Universidad Autónoma de Madrid

Master's Thesis

Ensemble Learning in the Presence of Noise

Author:
Maryam Sabzevari

Supervisors:
Dr. Gonzalo Martínez Muñoz,
Dr. Alberto Suárez González

Submitted in partial fulfillment of the requirements for the degree of Master in Investigation and Innovation in Information and Communication Technologies

in the

Computer Science Department,
Escuela Politécnica Superior,
Universidad Autónoma de Madrid,
C/ Francisco Tomás y Valiente, 11,
Madrid 28049 Spain.

June 2015

Declaration of Authorship

I, Maryam Sabzevari, declare that this thesis titled, 'Ensemble Learning in the Presence of Noise' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: __________________________________

Date: __________________________________

"He who loves practice without theory is like the sailor who boards ship without a rudder and compass and never knows where he may be cast."

Leonardo da Vinci

UNIVERSIDAD AUTÓNOMA de MADRID

Abstract

Escuela Politécnica Superior

Ensemble Learning in the Presence of Noise

by Maryam Sabzevari

Learning in the presence of noise is an important issue in machine learning. The design and implementation of effective strategies for automatic induction from noisy data is particularly important in real-world problems, where noise from defective collecting processes, data contamination or intrinsic fluctuations is ubiquitous. There are two general strategies to address this problem. One is to design a robust learning method. Another one is to identify noisy instances and eliminate or correct them.

In this thesis we propose to use ensembles to mitigate the negative impact of mislabelled data in the learning process. In ensemble learning the predictions of individual learners are combined to obtain a final decision. Effective combinations take advantage of the complementarity of these base learners. In this manner the errors incurred by a learner can be compensated by the predictions of other learners in the combination.

A first contribution of this work is the use of subsampling to build bootstrap ensembles, such as bagging and random forest, that are resilient to class label noise. By using lower sampling rates, the detrimental effect of mislabelled examples on the final ensemble decisions can be tempered. The reason is that each labelled instance is present in a smaller fraction of the training sets used to build individual learners. Ensembles can also be used as a noise detection procedure to improve the quality of the data used for training. In this strategy, one attempts to identify noisy instances and either correct (by switching their class label) or discard them. A particular example is identified as noise if a specified percentage (greater than 50%) of the learners disagree with the given label for this example. Using an extensive empirical evaluation we demonstrate the use of subsampling as an effective tool to detect and handle noise in classification problems.

Acknowledgements

Foremost, I wish to express my sincere gratitude to my supervisors Dr. Alberto Suárez González and Dr. Gonzalo Martínez Muñoz who have supported and helped me with their invaluable instructions, not only in my master thesis, but also during my whole study in UAM.

Dr. Alberto Suárez González gave me the opportunity to conduct my study in machine learning. He has helped me with his excellent guidance from the beginning of my study in UAM and I would like to express my deepest appreciation for his generous contribution of knowledge and his invaluable helps. I am truly thankful to Dr. Gonzalo Martínez Muñoz, for his extremely helpful suggestions, guidance and advice. I really appreciate him for always willing to help and share his knowledge and experiences.

I would like to thank professors in the machine learning group in UAM, specially Dr. José Ramón Dorronsoro Ibero and Dr. Daniel Hernández Lobato. I take this opportunity to thank all of my master professors for many things that I could learn from them. I would like also to express my gratitude to Dr. Ángela Fernández Pascual and Dr. Carlos María Alaíz Gudín for their friendships and academic help.

Finally, my sincere thanks also goes to my family: my husband, Vahid, for his encouragement, attention, unconditional support and help in academic and non-academic issues and my parents and my sister who always have supported me kindly throughout my life.

Contents

Chapter 1

Introduction

The success of margin-based predictors, such as SVM's or Adaboost ensembles in many classification tasks of practical interest [1–4] has prompted many researchers to posit that large margins are a key feature in explaining the generalization capacity of these types of classifiers. In ensembles, the margin is defined as the weighted sum of votes for the correct class minus the weighted sum of votes for the class that receives most votes and is different from the correct one. The predictive accuracy of boosting has been ascribed to the fact that the classification margins of the training examples are effectively increased by the progressive focus on training examples that are difficult to classify [5]. Nonetheless there are some empirical studies that put in doubt the general validity of this explanation [6, 7]. Furthermore, efforts to directly optimize the margin (or the minimal margin) have met with mixed results [8, 9]. In contrast to boosting, bagging [10] and random forest [11] do not tend to increase the margin of the ensemble. As a matter of fact there are other types ensembles, such as class-switching ensembles, that are effective predictors and yet have *small margins* by construction [12, 13].

In this thesis we provide further evidence that small margin ensembles can achieve good generalization performance and be robust to noise in the class labels of the training examples. As discussed in [14, 15], class label noise can be more harmful for classification accuracy than noise in the features. Therefore, designing classifiers whose performance does not significantly deteriorate in the presence of the former kind of noise is an important issue in machine learning applications. The excessive weights assigned to incorrectly labelled examples by the standard boosting algorithms, such as Adaboost [16], makes them ill-suited to handling these type of noisy learning tasks. There are robust versions of boosting [4, 8], generally based on the use of regularizations techniques, whose goal is to avoid emphasizing instances that are identified as outliers (mislabelled instances). Bagging is a robust classifier [17–19].

In bagging, the predictors are built using different bootstrap samples from the original training data with replacement. Each bootstrap sample in standard bagging contains the same number of instances as the original training set. The final decision is obtained by majority voting. Bagging generally improves the classification accuracy by reducing the variance of the classifiers. By contrast, boosting works mainly by reducing the classification bias [20, 21]. The deterioration in the predictive accuracy in noisy problems is mainly due to an increment in the variance of the classifiers [18, 19, 22]. Therefore bagging, which is a variance reduction mechanism, is generally more robust than standard boosting in presence of noisy instances [18, 19, 22].

The prescription used in bagging for the size of the bootstrap samples on which the ensemble predictors are trained need not be optimal. In fact, the generalization capacity of the ensembles can significantly improve with subsampling [23–25]. Using small sampling ratios has an important effect on how isolated instances are effectively handled. By isolated instances we refer to those located in a region in which the majority of instances belong to another class. The intuition is that, when the bootstrap samples in bagging contain less than 50% of the instances of the original training data, the classification given by the ensemble on a given instance is dominated by the surrounding instances [23, 25]. This is the case because each instance is present in less than half of the training sets of the classifiers in the ensemble. As a consequence its classification is dominated by classifiers trained on bootstrap samples that do not contain that particular instance. Incorrectly labeled instances are often isolated instances. Hence, using small sampling rates can help reduce the influence of these isolated instances and lead to ensembles that are more robust.

The thesis is organized as follows: Chapter 2 provides an introduction to ensemble learning. In chapter 3, we review previous work on how to deal with class-label noise in classification problems. There are two main approaches to address this issue. In one approach noisy examples are detected; these examples are then either removed or cleansed; (i.e. their class labels are corrected). A second strategy is to design learning algorithms that are robust to this kind of noise. In chapter 4 we present a comprehensive empirical assessment of the accuracy and robustness to label noise of bagging ensembles composed of decision trees, and random forests, as a function of the bootstrap sampling ratio. The results of this study provide evidence that small margin classifiers can be resilient to class label noise. In Chapter 5, we describe a noise detection method based on subsampling in bagging. Finally, the conclusions of the thesis and proposals for future work are presented in chapter 6.

Chapter 2

Classification with Ensembles

Machine learning (ML) is a branch of computer science devoted to the study of computational systems learning. By applying machine learning techniques we are able to identify patterns from data. These patterns can then be used to make predictions on new instances [26, 27]. ML algorithms can be classified into two categories: supervised learning and unsupervised learning. The goal of supervised learning is to induce a model form a set of labelled instances in the form of input-output pairs. From these data we induce a mapping from the attributes that characterize the instances (the inputs) to their corresponding target labels (the outputs). The induced mapping captures underlying regularities in the data. Finally, the mapping can be used to predict the labels of unseen instances. In unsupervised learning, the instances do not have an associated label. The goal is to learn a plausible partitioning of the data into groups or clusters. Semi-supervised learning is another class of problems that shows some characteristics with supervised and with unsupervised learning. In semi-supervised learning, the targets of some instances are available (usually for a small subset of the data), while most of the instances are unlabelled. In these types of problems, the unlabelled instances are used to improve the predictions made by the learners induced from the labelled instances [28, 29].

The focus of this thesis is on supervised learning and, in particular, on classification. The goal of classification is the prediction of discrete target labels. Some well-known classification systems are: neural networks (NN) [30–32], support vector machines (SVM) [33–35], k-nearest neighbours (k-NN) [36–38], decision trees (DT) [39–41], ensemble learning [11, 17, 42, 43], etc. In this thesis we focus on ensemble learning to improve the predictive accuracy of the model. In this family of methods, the predictions of a collection of learners are combined to improve the generalisation capacity of individual classifiers. Averaging the individual predictions or voting are common techniques for determining the final decision in an ensemble of learners [27, 29]. Different types of classifiers such as NN [42, 44–46], SVM [47–50], k-NN [51–53] and DT [11, 54–56] can be used as base learners in an ensemble. DTs are one of the most commonly used base learners in ensembles of classifiers. DTs are simple, fast and flexible predictors [27, 57]. For these reasons DTs are chosen as the base learners of the ensembles analysed in this thesis.

In classification, the margin of an instance is a measure of its distance to the decision boundary. This concept is interesting because it can be used to set an upper bound on the generalization error [58]. One of the best-known margin-based learner models is SVM [58, 59]. The key idea in SVM is to find a linear separator that correctly classifies the training instances while maximizing their distance to the decision boundary. Intuitively, a narrow margin implies that the model is sensitive to small displacements of the instances near the boundary. This lack of stability often reduces the generalization capacity of the model. In the context of ensemble learning, the margin of an instance is defined as the number of (weighted) votes for the correct class minus the number of (weighted) votes for the most voted incorrect class. A higher margin means that the ensemble prediction is more certain, because most of the classifiers agree. The idea of margin maximization was first introduced by Schapire [5] for Adaboost. The progressive emphasis of boosting on misclassified instances increases the margin of the training examples. This property has been proposed as an explanation for the effectiveness of boosting [5]. However, methods that directly optimize the margin yield mixed results [8, 9]. In addition, some well-known ensemble methods, such as bagging [10], random forest (RF) [11] and class-switching [12, 13], do not show a margin maximization behaviour and can also be effective predictors. Another example that small margin can be also accurate and robust is subbagging. In subbagging one uses samples that are smaller than the original training data to build the individual ensemble classifiers. The samples are generated using sampling without replacement from the original training data. The experiments performed in this thesis and in [25, 60] illustrate that bagging and random forest using bootstrap samples of size smaller than the standard prescription are robust to noise in the class labels, in spite of having smaller margins.

In this chapter we will describe how to build ensembles that are effective classifiers. The material is organized as follows: Section (2.1) is devoted to classification and describes how learners are validated. In this section we also discuss the decomposition of the classification error into bias and variance. Then, we describe how to build Classification And Regression Trees (CART). CART trees are the base learners used in the ensembles analysed in this thesis. In section (2.2) we provide an overview of ensemble learning and the justification of why ensembles have the potential to improve the accuracy of individual predictors. Finally, bagging, boosting, random forest and class switching ensembles are described. Section (2.3) introduces and discuses margins in the context of ensembles. An overview of the Structural Risk Minimization (SRM) principle and how it can be used to select hypotheses with good generalization properties is presented in this section. Finally, the conclusions of this chapter are given in section (2.4).

2.1 Classification

Classification is a type of supervised learning in which the goal is to predict the discrete class labels of the input data instances. As in many learning problems, classification can be divided in two phases. In the first phase, a predictor is built using the available training data. These data consist of a collection of labelled instances. In the second phase, the induced hypothesis is used to predict the class labels for new unlabelled instances. To build effective predictors, it is essential to have a sufficient amount of training data, as well as to consider learning models with the adequate expressive capacity [61].

Consider the training set

$$\mathcal{D}_{train} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{train}}, \tag{2.1}$$

where $\mathbf{x}_i \in \mathcal{X}$ is the input vector, $y_i \in \mathcal{Y}$ is the class label, $\mathcal{Y} \in \{c_1, \cdots, c_K\}$. K is the number of class labels. N_{train} is the number of training instances. In classification, the i^{th} instance is characterized by a value of the input vector $\mathbf{x}_i$ and of the class label y_i. In this thesis we will assume that the instances are independently and identically distributed (iid) random variables sampled from a fixed but unknown distribution S.

A hypothesis h is a function that maps the set of input vectors ($\mathcal{X}$) onto the corresponding set of class labels ($\mathcal{Y}$). This mapping function partitions the input region $\mathcal{X}$ into K different regions. Each of these regions corresponds to one class. The separations between these regions are called decision boundaries.

The generalization error is the probability of misclassifying instance $(\mathbf{x}, y)$ drawn from the distribution S

$$Error(h) = Pr_{(\mathbf{x},y)\sim S}[h(\mathbf{x}) \neq y]. \tag{2.2}$$

The calculation of the exact generalization error is seldom possible because the underlying distribution of the p is usually unknown [29]. Therefore, to evaluate a given hypothesis, the empirical error is calculated on a test or validation set $\mathcal{D}_{test} = \{(\mathbf{x}_j, y_j)\}_{j=1}^{N_{test}}$ that is drawn (iid) from the distribution S, independently of the training set. N_{test} is the number of test instances, $x_j \in \mathcal{X}$ is the input vector and $y_j \in \mathcal{Y}$ is the class label.

The empirical error is the average misclassification error of the instances in $\mathcal{D}_{test}$ [29]

$$Error_{test}(h) = \frac{1}{N_{test}} \sum_{j=1}^{N_{test}} \mathbb{1} h(\mathbf{x_j}) \neq y_j, \tag{2.3}$$

where $\mathbb{1}$ is the indicator function. In learning, the hypothesis is built by induction from the training instances. The induced hypothesis is expected to have the ability to generalize well; that is, to make correct predictions also for new unseen instances. In other words, generalization refers to the capacity of learning from past experiences instead of memorizing them [62]. The generalization capacity of a learning algorithm depends on its expressive capacity or complexity (flexibility) and also on the number of the instances available for training. Training a complex model using few instances can lead to overfitting. Overfitting refers to the situation in which the model performs well on the training instances but does not perform well on independent test instances. This situation is more likely when the space of hypothesis considered in the learning algorithm has a high flexibility. This increased expressive capacity means that small changes in the training data can make the learner decisions very different. For a model with fixed expressive capacity, increasing the size of the training set reduces the risk of overfitting. If the expressive capacity of the considered learning algorithm is too small to describe the regularities of the underlying problem, then underfitting can occur. In such a situation, the learner has a high bias, because its capacity of representation is not sufficiently large to capture the underlying structure of the data [27, 57, 63]. If either underfitting or overfitting occur, the induced model does not attain the good generalization performance.

The lowest error rate that can be achieved by a classifier in a particular problem is the Bayes error. The optimal classification for instance $\mathbf{x}$, drawn from the distribution S, is

$$h^*_S(\mathbf{x}) = \underset{y}{\operatorname{argmax}}\, P(y|\mathbf{x}). \tag{2.4}$$

The Bayes error can be accurately estimated only when the complete distribution of the data is known.

There are different ways to decompose the prediction error. In [64–66], the error is broken up into three terms: the bias, the variance and Bayes error. However, estimating Bayes error is not possible in a dataset with limited samples from an unknown distribution. Different definitions of bias and variance are given in the decomposition proposed by Webb in [21]. Consider a learning algorithm $\mathcal{L}$. Let $\mathcal{L}(\mathcal{D}_{train})(\cdot)$ denote the hypothesis induced by the learning algorithm $\mathcal{L}$ from the training data $\mathcal{D}_{train}$. The generalization error can be expressed as the sum of two terms

$$Error = bias + variance, \tag{2.5}$$

$$bias = P_{S,\mathcal{D}_{train}}(\mathcal{L}(\mathcal{D}_{train})(\mathbf{x}) \neq y \wedge \mathcal{L}(\mathcal{D}_{train})(\mathbf{x}) = \overset{\circ}{h}_{\mathcal{L},\mathcal{D}_{train}}(\mathbf{x})), \tag{2.6}$$

$$variance = P_{S,\mathcal{D}_{train}}(\mathcal{L}(\mathcal{D}_{train})(\mathbf{x}) \neq y \wedge \mathcal{L}(\mathcal{D}_{train})(\mathbf{x}) \neq \overset{\circ}{h}_{\mathcal{L},\mathcal{D}_{train}}(\mathbf{x})), \tag{2.7}$$

where $\overset{\circ}{h}_{\mathcal{L},\mathcal{D}_{train}}(\mathbf{x})$ is the most probable class label for instance $\mathbf{x}$. This central tendency is estimated using the training set instances. In this decomposition, bias refers to the deviations of the central tendency from the actual class labels. Variance is defined as the probability of errors that correspond to deviations of the individual predictors from the central tendency.

2.1.1 Decision Trees

A Decision Tree (DT) is a hierarchical learning model, which can be used in regression and classification problems. Besides giving fast predictions, decision trees are rule based predictors which are easy to understand and interpret. These aspects make DT a popular learning tool that has been employed in many applications in the medical [67], project management [68], finance [69] marketing [70] and other domains.

In decision tree learning, a tree shaped structure is used to make predictions. The tree is constructed based on rules that partition the space of attributes. These rules are induced from the training data. The construction of a decision tree involves two phases. In the first phase, the tree is grown by recursively splitting the space of attributes. To do so, a root node is created and all instances are assigned to it. Then, the algorithm searches for the most informative rule to partition the space. The root node is split into child nodes, each of which corresponds to one of the regions in which the space is partitioned. Each instance in the parent node is assigned to one of children nodes base on the values of the attributes and on the selected splitting rule. Impurity functions such as the entropy, the cross-entropy, the information gain or the Gini index are used to select the splitting rule at each step. The child nodes are then split further with the goal of achieving a higher degree of homogeneity in the successive children. The process is continued until a stopping condition is fulfilled. Possible stopping conditions include: (1) all instances assigned to the node considered for splitting have the same class label (2) no split is found to further partition the data (3) the number of instances in each terminal node is smaller than a predefined threshold (i.e. there are not enough instances to make a decision) (4) a further split does not reduce the impurity of the node (5) instances that are assigned to each node reach a degree of homogeneity, etc [71].

In the second phase, the tree is pruned. Pruning refers to eliminating some of the terminal branches of the tree. The goal is to limit the complexity of the tree with the objective of avoiding overfitting and thus improving its generalisation capacity [26, 72]. Complexity is usually determined in terms of one of the following measures: number of nodes, number of leaves, depth of the tree or number of variables used for splitting. In addition to an increased risk of overfitting, the storage requirements and the prediction times increase with the complexity of the tree. In the opposite situation, trees that are too simple have a limited expressive capacity, which could be insufficient to capture the underlying patterns in the data. This could imply a high bias and a poor prediction performance. Therefore, it is important to build trees of the right size (complexity): neither too small (risk of underfitting) nor too large (too complex and prone to overfitting).

Finally, the induced tree is used to make predictions on test instances. To make a prediction, the instance is first assigned to the root of the tree. Then, based on the value of the attribute on which the root node is split, it is assigned to one of the child nodes. This process is iterated at the corresponding inner nodes until the instance reaches a leave. The class label prediction for that instance is the majority label of the training instances assigned to that particular leave.

Some of the better-known DT algorithms include ID3 (Iterative Dichotomiser 3) [73], C4.5 [74] and CART [64]. Since CART is only the algorithm in which the base learners of RF (random trees) are based. CART is the algorithm in which the base learner algorithm of random forest (random tree) is based and it will be employed as the base learner in the ensembles analysed in this thesis. It can be employed to address classification and regression problems. CART trees are grown by recursive binary partitioning [75]. That is, each internal node has two children. These children are called left child and right child. The criterion used in the CART algorithm to select the node split is the Gini index criterion

$$Gini(t) = 1 - \sum_{i=1}^{K} P_i^2(t), \tag{2.8}$$

where $P_i(t)$ is the relative frequency of class i in node t. The Gini index takes values between 0 and $(1 - \frac{1}{K})$, where K is the number of class labels. The Gini index is equal to 0 when all instances in a node belong to the same class. The index takes its maximum value, $(1 - \frac{1}{K})$, when the instances assigned to a node are distributed equally between the K classes.

Let t be a node that is considered for splitting. The change of the Gini index when split s is made in node t is

$$Gini_{split}(s,t) = Gini(t) - P_L \cdot Gini(t_L) - P_R \cdot Gini(t_R), \tag{2.9}$$

where t_L and t_R are the left child and the right child of the node t, P_L and P_R are the fractions of the instances in t that are assigned to the left child and to the right child, respectively, and $s \in SP$ is a particular split in all possible set of splits SP. The split that maximizes $Gini_{split}(s,t)$ is then selected. The splitting process is repeated until one of the specified stopping conditions is fulfilled. In a second phase the fully grown tree is pruned using cost-complexity pruning. This type of pruning uses a global cost function that takes into account both the complexity and the accuracy of the resulting tree,

$$R_\alpha(t) = R(t) + \alpha \cdot C(t), \tag{2.10}$$

where $R(t)$ is the misclassification cost of the decision tree rooted at node t, which is estimated on the training set. The complexity of the tree is computed as the number of terminal nodes that it is denoted by $C(t)$. The parameter α specifies the relative weight between the accuracy and the complexity of the tree. Higher values of α lead to smaller trees. Lower values of α produce larger trees with higher accuracy in training. The value of α used in CART is estimated using cross-validation [64, 76].

2.2 Ensemble Learning

An ensemble is a collection of predictors whose outputs are combined to produce a global decision. The main idea in ensemble learning is to take advantage of the complementarity of the classifiers to improve their predictive accuracy [27, 29]. The aggregation of base learners is beneficial only if the individuals make different predictions. Accuracy and complementarity of the base learners are two fundamental aspects in the performance of an ensemble [17, 42, 43, 77, 78].

Consider for instance an ensemble composed of three base classifiers $\{h_1, h_2, h_3\}$. If the three hypotheses are equal, then an incorrect prediction by $h_1(\mathbf{x})$ co-occurs with incorrect predictions of the other two hypotheses. By contrast, if error of the classifiers are uncorrelated, an incorrect decision by one of the predictors can be compensated by the correct decisions of the other two. In this case majority voting would give the correct prediction. Ideally, the errors of the classifiers should be independent, so that the aggregation of the individual predictions is advantageous.

Assume that the predictions by the base learners for a particular instance $\mathbf{x}$ are independent. If the combination of the classifier outputs is made by majority voting, the probability of error is the probability that that more than half of the classifiers misclassify instance $\mathbf{x}$ is

$$Error_{test}(p_{\mathbf{x}}, T) = \sum_{\lfloor t=T/2 \rfloor+1}^{T} \binom{T}{t} p_{\mathbf{x}}^{t}(1-p_{\mathbf{x}})^{T-t}, \tag{2.11}$$

where $p_{\mathbf{x}}$ is the average error of an individual learner of the ensemble. If the expected accuracy of the individual classifiers is better than random guessing (i.e. $p_{\mathbf{x}} < \frac{1}{2}$), we can apply the Condorcet Jury Theorem and conclude that the decision based on the aggregation of the base learners is more accurate than the individual predictions [79]. Expression (2.11) is the area under the binomial distribution where more than half of the classifiers yield an incorrect prediction. As a numerical example, consider an ensemble of 21 independent classifiers. Assume that the error rate for each classifier on a given instance is 0.3. Using (2.11), the ensemble error is 0.026, which is much lower than the individuals error rate (0.3) [17]. Combining more such classifiers eventually reduces the error to zero.

In [17], Dietterich gives three reasons that explain why group learning can be an effective method to enhance the performance of the individuals (see figure 2.1). The first reason is statistical: assume that we are searching for an accurate hypothesis h in the hypothesis space $\mathcal{H}$. If the size of the training data is too small in relation with the size of $\mathcal{H}$, there might be several predictors with comparable performance. In this situation, averaging the predictions of these hypotheses can reduce the risk of selecting an incorrect one. The second reason is computational: the learning problem is usually transformed into an optimization problem, which needs to be solved numerically. If the objective function has several local minima it is possible that the algorithm converges to a suboptimal local minimum. Aggregation of several predictors, each of which is the result of an optimization with a different starting point, reduces the risk of selecting a local minimum far from the global one. Finally, it may be difficult to approximate the actual predictor function in the selected hypothesis space. By averaging several hypotheses, the representation capacity can be expanded.

According to the types of the classifiers whose decisions are combined, ensembles can be either homogeneous or heterogeneous. In homogeneous ensembles, all base learners are of the same kind. Heterogeneous ensembles are composed of base learners of different types.

There are several techniques to build diverse base learners in a homogeneous ensemble. Some ensemble learning algorithms use modified versions of the training set to train the base learners; others introduce changes in the learning algorithms. These strategies can also be used in combination.

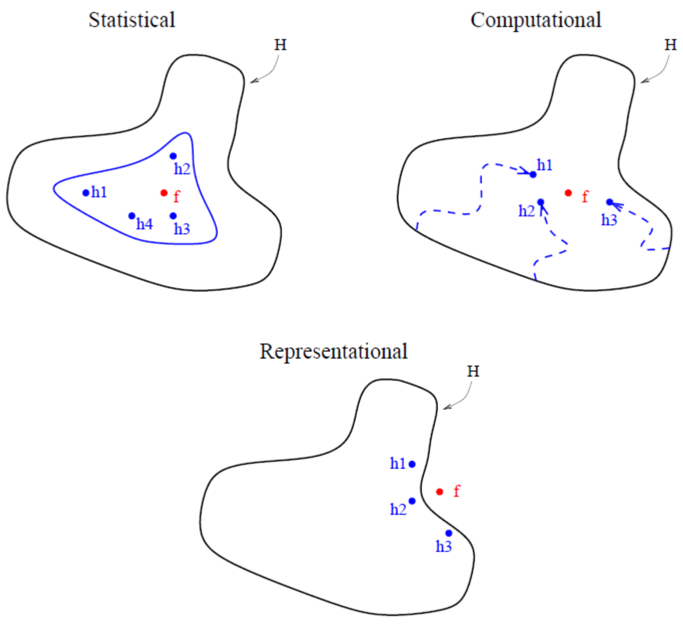

FIGURE 2.1: Illustration of the three reasons given by Dietterich [17] showing how ensembles might enhance the results of single base learners

Applying the same learning algorithm to different versions of the training set, can be an effective strategy to build diverse classifiers. Different versions of the training set can be generated by using surrogates of the original training set and/or by manipulating the training instances. The surrogates of the original training set can be generated using:

1. Resampling with replacement: In this technique, repeated instances can occur in a given bootstrap sample. Bagging [10] is an ensemble method that uses this technique.

2. Resampling without replacement: In this technique, all the instances in the bootstrap samples are different. Therefore, to have different versions of the training set, the number of the instances in each sample needs to be lower than the number of instances in the original training set. Subbagging is an ensemble method that uses this technique [80]. In subbagging the base learners are trained on subsamples of size $\frac{N_{train}}{2}$ drawn without replacement from the original training set.

3. Weighted resampling: The idea in this technique is to draw samples from the original training set taking into account the weights of the instances in the training set. Instances with higher weights have a higher probability to be selected in each sample. Boosting [55] and wagging [20] are two ensembles that are built using weighted sampling. In boosting, instances that are misclassified by the most recent ensemble classifier are assigned a higher weight to build the following classifier. Wagging assigns random weights to the instances using the Poisson distribution.

4. Using artificial data: In this technique, training samples include artificial instances. An example of this strategy is DECORATE (Diverse Ensemble Creation by Oppositional Relabelling of Artificial Training Examples). DECORATE is an ensemble method in which diversity is generated by adding artificial instances to the samples used to build the individual ensemble classifiers. The distribution of the training data is modelled using a Gaussian model. Artificial instances are then sampled from this distribution. Finally, the class label of a particular artificial instance is specified to be different from the current ensemble prediction for this instance [81].

Manipulation of the instances is another strategy for generating diverse predictors. It is possible to modify the attributes of the instances, their class labels or the weights that determine their importance in the learning process.

1. Altering the attributes: The base learners are trained using different feature subsets. This method is specially adequate in problems with redundant features. Attribute bagging [82], in which a random subset of features is used to train each base learner, is an ensemble method based on this technique. Another example of this strategy is using random feature subspaces in decision trees [83].
2. Altering the class labels: For example, in class-switching [84] to generate diverse base learners, the class labels of a portion of the data in each sample is switched to another randomly selected label. Error-correcting output coding (ECOC) [85] is another ensemble method based on modifying the class labels. In this method, to train each base learner, the labels are randomly categorized into two groups. For a particular test instance $\mathbf{x}$, the prediction is carried out based on the new labels. The vote count of all the original labels that are in the group of the predicted label is increased. The final prediction of the ensemble for the given instance $\mathbf{x}$ is the original class label with the highest number of votes.
3. Modifying the weights of the instances: Boosting [86] is one the ensemble methods in which training instances with their correspond weights attend in learning process. In this method the weights of the instances are initially equal. Afterwards, in an iterative process, the weights of the instances that are misclassified by the previous base learners are increased.

As mentioned earlier, a second method for creating diversity in homogeneous learners is to introduce modifications in the learning algorithm. This strategy can be implemented using:

1. Different parameter choices: Diversity in k-NN can be achieved by using different k values [53].

2. Different architectures in a learning model: In models such as neural network, diversity can be achieved by varying the architecture; for instance using different numbers of hidden layers or of nodes in the hidden layer [87].

3. Initialisation of the learning algorithm with different starting points: For instance using different initial weights to train a neural network [88].

4. Embedding global information in each hypothesis optimization space: An example of this strategy is Negative Correlation Learning (NCL) [89, 90]. In this method each neural network in the ensemble is trained by minimizing a cost function that is the sum of the mean squared error (MSE) and a term that penalizes the correlation of the individual and the ensemble predictions.

Combinations of these techniques can also be effective in ensemble construction [91–93]. For example, random forests are built by sampling with replacement from the original training set and also by selecting splits in a random subsets of the features [11].

Heterogeneous ensembles are composed of predictors that are generated with different learning algorithms. There is some evidence that heterogeneous ensembles can outperform homogeneous ensembles [94, 95], specially when their size is small [56]. A possible explanation is that classifiers of the same type tend to be affected by the same types of biases. Therefore, combining predictors whose biases are different increases the chance of having complementary predictors [95, 96]. Nevertheless, the improvements depend on the type of the base learners that are included in the ensemble and on the method used to combine their predictions. In any case, the empirical evidence is inconclusive: There are cases in which homogeneous ensembles outperform heterogeneous ensembles [97]. In any case, heterogeneous classifiers are more difficult to build and analyze than homogeneous ones: One needs to choose the adequate combination of base algorithms for each application. Then the parameters of the different base learning algorithms have to be tuned to optimize the accuracy not only of the individual predictors but also of the ensemble [98]. In this thesis, all the analyzed ensembles are homogeneous. Nonetheless, the strategies that are proposed can in principle be used to improve the effectiveness of heterogeneous ensembles as well.

After having generated the base learners, the next phase is to combine the individual decisions into a global decision. Different combination techniques can produce significant differences in the accuracy of the ensembles. The combination techniques can be divided into two groups:

1. Voting strategies: In majority voting each of the base learners in the ensemble predicts a class label for the instance under consideration. The ensemble prediction is the class label that is predicted most often by the base learners. Bagging [10] is a method in which majority voting is used to compute the final decision. Another variant of these types of methods is weighted majority voting. In this variant weights are used to determine the influence of the individual base learners in the final decision. Adaboost [99] is one of the methods that uses weighted majority voting. In Adaboost, the weight of each base learner in the global ensemble prediction depends on its accuracy on the training set.

2. Non voting strategies: These methods are used to combine predictors whose outputs quantify the confidence on the predictions. For instance, they could provide an estimate of class posterior probabilities. Some operations such as maximum, minimum, product, median and mean can be employed on the (weighted) confidence levels that are the output of the individual base learners. The final decision is determined by the most probable class label, based on the result of the employed operation [100–102].

Simple methods, such as (weighted) majority voting are effective in a wide range of applications [103]. Using the product in non voting strategies may be preferable if the base learners make independent errors [103, 104]. Nevertheless there is no winner strategy among the different combination techniques [101]. The optimal method is different base on the type of the base learners and the classification task under consideration [101, 104, 105].

The combination strategies that are described up to this point are static because the rule used to combine the outputs of the individual predictors is fixed. It is possible to design dynamic strategies, in which the combination depends on the particular instance whose class label is being predicted. Stacking is an example of a dynamic combination technique. In stacking the combination phase is accompanied by a learning process. First the base learners are trained on some version of the original training set (e.g. a bootstrap sample). After that, the predictions of the base learners are used as new feature vectors to train a second level learner (meta-learner). The key point in this strategy is to improve the guesses that are made by the base learners, by generalizing these guesses using a meta learner [106].

In the following subsections we give a description of some representative ensemble methods: bagging, boosting, random forest and class switching ensembles.

2.2.1 Bagging

Bagging [10] is an ensemble method in which individuals are trained on different bootstrap samples drawn from the original training data. Each bootstrap sample is created by selecting instances from the original training set with replacement. The standard procedure in bagging is to use bootstrap samples that contain the same number of instances as the original training set. This prescription produces samples containing on average 63.2% of different instances from the original training data. The rest are repeated examples. Each base learner is trained on an individual bootstrap sample. The final decision of the ensemble is given by simple majority voting. The pseudo-code of bagging is shown in algorithm (1). Averaging over diverse classifiers removes the uncorrelated errors of the individuals. In this manner, the variance error tends to decrease as a result of the aggregation of individual decisions. As mentioned earlier, bagging applies the strategy of resampling with replacement to produce diversity. Nevertheless, if the base learners are stable the variability introduced by the resampling process may not be sufficient to generate diverse classifiers [10].

Algorithm 1: Bagging algorithm[20]

```
Input: $\mathcal{D}_{train}$={$(\mathbf{x}_i, y_i)$}$_{i=1}^{N_{train}}$ % Training set
$T$ % Ensemble size
$\mathcal{L}$ % Base learning algorithm
1 for $t \leftarrow 1$ to $T$ do
2     $D_t \leftarrow Bootstrap(\mathcal{D}_{train})$% Bootstrap sample from $\mathcal{D}_{train}$, uniformly with
      replacement
3     $h_t(\cdot) \leftarrow \mathcal{L}(D_t)$ %Build a base learner applying $\mathcal{L}$ on bootstrap sample
  Output: $H(\cdot) = \arg\max_y \sum_{t:h_t(\cdot)=y} 1$
4 % Majority vote
```

Subbagging [80] is a variation of bagging in which sampling without replacement is used to generate bootstrap samples. Therefore, there are no repeated instances in the subsampled set. A common choice for the size of these subsampled sets is $\frac{N_{train}}{2}$, where N_{train} is the original training set size [107]. This prescription has been shown to be statistically equivalent to standard bagging [25, 107, 108]. As in bagging, the final decision is determined by a simple majority voting. The size of the bootstrap samples can be tuned in both bagging and subbagging to improve the accuracy of the ensemble prediction [25]. In this thesis, subsampling is used to build a variation of bagging which is robust in the presence of noise [60]. In addition to the improvements in the generalisation capacity of the ensemble, subsampling decreases the time complexity of the training also reduces the storage requirements and improves the classification speed [25].

The analysis of bagging in regression problems is straightforward [10] and illustrates why bagging can improve over the single learners. Consider the aggregate prediction on an instance characterized by the vector of attributes $\mathbf{x}$

$$H_A(\mathbf{x}) = \mathbb{E}_{\mathcal{D}_{train}}[h(\mathbf{x})], \tag{2.12}$$

where D_{train} indicates an average over all possible realizations of the training set. Assuming that y is the actual continuous target value for the input vector $\mathbf{x}$, the mean square prediction error of hypothesis $h(\mathbf{x})$ is

$$\mathbb{E}_{\mathcal{D}_{train}}[(h(\mathbf{x}) - y)^2] = y^2 - 2y\mathbb{E}_{\mathcal{D}_{train}}[h(\mathbf{x})] + \mathbb{E}_{\mathcal{D}_{train}}[h^2(\mathbf{x})]. \tag{2.13}$$

Using the fact that $\mathbb{E}_{\mathcal{D}_{train}}[h^2(\mathbf{x})] \geq (\mathbb{E}_{\mathcal{D}_{train}}[h(\mathbf{x})])^2$, we can write

$$\mathbb{E}_{\mathcal{D}_{train}}[(h(\mathbf{x}) - y)^2] \geq y^2 - 2y\mathbb{E}_{\mathcal{D}_{train}}[h(\mathbf{x})] + (\mathbb{E}_{\mathcal{D}_{train}}[h(\mathbf{x})])^2. \tag{2.14}$$

Considering the definition of the aggregated predictor in (2.12),

$$\mathbb{E}_{\mathcal{D}_{train}}[(h(\mathbf{x}) - y)^2] \geq y^2 - 2yH_A(\mathbf{x}) + [H_A(\mathbf{x})]^2 = (H_A(\mathbf{x}) - y)^2, \tag{2.15}$$

which indicates that the mean squared error for aggregated prediction is smaller or equal than the expected squared error of the individual predictions. The improvement in performance is given by difference between $\mathbb{E}_{\mathcal{D}_{train}}[h(\mathbf{x})^2]$ and $(\mathbb{E}_{\mathcal{D}_{train}}[h(\mathbf{x})])^2$. If most of the generated hypotheses $h(x)$ produce the same outputs, the two expectations converge to the same value and the aggregation will not improve over the accuracy of the base learners [10]. In bagging one averages not over different training sets, but over bootstrap samples from a single training set

$$H_B(\mathbf{x}) = \mathbb{E}_{\mathcal{D}_t(\mathcal{D}_{train})}[h(\mathbf{x})] \simeq \frac{1}{T}\sum_{t=1}^{T} h_t(\mathbf{x}) \tag{2.16}$$

where $\mathcal{D}_t(\mathcal{D}_{train})$ is the bootstrap sample drawn from the original training set. This bootstrap estimate is generally not as accurate as $H_A(\mathbf{x})$. Therefore, there are two opposing effects: On the one hand, the expected accuracy of the predictors built from bootstrap samples is generally lower than expected accuracy of the predictor trained on the original training data itself. On the other hand, the aggregation process tends to reduce the classification error by variance reduction. Bagging reduces the error only if the variance reduction dominates the error increase associated to using bootstrap samples. A similar analysis can be made in the case of classification problems [10].

2.2.2 Boosting

Boosting refers to an ensemble learning method in which a sequence of learners is built so that each new learner focuses on examples that are misclassified by the previous learner in the sequence [86]. Adaboost is one of the best-known boosting algorithm. It was introduced by Freund and Schapire in [99]. The pseudo-code of Adaboost is shown in algorithm (2). In this method, each instance in the training set is assigned a weight. To train the first classifier, all instances are assigned equal weights. In the subsequent classifiers, the instances weights are determined based on the correct or incorrect classification of these instances by the previous learners. To build the next classifier, the weights of the misclassified instances are increased and the weights of the correctly classified instances are decreased. This process is continued until either the ensemble reaches a specified size or the error of a base learner is higher than 0.5. The final prediction of the ensemble is determined by weighted majority voting. The weight of each particular base learner in the final decision is based on its individual accuracy on the training set [55]. A shortcoming of Adaboost is that it can assign excessive weights to instances that are difficult to classify. This can mislead Adaboost in problems with noisy class labels.

Algorithm 2: Adaboost algorithm [55]

```
Input: $\mathcal{D}_{train}=\{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{train}}$ % Training set
$\mathbf{w}_t = \{w_t(\mathbf{x}_i)\}_{i=1}^{N_{train}}$ % Vector of weights
$T$ % Ensemble size
$\mathcal{L}$ % Base learning algorithm
$w_1(\mathbf{x}_i) \leftarrow \frac{1}{N_{train}}$ for $i = 1, ..., N_{train}$ % Initialize the vector of weights
1  for $t \leftarrow 1$ to $T$ do
2      $D_t \leftarrow Bootstrap(\mathcal{D}_{train},\mathbf{w}_t)$% Bootstrap sample from $\mathcal{D}_{train}$ with weights $\mathbf{w}_t$
3      $h_t(\cdot) \leftarrow \mathcal{L}(D_t)$ % Train the base learner from $D_t$
4      $\epsilon_t = Pr_{\mathbf{w}_t}[h_t(\mathbf{x}_i) \neq y_i]$
5      if $\epsilon_t > \frac{1}{2}$ then
6          $T = t - 1$ and exit loop
7      $\alpha_t = 1/2 \ln(\frac{1-\epsilon_t}{\epsilon_t})$ % The weight of base learner $t$
8      for $i \leftarrow 1$ to $N_{train}$ do
9          $w_{t+1}(i) = \frac{w_t(i)exp(-\alpha_t y_i h_t(\mathbf{x}_i))}{Z_t}$
10         % $Z_t = \sum_{j=1}^{N_{train}} w_t(j)e^{-\alpha_t y_j h_t(\mathbf{x}_j)}$ is a normalization factor
Output: $H(\cdot) = sign(\sum_{t=1}^{T} \alpha_t h_t(\cdot))$
```

Boosting can be viewed as an optimization process in the hypothesis space of linear combinations of predictors of the same type [109]. Consider a binary classification problem. The training data are $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{train}}$, where $\mathbf{x}_i \in \mathcal{X}$ are the attributes and $y_i \in \{-1, 1\}$ are the class labels. We will build an ensemble whose output is a function of a linear combination of the base learner predictions. Let $\mathcal{F}$ denote the space of base learners, where $f_\tau \in \mathcal{F}$ and $f_\tau : \mathcal{X} \to \{-1, 1\}$. The ensemble prediction is of the form

$$H_T = sign[F_T(\mathbf{x})], \tag{2.17}$$

$$F_T(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t f_t(\mathbf{x}), \quad F_T : \mathcal{X} \to \mathbb{R}. \tag{2.18}$$

That is, $F_T \in lin(\mathcal{F})$, the set of linear combinations of function in $\mathcal{F}$. Assume that $lin(\mathcal{F})$ is endowed with an inner product

$$\langle F, G \rangle = \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} F(\mathbf{x}_i) G(\mathbf{x}_i), \tag{2.19}$$

where $F, G \in lin(\mathcal{F})$. Consider now a cost functional $C[F] : lin(\mathcal{F}) \to \mathbb{R}^+$ of the form

$$C[F] = \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} c(y_i F(\mathbf{x}_i)), \tag{2.20}$$

where c is the misclassification cost function. For F_T fixed, our goal is to find the values of $\alpha_{T+1} \in \mathbb{R}$ and $f_{T+1} \in \mathcal{F}$ that minimize

$$C[F_{T+1}], \qquad \text{where} \quad F_{T+1}(\mathbf{x}) = F_T(\mathbf{x}) + \alpha_{T+1} f_{T+1}(\mathbf{x}). \tag{2.21}$$

For a general definition of the inner product

$$\langle F, G \rangle = \int_{\mathcal{X}} F(\mathbf{x}) G(\mathbf{x}) dP(\mathbf{x}), \tag{2.22}$$

the expansion of the cost functional is

$$C[F + \epsilon \delta F] \approx C[F] + \epsilon \langle \nabla C[F], \delta F \rangle + \mathcal{O}(\epsilon^2), \tag{2.23}$$

where

$$\nabla C[F](\mathbf{x}) = \left. \frac{\partial C[F + \alpha \mathcal{I}_{\mathbf{x}}]}{\partial \alpha} \right|_{\alpha = 0}, \tag{2.24}$$

in terms of $\mathcal{I}_{\mathbf{x}}$, the indicator function of $\mathbf{x}$. For a cost function of the form (2.20)

$$\begin{aligned} \delta C[F] &= C[F + \epsilon \delta F] - C[F] = \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} c(y_i (F(\mathbf{x}_i) + \epsilon \delta F(\mathbf{x}_i))) - C[F] \\ &\approx \epsilon \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} c'(y_i F(\mathbf{x}_i)) y_i \delta F(\mathbf{x}_i) + \mathcal{O}(\epsilon^2), \end{aligned} \tag{2.25}$$

assuming that α_{T+1} is fixed, the condition on f_{T+1} in (2.21) so that $C[F_{T+1}] \leq C[F_T]$ is

$$-\langle \nabla C[F_T], f_{T+1} \rangle \leq 0. \tag{2.26}$$

Therefore, f_{T+1} is the function that minimizes

$$\begin{aligned}
\frac{N_{train}}{\alpha_{T+1}} \delta C[F_T] &= \sum_{i=1}^{N_{train}} c'(y_i F_T(\mathbf{x}_i)) y_i f(\mathbf{x}_i) = \sum_{i:y_i = f(\mathbf{x}_i)} c'(y_i F_T(\mathbf{x}_i)) - \sum_{i:y_i \neq f(\mathbf{x}_i)} c'(y_i F_T(\mathbf{x}_i)) \\
&= \sum_{i=1}^{N_{train}} c'(y_i F_T(\mathbf{x}_i)) - 2 \sum_{i:y_i \neq f(\mathbf{x}_i)} c'(y_i F_T(\mathbf{x}_i)) = \\
&= \sum_{i=1}^{N_{train}} c'(y_i F_T(\mathbf{x}_i)) \left(1 - 2 \sum_{i:y_i \neq f(\mathbf{x}_i)} \frac{c'(y_i F_T(\mathbf{x}_i))}{\sum_{j=1}^{N_{train}} c'(y_j F_T(\mathbf{x}_j))} \right) \\
&= -2 \sum_{i=1}^{N_{train}} c'(y_i F_T(\mathbf{x}_i)) \left(\sum_{i:y_i \neq f(\mathbf{x}_i)} w_i^{[T+1]} - \frac{1}{2} \right),
\end{aligned} \tag{2.27}$$

where

$$w_i^{[T+1]} = \frac{c'(y_i F_T(\mathbf{x}_i))}{\sum_{j=1}^{N_{train}} c'(y_j F_T(\mathbf{x}_j))}. \tag{2.28}$$

Assuming that the $c(m)$ is a monotonically decreasing function of m, then $-c'(m)$ is non-negative, and $\{w_i^{[T+1]}\}_{i=1}^{N_{train}}$ can be regarded as the weights of the instances. Therefore, minimizing (2.21) for F_T and α_{T+1} fixed is equivalent to minimizing the weighted error

$$\epsilon_{T+1} = \sum_{i:y_i \neq f(\mathbf{x}_i)} w_i^{[T+1]}; \qquad w_i^{[T+1]} = \frac{c'(y_i F_T(\mathbf{x}_i))}{\sum_{j=1}^{N_{train}} c'(y_j F_T(\mathbf{x}_j))}. \tag{2.29}$$

The stopping criteria is

$$\delta C[F_T] \geq 0 \quad \Longrightarrow \quad \sum_{i:y_i \neq f_{T+1}(\mathbf{x}_i)} w_i^{[T+1]} \geq \frac{1}{2}. \tag{2.30}$$

For the particular case of Adaboost

$$c(m) = e^{-m}, \qquad c'(m) = -e^{-m}. \tag{2.31}$$

Therefore, the weights of the instances are

$$w_i^{[T+1]} = \frac{e^{-y_i F_T(\mathbf{x}_i)}}{\sum_{j=1}^{N_{train}} e^{-y_j F_T(\mathbf{x}_j)}} = \frac{e^{-y_i \sum_{t=1}^{T} \alpha_t f_t(\mathbf{x}_i)}}{\sum_{j=1}^{N_{train}} e^{-y_j \sum_{t=1}^{T} \alpha_t f_t(\mathbf{x}_j)}} \quad i = 1, \cdots, N_{train} \tag{2.32}$$

Assume that, given F_T, we have chosen f_{T+1}. The value of α_{T+1} is determined by minimizing

$$C[F_T + \alpha f_{T+1}] = \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} c(y_i(F_T(\mathbf{x}_i) + \alpha f_{T+1}(\mathbf{x}_i)), \tag{2.33}$$

with respect to α. Taking the derivative with respect to α and setting its value to 0 when $\alpha = \alpha_{T+1}$ we obtain

$$\frac{1}{N_{train}} \sum_{i=1}^{N_{train}} c'(y_i(F_T(\mathbf{x}_i) + \alpha_{T+1} f_{T+1}(\mathbf{x}_i)))y_i f_{T+1}(\mathbf{x}_i) = 0. \tag{2.34}$$

The solution of this equations is

$$\sum_{i:f_{T+1}(x_i)=y_i} c'(y_i F_T(x_i) + \alpha_{T+1}) = \sum_{i:f_{T+1}(x_i)\neq y_i} c'(y_i F_T(x_i) + \alpha_{T+1}). \tag{2.35}$$

For Adaboost $c'(m) = -e^{(-m)}$

$$e^{2\alpha_{T+1}} = \frac{\sum_{i:f(\mathbf{x}_i)=y_i} e^{-y_i F_T(\mathbf{x}_i)}}{\sum_{i:f(\mathbf{x}_i)\neq y_i} e^{-y_i F_T(\mathbf{x}_i)}} = \frac{\sum_{i:f(\mathbf{x}_i)=y_i} w_i^{[T+1]}}{\sum_{i:f(\mathbf{x}_i)\neq y_i} w_i^{[T+1]}} = \frac{1-\epsilon_{T+1}}{\epsilon_{T+1}}, \tag{2.36}$$

where $\epsilon_{T+1} = \sum_{f(\mathbf{x}_i)\neq y_i} w_i^{[T+1]}$. The equation (2.36) implies

$$\alpha_{T+1} = \frac{1}{2} log \frac{1-\epsilon_{T+1}}{\epsilon_{T+1}}. \tag{2.37}$$

Using this result the expression for the weights can be simplified

$$w_i^{[T+1]} = \frac{e^{-y_i F_T(\mathbf{x}_i)}}{\sum_{j=1}^{N_{train}} e^{-y_j F_T(\mathbf{x}_j)}} = \frac{e^{-y_i \sum_{t=1}^{T} \alpha_t f_t(\mathbf{x}_i)}}{\sum_{j=1}^{N_{train}} e^{-y_j \sum_{t=1}^{T} \alpha_t f_t(\mathbf{x}_j)}} = \frac{1}{Z^{[T+1]}} w_i^{[T]} e^{-y_i \alpha_{T+1} f_{T+1}(\mathbf{x}_i)}, \tag{2.38}$$

where

$$Z^{[T+1]} = \sum_{i=1}^{N_{train}} w_i^{[T]} e^{-y_i \alpha_{T+1} f_{T+1}(\mathbf{x}_i)} = \sum_{i:f(\mathbf{x}_i)=y_i} w_i^{[T]} e^{-\alpha_{T+1}} + \sum_{i:f(\mathbf{x}_i)\neq y_i} w_i^{[T]} e^{\alpha_{T+1}}$$
$$= (1-\epsilon_{T+1})\sqrt{\frac{\epsilon_{T+1}}{1-\epsilon_{T+1}}} + \epsilon_{T+1}\sqrt{\frac{1-\epsilon_{T+1}}{\epsilon_{T+1}}} = 2\sqrt{\epsilon_{T+1}(1-\epsilon_{T+1})}. \quad (2.39)$$

2.2.3 Random Forest

Random Forest (RF) is a very effective ensemble method in which the individual learners are randomized trees [11]. As in bagging, each tree is generated using a different bootstrap sample drawn with replacement from the original training set. RF differs from bagging in the way that decision trees in the ensemble are built. In RF, in order to identify each split in the tree, a random subset of the features is sampled from the set of features. The best split within this subset is then selected. This process is repeated until all instances are correctly classified. The randomized trees are fully grown and no pruning is applied. As in bagging, the final decision in RF is obtained by majority voting. The pseudo-code of RF is shown in algorithm (3).

Since RFs have an additional randomization mechanism, the diversity in this types of ensembles is larger than in bagging. Numerous empirical studies show that this leads to improvements in the prediction accuracy in many problems of practical interest [11, 110–112].

Algorithm 3: Random Forest algorithm[11]

```
Input: $\mathcal{D}_{train}=\{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{train}}$ % Training set
$T$ % Ensemble size
$l$ % number of selected features in each split
$L$ %Data dimension
$\mathcal{L}$ % Base learning algorithm
1 for $t \leftarrow 1$ to $T$ do
2     $D_t \leftarrow Bootstrap(\mathcal{D}_{train})$% Bootstrap sample from $\mathcal{D}_{train}$ uniformly, with
      replacement
3     %Build tree on $D_t$
4     while nodes are not pure do
5         for each node in tree do
6             Select $l < L$ random features.
7             choose among the randomly selected feature, the one that provides the best
              split based on a specified objective function.
8             Split the current node on the selected feature.
Output: $H(\cdot) = \arg\max_y \sum_{t:h_t(\cdot)=y} 1$ %Majority voting
```

2.2.4 Class Switching Ensemble

Class switching is an ensemble method in which diversity is obtained by using different versions of the training data polluted with class label noise. Specifically, to train each base learner, the class label of each training point is changed to a different class label with probability p.

This idea was first proposed by Breiman [12]. In this work the class labels of the training instances are changed at random according to a probability matrix $\mathbf{P}$, whose elements are

$$\begin{aligned} P_{j\leftarrow i} &= w\ P_j \quad \text{for } i \neq j \\ P_{i\leftarrow i} &= 1 - w(1 - P_i) \end{aligned} \tag{2.40}$$

where $P_{j\leftarrow i}$ is the probability of changing a class label i to j, P_j is the fraction of instances with class label j in the training set and w is defined in terms of the switching rate p as

$$w = \frac{p}{1 - \sum\limits_j P_j^2} = \frac{p}{2\sum\limits_j \sum\limits_{k>j} P_j P_k}. \tag{2.41}$$

The probability matrix is such that the class distribution of the training set is maintained. The switching rate p in this method is chosen to be sufficiently small so that the training error tends to zero in large ensembles. Actually, p needs to be smaller than the proportion of the instances in the minority class. Otherwise, more than half of the instances in the minority class would have their class flipped. In that case the minority class instances would be misclassified by the ensemble. To address this shortcoming, Martínez-Muñoz and Suárez in [113] proposed to apply a fixed switching rate p, independently of the class distribution. The class label of each instance in the training set is randomly switched to an another label according to the transition probability matrix

$$\begin{aligned} P_{j\leftarrow i} &= \frac{p}{K-1} \quad \text{for } i \neq j, \\ P_{i\leftarrow i} &= 1 - p, \end{aligned} \tag{2.42}$$

where K is the total number of the classes. Using this prescription, the class distribution of the original training set is not maintained for unbalanced datasets. By increasing p, the proportion of the different class labels in the modified dataset becomes approximately equal. In this method, the probability of changing a class label needs to be lower than the probability of keeping the original class label ($P_{j\leftarrow i} < P_{i\leftarrow i}$) to guarantee that the ensemble correctly classifies the instances in the training set. Based on this consideration and (2.42) the upper bound of the switching rate is

$$p < p_{max} = \frac{K-1}{K}. \tag{2.43}$$

The final decision in this method is computed using majority vote among the base learners of the ensemble. The number of classifiers that need to be aggregated for convergence of the ensemble prediction is fairly large (> 1000) in most of the analysed classification tasks, specially for high values of p. This method is equivalent to Breiman's class flipping in class-balanced datasets, and exhibits better performance in unbalanced problems.

2.3 Classification Margins

In the context of classification the margin of an instance is defined as the distance to the decision boundary. The concept of margin has been a useful tool in the design and analysis of the effectiveness of numerous classification methods [114–119]. The observation that upper bounds on the generalization error decrease with increasing margins suggests that increasing the margin could improve the generalization capacity of the classifier and the confidence of its predictions. Support Vector Machine (SVM) [58, 59] is the best-known classification method based on margins. This method combines two ideas: first, instances are mapped to a high dimensional feature space in which the classification problem is linearly separable. Second, one searches for a hyperplane that correctly separates the training instances and maximizes the minimal distance between the hyperplane and the closest instances. Instances from different classes that are close to the separator hyperplane have higher uncertainty, due to their proximity to the region assigned to the other class.

The concept of the margin in an ensemble was introduced by the Schapire *et al.* in [5] in relation to boosting. In the context of ensembles, the margin is defined as the difference between the weighted sum of the votes for the correct class and the weighted sum of the votes for the most voted class other than the correct one

$$margin(\mathbf{x}, y) = \frac{1}{T}\sum_{t=1}^{T} \alpha_t \mathbb{1}(h_t(\mathbf{x}) = y) - max_{j \neq y}\ \frac{1}{T}\sum_{t=1}^{T} \alpha_t \mathbb{1}(h_t(\mathbf{x}) = j), \tag{2.44}$$

where α_t is the weight of base learner t in the ensemble and $\mathbb{1}$ is the indicator function. Assuming that the weights are normalized ($\sum_t \alpha_t = 1$), the margin is a real number in $[-1, 1]$. The margin is positive for correctly classified instances and negative for misclassified ones. A larger positive margin indicates a higher confidence on a correct classification. Large margins are often identified as a key point in the generalization capacity of a classifier [1–4]. Nonetheless there are some empirical studies that put in doubt the general validity of this view [6, 7]. Furthermore, efforts to directly optimize the margin (or the minimal margin) have met with mixed results [8, 9]. In the area of ensemble learning, boosting effectively increases the margins of the training example by progressively focusing on hard-to-classify instances. It defines a classification boundary that tends to minimize the overlap between classes. On the other hand, bagging and random forest do not tend to increase the margin [5]. Another example of classifiers that exhibit low margins is class-switching [12, 13]. This ensemble learning method, which has a generalization performance comparable to boosting [13], builds classifiers that exhibit small margins by construction. Furthermore, we will show in this thesis that subsampling improves the robustness of bagging in the presence of noise, while decreasing the margin [60].

2.3.1 VC dimension and Structural Risk Minimization

As discussed earlier in this thesis, the goal of machine learning is to use training data to automatically induce a system that make correct predictions on new instances. The learning algorithm should be able to generate predictions with an adequate capacity to properly capture the underlying regularities in the data. on the one hand, if the expressive capacity of the hypothesis space is insufficient, underfitting can occur. On the other hand, an overly complex hypothesis is prone to overfitting. Therefore, the complexity of the learning model should be tuned to the complexity of the classification problem.

The complexity of a family of hypothesis $\mathcal{H}$ can be quantified in terms of the Vapnik Chervonenkis dimension (VC). The VC dimension is defined as the largest number of instances that can be shattered by a hypothesis in the family considered. Shattering refers to separating instances for any possible class labeling. If for any sample set size N_{train}, every set with N_{train} instances can be shattered by $\mathcal{H}$, then the VC dimension of $\mathcal{H}$ is infinite [120]. Vapnik shows [120] that the generalization error of a classifier can be bounded with probability $1 - \mu$ as

$$Error(h) \leq Error_{test}(h) + \sqrt{\frac{VC(log(\frac{2N_test}{h}) + 1) - log(\frac{\mu}{4})}{N_{test}}}, \qquad (2.45)$$

where $Error_{test}(h)$ is the empirical error and $Error(h)$ is the generalization error, N_{test} is the size of the test set and VC is the value of the VC dimension of $\mathcal{H}$. The second term in (2.45) is called the VC confidence [121].

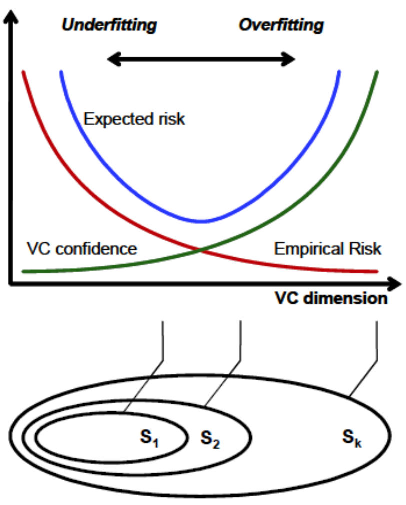

FIGURE 2.2: SRM structure in selection of the optimal learning model [122]

Structural Risk Minimization (SRM) refers to a technique for selecting an optimal hypothesis in which the complexity of the learning model and its generalization ability is adapted to the data available for training. In this method, a nested structure of families of hypothesis with increasing VC dimension are identified. In this structure, each family of hypothesis is more complex than the previous one. The family that is selected is the one with the minimum value of the generalization error upper bound (2.45). In this manner, SRM selects a learning model in which the complexity is adapted to the complexity of the data. Figure (2.2) shows the scheme of SRM in the model selection procedure. In this figure, the most complex family of hypothesis (S_k) is the one with the highest VC dimension and in consequence with the highest value of the VC confidence. This family has the lowest value of the empirical error, but is prone to overfitting. As the VC dimension goes down the empirical risk increases and underfitting is more probable. Choosing an appropriate VC dimension leads to the lower expected risk (generalization error). The VC dimension is difficult to calculate in nonlinear models such as neural network. The estimation is simpler in linear models such as SVM [59]. It can be shown [123, 124] that an upper bound of VC dimension of a linear model, with margin ρ is

$$VC \leq min\{d, \frac{4r^2}{\rho^2}\} + 1, \tag{2.46}$$

where r is the radius of the smallest sphere containing all instances and d is the dimensionality of the feature space. As it has seen in (2.46), larger margins correspond to the learning models with lower VC dimension.

2.4 Conclusions

In this chapter we have discussed how ensembles can be used to address classification problems. Classification refers to the prediction of the class label of an instance from the vector of attributes that characterizes it. The learning strategy is to identify patterns in the training data that allow us to make correct predictions on new instances. To do so, the learner should be sufficiently flexible to capture the complexity of the regularities in the training data. However, a model that is too flexible has the risk of overfitting and may not generalize well. Ensemble methods attempt to alleviate some limitations of individual learners by combining the decision of several base learners. Accuracy and complementarity of the base learner decisions are key features to build effective ensembles. The prediction accuracy of each base learner should be at least better than random guessing. In randomized ensembles, such as bagging, random forest and class switching, complementarity is an indirect consequence of the diversity among the classifiers. In some other ensemble learning methods, such as boosting or negative correlation learning, one actively seeks to generate complementary classifiers.

Finally, we introduced the concept of classification margin and discussed its relevance to the generalization capacity of a predictor. Margin of an instance refers to its distance to the decision boundary. The best-known margin-based classifier is SVM. In SVM, one identifies a separator hyperplane that correctly classifies the training instances while maximizing the margin. In the context of ensembles, the margin is defined as the number of the votes for the correct class label minus the number of the votes for the class that receives the highest number of votes and that is different from the correct one. This definition of margin is important to understand the performance of boosting ensembles such as Adaboost [5]. In this method, the margin tends to increase as the ensemble grows. However not all ensemble methods work by increasing the margin. Bagging, random forest and class-switching ensembles do not increase the margin. In this thesis we analyse the performance of bagging with sampling rates that are lower than the standard prescription. Lower sampling rates in bagging decrease the margin in comparison with standard bagging. The experiments performed in this thesis illustrate that bagging and random forest using lower rate sampling to train the base learners, are robust to noise in the class labels, in spite of having smaller margins.

Chapter 3

Learning in the Presence of Noise

Poor data quality and contamination by noise are unavoidable in many real-world classification problems [14, 15]. This can mislead the learning algorithms used for automatic induction from these data. Two types of noise can be present in these problems: class-label noise and polluted feature values [14, 15]. Class-label noise is the consequence of incorrect manual labeling, missing information or failures in the measuring process. Feature noise is often the result of a faulty data gathering process [14, 15]. Class-label noise typically has a more pronounced misleading effect than feature noise, except when most of the feature values are corrupted [14].

Frénay et al. [15] identify three types of label noise, characterized by different statistical models: The Noisy Completely at Random Model (NCAR), in which the probability of a class-label error is independent of the values of the features, the actual class of the instance and the noise rate. To simulate this type of noise the class labels of randomly selected instances are changed to a different class label at random. The second model is Noisy at Random (NAR). Labelling errors in this model are assumed to occur with a different probability for each class. NAR is useful to characterize tasks in which some classes are more susceptible to mislabeling than others. The third model is Noisy Not at Random (NNAR). In this case, the probability of an error depends on the actual class label and on the values of the features. This model should be used when some regions of the feature space, such as boundaries or sparse regions, are more prone to noise than others.

Noise can be handled in a preprocessing step (data cleansing) or during the learning process, assuming that the algorithms used for induction from the contaminated data are robust [15].

3.1 Data cleansing

To mitigate their harmful effects, noise and outliers can be eliminated in a preprocessing step, before the selected learning algorithm is applied. For instance, it is possible to use statistical models or clustering-based methods to detect outliers. Patterns and association rules can also be used in the cleansing process [125]. An example of a pattern-based data cleansing algorithm is described in [126, 127]. In this method, local SVM's are used to identify and remove instances that are suspected to be noise. For each particular training instance, k-NN is applied to locate nearby instances. An SVM is then trained on these instances to find the optimal separating hyperplane in that neighborhood. If the label predicted by this locally trained SVM does not coincide with the actual label, the instance is identified as noisy and discarded. This cleansing method has been tested on real and artificial datasets, where it showed improvements over k-NN. In [128], noisy instances are removed based on wrappers of different classification methods. In this study, the best results were obtained by removing or cleaning instances based on the prediction of a SVM built with the rest of the training data. Noisy instances are often included in the set of support vectors by a SVM classifier. Based on this observation, Fefilatyev et al. [129] propose to manually remove support vectors that are identified as noise by an expert. Then, a new SVM is built on the cleansed dataset. This process is iterated until no more support vectors are identified as noisy instances.

Noise handling has also been considered in the context of ensembles of classifiers. In [130], an ensemble of three classifiers (an univariate decision tree, a k-nearest neighbor and a linear machine) is used to identify noisy instances. Since different learning algorithms have different biases, employing information from a variety of learning algorithms can be more informative than a single one [130]. The noise detection protocol in this method is the following: The training set is partitioned into 4-folds. Then, an ensemble is trained using three of these folds. Finally, the ensemble is used to identify noisy instances in the fold not used for training. In this study two filtering strategies are considered: majority filtering and consensus filtering. In majority filtering a particular instance of the leave-out fold is identified as noisy when the predictions of more than half of the learners do not coincide with the label of that instance. The consensus filtering rule is more conservative. It identifies an instance as noisy when all the members of ensemble misclassify it. This process is repeated for each different fold to identify noisy instances in all 4 folds. In this study, majority filtering had better overall performance than consensus filtering.

In a similar approach, the authors of [131] used an ensemble of 25 classifiers of different types to detect noisy instances. The increased diversity of this heterogeneous ensemble reduces the risk of false positives in the noise detection process. In addition, different intermediate strategies between majority and consensus filtering are explored. In the problems investigated, the optimal threshold of erroneous ensemble predictions used to identify an instance as noisy is 23 out of 25, which is close to consensus filtering.

In [132], an ensemble of Top-down Induction of Logical Decision Trees (Tilde) is used to filter the training data. In this study either cross validation or bootstrap sampling are used to generate the training sets on which the base learners are built. Majority and consensus filtering are then applied to identify noisy instances. The best results are obtained with majority filtering. In addition, boosting is proposed as a noise detection strategy. The idea is to filter instances that have higher weights after a number of pre-defined iterations of the boosting algorithm. In the problems investigated this strategy does not perform well. However, in datasets with different number of instances and noise levels the results can be different.

In [133], a distributed method for large datasets is presented. In this method the original training set is partitioned into small subsets. A classifier is induced from each of the these subsets. These classifiers are used then to classify the instances in the original training data. The local error of an instance is defined as the fraction of classifiers whose training data included that particular instance and misclassified it. The global error for an instance is computed using those classifiers that did not include that instance in their training sets. Majority and consensus filtering are used to identify noisy instances. A necessary condition for an instance to be identified as noisy is that it be misclassified by the base learners whose training sets induced it. The justification is that a classifier has usually higher accuracy on instances that are in its training set. In this work majority filtering is found to yield better results than consensus filtering.

In [134], a noise detection strategy that combines ideas from bagging and boosting is investigated. In this method the weights are assigned to the instances. The instances are initialized with equal weights. Then bootstrap samples from the original training data are used to build different classifiers. For each instance a noise count is computed as the number of base learners that have misclassified the instance. The noise count is used in a boosting-like iterative process in which the weight of the instances with higher noise counts is decreased. This process is iterated for a predefined number of rounds. Finally, instances with noise count values higher than a threshold are labelled as noisy.

In most of the noise detection methods based on ensembles, majority and consensus filtering are used. More instances are identified as noise when majority filtering is applied with respect to consensus filtering. Majority filtering was found to be better than consensus filtering in most cases. This is probably due to the fact that consensus filtering is too restrictive and does not remove enough instances. However, in [131] strategies very close to consensus filtering worked best. This indicates that, the optimal filtering strategy depends on the noise level and on the particular classification problem considered.

3.1.1 Robust learning algorithms

Another strategy to deal with noise is the design of robust learning algorithms. For instance, pruning is used in decision trees to reduce overfitting : the presence of noise tends to increase the size of the decision trees induced from the contaminated training data. Pruning can thus be an effective way to improve the robustness of decision trees [64, 130]. Another robustifying strategy is to explicitly incorporate in the learning algorithm the fact that the values of the features and the class labels can be polluted by noise. This strategy is adopted in the construction of Credal Decision Trees [135]. These types of trees are grown using the Imprecise Info-Gain Ratio (IIGR) as a splitting criterion. In this method the values of the features and class labels are approximated using probabilities and uncertainty measures.

It is also possible to adapt the algorithms used to build Support Vector Machines to improve their robustness to class-label noise. For instance, in [136] the hinge loss is replaced by a related loss function that takes into account the amount of noise in the data. Unfortunately, with this loss function the optimization problem becomes non-convex. Heuristic optimization methods are then used to search for the global minimum of this non-convex problem. Promising results were obtained by this robust SVM in problems with asymmetric class noise (NAR model). A drawback of this method is that it is necessary to estimate the amount of noise in the data before hand. Another robust version of SVM, called P-SVM (Probabilistic SVM) is proposed in [137] to classify magnetic resonance medical images. The P-SVM takes as inputs not only class labels but also class probability estimates. These probabilities are used to estimate the confidence on the labeling of each instance. The lower the confidence on the label, the lower the weight of that instance in the learning process. A practical limitation of this method is that one needs both qualitative (class labels) and quantitative (class posterior probabilities) information on the classes.

The problem of induction from noisy data has also been extensively addressed in the area of ensemble learning. In [138], Ali and Pazzani analyze the behavior of multiple classifier systems in the presence class-label noise. They observed that the improvements of the ensemble with respect to a single learner are generally smaller when the training data are contaminated with class-label noise. However, the reduction is not uniform and depends on the type of ensemble used.

Noise is not always harmful. In fact, noise injection is a powerful regularization mechanism that has the potential of improving the generalization capacity and robustness of prediction systems. In particular, randomization is used to build diverse ensembles that have good generalization capacity [11–13, 17, 18, 20, 84, 139–143]. Furthermore, randomized ensembles, such as bagging and random forests, have been shown to be robust classifiers. By contrast, adaptive ensembles, such as boosting, are very sensitive to class-label noise [17, 18, 20, 139, 140]. The differences between these two types of ensembles can be explained by how errors are handled during the training phase: In bagging and random forest, the randomness injected during the construction of the ensemble is not correlated with the noise. For this reason, the influence of the different instances is equalized during training process [144]. By contrast, boosting increases the weights of misclassified instances irrespective of whether they are correctly labeled or not. The emphasis on correctly labeled instances that are difficult to classify is beneficial, because it reduces the classification bias. However, the focus on outliers tends to mislead the learning process. The adaptivity that makes boosting such a powerful learner also renders it overly susceptible to noise.

There are many proposals to improve the robustness of boosting to class-label noise. In most of these variants the weight update rule is modified to reduce boosting's sensitivity to noise. A successful strategy is to use less aggressive weight updates. In standard boosting the weight updates are exponential. Using slower updating scheme moderates the emphasis on misclassified instances. This is generally advantageous because some of this misclassified instances could be outliers [145]. In BrownBoost [146] misclassified instances with small negative margins are assigned higher weights, as in Adaboost. By contrast, instances whose margin is negative and above a specified threshold receive lower weights. The rationale behind this weight updating strategy is that instances in regions with a large class overlap tend to have low margins. By emphasizing these instances it is possible to model the classification boundary in more detail. Large negative margins correspond to isolated instances, which are far from the classification boundary. These instances are likely to be outliers and should therefore be discarded. In [140], Brownboost is shown to be more robust than Adaboost in a limited experimental setting (5 datasets for 20% class-label noise). Another way of avoiding excessive emphasis on misclassified instances is to discard instances whose weight is above a threshold [147]. The value of the threshold can be determined using a validation set. This algorithm is shown to be more robust than standard Adaboost in 8 datasets with low to medium class-label noise (up to 10%). None of these studies [140, 147] compares the results of robust boosting ensembles with bagging. Finally, it is possible to combine bagging and boosting strategies to improve the accuracy and robustness of the resulting ensembles [21, 148].

However, as far as we are aware, the effectiveness of these hybrid ensembles has not been systematically evaluated in experiments with class-label noise.

In [22] the authors propose to use credal decision trees to improve bagging's resilience to label noise. The results obtained with these types of ensembles in the low to medium noise regime (0%-10% class-label noise) are comparable to bagging of C4.5 trees. For higher noise levels (20%-30%) bagging of credal trees is more accurate than bagging of C4.5 trees.

3.1.2 Subsampling as a regularization mechanism

Subsampling can also be used to design robust bootstrap ensembles. The individual classifiers of a bagging ensemble are built by applying the same base learning algorithm to different m-out-of-n bootstrap samples from the original training data. In standard bagging the number of instances in the bootstrap sample, m, is equal to the number of instances in the original training data, n (i.e. $m = n$). This choice of m need not be optimal. As an illustration, the performance of bagged nearest neighbors is comparable to the nearest neighbor algorithm itself [10]. However, if each bootstrap sample contains on average less than 50% distinct instances from the training set, the accuracy of bagged nearest neighbors can actually improve. In fact, if the sampling ratio tends to 0 as the training set size tends to ∞, the performance of bagged nearest neighbor tends to the Bayes (optimal) error [23]. Another study [24] shows that subbagging with low sampling ratios generally improves the accuracy of bagging when stable classifiers are combined. The optimal subsampling ratio can be effectively determined using out-of-bag data [25]. In chapter 4 of this thesis we show that subsampling has potential to improve the robustness of bagging to class-label noise in some classification problems [60].

One way of understanding how the sampling ratio can influence the performance of bagging ensembles is to consider the average number of distinct instances in each bootstrap sample. The dependence of this value with the sampling ratio is displayed in Figure 3.1. In standard bagging (100% sampling ratio) each bootstrap sample contains on average 63.2% different instances from the original training data [149]. The remaining 36.8% are repeated instances. As the sampling ratio becomes smaller, the number of distinct instances in each bootstrap sample decreases. Eventually, only one instance is sampled for a sampling ratio of $1/N$, where N is the size of the training set. The classifier built on such a sample would predict the class label of the single instance in the sample. Hence, the ensemble decision would be the majority class in the training data, irrespective of the values of the features. On the other extreme, bootstrap samples obtained with high sampling ratios contain most of the training instances. In such cases most base learners

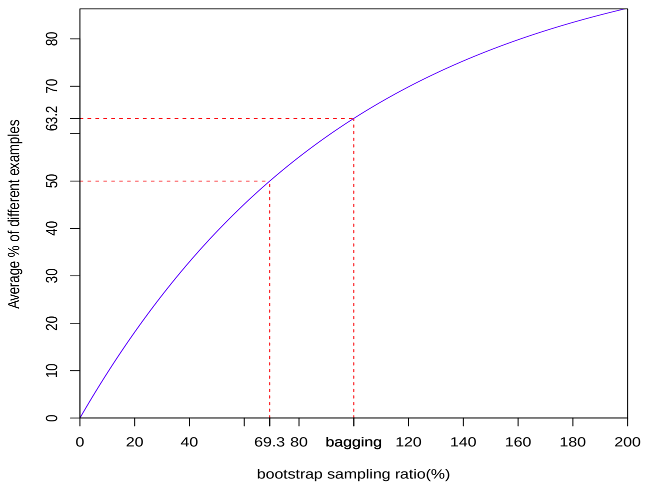

Figure 3.1: Average percentage of the unique training instances with respect to the size of the bootstrap sample

are very similar; the diversity arises only from having different repeated examples in the different bootstrap samples. Ensembles built using these extreme values of the sampling ratio will not in general have good generalization. The optimal performance is generally obtained at intermediate values of the sampling ratio [25]. Furthermore, the optimal sampling ratio need not coincide with the standard prescription (100%).

An interesting regime corresponds to sampling ratios smaller than 69.3% (see Figure 3.1). For values below this threshold, fewer than 50% of the original training instances are included in each bootstrap sample. This means that each instance is present in less than half of the classifiers of the ensemble. In this regime, the class label given by the ensemble for each training instance is strongly influenced by the class label of nearby instances. In consequence, subsampling has the potential to increase the diversity of the classifiers in the ensemble. Higher diversity results in more variability in the votes and therefore in lower margins. We conjecture that using sampling ratios in this regime is an effective strategy to handle class-label noise in classification ensembles.

Chapter 4

Small Margin Ensembles

In this chapter we explore how subsampling affects the classification margins in ensembles. The goal is to understand the relation between ensemble diversity, margins and robustness. We first present the results of a set of experiments that illustrate the effect of subsampling on the classification margin. Then we analyze how subsampling can act as a regularization mechanism that reduces the influence of mislabeled data.

4.1 Subsampling and margins

To understand how classification margins are affected by susbsampling we have carried out a series of experiments in the classification problems *Threenorm*, *Twonorm* and *Ringnorm* [6]. These are synthetic datasets for which the optimum Bayes decisions are known. Bagging ensembles and random forests of 500 trees were trained using different bootstrap sampling ratios: 100%, which is the standard prescription, 20% and 5%. Ensembles trained on a noiseless set are used as a baseline. The bagging and random forest ensembles were built on the same training sets, which consist of 300 instances. The boosting ensembles were built on different sets of the same size. Additional ensembles were then built on copies of these sets contaminated with 20% label noise. The noise was simulated using the NCAR model. Bagging and random forest ensembles were tested using the out-of-bag error [149]. The out-of-bag data of a particular classifier consists of those instances which are not included in the bootstrap sample used to build that classifier. Since they are not used for training, they can be employed as independent test data. Thus, to compute the out-of-bag error, each instance in the training set is classified using only the votes of those predictors whose training sets do not include that particular instance. Besides providing a good estimate of the generalization capacity, the out-of-bag method allows us to analyze how the injected noise is handled by the ensemble: The same instances, including those whose class labels have been altered, are used both for training and for testing. To allow comparisons across ensembles, the performance of boosting was evaluated on the training data used to build the bagging and random forest ensembles.

Scatter plots of the posterior probability of class 2 versus the fraction of class 2 votes for the instances in the evaluation set are given in figures 4.1 and 4.2. The results displayed correspond to experiments with the different ensembles, sampling ratios and class-label noise levels. In bagging and random forest, the fraction of class 2 votes for a particular instance is estimated using the classifiers for which that instance was in the out-of-bag set (i.e. the set of instances not used to train that particular classifier). For boosting, all the classifiers in the ensemble were used. Figure 4.1 presents the scatter plots for an execution of *Threenorm*. Similar results are obtained in the other datasets. The plots included in this figure display (by rows) the results for standard bagging (100% sampling ratio), bagging using 20% sampling, 5% sampling and boosting. The results for a noiseless training set are presented in the first column. The results for a training set with 20% injected label noise are presented in the second column. Figure 4.2 shows the corresponding plots for random forest. In all plots the class 1 (class 2) instances are marked as empty circles (triangles). The instances whose class has been changed into class 1 (class 2) are marked as filled circles (triangles). The lines shown in the plots define the decision boundaries for the Bayes classifier (horizontal line) and the ensemble (vertical line). In addition, the errors for the ensembles and the Bayes classifier are displayed on the right bottom corner of the plots. For the problems with injected label noise, error values considering noise (N) and without noise (O) are given. The Bayes classifier and the ensembles agree in the classification of instances located in the upper right and bottom left quadrants. The ensemble and the Bayes predictions are different for the remaining instances.

Several noteworthy features are revealed in these plots. In the noiseless problem (left column), the Bayes classifier assigns fairly high margins to most instances. The classification margins of bagging ensembles are lower than those of the Bayes classifier. Furthermore, they become smaller as the sampling ratio decreases. However, bagging ensembles with sampling ratios of 20% (second row) are more accurate than standard bagging, with 100% sampling (first row), in spite of the fact that the margins are smaller. The accuracy obtained with a sampling ratio of 5% is comparable to standard bagging. This is contrary to the view that accuracy should improve with increasing margin. A possible explanation of this behavior is that the different bootstrap samples have fewer common instances as the sampling ratio decreases. In consequence, the base classifiers become more diverse. This increased diversity initially leads to accuracy improvements. However, if the sampling ratio is reduced beyond a threshold, the individual classifiers

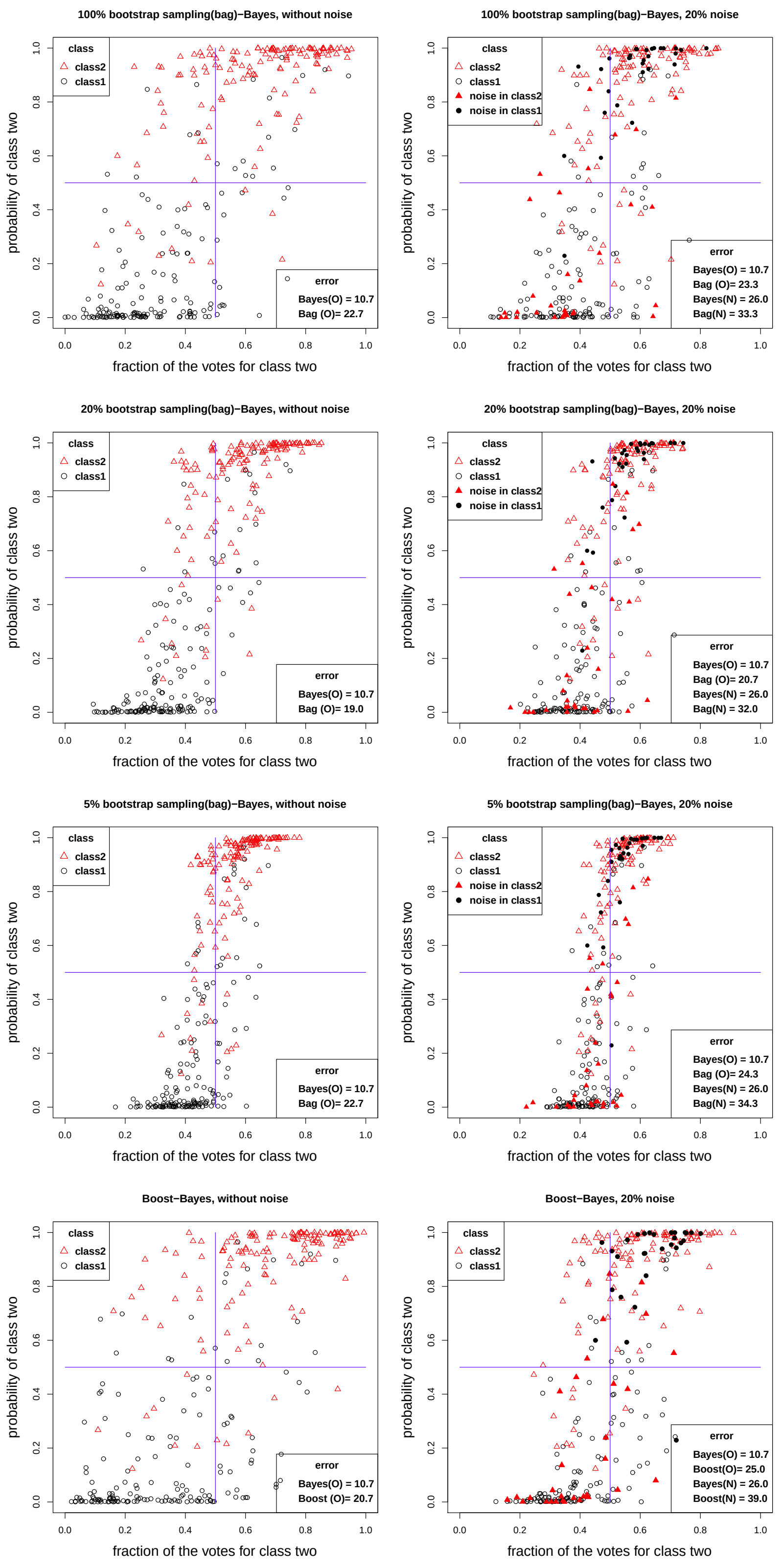

Figure 4.1: Scatter plots of the posterior probability of class 2 versus the fraction of ensemble class 2 votes for each instance in the evaluation set. Results are given for *Threenorm* without noise (left column) and with 20% noise (right column). The plots correspond to bagging ensembles with sampling ratios: 100% (first row), 20% (second row) and 5% (third row). The results for boosting are presented in the forth row.

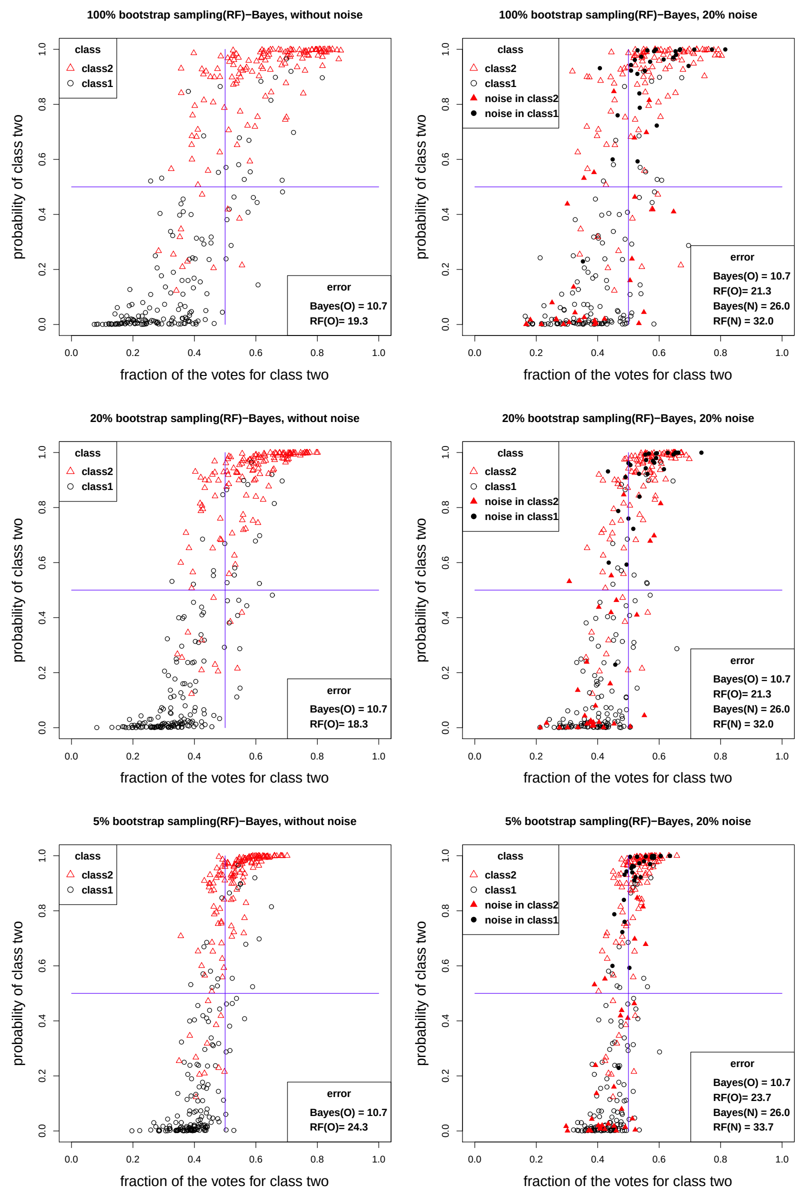

FIGURE 4.2: Scatter plots of the posterior probability of class 2 versus the fraction of ensemble class 2 votes for each instance in the evaluation set. Results are given for *Threenorm* without noise (left column) and with 20% noise (right column). The plots correspond to random forest ensembles with sampling ratios: 100% (first row), 20% (second row) and 5% (third row).

become inaccurate. The error reduction that results from the aggregation of their decisions in the ensemble is not sufficient to compensate the lack of accuracy of the base learners. As a result, the fraction of instances with small and negative margins increases (see 5% sampling, third row, left plot).

A similar behavior is observed when label noise is present in the training set (right column): The classification margins are now smaller in all cases, relative to the noiseless situation. The test error (second row, right column) initially improves with decreasing sampling rates. However, if the sampling ratio is too low the performance of the ensemble eventually deteriorates. A similar behavior has been reported in class-switching ensembles [13].

The behavior for boosting (last row) is somewhat different. Because of its adaptive nature, boosting produces larger margins than bagging. While this is effective in the noiseless setting, it can be disruptive in noisy problems. In particular, when 20% class-label noise is injected boosting has the worst accuracy.

The results for random forest (shown in Figure 4.2) are qualitatively similar to those of bagging. However, the margins in random forest ensemble are typically smaller than in bagging or boosting. This is a consequence of the higher diversity provided by the random trees that make up the ensemble. From the experiments performed in this study the best overall results are achieved by random forests built with the standard 100% sampling ratio. The larger initial diversity of random forest implies that there is less room for improvement as the sampling ratio decreases. The variability introduced by subsampling could in fact be detrimental to the accuracy of the ensemble. Therefore, subsampling is in general not as effective in random forest as it is in bagging. The validity of this qualitative analysis is confirmed by the empirical evidence presented in the section on experiments.

4.2 Experimental evaluation

In this section we present the results of an empirical investigation of the performance of bootstrap ensembles in the presence of label noise. The experiments are designed to assess how different sampling ratios affect the robustness of such ensembles. A total of 25 datasets from the UCI repository [150] and other sources [6] are used. They include synthetic data (*Ringnorm*, *Twonorm*, *Threenorm* and *Tic-tac-toe*) and classification problems from different application domains. The characteristics of the datasets are summarized in Table 5.1. They have been selected to cover a wide spectrum: there are problems with high and low numbers of attributes (e.g. *Sonar* and *Balance*, respectively),

Dataset	**Instances**	**Test**	**Attrib.**	**Classes**
Australian	690	230	14	2
Balance	625	198	4	3
Breast W.	699	233	9	2
Diabetes	768	256	8	2
German	1000	333	20	2
Heart	270	92	13	2
Hepatitis	155	51	19	2
Horse-Colic	368	122	21	2
Ionosphere	351	117	34	2
Iris	150	50	4	3
Labor	57	38	16	2
Liver	345	115	6	2
Lung Cancer	32	10	56	3
Magic	19020	6340	11	2
New-thyroid	215	143	5	3
Ringnorm	300	2000	20	2
Segment	2310	1540	19	7
Sonar	208	699	60	2
Threenorm	300	2000	20	2
Tic-tac-toe	958	319	9	2
Twonorm	300	5000	20	2
Vehicle	846	564	18	4
Votes	435	145	16	2
Waveform	300	5000	21	3
Wine	178	59	13	3

TABLE 4.1: Characteristics of the classification problems and testing method

with small and large number of instances (e.g. *Magic04* and *Lung Cancer*, respectively), and with different numbers of classes.

The protocol used in the experiments is similar for all datasets. The only difference is in the generation of the training and test sets. For the synthetic datasets (*Threenorm*, *Ringnorm* and *Twonorm*) we generate a training set of 300 instances and a test set of 2000 instances. For the remaining datasets, 2/3 of the available data are used for training and 1/3 for testing. Stratified sampling is used to guarantee that the class distributions in the training and test sets are similar to the complete dataset class distribution. For each problem and realization of the training and test sets, the following steps are carried out:

1. Label noise is injected in the training set with different rates: 0% (no noise), 5%, 10% and 20%. In each case the class label of the randomly selected training instances is changed to a different class, also at random. This corresponds to the

Noisy Completely At Random noise (NCAR) model [15]. Uniform noise was used to avoid making specific assumptions about the structure of the noise.

2. For each contaminated training set, six bagging ensembles composed of 500 unpruned CART (Classification And Regression Tree) trees [64] were built. The bootstrap sampling ratios used are: 10%, 20%, 40%, 60%, 80% and 100% (standard bagging). The CART trees were grown until pure class nodes were obtained. No pruning was applied to the fully grown decision trees. Random forest ensembles were built on the same training sets and sampling ratios. Random forest is a bagging ensemble composed of random trees. In random trees the splits at the inner nodes of the tree are selected from those that involve only a random subset of features. The size of these subsets was set to the square root of the number of features for each dataset [151].

3. The generalization performance of all ensembles is gauged using the error on the test set. To obtain comparable results across all the ensembles considered no noise was injected in the test set.

The test errors reported in the tables are averages over the 100 realizations of the training and test sets.

4.2.1 Results

To give an overall view of the results, we have computed the averages of the test error changes in the 25 problems investigated, for each noise level, sampling strategy and ensemble method (bagging and random forest). The results are presented in Table 4.2 as the relative error change, using standard bagging in the noiseless setting as the reference value. This reference value is marked in boldface in the Table. Values below 1 indicate that, on average, the corresponding method outperforms standard bagging in the noiseless setting. Values above 1 signal a higher average test error.

In addition, the average error changes with respect to the noiseless setting for each ensemble type are shown in Table 4.3. The reference values are highlighted in boldface. These results serve to analyze how the accuracy of ensembles built with the different sampling ratios is affected by class-label noise. The specific average test errors for the individual datasets are presented in the appendix: Tables A.1, A.2 and A.3 for bagging and Tables A.4, A.5 and A.6 for random forest ensembles.

An analysis of the results presented in Table 4.3 reveals that the loss of accuracy with respect to the noiseless setting is very different for different sampling ratios. For standard

Table 4.2: Relative error change for bagging and random forest for the different levels of noise and sampling ratios. The reference value corresponds to standard bagging in the noiseless case (marked in boldface as **1.00±0.00** in the table).

	Noise	10	20	40	60	80	100
Bag	0	1.38±1.43	1.06±0.41	0.98±0.16	0.96±0.08	0.98±0.08	**1.00±0.00**
	5	1.41±1.58	1.11±0.64	1.05±0.27	1.08±0.25	1.10±0.23	1.18±0.25
	10	1.45±1.70	1.19±0.92	1.13±0.43	1.18±0.44	1.28±0.47	1.38±0.57
	20	1.55±1.98	1.42±1.51	1.44±1.16	1.60±1.10	1.72±1.13	1.83±1.19
RF	0	1.77±2.85	1.49±2.24	1.21±1.44	1.06±0.91	0.97±0.58	0.94±0.42
	5	1.75±2.62	1.48±2.08	1.23±1.31	1.13±0.91	1.05±0.68	1.01±0.55
	10	1.77±2.56	1.48±1.98	1.27±1.36	1.20±1.03	1.15±0.82	1.16±0.76
	20	1.81±2.43	1.62±2.03	1.51±1.55	1.48±1.36	1.51±1.31	1.53±1.26

Table 4.3: Relative error change averaged over all datasets for bagging and random forest for the different levels of noise. The reference values are the test errors bagging and random forest noiseless case (marked in boldface in the first and fifth rows of the table).

	Noise	10	20	40	60	80	100
Bag	0	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**
	5	1.01±0.09	1.03±0.14	1.06±0.14	1.12±0.19	1.12±0.20	1.18±0.25
	10	1.03±0.11	1.07±0.22	1.13±0.30	1.22±0.38	1.30±0.45	1.38±0.57
	20	1.08±0.13	1.23±0.44	1.43±0.95	1.66±1.02	1.75±1.12	1.83±1.19
RF	0	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**
	5	1.05±0.18	1.02±0.08	1.05±0.08	1.09±0.16	1.08±0.11	1.06±0.11
	10	1.09±0.30	1.06±0.18	1.09±0.16	1.14±0.15	1.18±0.18	1.21±0.26
	20	1.18±0.51	1.22±0.39	1.35±0.37	1.44±0.39	1.55±0.49	1.59±0.53

bagging with 20% noise injected, the average error increase with respect to the noiseless case is 83%. This large increase should be expected, given the high level of noise injected. By contrast, if a 10% sampling ratio is used, the average error increase is only 1.0%, 3.0% and 8.0% for the 5%, 10% and 20% label noise rates, respectively. An interesting observation is that these error increments are significantly lower than the corresponding levels of injected noise. Using lower sampling ratios in bagging tends to increase the variability of the base classifiers. This larger ensemble diversity generally translates into more robust classification. The remarkable robustness to class-label noise of these ensembles is illustrated in greater detail by the results presented in Tables A.1, A.2 and A.3 in the appendix. In some cases, there is even an improvement in the classification accuracy when noise is injected. For instance, the best overall accuracy of bagging in *Breast* with 20% noise is achieved using a 10% sampling ratio: The test error goes from 4.1% when no noise is injected to 3.5% when the training data has 20% noise. By contrast, when standard bagging is used, the test error increases almost 5 percentage points (from 4.3% with no noise to 9.2% with 20% noise).

For random forest ensembles, a similar, albeit less marked effect, is observed in Table 4.3: The deterioration with the level of noise injected is more pronounced for larger sampling ratios (18% increment with a 10% sampling ratio and 59% with a 100% sampling ratio). However, the baseline accuracy of random forest ensembles at low sampling ratios is rather poor: In the noiseless setting, the average error rate of random forest with a 10% sampling ratio is 77% larger than standard bagging (see Table 4.2). One of the reasons why subsampling is not as effective, is that random forests are typically more diverse than bagging ensembles. This diversity makes standard random forest more robust to noise (see rightmost column of Table 4.2). Using lower sampling ratios is not as effective in increasing the diversity of the random trees. Therefore, subsampling does not lead to systematic accuracy improvements in this type of ensemble.

Finally, from the analysis of the results displayed in Table 4.2 one concludes that the best overall performance in the noiseless setting is achieved using standard random forests (0.94). The difference with standard bagging is 6 percentage points on average. However, the difference between standard random forest and bagging using 60% sampling ratio is only of two percentage points (values 0.96 and 0.94 in Table 4.2). As the noise level increases the best overall accuracy corresponds to bagging using 20-40% sampling ratios (1.42 and 1.44 in the Table 4.2 for a 20% noise rate).

4.2.2 Accuracy as a function of ensemble size

The error curves displayed in Figures 4.3 and 4.4 trace the dependence of the average test error of bagging on the number of classifiers in the ensemble. The classification problems used to illustrate this dependence are *Australian* (Figure 4.3) and *Threenorm* (Figure 4.4). The curves displayed correspond to different sampling ratios and noise levels: noiseless setting (top left plot), 5% (top right plot), 10% (bottom left plot) and 20% (bottom right plot) noise rates. The qualitative features of these error curves are similar in all the classification problems investigated.

When no noise is injected, the error curves for *Australian* converge to their asymptotic (infinite ensemble) limit after approximately 50 trees. As more noise is injected larger sizes are required for convergence. In this dataset the qualitative behavior of the error as a function of ensemble size is similar for the different sampling ratios. By contrast, in *Threenorm* (Figure 4.4), the convergence of the ensemble error curves is slower for smaller sampling ratios.

4.2.3 Statistical significance of the results

A record of the statistically significant differences in accuracy with respect to the standard ensembles in the 25 classification problems investigated is given in Tables 4.4 and 4.5 for bagging and random forest, respectively. In each cell of these tables the number of times a given method wins, draws or looses against standard bagging (Table 4.4) or

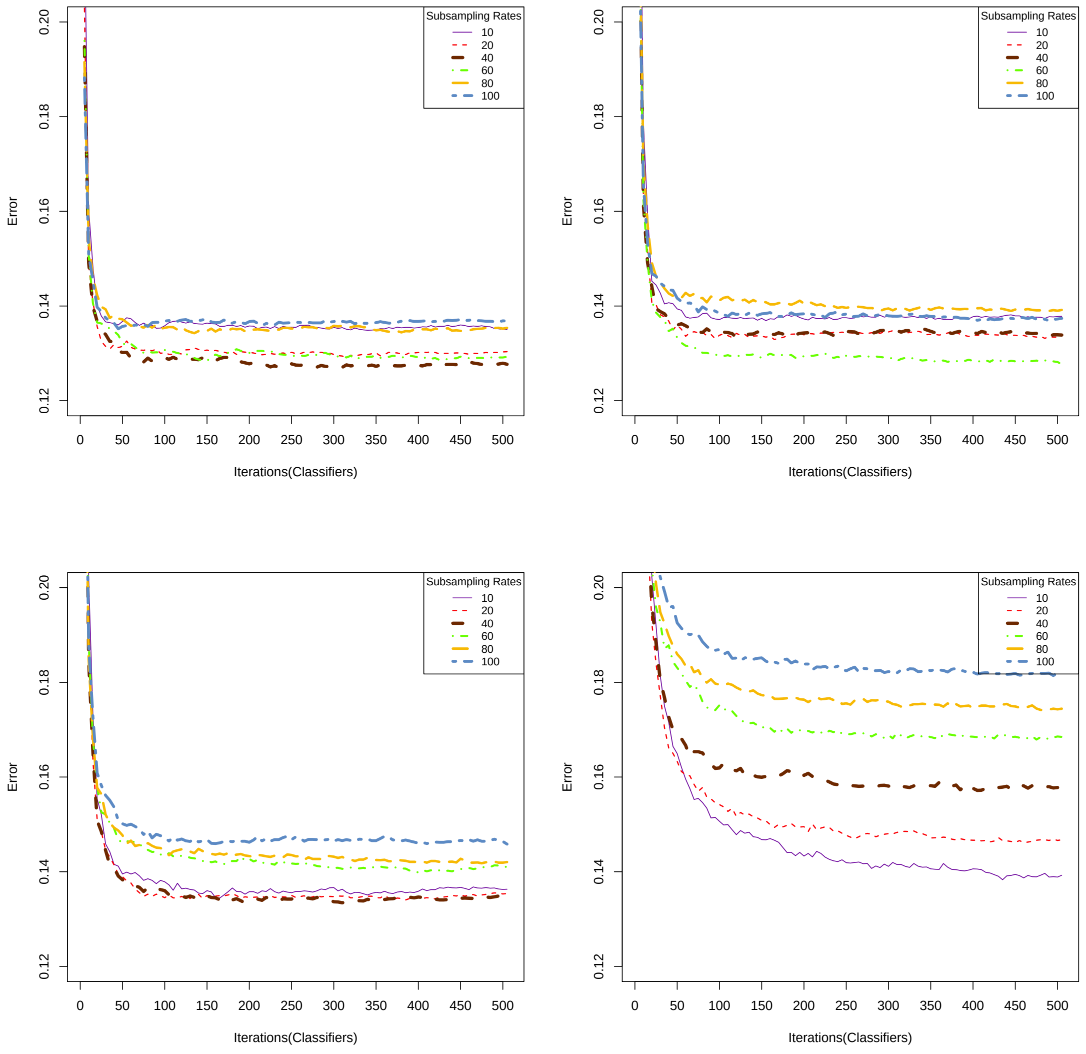

FIGURE 4.3: Average test error of bagging in the *Australian* dataset: Noiseless setting (top left); 5% (top tight), 10% (bottom left) and 20% (bottom right) noise rates. The different curves in each plot correspond to different sampling ratios.

standard random forest (Table 4.5) is displayed. Paired t-tests with $\alpha = 0.05$ are used to determine the significant wins and losses. A draw is recorded if the differences between the test errors are not statistically significant.

From the results presented in these tables one concludes that subsampling is more effective at higher levels of label noise. For instance, from Table 4.4, bagging using a 10% sampling ratio and 0% noise significantly outperforms standard bagging in 9 datasets and obtains lower accuracy in 11 datasets. When the noise rate is increased to 20%, the situation is reversed: there are 20 wins and only 3 significant losses.

An analysis of the results for random forest in Table 4.5 leads to similar conclusions.

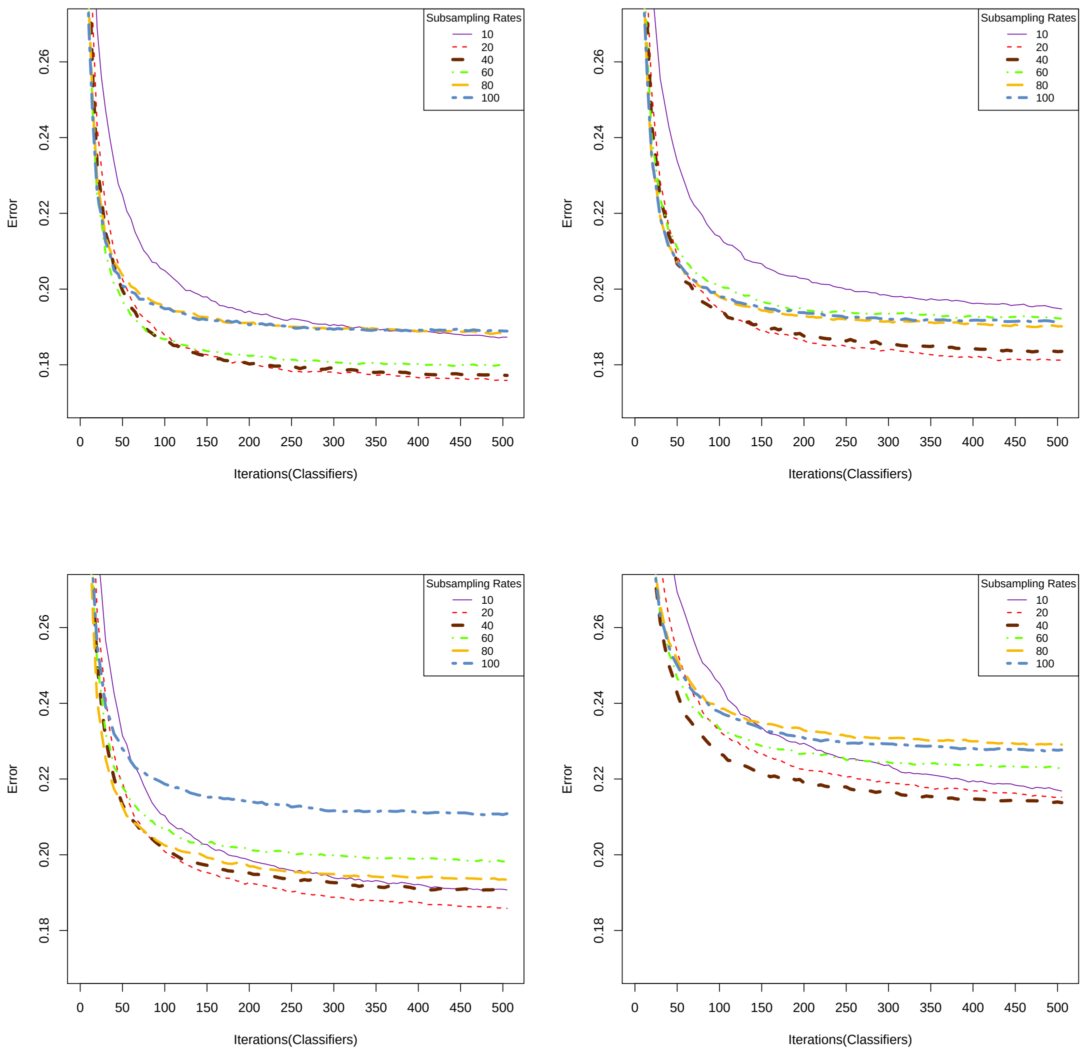

FIGURE 4.4: Average test error of bagging in the *Threenorm* dataset: Noiseless setting (top left); 5% (top tight), 10% (bottom left) and 20% (bottom right) noise rates. The different curves in each plot correspond to different sampling ratios.

Subsampling becomes more effective also at higher noise levels. The effect, however, is less salient than in bagging. In the noiseless case random forest using 20% bootstrap sampling outperforms the standard version in only one dataset and losses in 21 datasets. When the noise rate is increased to 20% the number of wins increases to 11 and the number of losses decreases to 9. Random forests built using the standard prescription (100% sampling ratio) have the best overall performance in the problems investigated for all noise levels. However, as the amount of class-label noise increases, subsampling becomes more effective and is actually advantageous in some problems.

Finally, the method proposed by Demšar in [152] is used to compare the performance of the ensembles across the different datasets. The comparison is made in terms of

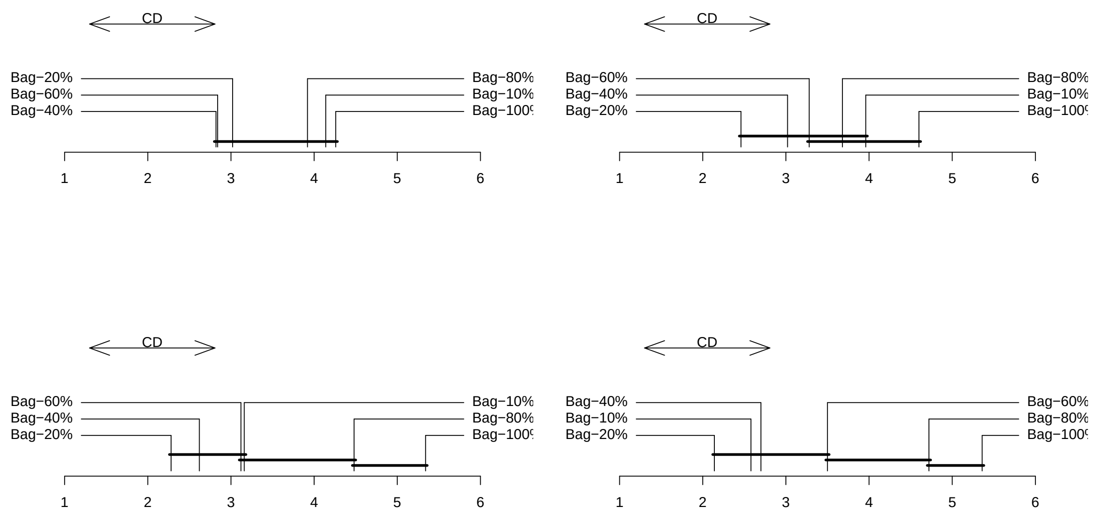

Figure 4.5: Comparison of bagging with different sampling ratios using the Nemenyi test, for datasets without noise (top left) and with 5% (top right), 10% (bottom left) and 20% (bottom right) noise rates. Horizontal lines connect sampling ratios whose average ranks are not significantly different (p-value < 0.05).

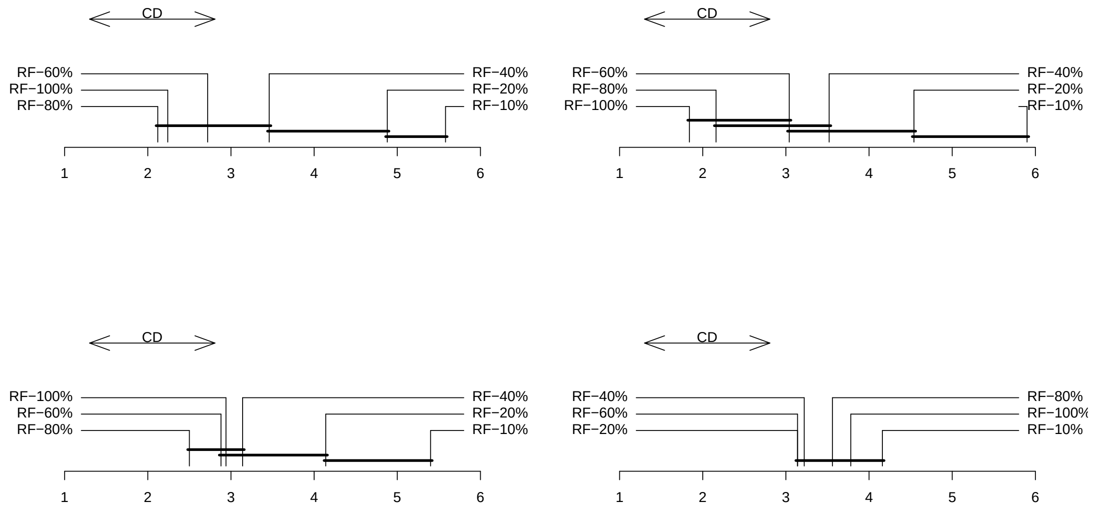

Figure 4.6: Comparison of random forest with different sampling ratios using the Nemenyi test, for datasets without noise (top left) and with 5% (top right), 10% (bottom left) and 20% (bottom right) noise rates. Horizontal lines connect sampling ratios whose average ranks are not significantly different (p-value < 0.05).

Noise (%)	10%	20%	40%	60%	80%
0	9/5/11	15/3/7	13/8/4	13/12/0	1/23/1
5	11/6/8	17/2/6	16/6/3	14/11/0	7/18/0
10	15/5/5	19/2/4	17/7/1	13/12/0	7/18/0
20	20/2/3	21/1/3	18/6/1	14/11/0	9/16/0

Table 4.4: Records for statistically significant wins/draws/losses for bagging with subsampling for different sampling ratios with respect to standard bagging (100 % sampling ratio).

Noise (%)	10%	20%	40%	60%	80%
0	1/1/23	1/3/21	2/14/9	5/12/8	3/19/3
5	0/5/20	1/6/18	2/14/9	1/19/5	1/22/2
10	3/4/18	3/9/13	3/16/6	4/19/2	3/21/1
20	8/6/11	11/5/9	9/10/6	6/16/3	3/19/3

Table 4.5: Records for statistically significant wins/draws/losses for random forest with subsampling for different sampling ratios with respect to standard random forest (100 % sampling ratio).

the average rank of each classifier in the problems considered. For a given dataset, the rank of the different ensembles is computed on the basis of the average test errors in the different realizations of the training and test sets. Figure 4.5 presents the results of these tests for different noise levels and sampling ratios. A Nemenyi test with p-value$<$ 0.05 is used to determine the statistical significance of the differences between average ranks. The critical distance above which these differences are considered significant is shown for reference (CD = 1.5 for 6 methods, 25 dataset and p-value$<$ 0.05). In this diagrams, if two methods are connected with a horizontal solid line, the difference between their average ranks is not statistically significant.

Figure 4.5 displays the results of the Demšar test for bagging ensembles in the noiseless setting (top left), and with 5% (top right), 10% (bottom left) and 20% (bottom right) noise rates. In all cases, standard bagging with 100% sampling ratio has the lowest average rank. When no noise is injected the highest accuracy corresponds to bagging with 20%, 40% and 60% sampling ratios. However, the differences with other sampling ratios are not statistically significant. The improvements over standard bagging for 20% and 40% sampling ratios are statistically significant in the problems with noise rates 5%, 10% and 20%. For the 20% noise rate, bagging ensembles that use 10%, 20%, 40% and 60% sampling ratios are significantly better than standard bagging (100% sampling ratio).

The results of the Demšar test for random forest are displayed in Figure 4.6. Standard random forest (i.e. with 100% sampling ratio) is the best method for the noiseless setting (top left plot) and for the data contaminated with a 5% noise rate (top right plot). However, the differences with ensembles built with 80%, 60% and 40% sampling ratios are not statistically significant. For these noise rates standard random forest significantly outperform ensembles built using 20% and 10% sampling ratios. When higher noise levels are injected (10%), the best performance corresponds to random forest with 80% sampling ratio. The improvements over ensembles built with 10% and 20% sampling ratios are statistically significant. For the highest noise level (20%) the method with the highest average rank is random forest with a 20% sampling ratio. However, in this case, none of the differences between the average ranks of the different ensembles are statistically significant.

4.3 Conclusion

In this chapter we have analyzed the resilience to class-label noise of bootstrap aggregation ensembles as a function of the size of the bootstrap samples used to train the individual predictors. The results of an extensive empirical evaluation show that bagging composed of unpruned decision trees trained on bootstrap samples whose size is between 10% and 40% of the size of the original training set are more resilient to label noise than standard bagging (i.e. using a 100 % sampling ratio). For random forests subsampling is effective only in noisy domains ($\approx$ 20% noise in the class labels) and in specific classification tasks. In most problems, for low noise levels the best results are obtained using high sampling ratios. In fact, using the standard sampling ratio to build random forests is a reasonable choice with a good overall performance in the problems investigated, especially in the absence of class-label noise. However, in noisy tasks, it is worth to explore the possibility of subsampling, because the optimal size of the bootstrap samples is problem dependent.

Experiments in synthetic data have been used to illustrate that the classification margins become smaller as the sampling ratio decreases. This effect occurs both in the noiseless setting and when class-label noise is injected. They provide empirical evidence that using smaller bootstrap samples to build the individual ensemble classifiers can lead to improvements in accuracy, especially in noisy domains. However, if the sampling ratio decreases beyond a threshold the accuracy of the ensemble abruptly drops. This abrupt deterioration of performance occurs at higher sampling rates in random forests than in bagging. The reason is that the margins are typically larger in bagging than in random forests. Since lower sampling ratios tend to reduce the margin, the potential improvements of subsampling for random forest are realized only in problems with high levels of class-label noise.

Chapter 5

Noise Detection

In the previous chapter we have shown that subsampling can be used to generate subbagging ensembles that are resilient to class label noise [60]. In this chapter we design a noise cleansing method based also on subsampling to improve the quality of the data before the training process. The idea is to predict the class label of each example in the training set using subbagging. Then, if the predictions of a specified fraction of the base classifiers do not coincide with the actual label, this instance is considered as noise.

The pseudo-code of the algorithm is shown in Algorithm 4. The first step is to estimate the optimum bootstrap sampling rate. Several bagging ensembles with different bootstrap sampling rates are trained. The one with the lowest out-of-bag error, bag^*, is kept. This ensemble is then used to detect the noisy or outlier instances. A particular instance is labeled as noisy when its label does not coincide with at least a fraction $\theta \leq 0.5$ of the classifiers in the ensemble bag^*. For instance, for $\theta = 0.2$, the class 1 (class 2) instances that receive more than 80% of the votes for class 2 (class 1) are marked as noisy instances. In general, the value of θ should be below 0.5. Otherwise, correctly classified instances would be marked as noise. Finding the optimum filtering parameter, θ^*, depends on the type of data and on the amount of noise. The parameter θ^* is determined by 3-fold cross-validation. A value of θ between 0 and 0.5 is used to cleanse the data in two of the folds using the ensemble bag^*. Then a classifier (standard bagging in our case) is trained on the cleansed data. Finally, the error is computed on the leave-out fold. The process is repeated exchanging the roles of the folds as test and training data. The optimum value for θ^* is determined as the one that minimizes the average cross-validation error. Finally, the instances in the training set that receive a percentage of votes greater than $(1 - \theta^*)$ and are incorrectly classified are marked as noise.

Algorithm 4: Noise detection by subbagging

```
Input:
$\mathcal{D}_{train} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{train}}$ % Training set
$y_i \in \{-1, 1\}$ %Class labels
$T$ % Ensemble size
$\mathcal{L}$ % Base learning algorithm

1  $rates = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$
2  min_error $\leftarrow \infty$
3  for $i \leftarrow 1$ to $length(rates)$ do
4      $sr \leftarrow rates[i]$ % Sampling rate
5      for $t \leftarrow 1$ to $T$ do
           $D_t \leftarrow Bootstrap_{sr}(\mathcal{D}_{train})$
           % Bootstrap is sampled from $\mathcal{D}_{train}$, uniformly with replacement, with sampling rate $sr$
           $h_t(\cdot) \leftarrow \mathcal{L}(D_t)$ %Train the base learner from $D_t$
6      $bag[i] \leftarrow \{h_t\}_{t=1}^{T}$ %Aggregation of the base learners using $rates[i]$
7      Calculate $oob\_error[i]$ (out-of-bag error of $bag[i]$)
8      Calculate $aggregated\_votes[i]$ (aggregated votes of $bag[i]$)
9      if $oob\_error[i] < min\_error$ then
           $min\_error \leftarrow oob\_error[i]$ %Out-of-bag error
           $sr^* \leftarrow rates[i]$
           $votes^* \leftarrow aggregated\_votes[i]$ %Save aggregated votes for optimal subbagging ensemble
10 $\theta$ =[0, 0.1, 0.2, 0.3, 0.4, 0.5] % Candidates for cleansing parameters
11 min_error $\leftarrow \infty$
12 for $i \leftarrow 1$ to $length(\theta)$ do
13     for $n \leftarrow 1$ to $N_{train}$ do
14         if $y(n)$ == -1 AND $votes^*(n)$ >= 1-$\theta[i]$ OR $y(n)$== 1 AND $votes^*(n)$ <= $\theta[i]$
           then
15             Mark instance n as noisy
16     cv_error[i] $\leftarrow$ Calculate error by 3 fold cross-validation
17     if $cv\_error[i]) < min\_error$ then
18         $\theta^* \leftarrow \theta$[i]
19         noise_set $\leftarrow$ save detected noisy set using current cleansing parameter
20 Cleanse dataset using aggregated votes ($votes^*$) of the optimum sampling rate and
   optimum cleansing parameter ($\theta^*$)
```

Dataset	**Instances**	**Test**	**Attrib.**	**Classes**
Australian	690	230	14	2
Breast W.	699	233	9	2
Diabetes	768	256	8	2
German	1000	333	20	2
Heart	270	92	13	2
Ionosphere	351	117	34	2
Liver	345	115	6	2
Sonar	208	699	60	2
Twonorm	300	5000	20	2
Votes	435	145	16	2

Table 5.1: Characteristics of the classification problems and testing method

Once the instances are marked as noisy they can be dealt with in two ways: either the class label is switched to the majority class given by *bag** or the instance that is marked as noise is removed from the training set.

5.1 Experimental Results

To assess the effectiveness of this noise detection procedure we have conducted an analysis on 10 datasets from the UCI repository [150]. The characteristics of the datasets are shown in Table 5.1. In all the experiments, unpruned CART trees are used [64].

In all datasets with the exception of *Twonorm*, which is a synthetic problem, the data are randomly divided into training and test sets with sizes equal to 2/3 and 1/3 of the data respectively. For *Twonorm*, 300 examples are used for training and 2000 for testing. In all cases, stratified sampling was used. The reported results are averages over 50 independent runs. The following testing protocol is followed for each dataset and run:

1. First, class noise is injected in the training set by randomly switching the class label of a random subset of the training instances. Different noise rates are considered: 0% (no injected noise), 10% and 20%. This type of label noise is know as completely at random noise (NCAR) [15].

2. For each noise level, six bagging ensembles composed of 500 trees are trained using the following bootstrap sampling ratios: 10%, 20%, 40%, 60%, 80% and 100% (standard bagging). The out-of-bag error is computed in each of these sets. The ensemble with the best out-of-bag accuracy (*bag**) is kept for the next step.

3. The bag^* ensemble is used to compute the percentage of correct votes for each of the training instances using out-of-bag. That is, for each instance, the votes are tallied using only those trees whose training set does not include that instance. Then, the percentage of correct predictions (votes) of the individual ensemble members is computed.

4. To detect noisy instances, if less than θ% of the classifiers for a particular instance agree on the class label prediction then the instance is considered as noise and will be either cleansed (i.e. its class label corrected) or removed. The following values of θ are tested: 0 (no cleansing), 0.1, 0.2, 0.3, 0.4 and 0.5. The optimum value of θ^* is selected by 3-fold cross-validation within the training set.

5. Training instances with a percentage of votes for the correct class below θ^* are identified as noise.

6. Instances marked as noise are either corrected or removed from the training set. Finally a standard bagging classifier composed of 500 decision trees is trained in each data set. For reference a bagging ensemble is trained on the full original training set without any noise cleansing process.

7. The performance of the resulting ensembles is estimated on the test set.

In this chapter we have used the filtering strategy described in [130] as a reference for comparison. In [130], an ensemble of three classifiers (an univariate decision tree, a k-nearest neighbor and a linear machine) is used for noise detection. In that paper majority and consensus filtering are used to mark noisy instances. Majority filtering had a better overall performance than consensus filtering.

To gain insight into how the proposed method works we have plotted the percentage of removed instances as a function of the threshold θ. Figures 5.1 and 5.2 show the percentage of instances marked as noisy (dashed purple line) as a function of the detection threshold, θ. In addition, the plots show the percentage of instances marked as noisy that are actually part of the injected noise (solid black line). In these plots the blue asterisk (with dashed lines) shows the percentage of instances that are marked as noisy for the selected threshold θ^*. For this same threshold the red circle (with dotted lines) shows the percentage of instances marked as noise that correspond to injected noise. In Figure 5.1 the results for *Heart* (first row of plots) and *Diabetes* (second row) are shown. In Figure 5.2, *Sonar* (first row) and *Votes* (second row) datasets are shown. In both figures, the left and right plots correspond to 10%, 20% injected noise, respectively.

The extreme values of θ correspond to the cases in which no instances ($\theta = 0$) and all instances ($\theta = 1$) are marked as noise respectively. For $\theta = 1$ the percentage of instances marked as noise that correspond to injected noise is 10% for the left plots and 20% for the plots on the right. From these plots we can have a rough estimate of the best choice of θ. When both lines coincide, all marked instances correspond to injected noise. From these plots we observe that choosing a small value of θ marks only a small percentage of instances as noise. However, the marked instances as noise are very likely to be injected noise. If we increase the value of θ the lines start to diverge and the percentage of instances that are detected increases. The objective is to select a value of θ that detects most noisy instances without removing valid ones. In general this value should be close to the point where the two lines start to diverge. From these plots, we conclude that in the investigated problems the optimal choice for θ for 10% injected noise is between 0.2 and 0.3. For 20% of injected noise the optimal choice of θ is higher, between 0.3 and 0.4. Another interesting aspect is that the performance of this noise detection procedure depends on the problem. For *Sonar* (Fig. 5.2 top row) the proposed method does not manage to detect adequately all the injected noises. By contrast, in *Votes* (Fig. 5.2 bottom row) most noisy instances are correctly identified and few noiseless instances are marked as noise.

Table 5.2 shows the average test errors for each method and dataset. For each dataset the errors for the cases of no injected noise, 10% and 20% noise are reported. The first column displays the test error of bagging trained on the original training set without any cleansing. The second column shows the average test error for bagging when trained on the cleansed dataset (i.e. in which the label of instances marked as noise is modified to the purportedly correct class label). The third column shows the test error when bagging trained on the dataset without the instances marked as noisy. In the last two columns, the test errors of standard bagging on data sets that their noises are marked using $\theta^* = 0.5$ (majority filtering as proposed by Brodley [130]) are shown. Columns for and five show the test error of bagging on datasets after cleansing and eliminating marked as noisy instances, respectively. The best result for each dataset and noise level is highlighted in boldface.

From the results presented in Table 5.2 one concludes that the method proposed in this chapter is more effective when more noise is injected in the training sets. For the noiseless case, the proposed method using cleansing outperforms bagging only in *Votes* and draws in other two problems (*Twonorm* and *Brest*). The method based on noise elimination performs similarly to the previous one. As more noise is injected, the proposed method outperforms bagging in seven datasets. In two datasets, (*Liver* and *Twonorm*), the accuracy of the proposed method is similar and in *Sonar* it is inferior to bagging. In several datasets the improvement is significant. For example in *Breast* our method achieves 4.9% error with 20% injected noise. Bagging trained on the original dataset achieves an error that is more than double this value (10.8%). The differences

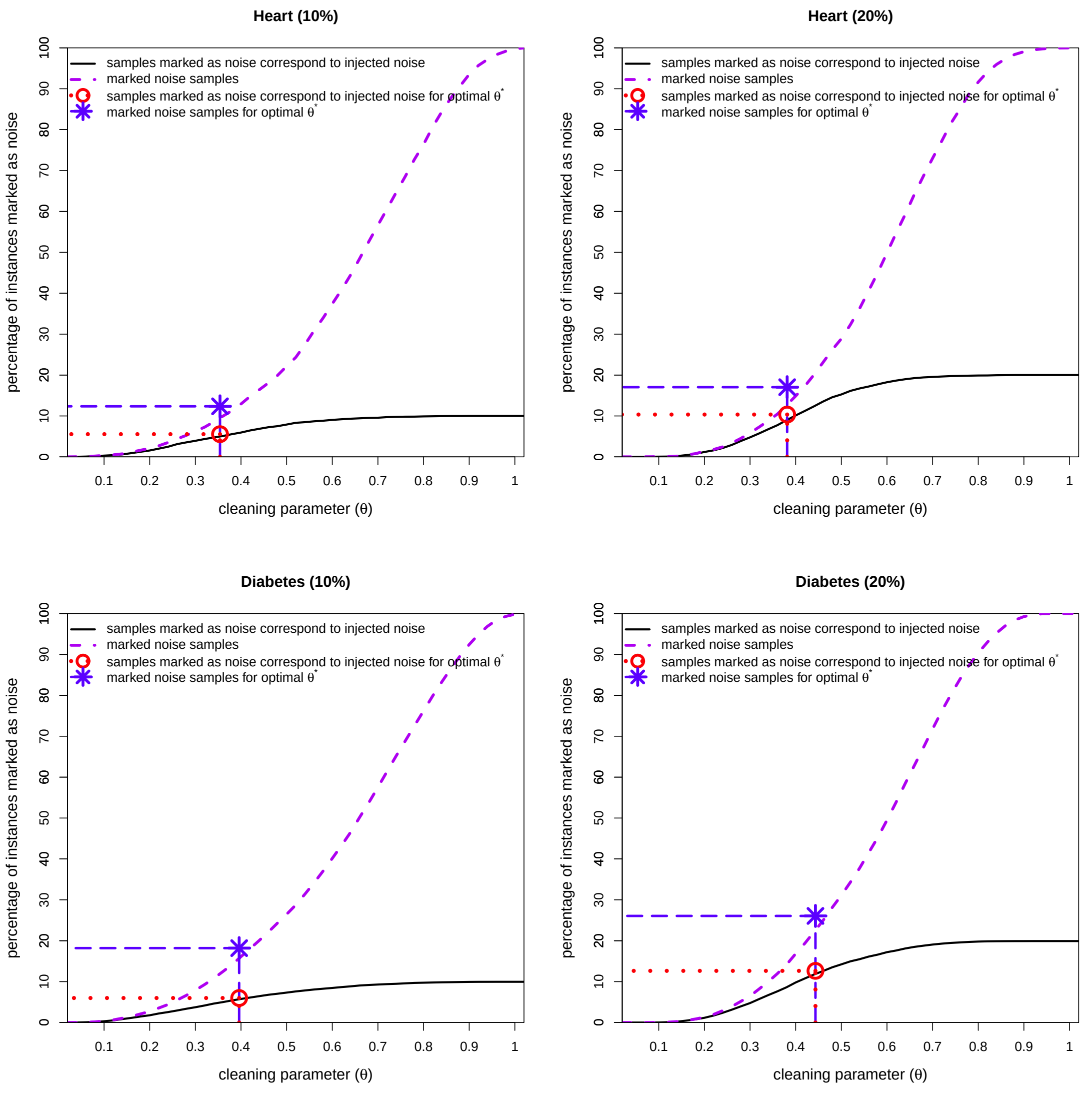

FIGURE 5.1: The Percentage of instances in training set, detected as noise and the percentage of correct detected noise instances in training set, correspond to each cleansing parameter, for *heart* and *pima* datasets with 10 and 20% noise. first row: *heart* 10 and 20% noise respectively. second row: *pima* 10 and 20% noise respectively.

between no cleansing and eliminating marked as noisy instances are small. In all cases the proposed method outperforms majority filtering.

In Table 5.3, the average values for the selected sampling rate sr^* (third column), threshold θ^* (fourth column), the percentage of marked instances (fifth column) and the percentage of injected noise detected (sixth column) are given. Note that the percentage of injected noise detected is relative to the total number of instances. In consequence, it cannot be above the actual percentage of noise injected.

As more noise is injected in the training set, the method marks a higher percentage of

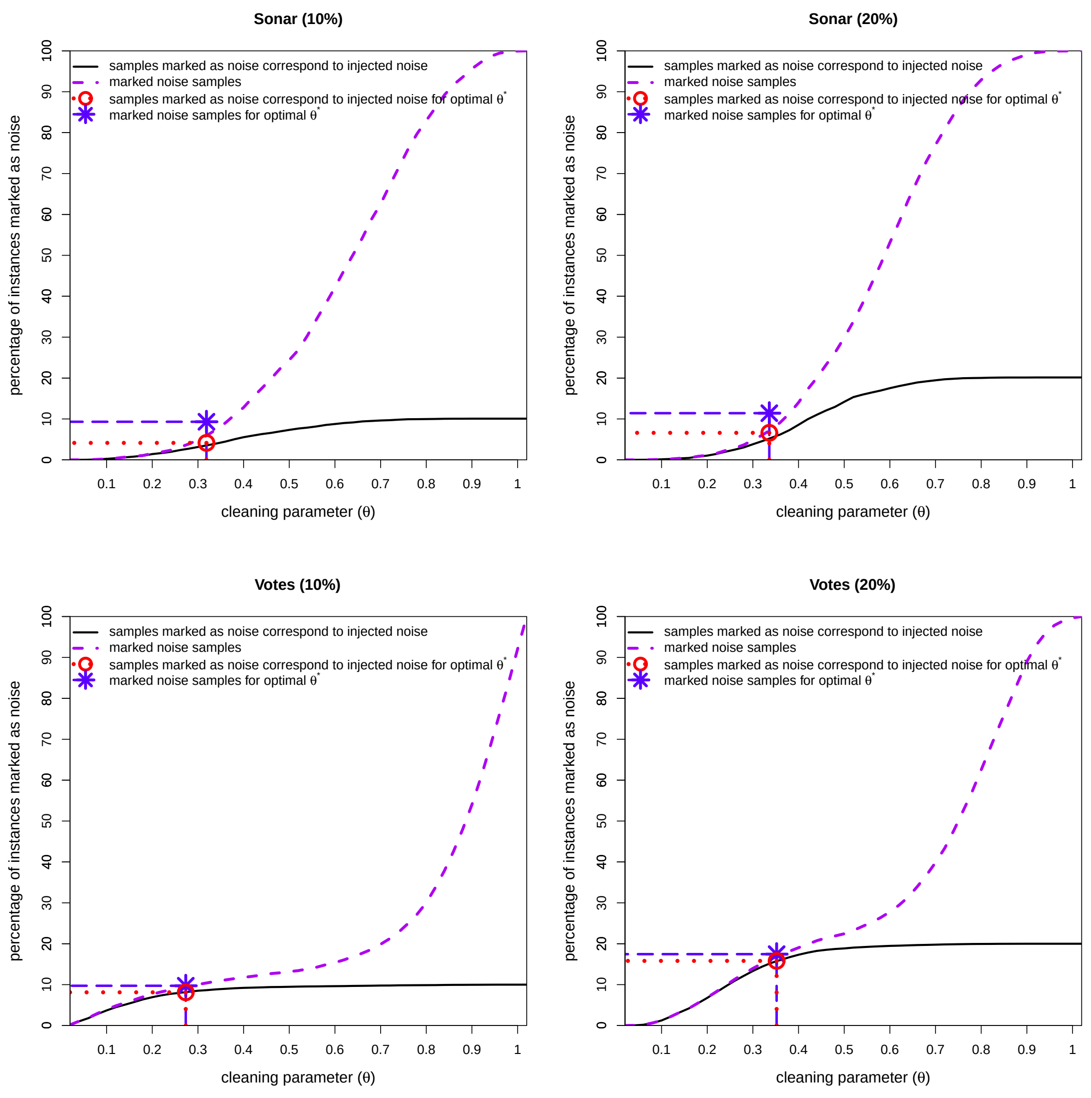

FIGURE 5.2: The Percentage of instances in training set, detected as noise and the percentage of correct detected noise instances in training set, correspond to each cleansing parameter, for *sonar* and *vote* datasets with 10 and 20% noise. first row: *sonar* 10 and 20% noise respectively. second row: *vote* 10 and 20% noise respectively.

instances. Simultaneously, the value estimated for the optimal θ^* also increases. This indicates that the proposed method adapts to the amount of noise present in the dataset. Interestingly, in some datasets the method manages to detect a rather high percentage of the injected noise (*Australian*, *Breast* and *Votes*) without removing a significant number of noiseless instances. For these datasets the improvement in accuracy is apparent (see Table 5.2). In addition, for many datasets, the percentage of instances labelled as noise, when no noise is injected is quite low, suggesting that the datasets contain little intrinsic noise. A clear case for this is *Twonorm*. Being a synthetic dataset, we know that it is a noiseless dataset. For this dataset, when no noise is injected, the proposed method only

TABLE 5.2: Bagging on original datasets and on datasets after cleansing and removing detected noise instances

Dataset	Noise (in %)	Original Dataset	Cleaned Dataset	Filtered Dataset	Cleaned Dataset($\theta = 0.5$)	Filtered Dataset($\theta = 0.5$)
Australian	0	13.2±2.2*	13.4±2.0	13.4±2.1	13.8±2.0	14.0±2.1
	10	15.0±2.2	**13.9±2.1***	**14.0±2.0**	14.6±2.1	14.5±2.3
	20	18.1±2.8	**15.1±1.8***	**15.2±2.2**	**16.9±3.2**	**16.2±2.6**
Breast W.	0	3.9±1.2*	4.1±1.1	4.0±1.2	4.4±1.4	4.4±1.2
	10	6.1±1.5	**4.5±1.6***	**4.5±1.6***	**5.4±1.7**	**4.8±1.5**
	20	9.8±2.2	**5.5±2.3**	**5.3±2.2***	**7.7±1.9**	**6.3±1.7**
Diabetes	0	24.2±2.4*	24.4±2.2	24.3±2.3	24.5±2.6	24.6±2.5
	10	26.1±2.1	**24.6±2.5***	**24.6±2.2***	**25.5±2.7**	**25.0±2.7**
	20	28.8±2.7	**25.2±2.1***	**25.3±2.1**	**26.4±2.1**	**26.6±2.4**
German	0	24.4±1.9*	25.4±1.8	25.1±1.6	26.5±1.7	25.3±1.8
	10	26.0±1.8	26.0±2.0	25.8±2.0*	27.0±1.9	26.0±2.1
	20	28.0±2.4	**27.2±1.9**	**26.9±1.9***	28.0±1.8	**27.3±2.2**
Heart	0	19.9±3.4*	20.8±3.5	20.6±3.5	21.4±3.8	22.0±3.7
	10	22.7±4.3	**21.4±4.0**	**21.1±3.9***	23.1±4.3	23.2±4.5
	20	25.1±3.4	**22.2±4.0***	**22.9±3.6**	25.2±4.9	25.3±4.7
Ionosphere	0	7.7±2.8*	8.0±2.8	7.9±2.8	9.6±2.8	9.9±3.0
	10	9.2±3.1	8.7±3.4*	8.9±3.3	10.6±3.4	9.8±3.2
	20	12.9±3.9	**10.5±3.7**	**10.5±3.8**	12.3±3.8	**10.3±3.9***
Liver	0	29.4±4.5*	30.6±3.9	30.1±3.9	33.6±3.8	32.6±4.1
	10	31.3±4.7*	33.6±3.4	33.3±3.7	34.3±4.0	34.5±3.4
	20	35.5±5.0	35.5±4.9	35.4±4.4*	36.9±4.7	36.0±4.8
Sonar	0	21.7±5.1*	23.1±5.1	23.3±5.4	26.3±4.9	26.3±4.4
	10	23.5±5.7*	24.6±5.9	24.4±5.4	26.9±4.8	26.3±5.1
	20	27.7±5.3*	29.3±6.2	28.5±6.2	31.5±6.9	31.0±6.2
Twonorm	0	6.3±1.6*	6.3±1.6*	6.4±1.6	11.4±3.9	10.0±3.7
	10	7.3±1.4	7.4±1.5	7.3±1.5*	10.6±3.4	10.8±3.9
	20	9.8±2.1	9.7±2.2	9.6±2.1*	11.2±3.1	12.9±4.1
Votes	0	5.1±1.8	**4.5±1.8***	**4.5±1.8***	5.1±1.7	4.8±1.5
	10	7.8±2.8	**5.8±2.6**	**5.8±2.5**	**6.6±2.3**	**5.5±2.4***
	20	12.1±3.5	**6.8±3.2***	**6.9±3.1**	11.0±3.8	**7.8±3.1**

detects 0.16% of instances. Another interesting case is *Diabetes*, generally considered as a noisy dataset. In this dataset, the percentage of removed instances that do not correspond to the injected noise ones, is very high. In the noiseless case the percentage of detected instances is 15.5%.

5.2 Conclusions

In this chapter we have proposed a noise detection procedure based on the level of agreement of the predictions given by individual ensemble members. To detect noisy instances in the training set, we have used subsampling, which, as shown in the previous chapter, is more resilient to noise than standard bagging. In this procedure a particular instance is marked as noisy if more than a specified fraction $(1-\theta)$ of the individual learners predictions are different from the actual class label. For $\theta = 0$, no instance is marked as noise. When $\theta = 0.5$, an instance is marked as noisy, when it is misclassified by more than half of the learners. For $\theta = 1$ only instances misclassified by all learners are marked as noise. Using $\theta = 0.5$ and $\theta = 1$ in our procedure corresponds to the majority filtering and consensus filtering methods proposed by Brodley in [130]. In our proposal, the optimal value of θ^* is determined using cross-validation. This optimal

Table 5.3: Average values of sr^*, θ^* and percentage of marked as noisy instances and percentage of detected noises that have been injected.

Dataset	Noise (in %)	sr^*	θ^*	%Marked instances	%Marked instances that correspond to injected noise
Australian	0	0.42	0.32	8.0	0
	10	0.23	0.32	12.4	7.0
	20	0.17	0.40	20.0	14.1
Breast W.	0	0.35	0.17	1.34	0
	10	0.16	0.27	9.6	8.4
	20	0.11	0.35	18	16.7
Diabetes	0	0.30	0.34	15.5	0
	10	0.21	0.38	18.1	6.0
	20	0.13	0.42	26.0	12.6
German	0	0.43	0.42	16.1	0
	10	0.32	0.41	22.8	6.5
	20	0.26	0.43	29.4	12.9
Heart	0	0.24	0.28	5.7	0
	10	0.17	0.33	12.3	5.5
	20	0.17	0.36	17.0	10.3
Ionosphere	0	0.37	0.20	2.4	0
	10	0.32	0.27	9.8	6.9
	20	0.22	0.34	16.5	13.5
Liver	0	0.35	0.34	12.0	0
	10	0.28	0.41	19.6	4.7
	20	0.23	0.42	26.4	9.8
Sonar	0	0.44	0.33	3.0	0
	10	0.55	0.27	9.3	4.1
	20	0.44	0.33	11.5	6.6
Twonorm	0	0.18	0.17	0.16	0
	10	0.16	0.25	3.9	3.7
	20	0.16	0.30	9.9	9.5
Votes	0	0.32	0.14	1.3	0
	10	0.20	0.23	9.7	8
	20	0.13	0.35	17.4	15.8

value depends the problem under consideration and the amount of noise that it contains. In datasets with higher test error, the selected value of θ^* is higher. As the amount of injected noise increases, the value of θ^* also increases. This dynamic behaviour of the proposed method in the presence of noise explains the improvement in accuracy with respect to the fixed filtering strategy of Brodley [130].

Chapter 6

Conclusions and Future Work

With increasing amounts of information, data analysis is nowadays a key tool in many scientific fields. The goal of machine learning is to identify regular patterns in the data that make generalization possible. However, these patterns could be masked or disturbed by noise and outliers. These irregularities can mislead the learning algorithms and limit or reduce the accuracy of the predictors induced from the data. There are two main strategies to address these issues. The first is to design robust learning algorithms that are resilient to noise and/or outliers. The second strategy is to detect noisy instances in a preprocessing step. By either correcting or eliminating these instances it is possible to improve the quality of the data that are used for induction.

In this thesis we have described different ensemble learning techniques especially designed to cope with the difficulties associated to learning from noisy data. Ensembles are multiple classifier systems that can provide a certain flexibility for robust decision making. In this thesis, we have investigated robustified versions of ensembles based on bootstrapping (bagging and random forest). In addition, we have proposed an ensemble-based procedure for noise detection.

The first idea is to use subsampling in bootstrap ensembles such as bagging and random forest to improve their robustness in the presence of noise. In bagging, the base learners in the ensemble are trained using bootstrap samples (with replacement) from the original training set. In the standard version of the algorithm, each bootstrap sample has the size as the original training set. When subsampling is used, the number of unique instances in each bootstrap sample is smaller than in the standard prescription. This can lead to an increase in the diversity of the base learners. Higher diversity can result in improvements in generalization capacity of the ensemble. Subsampling can also have other beneficial effects: When the sampling rate is below 69.3%, the percentage of unique instances in each bootstrap sample is on average below 50%. This means that each instance is present on average in less than half of the bootstrap samples used to train each of the base learners. In this regime, the predictions on a given instance in the training set is dominated by the labels of their neighbouring examples. This regularization effect limits the impact of this incorrectly labelled instances on the final decision.

To show the effectiveness of subsampling in bootstrap ensembles, extensive experiments on real-world and synthetic datasets have been carried out. The results show that bagging with bootstrap samples of sizes between 10% and 40% of the original training set are more robust to label noise than standard bagging. In random forest the improvement in robustness is observed only in some problems when high levels of noise are present. When the level of noise is low, standard random forest is the preferred choice. In any case, the optimal bootstrap samples size strongly depends on the problem and on the noise level. Nevertheless, this optimal size can be determined by using cross-validation.

A second proposal of this thesis is a method to deal with noise in a pre-processing step, i.e. a data-cleansing method. The idea is to use bagging with subsampling to detect noisy instances. An instance is marked as noise if a percentage (greater than 50%) of the base learners misclassify it. The value of the optimum threshold is selected using 3-fold cross-validation. We have carried out extensive experiments to demonstrate the effectiveness of the proposed cleansing method.

6.1 Future Work

As we discussed in this thesis, ensemble learning can be an effective strategy to overcome the limitations that arise from the lack of expressive capacity of a single learner. However, complementarity among base learners is an prerequisite to build a powerful ensemble. One of our open questions is to find out how strong classifiers such as SVMs or deep belief networks (DBNs) can be modified to introduce complementarity in their decisions. One of our ideas is that subsampling can increase diversity in learners and can improve chance of their being complementary.

Another line of work is to design robust boosting ensembles. Boosting is an ensemble method that improves the accuracy of individual learners in an iterative manner. In each iteration, the weight of the examples misclassified by the previous learner are increased. This process leads to the concentration of the learners on the examples that are difficult to classify. Nevertheless, in addition to the difficult examples, the noisy examples or outliers also are assigned higher weights and therefore overfitting may occur. One possible path to explore is to use subsampling together with boosting. By using subsampling a smaller amount of examples are selected in the training sets of the base learners. Therefore the risk of having excessive noise examples in the training sets decreases.

Appendix A

Lower Sampling Rate Error Tables

In Tables A.1, A.2 and A.3, the average generalization error (with the standard deviation after the $\pm$ sign) is shown for bagging ensembles for different sampling ratios and injected noise levels. The results are split into three tables. For each row the lowest error is highlighted with an asterisk (*). In addition, for each noise level and dataset (that is for each row) the results that are significantly better (using a paired t-test with p-value=0.05) than standard bagging (column 100%) are highlighted in boldface. Underlined those that are significantly worse than standard bagging. In the same manner, the detailed results for random forest are shown in Tables A.4, A.5 and A.6.

Table A.1: Average generalization error for bagging and subbagging (I)

	Noise	Bootstrap sampling ratio					
Dataset	(in %)	10%	20%	40%	60%	80%	100%
Australian	0	13.5±1.9	**13.0±1.9**	**12.8±1.6***	**12.9±2.0**	13.5±1.9	13.7±2.2
	5	13.8±1.9	13.3±1.9	13.4±2.0	**12.8±2.0***	13.9±2.0	13.7±2.0
	10	**13.6±1.9**	**13.5±2.2***	**13.5±1.9***	14.1±1.7	14.2±2.0	14.6±2.2
	20	**13.9±2.1***	**14.7±1.8**	**15.8±2.3**	**16.8±2.6**	**17.4±2.5**	18.2±2.6
Balance	0	**10.2±0.9***	**11.4±1.7**	**13.8±1.4**	**16.0±1.9**	17.4±1.8	18.3±2.1
	5	**11.0±1.1***	**11.9±1.3**	**14.7±1.2**	**17.1±1.6**	18.3±1.4	19.8±2.0
	10	**11.2± 1.2***	**12.9±1.5**	**16.3±1.6**	**17.9±1.4**	19.2±1.9	20.3±2.1
	20	**12.8±1.1***	**14.9 ±1.6**	**18.5±1.8**	**20.9 ±1.8**	23.6±3.0	25.1±3.2
Breast W.	0	4.1±1.3	**3.7±1.1***	**3.8±1.0**	**3.9±1.1**	4.1±1.0	4.3±1.1
	5	**3.7±1.0**	**3.5±1.0***	**3.7±1.0**	**4.2±1.3**	4.7±1.2	5.0±1.6
	10	**3.5±1.2***	**3.6±1.0**	**3.7±1.2**	**4.4±1.2**	**5.3±1.5**	6.1±1.7
	20	**3.5±1.0***	**4.2±1.3**	**5.1±1.4**	**6.4±1.7**	**8.0±2.1**	9.2±2.2
Diabetes	0	**23.7±2.2***	**23.8±2.1**	24.1±2.6	**23.8±1.9**	24.6±2.3	24.4±2.3
	5	**23.7±2.4***	**24.2±1.9**	**24.2±2.3**	**24.2±2.1**	24.8±2.2	25.2±2.3
	10	**23.4±2.2***	**24.2±2.4**	**24.6±2.1**	**25.1±2.5**	25.8±2.0	25.9±2.3
	20	**24.5±2.3***	**24.9±2.3**	**26.8±2.4**	**27.1±2.9**	**27.7±3.0**	28.5±2.5
German	0	25.0±1.8	24.2±1.6	23.9±2.0*	23.9±1.8*	24.0±1.8	24.3±1.8
	5	25.4±1.6	**24.1±1.8***	**24.3±1.9**	**24.2±1.7**	**24.5±1.9**	25.1±1.8
	10	25.5±1.6	**24.6±1.8***	**24.6±1.8***	25.4±2.1	25.8±2.0	26.0±2.3
	20	**26.5±1.8**	**25.8±1.9***	**26.4±2.1**	**27.6±2.5**	28.0±2.5	28.5±2.1
Heart	0	**17.0±3.5***	**17.1±3.7**	**18.7±4.4**	19.0±3.7	19.3±3.8	19.9±3.4
	5	**17.4±4.0***	**18.6±4.2**	**18.8±4.0**	**19.3±4.3**	**20.4±3.7**	21.8±4.6
	10	**18.1±3.9***	**19.1±4.6**	**19.5±3.7**	21.5±4.0	**21.2±4.4**	22.3±4.1
	20	**20.3±3.8***	**22.1±4.7**	**22.4±4.7**	**24.3±4.4**	**24.3±5.1**	25.9±4.3
Hepatitis	0	**21.2±0.4**	**19.7±2.0***	**19.8±1.9**	**20.7±2.7**	21.5±4.0	22.2±3.3
	5	**20.1±2.5**	**19.8±2.0***	**20.8±2.6**	**21.3±2.9**	**22.2±3.6**	23.3±4.1
	10	**20.0±2.6**	**19.8±1.8***	**21.1±3.1**	**22.4±3.7**	**23.5±4.4**	25.0±4.8
	20	**20.2±3.6***	**20.2±2.6***	**24.9±4.2**	**25.8±4.4**	**27.9±5.6**	31.4±5.2
Horse-Colic	0	25.2±2.1	19.9±0.9	**16.1±0.4***	**16.1±0.5***	17.2±0.7	16.4±0.9
	5	25.8±2.4	21.8±2.2	17.8±2.1	17.1±2.1	17.1±1.8	17.0±2.5*
	10	26.4±2.5	23.3±2.9	19.4±2.8	18.5±2.9*	18.5±3.2*	18.6±2.8
	20	27.5±3.8	25.8±3.9	22.4±3.5	22.4±3.8	21.3±3.6*	21.9±3.8
Ionosphere	0	9.6±2.8	**6.8±1.9***	7.5±2.2	**7.2±2.1**	7.7±2.5	8.0±2.4
	5	9.1±2.5	**7.2±2.3***	**7.6±2.3**	8.6±2.9	8.5±2.6	8.4±2.5
	10	9.6±2.2	**7.4±2.3***	**7.9±2.6**	**8.1±2.7**	9.1±2.7	9.6±2.8
	20	**10.3±3.1**	**9.8±3.2***	**10.1±3.0**	**11.2±3.5**	12.7±3.4	13.0±3.7
Iris	0	12.3±4.6	**4.5±2.6***	5.2±2.7	5.3±2.4	5.2±2.8	5.3±2.4
	5	8.6±6.0	**5.1±3.2***	**5.3±2.8**	**5.3±2.5**	7.4±3.3	7.9±3.7
	10	**4.6±6.8***	**4.7±3.4**	**5.3±2.6**	**6.4±3.4**	**8.9±4.0**	10.6±5.0
	20	**5.0±3.1***	**6.1±3.5**	**7.0±4.6**	**10.8±5.1**	**13.4±5.4**	16.0±6.2

Table A.2: Average generalization error for bagging and subbagging (II)

	Noise	Bootstrap sampling ratio					
Dataset	(in %)	10%	20%	40%	60%	80%	100%
Labor	0	16.2±8.8	14.7±8.5	13.3±9.8	11.8±7.6*	13.6±5.1	12.0±6.4
	5	16.0±10.2	13.3±8.8	14.2±8.8	12.4±8.5	11.8±7.0	10.4±5.6*
	10	14.2±7.2	11.8±6.3*	17.6±14.6	15.8±6.9	17.8±10.3	16.0±9.4
	20	18.9±8.3	17.3±9.0*	17.8±8.3	18.4±9.6	22.9±10.4	20.0±10.0
Liver	0	**28.6±4.0**	**27.4±3.4***	**27.5±3.7**	**27.8±3.4**	**28.7±3.6**	29.9±3.6
	5	**29.0±3.5**	**28.5±3.8***	**28.5±4.3***	**29.5±4.3**	30.1±3.9	30.7±3.9
	10	**29.9±4.0**	**29.1±3.4**	**29.0±4.0***	30.4±3.8	31.0±3.3	31.4±3.7
	20	**32.4±4.3**	**31.9±4.5***	**32.3±4.4**	**32.9±4.3**	34.3±4.1	34.8±4.3
Lung Cancer	0	**42.0±8.9***	53.5±10.2	**42.0±11.2***	45.5±11.1	45.5±11.8	45.0±12.9
	5	**44.0±8.5***	53.0±10.8	49.0±11.5	47.5±12.2	46.5±11.5	49.5±12.7
	10	**42.0±9.1***	50.5±10.4	47.0±11.7	45.5±12.0	49.0±11.9	49.0±11.8
	20	**49.5±9.0**	**53.5±11.1**	**48.0±12.0**	43.5±12.8*	55.0±12.2	54.0±13.7
Magic	0	13.0±0.4	12.5±0.4	12.3±0.3	12.3±0.4	12.2±0.4*	12.2±0.4*
	5	13.1±0.4	12.7±0.4	12.5±0.4	**12.3±0.3***	12.4±0.4	12.5±0.3
	10	13.0±0.4	12.8±0.3	**12.7±0.4***	**12.7±0.4***	12.9±0.3	12.9±0.4
	20	**13.4±0.4**	**13.2±0.4***	**13.3±0.4**	**13.6±0.4**	**13.8±0.4**	14.2±0.4
new-thyroid	0	5.4±3.0	6.4±2.9	6.9±3.2	5.2±3.2*	5.6±3.1	5.7±2.7
	5	6.5±2.5	**4.5±1.8***	**5.3±2.7**	6.7±3.0	**5.0±2.0**	8.3±2.8
	10	6.3±3.8	**5.4±2.8**	5.2±2.5	**5.0±2.3***	6.7±3.5	7.9±3.6
	20	**5.2±3.1**	**5.1±2.7***	**5.8±2.5**	10.1±4.4	11.0±3.6	10.6±5.4
Ringnorm	0	12.1±1.1	**8.1±1.1**	**7.6±1.3***	**8.2±1.8**	8.6±1.7	8.8±1.9
	5	11.4±1.7	**7.9±1.3**	**7.4±1.3***	**8.0±1.7**	**8.4±1.6**	9.1±1.8
	10	11.3±1.9	**7.8±1.5**	**7.5±1.5***	**8.4±1.6**	**8.7±1.6**	9.5±1.9
	20	11.5±2.1	**8.6±1.5***	**9.1±1.9**	**9.7±1.9**	**10.1±1.7**	11.2±1.9
Segment	0	3.4±1.4	3.0±1.2	2.6± 1.7	2.3± 1.5	2.2± 1.0	2.1± 0.9 *
	5	**3.2±1.5**	**3.1±1.3***	**3.4±1.9**	3.8±1.9	3.6±0.7	3.8±1.1
	10	**3.2±1.2**	**3.1±1.3***	**4.2±1.1**	4.6±1.7	5.2±1.2	6.6±1.3
	20	**3.5±2.1**	**3.2±1.5***	**4.0±1.6**	5.7±1.5	7.2±1.4	7.4±1.5
Sonar	0	22.5±4.4	23.6±4.3	23.0±4.6	21.5±4.9 *	22.0±4.7	21.0±4.9
	5	24.7±4.2	24.0±5.3	23.2±4.2	21.3±4.5*	22.4±4.6	22.7±5.3
	10	24.8±4.6	22.7±5.1	23.9±5.2	21.7±4.8*	24.1±5.4	21.8±5.0
	20	**25.6±5.1**	**25.7±5.5**	26.8±5.8	25.2±5.4*	26.3±5.9	26.2±6.0
Threenorm	0	18.7±1.2	**17.6±1.3***	**17.7±1.4**	**18.0±1.7**	18.8±1.6	18.9±1.8
	5	19.5±1.3	**18.1±1.6***	**18.4±1.4**	19.2±1.8	19.0±1.6	19.1±1.5
	10	**19.1±1.5**	**18.6±1.3***	**19.1±1.5**	**19.8±1.5**	**19.3±1.8**	21.1±1.7
	20	**21.7±1.9**	**21.5±1.9**	**21.4±1.8***	22.3±1.9	22.9±1.9	22.8±2.0
Tic-tac-toe	0	15.4±2.0	5.1±2.0	2.2±0.9	2.0±0.9	1.9±0.8*	1.9±0.7 *
	5	16.9±2.5	7.6±2.3	3.5±1.3	**3.1±1.2***	3.3±1.2	3.6±1.4
	10	18.0±2.3	10.4±2.1	5.4±1.7	**5.1±1.6***	5.4±1.6	5.6±1.6
	20	20.8±2.6	16.3±2.7	13.0±2.4	12.2±2.1*	12.2±2.1*	12.7±2.7

Table A.3: Average generalization error for bagging and subbagging (III)

	Noise	Bootstrap sampling ratio					
Dataset	(in %)	10%	20%	40%	60%	80%	100%
Twonorm	0	**4.9±1.1**	**4.6±0.8***	**5.1±1.0**	**5.2±0.7**	6.3±1.5	6.6±1.4
	5	**4.4±0.7***	**5.1±1.1**	5.5±1.1	6.2±1.9	6.2±1.0	7.1±2.0
	10	**5.0±0.8**	**4.8±0.6***	**5.9±0.7**	6.6±1.2	6.8±1.0	7.3±1.3
	20	**6.0±0.5***	**7.2±1.8**	7.3±1.8	7.8±1.1	8.4±0.6	9.1±1.7
Vehicle	0	26.0±2.5	25.5±2.3	25.5±2.1	25.2±2.0	25.7±1.0	25.1±1.1
	5	30.1±2.4	28.2±2.2	27.6±2.0	27.4±1.3	27.2±1.6	26.5±1.5
	10	31.8±2.3	**28.4±2.2**	27.9±2.0	27.5±1.8	28.1±1.7	28.5±1.2
	20	32.3±2.6	29.9±2.7	28.7±2.5	29.0±2.2	29.5±2.0	29.8±1.7
Votes	0	**4.4±1.6**	**4.0±1.6***	**4.0±1.5***	**4.5±1.6**	4.7±1.9	5.0±1.5
	5	**4.4±1.4**	**4.3±1.5***	**4.4±1.8**	**4.5±1.5**	**5.1±1.7**	5.9±2.1
	10	**4.5±1.5***	**4.7±1.5**	**4.8±1.8**	**5.7±1.8**	**6.7±2.3**	7.3±2.0
	20	**4.8±1.7***	**5.8±1.9**	**7.8±2.9**	**9.5±2.9**	**11.2±3.1**	12.9±3.7
Waveform	0	**17.5±2.5***	**17.9±2.4**	**17.8±2.0**	18.8±1.4	19.0±1.0	20.1±1.2
	5	**17.0±2.6***	**17.3±2.0**	**17.7±1.9**	19.1±1.6	19.3±1.6	19.5±1.5
	10	**17.5±2.2***	**17.8±2.2**	19.5±1.6	20.8±1.7	21.2±1.7	21.9±1.8
	20	**18.1± 2.7***	**19.5±2.6**	19.3±2.0	22.0±1.5	22.2±1.8	22.8±1.7
Wine	0	7.6±4.5	**4.5± 2.5**	5.2±4.4	**4.4±2.4**	3.9±3.3*	5.1±3.1
	5	6.2±3.9	**3.4±2.4***	**5.2±2.9**	**4.9±3.0**	**4.7±2.9**	6.1±3.6
	10	**5.9±3.3**	**3.5±2.2***	**4.0±2.3**	**4.3±3.4**	5.8±4.2	7.3±4.1
	20	**5.6±3.4**	**4.2±2.4***	**5.9±3.9**	**7.0±3.1**	8.7±3.4	10.5±4.3

Table A.4: Random forest generalization error (I)

	Noise	Bootstrap sampling ratio					
Dataset	(in %)	10%	20%	40%	60%	80%	100%
Australian	0	13.3±1.3	6.5±0.6	**4.9±0.5**	**4.7±0.5***	**4.9±0.6**	5.1±0.8
	5	14.4±4.7	7.8±2.2	5.7±1.4	5.4±1.1	5.2±0.7*	5.4±0.8
	10	16.6±5.2	9.4±3.5	6.5±1.8	6.0±1.3	**5.7±1.0***	6.1±1.1
	20	21.5±6.5	13.5±4.8	9.6±2.5	8.9±2.0	8.7±2.0	8.3±1.5
Balance	0	16.9±2.0	15.4±1.8	14.5±1.9	**14.3±1.6**	**14.0±1.5***	15.0±1.8
	5	16.1±2.2	**15.2±2.0**	**14.6±2.0***	**14.8±1.8**	**15.4±1.9**	16.1±2.1
	10	**15.2±2.5**	**14.6±1.9***	**15.7±2.2**	**16.2±2.1**	17.1±2.2	17.6±2.4
	20	**15.2±2.2***	**16.0±2.1**	**17.5±2.6**	**19.1±2.4**	19.8±2.3	20.3±2.9
Breast W.	0	3.5±1.0	3.3±1.0	3.2±1.0	3.1±1.0	3.0±0.9*	3.0±1.0*
	5	3.4±1.1	3.3±0.9	3.2±1.0	3.2±1.0	3.2±0.9	3.1±1.0*
	10	**3.2±1.1***	**3.4±1.1**	3.7±1.3	3.8±1.2	3.8±1.1	3.8±1.1
	20	**3.7±1.2***	**4.2±1.4**	**5.0±1.5**	6.1±1.9	**5.8±1.5**	6.6±1.7
Diabetes	0	25.8±2.3	24.7±2.7	24.4±2.3	24.3±2.3	24.2±2.2	23.9±2.2*
	5	25.0±2.7	24.7±2.7	24.6±2.3	24.6±2.2	24.2±2.3*	24.4±2.2
	10	25.2±2.2	24.6±2.5*	24.7±2.3	25.1±2.1	25.0±2.3	24.7±2.1
	20	**25.4±2.5**	**25.3±2.5***	**26.5±2.5**	27.2±3.0	27.0±2.7	27.4±2.9
German	0	29.6±0.4	28.4±0.7	27.0±1.0	25.8±1.3	25.3±1.4	24.9±1.3*
	5	29.2±0.7	27.9±1.0	26.7±1.1	26.0±1.2	25.3±1.4	24.9±1.5*
	10	28.6±0.9	27.5±1.3	25.8±1.3	25.6±1.6	25.5±1.7*	25.5±1.5*
	20	28.0±1.3	27.2±1.6	26.6±1.6	26.4±1.6*	27.0±2.2	26.8±1.9
Heart	0	20.9±3.4	19.6±3.9	17.5±3.1	17.4±3.4	17.2±3.5*	17.5±3.2
	5	19.8±3.1	18.2±3.3	17.6±3.3*	18.7±3.7	18.4±3.3	17.7±3.4
	10	19.5±3.7	18.8±3.7	18.5±4.2*	19.1±3.4	19.8±3.7	18.9±3.8
	20	**19.7±4.3***	**20.3±4.7**	**21.5±3.5**	22.0±4.2	22.4±3.9	22.8±4.9
Hepatitis	0	20.5±1.1	17.9±2.6	14.9±3.0	13.7±3.4	13.1±3.3	12.7±3.6*
	5	19.6±2.2	15.7±3.1	13.9±3.3	13.0±3.3	13.0±3.7	12.5±3.7*
	10	17.7±3.5	15.7±3.7	13.9±3.8	13.9±3.6	13.2±3.4*	13.5±3.9
	20	16.3±4.2	15.5±4.0	15.8±4.2	**15.2±4.4***	16.1±4.4	16.5±3.8
Horse-Colic	0	30.2±1.7	27.6±1.6	26.5±1.8	26.5±1.9	26.2±1.8	25.3±1.8*
	5	31.0±3.1	27.6±2.7	26.3±2.2	25.8±3.0	25.8±2.9	24.8±2.9*
	10	31.2±3.4	28.3±3.6	27.1±3.0	25.7±3.6	25.8±3.3	25.6±3.3*
	20	31.2±4.1	29.8±3.8	28.1±3.6	27.4±4.3	27.2±3.9	26.5±3.7
Ionosphere	0	12.6±2.4	7.8±1.9	6.6±2.0	6.8±1.9	6.2±1.9	6.1±1.8*
	5	10.3±3.0	7.8±2.3	7.2±2.2	7.2±2.3	6.8±2.0*	7.1±2.3
	10	10.7±2.9	8.1±2.2	**7.4±2.3***	**7.5±2.3**	7.6±2.2	8.3±2.7
	20	11.1±3.1	**9.5±2.4***	**9.6±3.1**	**9.7±2.6**	10.8±3.2	10.6±2.7
Iris	0	4.4±2.5*	4.6±2.2	4.4±1.9*	4.7±2.5	5.0±2.6	4.8±2.4
	5	6.2±4.9	4.9±3.1*	5.5±2.8	5.0±2.7	5.3±2.9	5.4±2.9
	10	7.6±5.5	6.8±4.5	5.6±3.5*	5.7±2.8	5.7±3.0	6.5±3.9
	20	**8.2±4.8**	**7.8±5.0***	**8.9±5.0**	**8.9±5.3**	10.6±5.1	11.8±5.4

Table A.5: Random forest generalization error (II)

	Noise	Bootstrap sampling ratio					
Dataset	(in %)	10%	20%	40%	60%	80%	100%
Labor	0	12.0±3.7	12.7±3.2	11.7±2.8	11.1±2.9	9.0±2.8	8.9±2.2 *
	5	12.7±3.8	12.5± 3.3	13.5±3.6	12.5±4.8	12.4±3.8	11.0±3.3 *
	10	**12.7±4.2****	**12.8±4.5**	14.5±4.1	14.8±4.0	15.6±4.5	15.7±4.2
	20	**13.0±5.8****	**13.5±5.3**	15.5±5.1	15.7±4.8	16.4±4.7	16.4±4.5
Liver	0	36.8±2.0	33.5±2.2	29.7±3.0	28.1±2.9	27.5±3.2	27.1±3.2*
	5	35.1±3.2	32.7±3.0	29.9±3.2	29.2±3.5	28.8±3.6	28.5±4.0*
	10	33.6±2.9	31.4±3.9	30.8±4.2	30.4±3.6	30.3±3.7*	30.6±3.4
	20	33.9±3.8	33.2±4.2*	33.7±4.4	34.3±4.7	33.5±4.4	34.4±4.6
Lung Cancer	0	57.9±9.1	53.8±11.7	48.2±12.9	**43.0±13.1****	46.8±13.2	48.4±14.6
	5	60.7±7.4	55.8±11.7	49.2±13.0	49.3±12.3	47.6±13.7	47.4±12.6*
	10	61.8±9.3	55.9±11.7	54.2±12.9	50.2±15.7*	51.2±13.5	51.8±12.7
	20	61.8±10.9	58.7±12.1	55.4±11.5	54.7±13.0	**50.5±13.3****	54.8±14.2
Magic	0	14.4±0.4	13.6±0.4	12.9±0.3	12.6±0.4	12.4±0.3*	12.4±0.4*
	5	13.5±0.4	13.2±0.4	12.8±0.4	12.7±0.4	12.5±0.3*	12.5±0.3*
	10	13.3±0.4	13.0±0.4	12.9±0.4	12.8±0.4	12.7±0.4*	12.7±0.4*
	20	**13.6±0.4**	**13.5±0.4****	**13.5±0.4****	**13.6±0.4**	13.7±0.4	13.8±0.4
New-thyroid	0	8.4±2.3	7.3±2.5	5.1±2.0	3.3±1.8	3.0±1.0	4.4±1.2
	5	8.1±2.4	8.2±2.5	5.6±2.3	5.8±1.9	4.0±1.6	3.4±1.5
	10	8.2±2.8	5.9±2.1	3.3±2.4	4.8±1.8	4.3±1.7	3.2±1.7
	20	**6.1±2.5****	**6.2±3.0**	7.2±2.8	8.0±2.5	8.4±2.7	8.5±2.6
Ringnorm	0	13.2±1.4	6.5±0.7	4.9±0.5	**4.8±0.6****	**4.8±0.6****	5.0±0.7
	5	14.9±4.7	7.4±2.3	5.5±1.2	5.2±0.9*	5.2±0.9*	5.4±0.8
	10	16.7±5.2	9.3±3.1	6.2±1.4	6.1±1.2	6.0±1.3*	6.1±1.4
	20	21.4±5.7	14.3±4.8	9.7±2.4	8.6±2.3	8.5±1.7	8.2±1.8
Segment	0	5.9±0.9	4.4±0.9	3.5±0.5	2.9±0.6	2.7±0.7	2.6±0.6*
	5	5.9±0.8	4.5±0.9	3.7±0.7	3.1±0.8*	3.1±0.7*	3.2±0.8
	10	5.8±1.1	4.6±1.1	**3.4±0.6**	**3.0±0.7****	**3.4±0.7**	4.1±0.7
	20	5.7±0.7	**4.9±0.8**	**4.0±0.9****	**4.5±0.8**	**5.2±0.7**	6.1±0.7
Sonar	0	31.1±4.8	24.6±4.9	21.5±4.6	19.6±4.5	18.3±4.6*	18.6±4.4
	5	28.6±6.1	24.9±5.2	21.3±5.0	20.6±5.2	20.5±4.5	19.9±3.8*
	10	28.7±6.5	24.4±5.0	21.2±4.5	20.7±4.5	20.9±5.2	20.6±4.5*
	20	27.8±6.4	26.3±5.2	24.9±4.7	24.4±5.5	24.2±5.9	23.9±5.2*
Threenorm	0	18.2±0.9	16.9±0.9	16.0±0.9*	16.8±1.0	16.2±1.0	16.0±1.1*
	5	22.3±3.2	19.3±1.9	18.4±1.2	17.2±1.1	17.2±1.1	16.9±1.0*
	10	24.7±3.9	21.7±2.5	20.0±1.6	19.5±1.5	**18.6±1.6****	19.0±1.5
	20	30.6±4.5	26.3±3.2	23.4±2.0	22.9±2.2	22.3±2.0	21.6±1.7*
Tic-tac-toe	0	29.0±1.3	23.0±1.4	15.2±1.7	10.0±2.0	6.6±2.0	4.9±1.7*
	5	26.9±1.9	21.5±1.9	14.0±1.9	10.1±2.3	7.7±2.0	6.3±1.9*
	10	26.3±2.2	20.6±2.2	14.5±2.4	11.3±2.4	9.1±2.1	8.4±2.3*
	20	25.2±2.5	21.3±2.7	16.8±2.6	14.9±2.5	14.3±2.4	13.6±2.4*

Table A.6: Random forest generalization error (III)

	Noise	Bootstrap sampling ratio					
Dataset	(in %)	10%	20%	40%	60%	80%	100%
Twonorm	0	**3.3±0.3***	**3.3±0.3***	**3.3±0.3***	**3.4±0.3**	3.6±0.3	3.6±0.4
	5	4.5±1.3	3.9±0.8*	**3.9±0.5***	4.0±0.5	3.9±0.5*	4.0±0.4
	10	5.9±2.0	4.8±1.2	4.4±0.7*	4.6±0.9	4.4±0.6*	4.5±0.6
	20	9.5±4.4	7.4±2.7	6.3±1.4	6.0±1.2*	6.1±1.0	6.3±1.1
Vehicle	0	30.7±2.4	29.6±1.7	27.2±1.9	26.3±1.7	25.9±1.7*	26.1±1.6
	5	30.9±3.0	29.0±2.0	27.3±2.0	26.1±1.8	26.2±2.1	25.6±2.5*
	10	30.1±2.2	27.8±2.2	27.9±2.4	26.2±1.7	26.3±2.0	25.9±1.8*
	20	30.5±2.2	29.6±2.4	28.2±2.2	27.0±2.5	27.4±1.8	26.9±2.6*
Votes	0	5.3±1.7	4.4±1.5	3.9±1.4	3.6±1.3*	3.6±1.4*	3.6±1.3*
	5	5.4±1.7	4.4±1.5	4.0±1.4	3.9±1.4	3.7±1.5*	3.7±1.7*
	10	5.7±1.6	5.0±1.7	4.5±1.5	**4.1±1.5***	4.2±2.0	4.6±1.9
	20	6.3±2.2	**5.5±2.1***	**5.9±2.2**	6.2±2.3	6.5±2.6	6.7±2.5
Waveform	0	15.5±0.7	14.9±0.8	14.8±0.8	14.5±0.6*	14.6±0.6	14.6±0.6
	5	15.3±0.9	15.1±0.9	14.9±1.1	15.0±0.8	14.8±0.8*	14.8±0.6*
	10	15.1±0.6	14.8±0.5	14.8±0.8	14.9±0.8	14.6±0.9*	15.0±0.7
	20	14.9±1.0*	15.0±0.6	15.3±0.8	15.4±0.7	15.4±1.0	14.9±0.7*
Wine	0	3.0±1.8	3.1±1.9	2.5±1.7	2.3±1.6	2.1±1.7	1.9±1.6*
	5	4.8±3.3	3.6±2.7	2.7±2.0	2.9±2.2	2.8±2.1	2.4±2.0*
	10	5.8±4.3	4.1±3.0	3.7±2.6	3.4±2.6	3.2±2.4*	3.4±2.6
	20	7.0±4.4	5.8±3.6	5.9±3.5	5.1±3.1	5.3±3.5	5.0±3.0*

Bibliography

[1] Alexander J. Smola and Peter J. Bartlett, editors. *Advances in Large Margin Classifiers*. MIT Press, Cambridge, MA, USA, 2000. ISBN 0262194481.

[2] Llew Mason, Peter Bartlett, and Jonathan Baxter. Direct optimization of margins improves generalization in combined classifiers. *Advances in Neural Information Processing Systems*, pages 288–294, 1999.

[3] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:2000, 1998.

[4] Yoav Freund. A more robust boosting algorithm. *arXiv preprint arXiv:0905.2138*, 2009.

[5] 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*, 26(5):1651–1686, 1998.

[6] Leo Breiman. Arcing classifiers. *Annals of Statistics*, 26:801–823, 1998.

[7] David Mease and Abraham Wyner. Evidence contrary to the statistical view of boosting. *Journal of Machine Learning Research*, 9:131–156, June 2008. ISSN 1532-4435.

[8] Gunnar Rätsch. Robust boosting via convex optimization: Theory and applications (doctoral thesis), 2001.

[9] Gunnar Rätsch and Manfred K. Warmuth. Maximizing the margin with boosting. In Jyrki Kivinen and Robert H. Sloan, editors, *COLT*, volume 2375 of *Lecture Notes in Computer Science*, pages 334–350. Springer, 2002.

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

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

[12] Leo Breiman. Randomizing outputs to increase prediction accuracy. *Machine Learning*, 40(3):229–242, 2000.

[13] Gonzalo Martínez-Muñoz and Alberto Suárez. Switching class labels to generate classification ensembles. *Pattern Recognition*, 38(10):1483–1494, 2005.

[14] Xingquan Zhu and Xindong Wu. Class noise vs. attribute noise: A quantitative study. *Artificial Intelligence Review*, 22(3):177–210, 2004.

[15] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: A survey. *IEEE Trans. Neural Netw. Learning Syst.*, 25(5):845–869, 2014.

[16] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96), Bari, Italy, July 3-6, 1996*, pages 148–156, 1996.

[17] Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier SYSTEMS, LBCS-1857*, pages 1–15. Springer, 2000.

[18] Prem Melville, Nishit Shah, Lilyana Mihalkova, and Raymond J. Mooney. Experiments on ensembles with missing and noisy data. In *Proceedings of the Workshop on Multi Classifier Systems*, pages 293–302. Springer Verlag, 2004.

[19] Christian Pölitz and Ralf Schenkel. Robust Ranking Models Using Noisy Feedback. In *Workshop "Information Retrieval Over Query Sessions" (SIR 2012) at ECIR 2012*, pages 1 – 6, 2012.

[20] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1-2):105–139, 1999.

[21] Geoffrey I. Webb. Multiboosting: A technique for combining boosting and wagging. *Machine Learning*, 40(2):159–196, 2000.

[22] Joaquín Abellán and Andrés R Masegosa. Bagging decision trees on data sets with classification noise. In *Foundations of Information and Knowledge Systems*, pages 248–265. Springer, 2010.

[23] Peter Hall and Richard J Samworth. Properties of bagged nearest neighbour classifiers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(3):363–379, 2005.

[24] Faisal Zaman and Hideo Hirose. Effect of subsampling rate on subbagging and related ensembles of stable classifiers. In *Pattern Recognition and Machine Intelligence*, pages 44–49. Springer, 2009.

[25] Gonzalo Martínez-Muñoz and Alberto Suárez. Out-of-bag estimation of the optimal sample size in bagging. *Pattern Recognition*, 43(1):143–152, 2010.

[26] Tom M. Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1 edition, March 1997. ISBN 0070428077.

[27] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387310738.

[28] David Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[29] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012. ISBN 026201825X, 9780262018258.

[30] V. Kalaichelvi and Ahammed Shamir Ali. Article: Application of neural networks in character recognition. *International Journal of Computer Applications*, 52(12): 1–6, August 2012.

[31] Stefan Knerr, Léon Personnaz, and Gérard Dreyfus. Handwritten digit recognition by neural networks with single-layer training. *IEEE Transactions on Neural Networks*, 3(6):962–968, 1992. URL `http://dblp.uni-trier.de/db/journals/tnn/tnn3.html#KnerrPD92`.

[32] W.G. Baxt. Use of an artificial neural network for data analysis in clinical decision-making: The diagnosis of acute coronary occlusion. *Neural Computation*, 2(4): 450–489, 1990.

[33] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2001.

[34] Yuchun Tang, Yan-Qing Zhang, Nitesh V. Chawla, and Sven Krasser. Svms modeling for highly imbalanced classification. *Trans. Sys. Man Cyber. Part B*, 39(1): 281–288, February 2009. ISSN 1083-4419.

[35] O. Chapelle, P. Haffner, and V. N. Vapnik. Support vector machines for histogram-based image classification. *Trans. Neur. Netw.*, 10(5):1055–1064, September 1999. ISSN 1045-9227. doi: 10.1109/72.788646. URL `http://dx.doi.org/10.1109/72.788646`.

[36] Yong Zhou, Youwen Li, and Shixiong Xia. An improved knn text classification algorithm based on clustering. *Journal of computers*, 4(3):230–237, 2009.

[37] PAN Jeng-Shyang, QIAO Yu-Long, and SUN Sheng-He. A fast¡ i¿ k¡/i¿ nearest neighbors classification algorithm. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 87(4):961–963, 2004.

[38] Tsang-Long Pao, Yu-Te Chen, Jun-Heng Yeh, and Yuan-Hao Chang. Emotion recognition and evaluation of mandarin speech using weighted d-knn classification. In *ROCLING*, 2005.

[39] Mark A Friedl and Carla E Brodley. Decision tree classification of land cover from remotely sensed data. *Remote sensing of environment*, 61(3):399–409, 1997.

[40] Qiang Ding, Qin Ding, and William Perrizo. Decision tree classification of spatial data streams using peano count trees. In *Proceedings of the 2002 ACM symposium on Applied computing*, pages 413–417. ACM, 2002.

[41] Antonia Vlahou, John O Schorge, Betsy W Gregory, and Robert L Coleman. Diagnosis of ovarian cancer using decision tree classification of mass spectral data. *BioMed Research International*, 2003(5):308–314, 2003.

[42] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 1990.

[43] Terry Windeatt. Accuracy/diversity and ensemble MLP classifier design. *IEEE Transactions on Neural Networks*, 17(5):1194–1211, 2006.

[44] Vladimir M. Krasnopolsky. Reducing uncertainties in neural network jacobians and improving accuracy of neural network emulations with NN ensemble approaches. *Neural Networks*, 20(4):454–461, 2007.

[45] David W. Opitz and Jude W. Shavlik. Actively searching for an effective neural network ensemble. *Connect. Sci.*, 8(3):337–354, 1996.

[46] Meng Chao, Sun Zhi Xin, and Liu San Min. Neural network ensembles based on copula methods and distributed multiobjective central force optimization algorithm. *Eng. Appl. of AI*, 32:203–212, 2014.

[47] Hyun-Chul Kim, Shaoning Pang, Hong-Mo Je, Daijin Kim, and Sung Yang Bang. Constructing support vector machine ensemble. *Pattern Recognition*, 36(12):2757–2767, 2003.

[48] Tomasz Malisiewicz, Abhinav Gupta, and Alexei A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*, pages 89–96, 2011.

[49] Hyun-Chul Kim, Shaoning Pang, Hong-Mo Je, Daijin Kim, and Sung Yang Bang. Support vector machine ensemble with bagging. In *Pattern Recognition with Support Vector Machines, First International Workshop, SVM 2002, Niagara Falls, Canada, August 10, 2002, Proceedings*, pages 397–407, 2002.

[50] Shaoning Pang, Daijin Kim, and Sung Yang Bang. Membership authentication in the dynamic group by face classification using SVM ensemble. *Pattern Recognition Letters*, 24(1-3):215–225, 2003.

[51] Xiaoyang Tan, Songcan Chen, Zhi-Hua Zhou, and Fuyan Zhang. Recognizing partially occluded, expression variant faces from single training image per person with SOM and soft k-nn ensemble. *IEEE Transactions on Neural Networks*, 16 (4):875–886, 2005.

[52] Guodong Zhou. Discriminative hidden markov modeling with long state dependence using a knn ensemble. In *COLING 2004, 20th International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2004, Geneva, Switzerland*, 2004. URL http://www.aclweb.org/anthology/C04-1004.

[53] Ulf Johansson, Rikard König, and Lars Niklasson. Genetically evolved knn ensembles. In *Data Mining - Special Issue in Annals of Information Systems*, pages 299–313. 2010.

[54] J. Ross Quinlan. Bagging, boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, August 4-8, 1996, Volume 1.*, pages 725–730, 1996.

[55] Robert E. Schapire. A brief introduction to boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages*, pages 1401–1406, 1999.

[56] Michael Gashler, Christophe G. Giraud-Carrier, and Tony R. Martinez. Decision tree ensemble: Small heterogeneous is better than large homogeneous. In *Seventh International Conference on Machine Learning and Applications, ICMLA 2008, San Diego, California, USA, 11-13 December 2008*, pages 900–905, 2008.

[57] Ethem Alpaydin. *Introduction to Machine Learning*. Adaptive computation and machine learning. MIT Press, 2010. ISBN 9780262012430. URL http://books.google.es/books?id=4j9GAQAAIAAJ.

[58] Vladimir Vapnik. *Statistical learning theory*. Wiley, 1998. ISBN 978-0-471-03003-4.

[59] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. doi: 10.1007/BF00994018. URL http://dx.doi.org/10.1007/BF00994018.

[60] Maryam Sabzevari, Gonzalo Martínez-Muñoz, and Alberto Suárez. Improving the robustness of bagging with reduced sampling size. In *Proceedings of the European Symposium on Artificial Neural Networks*, page (in press), 2014.

[61] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2Nd Edition)*. Wiley-Interscience, 2000. ISBN 0471056693.

[62] Pedro Domingos. A few useful things to know about machine learning. *Commun. ACM*, 55(10):78–87, 2012.

[63] Thomas G. Dietterich. Machine learning in nature encyclopedia of cognitive science, 2003.

[64] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[65] Ron Kohavi and David Wolpert. Bias plus variance decomposition for zero-one loss functions. In *Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96), Bari, Italy, July 3-6, 1996*, pages 275–283, 1996.

[66] Jerome H. Friedman. On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Min. Knowl. Discov.*, 1(1):55–77, 1997.

[67] Ahmad Taher Azar and Shereen M. El-Metwally. Decision tree classifiers for automated medical diagnosis. *Neural Computing and Applications*, 23(7-8):2387–2403, 2013.

[68] Prasanta Kumar Dey. Project risk management: a combined analytic hierarchy process and decision tree approach. *Cost Engineering*, 44(3):13–27, 2002.

[69] John F Magee. How to use decision trees in capital investment. *Harvard Business Review*, 42(5):79–96, 1964.

[70] Jong Woo Kim, Byung Hun Lee, Michael J Shaw, Hsin-Lu Chang, and Matthew Nelson. Application of decision-tree induction techniques to personalized advertisements on internet storefronts. *International Journal of Electronic Commerce*, 5:45–62, 2001.

[71] Oded Maimon and Lior Rokach, editors. *Data Mining and Knowledge Discovery Handbook, 2nd ed.* Springer, 2010.

[72] J. Ross Quinlan. Simplifying decision trees. *International journal of man-machine studies*, 27(3):221–234, 1987.

[73] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[74] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993. ISBN 1-55860-238-0.

[75] Charu C Aggarwal. *Data Classification: Algorithms and Applications*. CRC Press, 2014.

[76] K. P. Soman, Shyam Diwakar, and V. Ajay. *Insight into Data Mining: Theory and Practice*. Prentice-Hall of India Pvt.Ltd. ISBN 8120328973.

[77] Robert E. Banfield, Lawrence O. Hall, Kevin W. Bowyer, and W. Philip Kegelmeyer. Ensemble diversity measures and their application to thinning. *Information Fusion*, 6:2005, 2004.

[78] Kamal Ali. On the link between error correlation and error reduction in decision tree ensembles. Technical report, 1995.

[79] Shmuel. Nitzan and Jacob. Paroush. *Collective decision making : an economic outlook / Shmuel Nitzan and Jacob Paroush*. Cambridge University Press Cambridge [Cambridgeshire] ; New York, 1985. ISBN 0521303265.

[80] Peter Büchlmann and Bin Yu. Analyzing bagging. *Annals of Statistics*, pages 927–961, 2002.

[81] Prem Melville and Raymond J. Mooney. Creating diversity in ensembles using artificial data. *Information Fusion*, 6(1):99–111, 2005.

[82] Robert K. Bryll, Ricardo Gutierrez-Osuna, and Francis K. H. Quek. Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets. *Pattern Recognition*, 36(6):1291–1302, 2003.

[83] Tin Kam Ho. Random decision forests. In *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*, ICDAR '95, pages 278–. IEEE Computer Society, 1995. ISBN 0-8186-7128-9. URL http://dl.acm.org/citation.cfm?id=844379.844681.

[84] Gonzalo Martínez-Muñoz, Aitor Sánchez-Martínez, Daniel Hernández-Lobato, and Alberto Suárez. Building ensembles of neural networks with class-switching. In *Artificial Neural Networks–ICANN 2006*, pages 178–187. Springer, 2006.

[85] Eun Bae Kong and Thomas G. Dietterich. Error-correcting output coding corrects bias and variance. In *Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, July 9-12, 1995*, pages 313–321, 1995.

[86] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5: 197–227, 1990.

[87] G. P. Zhang and V. L. Berardi. Time Series Forecasting with Neural Network Ensembles: An Application for Exchange Rate Prediction. *Journal of the Operational Research Society*, 52(6):652, 2001.

[88] Bambang Parmanto, Paul W Munro, and Howard R Doyle. Improving committee diagnosis with resampling techniques. In *Advances in neural information processing systems*, pages 882–888, 1996.

[89] Yong Liu and Xin Yao. Simultaneous learning of negatively correlated neural networks. In *Journal of Artificial Life and Robotics*, pages 183–187, 1998.

[90] Yong Liu and Xin Yao. Ensemble learning via negative correlation. *Neural Networks*, 12(10):1399–1404, 1999.

[91] Peng Cao, Jinzhu Yang, Wei Li, Dazhe Zhao, and Osmar R. Zaïane. Ensemble-based hybrid probabilistic sampling for imbalanced data learning in lung nodule CAD. *Comp. Med. Imag. and Graph.*, 38(3):137–150, 2014.

[92] Sahand Khakabimamaghani, Farnaz Barzinpour, and Mohammad Reza Gholamian. A high diversity hybrid ensemble of classifiers. In *Software Engineering and Data Mining (SEDM), 2010 2nd International Conference on*, pages 461–466. IEEE, 2010.

[93] Louis Wehenkel, Damien Ernst, and Pierre Geurts. Ensembles of extremely randomized trees and some generic applications. *Proceedings of Robust Methods for Power System State Estimation and Load Forecasting*, 2006.

[94] Sean Whalen and Gaurav Pandey. A comparative analysis of ensemble classifiers: Case studies in genomics. In *2013 IEEE 13th International Conference on Data Mining, Dallas, TX, USA, December 7-10, 2013*, pages 807–816, 2013.

[95] Sean A Gilpin and Daniel M Dunlavy. Relationships between accuracy and diversity in heterogeneous ensemble classifiers. *Department of Energy's Nuclear Security Administration under Contract DE-AC04-94AL85000,, SAND2009, 694OC*, 2009.

[96] Kuo-Wei Hsu and Jaideep Srivastava. Diversity in combinations of heterogeneous classifiers. In *Advances in Knowledge Discovery and Data Mining, 13th Pacific-Asia Conference, PAKDD 2009, Bangkok, Thailand, April 27-30, 2009, Proceedings*, pages 923–932, 2009.

[97] Mahmoud O Elish, Tarek Helmy, and Muhammad Imtiaz Hussain. Empirical study of homogeneous and heterogeneous ensemble models for software development effort estimation. *Mathematical Problems in Engineering*, 2013, 2013.

[98] Daniel M. Dunlavy and Sean A. Gilpin. Heterogeneous ensemble classification. In *Proceedings of the 2008 Sandia Workshop on Data Mining and Data Analysis*, number SAND2008-6109, pages 33–35. Sandia National Laboratories, Albuquerque, NM and Livermore, CA, 2008.

[99] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, EuroCOLT '95, pages 23–37, London, UK, UK, 1995. Springer-Verlag. ISBN 3-540-59119-2.

[100] Duin Robert P. W. Tax David M. J. Hartog J. E. den Breukelen, M. Handwritten digit recognition by combined classifiers. *Kybernetika*, 34(4):[381]–386, 1998. URL http://eudml.org/doc/33365.

[101] Robert P. W. Duin and David M. J. Tax. Experiments with classifier combining rules. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, MCS '00, pages 16–29, London, UK, UK, 2000. Springer-Verlag. ISBN 3-540-67704-6.

[102] Anil K. Jain, Robert P. W. Duin, and Jianchang Mao. Statistical pattern recognition: A review. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 22(1):4–37, 2000.

[103] Josef Kittler, Mohamad Hatef, Robert P. W. Duin, and Jiri Matas. On combining classifiers. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(3):226–239, March 1998. ISSN 0162-8828. doi: 10.1109/34.667881. URL http://dx.doi.org/10.1109/34.667881.

[104] David M. J. Tax, Martijn van Breukelen, Robert P. W. Duin, and Josef Kittler. Combining multiple classifiers by averaging or by multiplying? *Pattern Recognition*, 33(9):1475–1485, 2000. doi: 10.1016/S0031-3203(99)00138-7. URL http://dx.doi.org/10.1016/S0031-3203(99)00138-7.

[105] David M.J. Tax, Robert P.W. Duin, and Martijn Van Breukelen. Comparison between product and mean classifier combination rules. In *In Proc. Workshop on Statistical Pattern Recognition*, pages 165–170, 1997.

[106] David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992. doi: 10.1016/S0893-6080(05)80023-1. URL http://dx.doi.org/10.1016/S0893-6080(05)80023-1.

[107] On bagging and nonlinear estimation. *Journal of Statistical Planning and Inference*, 137(3):669–683, March 2007. ISSN 03783758. doi: 10.1016/j.jspi.2006.06.002. URL http://dx.doi.org/10.1016/j.jspi.2006.06.002.

[108] Andreas Buja and Werner Stuetzle. Observations on bagging. *Statistica Sinica*, 16(2):323, 2006.

[109] Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Boosting algorithms as gradient descent in function space. NIPS, 1999.

[110] Pall Oskar Gislason, Jon Atli Benediktsson, and Johannes R. Sveinsson. Random forests for land cover classification. *Pattern Recogn. Lett.*, 27(4):294–300, March 2006. ISSN 0167-8655.

[111] Rich Caruana, Nikolaos Karampatziakis, and Ainur Yessenalina. An empirical evaluation of supervised learning in high dimensions. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, pages 96–103, 2008. doi: 10.1145/1390156.1390169.

[112] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, pages 161–168, 2006. doi: 10.1145/1143844.1143865.

[113] Gonzalo Martínez-Muñoz and Alberto Suárez. Switching class labels to generate classification ensembles. *Pattern Recognition*, 38(10):1483–1494, 2005.

[114] Hideyuki Watanabe, Tsukasa Ohashi, Shigeru Katagiri, Miho Ohsaki, Shigeki Matsuda, and Hideki Kashioka. Robust and efficient pattern classification using large geometric margin minimum classification error training. *Signal Processing Systems*, 74(3):297–310, 2014.

[115] Ofer Dekel, Joseph Keshet, and Yoram Singer. Large margin hierarchical classification. In *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*, 2004.

[116] Kilian Q. Weinberger and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 2009.

[117] Peter L. Bartlett, Michael Collins, Benjamin Taskar, and David A. McAllester. Exponentiated gradient algorithms for large-margin structured classification. In *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada]*, 2004.

[118] Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. In *Machine Learning*, pages 277–296, 1998.

[119] Jim Jing-Yan Wang, Majed Alzahrani, and Xin Gao. Large margin image set representation and classification. *CoRR*, 2014.

[120] Vladimir Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999. doi: 10.1109/72.788640. URL http://doi.ieeecomputersociety.org/10.1109/72.788640.

[121] Peter Bartlett and John Shawe-taylor. Generalization performance of support vector machines and other pattern classifiers, 1998.

[122] Vladimir S. Cherkassky and Filip Mulier. *Learning from Data: Concepts, Theory, and Methods*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1998. ISBN 0471154938.

[123] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995. ISBN 0-387-94559-8.

[124] Ulrike Von Luxburg and Bernhard Schölkopf. Statistical learning theory: Models, concepts, and results. *arXiv preprint arXiv:0810.4752*, 2008.

[125] Jonathan I. Maletic and Andrian Marcus. Data cleansing: Beyond integrity analysis. In *Fifth Conference on Information Quality (IQ 2000)*, pages 200–209, 2000.

[126] Nicola Segata, Enrico Blanzieri, Sarah Jane Delany, and Pádraig Cunningham. Noise reduction for instance-based learning with a local maximal margin approach. *Journal of Intelligent Information Systems*, 35(2):301–331, 2010.

[127] Nicola Segata, Enrico Blanzieri, and Pádraig Cunningham. A scalable noise reduction technique for large case-based systems. In *Case-Based Reasoning Research and Development*, pages 328–342. Springer, 2009.

[128] Jonathan Young, John Ashburner, and Sebastien Ourselin. Wrapper methods to correct mislabelled training data. In *Proceedings of the 2013 International Workshop on Pattern Recognition in Neuroimaging*, PRNI '13, pages 170–173. IEEE Computer Society, 2013. ISBN 978-0-7695-5061-9.

[129] Sergiy Fefilatyev, Matthew Shreve, Kurt Kramer, Lawrence O. Hall, Dmitry B. Goldgof, Rangachar Kasturi, Kendra Daly, Andrew Remsen, and Horst Bunke. Label-noise reduction with support vector machines. In *ICPR*, pages 3504–3508, 2012.

[130] Carla E. Brodley and Mark A. Friedl. Identifying mislabeled training data. *CoRR*, abs/1106.0219, 2011.

[131] Taghi M Khoshgoftaar, Shi Zhong, and Vedang Joshi. Enhancing software quality estimation using ensemble-classifier based noise filtering. *Intelligent Data Analysis*, 9(1):3–27, 2005.

[132] Sofie Verbaeten and Anneleen Van Assche. Ensemble methods for noise elimination in classification problems. In *Multiple classifier systems*, pages 317–325. Springer, 2003.

[133] Xingquan Zhu, Xindong Wu, and Qijun Chen. Eliminating class noise in large datasets. In *ICML*, volume 3, pages 920–927, 2003.

[134] Shi Zhong, Wei Tang, and Taghi M Khoshgoftaar. Boosted noise filters for identifying mislabeled data. Technical report, Technical report, Department of computer science and engineering, Florida Atlantic University, 2005.

[135] Carlos Javier Mantas and Joaquín Abellán. Credal decision trees to classify noisy data sets. In *HAIS*, pages 689–696, 2014.

[136] Guillaume Stempfel and Liva Ralaivola. Learning svms from sloppily labeled data. In *ICANN (1)*, pages 884–893, 2009.

[137] Emilie Niaf, Rémi Flamary, Olivier Rouvière, Carole Lartizien, and Stéphane Canu. Kernel-based learning from both qualitative and quantitative labels: Application to prostate cancer diagnosis based on multiparametric mr imaging. *IEEE Transactions on Image Processing*, 23(3):979–991, 2014.

[138] Kamal M. Ali and Michael J. Pazzani. Error reduction through learning multiple descriptions. *Machine Learning*, 24(3):173–202, 1996.

[139] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

[140] Ross A McDonald, David J Hand, and Idris A Eckley. An empirical comparison of three boosting algorithms on real data sets with artificial class noise. In *Multiple Classifier Systems*, pages 35–44. Springer, 2003.

[141] Eibe Frank and Bernhard Pfahringer. Improving on bagging with input smearing. In *PAKDD*, pages 97–106, 2006.

[142] Gonzalo Martínez-Muñoz, Aitor Sánchez-Martínez, Daniel Hernández-Lobato, and Alberto Suárez. Class-switching neural network ensembles. *Neurocomputing*, 71 (13):2521–2528, 2008.

[143] David P. Williams. Label alteration to improve underwater mine classification. *IEEE Geoscience and Remote Sensing Letters*, 8(3):488–492, 2011.

[144] Yves Grandvalet. Bagging equalizes influence. *Machine Learning*, 55(3):251–270, 2004.

[145] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:2000, 1998.

[146] Yoav Freund. An adaptive version of the boost by majority algorithm. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, pages 102–113, 2000.

[147] Amitava Karmaker and Stephen Kwek. A boosting approach to remove class label noise. *International Journal of Hybrid Intelligent Systems*, 3(3):169–177, 2006.

[148] Abba Krieger, Chuan Long, and Abraham Wyner. Boosting noisy data. In *ICML*, pages 274–281, 2001.

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

[150] Catherine Blake and Christopher J Merz. {UCI} repository of machine learning databases. 1998.

[151] Simon Bernard, Laurent Heutte, and Sébastien Adam. Influence of hyperparameters on random forest accuracy. In *Multiple Classifier Systems, 8th International Workshop, MCS 2009, Reykjavik, Iceland, June 10-12, 2009. Proceedings*, pages 171–180, 2009.

[152] Janez Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.