and Query by Boosting. Apart from optimizing classification accuracy, in many applications, producing good class probability estimates is also important, e.g., in fraud detection, which has unequal misclassification costs. This thesis introduces a novel approach to active learning based on ACTIVEDECORATE which uses Jensen-Shannon divergence (a similarity measure for probability distributions) to improve the selection of training examples for optimizing probability estimation. Comprehensive experimental results demonstrate the benefits of our approach.

Unlike the active learning setting, in many learning problems the class labels for all instances are known, but feature values may be missing and can be acquired at a cost.

For building accurate predictive models, acquiring complete information for all instances is often quite expensive, while acquiring information for a random subset of instances may not be optimal. We formalize the task of *active feature-value acquisition*, which tries to reduce the cost of achieving a desired model accuracy by identifying instances for which obtaining complete information is most informative. We present an approach, based on DECORATE, in which instances are selected for acquisition based on the current model's accuracy and its confidence in the prediction. Experimental results demonstrate that our approach can induce accurate models using substantially fewer feature-value acquisitions than random sampling.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

For many predictive modeling tasks, acquiring supervised training data for building accurate classifiers (models) is often difficult or expensive. In some cases, the amount of available labeled training data is quite limited. In other cases, it may be possible to acquire additional data, but there is a significant cost of acquisition. Hence, it is important to be able to build accurate classifiers with limited data, or with the most cost-effective acquisition of additional data. We study this problem of learning with reduced supervision in the following three settings.

- **Passive Supervised Learning**

  Most of machine learning research has focused on this setting, where we are given a fixed set of training examples $\{(x_1, y_1), ..., (x_m, y_m)\}$ for some unknown function $y = f(x)$. The values of y are typically drawn from a discrete set of classes. A learning algorithm is trained on the set of training examples, to produce a classifier, which is a hypothesis about the true (target) function f. Given a new example x, the classifier predicts the corresponding y value. The aim of this classification task is to learn a classifier that minimizes the error in predictions on an independent test set of examples.

In some domains, there is inherently a limited amount of training data available, e.g., patient diagnostic data for a newly identified disease. In other domains, such as personalization, if the model learned does not produce accurate predictions with very little feedback (examples) from the user, then the user may stop using the system. In both these types of domains, it is important to be able to maximize the utility of small training sets. Hence, the first part of our study focuses on building accurate classifiers given limited training data.

- **Active Learning**

  In some domains, there are a large number of unlabeled examples available, that can be labeled at a cost. For instance, in the task of web page classification, it is easy to gain access to a large number of unlabeled web pages, but it takes some effort to provide class labels to each of these pages. In such settings, the learner can be used to select the most informative examples to be labeled, so that acquiring these labels will increase the accuracy of the current classifier. Actively selecting the most useful examples to train on is good approach to reducing the amount of supervision required for effective learning. The second part of this study focuses on this *active learning* setting (Cohn, Atlas, & Ladner, 1994).

- **Active Feature-value Acquisition**

  In many tasks, the class labels of instances are known, but they may be missing feature values that can be acquired at a cost. For example, online customer-profiling data may contain incomplete customer information that can be filled in by an intermediary. For building accurate predictive models, acquiring complete information for all instances is often prohibitively expensive, while acquiring information for a random subset of instances may not be most effective. The third part of this study introduces the task of *active feature-value acquisition* (Melville, Saar-Tsechansky, Provost, & Mooney, 2004), in which the learner tries to reduce the cost of achieving a desired model accuracy by identifying instances for which obtaining complete information is most informative.

The main contribution of this thesis is the development of a new method for building an ensemble of classifiers, that can be used to reduce the amount of supervision required in each of the above three settings. As a result, we are able to build more accurate predictive models than existing methods, at a lower costs of data acquisition.

1.1 The DECORATE Approach

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, ADABOOST (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, 1996) and ADABOOST which can be applied to any base learner to produce an effective committee (Witten & Frank, 1999).

This thesis presents 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 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.

In this thesis, we motivate the use of DECORATE for each of the three settings discussed in the previous section, and we provide empirical results that confirm its effectiveness. In particular, for the passive supervised setting, we show that when training data is limited, DECORATE produces more accurate classifiers than competing ensemble methods — Bagging, ADABOOST, and Random Forests (Breiman, 2001). In the active learning setting, experiments demonstrate that DECORATE ensembles perform very well at selecting the most informative examples to be labeled, so as to improve classification accuracy. Experiments on active feature-value acquisition, show that DECORATE can be used very effectively in making cost-effective decisions of the most informative instances for which to acquire missing feature values.

1.2 Thesis Outline

Below is a summary of the rest of the thesis:

- **Chapter 2. Background:** We provide a review of ensemble methods for classification, and describe some commonly-used ensemble approaches — Bagging, ADABOOST and Random Forests.

- **Chapter 3. The** DECORATE **Algorithm:** This chapter presents the details of our ensemble method DECORATE, and discusses some related approaches.

- **Chapter 4. Passive Supervised Learning:** In this chapter, we present experiments on the *passive* learning setting. It is shown that, when training data is limited, DECORATE outperforms Bagging, ADABOOST and Random Forests. Moreover, even on larger training sets, DECORATE performs better than Bagging and Random Forests, and is competitive with ADABOOST. This chapter also presents several additional studies analyzing the DECORATE algorithm.

- **Chapter 5. Imperfections in Data:** We compare the sensitivity of Bagging, ADABOOST, and DECORATE to three types of imperfect data: missing features, classification noise, and feature noise. Experimental results demonstrate the resilience of DECORATE to these imperfections in data.

- **Chapter 6. Active Learning for Classification Accuracy:** This chapter discusses the task of active learning, and presents an algorithm ACTIVEDECORATE, which uses DECORATE ensembles to reduce the number of labeled training examples required to achieve high classification accuracy. Extensive experimental results demonstrate that, in general, ACTIVEDECORATE outperforms other active learners — Query by Bagging and Query by Boosting (Abe & Mamitsuka, 1998).

- **Chapter 7. Active Learning for Class Probability Estimation:** In this chapter, we examine the task of active learning, when the objective is improving class probability estimation, as opposed to classification accuracy. We propose the use of Jensen-Shannon divergence as a measure of the utility of acquiring labeled examples. We improve on an existing active probability estimation method, and also extend ACTIVEDECORATE to effectively select training examples that improve class probability estimates.

- **Chapter 8. Active Feature-value Acquisition:** In this chapter, we present a general framework for the task of active feature-value acquisition. Within this framework, we propose a method that significantly outperforms alternative approaches. Experimental results using DECORATE demonstrate that our approach can induce accurate models using substantially fewer feature-value acquisitions as compared to the baseline.

- **Chapter 9. Future Work:** This chapter discusses future research directions for the work presented in this thesis.

- **Chapter 10. Conclusions:** In this chapter, we review the main contributions of our work.

This thesis introduces the DECORATE algorithm, which produces a diverse set of classifiers by manipulating artificial training examples. We demonstrate that the diverse ensembles produced by DECORATE can be used to learn accurate classifiers in settings where there is a limited amount of training data, and in *active* settings, where the learner can acquire class labels for unlabeled examples or additional feature-values for examples with missing values. As a result, we are able to build more accurate predictive models than existing methods, with reduced supervision, which translates to lower costs of data acquisition.

Chapter 2

Background

In this chapter, we provide a brief background on the supervised learning task and ensemble methods for classification. We also review some commonly-used ensemble approaches.

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 training instances.

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

In supervised learning, a learning algorithm is given a set of training examples or instances 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 learning 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* base learner. 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.

It the following sections, we present some ensemble approaches that are most relevant to this study. For an excellent survey on ensemble methods see (Dietterich, 2000).

2.2 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.

Bagging is typically applied to learning algorithms that are *unstable*, i.e., a small change in the training set leads to a noticeable change in the model produced. 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. Bagging of *stable* learners, such as Naive Bayes, does not reduce error.

2.3 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 by Freund and Schapire (1996) (see Algorithm 1). This algorithm assumes that the base learner can handle weighted examples. If the learner cannot directly handle weighted examples, then the training set can be sampled according to a weight distribution to produce a new training set to be used by the learner. ADABOOST maintains a set of weights over the training examples; and 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.

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, and is considered one of the best off-the-shelf classification methods (Hastie et al., 2001). 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

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

is consistent with the Boosting theory. Boosting also does not perform well when there is a large amount of classification noise (i.e. training examples with incorrect class labels) (Dietterich, 2000; Melville, Shah, Mihalkova, & Mooney, 2004).

2.4 Random Forests

Breiman (2001) introduces Random Forests, where he combines Bagging with random feature selection for decision trees. In this method, each member of the ensemble is trained on a bootstrap replicate as in Bagging. Decision trees are then grown by selecting the feature to split on at each node from F randomly selected features. In our experiments, following Breiman (2001), we set F to $\lfloor \log_2(k+1) \rfloor$, where k is the total number of features. And we also do not perform any pruning on the random trees.

Dietterich (2002) recommends Random Forests as the method of choice for decision trees, as it compares favorably to ADABOOST and works well even with noise in the training data. The focus of our work has been the development of ensemble methods that are *meta-learners*. Random Forests do not fall in this class, as they can only be applied to decision trees. However, as we applied our methods to tree induction we chose to also compare our results with Random Forests.

Chapter 3

The DECORATE Algorithm

In this chapter, we discuss the notion of ensemble diversity, and explain our algorithm DECORATE in detail. We also discuss other studies that are most closely related to our approach.

3.1 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.

3.1.1 Our diversity measure

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

$$d_i(x) = \begin{cases} 0: & \text{if } C_i(x) = C^*(x) \\ 1: & \text{otherwise} \end{cases} \tag{3.1}$$

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

$$\frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} d_i(x_j) \tag{3.2}$$

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.

3.2 DECORATE: Algorithm Definition

Melville and Mooney (2003, 2004a) 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. 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 should 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 training 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 $\hat{P}_{C_i,y}(x)$ is the estimated 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:

$$\hat{P}_y(x) = \frac{\sum\limits_{C_i \in C^*} \hat{P}_{C_i,y}(x)}{|C^*|}$$

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

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. $trials = trials + 1$

3.2.1 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 Section 4.6 we present experiments on alternate approaches to generating artificial data.

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 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 $\hat{P}_y(x)$, then a new label is selected based on the new distribution $\hat{P}'$, where:

$$\hat{P}'_y(x) = \frac{1/\hat{P}_y(x)}{\sum_y 1/\hat{P}_y(x)}$$

3.3 Why DECORATE Should Work

Ensembles of classifiers are often more accurate than their component classifiers if 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, using the predictions of a DECORATE ensemble 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. Random Forests produce diversity by training on different subsets of the data and feature sets. However, all 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, ADABOOST and Random Forests low on the learning curve i.e. when training sets are small.

We empirically validate these hypotheses in the following chapter.

3.4 Related Work

3.4.1 Explicit Diversity-Based Approaches

DECORATE differs from ensemble methods, such as Bagging, in that it *explicitly* tries to foster ensemble diversity. There have been other approaches to using diversity to guide ensemble creation. We list some of them below.

Liu and Yao (1999) and Rosen (1996) simultaneously train neural networks in an ensemble using a correlation penalty term in their error functions. McKay and Abbass

(2001) use a similar method with a different penalty function. 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 of features that are eliminated. This method, called *input decimation*, has been further explored by Tumer and 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, to the best of our knowledge, apart from Opitz (1999), none of the previous studies compared their methods with standard ensemble approaches such as Boosting and Bagging.

3.4.2 Use of Artificial Examples

One ensemble approach that also utilizes artificial training data is the active learning method introduced by Cohn et al. (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.

Craven and Shavlik (1995) use artificial examples to learn decision trees from trained neural networks. As in CMMs, the goal here is to create more comprehensible models from existing classifiers. The artificial examples created are labeled by a given neural network, and then used in constructing an equivalent decision tree.

To prevent overfitting in neural networks often noise is added to the inputs during training. This is generally done by adding a random vector to the feature vector of each training example. These *perturbed* or *jittered* examples may also be considered as artificial examples. Quite often training with noise improves network generalization (Bishop, 1995; Raviv & Intrator, 1996). Adding noise to training examples differs from our method of constructing examples from the data distribution. Furthermore, unlike adding noise, DECORATE systematically labels artificial examples to improve generalization.

Chapter 4

Passive Supervised Learning

In this chapter, we consider the *passive* supervised learning setting, where the training set is randomly sampled from the data distribution. In Chapters 6-8, we will look at different *active* settings, where the learner can influence the process of data acquisition. In the following sections, we present experiments comparing DECORATE with the leading ensemble methods, Bagging, AdaBoost and Random Forests. We also discuss several 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, Random Forests 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 meth-

Table 4.1: Summary of Data Sets

Name	Examples	Classes	Features	
			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

ods, 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 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.

4.2 Results

Our results are summarized in Tables 4.2-4.5. 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. To better visualize the results from the tables, we present scatter-plots in Figures 4.1-4.4. Each plot presents a comparison of DECORATE versus another learner for one point on the learning curve. Each point in the scatter-plot represents one of the 15 datasets. The points above the diagonal

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	**72.14**/65.93	**76.8**/72.75	79.48/78.25	**80.7**/78.61	**81.81**/78.63	**81.65**/79.35	**83.19**/79.57	**82.99**/79.04	**82.62**/79.22
colic	**58.37**/52.85	66.58/65.31	75.85/74.37	79.54/79.94	81.33/**82.71**	82.47/**83.41**	83.02/83.55	83.1/**84.66**	84.02/**85.18**	84.69/85.16
iris	33.33/33.33	**50.27**/33.33	**80.67**/59.33	**91.53**/84.33	**93.2**/91.33	**94.2**/92.73	**94.73**/93	**94.4**/93.33	94.53/94.07	94.67/94.73
labor	54.27/54.27	54.27/54.27	**67.63**/58.93	**70.23**/64.77	**79.77**/70.07	**83**/73.7	**84.17**/75.17	**83.43**/75.8	**89.73**/77.4	**89.73**/78.8
lymph	48.39/48.39	**53.62**/46.64	**65.06**/60.39	**71.2**/68.21	**76.74**/70.79	**78.84**/73.58	**78.17**/74.53	**78.99**/73.34	**79.14**/75.63	**79.08**/76.06
segment	**67.03**/52.43	**81.16**/73.26	**89.61**/85.41	**92.83**/89.34	**94.88**/92.22	**95.94**/93.37	**96.47**/94.34	**96.93**/94.77	**97.58**/95.94	**98.03**/96.79
soybean	**19.51**/13.69	**32.4**/22.32	**55.36**/42.94	**73.06**/59.04	**85.14**/74.49	**88.27**/81.59	**90.22**/84.78	**91.4**/86.89	**92.75**/89.44	**93.89**/91.76
splice	**62.77**/59.92	67.8/68.69	77.37/77.49	82.55/82.58	**88.24**/87.98	90.47/90.44	91.84/91.77	92.41/92.4	93.44/93.47	93.92/**94.03**
Win/Draw/Loss	15/0/0	13/0/2	13/0/2	13/0/2	14/0/1	14/0/1	14/0/1	14/0/1	13/0/2	12/0/3
Sig. W/D/L	7/8/0	9/5/1	11/4/0	10/5/0	12/2/1	12/2/1	13/2/0	13/1/1	10/4/1	10/4/1
GM error ratio	0.8627	0.8661	0.8099	0.8104	0.8172	0.8056	0.8081	0.8251	0.8173	0.8303

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

Table 4.4: DECORATE vs Random Forests

Dataset	1%	2%	5%	10%	20%	30%	40%	50%	75%	100%
anneal	**75.29**/72.07	**78.14**/76.69	**85.24**/84.21	**92.26**/90.89	**96.48**/95.71	97.36/**97.54**	97.73/**98.16**	98.16/**98.64**	98.39/**99.01**	98.71/**99.23**
audio	**16.66**/12.98	**23.73**/20.47	**41.72**/26.61	**55.42**/30.73	**64.09**/41.93	**67.62**/51.14	**70.46**/57.05	**72.82**/60.69	**77.8**/69.43	**82.1**/73.47
autos	**24.33**/22.16	29.6/**31.65**	36.73/**36.76**	42.89/**44.76**	52.2/**57.04**	59.86/**63.53**	64.77/**69.43**	68.6/**73.81**	78/**79.95**	83.64/**85.24**
breast-w	**92.38**/81.52	**94.12**/88.7	**95.06**/92.07	**95.64**/93.49	**95.55**/94.37	**95.91**/94.94	**96.2**/95.41	**96.01**/95.77	**96.28**/95.84	**96.31**/95.85
credit-a	**71.78**/60.61	**74.83**/64.65	**80.61**/70.38	**83.09**/72.87	**84.38**/76.55	**84.68**/78.36	**85.22**/79.54	**85.57**/81.13	**85.61**/82.35	**85.93**/83.25
glass	**31.69**/24.85	**35.86**/31.79	**44.5**/42.19	**55.4**/52.84	**61.77**/59.96	**66.01**/63.4	**68.07**/67.06	68.85/**69.14**	72.73/**73.55**	72.77/**76.4**
heart-c	**58.66**/50.06	**65.11**/54.78	**73.55**/66.86	**75.05**/72.61	**77.66**/76.14	**78.34**/76.52	**79.09**/77.63	**79.46**/78.58	78.74/**79.28**	78.48/**79.92**
hepatitis	**52.33**/52.33	**72.14**/70.36	**76.8**/74.51	**79.48**/77.26	**80.7**/80.37	**81.81**/81.7	**81.65**/81	**83.19**/81.72	82.99/**83.05**	82.62/**82.9**
colic	**58.37**/52.73	**66.58**/56.62	**75.85**/64.52	**79.54**/68.03	**81.33**/74.6	**82.47**/77.15	**83.02**/79.54	**83.1**/81	**84.02**/83.36	**84.69**/84.34
iris	**33.33**/33.33	**50.27**/47	**80.67**/67.07	**91.53**/83.33	**93.2**/91.13	**94.2**/94	**94.73**/94.47	**94.4**/94.33	**94.53**/94.4	**94.67**/94.2
labor	**54.27**/54.27	**54.27**/54.27	**67.63**/65.3	**70.23**/69.57	**79.77**/75.23	**83**/79.6	**84.17**/80.03	**83.43**/81.6	**89.73**/82.83	**89.73**/88.1
lymph	**48.39**/48.39	**53.62**/52.06	**65.06**/60.55	**71.2**/65.48	**76.74**/68.18	**78.84**/71.37	**78.17**/73.55	**78.99**/76.34	**79.14**/77.51	79.08/**79.28**
segment	**67.03**/59.46	**81.16**/74.16	**89.61**/86.45	**92.83**/91.25	**94.88**/94.16	**95.94**/95.42	**96.47**/95.99	**96.93**/96.39	**97.58**/97.18	**98.03**/97.59
soybean	19.51/**25.82**	32.4/**38.3**	**55.36**/54.57	**73.06**/66.52	**85.14**/78.4	**88.27**/83.94	**90.22**/87	**91.4**/88.54	**92.75**/90.73	**93.89**/91.38
splice	**62.77**/49.37	**67.8**/51.34	**77.37**/51.92	**82.55**/51.97	**88.24**/52.03	**90.47**/52.11	**91.84**/52.17	**92.41**/52.23	**93.44**/52.42	**93.92**/52.59
Win/Draw/Loss	14/0/1	13/0/2	14/0/1	14/0/1	14/0/1	13/0/2	13/0/2	12/0/3	10/0/5	9/0/6
Sig. W/D/L	10/4/1	8/6/1	10/5/0	13/2/0	11/3/1	10/4/1	10/3/2	7/6/2	7/6/2	6/5/4
GM error ratio	0.8603	0.8495	0.7814	0.7433	0.7486	0.7763	0.7915	0.8203	0.8171	0.8364

Table 4.5: 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

indicate that the accuracy of DECORATE is higher than the learner to which it is being compared. We present these plots comparing DECORATE with the other learners given 1% and 20% of the available training data.

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 error ratio. This suggests that even in cases where Bagging beats DECORATE the improvement is less than DECORATE's improvement on Bagging on the rest of the cases.

Similar results are observed in the comparison of DECORATE with Random Forests (see Table 4.4). DECORATE exhibits superior performance through out the learning curve on both wins/loss records as well as error ratios. The poor performance of Random Forests maybe because we are using only 15 trees. Random Forests may benefit from using larger ensembles; more so than other methods. However, to do a fair comparison we use the same ensemble size for all methods. Section 4.5 presents experiments on larger ensembles, which support our claims here.

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,

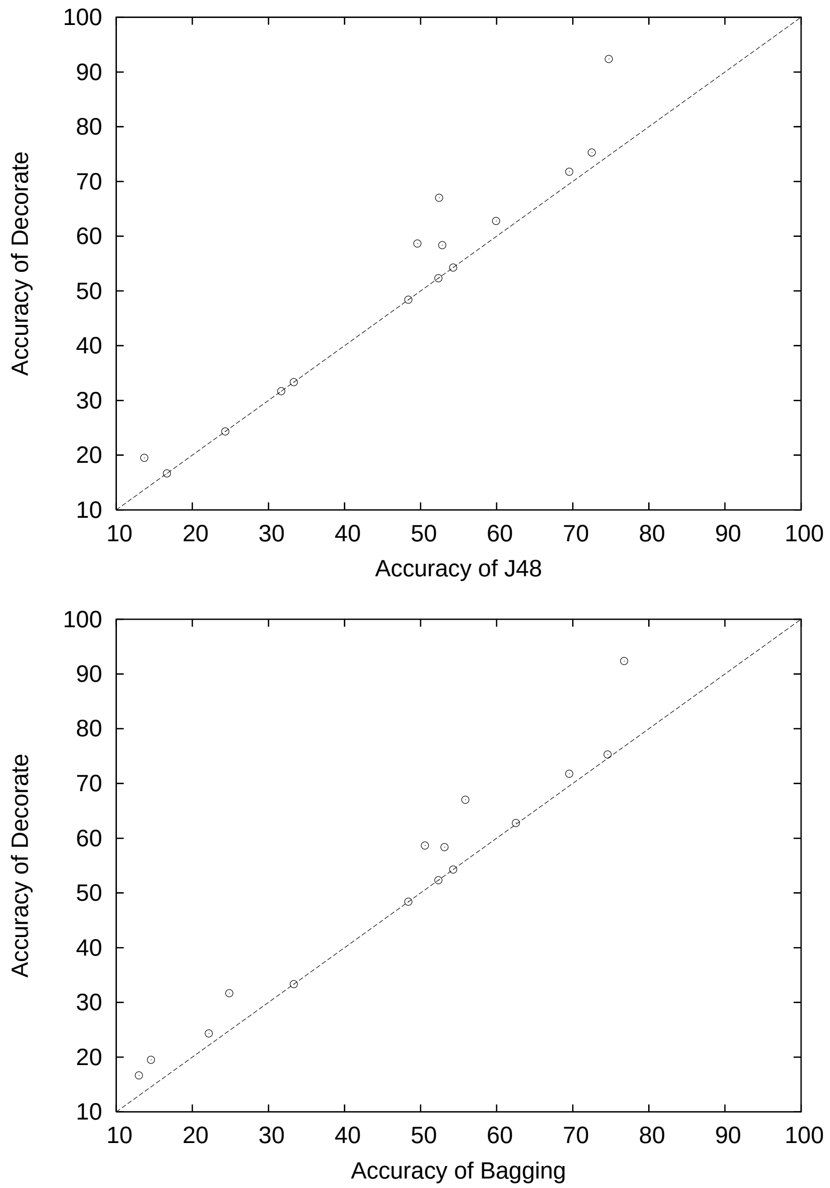


Figure 4.1: Comparing DECORATE with other learners on 15 datasets given 1% of the data.

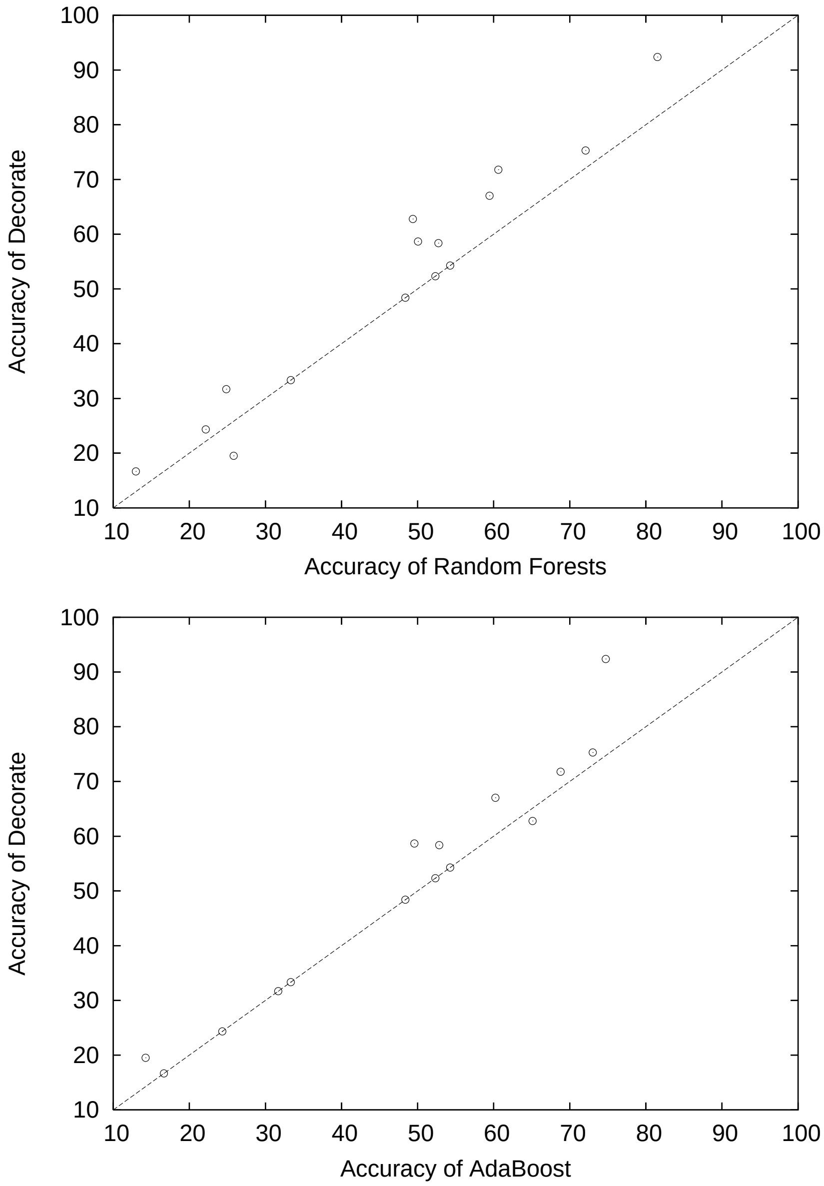


Figure 4.2: Comparing DECORATE with other learners on 15 datasets given 1% of the data.

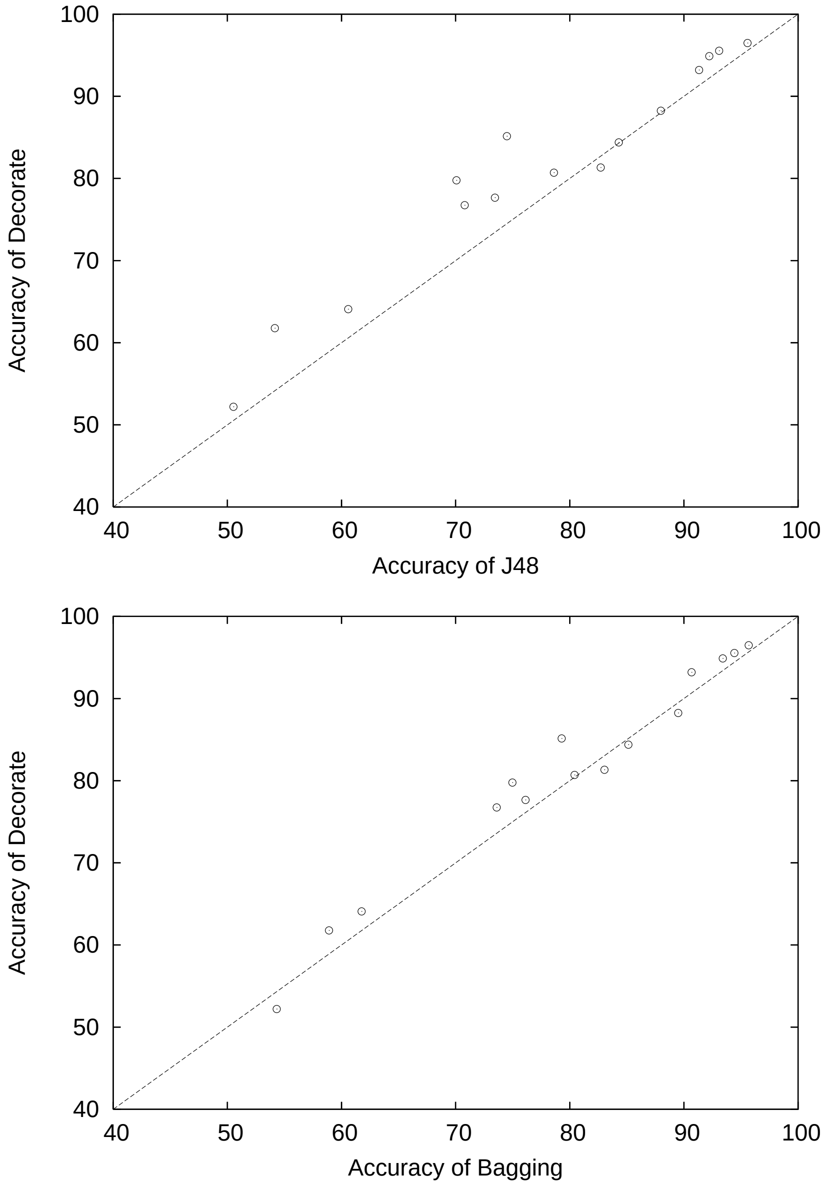


Figure 4.3: Comparing DECORATE with other learners on 15 datasets given 20% of the data.

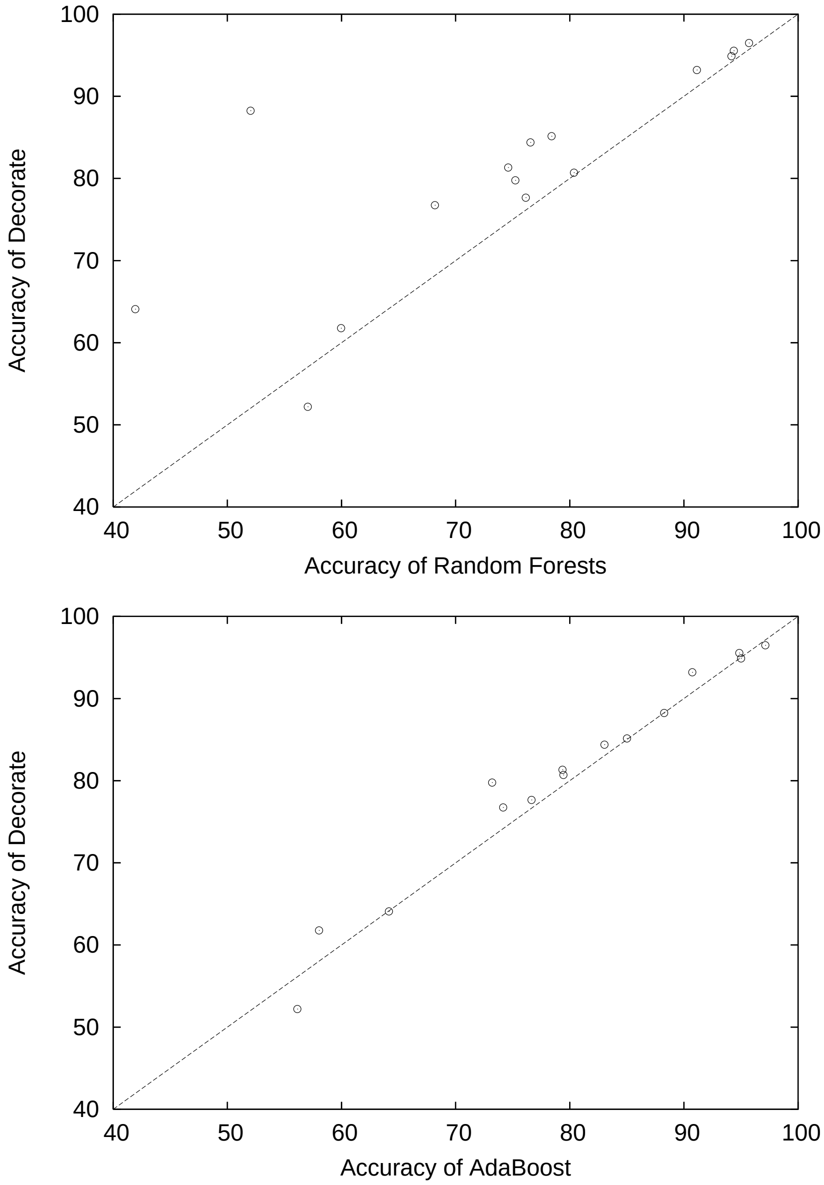


Figure 4.4: Comparing DECORATE with other learners on 15 datasets given 20% of the data.

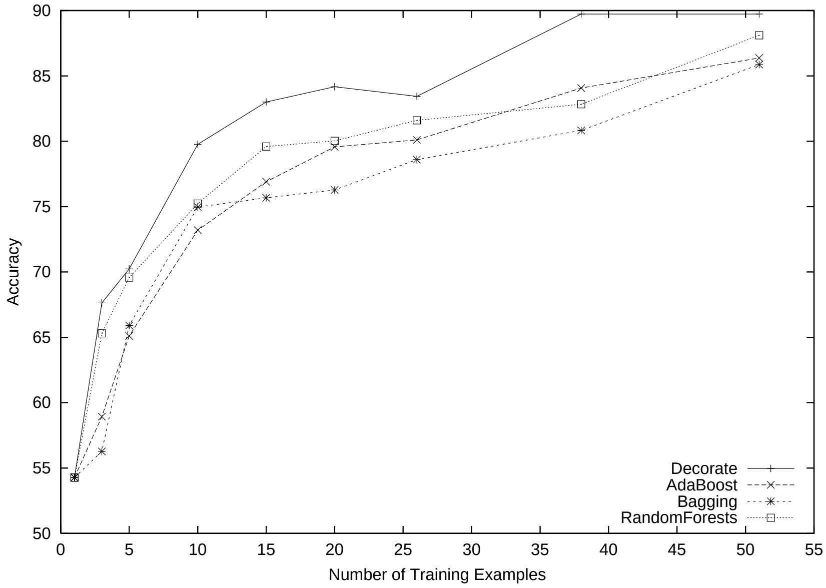
ADABOOST or Random Forests with far fewer training examples. Figures 4.5 and 4.6 show learning curves that clearly demonstrate this point. Hence, in domains where little data is available or acquiring labels is expensive, DECORATE has a significant advantage over other ensemble methods.

4.3 DECORATE with Large Training Sets

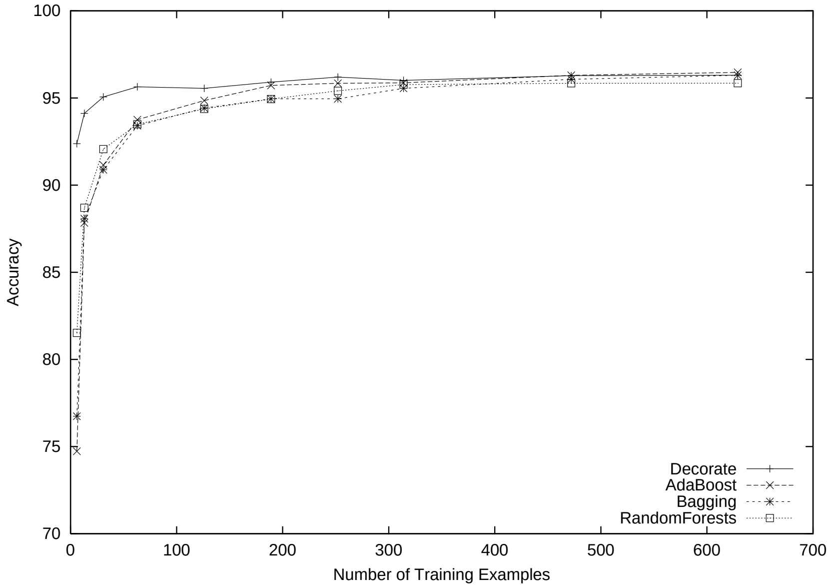
The learning curve evaluation clearly shows DECORATE's advantage when training sets are small. The results also indicate that DECORATE begins to lose out to ADABOOST with larger training sets. However, we claim that the performance of both systems on large training sets is comparable. To support this we ran additional experiments comparing DECORATE with ADABOOST on a larger collection of 33 UCI datasets. We ran 10 fold cross-validation using all the available training examples for each of the datasets. The results of this study are summarized in Table 4.6. We observe that on 25 of the 33 datasets there was no statistically significant difference between the two systems. And DECORATE significantly outperforms ADABOOST on four of the eight remaining datasets. We conjecture that when the training set is large enough the classifiers produced may be reaching the Bayes-optimal performance, which makes improvements impossible. Such a ceiling effect has been observed in other empirical comparisons of ensemble methods (Bauer & Kohavi, 1999). However, by looking at performance on varying training set sizes we can get a better understanding of the relative effectiveness of two learners. Therefore we strongly believe that generating learning curves is crucial for making a good comparison between systems.

4.4 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

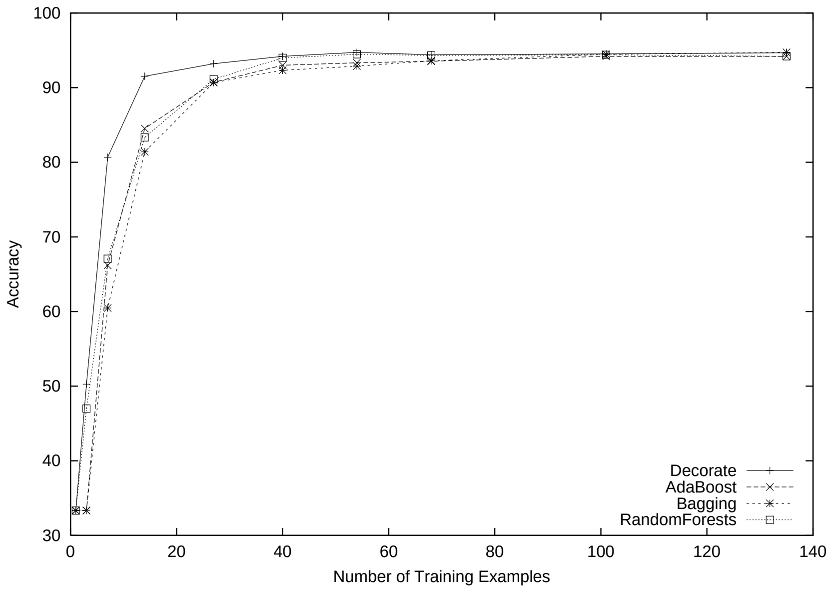


LABOR

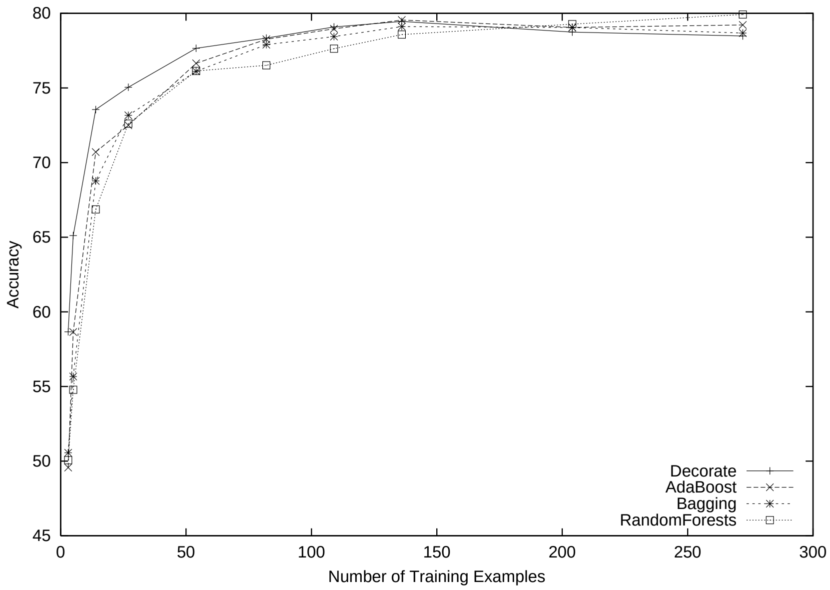


BREAST-W

Figure 4.5: DECORATE compared to ADABOOST, Bagging and Random Forests



IRIS



HEART-C

Figure 4.6: DECORATE compared to ADABOOST, Bagging and Random Forests

Dataset	AdaBoost	Decorate
audio	84.45	83.6
anneal	**99.55**	98.66
colic	83.13	85.58
balance-scale	78.56	**80.98**
credit-g	72.40	73.6
pima-diabetes	72.52	**75.52**
glass	76.58	72.34
heart-c	81.15	77.51
heart-h	78.56	79.98
credit-a	85.94	87.39
autos	86.33	85.79
kr-vs-kp	99.56	99.41
labor	88.33	83.00
lymph	82.43	78.29
mushroom	100.00	100.00
sonar	80.29	82.21
soybean	92.82	**94.58**
splice	93.17	93.89
vehicle	76.48	75.42
vote	95.17	95.18
vowel	93.94	**96.87**
breast-y	67.88	68.21
breast-w	96.42	96.85
heart-statlog	81.11	81.85
hepatitis	85.17	81.17
hypothyroid	**99.66**	98.6
ionosphere	93.75	92.6
iris	92.67	93.33
primary-tumor	40.09	44.53
segment	**98.57**	97.97
sick	**99.23**	98.49
waveform	81.58	80.92
zoo	96.18	94.18

Table 4.6: Decorate versus AdaBoost with large training sets

Table 4.7: Comparing ensemble diversity: Win-loss records.

	Number of Training Examples				
	10	15	20	25	30
Decorate vs Bagging	14-1	14-1	14-1	13-2	13-2
Decorate vs AdaBoost	15-0	14-1	14-1	14-1	14-1

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. 3.2). 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.

By exploiting artificial examples, the DECORATE algorithm forces the construction of a *diverse* set of hypotheses that are consistent with the training data. We believe that this ensemble diversity is the key to the success of DECORATE when training data is limited. We ran additional experiments to verify that DECORATE does indeed produce more diverse committees than Bagging or ADABOOST.

The diversity of each ensemble method was evaluated using 10-fold cross-validation on 15 UCI datasets. To test performance on varying amounts of data, each system was

[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 (Spatz & Johnston, 1984).

evaluated on the testing data, after training on increasing subsets of the training data. We focused on points early on the learning curve, where DECORATE is most effective. The results (Table 4.7) are summarized in terms of significant win/loss records; where a win or loss is only counted if the difference in *diversity* (not accuracy) is determined to be significant at the 0.05 level by a paired t-test. These results confirm that in most cases DECORATE does indeed produce significantly more diverse ensembles than Bagging or ADABOOST.

4.5 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 the 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 4.7). 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.

In our main results in Section 4.2 we used committees of size 15 for all methods. However, different ensemble methods may be affected to varying extents by committee size. To verify that the other ensemble methods are not being disadvantaged by smaller ensembles, we ran additional experiments with ensemble size set to 100. Learning curves were generated as in Section 4.1 on the four datasets presented in Figures 4.5 and 4.6. For these experiments, we set the maximum number of iterations in DECORATE to 300. The results of testing with larger ensembles is presented in Figures 4.8 and 4.9. Apart from slight improvements in accuracies for all methods, the trends of the results are the same as with ensembles of size 15.

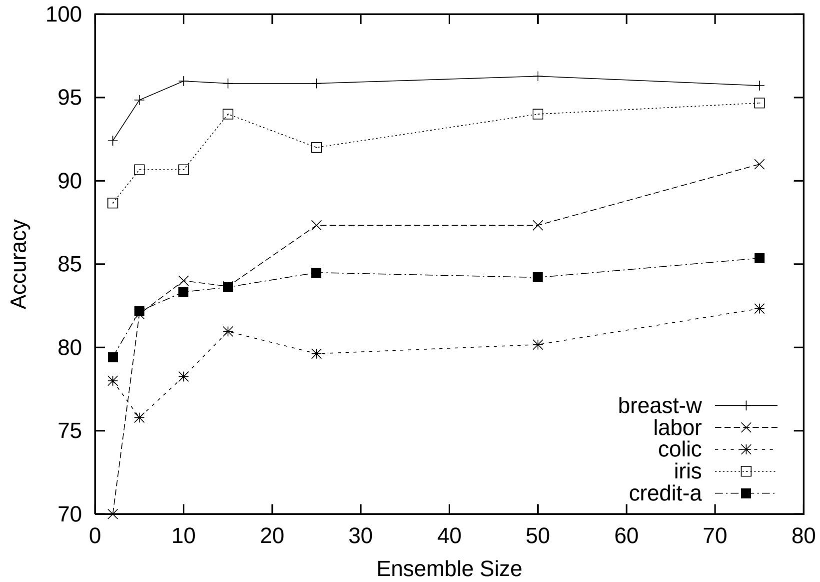
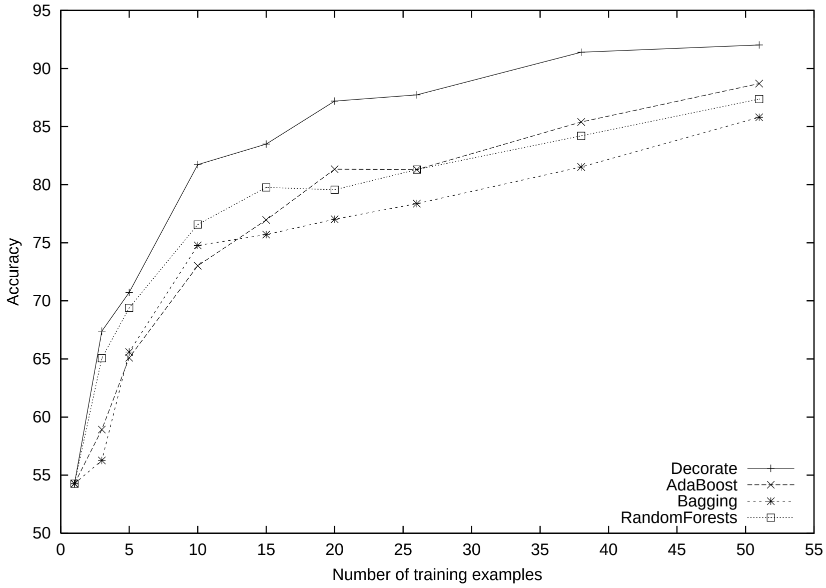


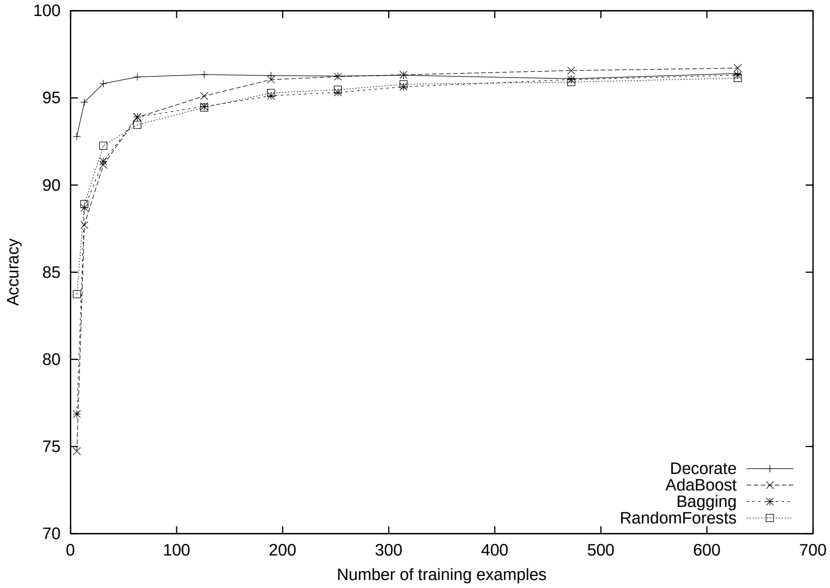
Figure 4.7: DECORATE at different ensemble sizes

4.6 Generation of Artificial Data

The DECORATE algorithm uses a fairly simple approach to generating artificial training examples. It generates feature values based on the training data distribution, assuming feature independence and making simple assumptions about the underlying models generating the data (Section 3.2.1). It is possible to model the data more accurately, but it is unclear if that is particularly beneficial. To verify this, we ran experiments using unlabeled *real* data in place of artificial data. Using unlabeled data corresponds to perfectly modeling the data, since they come from the same distribution as the training data. As a control experiment, we also tried relaxing our assumptions about the data, by assuming that all feature values come from uniform distributions. We describe these alternatives in more detail below.

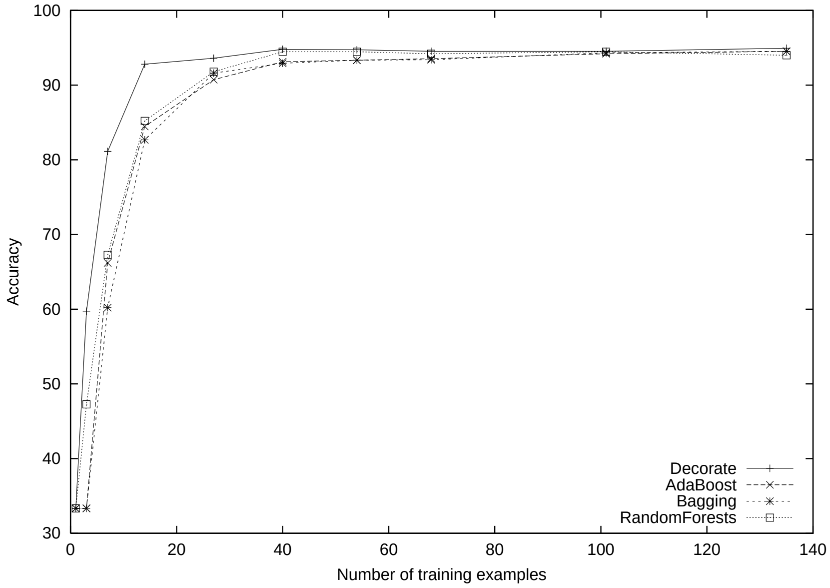


LABOR

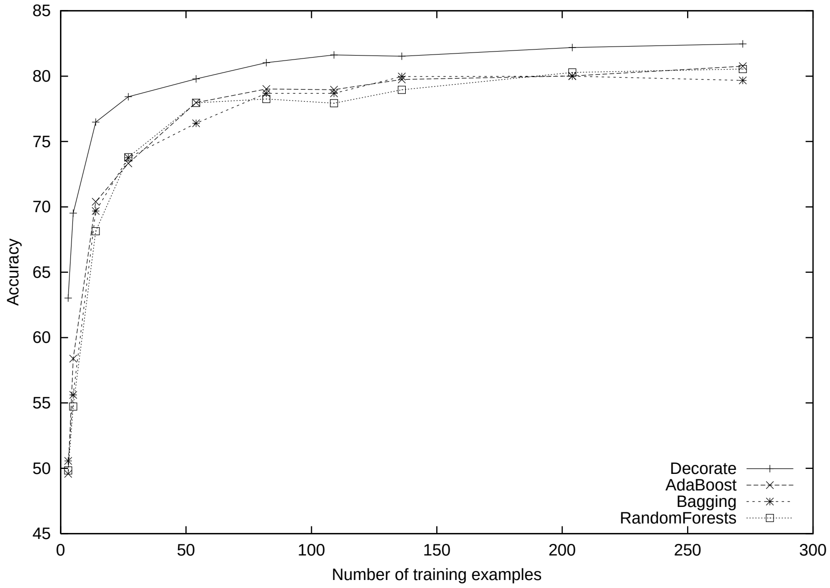


BREAST-W

Figure 4.8: Ensembles of size 100. DECORATE compared to ADABOOST, Bagging and Random Forests



IRIS



HEART-C

Figure 4.9: Ensembles of size 100. DECORATE compared to ADABOOST, Bagging and Random Forests

4.6.1 Using Unlabeled Data

In some domains, such as web page classification, we often have access to a large amount of unlabeled examples. In such domains, it is possible to exploit unlabeled data in place of artificial examples in the DECORATE algorithm. Unlike artificially-generated examples, there is not an infinite supply of unlabeled examples; so we need to slightly modify the DECORATE algorithm (see Algorithm 3). In the modified algorithm, we begin with a pool of unlabeled examples, and at each iteration, we sample a set of examples from this pool. This set of unlabeled examples is then used in the same way that artificial examples are used in DECORATE algorithm. We refer to this variation of the algorithm as DECORATE (Unlabeled). We compared this variation to the original DECORATE. In DECORATE we can generate an arbitrary amount of artificial examples, but in DECORATE (Unlabeled) we have a fixed amount of unlabeled examples to sample from. So to make a fair comparison, we implemented another version of DECORATE, which we call DECORATE (Sampled Artificial). In this approach, we initialize a fixed pool of artificial examples, and, similarly to DECORATE (Unlabeled), we sample from this pool at each iteration.

We ran experiments comparing the three versions of DECORATE and the base learner, J48. The performance of each algorithm was averaged over 10 runs of 10-fold cross-validation. In each fold of cross-validation, we generated learning curves in the following way. Initially, all examples are treated as unlabeled examples. For each point on the curve, a subset of the examples are randomly sampled and their true class labels are given to the learner. The remaining examples serve as the pool of unlabeled examples for DECORATE (Unlabeled). The performance of each learner is evaluated on increasing amounts of labeled training examples to produce points on the learning curve. Since there is a fixed set of available examples for each dataset, as the number of labeled examples increases, the size of the unlabeled set decreases. Hence, we only run these curves till 50% of the dataset is used as labeled examples, so that the rest may be used as the unlabeled pool.

The results of these experiments are summarized in Tables 4.8-4.10. The results

Algorithm 3 The DECORATE (Unlabeled) 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 examples
C_{size} - desired ensemble size
I_{max} - maximum number of iterations to build an ensemble
R_{size} - factor that determines number of additional examples to use

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.   Sample $R_{size} \times |T|$ examples with repetition from U, to give set R
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 additional 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. $trials = trials + 1$

show that unlabeled data can be used effectively in place of artificial data to produce ensembles that are more accurate than the base classifier. However, using artificially-generated data, both in DECORATE and DECORATE (Sampled Artificial), perform comparably or better than using unlabeled data. Unlabeled data has a slight disadvantage over artificial data, because if the current ensemble has high accuracy, then its predictions for the unlabeled data are likely to be correct. Flipping these labels may then make it difficult to find a hypothesis that is consistent with the training data and these new data. This does not happen as often in the case of artificial data, which are unlikely to contain mislabeled *real* examples. Nevertheless, when a large pool of unlabeled data is available, it can still be exploited in the DECORATE framework to improve over the base classifier. Using unlabeled data has the advantage that it is computationally less expensive than using artificial examples, since we avoid the step of generating artificial examples. The results also indicate that DECORATE and DECORATE (Sampled Artificial) exhibit similar performance on most datasets. The trends discussed above can be clearly seen in Figures 4.10-4.11.

4.6.2 Using Uniform Distributions

An alternate approach to generating artificial examples is to assume the feature values are sampled from uniform distributions. In this case, for a nominal feature we pick a value from the set of distinct values in its domain, selected uniformly at random. For a numeric feature, we select a random real number in the range defined by the minimum and maximum values observed in the training data. We refer to this version of DECORATE as DECORATE (Uniform).

Experiments were run as in Section 4.1, comparing DECORATE (Uniform), DECORATE and J48. The results are summarized in Tables 4.11-4.12. Generating artificial data assuming uniform distributions still produces significant improvements over the base classifer. For some datasets, the performance of DECORATE (Uniform) is similar to that of DECORATE, but on most others DECORATE performs better (see Figures 4.12 and 4.13).

Table 4.8: DECORATE(Unlabeled) vs. J48

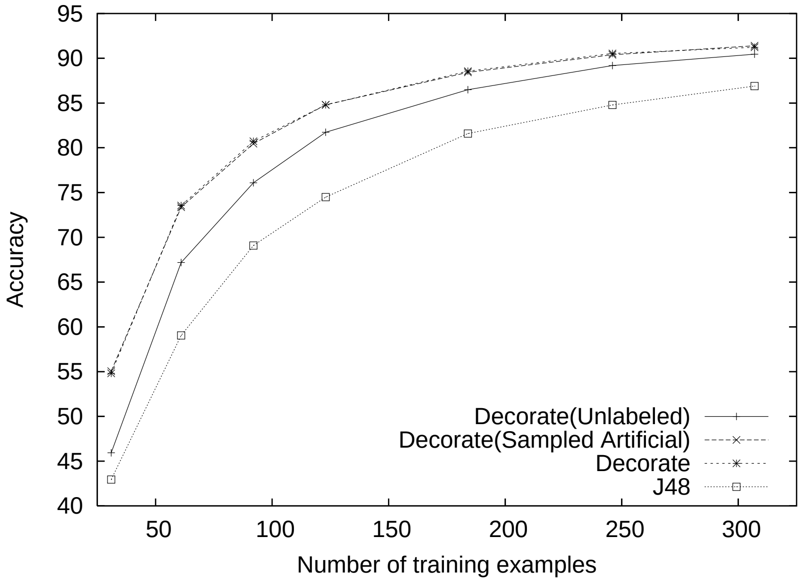
	5%	10%	15%	20%	30%	40%	50%
Win/Draw/Loss	9/0/6	7/0/8	9/0/6	11/0/4	11/0/4	12/0/3	12/0/3
Sig. W/D/L	6/6/3	4/9/2	5/8/2	5/8/2	6/8/1	5/9/1	8/6/1
GM error ratio	0.9337	0.9726	0.9675	0.9664	0.9426	0.9187	0.9234

Table 4.9: DECORATE(Unlabeled) vs. DECORATE (Sampled Artificial)

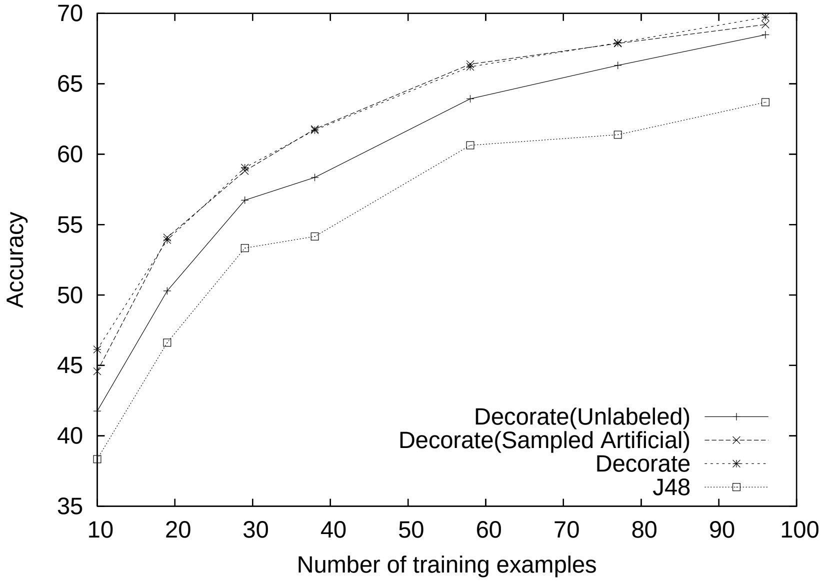
	5%	10%	15%	20%	30%	40%	50%
Win/Draw/Loss	2/0/13	2/0/13	2/0/13	1/0/14	2/0/13	2/0/13	2/0/13
Sig. W/D/L	0/8/7	0/6/9	0/5/10	0/6/9	0/8/7	0/8/7	0/10/5
GM error ratio	1.151	1.1733	1.2003	1.1857	1.170	1.1357	1.1095

Table 4.10: DECORATE(Unlabeled) vs. DECORATE

	5%	10%	15%	20%	30%	40%	50%
Win/Draw/Loss	2/0/13	3/0/12	1/0/14	0/0/15	1/0/14	1/0/14	3/0/12
Sig. W/D/L	0/6/9	0/6/9	0/5/10	0/6/9	0/6/9	0/8/7	0/9/6
GM error ratio	1.0066	1.024	1.0054	1.0061	0.9997	1.0215	1.0167

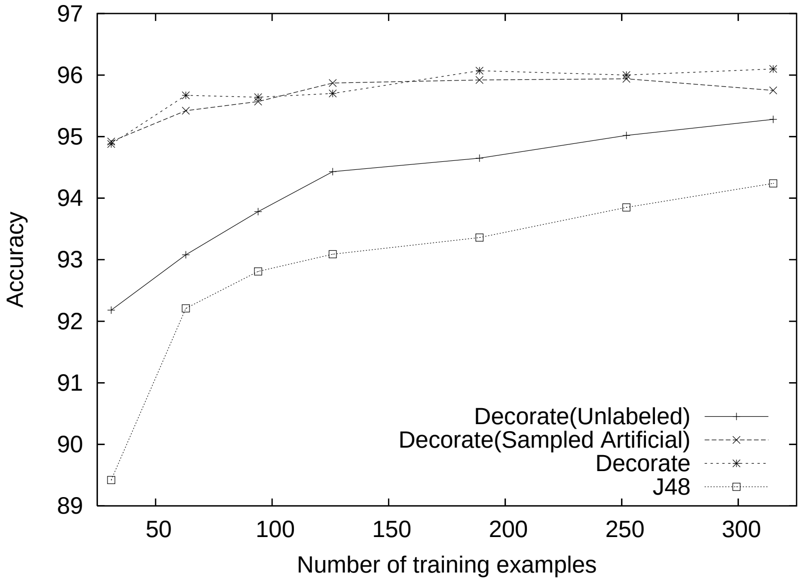


SOYBEAN

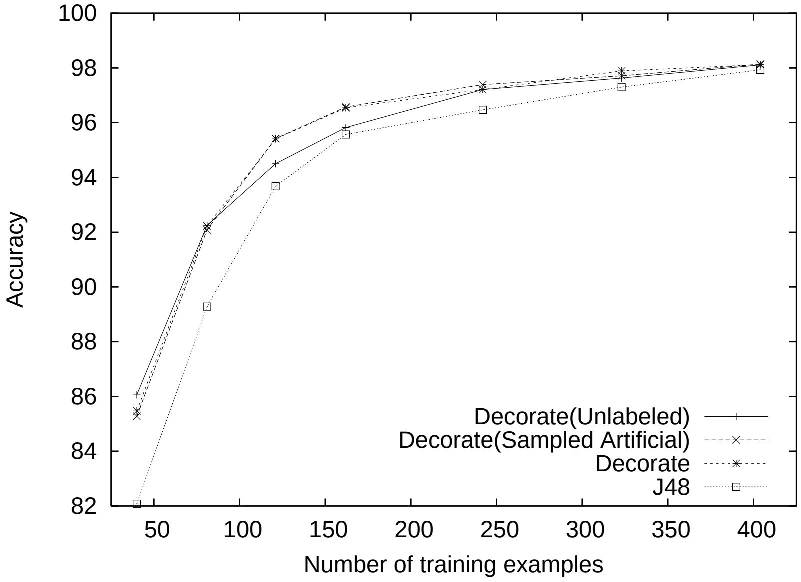


GLASS

Figure 4.10: Comparing the use of unlabeled examples versus artificial examples in DECORATE.



BREAST-W



ANNEAL

Figure 4.11: Comparing the use of unlabeled examples versus artificial examples in DECORATE.

The results show that, when generating artificial data, it is beneficial not to make very relaxed assumptions about the data, as in the case of uniform distributions; but it is also less effective to perfectly model the data, as in the case of unlabeled examples. Using an intermediate level of data modeling, as done in DECORATE, seems to work the best.

4.7 Importance of the Rejection Criterion

In building ensembles of classifiers, there is usually a tradeoff between diversity and average error of ensemble members. As such, in DECORATE, we try to foster diversity, while still maintaining the overall ensemble's accuracy. We do this by rejecting a new classifier if adding it to the existing ensemble decreases its training accuracy. To test the importance of this rejection criterion, we conducted an ablation study, in which we created a version of DECORATE without the rejection criterion, i.e., we excised steps 13-18 from Algorithm 2. We refer to this version of the algorithm as DECORATE (No Rejection). Experiments were run as in Section 4.1, and our results are summarized in Tables 4.13-4.16. We see that for most datasets removing the rejection criterion does not significantly hurt the performance of DECORATE (see, e.g., Figure 4.14(a)). However, on *splice* (Figure 4.14(b)), we see that DECORATE (No Rejection) performs very poorly compared to DECORATE. Therefore, having the rejection criterion is a good safety mechanism to guard against the unlikely event that DECORATE introduces too much diversity at the cost of generalization accuracy.

4.8 Experiments on Neural Networks

Since DECORATE is a meta-learning algorithm, it can be applied to any base learner to produce an ensemble of classifiers. In most experiments on DECORATE we have used decision tree induction as the base learner. To verify that our results generalize to other base learners, we ran additional experiments using neural networks. Specifically, we used the Weka implementation of neural networks, which uses backpropagation learning (Witten

Table 4.11: DECORATE(Uniform) vs. J48

	1%	2%	5%	10%	20%	30%	40%	50%	75%	100%
Win/Draw/Loss	14/0/1	14/0/1	13/0/2	14/0/1	15/0/0	14/0/1	14/0/1	13/0/2	14/0/1	12/0/3
Sig. W/D/L	9/5/1	12/3/0	10/5/0	11/4/0	11/4/0	12/3/0	11/4/0	11/4/0	11/4/0	9/5/1
GM error ratio	0.868	0.8623	0.8373	0.8423	0.8447	0.8583	0.845	0.8781	0.857	0.8889

Table 4.12: DECORATE(Uniform) vs. DECORATE

	1%	2%	5%	10%	20%	30%	40%	50%	75%	100%
Win/Draw/Loss	8/0/7	6/0/9	6/0/9	6/0/9	7/0/8	3/0/12	4/0/11	5/0/10	5/0/10	3/0/12
Sig. W/D/L	0/12/3	3/9/3	0/13/2	2/9/4	1/9/5	0/10/5	0/9/6	1/7/7	3/7/5	0/6/9
GM error ratio	1.0131	1.0222	1.039	1.0405	1.0427	1.065	1.067	1.0727	1.0377	1.0975

Table 4.13: DECORATE(No Rejection) vs. DECORATE

	1%	2%	5%	10%	20%	30%	40%	50%	75%	100%
Sig. W/D/L	0/15/0	0/15/0	2/11/2	0/12/3	0/11/4	0/11/4	1/11/3	2/12/1	0/14/1	1/11/3
Win/Draw/Loss	11/0/4	7/0/8	10/0/5	5/0/10	5/0/10	4/0/11	7/0/8	7/0/8	5/0/10	7/0/8
GM error ratio	1.0005	1.0014	1.0151	1.0278	1.0251	1.0263	1.0224	1.0182	1.0189	1.014

Table 4.14: DECORATE(No Rejection) vs. J48

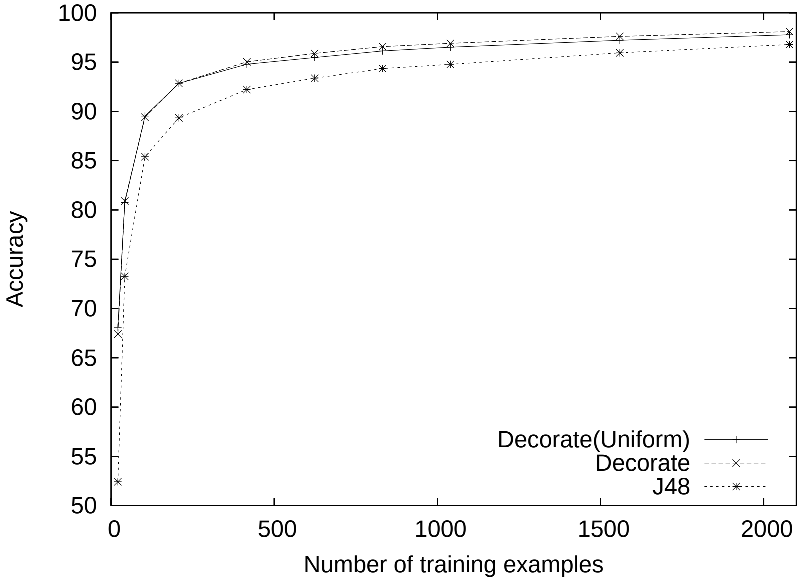
	1%	2%	5%	10%	20%	30%	40%	50%	75%	100%
Sig. W/D/L	9/5/1	10/3/2	11/3/1	11/2/2	12/2/1	13/1/1	13/1/1	11/2/2	11/2/2	10/4/1
Win/Draw/Loss	14/0/1	13/0/2	14/0/1	13/0/2	13/0/2	13/0/2	13/0/2	13/0/2	12/0/3	12/0/3
GM error ratio	0.9036	0.8801	0.9284	0.9679	0.9751	0.9806	0.9788	0.9818	0.9869	0.9825

Table 4.15: DECORATE(No Rejection) vs. Bagging

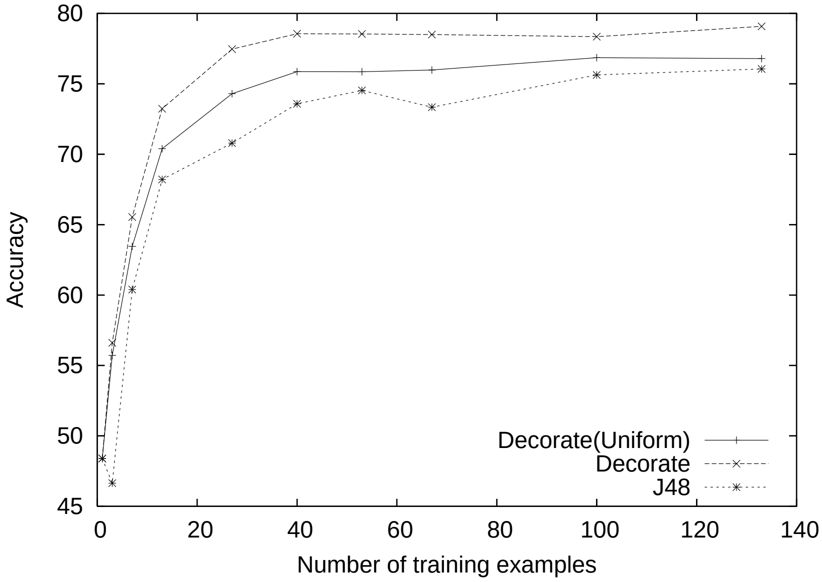
	1%	2%	5%	10%	20%	30%	40%	50%	75%	100%
Sig. W/D/L	10/5/0	11/2/2	11/3/1	9/4/2	9/5/1	7/7/1	6/8/1	6/7/2	4/8/3	7/5/3
Win/Draw/Loss	15/0/0	13/0/2	12/0/3	11/0/4	11/0/4	12/0/3	12/0/3	11/0/4	8/0/7	10/0/5
GM error ratio	0.8834	0.8841	0.9462	0.9929	1.005	1.0093	1.0061	1.0075	1.0095	1.0062

Table 4.16: DECORATE(No Rejection) vs. AdaBoost

	1%	2%	5%	10%	20%	30%	40%	50%	75%	100%
Sig. W/D/L	8/7/0	8/6/1	12/1/2	7/6/2	7/5/3	7/6/2	5/4/6	3/6/6	4/3/8	4/4/7
Win/Draw/Loss	13/0/2	13/0/2	13/0/2	13/0/2	9/0/6	8/0/7	9/0/6	9/0/6	6/0/9	7/0/8
GM error ratio	0.9194	0.8949	0.9576	0.9967	1.009	1.0148	1.0166	1.015	1.0212	1.0153

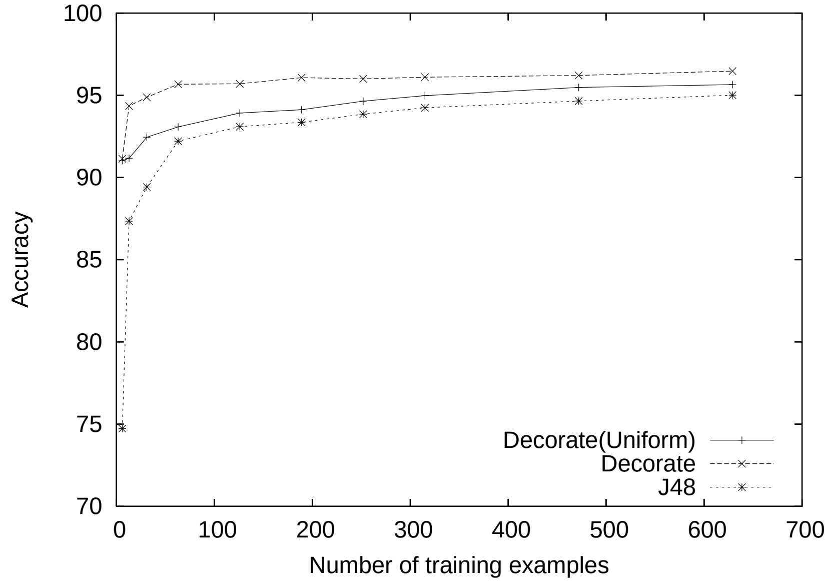


SEGMENT

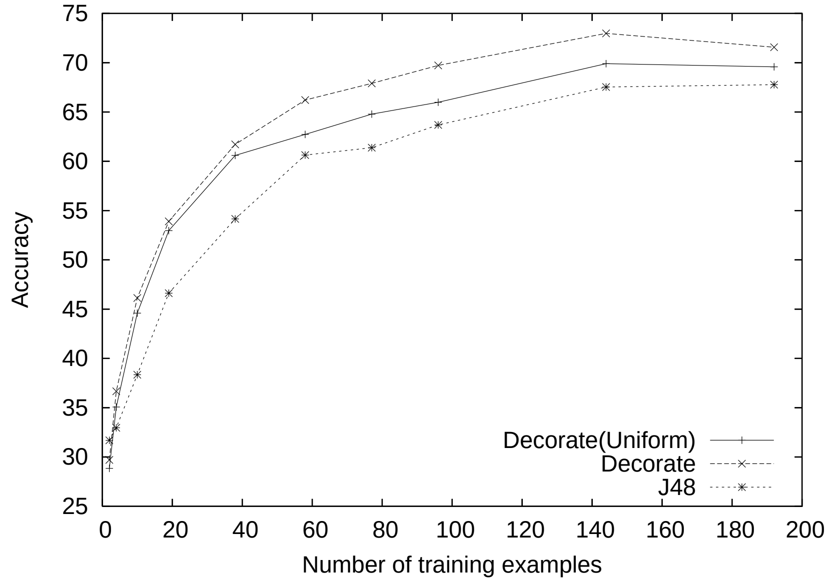


LYMPH

Figure 4.12: Comparing different approaches to generating artificial examples for DECORATE.

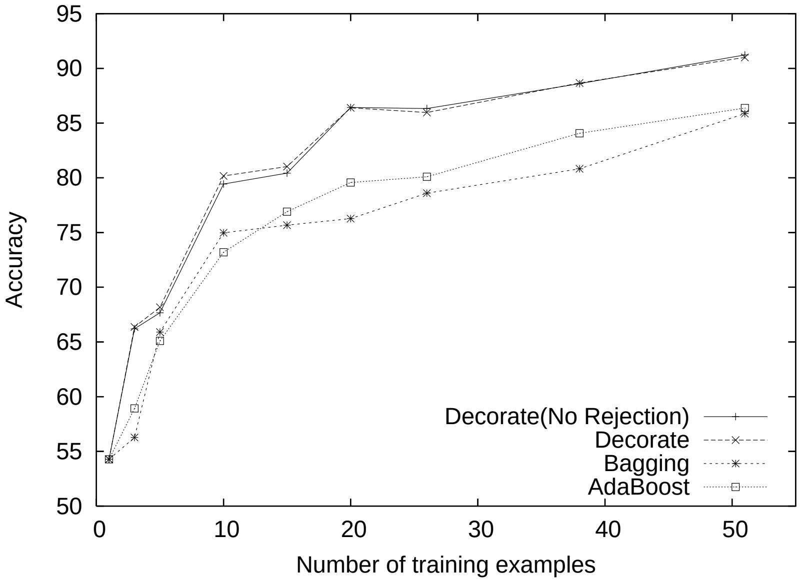


BREAST-W

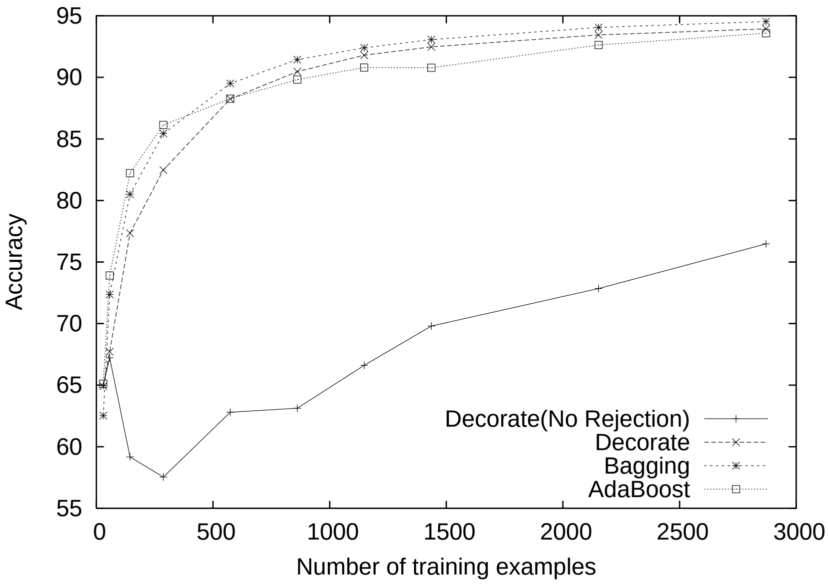


GLASS

Figure 4.13: Comparing different approaches to generating artificial examples for DECORATE.



(a) LABOR



(b) SPLICE

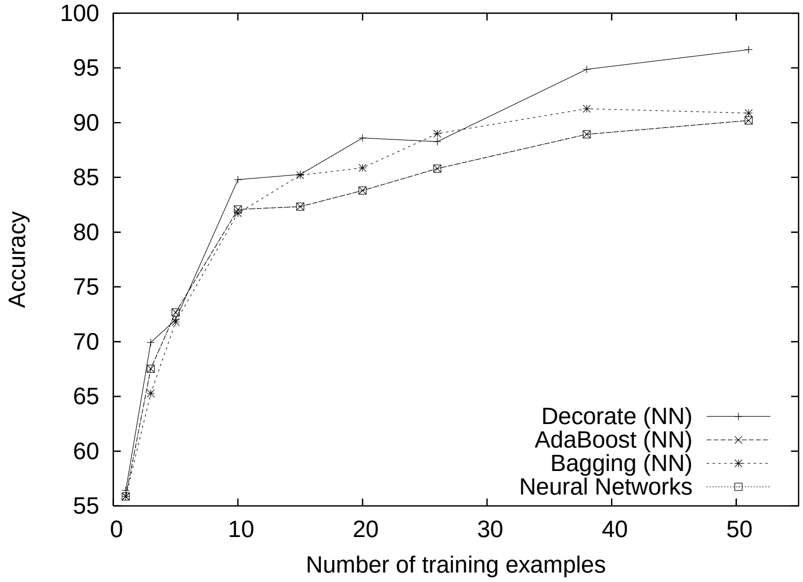
Figure 4.14: Comparison of DECORATE with and without the rejection criterion.

& Frank, 1999). For the network parameters, we set the learning rate to 0.15 and the momentum term to 0.9, as done in a similar study on ensemble methods (Opitz & Maclin, 1999). The number of hidden layers was set to half the sum of the number of attributes and classes for each dataset. We trained the networks for 80 epochs, which was the maximum used by Opitz and Maclin (1999). Experiments were run as in Section 4.1. However, since the training time for neural networks is much longer than for decision trees, we only ran five runs of 10-fold cross-validation on two datasets. We selected datasets on which DECORATE applied to decision trees performed well, so that we could verify that these results were not an artifact of the base learner.

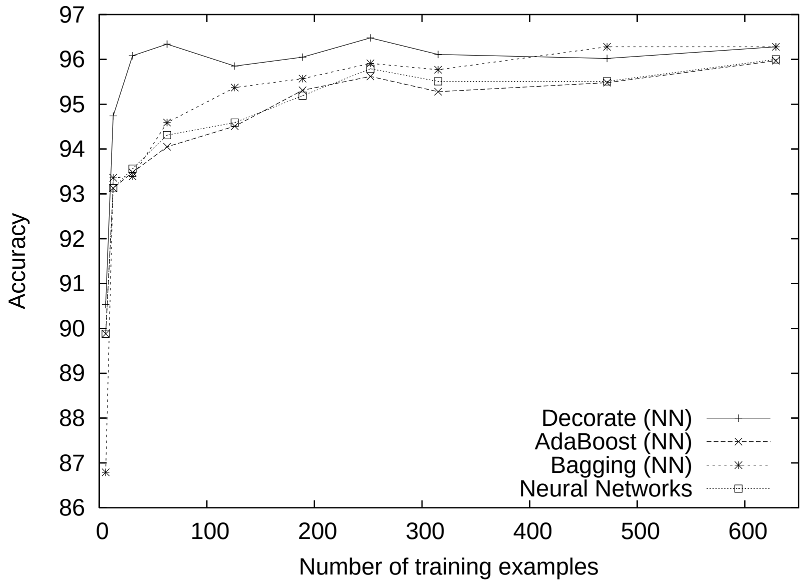
The resulting learning curves are presented in Figure 4.15. The results demonstrate that DECORATE significantly improves on the performance of neural networks, and is also better than Bagging and ADABOOST. The significant advantage of DECORATE early in learning is clearly visible on the *breast-w* dataset. On both datasets, ADABOOST does not produce a noticeable improvement over the base learner.

4.9 Experiments on Naive Bayes

The Naive Bayes algorithm (Duda & Hart, 1973) is a *stable* learner, i.e., small changes in the training set do not lead to significant changes in the classifier produced. However, most ensemble methods are best suited to use *unstable* base learners, as they facilitate the creation of a diverse set of classifiers. To test the effectiveness of ensemble methods on stable learners, we ran experiments comparing Bagging, ADABOOST and DECORATE applied to Naive Bayes. We observed that none of the ensemble methods consistently produce notable improvements over the base classifier. One can expect to see similar results with other stable learners, such as nearest neighbor classifiers. Our observations are supported by a recent study by Davidson (2004), which shows that Bagging, boosting and DECORATE are not very effective when applied to stable learners.



LABOR



BREAST-W

Figure 4.15: Comparing different ensemble methods applied to Neural Networks.

Chapter 5

Imperfections in Data

In addition to their many other advantages, classifier ensembles hold the promise of developing learning methods that are robust in the presence of imperfections in the data in terms of missing features and noise in both the class labels and the features. Noisy training data tends to increase the variance in the results produced by a given classifier; however, by learning a committee of hypotheses and combining their decisions, this variance can be reduced. In particular, variance-reducing methods such as *Bagging* (Breiman, 1996) have been shown to be robust in the presence of fairly high levels of noise, and can even *benefit* from low levels of noise (Dietterich, 2000).

Bagging is a fairly simple ensemble method which is generally outperformed by more sophisticated techniques such as ADABOOST (Freund & Schapire, 1996; Quinlan, 1996a). However, ADABOOST has a tendency to overfit when there is significant noise in the training data, preventing it from learning an effective ensemble (Dietterich, 2000). Therefore, there is a need for a general ensemble meta-learner that is at least as accurate as ADABOOST when there is little or no noise, but is more robust to higher levels of random error in the training data.

This chapter presents experiments from (Melville et al., 2004), in which we explore the resilience of DECORATE to the various forms of imperfections in data. In our experiments, the training data is corrupted with missing features, and random errors in the values of both the category and the features. Results on a variety of UCI data demonstrate that, in general, DECORATE continues to improve on the accuracy of the base learner, despite the presence of each of the three forms of imperfections. Furthermore, DECORATE is clearly more robust to missing features than the other ensemble methods.

5.1 Experimental Evaluation

5.1.1 Methodology

Three sets of experiments were conducted in order to compare the performance of ADABOOST, Bagging, DECORATE, and the base classifier J48, under varying amounts of three types of imperfections in the data: Each set of experiments differed from the other two only in the type of noise that was introduced:

1. **Missing features:** To introduce $N\%$ missing features to a data set of D instances, each of which has F features (excluding the class label), we select randomly with replacement $\frac{N \cdot D \cdot F}{100}$ instances and for each of them delete the value of a randomly chosen feature. Missing features were introduced to both the training and testing sets.

2. **Classification noise:** To introduce $N\%$ classification noise to a data set of D instances, we randomly select $\frac{N \cdot D}{100}$ instances with replacement and change their class labels to one of the *other* values chosen randomly with equal probability. Classification noise was introduced only to the training set and not to the test set.

3. **Feature noise:** To introduce $N\%$ feature noise to a data set of D instances, each of which has F features (excluding the class label), we randomly select with replacement $\frac{N \cdot D \cdot F}{100}$ instances and for each of them we change the value of a randomly selected feature. For nominal features, the new value is chosen randomly with equal

probability from the set of *all* possible values. For numeric features, the new value is generated from a Normal distribution defined by the mean and the standard deviation of the given feature, which are estimated from the data set. Feature noise was introduced to both the training and testing sets.

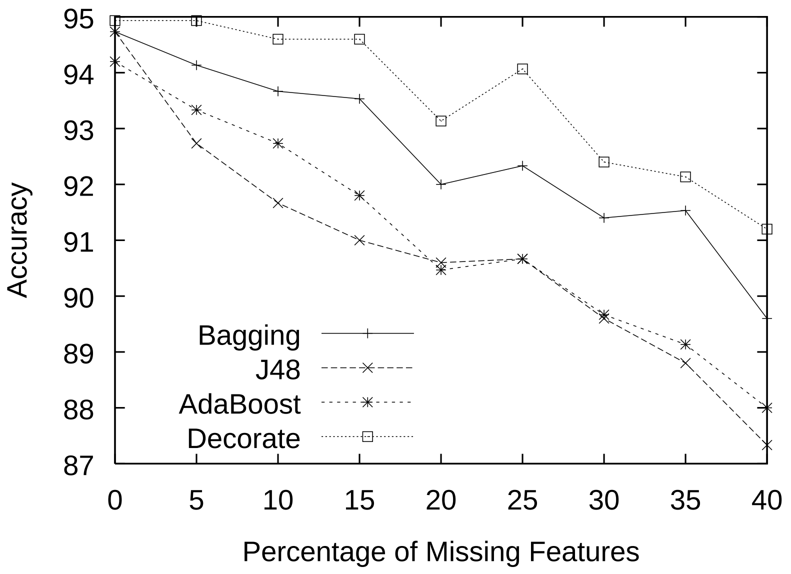
In each set of experiments, ADABOOST, Bagging, DECORATE, and J48 were compared on 11 UCI data sets using the Weka implementations of these methods (Witten & Frank, 1999). Table 5.1 presents some statistics about the data sets. The target ensemble size of the first three methods was set to 15. In the case of DECORATE, this size is only an upper bound on the size of the ensemble, and the algorithm may terminate with a smaller ensemble if the number of iterations exceeds the maximum limit. As in Chapter 4, this maximum limit was set to 50 iterations, and the number of artificially generated examples was equal to the training set size.

To ascertain that no ensemble method was being disadvantaged by the small ensemble size, we ran additional experiments on some datasets with the ensemble size set to 100. The trends of the results are similar to those with ensembles of size 15. So, in this chapter, we only present the results on ensembles of size 15.

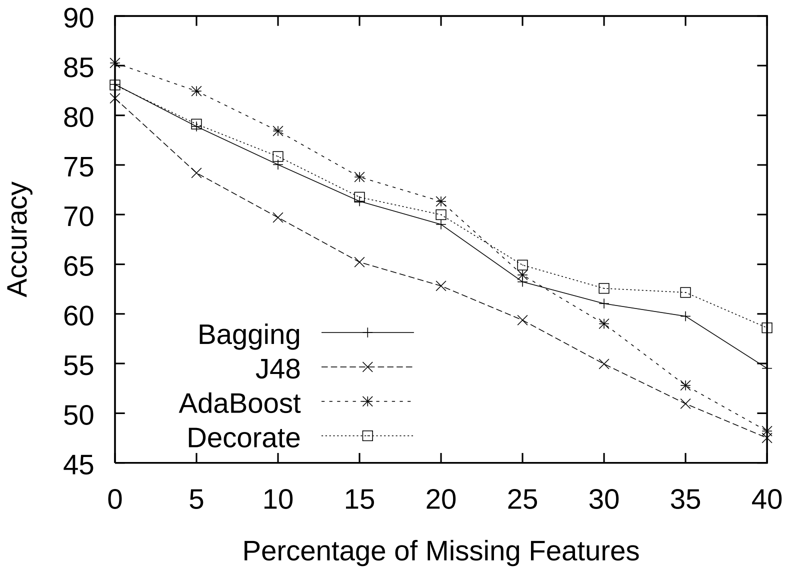
For each set of experiments, the performance of each of the learners was evaluated at increasing noise levels from 0% to 40% at 5% intervals using 10 complete 10-fold cross validations. As in Chapter 4, to compare two learning algorithms, we employ the statistics used by (Webb, 2000), namely, the significant win/draw/loss record and the geometric mean error ratio.

5.1.2 Results

This section presents the results of running the four algorithms on each of the 11 datasets summarized in Table 5.1.



(a) Iris



(b) Autos

Figure 5.1: Missing Features

Table 5.1: Summary of Data Sets

Name	Cases	Classes	Attributes	
			Numeric	Nominal
autos	205	6	15	10
balance-scale	625	3	4	–
breast-w	699	2	9	–
colic	368	2	10	12
credit-a	690	2	6	9
glass	214	6	9	–
heart-c	303	2	8	5
hepatitis	155	2	6	13
iris	150	3	4	–
labor	57	2	8	8
lymph	148	4	–	18

Missing Features

The results of running the algorithms when missing features are introduced, are presented in Tables 5.2–5.4. Each table compares the accuracy of DECORATE versus another algorithm for increasing percentages of missing features. The last two lines of each table present the win/draw/loss record and the GM error ratio respectively.

These results demonstrate that DECORATE is fairly robust to missing features, consistently beating the base learner, J48, at all noise levels (Table 5.2). In fact, when the amount of missing features is 20% or higher, DECORATE produces statistically significant wins over J48 on all datasets. The amount of error reduction produced by using DECORATE is also considerable, as is shown by the mean error ratios.

For this kind of imperfection in the data, in general, all of the ensemble methods produce some increase in accuracy over the base learner. However, the improvements brought about by using DECORATE are higher than those obtained by both Bagging and ADABOOST. The amount of error reduction achieved by DECORATE also increases with greater amounts of missing features; as is clearly demonstrated by the GM error ratios.

Figure 5.1(a) shows the results on a dataset that clearly demonstrates DECORATE's superior performance at all levels of missing features. In Figure 5.1(b), we see a dataset on

Table 5.2: Missing Features: DECORATE vs J48

Noise Level %	0	5	10	15	20	25	30	35	40
autos	83.05/81.72	**79.11**/74.19	**75.86**/69.7	**71.76**/65.21	**69.99**/62.82	**64.92**/59.38	**62.56**/54.96	**62.16**/50.95	**58.6**/47.5
balance-scale	**81.39**/77.85	**80.57**/77.11	**79.66**/76.4	**79.07**/75.29	**77.05**/75.07	**76.03**/72.86	**74.97**/71.53	**73.48**/70.47	**72.44**/69.96
breast-w	**96.47**/95.01	**96.12**/94.69	**96.1**/94.69	**95.81**/94.25	**95.87**/94.13	**95.39**/93.89	**95.18**/93.66	**94.72**/93.19	**94.32**/92.91
colic	84.91/85.16	83.72/83.93	82.71/82.63	**82.9**/81.89	**81.95**/80.56	**81.25**/79.23	**81.49**/78.56	**80.75**/77.26	**80.2**/76.39
credit-a	**86.16**/85.57	**85.83**/84.33	**85.16**/83.49	**84.72**/82.59	**84.42**/82.19	**82.97**/80.77	**82.28**/80.25	**81.87**/79.61	**81.26**/79.07
glass	**71.57**/67.77	**72.9**/68.36	**72.47**/68.73	**70.51**/65.29	**68.69**/66.15	**66.09**/63.95	**67.19**/61.46	**63.1**/59.46	**61.34**/57.4
heart-c	**78.42**/77.17	**78.52**/76.76	**79.37**/77.48	78.76/77.7	**78.84**/76.92	**78.15**/76.7	**78.97**/76.54	**77.02**/75.25	**77.71**/74.79
hepatitis	**83.58**/79.22	**82.41**/80.09	**84.46**/80.28	**82.78**/79.83	**82.89**/80.61	**83.19**/80.96	**83.05**/81.02	**82.93**/80.72	**83.56**/80.84
iris	94.93/94.73	**94.93**/92.73	**94.6**/91.67	**94.6**/91	**93.13**/90.6	**94.07**/90.67	**92.4**/89.6	**92.13**/88.8	**91.2**/87.33
labor	**91**/78.8	**91.07**/77.17	**90.23**/77.4	**89.33**/75.23	**89.4**/73.07	**86.97**/71.13	**87.07**/72.13	**85.03**/71.53	**84.57**/70.53
lymph	**79.08**/76.06	**78.87**/74.58	**77.48**/74.99	**77.41**/74.78	**78.55**/74.53	**75.74**/72.49	**77.86**/74.5	**75.84**/72.42	**75.05**/70.33
Sig. W/D/L	8/3/0	10/1/0	10/1/0	10/1/0	11/0/0	11/0/0	11/0/0	11/0/0	11/0/0
GM Error Ratio	0.8286	0.7882	0.7877	0.7815	0.7921	0.8039	0.8004	0.8095	0.8047

Table 5.3: Missing Features: DECORATE vs Bagging

Noise Level %	0	5	10	15	20	25	30	35	40
autos	83.05/83.12	79.11/78.87	75.86/75.02	71.76/71.35	69.99/69.01	**64.92**/63.22	62.56/61.05	**62.16**/59.77	**58.6**/54.52
balance-scale	81.39/**81.93**	80.57/**81.3**	79.66/80.25	79.07/79.16	77.05/**77.8**	76.03/75.74	74.97/74.48	73.48/73.43	**72.44**/71.84
berast-w	96.47/96.3	96.12/96.2	96.1/96.04	95.81/95.82	95.87/95.55	95.39/95.65	95.18/95.31	94.72/95.05	94.32/94.71
colic	84.91/85.34	83.72/83.99	82.71/82.63	**82.9**/81.7	**81.95**/80.67	**81.25**/79.61	**81.49**/78.2	**80.75**/77.26	**80.2**/76.58
credit-a	86.16/85.96	85.83/85.72	85.16/84.96	84.72/84.45	**84.42**/83.71	82.97/82.45	**82.28**/81.67	**81.87**/80.94	**81.26**/80.16
glass	71.57/**74.67**	72.9/72.47	**72.47**/70.53	70.51/70.55	68.69/69.85	66.09/66.7	67.19/66.66	63.1/63.62	61.34/62.48
heart-c	78.42/78.68	78.52/79.55	79.37/80.53	78.76/**79.81**	78.84/79.83	78.15/79.04	78.97/79.38	77.02/**78.24**	77.71/77.99
hepatitis	**83.58**/81.34	82.41/81.31	**84.46**/82.18	82.78/81.59	82.89/81.7	83.19/81.9	**83.05**/81.44	**82.93**/80.79	**83.56**/79.93
iris	94.93/94.73	**94.93**/94.13	**94.6**/93.67	**94.6**/93.53	**93.13**/92	**94.07**/92.33	92.4/91.4	92.13/91.53	**91.2**/89.6
labor	**91**/85.87	**91.07**/83.13	**90.23**/82.27	**89.33**/79.77	**89.4**/78.67	**86.97**/74.8	**87.07**/76.17	**85.03**/72.67	**84.57**/71.2
lymph	79.08/77.97	78.87/77.85	77.48/78	77.41/77.03	**78.55**/76.54	75.74/75.42	77.86/77.77	75.84/74.94	**75.05**/72.8
Sig. W/D/L	2/7/2	2/8/1	4/7/0	3/7/1	5/5/1	4/7/0	4/7/0	5/5/1	8/3/0
GM Error Ratio	0.952	0.9298	0.9201	0.9177	0.9041	0.9083	0.9085	0.915	0.8882

Table 5.4: Missing Features: DECORATE vs ADABOOST

Noise Level %	0	5	10	15	20	25	30	35	40
autos	83.05/**85.28**	79.11/**82.44**	75.86/**78.42**	71.76/**73.79**	69.99/71.33	64.92/63.93	**62.56**/59.01	**62.16**/52.79	**58.6**/48.19
balance-scale	**81.39**/77.76	**80.57**/77.06	**79.66**/76.91	**79.07**/77.34	77.05/76.73	**76.03**/74.75	**74.97**/73.32	**73.48**/71.8	**72.44**/70.54
breast-w	96.47/96.47	96.12/96.25	96.1/96.18	95.81/95.71	95.87/95.84	95.39/95.47	95.18/95.19	94.72/95.04	94.32/94.81
colic	**84.91**/81.93	**83.72**/81.22	**82.71**/79.63	**82.9**/78.48	**81.95**/79.08	**81.25**/77.85	**81.49**/77.99	**80.75**/77.59	**80.2**/77.45
credit-a	**86.16**/85.42	**85.83**/83.77	**85.16**/82.99	**84.72**/81.74	**84.42**/80.84	**82.97**/80.1	**82.28**/79.93	**81.87**/79.74	**81.26**/79.55
glass	71.57/**76.06**	72.9/73.93	72.47/72.52	70.51/70.2	68.69/69.34	66.09/67.21	**67.19**/64.22	63.1/63.98	61.34/60.75
heart-c	78.42/79.22	78.52/**79.94**	79.37/78.36	78.76/78.22	78.84/78.2	78.15/77.06	**78.97**/76.64	77.02/76.67	**77.71**/75.61
hepatitis	83.58/82.71	82.41/82.61	**84.46**/82.65	82.78/82.57	82.89/81.44	**83.19**/81.47	83.05/81.63	82.93/81.27	**83.56**/80.79
iris	94.93/94.2	**94.93**/93.33	**94.6**/92.73	**94.6**/91.8	**93.13**/90.47	**94.07**/90.67	**92.4**/89.67	**92.13**/89.13	**91.2**/88
labor	**91**/86.37	**91.07**/86.93	**90.23**/87.53	89.33/86.63	**89.4**/85.63	**86.97**/83.07	**87.07**/82.83	**85.03**/79.63	**84.57**/78.23
lymph	79.08/**81.75**	78.87/80.32	77.48/78.99	77.41/77.39	78.55/79.11	75.74/76.52	77.86/76.48	75.84/75.85	75.05/75.61
Sig. W/D/L	4/4/3	5/4/2	6/4/1	4/6/1	4/7/0	6/5/0	8/3/0	6/5/0	8/3/0
GM Error Ratio	0.9534	0.9382	0.9197	0.9024	0.9109	0.8982	0.8827	0.8968	0.8876

which AdaBoost has the best performance when there are no missing features; but with increasing amounts of missing features, both Bagging and Decorate outperform it.

Classification Noise

The comparison of each ensemble method with the base learner, in the presence of classification noise are summarized in Tables 5.5-5.7. The tables provide summary statistics, as described above, for each of the noise levels considered.

The win/draw/loss records indicate that, both Bagging and Decorate consistently outperform the base learner on most of the datasets at almost all noise levels; demonstrating that both are quite robust to classification noise. In the range of 10-35% of classification noise, Bagging performs a little better than Decorate, as is seen from the error ratios. This is because, occasionally, the addition of noise helps Bagging, as was also observed in (Dietterich, 2000).

Unlike, Bagging and Decorate, AdaBoost is very sensitive to noise in classifications. Though AdaBoost significantly outperforms J48 on 7 of the 11 datasets in the absence of noise, its performance degrades rapidly at noise levels as low as 10%. With 35-40% noise, AdaBoost performs significantly worse than the base learner on 7 of the datasets. Our results on the performance of AdaBoost agree with previously published studies (Dietterich, 2000; Bauer & Kohavi, 1999; McDonald, Eckley, & Hand, 2002). As pointed out in these studies, AdaBoost degrades in performance because it tends to place a lot of weight on the noisy examples.

Figure 5.2(a) shows a dataset on which Decorate has a clear advantage over other methods, at all levels of noise. Figure 5.2(b) presents a dataset on which Bagging outperforms the other methods at most noise levels. This figure also clearly demonstrates how rapidly the accuracy of AdaBoost can drop below that of the base learner. These results confirm that, in domains with appreciable levels of classification noise, it is beneficial to use Decorate or Bagging, but detrimental to apply AdaBoost.

Table 5.5: Class Noise: DECORATE vs J48

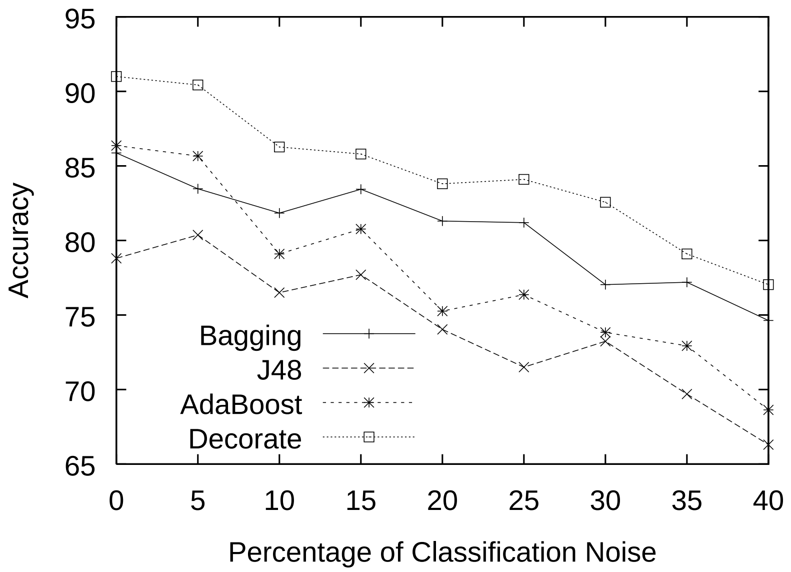
Noise Level %	0	5	10	15	20	25	30	35	40
autos	83.05/81.72	79.73/77.97	**77.63**/75.58	**74.4**/71.65	**71.7**/68.71	66.65/65.49	64.52/62.79	**63**/60.2	**61.41**/58.43
balance-scale	**81.39**/77.85	**81.58**/78.65	**81.8**/79.16	**81.08**/78.1	**80.44**/77.56	**79.64**/76.99	**79.36**/76.13	**77.79**/75.23	**77.38**/73.82
breast-w	**96.47**/95.01	**95.85**/94.34	**95.12**/94.29	**95.11**/93.41	**93.83**/93.21	**93.63**/92.73	**93.11**/91.97	**91.96**/90.96	**91.62**/90.51
colic	84.91/85.16	84.37/84.91	84.05/84.78	82.66/**84.58**	82.16/**84.47**	80.13/**83.83**	79.05/**83.47**	78.25/**81.49**	77.2/**81.78**
credit-a	**86.16**/85.57	85.12/85.09	84.88/85.09	83.09/**84.86**	81.54/**83.41**	80.29/**82.9**	77.99/**82.12**	74.87/**79.57**	73.16/**78.51**
glass	**71.57**/67.77	**72.92**/67.29	**71.6**/64.4	**70.63**/63.13	**71.03**/62.4	**69.89**/60.05	**66.42**/57	**65.55**/55.68	**64.62**/53.85
heart-c	**78.42**/77.17	**78.94**/76.6	**78.38**/76.47	**76.74**/74.97	**76.46**/74.76	**75.58**/73.37	**73.3**/70.08	**72.08**/69.04	**71.45**/67.09
hepatitis	**83.58**/79.22	**81.1**/78.09	**80.17**/78.06	**79.5**/76.72	**78.52**/75.8	76.65/76.3	74.14/73.15	73.18/73.02	72.15/71.98
iris	94.93/94.73	94.67/94.2	94/93.2	92.8/91.8	92.13/90.93	90.4/89.87	89.33/88.47	87/86.47	**87.6**/83.67
labor	**91**/78.8	**90.43**/80.37	**86.27**/76.5	**85.8**/77.7	**83.8**/74.03	**84.1**/71.5	**82.57**/73.23	**79.1**/69.7	**77.03**/66.3
lymph	**79.08**/76.06	**78.86**/75.02	**78.92**/73.65	**78.74**/75.01	**78.14**/72.17	**77.58**/71.35	**76.5**/71.12	**73.73**/68.6	**72.95**/67.15
Sig. W/D/L	8/3/0	7/4/0	8/3/0	8/1/2	8/1/2	6/3/2	6/3/2	7/2/2	8/1/2
GM Error Ratio	0.8286	0.8398	0.8633	0.8734	0.8809	0.896	0.9121	0.9229	0.8995

Table 5.6: Class Noise: Bagging vs J48

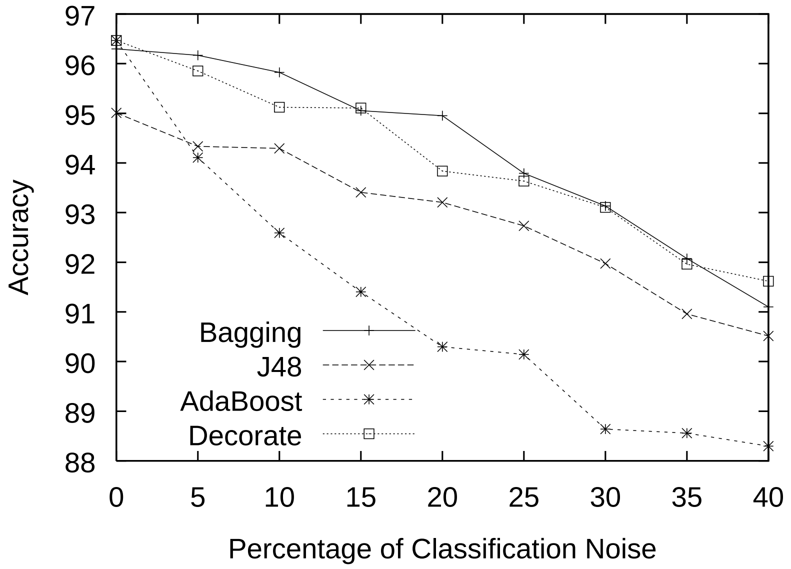
Noise Level %	0	5	10	15	20	25	30	35	40
autos	83.12/81.72	**81.38**/77.97	**79.68**/75.58	**76.6**/71.65	**74.54**/68.71	**69.6**/65.49	**67.81**/62.79	**64.03**/60.2	**63.1**/58.43
balance-scale	**81.93**/77.85	**82.09**/78.65	**81.69**/79.16	**81.58**/78.1	**81.04**/77.56	**79.9**/76.99	**78.82**/76.13	**77.68**/75.23	**76.67**/73.82
breast-w	**96.3**/95.01	**96.17**/94.34	**95.82**/94.29	**95.05**/93.41	**94.95**/93.21	**93.79**/92.73	**93.13**/91.97	**92.08**/90.96	91.1/90.51
colic	85.34/85.16	85.18/84.91	84.75/84.78	84.45/84.58	84.39/84.47	83.42/83.83	83.31/83.47	81.38/81.49	81.84/81.78
credit-a	85.96/85.57	**86.26**/85.09	**85.75**/85.09	85.35/84.86	83.9/83.41	82.12/82.9	79.71/**82.12**	77.06/**79.57**	76.26/**78.51**
glass	**74.67**/67.77	**72.71**/67.29	**72.62**/64.4	**69.45**/63.13	**69.57**/62.4	**69.93**/60.05	**68.98**/57	**65.79**/55.68	**62.18**/53.85
heart-c	**78.68**/77.17	**79.74**/76.6	**79.31**/76.47	**79.51**/74.97	**77.73**/74.76	**77.78**/73.37	**76.61**/70.08	**76.68**/69.04	**73.37**/67.09
hepatitis	**81.34**/79.22	**81.02**/78.09	**81.3**/78.06	**80.59**/76.72	**80.47**/75.8	78.3/76.3	**77.22**/73.15	74.88/73.02	**75.09**/71.98
iris	94.73/94.73	94.13/94.2	93.8/93.2	**92.73**/91.8	91.87/90.93	89.87/89.87	88.13/88.47	85.47/86.47	83.67/83.67
labor	**85.87**/78.8	**83.47**/80.37	**81.83**/76.5	**83.43**/77.7	**81.3**/74.03	**81.2**/71.5	**77.03**/73.23	**77.2**/69.7	**74.63**/66.3
lymph	**77.97**/76.06	**77.05**/75.02	**78.09**/73.65	**77.69**/75.01	**76.69**/72.17	**76.29**/71.35	**75.78**/71.12	**73.03**/68.6	**71.75**/67.15
Sig. W/D/L	7/4/0	9/2/0	9/2/0	9/2/0	8/3/0	7/4/0	8/2/1	7/3/1	7/3/1
GM Error Ratio	0.8704	0.8687	0.8526	0.8508	0.8443	0.8719	0.8867	0.8972	0.8995

Table 5.7: Class Noise: AdaBoost vs j48

Noise Level %	0	5	10	15	20	25	30	35	40
autos	**85.28**/81.72	**79.96**/77.97	76.67/75.58	70.6/71.65	66.95/68.71	63.8/65.49	60.89/62.79	58.69/60.2	57.56/58.43
balance-scale	77.76/77.85	76.49/**78.65**	75.18/**79.16**	73.6/**78.1**	70.81/**77.56**	69.78/**76.99**	69.06/**76.13**	67.41/**75.23**	66.86/**73.82**
breast-w	**96.47**/95.01	94.11/94.34	92.59/**94.29**	91.4/**93.41**	90.3/**93.21**	90.14/**92.73**	88.64/**91.97**	88.56/**90.96**	88.3/**90.51**
colic	81.93/**85.16**	79.43/**84.91**	77.04/**84.78**	75.92/**84.58**	73.28/**84.47**	69.95/**83.83**	69.54/**83.47**	67.38/**81.49**	65.26/**81.78**
credit-a	85.42/85.57	82.46/**85.09**	81.1/**85.09**	77.51/**84.86**	74.41/**83.41**	73.32/**82.9**	70.87/**82.12**	68.52/**79.57**	67.93/**78.51**
glass	**76.06**/67.77	**75.42**/67.29	**70.9**/64.4	**67.27**/63.13	**65.48**/62.4	**65.83**/60.05	**61.65**/57	**60.66**/55.68	**58.43**/53.85
heart-c	**79.22**/77.17	**78.82**/76.6	77.7/76.47	75.58/74.97	72.41/**74.76**	71.49/**73.37**	70.59/70.08	69.03/69.04	65.51/67.09
hepatitis	**82.71**/79.22	**79.82**/78.09	77.47/78.06	75.92/76.72	74.86/75.8	73.6/**76.3**	72.8/73.15	69.53/**73.02**	69.16/**71.98**
iris	94.2/94.73	91.33/**94.2**	88.67/**93.2**	85/**91.8**	81.33/**90.93**	80.6/**89.87**	77.27/**88.47**	76.07/**86.47**	75.27/**83.67**
labor	**86.37**/78.8	**85.67**/80.37	79.1/76.5	80.77/77.7	75.27/74.03	**76.37**/71.5	73.83/73.23	72.93/69.7	68.63/66.3
lymph	**81.75**/76.06	**79.75**/75.02	**77.78**/73.65	76/75.01	74/72.17	69.75/71.35	68.06/**71.12**	64.97/**68.6**	63.63/**67.15**
Sig. W/D/L	7/3/1	6/1/4	2/4/5	1/5/5	1/4/6	2/2/7	1/4/6	1/3/7	1/3/7
GM Error Ratio	0.8691	0.993	1.0984	1.1604	1.2322	1.2242	1.2431	1.212	1.1989



(a) Labor



(b) Breast-W

Figure 5.2: Classification Noise

Feature Noise

The results of running the algorithms with noise in the features are presented in Tables 5.8–5.10. Each table compares the accuracy of each ensemble method versus J48 for increasing amounts of feature noise.

In most cases, all ensemble methods improve on the accuracy of the base learner, at all levels of feature noise. Bagging performs a little better than the other methods, in terms of significant wins according to the win/draw/loss record. In general, all systems degrade in performance with added feature noise. The drop in accuracy of the ensemble methods seems to mirror that of the base learner, as can be seen in Figure 5.3. The performance of the ensemble methods seems to be tied to how well the base learner deals with feature noise.

5.2 Related Work

Several previous studies have focused on exploring the performance of various ensemble methods in the presence of noise. A thorough comparison of Bagging, ADABOOST, and Randomization (a method for building a committee of decision trees, which randomly determine the split at each internal tree node) is presented by Dietterich (2000). This study concludes that while ADABOOST outperforms Bagging and Randomization in settings where there is no noise, it performs significantly worse when classification noise is introduced. This behavior of ADABOOST is attributed to its tendency to overfit by assigning large weights to noisy examples.

Other studies have reached similar conclusions about AdaBoost (Bauer & Kohavi, 1999; McDonald et al., 2002), and several variations of AdaBoost have been developed to address this issue. For example, Kalai and Servedio (2003) present a new boosting algorithm and prove that it can attain arbitrary accuracy when classification noise is present. Another algorithm, Smooth Boosting, that is proven to tolerate a combination of classifica-

Table 5.8: Feature Noise: DECORATE vs J48

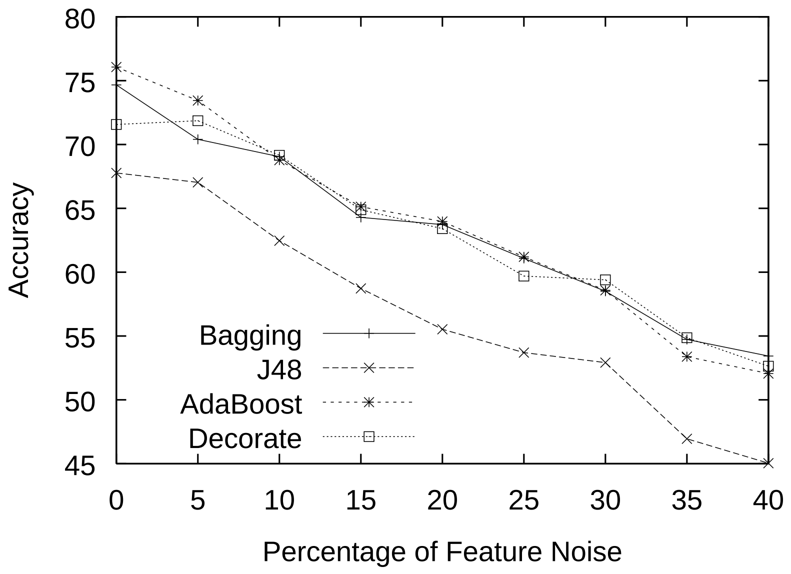
Noise Level %	0	5	10	15	20	25	30	35	40
autos	83.05/81.72	**78.22**/72.9	**72.86**/65.45	**67.62**/59.69	**63**/55.23	**60.75**/52.69	**56.85**/49.51	**50.48**/45.23	**49**/43.21
balance-scale	**81.39**/77.85	**80.17**/77.37	**78.22**/75.16	**76.53**/73.42	**74.56**/71.52	**72.93**/69.49	**72.19**/68.37	**69.26**/66.85	**68.96**/65.16
breast-w	**96.47**/95.01	**96.08**/93.91	**95.71**/93.26	**95.55**/92.72	**94.44**/91.75	**94.52**/90.42	**94.01**/90.36	**93.58**/89.47	**93.79**/88.69
colic	84.91/85.16	82.98/83.64	81.6/81.9	79.6/79.94	77.5/77.83	76.63/76.54	76.2/75.41	74.05/73.26	71.36/71.52
credit-a	**86.16**/85.57	83.96/84.07	82.03/81.8	80.19/80.55	78.09/**79**	78.3/78.97	76.03/76.52	74.29/74.7	73.16/73.64
glass	**71.57**/67.77	**71.86**/67.03	**69.15**/62.46	**64.89**/58.72	**63.41**/55.53	**59.69**/53.7	**59.4**/52.92	**54.86**/46.95	**52.64**/45.04
heart-c	**78.42**/77.17	76.76/75.92	76.46/75.34	73.25/73.32	72.33/72.56	72.42/71.55	72.41/71.49	69.43/69.74	69.75/70.28
hepatitis	**83.58**/79.22	**81.4**/78.44	**80.04**/78.22	**81.18**/78.36	79.11/78.15	79.24/77.7	77.28/77.25	**79.72**/77.12	77.47/77.12
iris	94.93/94.73	93.33/92.33	**91.27**/90	**91.2**/88	**88.33**/85.4	**85.33**/81.73	**84.6**/82.47	**81.2**/76.8	**81.6**/77.2
labor	**91**/78.8	**89.03**/78.47	**87.13**/76.9	**84.13**/73.77	**81.2**/70.93	**79.17**/71.27	**82.3**/72.4	**76.57**/68.23	**76.87**/70.57
lymph	**79.08**/76.06	**76.8**/74.31	71.85/71.77	**74.02**/71.31	**72.43**/68.44	**70.25**/67.22	68.79/68.32	65.56/64.46	64.69/63.69
Sig. W/D/L	8/3/0	7/4/0	7/4/0	8/3/0	7/3/1	7/4/0	6/5/0	7/4/0	6/5/0
GM Error Ratio	0.8286	0.8335	0.8434	0.8329	0.8593	0.8554	0.8690	0.8723	0.8782

Table 5.9: Feature Noise: Bagging vs. J48

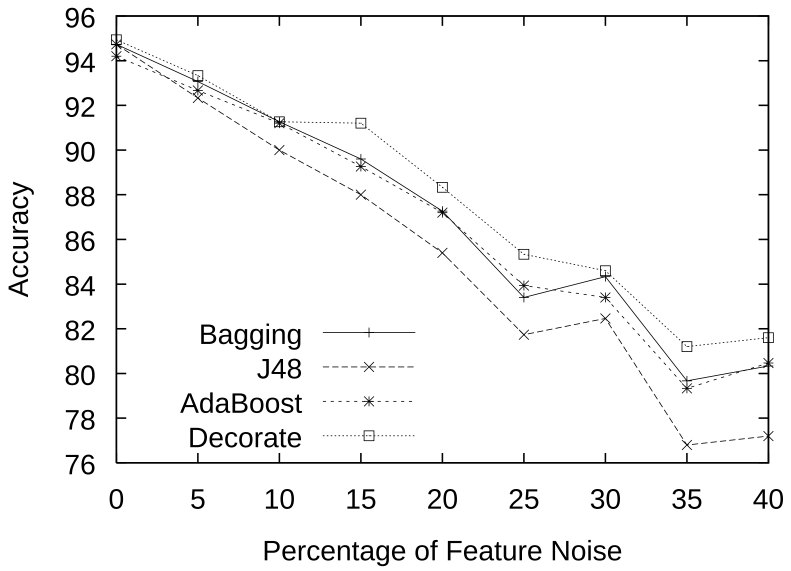
Noise Level %	0	5	10	15	20	25	30	35	40
autos	83.12/81.72	**78.12**/72.9	**72.71**/65.45	**67.82**/59.69	**64.64**/55.23	**62.89**/52.69	**57.84**/49.51	**55.31**/45.23	**52.55**/43.21
balance-scale	**81.93**/77.85	**80.83**/77.37	**78.79**/75.16	**76.9**/73.42	**74.84**/71.52	**73.15**/69.49	**72**/68.37	**69.55**/66.85	**68.8**/65.16
breast-w	**96.3**/95.01	**95.77**/93.91	**95.48**/93.26	**94.96**/92.72	**94.35**/91.75	**93.76**/90.42	**93.46**/90.36	**92.46**/89.47	**91.95**/88.69
colic	85.34/85.16	83.44/83.64	82.01/81.9	80.52/79.94	78.15/77.83	77.01/76.54	**76.38**/75.41	**74.65**/73.26	**73.29**/71.52
credit-a	85.96/85.57	**84.94**/84.07	**82.81**/81.8	**81.49**/80.55	**80.22**/79	79.29/78.97	**78.03**/76.52	75.42/74.7	**74.78**/73.64
glass	**74.67**/67.77	**70.4**/67.03	**69.04**/62.46	**64.29**/58.72	**63.73**/55.53	**61.08**/53.7	**58.51**/52.92	**54.73**/46.95	**53.44**/45.04
heart-c	**78.68**/77.17	**78.94**/75.92	**78.35**/75.34	**77.6**/73.32	**76.5**/72.56	**75.84**/71.55	**76.16**/71.49	**74.03**/69.74	**72.86**/70.28
hepatitis	**81.34**/79.22	**82.31**/78.44	**82.03**/78.22	**81.73**/78.36	**80.61**/78.15	**80.03**/77.7	**80.85**/77.25	**79.83**/77.12	**80.21**/77.12
iris	94.73/94.73	**93.07**/92.33	**91.27**/90	**89.6**/88	**87.27**/85.4	**83.4**/81.73	**84.33**/82.47	**79.67**/76.8	**80.33**/77.2
labor	**85.87**/78.8	**82.67**/78.47	**81.83**/76.9	**78.5**/73.77	**77.27**/70.93	73.8/71.27	**76.43**/72.4	**73.6**/68.23	74/70.57
lymph	**77.97**/76.06	**77.46**/74.31	**74.6**/71.77	**75.3**/71.31	**72.78**/68.44	**71.88**/67.22	70.44/68.32	**70.38**/64.46	**69.33**/63.69
Sig. W/D/L	7/4/0	10/1/0	10/1/0	10/1/0	10/1/0	8/3/0	10/1/0	10/1/0	10/1/0
GM Error Ratio	0.8704	0.8586	0.8450	0.8496	0.8473	0.8627	0.8634	0.8614	0.8661

Table 5.10: Feature Noise: AdaBoost vs. J48

Noise Level %	0	5	10	15	20	25	30	35	40
autos	**85.28**/81.72	**80.82**/72.9	**74.02**/65.45	**70.77**/59.69	**66.86**/55.23	**63.64**/52.69	**59.83**/49.51	**52.92**/45.23	**52.85**/43.21
balance-scale	77.76/77.85	76.8/77.37	75.04/75.16	73.42/73.42	72.23/71.52	**70.48**/69.49	**69.61**/68.37	67.49/66.85	**67.53**/65.16
breast-w	**96.47**/95.01	**96.11**/93.91	**95.74**/93.26	**95.65**/92.72	**94.62**/91.75	**94.48**/90.42	**94.16**/90.36	**93.32**/89.47	**92.81**/88.69
colic	81.93/**85.16**	80.68/**83.64**	78.28/**81.9**	76.58/**79.94**	75.68/**77.83**	73.65/**76.54**	72.41/**75.41**	71.69/73.26	69.98/71.52
credit-a	85.42/85.57	83.17/**84.07**	81.38/81.8	80.23/80.55	77.88/**79**	77.03/**78.97**	75.7/76.52	73.97/74.7	72.51/**73.64**
glass	**76.06**/67.77	**73.45**/67.03	**68.78**/62.46	**65.13**/58.72	**63.97**/55.53	**61.2**/53.7	**58.58**/52.92	**53.38**/46.95	**52.06**/45.04
heart-c	**79.22**/77.17	**78.77**/75.92	**78.18**/75.34	**77**/73.32	**75.84**/72.56	**75.3**/71.55	**75.11**/71.49	**72.38**/69.74	**73.07**/70.28
hepatitis	**82.71**/79.22	**82.62**/78.44	**81.98**/78.22	**82.43**/78.36	**80.8**/78.15	79.26/77.7	**79.51**/77.25	**80.9**/77.12	78.47/77.12
iris	94.2/94.73	92.67/92.33	**91.2**/90	89.27/88	**87.2**/85.4	**83.93**/81.73	83.4/82.47	**79.33**/76.8	**80.47**/77.2
labor	**86.37**/78.8	**85.3**/78.47	**81.97**/76.9	**81.4**/73.77	**80.03**/70.93	**78.83**/71.27	**77.83**/72.4	**73.17**/68.23	**77**/70.57
lymph	**81.75**/76.06	**81.38**/74.31	**78.8**/71.77	**76.51**/71.31	**75.48**/68.44	**74.16**/67.22	**71.48**/68.32	66.41/64.46	**68.98**/63.69
Sig. W/D/L	7/3/1	7/2/2	8/2/1	7/3/1	8/1/2	8/1/2	8/2/1	7/4/0	8/2/1
GM Error Ratio	0.8691	0.8449	0.8575	0.8455	0.8463	0.8564	0.8830	0.8900	0.8750



(a) Glass



(b) Iris

Figure 5.3: Feature Noise

tion and feature noise is presented in (Servedio, 2003). McDonald et al. (2003) compare ADABOOST to two other boosting algorithms, LogitBoost and BrownBoost, and conclude that BrownBoost is quite robust to noise. In an earlier study an extension to BrownBoost for multi-class problems was presented and shown empirically to outperform ADABOOST on noisy data (McDonald et al., 2002). However, BrownBoost's drawback is that it requires a time-out parameter to be set, which can be done only if the user can estimate the level of noise.

5.3 Chapter Summary

This chapter evaluates the performance of three ensemble methods, Bagging, ADABOOST and DECORATE, in the presence of different kinds of imperfections in the data. Experiments using J48 as the base learner, show that in the case of missing features, DECORATE significantly outperforms the other approaches. In the case of classification noise, both DECORATE and Bagging are effective at decreasing the error of the base learner; whereas ADABOOST degrades rapidly in performance, often performing worse than J48. In general, Bagging performs the best at combatting high amounts of classification noise. In the presence of noise in the features, all ensemble methods produce consistent improvements over the base learner. These results suggest that, when there are many missing features in the data, or noise in the classification labels, it is better to use DECORATE or Bagging over ADABOOST.

Chapter 6

Active Learning for Classification Accuracy

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 upon which it is trained. The ability to actively select the most useful training examples is an important approach to reducing the amount of supervision required for effective learning. In particular, *pool-based sample selection*, in which the learner chooses the best instances for labeling from a given set of unlabeled examples, is the most practical approach for problems in which unlabeled data is relatively easily available (Cohn et al., 1994). A theoretically well-motivated approach to sample selection is *Query by Committee* (Seung, Opper, & Sompolinsky, 1992), in which an ensemble of hypotheses is learned and examples that cause maximum disagreement amongst this committee (with respect to the predicted categorization) are selected as the most informative. Popular ensemble learning algorithms, such as Bagging and Boosting, have been used to efficiently learn effective committees for active learning (Abe & Mamitsuka, 1998). Meta-learning ensemble algorithms, such as Bagging and Boosting, that employ an arbitrary base classifier are particularly useful since they are general purpose and can be applied to improve any learner that is effective for a given domain.

An important property of a good ensemble for committee-based active learning is diversity. Only a committee of hypotheses that effectively samples the version space of all consistent hypotheses is productive for sample selection (Cohn et al., 1994). Since DECORATE explicitly builds such committees, it is well suited for this task. We believe that the added diversity of DECORATE ensembles should help select more informative examples than other Query by Committee methods. Melville and Mooney (2004b) introduced a new approach to active learning, ACTIVEDECORATE, which uses committees produced by DECORATE to select examples for labeling. Extensive experimental results on several real-world datasets show that using this approach produces substantial improvement over using DECORATE with random sampling. ACTIVEDECORATE requires far fewer examples than DECORATE, and on average also produces considerable reductions in error. In general, our approach also outperforms both Query by Bagging and Query by Boosting.

In this chapter, we will focus on active learning of classifiers, where the objective is to improve classification accuracy. In Chapter 7, we will discuss the related problem of active learning to improve class probability estimation.

6.1 Query by Committee

Query by Committee (QBC) is a very effective active learning approach that has been successfully applied to different classification problems (McCallum & Nigam, 1998; Dagan & Engelson, 1995; Liere & Tadepalli, 1997). A generalized outline of the QBC approach is presented in Algorithm 4. Given a pool of unlabeled examples, QBC iteratively selects examples to be labeled for training. In each iteration, it generates a committee of classifiers based on the current training set. Then it evaluates the potential utility of each example in the unlabeled set, and selects a subset of examples with the highest expected utility. The labels for these examples are acquired and they are transfered to the training set. Typically, the utility of an example is determined by some measure of *disagreement* in the committee about its predicted label. This process is repeated until the number of available requests for labels is exhausted.

Freund, Seung, Shamir, and Tishby (1997) showed that under certain assumptions, Query by Committee can achieve an exponential decrease in the number of examples required to attain a particular level of accuracy, as compared to random sampling. However, these theoretical results assume that the Gibbs algorithm is used to generate the committee of hypotheses used for sample selection. The Gibbs algorithm for most interesting problems is computationally intractable. To tackle this issue, Abe and Mamitsuka (1998) proposed two variants of QBC, Query by Bagging and Query by Boosting, where Bagging and ADABOOST are used to construct the committees for sample selection. In their approach, they evaluate the utility of candidate examples based on the *margin* of the example; where the margin is defined as the difference between the number of votes in the current committee for the most popular class label, and that for the second most popular label. Examples with smaller margins are considered to have higher utility.

6.2 ACTIVEDECORATE

It is beneficial in QBC to use an ensemble method that builds a *diverse* committee, in which each hypothesis is as different as possible, while still maintaining consistency with the training data. Since DECORATE explicitly focuses on creating ensembles that are diverse, we propose a variant of Query by Committee, ACTIVEDECORATE, that uses DECORATE (in Algorithm 4) to construct committees for sample selection.

To evaluate the expected utility of unlabeled examples, we also used the margins on the examples, as done by Abe and Mamitsuka (1998). We generalized their definition, to allow the base classifiers in the ensemble to provide class probabilities, instead of just the most likely class label. Given the class membership probabilities predicted by the committee, the margin is then defined as the difference between the highest and second highest

Algorithm 4 Generalized Query by Committee

Given:
T - set of training examples
U - set of unlabeled training examples
$BaseLearn$ - base learning algorithm
k - number of selective sampling iterations
m - size of each sample

1. Repeat k times
2. Generate a committee of classifiers,
   $C^* = EnsembleMethod(BaseLearn, T)$
3. $\forall x_j \in U$, compute $Utility(C^*, x_j)$, based on the current committee
4. Select a subset S of m examples that maximizes utility
5. Label examples in S
6. Remove examples in S from U and add to T
7. Return $EnsembleMethod(BaseLearn, T)$

predicted probabilities.

6.3 Experimental Evaluation

6.3.1 Methodology

To evaluate the performance of ACTIVEDECORATE, we ran experiments on 15 representative data sets from the UCI repository (Blake & Merz, 1998). We compared the performance of ACTIVEDECORATE with that of Query by Bagging (QBag), Query by Boosting (QBoost) and DECORATE, all using an ensemble size of 15. J48 decision-tree induction was used as the base learner for all methods.

The performance of each algorithm was averaged over two runs of 10-fold cross-validation. In each fold of cross-validation, we generated learning curves in the following fashion. The set of available training examples was treated as an unlabeled pool of examples, and at each iteration the active learner selected a sample of points to be labeled and added to the training set. For DECORATE, the examples in each iteration were selected randomly. The resulting curves evaluate how well an active learner orders the set of available examples in terms of utility. At the end of the learning curve, all algorithms see exactly the same training examples.

To maximize the gains of active learning, it is best to acquire a single example in each iteration. However to make our experiments computationally feasible, we choose larger sample sizes for the bigger data sets. In particular, we used a sample size of two for the *primary* dataset, and three for *breast-w, soybean, diabetes, vowel* and *credit-g*.

The primary aim of active learning is to reduce the amount of training data needed to induce an accurate model. To evaluate this, we first define the *target error rate* as the error that DECORATE can achieve on a given dataset, as determined by its error rate averaged over the points on the learning curve corresponding to the last 50 training examples. We then record the smallest number of examples required by a learner to achieve the same or lower error. We define the *data utilization ratio*, as the number of examples an active learner requires to reach the target error rate divided by the number DECORATE requires. This metric reflects how efficiently the active learner is using the data and is similar to a measure used by Abe and Mamitsuka (1998).

Another metric for evaluating an active learner is how much it improves accuracy over random sampling given a fixed amount of labeled data. Therefore, we also compute the percentage reduction in error over DECORATE averaged over points on the learning curve. As mentioned above, towards the end of the learning curve, all methods will have seen almost all the same examples. Hence, the main impact of active learning is lower on the learning curve. To capture this, we report the percentage error reduction averaged over only the 20% of points on the learning curve, where the largest improvements are produced. This is similar to a measure reported by Saar-Tsechansky and Provost (2001). When computing the error reduction of one system over another, the reduction is considered *significant* if the difference in the errors of the two systems averaged across the selected points on the learning curve is determined to be statistically significant according to paired t-tests ($p < 0.05$).

6.3.2 Results

The data utilization of the different active learners with respect to DECORATE is summarized in Table 6.1. We present the number of examples required for each system to achieve the target error rate and, in parentheses, the data utilization ratio. The smallest number of examples needed for each dataset is presented in bold font. On all but one dataset, ACTIVEDECORATE produces improvements over DECORATE in terms of data utilization. Furthermore, ACTIVEDECORATE outperforms both the other active learners on 10 of the datasets. QBag and QBoost were unable to achieve the target error rate on *vowel*; and QBoost also failed to achieve the target error on *primary*. Furthermore, on several datasets QBag and QBoost required more training examples than DECORATE. On average, ACTIVE-

Table 6.1: Data utilization with respect to Decorate

Dataset	Tot. Size	Decorate	QBag	QBoost	ActiveDecorate	Target Err.(%)
Soybean	615	492(1.00)	267(0.54)	219(0.45)	**144(0.29)**	6.59
Vowel	891	840(1.00)	-	-	**477(0.57)**	3.81
Statlog	243	81(1.00)	84(1.04)	89(1.10)	**46(0.57)**	19.21
Hepatitis	140	39(1.00)	30(0.77)	43(1.10)	**23(0.59)**	16.96
Primary	305	238(1.00)	202(0.85)	-	**164(0.69)**	56.23
Heart-c	273	50(1.00)	57(1.14)	41(0.82)	**36(0.72)**	20.97
Sonar	187	125(1.00)	186(1.49)	131(1.05)	**99(0.79)**	18.39
Heart-h	265	49(1.00)	**31(0.63)**	47(0.96)	39(0.80)	19.93
Glass	193	118(1.00)	**97(0.82)**	101(0.86)	100(0.85)	27.00
Diabetes	691	234(1.00)	**114(0.49)**	393(1.68)	201(0.86)	25.09
Lymph	133	27(1.00)	40(1.48)	40(1.48)	**24(0.89)**	22.21
Labor	51	13(1.00)	26(2.00)	19(1.46)	**12(0.92)**	15.14
Iris	135	32(1.00)	33(1.03)	125(3.91)	**30(0.94)**	5.25
Credit-g	900	498(1.00)	**213(0.43)**	243(0.49)	495(0.99)	26.36
Breast-w	629	**30(1.00)**	45(1.50)	75(2.50)	39(1.30)	3.94
No. of Wins		1	4	0	10	

DECORATE required 78% of the number of examples that DECORATE used to reach the target error. It is important to note that DECORATE itself achieves the target error with far fewer examples than available in the full training set, as seen by comparing to the total dataset sizes. Hence, improving on the data utilization of DECORATE is a fairly difficult task. Figure 6.1 presents learning curves that clearly demonstrate the advantage of ACTIVEDECORATE. On one dataset, *breast-w*, ACTIVEDECORATE requires a few more examples than DECORATE. This dataset exhibits a ceiling effect in learning, where DECORATE manages to reach the target error rate using only 30 of the 629 available examples, making it difficult to improve on (Figure 6.2).

Our results on error reductions are summarized in Table 6.2. The significant values are presented in bold font. We observed that on almost all datasets, ACTIVEDECORATE produces substantial reductions in error over DECORATE. Furthermore, on 8 of the datasets, ACTIVEDECORATE produces higher reductions in error than the other active-learning methods. Depending on the dataset, ACTIVEDECORATE produces a wide range of improvements, from moderate (4.16% on *credit-g*) to high (70.68% on *vowel*). On average, ACTIVE-

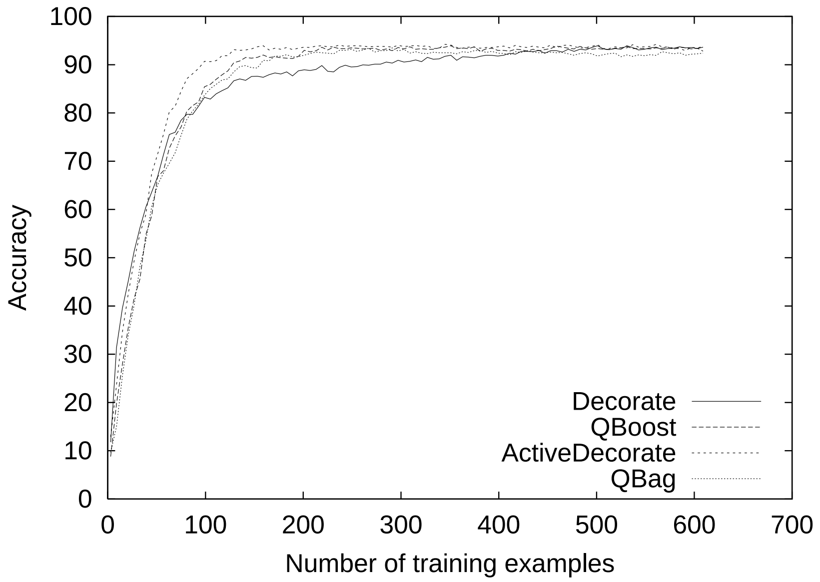


Figure 6.1: Comparing different active learners on *Soybean*.

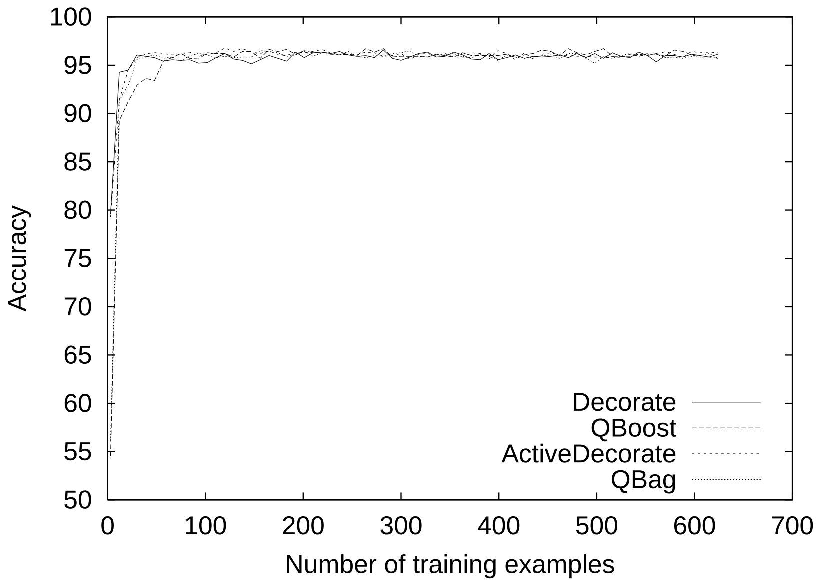


Figure 6.2: Ceiling effect in learning on *Breast-W*.

Table 6.2: Top 20% percent error reduction over Decorate

Dataset	QBag	QBoost	ActiveDecorate
Soybean	**30.50**	**34.17**	**45.84**
Vowel	**22.65**	**42.09**	**70.68**
Statlog	**11.31**	**10.34**	**11.43**
Hepatitis	**12.13**	**16.68**	**19.31**
Primary	**3.23**	0.43	**5.74**
Heart-c	**15.40**	**19.40**	**12.56**
Sonar	1.88	**8.09**	**16.47**
Heart-h	**16.22**	**14.68**	**12.14**
Glass	**10.58**	**16.88**	**15.83**
Diabetes	**8.68**	**4.01**	**5.94**
Lymph	**19.65**	**28.51**	**18.84**
Labor	-2.61	12.55	**36.33**
Iris	**22.78**	1.22	**22.53**
Credit-g	**9.43**	**6.71**	**4.16**
Breast-w	**15.12**	**18.89**	**19.51**
Mean	13.13	15.64	21.15
No. of Wins	4	3	8

DECORATE produces a 21.15% reduction in error.

6.4 Additional Experiments

6.4.1 Jensen-Shannon Divergence

There are two main aspects to any Query by Committee approach. The first is the method employed to construct the committee, and the second is the measure used to rank the utility of unlabeled examples given this committee. Thus far, we have only compared different methods for constructing the committees. Following Abe and Mamitsuka (1998), we ranked unlabeled examples based on the margin of the committee's prediction for the example.

An alternate approach is to use an information theoretic measure such as Jensen-Shannon (JS) divergence (Lin, 1991) to evaluate the potential utility of examples. JS-divergence is a measure of the "distance" between two probability distributions which can also be generalized to measure the distance (similarity) between a finite number of distributions (Dhillon, Mallela, & Kumar, 2002). JS-divergence is a natural extension of the Kullback-Leibler (KL) divergence to a set of distributions. KL divergence is defined between two distributions, and the JS-divergence of a set of distributions is the average KL divergence of each distribution to the mean of the set. Unlike KL divergence, JS-divergence is a true metric and is bounded. If a classifier can provide a distribution of class membership probabilities for a given example, then we can use JS-divergence to compute a measure of similarity between the distributions produced by a set (ensemble) of such classifiers. If $P_i(x)$ is the class probability distribution given by the i-th classifier for the example x (which we will abbreviate as P_i) we can then compute the JS-divergence of a set of size n as:

$$JS(P_1, P_2, ..., P_n) = H(\sum_{i=1}^{n} w_i P_i) - \sum_{i=1}^{n} w_i H(P_i)$$

where w_i is the vote weight of the i-th classifier in the set;[1] and $H(P)$ is the Shannon entropy of the distribution $P = \{p_j : j = 1, ..., K\}$, defined as:

$$H(P) = -\sum_{j=1}^{K} p_j \, log \, p_j$$

Higher values for JS-divergence indicate a greater spread in the predicted class probability distributions, and it is zero if and only if the distributions are identical. A similar measure was used for active learning for text categorization by McCallum and Nigam (1998).

We implemented a version of ACTIVEDECORATE that selects the unlabeled examples with the highest JS-divergence. This measure incorporates more information about the predicted class distribution than using margins, and hence could result in the selection of more informative examples. To test the effectiveness of using JS-divergence, we ran experiments comparing it to using the margin measure. The experiments were conducted as described in Section 6.3.1. Table 6.3 summarizes the results of the comparison of the two

[1]Our experiments use uniform vote weights, normalized to sum to one.

Table 6.3: Comparing measures of utility: Data utilization and top 20% error reduction with respect to Decorate.

	Data Utilization		%Error Reduction	
Dataset	Margins	JS Div.	Margins	JS Div.
Soybean	**144(0.29)**	369(0.75)	**45.84**	18.67
Vowel	**477(0.57)**	525(0.62)	**70.68**	63.26
Statlog	**46(0.57)**	76(0.94)	11.43	**11.52**
Hepatitis	23(0.59)	**19(0.49)**	**19.31**	15.90
Primary	**164(0.69)**	212(0.89)	**5.74**	3.84
Heart-c	36(0.72)	**28(0.56)**	12.56	**13.97**
Sonar	99(0.79)	**94(0.75)**	16.47	**16.71**
Heart-h	39(0.80)	**38(0.78)**	**12.14**	10.81
Glass	**100(0.85)**	118(1.00)	**15.83**	10.46
Diabetes	201(0.86)	**150(0.64)**	**5.94**	5.03
Lymph	24(0.89)	**20(0.74)**	**18.84**	12.18
Labor	12(0.92)	**10(0.77)**	**36.33**	29.77
Iris	**30(0.94)**	41(1.28)	22.53	**23.01**
Credit-g	495(0.99)	**330(0.66)**	**4.16**	3.91
Breast-w	**39(1.30)**	45(1.50)	**19.51**	19.20
Mean	0.78	0.83	21.15	17.22
No. of Wins	7	8	11	4

measures. All the error reductions are significant ($p < 0.05$), so we only present the better of the two columns in bold font. In terms of data utilization, the methods seem equally matched; JS-divergence performs better than margins on 8 of the 15 datasets. However, on the error reduction metric, using margins outperforms JS-divergence on 11 of the datasets. The results also show, that there are datasets on which JS-divergence and margins achieve the target error rate with comparable number of examples, but the error reduction produced by margins is higher. Figure 6.3 clearly demonstrates this phenomenon.

Note that while ACTIVEDECORATE using either measure of utility produces substantial error reductions, in general using margins produces greater improvements. Using the JS-divergence measure tends to select examples that would reduce the uncertainty of the predicted class membership probabilities, which helps to improve classification accuracy. On the other hand, using margins focuses more directly on determining the decision bound-

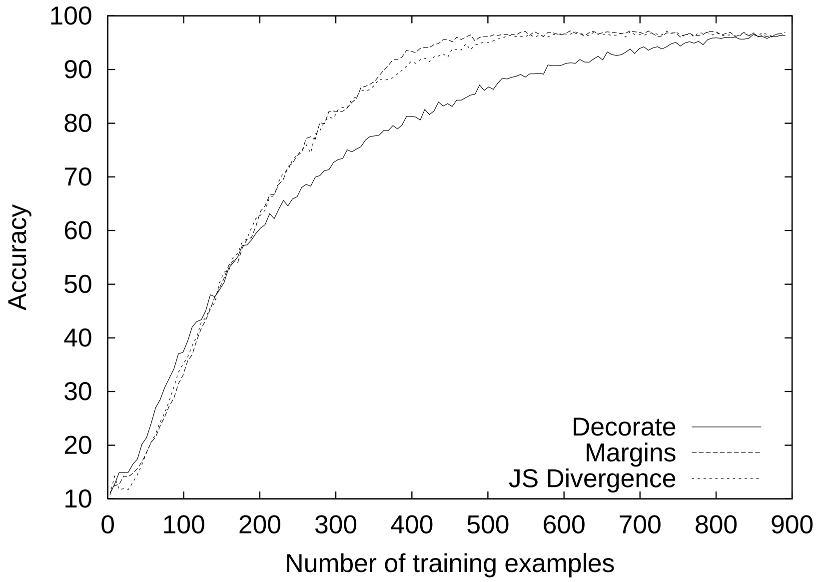


Figure 6.3: Comparing measures of utility: JS Divergence vs Margins on *Vowel*.

ary. This may account for its better performance. For making cost-sensitive decisions, it is very useful to have accurate class probability estimates (Saar-Tsechansky & Provost, 2001). In such cases, we conjecture that using JS-divergence could be a more effective approach. This conjecture is empirically validated in Chapter 7.

6.4.2 Committees for Sample Selection vs. Prediction

All the active learning methods that we have described use committees to determine which examples to select. But in addition to using committees for sample selection, these methods also use the committees for prediction. So we are *not* evaluating which method selects the best queries for the *base learner*, but which combination of sample selection and ensemble method works the best. The fact that ACTIVEDECORATE performs better than QBag may just be testament to the fact that DECORATE performs better than Bagging. However, we claim that not only does DECORATE produce accurate committees, but the committees produced are also more effective in sample selection. To verify this, we implemented an

Table 6.4: Comparing different ensemble methods for selection for Active-Decorate: Percentage error reduction over Decorate.

Dataset	Maximum Train Size	Select w/ Bagging	Select w/ AdaBoost	Select w/ Decorate
Soybean	300	**18.55**	**17.27**	**27.38**
Glass	100	**6.57**	**4.72**	**8.85**
Primary	200	0.2	**2.46**	**3.75**
Statlog	100	-1.79	-1.18	1.73

alternate version of ACTIVEDECORATE, where at each iteration a committee constructed by Bagging is used to select the examples given to DECORATE. In this way, we separate the evaluation of the method used for sample selection from the method used for prediction. Similarly, we implemented a version of ACTIVEDECORATE using ADABOOST to perform the sample selection.

We compared the three methods of sample selection for DECORATE on four of the datasets on which ACTIVEDECORATE exhibited good performance. We generated learning curves as described in Section 6.3.1. However, we did not run the learning curve trials until all the available training data was exhausted, since the active learning methods need fewer examples to achieve the target error rates.

The error reductions over DECORATE averaged across all the points on the learning curve are presented in Table 6.4.[2] The significant error reductions are shown in bold. The table also includes the maximum training set size, which corresponds to the last point on the learning curve. The results show that, on 3 of the 4 datasets, using any of the ensemble sample selection methods in conjunction with DECORATE produces better results than DECORATE. Furthermore, DECORATE committees select more informative examples for training DECORATE than the other committee sample selection methods. These trends are clearly seen in Figure 6.4. It would also be interesting to run similar experiments, using DECORATE ensembles to pick examples for training Bagging, ADABOOST, or J48.

[2]These results are not directly comparable to those in Table 6.2.

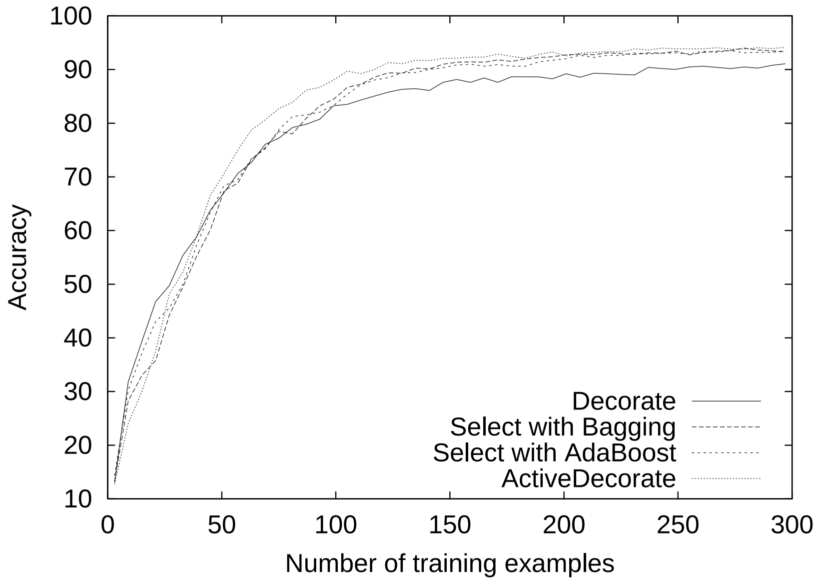


Figure 6.4: Comparing different ensembles methods for selecting samples for DECORATE on *Soybean*.

6.5 Related Work

In their QBC approach, Dagan and Engelson (1995) measure the utility of examples by *vote entropy*, which is the entropy of the class distribution, based on the majority votes of each committee member. McCallum and Nigam (1998) showed that *vote entropy* does not perform as well as JS-divergence for pool-based sample selection. Another recently developed effective committee-based active learner is Co-Testing (Muslea, Minton, & Knoblock, 2000); however, it requires two redundant *views* of the data. Since most data sets do not have redundant views, Co-Testing has rather limited applicability. Another general approach to sample selection is *uncertainty sampling* (Lewis & Catlett, 1994); however, this approach requires a learner that accurately estimates the uncertainty of its decisions, and tends to over-sample the boundaries of its current incomplete hypothesis (Cohn et al., 1994). Finally, *expected-error reduction* methods for active learning (Cohn et al., 1996; Roy & Mc-

Callum, 2001; Zhu, Lafferty, & Ghahramani, 2003) attempt to statistically select training examples that are expected to minimize error on the actual test distribution. This approach has the advantage of avoiding the selection of outliers whose labeling will not improve accuracy on typical examples. However, this method is computationally intense, and must be carefully tailored to a specific learning algorithm (e.g. naive Bayes); and hence, cannot be used to select examples for an arbitrary learner. Active meta-learners like Query by Bagging/Boosting and ACTIVEDECORATE have the advantage of being able to select queries to improve any learner appropriate for a given domain.

6.6 Chapter Summary

ACTIVEDECORATE is a simple, yet effective approach to active learning for improving classification accuracy. Experimental results show that, in general, this approach leads to more effective sample selection than Query by Bagging and Query by Boosting. On average, ACTIVEDECORATE requires only 78% of the number of training examples required by DECORATE with random sampling. As shown in Section 4.4, for small training sets DECORATE produces more diverse ensembles than Bagging or ADABOOST. We believe this increased diversity is the key to ACTIVEDECORATE's superior performance.

Our results also show that using JS-divergence to evaluate the utility of examples is less effective for improving classification accuracy than using margins. JS-divergence may be a better measure when the objective is improving class probability estimates. This conjecture is explored in detail in the next chapter.

Chapter 7

Active Learning for Class Probability Estimation

Many supervised learning applications require more than a simple classification of instances. Often, also having accurate Class Probability Estimates (CPEs) is critical for the task. Class probability estimation is a fundamental concept used in a variety of applications including marketing, fraud detection and credit ranking. For example, in direct marketing the probability that each customer would purchase an item is employed in order to optimize marketing budget expenditure. Similarly, in credit scoring, class probabilities are used to estimate the utility of various courses of actions, such as the profitability of denying or approving a credit application. While prediction accuracy of CPE improves with the availability of more labeled examples, acquiring labeled data is sometimes costly. For example, customers' preferences may be induced from customers' responses to offerings; but solicitations made to acquire customer responses (labels) may be costly, because unwanted solicitations can result in negative customer attitudes. It is therefore beneficial to use active learning to reduce the number of label acquisitions necessary to obtain a desired CPE accuracy.

Almost all prior work in active learning has focused on acquisition policies for inducing accurate *classification* models and thus are aimed at improving classification accuracy. Although active learning algorithms for classification can be applied for learning accurate CPEs, they may not be optimal. Active learning algorithms for classification may (and indeed should) avoid acquisitions that can improve CPEs but are not likely to impact classification. Accurate classification only requires that the model accurately assigns the highest CPE to the correct class, even if the CPEs across classes may be inaccurate. Therefore, to perform well, active learning methods for classification ought to acquire labels of examples that are likely to change the rank-order of the most likely class. To improve CPEs, however, it is necessary to identify potential acquisitions that would improve the CPE accuracy, regardless of the implications for classification accuracy.

In Chapter 6, we introduced a method, ACTIVEDECORATE, for active learning for classification. Melville, Yang, Saar-Tsechansky, and Mooney (2005) extended this work to active learning for probability estimation. In particular, we propose the use of Jensen-Shannon (JS) divergence (Section 6.4.1) to measure the utility of acquiring labels for examples, when the objective is to improve class probability estimates. In this chapter, we demonstrate that, for the task of active learning for CPE, ACTIVEDECORATE using JS-divergence indeed performs significantly better than using margins.

To the best of our knowledge, Bootstrap-LV (Saar-Tsechansky & Provost, 2001) is the only prior approach to active probability estimation. This methods was designed specifically to improve CPEs for binary class problems. The method acquires labels for examples for which the current model exhibits high variance for its CPEs. BOOTSTRAP-LV was shown to significantly reduce the number of label acquisitions required to achieve a given CPE accuracy compared to random acquisitions and existing active learning approaches for classification.

This chapter also presents two new active learning approaches based on BOOTSTRAP-LV. In contrast to BOOTSTRAP-LV, the methods we propose can be applied to acquire labels to improve the CPEs of an arbitrary number of classes. The two methods differ by the measures each employs to identify informative examples: the first approach, BOOTSTRAP-JS, employs the JS-divergence measure. The second approach, BOOTSTRAP-LV-EXT, uses a measure of variance inspired by the local variance measure proposed in BOOTSTRAP-LV. We demonstrate that for binary class problems, BOOTSTRAP-JS is superior to BOOTSTRAP-LV. In addition, we establish that for multi-class problems, BOOTSTRAP-JS and BOOTSTRAP-LV-EXT identify particularly informative examples that significantly improve the CPEs compared to random sampling.

7.1 ActiveDecorate and JS-divergence

In the previous chapter, we compared two measures of utility for ACTIVEDECORATE—*margins* and JS-divergence. It was shown that ACTIVEDECORATE using either measure of utility produces substantial error reductions in classification compared to random sampling. However, in general, using margins produces greater improvements. Using JS-divergence tends to select examples that reduce the uncertainty in CPE, which indirectly helps to improve classification accuracy. On the other hand, ACTIVEDECORATE using margins focuses more directly on determining the decision boundary. This may account for its better classification performance. It was conjectured that if the objective is improving CPEs, then JS-divergence may be a better measure.

In this chapter, we validate this conjecture. In addition to using JS-divergence, we made two more changes to the original algorithm, each of which independently improved its performance. First, each example in the unlabeled set is assigned a probability of being sampled, which is proportional to the measure of utility for the example. Instead of selecting the examples with the m highest utilities, we sample the unlabeled set based on the assigned probabilities (as in BOOTSTRAP-LV). This sampling has been shown to improve the selection mechanism as it reduces the probability of adding outliers to the training data and avoids selecting many similar or identical examples (Saar-Tsechansky & Provost, 2004).

The second change we made is in the DECORATE algorithm. DECORATE ensembles are created iteratively; where in each iteration a new classifier is trained. If adding this new classifier to the current ensemble increases the ensemble training error, then this classifier is rejected, else it is added to the current ensemble. In previous work, training error was evaluated using the 0/1 loss function; however, DECORATE can use any loss (error) function. Since we are interested in improving CPE we experimented with two alternate error functions — Mean Squared Error (MSE) and Area Under the Lift Chart (AULC) (defined in Section 7.3.1). Using MSE performed better on the two metrics used, so we present these results in the rest of this chapter. Our approach, ACTIVEDECORATE-JS, is shown in Algorithm 5.

Algorithm 5 ActiveDecorate-JS

Given:
T - set of training examples
U - set of unlabeled training examples
$\mathcal{L}$ - base learning algorithm
n - desired ensemble size
m - size of each sample

1. Repeat until stopping criterion is met
2. Generate an ensemble of classifiers, $C^* = Decorate(\mathcal{L}, T, n)$
3. For each $x_j \in U$
4. $\forall C_i \in C^*$ generate CPE distribution $P_i(x_j)$
5. $score_j = JS(P_1, P_2, ..., P_n)$
6. $\forall x_j \in U, D(x_j) = score_j / \sum_j score_j$
7. Sample a subset S of m examples from U based on the distribution D
8. Remove examples in S from U and add to T
9. Return $Decorate(\mathcal{L}, T, n)$

7.2 Bootstrap-LV and JS-divergence

To the best of our knowledge, Bootstrap-LV (Saar-Tsechansky & Provost, 2001) is the only active learning algorithm designed for learning CPEs. It was shown to require significantly fewer training examples to achieve a given CPE accuracy compared to random sampling and *uncertainty sampling*, which is an active learning method focused on classification accuracy (Lewis & Catlett, 1994). Bootstrap-LV reduces CPE error by acquiring examples for which the current model exhibits relatively high local variance (LV), i.e., the variance in CPE for a particular example. A high LV for an unlabeled example indicates that the model's estimation of its class membership probabilities is likely to be erroneous, and the example is therefore more desirable to be selected for learning.

Bootstrap-LV, as defined by Saar-Tsechansky and Provost (2001) is only applicable to binary class problems. We first provide the details of this method, and then describe how we extended it to solve multi-class problems. Bootstrap-LV is an iterative algorithm that can be applied to any base learner. At each iteration, we generate a set of n bootstrap samples (Efron & Tibshirani, 1993) from the training set, and apply the given learner $\mathcal{L}$ to each sample to generate n classifiers $C_i : i = 1, ..., n$. For each example in the unlabeled set U, we compute a score which determines its probability of being selected, and which is proportional to the variance of the CPEs. More specifically, the score for example x_j is computed as $(\sum_{i=1}^{n} (p_i(x_j) - \overline{p}_j)^2)/\overline{p}_{j,min}$; where $p_i(x_j)$ denotes the estimated probability the classifier C_i assigns to the event that example x_j belongs to class 0 (the choice of performing the calculation for class 0 is arbitrary, since the variance for both classes is identical), $\overline{p}_j$ is the average estimate for class 0 across classifiers C_i, and $\overline{p}_{j,min}$ is the average probability estimate assigned to the minority class by the different classifiers. Saar-Tsechansky and Provost (2001) attempt to compensate for the under-representation of the minority class by introducing the term $\overline{p}_{j,min}$ in the utility score. The scores produced for the set of unlabeled examples are normalized to produce a distribution, and then a subset of unlabeled examples are selected based on this distribution. The labels for these examples are acquired and the process is repeated.

The model's CPE variance allows the identification of examples that can improve CPE accuracy. However as noted above, the local variance estimated by Bootstrap-LV captures the CPE variance of a single class and thus is not applicable to multi-class problems. Since we have a set of probability distributions for each example, we can instead, use an information theoretic measure, such as JS-divergence to measure the utility of an example. The advantage to using JS-divergence is that it is a distance measure for probability distributions (Lin, 1991) that can be used to capture the uncertainty of the class distribution estimation; and furthermore, it naturally extends to distributions over multiple classes. We propose a variation of BOOTSTRAP-LV, where the utility score for each example is computed as the JS-divergence of the CPEs produced by the set of classifiers C_i. This approach, BOOTSTRAP-JS, is presented in Algorithm 6.

Our second approach, BOOTSTRAP-LV-EXT, is inspired by the Local Variance concept proposed in BOOTSTRAP-LV. For each example and for each class, the variance in the prediction of the class probability across classifiers $C_i, i = 1, ..., n$ is computed, capturing the uncertainty of the CPE for this class. Subsequently, the utility score for each potential acquisition is calculated as the mean variance across classes, reflecting the average uncertainty in the estimations of all classes. Unlike BOOTSTRAP-LV, BOOTSTRAP-LV-EXT does not incorporate the factor of $\overline{p}_{j,min}$ in the score for multi-class problems.

7.3 Experimental Evaluation

7.3.1 Methodology

To evaluate the performance of the different active CPE methods, we ran experiments on 24 representative data sets from the UCI repository (Blake & Merz, 1998). 12 of these datasets were two-class problems, the rest being multi-class. For three datasets (*kr-vs-kp*, *sick*, and *optdigits*), we used a random sample of 1000 instances to reduce experimentation time.

Algorithm 6 Bootstrap-JS

Given:
T - set of training examples
U - set of unlabeled training examples
$\mathcal{L}$ - base learning algorithm
n - number of bootstrap samples
m - size of each sample

1. Repeat until stopping criterion is met
2. Generate n bootstrap samples $B_i, i = 1, ..., n$ from T
3. Apply learner $\mathcal{L}$ to each sample B_i to produce classifier C_i
4. For each $x_j \in U$
5. $\forall C_i$ generate CPE distribution $P_i(x_j)$
6. $score_j = JS(P_1, P_2, ..., P_n)$
7. $\forall x_j \in U, D(x_j) = score_j / \sum_j score_j$
8. Sample a subset S of m examples from U based on the distribution D
9. Remove examples in S from U and add to T
10. Return $C = \mathcal{L}(T)$

All the active learning methods we discuss in this chapter are meta-learners, i.e., they can be applied to any base learner. For our experiments, as a base classifier we use a Probability Estimation Tree (PET) (Provost & Domingos, 2003), which is an unpruned C4.5 decision tree for which Laplace correction is applied at the leaves. Saar-Tsechansky and Provost (2001) showed that using Bagged-PETs for prediction produced better probability estimates than single PETs for BOOTSTRAP-LV; so we used Bagged-PETs for both BOOTSTRAP-LV and BOOTSTRAP-JS. The number of bootstrap samples and the size of ensembles in ACTIVEDECORATE was set to 15.

The performance of each algorithm was averaged over 10 runs of 10-fold cross-validation. In each fold of cross-validation, we generated learning curves as follows. The set of available training examples was treated as an unlabeled pool of examples, and at each iteration the active learner selected a sample of points to be labeled and added to the training set. Each method was allowed to select a total of 33 batches of training examples, measuring performance after each batch in order to generate a learning curve. To reduce computation costs, and because of diminishing variance in performance for different selected examples along the learning curve, we incrementally selected larger batches at each acquisition phase. The resulting curves evaluate how well an active learner orders the set of available examples in terms of utility for learning CPEs. As a baseline, we used random sampling, where the examples in each iteration were selected randomly.

To the best of our knowledge, there are no publicly-available datasets that provide true class probabilities for instances; hence there is no direct measure for the accuracy of CPEs. Instead, we use two indirect metrics proposed in other studies for CPEs (Zadrozny & Elkan, 2001). The first metric is squared error, which is defined for an instance x_j, as $\sum_y (P_{true}(y|x_j) - P(y|x_j))^2$; where $P(y|x_j)$ is the predicted probability that x_j belongs to class y, and $P_{true}(y|x_j)$ is the true probability that x_j belongs to y. We compute the Mean Squared Error (MSE) as the mean of this squared error for each example in the test set. Since we only know the true class labels and not the probabilities, we define $P_{true}(y|x_j)$ to be 1 when the class of x_j is y and 0 otherwise. Given that we are comparing with this extreme distribution, squared error tends to favor classifiers that produce accurate classification, but with extreme probability estimates. Hence, we do not recommend using this metric by itself.

The second measure we employ is the area under the lift chart (AULC) (Nielsen, 2004), which is computed as follows. First, for each class k, we take the $\alpha\%$ of instances with the highest probability estimates for class k. r_α is defined to be the proportion of these instances actually belonging to class k; and r_{100} is the proportion of all test instances that are from class k. The lift $l(\alpha)$, is then computed as $\frac{r_\alpha}{r_{100}}$. The AULC$_k$ is calculated by numeric integration of $l(\alpha)$ from 0 to 100 with a step-size of 5. The overall AULC is computed as the weighted-average of AULC$_k$ for each k; where AULC$_k$ is weighted by the prior class probability of k according to the training set. AULC is a measure of how good the probability estimates are for ranking examples correctly, but not how accurate the estimates are. However, in the absence of a direct measure, an examination of MSE and AULC in tandem provides a good indication of CPE accuracy. We also measured log-loss or cross-entropy, but these results were highly correlated with MSE, so we do not report them here.

To effectively summarize the comparison of two algorithms, we compute the percentage reduction in MSE of one over the other, averaged along the points of the learning curve. We consider the reduction in error to be *significant* if the difference in the errors of the two systems, averaged across the points on the learning curve, is determined to be statistically significant according to paired t-tests ($p < 0.05$). Similarly, we report the percentage *increase* in AULC, since a larger AULC usually implies better probability estimates.

7.3.2 Results

The results of all our comparisons are presented in Tables 7.1-7.3. In each table we present two active learning methods compared to random sampling as well as to each other. We present the statistics *% MSE reduction* and *% AULC increase* averaged across the learning curves. All statistically significant results are presented in bold font. The bottom of each table presents the win/draw/loss (w/d/l) record; where a win or loss is only counted if the improved performance is determined to be significant as defined above.

7.3.3 ActiveDecorate: JS-divergence versus Margins

Table 7.1 shows the results of using JS-divergence versus margins for ACTIVEDECORATE. In Chapter 6, it was shown that ACTIVEDECORATE, with both these measures, performs very well on the task of active learning for classification. Our results here confirm that both measures are also effective for active learning for CPE. ACTIVEDECORATE using mar-

Table 7.1: ACTIVEDECORATE-JS versus Margins

Data set	% MSE Reduction			% AULC Increase		
	Margin vs. Rand.	JS vs. Rand.	JS vs. Margin	Margin vs. Rand.	JS vs. Rand.	JS vs. Margin
breast-w	**9.32**	**23.91**	**12.73**	**0.29**	-0.50	**-0.79**
colic	**8.65**	**17.99**	**10.17**	**4**	**2.44**	**-1.47**
credit-a	**15.83**	**21.97**	**7.08**	**2.85**	**2.98**	0.07
credit-g	**7.06**	**8.91**	**2.02**	**6.98**	**7.79**	**0.75**
diabetes	**-3.11**	0.07	**2.9**	**4.98**	**0.84**	**-3.94**
heart-c	**4.66**	**6.3**	**1.72**	**1.54**	**0.53**	**-0.99**
hepatitis	**4.49**	**7.34**	**2.99**	**1.93**	**0.14**	**-1.95**
ion	**29.23**	**36.51**	**10.01**	**5.73**	**5.53**	-0.2
kr-vs-kp	**34**	**65.27**	**50.77**	**6.46**	**2.19**	**-3.99**
sick	**39.18**	**64.38**	**42.24**	**10.49**	**9.11**	**-1.24**
sonar	**9.3**	**9.31**	0.15	**5.84**	**5.37**	-0.41
vote	**12.15**	**45.79**	**38.12**	**0.81**	**-0.51**	**-1.31**
anneal	**45.51**	**63.8**	**32.1**	**7.62**	**11.14**	**3.27**
autos	**8.32**	**11.38**	**3.57**	**15.34**	**11.52**	**-3.34**
balance-s	**14.1**	**24.63**	**12.05**	**5.24**	**6.14**	**0.86**
car	**2.9**	**53.32**	**52.27**	**5.56**	**16.23**	**10.3**
glass	**7.62**	**12.31**	**5.02**	**8.62**	**10.51**	**1.82**
hypo	**31.37**	**89.87**	**86.34**	**4.03**	**4.7**	**0.65**
iris	**-1.32**	**34.32**	**32.7**	**-1.56**	**1.52**	**3.16**
nursery	**2.62**	**69.99**	**69.52**	**0.56**	**6.43**	**5.9**
optdigits	**32.56**	**39.8**	**10.67**	**19.38**	**17.79**	**-1.4**
segment	**56.95**	**71.12**	**27.27**	**6.11**	**6.85**	**0.71**
soybean	**15.82**	**21.84**	**7.42**	**21.1**	**34.35**	**10.89**
wine	**17.09**	**28.85**	**13.81**	**1.66**	**1.17**	**-0.5**
w/d/l	22/0/2	23/1/0	23/1/0	23/0/1	22/2/0	10/3/11

gins focuses on picking examples that reduce the uncertainty of the classification boundary. Since having better probability estimates usually improves accuracy, it is not surprising that a method focused on improving classification accuracy selects examples that may also improve CPE. However, using JS-divergence directly focuses on reducing the uncertainty in probability estimates and hence performs much better on this task than margins. On the AULC metric both measures seem to perform comparably; however, on MSE, using JS-divergence shows clear and significant advantages over using margins. As noted above, one needs to analyze a combination of these metrics to effectively evaluate any active CPE method. Figure 7.1 presents the comparison of ACTIVEDECORATE-JS versus using margins on the AULC metric on *glass*. The two methods appear to be comparable, with JS-divergence performing better earlier in the curve and margins performing better later. However, when the two methods are compared on the same dataset, using the MSE metric (Figure 7.2), we note that JS-divergence outperforms margins throughout the learning curve. Based on the combination of these results, we may conclude that using JS-divergence is more likely to produce accurate CPEs for this dataset. This example reinforces the need for examining multiple metrics.

7.3.4 Bootstrap-JS, Bootstrap-LV and Bootstrap-LV-EXT

We first examine the performance of BOOTSTRAP-JS for binary-class problems and compared it with that of BOOTSTRAP-LV and of random sampling. As shown in Table 7.2, BOOTSTRAP-JS often exhibits significant improvements over BOOTSTRAP-LV, or is otherwise comparable to BOOTSTRAP-LV. For all data sets, BOOTSTRAP-JS shows substantial improvements with respect to examples selected uniformly at random on both MSE and AULC. The effectiveness of BOOTSTRAP-JS can be clearly seen in Figure 7.3. (The plot shows the part of learning curve where the two active learners diverge in performance.)

Since BOOTSTRAP-LV cannot be applied to multi-class problems, we compare BOOTSTRAP-JS and BOOTSTRAP-LV-EXT with acquisitions of a representative set of examples selected uniformly at random. Table 7.3 presents results on multi-class datasets for BOOTSTRAP-JS and BOOTSTRAP-LV-EXT. Both active methods acquire particularly informative examples, such that for a given number of acquisitions, both methods produce significant reductions in error over random sampling. The two active methods perform comparably to each other for most data sets, and JS-divergence performs slightly better in some domains. Because JS-divergence successfully measures the uncertainty of the distribution estimation over all classes, we would recommend using BOOTSTRAP-JS for actively learning CPE models in multi-class domains.

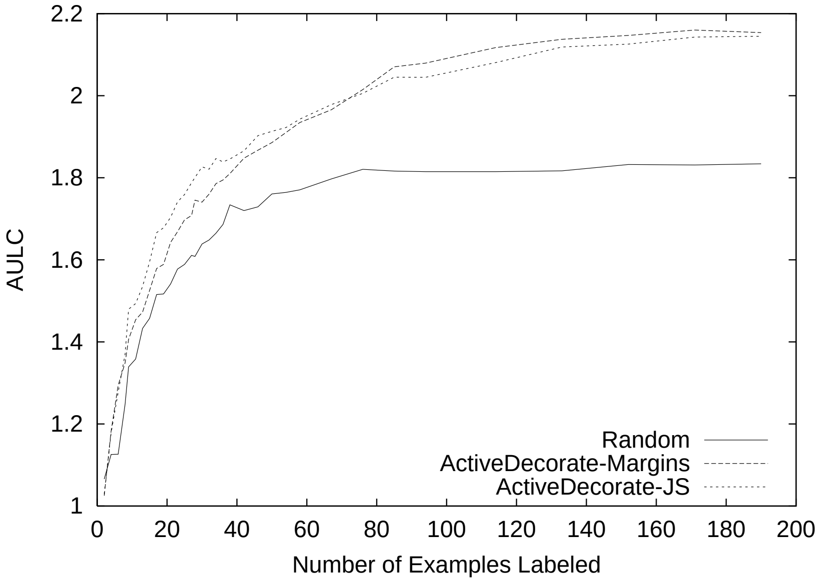


Figure 7.1: Comparing AULC of different algorithms on *glass*

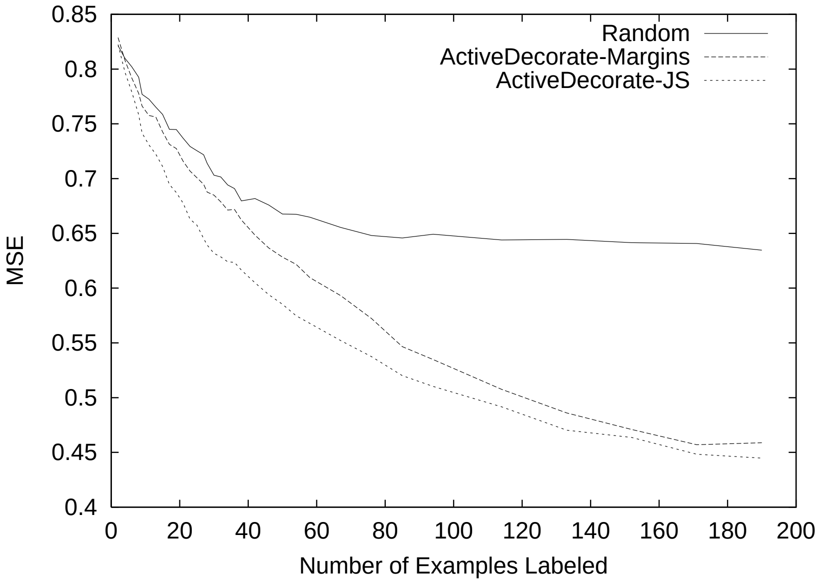


Figure 7.2: Comparing MSE of different algorithms on *glass*

Table 7.2: BOOTSTRAP-JS versus BOOTSTRAP-LV on binary datasets

Data set	%MSE Reduction			%AULC Increase		
	LV vs. Random	JS vs. Random	JS vs. LV	LV vs. Random	JS vs. Random	JS vs. LV
breast-w	**14.92**	**14.81**	-0.12	**0.55**	**0.52**	-0.02
colic	**-1.45**	-0.04	**1.39**	**-0.95**	**-0.56**	**0.41**
credit-a	**2.1**	**3.98**	**1.92**	**-0.49**	-0.01	**0.48**
credit-g	-0.16	**0.77**	**0.93**	-0.01	**0.3**	**0.32**
diabetes	**1.01**	**1.75**	**0.75**	0.18	**0.58**	**0.4**
heart-c	**1.68**	0.29	**-1.43**	**0.57**	-0.08	**-0.64**
hepatitis	0.19	**2.64**	**2.43**	0.19	**1.03**	**0.84**
ion	**10.65**	**12.26**	**1.82**	**1.13**	**0.96**	-0.16
kr-vs-kp	**38.97**	**43**	**8.07**	**1.64**	**1.79**	**0.15**
sick	**19.97**	**20.84**	**1.03**	**0.62**	**0.41**	**-0.21**
sonar	**2.44**	**1.32**	**-1.17**	**0.58**	**0.74**	**0.16**
vote	**6.3**	**9.14**	**3.08**	**0.28**	**0.46**	**0.18**
w/d/l	9/2/1	10/2/0	9/1/2	7/3/2	9/2/1	8/2/2

Table 7.3: BOOTSTRAP-JS versus BOOTSTRAP-LV-EXT on multi-class datasets

Data set	% MSE Reduction			% AULC Increase		
	LV-Ext vs. Rand.	JS vs. Rand.	JS vs. LV-Ext	LV-Ext vs. Rand.	JS vs. Rand.	JS vs. LV-Ext
anneal	**12.27**	**13.06**	**0.89**	0.05	**0.5**	**0.45**
autos	**0.96**	**0.38**	**-0.58**	**1.51**	**0.83**	**-0.66**
balance-s	**1.39**	**0.92**	-0.48	**0.72**	**0.58**	-0.14
car	**7.21**	**6.93**	**-0.31**	**1.53**	**1.41**	**-0.12**
glass	**-0.55**	-0.19	**0.36**	**0.61**	**0.48**	-0.11
hypo	**46.62**	**46.41**	-0.9	**0.49**	**0.47**	-0.02
iris	**6.64**	**10.79**	**4.58**	**0.46**	**0.83**	**0.39**
nursery	**14.37**	**14.25**	-0.20	**0.44**	**0.42**	-0.01
optdigits	0.35	**0.71**	**0.35**	**0.9**	**1.13**	**0.23**
segment	**11.08**	**11.19**	0.08	**0.83**	**0.79**	-0.04
soybean	**1.5**	**0.78**	**-0.74**	**-0.46**	**0.4**	**0.87**
wine	**13.13**	**13.34**	0.36	**1.11**	**1.08**	-0.02
w/d/l	10/1/1	11/1/0	4/5/3	10/1/1	12/0/0	4/6/2

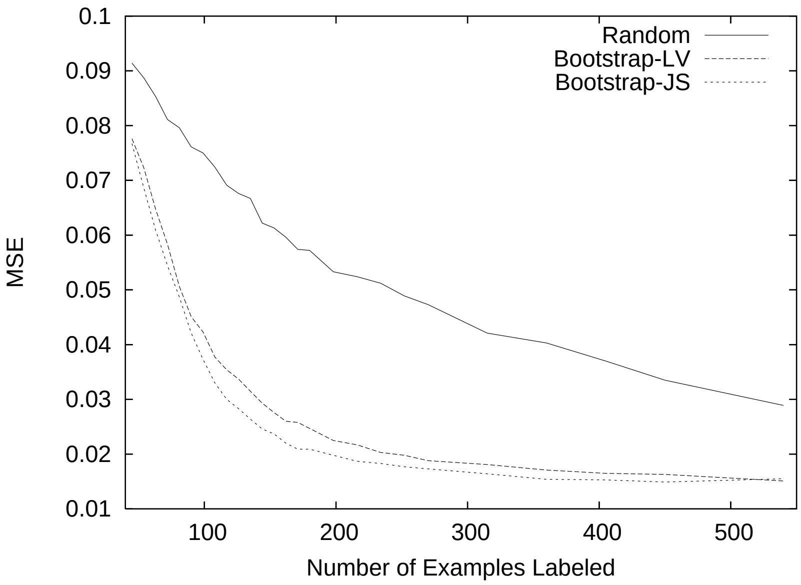


Figure 7.3: Comparing different algorithms on *kr-vs-kp*

7.3.5 ActiveDecorate-JS vs Bootstrap-JS

In addition to demonstrating the effectiveness of JS-divergence, we also compare the two active CPE methods that use JS-divergence. The comparison is made in two scenarios. In the *full dataset* scenario, the setting is the same as in previous experiments. In the *early stages* scenario, each algorithm is allowed to select 1 example at each iteration starting from 5 examples and going up to 20 examples. This characterizes the performance at the beginning of the learning curve. Table 7.4 summarizes the results in terms of win/draw/loss records on the 24 datasets. For the *full dataset*, on the AULC metric, the methods perform comparably, but BOOTSTRAP-JS outperforms ACTIVEDECORATE-JS on MSE. However, for most datasets, ACTIVEDECORATE-JS shows significant advantages over BOOTSTRAP-JS in the *early stages*. These results could be explained by the fact that DECORATE (used byACTIVEDECORATE-JS) has a clear advantage over Bagging (used by BOOTSTRAP-JS) when training sets are small, as explained in Chapter 4.

Table 7.4: BOOTSTRAP-JS vs. ACTIVEDECORATE-JS: Win/Draw/Loss records

	% MSE Reduction	% AULC Increase
Full dataset	18/0/6	13/0/11
Early stages	8/2/14	2/5/17

For DECORATE, we only specify the desired ensemble size; the ensembles formed could be smaller depending on the maximum number of classifiers it is permitted to explore. In our experiments, the desired size was set to 15 and a maximum of 50 classifiers were explored. On average DECORATE ensembles formed by ACTIVEDECORATE-JS are much smaller than those formed by Bagging in BOOTSTRAP-JS. Having larger ensembles generally increases classification accuracy (Melville & Mooney, 2003) and may improve CPE. This may account for the weaker overall performance of ACTIVEDECORATE-JS to BOOTSTRAP-JS; and may be significantly improved by increasing the ensemble size.

7.4 Chapter Summary

In this chapter, we propose the use of Jensen-Shannon divergence as a measure of the utility of acquiring labeled examples for learning accurate class probability estimates. Extensive experiments have demonstrated that JS-divergence effectively captures the uncertainty of class probability estimation and allows us to identify particularly informative examples that significantly improve the model's class distribution estimation. In particular, we show that when JS-divergence is used with ACTIVEDECORATE, an active learner for classification, it produces substantial improvements over using margins, which focuses on classification accuracy. We have also demonstrated that for binary-class problems, BOOTSTRAP-JS which employs JS-divergence to acquire training examples is either comparable or significantly superior to BOOTSTRAP-LV, an existing active CPE learner for binary class problems. BOOTSTRAP-JS maintains its effectiveness for multi-class domains as well: it acquires informative examples which result in significantly more accurate models as compared to models induced from examples selected uniformly at random. Furthermore, our results indicate that, on average, BOOTSTRAP-JS with Bagged-PETs is a preferable method for active CPE compared to ACTIVEDECORATE-JS. However, if one is concerned primarily with the early stages of learning, then ACTIVEDECORATE-JS has a significant advantage.

Chapter 8

Active Feature-value Acquisition

Unlike the active learning setting, in many predictive modeling tasks, the class labels for all instances are known, but feature values may be missing and can be acquired at a cost. For building accurate models, ignoring instances with missing values leads to inferior model performance (Quinlan, 1989; Leigh & James, 2004), while acquiring complete information for all instances often is prohibitively expensive or unnecessary. To reduce the cost of acquiring feature information, it is desirable to identify a subset of the instances for which complete information is most informative to acquire.

The setting we explore was first introduced by Zheng and Padmanabhan (2002), and applies to a variety of business and other domains. Consider an on-line retailer learning a predictive model to estimate customers' propensities to buy. The retailer may use private information on its customers and their buying behavior over time, as captured from the retailer's own web log-files. To improve the model, the retailer may also acquire additional information capturing its customers' buying preferences and lifestyle choices from a third-party information intermediary (Hagel & Singerare, 1999). Acquiring complete data for all customers may be prohibitively expensive (New York Times, 1999). Hence, the retailer could benefit from having a cost-efficient feature acquisition strategy that can select the customers it should acquire complete information for, so as to most benefit the predictive model. A similar challenge is faced by marketing research firms that, in order to model consumer behavior, often obtain consumer responses to a short survey, and due to the cost of acquiring information, acquire responses to an extended survey from only a small, representative subset of those consumers. An effective acquisition strategy that acquires complete responses from consumers that are particularly informative for the model, can increase the accuracy of the model compared to that induced with the default strategy.

In this chapter, we address this problem of *active feature-value acquisition* (AFA) for classifier induction (Melville et al., 2004): given a model built on incomplete training data, identify the instances with missing values for which acquiring complete feature information will result in the greatest increase in model accuracy. Formally, assume m instances, each represented by n features $a_1, ..., a_n$. For all instances, the values of a subset of the features $a_1, \ldots, a_i$ are known, along with the class labels. The values of the remaining features $a_{i+1}, \ldots, a_n$ are unknown and can be acquired at a cost.

The approach we present for active feature acquisition is based on the following three observations: **(1)** Most classification models provide estimates of the confidence of classification, such as estimated probabilities of class membership. Therefore principles underlying existing active-learning methods like uncertainty sampling (Cohn et al., 1994) can be applied. **(2)** For the data items subject to active feature-value acquisition, the correct classifications are known during training. Therefore, unlike with traditional active learning, it is possible to employ direct measures of the current model's accuracy for estimating the value of potential acquisitions. **(3)** Class labels are available for all complete and incomplete instances. Therefore, we can exploit all instances (including incomplete instances) to induce models, and to guide feature acquisition.

The approach we propose is simple-to-implement, computationally efficient and results in significant improvements compared to random sampling and a computationally-intensive method proposed earlier for this problem (Zheng & Padmanabhan, 2002).

8.1 Task Definition and Algorithm

8.1.1 Pool-based Active Feature Acquisition

Assume a classifier induction problem, where each instance is represented with n feature values and a class label. For a subset G of the training set T, the values of all n features are known. We refer to these instances as complete instances. For all other instances in T, only the values of a subset of the features $a_1, \ldots, a_i$ are known. The values of the remaining features $a_{i+1}, \ldots, a_n$ are missing and the set can be acquired at a fixed cost. We refer to these instances as incomplete instances, and the set of all incomplete instances is denoted as I. The class labels of all instances in T are known.

Unlike prior work (Zheng & Padmanabhan, 2002), we assume that models are induced from the entire training set (rather than just from G). This is because both parametric and non-parametric models induced from all available data have been shown to be superior to models induced when instances with missing values are ignored (Leigh & James, 2004). Beyond improved accuracy, the choice of model induction setting also bears important implications for the active acquisition mechanism, because the estimation of an acquisition's marginal utility is derived with respect to the model. We discuss this issue and its implications in detail in Section 8.3. Note that some induction algorithms (e.g., C4.5) include an internal mechanism for incorporating instances with missing feature-values (Quinlan, 1989); other induction algorithms require that missing values be imputed first before induction is performed (Leigh & James, 2004). For the latter learners, many imputation mechanisms are available to fill in missing values (e.g., multiple imputation, nearest neighbor) (Little & Rubin, 1987; Batista & Monard, 2003)). Henceforth, we assume that the induction algorithm includes some treatment for instances with missing values.

We study active feature-value acquisition policies within a generic iterative framework, shown in Algorithm 7. Each iteration estimates the utility of acquiring complete feature information for each available incomplete example. The missing feature values of a subset $S \in I$ of incomplete instances with the highest utility values are acquired and added to T (these examples move from I to G). A new model is then induced from T, and the process is repeated. Different AFA policies correspond to different measures of utility employed to evaluate the informativeness of acquiring features for an instance. Our baseline policy, random selection, selects acquisitions at random, which implicitly tends to prefer examples from dense areas of the example space (Saar-Tsechansky & Provost, 2004). In this study, we propose the use of *Error Sampling*, described below, which is based on the observations made in the previous section.

Algorithm 7 Active Feature-Value Acquisition Framework

Given:
G - set of complete instances
I - set of incomplete instances
T - set of training instances, $G \cup I$
$\mathcal{L}$ - learning algorithm
m - size of each sample

1. Repeat until stopping criterion is met
2. Generate a classifier, $C = \mathcal{L}(T)$
3. $\forall x_j \in I$, compute $Score(C, x_j)$ based on the current classifier
4. Select a subset S of m instances with the highest utility based on the score
5. Acquire values for missing features for each instance in S
6. Remove instances in S from I and add to G
7. Update training set, $T = G \cup I$
8. Return $\mathcal{L}(T)$

8.1.2 Error Sampling

For a model trained on incomplete instances, acquiring missing feature-values is effective if it enables a learner to capture additional discriminative patterns that improve the model's prediction. Specifically, acquired feature-values are likely to have an impact on subsequent model induction when the acquired values pertain to a misclassified example and may embed predictive patterns that can be potentially captured by the model and improve the model. In contrast, acquiring feature-values of instances for which the current model already embeds correct discriminative patterns is not likely to impact model accuracy considerably. Motivated by this reasoning, our approach *Error Sampling* prefers to acquire feature-values for instances that the current model misclassifies. At each iteration, it randomly selects m incomplete instances that have been misclassified by the model.

If there are fewer than m misclassified instances, then *Error Sampling* selects the remaining instances based on the *Uncertainty* score which we describe next. The notion of uncertainty, in this context, originated in work on optimum experimental design (Federov, 1972) and has been extensively applied in the active learning literature (Cohn et al., 1994; Saar-Tsechansky & Provost, 2004). The *Uncertainty* score captures the model's ability to distinguish between cases of different classes and prefers acquiring information regarding instances whose predictions are most uncertain. The acquisition of additional information for these cases is more likely to impact prediction, whereas information pertaining to strong discriminative patterns captured by the model is less likely to change the model. For a probabilistic model, the absence of discriminative patterns in the data results in the model assigning similar likelihoods for class membership of different classes. Hence, the *Uncertainty* score is calculated as the *margin* (Chapter 6), i.e., the absolute difference between the estimated class probabilities of the two most likely classes. Formally, for an instance x, let $P_y(x)$ be the estimated probability that x belongs to class y as predicted by the model. Then the *Uncertainty* score is given by $P_{y_1}(x) - P_{y_2}(x)$, where $P_{y_1}(x)$ and $P_{y_2}(x)$ are the first-highest and second-highest predicted probability estimates respectively. Formally, the

Error Sampling score for a potential acquisition is set to -1 for misclassified instances; and for correctly classified instances we employ the *Uncertainty* score. At each iteration of the AFA algorithm, complete feature information is acquired for the m incomplete instances with the lowest scores.

8.2 Experimental Evaluation

8.2.1 Methodology

We first compared *Error Sampling* to random feature acquisition. The performance of each system was averaged over five runs of 10-fold cross-validation. In each fold, we generated learning curves in the following fashion. Initially, the learner has access to all incomplete instances, and is given complete feature-values for a randomly selected subset, of size m, of these instances. The learner builds a classifier based on this data. For the active strategies, a sample of instances is then selected from the pool of incomplete instances based on the measure of utility using the current classification model. The missing values for these instances are acquired, making them complete instances. A new classifier is then generated based on this updated training set, and the process is repeated until the pool of incomplete instances is exhausted.

In the case of random selection, the incomplete instances are selected uniformly at random from the pool. Each system is evaluated on the held-out test set after each iteration of feature acquisition. As in the work of Zheng and Padmanabhan (2002), the test data set contains only complete instances, since we want to estimate the true generalization accuracy of the constructed model given complete data. The resulting learning curves evaluate how well an active feature-value acquisition method orders its acquisitions as reflected by model accuracy. Note that, at the end of the learning curve, all algorithms see exactly the same set of complete training instances. To maximize the gains of AFA, it is best to acquire features for a single instance in each iteration; however, to make our experiments computationally feasible, we selected instances in batches of 10 (i.e., sample size m = 10).

We can compare the performance of any two schemes, A and B, by comparing the errors produced by both, given that we are limited to acquiring a fixed number of complete instances. To measure this, we compute the percentage reduction in error of A over B and report the average over all points on the learning curve. The reduction in error is considered to be *significant* if the average errors across the points on the learning curve of A is lower than that of B according to a paired t-test ($p < 0.05$).

As mentioned above, towards the end of the learning curve, all methods will have seen almost all the same training examples. Hence, the main impact of AFA is lower on the learning curve. To capture this, we also report the percentage error reduction averaged over only the 20% of points on the learning curve where the largest improvements are produced. We refer to this as the *top-20% percentage error reduction*, which is similar to a measure reported by Saar-Tsechansky and Provost (2001).

All the experiments were run on 5 web-usage datasets (used by Padmanabhan, Zheng, and Kimbrough (2001)) and 5 datasets from the UCI machine learning repository (Blake & Merz, 1998). The web-usage data contain information from popular on-line retailers about customer behavior and purchases. These datasets exhibit a natural dichotomy, with a subset of features owned by a particular retailer and a set of features that the retailer may acquire at a cost. In particular, each retailer privately owns information about its customers' behavior as captured by web logfiles. The retailer's private data contain features such as user demographics, the time of the session or whether the session occurred on a weekday. These are referred to as *site-centric* features. In addition, the data contain information that is not owned by any individual retailer, capturing each customer's aggregated behavior and purchasing patterns across a variety of on-line retailers. These are referred to as *user-centric* features. The learning task is to induce models to predict whether a customer will purchase an item during a visit to the store. The web usage data has a clear division of features—the first 15 are site-centric and the rest are user-centric. Hence the pool of incomplete instances was initialized with only the first 15 features. We selected several UCI datasets that had more than 25 features. For these datasets, 30% of the features were randomly selected to be used in the incomplete instances. A different set of randomly selected features was used for each train-test split of the data. All the datasets used in this study are summarized in Table 8.1.

Table 8.1: Summary of Data Sets

Name	Instances	Classes	Features
bmg	2417	2	40
expedia	3125	2	40
qvc	2152	2	40
etoys	270	2	40
priceline	447	2	40
anneal	898	6	38
soybean	683	19	35
kr-vs-kp	3196	2	36
hypo	3772	4	29
autos	205	6	25

The AFA framework we have proposed can be implemented using an arbitrary probabilistic classifier as a learner. We experimented with two learners — J48 and DECORATE.

8.2.2 Results using J48 Tree Induction

The results comparing *Error Sampling* to random selection are summarized in Table 8.2. All error reductions reported are statistically significant. The results show that for all data sets using *Error Sampling* significantly improves on the model accuracy compared to random sampling. Figures 8.1 and 8.2 present learning curves that demonstrate the advantage of using an AFA scheme over random acquisition. Apart from average reduction in error, a good indicator of the effectiveness of an active feature-value acquisition scheme is the number of acquisitions required to obtain a desired accuracy. For example, on the *anneal* data set, *Error Sampling* achieves an accuracy of 98% with only 200 acquisitions of complete instances. In contrast, random selection requires more than 400 complete instances to achieve the same accuracy level.

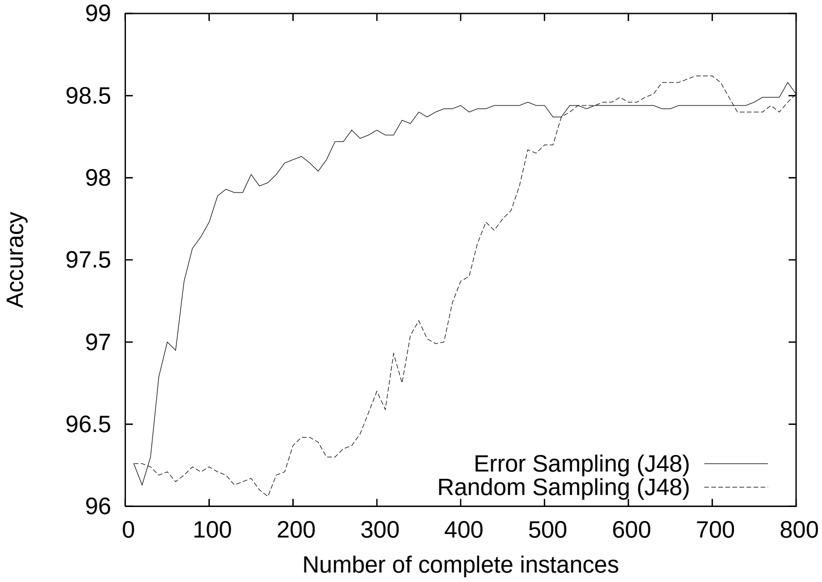


Figure 8.1: *Error Sampling* vs. *Random Sampling* on *anneal*.

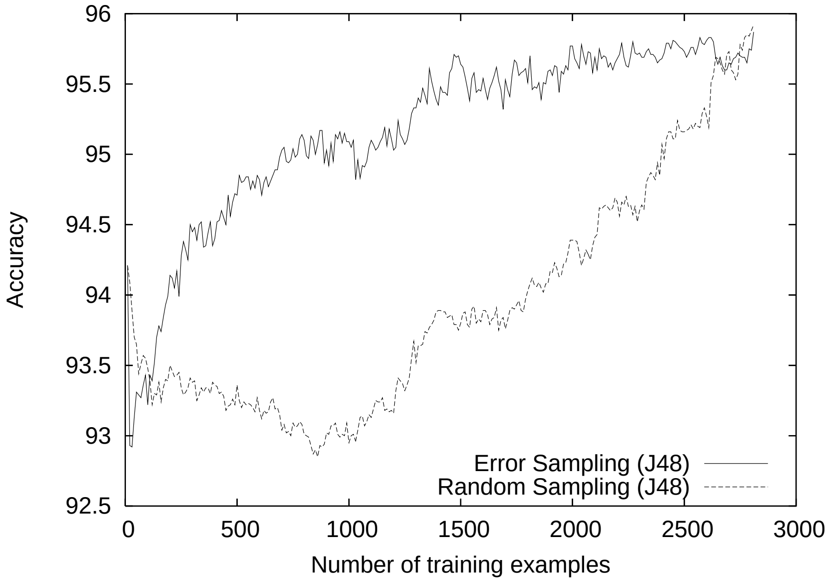


Figure 8.2: *Error Sampling* vs. *Random Sampling* on *expedia*.

Table 8.2: Error reduction of *Error Sampling* with respect to random sampling.

Dataset	%Error Reduction	Top-20% %Err. Red.
bmg	10.67	17.77
etoys	10.34	23.88
expedia	19.83	29.12
priceline	24.45	34.49
qvc	15.44	24.75
anneal	22.65	49.27
soybean	8.03	14.79
autos	4.24	10.50
kr-vs-kr	36.82	53.23
hypo	16.79	40.48
Mean	16.93	29.83

8.2.3 Results on DECORATE

In addition to J48, we also tested *Error Sampling* using DECORATE as a learner. DECORATE is well-suited for this task for the following three reasons: (1) DECORATE ensembles of decision trees produce higher accuracies than single trees (Chapter 4); (2) DECORATE has been successfully used for active learning using the *Uncertainty* measure described here (Chapter 6); and (3) DECORATE is more resilient to missing features than single decision trees, Bagging, and ADABOOST (Chapter 5).

In our experiments, we built DECORATE ensembles of 15 classifiers, using J48 as our base learner and generated learning curves as described in Section 8.2.1. In each iteration of AFA, we selected instances in batches of 20. The results comparing *Error Sampling* to random selection for DECORATE are summarized in Table 8.3. The error reductions on all datasets, except *etoys*, are significant. DECORATE with random sampling is more accurate than single trees; hence, improving on it through active sampling is a more challenging task. But as can be seen from the results, using *Error Sampling* gives considerable improvements in accuracy over DECORATE using random sampling. Figures 8.3 and 8.4 presents datasets which clearly demonstrate the advantage of using active feature-value acquisition over random selection for DECORATE. For example, on *qvc*, once random sampling ac-

Table 8.3: Error reduction of *Error Sampling* with respect to random sampling.

Dataset	%Error Reduction	Top-20% %Err. Red.
bmg	16.26	21.89
etoys	1.93	10.07
expedia	16.75	23.82
priceline	28.31	41.49
qvc	24.44	35.91
anneal	21.41	44.51
soybean	9.99	17.67
autos	3.95	8.49
kr-vs-kr	27.79	54.91
hypo	21.61	39.49
Mean	17.24	29.83

quires approximately 1200 complete instances, it induces a model with an accuracy of 90%; while, *Error Sampling* requires approximately 200 complete instances to achieve the same accuracy. This could translate to a substantial reduction in the cost of data acquisition.

8.3 Comparison with GODA

The most closely related work to ours, is the study by Zheng and Padmanabhan (2002) of the active feature-value acquisition scheme GODA. GODA measures the utility of acquiring feature-values for a particular incomplete instance in the following way. It adds the instance to the training set, imputing the values that are missing. It then induces a new model and measures its performance on the training set. This process is repeated for each incomplete instance, and the instance that leads to the model with the best expected performance is selected for feature-value acquisition.

GODA has an important difference from the methods we have proposed: it induces its models from only the complete instances—ignoring the incomplete instances. Whether one chooses to use or to ignore incomplete instances when inducing a model has a significant bearing on the acquisition scheme. GODA estimates the value of potential acquisitions by the model's improved performance resulting from adding the example to the training

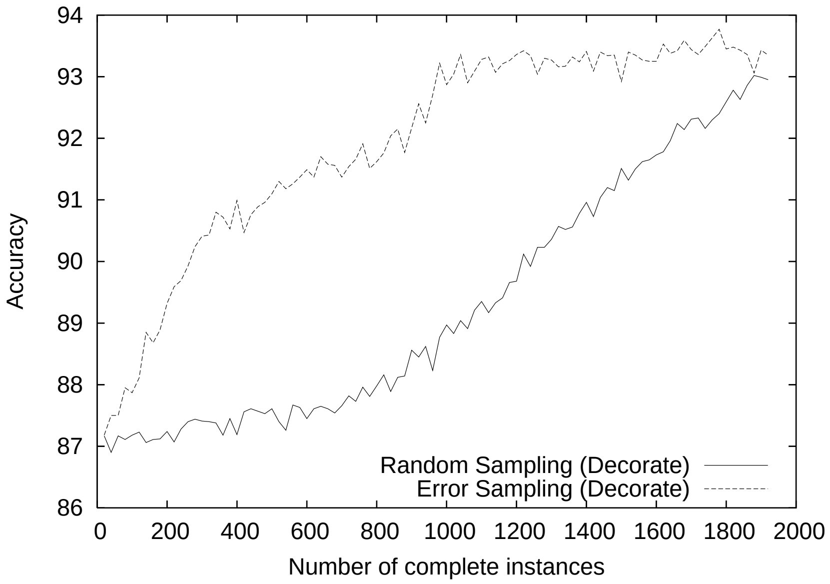


Figure 8.3: *Error Sampling* vs. *Random Sampling* on *qvc*.

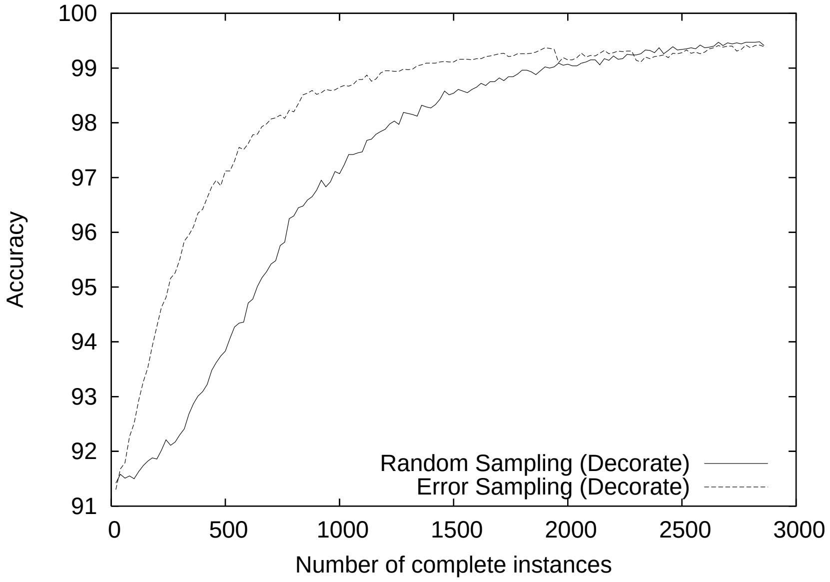


Figure 8.4: *Error Sampling* vs. *Random Sampling* on *kr-vs-kp*.

set. This confounds the improvement due to acquiring the previously unknown feature values with the improvement due to including the already known feature values. In contrast, the policies we propose estimate the marginal utility of missing feature acquisition with respect to a model induced from all available data. GODA's measure of utility cannot be employed directly when the models are induced from all incomplete instances including imputations of their missing features. Nevertheless, since GODA is (to our knowledge) the only other technique designed for the same acquisition setting, it is informative to compare performance with our approach.

To compare to our approach, we implemented GODA as described in (Zheng & Padmanabhan, 2002), using J48 tree induction as the learner and using accuracy as the *goodness measure* of the model. As in (Zheng & Padmanabhan, 2002), we use *multiple imputation* with Expectation-Maximization to impute missing values for incomplete instances. Experiments comparing *Error Sampling* using J48 to GODA were run as in Section 8.2.1. However, due to GODA's tremendous computational requirements, we only ran one run of 10-fold cross-validation on three of the datasets. The datasets were also reduced in size to make running GODA feasible.

A summary of the results, along with the reduced dataset sizes, is presented in Table 8.4. The results show that in spite of the high computational complexity of GODA, it results in inferior performance compared to *Error Sampling* for all three domains. All improvements obtained by *Error Sampling* with respect to GODA are statistically significant. Figure 8.5 presents learning curves for the *priceline* dataset that clearly demonstrate the superior performance of *Error Sampling*. These results suggest that the ability of *Error Sampling* to capitalize on information from incomplete instances, and to utilize this knowledge in feature acquisition, allows it to capture better predictive patterns compared to those captured by GODA.

Recall that when an instance is selected for acquisition, *Error Sampling* adds to the training data only the acquired feature values. GODA, however, adds to the training data the entire instance, i.e., the feature values that are known ex ante (but that are not used for induction by GODA [1]) as well as the acquired feature values and the instance's class membership. Hence, even when the same instance is selected by GODA and by *Error Sampling*, the relative increase in accuracy for GODA is likely to be greater than the increase obtained for a model induced with *Error Sampling*. This difference contributes to the steep learning curve exhibited by the model generated in GODA.

In addition to superior accuracy for a given number of acquisitions, our approach also has the advantage of being simple-to-implement and having a relatively low computational complexity. GODA, on the other hand, requires inducing a different model for estimating each potential acquisition (i.e., $|I|$ models are induced). Hence for even moderately large data sets this approach is prohibitively expensive, except (perhaps) with an incremental learner such as Naive Bayes. Our AFA framework is significantly more efficient because only a single model is induced for estimating the utilities of an arbitrarily large number of potential feature acquisitions.

Table 8.4: Comparing *Error Sampling* with GODA: Percent error reduction.

Dataset	Size	% Error Reduction
bmg	200	19.48
qvc	100	20.03
priceline	100	17.75

8.4 Related Work

Recent work on *budgeted learning* (Lizotte, Madani, & Greiner, 2003) also addresses the issue of active feature-value acquisition. However, the policies developed by Lizotte et al. (2003) assume feature-values are discrete, and consider the acquisition of individual feature-values for instances of a given class (i.e., queries are of the form "acquire value of

[1] This explains why GODA starts with lower accuracy.

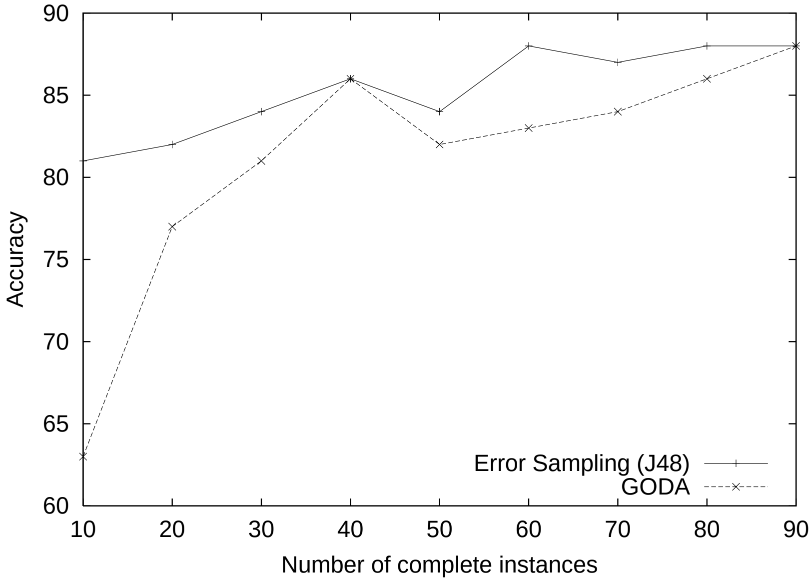


Figure 8.5: Comparing *Error Sampling* to GODA on *priceline*

feature f for some instance in class c."). Therefore, unlike our approach, the policies do not consider requesting additional features for a specific incomplete instance. In addition, the policies cannot be directly applied to estimate the value of acquiring sets of features (as is required in our problem setting). Another important aspect of the policies proposed by Lizotte et al. (2003) is that for each feature and class membership they require estimating the performance of all models induced from each possible value assignment. The induction of most learners is not incremental, hence for each feature class pair, a new model is required to be induced for each value assignment. Although the framework proposed by Lizotte et al. (2003) was not designed to solve the problem discussed here, one may consider an extension to this framework for estimating the utility of acquiring values for a set of features for incomplete instances. However, the number of possible value assignments, and consequently the number of model inductions required will increase considerably. It is unclear whether an algorithm with such a high complexity would be feasible in practice.

Some work on *cost sensitive* learning (Turney, 2000) has addressed the issue of inducing economical classifiers when there are costs associated with obtaining feature values. However, most of this work assumes that the *training* data are complete and focuses on learning classifiers that minimize the cost of classifying incomplete *test* instances. An exception, CS-ID3 (Tan & Schlimmer, 1990), also attempts to minimize the cost of acquiring features during training; however, it processes examples incrementally and can only request additional information for the current training instance. CS-ID3 uses a simple greedy strategy that requests the value of the cheapest unknown feature when the existing hypothesis is unable to correctly classify the current instance. It does not actively select the most useful information to acquire from a pool of incomplete training examples. The LAC* algorithm (Greiner, Grove, & Roth, 2002) also addresses the issue of economical feature acquisition during both training and testing; however, it also adopts a very simple strategy that does not actively select the most informative data to collect during training. Rather, LAC* simply requests complete information on a random sample of instances in repeated *exploration* phases that are intermixed with *exploitation* phases that use the current learned classifier to economically classify instances.

8.5 Chapter Summary

We have presented a general framework for active feature-value acquisition that can be applied to different learners and can use alternate measures of utility for ranking acquisitions. Within this framework, we present an approach in which instances are selected for acquisition based on the current model's accuracy and its confidence in prediction. We show empirically that this approach, *Error Sampling*, significantly improves the accuracy of models learned for fixed feature acquisition budgets, when compared with a policy that requests features randomly. In particular, we have shown that using *Error Sampling* with DECORATE ensembles is very effective for the task of active feature-value acquisition.

A direct comparison of *Error Sampling* with GODA, an alternate AFA approach, demonstrates that in spite of its simplicity, *Error Sampling* exhibits superior performance.

Error Sampling's utilization of all known feature-values and of a simple measure of the potential for improvement from an acquisition, results in advantages both in computation time and model accuracy.

The effectiveness, simplicity, and computational efficiency of *Error Sampling* argues that this policy should be considered by any practitioner or researcher faced with the problem of feature set acquisition. From a research perspective, we suggest that the *Error Sampling* policy be a baseline (in addition to random selection) for future studies of active feature selection.

Chapter 9

Future Work

In this chapter, we discuss some future directions for the research presented in this thesis.

9.1 Further Analysis on DECORATE

Our current study has focused primarily on building ensembles of decision trees. However, DECORATE, being a meta-learner, can be applied to any learning algorithm. One direction for future work is to experiment with other base learners. Initial experiments on applying DECORATE to neural networks look promising. It would be good to perform a more thorough study, and compare DECORATE to diversity-based ensemble methods designed specifically for neural networks (Opitz & Shavlik, 1996; Rosen, 1996; Liu & Yao, 1999).

DECORATE has been tested extensively on many datasets from the UCI repository. However, these datasets are fairly low-dimensional, having at most a few hundred features. It would be useful to see how effective DECORATE is for domains with high-dimensional data, having tens of thousands of features, such as text categorization. Boosting has been successfully used for text categorization, in a system called BoosTexter (Schapire & Singer, 2000), so it would interesting to see if DECORATE can perform better.

In Chapter 5, we studied the impact of imperfections in data on different ensemble methods. Our results showed that ADABOOST is very sensitive to classification noise. Several variations of ADABOOST have been recently developed to address this issue (Servedio, 2003; Oza, 2004; McDonald et al., 2003). An interesting avenue for future work would be to compare the performance of DECORATE with these new boosting algorithms.

The empirical success of DECORATE in the classification task, raises the issue of the need for a sound theoretical understanding of its effectiveness. In particular, it would be useful to provide a theoretical guarantee that the DECORATE algorithm improves the bound on generalization error. Furthermore, it would be useful to study the connection between DECORATE and methods that attempt to maximize the margins on the training sample, such as AdaBoost (Schapire, Freund, Bartlett, & Lee, 1998).

Recent studies have analyzed how different ensemble methods affect the contribution of *bias* and *variance* to generalization error (Suen, Melville, & Mooney, 2005; Bauer & Kohavi, 1999; Webb, 2000). Performing a similar bias-variance analysis of DECORATE may provide some useful insights about the algorithm.

9.2 Active Learning for Probability Estimation

In our experiments on active probability estimation in Chapter 7, we are forced to use indirect metrics to measure CPE accuracy, since we do not have datasets that provide true class probabilities for instances. However, in the absence of real data with class probabilities, it would be useful to also evaluate our methods on synthetic data, as done by Margineantu and Dietterich (2002).

Our study uses standard metrics for evaluating CPE employed in existing research (Nielsen, 2004). However, we have shown that JS-divergence is a good measure for selecting examples for improving CPE; and therefore it should also be a good measure for evaluating CPE. In future work, when the true class probabilities are known, we suggest evaluating CPE by computing the JS-divergence between the estimated and the true class distributions.

9.3 Active Feature-value Acquisition

In our current AFA setting in Chapter 8, we assume that for all instances, the values of a subset of the features $a_1, \ldots, a_i$ are known, and the values of the remaining features $a_{i+1}, \ldots, a_n$ are unknown and can be acquired at a cost. We also assume that for a selected instance, the entire set of missing features-values can be acquired at once. Furthermore, we assume that the cost of acquiring complete information is the same for different instances. These assumptions were based on the web-usage datasets(Padmanabhan et al., 2001) that motivated our study. However, these assumption may not be very realistic for other domains. As such, in recent work (Melville, Saar-Tsechansky, Provost, & Mooney, 2005b, 2005a), we have studied a more general form of the AFA problem, where the learner may request the value of a specific feature for a selected instance. In this setting, we also assume that the cost of acquiring each feature-value may vary. We present an approach that acquires feature values for inducing a classification model based on an estimation of the expected improvement in model accuracy per unit cost. Experimental results demonstrate that our approach consistently reduces the cost of producing a model of a desired accuracy compared to random feature acquisitions.

Similarly to previous studies on active feature acquisition (Zheng & Padmanabhan, 2002) the test instances in this study are complete. We test on complete instances in order to estimate the model's performance without confounding effects of incomplete values in test instances. However, it is important to explore the setting in which feature values can also be acquired for incomplete test instances. Some work in this direction has recently been done by Kapoor and Greiner (2005).

Chapter 10

Conclusions

This thesis has introduced the DECORATE algorithm, which is a simple yet effective method that uses diversity to guide ensemble construction. By manipulating artificial training examples, DECORATE is able to use a strong base learner to produce an accurate and diverse set of classifiers. This thesis demonstrates that the diverse ensembles produced by DECORATE can be used to learner accurate classifiers in settings where there is a limited amount of training data, and in *active* settings, where the learner can acquire class labels for unlabeled examples or additional feature-values for examples with missing values.

We first examined the *passive* learning setting, where the training set is randomly sampled from the data distribution. Experimental results demonstrate that DECORATE produces highly accurate ensembles that outperform Bagging, ADABOOST and Random Forests low on the learning curve. Moreover, even on larger training sets, DECORATE outperforms Bagging and Random Forests, and is competitive with ADABOOST.

We ran additional experiments comparing the sensitivity of Bagging, ADABOOST, and DECORATE to three types of imperfect data: missing features, classification noise, and feature noise. Our experiments, using J48 as a base learner, show that in the case of missing features, DECORATE significantly outperforms the other approaches. In the case of classification noise, both DECORATE and Bagging are effective at decreasing the error of the base learner. However, ADABOOST degrades rapidly in performance, even with small amounts of classification noise, often performing worse than J48. In the presence of noise in the features, all ensemble methods produce consistent improvements over the base learner. These results suggest that, when there are many missing features in the data, or appreciable noise in the classification labels, it is advisable to use DECORATE or Bagging over ADABOOST.

For the task of active learning, we propose the algorithm ACTIVEDECORATE, which uses DECORATE ensembles to help select the most informative examples to be labeled. Empirical results show that this approach is very effective at reducing the number of labeled training examples required to achieve high classification accuracy. On average, ACTIVE-DECORATE requires only 78% of the number of training examples required by DECORATE using random sampling. Experimental results also demonstrate that, on average, ACTIVE-DECORATE performs better that the competing active learners — Query by Bagging and Query by Boosting.

Another contribution of this thesis, is proposing the use of Jensen-Shannon divergence for measuring the utility of acquiring labeled examples for active learning of probability estimates. Extensive experiments have demonstrated that JS-divergence effectively captures the uncertainty of class probability estimation and allows us to identify particularly informative examples that significantly improve the model's class distribution estimation. In particular, we show that when JS-divergence is used with ACTIVEDECORATE it produces substantial improvements over using margins, which focuses on classification accuracy. We also improve on BOOTSTRAP-LV, an existing active CPE learner for binary class problems, by using JS-divergence in place of its *local variance* measure. Apart from requiring fewer labeled examples to achieve accurate probability estimates, our methods have the advantage of being applicable to multi-class domains.

This thesis also presents a general framework for the task of active feature-value acquisition (AFA). Within this framework, we present an approach in which instances are selected for acquisition based on the current model's accuracy and its confidence in prediction. Experiments on this approach, *Error Sampling*, using DECORATE demonstrate that our method can induce accurate models using substantially fewer feature-value acquisitions as compared to a random acquisition policy. A direct comparison of *Error Sampling* with GODA, an alternate AFA approach, demonstrates that in spite of its simplicity, *Error Sampling* exhibits superior performance. *Error Sampling*'s utilization of all known feature-values and of a simple measure of the potential for improvement from an acquisition, makes it computationally more efficient and leads to more accurate classifiers than GODA.

This thesis introduces the DECORATE algorithm, which produces a diverse set of classifiers by manipulating artificial training examples. We demonstrate that the diverse ensembles produced by DECORATE can be used to learn accurate classifiers in settings where there is a limited amount of training data, and in *active* settings, where the learner can acquire class labels for unlabeled examples or additional feature-values for examples with missing values. As a result, we are able to build more accurate predictive models than existing methods, with reduced supervision, which translates to lower costs of data acquisition.

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*.

Batista, G., & Monard, M. C. (2003). An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, *17*, 519–533.

Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning*, *36*(1-2), 105–139.

Bishop, C. M. (1995). *Neural Networks for Pattern Recogntion*. Oxford University Press.

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

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

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

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)*, pp. 67–74, 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 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pp. 489–496, Philadelphia, PA.

Craven, M. W., & Shavlik, J. W. (1995). Extracting tree-structured representations of trained networks. In Touretzky, D. S., Mozer, M. C., & Hasselmo, M. E. (Eds.), *Advances in Neural Information Processing Systems*, Vol. 8, pp. 24–30. The MIT Press.

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.

Davidson, I. (2004). An ensemble technique for stable learners with performance bounds. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-2004)*, pp. 330–335.

Dhillon, I., Mallela, S., & Kumar, R. (2002). Enhanced word clustering for hierarchical classification. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002)*, Edmonton.

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

Dietterich, T. (2002). *The Handbook of Brain Theory and Neural Networks*, chap. Ensemble Learning, pp. 405–408. The MIT Press.

Dietterich, T. (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 (ICML-97)*, pp. 98–106, Nashville, TN. Morgan Kaufmann.

Duda, R. O., & Hart, P. E. (1973). *Pattern Classification and Scene Analysis*. Wiley, New York.

Efron, B., & Tibshirani, R. J. (1993). *An Introduction to the Bootstrap*. Chapman and Hall, New York, NY.

Federov, V. (1972). *Theory of optimal experiments*. Academic Press.

Freund, Y., Iyer, R., Schapire, R. E., & Singer, Y. (1998). An efficient boosting algorithm for combining preferences. In Shavlik, J. W. (Ed.), *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-98)*, 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)*, pp. 148–156. 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.

Greiner, R., Grove, A., & Roth, D. (2002). Learning cost-sensitive active classifiers. *Artificial Intelligence*, *139*(2), 137–174.

Hagel, J., & Singerare, M. (1999). *Net Worth: Shaping Markets When Customers Make the Rules*. Harvard Business School Press.

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.

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.

Kalai, A., & Servedio, R. A. (2003). Boosting in the presence of noise. In *Thirty-Fifth Annual ACM Symposium on Theory of Computing*.

Kapoor, A., & Greiner, R. (2005). Learning and classifying under hard budgets. In *Proceedings of the European Conference on Machine Learning (ECML-05)*, Porto, Portugal.

Krogh, A., & Vedelsby, J. (1995). Neural network ensembles, cross validation and active learning. In *Advances in Neural Information Processing Systems 7*, pp. 231–238.

Kuncheva, L., & Whitaker, C. (2003). Measures of diversity in classifier ensembles and their relationship with ensemble accuracy. *Machine Learning*, *51*(2), 181–207.

Leigh, M., & James, L. (2004). Comparison of imputation techniques: Accuracy of imputations, imputed data parameters, imputed model, parameters and quality of marketing decisions implied by the estimated models. Working paper, Department of Marketing, Red McCombs School of Business, University of Texas at Austin.

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.

Lin, J. (1991). Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, *37*(1), 145–151.

Little, R., & Rubin, D. (1987). *Statistical Analysis with Missing Data*. John Wiley and Sons,.

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

Lizotte, D., Madani, O., & Greiner, R. (2003). Budgeted learning of naive-Bayes classifiers. In *Proceedings of 19th Conference on Uncertainty in Artificial Intelligence (UAI-2003)*, Acapulco, Mexico.

Maclin, R., & Opitz, D. (1997). An empirical evaluation of bagging and boosting. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pp. 546–551, 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.

McDonald, R. A., Eckley, I. A., & Hand, D. J. (2002). A multi-class extension to the brownboost algorithm. *International Journal of Pattern Recognition and Artificial Intelligence: Special issue on classifier fusion*.

McDonald, R. A., Hand, D. J., & Eckley, I. A. (2003). An empirical comparison of three boosting algorithms on real data sets with artificial class noise. In *Fourth International Workshop on Multiple Classifier Systems*, pp. 35–44. Springer.

McKay, R., & Abbass, H. (2001). Analyzing anticorrelation in ensemble learning. In *Proceedings of 2001 Conference on Artificial Neural Networks and Expert Systems*, pp. 22–27, Otago, New Zealand.

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

Melville, P., & Mooney, R. J. (2004a). Creating diversity in ensembles using artificial data. *Journal of Information Fusion: Special Issue on Diversity in Multi Classifier Systems*, *6*(1), 99–111.

Melville, P., & Mooney, R. J. (2004b). Diverse ensembles for active learning. In *Proceedings of 21st International Conference on Machine Learning (ICML-2004)*, pp. 584–591, Banff, Canada.

Melville, P., Saar-Tsechansky, M., Provost, F., & Mooney, R. (2004). Active feature-value acquisition for classifier induction. In *Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM-04)*, pp. 483–486.

Melville, P., Saar-Tsechansky, M., Provost, F., & Mooney, R. (2005a). Economical active feature-value acquisition through expected utility estimation. In *Proceedings of the KDD-05 Workshop on Utility-Based Data Mining*, pp. 10–16, Chicago, IL.

Melville, P., Saar-Tsechansky, M., Provost, F., & Mooney, R. (2005b). An expected utility approach to active feature-value acquisition. In *Proceedings of the International Conference on Data Mining*, pp. 745–748, Houston, TX.

Melville, P., Shah, N., Mihalkova, L., & Mooney, R. J. (2004). Experiments on ensembles with missing and noisy data. In F. Roli, J. K., & Windeatt, T. (Eds.), *Lecture Notes in Computer Science: Proceedings of the Fifth International Workshop on Multi Classifier Systems (MCS-2004)*, Vol. 3077, pp. 293–302, Cagliari, Italy. Springer Verlag.

Melville, P., Yang, S. M., Saar-Tsechansky, M., & Mooney, R. (2005). Active learning for probability estimation using Jensen-Shannon divergence. In *Proceedings of the European Conference on Machine Learning (ECML-05)*, pp. 268–279, Porto, Portugal.

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.

New York Times (1999). Doubleclick to buy retailing database keeper. June 5.

Nielsen, R. D. (2004). MOB-ESP and other improvements in probability estimation. In *Proceedings of 20th Conference on Uncertainty in Artificial Intelligence (UAI-2004)*, pp. 418–425, Banff, Canada.

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.

Opitz, D., & Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, *11*, 169–198.

Oza, N. (2004). AveBoost2: Boosting for noisy data. In F. Roli, J. K., & Windeatt, T. (Eds.), *Lecture Notes in Computer Science: Proceedings of the Fifth International Workshop on Multi Classifier Systems (MCS-2004)*, Vol. 3077, pp. 31–40, Cagliari, Italy. Springer Verlag.

Padmanabhan, B., Zheng, Z., & Kimbrough, S. O. (2001). Personalization from incomplete data: what you don't know can hurt.. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2001)*, pp. 154–163.

Provost, F., & Domingos, P. (2003). Tree induction for probability-based rankings. *Machine Learning*, *52*(3), 199–215.

Quinlan, J. R. (1989). Unknown attribute values in induction. In *Proceedings of the Sixth International Workshop on Machine Learning*, pp. 164–168, Ithaca, NY.

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.

Raviv, Y., & Intrator, N. (1996). Bootstrapping with noise: An effective regularization technique. *Connection Science*, *8*(3-4), 356–372.

Rosen, B. (1996). Ensemble learning using decorrelated neural networks. *Connection Science*, *8*, 373–384.

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)*, pp. 441–448. Morgan Kaufmann, San Francisco, CA.

Saar-Tsechansky, M., & Provost, F. (2004). Active sampling for class probability estimation and ranking. *Machine Learning*, *54*, 153–178.

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)*.

Servedio, R. A. (2003). Smooth boosting and learning with malicious noise. *The Journal of Machine Learning Research*, *4*, 633–648.

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

Spatz, C., & Johnston, J. (1984). *Basic Statistics* (3 edition)., chap. 9, pp. 201–202. Brooks/Cole Publishing Company.

Suen, Y. L., Melville, P., & Mooney, R. J. (2005). Combining bias and variance reduction techniques for regression trees. In *Proceedings of the European Conference on Machine Learning (ECML-05)*, pp. 741–749, Porto, Portugal.

Tan, M., & Schlimmer, J. C. (1990). Two case studies in cost-sensitive concept acquisition. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pp. 854–860, Boston, MA.

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. D. (2000). Types of cost in inductive concept learning. In *Proceedings of the Workshop on Cost-Sensitive Learning at the 17th International Conference on Machine Learning*, Palo Alto, CA.

Webb, G. (2000). Multiboosting: A technique for combining boosting and wagging. *Machine Learning*, *40*(2), 159–196.

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

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.

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*, pp. 576–587.

Zheng, Z., & Padmanabhan, B. (2002). On active learning for data acquisition. In *Proceedings of IEEE International Conference on Data Mining*.

Zhu, X., Lafferty, J., & Ghahramani, Z. (2003). Combining active learning and semi-supervised learning using Gaussian fields and harmonic functions. In *Proc. of the ICML Workshop on the Continuum from Labeled to Unlabeled Data*, pp. 58–65.

Vita

Prem Melville was born in Bombay, India, on December 9th, 1977, much to his own surprise. He graduated from Cathedral and John Connon High School in 1995. He subsequently went to Brandeis University, where he majored in Computer Science and Mathematics, and graduated *summa cum laude* in 1999. Following this, he enrolled in the doctoral program at the University of Texas at Austin, where he happily hid for a few years. He is now at large, and may soon be sighted at the IBM T.J. Watson Research Center.

Permanent Address: Department of Computer Sciences
University of Texas at Austin
2.124 Taylor Hall
Austin, TX 78712
melville@cs.utexas.edu

This dissertation was typeset with LaTeX 2_ε[1] by the author.

[1] LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay, James A. Bednar, and Ayman El-Khashab.