View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Research Commons@Waikato

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

Research Commons

http://researchcommons.waikato.ac.nz/

Research Commons at the University of Waikato

Copyright Statement:

The digital copy of this thesis is protected by the Copyright Act 1994 (New Zealand).

The thesis may be consulted by you, provided you comply with the provisions of the Act and the following conditions of use:

- Any use you make of these documents or images must be for research or private study purposes only, and you may not make them available to any other person.
- Authors control the copyright of their thesis. You will recognise the author's right to be identified as the author of the thesis, and due acknowledgement will be made to the author where appropriate.
- You will obtain the author's permission before publishing any material from the thesis.

Department of Computer Science

Hamilton, NewZealand

Smoothing in Probability Estimation Trees

Zhimeng Han

This thesis is submitted in partial fulfilment of the requirements for the degree of Master of Science at The University of Waikato.

March 2011

© 2011 Zhimeng Han

Abstract

Classification learning is a type of supervised machine learning technique that uses a classification model (e.g. decision tree) to predict unknown class labels for previously unseen instances. In many applications it canbe very useful to additionally obtain class probabilities for the different class labels. Decision trees that yield these probabilities are also called probability estimation trees(PETs). Smoothing is a technique used to improve the probability estimates. There are several existing smoothing methods, such as the Laplace correction (Provost & Domingos, 2003), M-Estimate smoothing (Dzeroski, Cestnik & Petrovski, 1993) and M-Branch smoothing (Ferri, Flach & Hernández-Orallo, 2003). Smoothing does not just apply to PETs. In the field of text compression, PPM (Cleary & Witten, 1984) in particular, smoothing methods play a important role. This thesis migrates smoothing methods from text compression to PETs. The newly migrated methods in PETs are compared with the best of the existing smoothing methods considered in this thesis under different experiment setups. Unpruned, pruned and bagged tree are considered in the experiments. The main finding is that the PPM-based methods yield the best probability estimates when used with bagged trees, but not when used with individual (pruned or unpruned) trees.

Acknowledgments

I would like to thank my supervisor Assoc. Prof. Eibe Frank for the guidance throughout not just the thesis, but also with code implemented in Weka. He provided extensive help and materials for the problems I encountered during the thesis write up and the development. He guided me with lots of patience and understanding.

I would also like to thank Sun Quan, who is a member of the machine learning group at the University of Waikato. He gave me lots of help with the UCSD machine learning competition.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

Consider the following definition of machine learning, as given by Arthur Samuel (Samuel, 1959):

> Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed.

In machine learning, there are many ways to achieve the goal, such as regression, classification and clustering. This thesis investigates a learning technique called probability estimation tree learning. It evaluates how well smoothing methods used in text compression work in probability estimation trees. Can smoothing methods from text compression be used to improve the accuracy of the probability estimates provided by probability estimation trees?

Probability estimation is important in cost-sensitive classification. In practical classification tasks, there are errors in predictions made by the learner. Some errors have a higher cost than others, and it is best to minimize the cost of errors when making predictions. Information on miss-classification costs can be used to build cost-sensitive models or make cost sensitive predictions by minimizing the expected cost (Witten & Frank, 2005). Class probabilities can be used to perform cost-sensitive predictions, but accurate probability estimates are needed to yield good performance.

The remainder of this chapter is structured as follows. Section 1.1 outlines the basic concepts. Section 1.2 discusses the ideas necessary for understanding probability estimation trees. Section 1.2.1 covers the basics about smoothing. Section 1.3 introduces the motivation and objectives of this thesis. Section 1.4 lists the structure of the rest of this thesis.

1.1 Basic machine learning concepts

As discussed above, machine learning is a set of techniques that discovers and extracts patterns from data. The process will involve the input data, and the output. The input involves datasets, attributes and instances. Tables 1.1 and 1.2 each show a small part of a dataset. The rows of the table are instances, and each row represents a single instance in the dataset. The columns are attributes: *sepal length*, *sepal width*, *petal length*, *petal width* and flower type are the attributes in the iris dataset.

The learning process, is a potentially complex process that can involves various techniques such as classification, regression, clustering and association. Classification learning takes a dataset, which has instances with class labels, to build a model based on this data. Then it applies the model to data that does not have any class label to predict the class value. In contrast, clustering divides the data into several natural groups instead of labeling each instance with a class value based on labeled training data. Regression is similar to classification, but it uses numeric values as class labels. Association learning is the technique that finds all kinds of relationships between instances, and does not just predicting the class values.

1.2 Probability estimation trees

A probability estimation tree (PET) is a model used in classification learning. It is built using a dataset with class values, and used to predict class probabilities for previously unseen data without class values.

PETs are usually constructed using a top-down fashion. One of the attributes will be selected to define the split at the root node. The way to choose which attribute to split on is to measure the purity of the child nodes after the split. The most common measurement of impurity is called information (Quinlan, 1986) or entropy (Witten & Frank, 2005).

Figure 1.1 shows a PET induced for the iris data. As mentioned above, table 1.2 shows an excerpt of this data. The last column is the class attribute. The 5 attributes in total: sepal length, sepal width, petal length, petal width and class values. The tree predicts probabilities for the class values given the other attributes. In this case only two of the attributes are used in the tree.

Sepal length	Sepal width	Petal length	Petal width	Flower type
6.0	2.9	4.5	1.5	Unknown

Table 1.1: The iris data

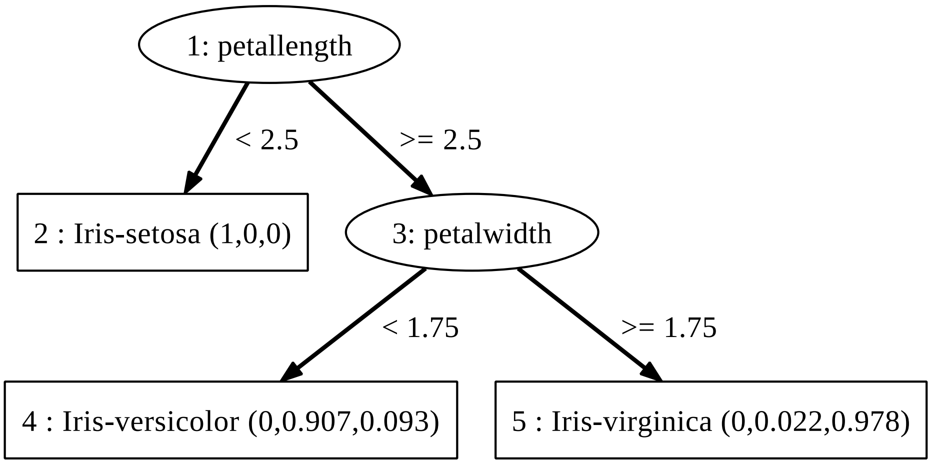

Figure 1.1: A PET built with Iris data

The tree can be used to read off probability estimates for new test cases. For example assume we are predicting an iris flower with attribute values shown in Table 1.1: Prediction starts at the root of tree, where the splitting attribute is petal length. The petal length of the iris is greater than 2.5, so we go to the right branch. The second split attribute is petal width, and petal width of the iris is less than 1.75, so we follow the left branch. Then we do not have any further splits, so we can conclude this iris is of type versicolor, and the predicted probability is 0.907. The estimated probabilities for the other classes are: Iris-setosa 0, Iris-virginica 0.093.

1.2.1 Smoothing

Smoothing is a technique widely used in the field of machine learning to compensate for data sparseness. Lack of data can lead to highly variable estimates. Smoothing provides a way to make estimates more robust. Smoothing is used in other fields too, such as text compression. The text compression method PPM (Prediction by Partial Matching) and its variants (PPMD, PPMP and etc.) are widely used in everyday application. In (Bell, Cleary & Witten, 1990), three smoothing methods that are generally used in PPM are introduced. This thesis empirically investigates the performance of these smoothing methods when used in PETs. Without smoothing, probability estimates at the leaf

	Sepal length	Sepal width	Petal length	Petal width	Flower type
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5.0	3.6	1.4	0.2	Iris-setosa
...					
51	7.0	3.2	4.7	1.4	Iris-versicolor
52	6.4	3.2	4.5	1.5	Iris-versicolor
53	6.9	3.1	4.9	1.5	Iris-versicolor
54	5.5	2.3	4.0	1.3	Iris-versicolor
55	6.5	2.8	4.6	1.5	Iris-versicolor
...					
101	6.3	3.3	6.0	2.5	Iris-virginica
102	5.8	2.7	5.1	1.9	Iris-virginica
103	7.1	3.0	5.9	2.1	Iris-virginica
104	6.3	2.9	5.6	1.8	Iris-virginica
105	6.5	3.0	5.8	2.2	Iris-virginica
...					

Table 1.2: The iris data

nodes are based simply on observed relative frequencies of class values at the leaf nodes. Smoothing adjusts these estimates to make them more robust.

Let us consider the effect of smoothing using a so-called ROC curve (Green & Swets, 1966). Probability estimation methods are often evaluated using ROC curves, which will be discussed in more detail in the next chapter. The larger the area under the ROC curve, which plots true positive predictions against false positive ones, the better the probability estimation method being evaluated. ROC curves are only defined for 2-class problems, where one class is considered negative and the other positive. The area under curve gives the probability that a randomly chosen positive instance receives a higher positive class probability than a randomly chosen negative one.

The plots in Figures 1.2 were generated using the same dataset and the same classifier. The only difference between them is that the red curve is based on no smoothing at all, and the blue curve is based on a technique called M-Branch smoothing discussed later in this thesis. As we can see, the area under the ROC curves is different. Ideally the area in one, which means all positive instances are ranked above all negative ones. Smoothing increases the area under the curve, which is the desired results.

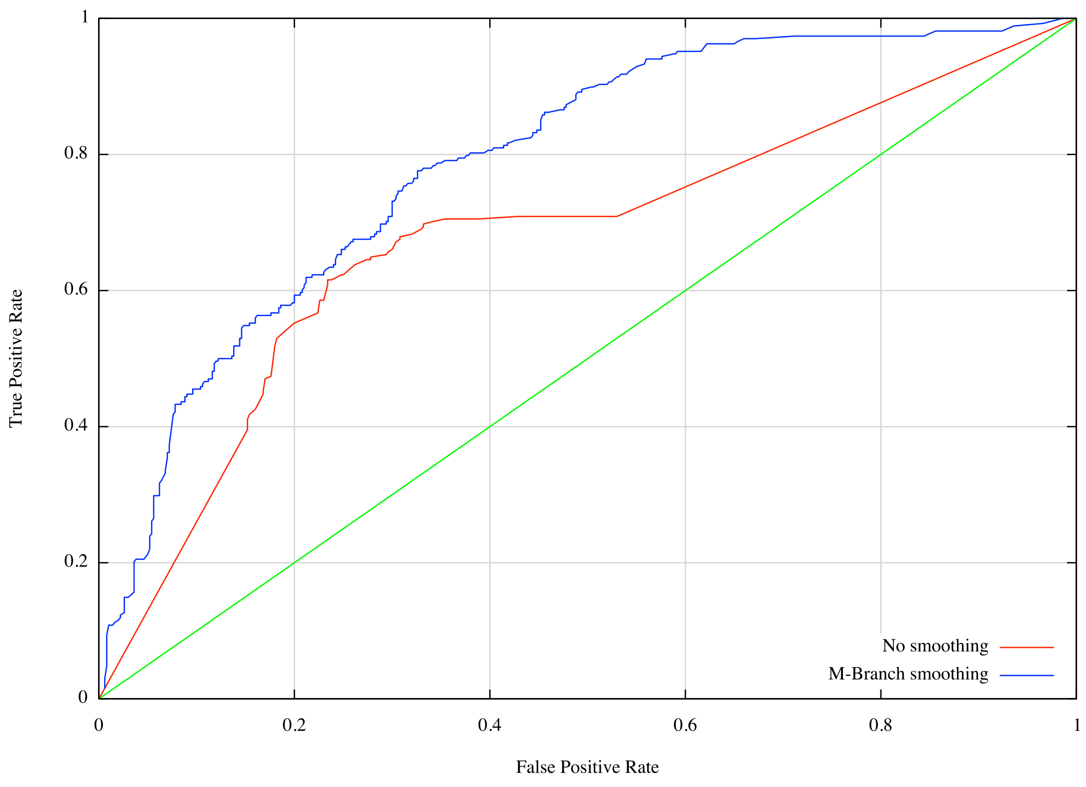

Figure 1.2: ROC curve with and without smoothing

1.2.2 Cost sensitive classification

Cost sensitive classification is a technique used to minimize the cost of the errors made by classifiers. For example the Weka machine learning workbench (Witten & Frank, 2005) assumes that the errors have the same cost by default, but if a cost matrix is given, it can use this information to pick predictions that minimize the expected cost. In that case, the probability estimates of the classifier are used to calculated the expected cost for each prediction, and the one that minimizes the expected cost is chosen as the final prediction. Cost-sensitive prediction, as opposed to coast-sensitive learning, has the advantage that costs can be varied at prediction time without changing the classifier. However, accurate probability estimates are required for this.

For example, we are trying to classify a patient as having meningitis or not having meningitis, where the estimated probability from the classifier is 0.9 that the patient does not have meningitis. Without any cost on the estimated outcomes, we would classify the patient as "no meningitis" case. However, this is different if the outcomes have costs. If the cost of misclassifying a true meningitis case (false negative) is 100 times higher than the cost of a false positive. The predicted outcome should have the minimum expected cost rather than just the most likely value, which means in this particular case, we classify the patient as potentially having meningitis and keep him/her under observation, perhaps performing further medical tests.

1.3 Motivation and Objectives

It is well-known that smoothing techniques can improve the accuracy of class probability estimation. The most basic smoothing method is the so-called Laplace correction (Provost & Domingos, 2003) and it is also implemented in the well-known Weka software. However, there are other more advanced smoothing methods that are not implemented in Weka, such as M-Branch smoothing (Ferri, Flach & Hernández-Orallo, 2003). In the field of text compression, there are also some smoothing techniques, and it has not been investigated how well these text compression smoothing methods perform in tree learners. In order to compare smoothing techniques for PETs in Weka, the objectives of this thesis are as follows:

1. Implement the smoothing methods introduced in (Ferri, Flach & Hernández-Orallo, 2003) as a state-of-the-art baseline to compare against.

2. Implement the PPM smoothing methods introduced in (Bell, Cleary & Witten, 1990)

3. Run experiments on the different smoothing method implemented

4. Summarize the experiment results and draw conclusions regarding relative performance.

1.4 Thesis Structure

Chapter 2 will present some background on important concept used in this thesis, such as class probability estimation trees, how to evaluate class probability estimates, and how to blend and adjust probability estimates. In this chapter, the datasets used in the experiments are also introduced and discussed. The data-mining tool Weka will be briefly introduced in this chapter too.

Chapter 3 will describe the different smoothing methods considered in this thesis in depth. Eight kinds of smoothing method will be considered in this chapter: Laplace correction, M-Branch smoothing (Ferri, Flach & Hernández-Orallo, 2003) and M-Estimate smoothing (Dzeroski, Cestnik & Petrovski, 1993), and 5 PPM methods: PPMA, PPMB, PPMC, PPMD and PPMP. Note that there are more PPM-related smoothing methods, but they are beyond the scope of this thesis. Chapter 3 will discuss the code implemented into one of the tree learners in Weka. The effect on probability estimates will be discussed along with the pseudo code.

The experimental results will be presented in Chapter 4 in tabular and graphical form.

Chapter 5 will summarize the previous chapters, draw conclusions and describe potential future work.

Chapter 2
Background and Related work

This chapter discusses background knowledge relevant for this thesis: the basic method for building a class probability estimation tree using information gain and the Laplace correction, building ensemble of trees using bagging, the PPM text compression method, and the datasets and evaluation measures used to evaluate probability estimations in this thesis.

This chapter has the following structure. Section 2.1 introduce the tree learner used in this thesis. Section 2.2 explains the bagging method for ensemble learning. Section 2.3 describes smoothing in PPM text compression. Section 2.4 discusses the datasets used. 2.5 lists and explains different analysis and evaluation techniques.

2.1 Learning Tree Classifiers

The coding done for this thesis was based on Weka (Witten & Frank, 2005). Weka is a collection of state-of-the-art machine learning algorithms and data preprocessing tools. The tree learner used to test and experiment with in this thesis is called REPTree. It is a fast tree learner that uses information gain to find split points and attributes. It is fast because it sorts values for numeric attributes only once. Another reason REPTree is used in this thesis is that it is similar to the tree learner described in the paper by Ferri, C., Flach, P., and Hernández-Orallo, J. (2003), which describes the state-of-the-art M-Branch smoothing technique that we will compare to. In this way our results are more comparable with those in the original paper.

There is a simple top-down method for building trees. Information gain is used to decide which attribute is used to split nodes in this process. Higher information gain represents higher purity of child nodes. An impure node has a very mixed distribution of class values. The reason to have a pure child node is that, the more pure the child is, the smaller the subtree is likely to be. The basic algorithm to construct a fully-expanded

```
Start on the root of the tree.
If node is not pure or no further splitting is possible:
  Find which attribute is the best to split on using information gain
  Split the node into subsets on possible values
  Go to each child node and recurse.
```

Figure 2.1: Pseudo code for constructing a PET

```
Start on the root of the tree.
If current node is not a leaf node:
  Find the split attribute and find the correct path to the child node.
Else
  return class proabbility distribution
```

Figure 2.2: Pseudo code for making a prediction using a PET

PET is given in Figure 2.1.

For making a prediction using a PET, we start off at the root node of the tree and check if the node is a leaf node. If it is, then the prediction is over. If it is not, we check the split attribute and find the correction path to go to the child node, then we do the same process recursively until the instance reaches a leaf node. The basic prediction pseudo code for a PET is given in Figure 2.2.

2.1.1 Pruning

Sometimes, a fully-grown tree does not perform as well as a smaller tree. To reduce the vulnerability to noise and variability in the fully grown tree, a process is needed to make the tree smaller and more straight-forward to use. Pruning is a technique that is widely used to reduce a tree to the right size. A pruned tree may not represent a dataset exactly, but a precisely constructed tree does not imply better probability estimates because it may overfit the training data.

For example, Figure 2.3 shows an unpruned probability estimation tree for the iris data. In contrast to Figure 1.1, the unpruned version of the PET has 11 nodes instead of 5 in Figure 1.1. The right branch from node number 3 (labelled *petal width*) has been pruned to approximate the unpruned tree. In the unpruned tree, as we can see, after splitting by petal length in node 4 there is a further split on petal width again, which is the same splitting attribute as in node 3.

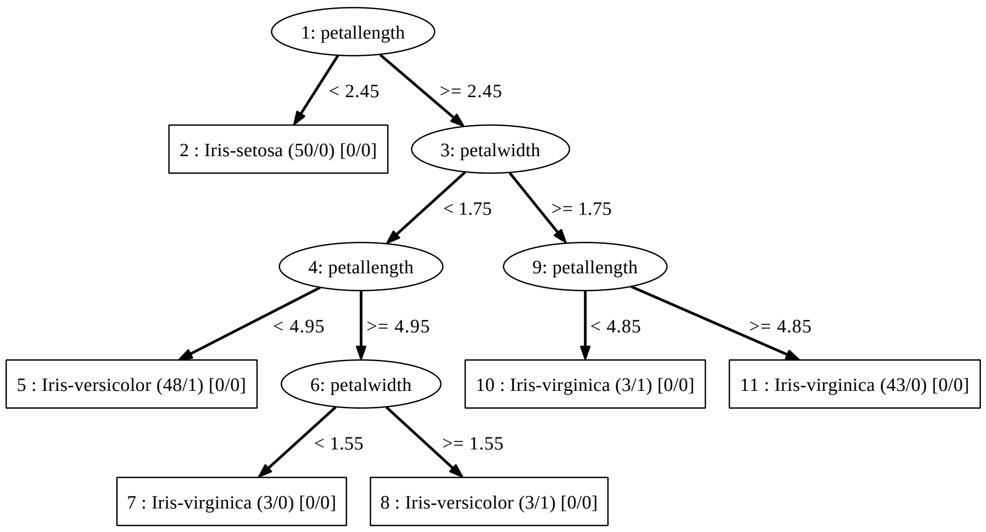
1: petallength
< 2.45
>= 2.45
2 : Iris-setosa (50/0) [0/0]
3: petalwidth
< 1.75
>= 1.75
4: petallength
9: petallength
< 4.95
>= 4.95
< 4.85
>= 4.85
5 : Iris-versicolor (48/1) [0/0]
6: petalwidth
10 : Iris-virginica (3/1) [0/0]
11 : Iris-virginica (43/0) [0/0]
< 1.55
>= 1.55
7 : Iris-virginica (3/0) [0/0]
8 : Iris-versicolor (3/1) [0/0]

Figure 2.3: Unpruned PET for iris data

Observing the leaf nodes 7 and 8, the instance counts in these two nodes are relatively small. If we look at node number 9, we can see that the child nodes under it have the same class: *Iris-virginica*, which means the split is not necessary to minimize classification error. After pruning, the subtree from node 4 and below becomes a leaf node with class label *Iris-versicolor* and the instance counts of all the nodes below it are combined. The same process has been done with the subtree attached to node 9 and its child nodes.

Pruning PETs sometimes makes the tree perform better than the original tree. Two general kinds of pruning exist (Witten & Frank, 2005): pre-pruning and post-pruning, where pre-pruning tries to stop unnecessary splits while growing the tree; on the other hand post-pruning grows the entire tree first then tries to prune the tree based on pruning criteria.

Reduced-error pruning (Quinlan, 1987) is a kind of pruning method where we split the data up. Normally two thirds of the data is used to construct the tree and the rest of the data is used to prune the tree by minimizing an error criterion on the pruning data. This method is implemented in the REPTree classifier that is used in this thesis.

2.1.2 Adjusting probability estimates

When the PET is grown, the training dataset used to grow the tree can potentially have zero appearance for an outcome (i.e. class value) at a particular leaf node. Without any smoothing methods, this outcome will be assigned a probability of zero, but assigning a zero probability outcome is problematic for probability estimations in real-world problems. This problem is called the zero-frequency problem (Roberts, 1982).

The zero-frequency problem can occur frequently in practice. This is bad because before predictions are made one or more of the outcomes have already been completely eliminated: Even though the event has never happened before, it does not imply it will never happen in the future. To prevent this problem, an adjustment process is necessary, to adjust the probability estimates of the outcomes, so that we do not have the zero-frequency problem. There are many different ways to adjust the probability estimates. For example, the Laplace correction is a simple smoothing method used to solve this problem. It adds 1 to each class count in the leaf nodes, and it thus effectively solves the zero-frequency problem. However, in practice adding one to each class count may cause some other problems, such as over-smoothing the probability estimates, especially when the dataset is small, perhaps becuase the tree has many branches and each leaf node contains a very small number of instances. To solve this problem a smarter way of smoothing probability estimates should be carried out, and several methods will be discussed in the next chapter.

2.2 Bagging trees

In practice, scientific experiments are usually being repeated using the same or a similar set up several times. Scientists then calculate the averaged outcome for a more accurate result. To improve predictive performance in machine learning there are some similar techniques, such as bagging (Breiman, 1996) and boosting (Freund & Schapire, 1996).

Bagging is the technique used in this thesis. As it resamples instances with replacement to construct trees, it throws some of the old instances out and replaces them with duplicate instances, so that the trees are built based on variants of the same dataset. The algorithm for bagging is given in Figure 2.4

For example, in bagging with 10 iterations, in each iteration a tree is generated, and the data used are generated by sampling with replacement from the full dataset. Each resampled dataset contains the same number of instances as the full dataset.

```
Let n be the number of instances in the training data.
For each of iterations:
  Sample n instance with replacement from training data.
  Build classifier from subsample.
```

Figure 2.4: Pseudo code for bagging

```
For each of the models generated
  Calculate the class probability estimates using current model
Return the average of all probability estimates
```

Figure 2.5: Pseudo code for prediction with bagging

At prediction time, for a test case with an unknown class value, the instance actually go traverses all the 10 PETs made from the previous step to get probability estimates from each tree. Then the class probabilities predicted by the 10 trees are averaged to form an overall class probability estimate. This is the standard process when probability estimates are calculated using bagging. The pseudo code for prediction in bagged trees is in Figure 2.5.

2.3 Text compression and PPM

Text compression is another field of study in computer science, but it shares many characteristics with machine learning. Its performance is usually measured in terms of compression ratio, memory and time used in the process.

2.3.1 PPM

PPM (Cleary & Witten, 1984) stands for Prediction by Partial Matching. It is an adaptive data compression technique based on context modeling and prediction. A technique called blending is used in PPM for solving the zero-frequency problem. As discussed above the zero-frequency problem occurs when an event has not been encountered before: an estimate of probability based on relative frequency will then be zero. This makes predictive compression impossible. PPM combines several predictions into a single over all probability to solve the zero-frequency problem.

In PPM, blending is the name of the type technique known in tree learners as smoothing. The Laplace correction is a simple way of blending an estimated distribution with the uniform distribution. Several different blending methods are used in PPM (for

#	Dataset	Size
1	Monks1	556
2	Monks2	601
3	Monks3	554
4	Tic Tac Toe	958
5	Mushroom	8124
6	Wisconsin Breast Cancer	669
7	Kr vs Kp	3196
8	Sonar	208
9	Pima Diabetes	768
10	Vote	435
11	Yeast	1484
12	Hepatitis	155
13	Liver disorders	345
14	Spambase	4601
15	Ionosphere	351
16	Sick	3772
17	Spect	267

Table 2.1: Datasets used in experiments

example, PPMA, PPMB). The only difference between these methods is the way of calculating *escape probabilities*. In PPM, the blended (i.e. smoothed) probabilities are calculated as follows:

$$p(\phi) = \sum_{o=-1}^{m} w_o p_o(\phi) \tag{2.1}$$

where o denotes the order of the context (the analogue of the depth of a node in a PET), $p_o(\phi)$ is the probability before blending and w_o is the weight, calculated using the following equation:

$$w_o = (1 - e_o) \prod_{i=o+1}^{l} e_i \qquad -1 \leq o < l \tag{2.2}$$

Where e is called escape probability and calculated differently in different methods. These methods will be discussed further later in the thesis. The probability e_o gives the probability of escaping from the order-o context to the order-$(o-1)$ context. In a tree, e_m would be the escape probability associated with a leaf node.

2.4 Datasets

The datasets used in this thesis to test the newly implemented methods are most of the datasets from the paper that introduced M-Branch smoothing (Ferri, Flach & Hernández-Orallo, 2003). They are shown in Table 2.1.

The first three datasets are the monk's problems. They all have the same attributes, but one of them has noise added in. They have 8 attributes including the class value.

Tic tac toe is a datasets that contains a number of different tic tac toe game boards and the outcome of the game as the class value. It has 10 attributes. The first 9 represent the 9 game board spaces and the tenth is the game result.

The mushroom dataset is a relatively bigger dataset considering both the number of instances and attributes. It has 23 attributes including the class value. The first 22 attributes describe the different appearances of the mushrooms and the class value defines whether it is edible or poisonous.

The Wisconsin breast cancer dataset contains 10 attributes including the class value. It is a purely numeric dataset. The first 9 attributes are the symptoms and the class attribute tells us whether the cancer is benign or malignant.

Kr vs Kp is a chess game dataset. It contains 36 attributes including the class attribute. Each instance contains the setup of a chessboard, where white has a king and a rook and black has a king and a pawn. The class attribute shows if white can win at the end of the game.

Sonar has 61 attributes including the class value. The first 60 attributes are readings from the sonar, and the class value determines whether the signal the instance corresponds to is a mine or a rock.

Pima diabetes has 9 attributes including the class value. The first 8 attributes are different properties of a patient and the class value is positive or negative.

Vote is the 1984 United Sates Congressional voting record. It has 17 attributes including a class value, for democrat or republican respectively.

Yeast has 9 attributes. It is a dataset about proteins. The first 8 attributes are different characteristics of the protein, and the class value shows the sequence name of that protein.

Hepatitis has 20 attributes including the class value, indicating whether the patient is still alive. Other attributes are symptoms and characteristics of the patient.

Liver disorder has 7 attributes. The first 5 attributes are aspects of blood tests that are related to liver disorders. The 6th attribute is the amount of alcohol consumed by the patient per day. The last attribute is the class value, which separates the data into two sets.

Spambase is a spam e-mail database. It has 58 attributes and the class value indicates whether the corresponding message is spam or not.

The ionosphere dataset has 35 attributes including a class value of good or bad. This is a radar dataset collected by the system in Goose Bay, Labrador. The first 34 attributes, they represent 17 pulse numbers, with two attributes per pulse.

The sick dataset contains 30 attributes. The first 29 represent different characteristics, symptoms and the medical history of the patient, such as age, sex and pregnancy. The last one is the class value which shows if the patient is sick or not.

The spect dataset contains 23 attributes. The first 22 attributes are diagnostic features of cardiac SPECT images. The class value shows the patient is normal or abnormal.

All of the above datasets have two classes. They were chosen because the tree learner used in this thesis behaves very similarly with binary-class datasets to the one used in (Ferri, Flach & Hernández-Orallo, 2003). The aim was to make the end result more comparable.

2.5 Evaluation methods

To measure the performance of the smoothing methods, several different evaluation methods are considered to make sure all aspects of the methods are evaluated. The following evaluation measures are used:

- Root mean squared error
- Area under ROC curve

- Entropy gain

2.5.1 Root mean squared error

Mean squared error or MSE is the average squared error between the predicted value and the actual value. Root mean squared error or RMSE is the squareroot of MSE. Root mean squared error is used often to measure precision of estimates. The squared error is also called the quadratic loss (Witten & Frank, 2005).

$$MSE = \frac{1}{n}\sum_{j=1}^{n}\frac{1}{k}\sum_{i=1}^{k}(p_{ji} - a_{ji})^2 \qquad \textit{Mean squared error} \tag{2.3}$$

Here, there are n test instances and k class values. The p_{ji} are the predicted class probabilities and the a_{ji} are the observed values (either 0 or 1). Taking the square root yields the root mean squared error.

$$RMSE = \sqrt{MSE} \qquad \textit{Root mean squared error} \tag{2.4}$$

2.5.2 Area under ROC curve

ROC stands for receiver operating characteristic. It represents the performance of a classifier without explicit error costs (Witten & Frank, 2005). The horizontal axis is the false positives rate and the vertical axis is the true positives rate, assuming binary classification. The area under the curve (AUC) (Ling, Huang & Zhang, 2003) is the performance measure used in this thesis. For multi-class probability estimation, the AUC value is first calculated for each class in turn, by considering all other classes as the "negative" class, and then the different AUC values are averaged, using weights based on the relative popularity of each class.

To generate an ROC curve from probability estimates for the positive class, the actual class values in the test data are needed. First, the probability estimates for the test instances are sorted in descending order, thus the highest probability comes first. In other words, the earlier the prediction is in the list, the more likely for it to be positive according to the classifier. Then we use the actual class value to find if the predicted probability is accurate. We start from the origin in the plot, and the top of our ranked list, and go up if the predicted value is true, and go right if the predicted value is false. After all the predictions are drawn on the coordinate system, a jagged line is formed. That is the ROC curve.

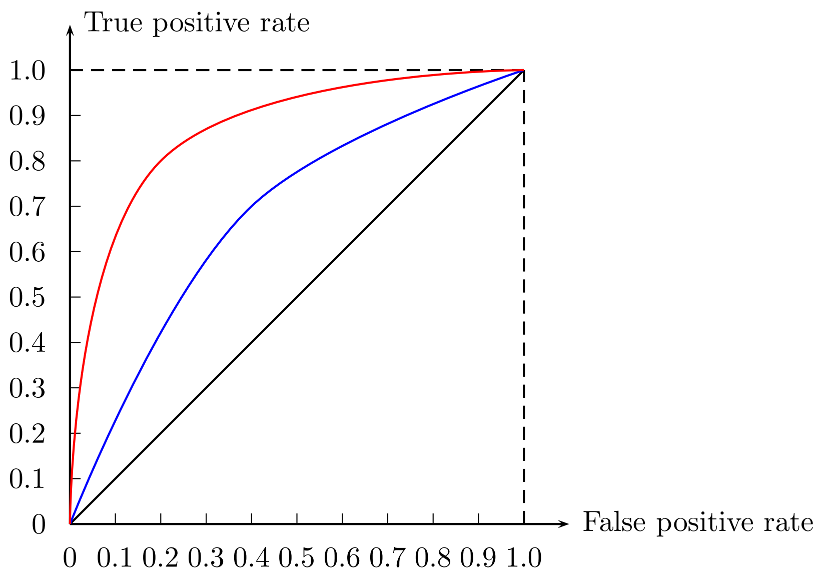

Figure 2.6: Example ROC curves

Figure 2.5.2 shows three ROC curves, which correspond to three different hypothetical classifiers. The red curve has the largest area under the curve. The blue curve has a larger area under it than the black line, but the area under it is smaller than the area under the red curve. The black line has a perfect 50% area under it, which is equivalent to a random prediction in a two class dataset.

2.5.3 Entropy gain

Entropy gain is a measure closely related to compression performance. The so-called informational loss for an event with estimated probability p is -$\log_2 p$ (Witten & Frank, 2005). If p is 1 for every predicted value, the model makes perfect predictions and no bits are needed to correct its mistakes. The number of bits needed is given by the informational loss. In Weka, entropy gain is calculated as

$$ENTROPY\_GAIN = \sum_{j=1}^{n} - \log P_i - \sum_{j=1}^{n} - \log p_i \tag{2.5}$$

Where P_i is the "default" prediction for the class value of test instance i and p_i is the model's prediction. The default prediction is an estimate of the prior probability of each class (e.g. 1/3 in the case of the iris data where each type of iris flower is equally likely).

2.6 Summary

This chapter discussed the background knowledge relevant for this thesis. The tree classifier that is used was introduced first, then the algorithm for constructing a PET was presented and explained with pseudo code. Then some further algorithmic techniques used in this thesis were discussed, such as pruning and bagging. Thirdly, smoothing was introduced, followed by the Laplace correction as a simple existing PET smoothing method. Then the PPM text compression method was introduced, followed by the different smoothing methods used by PPM, its equations, and the escape probability calculation method. The dataset used in the experiments in the thesis were listed and described in detail next. Finally, the evaluation criteria where discussed: area under ROC curve, root mean squared error and entropy gain.

Chapter 3
Smoothing methods

In this chapter, the different smoothing methods evaluated using PETs are discussed in detail. The first two smoothing methods are the Laplace correction and the M-Estimate smoothing (Dzeroski, Cestnik & Petrovski, 1993). They are very simple methods that are included for completeness. From the third method on, the methods are discussed in more detail. All these methods will be experimented on and analysed in detail in the next chapter.

3.1 Existing smoothing methods in probability estimation trees

There are many smoothing techniques available nowadays. To choose a method that will fit tree learning well therefore requires detailed experiments and analysis. We now discuss the basic methods that will be evaluated later.

3.1.1 Laplace Correction and M-Estimate

The Laplace correction, the strategy of adding one to each count, will eliminate a 0 count for a predicted outcome. It is the most basic technique for smoothing probability estimates and widely used in practice.

The M-Estimate (Ferri, Flach & Hernández-Orallo, 2003) is another commonly used smoothing technique, where M is a constant defined by the user. The Laplace correction and the M-Estimate are defined as follows:

$$p_i = \frac{n_i + 1}{\left(\sum_{i \in C} n_i\right) + c} \qquad \textit{Laplace Correction} \tag{3.1}$$

$$p_i = \frac{n_i + p \cdot m}{\left(\sum_{i \in C} n_i\right) + c} \qquad M - Estimate \tag{3.2}$$

Here n_i is the count of instances for class i from the leaf node concerned, c is the number of classes. The M-Estimate, if $p = 1/m$, becomes the formula shown in Equation 3.1, which is Laplace correction. In the experiments with PETs in (Ferri, Flach & Hernández-Orallo, 2003), $M = 4$ is used, because it is the best experimental value.

3.1.2 M-Branch Smoothing

M-Branch smoothing is introduced as a new method in (Ferri, Flach & Hernández-Orallo, 2003). It is more complicated than the Laplace correction and the M-Estimate because it combines multiple probability estimates. It is defined as follows:

$$p_i^j = \frac{n_i^j + m \cdot p_i^{j-1}}{\left(\sum_{i \in C} n_i^j\right) + m} \qquad M - Branch\ Smoothing \tag{3.3}$$

Here m is not just a constant. It varies according to the formula below:

$$m = M \cdot (1 + (1 - 1/h) \cdot \sqrt{N}) \tag{3.4}$$

where M here is a again a user-specified constant as in the M-Estimate, and N is the global cardinality of the dataset. The value h is the height of the node in the tree.

According to Equation 3.3, the smoothing method is no longer applied to just the leaf nodes as in the Laplace correction and in M-Estimate smoothing. It smoothes the probability along the path of prediction in a PET based on the probability estimates p_i^j for each node j. According to Equation 3.4, m is calculated based on h, the height of the current node. By observing the equation, it is not hard to find out that if the node is higher up in the tree, it is smoothed more heavily, where the normalised height of a node is $1-1/h$.

The probability of each outcome is fixed at $1/c$ above the root node, c being the number of possible prediction outcomes. Then we pass that probability to the root node of the PET, calculating the smoothed probability estimate until we reach the leaf node, where the normalized height of the node is 0, thus $m = M$.

For example in Figure 1.1, we can take the path from the root node to the versicolor node The relevant calculations are shown in Table 3.1.

Height	Node	Counts	Unsmoothed probability estimates			Normalized Height	
		N	Setosa	Versicolor	Virginica	Δ	m
-1	-	-	$\frac{1}{3}$ -	$\frac{1}{3}$ -	$\frac{1}{3}$ -	-	-
3	1	150	$\frac{50}{150}$ 50	$\frac{50}{150}$ 54	$\frac{50}{150}$ 50	$\frac{2}{3}$	36.66
2	3	150	0 0	$\frac{54}{100}$ 54	$\frac{50}{100}$ 50	$\frac{1}{2}$	28.50
1	4	150	0 0	$\frac{49}{54}$ 54	$\frac{5}{54}$ 5	0	4.00
Smoothed probabilities			0.00510	0.87676	0.11814		

Table 3.1: M-Branch smoothing effects along a path of prediction

3.2 PPM and escape probabilities

In PPM, blended probabilities are calculated using the equation mentioned in the previous chapter in Equation 2.1. In this equation, there is the variable w_o, which is the weight of each context In PETs, the context is provided by the path to the current node. The weight is calculated using Equation 2.2, where e is the escape probability.

The escape probability can be viewed as the probability of "escaping" from the current node to the parent node to determine the prediction for a new test instance, based on the probability that a previously unseen class value is encountered. There are a variety of ways of calculating the escape probability. They are named by combining PPM with the probability method code. PPMA is the PPM method using the escape probability calculation method A.

It is very hard to say which escape probability calculation method is the best of all, or to even make this statement about just two of them, but in practice there is one that is the most suitable one for the problem at hand. The more escape calculation methods there are, the more options we have when we have a problem. In this thesis, 5 of the most popular PPM escape methods are implemented and experimented on.

3.2.1 PPMA

The first escape probability calculation method is method A (Cleary & Witten, 1984). It is a simple calculation:

$$e_o = \frac{1}{C_o + 1} \tag{3.5}$$

In the text compression case, the number of characters we have seen before in the current context o is given by C_o. In PETs this will be the total count of instances

Height	Node	Counts	Unsmoothed probability estimates						Weight	Escape
		C_o	Setosa		Versicolor		Virginica		w_o	e_o
-1	-	-	$\frac{1}{3}$	-	$\frac{1}{3}$	-	$\frac{1}{3}$	-	1.1×10^{-6}	0
3	1	150	$\frac{50}{150}$	50	$\frac{50}{150}$	50	$\frac{50}{150}$	50	0.00018	0.0066
2	3	100	0	0	$\frac{50}{100}$	50	$\frac{50}{100}$	50	0.0180	0.0099
1	4	54	0	0	$\frac{49}{54}$	49	$\frac{5}{54}$	5	0.9818	0.0182
Smoothed probabilities			0.00006		0.89995		0.09997			

Table 3.2: PPMA smoothing effects along a path of prediction

at the node concerned. The probability of seeing a new type of character or a new class value respectively is 1 over the "new" total number of characters or instance count respectively. As C_o grows bigger, the escape probability becomes smaller and smaller. In text compression, after a number of characters have appeared in a context, the probability of an entirely new character showing up in upcoming text is very low. Similarly, in PETs, after the model has been built with a fair number of instances, the probability of seeing an entirely new class value of a particular node decreases.

If the same path used in Table 3.1 is smoothed with PPMA the result is as it is shown in Table 3.2

3.2.2 PPMB

The second escape probability calculation method is method B (Cleary & Witten, 1984). It uses a different approach to calculates the escape probability:

$$e_o = \frac{q_o}{C_o} \tag{3.6}$$

Here, q_o is the number of different characters that have occurred in the corresponding context context of order o.

In this approach, the number of different characters that have been seen is taken into account in the calculation of the escape probability and it is proportional to the escape probability. That means if fewer different character have been seen before, it is less likely to see a new character in the future.

This technique effectively only takes an observation into account after it has oc-

Height	Node	Counts	Unsmoothed probability estimates						Weight	Escape
		C_o	Setosa		Versicolor		Virginica		w_o	e_o
-1	-	-	$\frac{1}{3}$	-	$\frac{1}{3}$	-	$\frac{1}{3}$	-	1.48×10^{-5}	0
3	1	150	$\frac{50}{150}$	50	$\frac{50}{150}$	50	$\frac{50}{150}$	50	0.00073	0.02
2	3	100	0	0	$\frac{50}{100}$	50	$\frac{50}{100}$	50	0.0363	0.02
1	4	54	0	0	$\frac{49}{54}$	49	$\frac{5}{54}$	5	0.9630	0.037
Smoothed probabilities			0.00025		0.89223		0.10756			

Table 3.3: PPMB smoothing effects along a path of prediction

curred twice. It is inspired by the consideration that a one-off event may be an error.

In PETs, if fewer classes values have been seen in a node, that means the probability of a new class occurring at this node is low. Vice versa, if a node is relatively less pure, it is more likely that we will see more different classes occurring at this node.

The same path used in Table 3.1 is smoothed with PPMB and the result is shown in Table 3.3

3.2.3 PPMC

The third method is method C (Moffat, 1988). It is similar to method B:

$$e_o = \frac{q_o}{C_o + q_o} \tag{3.7}$$

This approach is based on the observation that PPMB effectively only takes an observation into account when it has already occurred twice, which seems wasteful. On the other hand, escape probability method A can be problematic, if a context occurs frequently, but with different characters. As a compromise between PPMA and PPMB, PPMC, a hybrid, was introduced. It gets the advantages of both method A and B. Looking at the equation, the only difference is that the denominator has an added value q_o. This will lower the escape probability of the entire context.

In PETs, PPMB has a similar disadvantage: if a node has a small number of instances, adding one to the class count will affect the probability estimates dramatically.

Height	Node	Counts	Unsmoothed probability estimates						Weight	Escape
		C_o	Setosa		Versicolor		Virginica		w_o	e_o
-1	-	-	$\frac{1}{3}$	-	$\frac{1}{3}$	-	$\frac{1}{3}$	-	1.37×10^{-5}	0
3	1	150	$\frac{50}{150}$	50	$\frac{50}{150}$	50	$\frac{50}{150}$	50	0.00069	0.0196
2	3	100	0	0	$\frac{50}{100}$	50	$\frac{50}{100}$	50	0.035	0.0196
1	4	54	0	0	$\frac{49}{54}$	49	$\frac{5}{54}$	5	0.9643	0.0357
Smoothed probabilities			0.00023		0.89275		0.10702			

Table 3.4: PPMC smoothing effects along a path of prediction

By using PPMC, we lower the estimated probability of a new a class occurring at the current node.

The same path used in Table 3.1 is smoothed with PPMC and the result is shown in Table 3.4

3.2.4 PPMD

The fourth method is method D (Howard, 1993). It is similar to method B too:

$$e_o = \frac{q_o/2}{C_o} \tag{3.8}$$

As in PPMB, if a new symbol occurs for the first time, 1 will be added to q_o, as the number of different characters that have occurred. However, in PPMD, when a new character occurs for the first time, 1/2 is added instead of 1 to the numerator of the escape probability. Thus, as in PPMC, a context will be used even if all observed characters are different.

The same path used in Table 3.1 is smoothed with PPMD and the result is shown in Table 3.5

Height	Node	Counts	Unsmoothed probability estimates						Weight	Escape
		C_o	Setosa		Versicolor		Virginica		w_o	e_o
-1	-	-	$\frac{1}{3}$	-	$\frac{1}{3}$	-	$\frac{1}{3}$	-	1.85×10^{-6}	0
3	1	150	$\frac{50}{150}$	50	$\frac{50}{150}$	50	$\frac{50}{150}$	50	1.83×10^{-4}	0.01
2	3	100	0	0	$\frac{50}{100}$	50	$\frac{50}{100}$	50	0.01831	0.01
1	4	54	0	0	$\frac{49}{54}$	49	$\frac{5}{54}$	5	0.9815	0.0185
Smoothed probabilities			0.00006		0.89984		0.10010			

Table 3.5: PPMD smoothing effects along a path of prediction

3.2.5 PPMP

The fifth method is method P (Witten & Bell, 1991). It is pretty unique compared to the other escape probability calculation methods. The formula for PPMP is as follows:

$$e_o = \frac{n_1}{C_o} - \frac{n_2}{C_o^2} + \cdots \tag{3.9}$$

As the equation shows, there are different fractions connect by a plus sign or a minus sign, and the plus or minus sign take turns. The numerator of each fraction is the number of classes that has occurred a certain number of times: for example, n_1 is the number of classes that occurred has 1 time in the current node.

Note that here n_1 can be 0, but a probability cannot be a negative number. This problem will be discussed in Section 3.4. When $n = C_o$, and it is the only fraction in the equation, the probability is 1, which should be avoided. However this problem does not happen in this thesis, because it does not occur in the datasets considered.

Differently from all the escape probability calculation methods above, the method is based on an open equation. The number of fractions of the equation is not fixed. However, as the denominater grows exponentially, the later fractions reduce exponentially, and their combination will become insignificant after the 9th or 10th fraction, depending on the number of instances in the dataset. In this thesis, 5 was chosen to be the number of fractions used in the equation because even the smallest dataset, hepatitis, has 155 instances. The denominator will be 89466096875 for the 5th fraction, the result of that fraction is a very small number.

	Setosa	Versicolor	Virginica
PPMA	0.00006	0.89995	0.09997
PPMB	0.00025	0.89223	0.10756
PPMC	0.00023	0.89275	0.10702
PPMD	0.00006	0.89984	0.10010

Table 3.6: Smoothing effects compared

```
For each item $d$ in the array of class counts
  $d = d + 1$
```

Figure 3.1: Pseudo code for Laplace correction in REPTree

3.3 Smoothing effects

Several PPM-based smoothing methods and smoothed probability estimates are have been discussed above. All the smoothed probability estimates are shown in Table 3.6 for comparison. By observing the table, notice that the probability estimates of different the methods are very similar. That is because of the characteristics of the iris data, which yields a large and almost pure leaf node..

From the little difference of the probability estimates, we can see that PPMA and PPMD has smaller smoothing effects, which correctly reflects the smoothing effect of Equation 3.8.

3.4 Implementing the smoothing methods

In order to test all the methods described above, the methods have to be implemented into a suitable tree learner. Firstly we have to find a tree learner that is similar to the one used in the literature (Ferri, Flach & Hernández-Orallo, 2003). The most suitable tree learner in Weka has been picked out, namely REPTree, a fast decision tree learner that uses information gain.

The simple Laplace correction smoothing has been implemented into REPTree first. The pseudo code is shown in Figure 3.1

Similar to the Laplace correction, the M-Estimate has been implemented in REP-

```
For each item $d$ in the array of class counts
  $d = d + (m \cdot \frac{1}{Number\ of\ classes})$
```

Figure 3.2: Pseudo code for M-Estimate in REPTree

```
if leafNode
  return 1
else
  if the attribute is missing
        Loop through each branch and calculate the height;
        then combine based on split proportions
        height = combind height
  else if nominal attribute
        height = height of the appropriate child node
        height++
  else if attribute is greater than split point
        //For numeric attributes
        height = height of the first child node
        height++
  else
        height = height of the second child node
        height++

  return height
```

Figure 3.3: Pseudo code for height calculation in REPTree

Tree. The code is very similar. The only difference is the different addition made to the original class counts. In both cases estimated probabilities are then calculated by normalizing the array of class counts. The pseudo code for M-Estimate smoothing is shown in Figure 3.2.

3.4.1 M-Branch Smoothing

Implementing M-Branch smoothing is different from implementing the Laplace correction and the M-Estimate. As it uses the height of the node, the code has to be in the prediction portion of the code instead of the PET building portion of the code. For calculating the height of the node, a simple method has been implemented to do it recursively along the prediction path. This code is shown in Figure 3.4. The "expected" height is used when the instance has a missing value for the node concerned.

The actual code for smoothing the probabilities is in the *distributionForInstance()* method, which is called at prediction time. As mentioned above, it is quite different

```
Get height for current node
Let N be total number of instances for current node
p = smoothed probabilities from parent node
calculate normalized height $\Delta$ = 1 - ( 1 / height )
m = M · (1 + $\Delta$ · $\sqrt{numInstances}$)
foreach raw count $c_i$ at node
smoothed probability = ($c_i$ + m · $p_i$) / (N + m)
```

Figure 3.4: Pseudo code for M-Branch smoothing in REPTree

```
Let weight = 0
Let N be total number of instances for current node
get escape probability e for current node
weight = (1 - e) · product of escape probabilities from path below
foreach raw count $c_i$ at node
smoothed probability += ($c_i$ / N) * weight
```

Figure 3.5: Pseudo code for PPM in REPTree

from the two methods above, because in M-Branch smoothing, the height of the current node is needed, and multiple probability estimates are combined. The pseudo code for M-Branch smoothing is in Figure 3.4

3.4.2 PPM Methods

The PPM methods are placed in the same location as M-Branch smoothing as they need similar information, and are also applied recursively. The difference between different PPM methods is in the escape probability calculations, thus there is only one piece of code applying the smoothing to the probability estimates. The pseudo code for this is in Figure 3.5.

The method for returning the product of escape probabilities in Figure 3.5, needed for the calculation in Equation 2.2, is a similar procedure as the get height method for M-Branch smoothing. The pseudo code is in Figure 3.6. Note that it need to be called on the appropriate child node in line 4 of the algorithms in Figure 3.5.

For different escape probability calculation methods, a separate function has been created to detect which escape probability method has been chosen by the user and calculate the escape probability required. The pseudo code for getting the escape probability is in Figure 3.7

```
Let product = 0
if leafNode
  return 1
else
  if the attribute is missing
        Loop through each branch and calculate the product;
        combine products based on split proportions
        product = combined product
  else if nominal attribute
        product = calculate product for child node
        e = escape probability for current node
        product = product * e
  else if attribute is greater than split point
        //For numeric attributes
        product = calculate product for the first child node
        e = escape probability for current node
        product = product * e
  else
        product = calculate product for the second child node
        e = escape probability for current node
        product = product * e

  return product
```

Figure 3.6: Pseudo code for getProduct in REPTree

3.5 Summary

This chapter discussed two simple smoothing methods of PETs first, the Laplace correction and M-Estimate smoothing. These methods are used as baseline methods to test the behaviour of the PET we used. Then the smoothing method from the paper by Ferri, Flach & Hernández-orallo (2003) was introduced and explained with the aid of its probability estimate calculation equation. A simple smoothing example with the iris dataset was presented in a tabular form.

Then, each of the PPM smoothing methods were introduced. The difference between the PPM methods is the escape probability calculation method. With the help of equations for the escape probability, the motivation for each escape probability calculation method was discussed. PPMA simply gives the newly occurred character, in the case of PETs, a new instance, one increment on the total count, which is simple enough to understand: after quite a few number characters have been encountered, the probability that a new character shows up is quite low. PPMB takes a different approach, the character probability calculation is inspired by the rule that observations are counted only

```
Let Escape Probability e = 0
Let N be the total count of instances occuring at current node
Let q be the number of classes occuring at current node
if method A
  e = 1 / (1 + N)
else if method B
  if (N = 0 or q = 0)
    return 1
  e = q / N
else if method C
  if (N = 0)
    return 1
  e = q / (N + q)
else if method D
  if (N = 0 or q = 0)
    return 1
  e = (q / 2) / N
else if method P
  if (N = 0 or q = 0)
    return 1
  if no class occured 1 time in the current node
    Calculate e as method C
  for i from 1 to 5
    get number of classes occured i times n
    e -= n / $N^i \cdot -1^i$

return e
```

Figure 3.7: Pseudo code for getEscProb method in REPTree

if they have happened twice. Thus the escape probability also uses the number of different characters seen so far. PPMC is a compromise between PPMA and PPMB, which gets the advantage of both methods. PPMD has the least smoothing effect on probability estimates, as it only adds half a count to the number of seen characters of the escape probability, classes in the case of PETs, when a new character occurs. PPMP has the most special approach amongst all the PPM methods considered. The number of classes that have occurred exactly n times are involved in the calculation of escape probabilities, n being a natural number which increments by 1 each time. As the denominator of the fractions involved, if the number of classes is 0 for the first fraction, the end result of the calculation will be negative. As probabilities cannot be negative, the escape probability calculation falls back to method C in this case.

Lastly, the pseudo code of all the smoothing methods discussed in this chapter was presented and explained, including the helper methods: getHeight for M-Branch smoothing, getProduct for calculating the escape probabilities, and getEscProb to get the correct escape probability calculation method based on user input.

Chapter 4
Experiments

This chapter evaluates the different smoothing methods on the datasets discussed in Chapter 2. The methodology and experiment setups are discussed in Section 4.1. Section 4.2 evaluates results from the first two setups, using unpruned REPTree without modifications and with the Laplace correction and M-Estimate smoothing. It compares the unsmoothed probability estimates with those smoothed by the Laplace correction and the M-Estimate respectively. Section 4.3 evaluates M-Branch smoothing with comparison to the M-Estimate. Section 4.4 discusses the newly implemented PPM methods with M-Branch smoothing. Bagging unsmoothed trees is discussed in Section 4.5 The effects of bagging trees with the basic smoothing methods are discussed in Section 4.6. Section 4.7 compares the effect of smoothing bagged trees with M-Branch smoothing to bagged unpruned trees. Section 4.8 considers bagged trees with PPM smoothing. Sections 4.9 to 4.12 consider pruned trees. Section 4.13 presents results for a specific case study based on an e-commerce dataset.

4.1 Methodology and Experiment Setups

For ease of comparing the performance of all the smoothing methods implemented in REPTree in Weka (Witten & Frank, 2005), 17 datasets were selected to be used in the experiments. They were listed in Table 2.1. Some of these dataset contain solely numeric attributes or nominal attributes, some contain both nominal attributes and numeric ones, and some contain missing values.

In practice, one dataset will be divided into two subsets, one subset is used to build the PET, and the other one is used to test the PET built from the training subset. The split percentage can be chosen by the experimenter. For example, we can divide the dataset into disjoint 3 parts, where the dataset is divided randomly. Each of the 3 subsets take turns to test the PET built based on the other two subsets. Then the 3 error estimates are averaged to yield an overall error estimate. This is called a

3-fold cross-validation. The number of subsets we used to divide the datasets up is the number of folds. Sometimes a single cross-validation might not be reliable. Different cross-validations can be performed using the same algorithm, and we can calculate the average of the error estimates. Usually the number of folds in a cross-validation is set to 10, which is found to be the best number to generate an accurate error estimate (Kohavi, 1995). However, in this thesis, a 20 times 5-fold cross-validation is used, because it is the setup used in the paper by Ferri, Flach & Hernández-orallo (2003). This way the experimental results are more comparable. The experiments use the paired corrected t-tester (Nadeau & Bengio, 2001) for significance testing with a significance level of 0.05.

4.2 Smoothing effect of M-Estimate smoothing and the Laplace correction

We first compare unpruned trees to those smoothed with the Laplace correction and the M-Estimate respectively.

4.2.1 Area under ROC curve

Table 4.1 shows the AUC value from the experiments for REPTree probability estimates with no smoothing versus M-Estimates and Laplace correction. The second column is used as the test base, thus both the third and the fourth columns are compared with the second column. $\bullet$ in the third column indicates that M-Estimate has significantly improved AUC over the same classifier with no smoothing at all.

In this table, the numbers after the $\pm$ are standard deviations. By observing the table, we can see that 10 out of 17 datasets have significant improvements after smoothing with M-Estimate. In face, all the datasets have some improvement with M-Estimate smoothing. In contrast, the Laplace correction has no significant differences compared to M-Estimate smoothing. However, comparing M-Estimate with Laplace correction more closely, it has larger AUC estimates for 6 out of 17 datasets.

4.2.2 Root mean squared error

Table 4.2 shows the root mean squared error value from the same experiments as in Table 4.1. Again, the second column is used as the test base, both the third and the fourth columns are compared with the second column. The $\circ$ symbol in the third column indicates M-Estimate smoothing has a significantly smaller root mean squared error

Dataset	M-Estimate	No Smoothing		Laplace Correction
monks-1	0.979± 0.03	0.974±0.03		0.980±0.03
monks-2	0.604± 0.06	0.568±0.06	●	0.602±0.06
monks-3	0.991± 0.01	0.990±0.01		0.991±0.01
tic-tac-toe	0.930± 0.02	0.879±0.03	●	0.930±0.02
mushroom	1.000± 0.00	1.000±0.00		1.000±0.00
breast-cancer	0.977± 0.01	0.952±0.03	●	0.977±0.01
kr-vs-kp	0.999± 0.00	0.997±0.00		0.999±0.00
sonar	0.803± 0.07	0.751±0.08	●	0.801±0.07
pima-diabetes	0.770± 0.04	0.684±0.05	●	0.766±0.04
vote	0.985± 0.01	0.984±0.01		0.986±0.01
yeast	0.725± 0.03	0.690±0.03	●	0.727±0.03
hepatitis	0.748± 0.09	0.717±0.11		0.747±0.10
liver-disorders	0.681± 0.07	0.655±0.07		0.680±0.07
spambase	0.968± 0.01	0.932±0.01	●	0.967±0.01
ionosphere	0.934± 0.03	0.892±0.04	●	0.934±0.03
sick	0.990± 0.01	0.965±0.02	●	0.990±0.01
spect	0.718± 0.06	0.678±0.07	●	0.711±0.06

○, ● statistically significant difference

Table 4.1: Area under ROC curve

compared to the same classifier with no smoothing. The ● symbol in the third column indicates a significantly bigger root mean squared error. Similarly, the fourth column has the same indicators. In Table 4.2, the Laplace correction improves the probability estimates from no smoothing, and M-Estimate has further improvements from Laplace correction. The raw results in the table also show improvements in most datasets. 11 out of 17 datasets have significant improvements for M-Estimate smoothing over no smoothing and 9 out of 17 datasets have significant improvements for M-Estimates over Laplace correction.

By observing both Table 4.1 and Table 4.2, it is not hard to notice that datasets which do not yield any improvements are similar. These datasets are *monks problems 1*, *monks problems 3*, *mushroom*, *kr vs kp* and *vote*. In these 5 datasets, *mushroom* and *kr vs kp* have perfect ROC curves, 100% for *mushroom* and 99.7% for *kr vs kp*, which leaves very little potential for improvements. In Table 4.2 the root mean squared error for *mushroom* has increased from 0 to 0.018. This is a typical case of over smoothing.

4.2.3 Entropy gain

Table 4.3 lists the entropy gain for M-Estimate smoothing compared with no smoothing and the Laplace correction, with M-Estimates as the test base. The ● symbol in the

Dataset	M-Estimate	No Smoothing		Laplace correction	
monks-1	0.224± 0.07	0.141±0.12	●	0.188±0.08	●
monks-2	0.482± 0.02	0.586±0.04	○	0.500±0.02	○
monks-3	0.163± 0.02	0.122±0.04	●	0.141±0.02	●
tic-tac-toe	0.323± 0.02	0.351±0.03	○	0.319±0.02	
mushroom	0.018± 0.00	0.000±0.00	●	0.010±0.00	●
breast-cancer	0.211± 0.03	0.227±0.04	○	0.214±0.04	
kr-vs-kp	0.084± 0.01	0.073±0.02	●	0.077±0.02	●
sonar	0.441± 0.06	0.490±0.07	○	0.456±0.06	○
pima-diabetes	0.452± 0.03	0.518±0.03	○	0.471±0.03	○
vote	0.220± 0.02	0.191±0.03	●	0.203±0.03	●
yeast	0.254± 0.01	0.280±0.01	○	0.256±0.00	○
hepatitis	0.389± 0.05	0.419±0.06	○	0.396±0.05	○
liver-disorders	0.491± 0.04	0.559±0.05	○	0.510±0.04	○
spambase	0.244± 0.01	0.263±0.01	○	0.248±0.01	○
ionosphere	0.288± 0.04	0.311±0.05	○	0.294±0.04	○
sick	0.096± 0.01	0.096±0.02		0.095±0.01	
spect	0.463± 0.03	0.516±0.04	○	0.475±0.03	○

○, ● statistically significant difference

Table 4.2: Root mean squared error

third column indicates M-Estimate smoothing has a significantly larger entropy gain over no smoothing. On the other hand, ○ indicates significantly smaller entropy gain for M-Estimate smoothing. The fourth column has the same indicators for Laplace correction. Table 4.3 also confirms the fact that M-Estimate has the best performance among the three methods: except for *mushroom* all the datasets exhibit improvements with M-Estimate smoothing. Moreover, 14 out of 17 datasets have significant improvements for M-Estimate smoothing compared to no smoothing. Laplace correction has better performance compared to no smoothing at all: 16 out of 17 datasets have improvements with Laplace correction compared to unsmoothed probability estimates, but for 9 out of 17 datasets we prefer M-Estimate smoothing over Laplace correction.

Discussion

In the tables of experimental results, a larger AUC and a smaller root mean squared error and a larger entropy gain are indicators of a better smoothing method. Although there are some significantly degraded experimental results in Tables 4.2 and 4.3, but the over-all results for M-Estimate are better than those for the Laplace correction and the probability estimates before smoothing. The results are quite similar to the results in (Ferri, Flach & Hernández-Orallo, 2003), which indicates that the implementation of M-Estimates is correct and that REPTree is similar in behaviour to the classifier used in (Ferri, Flach &

Dataset	M-Estimate	No smoothing	Laplace correction
monks-1	0.68± 0.12	-21.55±27.38	0.75±0.13 ○
monks-2	-0.01± 0.06	-266.96±50.83 ●	-0.08±0.08 ●
monks-3	0.80± 0.02	-9.50± 9.92 ●	0.85±0.03 ○
tic-tac-toe	0.44± 0.05	-88.80±26.04 ●	0.47±0.06 ○
mushroom	0.99± 0.00	1.00± 0.00 ○	0.99±0.00 ○
wisconsin-breast-cancer	0.69± 0.07	-35.92±18.79 ●	0.68±0.09
kr-vs-kp	0.95± 0.01	-1.55± 2.47 ●	0.96±0.01 ○
sonar	0.10± 0.22	-196.77±73.39 ●	-0.02±0.28 ●
pima-diabetes	0.06± 0.09	-216.49±41.15 ●	-0.06±0.12 ●
vote	0.69± 0.05	-6.49± 9.13	0.72±0.05 ○
yeast	0.25± 0.11	-338.23±33.63 ●	0.20±0.09 ●
hepatitis	0.05± 0.14	-49.84±48.98 ●	0.00±0.18 ●
liver-disorders	-0.03± 0.14	-253.37±68.16 ●	-0.16±0.18 ●
spambase	0.65± 0.03	-43.93± 7.82 ●	0.63±0.04 ●
ionosphere	0.51± 0.10	-68.86±34.71 ●	0.48±0.12 ●
sick	0.28± 0.01	-4.14± 2.75 ●	0.28±0.02
spect	0.08± 0.10	-170.25±58.98 ●	0.01±0.13 ●

○, ● statistically significant difference

Table 4.3: Entropy gain

Hernández-Orallo, 2003). This is important for the analysis of the results that follow.

4.3 M-Estimate versus M-Branch

We now compare the best smoothing method so far, M-Estimate smoothing, to the new smoothing method proposed in (Ferri, Flach & Hernández-Orallo, 2003).

4.3.1 Area under ROC curve

As discussed in previous chapters, the M-Branch smoothing technique is more complicated and sophisticated than both the Laplace correction and M-Estimate smoothing. Table 4.4 lists the AUC values for M-Estimate smoothing and M-Branch smoothing with M-Estimate as the test base. The ○ symbol in the third column indicates that M-Branch smoothing has a significantly larger area under the ROC curve. On the other hand ● indicates a significantly smaller AUC for M-Branch smoothing.

Table 4.4 shows that for most datasets, there is no signification difference: only one dataset has a significantly improved AUC. However, if we look at the values more closely, by comparing the values in the second and the third column, there are some difference that can be noticed. *Wisconsin breast cancer*, *pima diabetes*, *vote*, *liver disorder*

Dataset	M-Estimate	M-Branch
monks-1	0.979± 0.03	0.974±0.04
monks-2	0.604± 0.06	0.592±0.06 ●
monks-3	0.991± 0.01	0.990±0.01
tic-tac-toe	0.930± 0.02	0.924±0.02
mushroom	1.000± 0.00	1.000±0.00
breast-cancer	0.977± 0.01	0.978±0.01
kr-vs-kp	0.999± 0.00	0.999±0.00
sonar	0.803± 0.07	0.802±0.07
pima-diabetes	0.770± 0.04	0.779±0.04
vote	0.985± 0.01	0.989±0.01
yeast	0.725± 0.03	0.762±0.03 ○
hepatitis	0.748± 0.09	0.748±0.09
liver-disorders	0.681± 0.07	0.687±0.07
spambase	0.968± 0.01	0.967±0.01
ionosphere	0.934± 0.03	0.931±0.03
sick	0.990± 0.01	0.988±0.01
spect	0.718± 0.06	0.738±0.06

○, ● statistically significant difference

Table 4.4: Area under ROC curve (M-Estimate vs M-Branch)

and *spect* also show some improvement in terms of the estimated AUC, other than the significantly improved dataset *yeast*, which is consistent with the results in (Ferri, Flach & Hernández-Orallo, 2003).

4.3.2 Root mean squared error

Table 4.5 lists the root mean squared error for M-Estimate smoothing and M-Branch smoothing using REPTree, with M-Estimate as the test base. The ● symbol in the third column indicates M-Branch smoothing has a significantly smaller root mean squared error over M-Estimate smoothing.

In Table 4.5, there are three significant improvements that can be observed from the third column, but by looking at the values more closely, almost all the datasets have a small improvement, except for *monks problem 2*, *sonar* and *liver disorder*. Note that in Table 4.4 *monk's problem 2* and *sonar* do not have any improvements in area under the ROC curve for M-Branch smoothing over M-Estimate, so M-Branch smoothing does not seem appropriate for these datasets.

Dataset	M-Estimate	M-Branch
monks-1	0.224± 0.07	0.223±0.07
monks-2	0.482± 0.02	0.486±0.02
monks-3	0.163± 0.02	0.124±0.03 ●
tic-tac-toe	0.323± 0.02	0.323±0.02
mushroom	0.018± 0.00	0.020±0.01
breast-cancer	0.211± 0.03	0.206±0.04
kr-vs-kp	0.084± 0.01	0.083±0.01
sonar	0.441± 0.06	0.443±0.06
pima-diabetes	0.452± 0.03	0.451±0.03
vote	0.220± 0.02	0.177±0.04 ●
yeast	0.254± 0.01	0.247±0.01 ●
hepatitis	0.389± 0.05	0.388±0.05
liver-disorders	0.491± 0.04	0.496±0.04
spambase	0.244± 0.01	0.242±0.01
ionosphere	0.288± 0.04	0.288±0.04
sick	0.096± 0.01	0.096±0.01
spect	0.463± 0.03	0.461±0.03

○, ● statistically significant difference

Table 4.5: Root mean squared error (M-Estimate vs M-Branch)

Dataset	M-Estimate	M-Branch
monks-1	0.68± 0.12	0.69±0.12
monks-2	-0.01± 0.06	-0.02±0.07
monks-3	0.80± 0.02	0.86±0.04 ○
tic-tac-toe	0.44± 0.05	0.45±0.05
mushroom	0.99± 0.00	0.99±0.00 ○
breast-cancer	0.69± 0.07	0.67±0.11
kr-vs-kp	0.95± 0.01	0.95±0.01 ○
sonar	0.10± 0.22	0.00±0.29 ●
pima-diabetes	0.06± 0.09	-0.02±0.13 ●
vote	0.69± 0.05	0.78±0.08 ○
yeast	0.25± 0.11	0.57±0.12 ○
hepatitis	0.05± 0.14	-0.01±0.21
liver-disorders	-0.03± 0.14	-0.11±0.17 ●
spambase	0.65± 0.03	0.62±0.04 ●
ionosphere	0.51± 0.10	0.45±0.15
sick	0.28± 0.01	0.27±0.02
spect	0.08± 0.10	0.04±0.15

○, ● statistically significant difference

Table 4.6: Entropy gain (M-Estimate vs M-Branch)

4.3.3 Entropy gain

Table 4.6 lists the entropy gain for M-Estimate smoothing and M-Branch smoothing, with M-Estimates as the test base. The ∘ symbol in the third column indicates M-Branch smoothing has a significantly larger entropy gain than M-Estimate smoothing. On the other hand, the • symbols indicate significantly smaller entropy gain for M-Branch smoothing.

In Table 4.6, there are 5 significantly improved datasets for M-Branch smoothing and 4 significant degradations for. Note that the three datasets discussed for root mean squared error, *monk's problem 2*, *sonar* and *liver disorder*, also have degradation in entropy gain.

Discussion

The improvement of M-Branch smoothing over M-Estimate smoothing is not as significant as the improvement of M-Estimate smoothing over probability estimates without any smoothing. According to the results in the paper by Ferri, Flach & Hernández-orallo (2003), M-Branch smoothing performs better in multi-class datasets, but in this thesis we focus on binary-class datasets. Nevertheless, M-Branch smoothing has improved the probability estimates for many datasets. It is a better smoothing method compared to M-Estimate smoothing.

4.4 M-Branch smoothing versus PPM smoothing methods

We now consider the new PPM-based smoothing methods for PETs introduced in this thesis and compare them to M-Branch smoothing.

4.4.1 Area under ROC curve

Table 4.7 shows the AUC for M-Branch smoothing compared against all the PPM methods implemented for the thesis. As results in previous chapters have indicated, it is less useful to compare individual PPM methods against each other because of the small differences observed. In this table, all the values are compared with the left-most column, which is M-Branch smoothing. In all the columns except the first column, the ∘ symbols indicate a significant improvement for the area under the ROC curve.

By observing Table 4.7, it is clear to see that almost all PPM methods are worse

Dataset	M-Branch	PPMA	PPMB	PPMC	PPMD	PPMP
monks-1	0.974±0.04	0.972±0.04	0.968±0.04	0.972±0.04	0.974±0.03	0.968±0.04
monks-2	0.592±0.06	0.593±0.06	0.595±0.05	0.590±0.06	0.565±0.06 ●	0.602±0.06
monks-3	0.990±0.01	0.989±0.01	0.989±0.01	0.989±0.01	0.990±0.01	0.989±0.01
tic-tac-toe	0.924±0.02	0.925±0.02	0.923±0.02	0.925±0.02	0.878±0.03 ●	0.924±0.02
mushroom	1.000±0.00	1.000±0.00	1.000±0.00	1.000±0.00	1.000±0.00	1.000±0.00
breast-cancer	0.978±0.01	0.976±0.01	0.977±0.01	0.977±0.01	0.953±0.03 ●	0.977±0.01
kr-vs-kp	0.999±0.00	0.999±0.00	0.999±0.00	0.999±0.00	0.997±0.00	0.999±0.00
sonar	0.802±0.07	0.793±0.08	0.795±0.08	0.795±0.08	0.751±0.08 ●	0.794±0.08
pima-diabetes	0.779±0.04	0.753±0.04 ●	0.758±0.04 ●	0.754±0.04 ●	0.685±0.05 ●	0.754±0.04 ●
vote	0.989±0.01	0.987±0.01	0.988±0.01	0.987±0.01	0.987±0.01	0.987±0.01
yeast	0.762±0.03	0.741±0.03 ●	0.749±0.03 ●	0.744±0.03 ●	0.694±0.04 ●	0.743±0.03 ●
hepatitis	0.748±0.09	0.738±0.10	0.740±0.10	0.740±0.10	0.724±0.11	0.740±0.10
liver-disorders	0.687±0.07	0.678±0.07	0.681±0.07	0.678±0.07	0.656±0.07	0.679±0.07
spambase	0.967±0.01	0.961±0.01 ●	0.962±0.01 ●	0.962±0.01 ●	0.932±0.01 ●	0.962±0.01 ●
ionosphere	0.931±0.03	0.925±0.03	0.926±0.03	0.925±0.03	0.892±0.04 ●	0.925±0.03
sick	0.988±0.01	0.987±0.01	0.987±0.01	0.987±0.01	0.963±0.02 ●	0.987±0.01
spect	0.738±0.06	0.707±0.06 ●	0.715±0.06	0.709±0.06 ●	0.683±0.07 ●	0.707±0.06 ●

○, ● statistically significant difference

Table 4.7: Area under ROC curve (M-Branch vs PPM methods)

than M-Branch smoothing for all datasets except for *monk's problem 2*. For *pima diabetes*, *yeast*, *spambase* and *spect*, all PPM methods are significantly worse than M-Branch smoothing. PPMD is particularly poor. As mentioned in previous chapters, it has the least smoothing effects among all the PPM methods.

4.4.2 Root mean squared error

Similar to Table 4.7, Table 4.8 lists the root mean squared error for M-Branch smoothing and all the PPM methods with M-Branch as the test base. The ● symbol indicates improvements of PPM methods over M-Branch smoothing. The ○ symbol indicates a larger root mean squared error produced by the PPM methods, which corresponds to a degradation.

In Table 4.8, we can see that for some of the datasets PPM methods are better than M-Branch smoothing. For *monk's problem 1*, *monk's problem 3*, *mushroom* and *kr vs kp*, almost all the PPM methods exhibit significantly improved smoothing probability estimates. By looking at the raw values, we can also see that for *sick*, all the PPM methods are better than M-Branch smoothing.

Dataset	M-Branch	PPMA	PPMB	PPMC	PPMD	PPMP
monks-1	0.223±0.07	0.172±0.10 ●	0.196±0.10	0.170±0.10 ●	0.138±0.12 ●	0.189±0.10
monks-2	0.486±0.02	0.537±0.03 ○	0.517±0.03 ○	0.537±0.03 ○	0.587±0.04 ○	0.529±0.03 ○
monks-3	0.124±0.03	0.114±0.03 ●	0.103±0.04 ●	0.110±0.04 ●	0.111±0.04	0.107±0.04 ●
tic-tac-toe	0.323±0.02	0.329±0.03	0.330±0.03	0.329±0.03	0.351±0.03 ○	0.332±0.03
mushroom	0.020±0.01	0.007±0.00 ●	0.008±0.00 ●	0.007±0.00 ●	0.000±0.00 ●	0.007±0.00 ●
breast-cancer	0.206±0.04	0.216±0.04	0.212±0.04	0.215±0.04	0.223±0.04 ○	0.215±0.04
kr-vs-kp	0.083±0.01	0.074±0.02 ●	0.076±0.02 ●	0.074±0.02 ●	0.072±0.02 ●	0.074±0.02 ●
sonar	0.443±0.06	0.469±0.06 ○	0.463±0.06 ○	0.467±0.06 ○	0.487±0.07 ○	0.469±0.06 ○
pima-diabetes	0.451±0.03	0.491±0.03 ○	0.480±0.03 ○	0.488±0.03 ○	0.513±0.03 ○	0.490±0.03 ○
vote	0.177±0.04	0.186±0.03	0.181±0.03	0.183±0.03	0.185±0.03	0.183±0.03
yeast	0.247±0.01	0.267±0.01 ○	0.259±0.01 ○	0.264±0.01 ○	0.275±0.01 ○	0.266±0.01 ○
hepatitis	0.388±0.05	0.406±0.06 ○	0.396±0.05	0.401±0.05 ○	0.407±0.06 ○	0.401±0.05 ○
liver-disorders	0.496±0.04	0.533±0.05 ○	0.525±0.05 ○	0.531±0.05 ○	0.556±0.05 ○	0.533±0.05 ○
spambase	0.242±0.01	0.253±0.01 ○	0.250±0.01 ○	0.252±0.01 ○	0.261±0.01 ○	0.252±0.01 ○
ionosphere	0.288±0.04	0.300±0.04	0.296±0.04	0.298±0.04	0.308±0.04	0.299±0.04
sick	0.096±0.01	0.094±0.02	0.095±0.01	0.094±0.02	0.095±0.02	0.095±0.02
spect	0.461±0.03	0.493±0.04 ○	0.484±0.04 ○	0.491±0.04 ○	0.512±0.04 ○	0.493±0.04 ○

○, ● statistically significant difference

Table 4.8: Root mean squared error (M-Branch vs PPM methods)

4.4.3 Entropy gain

Table 4.9 lists the entropy gain of M-Branch smoothing and all the PPM methods mentioned in previous chapters, with M-Branch as the test base. The ○ symbol in the second to the sixth column indicates improvements for PPM methods correspondingly. ● indicates degradation of PPM methods compared to M-Branch smoothing.

By looking at Table 4.9, we can see that four datasets have significantly improved entropy gain for PPMA, two for PPMB, four for PPMC, one for PPMD and three for PPMP. On the other hand, most of the datasets prefer M-Branch smoothing. The results confirm that for the datasets mentioned in the last subsection, PPM methods performs better than M-Branch smoothing. However for most of the datasets, M-Branch smoothing performs better than all PPM methods.

Discussion

To sum up, M-Branch smoothing has a better effect on smoothing probability estimates for unpruned PETs, although for some datasets, PPM methods are better. Considering these datasets, *monk's problem 1* and *monk's problem 3* are medium sized datasets, *mushroom*, *kr vs kp* and *sick* are large datasets. Further experiments would need to be conducted to

Dataset	M-Branch	PPMA	PPMB	PPMC	PPMD	PPMP
monks-1	0.69±0.12	0.77±0.17 ○	0.74±0.18	0.78±0.17 ○	-21.55±27.38	0.75±0.18
monks-2	-0.02±0.07	-0.31±0.15 ●	-0.20±0.12 ●	-0.30±0.14 ●	-266.96±50.83 ●	-0.27±0.14 ●
monks-3	0.86±0.04	0.89±0.06 ○	0.90±0.06 ○	0.89±0.06 ○	-9.50± 9.92 ●	0.90±0.06 ○
tic-tac-toe	0.45±0.05	0.42±0.09	0.43±0.08	0.43±0.09	-88.80±26.04 ●	0.42±0.09
mushroom	0.99±0.00	1.00±0.00 ○	1.00±0.00 ○	1.00±0.00 ○	1.00± 0.00 ○	1.00±0.00 ○
breast-cancer	0.67±0.11	0.59±0.14 ●	0.61±0.14 ●	0.60±0.14 ●	-34.35±18.67 ●	0.60±0.14 ●
kr-vs-kp	0.95±0.01	0.96±0.02 ○	0.96±0.02	0.96±0.02 ○	-1.55± 2.47 ●	0.96±0.02 ○
sonar	0.00±0.29	-0.38±0.44 ●	-0.33±0.43 ●	-0.36±0.43 ●	-196.75±73.39 ●	-0.37±0.44 ●
pima-diabetes	-0.02±0.13	-0.41±0.20 ●	-0.35±0.20 ●	-0.40±0.20 ●	-216.47±41.15 ●	-0.41±0.20 ●
vote	0.78±0.08	0.74±0.10 ●	0.74±0.10	0.74±0.10 ●	-6.46± 9.13	0.74±0.10 ●
yeast	0.57±0.12	-1.30±0.42 ●	-0.47±0.31 ●	-0.67±0.32 ●	-200.11±32.57 ●	-0.92±0.36 ●
hepatitis	-0.01±0.21	-0.17±0.31 ●	-0.12±0.29 ●	-0.14±0.30 ●	-47.98±45.84 ●	-0.14±0.30 ●
liver-disorders	-0.11±0.17	-0.56±0.30 ●	-0.49±0.29 ●	-0.54±0.30 ●	-253.35±68.16 ●	-0.56±0.30 ●
spambase	0.62±0.04	0.54±0.06 ●	0.55±0.06 ●	0.54±0.06 ●	-43.92± 7.82 ●	0.54±0.06 ●
ionosphere	0.45±0.15	0.33±0.21 ●	0.35±0.21 ●	0.34±0.21 ●	-68.85±34.71 ●	0.34±0.21 ●
sick	0.27±0.02	0.26±0.03	0.27±0.03	0.27±0.03	-3.61± 2.66 ●	0.27±0.03
spect	0.04±0.15	-0.25±0.26 ●	-0.19±0.25 ●	-0.23±0.26 ●	-170.24±58.98 ●	-0.25±0.26 ●

○, ● statistically significant difference

Table 4.9: Entropy gain (M-Branch vs PPM methods)

verify whether PPM methods work better with larger datasets in general.

4.5 Effect of Bagging Trees

As discussed in Chapter 2, bagging is a technique that resamples instances based on the same dataset. It then calculates the probability estimates individually first based on each tree in the ensemble, and an average is calculated to form a final predicted class probability. Experiments with bagging trees are also a part of this thesis. The default bagging setup in Weka is used, thus 10 trees are created with each algorithm on the training data in the experiments. To demonstrate the effect of bagging along, Table 4.10 shows the area under the ROC curve for bagged REPTree without any smoothing compared against normal unpruned REPTree. By observing the table, we can see that all the datasets work better with bagging trees except for *mushroom*, which has a perfect ROC curve (i.e. 100% area under the ROC curve). This table shows that bagging definitely has a positive effect on probability estimates for all datasets considered in this thesis. Similar improvements can be obtained for root mean squared error and entropy gain (see Appendix A).

Dataset	No smoothing	No smoothing with bagging	
monks-1	0.974± 0.03	1.000±0.00	
monks-2	0.568± 0.06	0.609±0.05	
monks-3	0.990± 0.01	0.992±0.01	
tic-tac-toe	0.879± 0.03	0.980±0.01	○
mushroom	1.000± 0.00	1.000±0.00	
breast-cancer	0.952± 0.03	0.989±0.01	○
kr-vs-kp	0.997± 0.00	0.999±0.00	
sonar	0.751± 0.08	0.878±0.06	○
pima-diabetes	0.684± 0.05	0.802±0.03	○
vote	0.984± 0.01	0.989±0.01	
yeast	0.690± 0.03	0.784±0.03	○
hepatitis	0.717± 0.11	0.827±0.08	○
liver-disorders	0.655± 0.07	0.740±0.06	○
spambase	0.932± 0.01	0.981±0.00	○
ionosphere	0.892± 0.04	0.955±0.03	○
sick	0.965± 0.02	0.987±0.01	○
spect	0.678± 0.07	0.739±0.06	

○, ● statistically significant difference

Table 4.10: Area under ROC curve

4.6 Bagging with simple smoothing methods

We now consider the effect of Laplace smoothing and M-Estimate smoothing when using bagging.

4.6.1 Area under ROC curve

Table 4.11 contains the same comparison as Table 4.1, but with bagging. We can observe that there are fewer significantly improved AUC values in Table 4.1. This is because bagged trees with no smoothing already yield a dramatic improvement as was shown in Table 4.10. However, M-Estimate and Laplace correction do yield some improvements over the non-smoothed results. Laplace correction has very similar effects as M-Estimate in this table, but M-Estimate is still preferable for 6 of the 17 datasets, and on 3 datasets it performs significantly better.

4.6.2 Root mean squared error

Table 4.12 shows the root mean squared error value of M-Estimate versus no smoothing and the Laplace correction. The second column is used as the test base, the third and the fourth column are compared to the second column. The ○ in the third and fourth column indicates M-Estimate smoothing has a significantly smaller root mean squared error.

Dataset	M-Estimate	No Smoothing	Laplace correction
monks-1	1.000± 0.00	1.000±0.00	1.000±0.00
monks-2	0.644± 0.05	0.609±0.05 ●	0.633±0.05 ●
monks-3	0.991± 0.01	0.992±0.01	0.991±0.01
tic-tac-toe	0.986± 0.01	0.980±0.01 ●	0.985±0.01 ●
mushroom	1.000± 0.00	1.000±0.00	1.000±0.00
breast-cancer	0.989± 0.01	0.989±0.01	0.989±0.01
kr-vs-kp	1.000± 0.00	0.999±0.00	1.000±0.00
sonar	0.877± 0.06	0.878±0.06	0.878±0.06
pima-diabetes	0.813± 0.03	0.802±0.03 ●	0.810±0.03 ●
vote	0.990± 0.01	0.989±0.01	0.990±0.01
yeast	0.788± 0.03	0.784±0.03	0.788±0.03
hepatitis	0.833± 0.08	0.827±0.08	0.832±0.07
liver-disorders	0.744± 0.06	0.740±0.06	0.743±0.07
spambase	0.981± 0.00	0.981±0.00	0.981±0.00
ionosphere	0.956± 0.03	0.955±0.03	0.956±0.03
sick	0.991± 0.01	0.987±0.01	0.991±0.01
spect	0.754± 0.06	0.739±0.06	0.750±0.06

○, ● statistically significant difference

Table 4.11: Area under ROC curve with Bagging

From Table 4.12, we can see that M-Estimate smoothing does not have an advantage compared with the other two algorithms. Laplace correction has 10 out of 17 datasets where it is better than M-Estimate, and no smoothing has 8 of 17 datasets where it is better than M-Estimate. Comparing the Laplace correction with no smoothing, there are 12 out of 17 datasets that have better root mean squared error values with no smoothing. To sum up, with bagged trees, no smoothing has the best performance in terms of root mean squared error.

4.6.3 Entropy gain

Table 4.13 shows the entropy gain of M-Estimate smoothing versus no smoothing and the Laplace correction, with M-Estimate as the test base.

From the table we can see that M-Estimate smoothing has only 2 significant improvement over no smoothing, but the Laplace correction has 9 out 17 datasets with significantly better results than M-Estimate. The Laplace correction also wins against unsmoothed trees on 12 datasets. This makes the Laplace correction the best method when considering entropy gain.

Dataset	M-Estimate	No Smoothing	Laplace correction
monks-1	0.208± 0.03	0.092±0.03 ●	0.163±0.03 ●
monks-2	0.465± 0.01	0.502±0.02 ○	0.471±0.02 ○
monks-3	0.146± 0.02	0.112±0.03 ●	0.127±0.03 ●
tic-tac-toe	0.289± 0.01	0.251±0.02 ●	0.270±0.01 ●
mushroom	0.019± 0.00	0.002±0.00 ●	0.011±0.00 ●
breast-cancer	0.183± 0.03	0.180±0.03	0.181±0.03 ●
kr-vs-kp	0.085± 0.01	0.071±0.02 ●	0.078±0.01 ●
sonar	0.380± 0.04	0.376±0.04	0.378±0.04
pima-diabetes	0.408± 0.02	0.419±0.02 ○	0.411±0.02 ○
vote	0.200± 0.02	0.178±0.03 ●	0.188±0.03 ●
yeast	0.238± 0.00	0.237±0.01	0.247±0.00 ○
hepatitis	0.353± 0.03	0.356±0.04	0.353±0.04
liver-disorders	0.447± 0.03	0.454±0.03	0.448±0.03
spambase	0.214± 0.01	0.211±0.01 ●	0.212±0.01 ●
ionosphere	0.260± 0.03	0.254±0.04	0.257±0.03 ●
sick	0.093± 0.01	0.088±0.01 ●	0.090±0.01 ●
spect	0.444± 0.02	0.458±0.03 ○	0.447±0.03

○, ● statistically significant differences

Table 4.12: Root mean squared error with bagging

Discussion

From the three tables, Table 4.11, Table 4.12 and Table 4.13, it is difficult to tell which smoothing method is the best of all by just looking at the significant differences. However, they confirm that bagging improves the probability estimates of unsmoothed trees, to a level such that smoothing does not bring much additional benefit. M-Estimate smoothing has a lead in area under ROC curve, no smoothing has a significant lead in root mean squared error, and the Laplace correction leads in the entropy gain. Because of the poor performance in entropy gain for unsmoothed trees, M-Branch smoothing will be compared with the Laplace correction in the next section: it is the second best in both AUCs and root mean squared error, and it leads in entropy gain. In what follows we use Lapalace smoothing as the baseline because it seems to exhibit the most robust behaviour overall.

4.7 M-Branch smoothing with Bagging

In this section, M-Branch smoothing and Laplace smoothing, both with bagging, are compared, to find out the best smoothing method before the PPM methods with bagging are considered. The methods are compared according to the three standard criteria.

Dataset	M-Estimate	No Smoothing	Laplace correction
monks-1	0.70± 0.05	0.92± 0.04 ○	0.78±0.04 ○
monks-2	0.04± 0.04	-4.28± 7.02	0.02±0.05
monks-3	0.82± 0.02	-6.69± 8.41	0.86±0.03 ○
tic-tac-toe	0.48± 0.02	0.28± 1.92	0.54±0.03 ○
mushroom	0.99± 0.00	1.00± 0.00 ○	0.99±0.00 ○
breast-cancer	0.75± 0.04	-1.70± 5.09	0.76±0.04 ○
kr-vs-kp	0.94± 0.01	0.25± 1.17	0.96±0.01 ○
sonar	0.35± 0.10	-0.42± 4.45	0.36±0.11
pima-diabetes	0.21± 0.06	-3.63± 5.89	0.19±0.07 ●
vote	0.72± 0.04	-0.69± 4.03	0.75±0.04 ○
yeast	0.65± 0.07	-47.91±13.70 ●	0.45±0.05 ●
hepatitis	0.17± 0.08	-0.89± 5.93	0.17±0.10
liver-disorders	0.13± 0.08	-3.81± 9.48	0.12±0.10
spambase	0.72± 0.02	0.50± 0.59	0.73±0.02 ○
ionosphere	0.59± 0.08	-2.62±10.94	0.59±0.09
sick	0.28± 0.01	-0.36± 1.20	0.28±0.01 ○
spect	0.14± 0.07	-7.34±13.57	0.12±0.09

○, ● statistically significant differences

Table 4.13: Entropy gain with bagging

4.7.1 Area under ROC curve

Table 4.14 shows the area under the ROC curve for Laplace correction versus M-Branch smoothing. ○ in the second column shows the datasets that have a significantly increased AUC with M-Branch smoothing.

From the ○ symbols, we can see there are two significant improvements for M-Branch smoothing over the Laplace correction and 3 significant degradations. It is hard to tell which one is better from these results. By comparing the values more closely, there are 6 wins, 4 ties, and 7 losses in AUC, so the Laplace correction wins by 1, hence it is very hard to tell which one is better.

4.7.2 Root mean squared error

Table 4.15 shows the root mean squared error for the Laplace correction and M-Branch smoothing, with the Laplace correction as the test base. ○ indicates significantly increased error for M-Branch smoothing, ● indicates significantly decreased root mean squared error, which is what we are looking for.

There are 7 significant degradations for M-Branch smoothing over Laplace smooth-

Dataset	Laplace correction	M-Branch
monks-1	1.000±0.00	1.000±0.00
monks-2	0.633±0.05	0.616±0.06 ●
monks-3	0.991±0.01	0.990±0.01
tic-tac-toe	0.985±0.01	0.978±0.01 ●
mushroom	1.000±0.00	1.000±0.00
breast-cancer	0.989±0.01	0.989±0.01
kr-vs-kp	1.000±0.00	0.999±0.00
sonar	0.878±0.06	0.873±0.06
pima-diabetes	0.810±0.03	0.818±0.03 ○
vote	0.990±0.01	0.991±0.01
yeast	0.788±0.03	0.796±0.03 ○
hepatitis	0.832±0.07	0.834±0.08
liver-disorders	0.743±0.07	0.746±0.06
spambase	0.981±0.00	0.980±0.00 ●
ionosphere	0.956±0.03	0.955±0.03
sick	0.991±0.01	0.991±0.01
spect	0.750±0.06	0.763±0.06

○, ● statistically significant differences

Table 4.14: Area under ROC curve with bagging (Laplace correction vs M-Branch)

ing. There are 2 significant improvement for M-Branch smoothing. The datasets are *vote* and *yeast*. The Laplace correction is clearly the better algorithm for root mean squared error by looking at the indicators for significant differences. Also, if we look at the table more closely, there are 9 datasets where we are better off with Laplace smoothing based on raw estimates.

4.7.3 Entropy gain

Table 4.16 shows the entropy gain of the Laplace correction compared against M-Branch smoothing with Laplace correction as the test base. ○ in the second column shows significant improvements of entropy gain for Laplace smoothing, ● shows significant degradation.

From the table, we can observe that there are 6 significant degradations and two significant improvement for M-Branch smoothing over the Laplace correction. The improved datasets are identical to those in the case of root mean squared error: *vote* and *yeast*. By comparing the entropy gain values directly, there 10 datasets where we are better off with the Laplace correction.

Dataset	Laplace correction	M-Branch
monks-1	0.163±0.03	0.207 ± 0.03 ○
monks-2	0.471±0.02	0.469 ± 0.01
monks-3	0.127±0.03	0.119 ± 0.03
tic-tac-toe	0.270±0.01	0.289 ± 0.01 ○
mushroom	0.011±0.00	0.021 ± 0.01 ○
breast-cancer	0.181±0.03	0.182 ± 0.03
kr-vs-kp	0.078±0.01	0.084 ± 0.01 ○
sonar	0.378±0.04	0.381 ± 0.04
pima-diabetes	0.411±0.02	0.408 ± 0.02
vote	0.188±0.03	0.174 ± 0.03 ●
yeast	0.247±0.00	0.232 ± 0.01 ●
hepatitis	0.353±0.04	0.352 ± 0.03
liver-disorders	0.448±0.03	0.446 ± 0.03
spambase	0.212±0.01	0.216 ± 0.01 ○
ionosphere	0.257±0.03	0.264 ± 0.03 ○
sick	0.090±0.01	0.094 ± 0.01 ○
spect	0.447±0.03	0.442 ± 0.03

○, ● statistically significant differences

Table 4.15: Root mean squared error with bagging (Laplace correction vs M-Branch)

Discussion

To sum up, after comparing the Laplace correction with M-Branch smoothing when used in bagged trees, there is no clear conclusion regarding which one is better from the area under the ROC curve table. However, observing the other two measures of smoothing effects, the Laplace correction is the better one, considering the differences in root mean squared error and entropy gain are more significant and easier to observe. The Laplace correction with bagging will be compared with PPM methods in the next section to see which smoothing algorithm has the best effect on probability estimates in bagged trees.

4.8 PPM with Bagging

In this section, the best performing smoothing method for bagged trees from previous sections, the Laplace correction is compared with PPM methods. As before, the methods are going to be evaluated in three different aspects: AUC, root mean squared error and entropy gain.

Dataset	Laplace correction	M-Branch
monks-1	0.782±0.04	0.714±0.05 •
monks-2	0.024±0.05	0.028±0.04
monks-3	0.864±0.03	0.864±0.03
tic-tac-toe	0.538±0.03	0.508±0.03 •
mushroom	0.994±0.00	0.992±0.00 •
breast-cancer	0.756±0.04	0.754±0.05
kr-vs-kp	0.956±0.01	0.951±0.01 •
sonar	0.355±0.11	0.346±0.12
pima-diabetes	0.191±0.07	0.194±0.08
vote	0.754±0.04	0.791±0.07 ◦
yeast	0.453±0.05	0.870±0.09 ◦
hepatitis	0.168±0.10	0.167±0.10
liver-disorders	0.122±0.10	0.126±0.10
spambase	0.727±0.02	0.720±0.02 •
ionosphere	0.593±0.09	0.561±0.11 •
sick	0.283±0.01	0.278±0.02
spect	0.123±0.09	0.131±0.11

◦, • statistically significant differences

Table 4.16: Entropy gain with bagging(Laplace correction vs M-Branch)

4.8.1 Area under ROC curve

Table 4.17 shows the AUC for Laplace correction and the PPM methods with Laplace correction as the test base. The • symbols in the columns other than the left-most column show significant degradation in AUCs for PPM methods.

From the table all we can see are significant degradations, no PPM method has significant improvements over the Laplace correction. *monk's problem 2* and *tic tac toe* are the two datasets for which all PPM methods are significantly worse than the Laplace correction. Therefore the Laplace correction is the best of all methods in this table. However, the differences in AUC are very small in almost all cases.

4.8.2 Root mean squared error

Table 4.18 shows the root mean squared error of the Laplace correction versus the PPM methods with Laplace correction as the test base. • symbols indicate significantly improved performance of PPM methods. ◦ symbols shows a significantly increased root mean squared error.

By looking at the table, we can see that there are many significant differences,

Dataset	Laplace correction	PPMA	PPMB	PPMC	PPMD	PPMP
monks-1	1.000±0.00	1.000±0.00	1.000±0.00	1.000±0.00	1.000±0.00	1.000±0.00
monks-2	0.633±0.05	0.612±0.05 ●	0.609±0.05 ●	0.609±0.05 ●	0.604±0.05 ●	0.613±0.05 ●
monks-3	0.991±0.01	0.989±0.01	0.989±0.01	0.989±0.01	0.992±0.01	0.989±0.01
tic-tac-toe	0.985±0.01	0.981±0.01 ●	0.978±0.01 ●	0.980±0.01 ●	0.979±0.01 ●	0.979±0.01 ●
mushroom	1.000±0.00	1.000±0.00	1.000±0.00	1.000±0.00	1.000±0.00	1.000±0.00
breast-cancer	0.989±0.01	0.990±0.01	0.990±0.01	0.990±0.01	0.989±0.01	0.990±0.01
kr-vs-kp	1.000±0.00	0.999±0.00	0.999±0.00	0.999±0.00	0.999±0.00	0.999±0.00
sonar	0.878±0.06	0.879±0.06	0.878±0.06	0.879±0.06	0.878±0.06	0.878±0.06
pima-diabetes	0.810±0.03	0.809±0.03	0.812±0.03	0.810±0.03	0.803±0.03 ●	0.809±0.03
vote	0.990±0.01	0.990±0.01	0.991±0.01	0.991±0.01	0.990±0.01	0.991±0.01
yeast	0.788±0.03	0.789±0.03	0.792±0.03	0.790±0.03	0.786±0.03	0.789±0.03
hepatitis	0.832±0.07	0.831±0.08	0.832±0.07	0.831±0.08	0.829±0.08	0.831±0.08
liver-disorders	0.743±0.07	0.743±0.06	0.745±0.06	0.744±0.06	0.741±0.06	0.744±0.06
spambase	0.981±0.00	0.981±0.00	0.981±0.00	0.981±0.00	0.981±0.00	0.981±0.00
ionosphere	0.956±0.03	0.956±0.03	0.956±0.03	0.956±0.03	0.955±0.03	0.956±0.03
sick	0.991±0.01	0.990±0.01	0.990±0.01	0.990±0.01	0.986±0.02	0.990±0.01
spect	0.750±0.06	0.748±0.06	0.752±0.06	0.749±0.06	0.741±0.06	0.748±0.06

○, ● statistically significant differences

Table 4.17: Area under ROC curve (Laplace correction vs PPM methods

more than in Table 4.17. There are 8 out of 17 significantly improved root mean squared error values for PPMA, 3 for PPMB, 7 for PPMC, 8 for PPMD and 7 for PPMP. On the other hand, there are 2 significantly degraded root mean squared error values for PPMA, 1 for PPMB, 2 for PPMC, 3 for PPMD and 2 for PPMP. These results show that PPM methods are better than the Laplace correction for root mean squared error. *monk's problem 2* and *pima diabetes* are the two odd datasets for which the Laplace correction is preferred.

4.8.3 Entropy gain

Table 4.19 has the comparison for the Laplace correction and PPM methods for entropy gain. ○ symbols indicate significant improvement in entropy gain for PPM methods.

From the table we can see there are only 2 degradations for PPM methods, *monk's problem 2* and *pima diabetes*, which is identical to root mean squared error. There are no significant degradations for PPMD, but fewer improvements too. There are 5 out of 17 significant improvements for PPMA, 5 for PPMB, 6 for PPMC, 2 for PPMD and 5 for PPMP.

Dataset Dataset	Laplace correction	PPMA	PPMB	PPMC	PPMD	PPMP
monks-1	0.163±0.03	0.138±0.03 ●	0.162±0.04	0.137±0.03 ●	0.091±0.03 ●	0.152±0.04 ●
monks-2	0.471±0.02	0.486±0.02 ○	0.481±0.02 ○	0.487±0.02 ○	0.504±0.02 ○	0.484±0.02 ○
monks-3	0.127±0.03	0.108±0.03 ●	0.101±0.04 ●	0.106±0.04 ●	0.107±0.04 ●	0.104±0.04 ●
tic-tac-toe	0.270±0.01	0.260±0.02 ●	0.268±0.02	0.261±0.02 ●	0.252±0.02 ●	0.264±0.02 ●
mushroom	0.011±0.00	0.009±0.00 ●	0.010±0.01	0.009±0.00 ●	0.002±0.00 ●	0.009±0.00 ●
breast-cancer	0.181±0.03	0.179±0.03	0.180±0.03	0.179±0.03	0.180±0.03	0.179±0.03
kr-vs-kp	0.078±0.01	0.076±0.01 ●	0.078±0.01	0.076±0.01 ●	0.071±0.02 ●	0.076±0.01 ●
sonar	0.378±0.04	0.376±0.04	0.377±0.04	0.377±0.04	0.376±0.04	0.376±0.04
pima-diabetes	0.411±0.02	0.415±0.02 ○	0.413±0.02	0.414±0.02 ○	0.418±0.02 ○	0.415±0.02 ○
vote	0.188±0.03	0.176±0.03 ●	0.175±0.03 ●	0.175±0.03 ●	0.176±0.03 ●	0.175±0.03 ●
yeast	0.247±0.00	0.235±0.01 ●	0.234±0.01 ●	0.235±0.01 ●	0.237±0.01 ●	0.235±0.01 ●
hepatitis	0.353±0.04	0.354±0.04	0.353±0.04	0.353±0.04	0.354±0.04	0.353±0.04
liver-disorders	0.448±0.03	0.450±0.03	0.449±0.03	0.449±0.03	0.453±0.03	0.450±0.03
spambase	0.212±0.01	0.212±0.01 ●	0.212±0.01	0.212±0.01	0.211±0.01 ●	0.212±0.01
ionosphere	0.257±0.03	0.256±0.03	0.257±0.03	0.256±0.03	0.254±0.04	0.256±0.03
sick	0.090±0.01	0.089±0.01	0.092±0.01	0.090±0.01	0.089±0.01	0.090±0.01
spect	0.447±0.03	0.452±0.03	0.450±0.03	0.451±0.03	0.457±0.03 ○	0.452±0.03

○, ● statistically significant differences

Table 4.18: Root mean squared error (Laplace correction vs PPM methods)

Discussion

From the three tables in this section, we can see that PPM smoothing methods have improved the probability estimates from the Laplace correction considering that they perform better in both root mean squared error and entropy gain. As the best tested method in previous sections, Laplace correction is a very good smoothing method with bagged trees, but the experiments in this section show that PPM methods are better than the Laplace correction with bagged trees. PPM methods are the best of all smoothing methods considered when using in bagged trees.

4.9 Smoothing pruned trees

In this section, pruned trees are used to test the performance of the smoothing methods considered in previous sections. Reduced-error pruning is used for this. As discussed in Chapter 2, it is a technique that uses a part of the dataset that is not the same as the part used for building the tree to make the tree smaller. Note that class counts from the pruning data are backfitted into the tree once it has been pruned. This way we aim to have more accurate PETs, increasing the probability estimate performance. To demonstrate the effect of pruning, Table 4.20 shows the area under the ROC curve for default REPTree

Dataset	Laplace correction	PPMA	PPMB	PPMC	PPMD	PPMP
monks-1	0.782±0.04	0.835±0.04 ○	0.802±0.05 ○	0.837±0.04 ○	0.920± 0.03 ○	0.819±0.05 ○
monks-2	0.024±0.05	-0.028±0.07 ●	-0.009±0.06 ●	-0.030±0.07 ●	-4.289± 7.02	-0.018±0.07 ●
monks-3	0.864±0.03	0.898±0.05 ○	0.902±0.06 ○	0.900±0.05 ○	-6.690± 8.41	0.900±0.06 ○
tic-tac-toe	0.538±0.03	0.580±0.03 ○	0.564±0.03 ○	0.578±0.03 ○	0.275± 1.92	0.574±0.03 ○
mushroom	0.994±0.00	0.997±0.00 ○	0.997±0.00 ○	0.997±0.00 ○	0.999± 0.00 ○	0.997±0.00 ○
breast-cancer	0.756±0.04	0.754±0.06	0.754±0.06	0.755±0.06	-0.926± 4.02	0.754±0.06
kr-vs-kp	0.956±0.01	0.962±0.01 ○	0.961±0.01	0.962±0.01 ○	0.253± 1.17	0.962±0.01 ○
sonar	0.355±0.11	0.355±0.13	0.354±0.13	0.354±0.13	-0.422± 4.44	0.355±0.13
pima-diabetes	0.191±0.07	0.151±0.09 ●	0.156±0.09 ●	0.153±0.09 ●	-3.623± 5.89	0.150±0.09 ●
vote	0.754±0.04	0.778±0.08	0.783±0.08	0.781±0.08	-0.686± 4.03	0.781±0.08
yeast	0.453±0.05	0.515±0.18	0.669±0.14 ○	0.629±0.15 ○	-0.425± 2.27	0.580±0.16
hepatitis	0.168±0.10	0.155±0.13	0.160±0.12	0.158±0.12	-0.538± 4.88	0.158±0.12
liver-disorders	0.122±0.10	0.093±0.12	0.099±0.12	0.095±0.12	-3.803± 9.48	0.093±0.12
spambase	0.727±0.02	0.725±0.03	0.724±0.03	0.725±0.03	0.505± 0.59	0.725±0.03
ionosphere	0.593±0.09	0.578±0.12	0.575±0.12	0.578±0.12	-2.615±10.94	0.578±0.12
sick	0.283±0.01	0.280±0.02	0.278±0.02	0.279±0.02	-0.019± 0.82	0.279±0.02
spect	0.123±0.09	0.077±0.14	0.084±0.14	0.079±0.14	-7.338±13.57	0.076±0.14

○, ● statistically significant differences

Table 4.19: Entropy gain (Laplace correction vs PPM methods)

and a pruned REPTree. By observing the table, we can see three ○ in the third column. That means there are three significant improvements of pruned REPTree over the default REPTree. There are no significant degradations for pruned trees. The pruning process thus has a positive effects on the probability estimates. Similar effects can be observed for root mean squared error and entropy gain (see Appendix A).

4.10 Pruning with simple smoothing methods

In this section, the simple smoothing methods are applied to pruned trees instead of unpruned trees. As before this section involves 2 smoothing methods, the Laplace correction and M-Estimate smoothing.

4.10.1 Area under ROC curve

Table 4.21 shows the AUC values for M-Estimate smoothing versus no smoothing and Laplace correction, with M-Estimate as the test base. The ● symbol shows significantly smaller AUC value compared to M-Estimate.

From the table, we can see there is only one significant difference. *spambase* has a

Dataset	No smoothing	No Smoothing with pruning
monks-1	0.974±0.03	0.969 ± 0.04
monks-2	0.568±0.06	0.527 ± 0.06
monks-3	0.990±0.01	0.992 ± 0.01
tic-tac-toe	0.879±0.03	0.867 ± 0.04
mushroom	1.000±0.00	1.000 ± 0.00
breast-cancer	0.952±0.03	0.958 ± 0.02
kr-vs-kp	0.997±0.00	0.998 ± 0.00
sonar	0.751±0.08	0.725 ± 0.08
pima-diabetes	0.684±0.05	0.757 ± 0.05 ∘
vote	0.984±0.01	0.974 ± 0.02
yeast	0.690±0.03	0.744 ± 0.03 ∘
hepatitis	0.717±0.11	0.650 ± 0.13
liver-disorders	0.655±0.07	0.642 ± 0.07
spambase	0.932±0.01	0.954 ± 0.01 ∘
ionosphere	0.892±0.04	0.905 ± 0.04
sick	0.965±0.02	0.964 ± 0.03
spect	0.678±0.07	0.705 ± 0.06

∘, • statistically significant differences

Table 4.20: Area under ROC curve (No smoothing vs No smoothing pruned)

significantly smaller AUC value for no smoothing, which means there is an improvement for M-Estimate smoothing over no smoothing. There is no significant difference between M-Estimate smoothing and the Laplace correction. By comparing the numbers more closely, M-Estimate smoothing has 13 wins over no smoothing, but only 9 wins and 8 losses over Laplace correction. It is easily observed that the M-Estimate is better than no smoothing in terms of AUC, but it is difficult to draw a conclusion between M-Estimate smoothing and the Laplace correction.

4.10.2 Root mean squared error

Table 4.22 shows the root mean squared error for M-Estimate smoothing versus no smoothing and the Laplace correction, with M-Estimate as the test base. The ∘ symbols show a significantly larger root mean squared error for M-Estimate smoothing. • symbols indicates significantly smaller error value.

From Table 4.22, there are many significant differences compared to Table 4.21. In the second column, we can see 4 significant improvements of no smoothing over M-Estimate, but there are more datasets that yield significantly larger root mean squared error. Comparing the values more closely, it is easier to see the better smoothing

Dataset	M-Estimate	No smoothing		Laplace correction
monks-1	0.964± 0.04	0.969±0.04		0.968±0.04
monks-2	0.534± 0.06	0.527±0.06		0.531±0.06
monks-3	0.991± 0.01	0.992±0.01		0.991±0.01
tic-tac-toe	0.871± 0.04	0.867±0.04		0.874±0.04
mushroom	1.000± 0.00	1.000±0.00		1.000±0.00
breast-cancer	0.965± 0.02	0.958±0.02		0.964±0.02
kr-vs-kp	0.998± 0.00	0.998±0.00		0.998±0.00
sonar	0.734± 0.07	0.725±0.08		0.730±0.07
pima-diabetes	0.771± 0.05	0.757±0.05		0.766±0.05
vote	0.977± 0.02	0.974±0.02		0.977±0.02
yeast	0.750± 0.03	0.744±0.03		0.751±0.03
hepatitis	0.660± 0.13	0.650±0.13		0.660±0.13
liver-disorders	0.649± 0.07	0.642±0.07		0.647±0.07
spambase	0.960± 0.01	0.954±0.01	●	0.959±0.01
ionosphere	0.909± 0.03	0.905±0.04		0.909±0.03
sick	0.972± 0.02	0.964±0.03		0.972±0.02
spect	0.713± 0.05	0.705±0.06		0.708±0.05

○, ● statistically significant differences

Table 4.21: Area under ROC curve with pruning

method: the M-Estimate wins in 12 datasets. In the third column, there are 4 significant improvements and 4 significant degradations, which makes M-Estimate smoothing and Laplace correction yield a draw in root mean squared error as well.

4.10.3 Entropy gain

Table 4.23 shows the entropy gain values of M-Estimate smoothing versus no smoothing and the Laplace correction, with M-Estimate as the test base. ● and ○ shows significantly improved or degraded entropy gain respectively.

By observing the table, we can see 3 significant improvements for M-Estimate smoothing over no smoothing. After comparing the values, it is easy to see that *mushroom* is the only dataset where no smoothing shows an advantage. This is because the default tree built from *mushroom* is almost perfect in every aspect and thus does not need any smoothing. The improvements observed make M-Estimate smoothing the better method. In the third column, 5 improvements and 5 degradations are observed with respect to the Laplace correction, making it a tie again. Comparing the values more closely, there are 10 out of 17 datasets where we would prefer M-Estimate smoothing.

Dataset	M-Estimate	No Smoothing	Laplace correction
monks-1	0.246± 0.07	0.196±0.10 ●	0.221±0.08 ●
monks-2	0.482± 0.01	0.495±0.02 ○	0.486±0.02 ○
monks-3	0.116± 0.03	0.094±0.05 ●	0.103±0.04 ●
tic-tac-toe	0.360± 0.02	0.363±0.03	0.358±0.03
mushroom	0.022± 0.01	0.008±0.02 ●	0.016±0.01 ●
breast-cancer	0.214± 0.03	0.217±0.03	0.215±0.03
kr-vs-kp	0.103± 0.02	0.099±0.02 ●	0.100±0.02 ●
sonar	0.454± 0.04	0.465±0.05 ○	0.458±0.05
pima-diabetes	0.426± 0.02	0.434±0.03 ○	0.429±0.02 ○
vote	0.187± 0.04	0.188±0.04	0.187±0.04
yeast	0.242± 0.01	0.245±0.01 ○	0.243±0.00
hepatitis	0.389± 0.03	0.394±0.04	0.390±0.03
liver-disorders	0.481± 0.03	0.495±0.03 ○	0.485±0.03 ○
spambase	0.257± 0.01	0.261±0.01 ○	0.258±0.01 ○
ionosphere	0.293± 0.04	0.295±0.04	0.293±0.04
sick	0.109± 0.01	0.109±0.01	0.109±0.01
spect	0.449± 0.03	0.454±0.03	0.450±0.03

○, ● statistically significant differences

Table 4.22: Root mean squared error with pruning

Discussion

In this section, two simple smoothing methods are applied to pruned REPTree classifiers. The two methods, M-Estimate smoothing and Laplace correction, are better than no smoothing at all, which is a different result from bagged trees. Furthermore, M-Estimate smoothing and the Laplace correction have a very similar effect on smoothing pruned trees. In two of the three tables, it is very hard to tell which one is better than the other, but in the entropy gain table, M-Estimate has a small advantage over the Laplace correction. Thus, it will be used to compare to M-Branch smoothing in the next section.

4.11 M-Branch smoothing on pruned trees

In this section, the best smoothing method on pruned trees so far, M-Estimate smoothing, and M-Branch smoothing are compared based on the standard three criteria.

4.11.1 Area under ROC curve

Table 4.24 shows the AUC values of M-Estimate smoothing and M-Branch smoothing.

From the table, we cannot see any significant difference. However, by comparing

Dataset	M-Estimate	No Smoothing	Laplace correction
monks-1	0.653±0.12	-0.932± 6.22	0.713±0.13 ○
monks-2	-0.640±3.40	-18.857±19.62	-0.653±3.40 ●
monks-3	0.863±0.03	-4.858± 6.75	0.891±0.04 ○
tic-tac-toe	-0.213±2.03	-17.553±11.41 ●	-0.196±2.03 ○
mushroom	0.988±0.00	0.998± 0.00 ○	0.993±0.00 ○
breast-cancer	0.669±0.07	-3.637± 7.33	0.665±0.08
kr-vs-kp	0.926±0.02	0.256± 1.09	0.934±0.02 ○
sonar	0.108±0.14	-17.730±29.55	0.081±0.16
pima-diabetes	0.149±0.07	-12.979±13.81	0.134±0.08 ●
vote	0.753±0.07	0.382± 2.10	0.756±0.07
yeast	0.704±0.08	-62.601±19.92 ●	0.667±0.08 ●
hepatitis	0.049±0.09	-4.462±16.75	0.042±0.10
liver-disorders	0.026±0.09	-25.159±28.87	0.002±0.10 ●
spambase	0.628±0.03	-4.914± 3.55 ●	0.624±0.03 ●
ionosphere	0.498±0.09	-5.153±10.59	0.496±0.10
sick	0.256±0.02	-0.608± 1.24	0.258±0.02
spect	0.119±0.08	-7.319±16.79	0.110±0.09

○, ● statistically significant differences

Table 4.23: Entropy gain with pruning

the values, we can find out that there are 7 small improvements for M-Branch smoothing over M-Estimate smoothing, and there are 2 degradations and 8 ties. It is too early to tell which one is better based on area under the ROC curve alone, but M-Branch smoothing has a small advantage.

4.11.2 Root mean squared error

Table 4.25 shows the root mean squared error values for M-Estimate smoothing versus M-Branch smoothing, with M-Estimate as the test base. The ● symbol in the second column indicates significantly smaller root mean squared error.

From the table, we can see that there is only one significant difference: *yeast* has a better probability estimate with M-Branch smoothing. Comparing the values, we find that there are 6 datasets that have a smaller root mean squared error with M-Branch smoothing. There are 8 ties in this table too, which makes 3 of the 17 datasets favour M-Estimate smoothing. Considering the small differences, M-Branch smoothing has a very small advantage in root mean squared error again, just like in area under ROC curves.

Dataset	M-Estimate	M-Branch
monks-1	0.964± 0.04	0.966±0.04
monks-2	0.534± 0.06	0.525±0.06
monks-3	0.991± 0.01	0.991±0.01
tic-tac-toe	0.871± 0.04	0.873±0.04
mushroom	1.000± 0.00	1.000±0.00
breast-cancer	0.965± 0.02	0.965±0.02
kr-vs-kp	0.998± 0.00	0.998±0.00
sonar	0.734± 0.07	0.735±0.07
pima-diabetes	0.771± 0.05	0.770±0.05
vote	0.977± 0.02	0.977±0.02
yeast	0.750± 0.03	0.756±0.03
hepatitis	0.660± 0.13	0.661±0.13
liver-disorders	0.649± 0.07	0.651±0.07
spambase	0.960± 0.01	0.960±0.01
ionosphere	0.909± 0.03	0.909±0.03
sick	0.972± 0.02	0.972±0.02
spect	0.713± 0.05	0.715±0.05

○, ● statistically significant differences

Table 4.24: Area under AUC curve with pruning (M-Estimate vs M-Branch)

4.11.3 Entropy gain

Table 4.26 shows the entropy gain of M-Estimate smoothing and M-Branch smoothing, with M-Estimate as the test base. The ○ symbols in the second column indicate a significantly improved entropy gain with M-Branch smoothing.

From the table, we observe 3 significant improvements with M-Branch smoothing. *Monk's problem 3*, *mushroom* and *kr vs kp* have a better entropy gain with this technique. There are no significant degradations in the table. Thus M-Branch smoothing performs better considering entropy gain.

Discussion

In this section, M-Estimate smoothing and M-Branch smoothing on pruned REPTree are compared in detail. M-Branch smoothing has a small advantage in area under ROC curve, a small advantage in root mean squared error, and a relatively bigger advantage in entropy gain. To sum up, M-Branch smoothing has better effects over M-Estimate smoothing on the probability estimates of pruned REPTree.

Dataset	M-Estiamte	M-Branch
monks-1	0.246± 0.07	0.244±0.07
monks-2	0.482± 0.01	0.482±0.01
monks-3	0.116± 0.03	0.113±0.03
tic-tac-toe	0.360± 0.02	0.359±0.02
mushroom	0.022± 0.01	0.024±0.01
breast-cancer	0.214± 0.03	0.214±0.03
kr-vs-kp	0.103± 0.02	0.103±0.02
sonar	0.454± 0.04	0.454±0.04
pima-diabetes	0.426± 0.02	0.426±0.02
vote	0.187± 0.04	0.186±0.04
yeast	0.242± 0.01	0.241±0.01 ●
hepatitis	0.389± 0.03	0.388±0.03
liver-disorders	0.481± 0.03	0.481±0.03
spambase	0.257± 0.01	0.257±0.01
ionosphere	0.293± 0.04	0.294±0.04
sick	0.109± 0.01	0.110±0.01
spect	0.449± 0.03	0.449±0.03

○, ● statistically significant differences

Table 4.25: Root mean squared error with pruning (M-Estimate vs M-Branch)

4.12 PPM with pruning

In this section, M-Branch smoothing, the best smoothing method on pruned REPTree classifiers so far, is compared with the new PPM-based smoothing methods.

4.12.1 Area under ROC curve

Table 4.27 shows the area under ROC curve for M-Branch smoothing and the PPM methods, with M-Estimate as the test base, which means all the PPM methods are compared to M-Branch smoothing, the left most column. ● indicates significant degradation in area under ROC curve for PPM methods.

We can see only two significant degradations, both of them are for the PPMD method. *yeast* and *spambase* are the two datasets where M-Branch smoothing has a significant advantage. As the table shows, it is very hard to tell which smoothing method is better just by the significant differences. If we look at the raw values more closely, M-Branch smoothing has 10 wins, 3 losses and 4 ties over PPMA, 8 wins, 3 losses and 6 ties over PPMB, 9 wins, 3 losses and 5 ties over PPMC, 11 wins, 3 losses and 3 ties over PPMD, and finally, it has 9 wins, 3 losses and 5 ties over PPMP. In the table, we can observe that *mushroom*, *kr vs kp* and *vote* have the same AUC values across the

Dataset	M-Estimate	M-Branch
monks-1	0.653±0.12	0.660±0.12
monks-2	-0.640±3.40	-0.018±0.05
monks-3	0.863±0.03	0.871±0.04 ○
tic-tac-toe	-0.213±2.03	0.354±0.07
mushroom	0.988±0.00	0.991±0.00 ○
breast-cancer	0.669±0.07	0.665±0.08
kr-vs-kp	0.926±0.02	0.930±0.02 ○
sonar	0.108±0.14	0.098±0.15
pima-diabetes	0.149±0.07	0.144±0.07
vote	0.753±0.07	0.756±0.07
yeast	0.704±0.08	0.691±0.10
hepatitis	0.049±0.09	0.047±0.10
liver-disorders	0.026±0.09	0.019±0.10
spambase	0.628±0.03	0.626±0.03
ionosphere	0.498±0.09	0.492±0.10
sick	0.256±0.02	0.256±0.02
spect	0.119±0.08	0.116±0.09

○, ● statistically significant differences

Table 4.26: Entropy gain with pruning (M-Estimate vs M-Branch)

whole table. *monk's problem 1* and *monk's problem 2* are the two datasets that have better results with PPM methods across the table. Overall, M-Branch smoothing has a small advantage over the PPM methods with no significant differences except compared to PPMD.

4.12.2 Root mean squared error

Table 4.28 shows the root mean squared error for M-Branch smoothing and PPM smoothing methods, with M-Branch as the test base. ○ indicates significant degradation in root mean squared error for PPM methods over M-Branch smoothing, ● indicates significantly improved root mean squared error.

In contrast to the area under ROC curve table, this table has many significant differences. Just by looking at them it is hard to tell which method is the better one. M-Branch smoothing seems to be tieing with all the PPM methods. It has 5 wins, 4 losses and eight ties over PPMA, 3 wins, 4 loses and 10 ties over PPMB, 4 wins, 4 losses and 9 ties over PPMC, 5 wins, 4 losses and 8 ties over PPMD, and at last, it has 5 wins, 4 losses and 8 ties with PPMP.

Dataset	M-Branch	PPMA	PPMB	PPMC	PPMD	PPMP
monks-1	0.966±0.04	0.968±0.04	0.967±0.04	0.969±0.04	0.969±0.04	0.968±0.04
monks-2	0.525±0.06	0.528±0.06	0.528±0.06	0.527±0.06	0.528±0.06	0.527±0.06
monks-3	0.991±0.01	0.990±0.01	0.991±0.01	0.991±0.01	0.992±0.01	0.991±0.01
tic-tac-toe	0.873±0.04	0.874±0.04	0.873±0.04	0.874±0.04	0.867±0.04	0.874±0.04
mushroom	1.000±0.00	1.000±0.00	1.000±0.00	1.000±0.00	1.000±0.00	1.000±0.00
breast-cancer	0.965±0.02	0.964±0.02	0.964±0.02	0.964±0.02	0.959±0.02	0.964±0.02
kr-vs-kp	0.998±0.00	0.998±0.00	0.998±0.00	0.998±0.00	0.998±0.00	0.998±0.00
sonar	0.735±0.07	0.727±0.07	0.730±0.07	0.729±0.07	0.724±0.08	0.729±0.07
pima-diabetes	0.770±0.05	0.763±0.05	0.765±0.05	0.764±0.05	0.760±0.05	0.763±0.05
vote	0.977±0.02	0.977±0.02	0.977±0.02	0.977±0.02	0.977±0.02	0.977±0.02
yeast	0.756±0.03	0.750±0.03	0.754±0.03	0.753±0.03	0.747±0.03 •	0.751±0.03
hepatitis	0.661±0.13	0.659±0.13	0.659±0.13	0.660±0.13	0.654±0.13	0.660±0.13
liver-disorders	0.651±0.07	0.646±0.07	0.647±0.07	0.646±0.07	0.646±0.07	0.646±0.07
spambase	0.960±0.01	0.958±0.01	0.958±0.01	0.958±0.01	0.954±0.01 •	0.958±0.01
ionosphere	0.909±0.03	0.908±0.04	0.908±0.04	0.908±0.04	0.907±0.04	0.908±0.04
sick	0.972±0.02	0.972±0.02	0.972±0.02	0.972±0.02	0.964±0.03	0.972±0.02
spect	0.715±0.05	0.708±0.05	0.711±0.05	0.709±0.05	0.708±0.05	0.709±0.05

○, • statistically significant differences

Table 4.27: Area under ROC curve with pruning(M-Branch vs PPM methods

4.12.3 Entropy gain

Table 4.29 show the entropy gain values of M-Branch smoothing and PPM-based smoothing methods, with M-Branch as the test base. As before, ○ shows significant improvement of PPM methods over M-Branch smoothing. The • symbols indicates significant degradations.

By observing the table, there are many significant difference in this table too, but again, M-Branch smoothing and PPM methods seems to be a tieing in terms of entropy gain. M-Branch smoothing has 5 wins, 4 losses and 8 ties over PPMA, 3 wins, 4 losses and 10 ties over PPMB, 4 wins, 4 losses over PPMC, 3 wins, 1 loss and 13 ties over PPMD, and finally it has 4 wins, 4 losses and 9 ties over PPMP. The score is almost identical to root mean squared error. At this stage, it is still hard to tell which method is better. Further experiments would need to be done.

Discussion

From the experimental results presented in the tables in this section, it is hard to tell if the PPM methods are better or M-Branch smoothing is better for pruned trees. Further experiments would need to be done, with a larger collection of datasets.

Dataset	M-Branch	PPMA	PPMB	PPMC	PPMD	PPMP
monks-1	0.244±0.07	0.208±0.09 ●	0.213±0.09 ●	0.208±0.09 ●	0.196±0.10 ●	0.209±0.09 ●
monks-2	0.482±0.01	0.489±0.02 ○	0.486±0.02 ○	0.488±0.02 ○	0.493±0.02 ○	0.488±0.02 ○
monks-3	0.113±0.03	0.097±0.04 ●	0.098±0.04 ●	0.097±0.04 ●	0.095±0.05 ●	0.097±0.04 ●
tic-tac-toe	0.359±0.02	0.359±0.03	0.358±0.03	0.358±0.03	0.362±0.03	0.359±0.03
mushroom	0.024±0.01	0.013±0.01 ●	0.014±0.01 ●	0.013±0.01 ●	0.008±0.01 ●	0.013±0.01 ●
breast-cancer	0.214±0.03	0.215±0.03	0.215±0.03	0.215±0.03	0.216±0.03	0.215±0.03
kr-vs-kp	0.103±0.02	0.099±0.02 ●	0.099±0.02 ●	0.099±0.02 ●	0.098±0.02 ●	0.099±0.02 ●
sonar	0.454±0.04	0.461±0.05	0.459±0.05	0.460±0.05	0.463±0.05	0.460±0.05
pima-diabetes	0.426±0.02	0.431±0.02 ○	0.429±0.02 ○	0.430±0.02 ○	0.433±0.03 ○	0.430±0.02 ○
vote	0.186±0.04	0.187±0.04	0.187±0.04	0.187±0.04	0.187±0.04	0.186±0.04
yeast	0.241±0.01	0.244±0.01 ○	0.242±0.01	0.243±0.01 ○	0.244±0.01 ○	0.243±0.01 ○
hepatitis	0.388±0.03	0.391±0.04	0.390±0.03	0.390±0.04	0.392±0.04	0.390±0.04
liver-disorders	0.481±0.03	0.490±0.03 ○	0.486±0.03 ○	0.488±0.03 ○	0.492±0.03 ○	0.488±0.03 ○
spambase	0.257±0.01	0.259±0.01 ○	0.258±0.01	0.258±0.01	0.260±0.01 ○	0.258±0.01 ○
ionosphere	0.294±0.04	0.294±0.04	0.293±0.04	0.293±0.04	0.294±0.04	0.294±0.04
sick	0.110±0.01	0.108±0.01	0.109±0.01	0.109±0.01	0.109±0.01	0.109±0.01
spect	0.449±0.03	0.452±0.03	0.451±0.03	0.451±0.03	0.453±0.03	0.451±0.03

○, ● statistically significant differences

Table 4.28: Root mean squared error with pruning (M-Branch vs PPM methods)

4.13 Case study: UCSD machine learning competition

In this section, we consider the dataset from the 2010 UCSD machine learning competition. The different smoothing methods are applied to this dataset to measure the performance based on smoothing effects, based on unpruned REPTree classifier.

4.13.1 Dataset and Experiment setup

There were two different dataset in the 2010 UCSD machine learning competition, but our case study focuses on the first dataset, the *E-commerce Customer Identification (Raw)* dataset. There are three files provided for this dataset, a training dataset file, a class label file for the training dataset and a test dataset file. The training dataset has 130475 instances, 334 attributes, with a class label of 0 or 1. The test dataset has 86691 instances with the same format as the training file. The first 334 attributes are identification and characteristics of people, the class label indicates if the corresponding instance is a customer or not. The task of this competition was to build a model on the training dataset, and output the probability estimates on the test dataset instances indicate weather the test instance is going to be a potential new customer. Thus the task was to generate a label file like the second file mentioned above, but for the test dataset and with probability attached. The output results for the test dataset were measured by

Dataset	M-Branch	PPMA	PPMB	PPMC	PPMD	PPMP
monks-1	0.660±0.12	0.751±0.14 ○	0.741±0.14 ○	0.752±0.14 ○	-0.930± 6.22	0.750±0.14 ○
monks-2	-0.018±0.05	-0.051±0.07 ●	-0.036±0.06	-0.045±0.07 ●	-18.848±19.62	-0.047±0.07 ●
monks-3	0.871±0.04	0.904±0.05 ○	0.902±0.05 ○	0.902±0.05 ○	-4.860± 6.75	0.904±0.05 ○
tic-tac-toe	0.354±0.07	0.365±0.08	0.368±0.07	0.368±0.08	-17.547±11.41 ●	0.366±0.08
mushroom	0.991±0.00	0.996±0.00 ○	0.996±0.00 ○	0.996±0.00 ○	0.998± 0.00 ○	0.996±0.00 ○
breast-cancer	0.665±0.08	0.654±0.09	0.657±0.09	0.656±0.09	-3.619± 7.33	0.656±0.09
kr-vs-kp	0.930±0.02	0.938±0.02 ○	0.938±0.02 ○	0.938±0.02 ○	0.257± 1.09	0.938±0.02 ○
sonar	0.098±0.15	0.048±0.18	0.062±0.17	0.058±0.18	-17.717±29.55	0.055±0.18
pima-diabetes	0.144±0.07	0.114±0.09 ●	0.123±0.09 ●	0.119±0.09 ●	-12.972±13.81	0.117±0.09 ●
vote	0.756±0.07	0.756±0.08	0.756±0.08	0.756±0.08	0.388± 2.10	0.756±0.08
yeast	0.691±0.10	0.308±0.24 ●	0.528±0.17 ●	0.505±0.17 ●	-8.909± 8.10 ●	0.407±0.21 ●
hepatitis	0.047±0.10	0.029±0.12	0.037±0.11	0.034±0.12	-4.451±16.75	0.034±0.12
liver-disorders	0.019±0.10	-0.034±0.13 ●	-0.017±0.12	-0.025±0.12	-25.145±28.86	-0.028±0.13
spambase	0.626±0.03	0.617±0.04 ●	0.619±0.04 ●	0.618±0.04 ●	-4.912± 3.55 ●	0.618±0.04 ●
ionosphere	0.492±0.10	0.489±0.11	0.492±0.11	0.491±0.11	-5.149±10.59	0.490±0.11
sick	0.256±0.02	0.256±0.02	0.256±0.02	0.256±0.02	-0.563± 1.23	0.256±0.02
spect	0.116±0.09	0.099±0.10	0.104±0.10	0.102±0.10	-7.314±16.79	0.101±0.10

○, ● statistically significant differences

Table 4.29: Entropy gain with pruning (M-Branch vs PPM methods)

area under ROC curve in the competition.

The experiments presented here are based on the training dataset. The experiment are based on a percentage split instead of cross validations considering the memory usage and the time used for the experiment. The dataset was split on 66% which means 66% of the dataset was used as training data and the rest (34%) was used as testing data.

4.13.2 Simple smoothing methods

In this subsection, we consider the smoothing effect of the Laplace correction and M-Estimate smoothing on the *E-commerce Customer Identification (Raw)* dataset. Figure 4.1 shows the different ROC curves for the simple smoothing methods and unsmoothed trees. The green straight line shows the 50% AUC curve, which is a random classifier.

By observing the curves, we can see that no smoothing has a slightly larger AUC than the green line, but the curve becomes a straight line for the portion from $x = 0.2$ onwards. The curves of M-Estimate smoothing and the Laplace correction are almost on top of each other, we can only differentiate them for the portion $0.1 < x < 0.4$. In the figure, M-Estimate smoothing has the largest area under the ROC curve.

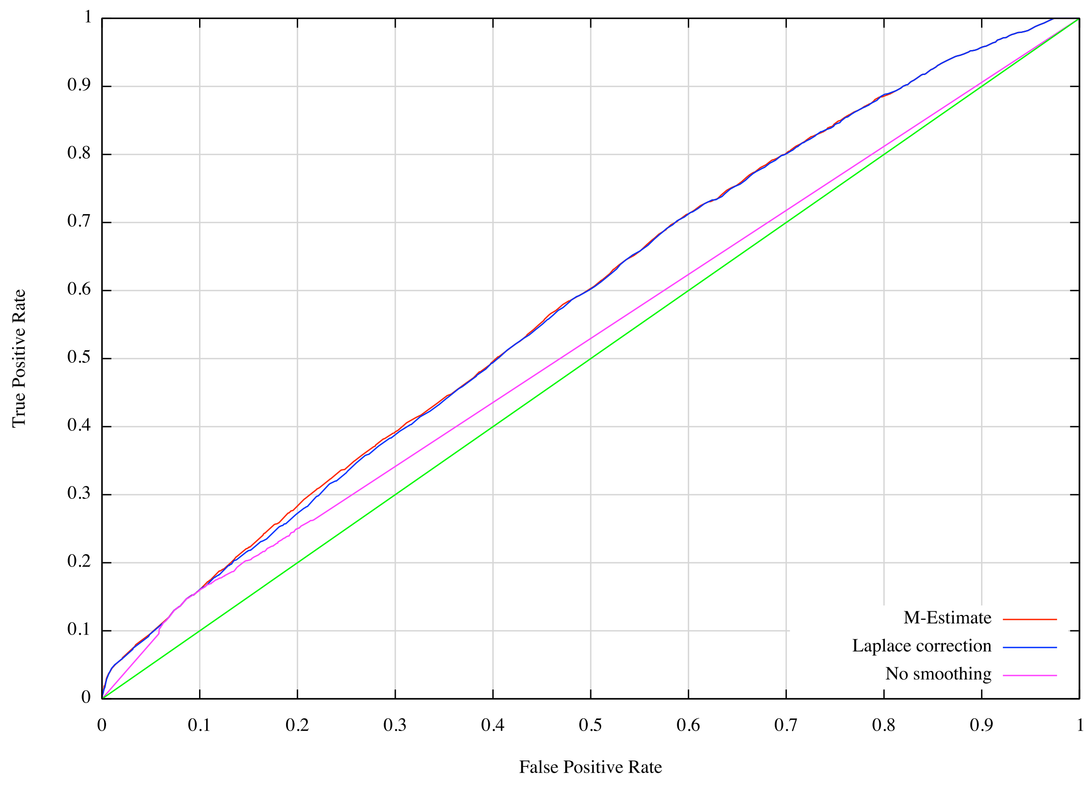

Figure 4.1: ROC curves of simple smoothing methods

4.13.3 M-Branch smoothing

Figure 4.2 shows the ROC curve of M-Branch smoothing and M-Estimate smoothing on the *E-commerce Customer Identification (Raw)* dataset.

From the figure, we can observe that the AUC for M-Branch smoothing is clearly larger than for M-Estimate smoothing: the curve of M-Branch smoothing is above the one for M-Estimate smoothing through at the whole figure.

4.13.4 PPM smoothing methods

Figure 4.3 shows the ROC curves of M-Branch smoothing and PPM smoothing methods.

From the figure, we can see that all the PPM smoothing methods have almost exactly the same curve apart from PPMD. Compared to M-Branch smoothing, the PPMD method does not have advantage for this particular dataset, in fact all the PPM method curves are under the M-Branch smoothing curve. The curve for PPMD, it is almost identical to the one for no smoothing in Figure 4.1.

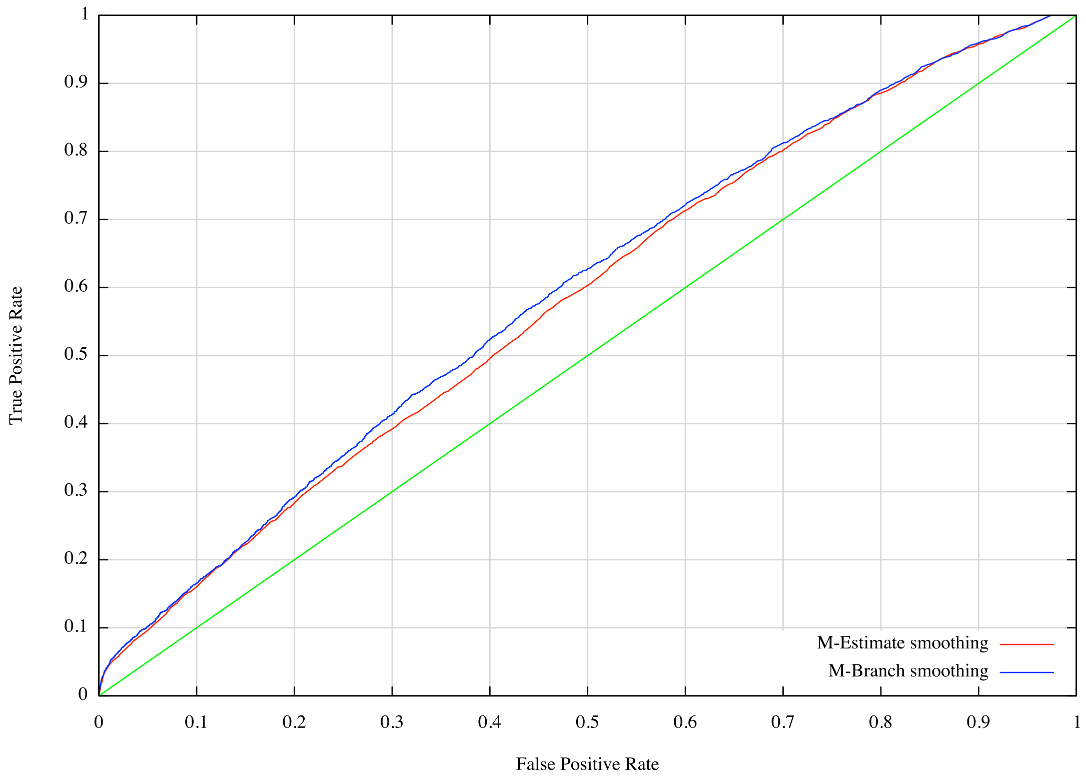

Figure 4.2: ROC curves of M-Estimate vs M-Branch smoothing

Discussion

From the figures in this section, we can see that in a single unpruned PET, M-Branch smoothing has an advantage in area under ROC curve, which is consistent with the results presented earlier in thie chapter. Compared to other datasets used in the experiments, this dataset is very large, both in terms of the number of instances and the number of attributes.

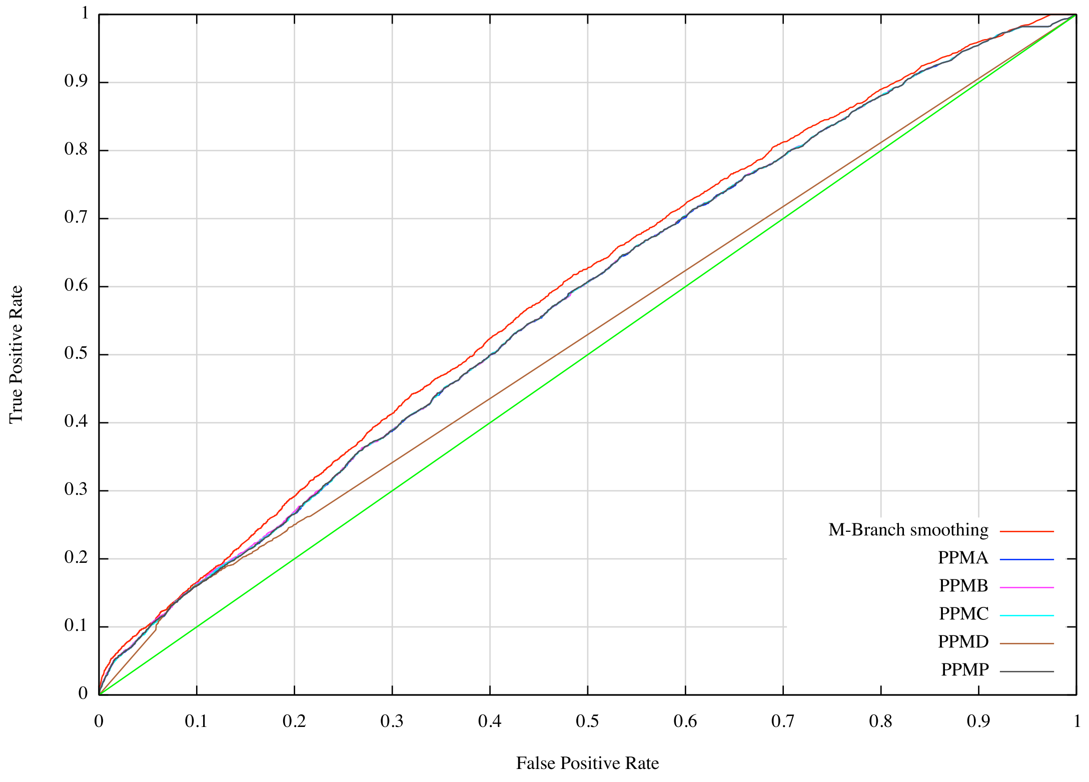

Figure 4.3: ROC curves of M-Branch smoothing vs PPM methods

Chapter 5
Conclusions

Smoothing is a useful technique that improves probability estimates. The results in this thesis shows that different smoothing method can have different effects on different datasets, some perform better for pruned PETs, others perform better for pruned PETs. It is not possible to find out a smoothing methods that smoothes probability estimates the best on all datasets, for all tree-based classifiers compared. This thesis presents a smoothing method used in data compression and applies it to PETs. The experimental results generated using the new method are compared with those of a state-of-art smoothing method: M-Branch smoothing (Ferri, Flach & Hernández-Orallo, 2003). Two other simple smoothing methods, the Laplace correction and M-Estimate smoothing, are also compared against. This chapter summarizes the work presented in the previous chapters, draws conclusions and indicates future work.

5.1 Summary and conclusion

At the beginning of this thesis, relevant background knowledge was introduced in Chapter 2. It covered several different concepts and learning techniques used in this thesis, such as PETs, pruning and bagging. Pruning and bagging were discussed in detail, with pseudo code for both learning and applying PETs and bagged classifiers. An example pruned PET was compared to an unpruned PET in graphical form. Secondly, the PPM data compression concepts were introduced with a brief description of the smoothing method involved in it. The simple Laplace smoothing method was also discussed briefly. Then the datasets used in this thesis were listed and described in detail. Finally, the evaluation methods were introduced and discussed.

The third chapter discussed the smoothing methods in detail. First, the simple smoothing methods were discussed, the Laplace correction and M-Estimate smoothing. Secondly, M-Branch smoothing was discussed, and an example was giving to calculate the probability estimates in a PET using M-Branch smoothing. Then, the PPM smoothing method and the differences between its escape probability calculations were introduced and discussed. The motivation for each escape probability calculation was discussed as well. Each of the PPM smoothing methods were applied to the same dataset and the same PET prediction path used in the M-Branch smoothing example. The smoothed probability estimates of the PPM methods were compared in a table and relatively small differences were observed. At last, the pseudo code used for implementing each smoothing method discussed above was listed and explained.

At the very beginning of Chapter 4, the experiment set ups were introduced. The experiments were divided into three sections. The first part consisted experiments on a single unpruned REPTree with the 17 datasets selected for this thesis. We first discussed the experiment comparing no smoothing with the Laplace correction and M-Estimate smoothing. The experimental results were evaluated base on three critreia discussed: AUC values first, followed by root mean squared error and entropy gain. M-Estimate smoothing had an advantage over no smoothing and the Laplace correction in the experimental results, especially in root mean squared error. M-Estimate smoothing won over the Laplace correction for large majority majority of the datasets. Then, M-Estimate smoothing was compared with M-Branch smoothing with the same experiment set up and comparisons. M-Branch smoothing was the better smoothing method of the two. Lastly, M-Branch smoothing was compared with the PPM smoothing methods. In all comparisons, M-Branch was the best smoothing method among the smoothing methods in this experiment, especially in terms of root mean squared error and entropy gain.

The second set of experiments applied the smoothing methods to bagged trees and compared the smoothing effects on them. As in the previous experiments, the simple smoothing methods were compared first: no smoothing, the Laplace correction and M-Estimate smoothing. Differently from the last experimental results, the smoothing effects on the bagged trees were more difficult to summarize. With the help of bagging, even the unsmoothed REPTree has much better probability estimates, but with the poor performance in entropy gain, no smoothing is not the best method. On the other hand, the Laplace correction shows robustness in all three tests. It was thus used to compare with M-Branch smoothing. After the comparison with M-Branch smoothing, the Laplace correction was still the best smoothing method on the dataset selected. Finally, the Laplace correction was compared with the PPM methods, and the PPM methods showed an advantage in all three test categories. To sum up, according to the experiments, the

PPM methods were the best smoothing methods for bagged trees.

The third set of experiments applied the smoothing methods to a single pruned REPTree. The simple smoothing methods were compared first. It was very difficult to tell which method is better because the test outcome showed a tie (or a result close to a tie) in all three tables. Nevertheless, M-Estimate smoothing was identified as the best smoothing methods because of its overall robustness. M-Estimate smoothing was then compared with M-Branch smoothing. With very little difference between these two smoothing methods, M-Branch smoothing still had a small advantage, although the improvement was not comparable with the situation in the case of unpruned REPTree classifier. Finally, M-Branch smoothing, the best smoothing method up to this point on pruned trees, was compared to the PPM smoothing methods. According to the results, it was even more difficult to tell which method is the best of all, with all three test exhibiting ties. A further experiment would need to be done with a more extensive collection of datasets to distinguish the performance difference between the two methods. In a last experiment, a case study, the smoothing methods were compared on e-commerce data based on unpruned trees and ROC curves. The M-Branch method performed best in this experiment. This was consistent with previous results.

5.2 Future work

This thesis only concerns binary datasets. As described in the paper by Ferri, Flach & Hernández-Orallo (2003), M-Branch smoothing has better smoothing effects on multi-class datasets. It would be useful to identify a classifier that behaves similarly to the classifier in (Ferri, Flach & Hernández-Orallo, 2003) for multi-class data and compare the PPM smoothing methods with M-Branch smoothing on multi-class datasets. It would also be useful if an experiment with the MetaCost (Domingos, 1999) classifier were carried out, to compare the smoothing performance with cost-sensitive classification.

Appendix A
More experimental results for bagged trees and pruned trees vs unpruned trees

Dataset	No smoothing	No smoothing with bagging
monks-1	0.141±0.12	0.092 ± 0.03
monks-2	0.586±0.04	0.502 ± 0.02 ●
monks-3	0.122±0.04	0.112 ± 0.03
tic-tac-toe	0.351±0.03	0.251 ± 0.02 ●
mushroom	0.000±0.00	0.002 ± 0.00
breast-cancer	0.227±0.04	0.180 ± 0.03 ●
kr-vs-kp	0.073±0.02	0.071 ± 0.02
sonar	0.490±0.07	0.376 ± 0.04 ●
pima-diabetes	0.518±0.03	0.419 ± 0.02 ●
vote	0.191±0.03	0.178 ± 0.03
yeast	0.280±0.01	0.237 ± 0.01 ●
hepatitis	0.419±0.06	0.356 ± 0.04 ●
liver-disorders	0.559±0.05	0.454 ± 0.03 ●
spambase	0.263±0.01	0.211 ± 0.01 ●
ionosphere	0.311±0.05	0.254 ± 0.04 ●
sick	0.096±0.02	0.088 ± 0.01
spect	0.516±0.04	0.458 ± 0.03 ●

○, ● statistically significant differences

Table A.1: Root mean squared error with bagging

Dataset	No smoothing	No smoothing with bagging
monks-1	-21.551±27.38	0.918± 0.04
monks-2	-266.961±50.83	-4.281± 7.02 ○
monks-3	-9.504± 9.92	-6.694± 8.41
tic-tac-toe	-88.803±26.04	0.278± 1.92 ○
mushroom	0.999± 0.00	0.999± 0.00
breast-cancer	-35.925±18.79	-1.701± 5.09 ○
kr-vs-kp	-1.554± 2.47	0.252± 1.17
sonar	-196.767±73.39	-0.423± 4.45 ○
pima-diabetes	-216.485±41.15	-3.627± 5.89 ○
vote	-6.486± 9.13	-0.693± 4.03
yeast	-338.235±33.63	-47.908±13.70 ○
hepatitis	-49.837±48.98	-0.888± 5.93 ○
liver-disorders	-253.373±68.16	-3.808± 9.48 ○
spambase	-43.928± 7.82	0.504± 0.59 ○
ionosphere	-68.855±34.71	-2.616±10.94 ○
sick	-4.138± 2.75	-0.357± 1.20 ○
spect	-170.254±58.98	-7.341±13.57 ○

○, ● statistically significant differences

Table A.2: Entropy gain with bagging

Dataset	No smoothing	No smoothing with pruning
monks-1	0.141±0.12	0.196 ± 0.10
monks-2	0.586±0.04	0.495 ± 0.02 ●
monks-3	0.122±0.04	0.094 ± 0.05
tic-tac-toe	0.351±0.03	0.363 ± 0.03
mushroom	0.000±0.00	0.008 ± 0.02
breast-cancer	0.227±0.04	0.217 ± 0.03
kr-vs-kp	0.073±0.02	0.099 ± 0.02 ○
sonar	0.490±0.07	0.465 ± 0.05
pima-diabetes	0.518±0.03	0.434 ± 0.03 ●
vote	0.191±0.03	0.188 ± 0.04
yeast	0.280±0.01	0.245 ± 0.01 ●
hepatitis	0.419±0.06	0.394 ± 0.04
liver-disorders	0.559±0.05	0.495 ± 0.03 ●
spambase	0.263±0.01	0.261 ± 0.01
ionosphere	0.311±0.05	0.295 ± 0.04
sick	0.096±0.02	0.109 ± 0.01
spect	0.516±0.04	0.454 ± 0.03 ●

○, ● statistically significant differences

Table A.3: Root mean squared error with pruning

Dataset	No smoothing	No smoothing with pruning
monks-1	-21.551±27.38	-0.932± 6.22
monks-2	-266.961±50.83	-18.857±19.62 ○
monks-3	-9.504± 9.92	-4.858± 6.75
tic-tac-toe	-88.803±26.04	-17.553±11.41 ○
mushroom	0.999± 0.00	0.998± 0.00
breast-cancer	-35.925±18.79	-3.637± 7.33 ○
kr-vs-kp	-1.554± 2.47	0.256± 1.09
sonar	-196.767±73.39	-17.730±29.55 ○
pima-diabetes	-216.485±41.15	-12.979±13.81 ○
vote	-6.486± 9.13	0.382± 2.10
yeast	-338.235±33.63	-62.601±19.92 ○
hepatitis	-49.837±48.98	-4.462±16.75
liver-disorders	-253.373±68.16	-25.159±28.87 ○
spambase	-43.928± 7.82	-4.914± 3.55 ○
ionosphere	-68.855±34.71	-5.153±10.59 ○
sick	-4.138± 2.75	-0.608± 1.24 ○
spect	-170.254±58.98	-7.319±16.79 ○

○, ● statistically significant differences

Table A.4: Entropy gain with pruning

Bibliography

Bell, T., Cleary, J. & Witten, I. (1990). *Text compression.* Prentice Hall advanced reference series: Computer science. Prentice Hall.

Breiman, L. (1996). Bagging predictors. In *Machine Learning* (pp. 123–140).

Cleary, J. G. & Witten, I. H. (1984). Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, *32*, 396–402.

Domingos, P. (1999). Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 155–164).

Dzeroski, S., Cestnik, B. & Petrovski, I. (1993). Using the m-estimate in rule induction. *Journal of Computing and Information Technology*, *1(1)*, 37–46.

Ferri, C., Flach, P. & Hernández-Orallo, J. (2003). Improving the AUC of probabilistic estimation trees. *Machine Learning: ECML 2003* (pp. 121–132).

Freund, Y. & Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *the Thirteenth International Conference on Machine Learning* (pp. 148–156).

Green, D. M. & Swets, J. A. (1966). *Signal detection theory and psychophysics.* John Wiley and Sons Inc.

Howard, P. G. (1993). *The Design and Analysis of Efficient Lossless Data Compression Systems.* PhD thesis, Brown University.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *The Fourteenth International Joint Conference on Artificial Intelligence* (pp. 1137–1143). Morgan Kaufmann.

Ling, C., Huang, J. & Zhang, H. (2003). AUC: a statistically consistent and more discriminating measure than accuracy. In *International Joint Conference on Artificial Intelligence*, Volume 18 (pp. 519–526).

Moffat, A. (1988). A note on the ppm data compression algorithm. Research Report 88/7, Department of Computer Science, University of Melbourne, Parkville, Victoria, Australia.

Nadeau, C. & Bengio, Y. (2001). Inference for the generalization error. *Machine Learning*.

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

Quinlan, J. (1986). Induction of decision trees. *Machine learning*, *1(1)*, 81–106.

Quinlan, J. R. (1987). Simplifying decision trees. *International Journal of Man-Machine Studies*, *27(3)*, 221–234.

Roberts, M. G. (1982). *Local-order-estimating Markovian analysis for noiseless source coding and authorship identification*. PhD thesis, Stanford Univerisity.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, *3(3)*, 211–229.

Witten, I. H. & Bell, T. J. (1991). The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, *37(4)*, 1085–1094.

Witten, I. H. & Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques* (2 Ed.). San Francisco: Morgan Kaufmann.