Ensemble approaches to semi-supervised learning
Tanha, J.

Citation for published version (APA):
Tanha, J. (2013). Ensemble approaches to semi-supervised learning

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Chapter 3

Self-Training With Decision Trees

This chapter focuses on Semi-Supervised Learning with a decision tree learner. Standard decision tree learning cannot be used as the base learner in a self-training method for semi-supervised learning. We demonstrate this with experiments. Good probability estimations are needed in self-training for selecting unlabeled examples for labeling, as selection metric, but a basic decision tree learner does not produce reliable probability estimation. We consider the effect of several modifications to the basic decision tree learner which produce better probability estimations than using the distributions at the leaves of the tree. We show that these modifications do not produce better performance when used on the labeled data only, but they do benefit more from the unlabeled data in self-training. The modifications that we consider are Naive Bayes Tree, a combination of No-pruning and Laplace correction, grafting, and using a distance-based measure. Next we extend this improvement to algorithms for ensembles of decision trees. We show that when a modified decision tree learner is used as the base learner, the ensemble learner does benefit from unlabeled data in self-training.

This chapter is an extension of the published paper by Tanha et al. [92], at the IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS) in 2011.

3.1 Introduction

In this chapter we focus on self-training with a decision tree learner as the base learner. Self-training is a commonly used technique for semi-supervised learning. In self-training, a classifier is first trained on a small set of labeled data. This classifier is then used to classify the unlabeled data. Next, a set of unlabeled data points, of which class labels can be predicted with high confidence, is added to the training set. The classifier is re-trained and this procedure is repeated until it reaches a stopping condition. In self-training misclassified examples are used for further learning and thus can have a strong effect on the result. There is thus
a need to find a metric for the selection from the unlabeled data. In a decision 
tree classifier the class distribution at the leaves is normally used as probability 
estimation for the predictions. We show below that a standard decision tree 
learning algorithm like C4.5 performs poorly as the base learner in self-training. 
Accuracy after self-training is almost the same as after supervised learning from 
only the labeled data. The algorithm does not benefit from the unlabeled data. 
Provost and Domingos [73] showed that the distributions at the leaves of decision 
trees provide poor ranking for probability estimates. Using class frequencies at the 
leaves for probability estimation brings two problems when ranking and selecting 
instances: the first is that the sample size at the leaves is almost always small, 
which produces poor estimates, and the second is that all instances at a leaf 
are assigned the same probability. Examples that are far from the boundary 
are more likely to belong to a class than those close to the boundary. Because 
of poor probability estimation, selecting newly-labeled data in self-training is 
error-prone and these errors will propagate to later iterations. However, in many 
domains decision trees are the best choice as the base learner, see [73]. This has 
motivated us to look for improvements of the probability estimation for decision 
tree learning.

We consider several methods that may improve the probability estimation of 
tree classifiers: (a) No-pruning and applying the Laplacian Correction (C4.4) [73], 
(b) Grafting [103], (c) a combination of Grafting with Laplacian Correction and 
No-pruning, (d) a Naive Bayes Tree classifier (NBTree) [53], (e) using a distance-
based measure combined with the improved decision tree learners. Our hypothesis 
is that these modified decision tree learners will show accuracy similar to the 
standard decision tree learner when applied to the labeled data only, but will 
benefit from the unlabeled data when used as the base classifier in self-training. 
Next we extend our analysis from single decision tree as the base classifier to 
ensembles of decision trees, in particular the Random Subspace Method (RSM) 
[44] and Random Forests (RF) [15]. In this case, probability estimation can be 
estimated from the predictions of multiple trees. However, since these trees suffer 
from poor probability estimation, the ensemble learner will not benefit much from 
self-training on unlabeled data. For the same reason as for the single decision tree 
learner, we also expect that using the modified decision tree learners as the base 
learner for the ensemble will improve self-training with the ensemble classifier as 
the base learner. The results of the experiments on the several benchmark UCI 
datasets and visual features of web-pages datasets [10] confirm this.

The rest of this chapter is organized as follows. Section 3.2 reviews related 
work on semi-supervised learning. In Section 3.3, decision tree learning algorithms 
as the supervised learner in self-training are evaluated. In Section 3.4 and 3.5 
we address the improvements for self-training. The experimental setup of the 
experiments is presented in Section 3.6. Section 3.7 presents the results of the 
experiments and in Section 3.8, we present our conclusions.
3.2 Overview of Self-Training

There are several kinds of learning algorithms for semi-supervised learning. The generative approach is one of the best-known. In this method the labels of the unlabeled data are viewed as missing values of model parameters, and the EM (Expectation Maximization) algorithm [26] is used to find both the maximum likelihood estimation of the model parameters and estimates for the missing values. Generative methods differ from each other by the type of distribution used to fit the data, for example a mixture of Gaussians [86] and mixture of experts [68]. Nigam et al. [71] apply the EM algorithm to a mixture of multinomials for text classification. They showed that the resulting classifiers perform better than those trained only on originally labeled data. However, Nigam and Ghani [72] also found that there were examples in which EM did not work properly. One issue in EM when it is used in semi-supervised learning is that locally maximizing the likelihood may not lead to the optimal classification performance [78]. The EM approach is an example of the iterative approach or briefly iterative methods, as addressed in Subsection 2.2.2. Other methods based on iterative approach are methods, such as co-training [9], Co-forest [108], and Ensemble-Co-Training [93]. Self-training is also an iterative method.

Self-training has been applied to several natural language processing tasks. Yarowsky [105] uses self-training for word sense disambiguation. A self-training algorithm is used to recognize different nouns in [76]. Maeireizo et al. [64] propose a self-training algorithm to classify dialogues as “emotional” or “non-emotional” with a procedure involving two classifiers. In [100] a semi-supervised self-training approach using a hybrid of Naive Bayes and decision trees is used to classify sentences as subjective or objective.

Rosenberg et al. [78] proposed a self-training approach to object detection. They implemented a wrapper around the training process of an existing object detector and presented empirical results. The main contribution of this empirical study is to show that a model trained on a small set of labeled instances can achieve results comparable to a model trained in the supervised manner using a larger set of labeled data. The Nearest Neighbor classifier was used as the base classifier in self-training.

Li et al. [61] propose a self-training semi-supervised support vector machine algorithm and its selection metric, which are designed to train a classifier from a limited number of training data. Two examples show the validity of the proposed algorithm and selection metric. The authors apply the proposed algorithm to a data set collected from a P300-based brain computer interface speller. This algorithm is shown to significantly reduce training effort.

In general, self-training is a wrapper algorithm, and is hard to analyze. However, for specific base classifiers, there have been some studies, such as [41]. Haffari and Sarkar [41] showed that the Yarowsky’s algorithm [105] minimizes an upper-bound on a new definition of cross entropy based on a specific instantiation of the
Bregman distance. In this chapter, we analyze the effect of modifying a decision tree learner to improve the probability estimation of its predictions, when the decision tree is used as the base learner in self-training.

3.3 Semi-Supervised Self-Training with Decision Trees

In this section we first provide a basic definition of semi-supervised learning and then address the self-training algorithm using decision tree classifiers.

3.3.1 Basic Definitions

In semi-supervised learning there is a small amount of labeled data and a large pool of unlabeled data. Data points can be divided into the points $X_l = (x_1, x_2, ..., x_l)$, for which labels $Y_l = \{+1, -1\}$ are provided, and the points $X_u = (x_{l+1}, x_{l+2}, ..., x_{l+u})$, the labels of which are not known. We assume that labeled and unlabeled data are drawn independently from the same data distribution. In this chapter we consider datasets for which $l \ll u$, where $l$ and $u$ are the number of labeled data and unlabeled data respectively.

3.3.2 The Self-Training Algorithm

The self-training algorithm uses its own predictions to obtain new training data. A base learner is first trained with a small number of labeled examples, the initial training set. The classifier is then used to predict the labels for the unlabeled examples (prediction step) based on the classification confidence. Next, a subset $S$ of the unlabeled examples, together with their predicted labels, is selected to train a new classifier (selection step). Typically, $S$ consists of a few unlabeled examples with high-confidence predictions. The classifier is then re-trained on the new set of labeled examples, and the procedure is repeated (re-training step) until it reaches a stopping condition. Here we employ the decision tree classifier as the base learner in self-training. The most well-known algorithm for building decision trees is C4.5 [74], an extension of Quinlan’s earlier ID3 algorithm. Decision trees are one of the widely used classification methods in many domains, for example [16], [7], and [36]. They are fast and effective in many domains. They work well with little or no tweaking of parameters which has made them a popular tool for many domains [73]. This makes it useful to find a semi-supervised method for learning decision trees. Algorithm 2 presents the main structure of the self-training algorithm.

The goal of the selection function in Algorithm 2 is to find a subset of high-confidence predictions. For that, the selection function selects the newly-labeled examples of which the probability estimations are above the threshold $T$, at
Algorithm 2 Outline of the Self-Training algorithm

Initialize: L, U, F, T ; L: Labeled data; U: Unlabeled data;
F : underlying classifier; T : threshold for selection;
Max-Iterations : number of iterations; \( \{P_l\}_{l=1}^{M} \): Prior probability;
t ← 1;
while (U ! = empty) and (t < Max - Iterations ) do
  - Train Classifier F on L;
  for each \( x_i \in U \) do
    - Assign pseudo-label to \( x_i \) based on classification confidence
    - Sort Newly-Labeled examples based on the confidence
    - Select a set \( S \) of the high-confidence predictions according to \( n_l \propto P_l \) and threshold \( T \) // Selection Step
    - Update U = U - S; L = L U S;
    - Re-Train F by the new training set
  end while
Output: Generate final hypothesis based on the new training set

each iteration of the training process. This is important, because a misclassified prediction will propagate to produce further classification errors. At each iteration the newly-labeled instances are added to the original labeled data for constructing a new classifier based on the new training set. The number of iterations in Algorithm 2 depends on the threshold \( T \) and also on pre-defined maximal number of iterations \( Max - Iterations \).

3.4 Self-Training by Improving Probability Estimates

The main difficulty in self-training is to find a set of high-confidence predictions, especially when the base learner is a decision tree learner. Although decision tree classifiers are effective in many applications, they provide poor probability estimation. Therefore, good probability estimates are needed to find a set of high-confidence predictions in self-training. The distribution at the leaf of a decision tree gives the probability that the instance belongs to the majority class but these probabilities are based on very few data points, due to the fragmentation of data over the decision tree. For example, if a leaf node has subset of 50 examples of which 45 examples belong to one class, then any example that corresponds to this leaf will get 0.9% probability where a leaf with 3 examples of one class get a probability of 1.0. In semi-supervised learning this problem is very serious because the size of initial training set is small from the beginning. Provost and Domingos [73] showed that the probability estimates obtained from the distributions in the
Chapter 3. Self-Training With Decision Trees

leaves of decision trees are not very accurate. It seems that pruning methods tend to prune too much so that the resulting decision tree tends to “underfit” the data. Provost and Domingos propose several modifications of the C4.5 decision tree learner for finding better probability estimations. They propose decision tree learner C4.4 which does not prune and uses the Laplace correction to smooth probabilities at leaves. Experiments show that this improves the probability estimation of the decision tree learner.

Here, we consider several methods for improving the probability estimates at the leaves of decision trees. Beside C4.4, these are Naive Bayes Tree, Grafted Decision Tree [103], and global distance-based measure.

3.4.1 C4.4: No-Pruning and Laplacian Correction

One candidate improvement is the Laplacian correction (or Laplace estimator) which smooths the probability values at the leaves of the decision tree [73]. In fact smoothing of probability estimates from small samples is a well-studied statistical problem [89]. Assume there are K instances of a class out of N instances at a leaf, and C classes. The Laplacian correction calculates the estimated probability $P(\text{class})$ as $(K+1)/(N+C)$. Therefore, while the frequency estimate yields a probability of 1.0 from $K=10$, $N=10$ leaf, for a binary classification problem the Laplace estimate produces a probability of $(10+1)/(10+2)=0.92$. For sparse data, the Laplacian correction at the leaves of a tree yields a more reliable estimation that is crucial for the selection step in self-training. Without it, regions with low density will show relatively extreme probabilities that are based on very few data points. These have a high probability of being used for self-training, which is problematic because of misclassified examples with high probability of being correct.

Another possible improvement is a decision tree learner that does not do any pruning. Although this introduces the risk of “overfitting”, it may be a useful method because of the small amount of training data. If there are few training data then pruning methods can easily produce underfitting and No-pruning avoids this. In applications of semi-supervised learning, “underfitting” is a potential problem. Although it produces even fewer data at leave nodes, No-pruning may therefore still provide better estimates, especially if combined with the Laplace correction. We therefore include this combination in our analysis as C4.4.

3.4.2 NBTree

The Naive Bayesian Tree learner, NBTree [53], combines the Naive Bayes Classifier with decision tree learning. In an NBTree, a local Naive Bayes Classifier is constructed at each leaf of decision tree that is built by a standard decision tree learning algorithm like C4.5. NBTree achieves in some domains higher accuracy
than either a Naive Bayes Classifier or a decision tree learner. NBTree uses additional attributes and gives a posterior probability distribution that can be used to estimate the confidence of the classifier. Similar to the standard decision tree, a threshold for numerical attributes is chosen using entropy minimization strategy.

For numerical attributes, NBTree fits a Gaussian distribution to estimate the conditional probabilities. Unlike the standard decision tree learner, this uses the distance of values to the decision boundary to estimate the posterior probabilities. This will improve the probability estimates.

### 3.4.3 Grafted Decision Tree

A grafted decision tree classifier generates a "grafted" decision tree from a standard decision tree. The idea behind grafting is that some regions in the data space are more sparsely populated. The class label for sparse regions can better be estimated from a larger region. The grafting technique [103] searches for regions of the multidimensional space of attributes that are labeled by the decision tree but they contain no or very sparse training data. These regions are then split by the region that corresponds to a leaf and labeling the empty or sparse areas by the label of the majority above the previous leaf node. Consider the example in Figure 3.1. It shows the resulting grafted tree. As can be seen, there are two cuts in the decision trees at nodes 12 and 4. After grafting, branches are added by the grafting technique. Grafting performs a kind of local “unpruning” for low-density areas. This can improve the resulting model, see [103]. In fact, grafted decision tree learning gives better decision tree in case of sparse data and also improves the probability estimates.

### 3.4.4 Combining No-pruning, Laplace correction and Grafting

We combine Grafting with the Laplacian correction and No-pruning, which is called C4.4graft. We expect that C4.4graft gives better decision tree than C4.5 in the case of sparse data and it also improves probability estimates due to using the Laplacian correction and No-pruning, see Section 3.7.1.

### 3.4.5 Global Distance-Based Measure

Finally we use a confidence measure based on the distance between an example and labeled examples from both classes. Specifically we use the difference in average Mahalanobis distance between an unlabeled example and all labeled examples. As addressed in Algorithm 2, the probability estimation is used to select high-confidence predictions, which may not always be optimal because of some misclassified examples with high-confidence probability estimation. Another way to select from the unlabeled examples is to use a distance-based approach. In
Chapter 3. Self-Training With Decision Trees

this method, for each unlabeled example the distance between the corresponding example and all other positive examples is computed based on the Mahalanobis distance method which differs from Euclidean distance in that it takes into account the correlation of the data. It is defined as follows:

\[ D(X) = \sqrt{(X - \bar{X})^T S^{-1}(X - \bar{X})} \]  

where \( X = (X_1, ..., X_n) \) is a multivariate vector, \( \bar{X} = (\bar{X}_1, ..., \bar{X}_n) \) is the mean, and \( S \) is the covariance matrix. For each unlabeled example, the distance is calculated from all positive and negative examples separately. Then the absolute value of the difference between \( p_i \) and \( q_i \) is calculated as a score for each unlabeled example. Next, these scores are used for selection metric. Algorithm 3 shows the procedure for selection metric using Mahalanobis distance. As shown, for each example a score is assigned. Then, a set of examples with highest score is selected from unlabeled examples. We use this selection metric along with the classifier selection metric, probability estimation, to select from the unlabeled examples.

In Algorithm 3, a set of unlabeled examples is first selected. This subset is then used in the self-training process to assign “pseudo-labels”. Next, the high-confidence predictions from this subset is added with their “pseudo-labels” to the training set. This procedure is repeated until it reaches a stopping condition. Algorithm 4 shows the self-training procedure.
3.5 Self-Training for Ensembles of Decision Trees

In this section we extend the analysis from decision trees to ensembles of decision trees. An ensemble combines many, possibly weak, classifiers to produce a (hopefully) strong classifier [29]. Ensemble methods differ in their base learner and in how classifiers are combined. Examples of the ensemble methods are bagging [13], boosting [33], Random Forest (RF) [15], and Random Subspace Method (RSM) [44]. For an ensemble classifier, the probability estimation can be estimated by combining the confidences of their components. This tends to improve both the classification accuracy and the probability estimation [63]. However, if a standard decision tree learner is used as the base learner then the problems that we noted above carry over to the ensemble. We therefore expect that improving the probability estimates of the base learner will enable the ensemble learner to benefit more from the unlabeled data than if the standard decision tree learner is used. Algorithm 5 shows the generation of the trees in an ensemble classifier. In the experiments we use Random Forest and the Random Subspace Method to generate ensembles of trees.

The RSM and RF ensemble classifiers are well-suited for data that are high-dimensional. A single decision tree minimizes the number of features that are used in the decision tree and does not exploit features that are correlated and
Chapter 3. Self-Training With Decision Trees

Algorithm 4 Self-Training with Distance-based Selection Metric

Initialize: L, U, F, T, P; L: Labeled data; U: Unlabeled data;
F: Underlying classifier; T: threshold for selection;
P: Number of selected unlabeled examples at each iteration;
Max-Iterations: Number of iterations; \( \{P_i\}_{i=1}^{M} \): Prior probability;
t ← 1;
while (U ! = empty) and (t < Max – Iterations) do
    - \( H^{t-1} \leftarrow \text{BaseClassifier}(L, F); \)
    - \( U' \leftarrow \text{SelectionMetric}(U, P); \)
    for each \( x_i \in U' \) do
        - Assign pseudo-label to \( x_i \) based on \( H^{t-1} \) prediction
        - Measure confidence for the assigned pseudo-label to \( x_i \)
    end for
    - Select a set \( S \) of high-confidence predictions according to distribution of classes and threshold \( T \); // Selection Step
    - Update U = U - S; L = L U S;
    - Re-Train \( H^t \leftarrow \text{BaseClassifier}(L, F); \)
    - t ← t + 1
end while
Output: Classifier \( H^T \) is the final hypothesis

Algorithm 5 Ensemble of Decision Trees

Initialize: L, F, N; L: Labeled data;
F: Base classifier; N: Number of trees;
for i=1 to N do
    - \( L_i \leftarrow \text{BootstrapSample}(L); \) // or RandomSubSpaceSample(L);
    - \( h_i \leftarrow F(L_i) \)
    - \( H \leftarrow H + h_i \)
end for
Output: Generate ensemble \( H \)

all have little predictive power. The ensemble classifiers do not suffer from this problem because the random component and the ensemble allow including more features.

3.5.1 Random Forest

A Random Forest is an ensemble of decision trees in which each tree is first constructed independently of other trees in the forest. These \( n \) decision trees are trained on different subsets of the training set, generated from original labeled data by bagging [13], as addressed in Section 2.1.1. In bagging each training set is sampled from original data with replacement. For each tree those examples that
are not included in the training set are called Out-Of-Bag data for that tree and these are used for estimating the error, called Out Of Bag Error. Therefore, there is no need for cross-validation or a separate test set to get an unbiased test set error in this method. Random Forest uses randomized feature selection while the tree is growing. In the case of multidimensional datasets this property is indeed crucial, because when there are hundreds or thousand features, for example in medical diagnosis and documents, many weakly relevant features may not appear in a single decision tree at all. The final hypothesis in Random Forest is produced by using majority voting method among the trees. Many semi-supervised algorithms employ the Random Forest approach, such as Co-Forest [108].

3.5.2 The Random Subspaces Method

A Random Subspace method [44] is an ensemble method that combines randomly chosen feature subspaces of the original feature space. In the Random Subspaces method instead of using a subsample of data points, subsampling is performed on the feature space. The Random Subspaces method constructs a decision forest that maintains highest accuracy on training data and improves on generalization accuracy as it grows in complexity.

Assume that the i-th tree of the ensemble be defined as \( h_i(X, S_i) : X \mapsto Y \), where \( X \) are the data points, \( Y \) are the labels, and the \( S_i \) are independent identically distributed (i.i.d) random vectors. Let’s denote the ensemble \( H \) as \( H = \{h_1, h_2, ..., h_N\} \), where \( N \) is the number of trees in the forest. Then, the probability estimation for predicting example \( x \) is defined as follows:

\[
\arg\max_k P(k|x)
\]

(3.2)

where

\[
P(k|x) = \frac{1}{N} \sum_{i=1}^{N} P_i(k|x)
\]

(3.3)

and \( k \) is the label and \( P_i(k|x) \) is the probability estimation of the i-th tree for sample \( x \). As seen, this method produces a good ranking of probability estimation using voting methods, if each component classifier produces a reliable probability estimation.

Boosting also is one of the promising ensemble methods which sequentially constructs a weighted combination of the classifiers. However, it has two main drawbacks [29]: it performs poorly in the case of a small training set and noisy labeled data. For this, we focus on RF and RSM as the ensemble classifiers in this chapter.
3.5.3 The Ensemble Self-Training algorithm

Since our goal is to improve the probability estimates of decision trees, we employ modified version of C4.5, as mentioned earlier, to generate the trees in an ensemble classifier. Algorithm 6 shows the ensemble self-training algorithm. As can be seen, first the decision trees are generated by bagging or the random subspace method. Next the ensemble classifier assigns “pseudo-labels” and confidence to the unlabeled examples at each iteration. Labeling is performed by using different voting strategies, such as majority voting or average probability. Then a set of high-confidence predictions based on the pre-defined threshold $T$ is selected for the next iterations. A new training set is constructed by combination of selected newly-labeled and original labeled examples. Next, a new ensemble of trees based on the new training set is generated. The training process is repeated until it reaches a stopping condition.

**Algorithm 6 Ensemble Self-Training**

```
Initialize: L, U, F, T, N ; L: Labeled data; U: Unlabeled data; 
F : Underlying classifier; T : threshold for selection; N: Number of trees; 
Max-Iterations : Number of iterations; \{P_j\}_{j=1}^M : Prior probability; 
t ← 1;
while (U ! = empty) and (t < Max – Iterations ) do
    - $H^{t-1}$ ← EnsembleTrees($L,F,N$);
    for each $x_i \in U$ do
        - Assign pseudo-label to $x_i$ based on $H^{t-1}$ prediction
        - Measure confidence for the assigned pseudo-label to $x_i$
    end for
    - Select a set $S$ of high-confidence predictions according to distribution of classes and threshold $T$ // Selection Step
    - Update U = U - S; L = L U S;
    - Re-Train $H^t$ ← EnsembleTrees($L,F,N$);
    - $t$ ← $t + 1$
end while
Output: Ensemble classifier $H^T$
```

The selection metric in this algorithm is based on the ensemble classifier, which on one hand improves the classification accuracy and on the other hand improves the probability estimation of the trees. As a result, the probability estimation as the selection metric in this algorithm may lead to select a high-confidence predictions and performs better than a single classifier.
3.6 Experiments

In the experiments we compare the performance of the supervised decision tree learning algorithms to self-training. We expect to see that the decision tree learners that give better probability estimates for their predictions will benefit more from unlabeled examples. In other words, the difference in performance between supervised learning from the labeled data and semi-supervised learning on all data should be larger when the probability estimates of the predictions are more accurate. We then make the same comparison for ensemble learners.

3.6.1 Datasets

We use a number of UCI datasets [32] and web-pages classification task [10] to evaluate the performance of our proposed methods. Recently the UCI datasets have been extensively used for evaluating semi-supervised learning methods. Consequently, we adopt UCI datasets to assess the performance of the proposed algorithm in comparison to supervised learning.

UCI datasets

Fourteen datasets from the UCI repository are used in our experiments. We selected these datasets, because they differ from the number of features and examples, and distribution of classes. Information about these datasets is in Table 3.1. All sets have two classes and Perc. represents the percentage of the largest class.

Table 3.1: Overview of UCI Datasets

<table>
<thead>
<tr>
<th>Dataset (classes)</th>
<th>Attributes</th>
<th>Size</th>
<th>Perc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breath-cancer (1,2)</td>
<td>10</td>
<td>286</td>
<td>70</td>
</tr>
<tr>
<td>Bupa (1,2)</td>
<td>6</td>
<td>345</td>
<td>58</td>
</tr>
<tr>
<td>Car (1,2)</td>
<td>7</td>
<td>1594</td>
<td>76</td>
</tr>
<tr>
<td>Cmc (1,3)</td>
<td>10</td>
<td>1140</td>
<td>55</td>
</tr>
<tr>
<td>Colic (1,2)</td>
<td>22</td>
<td>368</td>
<td>63</td>
</tr>
<tr>
<td>Diabetes (1,2)</td>
<td>6</td>
<td>768</td>
<td>65</td>
</tr>
<tr>
<td>Heart Statlog (1,2)</td>
<td>13</td>
<td>270</td>
<td>55</td>
</tr>
<tr>
<td>Hepatitis (1,2)</td>
<td>19</td>
<td>155</td>
<td>79</td>
</tr>
<tr>
<td>Ionosphere (1,2)</td>
<td>34</td>
<td>351</td>
<td>36</td>
</tr>
<tr>
<td>Liver (1,2)</td>
<td>7</td>
<td>345</td>
<td>58</td>
</tr>
<tr>
<td>Sonar (1,2)</td>
<td>61</td>
<td>208</td>
<td>53</td>
</tr>
<tr>
<td>Tic-tac-toe (1,2)</td>
<td>9</td>
<td>958</td>
<td>65</td>
</tr>
<tr>
<td>Vote (1,2)</td>
<td>16</td>
<td>435</td>
<td>61</td>
</tr>
<tr>
<td>Wave (1,2)</td>
<td>41</td>
<td>3345</td>
<td>51</td>
</tr>
</tbody>
</table>
Web-Pages Datasets

To automatically classify and process web-pages in [10], the visual appearance of the web-pages are considered as one of the main features. The current systems for web-pages classification usually use the textual content of those pages, including both the displayed content and the underlying (HTML) code. However, the visual appearance of the web-pages can be a very important feature. Using generic visual features for classifying the web-pages for several different types of tasks is main task in [10]. The simple color and edge histograms, Gabor and texture features are used for classification task. These were extracted using an off-the-shelf visual feature extraction package (Lire, written in Java, [62]). The learning task is to find a classifier that recognizes the aesthetic value and the recency of a page. In this task the number of attributes is much larger than in the used UCI datasets in this study, which is more suitable for ensemble methods.

![Figure 3.3: Examples of web-pages](image)

Information about these datasets is in Table 3.2. All datasets have two classes and Prc. represents the percentage of the largest class label.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Attributes</th>
<th>Size</th>
<th>Perc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aesthetic</td>
<td>192</td>
<td>60</td>
<td>50</td>
</tr>
<tr>
<td>Recency</td>
<td>192</td>
<td>60</td>
<td>50</td>
</tr>
</tbody>
</table>

For each web-page a number of low-level features are computed, such as colors and edge histograms, Tamura, and Gabor features [10]. In this experiment we use binary classes Ugly and Beautiful as labels for Aesthetic dataset. These labels were assigned by human evaluators with a very high inter-rater agreement.
3.6. Experiments

Figure 3.3 shows samples of Ugly and Beautiful web-pages. For the Recency dataset, we look at Old and New designed web-pages.

3.6.2 Setup of the Experiments

For each dataset, 30 percent of the data are kept as test set, and the rest is used as training data. Training data in each experiment are first partitioned into 90 percent unlabeled data and 10 percent labeled data, keeping the class proportions in all sets similar to the original data set. We run each experiment 10 times with different subsets of training and testing data. The results reported refer to the test set. To provide a statistical basis for the main comparisons we use the following statistical tests.

3.6.3 Statistical Test

We compare the performance of the proposed algorithms by the given method in [27]. We first apply Friedman’s test as a nonparametric test equivalent to the repeated measures ANOVA. Under the null hypothesis Friedman’s test states that all the algorithms are alike and the rejection of this hypothesis implies the differences among the performance of the algorithms. It ranks the algorithms based on their performance for each dataset separately, then it assigns the rank 1 to the best performing algorithm, rank 2 to the second best, and so on. In case of ties average ranks are assigned to the related algorithms.

Let’s define \( r_{ij} \) as the rank of the \( j \)-th algorithm on the \( i \)-th datasets. The Friedman test compares the average ranks of algorithms, \( R_j = \frac{1}{N} \sum_i r_{ij} \), where \( N \) is the number of datasets and \( k \) is the number of the classifiers. Friedman’s statistic

\[
\chi_F^2 = \frac{12N}{k(k+1)} \left[ \sum_j R_j^2 - \frac{k(k+1)^2}{4} \right]
\]  

(3.4)

is distributed according to \( \chi_F^2 \) with \( k - 1 \) degrees of freedom. Iman and Davenport [46] showed that Friedman’s \( \chi_F^2 \) is conservative. Then they present a better statistic based on Friedman’s test:

\[
F_F = \frac{(N - 1)\chi_F^2}{N(k - 1) - \chi_F^2}
\]

(3.5)

This statistics is distributed according to the \( F - distribution \) with \( k - 1 \) and \( (k - 1)(N - 1) \) degrees of freedom.

As mentioned earlier, Friedman’s test shows whether there is a significant difference between the averages or not. The next step for comparing the performance of the algorithms with each other is to use a post-hoc test. We use Holm’s method [45] for post-hoc tests. This sequentially checks the hypothesis ordered
Chapter 3. Self-Training With Decision Trees

by their performance. The ordered \( p \)-values are denoted by \( p_1 \leq p_2 \leq \ldots \leq p_{k-1} \). Then each \( p_i \) is compared with \( \frac{\alpha}{(k-i)} \), starting from the most significant \( p \)-value. If \( p_1 \) is less than \( \frac{\alpha}{(k-1)} \), then the corresponding hypothesis is rejected and \( p_2 \) is compared with \( \frac{\alpha}{(k-2)} \). As soon as a certain hypothesis cannot be rejected, all remaining averages are taken as not significantly different. The test statistic for comparing two algorithms is

\[
    z = \frac{R_i - R_j}{\sqrt{\frac{k(k+1)}{6N}}}
\]  

The value \( z \) is used to find the corresponding probability from the normal distribution table, which is compared with the corresponding value of \( \alpha \).

3.6.4 Decision Tree Learners

We use several decision tree classifiers as the base classifier in self-training: J48 (the Java implementation of C4.5), C4.4, NBTree, C4.4graft, J48graft, and ensemble of trees. For our experiments we use the WEKA \([42]\) implementation of the classifiers in Java.

3.7 Results

Tables 3.3, 3.4, 3.7, and 3.8 give the classification accuracies for the experiments. For each base classifier, the performance of the supervised learning only on labeled data and self-training on both labeled and unlabeled data are reported.

3.7.1 Self-Training with a Single Classifier

At first, we compare the classification accuracy achieved by applying a decision tree learner to a set of labeled data only and self-training on these labeled data and a large set of unlabeled data. Table 3.3 compares the results of the standard decision tree learner J48 (DT) to its self-training version (ST-DT) and the same for the C4.4, grafting (GT), the combination of grafting and C4.4 (C4G), and the Naive Bayes Decision trees (NBTree). We expect that the modified algorithms show results similar to J48 when only labeled data are used but improved classification accuracies when they are used as the base learner in self-training.

Self-training with J48 decision tree learner

In Table 3.3, the columns DT and ST-DT show the classification accuracy of J48 base learner and self-training respectively. As can be seen, self-training does not benefit from unlabeled data and there is no difference in accuracy between
Table 3.3: Average classification accuracy of supervised learning and self-training with different base classifiers

<table>
<thead>
<tr>
<th>Dataset</th>
<th>DT</th>
<th>ST-DT</th>
<th>C4.4</th>
<th>ST-C4.4</th>
<th>GT</th>
<th>ST-GT</th>
<th>C4G</th>
<th>ST-C4G</th>
<th>NB</th>
<th>ST-NB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breath-cancer</td>
<td>68.00</td>
<td>69.00</td>
<td>66.25</td>
<td>67.00</td>
<td>68.00</td>
<td>70.01</td>
<td>66.25</td>
<td>70.12</td>
<td>72.50</td>
<td>75.75</td>
</tr>
<tr>
<td>Bupa</td>
<td>58.62</td>
<td>57.09</td>
<td>58.62</td>
<td>58.68</td>
<td>58.62</td>
<td>59.25</td>
<td>58.62</td>
<td>61.40</td>
<td>58.20</td>
<td>58.20</td>
</tr>
<tr>
<td>Car</td>
<td>86.08</td>
<td>86.04</td>
<td>85.48</td>
<td>87.48</td>
<td>86.08</td>
<td>87.28</td>
<td>85.48</td>
<td>88.28</td>
<td>85.08</td>
<td>87.68</td>
</tr>
<tr>
<td>Cmc</td>
<td>57.00</td>
<td>58.25</td>
<td>56.75</td>
<td>59.05</td>
<td>57.00</td>
<td>59.13</td>
<td>56.75</td>
<td>60.12</td>
<td>54.25</td>
<td>58.00</td>
</tr>
<tr>
<td>Colic</td>
<td>72.83</td>
<td>72.36</td>
<td>70.56</td>
<td>73.70</td>
<td>72.84</td>
<td>74.80</td>
<td>70.56</td>
<td>75.03</td>
<td>74.60</td>
<td>76.71</td>
</tr>
<tr>
<td>Diabetes</td>
<td>67.82</td>
<td>67.83</td>
<td>67.51</td>
<td>69.18</td>
<td>68.46</td>
<td>69.40</td>
<td>68.14</td>
<td>71.79</td>
<td>71.14</td>
<td>72.59</td>
</tr>
<tr>
<td>Heart</td>
<td>67.27</td>
<td>67.27</td>
<td>68.63</td>
<td>70.50</td>
<td>67.27</td>
<td>69.10</td>
<td>68.63</td>
<td>72.12</td>
<td>71.81</td>
<td>73.85</td>
</tr>
<tr>
<td>Hepatitis</td>
<td>76.00</td>
<td>75.60</td>
<td>76.00</td>
<td>76.40</td>
<td>76.00</td>
<td>76.60</td>
<td>76.00</td>
<td>80.60</td>
<td>78.40</td>
<td>82.40</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>70.47</td>
<td>70.67</td>
<td>70.47</td>
<td>71.46</td>
<td>70.37</td>
<td>71.56</td>
<td>70.37</td>
<td>73.72</td>
<td>79.97</td>
<td>82.57</td>
</tr>
<tr>
<td>Liver</td>
<td>56.80</td>
<td>56.60</td>
<td>57.00</td>
<td>60.80</td>
<td>57.00</td>
<td>59.80</td>
<td>57.00</td>
<td>59.98</td>
<td>57.00</td>
<td>59.90</td>
</tr>
<tr>
<td>Sonar</td>
<td>63.40</td>
<td>63.40</td>
<td>63.40</td>
<td>63.76</td>
<td>63.40</td>
<td>64.92</td>
<td>63.40</td>
<td>65.40</td>
<td>59.60</td>
<td>63.60</td>
</tr>
<tr>
<td>Tic-tac-toe</td>
<td>66.40</td>
<td>68.20</td>
<td>63.40</td>
<td>68.80</td>
<td>66.40</td>
<td>70.10</td>
<td>63.80</td>
<td>69.60</td>
<td>65.20</td>
<td>68.60</td>
</tr>
<tr>
<td>Vote</td>
<td>89.08</td>
<td>89.08</td>
<td>89.05</td>
<td>90.30</td>
<td>89.08</td>
<td>89.80</td>
<td>89.05</td>
<td>90.48</td>
<td>90.00</td>
<td>92.74</td>
</tr>
<tr>
<td>Wave</td>
<td>83.10</td>
<td>83.63</td>
<td>82.85</td>
<td>85.13</td>
<td>84.10</td>
<td>85.25</td>
<td>83.60</td>
<td>86.25</td>
<td>84.75</td>
<td>88.00</td>
</tr>
</tbody>
</table>

learning from the labeled data only and self-training from labeled and unlabeled data. The average improvement over all datasets is 0.15 %.

Self-Training with C4.4, Grafting, and NBTree

Table 3.3 gives the classification accuracy of C4.4 and ST-C4.4. Using C4.4 as the base learner in self-training algorithm does not result in higher classification accuracy if only the labeled data are used, but unlike the basic decision tree learner, C4.4 enables self-training to become effective for nearly all the datasets. The average improvement over the used datasets is 1.9%. The reason for improvement is that using Laplacian correction and No-pruning give better rank for probability estimation of the decision tree, which leads to select a set of high-confidence predictions.

In Table 3.3, we can see the same observation for J48graft (grafted decision tree). When using only labeled data we see no difference with the standard algorithm but self-training improves the performance of the supervised J48graft on all datasets. The average improvement over all datasets is 1.6%. Next, we combine Laplacian correction and No-pruning in J48graft, C4.4graft. This modification in grafted decision tree gives better results when it is used as the base learner in self-training. The results show that self-training algorithm outperforms the supervised C4.4graft classifier on all datasets. A T-test on the results shows that self-training significantly improves the classification performance of C4.4graft classifier on 10 out of 14 datasets and the average improvement over all datasets is 3.5%. Finally, the results of experiments on NBTree classifier as the base learner in self-training show that it improves the performance of NBTree classifier on 13 out of 14 datasets and the average improvement is 2.7%.
Self-Training with Single Classifier and Distance-based Measure

To evaluate the impact of the selection metric for self-training (as in Algorithms 4 and 3), we run another set of experiments. Table 3.4 shows the results of the experiments on UCI dataset using Algorithm 4. The columns ST-DT, ST-C4.4, ST-GT, ST-C4G, and ST-NB show the classification performance of self-training with J48, C4.4, J48graft, C4.4graft, and NBTree as the base learners respectively.

The results show that in general using both the Mahalanobis distance and probability estimation as the selection metric improves the classification performance of all self-training algorithms and emphasizes the effectiveness of this selection metric. For example, comparing the results of self-training in Tables 3.3 and 3.4, when the base classifier is C4.4graft, we observe that self-training in Table 3.4, using distance-based selection metric, outperforms the self-training in Table 3.3 on 10 out of 14 datasets. The same results can be seen for J48 as the base learner.

Table 3.4: Average performance of supervised learning and self-training using the Mahalanobis distance along with the probability estimation as the selection metric

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Supervised</th>
<th>DT</th>
<th>ST-DT</th>
<th>ST-C4.4</th>
<th>ST-GT</th>
<th>ST-C4G</th>
<th>ST-NB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breath-cancer</td>
<td>68.00</td>
<td>70.07</td>
<td>70.12</td>
<td>70.25</td>
<td>71.27</td>
<td>76.14</td>
<td></td>
</tr>
<tr>
<td>Bupa</td>
<td>58.62</td>
<td>61.60</td>
<td>61.60</td>
<td>61.80</td>
<td>62.37</td>
<td>61.76</td>
<td></td>
</tr>
<tr>
<td>Car</td>
<td>86.08</td>
<td>87.04</td>
<td>87.56</td>
<td>87.28</td>
<td>89.01</td>
<td>88.04</td>
<td></td>
</tr>
<tr>
<td>Cmc</td>
<td>57.00</td>
<td>59.00</td>
<td>60.01</td>
<td>60.77</td>
<td>61.01</td>
<td>60.00</td>
<td></td>
</tr>
<tr>
<td>Colic</td>
<td>72.83</td>
<td>74.29</td>
<td>75.17</td>
<td>75.49</td>
<td>76.25</td>
<td>77.34</td>
<td></td>
</tr>
<tr>
<td>Diabetes</td>
<td>67.82</td>
<td>69.80</td>
<td>70.80</td>
<td>70.90</td>
<td>71.20</td>
<td>72.40</td>
<td></td>
</tr>
<tr>
<td>Heart</td>
<td>67.27</td>
<td>68.50</td>
<td>71.61</td>
<td>70.27</td>
<td>71.63</td>
<td>74.00</td>
<td></td>
</tr>
<tr>
<td>Hepatitis</td>
<td>76.00</td>
<td>77.12</td>
<td>77.29</td>
<td>78.71</td>
<td>80.97</td>
<td>82.40</td>
<td></td>
</tr>
<tr>
<td>Ionosphere</td>
<td>70.47</td>
<td>72.62</td>
<td>71.61</td>
<td>72.29</td>
<td>73.43</td>
<td>82.15</td>
<td></td>
</tr>
<tr>
<td>Liver</td>
<td>56.80</td>
<td>57.28</td>
<td>60.88</td>
<td>61.10</td>
<td>60.80</td>
<td>58.32</td>
<td></td>
</tr>
<tr>
<td>Sonar</td>
<td>63.40</td>
<td>64.12</td>
<td>65.00</td>
<td>65.10</td>
<td>66.43</td>
<td>64.31</td>
<td></td>
</tr>
<tr>
<td>Tic-tac-toe</td>
<td>66.40</td>
<td>67.40</td>
<td>66.02</td>
<td>68.40</td>
<td>67.33</td>
<td>67.17</td>
<td></td>
</tr>
<tr>
<td>Vote</td>
<td>89.08</td>
<td>90.43</td>
<td>91.37</td>
<td>92.12</td>
<td>92.18</td>
<td>92.15</td>
<td></td>
</tr>
<tr>
<td>Wave</td>
<td>83.10</td>
<td>84.63</td>
<td>85.13</td>
<td>85.67</td>
<td>86.75</td>
<td>88.00</td>
<td></td>
</tr>
</tbody>
</table>

Statistical Analysis

In this section the results in Table 3.3 are analyzed using statistical tests. Table 3.5 shows rank of each algorithm for each dataset according to Friedman’s test. Average ranks are reported in the last row. Friedman’s test checks whether the measured average ranks are significantly different from the mean rank $R_j = 2.96$ expected under the null-hypothesis. Then, according to the equations (4.17) and (3.5), $\chi^2_F = 39.54$ and $F_F = 31.24$. 
### 3.7. Results

Table 3.5: Statistical Rank (Friedman’s test)

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Decision Tree</th>
<th>C4.4</th>
<th>J48graft</th>
<th>C4.4graft</th>
<th>NBTree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breath-cancer</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Bupa</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Car</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Cmc</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Colic</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Diabetes</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Heart</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Hepatitis</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Liver</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Sonar</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Tic-tac-toe</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Vote</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Wave</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td><strong>Average Rank</strong></td>
<td><strong>4.93</strong></td>
<td><strong>3.50</strong></td>
<td><strong>2.93</strong></td>
<td><strong>1.64</strong></td>
<td><strong>1.93</strong></td>
</tr>
</tbody>
</table>

With five algorithms and 14 datasets, \( F_F \) is distributed according to the \( F - distribution \) with \( 5 - 1 = 4 \) and \( (5 - 1)(14 - 1) = 39 \) degrees of freedom. The critical value of \( F(4, 39) \) for \( \alpha = 0.05 \) is 2.61, so we reject the null-hypothesis. In the next step we use Holm’s test. For that, we have to compute and order the corresponding statistics and \( p \) values.

Table 3.6: Statistical Rank (Holm’s test)

<table>
<thead>
<tr>
<th>( i )</th>
<th>classifier</th>
<th>( Z )</th>
<th>P-value</th>
<th>( \alpha/(k - i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C4.4graft</td>
<td>5.498051603</td>
<td>0.00000004</td>
<td>0.0125</td>
</tr>
<tr>
<td>2</td>
<td>NBTree</td>
<td>4.780914437</td>
<td>0.00000174</td>
<td>0.016666667</td>
</tr>
<tr>
<td>3</td>
<td>J48graft</td>
<td>3.82473155</td>
<td>0.00013092</td>
<td>0.025</td>
</tr>
<tr>
<td>4</td>
<td>C4.4</td>
<td>2.031888636</td>
<td>0.0421658</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Table 3.6 shows the results. The Holm procedure rejects the first, second, third, and then fourth hypotheses, since the corresponding \( p \) values are smaller than the adjusted \( \alpha \)'s (0.05). We conclude that the performance of C4.4graft, NBTree, J48graft, and C4.4 as the base learner in self-training algorithm are significantly different from standard decision tree (J48).

Based on the results of the experiments we conclude that improving the probability estimation of the tree classifiers leads to better selection metric for the self-training algorithm and produces better classification model. We observe that using Laplacian correction, No-pruning, grafting, and NBTree produce better probability estimation in tree classifiers. We also observed that Mahalanobis distance method for sampling is useful and guides a decision tree learner to select a set of high-confidence predictions.
3.7.2 Self-Training with an Ensemble of Trees

In this experiment, we analyze the effect of modifying the decision tree learner when it is used as the base learner in an algorithm for learning an ensemble of trees. The ensemble learner itself is the base learner in self-training. Because for many domains ensembles show better results than a single base learner. We expect that the ensemble learner will perform somewhat better than the basic decision tree learner when used on the labeled data only. More interesting thought, we expect that the ensemble, with improved base learner, can improve the probability estimation and therefore if it is used as the base learner in self-training it will benefit even more from the unlabeled data than a single modified decision tree learner.

As a base classifier in RF we use C4.4graft. Also, REPTree, NBTree, and C4.4graft classifiers are used in a RSM ensemble classifier as the base classifiers. REPTree is a fast decision tree learner. It builds a decision tree using information gain and prunes it using reduced-error pruning (with backfitting). REPTree only sorts values for numeric attributes once. Missing values are dealt with by splitting the corresponding examples into pieces like in C4.5 tree. It is a default decision tree learner for Random Subspace Method in WEKA.

Table 3.7: Average performance of supervised learning and self-training with ensemble classifiers

<table>
<thead>
<tr>
<th>Dataset</th>
<th>RFG</th>
<th>ST-RFG</th>
<th>RREP</th>
<th>ST-RREP</th>
<th>RG</th>
<th>ST-RG</th>
<th>RNB</th>
<th>ST-RNB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breath-cancer</td>
<td>66.25</td>
<td>68.75</td>
<td>68.50</td>
<td>69.50</td>
<td>68.50</td>
<td>71.50</td>
<td>74.50</td>
<td>75.50</td>
</tr>
<tr>
<td>Bupa</td>
<td>57.34</td>
<td>60.32</td>
<td>56.45</td>
<td>57.72</td>
<td>55.04</td>
<td>59.38</td>
<td>58.40</td>
<td>58.40</td>
</tr>
<tr>
<td>Car</td>
<td>87.00</td>
<td>88.32</td>
<td>77.60</td>
<td>80.40</td>
<td>80.60</td>
<td>83.02</td>
<td>78.00</td>
<td>80.80</td>
</tr>
<tr>
<td>Cmc</td>
<td>60.25</td>
<td>63.25</td>
<td>58.75</td>
<td>59.63</td>
<td>59.50</td>
<td>63.70</td>
<td>57.25</td>
<td>59.38</td>
</tr>
<tr>
<td>Colic</td>
<td>75.00</td>
<td>74.90</td>
<td>67.32</td>
<td>71.65</td>
<td>77.50</td>
<td>79.60</td>
<td>76.77</td>
<td>79.26</td>
</tr>
<tr>
<td>Diabetes</td>
<td>69.56</td>
<td>71.66</td>
<td>70.66</td>
<td>70.75</td>
<td>67.82</td>
<td>70.56</td>
<td>70.04</td>
<td>72.29</td>
</tr>
<tr>
<td>Heart</td>
<td>74.99</td>
<td>76.22</td>
<td>72.04</td>
<td>74.58</td>
<td>73.18</td>
<td>76.09</td>
<td>70.91</td>
<td>73.40</td>
</tr>
<tr>
<td>Hepatitis</td>
<td>80.00</td>
<td>80.80</td>
<td>80.00</td>
<td>80.00</td>
<td>80.40</td>
<td>81.80</td>
<td>79.60</td>
<td>82.00</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>80.00</td>
<td>82.00</td>
<td>71.20</td>
<td>73.80</td>
<td>73.04</td>
<td>77.76</td>
<td>78.31</td>
<td>81.10</td>
</tr>
<tr>
<td>Liver</td>
<td>56.60</td>
<td>58.14</td>
<td>56.00</td>
<td>58.40</td>
<td>61.40</td>
<td>63.00</td>
<td>56.80</td>
<td>58.00</td>
</tr>
<tr>
<td>Sonar</td>
<td>63.60</td>
<td>67.20</td>
<td>59.20</td>
<td>60.60</td>
<td>63.40</td>
<td>64.80</td>
<td>59.80</td>
<td>61.20</td>
</tr>
<tr>
<td>Tic-tac-toe</td>
<td>70.00</td>
<td>71.40</td>
<td>67.20</td>
<td>67.60</td>
<td>69.60</td>
<td>69.60</td>
<td>68.20</td>
<td>70.40</td>
</tr>
<tr>
<td>Vote</td>
<td>91.25</td>
<td>93.25</td>
<td>88.78</td>
<td>90.25</td>
<td>89.00</td>
<td>93.00</td>
<td>88.78</td>
<td>92.50</td>
</tr>
<tr>
<td>Wave</td>
<td>86.00</td>
<td>88.50</td>
<td>85.75</td>
<td>86.75</td>
<td>87.25</td>
<td>89.50</td>
<td>88.75</td>
<td>89.75</td>
</tr>
</tbody>
</table>

Table 3.7 gives the classification accuracies of the all experiments on the UCI datasets. The columns RFG, RREP, RG, and RNB show the classification performance of supervised classifiers Random Forest with C4.4graft and RSM with REPTree, C4.4graft, and NBTree respectively and their corresponding self-training algorithms. Using RF with C4.4graft as the base learner in self-training improves the classification performance of the supervised classifier RF, on 13 out of 14 datasets. As can be seen, the results are better than a single decision tree,
3.7. Results

but in most cases the differences are not significant. We suspect that it is due to using the bagging method for generating different training set in Random Forest. In the case of a small set of labeled data, bagging does not work very well [94], because the pool of labeled data is too small for re-sampling. However the average improvement is 1.9% over all datasets.

In the second experiment, we use the RSM ensemble classifier with improved versions of decision trees. We observe that using C4.4graft and NBTree as the base classifiers in RSM is more effective than using REPTree, when RSM is used as the base learner in self-training. The results show that RSM with C4.4graft as the base learner in self-training improves the classification performance of RSM on 13 out of 14 datasets and the average improvement over all datasets is 2.7%. The same results are shown in Table 3.7 for RSM, when the NBTree is the base learner.

Self-Training with Ensemble Classifier and Distance-based Measure

Table 3.8 shows the results of the experiments on UCI dataset using Algorithm 4. The columns ST-RFG, ST-RREP, ST-RG, and ST-RNB show the classification performance of self-training with RF and RSM with REPTree, C4.4graft, and NBTree as the base learners respectively. The results, as in single classifier, show that in general using the Mahalanobis distance along with the probability estimation as the selection metric improves the classification performance of all self-training algorithms and emphasizes the effectiveness of this selection metric.

Table 3.8: Average performance of supervised learning and self-training using the Mahalanobis distance along with the probability estimation as the selection metric

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Supervised</th>
<th>ST-RF</th>
<th>ST-RREP</th>
<th>ST-RG</th>
<th>ST-RNB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breath-cancer</td>
<td>68.00</td>
<td>69.80</td>
<td>70.50</td>
<td>71.95</td>
<td>76.93</td>
</tr>
<tr>
<td>Bupa</td>
<td>58.62</td>
<td>61.60</td>
<td>60.76</td>
<td>61.06</td>
<td>60.04</td>
</tr>
<tr>
<td>Car</td>
<td>86.08</td>
<td>89.27</td>
<td>81.40</td>
<td>84.29</td>
<td>81.18</td>
</tr>
<tr>
<td>Cmc</td>
<td>57.00</td>
<td>64.00</td>
<td>60.00</td>
<td>64.12</td>
<td>60.10</td>
</tr>
<tr>
<td>Colic</td>
<td>72.83</td>
<td>75.21</td>
<td>71.15</td>
<td>80.60</td>
<td>79.32</td>
</tr>
<tr>
<td>Diabetes</td>
<td>67.82</td>
<td>71.80</td>
<td>70.98</td>
<td>71.80</td>
<td>72.00</td>
</tr>
<tr>
<td>Heart</td>
<td>67.27</td>
<td>77.01</td>
<td>74.15</td>
<td>76.62</td>
<td>74.27</td>
</tr>
<tr>
<td>Hepatitis</td>
<td>76.00</td>
<td>81.81</td>
<td>81.25</td>
<td>82.15</td>
<td>83.00</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>70.47</td>
<td>82.34</td>
<td>74.79</td>
<td>78.56</td>
<td>81.91</td>
</tr>
<tr>
<td>Liver</td>
<td>56.80</td>
<td>59.00</td>
<td>57.92</td>
<td>63.60</td>
<td>58.52</td>
</tr>
<tr>
<td>Sonar</td>
<td>63.40</td>
<td>69.07</td>
<td>64.15</td>
<td>69.23</td>
<td>63.14</td>
</tr>
<tr>
<td>Tic-tac-toe</td>
<td>66.40</td>
<td>72.01</td>
<td>69.34</td>
<td>70.67</td>
<td>71.33</td>
</tr>
<tr>
<td>Vote</td>
<td>89.08</td>
<td>93.25</td>
<td>91.53</td>
<td>93.17</td>
<td>93.79</td>
</tr>
<tr>
<td>Wave</td>
<td>83.10</td>
<td>88.15</td>
<td>86.97</td>
<td>89.50</td>
<td>89.86</td>
</tr>
</tbody>
</table>
3.7.3 Results of Experiments on Web-page Datasets

As mentioned earlier, we use visual features of web-pages to classify the web-pages dataset. For this task in the experiments, we use the Mahalanobis distance along with the probability estimation as the selection metric in self-training. As before, we use 10% labeled examples of Aesthetics and Recency, i.e. only six labeled examples, which is well-fit to semi-supervised learning. The reason why we use these datasets is that they consist a large set of features which is well-suited for ensemble methods RSM and RF.

Tables 3.9 and 3.10 show the classification performance of the used datasets. As shown, consistent with our previous results on UCI datasets, NBTree and C4.4graft base classifiers in self-training achieve the best results.

Table 3.9: Average Classification Accuracy of supervised learning and self-training with single classifier on web-pages datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>DT</th>
<th>ST-DT</th>
<th>C4.4</th>
<th>ST-C4.4</th>
<th>NB</th>
<th>ST-NB</th>
<th>C4G</th>
<th>ST-C4G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aesthetics</td>
<td>53.47</td>
<td>57.70</td>
<td>54.12</td>
<td>57.87</td>
<td>54.40</td>
<td>59.22</td>
<td>54.48</td>
<td>60.45</td>
</tr>
<tr>
<td>Recency</td>
<td>65.27</td>
<td>68.08</td>
<td>66.01</td>
<td>70.71</td>
<td>67.00</td>
<td>72.47</td>
<td>67.50</td>
<td>74.36</td>
</tr>
</tbody>
</table>

In the second experiment we use ensemble self-training. Results in table 3.10 show that ensemble RSM classifier with NBTree and C4.4graft, as the base learners, achieves the best classification performance. Finally, comparing Table 3.9 to Table 3.10, shows that ensemble methods outperform the single classifier. The results also verify that improving both the classification accuracy and the probability estimates of the base learner in self-training are effective for improving the performance.

Table 3.10: Average performance of supervised RF, RSM-REPTree, RSM-NBTree, and RSM-C4.4graft and their self-training algorithms

<table>
<thead>
<tr>
<th>Dataset</th>
<th>RFG</th>
<th>ST-RFG</th>
<th>RREP</th>
<th>ST-RREP</th>
<th>RNB</th>
<th>ST-RNB</th>
<th>RG</th>
<th>ST-RG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aesthetics</td>
<td>58.55</td>
<td>61.91</td>
<td>60.88</td>
<td>61.04</td>
<td>60.41</td>
<td>65.01</td>
<td>63.88</td>
<td>68.91</td>
</tr>
<tr>
<td>Recency</td>
<td>68.49</td>
<td>71.87</td>
<td>70.37</td>
<td>72.3</td>
<td>71.29</td>
<td>75.57</td>
<td>70.37</td>
<td>78.74</td>
</tr>
</tbody>
</table>

Sensitivity to the Amount of Labeled Data

To study the sensitivity of the proposed algorithms to the number of labeled data, we run a set of experiments with different proportions of labeled data which vary from 10% to 40%. We expect that the difference between the supervised algorithms and the semi-supervised methods decreases when more labeled data are available. Figure 3.4 shows the performance of self-training with different base learners on two web-pages datasets. In this experiment, we use RF and RSM with C4.4graft as the base learner in self-training. For fair comparison we include
3.7. Results

a set of experiments with single classifiers, J48 and C4.5graft, as well. Figure 3.4 shows the performance obtained by self-training algorithms and supervised classifiers. Figure 3.4.a and 3.4.b show the performance of self-training with ensemble classifiers and Figure 3.4.c and 3.4.d give the performance of self-training with single classifiers on web-pages datasets. Consistent with our hypothesis we observe that difference between supervised algorithms and self-training methods decreases when the number of labeled data increases. Another interesting observation is that RF improves the classification performance of self-training when more labeled data are available, because with more labeled data the bagging approach, used in RF, generates diverse decision trees.

Figure 3.4: Average Performance of Self-Training (using ensemble classifier (a and b) and single classifier (c and d) as the base learner) with increasing proportions of labeled data on web-pages datasets
Chapter 3. Self-Training With Decision Trees

Sensitivity to the number of trees

In this experiment, we study the impact of the number of trees in ensemble methods RF and RSM, as the base learner of self-training, on the performance. Figure 3.5 shows the performance of self-training with different number of trees on two datasets. As before, we use 10% labeled data in this experiment. The NBTree, C4.4graft, and REPTree are used as the base learners in RSM meta classifier.

![Figure 3.5: Average Performance of Self-Training with increasing the number of trees on web-pages datasets](image)

It is not surprising that overall the classification accuracy is improved with increasing the number of trees. We suspect that when the number of trees is increasing in ensemble methods, the probability estimates will improve. Consequently, it improves the performance of the self-training algorithm.

3.8 Conclusion and Discussion

The main contribution of this chapter is the observation that when a learning algorithm is used as the base learner in self-training, it is very important that the confidence of prediction is correctly estimated, probability estimation. The standard technique of using the distribution at the leaves of decision tree as probability estimation does not enable self-training with a decision tree learner to benefit from unlabeled data. The accuracy is the same as when the decision tree learner is applied to only the labeled data. If a modified decision tree learner is used which has an improved technique for estimating probability, then self-training with the modified version does benefit from the unlabeled data. Although to a lesser extent, the same is true when the modified decision tree learners are used as the base learner in an ensemble learner.
Our experimental results showed that five modifications: No-pruning with Laplacian correction, Grafting, No-pruning with Laplacian correction with Grafting, NBTree, and global distance-based measure, enable self-training to benefit from unlabeled data, even when there is only a small number of labeled data. The best result based on experiments with a small amount of labeled instances (10%), which is the most relevant for semi-supervised settings, was obtained by a combination of grafting, No-pruning and Laplacian correction, called C4.4graft. This was useful in the Random Subspace Method as well. Random Forest suffers from the small amount of labeled data and therefore does not work well. Better probability-based ranking and high classification accuracy could select the high-confidence predictions in the selection step of self-training and therefore, these variations improved the performance of self-training.

Future work, can consider extending the proposed selection metric to multiclass classification problem. In co-training also the selection of a set of high-confidence predictions is vital to achieve better performance. It is interesting to see how the proposed methods in this chapter behave when they are used for co-training.