Chapter 1

Introduction

The research presented in this thesis concerns two different domains related to supervised automatic learning, namely decision tree induction and the treatment of temporal data. Both problems are approached using the bias/variance tradeoff paradigm. Both topics are defined and motivated in Section 1.1. Then, we define in Section 1.2 the general objectives of our research in this context and the way we have tackled these objectives. The organization of the thesis is given in Section 1.3. Eventually, we close this introduction with an enumeration of the main contributions of this thesis.

1.1 Context and motivations

1.1.1 Automatic learning

Automatic learning denotes methods which aim at extracting a model of a system from the sole observation (or the simulation) of this system in some situations. By model, we mean some relationships between the variables used to describe the system. The goal of this model may be to predict the behavior of this system in some unencountered situations or to help understanding its behavior.

The interest of automatic learning is obvious when it is impossible to analytically model a system as it is often easy to gather data from its direct observation. For example, a handwriting recognition system is very difficult to put into equation, notably because of the high variability of handwriting. However, it is easy to ask several people to write some letters or words and hence constitute a database for applying automatic learning techniques. Even in situations where a physical (analytical or numerical) model based on first principles is available, the application of automatic learning can be useful, for various reasons. First, from a practical point of view, automatic learning may provide simpler models which may be exploited more efficiently in practice than the existing ones. In addition, these models constructed by automatic learning may be customized so as to express the relationship among any desired *subset* of system variables. Second, if automatic learning produces models which are comprehensible (interpretable), it can help to understand the physical behavior of the system. Third, automatic learning may take into account uncertainties about some variables which can not be modeled in a deterministic or accurate way (for example, the external environment with which the system interacts, and in particular human operators or users of the system). In all these cases, observations may be gathered from numerical simulations based on the physical model, completed, if necessary, by randomized inputs from the environment in the form of randomized system parameters or disturbances. For example, electrical power systems are very complex systems which can nevertheless be modeled by (approximate) analytical models. However, the practical use of these models is very computationally demanding and they can not be used in real-time to predict the future behavior of the system. In this case, the power system can be simulated offline under different conditions and the particular relationship we want to observe is extracted from data by automatic learning (see [Weh98]).

Since its beginning, automatic learning has drawn interest from several communities: from the artificial intelligence community where it can solve several shortcomings of expert systems (by providing rules in an automatic way from observations), from the statistical community but also from the system theory community (where it is related to system identification). Each of these communities has brought different ways to state the problem and a large variety of techniques. Recent interest in this field has also grown because of the rapid progress of computer technology and also because of the rapid growth and high accessibility of the available data. The field has thus evolved into a broader field which is called KDD (knowledge discovery in databases). KDD is defined in [FPSS96] as "the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data". The emergence of this field has also emphasized several needs in automatic learning techniques such as the scalability of the algorithms, the possibility of using them in an adaptive way (in order to not repeat the knowledge extraction from scratch when data grows), and interpretability or understandability of the knowledge extracted.

1.1.2 Supervised learning

Supervised learning is the part of automatic learning which focuses on modeling input/output relationships. More precisely, the goal of supervised learning is to identify a mapping from some input variables to some output variables on the sole basis of a sample of observations of the values of these variables. The variables are often called (input or output) attributes, observations are called objects and the sample of object is the learning sample. For example, consider the bi-dimensional problem of Figure 1.1 and assume that the goal of learning is to find a function which reproduces at best the classification between empty triangle and full diamond. For example, each point of this space may represent a patient which is described by two variables which are the result of two medical tests. Patients marked by a full diamond are suffering from one disease while patients marked by a empty triangle are healthy. The goal of supervised learning is to provide a rule that may help a doctor to predict the state of a new patient on which the test results are known. Usually, of course, each point of the learning sample is described by more than one or two attributes and these attributes may have other values than a number (discrete values for example).

This kind of problem is solved by a (supervised) learning algorithm. Loosely speaking, a learning algorithm receives a learning sample (like the one represented in Figure 1.1) and returns a function h (an hypothesis) which is chosen in a set of candidate models (the hypothesis space). For example, a potential candidate function returned by a learning algorithm is the one given in Figure 1.1. Even though this function makes several mistakes on the learning sample, it seems that these mistakes are due to noise in the data (which may be due either to error in measurements or variables which have not been observed) and the function is a plausible explanation for the function which has served to generate the data.

The main criterion used to assess learning algorithms is their prediction accuracy, i.e. the way the model they produce generalizes to unseen data. Another important criterion, especially in the context of KDD, is the interpretability of this model. For example, in our medical application, it would be interesting to know if one of the results of the two tests

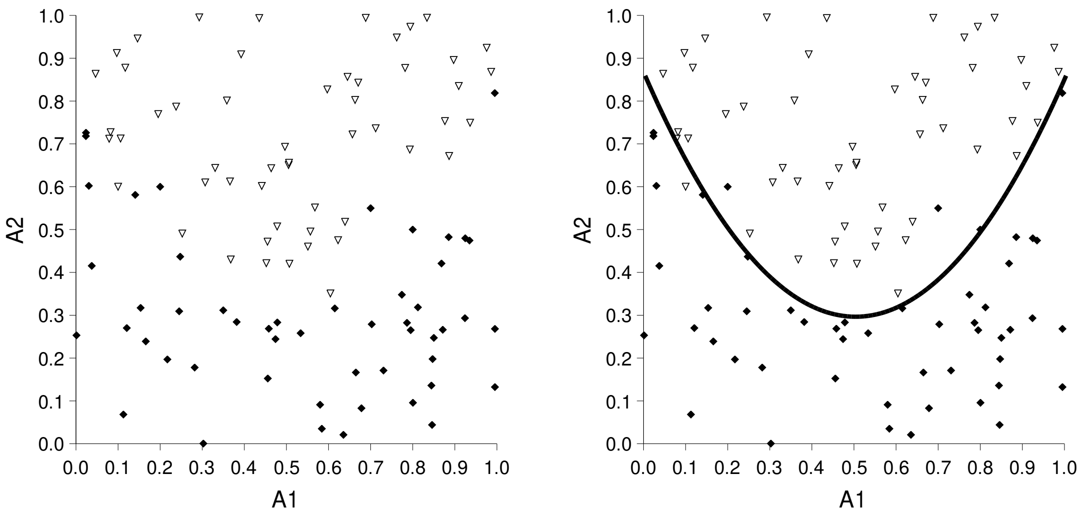

Figure 1.1: Left, a sample corresponding to a simple classification problem in a two-dimensional space. Right, the optimal classification boundary for this problem

is not correlated to the disease. Computational efficiency and scalability are also of great concern, especially when it comes to apply learning algorithms on very large dataset.

1.1.3 The Bias/variance tradeoff in supervised learning

To explain why the supervised learning problem is not trivial, it is useful to give some feeling about the bias/variance tradeoff in this context. Let us assume that we want to separate the two classes in Figure 1.1 with a linear decision boundary. The left part of Figure 1.2 shows one such model. With a linear model, it is impossible to separate perfectly the two classes. Patently, this model is not satisfactory to solve this task (it is very far from the optimal model of Figure 1.1). The chosen family of functions is not flexible enough to model our classification problem. This phenomenon responsible for the error in this case is called the **bias**.

Let us assume now that we use a more complex family of functions among which there exists one or several hypotheses which realize a perfect separation between classes in our sample. Then, from this family, the learning algorithm may output the decision boundary given in the right part of Figure 1.2. This time, the classification boundary perfectly separates elements of the different classes. Like the linear model however, this model is still not appropriate. This time, actually, the model is too complex and hence it learns "too much" information from our data. In this case, we say that the learning algorithm *overfits* the data. If we draw another learning sample for the same problem (by repeating the experiment on another group of patients), then it is very likely that the model found by the same learning algorithm will be very different from this one. In this case, we say that the learning algorithm suffers from **variance**. The hypothesis which it returns is very variable from one learning sample to another.

When it comes to apply the model on new objects, both bias and variance are sources of error and hence they should be minimized. As they are functions reacting in an opposite direction to a variation of the flexibility of the hypothesis space, there is a tradeoff between these two effects which must be taken into account when designing learning algorithms.

Because of its central role in automatic learning, this tradeoff has received a lot of attention in the supervised learning literature. There are mainly two ways to handle this

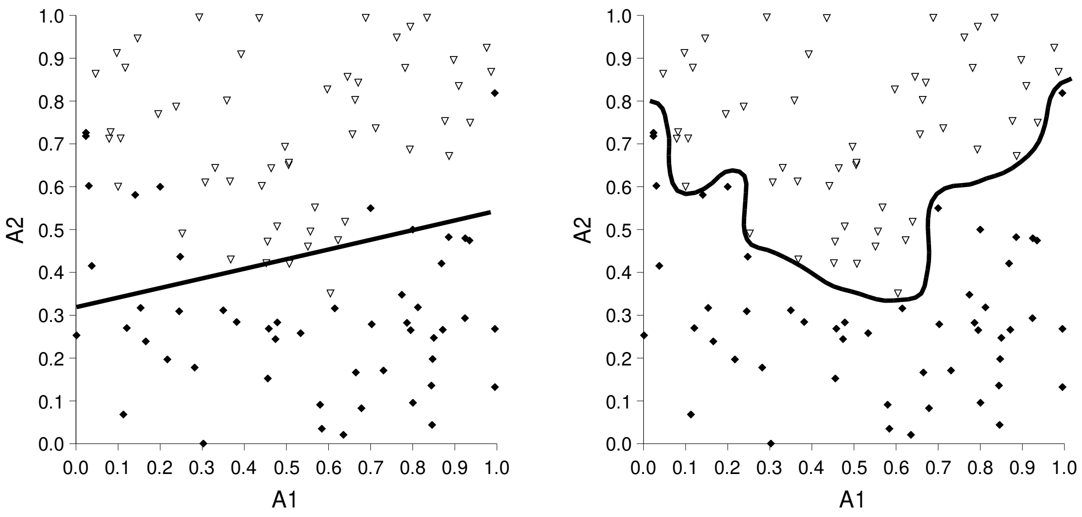

Figure 1.2: Left, a too simple model, right, a too complex one

tradeoff. The first one is to introduce in the learning algorithm some means to control the complexity of the hypothesis space and then adapt this complexity to the problem under consideration. This approach is necessary but it is not plainly satisfactory since it supposes to sacrifice some bias to reduce the variance. Recently, a series of methods have been proposed which reduce the variance without increasing the bias. The idea behind these methods is to aggregate the predictions given by several models to reduce the dependence on the random nature of the learning sample and hence the variance. In classification, this approach consists in building several models for the same sample which all overfit the data (to some extent, like the right one in Figure 1.2). To classify a new object, we make a count of the number of times each class is given by one model and then finally output the class which receives the majority of votes among all models. Intuitively, it is clear that such voting should stabilize the classification with respect to the individual classification given by every models (just like the majority vote among several people in a population is generally more stable than the vote of a random person taken from this population). These methods are called averaging techniques or ensemble methods. Among the most popular ensemble methods, there are bagging [Bre96b] and boosting [FS96]. Because of the very impressive results obtained by these methods, a lot of research has been carried out recently in this domain [Die00b].

1.1.4 Decision tree induction

Among learning algorithms, one of the most popular one is decision tree induction [BFOS84]. In its simple form, a decision tree combines several binary tests in a tree structure. For example, Figure 1.3 plots a decision tree and the resulting decision boundary for our bi-dimensional example problem. Each interior node of the tree is labeled with a test (here which compares the attribute value to a threshold) and each terminal node is labeled with a class. To produce a classification for a new object which attribute values are known, we simply propagate it into the tree from the top node according to the test answers. When a terminal node is reached, its classification is attributed to the object. By using such tests, the tree progressively partitions the input space into hyper-rectangular regions so as to yield regions where the classification is constant. In the representation of the decision tree in Figure 1.3, the areas of different colors at a node represent the proportions of patients

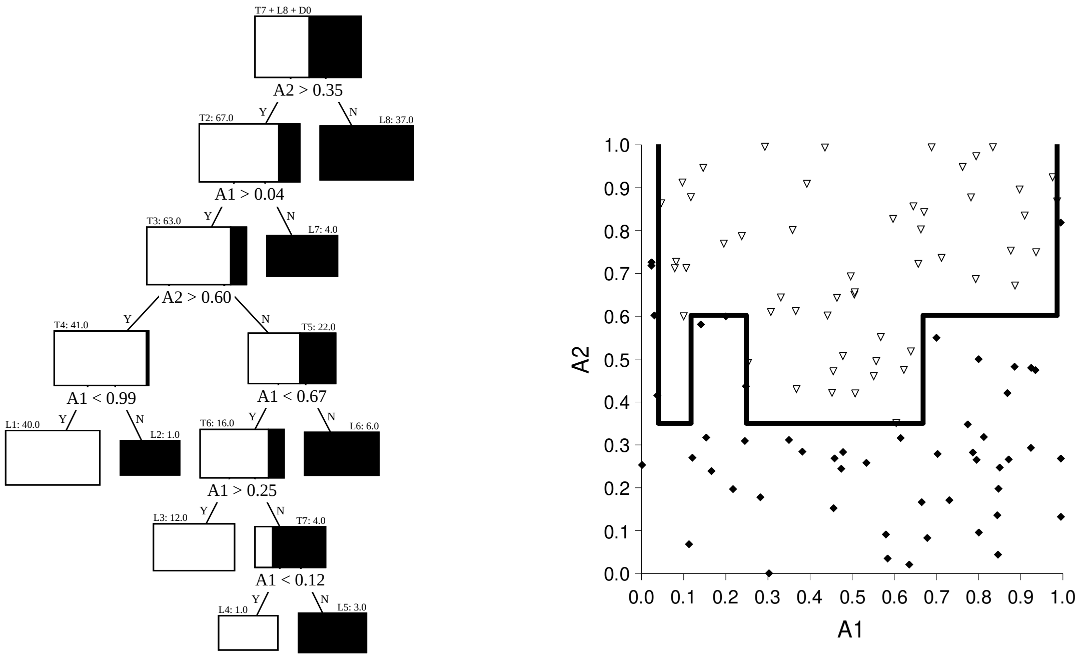

Figure 1.3: Left, a decision tree, right, the corresponding decision boundary

of each class which go through this node, highlighting how the tests progressively gather information about the output class.

The popularity of this learning algorithm is related to the fact that it combines several nice characteristics:

- **Interpretability.** Because of its structure, the way attributes interact to give a prediction is very readable. For example, the tree of Figure 1.3 tells to the doctor that if the result of the $A2$ test is lower than 0.35 then the patient suffers from the disease and there is no need to carry out the $A1$ test.

- **Efficiency.** The induction of such trees is done by a top-down algorithm which (in its most simple version) recursively splits terminal nodes of the current tree until they all contain elements of only one class. The resulting algorithm is very fast and can be used on very large datasets (e.g. of millions of objects and thousands of variables).

- **Flexibility.** This method is very open-minded. It does not make any hypothesis about the problem under consideration. It can handle both continuous and discrete attributes (by adapting the test choices). Predictions at leaf nodes may be symbolic (like in our example) or numerical (in which case, trees are called regression trees). Furthermore, the tree induction method can be easily extended by improving tests at tree nodes (introducing for example linear combinations of attributes) or providing a prediction at terminal nodes by means of another model.

It is one of the only methods in machine learning which present so many advantages at the same time (see [HTF01] for a comparison with other learning algorithms).

However, decision trees still suffer one drawback: their accuracy is often not competitive with other learning algorithms (like e.g. neural networks). This lack of accuracy is mainly due to their variance. Indeed, during induction, the decision tree can be extended as far as needed to perfectly separate objects from the learning sample. So, their bias is low. However, because of this high adaptivity to the learning sample, a small perturbation of the learning sample may result in very different decision trees. So, their variance is high. In addition to the negative effect on accuracy, this instability reduces also their interpretability: we can not really trust the choices of tests since they are changing from one learning sample to another.

There are several existing approaches to improve accuracy of decision trees by reducing their variance. First, pruning consists in simplifying the tree by closing some of its test nodes so as to find the best compromise between bias and variance. Although pruning may reduce substantially the complexity of the tree by removing useless parts of it, its effect on accuracy is often limited [Min89a]. A more efficient way to improve the accuracy of decision trees is to aggregate the predictions given by several of them for the same problem. Various techniques have been proposed in the literature to generate different decision trees from the same learning sample for aggregation and they often give very impressive improvement of accuracy. Nevertheless, they also destroy some of the main advantages of decision trees, namely computational efficiency and interpretability. Indeed, the computation time needed to build a model is multiplied by the number of trees which have to be constructed. Also, since the prediction is given by majority vote among several trees, the way attributes intervene to make up the final prediction is somewhat blurred by the process.

1.1.5 Temporal data

Classical machine learning algorithms like decision tree induction assume that objects of the learning sample are described by attributes which take simple values, essentially numerical or symbolic. However, because of the success of automatic learning in a lot of domains and because of the few hypotheses underlying such methods (they only need to gather data), automatic learning is applied to a lot of domains which need to handle more complex types of data, like image, text, or biological sequences. Among all these types of data, we focus here on temporal data. Restricting ourself to the supervised classification problem, objects of the learning sample correspond in this case to trajectories of a dynamic system which are described by several temporal attributes and the goal of learning is to discriminate among the trajectories according to a given classification. Many practical problems satisfy this definition. For example, in speech recognition, each object corresponds to the utterance of a word and is described by the evolution of the acoustic speech signal over time. The goal of learning is to recognize the word which has been pronounced. Other application domains concern for example, computer system intrusion detection, gestures recognition, and electrocardiogram (ECG) signal classification.

With the introduction of the temporal dimension, each object is in fact described by a large number of low-level values, each one corresponding to the value of an attribute at a given time point. Often, every such low-level value alone is useless for classification. Only the way they are related in time is useful for classification. The difficulties of this domain are manifold and related to this high dimensionality. First, automatic learning algorithms will suffer from a high variance. Indeed, because of the high number of low-level attributes, it is easy to perfectly separate a learning sample and the "chances" for overfitting are increased. So, variance deserves special attention in such domains. Also, temporal problems often correspond to very large datasets and application of automatic learning is often very demanding in terms of computation times and computer resources. Another difficulty is to produce interpretable rules. Indeed, interpretable methods applied on low-level attributes without taking into account the temporal dimension will likely produce very complex models which are very difficult to understand.

Several algorithms have been developed for handling temporal data in specific domains. Practically, the difficulties mentioned earlier are overcome in these systems by introducing a lot of a priori information about the problem. For example, many systems exist to solve the speech recognition problem. These systems usually use specific information about the physical voice generation mechanism to simplify the learning phase. If these problem specific approaches often give very good results in terms of accuracy, they are not widely applicable. Furthermore, they fail in providing interpretable models since on the contrary they assume a prior understanding of the problem to solve. In this context, what is missing is thus general approaches which produce accurate and/or interpretable models.

1.2 Objectives of the thesis and our way to solutions

The research presented in this thesis concerns the two domains motivated in the previous section, namely improvement of decision tree induction and treatment of temporal data. In our search for a solution to both these problems, the bias/variance tradeoff will be used to guide most of our choices. One of the side objectives of this thesis is to show how such a theoretical tool may help either to improve existing methods or to design algorithms to handle new types of problems.

1.2.1 Improvements of decision tree induction

In the context of decision tree induction, the main objective of this thesis is to analyze the sources and consequences of the high variance of decision trees and to investigate means to reduce this variance without jeopardizing efficiency and interpretability of this method.

The resolution of this question in this thesis is a four step process. First, this variance is studied in detail by several experiments. The main goals of these experiments are to highlight quantitatively the high variance of decision trees and to find out the main sources of this variance. One of the main conclusions of these experiments is that decision tree variance is (indeed) high and it results in a high variance of the parameters that are used to define the tree, mainly the discretization thresholds for numerical attributes.

The second step consists in trying to find methods which reduce the variance of the decision tree induction algorithm. In this goal, we evaluate classical pruning algorithms that adapt the complexity of decision trees and propose several means to reduce the discretization threshold variance. Although both techniques certainly improve the interpretability of decision trees (by reducing the complexity and stabilizing the parameters), we show that they have however little effect on decision tree accuracy. This experiment shows the limitations of what can be done in terms of accuracy by using a single decision tree. One conclusion of this second step is thus that it is necessary to extend the hypothesis space of decision trees if we want to obtain further improvement in terms of accuracy.

The next step is thus to consider ensemble techniques in the context of decision trees which use more complex hypothesis spaces. In the context of decision trees, several ensemble methods have been proposed which somewhat randomize the induction algorithm to produce different models from the same learning sample. Since our experiments highlight the high variance of test parameters, we introduce extremely randomized trees (Extra-trees) where tests are chosen almost randomly (i.e. without taking into account the learning sample). When several extra-trees are aggregated, the improvement in terms of accuracy by variance reduction is very impressive. Because of the extreme randomization, the result is also a computationally very efficient method which is competitive with single decision tree induction in spite of the fact that it needs to build several trees.

However, although very accurate and efficient, the resulting method is intrinsically not interpretable since it uses several models to produce a prediction and furthermore these models are random.

In the last step, we propose a new algorithm which tries to reproduce the prediction of an ensemble of models with only one model. This general algorithm generates several predictions by perturbing the attribute vector corresponding to an object and using a single model to produce a prediction for each of these perturbed vectors. We call this algorithm "dual perturb and combine". In the context of decision trees, a closed form approximation of this algorithm is developed which essentially replaces crisp thresholds at the test nodes of a tree by soft ones. Actually, this method bridges the gap between decision trees and ensembles of decision trees. Of the first, it saves the interpretability (by using only one model), and with ensemble methods, it shares some of the accuracy (by reducing the variance).

1.2.2 Treatment of temporal data

The main objective of this part of the thesis is to design general approaches to solve the problem of supervised classification with temporal attributes. We make the additional assumption that temporal signals in the dataset are represented in discrete time. Discrete time temporal signals are often called time series in the literature. In this domain, our ambition is not to give a final solution to the problem but rather to give some preliminary steps in the goal of giving accurate and interpretable methods. This objective is realized in three steps.

First, we explore the use of low-level features with classical learning algorithms, with the aim of highlighting the limitations of such approaches. On several problems, provided that we use strong variance reduction techniques (like extra-trees), this approach gives not so bad results. However, on problems which present typical behavior of the temporal domain, variance reduction techniques do not succeed in improving accuracy in an acceptable way.

Second, we propose a new approach which also uses classical methods on low-level features but which nevertheless exploits the temporal ordering of values. A time series is divided into all its subsequences of a given length and a model is constructed to classify these subsequences on the basis of low-level features. This algorithm is called "segment and combine" and its validation shows that it gives significant improvement especially on typical temporal problems.

Even if algorithms working with low-level features give acceptable results in terms of accuracy, they do not provide interpretable models even if the classical algorithm behind it was interpretable. So, the third step of this work is to propose a new algorithm which provides interpretable models. To this end, decision tree tests are extended to take into account local shift invariant properties in time series. Such patterns are extracted during tree induction from the time series appearing in the learning sample. To reduce the search space for candidate patterns and improve interpretability, time series are compressed by using piecewise constant models. Although they were introduced mainly to improve the interpretability of decision trees, such tests actually allow to improve also accuracy in problems which are characterized by local shift invariant properties (mainly because of a low bias). On other problems however, it deteriorates accuracy with respect to the best use of low-level features, mainly because of an increase of variance.

1.3 Organization of the thesis

The thesis is organized into three parts. The first part aims at introducing the supervised learning problem and the bias/variance tradeoff. The second part is devoted to the improvements of decision tree induction while the treatment of temporal data is studied in the third part. In order to improve readability, we have collected some technical discussions as well as the detailed presentation of empirical results in appendices. Finally, the conclusions chapter gives a general synthesis of the research developed in this thesis and discusses some ideas for future work. The three parts of the body of the thesis are structured as follows.

Part I: Supervised learning and the bias/variance tradeoff

In chapter 2, we define the supervised learning problem using a probabilistic framework. Several classical learning algorithms are described and discussed in terms of the practical criteria used to compare learning algorithms, namely accuracy, computational efficiency, and interpretability. Eventually, we conclude this chapter with a description of usual methods used to estimate the accuracy of models.

Chapter 3 is devoted to the bias/variance tradeoff in automatic learning. After an informal illustration of this tradeoff on a simple synthetic problem, we give the well-known decomposition of the mean square error into the two positive terms that reflect bias and variance. The problem of the decomposition of mean error rate in classification is then tackled. While the decomposition of square error is well agreed upon, several (actually, quite many) decompositions have been proposed in classification. One of them is selected which will be used throughout our experiments. Other decompositions are nevertheless given and related in Appendix A. In this chapter as well, several experiments are carried out which aim at providing some insight about the way bias and variance evolve according to the parameters of learning algorithms and the studied problem. Eventually, the chapter is concluded with some analytical results found in the literature related to the bias/variance tradeoff.

To conclude this part, we review in Chapter 4 classical variance reduction techniques. Two types of techniques are considered: techniques that control the bias/variance trade-off in the context of a given learning algorithm and ensemble methods which consists in aggregating several models. In this latter case, we propose a unifying framework based on random algorithms to describe ensemble methods which, like in [Bre96a], are termed "perturb and combine" algorithms. An analysis of the bias/variance decomposition of these algorithms identifies the conditions under which such algorithms should improve accuracy. In this chapter, we give also an introduction of the general dual perturb and combine algorithm which is studied in chapter 8 in combination with decision trees. Related techniques like Bayesian approaches and boosting type of algorithms are also discussed here.

The reader already familiar with supervised learning and the bias/variance tradeoff may skip most of the first part or refer to it only when necessary.

Part II: Bias and variance in decision tree induction

The second part of the thesis is devoted to the analysis and the improvement of the decision tree induction algorithm. This part is decomposed into five chapters.

Chapter 5 contains several experiments that aim at showing and analyzing the high variance of decision trees. This variance is analyzed from both the point of view of accuracy (measuring its effect on error rates) and interpretability (measuring the variance of the parameters that define the decision tree, i.e. the discretization thresholds and the attribute which is chosen at a test node). Among other things, we study the effect on variance of the score measure which is used to assess candidate tests during tree induction.

Chapter 6 is devoted to the reduction of decision tree variance. First, some experiments are carried with a pruning algorithm. Then, we propose several techniques to reduce the variance of the discretization thresholds of numerical attributes. The effect of these techniques on prediction accuracy is also assessed.

In chapter 7, we approach the problem of decision tree variance reduction by perturb and combine algorithms. After some experiments with the popular bagging algorithm, we give a short review of the perturb and combine algorithms specifically designed for decision trees. Then, we introduce our perturb and combine algorithm which builds extremely randomized trees. This algorithm is evaluated and compared with bagging and boosting algorithms. Eventually, we discuss the promising capabilities of this algorithm in terms of scalability and for bias reduction.

Chapter 8 is devoted to the application of the dual perturb and combine approach to decision trees. A closed-form implementation of this algorithm is developed which shows that it actually transforms a classical decision tree into a soft tree model. The method is validated and compared with bagging. Its behavior is related to the one of other perturb and combine algorithms and its combination with bagging is assessed. We conclude this chapter by discussing related work on other approaches to build soft tree models.

Eventually, chapter 9 concludes this part of the thesis by a synthetic comparison of all decision tree variants which have been studied and analyzed throughout this part of the thesis.

All experiments of this part of the thesis are carried out on 7 classification problems which are described in Appendix B. While only synthetic results appear in the core of the text, Appendix C collects detailed results on all problems for experiments of all chapters of this part of the thesis.

Part III: Knowledge extraction from time series

The third part of the thesis is devoted to the temporal domain. It is divided into three chapters.

In Chapter 10, we give the motivations and the challenges in considering temporal data (and more generally complex data) with learning algorithms. We then extend the supervised learning framework of Chapter 2 to handle temporal attributes. Several classifications of the temporal problems are given and peculiarities of the time series classification problem are discussed. Eventually, literature on time series classification is briefly reviewed and the 6 test problems we used throughout our experiments are introduced. The details concerning the generation of these problems are reported to Appendix B.

Chapter 11 investigates the two approaches based on low-level features. First, classical algorithms are used with non temporal attributes extracted from time series by two simple sampling approaches. Then, the segment and combine algorithm is described and validated. The limitation of this approach is discussed in the conclusion of this chapter.

Chapter 12 studies our extension of decision trees to handle time series data. Decision tree tests are extended to detect local shift invariant properties or patterns in time series. A large part of this chapter is devoted to the description of the proposed algorithm. Some algorithmic details are reported in Appendix D. Several experiments are carried out to validate the method in terms of accuracy and interpretability. This chapter is concluded by a section on related work and a discussion of possible improvements of the method.

1.4 Contributions and originality

The aim of this section is to highlight the main contributions of this thesis. Each item is related to a small number of selected references in order to allow one to appraise our own contributions. A more extended discussion of related works in all domains is of course given in the body of the thesis. Our contributions which have already been published in the machine learning literature are also mentioned.

So, we believe that the principal contributions of this thesis are (in order of appearance in the main text):

- The quantitative analysis of decision tree variance in Chapter 5. Although we can find several references in the literature to the fact that decision trees are highly variable and also that the variance of the discretization thresholds is large (see for example [DK95] or [HTF01]), we have no knowledge of previous publications that ana-lyse and estimate quantitatively how important this variance is. The experiments of this thesis have been motivated by [Weh97]. A summary has been published in [GW00].

- The stabilization techniques for discretization thresholds introduced in Chapter 6. This work is orthogonal to previous works on discretization which focus exclusively on improvements of global characteristics of trees (complexity, predictive accuracy, or computational efficiency). These methods have been motivated by the work in [Weh97] and some of the results have been published in [GW00].

- Extra-tree algorithm of Chapter 7. Random decision trees are not new (see [Bre01] or [Ho95]). With respect to existing algorithms, our method goes one step further by randomizing both attributes and discretization thresholds. This algorithm originates from our analysis of decision tree variance. The idea of randomizing the attribute at tree nodes already appears in [Geu00].

- The general dual perturb and combine algorithm and its closed-form implementation in the context of decision tree induction presented in Chapter 6. This method has already been published in [Geu01a] and is described in a broader extent in [GOW01]. The idea of generating several predictions at the prediction stage has also appeared independently of our work in [DKN01] under the name "virtual test sample" approach. In the context of decision trees, this method is also related to soft tree models which have been extensively studied in automatic learning under different names.

- The derivation, from the perspective of the bias/variance tradeoff, of the segment and combine algorithm for time series classification in Chapter 11. A similar scheme for image classification has been proposed in [FH00a].

- Pattern extraction techniques for decision tree induction in Chapter 12. The approach is substantially different from existing algorithms for time series classification. A slightly less advanced version of this method has been published in [Geu01b].

All methods investigated in this thesis, except for the classical decision tree induction algorithm, k-NN, and multilayer perceptrons, have been implemented by the author of the thesis in Common Lisp as add-ons to the ATDIDT software which is an extensible data mining package developed at the University of Liège. Validation of improvements or new algorithms rely heavily on empirical studies. For information, the results of the second part of the thesis (gathered in Appendix C) have required the construction of about 100,000 decision trees. Notice that, while computational efficiency of automatic learning algorithms is very important, even in research, our focus has been more on the intrinsic features of the methods (such as algorithm complexity and scalability, and more importantly bias, variance, and accuracy) than on the efficient implementation of the algorithms. Therefore, we report only on few occasions CPU times, and we do this with the sole objective of providing some feeling about *relative* performances.