UNIVERSIDAD AUTÓNOMA DE MADRID

Dynamic & Static Pruning Techniques for Classification Ensembles

by

Víctor Soto Martínez

A thesis submitted in partial fulfillment for the
degree of Masters of Science

in the
Escuela Politécnica Superior
Departamento de Ingeniería Informática

May 2011

"I have not failed. I've just found 10,000 ways that won't work."

Thomas Edison

UNIVERSIDAD AUTÓNOMA DE MADRID

Abstract

Escuela Politécnica Superior
Departamento de Ingeniería Informática

Masters of Science

by Víctor Soto Martínez

In ensemble learning the outputs of several diverse classifiers are combined into a unified decision. It has been proved in the literature that using this scheme of classification provides both good accuracy and robustness to the classification. However, ensemble learning has expensive requirements: it needs to keep every classifier in memory in order to be able to query them, and all the classifiers need to be queried to output a decision. In order to overcome these drawbacks, ensemble pruning techniques have been designed. Usually ensemble pruning techniques are categorized as static and dynamic techniques. Static techniques select a subensemble of classifiers that optimize a given heuristic, and discard the rest. Dynamic techniques sequentially query the classifiers and halt the process when a confidence-interval measure reaches a preset value.

In this Master's Thesis two contributions are made to the field of ensemble pruning techniques. First, a double pruning scheme is presented, that reduces the memory requirements of the ensembles and ameliorates the classification process by combining preexisting static and dynamic pruning techniques from the literature. Second, an updated version of the *Statistical Instance-Based Pruning* method is proposed. The new formulation of the method allows us to include specific knowledge about the classification problems, thus better fitting the disagreement between the full ensemble and its dynamically pruned counterpart, and further increasing the speed-up rates.

Acknowledgements

A huge thank you goes to my Thesis Director Gonzalo Martínez. Thanks for your guidance and insight before and during the making of this thesis, your suggestions were greatly appreciated. Without your unlimited patience and proofreading superpowers this work would have not been possible, much less readable. It has been great working with you.

Another heartfelt gratitude goes to my Master's Advisor Alberto Suárez. Thanks Alberto, for introducing me to the Machine Learning world, a world that has hooked me up inevitably by now. The long hours in your office discussing theorems and proofs about boosting will always be remembered by me. Your perseverance in making me understand the last tidbit of mathematical soundness knows no match.

I would like to thank all my friends, whom I will not list here for fear of missing one out: thank you! But foremost, thank you Bea and Dani. I am sure there are a thousand ways to say this, but only one feels right: you are the best friends anyone could have. Thank you for asking, caring, and supporting me throughout this period of exams, papers, interviews, fellowships, applications, ups and downs.

Finally but not less important, I want to thank my family: Alex, Lore, Mum and Dad. Thank you for being the last remnant of peace, for providing me with a space where neither Master's Thesis nor jobs exist. Your continuous efforts for being the most surreal and lovely family anyone can dream of are priceless. Thanks!

Contents

List of Figures

List of Tables

Symbols

$\mathcal{X}$	Feature space
$\mathbf{x}$	feature vector from the Feature space
$\mathcal{Y}$	Label space
M	dimensionality of the Feature Space
$(\mathbf{x}_i, y_i)$	instance of the dataset
N	number of instances in the dataset
l	cardinality of the Label space
h	singular classifier or hypothesis
H	ensemble classifier
$\hat{H}$	pruned ensemble classifier
T	size of the classification ensemble
t	size of the subensemble
$\mathbf{T}$	vector of votes of the full ensemble
$\mathbf{t}$	vector of votes of the subensemble

To all of you who helped me through this graduate school madness

Chapter 1

Introduction

The goal of supervised machine learning is to build an autonomous prediction system h, also known as classifier or hypothesis, by means of an induction process using a training dataset $\mathcal{Z}_{train}$. The training dataset is a collection of examples formed by pairs $(\mathbf{x}, y)$ where is $\mathbf{x} \in \mathcal{X}$ is a feature vector of independent variables and $y \in \mathcal{Y}$ is its class label or dependent variable. The prediction task consists in labeling each example with a class based only on its feature vector.

$$\begin{array}{rcl} h : \mathcal{X} & \longrightarrow & \mathcal{Y} \\ \mathbf{x} & \mapsto & \mathbf{y} \end{array}$$

When the class labels space $\mathcal{Y}$ is a discrete space where no order is established the prediction task is called *classification*, whereas when the class label is real valued the task is called *regression*. The quality of the prediction system is measured in its generalization performance capacity, i.e the accuracy achieved on a separate test dataset $\mathcal{Z}_{test}$ generated from the same training distribution.

$$accuracy = \frac{1}{\|\mathcal{Z}_{test}\|} \sum_{(\mathbf{x},y)\in\mathcal{Z}_{test}} \mathbb{I}(h(x) = y) \tag{1.1}$$

Ensemble learning is a branch of supervised learning concerned with combining multiple classifiers in order to generate a single, more accurate classifier [1–5]. Ensemble classifiers combine the output of every single classifier and output a unified prediction. There is extensive evidence in the literature that shows that combining multiple predictions is a good mechanism to improve generalization performance and improve the robustness of the prediction [6–8]. Specifically, combining diverse classifiers (those whose errors are independent and mostly complementary) further improves the performance of the final

aggregated classifier, since the errors made by some hypothesis are corrected by other hypothesis.

The prediction of the ensemble can be computed using different rules. The most common rule is the majority rule, where the most voted class is chosen. Supposing the ensemble has T classifiers, the rule is given by the following formula

$$y = \arg\max_i \sum_{t=1}^{T} \mathbb{I}(h_t(\mathbf{x}) = y_i) \tag{1.2}$$

An extensive analysis on this rule can be found in [1].

The biggest drawback of ensemble learning algorithms is the the elevated cost of space and time complexity: several classifiers have to be allocated in memory and be queried in order to output a final prediction. In order to ameliorate these costs, many ensemble pruning techniques have been proposed (see section 2.2). These ensemble pruning techniques can be separated in two different groups: static and dynamic techniques. Static techniques (or off-line techniques) focus in selecting a subset of classifiers of fixed size that improves the accuracy with respect to the full ensemble, discarding the rest of them. Dynamic pruning techniques, on the other hand, estimate the number of classifiers needed to obtain the final decision for each specific instance during the classification process. Dynamic pruning techniques (or on-line techniques) speed up the classification process but all the classifiers need to be stored in memory.

In this work two different contributions are made:

1. A double pruning algorithm is proposed. This algorithm consists in first reducing the set of classifiers by applying a static method, which ensures that both the accuracy and speed-up are improved. Then a dynamic method is used during the classification process, which furthers accelerates the classification process. Two variants of the double pruning algorithm are given for ensemble algorithms *Bagging* and *Adaboost*.

2. A variant of the dynamic pruning method *Statistical Instance-Based* is shown. The variation incorporates prior knowledge about the specific classification problem. It is shown that the new method improves the speed-up rates of the original ensemble.

The rest of the work is organized as follows. Chapter 2 reviews the state of the art of ensemble methods and static and dynamic pruning techniques, the main focus of the thesis. It also contains an extensive analysis on the SIBP method, including the involved mathematical proofs. Chapter 3 contains the first contribution of this thesis, the double

pruning technique. Chapter 4 presents a thorough analysis of the new updated SIBP method. Finally in chapter 5 the global conclusions of this thesis are expounded.

Chapter 2

Classification Ensembles and Pruning Techniques

In this chapter a revision of the vast state of the art on classification ensemble methods and its pruning techniques is given. Mainly the methods used throughout this work are explained, or those the author judged interesting or necessary for the reader. The first section contains the review of three most important ensemble algorithms. The second section offers a review of key ensemble pruning techniques. Finally, the third section presents the experiments performed over several benchmark classification problems using the SIBP method. These results will be relevant for Chapters 3 and 4.

2.1 Classification Ensembles

A condition both necessary and sufficient to build classification ensembles is if the hypothesis are accurate and diverse [1]. A hypothesis is accurate if its accuracy is better than random guessing, i.e higher than 50%. A set of ensembles are diverse if they make errors in different instances, i.e their errors are independent. Ensemble construction techniques can be categorized as [6]: enumerating the hypothesis, manipulating the training dataset, manipulating the input features, manipulating the output targets and injecting randomness. In this work, all the tested ensembles are built by manipulating the training dataset. Using this approach, the diverse hypothesis are built creating a different training set sampled from the original set. Random Forests also uses the *manipulating the input vectors approach* by selecting a reduced number of features in each split node of its tree classifiers.

In the next three sections we review the three most important ensemble algorithms, also used in this work.

2.1.1 Bagging

Bagging (**B**ootstrap **Agg**regat**ing**) [2] is an ensemble learning algorithm that creates diverse classifiers from the same dataset and aggregates their output to create a single predictor. The method creates an ensemble of T classifiers by creating T different bootstrap learning sets $\mathcal{Z}^t_{train}$ $t \in [1, T]$. Given a learning set of size N, bootstrap samples uniformly and with replacement N examples from the learning dataset. As a direct consequence of sampling with replacement, each bootstrap dataset will contain repeated examples and will miss some examples. Specifically, each bootstrap dataset will contain a fraction close to 0.632 of the N examples.

$$P(\mathbf{x}_i \notin \mathcal{Z}^t_{train}) = \left(\frac{N-1}{N}\right)^N = e^{N \ln \frac{N-1}{N}} \xrightarrow{N\to\infty} e^{N(\frac{N-1}{N}-1)} = e^{-1} = 0.368 \tag{2.1}$$

where it has been used the inequality $\lim_{x\to 1} \ln x = x - 1$. The examples that are not included in the bootstrap sampling (a fraction $1/e$ of the total) are called *out-of-bag* instances, and are used to compute estimates of the ensembles [9]. The bagging pseudocode can be observed in Algorithm 1.

Algorithm 1 Bagging Pseudocode

Require: A training set $E = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ and *Learn* algorithm

Ensure: Classification ensemble $\{h_t\}_{t=1}^T$

1: **for** $t = 1 \to T$ **do**
2: $E_t = BootStrapSample(E)$
3: $h_t \leftarrow Learn(E_t)$
4: **end for**
5: **return** Ensemble classifier given by the expression $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^T \alpha_t h_t \mathbf{x})\right)$

Once the T classifiers have been built, the aggregated predictor outputs classification decisions using the majority voting scheme rule

$$y = \arg\max_i \sum_{t=1}^{T} \mathbb{I}(h_t(\mathbf{x}) = y_i) \tag{2.2}$$

Bagging needs unstable predictors to work properly. An unstable prediction algorithm is one that generates very different (diverse) classifiers if the same training data is used

with small perturbations. It has been studied that neural networks and decision trees are unstable learning algorithms, while nearest neighbors methods are very stable [10].

2.1.2 Boosting

The state of the art of boosting is very extensive. In this section only a brief review is given following a chronological perspective. The first boosting algorithm comes up as an answer to a question posed in [11] within the PAC (probably approximately correct) learning framework: "A weak learning algorithm is given, capable of generating hypothesis with an accuracy better than random guessing for any distribution over the instances space. Does the existence of such weak algorithm imply the existence of a strong algorithm capable of generating classifiers of arbitrarily high accuracy?". The answer by *Robert E. Schapire* is affirmative [12], proving that it is possible to join the decision of several weak learners and form an aggregated strong classifier in polynomial time. Later, a new algorithm that works in the majority voting framework is proposed by *Yoav Freund* in [13]. Both algorithms were characterized by keeping a probability distribution over the training dataset and updating that distribution in each iteration in a way that often misclassified examples have more probability that correctly classified examples. Using this distribution to generate the training datasets of each hypothesis, it is more likely that successive classifiers will be able to correctly classify examples that were misclassified by the previous hypothesis.

Up to that point, the existent boosting algorithms had the undesirable characteristic of not being *adaptive*, that is, they needed an input parameter $\gamma \in (0, 1/2)$ that indicated the minimum accuracy $1/2 + \gamma$ required to each hypothesis. Hypothesis that failed to achieve a minimum accuracy, were discarded and not used in the ensemble classifier. Adaboost (**Ada**ptive **Boost**ing) [14, 15] was the first boosting algorithm to overcome this drawback. Unlike its predecessors, Adaboost used all the weak hypothesis generated during its training phase, being almost as efficient as boosting-by-majority. Due to its efficiency and simplicity, Adaboost quickly gained fame between the researchers of the field. Adaboost proved to be extremely efficient in driving the training error to zero, but more importantly, it proved to continue reducing the generalization error long after the training error was zero [16]. The Adaboost pseudocode is given in Algorithm 2.

There are two main implementations of boosting: boosting-by-resampling and boosting-by-reweighting. In boosting-by-resampling each time the weak learning algorithm is called, a resampled (with replacement) training set is fed directly to the algorithm. However, in boosting-by-reweighting the base learning algorithm is capable to handle the

weights over the examples space and both the probability distribution and the complete training set is passed.

Both bagging and boosting by resampling creates new training sets for each hypothesis by resampling the original training data. However, whereas the resampling performed in boosting is uniform over the space of instances and independently done for each classifier, in boosting the resampling follows a given probability distribution that is very dependent from the previous ones.

Breiman proposed in [16] a general term for adaptive boosting algorithms under the name of Arcing algorithms (**A**daptively **R**esample and **C**ombin**ing**) and gave a set of directions under which the arcing algorithms were constructed. Breiman's goal was to provide an analysis framework capable of explaining why Adaboost kept reducing the generalization error after terminating the training error. In the same work, Breiman conducts an analysis on arcing algorithms in which the generalization error is decomposed into a Bayes error term (the minimum misclassification rate), a bias term and a variance term. The definitions of both terms varies greatly depending on the author, but the concept remains the same: the bias is defined as the error directly related to the learning algorithm, and the variance is the error related to the fluctuations of generating single classifiers.

An alternative framework for analysis of arcing algorithms is proposed in [17], where the authors claim that the key to the boosting algorithms effectiveness lies not in reducing the training error to zero, but in maximizing the margins of the examples. The margin of an ensemble classifier over an example is defined as

$$margin(\mathbf{x}, y) = \sum_{t=1}^{T} w_t y h_t(\mathbf{x}) \tag{2.3}$$

where $w_t > 0$ is the weight associated to hypothesis h_t, such that $\sum_{t=1}^{T} w_t = 1$.

2.1.3 Random Forests

Random Forests [4] is an ensemble learning algorithm for decision trees. Starting from the same idea of bagging, Random Forests creates a bootstrap dataset to train each base classifier. The main difference between bagging and random forest is that during the training of the decision trees, the latter chooses randomly in each node m variables to calculate the best split. Usually the number of selected variables follows the identity $m = \log_2(M + 1)$, where M is the dimensionality of the feature space.

Algorithm 2 Adaboost Pseudocode

Require: A training set $E = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$ and $WeakLearn$ algorithm

Ensure: Classification ensemble formed by $\{(\alpha_t, h_t)\}_{t=1}^{T}$

1: $w_1(i) = 1/m$
2: **for** $t = 1 \rightarrow T$ **do**
3: $\quad h_t \leftarrow WeakLearn(S, w_t)$
4: $\quad \epsilon_t \leftarrow \sum_{i=1}^{m} w_t(i)\mathbb{I}(h_t(\mathbf{x}_i) \neq y_i)$
5: $\quad \alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
6: $\quad$ **for** $i = 1 \rightarrow m$ **do**
7: $\quad\quad w_{t+1}(i) = \frac{w_t(i)\exp\left(-\alpha_t y_i h_t(\mathbf{x}_i)\right)}{Z_t}$
8: $\quad$ **end for**
9: **end for**
10: **return** Ensemble classifier given by the expression $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T}\alpha_t h_t(\mathbf{x})\right)$

2.2 Pruning Techniques for Classification Ensembles

There is extensive empirical evidence that combining the predictions of complementary classifiers is a successful strategy to build robust classification systems with good generalization performance [6–8]. The main disadvantages of ensemble methods are the difficulties in the interpretation of the decisions of the ensemble and their large computational requirements. In particular, the training cost, the storage needs and the time of prediction increase linearly with the number of classifiers that are included in the ensemble. To alleviate these shortcomings, different ensemble pruning methods can be used [18–27]. The goal of these methods is to reduce the memory requirements and to speed-up the classification process while maintaining or, if possible, improving the accuracy of the original ensemble. Besides needing less storage space and predicting faster, pruned subensembles can actually outperform the original classification ensembles from which they are extracted [19–22, 24].

The pruning methods for classification ensembles are clustered into two different groups: static and dynamic (also known as off-line and on-line respectively). The goal of static pruning methods is to select a subensemble of classifiers that improves or maintains the generalization performance while accelerating the classification process with respect to the full ensemble [18, 23, 24]. Once the subensemble is identified, the rest of classifiers are discarded and only the subsensemble is stored in memory. Thus, memory storage needs is ameliorated and the classification stage is sped-up. Additionally, if the selected

classifiers make complementary errors, the generalization performance can be improved [24]. Static pruning ensemble methods have been successfully applied to both parallel ensembles [24] and sequential ensembles [18, 23].

Dynamic methods estimate the prediction of the full ensemble based on the results of a few queried classifiers. Given a high level of confidence α, all the classifiers are queried until the probability that both the full ensemble and the queried subensemble predictions is the same is above α [25]. Unlike static methods, dynamic methods need to store all the classifiers and thus do not alleviate memory storage needs. On the other hand they further improve the speed-up rates, which makes them suitable for on-line applications. Dynamic pruning ensemble methods have been applied only to parallel ensemble, given that its theoretical analysis assumes that the classifiers are built independently when conditioned to the training data.

2.2.1 Static Pruning Techniques

A possible approach to ensemble pruning is to select from the original ensemble a subset of representative classifiers whose combined performance is equivalent or better than the complete ensemble. A handicap is that the selection of classifiers has to be based on estimates on the training data. However, the objective is to identify a subensemble that has good generalization performance. Even if we can compute accurate estimates of the generalization accuracy on the basis of the training data only, finding the optimal subensemble is a computationally expensive problem that involves comparing all the possible 2^T-1 non-empty subensembles that can be extracted from the original ensemble.

An important subset of static pruning techniques are those called ordered aggregation techniques. These techniques use a greedy strategy based on modifying the order of the original ensemble. The greedy search starts from an initial subset of classifiers of the complete pool E_T and adds in each iteration the classifier that optimizes a given heuristic measure. From the subensemble $\mathcal{S}_{u-1}$ of size $u-1$, the subensemble of size $\mathcal{S}_u$ is constructed by incorporating a single classifier s_u from the remaining pool of classifiers $E_T \setminus S_{u-1}$. The result is a sequence of ordered hypothesis that can be pruned by selecting the first k classifiers. An exhaustive analysis of order aggregation techniques can be found in [24]. Next we review some of the most notable static pruning techniques in the literature. For each method, its own ordered aggregation rule is given.

2.2.1.1 Early Stopping

The early stopping or fix-rate pruning selects the first t hypothesis produced by the ensemble learning algorithm and discards the rest. Therefore, the $u-th$ selected classifier will be the same classifier in that position in the original ensemble

$$s_u = h_u \tag{2.4}$$

2.2.1.2 KL-divergence Pruning

Based on the assumption that a good indicator of the generalization performance of an ensemble is the diversity between its classifiers, this technique maximizes the diversity of the selected subensemble using the Kullback-Leibler divergence of the probability distributions over the training data. The KL-divergence $KL(p||q)$ measures how different two probability distributions p and q are. It is not a proper distance, as it does not satisfy the symmetrical and transitive properties, but it is often used as such in the literature. The definitions, for both discrete and continuous distributions are

$$KL(p||q) = \sum_{n=1}^{N} p(x_n) \log \frac{p(x_n)}{q(x_n)} \qquad KL(p||q) = \int_{\mathcal{X}} p(x) \log \frac{p(x)}{q(x)} dx \tag{2.5}$$

This technique initializes the subset of classifiers with the first hypothesis of the ensemble. Starting from a subensemble of size $u-1$, it creates the subensemble of size u by adding the classifier that maximizes the sum of the pairwise distances of all the classifiers. We make an abuse of notation and define the KL divergence between two classifiers $\tilde{KL}(h_i||h_j)$ as the KL divergence between the probability distributions over their training sets. The ordering rule follows:

$$s_u = \arg\min_k \sum_{i=1}^{u-1} \tilde{KL}(s_i||h_k) \tag{2.6}$$

2.2.1.3 Kappa Pruning

This technique chooses the classifiers that will form the pruned ensemble based on their pairwise diversity. In order to do so, it uses the κ statistic. The κ coefficient of a pair of classifiers h_α and h_β, is computed using the estimated probability that the classifiers coincide in the classification of an instance, Θ_1, and the estimated probability that the classifiers coincide by chance in the classification, Θ_2. The definition of both statistics

is

$$\Theta_1 = \frac{1}{m}\sum_{i}^{l} C_{ii} \qquad \Theta_2 = \sum_{i=1}^{l}\left(\frac{1}{m}\sum_{j}^{l} C_{ij}\right)\left(\frac{1}{m}\sum_{j}^{l} C_{ji}\right) \tag{2.7}$$

where C_{ij} is the number of instances in the training dataset for which $h_\alpha = y_i$ and $h_\beta = y_j$. The kappa statistic measures the agreement between the classifiers as

$$\kappa = \frac{\Theta_1 - \Theta_2}{1 - \Theta_2} \tag{2.8}$$

The κ statistic is interpreted as follows. If $\kappa = 1$ then the two classifiers are identical on their decisions and thus not diverse at all. If $\kappa = 0$ then their agreement rate is the same as their agreement expected by chance. Finally κ takes negative values when the agreement is lower than the expected by chance. This technique incorporates pairs of classifiers to the subensemble with minimum value of *kappa*, until t hypothesis have been selected:

$$s_u = \arg\min_k \kappa(h_k, H_{S_{u-1}}) \tag{2.9}$$

2.2.1.4 Boosting-Based Ordering

The boosting-based ordering [28] uses the weights updates of boosting on bagging ensembles. Given a bagging ensemble of size T, the boosting-based ordering starts by initializing the weights over the training dataset uniformly. In each iteration the method chooses the classifiers with the lowest weighted error over the training set from the pool of classifiers and updates the weights over the dataset according to the Adaboost rules. The weighted error of a classifier over the training dataset is given by

$$\epsilon_t = \frac{1}{N}\sum_{n=1}^{N} w_n \cdot \mathbb{I}(h_t(\mathbf{x}_n) \neq y_n) \tag{2.10}$$

and the Adaboost weight updates follow

$$w_i = \begin{cases} \frac{w_i}{2\epsilon_t} & \text{if } h_t(\mathbf{x}_i) \neq y_i \\ \frac{w_i}{2(1-\epsilon_t)} & \text{if } h_t(\mathbf{x}_i) = y_i \end{cases} \tag{2.11}$$

The ordering rule of Boosting-based Ordering is

$$s_u = \arg\min_k \epsilon(h_k) \tag{2.12}$$

2.2.1.5 Reduce-Error Pruning

The Reduce-Error pruning, based on the method of the same name for pruning decision trees, is introduced in [18]. This method, following a greedy strategy, iteratively adds to $\mathcal{S}_{u-1}$ the classifier h_u such that the voted combination of $\mathcal{S}_{u-1} \cup h_u$ minimizes the classification error. It is mentioned in [18] that better results can be obtained introducing a back-fitting stage after each addition to the pruned ensemble, although as noted in [29] this might not be necessarily true. The back-fitting stage consist in replacing one of the already added classifiers with one of the classifiers in the pool. If the classification rate is improved after the replacement, it tries to accomplish another replacement until the ensemble converges or a preset maximum number of iterations is reached. The ordering rule is given by

$$s_u = \arg\max_k \sum_{n=1}^{N} \mathbb{I}(H_{S_{u-1} \cup h_k}(\mathbf{x}_n) = y_n) \tag{2.13}$$

2.2.1.6 Complementary Measure Pruning

The Complementary Measure pruning technique [20] adds in each iteration the classifier that is most accurate over the set of misclassified instances of the current pruned ensemble. Initially the subensemble is initialized with the classifier with the lowest classification error, and iteratively incorporates the most diverse hypothesis. In this case the most diverse classifier is the one that correctly classifies the misclassified instances by the actual ordered ensemble. The ordering rule of the method follows

$$s_u = \arg\max_k \sum_{n=1}^{N} \mathbb{I}(h_k(\mathbf{x}_n) = y_n \ \& \ H_{S_{u-1}}(\mathbf{x}_n) \neq y_n) \tag{2.14}$$

2.2.1.7 Margin Distance Minimization

This method, first published in [20], uses the margin vector $c^t = (c_1^t, c_2^t, \ldots, c_N^t)$ of each classifier to create the pruned ensemble. The margin of the classifier h_t over the i-th example is defined as

$$c_n^t = 2\mathbb{I}(h_t(\mathbf{x}_n) = y_n) - 1 \tag{2.15}$$

In order to build the ensemble, the method iteratively adds the classifier that minimizes the distance between the aggregated margin of the pruned ensemble and a randomly chosen point $\mathbf{o}$ in the same space whose coordinates are all positive but close to zero $o_i = p \ p \in (0.05, 0.25)$, representing and ideal classifier that does not misclassify any

instance without being overly optimistic. The ordering rule of the method is

$$s_u = \arg\min_k d(\mathbf{o}, \frac{1}{T}(\mathbf{c}^k + \sum_{t=1}^{u-1} \mathbf{c}^t)) \tag{2.16}$$

The authors later proposed in [24] to update the point $\mathbf{o}$ as a function of the number of iteration, finding a good approximation $p(u) \propto \sqrt{u}$.

2.2.1.8 Orientation Ordering

This Orientation Ordering method, described in [29], sorts the classifiers of the ensemble by increasing order of the angles of the margin vector $\mathbf{c}^t$ and a reference vector $\mathbf{c}_{ref}$. In order to do so, the reference vector is chosen as to maximize the influence of the perfect classification performance (given by $\mathbf{o}$, which is any vector oriented along the diagonal of the first quadrant $\mathbf{o} = a \cdot \mathbf{e},\ a \in \mathbb{N}$ and $\mathbf{e}_i = 1\ \forall i$) over the margin vector of the full ensemble $\mathbf{c}_{ens}$:

$$\mathbf{c}_{ref} = \mathbf{o} + \lambda \mathbf{c}_{ens} \tag{2.17}$$

where

$$\mathbf{c}_{ens} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{c}^t \tag{2.18}$$

and λ is a constant such that $\mathbf{c}_{ens}$ and $\mathbf{c}_{ref}$ are orthogonal. The ordering rule of the method is given by

$$s_u = \arg\min_k \arccos\left(\frac{\mathbf{c}_{ref} \cdot \mathbf{c}^k}{\|\mathbf{c}_{ref}\|\|\mathbf{c}^k\|}\right) \tag{2.19}$$

2.2.1.9 Ensemble pruning via Semi-definite Programming

This technique, proposed in [23], formulates the ensemble pruning as a quadratic integer programming problem. The authors conjecture that the subensemble that achieves optimal values of some accuracy-diversity measure will also attain optimal generalization performance. Therefore the goal of the method is to find the subensemble of (fixed) size $t < T$ that minimizes the heuristic measure. The authors define *ad-hoc* the square matrix G of size T that contains both a measure of the accuracy of the individual classifiers and the pairwise diversity between classifiers. Specifically, the diagonal elements G_{ii} are the number of misclassified instances by the i-th hypothesis and elements $G_{ij}\ i \neq j$ is the number of common of misclassified instances by the i-th and j-th hypothesis.

The subset selection task is formulated as a quadratic programming problem to find a fixed size subset of classifiers that minimize the sum of the elements of the G matrix

$$\begin{aligned} \min_{x} \quad & x^T G x \\ \text{s.t.} \quad & x^T I x = \sum_{i}^{T} x_i = t \\ & x_i \in \{0, 1\} \end{aligned} \tag{2.20}$$

where $x_i = 1$ indicates that the i-th classifier belongs to the subset, and $x_i = 0$ the opposite. This problem is NP-hard in general, but it is very similar to the MC-k graph partitioning problem, which is known for having a very good approximate solution algorithm based on semi-definite programming (SDP). The original formulation of the MC-k problem is

$$\begin{aligned} \min_{y} \quad & y^T W y \\ \text{s.t.} \quad & \sum_{i}^{T} y_i = N_v - 2t \\ & y_i \in \{-1, 1\} \end{aligned} \tag{2.21}$$

where $W_{ii} = 0$, N_v is the number of vertices in the graph and t is the number of cuts needed to separate the graph. Both formulations are similar, except for the values of the variables x_i and y_i. The difference is fixed using the transformation $x_i = \frac{v_i+1}{2}$ where $v_i \in \{+1, -1\}$. The problem formulation becomes

$$\begin{aligned} \min_{v} \quad & \frac{1}{4}(v + e)^T G (v + e) \\ \text{s.t.} \quad & \frac{1}{4}(v + e)^T I (v + e) = t \\ & v_i \in \{-1, 1\} \end{aligned} \tag{2.22}$$

Another transformation is used to put the formulation back in quadratic form. In order to do so we extend the definition of vector v as $\tilde{v}^T = (1 \quad v)$ and define the new matrices H and D as

$$H = \begin{pmatrix} e^T G e & e^T G \\ G e & G \end{pmatrix} \qquad D = \begin{pmatrix} n & e^T \\ e & I \end{pmatrix} \tag{2.23}$$

The new problem (2.24) is equivalent to (2.22)

$$\begin{aligned} \min_{\tilde{v}} \quad & \tilde{v}^T H (\tilde{v}) \\ \text{s.t.} \quad & \tilde{v}^T D \tilde{v} = 4t \\ & \tilde{v}_i \in \{-1, 1\} \end{aligned} \tag{2.24}$$

Proof.

$$\begin{aligned}
\tilde{v}^T H \tilde{v} &= \begin{pmatrix} 1 & v^T \end{pmatrix} \begin{pmatrix} e^T G e & e^T G \\ G e & G \end{pmatrix} \begin{pmatrix} 1 \\ v \end{pmatrix} \\
&= \begin{pmatrix} e^T G e + v^T G e & e^T G + v^T G \end{pmatrix} \begin{pmatrix} 1 \\ v \end{pmatrix} \\
&= e^T G e + v^T G e + e^T G v + v^T G v \\
&= e^T G (e+v) + v^T G (e+v) = (e+v)^T G (e+v)
\end{aligned}$$

$$\begin{aligned}
\tilde{v}^T D \tilde{v} &= \begin{pmatrix} 1 & v^T \end{pmatrix} \begin{pmatrix} n & e^T \\ e & I \end{pmatrix} \begin{pmatrix} 1 \\ v \end{pmatrix} \\
&= \begin{pmatrix} n + v^T e & e^T + v^T I \end{pmatrix} \begin{pmatrix} 1 \\ v \end{pmatrix} \\
&= n + v^T e + e^T v + v^T I v = e^T I e + v^T I e + e^T I v + v^T I v \\
&= e^T I (e+v) + v^T I (e+v) = (e+v)^T I (e+v)
\end{aligned}$$

□

This formulation of the problem is equivalent to the MC-k partitioning problem. As mentioned before, the partitioning problem is NP-hard, and thus cannot be solved in polynomial time. But, the previous enunciation of the problem can be relaxed into a convex SDP problem. In order to do so, first notice that the matrix $V = vv^T$ if and only if $V \succeq 0$ and $\text{rank}(V) = 1$.

Proof. $\Rightarrow$

A matrix M is said to be positive-semidefinite if $x^T M x \geq 0 \; \forall x \in \mathbb{R}^n$. In our case, given $x \in \mathbb{R}^n$,

$$\begin{aligned}
x^T V x &= \begin{pmatrix} \sum_{i=1}^n x_i V_{i1} & \dots & \sum_{i=1}^n x_i V_{in} \end{pmatrix} x = \sum_{j=1}^n x_j \sum_{i=1}^n x_i V_{ij} \\
&= \sum_{j=1}^n x_j \sum_{i=1}^n x_i v_i v_j = \sum_{j=1}^n x_j v_j \sum_{i=1}^n x_i v_i = (\sum_{j=1}^n x_j v_j)^2 \geq 0
\end{aligned}$$

To demonstrate that rank$(V) = 1$, it is enough to observe that any row V_i of the matrix is linearly dependent to the others

$$\begin{aligned} V_i &= \begin{pmatrix} V_{i1} & V_{i2} & \dots & V_{in} \end{pmatrix} \\ &= \begin{pmatrix} v_i v_1 & v_i v_2 & \dots & v_i v_n \end{pmatrix} = v_i \begin{pmatrix} v_1 & v_2 & \dots & v_n \end{pmatrix} \end{aligned}$$

$\Leftarrow$ If $rank(V) = 1$ then all the rows of the matrix are linearly dependent of each other, for example the i-th is $V_i = \lambda_i \begin{pmatrix} v_1 & v_2 & \dots & v_n \end{pmatrix}$. Again, if V is positive-semidefinite, given any $x \in \mathbb{R}^n$, $x^T V x \geq 0$. In that case

$$\begin{aligned} x^T V x &= \begin{pmatrix} x_1 & x_2 & \dots & x_n \end{pmatrix} \begin{pmatrix} \lambda_1 v_1 & \lambda_1 v_2 & \dots & \lambda_1 v_n \\ \lambda_2 v_1 & \lambda_2 v_2 & \dots & \lambda_2 v_n \\ & & \ddots & \\ \lambda_n v_1 & \lambda_n v_2 & \dots & \lambda_n v_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \\ &= (\sum_{i=1}^{n} \lambda_i x_i) \begin{pmatrix} v_1 & v_2 & \dots & v_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \\ &= (\sum_{i=1}^{n} \lambda_i x_i)(\sum_{i=1}^{n} v_i x_i) \geq 0 \Rightarrow \lambda_i = v_i \forall i \Rightarrow V = v v^T \end{aligned}$$

□

In that case the optimization problem becomes

$$\begin{aligned} \min_{V} \quad & H \bullet V \\ \text{s.t.} \quad & D \bullet V = 4t \\ & diag(V) = e \\ & rank(V) = 1 \\ & V \succeq 0 \end{aligned} \tag{2.25}$$

The constraints $v_i \in \{+1, -1\}$ and $diag(V) = e$ are equivalent because the diagonal elements are $V_{ii} = v_i v_i = (\pm 1)^2 = 1$. The convex relaxation is obtained by dropping the constraint rank$(V) = 1$. Dropping this constraint is equivalent to allowing V to take real values, and not only values in ± 1. Once the problem is solved, it will be necessary to approximate a solution to the original values. The new formulation (2.26) is a convex

SDP problem that can be solved in polynomial time.

$$\begin{aligned} \min_{V} \quad & H \bullet V \\ \text{s.t.} \quad & D \bullet V = 4t \\ & diag(V) = e \\ & V \succeq 0 \end{aligned} \tag{2.26}$$

A greedy version of the algorithm can be attained by iteratively selecting the classifier s_u that, parting from an initial pool of classifiers S_{u-1} minimizes the value of 2.20 the most.

2.2.2 Dynamic Pruning Techniques

The dynamic pruning techniques focus on reducing the time devoted to make a prediction. In these techniques, the number of classifiers that need to be queried is computed for each instance. In order to do so, dynamic methods usually make estimations of the probability that the decision made by the first k hypothesis will coincide with the decision output by the full ensemble. When this estimation is above a confidence threshold, then the querying process is halted. Dynamic methods greatly improve the classification speed but do not reduce the storage requirements, because all classifiers need to be available during the classification process.

2.2.2.1 Dynamic Scheduling for Classification Ensembles

This method proposed in [30, 31] is formulated for cost-sensitive classification problems. Here only a schematic summary of the method is given. Let w_k the weight of hypothesis h_k and $T(\mathbf{x})$ the decision-threshold associated to instance $\mathbf{x}$. The objective of the ensemble is to detect if the label associated to the instance $\mathbf{x}$ is the class $y(\mathbf{x}) = y_0$ with the highest cost. Let $P_k(\mathbf{x})$ be the probability of a given instance being labeled as class y_0. For simplification purposes we suppose that the hypothesis of the ensemble are already sorted from highest to lowest weight, that is $w_k > w_{k+1}\ \forall k$.

The weighted probability computed by the k first queried hypothesis of the ensemble is given by

$$F_k(\mathbf{x}) = \frac{\sum_{i=1}^{k} w_i h_i(\mathbf{x})}{\sum_{i=1}^{k} w_i} \tag{2.27}$$

The goal of the method is to halt the querying process once there is enough confidence that the full ensemble would reach the same decision with respect to y_0. The weighted probability by the full ensemble $F_K(\mathbf{x})$ is unknown. To overcome this fact, the method

computes the probability distribution of the error over the training dataset, where the error of an instance is given by $\epsilon_k(\mathbf{x}) = F_k(\mathbf{x}) - F_K(\mathbf{x})$. The interval $[0, 1]$ is divided into ξ bins of the same length. If $F_k(\mathbf{x}) \in \left[\frac{i-1}{\xi}, \frac{i}{\xi}\right)$, then the statistics $\mu_{k,i}$ and $\sigma^2_{k,i}$, the mean and the variance of the error of the examples that have fallen in the i-th bin when using only the first k classifiers, are used in the following decision rule

$$\begin{cases} F_k(\mathbf{x}) - \mu_{k,i} - \beta \cdot \sigma_{k,i} > T(\mathbf{x}), \text{ then } y_0 \\ F_k(\mathbf{x}) + \mu_{k,i} + \beta \cdot \sigma_{k,i} \leq T(\mathbf{x}), \text{ then } y_1 \\ \text{else, uncertain} \end{cases} \tag{2.28}$$

where the parameter β is the confidence interval parameter. If the result at iteration k is uncertain, then the $k+1$-th hypothesis is queried and the process repeated.

2.2.2.2 Statistical Instance-based pruning

The usual ensemble classification scheme consists in querying (sequentially or in parallel) every hypothesis and then output the majority class. However it is not always necessary to query all the hypothesis to find the majority class. For example, in binary problems it is possible to output the class voted by at least the 50% of the ensemble. The equivalent rule for multiclass problems consists in labeling the most voted class when the difference in votes between the most voted class and the second most voted class is higher than the number of remaining hypothesis. In the following sections, we refer to this rule as ***full-confidence*** pruning rule ($\alpha = 1$). This pruning rule allows us to classify instances in the same way as the full ensemble without querying all the classifiers. However, it is also possible to halt the querying process if it is considered acceptable to classify an instance within a confidence interval $\alpha < 1$.

Before going any further, some notation will be introduced. Let T be the total number of hypothesis that form the classification ensemble. During the classification stage, each hypothesis outputs its predicted class $y = h(\mathbf{x})$ for instance $\mathbf{x}$. Assuming the ensemble is queried sequentially, after iteration $t < T$ we will have the following voting vector

$$\mathbf{t} = \{t_1, t_2, \ldots, t_l; \sum_{i=1}^{l} t_i = t\} \tag{2.29}$$

where t_i is the number of votes emitted to the class y_i by the first t hypothesis. The final voting vector $\mathbf{T}$ is

$$\mathbf{T} = \{T_1, T_2, \ldots, T_l; \sum_{i=1}^{l} T_i = T\} \tag{2.30}$$

where again T_i is the total number of votes assigned to class y_i by all T classifiers of the ensemble. The majority vote rule says that the label assigned to instance $\mathbf{x}$ is

$$y(\mathbf{x}) = \arg\max_i T_i(\mathbf{x}) \tag{2.31}$$

Therefore one way to define the pruning task will be to estimate vector $\mathbf{T}$ based on current $\mathbf{t}$ vector.

Statistical Instance-Based Pruning (SIBP) [25] is based in computing the probability that the label predicted by a queried subensemble of size t and the full ensemble of size T is the same with certainty α. For the computation of this probability it is only necessary to know: the number of hypothesis in the full ensemble T, the number of labels present in the classification problem l and the confidence interval α. Two important assumptions are made for the derivation: the hypothesis are independent and identically distributed when conditioned to the training data, and the a priori distribution of votes for the different classes is supposed to be equal. Following these assumptions, the authors formulate in a Bayesian framework the probability of obtaining the voting vector $\mathbf{T}$ after querying T classifiers given that a vector $\mathbf{t}$ has been observed after querying t classifiers.

$$\mathcal{P}(\mathbf{T}|\mathbf{t}) = \frac{(T-t)!}{\prod_{i=1}^{l}(T_i - t_i)!} \frac{\prod_{i=1}^{l}(t_i+1)_{T_i-t_i}}{(t+l)_{T-t}} \tag{2.32}$$

where $(a)_n = a(a+1)\cdots(a+n-1)$ is the Pochhammer symbol, or rising factorial, with a and n nonnegative integers. We show the proof to this result in the following Proposition.

Proposition 2.1. *Given a classification ensemble of size T of independent and identically distributed classifiers, the probability of obtaining the voting vector $\mathbf{T}$ given the vector $\mathbf{t}$ after querying the first t classifiers follows the formula*

$$\mathcal{P}(\mathbf{T}|\mathbf{t}) = \frac{(T-t)!}{\prod_{i=1}^{l}(T_i - t_i)!} \frac{\prod_{i=1}^{l}(t_i+1)_{T_i-t_i}}{(t+l)_{T-t}}$$

Proof. Let $\mathbf{p}(\mathbf{x})$ be the probability vector

$$\mathbf{p}(\mathbf{x}) = \{p_1(\mathbf{x}), p_2(\mathbf{x}), \ldots, p_l(\mathbf{x})\}, \qquad \sum_{i=1}^{l} p_i(\mathbf{x}) = 1 \tag{2.33}$$

where $p_i(\mathbf{x})$ is the probability of instance $\mathbf{x}$ is classified as class y_i by any hypothesis of the ensemble. At a given moment during the classification of the instance $\mathbf{x}$, t votes have been emitted. The distribution of the voting vector $\mathbf{t}$ restricted to $\mathbf{p}$ can be seen

as a multinomial distribution of parameters $(t_1+1, t_2+1, \ldots, t_l+1)$

$$\mathcal{P}(\mathbf{t}|\mathbf{p}) = \frac{t!}{t_1!t_2!\ldots t_l!}\prod_{i=1}^{l} p_i^{t_i} \tag{2.34}$$

Following the previous assumption that the a priori probability of the classes is equally likely, the prior distribution $\mathcal{P}(\mathbf{p})$ is uniform. Using the Bayes theorem we compute the posterior distribution

$$\mathcal{P}(\mathbf{p}|\mathbf{t}) = \frac{\mathcal{P}(\mathbf{t}|\mathbf{p})\mathcal{P}(\mathbf{p})}{\mathcal{P}(\mathbf{t})} = \frac{\Gamma(t+l)}{\prod_{i=1}^{l}\Gamma(t_i+1)}\prod_{i=1}^{l} p_i^{t_i} \tag{2.35}$$

where $\mathcal{P}(\mathbf{t})$ is the normalization constant

$$\mathcal{P}(\mathbf{t}) = \int_{\mathcal{D}} \mathcal{P}(\mathbf{t}|\mathbf{p})\mathcal{P}(\mathbf{p})d\mathbf{p} = \frac{t!}{t_1!t_2!\ldots t_l!}\int_{\mathcal{D}}\prod_{i=1}^{l} p_i^{t_i} d\mathbf{p} \tag{2.36}$$

$$= \frac{t!}{t_1!t_2!\ldots t_l!}\frac{\prod_{i=1}^{l}\Gamma(t_i+1)}{\Gamma(t+l)} \tag{2.37}$$

The posterior distribution $\mathcal{P}(\mathbf{p}|\mathbf{t})$ is a Dirichlet distribution of order l and parameters $(t_1+1, t_2+1, \ldots, t_l+1)$. Now we can compute the probability that given vector $\mathbf{t}$ we can obtain vector $\mathbf{T}$ when all the hypothesis have been queried

$$\begin{aligned}
\mathcal{P}(\mathbf{T}|\mathbf{t}) &= \int_{\mathcal{D}} \mathcal{P}(\mathbf{T}-\mathbf{t}|\mathbf{p})\mathcal{P}(\mathbf{p}|\mathbf{t})d\mathbf{p} \\
&= \frac{(T-t)!}{\prod_{i=1}^{l}(T_i-t_i)!}\frac{\Gamma(t+l)}{\prod_{i=1}^{l}\Gamma(t_i+1)}\int_{\mathcal{D}}\prod_{i=1}^{l} p_i^{T_i} d\mathbf{p} \\
&= \frac{(T-t)!}{\prod_{i=1}^{l}(T_i-t_i)!}\frac{\Gamma(t+l)}{\prod_{i=1}^{l}\Gamma(t_i+1)}\frac{\prod_{i=1}^{l}\Gamma(T_i+1)}{\Gamma(T+l)} \\
&= \frac{(T-t)!}{\prod_{i=1}^{l}(T_i-t_i)!}\frac{\prod_{i=1}^{l}(t_i+1)_{T_i-t_i}}{(t+l)_{T-t}}
\end{aligned}$$

□

Finally, the probability that the class label predicted by the full ensemble and the label predicted by the t first classifiers coincide is given by the expression

$$\mathcal{P}^*(\mathbf{t}, T) = \frac{(T-t)!}{(t+l)_{T-t}}\sum_{\mathbf{T}\in\mathcal{T}_\mathbf{t}}\frac{\prod_{i=1}^{l}(t_i+1)_{T_i-t_i}}{\prod_{i=1}^{l}(T_i-t_i)!} \tag{2.38}$$

where $\mathcal{T}_\mathbf{t}$ is the set of vectors $\mathbf{T}$ that obey the following restrictions:

1. $T_i \geq t_i \ \forall i$ and $\sum_{i=1}^{l} T_i = T$

2. Let $k_{\mathbf{t}}$ the majority class of $\mathbf{t}$, then $T_{k_{\mathbf{t}}} > T_j \ \forall j \neq k_{\mathbf{t}}$.

If $\mathcal{P}^*(\mathbf{t}, T) = 1$, then we have full confidence on the result of the final classification given by the ensemble and the querying process can be halted. However, if it is acceptable that, with a small probability $1 - \alpha$, the prediction of the partially polled ensemble and that of the complete ensemble disagree, the voting process can be stopped when the probability (2.38) exceeds the specified confidence level α. The final classification would be given as the combined decision of the polled classifiers only. In particular, the querying process can be halted after t classifiers have been queried, if the vector of class predictions of the current subensemble $\mathbf{t}$ is such that $\mathcal{P}^*(\mathbf{t}, T) \geq \alpha$.

2.3 Experiments on SIBP

Table 2.1 shows the results of applying the SIBP technique on several classification problems of the UCI Repository [32]. The experiments are performed on Random Forests Ensembles of $T = 101$ CART trees. The acceptance threshold is set on $\alpha = 0.99$. The table contains the following information: column 1 is the name of the problem, column 2 shows the test error of the full ensemble, columns 3 shows the test error of the pruned ensemble, column 4 shows the mean disagreement rate of between the full ensemble and the pruned ensemble, column 5 and 6 display the number of queried trees by the full ensemble and the pruned ensemble respectively, and finally column 7 contains the mean speed-up rate of the SIBP technique with respect to the *full-confidence* pruning rule ($\alpha = 1$). The mean error rate (MER) is given by the formula

$$MER = \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}(\mathcal{H}(\mathbf{x}_n) == y_n) \tag{2.39}$$

The mean disagreement rate (MDR) between the ensemble $\mathcal{H}$ and the SIB-pruned one $\hat{\mathcal{H}}$ is computed as

$$MDR = \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}(\mathcal{H}(\mathbf{x}_n) == \hat{\mathcal{H}}(\mathbf{x}_n)) \tag{2.40}$$

The numbers show significant speed-up rates, from 4.4 to 8.5. Notice that the speed-up rate is computed with respect to the *full-confidence* ($\alpha = 1$) pruning rule mentioned before, and not with respect to the 101 trees of the ensemble. In the problems investigated, the error rate of the pruned ensemble is very similar to the error of the full ensemble in all the analyzed problems The differences range from 0.1 in favor of SIBP in *echocardiogram* to 0.2 in favor of full ensemble in *german*, *sonar* and *threenorm*. In relation to the disagreement rates, it should be expected that the disagreement rates are

	Test error			# trees IB		
Problem	RF	IB-RF101	Dis-rate	$\alpha = 1$	$\alpha = 0.99$	Speed-Up
australian	13.0±3.7	13.1±3.7	0.3±0.6	62.2±1.4	16.1±2.1	6.6±0.4
breast	3.2±2.1	3.2±2.1	0.1±0.4	54.2±0.9	8.9±1.4	8.0±0.2
diabetes	24.3±4.2	24.3±4.1	0.6±0.9	68.8±1.8	24.9±3.2	5.6±0.4
echocardiogram	22.2±14.3	22.1±14.7	0.7±3.1	68.0±4.6	22.6±8.2	5.7±1.1
german	23.4±3.5	23.6±3.3	0.8±0.8	71.8±1.3	28.4±2.8	5.1±0.4
heart	18.3±6.9	18.4±7.0	0.8±1.8	67.2±2.5	22.5±4.2	5.8±0.6
liver	27.1±6.7	27.1±7.0	1.0±1.7	74.5±2.3	31.8±4.5	4.6±0.6
mushroom	0.0±0.0	0.0±0.0	0.0±0.0	51.0±0.0	6.0±0.0	8.5±0.0
ringnorm	7.6±1.3	7.7±1.2	0.5±0.2	68.6±0.8	22.9±1.1	5.5±0.3
sonar	16.3±8.7	16.5±8.7	0.9±2.0	73.9±3.0	32.1±6.6	4.7±0.7
threenorm	17.8±1.1	18.0±1.1	1.0±0.2	76.6±0.5	34.8±1.0	4.4±0.2
twonorm	4.7±0.6	4.8±0.6	0.4±0.1	67.2±0.2	21.0±0.5	5.8±0.1
votes	4.1±2.9	4.1±2.9	0.1±0.4	54.5±1.2	8.8±1.8	7.9±0.3

TABLE 2.1: Results of applying the SIBP technique to a Random Forest ensemble of 101 trees

close to the theoretical bound $1 - \alpha$, in this case 1%. However, most of them are much lower than 1%. Depending on the problem from 0.0 for *mushroom* to 1.0 for *liver* and *threenorm*. Although this might seem a positive feature, increasing the disagreement rate would help to better find the balance position between error and speed-up rates. We conjecture that this happens because no a priori knowledge is incorporated into the SIBP technique. In Chapter 4, we propose an update into the SIBP technique that does not assume a uniform prior distribution, and instead models different prior distributions using the training data.

Chapter 3

Double Pruning Techniques For Classification Ensembles

3.1 Introduction

The theoretical analysis of majority voting on which SIBP is grounded relies on the fact that, in parallel ensembles, the individual classifiers are generated under the same conditions and independently of each other. When the ensemble is sequential, the classifier that is added at one point in the sequence depends on the classifiers that have been included in the ensemble up to that point. As a result, correlations among classifiers are introduced, which can result in biases in the estimation of the final ensemble prediction on the basis of the outputs of the initial classifiers in the sequence.

The goal of this chapter is to determine whether SIBP can be also used in sequential ensembles. Specifically, SIBP is applied to two types of sequential ensembles: ordered bagging ensembles and static-pruned boosting ensembles. This results in a double pruning method that combines the advantages of static and dynamic pruning. Namely improved generalization performance, reduced storage requirements because only classifiers in the pruned ensemble need to be stored in memory, and improved speed-up rates during classification process.

The results of experiments on several benchmark classification problems carried out in this investigation show that the biases introduced by SIBP can cause some distortions in the estimation of the error rate of the complete ensemble when the classifiers of the ensemble are not independently generated. By contrast, SIBP is remarkably effective when it is used to halt the aggregation process in a previously pruned sub-ensemble.

Previous sections 2.2.1.4, 2.2.1.9 and 2.2.2.2 provide reviews of the Boosting-Based, SDP and SIBP pruning algorithms respectively.

This chapter is organized as follows. Section 3.2 explains the methodology of the double pruning technique, section 3.3 summarizes the results of experiments on benchmark classification tasks and demonstrates the effectiveness of the double pruning algorithm proposed and finally the conclusions of this chapter are exposed in section 3.4.

Part of the work presented in this Chapter was published in [33].

3.2 Double pruning applied to sequential ensembles

3.2.1 Double pruning in Ordered Bagging

In ordered bagging ensembles it is generally observed that the curves that trace the dependence of the error rate on the size of the ordered ensemble exhibit a minimum at intermediate ensemble sizes. This induces us to think that classifiers included at the beginning and the end in the ordered bagging ensemble have rather different statistical properties. As a matter of fact, the first classifiers of the ordered sequence produce a steep error descent that indicates a higher degree of uncorrelated errors than the classifiers included at the end. In consequence, estimations based on the first classifiers in the ensemble can be very different from the final decision, which takes into account all the classifiers in the ensemble. By contrast, in the case of randomly ordered bagging ensembles, the test error rate monotonically decreases with the size of the ensemble, which indicates that the statistics behind the outputs of the first classifiers are maintained throughout the whole voting process. The results of extensive experimental evaluation show that early stopping in the aggregation process allows us to identify pruned ensembles, whose size is $\approx 20\%$ of the complete ensemble [28], which outperform bagging and retain bagging's resilience to noise in the class labels of the examples (see Figure 3.1). This figure shows that by stopping the aggregation of classifiers at $\approx 20 - 30\%$ of the total number of elements in the ensemble, a significant reduction in the classification error is obtained. These error curves are representative of the general behavior of bagging and ordered bagging in all the datasets investigated. We will analyze empirically the applicability of SIBP to ordered bagging ensembles.

3.2.2 Double pruning in Boosting

Boosting ensembles induce an intrinsic ordering during the training phase. One way to avoid the correlations related to the ordered sequence is to shuffle the classifiers in

the ensemble by a random permutation. In order to test the applicability of SIBP to boosting ensembles, we do it following the original order of the classifiers and random permutations. Furthermore we will analyze the potential applicability of a static pruning strategy with SIBP. The boosting ensembles are statically pruned using the SDP algorithm. Unlike ordered aggregation techniques, SDP does not induce an ordered sequence in the selected classifiers. Furthermore, the greedy variant of SDP is also tested together with boosting ensembles.

3.3 Experiments

For each classification problem, bagging and boosting ensembles of 101 CART trees are built. Every ensemble is pruned off-line to a different size, depending on the type of ensemble. Then, the on-line SIBP algorithm is applied to all the ensembles and results are reported. The confidence parameter is set to $\alpha = 0.99$.

All the experiments are performed on twelve binary classification problems from the UCI repository [32]. The same learning setup is used to make comparisons possible. In all cases the results reported are averages over 10 independent 10-fold cross validation estimates. The protocol followed in each execution for a partition of the data into training and test is as follows: (i) Build an ensemble of classifiers composed of $T = 101$ CART trees [34] using the training set. The standard settings for the generation of the decision trees are used. The ordering of the initial ensemble is determined by the order of generation, which is random in bagging and sequential in boosting. (ii) Estimate the generalization error in the test set for the whole ensemble. Apply SIBP to the complete ensemble using $\alpha = 99\%$ recording the test error and the average number of trees used to classify the instances. (iii) Modify the sequence of aggregation of the trees in the ensemble using the chosen off-line technique (SDP and greedy for boosting and boosting-based for bagging). Estimate the test error for the pruned ensemble and the value for the number of selected trees for the off-line pruned ensemble. Compute the average test error and the average number of classifiers used to classify the instances. (iv) Finally, apply SIBP to the off-line pruned subensemble and record the number of trees and classification error.

In the following sections the results of the experiments are summarized in tables displaying the error rates, disagreement rates between full and pruned ensembles, number of queried trees and speed-up rates between full and pruned ensembles. For each dataset, the tables show the average of the corresponding measure and its standard deviation after the $\pm$ sign.

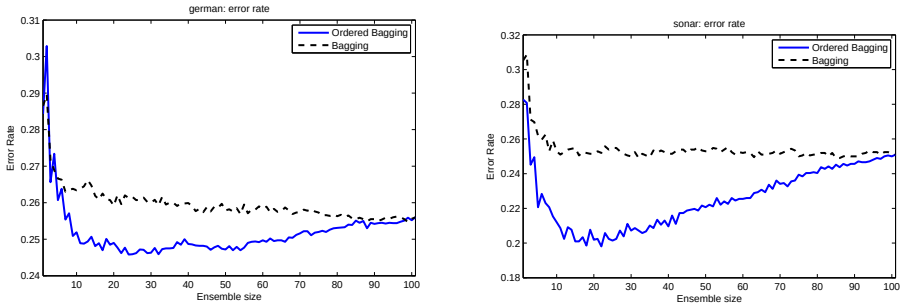


FIGURE 3.1: Test error curves with respect to the number of classifiers for bagging and bagging ordered using boosting based ordering

3.3.1 Boosting-based Ordered Bagging

In order to determine whether SIBP can be used in combination with ordered bagging several experiments are carried out. In the first set of experiments SIBP is applied to a standard (randomly ordered) bagging ensemble. As expected, the results of these experiments confirm the effectiveness of SIBP in parallel ensembles. A second batch of experiments show SIBP applied to ordered bagging. Finally, SIBP applied to a pruned ensemble that is obtained by selecting the first $\approx 20\%$ classifiers in the ordered bagging ensemble.

The results of applying SIBP to bagging are summarized in Table 3.1. For each dataset, the table shows the average test error for bagging (BAG101), bagging using the first 21 randomly generated classifiers (BAG21) and SIBP applied to the full bagging ensemble (IB-BAG101). The average number of trees used to classify each instance in IB-BAG101 is shown in the last column of the table. These experiments confirm the results reported in [25]. Table 3.1 shows that the generalization error of a bagging ensemble with 101 trees is generally better than a bagging ensemble composed of 21 classifiers. This also confirms the observation that increasing the size of parallel ensembles in which the generation of the individual classifiers involves some form of randomization generally improves the generalization performance of the ensemble [4]. In contrast, when SIBP is used to determine when to stop querying for the classification of new instances, a performance comparable to BAG101 is achieved in the studied datasets using on average a fairly small fraction of the classifiers. In particular, the average number of trees that need to be queried in IB-BAG101 ranges from 6.1 for *Votes* to 21.1 for *Liver*.

Table 3.2 compiles the results of the application of SIBP to ordered bagging ensembles. The column labeled BAG101 displays the test error rate obtained by a bagging ensemble composed of 101 trees. The third column presents the results of SIBP when applied to the complete ordered bagging ensemble (IB-OB101). The average number of trees used

TABLE 3.1: Average results for bagging (for each dataset the best method is highlighted in boldface)

	Test error			# trees IB
Problem	BAG101	BAG21	IB-BAG101	($\alpha = 99\%$)
australian	**14.5±3.8**	**14.5±3.8**	**14.5±3.8**	7.4±1.0
breast	**4.8±2.8**	**4.8±2.6**	**4.8±2.8**	8.6±1.3
diabetes	**24.9±3.9**	**24.9±4.0**	**24.9±3.9**	14.3±2.3
german	**25.6±3.1**	25.9±3.5	25.7±3.1	18.0±2.6
heart	19.6±8.0	19.9±8.0	**19.4±7.8**	18.5±4.8
horse-colic	17.8±6.3	17.9±6.0	**17.7±6.2**	9.9±3.0
ionosphere	**9.7±4.6**	9.8±4.5	**9.7±4.5**	8.6±1.9
labor	**13.4±12.8**	14.1±12.9	13.7±12.6	17.7±8.7
liver	**31.1±6.2**	31.9±6.9	**31.1±6.2**	21.1±5.2
sonar	25.1±9.7	25.1±9.2	**25.0±9.7**	19.4±5.4
tic-tac-toe	**1.6±1.3**	2.2±1.5	**1.6±1.3**	10.1±1.3
votes	**4.4±3.0**	**4.4±3.0**	**4.4±3.0**	6.1±0.3

by IB-OB101 is given in the sixth column. The results for a pruned ensemble composed of the first 21 trees of the ordered bagging ensemble are given in the column labeled OB21. These results show that the performance of ordered bagging with 21 classifiers is better than that of full bagging for all the datasets investigated except for *Votes*. Ordered bagging has two advantages over bagging: faster classification, because only a small fraction ($\approx 20\%$) of the original classifiers is used, and, in general, better accuracy in the test set. Instead of using a fixed number of classifiers, SIBP individually determines the number of classifiers that are needed to estimate the complete ensemble prediction for each particular instance. When SIBP is used in conjunction with ordered bagging (column IB-OB101 in Table 3.2), the number of queried classifiers is generally lower than the 21 trees used in pruned bagging (OB21). However, it is over the number of elements queried by SIBP for randomly ordered bagging (right most column of Table 3.1). In addition, the accuracy improvement with respect to bagging is not as ample as the improvement of OB21 over BAG101. This poorer performance is a consequence of the fact that IB-OB101 is making inference about the predictions of the complete ensemble on the basis of the predictions of only the first classifiers in the ordered sequence. These classifiers follow a distribution that is different from the overall distribution of classifiers in bagging. These results can be understood analyzing the plots displayed in Fig. 3.1. The curves depicted trace the dependence of the test error with the size of the ensemble using bagging and ordered bagging for the classification tasks *German* and *Sonar*.

In the final batch of experiments SIBP is applied to a pruned ensemble composed of the first 21 classifiers in ordered bagging. The results of these experiments are displayed in the fifth column of Table 3.2 (IB-OB21). The last column shows the average number of

TABLE 3.2: Average results for ordered bagging (for each dataset the best method is highlighted in boldface)

	Test error				# trees ($\alpha = .99$)	
Problem	BAG101	IB-OB101	OB21	IB-OB21	IB-OB101	IB-OB21
australian	14.5±3.8	14.3±3.9	**13.7±3.9**	**13.7±4.0**	11.3±1.7	7.0±0.5
breast	4.8±2.8	4.5±2.6	4.1±2.6	**4.0±2.6**	8.7±1.2	5.9±0.3
diabetes	24.9±3.9	24.7±4.0	**24.3±3.9**	**24.3±3.9**	17.2±2.3	8.7±0.6
german	25.6±3.1	25.2±3.3	24.8±3.7	**24.7±3.8**	21.1±2.5	9.3±0.6
heart	19.6±8.0	18.9±7.6	**18.6±7.2**	**18.6±7.1**	20.2±4.0	9.4±1.0
horse-colic	17.8±6.3	17.5±6.2	**16.3±6.6**	**16.3±6.5**	9.8±2.1	6.6±0.7
ionosphere	9.7±4.6	8.5±4.4	**7.5±4.2**	**7.5±4.1**	10.9±2.0	6.7±0.6
labor	13.4±12.8	10.0±11.3	**8.3±10.0**	8.5±10.0	14.8±7.5	7.9±1.9
liver	31.1±6.2	29.5±6.2	**28.2±6.5**	28.4±6.7	28.0±4.6	11.8±0.9
sonar	25.1±9.7	23.6±9.5	**20.2±10.7**	**20.2±10.7**	26.1±5.3	11.2±1.3
tic-tac-toe	1.6±1.3	**1.4±1.2**	**1.4±1.2**	1.5±1.2	9.4±1.0	6.5±0.4
votes	**4.4±3.0**	**4.4±3.1**	4.7±3.2	4.6±3.2	7.0±0.8	5.6±0.3

trees used by IB-OB21. These results, show that the generalization error of SIBP applied to OB-21 is equivalent to that of OB21 in the problems analyzed. Small variations of one or two tenths of a percent point both positive and negative can be observed for some datasets. Therefore, the improvements obtained by IB-OB21 over complete bagging (BAG101) are of the same magnitude as the improvements obtained by the pruned ensemble obtained by early stopping in ordered aggregation (OB21). The number of trees that need to be stored in memory is also reduced from 101 to 21 trees. Finally, the average number of trees that need to be queried is further reduced by the application of SIBP to the pruned ensemble OB21. Specifically, IB-OB21 employs an average number of trees that ranges from 5.6 (*Votes*) to 11.8 (*Liver*). In summary, the application of SIBP to the pruned ensemble obtained from ordered aggregation (OB21) improves the accuracy and reduces the memory requirements of bagging as much as OB21 does. It has the additional advantage that it predicts even faster than OB21.

The overall generalization performance of the different ensemble methods in the classification tasks analyzed is compared using the methodology proposed by Demšar [35]. Fig. 3.2 displays the rank of each method averaged over the results in the different classification tasks. In this diagram, the differences between methods connected with a horizontal line are not significant according to a Nemenyi test (p-value$<$ 0.05). The critical difference (CD=2.2 for 6 methods, 12 dataset and p-value$<$ 0.05) is shown for reference. The best overall performance corresponds to OB21 and IB-OB21. The performance of these two methods is equivalent in the classifications tasks investigated. According to this test the performance of OB21 and IB-OB21 in terms of average rank

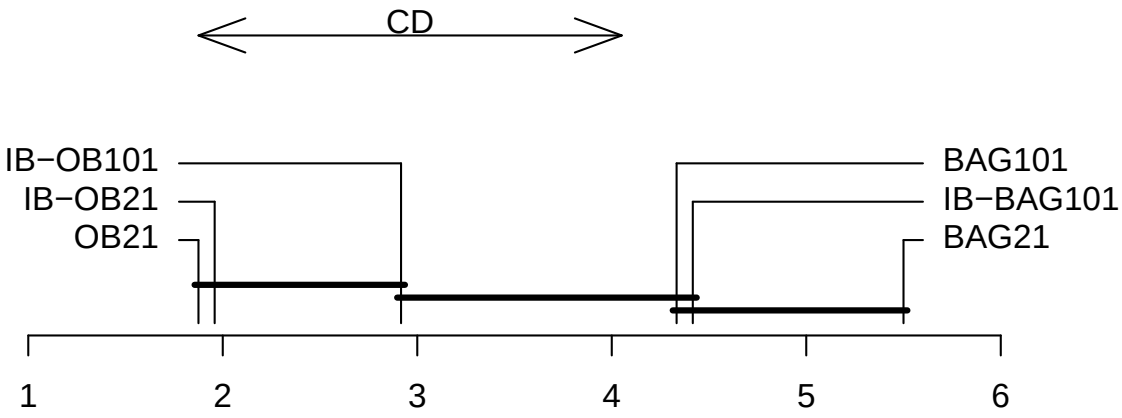


FIGURE 3.2: Comparison of the different methods using the Nemenyi test. Classification systems whose performance are not significantly different according to this test (p-value < 0.05) are connected by a line segment in the diagram.

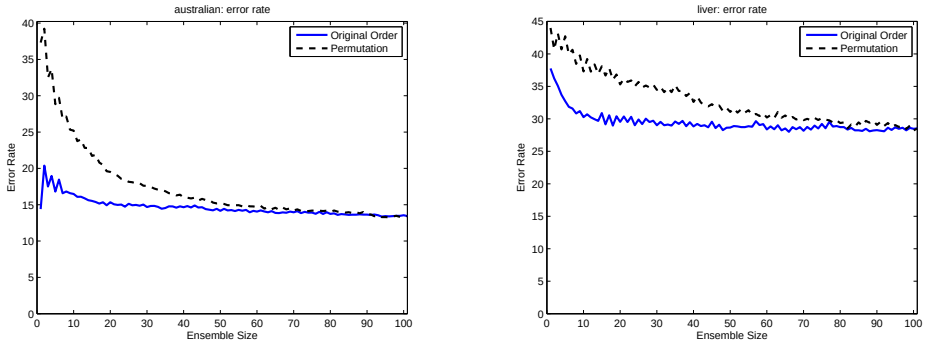
is significantly better than standard bagging. The performances of the remaining methods are not significantly different from bagging.

3.3.2 SDP-pruned Boosting

We perform several experiments to confirm whether SIBP can be applied to boosting ensembles and to pruned boosting ensembles obtained by the ordered aggregation method and the SDP method. First, SIBP is applied to boosting ensembles to see whether it can be used to reduce the error rates and the number of queried classifiers in several classification problems. Second SDP and its greedy variant is applied to the full ensembles. Finally, the on-line SIBP algorithm is applied to the ensembles pruned with the off-line technique.

Table 3.3 summarizes the mean error rate obtained by boosting ensembles with 101 classifiers (BOOST 101), ordered boosting ensembles with the first 51 classifiers (GREEDY 51) and pruned boosting ensembles using the SDP method with 51 classifiers (SDP 51). Every ensemble has been tested using the full ensemble (FULL), using SIBP following the natural order of the classifiers (IB-SEQ) or a random permutation of the classifiers in the ensemble (IB-PERM). Table 3.5 displays the average number of trees queried to classify every instance of a given problem. This table only shows the results obtained by the IB-pruned methods from table 3.3.

Although the classifiers in a boosting ensemble are not independent from each other, the experiments confirm that SIBP can be used to reduce the time and space cost of boosting ensembles. These results show that the mean error rate is usually the same or even better when the statistical instance based pruning algorithm is applied

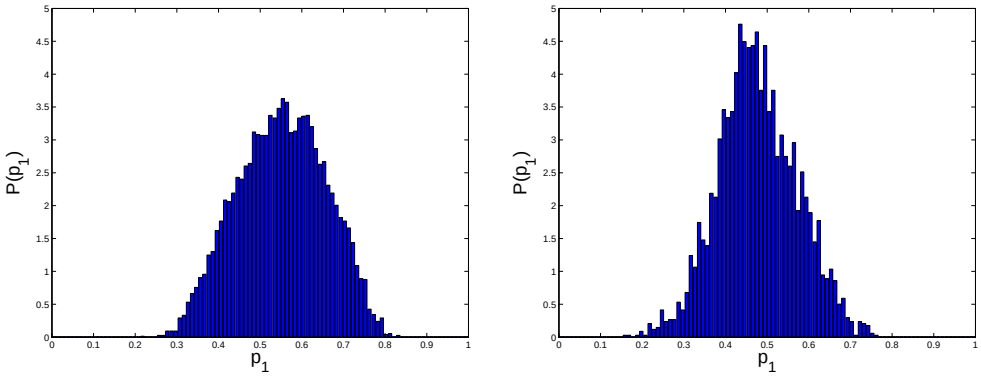


Australian dataset Liver dataset

FIGURE 3.3: Test error rate evolution following the original order and random permutations for boosting ensembles

to the original order of classifiers. Given that the SIBP algorithm is formulated on the hypothesis that the classifiers are uncorrelated, we test the method following a different random permuted order for each instance. However, several experiments show that SIB-pruned ensembles yield worse error and speed-up rates when applied over random ordered boosting ensembles instead of following their original order (columns 3 and 4 of tables 3.3 and 3.5). Moreover the disagreement rates achieved by the permuted ensemble are not acceptable, as they exceed the imposed bound 1% (column 3 of table 3.4). This can be explained looking at figure 3.3, where it can be observed that when the order of the hypothesis is altered, the error rate curve is generally above that of the original boosting. To understand this we conjecture the following: first, in the randomly ordered ensemble the voting statistics are given by prior distributions similar to the ones plotted in figure 3.4, and thus using a uniform prior distribution results in unrealistic and optimistic stopping rules; and second, when using the original sequence of boosting classifiers, the first hypothesis (those who have not been under the influence of the boosting algorithm for being created in the early iterations) probably have a more uniform voting distribution, and thus the best results. In conclusion, SIB-pruned boosting ensembles improve the accuracy of the original ensemble and ameliorates the classification process (only $30 - 50\%$ of the original trees are queried) only when the original order of classifiers is followed.

Comparing the performance of the statically pruned ensembles (by SDP or its greedy version) with that of the full boosting ensemble (column 1 versus columns 5 and 7 from tables 3.3 and 3.4). Unlike in the previous section, where the error rate curves of the ordered ensembles suggested the ideal size of the pruned ensemble, the error curves of neither method show an absolute minima. For this reason, 51 was chosen as the pruning point. The SDP and Greedy methods slightly improve the test error rates in our experiments, albeit some exceptions like *labor* and *sonar* where the error rates



Diabetes dataset Liver dataset

FIGURE 3.4: Prior distributions for hard to classify problems for boosting ensembles

are higher, and *horse-colic* and *liver* where the error rates are improved. There is no significant evidence that either Greedy or SDP is better than each other.

SIBP applied to SDP and Greedy subensembles have a generalization performance comparable to the off-line pruned subensembles (see table 3.3). Nevertheless, our main interest lies in speeding-up the classificacion process. In this regard, the double pruned subensembles, besides achieving similar or better results than the full ensemble, query between 14% (*votes*) and 33% (*liver*) of the trees queried by the original boosting ensemble (columns 6 and 8 against column 1 of 3.5). This fact confirms that the off-line pruning phase is useful to reduce the number of queried classifiers. Finally, the number of queried trees by the SIB-Greedy-pruned subensembles is usually lower when compared to SIB-SDP-pruned ensembles. Although the results obtained by both methods are very similar, the greedy version of SDP is recommended because it is computationally less expensive. Hence, this double-pruning method keeps and improves the error rates and speeds up the classification process. The method consists in summarizing the original boosting ensemble using the Greedy or SDP methods as explained previously and later applying the on-line instance-based pruning algorithm. It is important to emphasize that the method does not suppose an overhead of classification time as the execution of SDP is off-line and the SIBP does not add any complexity to the classification process.

3.4 Conclusions

We propose to combine two existing pruning algorithms (static and dynamic) to reduce the computational costs associated with the use of ensembles of classifiers for prediction and to improve their generalization performance. Two different combinations are proposed, depending on whether the ensemble learning algorithm used is bagging or boosting. The first strategy applies a static pruning method: boosting-based ordering

	BOOST101			GREEDY51		SDP51	
Problem	FULL	IB-SEQ	IB-PERM	FULL	IB-SEQ	FULL	IB-SEQ
australian	13.4±3.9	13.5±3.9	13.9±4.0	13.3±4.0	13.3±4.0	**13.3±3.9**	**13.3±3.9**
breast	3.3±2.1	3.3±2.1	3.4±2.1	3.2±2.0	3.3±2.0	3.2±2.0	**3.2 ±1.9**
diabetes	26.0±4.0	26.0±4.0	26.6±4.0	**25.1±4.1**	**25.1±4.0**	25.2±4.2	25.2±4.1
german	24.8±3.8	24.7±3.8	25.3±4.1	24.5±3.6	24.6±3.6	**24.5±3.4**	**24.5±3.5**
heart	19.7±7.1	**19.4±6.7**	20.1±7.5	19.5±7.1	19.6±7.1	19.6±7.0	19.6±7.0
horse-colic	21.3±6.7	20.9±6.6	22.0±6.6	17.7±5.7	17.5±5.8	**17.4±5.8**	17.5±5.9
ionosphere	6.3±3.5	6.3±3.5	6.3±3.4	**6.2±3.4**	6.3±3.4	**6.2±3.4**	6.2±3.5
labor	**6.7±9.8**	**6.7±9.8**	6.8±10.0	7.1±10.5	7.1±10.5	7.2±10.5	7.1±10.5
liver	28.5±7.2	28.5±7.0	29.0±7.0	27.8±7.0	27.8±6.9	**27.8±6.8**	**27.8±6.8**
sonar	13.9±8.3	**13.8±8.1**	14.4±8.3	15.2±8.2	15.1±8.1	15.1±8.5	15.1±8.3
tic-tac-toe	**0.7±0.9**	**0.7±0.9**	0.7±0.9	0.8±1.0	0.8±1.0	0.8±1.0	0.8±1.0
votes	**4.5±3.1**	**4.5±3.1**	4.6±3.1	4.6±2.9	4.7±2.9	4.6±2.9	4.6±2.9

TABLE 3.3: Error rates comparison of the double-pruning scheme for boosting ensembles (for each dataset the best method is high-lighted in boldface)

	BOOST101		GREEDY51	SDP51
Problem	IB-SEQ	IB-PERM	IB-SEQ	IB-SEQ
australian	0.1±0.4	1.1±1.2	0.2±0.6	0.2±0.6
breast	0.0±0.2	0.2±0.5	0.0±0.2	0.0±0.2
diabetes	0.2±0.5	1.9±1.6	0.5±0.8	0.2±0.5
german	0.6±0.7	1.9±1.4	0.7±0.7	0.4±0.6
heart	0.2±1.0	1.1±1.9	0.3±1.2	0.1±0.6
horse-colic	0.6±1.4	1.9±2.4	0.5±1.1	0.2±0.7
ionosphere	0.0±0.0	0.2±0.9	0.1±0.4	0.2±0.7
labor	0.0±0.0	0.9±4.1	0.0±0.0	0.2±1.7
liver	0.2±0.7	2.4±2.6	0.8±1.4	0.3±1.1
sonar	0.2±0.9	1.7±2.9	0.4±1.6	0.6±1.7
tic-tac-toe	0.0±0.2	0.2±0.4	0.0±0.2	0.0±0.2
votes	0.0±0.2	0.2±0.7	0.1±0.5	0.0±0.0

TABLE 3.4: Disagreement rates comparison of the double-pruning scheme for boosting ensembles

with early stopping in the case of bagging ensembles and SDP-pruning for boosting ensembles. The static methods reduce the storage requirements, speed-up the classification process and improve the generalization performance. The second strategy consists in applying the dynamic method Instance-Based Pruning. This method does not reduce the storage requirements, but further reduces the classification time without decreasing the accuracy significantly. SIBP is applied by computing the probability that the majority class obtained after having queried t classifiers is the same class given by the full ensemble. If this probability is above a specified confidence level α the classification process stops.

For bagging ensembles, SIBP applied to the original ensemble obtains error rates similar to the complete ensemble and reduces the average number of queries more than the

	BOOST101			GREEDY51		SDP51	
Problem	FULL	IB-SEQ	IB-PERM	FULL	IB-SEQ	FULL	IB-SEQ
australian	79.0±1.9	35.0±4.7	37.7±4.2	37.1±0.8	**19.6±1.8**	37.1±0.8	19.9±2.0
breast	62.2±1.5	12.7±2.0	14.9±1.9	29.1±0.5	**8.8±0.8**	29.4±0.5	9.2±1.0
diabetes	85.4±1.7	49.1±5.2	52.2±4.5	39.8±0.7	25.2±1.7	39.7±0.8	**25.1±2.0**
german	85.4±1.5	49.3±4.6	51.6±3.9	40.8±0.6	26.9±1.5	40.6±0.7	**26.2±1.7**
heart	81.7±2.1	42.1±6.0	42.8±5.8	38.5±1.1	22.9±2.7	38.4±1.2	**22.6±3.0**
horse-colic	86.0±2.2	45.7±6.1	53.8±6.2	39.1±1.1	**22.8±2.6**	39.5±1.1	23.4±2.7
ionosphere	71.8±1.7	21.6±3.8	25.5±3.7	32.3±1.0	**12.0±1.7**	33.0±0.9	13.1±1.7
labor	72.3±4.5	27.3±10.5	25.5±9.1	33.6±2.5	**14.2±5.1**	34.4±2.4	16.2±5.0
liver	88.1±1.7	59.1±6.2	57.4±5.6	42.0±1.0	**29.0±2.7**	42.1±1.0	29.9±2.8
sonar	82.4±2.2	44.8±7.0	45.0±7.3	39.7±1.3	**24.3±3.3**	39.8±1.3	25.1±3.0
tic-tac-toe	69.6±0.9	17.4±1.7	20.9±1.7	32.6±0.4	**11.5±0.9**	33.2±0.5	12.6±1.0
votes	66.3±1.8	15.6±3.2	18.9±2.5	30.0±0.7	**9.4±1.2**	30.5±0.8	10.4±1.4

TABLE 3.5: Number of trees comparison of the double-pruning scheme for boosting ensembles (for each dataset the best method is high-lighted in boldface)

pruned ensemble that is built by selecting the first 21 classifiers in the ordered ensemble. When SIBP is applied to the complete ordered ensemble its generalization accuracy is better than the complete ensemble. Nevertheless, this accuracy is still inferior to the pruned ordered ensemble. This is due to the fact that the distribution of the predictions of classifiers that appear first in the ordered ensemble is different from the last classifiers included. Therefore, one of the basic assumptions of IB does not hold, leading to suboptimal performance. Finally, when SIBP is applied to the pruned ordered ensemble itself, a significant speed-up is achieved with a performance that is similar to the pruned ensemble and much better than bagging.

For boosting ensembles, SIBP applied to the original ensemble improves very slightly its accuracy, despite being a sequential ensemble and thus not holding a basic SIBP assumption. The number of queried trees is reduced $50 - 70\%$ depending on the problem without increasing the error rates. The SDP-pruned ensembles are reduced to 51 classifiers and its accuracy is improved with respect to the full boosting ensemble. When the SIBP method is applied to the pruned ensembles, the error rates are increased, although they stay close to the rates achieved by the full ensemble. However, their speed-up rates are further improved.

The result is a double pruning algorithm that significantly improves the performance of the original ensemble learning algorithms: the accuracy is improved or maintained, depending on the problem and the ensemble algorithm. Moreover, it reduces the memory requirements with respect to the full ensemble, because only the classifiers that are selected in the pruned ordered ensemble need to be accessible for potential queries, and predicts much faster than both the original ensemble and the pruned one.

Chapter 4

Improved Statistical Instance Based Pruning

4.1 Introduction

The *Statistical Instance-Based Pruning* technique ([25] and section 2.2.2.2) is a dynamic pruning method for classification ensembles. The method works under the assumption that the classifiers of the ensemble are independent from each other given the training data. Iteratively, the method queries one classifier of the ensemble and updates the number of votes assigned to each class. Using the voting information received until that moment, SIBP computes the probability that the majority voted class by the full ensemble and the queried subset coincide. If this probability exceeds a given confidence threshold α, the classification process is halted.

After extensive empirical evidence, it has been observed that often the disagreement rates between the full ensemble and the SIB-pruned ensemble is usually below the theoretical bound $1 - \alpha$. We conjecture that this is due to the fact that no prior information on the classification problem is included in the derivation. The original method assumes a uniform prior distribution of the votes, which may result in unrealistic pruning points and disagreement rates lower than the user's imposed theoretical bound. By incorporating the prior knowledge about the specific problem, our goal is to improve the disagreement estimation between the full and pruned ensembles and to speed-up the classification time in those cases where the disagreement rate is underestimated.

In this chapter we propose three different ways to model the prior distribution over the probability vectors $\mathcal{P}(\mathbf{p})$ from the SIBP technique. In that formulation the prior distribution over the probability vector $\mathbf{p}(\mathbf{x})$ was considered uniform.

The probability vector $\mathbf{p}(\mathbf{x})$ is defined as

$$\mathbf{p}(\mathbf{x}) = \{p_1(\mathbf{x}), \ldots, p_l(\mathbf{x})\}, \quad \sum_{i=1}^{l} p_i(\mathbf{x}) = 1 \tag{4.1}$$

where $p_i(\mathbf{x})$ is the probability that instance $\mathbf{x}$ is predicted as class y_i by any hypothesis in the ensemble, regardless of the real class. The real distribution of this probability vector is unknown. However, approximated samples can be obtained from the training dataset using, for example, out-of-bag or cross-validation. Using these empirical samples the value of $p_i(\mathbf{x}_j)$ can be obtained as t_i^j/T, where t_i^j is the number of votes assigned to class i for instance $\mathbf{x}_j$. These empirical probability vector samples can be used to estimate the distribution over $\mathbf{p}$.

The two first proposals consider that $\mathbf{p}$ follow a Dirichlet distribution. The third alternative introduces a discrete approach to the instance based pruning technique and model the prior distribution of $\mathbf{p}$ using a discrete non-parametric distribution of the votes in the training data. In the next sections we develop the mathematical equations involved are derived and the results of the new proposed methods are described.

Our experiments prove that by incorporating prior knowledge about the problem, the disagreement rates come closer to the theoretical bound and the speed-up rates are improved significantly for those problems with prior distributions over $\mathbf{p}$ very different from the assumed uniform distribution. In addition, the test error rates are kept close to the error rates achieved by the original ensemble.

This chapter is organized as follows. In section 4.2 the prior distribution is modelled after a Dirichlet distribution. In section 4.3 a mixture of Dirichlet distribution is used to better fit the multi-modal distributions. Section 4.4 presents a reformulation of the SIBP technique that allows us to model the prior with a non-parametric distribution. Section 4.5 discusses the computational complexity of each method. Section 4.6 shows the experimental results of the proposed methods over a set of benchmarking problems. Finally, section 4.7 contains the conclusions of the whole chapter.

4.2 Prior following a Dirichlet distribution

We consider that the distribution $\mathcal{P}(\mathbf{p})$ follows a Dirichlet distribution

$$\mathcal{P}(\mathbf{p}; \boldsymbol{\beta}) = \frac{\Gamma(\beta)}{\Gamma(\beta_1) \ldots \Gamma(\beta_l)} \prod_{i=1}^{l} p_i^{\beta_i - 1} \tag{4.2}$$

where $\boldsymbol{\beta}$ is the vector $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_l)$ such that $\beta_i > 0$, $\beta = \sum_{i=1}^{l} \beta_i$ and $l \geq 2$. The Dirichlet distribution is equivalent to a beta distribution in a 2-dimensional problem. The support of the Dirichlet distribution is the hyperplane of points that satisfies $\sum_{i=1}^{l} p_i = 1$ and $0 < p_i < 1 \ \forall i$. We choose to model the prior after a Dirichlet distribution because it models random variables that can be interpreted as probabilities, which is the case with vector $\mathbf{p}(\mathbf{x})$. Moreover, given that the likelihood $\mathcal{P}(\mathbf{t}|\mathbf{p})$ is modeled after a multinomial distribution, the posterior distribution over $\mathbf{p}$ is also a Dirichlet distribution, which is a desirable property.

Before proceeding to the mathematical proof of the method, it is important to point out that by introducing this prior, the β_i parameters might no longer be integers. This must be taken into account when evaluating the gamma function $\Gamma(\beta_i)$. The gamma function is defined for complex numbers with a positive real part $z \in \mathbb{C}^+ = \{x \in \mathbb{C} : \mathrm{Re}\{x\} > 0\}$ by the following equation

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt \tag{4.3}$$

The Gamma function satisfies

$$\Gamma(z) = (z-1)\Gamma(z-1) \tag{4.4}$$

Notice that when n is a positive integer greater or equal to 1, the previous formula simplifies to $\Gamma(n) = (n-1)!$. Property 4.4 allows us to avoid the computation of the integral until $0 < \mathrm{Re}\{z\} < 1$, which is an expensive calculation. Fortunately, the Gamma functions in our calculations will always appear in fractions of the form $\Gamma(x+t)/\Gamma(x)$. This form allows us to simplify the computations to products of real numbers, without having to compute the integral 4.3, as:

$$\frac{\Gamma(x+t)}{\Gamma(x)} = \begin{cases} (x)\ldots(x+t-1) = (x)_t & t \in \mathbb{N}, \quad x \in \mathbb{R}^+ \\ 1 & t = 0 \end{cases}$$

After clearing up this implementation detail, let us go back to the derivation. Just as in the original SIBP, our goal is to compute the probability that the ensemble of size T classifies an instance with the same label as the one predicted by the sub-ensemble of size t. We denote that probability as $\mathcal{P}^*(\mathbf{t}, T)$.

Proposition 4.1. *Assuming that the distribution over vector* $\mathbf{p}(\mathbf{x})$ *follows a Dirichlet distribution of parameters* $\boldsymbol{\beta} = \{\beta_1, \ldots, \beta_l\}$ *and that the likelihood* $\mathcal{P}(\mathbf{t}|\mathbf{p})$ *follows a multinomial distribution, the conditional probability of vector* $\mathbf{T} = \{T_1, \ldots, T_l\}$ *given vector* $\mathbf{t} = \{t_1, \ldots, t_l\}$ *obtained after querying* t *classifiers is*

$$\mathcal{P}(\mathbf{T}|\mathbf{t}) = \frac{(T-t)!}{\prod_{i=1}^{l}(T_i - t_i)!} \frac{\prod_{i=1}^{l}(t_i + \beta_i)_{T_i - t_i}}{(t+\beta)_{T-t}} \tag{4.5}$$

Proof. Given a probability vector $\mathbf{p}$, the probability distribution of vector $\mathbf{t} = \{t_1, \ldots, t_l\}$ follows a multinomial distribution

$$\mathcal{P}(\mathbf{t}|\mathbf{p}) = \frac{t!}{t_1!t_2!\ldots t_l!}\prod_{i=1}^{l} p_i^{t_i} \tag{4.6}$$

The class predicted at time t is given by $k_{\mathbf{t}} = \arg\max_i t_i$. The posterior probability $\mathcal{P}(\mathbf{p}|\mathbf{t})$ can be obtained by applying Bayes Theorem and considering that the prior $\mathcal{P}(\mathbf{p})$ follows equation (4.2).

$$\mathcal{P}(\mathbf{p}|\mathbf{t}) = \frac{\mathcal{P}(\mathbf{t}|\mathbf{p})\mathcal{P}(\mathbf{p})}{\mathcal{P}(\mathbf{t})} = \frac{\Gamma(t+\beta)}{\prod_{i=1}^{l}\Gamma(t_i+\beta_i)}\prod_{i=1}^{l} p_i^{t_i+\beta_i-1} \tag{4.7}$$

where $\mathcal{P}(\mathbf{t})$, the normalization constant is given by:

$$\begin{aligned}\mathcal{P}(\mathbf{t}) &= \int_{\mathcal{D}} \mathcal{P}(\mathbf{t}|\mathbf{p})\mathcal{P}(\mathbf{p})d\mathbf{p} = \frac{\Gamma(\beta)}{\Gamma(\beta_1)\ldots\Gamma(\beta_l)}\frac{t!}{t_1!t_2!\ldots t_l!}\int_{\mathcal{D}}\prod_{i=1}^{l} p_i^{t_i+\beta_i-1}d\mathbf{p} \\ &= \frac{\Gamma(\beta)}{\Gamma(\beta_1)\ldots\Gamma(\beta_l)}\frac{t!}{t_1!t_2!\ldots t_l!}\frac{\prod_{i=1}^{l}\Gamma(t_i+\beta_i)}{\Gamma(t+\beta)}\end{aligned}$$

Thus the posterior distribution of $\mathbf{p}$ given the observed votes vector $\mathbf{t}$ is given by a Dirichlet distribution of order l and parameters $(t_1+\beta_1, t_2+\beta_2, \ldots, t_l+\beta_l)$.

Before inducing the equation of $\mathcal{P}(\mathbf{T}|\mathbf{t})$, it is necessary to understand two mathematical matters. First, the probability distribution $\mathcal{P}(\mathbf{T}|\mathbf{t})$ is the same as $\mathcal{P}(\mathbf{T}-\mathbf{t}|\mathbf{t})$. And second, the probability distribution $\mathcal{P}(\mathbf{T}-\mathbf{t}|\mathbf{p},\mathbf{t})$ is equivalent to $\mathcal{P}(\mathbf{T}-\mathbf{t}|\mathbf{p})$. Both remarks are direct consequences of the individual classifiers being independent from each other given the training data.

$$\begin{aligned}\mathcal{P}(\mathbf{T}|\mathbf{t}) &= \mathcal{P}(\mathbf{T}-\mathbf{t}|\mathbf{t}) = \int_{\mathcal{D}} \mathcal{P}(\mathbf{T}-\mathbf{t}|\mathbf{p},\mathbf{t})\mathcal{P}(\mathbf{p}|\mathbf{t})d\mathbf{p} = \int_{\mathcal{D}} \mathcal{P}(\mathbf{T}-\mathbf{t}|\mathbf{p})\mathcal{P}(\mathbf{p}|\mathbf{t})d\mathbf{p} \\ &= \frac{(T-t)!}{\prod_{i=1}^{l}(T_i-t_i)!}\frac{\Gamma(t+\beta)}{\prod_{i=1}^{l}\Gamma(t_i+\beta_i)}\int_{\mathcal{D}}\prod_{i=1}^{l} p_i^{T_i+\beta_i-1}d\mathbf{p} \\ &= \frac{(T-t)!}{\prod_{i=1}^{l}(T_i-t_i)!}\frac{\Gamma(t+\beta)}{\prod_{i=1}^{l}\Gamma(t_i+\beta_i)}\frac{\prod_{i=1}^{l}\Gamma(T_i+\beta_i)}{\Gamma(T+\beta)} \\ &= \frac{(T-t)!}{\prod_{i=1}^{l}(T_i-t_i)!}\frac{\prod_{i=1}^{l}(t_i+\beta_i)_{T_i-t_i}}{(t+\beta)_{T-t}}\end{aligned}$$

where $(t)_n = t(t+1)\cdots(t+n-1)$ is the Pochhammer symbol, t is a non-negative real number and n is a non-negative integer. □

Finally, the probability that the classes predicted by the subensemble of size t and the

full ensemble of size T coincide is the sum of the conditional probabilities $\mathcal{P}(\mathbf{T}|\mathbf{t})$ for each $\mathbf{T} \in \mathcal{T}_\mathbf{t}$, where $\mathcal{T}_\mathbf{t}$ is the set of vectors $\mathbf{T}$ such that $k_\mathbf{T} = k_\mathbf{t}$, $T_i \geq t_i$ and $\sum_{i=1}^{l} T_i = T$. Therefore we have

$$\mathcal{P}^*(\mathbf{t}, T) = \frac{(T-t)!}{(t+\beta)_{T-t}} \sum_{\mathbf{T}\in\mathcal{T}_\mathbf{t}} \frac{\prod_{i=1}^{l} (t_i+\beta_i)_{T_i-t_i}}{\prod_{i=1}^{l} (T_i-t_i)!} \tag{4.8}$$

Notice that if $\beta_i = 1\ \forall i$ (parameters corresponding to the uniform distribution), and thus $\beta = l$, the exact same formula from the regular instance-based method is recovered.

4.3 Prior following a mixture of Dirichlet distributions

Using only one Dirichlet distribution to model the prior might not always provide a good fit. The distributions $\mathcal{P}(\mathbf{p})$ for the examined datasets often show two or more different modes which cannot be accurately approximated with one Dirichlet distribution. Given that usually each mode corresponds to the distribution of instances of each class, we choose to fit the prior distribution with a mixture of Dirichlets, where the number of Dirichlets is given by the number of classes in the dataset and each Dirichlet is estimated using only examples of one class. Following this scheme, the prior distribution can be defined as

$$\mathcal{P}(\mathbf{p}) = \sum_{j=1}^{l} \mathcal{P}(c_j)\mathcal{P}(\mathbf{p}|c_j) = \sum_{j=1}^{l} w_j \frac{\Gamma(\beta^j)}{\Gamma(\beta_1^j)\ldots\Gamma(\beta_l^j)} \prod_{i=1}^{l} p_i^{\beta_i^j - 1} \tag{4.9}$$

where each Dirichlet is weighted by w_j, the class prior, that can be estimated by the frequency of each class, i.e. $w_j = \frac{n_j}{N}$ where $N = \sum_{j=1}^{l} n_j$. The remaining part of the mathematical proof follows the same steps as in the previous section, albeit the differences included with the new prior.

Proposition 4.2. *Assuming that the distribution over vector* $\mathbf{p}(\mathbf{x})$ *follows a mixture of Dirichlet distributions* (4.9)*, and that the likelihood* $\mathcal{P}(\mathbf{t}|\mathbf{p})$ *follows a multinomial distribution, the conditional probability of vector* $\mathbf{T} = \{T_1, \ldots, T_l\}$ *given vector* $\mathbf{t} = \{t_1, \ldots, t_l\}$ *obtained after querying* t *classifiers is*

$$\mathcal{P}(\mathbf{T}|\mathbf{t}) = \frac{(T-t)!}{\prod_{j=1}^{l}(T_j-t_j)!} \frac{\sum_{i=1}^{l} \frac{w_i}{(\beta^i)_T} \prod_{j=1}^{l} (\beta_j^i)_{T_j}}{\sum_{j=1}^{l} \frac{w_j}{(\beta^j)_t} \prod_{i=1}^{l} (\beta_i^j)_{t_i}} \tag{4.10}$$

Proof. The conditional probability distribution $\mathcal{P}(\mathbf{t}|\mathbf{p})$ follows a multinomial distribution (see Eq. (4.6)). In order to apply the Bayes Theorem, we first need to compute the

normalization constant $\mathcal{P}(\mathbf{t})$:

$$\begin{aligned}
\mathcal{P}(\mathbf{t}) &= \int_{\mathcal{D}} \mathcal{P}(\mathbf{t}|\mathbf{p})\mathcal{P}(\mathbf{p})d\mathbf{p} \\
&= \frac{t!}{t_1!\dots t_l!}\sum_{j=1}^{l} w_j \frac{\Gamma(\beta^j)}{\Gamma(\beta_1^j)\dots\Gamma(\beta_l^j)} \int_D \prod_{i=1}^{l} p_i^{t_i+\beta_i^j-1} d\mathbf{p} \\
&= \frac{t!}{t_1!\dots t_l!}\sum_{j=1}^{l} w_j \frac{\Gamma(\beta^j)}{\Gamma(t+\beta^j)} \prod_{i=1}^{l} \frac{\Gamma(t_i+\beta_i^j)}{\Gamma(\beta_i^j)} \\
&= \frac{t!}{t_1!\dots t_l!}\sum_{j=1}^{l} \frac{w_j}{(\beta^j)_t} \prod_{i=1}^{l} (\beta_i^j)_{t_i} \qquad (4.11)
\end{aligned}$$

The posterior distribution $\mathcal{P}(\mathbf{p}|\mathbf{t})$ is computed applying the Bayes Theorem. Unlike in the previous section, numerator and denominator cannot simplify, and the resulting equation is computationally more complex:

$$\mathcal{P}(\mathbf{p}|\mathbf{t}) = \frac{\mathcal{P}(\mathbf{t}|\mathbf{p})\mathcal{P}(\mathbf{p})}{\mathcal{P}(\mathbf{t})} = \frac{\sum_{j=1}^{l} w_j \Gamma(\beta^j) \prod_{i=1}^{l} \frac{p_i^{t_i+\beta_i^j-1}}{\Gamma(\beta_i^j)}}{\sum_{j=1}^{l} \frac{w_j}{(\beta^j)_t} \prod_{i=1}^{l} (\beta_i^j)_{t_i}} \qquad (4.12)$$

Finally, the probability of vector $\mathbf{T}$ given vector $\mathbf{t}$, or $\mathcal{P}(\mathbf{T}|\mathbf{t})$, is given by:

$$\begin{aligned}
\mathcal{P}(\mathbf{T}|\mathbf{t}) &= \int_{\mathcal{D}} \mathcal{P}(\mathbf{T}-\mathbf{t}|\mathbf{p})\mathcal{P}(\mathbf{p}|\mathbf{t})d\mathbf{p} \\
&= \frac{(T-t)!}{\prod_{i=1}^{l}(T_i-t_i)!} \frac{\sum_{j=1}^{l} w_j \Gamma(\beta^j) \int_D \prod_{i=1}^{l} \frac{p_i^{T_i+\beta_i^j-1}}{\Gamma(\beta_i^j)} d\mathbf{p}}{\sum_{j=1}^{l} \frac{w_j}{(\beta^j)_t} \prod_{i=1}^{l} (\beta_i^j)_{t_i}} \\
&= \frac{(T-t)!}{\prod_{j=1}^{l}(T_j-t_j)!} \frac{\sum_{i=1}^{l} \frac{w_i}{(\beta^i)_T} \prod_{j=1}^{l} (\beta_j^i)_{T_j}}{\sum_{j=1}^{l} \frac{w_j}{(\beta^j)_t} \prod_{i=1}^{l} (\beta_i^j)_{t_i}}
\end{aligned}$$

□

And the probability that the class predicted by the full ensemble and the partially queried ensemble of size t is

$$\mathcal{P}^*(\mathbf{t},T) = \sum_{\mathbf{T}\in\mathcal{T}_\mathbf{t}} \frac{(T-t)!}{\prod_{j=1}^{l}(T_j-t_j)!} \frac{\sum_{i=1}^{l} \frac{w_i}{(\beta^i)_T} \prod_{j=1}^{l} (\beta_j^i)_{T_j}}{\sum_{j=1}^{l} \frac{w_j}{(\beta^j)_t} \prod_{i=1}^{l} (\beta_i^j)_{t_i}} \qquad (4.13)$$

where the vectors $\mathbf{T} \in \mathcal{T}_\mathbf{t}$ satisfy that $k_\mathbf{T} = k_\mathbf{t}$, $T_i \geq t_i$ and $\sum_{i=1}^{l} T_i = T$.

4.4 A Reformulation of the SIBP Method

In this section we propose an equivalent formulation of the SIBP method using a discrete Hypergeometric distribution. This new formulation, equivalent to the original one, allows us to incorporate a discrete non-parametric prior distribution.

The classification process can be modelled after the classical urn models. Suppose we have a total of T balls in the urn of l different classes, of which T_k is the number of balls of class y_k with $1 \leq k \leq l$. The total number of balls and the number of balls for each class is fixed. After t balls have been extracted without replacement, a number of t_k of them belong to class y_k, $\sum_{i=1}^{l} t_i = t$. The probability of extracting t_k balls of class y_k after extracting a total of t, given a total of T balls in the urn, of which T_k belong to class y_k, can be described by the hypergeometric distribution:

$$\mathcal{P}(\mathbf{t}|\mathbf{T}) = \frac{\prod_{i=1}^{l} \binom{T_i}{t_i}}{\binom{T}{t}} \tag{4.14}$$

The posterior distribution $\mathcal{P}(\mathbf{T}|\mathbf{t})$ is computed applying the Bayes Theorem

$$\begin{aligned} \mathcal{P}(\mathbf{T}|\mathbf{t}) &= \frac{\mathcal{P}(\mathbf{t}|\mathbf{T})\mathcal{P}(\mathbf{T})}{\mathcal{P}(\mathbf{t})} = \frac{\mathcal{P}(\mathbf{t}|\mathbf{T})\mathcal{P}(\mathbf{T})}{\sum_{\mathbf{T}^* \in \Omega_{\mathbf{t}}} \mathcal{P}(\mathbf{t}|\mathbf{T}^*)\mathcal{P}(\mathbf{T}^*)} = \frac{\binom{T_1}{t_1} \dots \binom{T_l}{t_l}\mathcal{P}(\mathbf{T})}{\sum_{\mathbf{T}^* \in \Omega_{\mathbf{t}}} \binom{T_1^*}{t_1} \dots \binom{T_l^*}{t_l}\mathcal{P}(\mathbf{T}^*)} \\ &= \frac{\frac{T_1!}{(T_1-t_1)!} \dots \frac{T_l!}{(T_l-t_l)!}\mathcal{P}(\mathbf{T})}{\sum_{\mathbf{T}^* \in \Omega_{\mathbf{t}}} \frac{T_1^*!}{(T_1^*-t_1)!} \dots \frac{T_l^*!}{(T_l^*-t_l)!}\mathcal{P}(\mathbf{T}^*)} \end{aligned} \tag{4.15}$$

where $\Omega_{\mathbf{t}}$ is the set of vectors $\mathbf{T}$ such that $T_i \geq t_i \ \forall i$ and $\sum_i^l T_i = T$.

Assuming a uniform prior distribution $\mathcal{P}(\mathbf{T}) = \frac{1}{\|\mathbf{T}\|}$, where $\|\mathbf{T}\|$ stands for the number of possible $\mathbf{T}$ vectors, we recover the formula of the original SIBP method (2.38). A proof of that result is included in Appendix B (for binary problems).

The distribution $\mathcal{P}(\mathbf{T})$ can be modeled using a non-parametric prior. The values of this prior can be obtained from the training data by some in-sample validation: out-of-bag or cross validation. Out-of-bag is faster, since it does not require multiple generations of the ensemble. However, each instance $\mathbf{x}_n$ is classified only by a fraction of the classifiers $\tilde{T}^n$, such that $\tilde{T}^n = \tilde{T}_1^n + \ldots + \tilde{T}_l^n$, where $\tilde{T}_i^n$ is the number of out-of-bag votes assigned to class i for instance $\mathbf{x}_n$. In order to estimate the number of votes for each class with respect to T, we proceed

$$T_i^n \approx round\left(\frac{T\,\tilde{T}_i^n}{\tilde{T}^n}\right)$$

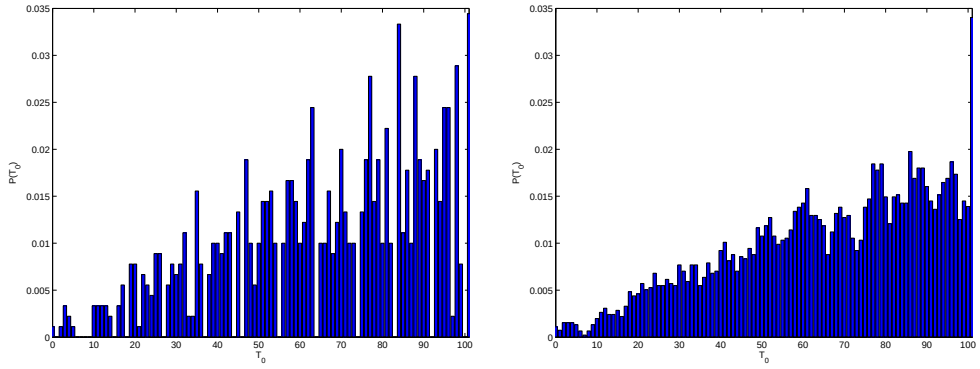


FIGURE 4.1: Prior of a selected fold from the German problem

This normalization process generates artifacts in the prior distributions. In figure 4.1(left) it can be observed that some probability values of T_i are zero, while nearby probability values of T_i are very high. In order to reduce the spurious effect of the normalization, the value of each probability is smoothed by averaging the values using a sliding window of size 5. Figure 4.1(right) shows the distribution $\mathcal{P}(\mathbf{T})$ on the left plot after being smoothed. Figures A.6 and A.7 in Appendix A show the smoothed prior distributions for all analyzed problems.

4.5 Computational complexity

The original SIBP formulation, with uniform prior distribution, permits to compute for binary problems a look-up table indexed by the number of votes of the minority class and whose values are the minimum number of votes of the majority class given confidence interval α. The consequence of using a uniform prior is that all classes are considered equivalent. Hence it is sufficient to compute one look-up table for all classes. When incorporating the prior knowledge, the prior probability of votes for each class is not necessarily the same and $\mathcal{P}(t_1 = n, t_2 = m)$ is not necessarily equal to $\mathcal{P}(t_1 = m, t_2 = n)$ for $n \neq m$. Therefore one effect of introducing prior knowledge of the $\mathbf{p}$ distribution, is that one look-up table per class will be necessary.

In addition, it is also necessary to compute a different set of tables for each dataset. The prior distribution is estimated from the training data and, hence, it is dependent on the problem. Furthermore, for experimental comparisons, since these instances are different for each partition of the data, it is necessary to recompute again the look-up tables for each realization.

Finally, the calculation of the look-up tables for the Dirichlet-based SIBP methods is computationally more expensive with respect to the original uniform SIBP method. In

	Products	Divisions
One Dirichlet	$2(T-t-1)$	1
Mixture of Dirichlets	$2l(T+t-1)$	$2l+1$
Difference	$2[(l-1)(T-1)+t(l+1)]$	$2l$

TABLE 4.1: Complexity comparison of the priod-based SIBP methods

the following paragraphs we offer some insight into some of the implementation details used to build the look-up tables. The hypergeometric formulation only uses integers in its computations, making it possible to use the tricks described in [36] to improve the computation speed of the look-up table. The authors of [36] propose to decompose all the terms in (2.38) in prime factors, then to simplify numerator and denominator, and to compute only the irreducible fractions, which avoids numerical overflows for high values of T. Moreover, for a fixed vector t, the denominator is the same for all the set of possible vectors T, allowing to compute it only once.

The Dirichlet-based methods, on the contrary, cannot take advantage of these tricks because the estimated parameters of the distribution are real numbers and therefore a decomposition in prime factors is not possible. Analyzing equations (4.5) and (4.10), it can be observed that the mixture of Dirichlets (4.10) is computationally more expensive. Ignoring the term $\frac{(T-t)!}{\prod_{j=1}^{l}(T_j-t_j)!}$ which is common to both equations, the number of operations for each equation is shown in Table 4.1. The table displays the number of products and divisions operations involved in the computation of $\mathcal{P}(\mathbf{T}|\mathbf{t})$, for both Dirichlet-based methods. The third row shows the additional number of operations that need to be performed by the mixture of Dirichlets method, with respect to the single Dirichlet approach.

4.6 Experiments

In order to test the prior-based SIBP methods a series of experiments have been carried out. The proposed modifications are compared with the full ensemble and the SIBP method in its original formulation with uniform prior. The performance is evaluated using as a benchmark several problems (binary and multiclass) included in the UCI Repository [32]. The fitting algorithm for the Dirichlet distributions used is described in [37] and the FastFit Matlab ToolBox [38].

For each problem, 100 partitions are created, and for each partition a Random Forest ensemble of size $T = 101$ is trained. We compute the mean error rate of the full ensemble

and the mean number of queried trees. It is important to point out that the speed-up rate is computed with respect to the real number of queried trees using the *full-confidence* rule ($\alpha = 1$), and not with respect to the size of the full ensemble (101 in this case). Then SIBP is applied to the ensemble querying process using the following priors: uniform, Dirichlet, mixture of Dirichlets and the non-parametric prior following the hypergeometric re-formulation. The confidence interval is set to $\alpha = 0.99$ for all the experiments. The ensembles are queried until the α confidence is reached. At that point the generalization error rate, number of queried trees, speed-up and the disagreement rates between the SIB-pruned ensemble and the original ones is computed.

Figures A.1, A.2, A.3 A.4 and A.5 display the prior distributions of the tested binary problems using a Dirichlet prior (left column) and a mixture of Dirichlets prior (right column). Figures A.6, A.7 and A.8 display the prior distributions when using the hypergeometric re-formulation of the pruning problem. All the figures display the histogram of the prior probability distribution over p_1 (or t_1 in the case of the hypergeometric re-formulation). In addition, the plots that display the Dirichlet priors show the estimated Dirichlet parameters and the Dirichlet curve itself. A look at the prior distribution plots can provide useful information about the problem in question. Easy classification problems concentrate most of its probability mass close to the end points $p = 0$ and $p = 1$, since they are characterized by easily separable classes. However, hard problems concentrate its mass probability close to $p = 0.5$ in one or two modes, making the task of classification much harder. By comparing the prior distribution plots and the disagreement rates of the random forest ensembles using the original SIBP method, it is clear that easy problems have disagreement rates closer to 0.0 and hard problems have disagreement rates close to 1.0.

Tables 4.2, 4.3, 4.4 and 4.5 show the error rates, disagreement rates, number of queried trees and speed-up rates respectively for all the tested classification problems. The tables display the results of the full ensemble (RF), the full ensemble using the original SIBP method (SIBP), the SIBP using a Dirichlet prior (1DIR), the SIBP using a prior formed by a mixture of Dirichlets (NDIR) and finally the hypergeometric reformulation with prior (HYPER). All the results are displayed as *mean* $\pm$ *std. deviation* over the 100 partitions. The best results for each problem are marked with bold fonts.

From table 4.2 it can be observed that the mean error rates obtained by all pruning methods are similar to the original Random Forests. Although for some problems the mean error rates are slightly worse than the rates obtained by the full ensemble, the difference between them is never greater than 0.3 percentage points. All proposed methods obtain disagreement rates that are closer to 1% than the SIBP method with uniform

Problem	RF	SIBP	1DIR	NDIR	HYPER
australian	**13.00±3.7**	13.09±3.7	13.14±3.7	13.14±3.7	13.25±3.8
breast	**3.22±2.1**	3.23±2.1	3.40±2.3	3.45±2.2	3.76±2.3
diabetes	24.34±4.2	24.25±4.1	24.31±4.1	**24.21±4.0**	24.23±4.0
echocardiogram	22.18±14.3	**22.05±14.7**	22.46±14.6	22.18±14.4	22.18±14.1
german	**23.43±3.5**	23.65±3.3	23.60±3.3	23.61±3.3	23.62±3.3
heart	**18.30±6.9**	18.37±7.0	18.44±7.0	18.44±7.1	18.37±7.2
horse-colic	15.47±5.6	**15.44±5.4**	15.47±5.4	15.49±5.4	15.44±5.4
ionosphere	**6.44±4.1**	6.44±4.1	6.50±4.0	6.55±4.0	6.52±3.9
labor	6.33±8.9	**6.17±8.8**	6.33±8.9	6.33±8.9	6.43±9.1
liver	27.10±6.7	27.09±7.0	27.04±6.9	27.01±6.9	**27.01±6.9**
mushroom	**0.00±0.0**	0.00±0.0	0.00±0.0	0.08±0.2	0.08±0.2
new-thyroid	**4.29±4.0**	4.38±4.0	4.61±4.3	4.94±4.5	4.52±4.3
ringnorm	**7.60±1.3**	7.72±1.2	7.74±1.2	7.78±1.2	7.82±1.2
sonar	**16.25±8.7**	16.45±8.7	16.40±8.7	16.50±8.7	16.45±8.8
spam	**4.59±1.5**	4.63±1.5	4.72±1.5	4.79±1.5	4.86±1.4
threenorm	**17.85±1.1**	18.04±1.1	17.94±1.1	17.95±1.1	17.97±1.1
tic-tac-toe	**1.05±1.1**	1.16±1.1	1.20±1.2	1.34±1.3	1.72±1.5
twonorm	**4.66±0.6**	4.77±0.6	4.76±0.6	4.90±0.6	4.90±0.6
votes	**4.05±2.9**	4.12±2.9	4.23±2.9	4.23±2.9	4.30±2.9
waveform	**17.30±0.9**	17.36±0.8	17.65±0.8	17.46±0.8	17.42±0.8
wine	**1.69±2.8**	1.74±2.8	2.19±3.2	2.24±3.4	2.13±3.2

TABLE 4.2: Error rates comparison of the SIBP variants (for each dataset the best method is high-lighted in boldface)

prior, except in problems *liver*, *sonar* and *threenorm*. In these problem the prior distributions over $\mathbf{p}$ (see Appendix A) are somehow similar to a uniform distribution and the disagreement for the original SIBP method is already very close to the expected one. Nonetheless, the disagreement rates of the proposed methods do not divert significantly from the expected 1%. From the proposed methods, the one that shows the poorest fit of the disagreement rate to the theoretial bound is 1DIR. NDIR improves significantly the fit of the disagreement, possibly because it can better model bimodal distributions. Finally the HYPER method achieves the best fit of the disagreement rates, except in the *echocardiogram* problem.

As for the number of queried trees (see table 4.4) and speed-up rates (table 4.5), the proposed methods improve with respect to the SIB-pruned ensemble in many of the tested datasets. The benefit of using the prior distribution becomes more evident in the *mushroom* problem. For this problem, the predictions of most classifiers agree as it can be observed in figures A.4(a) and A.7(c). That information is incorporated into the pruning method, that queries a mean number of just one or two classifiers before casting a decision, achieving a speed-up rate of 51 times. Other problems in which the proposed methods improve the speed-up rates are *australian*, *breast*, *ionosphere*,

Problem	SIBP	1DIR	NDIR	HYPER
australian	0.3±0.6	0.5±0.8	0.4±0.8	0.9±1.1
breast	0.1±0.4	0.3±0.7	0.5±0.9	1.0±1.1
diabetes	0.6±0.9	0.8±1.1	0.7±1.0	0.8±1.0
echocardiogram	0.7±3.1	0.8±3.3	0.8±3.3	1.4±4.6
german	0.8±0.8	0.9±0.9	0.8±0.8	0.8±0.9
heart	0.8±1.8	1.0±1.9	1.0±2.0	1.0±2.1
horse-colic	0.4±0.9	0.4±0.9	0.5±1.0	0.7±1.3
ionosphere	0.1±0.6	0.2±0.7	0.3±0.9	0.7±1.3
labor	0.2±1.7	0.3±2.3	0.3±2.3	1.2±4.5
liver	1.0±1.7	0.8±1.4	0.8±1.4	0.9±1.5
mushroom	0.0±0.0	0.0±0.0	0.1±0.2	0.1±0.2
new-thyroid	0.1±0.7	0.5±1.5	0.9±2.2	0.5±1.5
ringnorm	0.5±0.2	0.5±0.2	0.6±0.2	0.8±0.3
sonar	0.9±2.0	0.6±1.8	0.8±1.9	0.8±1.9
spam	0.1±0.2	0.4±0.3	0.5±0.4	0.7±0.4
threenorm	1.0±0.2	0.7±0.1	0.7±0.2	0.8±0.2
tic-tac-toe	0.1±0.4	0.2±0.4	0.3±0.6	0.7±1.0
twonorm	0.4±0.1	0.4±0.1	0.7±0.2	0.7±0.2
votes	0.1±0.4	0.4±1.1	0.5±1.2	1.0±1.8
waveform	0.6±0.1	1.7±0.4	1.0±0.2	0.8±0.2
wine	0.1±0.6	0.8±2.1	0.9±2.3	0.9±2.0

TABLE 4.3: Disagreement rates comparison of the SIBP variants (for each dataset the best method is high-lighted in boldface)

new-thyroid, *spam*, *tic-tac-toe*, *votes* and *wine*.

From the observations of tables 4.3 and 4.5 we consider that the Hypergeometric formulation with incorporated prior is the best of the new proposed methods in terms of best fitting the disagreement rates and speeding-up the classification process. This can be explained given that this method does not assume that the prior follows any parametric distribution.

4.7 Conclusions

The *Statistical Instance-Based Pruning* method is a dynamic pruning algorithm for classification ensembles. This pruning technique queries sequentially the classifiers of the ensemble until the probability that the output class by the classifiers queried up until that point and the predicted output class of the full ensemble is the same exceeds a preset threshold α. The original pruning technique is based on a scheme of majority voting and does not include prior knowledge about the classification problem. The disagreement rates between the full ensembles and the SIB-pruned ensembles are often much lower than the theoretical bound given by $1 - \alpha$. In order to better fit the empirical

Problem	RF	SIBP	1DIR	NDIR	HYPER
australian	62.2±1.4	16.1±2.1	14.6±2.2	14.7±2.1	**12.8±2.3**
breast	54.2±0.9	8.9±1.4	5.8±1.3	5.1±1.1	**4.0±1.0**
diabetes	68.8±1.8	24.9±3.2	**23.7±3.3**	24.5±3.3	24.0±3.2
echocardiogram	68.0±4.6	22.6±8.2	21.5±8.1	21.8±8.2	**20.0±8.0**
german	71.8±1.3	28.4±2.8	**27.3±2.8**	28.1±2.8	27.7±2.9
heart	67.2±2.5	22.5±4.2	21.7±4.3	21.4±4.2	**20.9±4.2**
horse-colic	66.2±2.1	20.2±3.5	19.6±3.5	19.0±3.5	**17.5±3.7**
ionosphere	57.9±1.5	11.9±2.3	9.9±2.3	9.3±2.3	**7.8±2.1**
labor	61.6±4.0	14.1±6.0	12.5±6.0	12.1±6.0	**9.7±5.3**
liver	74.5±2.3	31.8±4.5	32.6±4.5	32.5±4.5	**31.7±4.5**
mushroom	51.0±0.0	6.0±0.0	3.0±0.0	**1.0±0.0**	**1.0±0.0**
new-thyroid	55.2±1.8	10.7±2.6	6.2±2.4	**5.4±2.2**	6.3±2.4
ringnorm	68.6±0.8	22.9±1.1	22.6±1.3	21.3±1.4	**20.4±1.5**
sonar	73.9±3.0	**32.1±6.6**	32.7±6.6	**32.1±6.4**	32.6±6.8
spam	57.1±0.3	11.1±0.5	8.5±0.6	7.9±0.7	**7.2±0.6**
threenorm	76.6±0.5	**34.8±1.0**	36.6±1.1	36.3±1.1	35.8±1.6
tic-tac-toe	60.7±0.9	12.8±1.4	11.1±1.3	9.4±1.3	**7.8±1.2**
twonorm	67.2±0.2	21.0±0.5	20.9±0.5	18.6±0.6	**18.4±0.9**
votes	54.5±1.2	8.8±1.8	5.8±1.7	5.1±1.6	**4.1±1.4**
waveform	72.3±0.7	29.3±1.1	**24.9±1.1**	26.9±1.2	28.1±1.6
wine	57.3±2.1	11.4±2.7	6.7±2.4	**6.1±2.2**	**6.1±1.8**

TABLE 4.4: Number of queried trees comparison of the SIBP variants (for each dataset the best method is high-lighted in boldface)

disagreement rates to the theoretical bound, we propose to incorporate prior knowledge about the actual classification problem to the SIBP method. Three different methods have been developed and tested in this work that incorporate prior knowledge over the distribution of class votes from specific problem into the SIBP. The first method incorporates the prior knowledge by modelling the prior probability distribution after a Dirichlet distribution. The second method models the prior distribution after a mixture of Dirichlet distributions that can fit multimodal distributions. The third method proposes a reformulation of the SIBP method in which the voting process is modeled after a hypergeometric distribution. In this case the prior is modelled using a non-parametric distribution.

The proposed methods succeed in achieving disagreement rates that are closer to the preset $1 - \alpha$ with respect to the ones obtained by the original SIBP formulation. The method that obtains disagreement rates closer to the fixed one is the hypergeometric formulation with non-parametric prior. Obtaining disagreement rates close to the desired one means that the pruning technique has a more reliable and foreseeable behavior. Additionally, the proposed methods obtain a generalization performance very close to the one obtained by the full ensemble. Furthermore, the proposed methods significantly

Problem	SIBP	1DIR	NDIR	HYPER
australian	6.6±0.4	8.1±0.7	8.0±0.5	**10.0±0.9**
breast	8.0±0.2	14.8±0.4	16.3±0.5	**22.4±1.5**
diabetes	5.6±0.4	**6.8±0.8**	6.2±0.5	6.4±0.6
echocardiogram	5.7±1.1	6.8±1.7	6.6±1.4	**7.8±2.0**
german	5.1±0.4	**6.0±0.5**	5.7±0.5	5.8±0.5
heart	5.8±0.6	6.5±0.9	6.6±0.9	**6.8±1.0**
horse-colic	5.9±0.5	6.6±0.7	6.8±0.7	**7.9±1.2**
ionosphere	7.3±0.4	10.2±0.6	11.0±0.7	**14.0±1.0**
labor	6.7±1.3	8.7±1.8	8.6±1.9	**11.1±2.5**
liver	**4.6±0.6**	4.3±0.6	4.3±0.6	4.4±0.6
mushroom	8.5±0.0	17.0±0.1	**51.0±0.0**	**51.0±0.0**
new-thyroid	6.8±0.3	15.8±1.1	**19.3±3.0**	15.1±1.2
ringnorm	5.5±0.3	5.8±0.5	6.3±0.7	**6.6±0.6**
sonar	**4.7±0.7**	4.4±0.6	4.6±0.7	4.4±0.7
spam	7.5±0.1	11.4±0.2	13.2±1.2	**14.5±0.7**
threenorm	**4.4±0.2**	3.7±0.2	3.7±0.2	3.8±0.3
tic-tac-toe	6.9±0.3	9.3±0.5	10.7±0.6	**12.8±1.4**
twonorm	5.8±0.1	5.8±0.2	**7.2±0.2**	7.1±0.3
votes	7.9±0.3	14.5±0.7	16.2±0.8	**20.7±3.0**
waveform	4.5±0.1	**7.8±0.4**	5.9±0.3	5.2±0.4
wine	6.5±0.5	15.1±1.4	**15.2±1.3**	13.8±1.3

TABLE 4.5: Speed-up rates comparison of the SIBP variants (for each dataset the best method is high-lighted in boldface)

improve the classification speed, especially for datasets where the mass of the prior is close to 0 and 1. For the analyzed datasets, the method based on the hypergeometric reformulation with non-parametric prior achieves the best results both getting closer to the disagreement rate and speeding-up the classification process.

Chapter 5

Conclusions

In this work two contributions have been made to ensemble pruning. The first contribution is a double pruning scheme, presented in Chapter 3. This double scheme combines the two types of ensemble pruning techniques: static and dynamic. Specifically the scheme consists in applying a static method that reduces the number of classifiers of the original ensemble and improves the accuracy with respect to the full ensemble. Using only the classifiers selected by the static method, the classification is performed using the dynamic *Statistical Instance-Based Pruning* (SIBP) method. Applying SIBP allows us to further reduce the classification time and keep the accuracy of the statically pruned ensembles. Two variants of the double pruning scheme were presented: one for parallel ensembles and one for sequential ensembles. The *parallel* variant uses *boosting-based ordering* with *early-stopping* to reduce the size of the ensemble and the *sequential* counterpart uses SDP pruning (or its greedy version).

Based on the observation that ordered bagging ensemble reach a local minima in its accuracy after querying around 20% of its classifiers, the boosting-based bagging ensembles are pruned off-line at that point. When SIBP is applied to the off-line pruned ensemble, the number of queried classifiers is decreased to 5-12% of all classifiers of complete the ensemble, depending on the problem. For boosting ensembles the static method selects a subensemble with 50% of the original classifiers. After applying SIBP, the number of queried trees during the classification phase is reduced to 8-30% of all the original classifiers.

The second contribution of this Master Thesis, described in Chapter 4, is an updated version of the SIBP method. The SIBP queries sequentially the classifiers of the ensemble until the probability that the output class by the classifiers queried up until that point and the predicted output class of the full ensemble is the same exceeds a preset threshold α. The original formulation of the SIBP method does not have into account previous

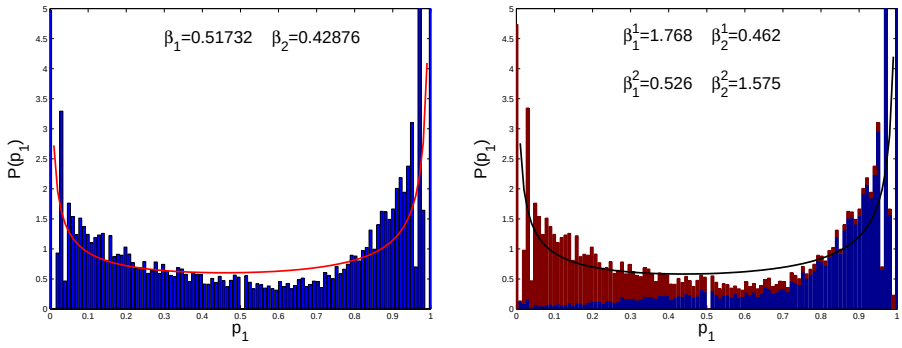
knowledge about the class votes distribution of the classification problem. We have proposed three different ways to include this prior knowledge from estimates obtained from the training data. The first proposal includes a Dirichlet prior distribution. The second one includes a probability distribution that follows a mixture of Dirichlets. Finally the third method, is a reformulation of the SIBP method based on the hypergeometric distribution, equivalent to the original, that permits to introduce a non-parametric prior distribution. The prior-based SIBP methods succeed at achieving disagreement rates (with respect to the original ensemble) closer to the theoretical bound given by $1 - \alpha$. One direct consequence is that the number of queried classifiers is further reduced, especially for easy separable problems. The main consequence is that the proposed pruning technique has a more reliable and foreseeable behavior with respect to the disagreement rate which is a necessary property for the application of these techniques. The proposal based on the non-parametric prior and hypergeometric distribution is the one that delivers better results.

Appendix A

Prior Probability Distributions

In this Appendix, all the prior distribution plots are included for the binary classification problems. It is important to remark that, in order to facilitate the visualization of the prior distributions, the plots displayed here contain all the out-of-bag votes from all the dataset partitions. Therefore, the plots might be more optimistic than the actual distributions used during the classification process, which only contain the out-of-bag information from one of the partitions.

(a) Australian dataset



(b) Breast dataset

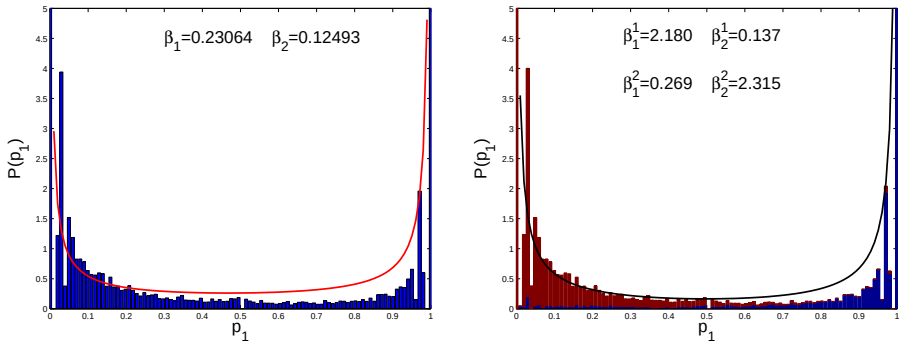
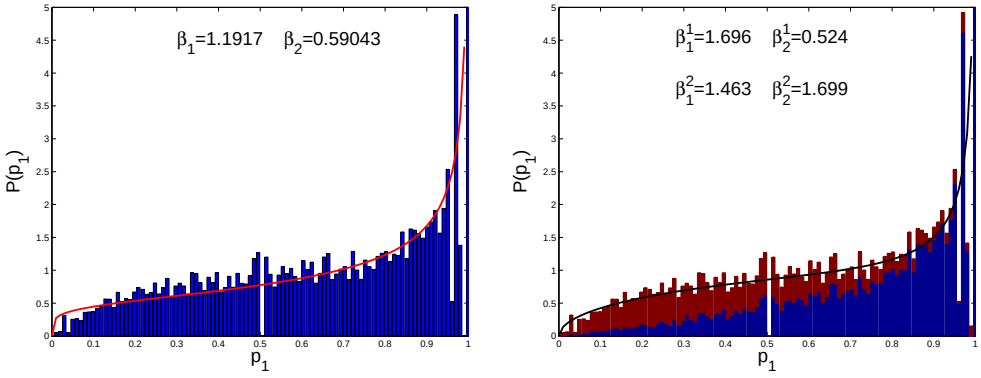
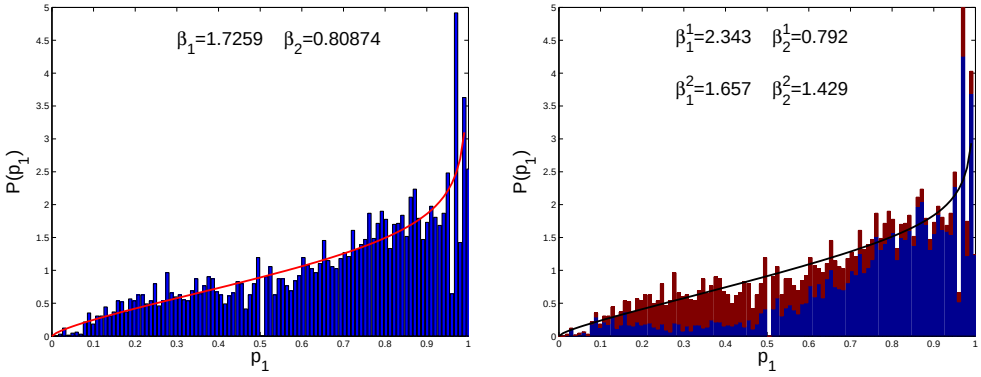


FIGURE A.1: Dirichlet-based Prior Distributions for datasets (a) Australian and (b) Breast

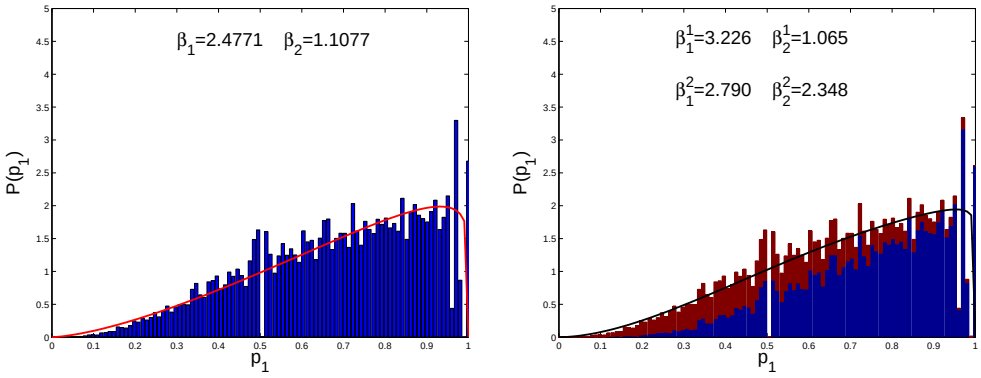
(a) Diabetes dataset



(b) Echocardiogram dataset



(c) German dataset



(d) Heart dataset

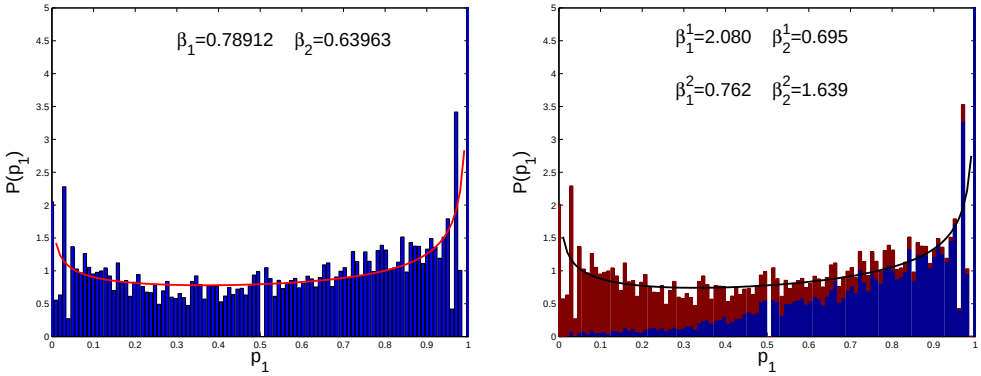
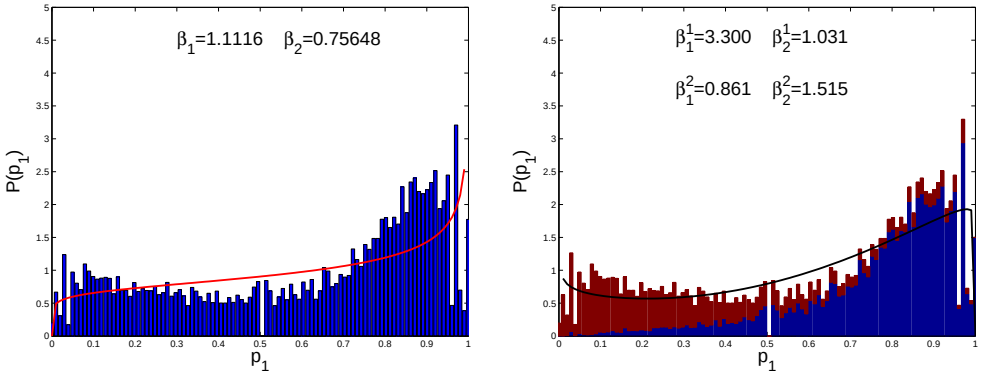
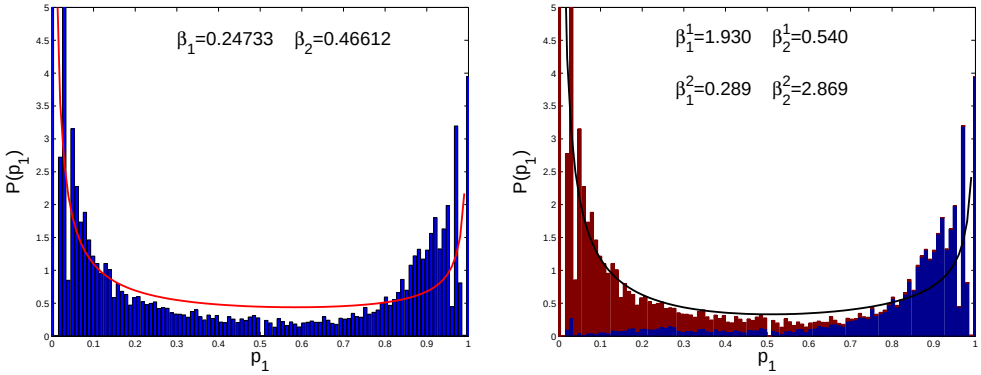


FIGURE A.2: Dirichlet-based Prior Distributions for datasets (a) Diabetes, (b) Echocardiogram, (c) German and (d) Heart

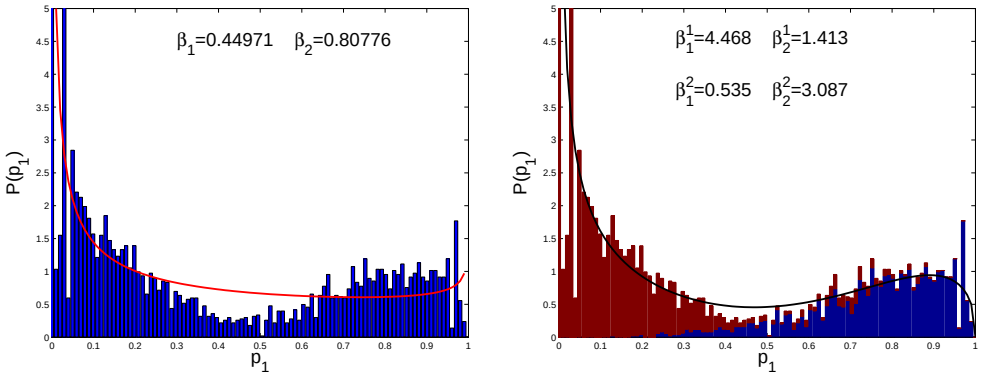
(a) Horse-colic dataset



(b) Ionosphere dataset



(c) Labor dataset



(d) Liver dataset

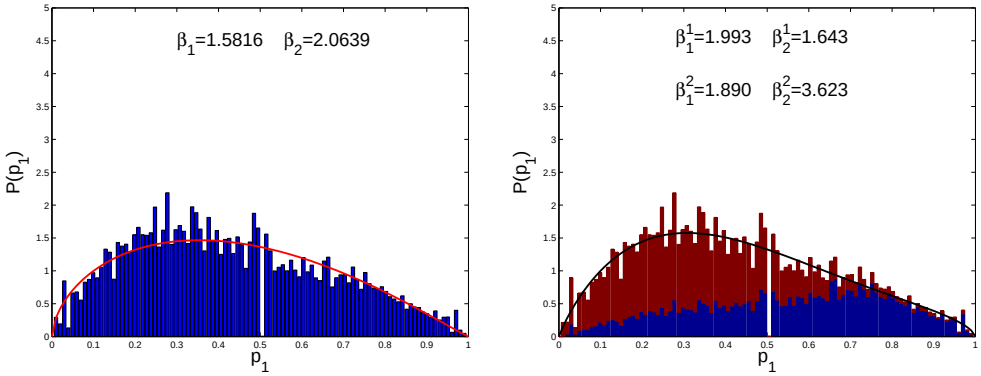


FIGURE A.3: Dirichlet-based Prior Distributions for datasets (a) Horse-colic, (b) Ionosphere, (c) Labor and (d) Liver

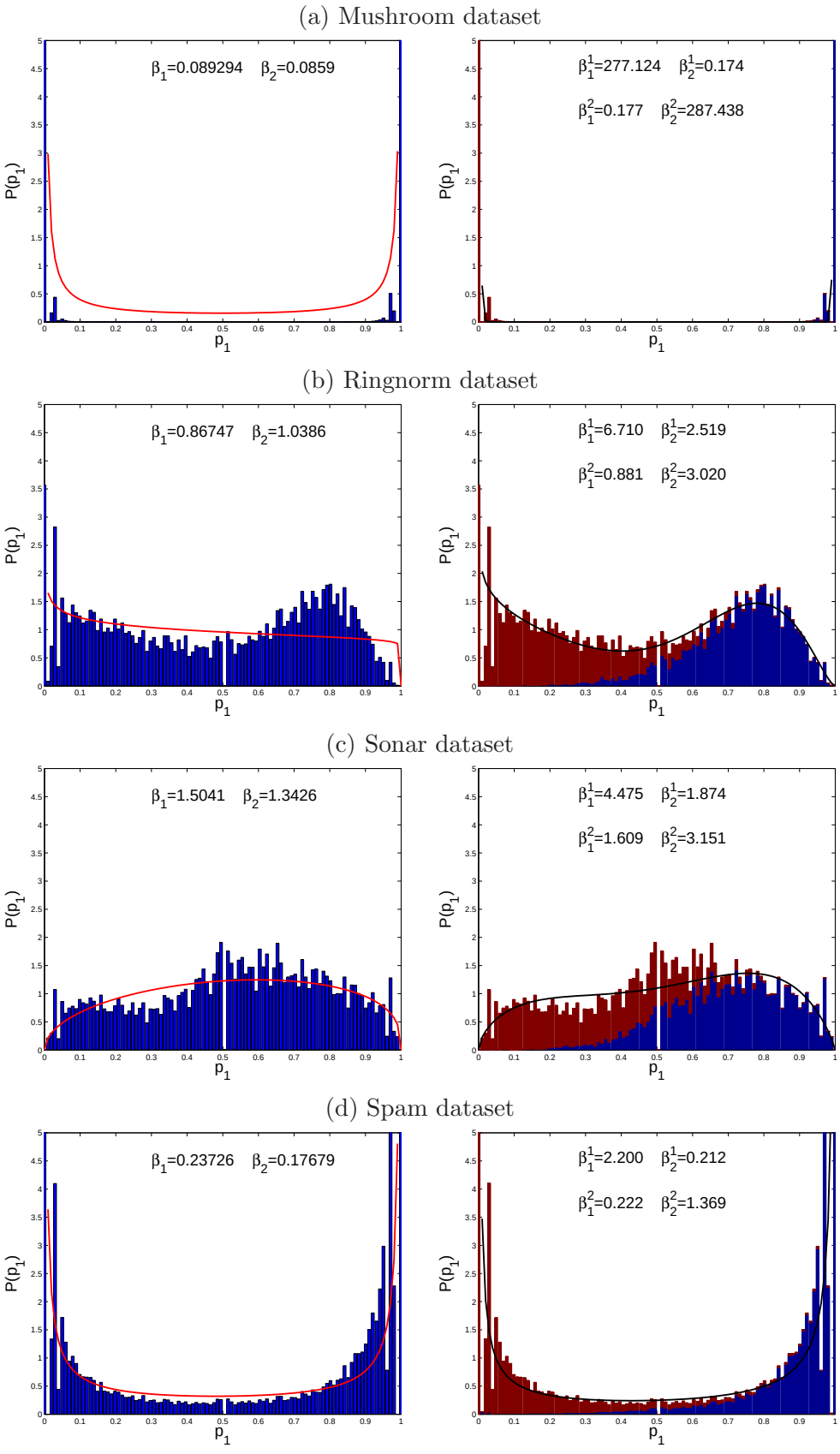


FIGURE A.4: Dirichlet-based Prior Distributions for datasets (a) Mushroom, (b) Ringnorm, (c) Sonar and (d) Spam

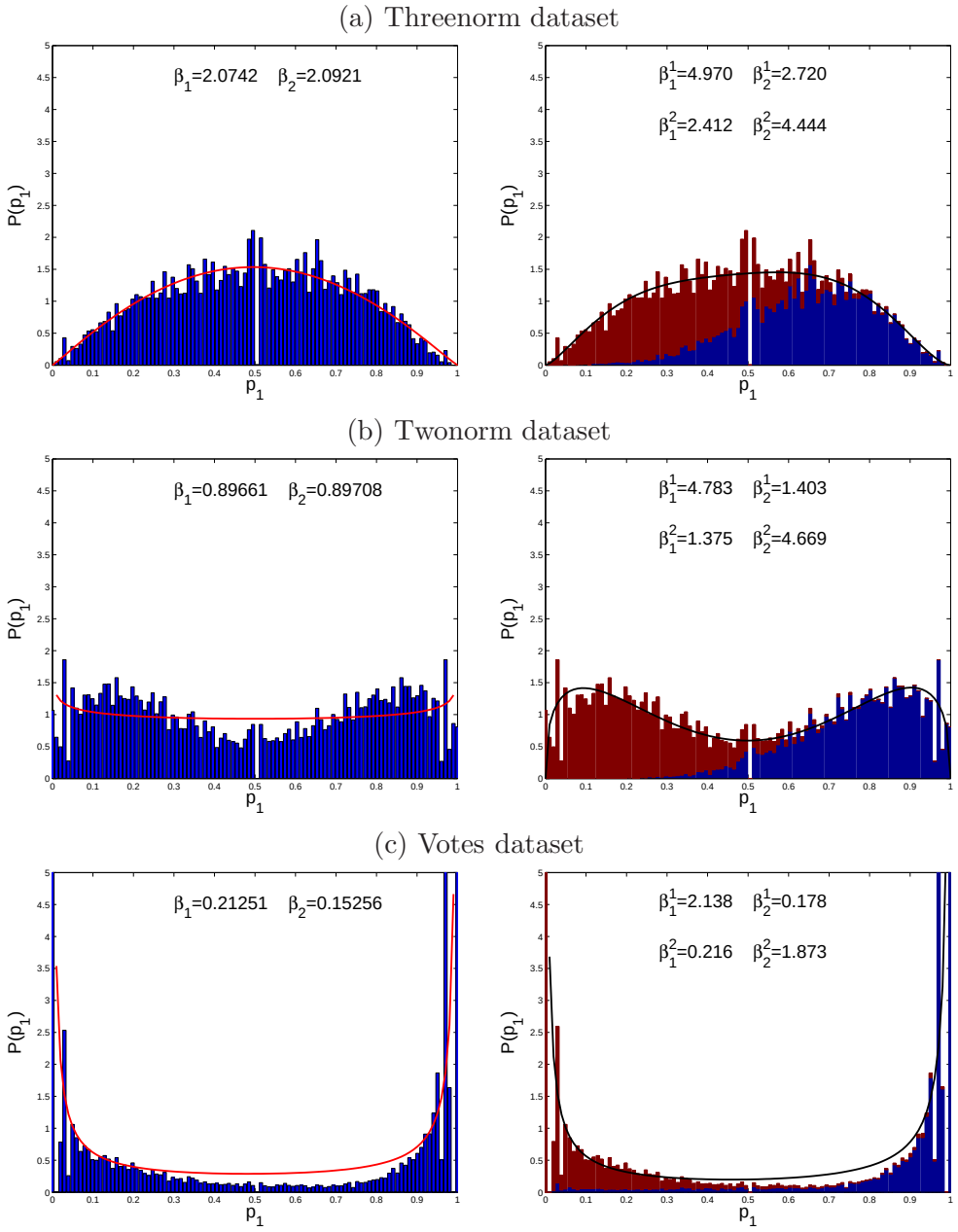


FIGURE A.5: Dirichlet-based Prior Distributions for datasets (a) Threenorm, (b) Twonorm and (c) Votes

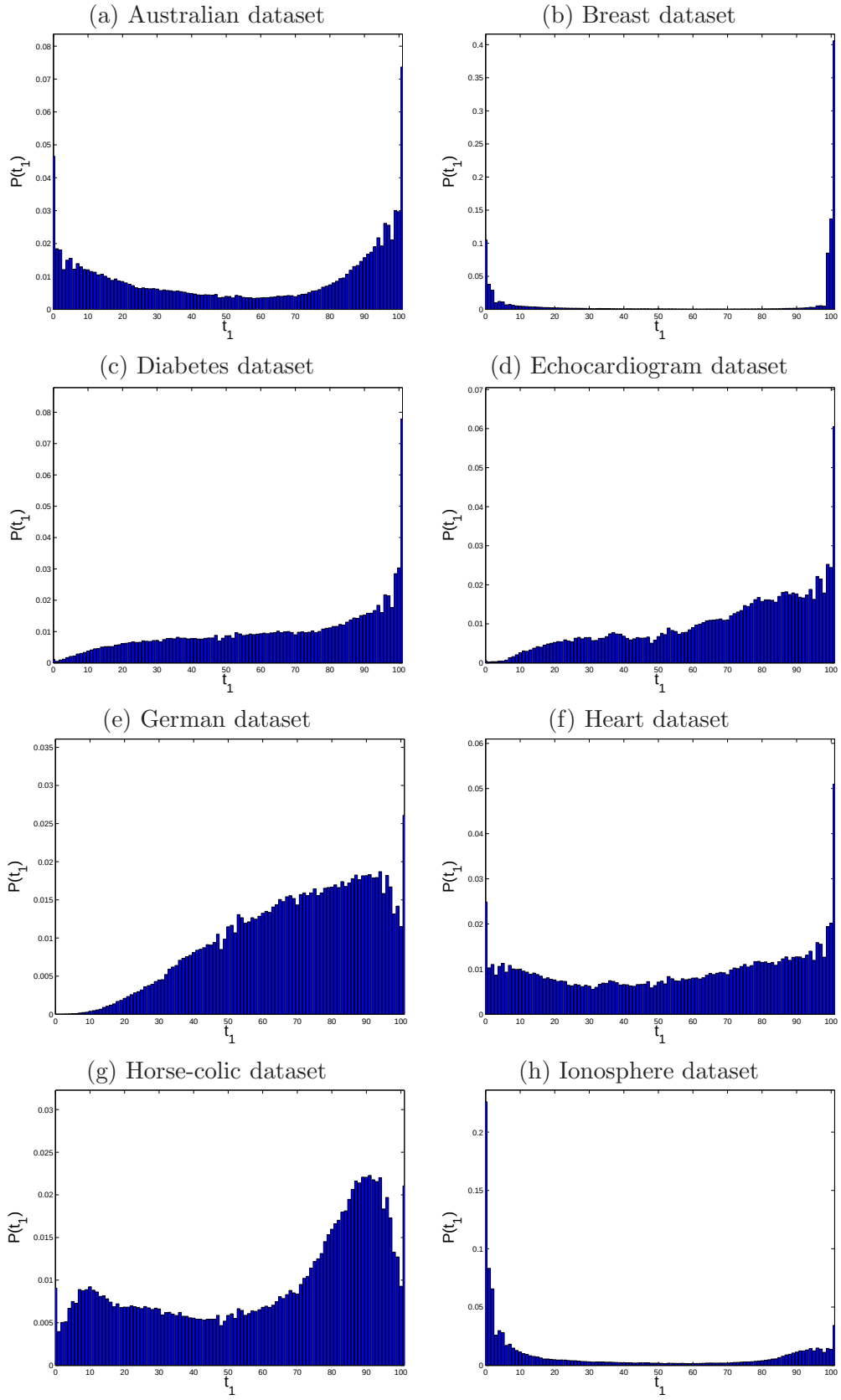


FIGURE A.6: Non-parametric Prior Distributions for datasets (a) Australian, (b) Breast, (c) Diabetes, (d) Echocardiogram, (e) German, (f) Heart, (g) Horse-colic and (h) Ionosphere

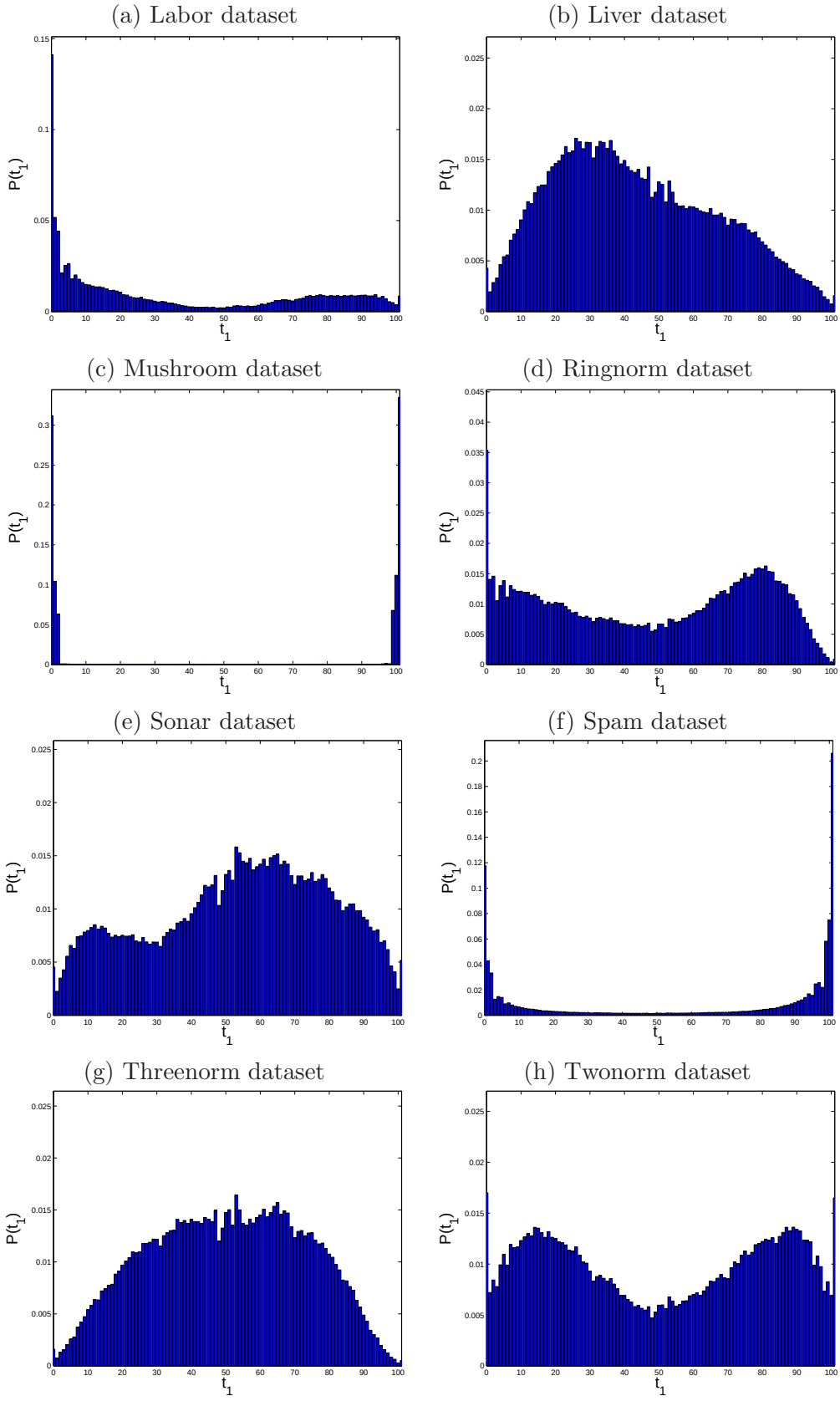


FIGURE A.7: Non-parametric Prior Distributions for datasets (a) Labor, (b) Liver, (c) Mushroom, (d) Ringnorm, (e) Sonar, (f) Spam, (g) Threenorm and (h) Twonorm

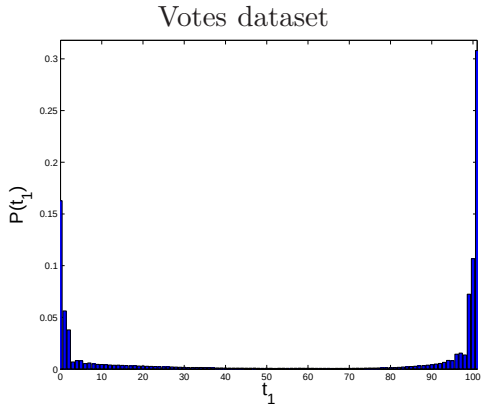


FIGURE A.8: Non-parametric Prior Distributions for Votes dataset

Appendix B

On the SIBP equivalence

In order to prove the equivalence between the original formulation of the SIBP method and the new formulation when the prior distribution is uniform, we first need to know three important results, shown in the theorem and the proposition below. Notice that in order to keep simple the notation and simplify the computations, it is assumed that the classification problem is binary. However it is entirely possible to generalize all the results shown here to a multiclass problem.

Theorem B.1. *Chu-Vandermonde Identity (Generalization of the Vandermonde Identity)*
Let $s, t \in \mathbb{C}$ and $n \in \mathbb{N}$, then

$$\binom{s+t}{n} = \sum_{k=0}^{n} \binom{s}{k}\binom{t}{n-k} \tag{B.1}$$

Proposition B.2. *Upper negation*
Let $r \in \mathbb{C}$ and $k \in \mathbb{Z}$, then

$$\binom{r}{k} = (-1)^k \binom{k-r-1}{k} \tag{B.2}$$

This proposition is the key to proving the equivalence between the two formulations. The previous theorem and proposition are used here.

Proposition B.3. *Let t_1 and t_2 be positive integers such that $t_1 + t_2 = t$ and $t \leq T$. Then*

$$\sum_{i=t_1}^{T-t_2} \binom{i}{t_1}\binom{T-i}{t_2} = \binom{T+1}{T-t} \tag{B.3}$$

Proof. The difficulty of this proof lies in the series indices being situated in the upper part of the Newton's binomial. We first use the symmetry property to bring down the

indices

$$\sum_{i=t_1}^{T-t_2}\binom{i}{t_1}\binom{T-i}{t_2}=\sum_{i=t_1}^{T-t_2}\binom{i}{i-t_1}\binom{T-i}{T-i-t_2} \tag{B.4}$$

The upper indices are removed by applying the upper negation property B.2

$$\sum_{i=t_1}^{T-t_2}\binom{i}{i-t_1}\binom{T-i}{T-i-t_2}=\sum_{i=t_1}^{T-t_2}\binom{-t_1-1}{i-t_1}\binom{-t_2-1}{T-i-t_2}(-1)^{i-t_1}(-1)^{T-i-t_2} \tag{B.5}$$

Now the Vandermonde Convolution B.1 can be applied

$$\sum_{i=t_1}^{T-t_2}\binom{-t_1-1}{i-t_1}\binom{-t_2-1}{T-i-t_2}(-1)^{i-t_1}(-1)^{T-i-t_2}=\binom{-t-2}{T-t}(-1)^{T-t} \tag{B.6}$$

Finally the upper negation is applied and the desired result is obtained

$$\binom{-t-2}{T-t}(-1)^{T-t}=\binom{T+1}{T-t}(-1)^{2(T-t)}=\binom{T+1}{T-t} \tag{B.7}$$

□

Proposition B.4. *Following the hypergeometric reformulation as explained in section 4.4 and assuming that the prior distribution follows a uniform distribution, the hypergeometric reformulation and its original are equivalent.*

Proof. Let $t_1, t_2 \in \mathbb{N}$ such that $t_1 + t_2 = t$ and $t \leq T = T_1 + T_2$.

$$\mathcal{P}(\mathbf{T}|\mathbf{t})=\frac{\mathcal{P}(\mathbf{t}|\mathbf{T})\mathcal{P}(\mathbf{T})}{\mathcal{P}(\mathbf{t})}=\frac{\mathcal{P}(\mathbf{t}|\mathbf{T})\mathcal{P}(\mathbf{T})}{\sum_{T_1^*=t_1}^{T-t_2}\mathcal{P}(\mathbf{t}|\mathbf{T}^*)\mathcal{P}(\mathbf{T}^*)} \tag{B.8}$$

Following the formulas in section 4.4, $\mathcal{P}(\mathbf{t}|\mathbf{T})=\frac{\binom{T_1}{t_1}\binom{T_2}{t_2}}{\binom{T}{t}}$ and $\mathcal{P}(\mathbf{T})=\frac{1}{T+1}$. We use the results on proposition B.3 to simplify the denominator and the original SIBP equation 2.38 is obtained:

$$\begin{aligned}\mathcal{P}(\mathbf{T}|\mathbf{t}) &= \frac{\binom{T_1}{t_1}\binom{T_2}{t_2}}{\sum_{T_1^*=t_1}^{T-t_2}\binom{T_1^*}{t_1}\binom{T_2^*}{t_2}}=\frac{\binom{T_1}{t_1}\binom{T_2}{t_2}}{\binom{T+1}{T-t}}=\frac{\frac{T_1!}{t_1!(T_1-t_1)!}\frac{T_2!}{t_2!(T_2-t_2)!}}{\frac{(T+1)!}{(T-t)!(t+1)!}}\\ &= \frac{(T-t)!}{\prod_{i=1}^{2}(T_i-t_i)!}\frac{\prod_{i=1}^{2}(t_i+1)_{T_i-t_i}}{(t+2)_{T-t}}\end{aligned} \tag{B.9}$$

□

Bibliography

[1] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:993–1001, 1990.

[2] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[3] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proc. of the 13th International Conf. on Machine Learning*, pages 148–156. Morgan Kaufmann, 1996.

[4] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[5] G. Martinez-Munoz and A. Suarez. Switching class labels to generate classification ensembles. *Pattern Recognition*, 38(10):1483–1494, 2005.

[6] Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems: First International Workshop*, pages 1–15, 2000.

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

[8] Ludmila I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms.* Wiley-Interscience, 2004. ISBN 0471210781.

[9] Leo Breiman. Out-of-bag estimation. Technical report, Statistics Department, University of California, 1996.

[10] Leo Breiman. Heuristics of instability and stabilization in model selection. *The Annals of Statistics*, 24(6):2350–2383, December 1996.

[11] Michael Kearns and Leslie Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the Association for Computing Machinery*, 41:67–95, January 1994.

[12] Robert E. Schapire. The strength of weak learnability. *Mach. Learn.*, 5:197–227, July 1990.

[13] Yoav Freund. Boosting a weak learning algorithm by majority. *Inf. Comput.*, 121: 256–285, September 1995. ISSN 0890-5401. doi: 10.1006/inco.1995.1136. URL http://portal.acm.org/citation.cfm?id=220262.220446.

[14] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proc. of the 2nd European Conference on Computational Learning Theory*, pages 23–37, 1995.

[15] Y. Freund and R. Schapire. A short introduction to boosting. *Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999.

[16] Leo Breiman. Arcing classifiers. *The Annals of Statistics*, 26(3):801–849, 1998.

[17] R.E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 12 (5):1651–1686, 1998.

[18] Dragos D. Margineantu and Thomas G. Dietterich. Pruning adaptive boosting. In *Proc. of the 14th International Conference on Machine Learning*, pages 211–218. Morgan Kaufmann, 1997.

[19] Z.-H. Zhou, J. Wu, and W. Tang. Ensembling neural networks: Many could be better than all. *Artificial Intelligence*, 137(1-2):239–263, 2002.

[20] Gonzalo Martínez-Muñoz and Alberto Suárez. Aggregation ordering in bagging. In *Proc. of the IASTED International Conference on Artificial Intelligence and Applications*, pages 258– 263. Acta Press, 2004.

[21] Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *Proc. of the 21st International Conference on Machine Learning*, page 18, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-828-5.

[22] Robert E. Banfield, Lawrence O. Hall, Kevin W. Bowyer, and W. Philip Kegelmeyer. Ensemble diversity measures and their application to thinning. *Information Fusion*, 6(1):49–62, 2005.

[23] Yi Zhang, Samuel Burer, and W. Nick Street. Ensemble pruning via semi-definite programming. *Journal of Machine Learning Research*, 7:1315–1338, 2006.

[24] Gonzalo Martínez-Muñoz, Daniel Hernández-Lobato, and Alberto Suárez. An analysis of ensemble pruning techniques based on ordered aggregation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):245–259, 2009.

[25] Daniel Hernández-Lobato, Gonzalo Martínez-Muñoz, and Alberto Suárez. Statistical instance-based pruning in ensembles of independent classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):364–369, 2009.

[26] Patrice Latinne, Olivier Debeir, and Christine Decaestecker. Limiting the number of trees in random forests. In *Multiple Classifier Systems*, pages 178–187. 2001. URL http://dx.doi.org/10.1007/3-540-48219-9_18.

[27] Amanda Sharkey, Noel Sharkey, Uwe Gerecke, and G. Chandroth. The Test and select approach to ensemble combination. In *Multiple Classifier Systems*, pages 30–44. 2000. URL http://dx.doi.org/10.1007/3-540-45014-9_3.

[28] Gonzalo Martínez-Muñoz and Alberto Suárez. Using boosting to prune bagging ensembles. *Pattern Recognition Letters*, 28(1):156–165, 2007.

[29] Gonzalo Martínez-Muñoz and Alberto Suárez. Pruning in ordered bagging ensembles. In *Proc. of the 23rd International Conference on Machine Learning*, pages 609–616, 2006.

[30] Wei Fan, Fang Chu, Haixun Wang, and Philip S. Yu. Pruning and dynamic scheduling of cost-sensitive ensembles. In *Proc. of the 18th National Conference on Artificial Intelligence*, pages 146–151. American Association for Artificial Intelligence, 2002.

[31] Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 226–235, New York, NY, USA, 2003. ACM Press. ISBN 1-58113-737-0. doi: http://doi.acm.org/10.1145/956750.956778.

[32] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007. URL http://www.ics.uci.edu/~mlearn/MLRepository.html.

[33] Víctor Soto, Gonzalo Martínez-Muñoz, Daniel Hernández-Lobato, and Alberto Suárez. A double pruning algorithm for classification ensembles. In *MCS*, pages 104–113, 2010.

[34] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman & Hall, New York, 1984.

[35] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006. ISSN 1533-7928.

[36] Gonzalo Martínez-Muñoz, Daniel Hernández-Lobato, and Alberto Suárez. Statistical instance-based ensemble pruning for multi-class problems. In *ICANN (1)*, pages 90–99, 2009.

[37] Thomas Minka. Estimating a dirichlet distribution. Website, 2003. URL http://research.microsoft.com/en-us/um/people/minka/papers/dirichlet/.

[38] Thomas Minka. Fastfit matlab toolbox. Website. URL http://research.microsoft.com/enus/um/people/minka/software/fastfit/.