Dipartimento di Informatica e
Scienze dell'Informazione

Giorgio Valentini

Ensemble methods based on bias–variance analysis

Theses Series DISI-TH-2003-04

DISI, Università di Genova

v. Dodecaneso 35, 16146 Genova, Italy http://www.disi.unige.it/

Università degli Studi di Genova

Dipartimento di Informatica e
Scienze dell'Informazione

Dottorato di Ricerca in Informatica

Ph.D. Thesis in Computer Science

Giorgio Valentini

Ensemble methods based on bias–variance analysis

April, 2003

Dottorato di Ricerca in Informatica
Dipartimento di Informatica e Scienze dell'Informazione
Università degli Studi di Genova

DISI, Univ. di Genova
via Dodecaneso 35
I-16146 Genova, Italy
`http://www.disi.unige.it/`

Ph.D. Thesis in Computer Science

Submitted by Giorgio Valentini
DISI, Univ. di Genova
`valenti@disi.unige.it`

Date of submission: April 2003

Title: Ensemble methods based on bias–variance analysis

Advisor: Prof. Francesco Masulli
Dipartimento di Informatica, Università di Pisa, Italy
`masulli@di.unipi.it`

Ext. Reviewers:

Prof. Ludmila Kuncheva
School of Informatics,
University of Wales, Bangor, UK
`l.i.kuncheva@bangor.ac.uk`

Prof. Fabio Roli
Dipartimento di Ingegneria
Elettrica ed Elettronica
Università degli Studi di Cagliari, Italy
`roli@diee.unica.it`

Abstract

Ensembles of classifiers represent one of the main research directions in machine learning. Two main theories are invoked to explain the success of ensemble methods. The first one consider the ensembles in the framework of large margin classifiers, showing that ensembles enlarge the margins, enhancing the generalization capabilities of learning algorithms. The second is based on the classical bias–variance decomposition of the error, and it shows that ensembles can reduce variance and/or bias.

In accordance with this second approach, this thesis pursues a twofold purpose: on the one hand it explores the possibility of using bias–variance decomposition of the error as an analytical tool to study the properties of learning algorithms; on the other hand it explores the possibility of designing ensemble methods based on bias–variance analysis of the error.

At first, *bias–variance decomposition of the error* is considered *as a tool to analyze learning algorithms*. This work shows how to apply Domingos and James theories on bias–variance decomposition of the error to the analysis of learning algorithms. Extended experiments with Support Vector Machines (SVMs) are presented, and the analysis of the relationships between bias, variance, kernel type and its parameters provides a characterization of the error decomposition, offering insights into the way SVMs learn.

In a similar way *bias–variance analysis* is applied *as a tool to explain the properties of ensembles of learners*. A bias–variance analysis of ensembles based on resampling techniques is conducted, showing that, as expected, bagging is a variance reduction ensemble method, while the theoretical property of canceled variance holds only for Breiman's random aggregated predictors.

In addition to analyzing learning algorithms, *bias–variance analysis can offer guidance to the design of ensemble methods*. This work shows that it provides a theoretical and practical tool to develop new ensemble methods well-tuned to the characteristics of a specific base learner.

On the basis of the analysis and experiments performed on SVMs and bagged ensembles of SVMs, *new ensemble methods based on bias–variance analysis* are proposed. In particular *Lobag* (*Lo*w bias *bag*ging) selects low bias base learners and then combines them through bootstrap aggregating techniques. This approach affects both bias, through the selection of low bias base learners, and variance, through bootstrap aggregation of the selected low bias base learners. Moreover a new potential class of ensemble methods (*heterogeneous* *ensembles of SVMs*), that aggregate different SVM models on the basis of their bias–variance characteristics, is introduced.

From an *applicative standpoint* it is also shown that the proposed ensemble methods can be successfully applied to the *analysis of DNA microarray data.*

Al mio caro topo

Acknowledgements

I would like to thank Tom Dietterich for his friendly indirect support to my thesis. Indeed the main idea behind my thesis, that is applying bias–variance analysis as a tool to study learning algorithms and to develop new ensemble methods comes from Tom. Moreover I would like to thank him for the long discussions (especially by e-mail) about different topics related to my thesis, for his suggestions and constructive criticism.

I would like also to thank Franco Masulli: without his support and encouragement probably I would have not finish my Ph.D. in Computer Science. I thank him also for the large degree of freedom I enjoyed during my research activity.

Thanks to DISI and to the University of Genova for allowing me to pursue my Ph.D activity, and to the DISI people who helped me during my Ph.D activity.

Thanks to INFM, Istituto Nazionale di Fisica della Materia, especially for the financial support to my research activity and for the cluster of workstations that I used for my experiments.

Finally I would like to thank my wife Cristina, for bearing the too many saturdays and sundays with her husband on his papers, books and computers. This thesis is dedicated to her, even if I suspect that she wisely prefers her ancient potteries and archeological sites.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Ensembles of classifiers represent one of the main research directions in machine learning [43, 184].

The success of this emerging discipline is the result of the exchange and interactions between different cultural backgrounds and different perspectives of researchers from diverse disciplines, ranging from neural networks, to statistics, pattern recognition and soft computing, as reported by the recent international workshops on Multiple Classifier System organized by Josef Kittler and Fabio Roli [106, 107, 157].

Indeed empirical studies showed that both in classification and regression problems ensembles are often much more accurate than the individual base learners that make them up [8, 44, 63], and different theoretical explanations have been proposed to justify the effectiveness of some commonly used ensemble methods [105, 161, 111, 2].

Nonetheless, the variety of terms and specifications used to indicate sets of learning machines that work together to solve a machine learning problem [123, 192, 193, 105, 92, 30, 51, 12, 7, 60], reflects the absence of an unified theory on ensemble methods and the youngness of this research area.

A large number of combination schemes and ensemble methods have been proposed in literature. The combination by *majority voting* [103, 147], where the class most represented among the base classifiers is chosen, is probably the first ensemble method proposed in the literature, far before the first computers appeared [38]. A refinement of this approach is represented by a combination through *Bayesian decision rules*, where the class with the highest posterior probability computed through the estimated class conditional probabilities and the Bayes' formula is selected [172]. The base learners can also be aggregated using simple operators as *Minimum*, *Maximum*, *Average* and *Product* and *Ordered Weight Averaging* [160, 20, 117], and if we can interpret the classifier outputs as the support for the classes, fuzzy aggregation methods can be applied [30, 190, 118]. Other methods consist in training the combining rules, using second-level learning machines on top of the set of the base learners [55] or meta-learning techniques [25, 150]. Other classes of ensemble methods try to improve the overall accuracy of the ensemble by directly boosting the accuracy and the diversity of the base learners. For instance they can modify the structure and the characteristics of the available input data, as in *resampling* methods [15, 63, 161] or in *feature selection* [81] methods. They can also manipulate the aggregation of the classes, as in *Output Coding* methods [45, 46], or they can select base learners specialized for a specific input region, as in *mixture of experts* methods [98, 90]. Other approaches can inject randomness at different levels to the base learning algorithm [44, 19], or can select a proper set of base learners evaluating the performances of the component base learners, as in *test-and-select* methods [166, 156].

Despite the variety and the differences between the diverse classes of ensemble methods proposed in literature, they share a common characteristic: they emphasize in particular the combination scheme, or more in general the way the base learners are aggregated. Of course this is a fundamental aspect of ensemble methods, as it represents the main structural element of any ensemble of learning machines. Indeed ensemble methods have been conceived quite independently of the characteristics of specific base learners, emphasizing the combination scheme instead of the properties of the applied basic learning algorithm.

However, several researchers showed that the effectiveness of ensemble methods depends on the specific characteristics of the base learners; in particular on their individual accuracy, on the relationship between diversity and accuracy of the base learners [77, 119, 72, 121], on their stability [16], and on their general geometrical properties [32]. In other words, the analysis of the features and properties of the base learners used in ensemble methods is another important item for the design of ensemble methods [43]. Then we could try to develop ensemble methods well-tuned to the characteristics of specific base learners.

According to this this standpoint, this research starts from the "bottom edge" of ensemble methods: trying to exploit the features of base learning algorithms in order to build around them ensemble methods well-tuned to the learning characteristics of the base learners.

This requires the analysis of their learning properties, discovering and using appropriate tools to perform/execute this task. In principle, we could use measures of accuracy, diversity and complexity to study and characterize the behaviour of learning algorithms. However, as shown by L. Kuncheva [121], diversity may be related in a complex way to accuracy, and it may be very difficult to characterize the behaviour of a learning algorithm in terms of capacity/complexity of the resulting learning machine [188].

Decomposition of the error in bias and variance is a classical topic in statistical learning [52, 68]. Recently, Domingos proposed an unified theory on bias-variance analysis of the error, independent of the particular loss function [47], and James extended the Domingos approach, introducing the notions of bias and variance effect [94].

Using the theoretical tools provided by Domingos, we tried to evaluate if the analysis of bias and variance can provide insights into the way learning algorithms work. In principle, we could use the knowledge obtained from this analysis to design new learning algorithms, as suggested by Domingos himself [49], but in this work we did not follow this research line.

We used bias-variance analysis as a tool to study the behavior of learning methods, focusing in particular on Support Vector Machines [35]. A related issue of this work was also to evaluate if it was possible to characterize learning in terms of bias–variance decomposition of the error, studying the relationships between learning parameters and the bias and variance components of the error. We considered also if and how this analysis could be extended from specific learning algorithms to ensemble methods, trying to characterize also the behaviour of ensemble methods based on resampling techniques in terms of bias–variance decomposition of the error.

Besides studying if bias-variance theory offers a rationale to analyze the behaviour of learning algorithms and to explain the properties of ensembles of classifiers, the second main research topic of this thesis consists in researching if the decomposition of the error in bias and variance can also give guidance to the design of ensemble methods by relating measurable properties of learning algorithms to expected performances of ensembles [182].

On the basis of the knowledge gained from the bias variance analysis of specific learning algorithms, we tried to understand if it was possible to design new ensemble methods well-tuned to the bias–variance characteristics of specific base learners. Moreover we studied also if we could design ensemble methods with embedded bias–variance analysis procedures in order to take into account bias–variance characteristics of both the base learner and the ensemble. For instance, we researched if ensemble methods based on resampling techniques (e.g. bagging) could be enhanced through the bias–variance analysis approach, or if we could build variants of bagging exploiting the bias–variance characteristics of the base learners. This research line was motivated also by an applicative standpoint, in order to consider low-sized and high-dimensional classification problems in bioinformatics.

Outline of the thesis

Chapter 2 (*Ensemble methods*) introduces the main subject of this thesis into the the general framework of ensemble methods. It presents an overview of ensembles of learning machines, explaining the main reasons why they are able to outperform any single classifier within the ensemble, and proposing a taxonomy based on the main ways base classifiers can be generated or combined together. New directions in ensemble methods research are depicted, introducing ensemble methods well-tuned to the learning characteristics of the base learners.

The main goal of this work is twofold: on one hand it consists in evaluating if bias-variance analysis can be used as a tool to study and characterize learning algorithms and ensemble methods; on the other hand it consists in evaluating if the decomposition of the error into bias and variance can guide the design of ensemble methods by relating measurable properties of algorithms to the expected performances of ensembles. In both cases bias–variance theory plays a central role, and chapter 3 (*Bias–variance decomposition of the error*) summarizes the main topics of Domingos bias–variance theory. After a brief outline of the literature on bias–variance decomposition, the chapter focuses on bias–variance decomposition for the 0/1 loss, as we are mainly interested in classification problems. We underline that in this context bias variance is not additive, and we present an analysis of the combined effect of bias, variance and noise on the overall error, comparing also the theoretical approaches of Domingos and James on this topic. Moreover we consider the procedures to measure bias and variance, distinguishing the case when "large" or "small" data sets are available. In the latter case we propose to use out-of-bag procedures, as they are unbiased and computationally less expensive compared with multiple hold-out and cross-validation techniques.

Chapter 4 (*Bias–variance analysis in single SVMs*) presents an experimental bias–variance analysis in SVMs. The main goal of this chapter is to study the learning properties of SVMs with respect to their bias–variance characteristics and to characterize their learning behavior. In particular this analysis gets insights into the way SVMs learn, unraveling the specific effects of bias, unbiased and biased variance on the overall error. Firstly, the experimental set-up, involving training and testing of more than half-million of different SVMs, using gaussian, polynomial and dot-product kernels, is presented. Then we analyzed the relationships of bias–variance decomposition with different kernels, regularization and kernel parameters, using both synthetic and "real world" data. In particular, with gaussian kernels we studied the reasons why usually SVMs do not learn with small values of the spread parameter, their behavior with large values of the spread, and the relationships between generalization error, training error, number of support vectors and capacity. We provided also a characterization of bias–variance decomposition of the error in gaussian kernels, distinguishing three main regions characterized by specific trends of bias and variance with respect to the values of the spread parameter σ. Similar characterizations were provided also for polynomial and dot-product kernels.

Bias-variance can also be a useful tool to analyze bias–variance characteristics of ensemble methods. To this purpose chapter 5 (*Bias–variance analysis in random aggregated and bagged ensembles of SVMs*) provides an extended experimental analysis of bias–variance decomposition of the error for ensembles based on resampling techniques. We consider theoretical issues about the relationships between random aggregating and bagging. Indeed bagging can be seen as an approximation of *random aggregating*, that is a process by which base learners, trained on samples drawn accordingly to an unknown probability distribution from the entire universe population, are aggregated through majority voting (classification) or averaging between them (regression). Breiman showed that random aggregating and bagging are effective with unstable learning algorithms, that is when small changes in the training set can result in large changes in the predictions of the base learners; we prove that there is a strict relationship between instability and the variance of the base predictors. Theoretical analysis shows that random aggregating should significantly reduce variance, without incrementing bias. Bagging also, as an approximation of random aggregating, should reduce variance. We performed an extended experimental analysis, involving training and testing of about 10 million SVMs, to test these theoretical outcomes. Moreover, we analyzed bias–variance in bagged and random aggregated SVM ensembles, to understand the effect of bagging and random aggregating on bias and variance components of the error in SVMs. In both cases we evaluated for each kernel the expected error and its decomposition in bias, net-variance, unbiased and biased variance with respect to the learning parameters of the base learners. Then we analyzed the bias–variance decomposition as a function of the number of the base learners employed. Finally, we compared bias and variance with respect to the learning parameters in random aggregated and bagged SVM ensembles and in the corresponding single SVMs, in order to study the effect of bagging and random aggregating on the bias and variance components of the error. With random aggregated ensembles we registered a very large reduction of the net-variance with respect to single SVMs. It was always reduced close to 0, independently of the type of kernel used. This behaviour is due primarily to the unbiased variance reduction, while the bias remains unchanged with respect to the single SVMs. With bagging we have also a reduction of the error, but not as large as with random aggregated ensembles. Indeed, unlike random aggregating, net and unbiased variance, although reduced, are not actually reduced to 0, while bias remains unchanged or slightly increases. An interesting byproduct of this analysis is that *undersampled bagging* can be viewed as another approximation of random aggregating (using a bootstrap approximation of the unknown probability distribution), if we consider the universe U as a data set from which undersampled data, that is data sets whose cardinality is much less than the cardinality of U, are randomly drawn with replacement. This approach should provide very significant reduction of the variance and could be in practice applied to data mining problems, when learning algorithms cannot comfortably manage very large data sets.

In addition to providing insights into the behavior of learning algorithms, the analysis of the bias–variance decomposition of the error can identify the situations in which ensemble methods might improve base learner performances. Indeed the decomposition of the error into bias and variance can guide the design of ensemble methods by relating measurable properties of algorithms to the expected performance of ensembles. Chapter 6 (*SVM ensemble methods based on bias–variance analysis*), presents two possible ways of applying bias–variance analysis to develop SVM-based ensemble methods. The first approach tries to apply bias–variance analysis to enhance both accuracy and diversity of the base learners. The second research direction consists in bootstrap aggregating low bias base learners in order to lower both bias and variance. Regarding the first approach, only some very general research lines are depicted. About the second direction, a specific new method that we named *Lobag*, that is *Lo*w bias *bag*ged SVMs, is introduced, considering also different variants. Lobag applies bias–variance analysis in order to direct the tuning of Support Vector Machines toward the optimization of the performance of bagged ensembles. Specifically, since bagging is primarily a variance-reduction method, and since the overall error is (to a first approximation) the sum of bias and variance, this suggests that SVMs should be tuned to minimize bias before being combined by bagging. The key-issue of this methods consists in efficiently evaluating the bias–variance decomposition of the error. We embed this procedure inside the Lobag ensemble method implementing a relatively inexpensive out-of-bag estimate of bias and variance. The pseudocode of Lobag is provided, as well as a C++ implementation (available on-line). Numerical experiments show that Lobag compares favorably with bagging, and some preliminary results show that it can be successfully applied to DNA microarray data analysis.

The conclusions summarize the main results achieved, and several open questions delineate possible future works and developments.

Chapter 2

Ensemble methods

Ensembles are sets of learning machines whose decisions are combined to improve the performance of the overall system. In this last decade one of the main research areas in machine learning has been represented by methods for constructing ensembles of learning machines. Although in the literature [123, 192, 193, 105, 92, 30, 51, 12, 7, 60] a plethora of terms, such as committee, classifier fusion, combination, aggregation and others are used to indicate sets of learning machines that work together to solve a machine learning problem, in this paper we shall use the term *ensemble* in its widest meaning, in order to include the whole range of combining methods. This variety of terms and specifications reflects the absence of an unified theory on ensemble methods and the youngness of this research area. However, the great effort of the researchers, reflected by the amount of the literature [167, 106, 107, 157] dedicated to this emerging discipline, achieved meaningful and encouraging results.

Empirical studies showed that both in classification and regression problem ensembles are often much more accurate than the individual base learner that make them up [8, 44, 63], and recently different theoretical explanations have been proposed to justify the effectiveness of some commonly used ensemble methods [105, 161, 111, 2].

The interest in this research area is motivated also by the availability of very fast computers and networks of workstations at a relatively low cost that allow the implementation and the experimentation of complex ensemble methods using off-the-shelf computer platforms. However, as explained in Sect. 2.1 there are deeper reasons to use ensembles of learning machines. motivated by the intrinsic characteristics of the ensemble methods.

This chapter presents a brief overview of the main areas of research, without pretending to be exhaustive or to explain the detailed characteristics of each ensemble method.

2.1 Reasons for Combining Multiple Learners

Both empirical observations and specific machine learning applications confirm that a given learning algorithm outperforms all others for a specific problem or for a specific subset of the input data, but it is unusual to find a single expert achieving the best results on the overall problem domain. As a consequence multiple learner systems try to exploit the local different behavior of the base learners to enhance the accuracy and the reliability of the overall inductive learning system. There are also hopes that if some learner fails, the overall system can recover the error. Employing multiple learners can derive from the application context, such as when multiple sensor data are available, inducing a natural decomposition of the problem. In more general cases we can dispose of different training sets, collected at different times, having eventually different features and we can use different specialized learning machine for each different item.

However, there are deeper reasons why ensembles can improve performances with respect to a single learning machine. As an example, consider the following one given by Tom Dietterich in [43]. If we have a dichotomic classification problem and L hypotheses whose error is lower than 0.5, then the resulting majority voting ensemble has an error lower than the single classifier, as long as the error of the base learners are uncorrelated. In fact, if we have 21 classifiers, and the error rates of each base learner are all equal to $p = 0.3$ and the errors are independent, the overall error of the majority voting ensemble will be given by the area under the binomial distribution where more than $L/2$ hypotheses are wrong:

$$P_{error} = \sum_{(i=\lceil L/2 \rceil)}^{L} \begin{pmatrix} L \\ i \end{pmatrix} p^i (1-p)^{L-i} \quad \Rightarrow P_{error} = 0.026 \ll p = 0.3$$

This result has been studied by mathematicians since the end of the XVIII century in the context of social sciences: in fact the *Condorcet Jury Theorem* [38]) proved that the judgment of a committee is superior to those of individuals, provided the individuals have reasonable competence (that is, a probability of being correct higher than 0.5). As noted in [122], this theorem theoretically justifies recent research on multiple "weak" classifiers [95, 81, 110], representing an interesting research direction diametrically opposite to the development of highly accurate and specific classifiers.

This simple example shows also an important issue in the design of ensembles of learning machines: the effectiveness of ensemble methods relies on the independence of the error committed by the component base learner. In this example, if the independence assumption does not hold, we have no assurance that the ensemble will lower the error, and we know that in many cases the errors are correlated. From a general standpoint we know that the effectiveness of ensemble methods depends on the *accuracy* and the *diversity* of the base learners, that is if they exhibit low error rates and if they produce different errors

[78, 174, 132]. The correlated concept of independence between the base learners has been commonly regarded as a requirement for effective classifier combinations, but Kuncheva and Whitaker have recently shown that not always independent classifiers outperform dependent ones [121]. In fact there is a trade-off between accuracy and independence: more accurate are the base learners, less independent they are.

Learning algorithms try to find an hypothesis in a given space $\mathcal{H}$ of hypotheses, and in many cases if we have sufficient data they can find the optimal one for a given problem. But in real cases we have only limited data sets and sometimes only few examples are available. In these cases the learning algorithm can find different hypotheses that appear equally accurate with respect to the available training data, and although we can sometimes select among them the simplest or the one with the lowest capacity, we can avoid the problem averaging or combining them to get a good approximation of the unknown true hypothesis.

Another reason for combining multiple learners arises from the limited representational capability of learning algorithms. In many cases the unknown function to be approximated is not present in $\mathcal{H}$, but a combination of hypotheses drawn from $\mathcal{H}$ can expand the space of representable functions, embracing also the true one. Although many learning algorithms present universal approximation properties [86, 142], with finite data sets these asymptotic features do not hold: the effective space of hypotheses explored by the learning algorithm is a function of the available data and it can be significantly smaller than the virtual $\mathcal{H}$ considered in the asymptotic case. From this standpoint ensembles can enlarge the effective hypotheses coverage, expanding the space of representable functions.

Many learning algorithms apply local optimization techniques that may get stuck in local optima. For instance inductive decision trees employ a greedy local optimization approach, and neural networks apply gradient descent techniques to minimize an error function over the training data. Moreover optimal training with finite data both for neural networks and decision trees is NP-hard [13, 88]. As a consequence even if the learning algorithm can in principle find the best hypothesis, we actually may not be able to find it. Building an ensemble using, for instance, different starting points may achieve a better approximation, even if no assurance of this is given.

Another way to look at the need for ensembles is represented by the classical bias–variance analysis of the error [68, 115]: different works have shown that several ensemble methods reduce variance [15, 124] or both bias and variance [15, 62, 114]. Recently the improved generalization capabilities of different ensemble methods have also been interpreted in the framework of the theory of large margin classifiers [129, 162, 2], showing that methods such as boosting and ECOC enlarge the margins of the examples.

2.2 Ensemble Methods Overview

A large number of combination schemes and ensemble methods have been proposed in literature. Combination techniques can be grouped and analyzed in different ways, depending on the main classification criterion adopted. If we consider the representation of the input patterns as the main criterion, we can identify two distinct large groups, one that uses the same and one that uses different representations of the inputs [104, 105].

Assuming the architecture of the ensemble as the main criterion, we can distinguish between serial, parallel and hierarchical schemes [122], and if the base learners are selected or not by the ensemble algorithm we can separate selection-oriented and combiner-oriented ensemble methods [92, 118]. In this brief overview we adopt an approach similar to the one cited above, in order to distinguish between *non-generative* and *generative* ensemble methods. Non-generative ensemble methods confine themselves to combine a set of given possibly well-designed base learners: they do not actively generate new base learners but try to combine in a suitable way a set of existing base classifiers. Generative ensemble methods generate sets of base learners acting on the base learning algorithm or on the structure of the data set and try to actively improve diversity and accuracy of the base learners.

Note that in some cases it is difficult to assign a specific ensemble method to either of the proposed general superclasses: the purpose of this general taxonomy is simply to provide a general framework for the main ensemble methods proposed in the literature.

2.2.1 Non-generative Ensembles

This large group of ensemble methods embraces a large set of different approaches to combine learning machines. They share the very general common property of using a predetermined set of learning machines previously trained with suitable algorithms. The base learners are then put together by a combiner module that may vary depending on its adaptivity to the input patterns and on the requirement of the output of the individual learning machines.

The type of combination may depend on the type of output. If only labels are available or if continuous outputs are hardened, then *majority voting*, that is the class most represented among the base classifiers, is used [103, 147, 124].

This approach can be refined assigning different weights to each classifier to optimize the performance of the combined classifier on the training set [123], or, assuming mutual independence between classifiers, a *Bayesian decision rule* selects the class with the highest posterior probability computed through the estimated class conditional probabilities and the Bayes' formula [193, 172]. A Bayesian approach has also been used in *Consensus based classification* of multisource remote sensing data [10, 9, 21], outperforming conventional multivariate methods for classification. To overcome the problem of the independence assumption (that is unrealistic in most cases), the Behavior-Knowledge Space (BKS) method [87] considers each possible combination of class labels, filling a look-up table using the available data set, but this technique requires a large volume of training data.

Where we interpret the classifier outputs as the support for the classes, fuzzy aggregation methods can be applied, such as simple connectives between fuzzy sets or the fuzzy integral [30, 29, 100, 190]; if the classifier outputs are possibilistic, *Dempster-Schafer* combination rules can be applied [154]. Statistical methods and similarity measures to estimate classifier correlation have also been used to evaluate expert system combination for a proper design of multi-expert systems [89].

The base learners can also be aggregated using simple operators as *Minimum*, *Maximum*, *Average* and *Product* and *Ordered Weight Averaging* and other statistics [160, 20, 117, 155]. In particular, on the basis of a common bayesian framework, Josef Kittler provided a theoretical underpinning of many existing classifier combination schemes based on the product and the sum rule, showing also that the sum rule is less sensitive to the errors of subsets of base classifiers [105].

Recently Ludmila Kuncheva has developed a global combination scheme that takes into account the decision profiles of all the ensemble classifiers with respect to all the classes, designing *Decision templates* that summarize in matrix format the average decision profiles of the training set examples. Different similarity measures can be used to evaluate the matching between the matrix of classifier outputs for an input x, that is the decision profiles referred to x, and the matrix templates (one for each class) found as the class means of the classifier outputs [118]. This general fuzzy approach produce soft class labels that can be seen as a generalization of the conventional crisp and probabilistic combination schemes.

Another general approach consists in explicitly *training combining rules*, using second-level learning machines on top of the set of the base learners [55, 191]. This stacked structure makes use of the outputs of the base learners as features in the intermediate space: the outputs are fed into a second-level machine to perform a trained combination of the base learners.

Meta-learning techniques can be interpreted as an extension of the previous approach [25, 26]. Indeed they can be defined as learning from learned knowledge and are characterized by meta-level training sets generated by the first level base learners trained on the "true" data set, and a meta-learner trained from the meta-level training set [150]. In other words, in meta-learning the integration rule is learned by the meta-learner on the basis of the behavior of the trained base learners.

2.2.2 Generative Ensembles

Generative ensemble methods try to improve the overall accuracy of the ensemble by directly boosting the accuracy and the diversity of the base learner. They can modify the structure and the characteristics of the available input data, as in *resampling* methods or in *feature selection* methods, they can manipulate the aggregation of the classes (*Output Coding* methods), can select base learners specialized for a specific input region (*mixture of experts* methods), can select a proper set of base learners evaluating the performance and the characteristics of the component base learners (*test-and-select* methods) or can randomly modify the base learning algorithm (*randomized* methods).

2.2.2.1 Resampling methods

Resampling techniques can be used to generate different hypotheses. For instance, bootstrapping techniques [56] may be used to generate different training sets and a learning algorithm can be applied to the obtained subsets of data in order to produce multiple hypotheses. These techniques are effective especially with unstable learning algorithms, which are algorithms very sensitive to small changes in the training data, such as neural-networks and decision trees.

In *bagging* [15] the ensemble is formed by making bootstrap replicates of the training sets, and then multiple generated hypotheses are used to get an aggregated predictor. The aggregation can be performed averaging the outputs in regression or by majority or weighted voting in classification problems [169, 170].

While in bagging the samples are drawn with replacement using a uniform probability distribution, in *boosting* methods the learning algorithm is called at each iteration using a different distribution or weighting over the training examples [160, 63, 161, 62, 164, 159, 50, 61, 51, 50, 17, 18, 65, 64]. This technique places the highest weight on the examples most often misclassified by the previous base learner: in this way the base learner focuses its attention on the hardest examples. Then the boosting algorithm combines the base rules taking a weighted majority vote of the base rules. Schapire and Singer showed that the training error exponentially drops down with the number of iterations [163] and Schapire et al. [162] proved that boosting enlarges the margins of the training examples, showing also that this fact translates into a superior upper bound on the generalization error. Experimental work showed that bagging is effective with noisy data, while boosting, concentrating its efforts on noisy data seems to be very sensitive to noise [153, 44]. Recently, variants of boosting, specific for noisy data, have been proposed by several authors [152, 39].

Another resampling method consists in constructing training sets by leaving out disjoint subsets of the training data as in *cross-validated committees* [143, 144] or sampling without replacement [165].

Another general approach, named *Stochastic Discrimination* [109, 110, 111, 108], is based on randomly sampling from a space of subsets of the feature space underlying a given problem, then combining these subsets to form a final classifier, using a set-theoretic abstraction to remove all the algorithmic details of classifiers and training procedures. By this approach the classifiers' decision regions are considered only in form of point sets, and the set of classifiers is just a sample into the power set of the feature space. A rigorous mathematical treatment starting from the "representativeness" of the examples used in machine learning problems leads to the design of ensemble of weak classifiers, whose accuracy is governed by the law of large numbers [27].

2.2.2.2 Feature selection methods

This approach consists in reducing the number of input features of the base learners, a simple method to fight the effects of the classical curse of dimensionality problem [66]. For instance, in the *Random Subspace Method* [81, 119], a subset of features is randomly selected and assigned to an arbitrary learning algorithm. This way, one obtains a random subspace of the original feature space, and constructs classifiers inside this reduced subspace. The aggregation is usually performed using weighted voting on the basis of the base classifiers accuracy. It has been shown that this method is effective for classifiers having a decreasing learning curve constructed on small and critical training sample sizes [168]

The *Input Decimation* approach [175, 139] reduces the correlation among the errors of the base classifiers, decoupling the base classifiers by training them with different subsets of the input features. It differs from the previous Random Subspace Method as for each class the correlation between each feature and the output of the class is explicitly computed, and the base classifier is trained only on the most correlated subset of features.

Feature subspace methods performed by partitioning the set of features, where each subset is used by one classifier in the team, are proposed in [193, 141, 20]. Other methods for combining different feature sets using genetic algorithms are proposed in [118, 116]. Different approaches consider feature sets obtained by using different operators on the original feature space, such as Principal Component Analysis, Fourier coefficients, Karhunen-Loewe coefficients, or other [28, 55]. An experiment with a systematic partition of the feature space, using nine different combination schemes is performed in [120], showing that there are no "best" combinations for all situations and that there is no assurance that in all cases a classifier team will outperform the single best individual.

2.2.2.3 Mixtures of experts methods

The recombination of the base learners can be governed by a supervisor learning machine, that selects the most appropriate element of the ensemble on the basis of the available input data. This idea led to the *mixture of experts* methods [91, 90], where a gating network performs the division of the input space and small neural networks perform the effective calculation at each assigned region separately. An extension of this approach is the *hierarchical mixture of experts* method, where the outputs of the different experts are non-linearly combined by different supervisor gating networks hierarchically organized [97, 98, 90].

Cohen and Intrator extended the idea of constructing local simple base learners for different regions of input space, searching for appropriate architectures that should be locally used and for a criterion to select a proper unit for each region of input space [31, 32]. They proposed a hybrid MLP/RBF network by combining RBF and Perceptron units in the same hidden layer and using a forward selection approach [58] to add units until a desired error is reached. Although the resulting *Hybrid Perceptron/Radial Network* is not in a strict sense an ensemble, the way by which the regions of the input space and the computational units are selected and tested could be in principle extended to ensembles of learning machines.

2.2.2.4 Output Coding decomposition methods

Output Coding (OC) methods decompose a multiclass–classification problem in a set of two-class subproblems, and then recompose the original problem combining them to achieve the class label [134, 130, 43]. An equivalent way of thinking about these methods consists in encoding each class as a bit string (named codeword), and in training a different two-class base learner (dichotomizer) in order to separately learn each codeword bit. When the dichotomizers are applied to classify new points, a suitable measure of similarity between the codeword computed by the ensemble and the codeword classes is used to predict the class.

Different *decomposition schemes* have been proposed in literature: In the One-Per-Class (OPC) decomposition [4], each dichotomizer f_i has to separate a single class from all others; in the *PairWise Coupling* (PWC) decomposition [79], the task of each dichotomizer f_i consists in separating a class C_i form class C_j, ignoring all other classes; the *Correcting Classifiers* (CC) and the *PairWise Coupling Correcting Classifiers* (PWC-CC) are variants of the PWC decomposition scheme, that reduce the noise originated in the PWC scheme due to the processing of non pertinent information performed by the PWC dichotomizers [137].

Error Correcting Output Coding [45, 46] is the most studied OC method, and has been successfully applied to several classification problems [1, 11, 69, 6, 177, 194]. This decomposition method tries to improve the error correcting capabilities of the codes generated by the decomposition through the maximization of the minimum distance between each couple of codewords [114, 130]. This goal is achieved by means of the redundancy of the coding scheme [187].

ECOC methods present several open problems. The tradeoff between error recovering capabilities and complexity/learnability of the dichotomies induced by the decomposition scheme has been tackled in several works [2, 176], but an extensive experimental evaluation of the tradeoff has to be performed in order to achieve a better understanding of this phenomenon. A related problem is the analysis of the relationship between codeword length and performances: some preliminary results seem to show that long codewords improve performance [69]. Another open problem, not sufficiently investigated in literature [69, 131, 11], is the selection of optimal dichotomic learning machines for the decomposition unit. Several methods for generating ECOC codes have been proposed: exhaustive codes, randomized hill climbing [46], random codes [93], and Hadamard and BCH codes [14, 148]. An open problems is still the joint maximization of distances between rows and columns in the decomposition matrix. Another open problem consists in designing codes for a given multiclass problem. An interesting greedy approach is proposed in [134], and a method based on soft weight sharing to learn error correcting codes from data is presented in [3]. In [36] it is shown that given a set of dichotomizers the problem of finding an optimal decomposition matrix is NP-complete: by introducing continuous codes and casting the design problem of continuous codes as a constrained optimization problem, we can achieve an optimal continuous decomposition using standard optimization methods.

The work in [131] highlights that the effectiveness of ECOC decomposition methods depends mainly on the design of the learning machines implementing the decision units, on the similarity of the ECOC codewords, on the accuracy of the dichotomizers, on the complexity of the multiclass learning problem and on the correlation of the codeword bits. In particular, Peterson and Weldon [148] showed that if errors on different code bits are dependent, the effectiveness of error correcting code is reduced. Consequently, if a decomposition matrix contains very similar rows (dichotomies), each error of an assigned dichotomizer will be likely to appear in the most correlated dichotomizers, thus reducing the effectiveness of ECOC. These hypotheses have been experimentally supported by a quantitative evaluation of the dependency among output errors of the decomposition unit of ECOC learning machines using mutual information based measures [132, 133].

2.2.2.5 Test and select methods

The *test and select* methodology relies on the idea of selection in ensemble creation [166]. The simplest approach is a greedy one [147], where a new learner is added to the ensemble only if the resulting squared error is reduced, but in principle any optimization technique can be used to select the "best" component of the ensemble, including genetic algorithms [138].

It should be noted that the time complexity of the selection of optimal subsets of classifiers is exponential with respect to the number of base learners used. From this point of view heuristic rules, as the "choose the best" or the "choose the best in the class", using classifiers of different types strongly reduce the computational complexity of the selected phase, as the evaluation of different classifier subsets is not required [145]. Moreover test and select methods implicitly include a "production stage", by which a set of classifiers must be generated.

Different selection methods based on different search algorithm mututated from feature selection methods (forward and backward search) or for the solution of complex optimization tasks (tabu search) are proposed in [156]. Another interesting approach uses clustering methods and a measure of diversity to generate sets of diverse classifiers combined by majority voting, selecting the ensemble with the highest performance [72]. Finally, *Dynamic Classifier Selection* methods [85, 192, 71] are based on the definition of a function selecting for each pattern the classifier which is probably the most accurate, estimating, for instance the accuracy of each classifier in a local region of the feature space surrounding an unknown test pattern [71, 74, 73].

2.2.2.6 Randomized ensemble methods

Injecting randomness into the learning algorithm is another general method to generate ensembles of learning machines. For instance, if we initialize with random values the initial weights in the backpropagation algorithm, we can obtain different learning machines that can be combined into an ensemble [113, 143].

Several experimental results showed that randomized learning algorithms used to generate base elements of ensembles improve the performances of single non-randomized classifiers. For instance in [44] randomized decision tree ensembles outperform single C4.5 decision trees [151], and adding gaussian noise to the data inputs, together with bootstrap and weight regularization can achieve large improvements in classification accuracy [153].

2.3 New directions in ensemble methods research

Ensemble methods are one of the most increasing research topic in machine learning. Without pretending to be exhaustive, here are summarized some new directions in ensemble method research, emphasizing those topics most related to my research interests.

Ensemble methods have been developed in classification and regression settings, but there are very few approaches proposed for unsupervised clustering problems. For instance, a multiple k-means method combines multiple approximate k-means solution to obtain a final set of cluster centers [59], and cluster ensembles based on partial sets of features or multiple views of data have been applied to data mining problems and to structure rules in a knowledge base [99, 136]. Recently a new interesting research direction has been proposed for unsupervised clustering problems [70, 171]. According to this approach multiple partitioning of a set of objects, obtained from different clustering algorithms or different instances of the same clustering algorithm, are combined without accessing the original input features, but using only the cluster labels provided by the applied clusterers. Then the "optimal" partition labeling is selected as the one that maximizes the mutual information with respect to all the provided labelings. This approach should also permit to integrate both different clustering algorithms and views of data, exploiting heterogeneous resources and data available in distributed environments.

Another research direction for ensemble methods could be represented by ensemble methods specific for feature selection. Indeed, if only small sized samples are available, ensemble methods could provide robust estimates of sets of features correlated with the output of a learning machines: several applications, for instance in bioinformatics, could take advantage of this approach.

Following the spirit of Breiman's random forests [19], we could use randomness at different levels to improve performances of ensemble methods. For instance, we know that random selection of input samples combined with random selection of features improve the performance of random forests. This approach could be in principle extended to other base learners. Moreover we could also extend this approach to other types of randomness, as the strong law of large numbers assures the convergence and no overfitting problems incrementing the number of base learners [19]. For instance we could design "forests", or, more appropriately in this context, nets of neural networks, exploring suitable ways to inject randomness in building ensembles, extending the original Breiman's approach "limited only" to random input and features.

Two main theories are invoked to explain the success of ensemble methods. The first one consider the ensembles in the framework of large margin classifiers [129], showing that ensembles enlarge the margins, enhancing the generalization capabilities of learning algorithms [162, 2]. The second is based on the the classical bias–variance decomposition of the error [68], and it shows that ensembles can reduce variance [16] and also bias [114].

Recently Domingos proved that Schapire's notion of margins [162] can be expressed in terms of bias and variance and viceversa [49], and hence Schapire's bounds of ensemble's generalization error can be equivalently expressed in terms of the distribution of the margins or in terms of the bias–variance decomposition of the error, showing the equivalence of margin-based and bias–variance-based approaches.

Despite these important results, most of the theoretical problems behind ensemble methods remain opened, and we need more research work to understand the characteristics and generalization capabilities of ensemble methods.

For instance, a substantially unexplored research field is represented by the analysis of the relationships between ensemble methods and data complexity [126]. The papers of Tin Kam Ho [82, 83, 84] represent a fundamental starting point to explore the relationships between ensemble methods (and more generally learning algorithms) and data complexity in order to characterize ensemble methods with respect to the specific properties of the data. Extending this approach we could also try to design ensemble methods well-tuned to the data characteristics, embedding analysis of data complexity and/or the evaluation of the geometrical or topological data characteristics into the ensemble method itself. An interesting step in this direction is represented bu the research of Cohen and Intrator [31, 33]. Even if they use a single learning machine composed by heterogeneous radial and sigmoidal units to properly fit geometrical data characteristics, their approach can be in principle extended to heterogeneous ensembles of learning machines.

From a different standpoint we could also try to develop ensemble methods well-tuned to the the characteristics of specific base learners. Usually ensemble methods have been conceived quite independently of the characteristics of specific base learners, emphasizing the combination scheme instead of the properties of the applied basic learning algorithm. Hence, a promising research line could consist in characterizing the properties of a specific base learner, building around it an ensemble method well-tuned to the learning characteristics of the base learner itself. Toward this research direction, bias-variance analysis [47] could in principle be used to characterize the properties of learning algorithms in order to design ensemble methods well-tuned to the bias–variance characteristics of a specific base learner [182].

Chapter 3

Bias–variance decomposition of the error

Our purpose consists in evaluating if bias–variance analysis can be used to characterize the behavior of learning algorithms and to tune the individual base classifiers so as to optimize the overall performance of the ensemble. As a consequence, to pursue these goals, we considered the different approaches and theories proposed in the literature, and in particular we propose a very general approach applicable to any loss function and in particular to the 0/1 loss [47], as explained below in this chapter.

Historically, the bias–variance insight was borrowed from the field of regression, using squared–loss as the loss function [68]. For classification problems, where the 0/1 loss is the main criterion, several authors proposed bias–variance decompositions related to 0/1 loss.

Kong and Dietterich [114] proposed a bias–variance decomposition in the context of ECOC ensembles [46], but their analysis is extensible to arbitrary classifiers, even if they defined variance simply as a difference between loss and bias.

In Breiman's decomposition [16] bias and variance are always non-negative (while Dietterich definition allows a negative variance), but at any input the reducible error (i.e. the total error rate less noise) is assigned entirely to variance if the classification is unbiased, and to bias if biased. Moreover he forced the decomposition to be purely additive, while for the 0/1 loss this is not the case. Kohavi and Wolpert approach [112] produced a biased estimation of bias and variance, assigning a non-zero bias to a Bayes classifier, while Tibshirani [173] did not use directly the notion of variance, decomposing the 0/1 loss in bias and an unrelated quantity he called "aggregation effect", which is similar to the James' notion of *variance effect* [94].

Friedman [66] showed that in classification problems, bias and variance are not purely additive: in some cases increasing variance increases the error, but in other cases can also reduce the error, especially when the prediction is biased.

Heskes [80] proposed a bias-variance decomposition using the Kullback-Leibler divergence as loss function. By this approach the error between the target and the predicted classifier densities is measured; anyway when he tried to extend this approach to the zero-one function interpreted as the limit case of log-likelihood type error, the resulting decomposition produces a definition of bias that losses his natural interpretation as systematic error committed by the classifier.

As briefly outlined, these decompositions suffer of significant shortcomings: in particular they lose the relationship to the original squared loss decomposition, forcing in most cases bias and variance to be purely additive.

We consider classification problems and the 0/1 loss function in the Domingos' unified framework of bias–variance decomposition of the error [49, 48]. In this approach bias and variance are defined for an arbitrary loss function, showing that the resulting decomposition specializes to the standard one for squared loss, but it holds also for the 0/1 loss [49].

A similar approach has been proposed by James [94]: he extended the notion of variance and bias for general loss functions, distinguishing also between bias and variance, interpreted respectively as the systematic error and the variability of an estimator, and the the actual effect of bias and variance on the error.

In the rest of this chapter we consider Domingos and James bias–variance theory, focusing on bias–variance for the 0/1 loss. Moreover we show how to measure bias and variance of the error in classification problems, suggesting diverse approaches for respectively "large" or "small" data sets.

3.1 Bias–Variance Decomposition for the 0/1 loss function

The analysis of bias–variance decomposition of the error has been originally developed in the standard regression setting, where the squared error is usually used as loss function. Considering a prediction $y = f(x)$ of an unknown target t, provided by a learner f on input x, with $\mathbf{x} \in \mathbb{R}^d$ and $y \in \mathbb{R}$, the classical decomposition of the error in bias and variance for the squared error loss is [68]:

$$\begin{aligned} E_{y,t}[(y-t)^2] &= E_t[(t-E[t])^2] + E_y[(y-E[y])^2] + (E[y]-E[t])^2 \\ &= Noise(t) + Var(y) + Bias^2(y) \end{aligned}$$

In words, the expected loss of using y to predict t is the sum of the variances of t (noise) and y plus the squared bias. $E_y[\cdot]$ indicates the expected value with respect to the distribution of the random variable y.

This decomposition cannot be automatically extended to the standard classification setting, as in this context the 0/1 loss function is usually applied, and bias and variance are not purely additive. As we are mainly interested in analyzing bias–variance for classification problems, we introduce the bias–variance decomposition for the 0/1 loss function, according to the Domingos unified bias–variance decomposition of the error [48].

3.1.1 Expected loss depends on the randomness of the training set and the target

Consider a (potentially infinite) population U of labeled training data points, where each point is a pair $(\mathbf{x}_j, t_j)$, $t_j \in \mathcal{C}$, $\mathbf{x}_j \in \mathbb{R}^d$, $d \in \mathbb{N}$, where $\mathcal{C}$ is the set of the class labels. Let $P(\mathbf{x}, t)$ be the joint distribution of the data points in U. Let D be a set of m points drawn identically and independently from U according to P. We think of D as being the training sample that we are given for training a classifier. We can view D as a random variable, and we will let $E_D[\cdot]$ indicate the expected value with respect to the distribution of D.

Let $\mathcal{L}$ be a learning algorithm, and define $f_D = \mathcal{L}(D)$ as the classifier produced by $\mathcal{L}$ applied to a training set D. The model produces a prediction $f_D(\mathbf{x}) = y$. Let $L(t, y)$ be the 0/1 loss function, that is $L(t, y) = 0$ if $y = t$, and $L(t, y) = 1$ otherwise.

Suppose we consider a fixed point $\mathbf{x} \in \mathbb{R}^d$. This point may appear in many labeled training points in the population. We can view the corresponding labels as being distributed according to the conditional distribution $P(t|\mathbf{x})$. Recall that it is always possible to factor the joint distribution as $P(\mathbf{x}, t) = P(\mathbf{x})P(t|\mathbf{x})$. Let $E_t[\cdot]$ indicate the expectation with respect to t drawn according to $P(t|\mathbf{x})$.

Suppose we consider a *fixed* predicted class y for a given $\mathbf{x}$. This prediction will have an expected loss of $E_t[L(t, y)]$. In general, however, the prediction y is not fixed. Instead, it is computed from a model f_D which is in turn computed from a training sample D.

Hence, the expected loss EL of learning algorithm $\mathcal{L}$ at point $\mathbf{x}$ can be written by considering both the randomness due to the choice of the training set D and the randomness in t due to the choice of a particular test point $(\mathbf{x}, t)$:

$$EL(\mathcal{L}, \mathbf{x}) = E_D[E_t[L(t, f_D(\mathbf{x}))]],$$

where $f_D = \mathcal{L}(D)$ is the classifier learned by $\mathcal{L}$ on training data D. The purpose of the bias-variance analysis is to decompose this expected loss into terms that separate the bias and the variance.

3.1.2 Optimal and main prediction.

To derive this decomposition, we must define two things: the *optimal prediction* and the *main prediction*: according to Domingos, bias and variance can be defined in terms of these quantities.

The *optimal prediction* y_* for point $\mathbf{x}$ minimizes $E_t[L(t, y)]$:

$$y_*(\mathbf{x}) = \arg\min_y E_t[L(t, y)] \tag{3.1}$$

It is equal to the label t that is observed more often in the universe U of data points. The *optimal model* $\hat{f}(\mathbf{x}) = y_*,\ \forall \mathbf{x}$ makes the optimal prediction at each point $\mathbf{x}$. The noise $N(\mathbf{x})$, is defined in terms of the optimal prediction, and represents the remaining loss that cannot be eliminated, even by the optimal prediction:

$$N(\mathbf{x}) = E_t[L(t, y_*)]$$

Note that in the deterministic case $y_*(\mathbf{x}) = t$ and $N(\mathbf{x}) = 0$.

The *main prediction* y_m at point $\mathbf{x}$ is defined as

$$y_m = \arg\min_{y'} E_D[L(f_D(\mathbf{x}), y')]. \tag{3.2}$$

This is a value that would give the lowest expected loss if it was the "true label" of $\mathbf{x}$. It expresses the "central tendency" of a learner, that is its systematic prediction, or, in other words, it is the label for $\mathbf{x}$ that the learning algorithm "wishes" were correct. For 0/1 loss, the main prediction is the class predicted most often by the learning algorithm $\mathcal{L}$ when applied to training sets D.

3.1.3 Bias, unbiased and biased variance.

Given these definitions, the *bias* $B(\mathbf{x})$ (of learning algorithm $\mathcal{L}$ on training sets of size m) is the loss of the main prediction relative to the optimal prediction:

$$B(\mathbf{x}) = L(y_*, y_m)$$

For 0/1 loss, the bias is always 0 or 1. We will say that $\mathcal{L}$ is *biased at point* $\mathbf{x}$, if $B(\mathbf{x}) = 1$.

The *variance* $V(\mathbf{x})$ is the average loss of the predictions relative to the main prediction:

$$V(\mathbf{x}) = E_D[L(y_m, f_D(\mathbf{x}))] \tag{3.3}$$

It captures the extent to which the various predictions $f_D(\mathbf{x})$ vary depending on D.

In the case of the 0/1 loss we can also distinguish two opposite effects of variance (and noise) on the error: in the unbiased case variance and noise increase the error, while in the biased case variance and noise decrease the error.

There are three components that determine whether $t = y$:

1. Noise: is $t = y_*$?
2. Bias: is $y_* = y_m$?
3. Variance: is $y_m = y$?

Note that bias is either 0 or 1 because neither y_* nor y_m are random variables. From this standpoint we can consider two different cases: the unbiased and the biased case.

In the unbiased case, $B(\mathbf{x}) = 0$ and hence $y_* = y_m$. In this case we suffer a loss if the prediction y differs from the main prediction y_m (variance) and the optimal prediction y_* is equal to the target t, or y is equal to y_m, but y_* is different from t (noise).

In the biased case, $B(\mathbf{x}) = 1$ and hence $y_* \neq y_m$. In this case we suffer a loss if the prediction y is equal to the main prediction y_m and the optimal prediction y_* is equal to the target t, or if both y is different from to y_m (variance), and y_* is different from t (noise). Fig. 3.1 summarizes the different conditions under which an error can arise, considering the combined effect of bias, variance and noise on the learner prediction.

Considering the above case analysis of the error, if we let $P(t \neq y_*) = N(\mathbf{x}) = \tau$ and $P(y_m \neq y) = V(\mathbf{x}) = \sigma$, in the unbiased case we have:

$$\begin{aligned} L(t,y) &= \tau(1-\sigma) + \sigma(1-\tau) \\ &= \tau + \sigma - 2\tau\sigma \\ &= N(\mathbf{x}) + V(\mathbf{x}) - 2N(\mathbf{x})V(\mathbf{x}) \end{aligned} \tag{3.4}$$

while, in the the biased case:

$$\begin{aligned} L(t,y) &= \tau\sigma + (1-\tau)(1-\sigma) \\ &= 1 - (\tau + \sigma - 2\tau\sigma) \\ &= B(\mathbf{x}) - (N(\mathbf{x}) + V(\mathbf{x}) - 2N(\mathbf{x})V(\mathbf{x})) \end{aligned} \tag{3.5}$$

Note that in the unbiased case (eq. 3.4) the variance is an additive term of the loss function, while in the biased case (eq. 3.5) the variance is a subtractive term of the loss function. Moreover the interaction terms $\tau\sigma$ will usually be small, because, for instance, if both noise and variance term will be both lower than 0.1, the interaction term $2N(\mathbf{x})V(\mathbf{x})$ will be reduced to less than 0.02

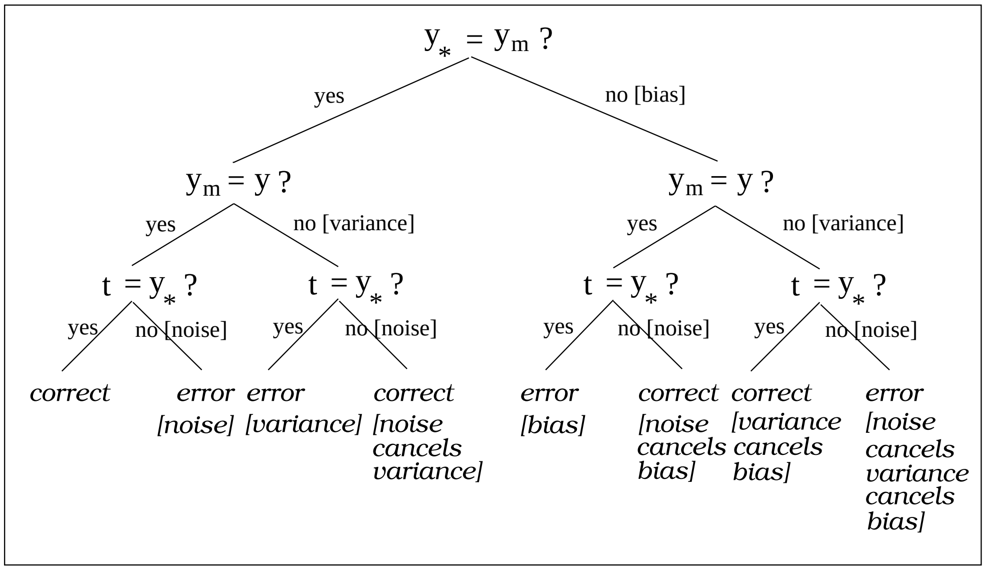


Figure 3.1: Case analysis of error.

In order to distinguish between these two different effects of the variance on the loss function, Domingos defines the *unbiased variance*, $V_u(\mathbf{x})$, to be the variance when $B(\mathbf{x}) = 0$ and the *biased variance*, $V_b(\mathbf{x})$, to be the variance when $B(\mathbf{x}) = 1$. We can also define the *net variance* $V_n(\mathbf{x})$ to take into account the combined effect of the unbiased and biased variance:

$$V_n(\mathbf{x}) = V_u(\mathbf{x}) - V_b(\mathbf{x})$$

Fig. 3.2 summarizes in graphic form the opposite effects of biased and unbiased variance on the error.

If we can disregard the noise, the unbiased variance captures the extents to which the learner deviates from the correct prediction y_m (in the unbiased case $y_m = y_*$), while the biased variance captures the extents to which the learner deviates from the incorrect prediction y_m (in the biased case $y_m \neq y_*$).

More precisely, for the two-class classification problem, with $N(x) = 0$, in the two cases we have:

1. If $B(\mathbf{x}) = 0$, $p_{corr}(\mathbf{x}) > 0.5 \Rightarrow V_u(\mathbf{x}) = 1 - p_{corr}(\mathbf{x})$.
2. If $B(x) = 1$, $p_{corr}(x) \leq 0.5 \Rightarrow V_b(x) = p_{corr}(x)$

where p_{corr} is the probability that a prediction is correct: $p_{corr}(\mathbf{x}) = P(y = t|\mathbf{x})$.

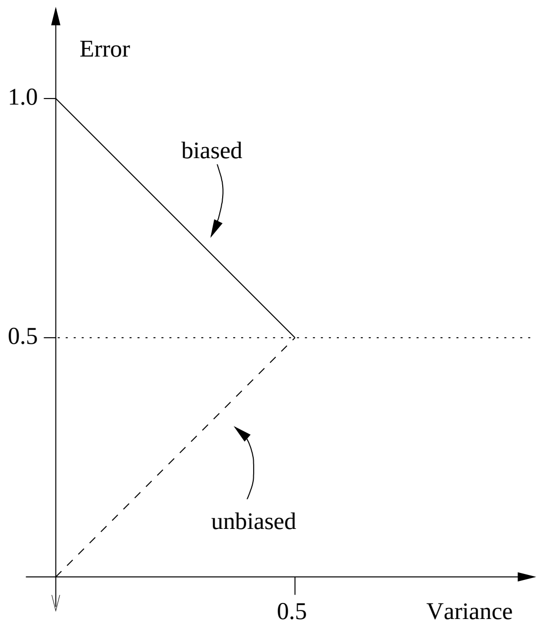


Figure 3.2: Effects of biased and unbiased variance on the error. The unbiased variance increments, while the biased variance decrements the error.

In fact, in the unbiased case:
$p_{corr}(\mathbf{x}) > 0.5 \Rightarrow y_m = t \Rightarrow P(y = y_m|\mathbf{x}) = p_{corr} \Rightarrow P(y \neq y_m) = 1 - p_{corr} \Rightarrow E_D[L(y_m, y)] = V(\mathbf{x}) = 1 - p_{corr}$.
Hence the variance $V(\mathbf{x}) = V_u(\mathbf{x}) = 1 - p_{corr}$ is given by the probability of an incorrect prediction, or equivalently expresses the deviation from the correct prediction.

In the biased case:
$p_{corr}(\mathbf{x}) \leq 0.5 \Rightarrow y_m \neq t \Rightarrow P(y = y_m|\mathbf{x}) = 1 - p_{corr} \Rightarrow P(y \neq y_m) = p_{corr} \Rightarrow E_D[L(y_m, y)] = V(\mathbf{x}) = p_{corr}$.
Hence the variance $V(\mathbf{x}) = V_b(\mathbf{x}) = p_{corr}$ is given by the probability of a correct prediction, or equivalently expresses the deviation from the incorrect prediction.

3.1.4 Domingos bias–variance decomposition.

For quite general loss functions L Domingos [47] showed that the expected loss is:

$$EL(\mathcal{L}, \mathbf{x}) = c_1 N(\mathbf{x}) + B(\mathbf{x}) + c_2 V(\mathbf{x}) \tag{3.6}$$

For the 0/1 loss, c_1 is $2P_D(f_D(\mathbf{x}) = y_*) - 1$ and c_2 is $+1$ if $B(\mathbf{x}) = 0$ and -1 if $B(\mathbf{x}) = 1$. Note that $c_2 V(\mathbf{x}) = V_u(\mathbf{x}) - V_b(\mathbf{x}) = V_n(\mathbf{x})$ (eq. 3.3), and if we disregard the noise, eq. 3.6

can be simplified to:

$$EL(\mathcal{L}, \mathbf{x}) = B(\mathbf{x}) + V_n(\mathbf{x}) \tag{3.7}$$

Summarizing, one of the most interesting aspects of Domingos' decomposition is that variance hurts on unbiased points $\mathbf{x}$, but it helps on biased points. Nonetheless, to obtain low overall expected loss, we want the bias to be small, and hence, we see to reduce both the bias and the unbiased variance. A good classifier will have low bias, in which case the expected loss will approximately equal the variance.

This decomposition for a single point $\mathbf{x}$ can be generalized to the entire population by defining $E_{\mathbf{x}}[\cdot]$ to be the expectation with respect to $P(\mathbf{x})$. Then we can define the *average bias* $E_{\mathbf{x}}[B(\mathbf{x})]$, the *average unbiased variance* $E_{\mathbf{x}}[V_u(\mathbf{x})]$, and the *average biased variance* $E_{\mathbf{x}}[V_b(\mathbf{x})]$. In the noise-free case, the expected loss over the entire population is

$$E_{\mathbf{x}}[EL(\mathcal{L}, \mathbf{x})] = E_{\mathbf{x}}[B(\mathbf{x})] + E_{\mathbf{x}}[V_u(\mathbf{x})] - E_{\mathbf{x}}[V_b(\mathbf{x})].$$

3.1.5 Bias, variance and their effects on the error

James [94] provides definitions of bias and variance that are identical to those provided by Domingos. Indeed bias and variance definitions are based on quantities that he named the systematic part sy of y and the systematic part st of t. These correspond respectively to the Domingos main prediction (eq.3.2) and optimal prediction (eq.3.1).

Moreover James distinguishes between bias and variance and *systematic* and *variance effects*. Bias and variance satisfy respectively the notion of the difference between the systematic parts of y and t, and the variability of the estimate y. Systematic effect SE represents the change in error of predicting t when using sy instead of st, and the variance effect VE the change in prediction error when using y instead of sy in order to predict t. Using Domingos notation (y_m for sy, and y_* for st) the variance effect is:

$$VE(y,t) = E_{y,t}[L(y,t)] - E_t[L(t, y_m)]$$

while the systematic effect corresponds to:

$$SE(y,t) = E_t[L(t, y_m)] - E_t[L(t, y_*)]$$

In other words the systematic effect represents the change in prediction error caused by bias, while the variance effect the change in prediction error caused by variance.

While for the squared loss the two sets of bias–variance definitions match, for general loss functions the identity does not hold. In particular for the 0/1 loss James proposes the following definitions for noise, variance and bias with 0/1 loss:

$$N(\mathbf{x}) \;\; = \;\; P(t \neq y_*)$$

$$\begin{aligned} V(\mathbf{x}) &= P(y \neq y_m) \\ B(\mathbf{x}) &= I(y_* \neq y_m) \end{aligned} \tag{3.8}$$

where $I(z)$ is 1 if z is true and 0 otherwise.

The variance effect for the 0/1 loss can be expressed in the following way:

$$\begin{aligned} VE(y,t) = E_{y,t}[L(y,t) - L(t,y_m)] &= P_{y,t}(y \neq t) - P_t(t \neq y_m) = \\ = 1 - P_{y,t}(y = t) - (1 - P_t(t = y_m)) &= P_t(t = y_m) - P_{y,t}(y = t) \end{aligned} \tag{3.9}$$

while the systematic effect is:

$$\begin{aligned} SE(y,t) = E_t[L(t,y_m)] - E_t[L(t,y_*)] &= P_t(t \neq y_m) - P_t(t \neq y_*) = \\ = 1 - P_t(t = y_m) - (1 - P_t(t = y_*)) &= P_t(t = y_*) - P_t(t = y_m) \end{aligned} \tag{3.10}$$

If we let $N(\mathbf{x}) = 0$, considering eq. 3.7, eq. 3.8, and eq. 3.9 the variance effect becomes:

$$\begin{aligned} VE(y,t) = P_t(t = y_m) - P_{y,t}(y = t) &= P(y_* = y_m) - P_y(y = y_*) = \\ = 1 - P(y_* \neq y_m) - (1 - P_y(y \neq y_*)) &= 1 - B(\mathbf{x}) - (1 - EL(\mathcal{L},\mathbf{x})) = \\ EL(\mathcal{L},\mathbf{x}) - B(\mathbf{x}) = V_n(\mathbf{x}) & \end{aligned} \tag{3.11}$$

while from eq. 3.8 and eq. 3.10 the systematic effect becomes:

$$\begin{aligned} SE(y,t) = P_t(t = y_*) - P_t(t = y_m) &= 1 - P_t(t \neq y_*) - (1 - P_t(t \neq y_m)) = \\ P(y* \neq y_m) = I(y* \neq y_m) = B(\mathbf{x}) & \end{aligned} \tag{3.12}$$

Hence if $N(\mathbf{x}) = 0$, it follows that the variance effect is equal to the net-variance (eq. 3.11), and the systematic effect is equal to the bias (eq. 3.12).

3.2 Measuring bias and variance

The procedures to measure bias and variance depend on the characteristics and on the cardinality of the data sets used.

For synthetic data sets we can generate different sets of training data for each learner to be trained. Then a large synthetic test set can be generated in order to estimate the bias–variance decomposition of the error for a specific learner model.

Similarly, if a large data set is available, we can split it in a large learning set and in a large testing set. Then we can randomly draw subsets of data from the large training set in order to train the learners; bias–variance decomposition of the error is measured on the large independent test set.

However, in practice, for real data we dispose of only one and often small data set. In this case, we can use cross-validation techniques for estimating bias–variance decomposition, but we propose to use out-of-bag [19] estimation procedures, as they are computationally less expensive.

3.2.1 Measuring with artificial or large benchmark data sets

Consider a set $\mathcal{D} = \{D_i\}_{i=1}^n$ of learning sets $D_i = \{\mathbf{x}_j, t_j\}_{j=1}^m$. Here we consider only a two-class case, i.e. $t_j \in \mathcal{C} = \{-1, 1\}, \quad \mathbf{x}_j \in \mathbf{X}$, for instance $\mathbf{X} = \mathbb{R}^d, \quad d \in \mathbb{N}$, but the extension to the multiclass case is straightforward.

We define $f_{D_i} = \mathcal{L}(D_i)$ as the model f_{D_i} produced by a learner $\mathcal{L}$ using a training set D_i. The model produces a prediction $f_{D_i}(\mathbf{x}) = y$.

In presence of noise and with the 0/1 loss, the optimal prediction y_* is equal to the label t that is observed more often in the universe U of data points:

$$y_*(\mathbf{x}) = \arg\max_{t \in \mathcal{C}} P(t|\mathbf{x})$$

The noise $N(\mathbf{x})$ for the 0/1 loss can be estimated if we can evaluate the probability of the targets for a given example $\mathbf{x}$:

$$N(\mathbf{x}) = \sum_{t \in \mathcal{C}} L(t, y_*) P(t|\mathbf{x}) = \sum_{t \in \mathcal{C}} ||t \neq y_*|| P(t|\mathbf{x})$$

where $||z|| = 1$ if z is true, 0 otherwise,

In practice, for "real world" data sets it is difficult to estimate the noise, and to simplify the computation we consider the noise free case. In this situation we have $y_* = t$.

The main prediction is a function of the $y = f_{D_i}(\mathbf{x})$. Considering a 0/1 loss, we have

$$y_m = \arg\max(p_1, p_{-1})$$

where $p_1 = P_D(y = 1|\mathbf{x})$ and $p_{-1} = P_D(y = -1|\mathbf{x})$, i.e. the main prediction is the mode. To calculate p_1, having a test set $\mathcal{T} = \{\mathbf{x}_j, t_j\}_{j=1}^r$, it is sufficient to count the number of learners that predict class 1 on a given input $\mathbf{x}$:

$$p_1(\mathbf{x}_j) = \frac{\sum_{i=1}^n \|f_{D_i}(\mathbf{x}_j) = 1\|}{n}$$

where $\|z\| = 1$ if z is true and $\|z\| = 0$ if z is false

The bias can be easily calculated after the evaluation of the main prediction:

$$B(\mathbf{x}) = \begin{cases} 1 & \text{if } y_m \neq t \\ 0 & \text{if } y_m = t \end{cases} = \left| \frac{y_m - t}{2} \right| \tag{3.13}$$

or equivalently:

$$B(\mathbf{x}) = \begin{cases} 1 & \text{if } p_{corr}(\mathbf{x}) \leq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

where p_{corr} is the probability that a prediction is correct, i.e. $p_{corr}(\mathbf{x}) = P(y = t|\mathbf{x}) = P_D(f_D(\mathbf{x}) = t)$.

In order to measure the variance $V(\mathbf{x})$, if we define $y_{D_i} = f_{D_i}(\mathbf{x})$, we have:

$$V(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} L(y_m, y_{D_i}) = \frac{1}{n}\sum_{i=1}^{n} ||(y_m \neq y_{D_i})||$$

The *unbiased variance* $V_u(\mathbf{x})$ and the *biased variance* $V_b(\mathbf{x})$ can be calculated evaluating if the prediction of each learner differs from the main prediction respectively in the unbiased and in the biased case:

$$V_u(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} ||(y_m = t) \text{ and } (y_m \neq y_{D_i})||$$

$$V_b(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} ||(y_m \neq t) \text{ and } (y_m \neq y_{D_i})||$$

In the noise-free case, the *average loss on the example* $\mathbf{x}$ $E_D(\mathbf{x})$ is calculated by a simple algebraic sum of bias, unbiased and biased variance:

$$E_D(\mathbf{x}) = B(\mathbf{x}) + V_u(\mathbf{x}) - V_b(\mathbf{x}) = B(\mathbf{x}) + (1 - 2B(\mathbf{x}))V(\mathbf{x})$$

In order to evaluate bias–variance decomposition on the entire set of examples, consider a test set $\mathcal{T} = \{\mathbf{x}_j, t_j\}_{j=1}^{r}$. We can easily calculate the *average bias, variance, unbiased, biased and net variance*, averaging over the entire set of examples:
Average bias:

$$E_{\mathbf{x}}[B(\mathbf{x})] = \frac{1}{r}\sum_{j=1}^{r} B(\mathbf{x}_j) = \frac{1}{r}\sum_{j=1}^{r} \left|\frac{y_m(\mathbf{x}_j) - t_j}{2}\right|$$

Average variance:

$$\begin{aligned} E_{\mathbf{x}}[V(\mathbf{x})] &= \frac{1}{r}\sum_{j=1}^{r} V(\mathbf{x}_j) \\ &= \frac{1}{nr}\sum_{j=1}^{r}\sum_{i=1}^{n} L(y_m(\mathbf{x}_j), f_{D_i}(\mathbf{x}_j)) \\ &= \frac{1}{nr}\sum_{j=1}^{r}\sum_{i=1}^{n} ||y_m(\mathbf{x}_j) \neq f_{D_i}(\mathbf{x}_j)|| \end{aligned}$$

Average unbiased variance:

$$E_{\mathbf{x}}[V_u(\mathbf{x})] = \frac{1}{r}\sum_{j=1}^{r} V_u(\mathbf{x}_j) = \frac{1}{nr}\sum_{j=1}^{r}\sum_{i=1}^{n} ||(y_m(\mathbf{x}_j) = t_j) \text{ and } (y_m(\mathbf{x}_j) \neq f_{D_i}(\mathbf{x}_j))||$$

Average biased variance:

$$E_{\mathbf{x}}[V_b(\mathbf{x})] = \frac{1}{r}\sum_{j=1}^{r} V_b(\mathbf{x}_j) = \frac{1}{nr}\sum_{j=1}^{r}\sum_{i=1}^{n} ||(y_m(\mathbf{x}_j) \neq t_j) \text{ and } (y_m(\mathbf{x}_j) \neq f_{D_i}(\mathbf{x}_j))||$$

Average net variance:

$$E_{\mathbf{x}}[V_n(\mathbf{x})] = \frac{1}{r}\sum_{j=1}^{r} V_n(\mathbf{x}_j) = \frac{1}{r}\sum_{j=1}^{r}(V_u(\mathbf{x}_j) - V_b(\mathbf{x}_j))$$

Finally, the average loss on all the examples (with no noise) is the algebraic sum of the average bias, unbiased and biased variance:

$$E_{\mathbf{x}}[L(t,y)] = E_{\mathbf{x}}[B(\mathbf{x})] + E_{\mathbf{x}}[V_u(\mathbf{x})] - E_{\mathbf{x}}[V_b(\mathbf{x})]$$

3.2.2 Measuring with small data sets

In practice (unlike in theory), we have only one and often small data set $\mathcal{S}$. We can simulate multiple training sets by bootstrap replicates $S' = \{\mathbf{x}|\mathbf{x} \text{ is drawn at random with replacement from } \mathcal{S}\}$.

In order to measure bias and variance we can use out-of-bag points, providing in such a manner an unbiased estimate of the error.

At first we need to construct B bootstrap replicates of $\mathcal{S}$ (e. g., $B = 200$): $S_1, \ldots, S_B$.

Then we apply a learning algorithm $\mathcal{L}$ to each replicate S_b to obtain hypotheses $f_b = \mathcal{L}(S_b)$.

Let $T_b = \mathcal{S} \backslash S_b$ be the data points that do not appear in S_b (out of bag points). We can use these data sets T_b to evaluate the bias–variance decomposition of the error; that is we compute the predicted values $f_b(\mathbf{x})$, $\forall \mathbf{x}$ s.t. $\mathbf{x} \in T_b$.
For each data point $\mathbf{x}$, we will now have the observed corresponding value t and several predictions $y_1, \ldots, y_K$, where $K = |\{T_b|\mathbf{x} \in T_b, 1 \leq b \leq B\}|$ depends on $\mathbf{x}$, $K \leq B$, and on the average $K \simeq B/3$, because about $1/3$ of the predictors is not trained on a specific input $\mathbf{x}$.

In order to compute the main prediction, for a two-class classification problem, we can define:

$$p_1(\mathbf{x}) = \frac{1}{K}\sum_{b=1}^{B} ||(\mathbf{x} \in T_b) \text{ and } (f_b(\mathbf{x}) = 1)||$$

$$p_{-1}(\mathbf{x}) = \frac{1}{K} \sum_{b=1}^{B} ||(\mathbf{x} \in T_b) \text{ and } (f_b(\mathbf{x}) = -1)||$$

The main prediction $y_m(\mathbf{x})$ corresponds to the mode:

$$y_m = \arg\max(p_1, p_{-1})$$

The bias can be calculated as in eq. 3.13, and the variance $V(\mathbf{x})$ is:

$$V(\mathbf{x}) = \frac{1}{K} \sum_{b=1}^{B} ||(\mathbf{x} \in T_b)\text{and}(y_m \neq f_b(\mathbf{x}))||$$

Similarly can be easily computed unbiased, biased and net–variance:

$$V_u(\mathbf{x}) = \frac{1}{K} \sum_{b=1}^{B} ||(\mathbf{x} \in T_b) \text{ and } (B(\mathbf{x}) = 0) \text{ and } (y_m \neq f_b(\mathbf{x}))||$$

$$V_b(\mathbf{x}) = \frac{1}{K} \sum_{b=1}^{B} ||(\mathbf{x} \in T_b) \text{ and } (B(\mathbf{x}) = 1) \text{ and } (y_m \neq f_b(\mathbf{x}))||$$

$$V_n(\mathbf{x}) = V_u(\mathbf{x}) - V_b(\mathbf{x})$$

Average bias, variance, unbiased, biased and net variance, can be easily calculated averaging over all the examples.

Chapter 4

Bias–Variance Analysis in single SVMs

The bias–variance decomposition of the error represents a powerful tool to analyze learning processes in learning machines. According to the procedures described in the previous chapter, we analyzed bias and variance in SVMs, in order to study the relationships with different kernel types and their parameters. To accomplish this task we computed bias–variance decomposition of the error on different synthetic and "real" data sets.

4.1 Support Vector Machines

In this section we provide a very brief overview of Support Vector Machines in order to introduce the main notions and concepts used in the rest of the this chapter. For more details, see, for instance [188, 37].

Given a data set $\mathcal{Z} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $\mathbf{x}_i \in \mathbb{R}^N, y_i \in \mathcal{Y} = \{-1, 1\}$, where y_i are the labels of two different classes of examples, a linear classifier computes a decision function $g(\mathbf{x}) = \mathrm{sign(f(\mathbf{x}))}$, where $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$.

For a point $\mathbf{x}_p$ on the separating hyperplane $f(\mathbf{x}_p) = \mathbf{w} \cdot \mathbf{x}_p + b = 0$ (Fig. 4.1), a point $\mathbf{x}_m$ on the margin whose width is γ can be expressed as

$$\mathbf{x}_m = \mathbf{x}_p + \frac{\mathbf{w}}{||\mathbf{w}||}\gamma$$

Then $f(\mathbf{x}_m) = \mathbf{w} \cdot \mathbf{x}_m + b = \mathbf{w} \cdot \mathbf{x}_p + \frac{\mathbf{w}\cdot\mathbf{w}}{||\mathbf{w}||}\gamma + b = \gamma||\mathbf{w}||$.

The *functional margin* is $\gamma||\mathbf{w}||$ and the *geometric margin* is $\gamma = \frac{f(\mathbf{x}_m)}{||\mathbf{w}||}$.

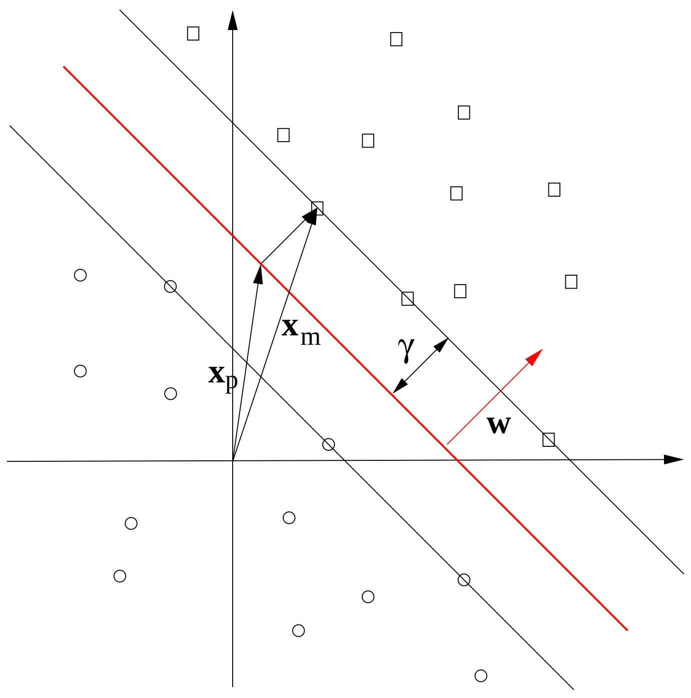


Figure 4.1: Separating hyperplane and margins in a two-class classification problem

To obtain the *canonical separating hyperplane* we need to normalize w.r.t the functional margin:

$$f_c(\mathbf{x}) = \frac{f(\mathbf{x})}{\gamma||\mathbf{w}||}$$

The *canonical functional margin* is

$$f_c(\mathbf{x}_m) = \frac{f(\mathbf{x}_m)}{\gamma||\mathbf{w}||} = 1$$

The *canonical margin* is $\gamma_c = \frac{1}{||\mathbf{w}||}$

From this point we consider only the canonical hyperplane (that is the hyperplane with canonical margin $1/||\mathbf{w}||$.

In order to maximize the margin $\gamma = \frac{1}{||\mathbf{w}||}$ and to correctly separate the examples we need to solve a constrained quadratic optimization problem:

$$\begin{array}{ll} \text{Minimize} & \mathbf{w}\cdot\mathbf{w} \\ \text{subject to} & y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 \\ & 1 \leq i \leq n \end{array}$$

The hyperplane $\mathbf{w}\cdot\mathbf{x}+b=0$ that solves this quadratic oprimization problem is the *maximal margin iperplane* with margin $\gamma=\frac{1}{||\mathbf{w}||}$

The lagrangian associated with the primal optimization problem is:

$$L(\mathbf{w},b,\alpha)=\frac{1}{2}\mathbf{w}\cdot\mathbf{w}-\sum_{i=1}^{n}\alpha_i(y_i(\mathbf{w}\cdot\mathbf{x}_i+b)-1)$$

leading to this set of optimality conditions:

$$\begin{aligned}\frac{\partial L(\mathbf{w},b,\alpha)}{\partial\mathbf{w}} &= \mathbf{w}-\sum_{i=1}^{n}y_i\alpha_i\mathbf{x}_i=\mathbf{0}\\ \frac{\partial L(\mathbf{w},b,\alpha)}{\partial b} &= \sum_{i=1}^{n}y_i\alpha_i=0\end{aligned}$$

hence

$$\begin{aligned}\mathbf{w} &= \sum_{i=1}^{n}y_i\alpha_i\mathbf{x}_i\\ 0 &= \sum_{i=1}^{n}y_i\alpha_i\end{aligned}$$

Putting the relations obtained into the primal we have:

$$\begin{aligned}L(\mathbf{w},b,\alpha) &= \frac{1}{2}\mathbf{w}\cdot\mathbf{w}-\sum_{i=1}^{n}\alpha_i(y_i(\mathbf{w}\cdot\mathbf{x}_i+b)-1)\\ &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}y_iy_j\alpha_i\alpha_j(\mathbf{x}_i\cdot\mathbf{x}_j)-\sum_{i=1}^{n}\sum_{j=1}^{n}y_iy_j\alpha_i\alpha_j(\mathbf{x}_i\cdot\mathbf{x}_j)+\sum_{i=1}^{n}\alpha_i\\ &= \sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}y_iy_j\alpha_i\alpha_j(\mathbf{x}_i\cdot\mathbf{x}_j)\end{aligned}$$

obtaining the associated dual optimization problem:

$$\begin{array}{ll}\text{Maximize} & \Phi(\alpha)=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}y_iy_j\alpha_i\alpha_j(\mathbf{x}_i\cdot\mathbf{x}_j)\\ \text{subject to} & \sum_{i=1}^{n}y_i\alpha_i=0\\ & \alpha_i\geq 0,\quad 1\leq i\leq n\end{array}$$

The hyperplane whose weight vector $\mathbf{w}^*=\sum_{i=1}^{n}y_i\alpha_i\mathbf{x}_i$ solves this quadratic optimization problem is the *maximal margin hyperplane* with geometric margin $\gamma=\frac{1}{||\mathbf{w}||}$.

The linear SVMs compute the family of linear functions:

$$\mathcal{F}(\mathbf{x}, \mathbf{w}, b) = \{\mathbf{x} \cdot \mathbf{w} + b, \mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R}\}$$

If α^* is the solution of the dual optimization problem then

- $\mathbf{w}^* = \sum_{i=1}^n y_i \alpha_i^* \mathbf{x}_i$ is the weight vector of the maximal margin hyperplane
- $f(\mathbf{x}) = \mathbf{w}^* \cdot \mathbf{x} + b^* = \sum_{i=1}^n y_i \alpha_i^* \mathbf{x}_i \cdot \mathbf{x} + b^*$ is the corresponding discriminant function.
- The decision function $g : \mathbb{R}^n \to \{-1, +1\}$ is $g(\mathbf{x}) = \text{sign}(\sum_{i=1}^n y_i \alpha_i^* \mathbf{x}_i \cdot \mathbf{x} + b^*)$

The SVM algorithm minimizes both the empirical risk and the confidence interval [188]. Indeed, maximizing the margin, that is equivalently minimizing $||\mathbf{w}||$, we minimize the Vapnik Chervonenkis (VC) dimension, and the confidence interval depends mainly on the ratio (VC) dimension/ cardinality of the training set.

In order to consider non lineraly separable data we nee to introduce soft margin SVM and kernels. In this setting we first add to the primal optimization problems a set of slack variables ξ_i, and a

$$\begin{array}{ll} \text{Minimize} & \mathbf{w} \cdot \mathbf{w} + C \sum_{i=1}^n \xi_i \\ \text{subject to} & y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \\ & 1 \leq i \leq n \end{array}$$

If $K(\mathbf{x}, \mathbf{x}')$ is a symmetric function satisfing *Mercer's conditions*, that is:

$$\int \int K(\mathbf{x}, \mathbf{x}') f(\mathbf{x}) f(\mathbf{x}') d\mathbf{x} d\mathbf{x}' \geq 0$$

for all f such that $\int f^2(\mathbf{x}) d\mathbf{x} < \infty$, then we can expand $K(\mathbf{x}, \mathbf{x}')$ in a some inner product feature space:

$$K(\mathbf{x}, \mathbf{x}') = \sum_{j=1}^{\infty} \lambda_j \phi(\mathbf{x}) \phi(\mathbf{x}')$$

Note that in the dual representation of linear SVMs the inputs appears only in a dot-product form:as a consequence we can substitute the dot-products in the input space with a kernel function obeying Mercer's conditions:

$$\begin{array}{ll} \text{Maximize} & \Phi(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i \mathbf{x}_j) \\ \text{subject to} & \sum_{i=1}^n y_i \alpha_i = 0 \\ & 0 \leq \alpha_i \leq C, \quad 1 \leq i \leq n \end{array}$$

The discriminant function obtained from the solution of this quadratic optimization problem is:

$$f(\mathbf{x}, \alpha^*, b) = \sum_{i=1}^{n} y_i \alpha_i^* K(\mathbf{x}_i, \mathbf{x}) + b^*$$

The SVM receives as inputs patterns $\mathbf{x}$ in the input space, but works in a high dimensional (possibly infinite) feature space, where it performs a linear separation of the data.

The symmetric function $K(\cdot, \cdot)$ must be chosen among the kernels of Reproducing Kernel Hilbert Spaces [189]; three possible choices are:

- Linear kernel: $K(\mathbf{u}, \mathbf{v}) = \mathbf{u} \cdot \mathbf{v}$
- Polynomial kernel: $K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^d$
- Gaussian kernel: $K(\mathbf{u}, \mathbf{v}) = \exp(-\|\mathbf{u} - \mathbf{v}\|^2/\sigma^2)$

The bias and variance of SVMs are typically controlled by two parameters. The parameter C controls the tradeoff between fitting the data (achieved by driving the ξ_i's to zero) and maximizing the margin (achieved by driving $\|\mathbf{w}\|$ to zero). Setting C large should tend to minimize bias.

The second parameter that controls bias arises only in SVMs that employ parameterized kernels such as the polynomial kernel (where the parameter is the degree d of the polynomial) and RBF kernels (where the parameter is the width σ of the gaussian kernel). Bias and variance depend critically on these parameters [182].

4.2 Experimental setup

We performed an extended bias–variance analysis of the error in Support Vector Machines, training and testing more than half million of different SVMs on different training and test sets.

4.2.1 Data sets

In the experiments we employed 7 different data sets, both synthetic and "real".

P2 is a synthetic bidimensional two–class data set; each region is delimited by one or more of four simple polynomial and trigonometric functions (Fig. 4.2).

The synthetic data set *Waveform* is generated from a combination of 2 of 3 "base" waves; we reduced the original three classes of *Waveform* to two, deleting all samples pertaining to class 0. The other data sets are all from the UCI repository [135].

Tab. 4.1 summarizes the main features of the data sets used in the experiments. The rest of this section explains in more detail the characteristics of the data sets.

Table 4.1: Data sets used in the experiments.

Data set	# of attr.	# of tr. samples	# of tr. sets	# base tr. set	# of test samples
P2	2	100	400	synthetic	10000
Waveform	21	100	200	synthetic	10000
Grey-Landsat	36	100	200	4425	2000
Letter	16	100	200	614	613
Letter w. noise	16	100	200	614	613
Spam	57	100	200	2301	2300
Musk	166	100	200	3299	3299

4.2.1.1 P2

We used a synthetic bidimensional two–class data set (Fig. 4.2). Each region, delimited by one or more of four simple polynomial and trigonometric functions, belongs to one of the two classes, according to the Roman numbers I and II. We generated a series of 400 training sets with 100 independent examples randomly extracted according to a uniform probability distribution. The test set (10000 examples) was generated through the same distribution. The application `gensimple`, that we developed to generate the data, is freely available on line at `ftp://ftp.disi.unige.it/person/ValentiniG/BV/gensimple`.

4.2.1.2 Waveform

It is a synthetic data set from the UCI repository. Each class is generated from a combination of 2 of 3 "base" waves. Using the application `waveform` we can generate an arbitrary number of samples from the same distribution. We reduced the original three classes of *Waveform* to two, deleting all samples pertaining to class 0.

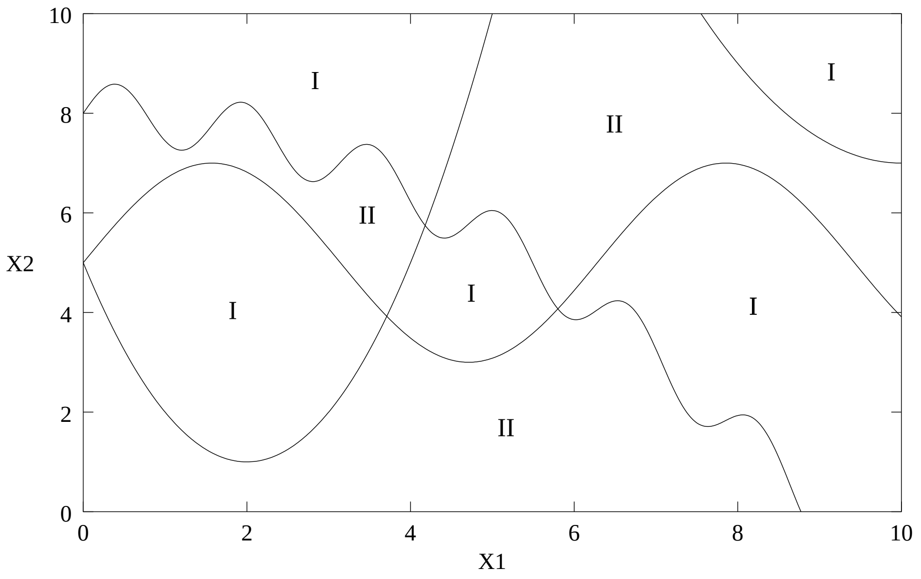


Figure 4.2: P2 data set, a bidimensional two class synthetic data set.

4.2.1.3 Grey-Landsat

It is a data set from the UCI repository, modified in order to be available for a dichotomic classification problem. The attributes represent intensity values for four spectral bands and nine neighbouring pixels, while the classification refers to the central pixel. Hence we have 9 data values for each spectral band for a total of 36 data attributes for each pattern. The data come from a rectangular area approximately five miles wide. The original data set Landsat (available from UCI repository) is a 6 way classification data set with 36 attributes. Following Scott and Langdon [125], classes 3, 4 and 7 were combined into one (positive gray), while 1, 2 and 5 became the negative examples (not-Gray).

4.2.1.4 Letter-Two

It is a reduced version of the Letter data set from UCI: we consider here only letter B versus letter R, taken from the letter recognition data set. The 16 attributes are integer values that refer to different features of the letters. We used also a version of Letter-Two with 20 % added classification noise (*Letter-Two with added noise* data set).

4.2.1.5 Spam

This data set from UCI separates "spam" e-mails from "non-spam' e-mails, considering mainly attributes that indicate whether a particular word or character frequently occurs in the e-mail. Of course, the concept of spam is somewhat subjective: in particular the creators of this data set selected non-spam e-mails that came from filed work and personal e-mails, while collection of spam e-mails came from their postmaster and individuals who had filed spam. However we have a relatively large data set with 4601 instances and 57 continuous attributes.

4.2.1.6 Musk

The dataset (available from UCI) describes a set of 102 molecules of which 39 are judged by human experts to be musks and the remaining 63 molecules are judged to be non-musks. The 166 features that describe these molecules depend upon the exact shape, or conformation, of the molecule. Because bonds can rotate, a single molecule can adopt many different shapes. To generate this data set, all the low-energy conformations of the molecules were generated to produce 6,598 conformations. Then, a feature vector was extracted that describes each conformation.

In these experiments the data set was used as a normal data set, considering directly the different conformations of the same molecule as a different instance. As a consequence, each feature vector represents a different example to be classified and the classifier does not classify a molecule as "musk" if any of its conformations is classified as a musk. In other words we used the data set without considering the many-to-one relationship between feature vectors and molecules that characterize the "multiple instance problem".

4.2.2 Experimental tasks

In order to perform a reliable evaluation of bias and variance we used small training set and large test sets. For synthetic data we generated the desired number of samples. For real data sets we used bootstrapping to replicate the data. In both cases we computed the main prediction, bias, unbiased and biased variance, net-variance according to the procedures explained in Sect. 3.2.1. In our experiments, the computation of variance effect and systematic effect is reduced to the measurement of the net-variance and bias, as we did not explicitly consider the noise (eq. 3.11 and 3.12).

4.2.2.1 Set up of the data

With synthetic data sets, we generated small training sets of about 100 examples and reasonably large test sets using computer programs. In fact small samples show bias and variance more clearly than having larger samples. We produced 400 different training sets for *P2* and 200 training sets for *Waveform*. The test sets were chosen reasonably large (10000 examples) to obtain reliable estimates of bias and variance.

For real data sets we first divided the data into a training $\mathcal{D}$ and a test $\mathcal{T}$ sets. If the data sets had a predefined training and test sets reasonably large, we used them (as in *Grey-Landsat* and *Spam*), otherwise we split them in a training and test set of equal size. Then we drew from $\mathcal{D}$ bootstrap samples. We chosen bootstrap samples much smaller than $|\mathcal{D}|$ (100 examples). More precisely we drew 200 data sets from $\mathcal{D}$, each one consisting of 100 examples uniformly drawn with replacement.

Fig. 4.3 outlines the experimental procedure we adopted for setting up the data and Fig. 4.4 the experimental procedure to evaluate bias–variance decomposition of the error.

```
Procedure Generate_samples
Input arguments:
  - Data set $\mathcal{S}$
  - Number $n$ of samples
  - Size $s$ of the samples
Output:
  - Set $\bar{D} = \{D_i\}_{i=1}^n$ of samples
begin procedure
  $[\mathcal{D}, \mathcal{T}] = \texttt{Split}(\mathcal{S})$
  $\bar{D} = \emptyset$
  for $i = 1$ to $n$
  begin
    $D_i = \texttt{Draw\_with\_replacement}(\mathcal{D}, s)$
    $\bar{D} = \bar{D} + D_i$
  end
end procedure.
```

Figure 4.3: Procedure to generate samples to be used for bias–variance analysis with single SVMs

```
Procedure Bias–Variance_analysis
Input arguments:
   - Test set $\mathcal{T}$
   - Number of samples $n$
   - Set of learning parameters $\mathcal{A}$
   - Set $\bar{D} = \{D_i\}_{i=1}^n$ of samples
Output:
   - Error, bias, net-variance, unbiased and biased variance $BV = \{bv(\alpha)\}_{\alpha \in \mathcal{A}}$
   of the SVMs with learning parameters $\alpha \in \mathcal{A}$.
begin procedure
   For each $\alpha \in \mathcal{A}$
   begin
      SVM_Set($\alpha$) = $\emptyset$
      for $i = 1$ to $n$
      begin
         svm($\alpha$, $D_i$) = svm_train ($\alpha$, $D_i$)
         SVM_Set($\alpha$) = SVM_Set($\alpha$) $\cup$ svm($\alpha$, $D_i$)
      end
      $bv(\alpha)$ = Perform_BV_analysis(SVM_Set ($\alpha$), $\mathcal{T}$)
      $BV = BV \cup bv(\alpha)$
   end
end procedure.
```

Figure 4.4: Procedure to perform bias–variance analysis on single SVMs

Samples D_i are drawn with replacement according to an uniform probability distribution from the training set $\mathcal{D}$ by the procedure `Draw_with_replacement`. This process is repeated n times (procedure `Generate_samples`, Fig. 4.3). Then the procedure `Bias--Variance_analysis` (Fig. 4.4) trains different SVM models, according to the different learning parameters α provided to the procedure `svm_train`). SVM_Set(α) is the set of the SVMs trained using the same learning parameter α and a set $\bar{D}$ of samples generated by the procedure `Generate_samples`.

The bias–variance decomposition of the error is performed on the separated test set $\mathcal{T}$ using the previously trained SVMs (procedure `Perform_BV_analysis`).

4.2.2.2 Tasks

To evaluate bias and variance in SVMs we conducted experiments with different kernels and different kernel parameters.

In particular we considered 3 different SVM kernels:

1. **Gaussian kernels**. We evaluated bias–variance decomposition varying the parameters σ of the kernel and the C parameter that controls the trade–off between training error and the margin. In particular we analyzed:

   (a) The relationships between average error, bias, net–variance, unbiased and biased variance and the parameter σ of the kernel.

   (b) The relationships between average error, bias, net–variance, unbiased and biased variance and the parameter C (the regularization factor) of the kernel.

   (c) The relationships between generalization error, training error, number of support vectors and capacity with respect to σ.

   We trained RBF-SVM with all the combinations of the parameters σ and C, taken from the following two sets:

$$\left\{ \begin{array}{lcl} \sigma & \in & \{0.01, 0.02, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100, 200, 300, 400, 500, 1000\} \\ C & \in & \{0.01, 0.1, 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000\} \end{array} \right.$$

   evaluating in such a way $17 \times 12 = 204$ different RBF-SVM models for each data set.

2. **Polynomial kernels**. We evaluated bias–variance decomposition varying the degree of the kernel and the C parameter that controls the trade–off between training error and the margin. In particular we analyzed:

   (a) The relationships between average error, bias, net–variance, unbiased and biased variance and the degree of the kernel.

   (b) The relationships between average error, bias, net–variance, unbiased and biased variance and the parameter C (the regularization factor) of the kernel.

   We trained polynomial-SVM with all the combinations of the parameters:

$$\left\{ \begin{array}{lcl} degree & \in & \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\} \\ C & \in & \{0.01, 0.1, 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000\} \end{array} \right.$$

   evaluating in such a way $10 \times 12 = 120$ different polynomial-SVM models for each data set. Following the heuristic of Jakkola, the dot product of polynomial kernel was divided by the dimension of the input data, to "normalize" the dot–product before to raise to the degree of the polynomial.

3. **Dot–product kernels**. We evaluated bias–variance decomposition varying the C parameter. We analyzed the relationships between average error, bias, net–variance, unbiased and biased variance and the parameter C (the regularization factor) of

the kernel. We trained dot–product-SVM considering the following values for the C parameter:

$$C \in \{0.01, 0.1, 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000\}$$

evaluating in such a way 12 different dot–product-SVM models for each data set.

Each SVM model required the training of 200 different SVMs, one for each synthesized or bootstrapped data set, for a total of $(204+120+12) \times 200 = 67200$ trained SVMs for each data set (134400 for the data set *P2*, as for this data set we used 400 data sets for each model).

Summarizing the experiments required the training of more than half million of SVMs, considering all the data sets and of course the testing of all the SVM previously trained in order to evaluate the bias–variance decomposition of the error of the different SVM models. For each SVM model we computed the main prediction, bias, net-variance, biased and unbiased variance and the error on each example of the test set, and the corresponding average quantities on the overall test set.

4.2.3 Software used in the experiments

In all our experiments we used the *NEURObjects* [185][1] C++ library and *SVM-light* [96] applications. In particular the synthetic data *P2* and *Waveform* were generated respectively through our C++ application `gensimple` and `waveform` from the UCI repository. The bootstrapped data for the real data sets were extracted using the C++ *NEURObjects* application `subsample`. The data then were normalized using the *NEURObjects* application `convert_data_format`. For some data sets in order to extract randomly a separated training and test set we used the *NEURObjects* application `dofold`. Training and testing of the SVM were performed using Joachim's *SVM-light* software, and in particular the applications `svm_learn` and `svm_classify`. We slightly modified `svm_learn` in order to force convergence of the SVM algorithm when the optimality conditions are not reached in a reasonable time. We developed and used the C++ application `analyze_BV`, to perform bias–variance decomposition of the error[2]. This application analyzes the output of a generic learning machine model and computes the main prediction, error, bias, net–variance, unbiased and biased variance using the 0/1 loss function. Other C++ applications have been developed for the automatic analysis of the results, using also Cshell scripts to train, test and analyze bias–variance decomposition of all the SVM models for a specific data set, considering respectively gaussian, polynomial and dot–product kernels.

[1]Download web site: `http://www.disi.unige.it/person/ValentiniG/NEURObjects`.

[2]The source code is available at `ftp://ftp.disi.unige.it/person/ValentiniG/BV`. Moreover C++ classes for bias–variance analysis have been developed as part of the *NEURObjects* library

4.3 Results

In this section we present the results of the experiments. We analyzed bias–variance decomposition with respect to the kernel parameters considering separately gaussian, polynomial and dot product SVMs, comparing also the results among different kernels. Here we present the main results. Full results, data and graphics are available by anonymous ftp at: `ftp://ftp.disi.unige.it/person/ValentiniG/papers/bv-svm.ps.gz`.

4.3.1 Gaussian kernels

Fig. 4.5 depicts the average loss, bias net–variance, unbiased and biased variance varying the values of σ and the regularization parameter C in *RBF-SVM* on the *Grey-Landsat* data set. We note that σ is the most important parameter: although for very low values of C the SVM cannot learn, independently of the values of σ, (Fig. 4.5 a), the error, the bias, and the net–variance depend mostly on the σ parameter. In particular for low values of σ, bias is very high (Fig. 4.5 b) and net-variance is 0, as biased and unbiased variance are about equal (Fig. 4.5d and 4.5e). Then the bias suddenly goes down (Fig. 4.5b), lowering the average loss (Fig. 4.5a), and then stabilizes for higher values of σ. Interestingly enough, in this data set (but also in others, data not shown), we note an increment followed by a decrement of the net–variance, resulting in a sort of "wave shape" of the net variance graph (Fig. 4.5c).

Fig. 4.6 shows the bias–variance decomposition on different data sets, varying σ, and for a fixed value of C, that is a sort of "slice" along the σ axis of the Fig. 4.5. The plots show that average loss, bias, and variance depend significantly on σ for all the considered data sets, confirming the existence of a "high biased region" for low values of σ. In this region, biased and unbiased variance are about equal (net–variance $V_n = V_u - V_b$ is low). Then unbiased variance increases while biased variance decreases (Fig. 4.6 a,b,c and d), and finally both stabilize for relatively high values of σ. Interestingly, the average loss and the bias do not increase for high values of σ, especially if C is high.

Bias and average loss increases with σ only for very small C values. Note that net-variance and bias show opposite trends only for small values of C (Fig. 4.6 c). For larger C values the symmetric trend is limited only to $\sigma \leq 1$ (Fig. 4.6 d), otherwise bias stabilizes and net-variance slowly decreases.

Fig. 4.7 shows more in detail the effect of the C parameter on bias-variance decomposition. For $C \geq 1$ there are no variations of the average error, bias and variance for a fixed value of σ. Note that for very low values of σ (Fig. 4.7a and b) there is no learning. In the Letter-Two data set, as in other data sets (figures not shown), only for small C values we have variations in bias and variance values (Fig. 4.7).

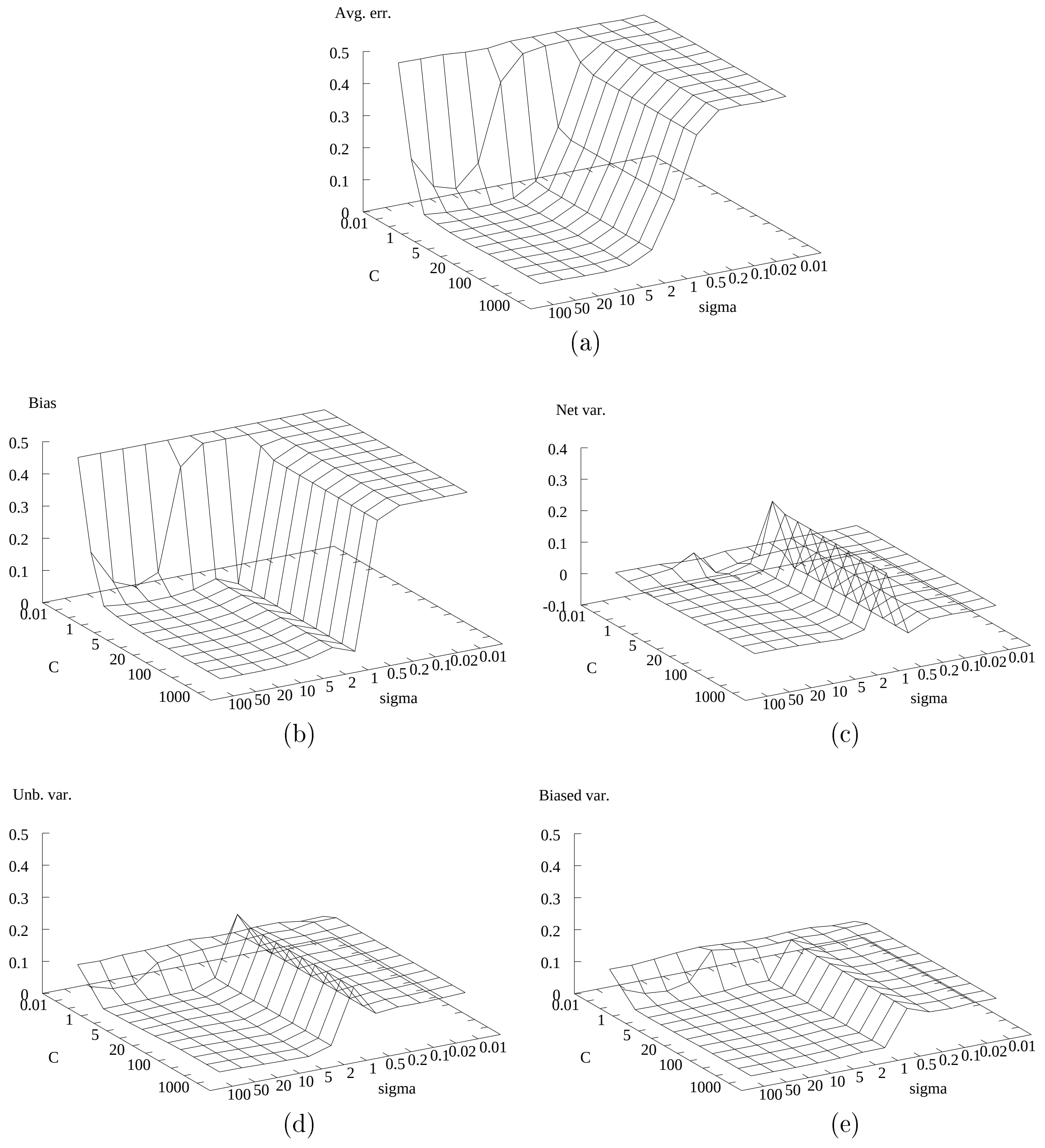
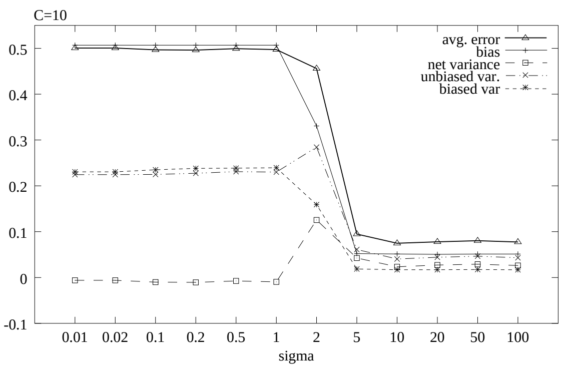
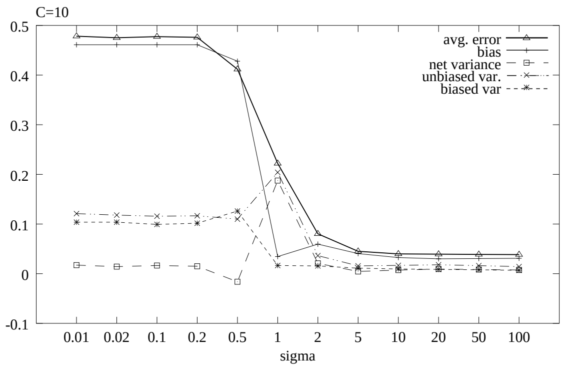


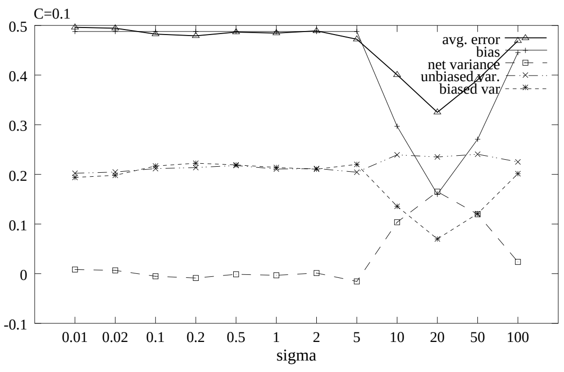
Figure 4.5: Grey-Landsat data set. Error (a) and its decomposition in bias (b), net variance (c), unbiased variance (d), and biased variance (e) in SVM RBF, varying both C and σ.



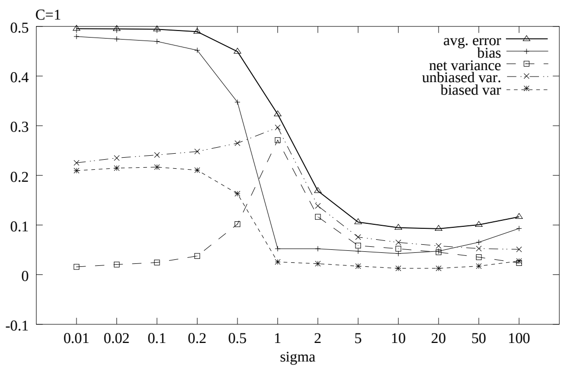
(a)



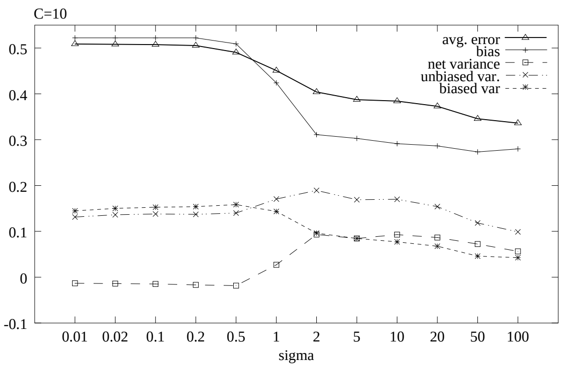
(b)



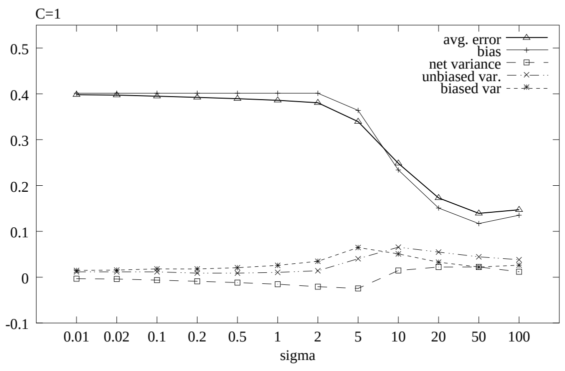
(c)



(d)

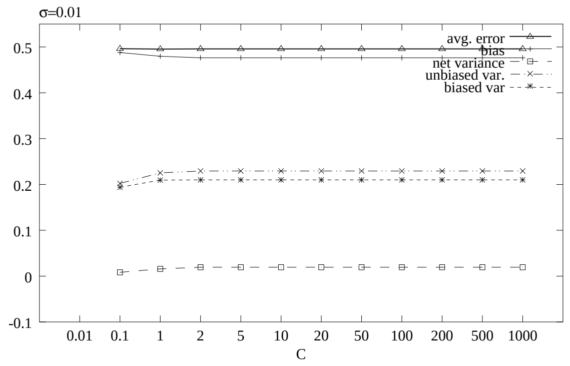


(e)

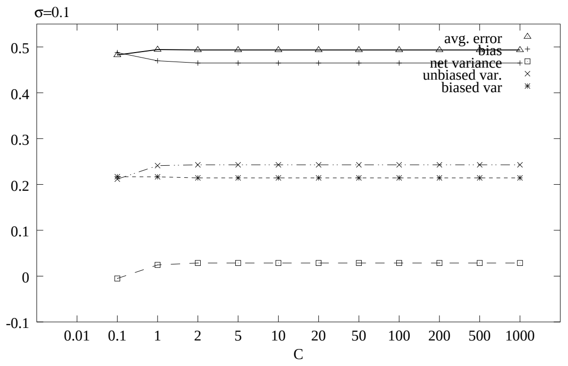


(f)

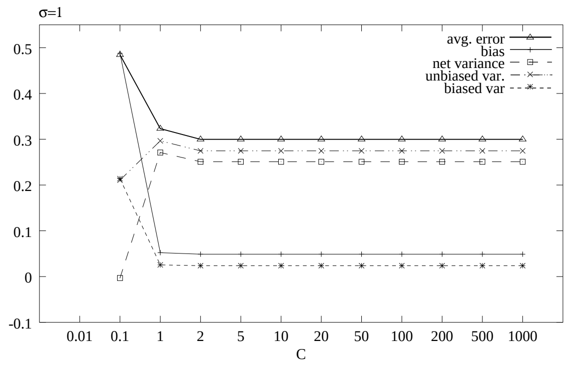
Figure 4.6: Bias-variance decomposition of the error in bias, net variance, unbiased and biased variance in SVM RBF, varying σ and for fixed C values: (a) Waveform, (b) Grey-Landsat, (c) Letter-Two with $C = 0.1$, (c) Letter-Two with $C = 1$, (e) Letter-Two with added noise and (f) Spam.



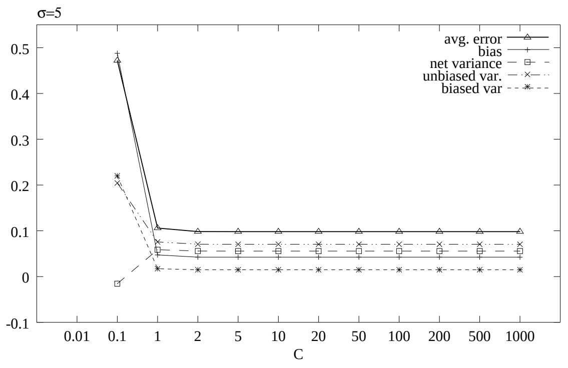
(a)



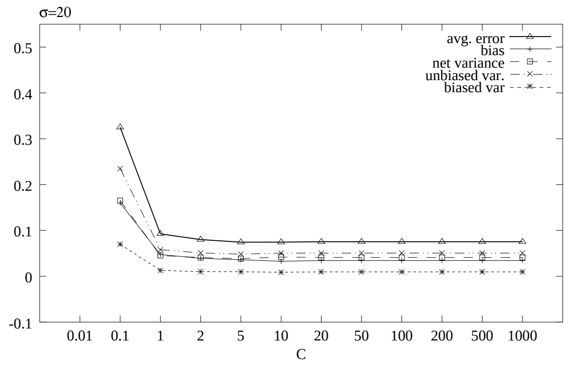
(b)



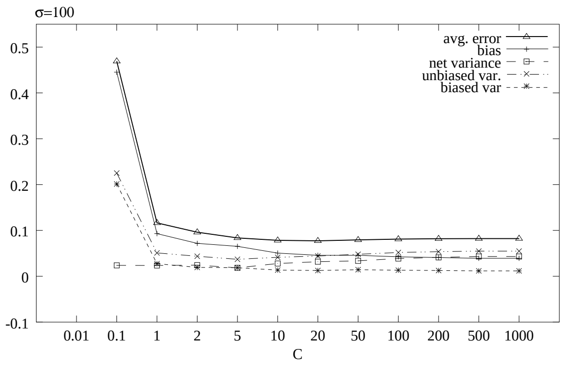
(c)



(d)



(e)



(f)

Figure 4.7: Letter-Two data set. Bias-variance decomposition of error in bias, net variance, unbiased and biased variance in SVM RBF, while varying C and for some fixed values of σ: (a) $\sigma = 0.01$, (b) $\sigma = 0.1$, (c) $\sigma = 1$, (d) $\sigma = 5$, (e) $\sigma = 20$, (f) $\sigma = 100$.

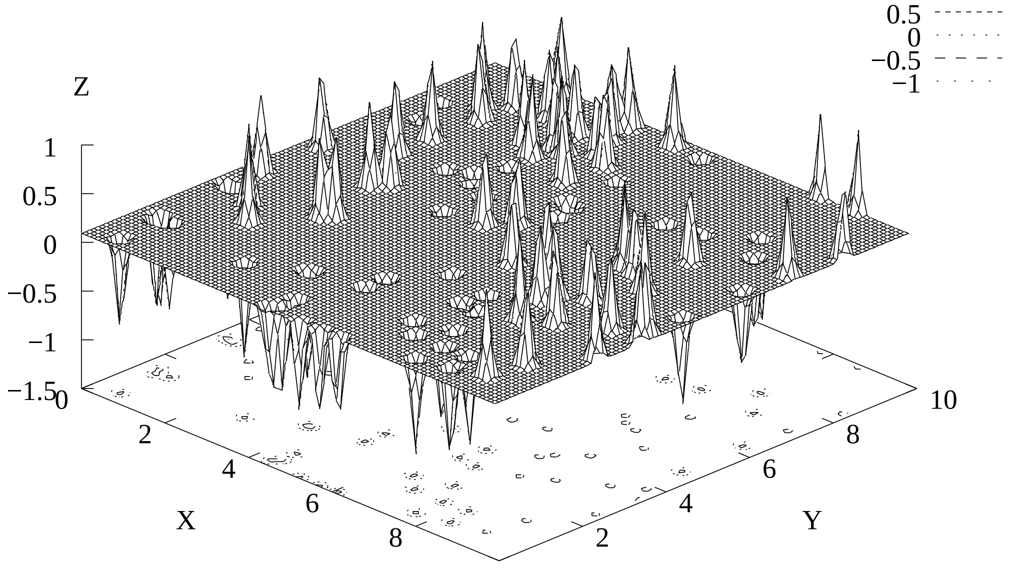


Figure 4.8: The discriminant function computed by the SVM on the P2 data set with $\sigma = 0.01$, $C = 1$.

4.3.1.1 The discriminant function computed by the SVM-RBF classifier

In order to get insights into the behaviour of SVM learning algorithm with gaussian kernels we plotted the real-valued functions computed without considering the discretization step performed through the sign function. The real valued function computed by a gaussian SVM is the following:

$$f(\mathbf{x}, \alpha, b) = \sum_{i \in SV} y_i \alpha_i \exp(-\|\mathbf{x_i} - \mathbf{x}\|^2/\sigma^2) + b$$

where the α_i are the Lagrange multipliers found by the solution of the dual optimization problem, the $\mathbf{x_i} \in SV$ are the support vectors, that is the points for which $\alpha_i > 0$.

We plotted the surface computed by the gaussian SVM with the synthetic data set *P2*. Indeed it is the only surface that can be easily visualized, as the data are bidimensional and the resulting real valued function can be easily represented through a wireframe three-dimensional surface. The SVMs are trained with exactly the same training set composed by 100 examples. The outputs are referred to a test set of 10000 examples, selected in an uniform way through all the data domain. In particular we considered a grid of equi-spaced

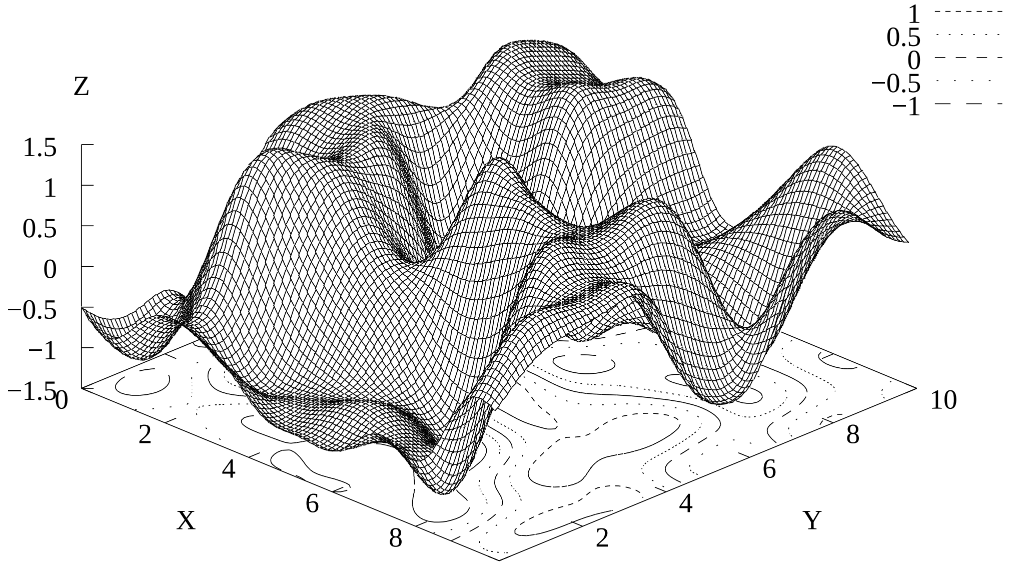


Figure 4.9: The discriminant function computed by the SVM on the P2 data set, with $\sigma = 1$, $C = 1$.

data at 0.1 interval in a two dimensional 10×10 input space. If $f(\mathbf{x}, \alpha, b) > 0$ then the SVM matches up the example $\mathbf{x}$ with class 1, otherwise with class 2.

With small values of σ we have "spiky" functions: the response is high around the support vectors, and is close to 0 in all the other regions of the input domain (Fig. 4.8). In this case we have overfitting: a large error on the test set (about 46 % with $\sigma = 0.01$ and 42.5 % with $\sigma = 0.02$), and a training error near to 0. If we enlarge the values of σ we obtain a wider response on the input domain and the error decreases (with $\sigma = 0.1$ the error is about 37 %). With $\sigma = 1$ we have a smooth function that fits quite well the data (Fig. 4.9). In this case the error drops down to about 13 %.

Enlarging too much σ we have a too smooth function (Fig. 4.10 (a)), and the error increases to about 37 %: in this case the high bias is due to an excessive smoothing of the function. Increasing the values of the regularization parameter C (in order to better fit the data), we can diminish the error to about 15 %: the shape of the function now is less smooth (Fig. 4.10 (b)).

Finally using very large values of sigma (e.g. $\sigma = 500$), we have a very smooth (in practice a plan) and a very biased function (error about 45 %), and if we increment C, we obtain

obtain better results, but always with a large error (about 35 %).

4.3.1.2 Behavior of SVMs with large σ values

Fig 4.5 and 4.6 show that the σ parameter plays a sort of smoothing effect, when the value of σ increases. In particular with large values of σ we did not observe any increment of bias nor decrement of variance. In order to get insights into this counter-intuitive behaviour we tried to answer these two questions:

1. Does the bias increase while variance decrease with large values of σ, and what is the combined effect of bias-variance on the error?

2. In this situation (large values for σ), what is the effect of the C parameter?

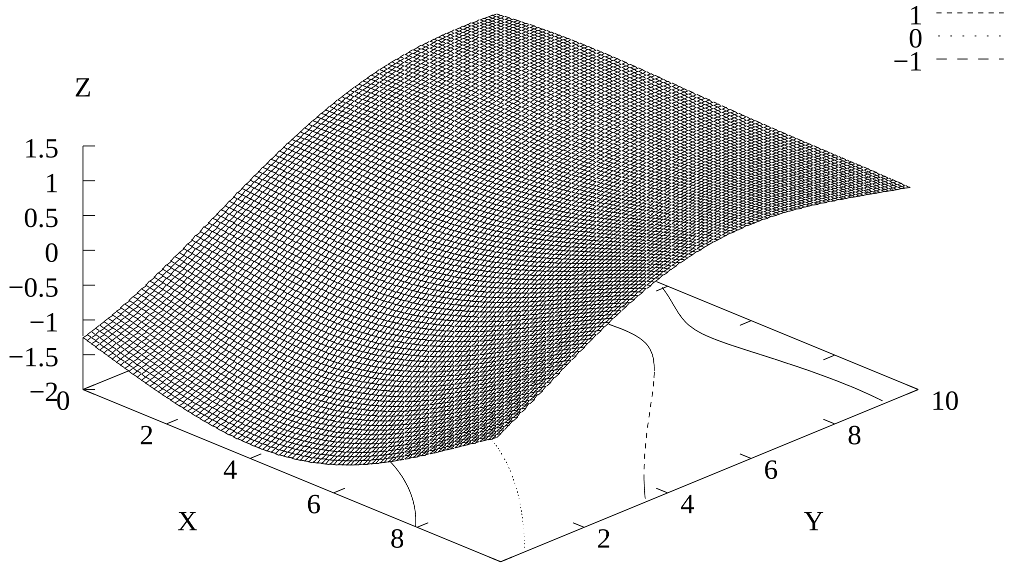
In Fig. 4.6 we do not observe an increment of bias with large values of σ, but we limited our experiments to values of $\sigma \leq 100$. Here we investigate the effect for larger values of σ (from 100 to 1000).

In most cases, also increasing the values of σ right to 1000 we do not observe an increment of the bias and a substantial decrement of the variance. Only for low values of C, that is $C < 1$ the bias and the error increase with large values of σ (Fig. 4.11).

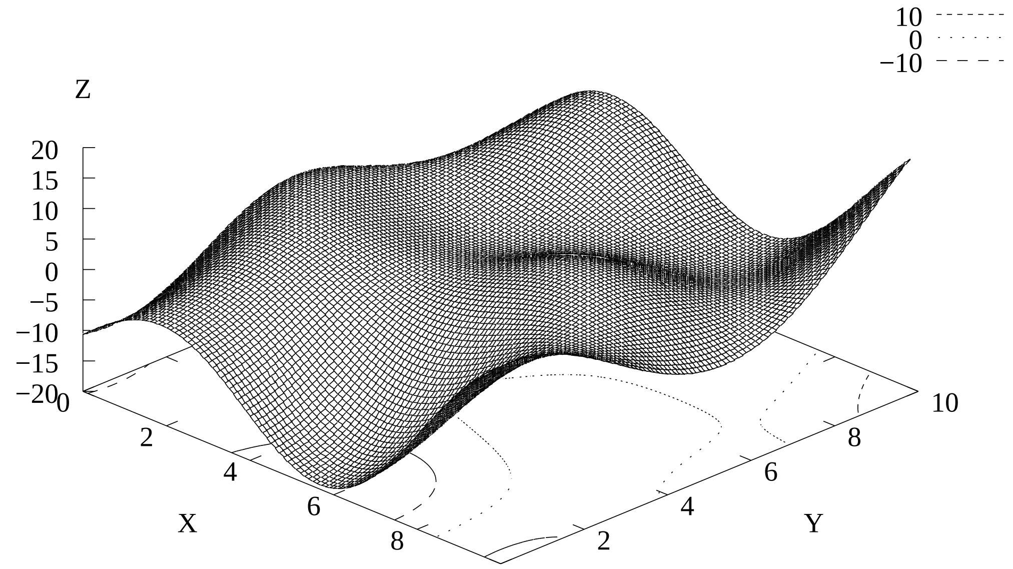
With the P2 data set the situation is different: in this case we observe an increment of the bias and the error with large values of σ, even if with large value of C the increment rate is lower (Fig. 4.12 a and b). Also with the musk data set we note an increment of the error with very large values of σ, but surprisingly this is due to an increment of the unbiased variance, while the bias is quite stable, at least for values of $C > 1$, (Fig. 4.12 c and d).

Larger values of C counter-balance the bias introduced by large values of σ. But with some distributions of the data too large values of σ produce too smooth functions, and also incrementing C it is very difficult to fit the data. Indeed, the real-valued function computed by the RBF-SVM with the P2 data set (that is the function computed without considering the sign function) is too smooth for large values of σ: for $\sigma = 20$, the error is about 37%, due almost entirely to the large bias, (Fig. 4.10 a), and for $\sigma = 500$ the error is about 45 % and also incrementing the C value to 1000, we obtain a surface that fits the data better, but with an error that remains large (about 35%).

Summarizing with large σ values bias can increment, while net-variance tends to stabilize, but this effect can be counter-balanced by larger C values.

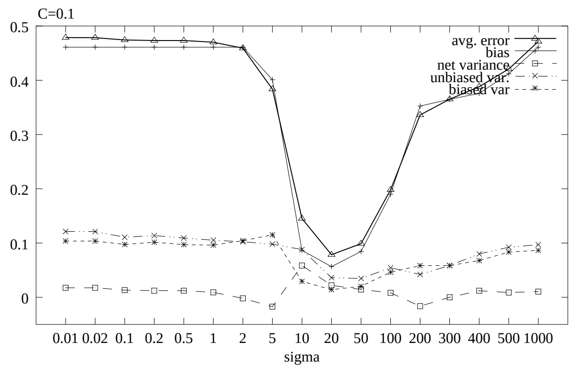


(a)

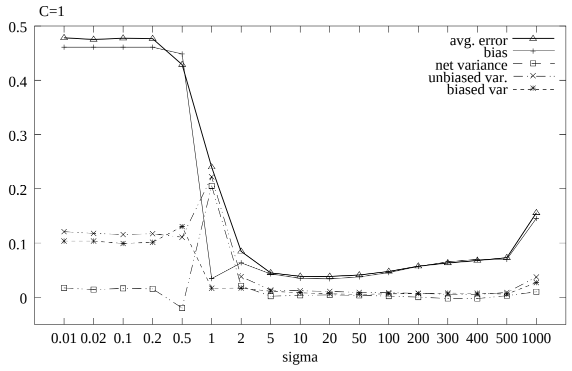


(b)

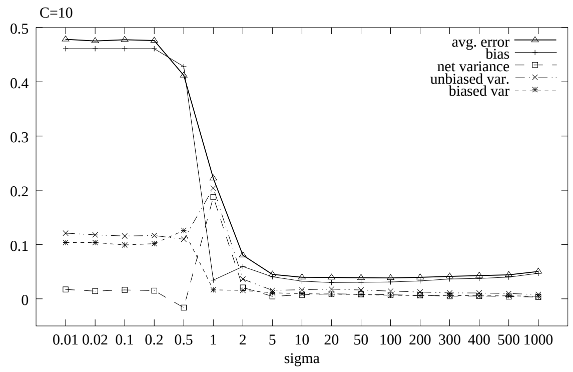
Figure 4.10: The discriminant function computed by the SVM on the P2 data set. (a) $\sigma = 20$, $C = 1$, (b) $\sigma = 20$ $C = 1000$.



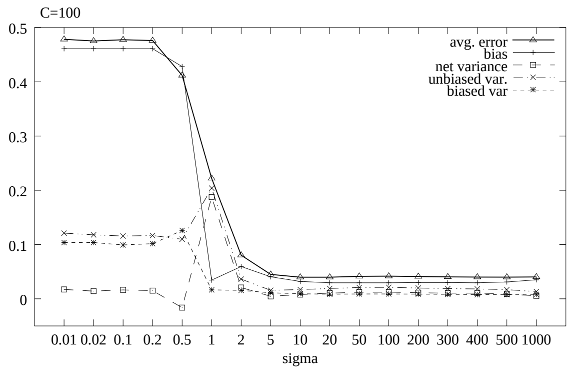
(a)



(b)

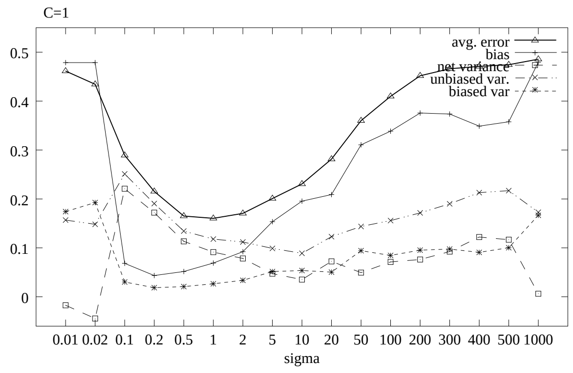


(c)

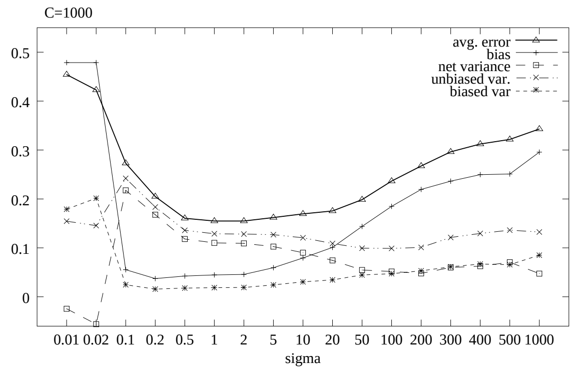


(d)

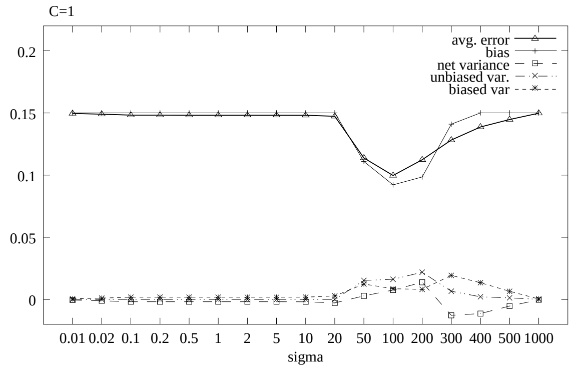
Figure 4.11: Grey-Landsat data set. Bias-variance decomposition of error in bias, net variance, unbiased and biased variance in SVM RBF, while varying σ and for some fixed values of C: (a) $C = 0.1$, (b) $C = 1$, (c) $C = 10$, (d) $C = 100$.



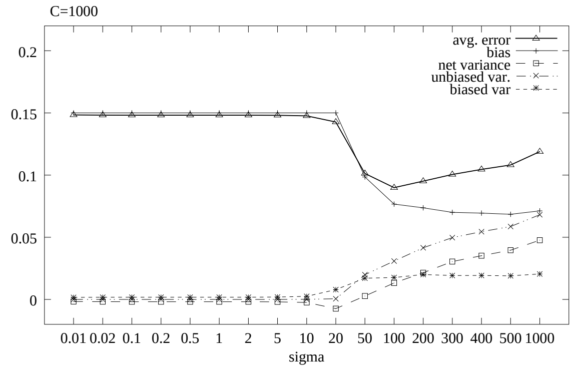
(a)



(b)



(c)



(d)

Figure 4.12: Bias-variance decomposition of error in bias, net variance, unbiased and biased variance in SVM RBF, while varying σ and for some fixed values of C: (a) P2, with $C = 1$, (b) P2, with $C = 1000$, Musk, with $C = 1$, (d) Musk, with $C = 1000$.

4.3.1.3 Relationships between generalization error, training error, number of support vectors and capacity

Looking at Fig. 4.5 and 4.6, we see that SVMs do not learn for small values of σ. Moreover the low error region is relatively large with respect to σ and C.

In this section we evaluate the relationships between the estimated generalization error, the bias, the training error, the number of support vectors and the estimated Vapnik Chervonenkis dimension [188], in order to answer the following questions:

1. Why SVMs do not learn for small values of σ?

2. Why we have a so large bias for small values of σ?
3. Can we use the variation of the number of support vectors to predict the "low error" region?
4. Is there any relationship between the bias, variance and VC dimension, and can we use this last one to individuate the "low error" region?

The generalization error, bias, training error, number of support vectors and the Vapnik Chervonenkis dimension are estimated averaging with respect to 400 SVMs (*P2* data set) or 200 SVMs (other data sets) trained with different bootstrapped training sets composed by 100 examples each one. The test error and the bias are estimated with respect to an independent and sufficiently large data set.

The *VC dimension* is estimated using the Vapnik's bound based on the radius R of the sphere that contains all the data (in the feature space), approximated through the sphere centered in the origin, and on the norm of the weights in the feature space [188]. In this way the VC dimension is overestimated but it is easy to compute and we are interested mainly in the comparison of the VC dim. of different SVM models:

$$VC \leq R^2 \cdot \|\mathbf{w}\|^2 + 1$$

where [37]

$$\|\mathbf{w}\|^2 = \sum_{i \in SV} \sum_{j \in SV} \alpha_i \alpha_j K(\mathbf{x_i}, \mathbf{x_j}) y_i y_j$$

and

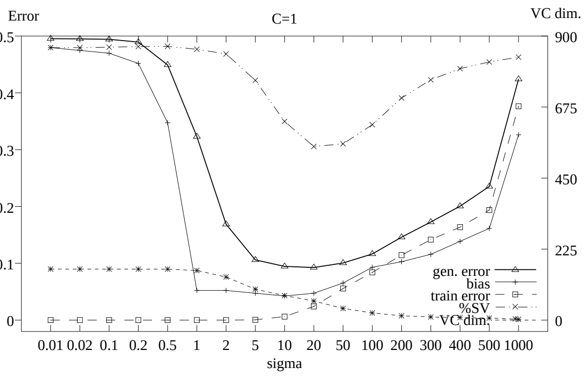
$$R^2 = \max_i K(\mathbf{x_i}, \mathbf{x_i})$$

The number of support vectors is expressed as the halved ratio of the number ($\%\ SV$) of support vectors with respect to the total number of the training data:

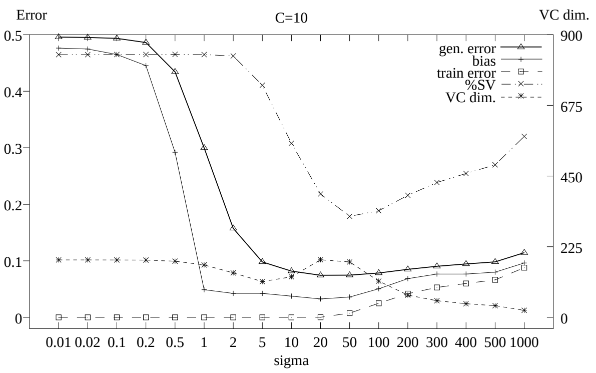
$$\%SV = \frac{\#SV}{\#trainingdata * 2}$$

In the graphs shown in Fig. 4.13 and Fig. 4.14, on the left y axis is represented the error, training error and bias, and the halved ratio of support vectors. On the right y axis is reported the estimated Vapnik Chervonenkis dimension.

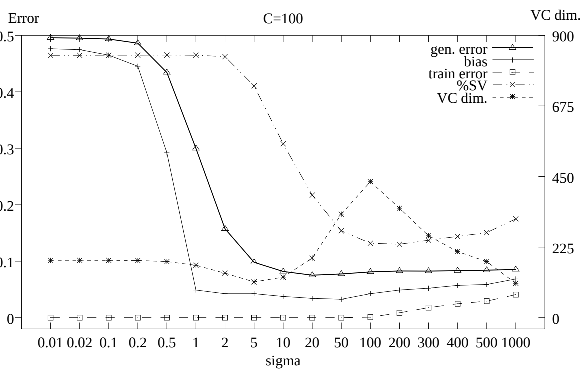
For very small values of σ the training error is very small (about 0), while the number of support vectors is very high, and high is also the error and the bias (Fig.4.13 and 4.14). These facts support the hypothesis of overfitting problems with small values of σ. Indeed the real-valued function computed by the SVM (that is the function computed without considering the sign function, Sect. 4.3.1.1) is very spiky with small values of σ (Fig. 4.8).



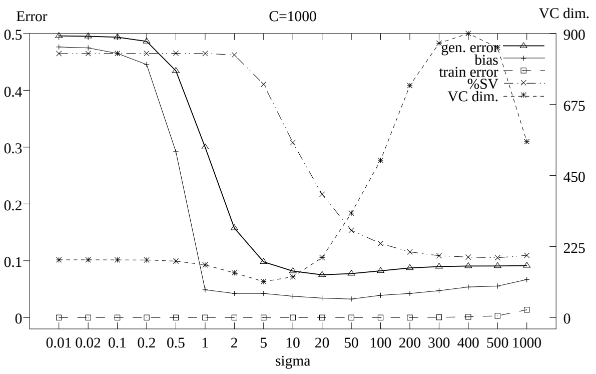
(a)



(b)



(c)



(d)

Figure 4.13: Letter-Two data set. Error, bias, training error, halved fraction of support vectors, and estimated VC dimension while varying the σ parameter and for some fixed values of C: (a) $C = 1$, (b) $C = 10$, (c) $C = 100$, and $C = 1000$.

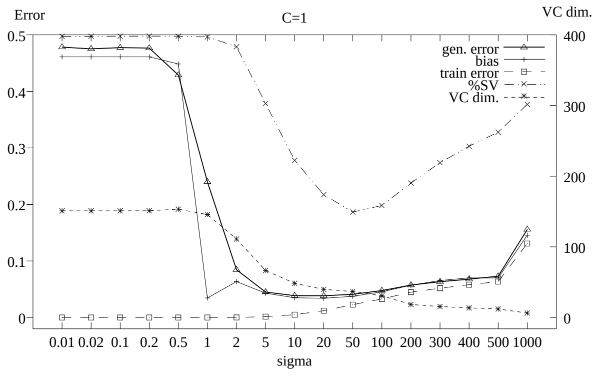
The response of the SVM is high only in small areas around the support vectors, while in all the other areas "non covered" by the gaussians centered to the support vectors the response is very low (about 0), that is the SVM is not able to get a decision, with a consequently very high bias.

Anyway, note that the pattern of dependency between σ and the bias–variance components of the error might be different if larger training sets were used. Indeed the high bias region will be smaller (that is limited to lower values of σ) if the number of the examples of the training set is larger. With an increasing number of examples, the "non-covered" regions of the input space will be reduced, and they might be only small exceptions if relatively large training sets are used. This should be not so surprising, as it is well-known that th error, as well as its bias variance components depend largely on the cardinality of the available training set [12, 188].

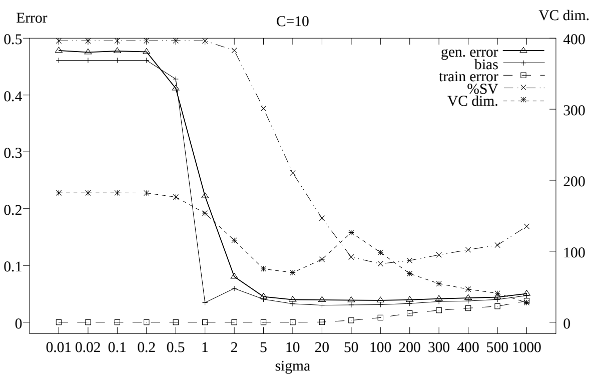
In the same region (small values for σ) the net variance is usually very small, for either one of these reasons: 1) biased and unbiased variance are almost equal but both different from 0 ; 2) biased and unbiased variance are about equal but both very near to 0 (Fig. 4.6 a, b and f). In the first case the SVM performs a sort of random guessing for the most part of the unknown data, resulting in a very biased response, but with a certain variability due to the fact both in the biased and in the unbiased regions SVM trained on different training sets provide different outputs, with a consequent biased and unbiased variance.

In the second case the SVMs tend to answer in the same way independently of a particular instance of the test set: they classify all the examples as positives or as negatives. As a consequence, both biased and unbiased variance are 0, and the error is equal to the bias. For instance, if the number of the examples in the test set is equal for the positive and the negative class, then the bias (and the error) will be 0.5. If the number of positive examples is n^+ and the number of negative examples is n^-, and the SVMs classifiy as positive all the examples the bias and the error will be $n^-/(n^+ + n^-)$, and of course will be $n^+/(n^+ + n^-)$ if the SVM classifies all the examples as negative. Enlarging σ we obtain a wider response on the input domain: the real-valued function computed by the SVM becomes smoother (Fig. 4.9), as the "bumps" around the support vectors become wider and the SVM can decide also on unknown examples. At the same time the number of support vectors decreases (Fig. 4.13 and 4.14).

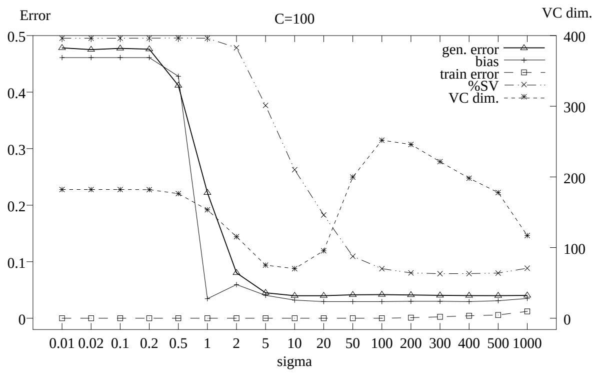
Considering the variation of the ratio of the support vectors with σ, in all data sets the trend of the halved ratio of support vectors follows the error, with a sigmoid shape that sometimes becomes an U shape for small values of C (Fig.4.13 and 4.14). This is not surprising because it is known that the support vector ratio offers an approximation of the generalization error of the SVMs [188]. Moreover, on all the data sets the halved ratio of support vectors decreases in the "stabilized" region, while in the transition region remains high. As a consequence the decrement in the number of support vectors shows that we are entering the "low error" region, and in principle we can use this information to detect this



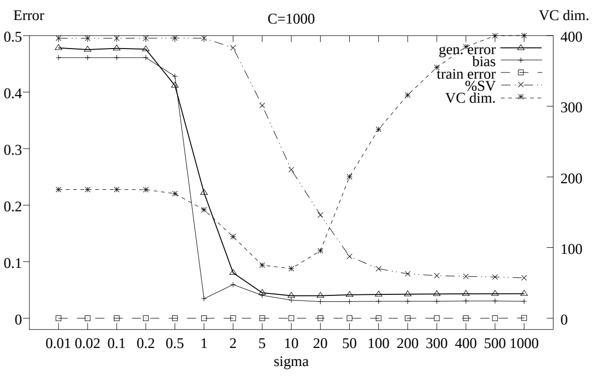
(a)



(b)



(c)



(d)

Figure 4.14: Grey-Landsat data set. Error, bias, training error, halved fraction of support vectors, and estimated VC dimension while varying the σ parameter and for some fixed values of C: (a) $C = 1$, (b) $C = 10$, (c) $C = 100$, and $C = 1000$.

region.

In order to analyze the role of the VC dimension on the generalization ability of learning machines, we know from Statistical learning Theory that the form of the bounds of the generalization error E of SVMs is the following:

$$E(f(\sigma, C)_n^k)) \leq E_{emp}(f(\sigma, C)_n^k)) + \Phi(\frac{h_k}{n}) \tag{4.1}$$

where $f(\sigma, C)_n^k$ represents the set of functions computed by an RBF-SVM trained with n examples and with parameters (σ_k, C_k) taken from a set of parameters $S = \{(\sigma_i, C_i), i \in \mathbb{N}\}$, E_{emp} represents the empirical error and Φ the confidence interval that depends on the cardinality n of the data set and on the VC dimension h_k of the set of functions identified by the actual selection of the parameters (σ_k, C_k). In order to obtain good generalization capabilities we need to minimize both the empirical risk and the confidence interval. According to Vapnik's bounds (eq. 4.1), in Fig. 4.13 and 4.14 the lowest generalization error is obtained for a small empirical risk and a small estimated VC dimension.

But sometimes with relatively small values of VC we have a very large error, as the training error and the number of support vectors increase with very large values of σ (Fig. 4.13 a and 4.14 a). Moreover with a very large estimate of the VC dimension and low empirical error (Fig. 4.13 and 4.14) we have a relatively low generalization error.

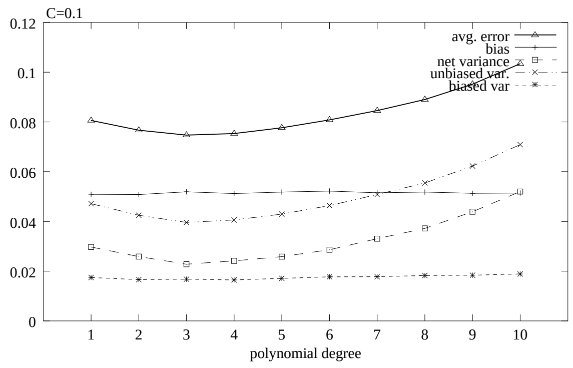
In conclusion it seems very difficult to use in practice these estimates of the VC dimension to infer the generalization abilities of the SVM. In particular it seems unreliable to use the VC dimension to infer the "low error" region of the RBF-SVM.

4.3.2 Polynomial and dot-product kernels

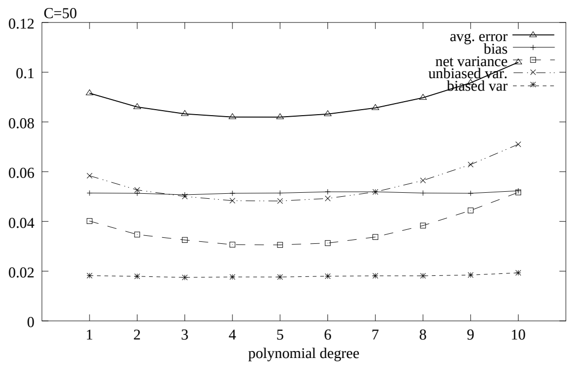
In this section we analyze the characteristics of bias–variance decomposition of the error in polynomial SVMs, varying the degree of the kernel and the regularization parameter C.

Error shows a U shape with respect to the degree. This shape depends on unbiased variance (Fig. 4.15 a and b), or both by bias and unbiased variance (Fig. 4.15 c and d). The U shape of the error with respect to the degree tends to be more flat for increasing values of C, and net-variance and bias show often opposite trends (Fig. 4.16).

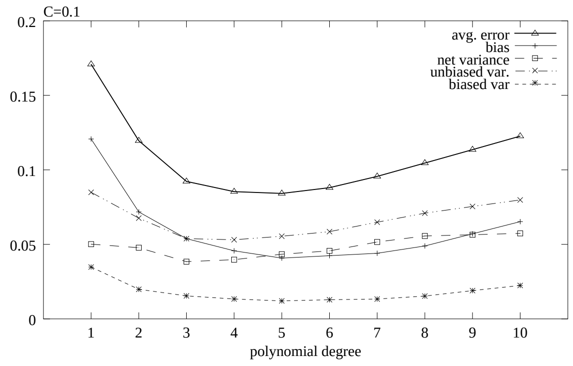
Average error and bias are higher for low C and degree values, but incrementing the degree the error is less sensitive to C values (Fig. 4.17). Bias is flat (Fig. 4.18 a) or decreasing with respect to the degree (Fig. 4.16 b), or it can be constant or decreasing, depending on C (Fig. 4.18 b). Unbiased variance shows an U shape (Fig. 4.15 a and b) or it increases (Fig. 4.15 c) with respect to the degree, and the net–variance follows the shape of the unbiased variance. Note that in the P2 data set (Fig. 4.16) bias and net–variance follow



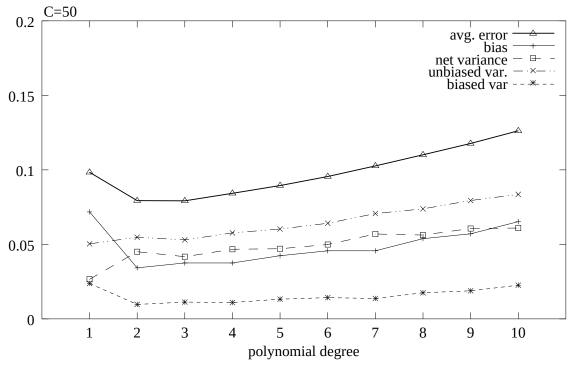
(a)



(b)



(c)



(d)

Figure 4.15: Bias-variance decomposition of error in bias, net variance, unbiased and biased variance in polynomial SVM, while varying the degree and for some fixed values of C: (a) Waveform, $C = 0.1$, (b) Waveform, $C = 50$, (c) Letter-Two, $C = 0.1$, (d) Letter-Two, $C = 50$.

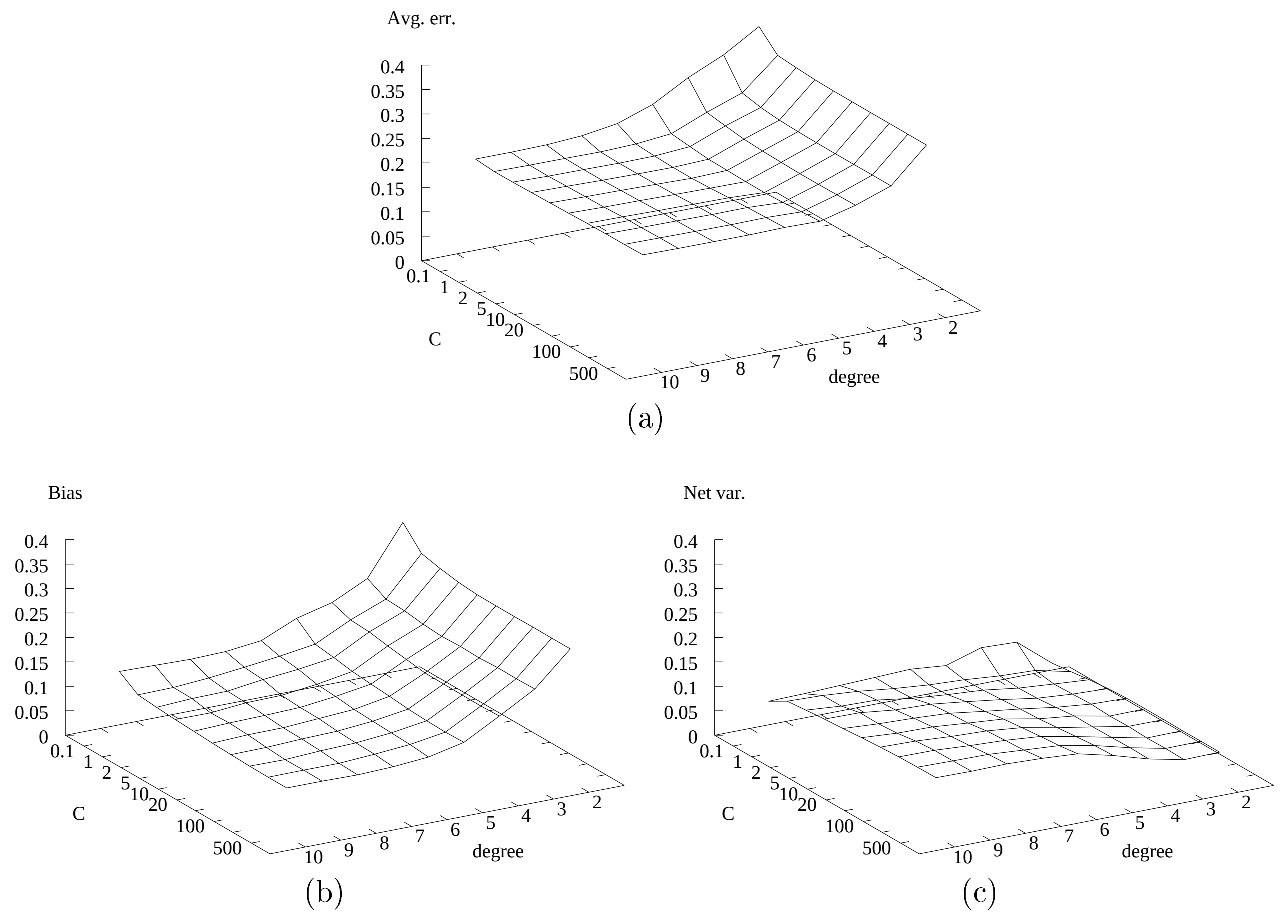
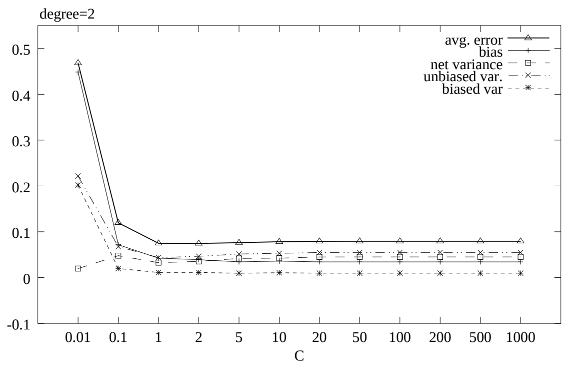


Figure 4.16: P2 data set. Error (a) and its decomposition in bias (b) and net variance (c), varying both C and the polynomial degree.

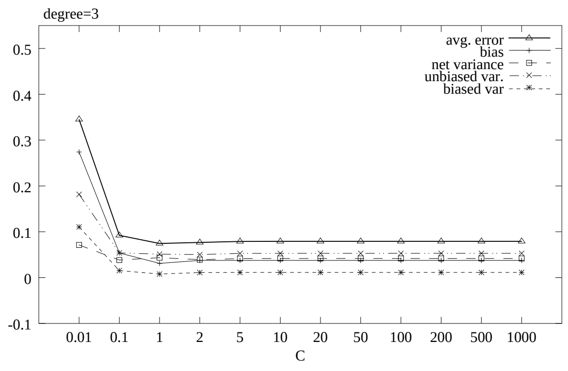
the classical opposite trends with respect to the degree. This is not the case with other data sets (see, e.g. Fig. 4.15).

For large values of C bias and net–variance tend to converge, as a result of the bias reduction and net–variance increment (Fig. 4.19), or because both stabilize at similar values (Fig. 4.17).

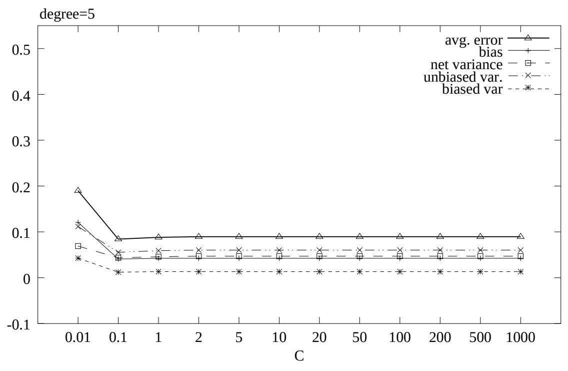
In dot–product SVMs bias and net–variance show opposite trends: bias decreases, while net–variance and unbiased variance tend to increase with C (Fig. 4.20). On the data set P2 this trend is not observed, as in this task the bias is very high and the SVM does not perform better than random guessing (Fig. 4.20a). The minimum of the average loss for relatively low values of C is the result of the decrement of the bias and the increment of the



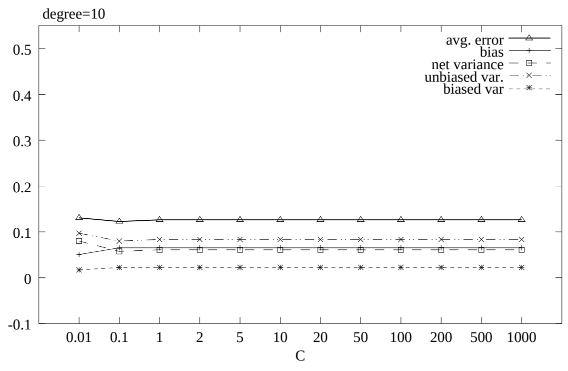
(a)



(b)



(c)



(d)

Figure 4.17: Letter-Two data set. Bias-variance decomposition of error in bias, net variance, unbiased and biased variance in polynomial SVM, while varying C and for some polynomial degrees: (a) $degree = 2$, (b) $degree = 3$, (c) $degree = 5$, (d) $degree = 10$

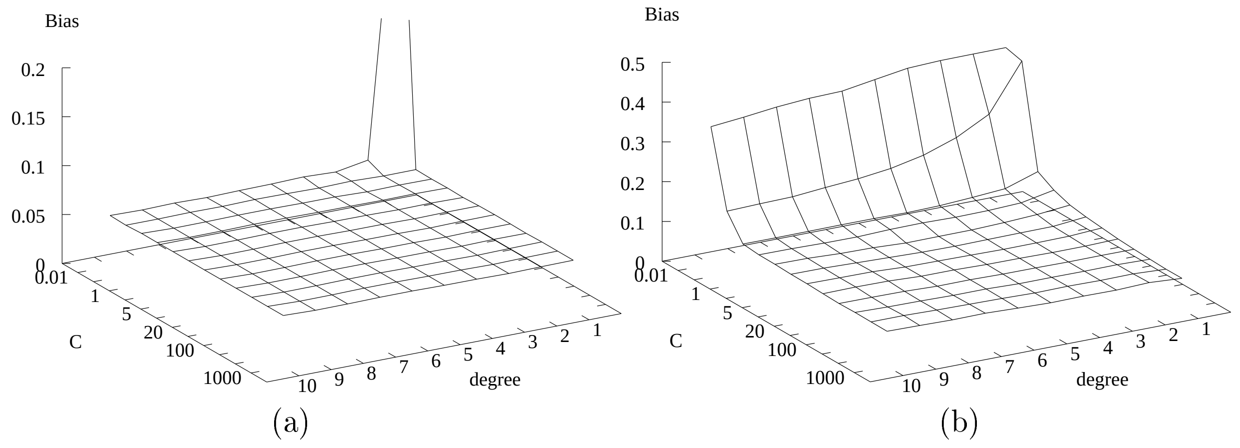


Figure 4.18: Bias in polynomial SVMs with (a) Waveform and (b) Spam data sets, varying both C and polynomial degree.

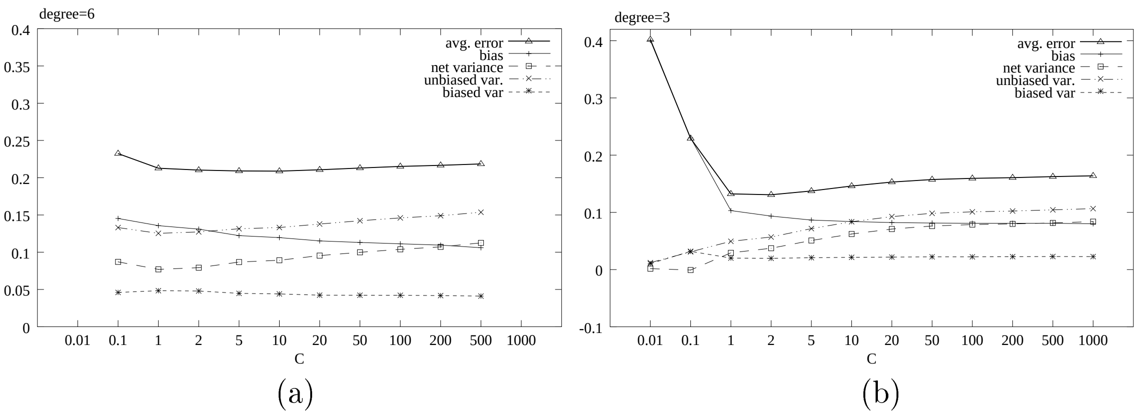


Figure 4.19: Bias-variance decomposition of error in bias, net variance, unbiased and biased variance in polynomial SVM, varying C: (a) P2 data set with $degree = 6$, (b) Spam data set with $degree = 3$.

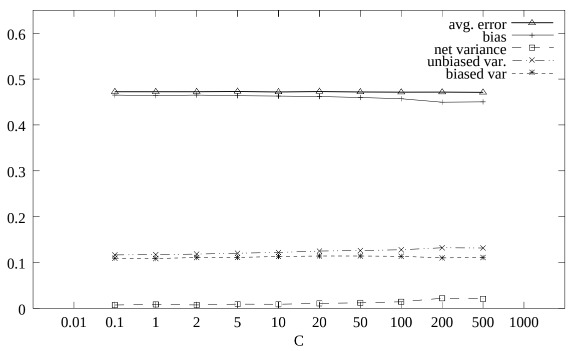
net–variance: it is achieved usually before the crossover of bias and net–variance curves and before the stabilization of the bias and the net–variance for large values of C. The biased variance remains small independently of C.

4.3.3 Comparing kernels

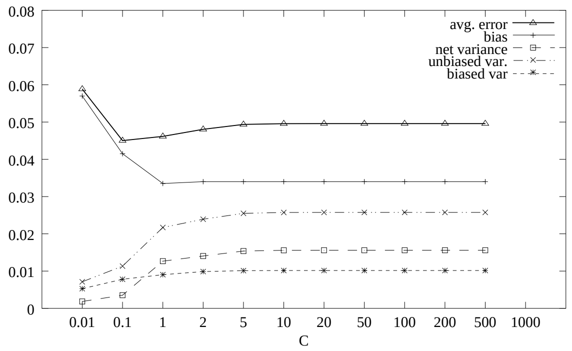
In this section we compare the bias–variance decomposition of the error with respect to the C parameter, considering gaussian, polynomial and dot–product kernels. For each kernel and for each data set the best results are selected. Tab. 4.2 shows the best results achieved by the SVM, considering each kernel and each data set used in the experiments. Interestingly enough in 3 data sets there are not significant differences in the error (Waveform, Letter-Two with added noise and Spam), but there are differences in bias, net–variance, unbiased or biased variance. In the other data sets gaussian kernels outperform polynomial and dot–product kernels, as bias, net–variance or both are lower. Considering bias and net–variance, in some cases they are lower for polynomial or dot–product kernel, showing that different kernels learn in different ways with different data.

Considering the data set *P2* (Fig. 4.21 a, c, e), RBF-SVM achieves the best results, as a consequence of a lower bias. Unbiased variance is comparable between polynomial and gaussian kernel, while net–variance is lower, as biased variance is higher for polynomial-SVM. In this task the bias of dot–product SVM is very high. Also in the data set *Musk* (Fig. 4.21 b, d, f) RBF-SVM obtains the best results, but in this case unbiased variance is responsible for this fact, while bias is similar. With the other data sets the bias is similar between RBF-SVM and polynomial-SVM, but for dot–product SVM often the bias is higher (Fig. 4.22 b, d, f).

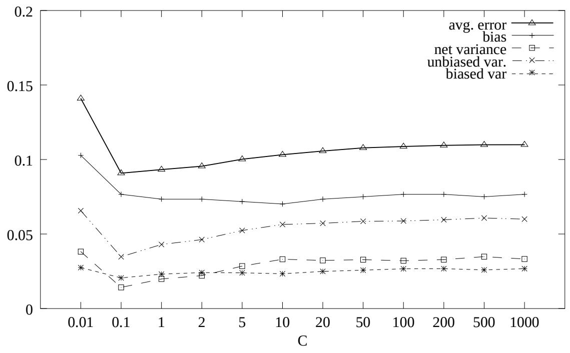
Interestingly enough RBF-SVM seems to be more sensible to the C value with respect to both polynomial and dot–product SVM: for $C < 0.1$ in some data sets the bias is much higher (Fig. 4.22 a, c, e). With respect to C bias and unbiased variance show sometimes opposite trends, independently of the kernel: bias decreases, while unbiased variance increases, but this does not occur in some data sets. We outline also that the shape of the error, bias and variance curves is similar between different kernels in all the considered data sets: that is, well-tuned SVM having different kernels tend to show similar trends of the bias and variance curves with respect to the C parameter.



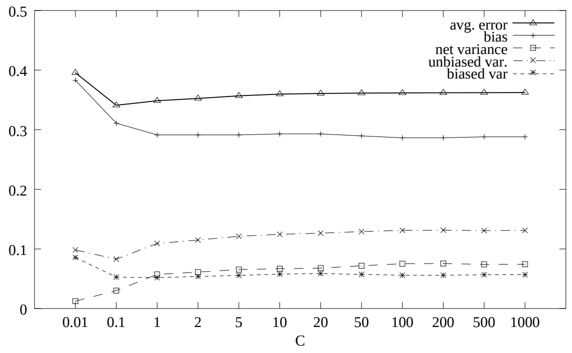
(a)



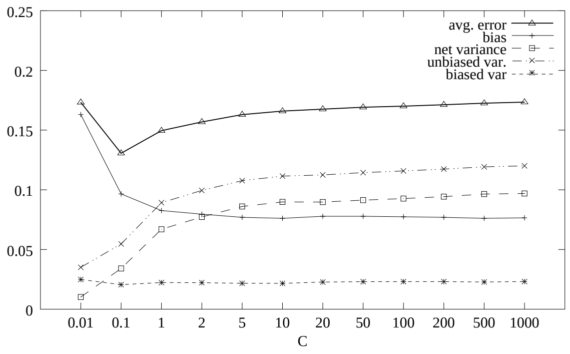
(b)



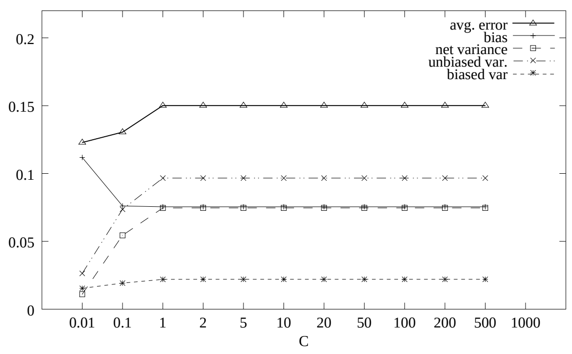
(c)



(d)



(e)



(f)

Figure 4.20: Bias-variance decomposition of error in bias, net variance, unbiased and biased variance in dot-product SVM, varying C: (a) P2, (b) Grey-Landsat, (c) Letter-Two, (d) Letter-Two with added noise, (e) Spam, (f) Musk.

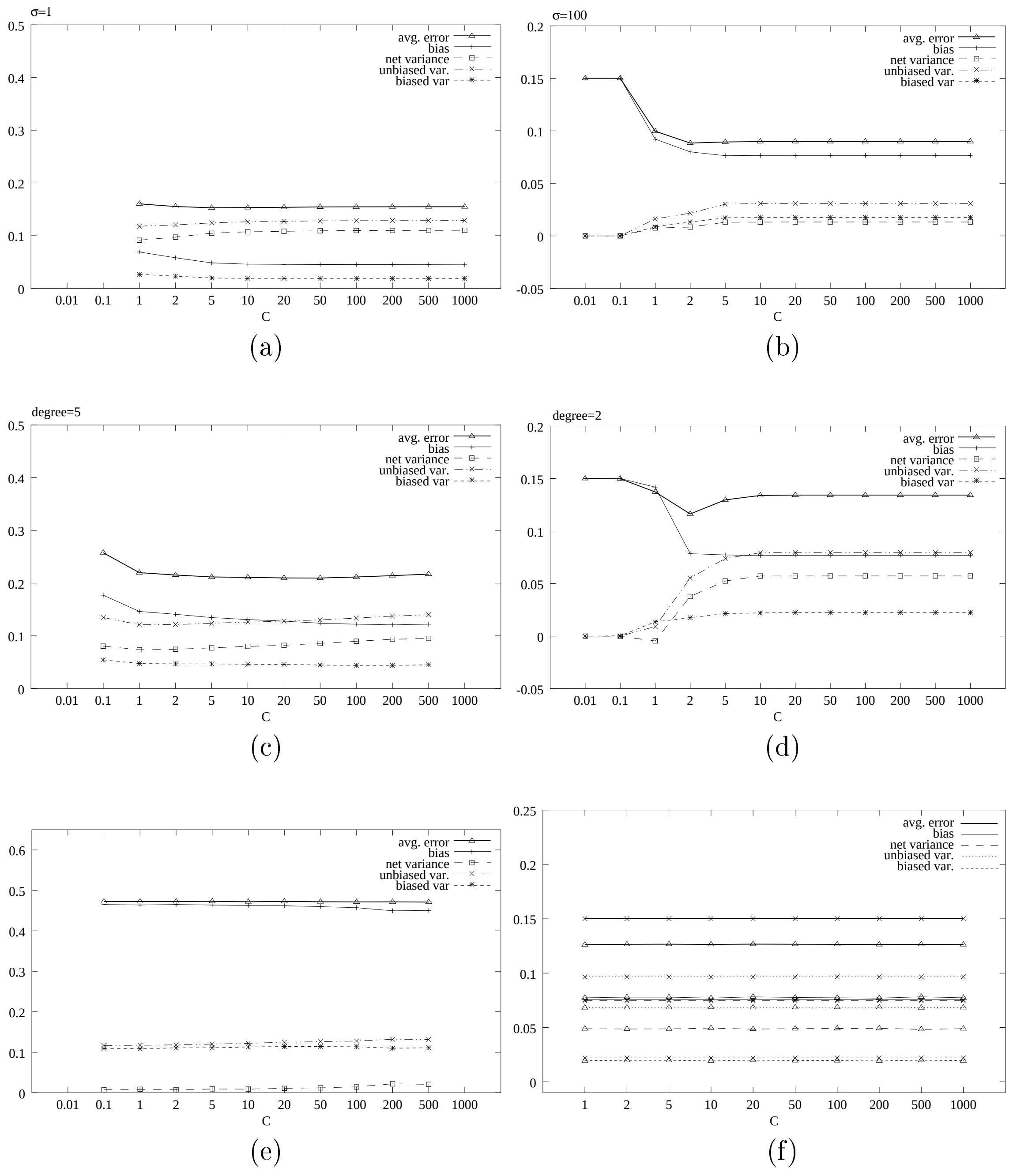
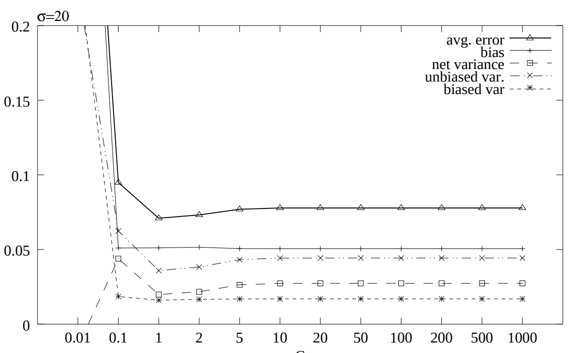
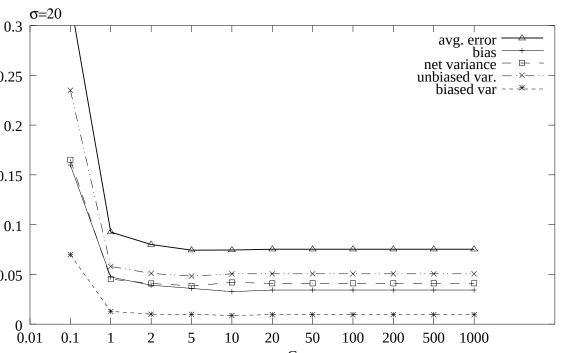


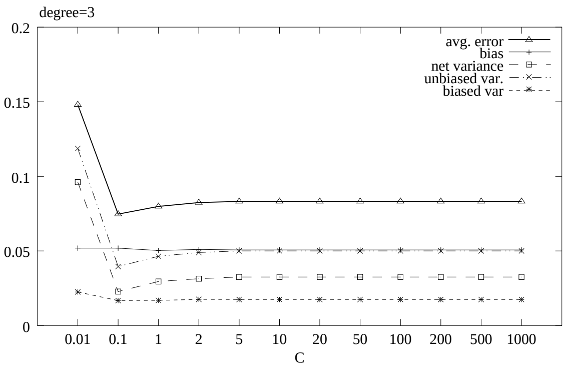
Figure 4.21: Bias-variance decomposition of error in bias, net variance, unbiased and biased variance with respect to C, considering different kernels. (a) P2, gaussian; (b) Musk, gaussian (c) P2, polynomial; (d) Musk, polynomial; (e) P2, dot–product; (f) Musk, dot–product.



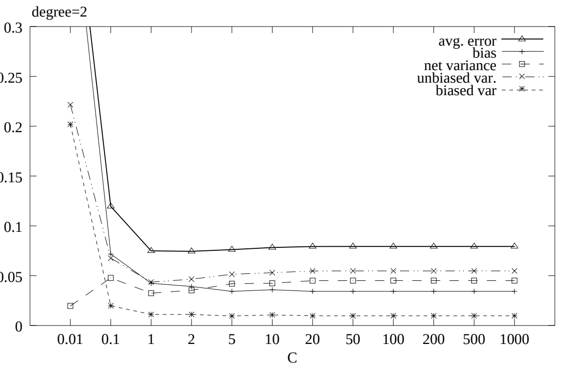
(a)



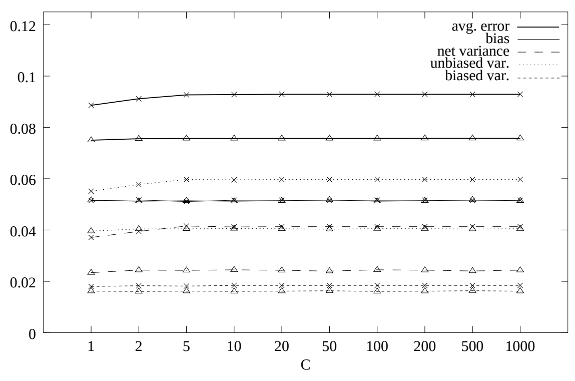
(b)



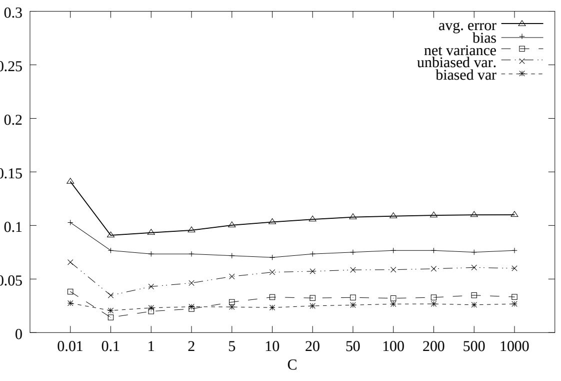
(c)



(d)



(e)



(f)

Figure 4.22: Bias-variance decomposition of error in bias, net variance, unbiased and biased variance with respect to C, considering different kernels. (a) Waveform, gaussian; (b) Letter-Two, gaussian (c) Waveform, polynomial; (d) Letter-Two, polynomial; (e) Waveform, dot–product; (f) Letter-Two, dot–product.

Table 4.2: Compared best results with different kernels and data sets. RBF-SVM stands for SVM with gaussian kernel; Poly-SVM for SVM with polynomial kernel and D-prod SVM for SVM with dot-product kernel. Var unb. and Var. bias. stand for unbiased and biased variance.

	Parameters	**Avg. Error**	**Bias**	**Var. unb.**	**Var. bias.**	**Net Var.**
Data set *P2*						
RBF-SVM	$C = 20, \quad \sigma = 2$	0.1516	0.0500	0.1221	0.0205	0.1016
Poly-SVM	$C = 10, \quad degree = 5$	0.2108	0.1309	0.1261	0.0461	0.0799
D-prod SVM	$C = 500$	0.4711	0.4504	0.1317	0.1109	0.0207
Data set *Waveform*						
RBF-SVM	$C = 1, \quad \sigma = 50$	0.0706	0.0508	0.0356	0.0157	0.0198
Poly-SVM	$C = 1, \quad degree = 1$	0.0760	0.0509	0.0417	0.0165	0.0251
D-prod SVM	$C = 0.1$	0.0746	0.0512	0.0397	0.0163	0.0234
Data set *Grey-Landsat*						
RBF-SVM	$C = 2, \quad \sigma = 20$	0.0382	0.0315	0.0137	0.0069	0.0068
Poly-SVM	$C = 0.1, \ degree = 5$	0.0402	0.0355	0.0116	0.0069	0.0047
D-prod SVM	$C = 0.1$	0.0450	0.0415	0.0113	0.0078	0.0035
Data set *Letter-Two*						
RBF-SVM	$C = 5, \quad \sigma = 20$	0.0743	0.0359	0.0483	0.0098	0.0384
Poly-SVM	$C = 2, \quad degree = 2$	0.0745	0.0391	0.0465	0.0111	0.0353
D-prod SVM	$C = 0.1$	0.0908	0.0767	0.0347	0.0205	0.0142
Data set *Letter-Two with added noise*						
RBF-SVM	$C = 10, \quad \sigma = 100$	0.3362	0.2799	0.0988	0.0425	0.0563
Poly-SVM	$C = 1, \quad degree = 2$	0.3432	0.2799	0.1094	0.0461	0.0633
D-prod SVM	$C = 0.1$	0.3410	0.3109	0.0828	0.0527	0.0301
Data set *Spam*						
RBF-SVM	$C = 5, \quad \sigma = 100$	0.1263	0.0987	0.0488	0.0213	0.0275
Poly-SVM	$C = 2, \quad degree = 2$	0.1292	0.0969	0.0510	0.0188	0.0323
D-prod SVM	$C = 0.1$	0.1306	0.0965	0.0547	0.0205	0.0341
Data set *Musk*						
RBF-SVM	$C = 2, \quad \sigma = 100$	0.0884	0.0800	0.0217	0.0133	0.0084
Poly-SVM	$C = 2, \quad degree = 2$	0.1163	0.0785	0.0553	0.0175	0.0378
D-prod SVM	$C = 0.01$	0.1229	0.1118	0.0264	0.0154	0.0110

4.4 Characterization of Bias–Variance Decomposition of the Error

Despite the differences observed in different data sets, common trends of bias and variance can be individuated for each of the kernels considered in this study. Each kernel presents a specific characterization of bias and variance with respect to its specific parameters, as explained in the following sections.

4.4.1 Gaussian kernels

Error, bias, net–variance, unbiased and biased variance show a common trend in the 7 data sets we used in the experiments. Some differences, of course, arise in the different data sets, but we can distinguish three different regions in the error analysis of *RBF-SVM*, with respect to increasing values of σ (Fig. 4.23):

1. **High bias region**. For low values of σ, error is high: it depends on a high bias. Net–variance is about 0 as biased and unbiased variance are equivalent. In this region there are no remarkable fluctuations of bias and variance: both remain constant, with high values of bias and comparable values of unbiased and biased variance, leading to net–variance values near to 0. In some cases biased and unbiased variance are about equal, but different from 0, in other cases they are equal, but near to 0.

2. **Transition region**. Suddenly, for a critical value of σ, the bias decreases rapidly. This critical value depends also on C: for very low values of C, we have no learning, then for higher values the bias drops. Higher values of C cause the critical value of σ to decrease (Fig. 4.5 (b) and 4.6). In this region the increase in net–variance is less than the decrease in bias: so the average error decreases. The boundary of this region can be determined at the point where the error stops decrementing. This region is characterized also by a particular trend of the net–variance. We can distinguish two main behaviours:

   (a) **Wave-shaped net–variance**. Net–variance first increases and then decreases, producing a wave-shaped curve with respect to σ. The initial increment of the net–variance is due to the simultaneous increment of the unbiased variance and decrement of the biased variance. In the second part of the transition region, biased variance stabilizes and unbiased variance decreases, producing a parallel decrement of the net–variance. The rapid decrement of the error with σ is due to the rapid decrement of the bias, after which the bias stabilizes and the further decrement of the error with σ is determined by the net–variance reduction (Fig. 4.5c, 4.6).

   (b) **Semi-wave-shaped net–variance**. In other cases the net–variance curve with σ is not so clearly wave-shaped: the descending part is very reduced (Fig. 4.6 e, f). In particular in the musk data set we have a continuous increment of the net–variance (due to the continuous growing of the unbiased variance with σ), and no wave-shaped curve is observed (at least for $C > 10$, Fig. 4.12 d).

In both cases the increment of the net–variance is slower than the increment in bias: so the average error decreases.

3. **Stabilized region**. This region is characterized by small or no variations in bias and net–variance. For high values of σ both bias and net–variance stabilize and the average error is constant (Fig. 4.5, 4.6). In other data sets the error increases with σ, because of the increment of the bias (Fig. 4.12 a,b) or the unbiased variance (Fig. 4.12 c,d).

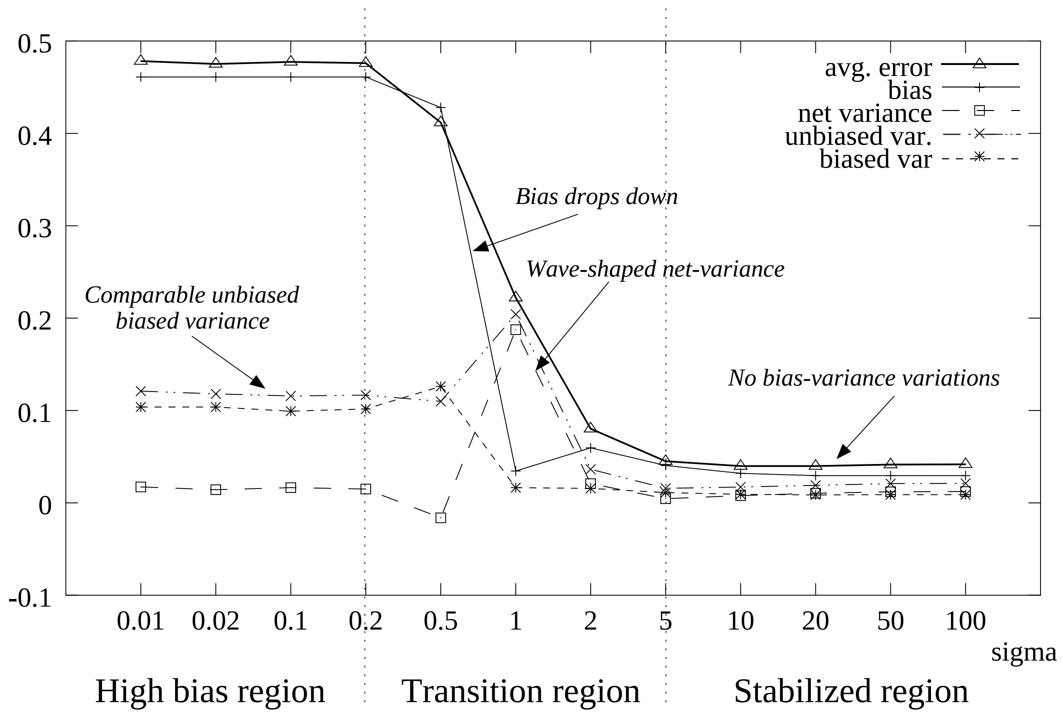


Figure 4.23: The 3 regions of error in RBF-SVM with respect to σ.

In the first region, bias rules SVM behavior: in most cases the bias is constant and close to 0.5, showing that we have a sort of random guessing, without effective learning. It appears that the area of influence of each support vector is too small (Fig. 4.8), and the learning machine overfits the data. This is confirmed by the fact that in this region the training error is about 0 and almost all the training examples are support vectors.

In the transition region, the SVM starts to learn, adapting itself to the data characteristics. Bias rapidly goes down (at the expenses of a growing net–variance), but for higher values of σ (in the second part of the transition region), sometimes net–variance also goes down, working to lower the error(Fig. 4.6).

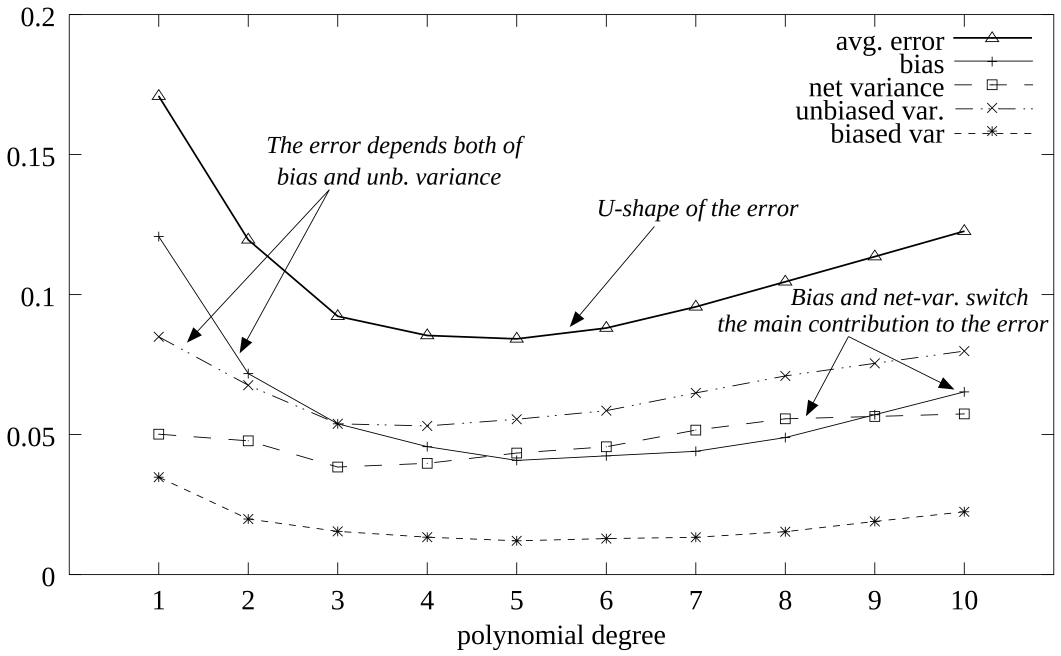


Figure 4.24: Behaviour of polynomial SVM with respect of the bias–variance decomposition of the error.

Even if the third region is characterized by no variations in bias and variance, sometimes for low values of C, the error increases with σ (Fig. 4.11 a, 4.13 a), as a result of the bias increment; on the whole RBF-SVMs are sensitive to low values of C: if C is too low, then bias can grow quickly. High values of C lower the bias (Fig. 4.13 c, d).

4.4.2 Polynomial and dot-product kernels

For polynomial and dot–product SVMs, we have also characterized the behavior of SVMs in terms of average error, bias, net–variance, unbiased and biased variance, even if we cannot distinguish between different regions clearly defined.

However, common trends of the error curves with respect to the polynomial degree, considering bias, net–variance and unbiased and biased variance can be noticed.

The average loss curve shows in general a U shape with respect to the polynomial degree, and this shape may depend on both bias and unbiased variance or in some cases mostly on the unbiased variance according to the characteristics of the data set. From these general observations we can schematically distinguish two main global pictures of the behaviour of polynomial SVM with respect to the bias–variance decomposition of the error:

1. **Error curve shape bias–variance dependent**.
   In this case the shape of the error curve is dependent both on the unbiased variance and the bias. The trend of bias and net–variance can be symmetric or they can also have non coincident paraboloid shape, depending on C parameter values (Fig. 4.15 c, d and 4.16). Note that bias and net variance show often opposite trends (Fig. 4.16).

2. **Error curve shape unbiased variance dependent**.
   In this case the shape of the error curve is mainly dependent on the unbiased variance. The bias (and the biased variance) tend to be degree independent, especially for high values of C (Fig. 4.15 a, b) .

Fig. 4.24 schematically summarizes the main characteristics of the bias–variance decomposition of error in polynomial SVM. Note however that the error curve depends for the most part on both variance and bias: the prevalence of the unbiased variance (Fig. 4.15 a, b) or the bias seems to depend mostly on the distribution of the data. The increment of

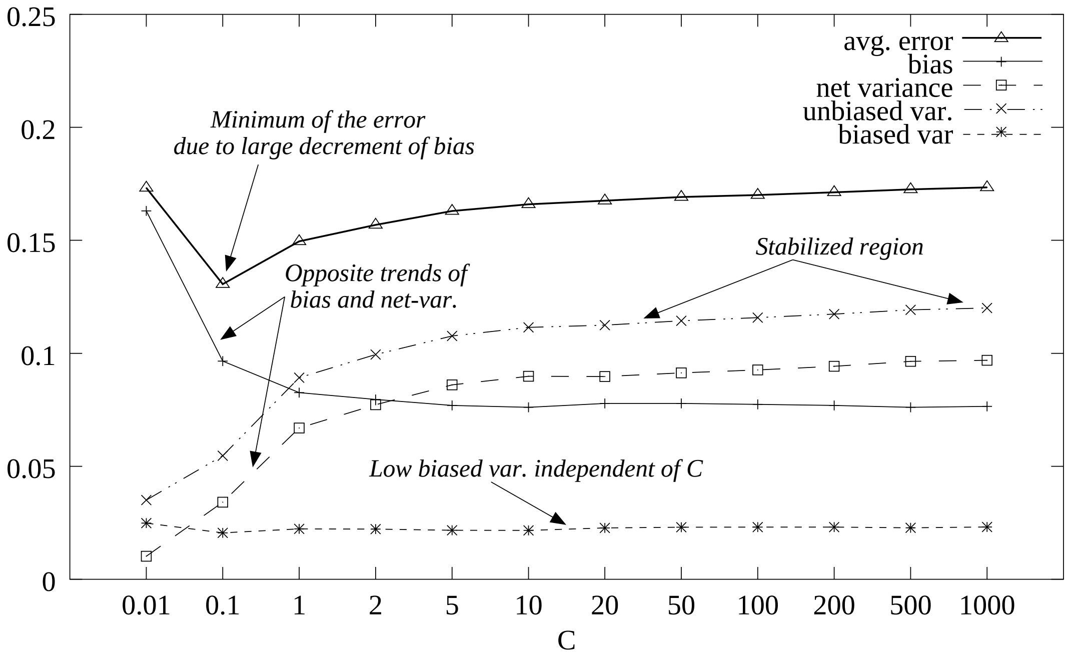


Figure 4.25: Behaviour of the dot–product SVM with respect of the bias–variance decomposition of the error.

the values of C tends to flatten the U shape of the error curve: in particular for large C values bias becomes independent with respect to the degree (Fig. 4.18). Moreover the C parameter plays also a regularization role (Fig. 4.19)

Dot–product SVM are characterized by opposite trends of bias and net–variance: bias decrements, while net–variance grows with respect to C; then, for higher values of C both stabilize. The combined effect of these symmetric curves produces a minimum of the error for low values of C, as the initial decrement of bias with C is larger than the initial increment of net–variance. Then the error slightly increases and stabilizes with C (Fig. 4.20). The shape of the net–variance curve is determined mainly by the unbiased variance: it increases and then stabilizes with respect to C. On the other hand the biased variance curve is flat, remaining small for all values of C. A schematic picture of this behaviour is given in Fig. 4.25.

Chapter 5

Bias–variance analysis in random aggregated and bagged ensembles of SVMs

Methods based on resampling techniques, and in particular on bootstrap aggregating (bagging) of sets of base learners trained on repeated bootstrap samples drawn from a given learning set, have been introduced in the nineties by Breiman [15, 16, 17].

The effectiveness of this approach, that have been shown to improve the accuracy of a single predictor [15, 63, 128, 44], is to be found in its property of reducing the variance component of the error.

Bagging can be seen as an approximation of *random aggregating*, that is a process by which base learners, trained on samples drawn accordingly to an unknown probability distribution from the entire universe population, are aggregated through majority voting (classification) or averaging between them (regression).

Breiman showed that in regression problem, aggregation of predictors always improve the performance of single predictors, while in classification problems this is not always the case, if poor base predictors are used [15].

The improvement depends on the stability of the base learner: random aggregating and bagging are effective with unstable learning algorithms, that is when small changes in the training set can result in large changes in the predictions of the base learners.

Random aggregating always reduce variance in regression and in classification with reasonably good base classifiers, while bias remains substantially unchanged. With bagging we can have a variance reduction but bias can also slightly increases, as the average sample size used by each base learner is only about 2/3 of the training set from which the samples are bootstrapped.

In general bagging unstable base learners is a good idea. As bagging is substantially a variance reduction method, we could also select low bias base learner in order to reduce both the bias and variance components of the error.

Random aggregating is only a theoretical ensemble method, as we need the entire universe of data from which random samples are drawn according to an usually unknown probability distribution. But with very large data sets, using a uniform probability distribution and undersampling techniques, we can simulate random aggregating (with the assumptions that the very large available data set and the uniform probability distribution are good approximations respectively of the "universe" population and the unknown probability distribution).

In the next sections we discuss some theoretical issues about the relationships between random aggregating and bagging. Then we verify the theoretical results of the expected variance reduction in bagging and random aggregating performing an extended experimental bias–variance decomposition of the error in bagged and random aggregated ensembles of SVMs. Finally, we consider an approximation of random aggregated ensembles for very large scale data mining problems.

5.1 Random aggregating and bagging

Let D be a set of m points drawn identically and independently from U according to P, where U is a population of labeled training data points $(\mathbf{x}_j, t_j)$, and $P(\mathbf{x}, t)$ is the joint distribution of the data points in U, with $\mathbf{x} \in \mathbb{R}^d$.

Let $\mathcal{L}$ be a learning algorithm, and define $f_D = \mathcal{L}(D)$ as the predictor produced by $\mathcal{L}$ applied to a training set D. The model produces a prediction $f_D(\mathbf{x}) = y$. Suppose that a sequence of learning sets $\{D_k\}$ is given, each i.i.d. from the same underlying distribution P. According to [15] we can aggregate the f_D trained with different samples drawn from U to get a better predictor $f_A(\mathbf{x}, P)$. For regression problems $t_j \in \mathbb{R}$ and $f_A(\mathbf{x}, P) = E_D[f_D(\mathbf{x})]$, where $E_D[\cdot]$ indicates the expected value with respect to the distribution of D, while in classification problems $t_j \in S \subset \mathbb{N}$ and $f_A(\mathbf{x}, P) = \arg\max_j |\{k | f_{D_k}(\mathbf{x}) = j\}|$.

As the training sets D are randomly drawn from U, we name the procedure to build f_A *random aggregating*. In order to simplify the notation, we denote $f_A(\mathbf{x}, P)$ as $f_A(\mathbf{x})$.

5.1.1 Random aggregating in regression

If T and $\mathbf{X}$ are random variables having joint distribution P the expected squared loss EL for the single predictor $f_D(\mathbf{X})$ is:

$$EL = E_D[E_{T,\mathbf{X}}[(T - f_D(\mathbf{X}))^2]] \tag{5.1}$$

while the expected squared loss EL_A for the aggregated predictor is:

$$EL_A = E_{T,\mathbf{X}}[(T - f_A(\mathbf{X}))^2] \tag{5.2}$$

Developing the square in eq. 5.1 we have:

$$\begin{aligned} EL &= E_D[E_{T,\mathbf{X}}[T^2 + f_D^2(\mathbf{X}) - 2Tf_D(\mathbf{X})]] \\ &= E_T[T^2] + E_D[E_{\mathbf{X}}[f_D^2(\mathbf{X})]] - 2E_T[T]E_D[E_{\mathbf{X}}[f_D(\mathbf{X})]] \\ &= E_T[T^2] + E_{\mathbf{X}}[E_D[f_D^2(\mathbf{X})]] - 2E_T[T]E_{\mathbf{X}}[f_A(\mathbf{X})] \end{aligned} \tag{5.3}$$

In a similar way, developing the square in eq. 5.2 we have:

$$\begin{aligned} EL_A &= E_{T,\mathbf{X}}[T^2 + f_A^2(\mathbf{X}) - 2Tf_A(\mathbf{X})] \\ &= E_T[T^2] + E_{\mathbf{X}}[f_A^2(\mathbf{X})] - 2E_T[T]E_{\mathbf{X}}[f_A(\mathbf{X})] \\ &= E_T[T^2] + E_{\mathbf{X}}[E_D[f_D(\mathbf{X})]^2] - 2E_T[T]E_{\mathbf{X}}[f_A(\mathbf{X})] \end{aligned} \tag{5.4}$$

Let be $Z = E_D[f_D(\mathbf{X})]$. Using $E[Z^2] \geq E[Z]^2$, considering eq. 5.3 and 5.4 we have that $E_D[f_D^2(\mathbf{X})] \geq E_D[f_D(\mathbf{X})]^2$ and hence $EL \geq EL_A$.

The reduction of the error in random aggregated ensembles depends on how much differ the two terms $E_{\mathbf{X}}[E_D[f_D^2(\mathbf{X})]]$ and $E_{\mathbf{X}}[E_D[f_D(\mathbf{X})]^2]$ of eq. 5.3 and 5.4. As outlined by Breiman, the effect of instability is clear: if $f_D(\mathbf{X})$ does not change too much with replicate data sets D, the two terms will be nearly equal and aggregation will not help. The more variable the $f_D(\mathbf{X})$ are, the more improvement aggregation may produce.

In other words the reduction of the error depends on the instability of the prediction, that is on how unequal the two sides of eq. 5.5 are:

$$E_D[f_D(\mathbf{X})]^2 \leq E_D[f_D^2(\mathbf{X})] \tag{5.5}$$

There is a strict relationship between the instability and the variance of the base predictor. Indeed the variance $V(\mathbf{X})$ of the base predictor is:

$$\begin{aligned} V(\mathbf{X}) &= E_D[(f_D(\mathbf{X}) - E_D[f_D(\mathbf{X})])^2] \\ &= E_D[f_D^2(\mathbf{X}) + E_D[f_D(\mathbf{X})]^2 - 2f_D(\mathbf{X})E_D[f_D(\mathbf{X})]] \\ &= E_D[f_D^2(\mathbf{X})] - E_D[f_D(\mathbf{X})]^2 \end{aligned} \tag{5.6}$$

Comparing eq.5.5 and 5.6 we see that higher the instability of the base classifiers, higher their variance is. The reduction of the error in random aggregation is due to the reduction of the variance component (eq. 5.6) of the error, as $V(\mathbf{X})$ will be small if and only if $E_D[f_D^2(\mathbf{X})] > E_D[f_D(\mathbf{X})]^2$, that is if and only if the base predictor is unstable (eq. 5.5).

5.1.2 Random aggregating in classification

With random aggregation of base classifiers, the same behaviour regarding stability holds, but in this case a more complex situation arises.

Indeed let be $f_D(\mathbf{X})$ a base classifier that predicts a class label $t \in \mathcal{C}, \mathcal{C} = \{1, \ldots, C\}$, and let be $\mathbf{X}$ a random variable as in previous regression case and T a random variable with values in $\mathcal{C}$.

Then the probability $p(D)$ of correct classification for a fixed data set D, considering a non deterministic assignment for the labels of the class, is:

$$p(D) = P(f_D(\mathbf{X}) = T) = \sum_{j=1}^{C} P(f_D(\mathbf{X}) = j | T = j) P(T = j) \tag{5.7}$$

In order to make independent the probability p of correct classification from the choice of a specific learning set we average over D:

$$\begin{aligned} p &= \sum_{j=1}^{C} E_D[P(f_D(\mathbf{X}) = j | T = j)] P(T = j) \\ &= \sum_{j=1}^{C} \int P(f_D(\mathbf{X}) = j | \mathbf{X} = \mathbf{x}, T = j) P(T = j | \mathbf{X} = \mathbf{x}) P_{\mathbf{X}}(d\mathbf{x}) \end{aligned} \tag{5.8}$$

Recalling that $f_A(\mathbf{X}) = \arg\max_i P_D(f_D(\mathbf{x}) = i)$, the probability p_A of correct classification for random aggregation is:

$$\begin{aligned} p_A &= \sum_{j=1}^{C} P(f_A(\mathbf{X}) = j | T = j) P(T = j) \\ &= \sum_{j=1}^{C} \int P(f_A(\mathbf{X}) = j | T = j) P(T = j | \mathbf{X} = \mathbf{x}) P_{\mathbf{X}}(d\mathbf{x}) \\ &= \sum_{j=1}^{C} \int P(\arg\max_i [P_D(f_D(\mathbf{X}) = i)] = j | T = j) P(T = j | \mathbf{X} = \mathbf{x}) P_{\mathbf{X}}(d\mathbf{x}) \end{aligned}$$

$$= \sum_{j=1}^{C} \int I(\arg\max_i [P_D(f_D(\mathbf{X}) = i] = j) P(T = j|\mathbf{X} = \mathbf{x}) P_{\mathbf{X}}(d\mathbf{x}) \tag{5.9}$$

where I is the indicator function.

The optimal prediction for a pattern $\mathbf{x}$ is the Bayesian prediction:

$$B^*(\mathbf{x}) = \arg\max_j P(T = j|\mathbf{X} = \mathbf{x}) \tag{5.10}$$

We split now the patterns in a set O corresponding to the optimal predictions performed by the aggregated classifier and in a set O' corresponding to non-optimal predictions. The set O of the optimally classified patterns is:

$$O = \{\mathbf{x}| \arg\max_j P(T = j|\mathbf{X} = \mathbf{x}) = \arg\max_j P_D(f_D(\mathbf{x}) = j)\}$$

According to the proposed partition of the data we can split the probability p_A of correct classification for random aggregation in two terms:

$$p_A = \int_{\mathbf{x}\in O} \max_j P(T = j|\mathbf{X} = \mathbf{x}) P_{\mathbf{X}}(d\mathbf{x}) + \int_{\mathbf{x}\in O'} \sum_{j=1}^{C} I(f_A(\mathbf{x}) = j) P(T = j|\mathbf{X} = \mathbf{x}) P_{\mathbf{X}}(d\mathbf{x}) \tag{5.11}$$

If $\mathbf{x} \in O$ we have:

$$\arg\max_j P(T = j|\mathbf{X} = \mathbf{x}) = \arg\max_j P_D(f_D(\mathbf{x}) = j) \tag{5.12}$$

In this case, considering eq. 5.8 and 5.9:

$$\sum_{j=1}^{C} P(f_D(\mathbf{X}) = j|T = j) P(T = j|\mathbf{X} = \mathbf{x}) \leq \arg\max_j P_D(f_D(\mathbf{x}) = j)$$

and hence $p_A(\mathbf{X}) \geq p(\mathbf{X})$. On the contrary, if $\mathbf{x} \in O'$ eq. 5.12 does not hold, and it may occur that:

$$\sum_{j=1}^{C} I(f_A(\mathbf{x}) = j) P(T = j|\mathbf{X} = \mathbf{x}) < \sum_{j=1}^{C} P(f_D(\mathbf{X}) = j|T = j) P(T = j|\mathbf{X} = \mathbf{x})$$

As a consequence if the set O of the optimally predicted patterns is large, that is, if we have relatively good predictors, aggregation improves performances. On the contrary, if the set O' is large, that is if we have poor predictors, aggregation can worsen performances.

Summarizing, unlike regression, aggregating poor predictors can lower performances, whereas, as in regression, aggregating relatively good predictors can lead to better performances, as long as the base predictor is unstable [15].

5.1.3 Bagging

In most cases we dispose only of data sets of limited size, and moreover we do not know the probability distribution underlying the data. In these case we could try to simulate random aggregation by bootstrap replicates of the data [56] and successively aggregating the predictors trained on the bootstrapped data.

The *b*ootstrap *agg*regat*ing* method (*bagging*) [15] can be applied both to regression and classification problems: the only difference is in the aggregation phase.

Consider, for instance, a classification problem. Let $\mathcal{C}$ be the set of class labels. Let $\{D_j\}_{j=1}^n$ be the set of n bootstrapped samples drawn with replacement from the learning set $\mathcal{D}$ according to an uniform probability distribution. Let $f_{D_j} = \mathcal{L}(D_j)$ be the decision function of the classifier trained by a learning algorithm $\mathcal{L}$ using the bootstrapped sample D_j.

Then the classical decision function $f_B(\mathbf{x})$ applied for aggregating the base learners in bagging is [15]:

$$f_B(\mathbf{x}) = \arg\max_{c \in \mathcal{C}} \sum_{j=1}^{n} I(f_{D_j}(\mathbf{x}) = c) \tag{5.13}$$

where $I(z) = 1$ if the boolean expression z is true, otherwise $I(z) = 0$. In words, the bagged ensemble selects the most voted class.

In regression the aggregation is performed averaging between the real values computed by the real function valued base learners $g_{D_j} : \mathbb{R}^d \rightarrow \mathbb{R}$:

$$f_B(\mathbf{x}) = \frac{1}{n} \sum_{j=1}^{n} g_{D_j}(\mathbf{x}) \tag{5.14}$$

Fig. 5.1 show the pseudo-code for bagging.

The learning algorithm $\mathcal{L}$ generates an hypothesis $h_t : \mathcal{X} \rightarrow \mathcal{Y}$ using a sample D_t bootstrapped from $\mathcal{D}$, and h_{fin} is the final hypothesis computed by the bagged ensemble, aggregating the base learners through majority voting (Fig. 5.1).

Bagging shows the same limits of random aggregating: only if the base learners are unstable we can achieve reduction of the error with respect to the single base learners. Of course if the base learner is near to the Bayes error we cannot expect improvements by bagging.

Moreover *bagging* is an approximation of random aggregating, for at least two reasons.

```
Algorithm Bagging
Input arguments:
    - Data set $\mathcal{D} = \{\mathbf{z}_i = (\mathbf{x}_i, y_i)\}_{i=1}^n$, $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^d$, $y_i \in \mathcal{Y} = \{1, \ldots, k\}$
    - A learning algorithm $\mathcal{L}$
    - Number of iterations (base learners) $T$
Output:
    - Final hypothesis $h_{fin} : \mathcal{X} \rightarrow \mathcal{Y}$ computed by the ensemble.
begin
    for $t = 1$ to $T$
    begin
        $D_t = \texttt{Bootstrap\_replicate}(\mathcal{D})$
        $h_t = \mathcal{L}(D_t)$
    end
    $h_{fin}(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} \sum_{t=1}^T ||h_t(\mathbf{x}) = y||$
end.
```

Figure 5.1: Bagging for classification problems.

First, bootstrap samples are not real data samples: they are drawn from a data set D that is in turn a sample from the population U. On the contrary f_A uses samples drawn directly from U.

Second, bootstrap samples are drawn from D through an uniform probability distribution that is only an approximation of the unknown true distribution P.

For these reasons we can only hope that this is a good enough approximation to f_A that considerable variance reduction (eq. 5.2) will result [17].

Moreover with bagging each base learner, on the average, uses only 63.2% of the available data for training and so we can expect for each base learner a larger bias, as the effective size of the learning set is reduced. This can also affect the bias of the bagged ensemble that critically depends on the bias of the component base learners: we could expect sometimes a slight increment of the bias of the bagged ensemble with respect to the unaggregated predictor trained on the entire available training set.

Bagging is a variance reduction method, but we cannot expect so large decrements of variance as in random aggregating. The intuitive reason consists in the fact that in random aggregating the base learners use more variable training sets drawn from U according to the distribution P. In this way random aggregating exploits more information from the population U, while bagging can exploit only the information from a single data set D drawn from U, through bootstrap replicates of the data from D.

5.2 Bias–variance analysis in bagged SVM ensembles

In this section we deal with the problem of understanding the effect of bagging on bias and variance components of the error in SVMs. Our aim consists in getting insights into the way bagged ensembles learn, in order to characterize learning in terms of bias–variance components of the error.

In particular we try to verify the theoretical property of the expected variance reduction in bagging, through an extended experimental bias–variance decomposition of the error in bagged SVM ensembles.

The plan of the experiments, whose results are summarized in the next sections, is the following. We performed experiments with gaussian, polynomial and dot-product kernels. At first, for each kernel, we evaluated the expected error and its decomposition in bias, net-variance, unbiased and biased variance with respect to the learning parameters of the base learners. Then we analyzed the bias–variance decomposition as a function of the number of the base learners employed. Finally we compared bias and variance with respect to the learning parameters in bagged SVM ensembles and in the corresponding single SVMs, in order to study the effect of bootstrap aggregation on the bias and variance components of the error.

The next sections reported only some examples and a summary of the results of bias–variance analysis in bagged SVM ensembles. Full data, results and graphics of the experimentation on bias–variance analysis in bagged ensembles of SVMs are reported in [180].

5.2.1 Experimental setup

To estimate the decomposition of the error in bias, net-variance, unbiased and biased variance with bagged ensembles of SVMs, we performed a bias-variance decomposition of the error on the data sets described in Chap. 4. At first we split the data in a separated learning set $\mathcal{D}$ and testing set $\mathcal{T}$. Then we drew with replacement from $\mathcal{D}$ n samples S_i of size s, according to a uniform probability distribution. From each D_i, $1 \leq i \leq n$ we generated by bootstrap m replicates D_{ij}, collecting them in n different sets $\bar{D}_i = \{D_{ij}\}_{j=1}^{m}$.

We used the n sets $\bar{D}_i$ to train n bagged ensembles, each composed by m SVMs, each one trained with different bootstrapped data, repeating this process for all the considered SVM models. In order to properly compare the effect of different choices of the learning parameters on bias–variance decomposition of the error, each SVM model is represented by a different choice of the kernel type and parameters and is trained with the same sets $\bar{D}_i$, $1 \leq i \leq n$ of bootstrapped samples.

For each SVM model, bias–variance decomposition of the error is evaluated on a separated test set $\mathcal{T}$, significantly larger than the training sets, using the bagged ensembles trained on the n sets $\bar{D}_i$.

The experimental procedure we adopted to generate the data and to manage bias–variance analysis are summarized in Fig. 5.2 and 5.3. For more detailed information on how to compute bias–variance decomposition of the error see Chap. 3.2.

Procedure Generate_samples

```
Input arguments:
    - Data set $\mathcal{S}$
    - Number of samples $n$
    - Size of samples $s$
    - Number of bootstrap replicate $m$
Output:
    - Sets $\bar{D}_i = \{D_{ij}\}_{j=1}^{m}$, $1 \leq i \leq n$ of bootstrapped samples
begin procedure
    $[\mathcal{D}, \mathcal{T}] =$ Split($\mathcal{S}$)
    for $i = 1$ to $n$
    begin
        $D_i =$ Draw_with_replacement($\mathcal{D}, s$)
        $\bar{D}_i = \emptyset$
        for $j = 1$ to $m$
        begin
            $D_{ij} =$ Bootstrap_replicate($D_i$)
            $\bar{D}_i = \bar{D}_i + D_{ij}$
        end
    end
end procedure.
```

Figure 5.2: Procedure to generate samples to be used for bias–variance analysis in bagging

The procedure `Generate_samples` (Fig. 5.2) generates sets $\bar{D}_i$ of bootstrapped samples, drawing at first a sample D_i from the training set $\mathcal{D}$ according to an uniform probability distribution (procedure `Draw_with_replacement`) and then drawing from D_i m bootstrap replicates (procedure `Bootstrap_replicate`). Note that $\bar{D}_i$ is a set of sets, and the plus sign in Fig. 5.2 indicates that the entire set D_{ij} is added as a new element of the set of sets $\bar{D}_i$.

```
Procedure Bias–Variance_analysis
Input arguments:
    - Test set $\mathcal{T}$
    - Number of bagged ensembles $n$
    - Number of bootstrap replicate $m$
    - Set of learning parameters $\mathcal{A}$
Output:
    - Error, bias, net-variance, unbiased and biased variance $BV = \{bv(\alpha)\}_{\alpha \in \mathcal{A}}$
    of the bagged ensemble having base learners with learning parameters $\alpha \in \mathcal{A}$.
begin procedure
    For each $\alpha \in \mathcal{A}$
    begin
        Ensemble_Set($\alpha$) = $\emptyset$
        for $i = 1$ to $n$
        begin
            bag($\alpha$, $\bar{D}_i$) = Ensemble_train ($\alpha$, $\bar{D}_i$)
            Ensemble_Set($\alpha$) = Ensemble_Set($\alpha$) $\cup$ bag($\alpha$, $\bar{D}_i$)
        end
        $bv(\alpha)$ = Perform_BV_analysis(Ensemble_Set ($\alpha$), $\mathcal{T}$)
        $BV = BV \cup bv(\alpha)$
    end
end procedure.
```

Figure 5.3: Procedure to perform bias–variance analysis on bagged SVM ensembles

The procedure `Bias-Variance_analysis` (Fig. 5.3) trains different ensembles of bagged SVMs (procedure `Ensemble_train`) using the same sets of bootstrap samples generated through the procedure `Generate_samples`. Then bias–variance decomposition of the error is performed on the separated test set $\mathcal{T}$ using the previously trained bagged SVM ensembles (procedure `Perform_BV_analysis`).

In our experiments we employed gaussian, polynomial and dot-product kernels evaluating 110 different SVM models, considering different combinations of the type of the kernel and learning parameters for each data set. For each model we set $s = 100$, $n = 100$, $m = 60$, training for each data set $110 \times 100 = 11000$ bagged ensembles and a total of $110 \times 100 \times 60 = 660000$ different SVMs. Considering all the data sets we trained and tested about 80000 different bagged SVM ensembles and a total of about 5 millions of single SVMs.

5.2.2 Bagged RBF-SVM ensembles

In this section are reported the results of bias–variance analysis in bagged SVMs, using base learners with gaussian kernels.

5.2.2.1 Bias–variance decomposition of the error

The decomposition of the error is represented with respect to different values of σ and for fixed values of C. The error follows and "U" (Fig. 5.4 a and b) or a "sigmoid" trend (Fig. 5.4 c and d). This trend is visible also in other data sets (data not shown). In the *P2* data set we can observe opposite trends of bias and net-variance varying the σ parameter, while in *Letter-Two* for large values of σ both bias and net-variance remain constant. The net–variance, for small σ values is about 0, as unbiased and biased variance are about equal, then rapidly increases, as at the same time unbiased variance increases and biased variance decreases. Anyway, when net–variance increases the error goes down as the bias decreases more quickly. Then, for slightly larger values of σ the error diminishes, mainly for the reduction of the net-variance. For large values of σ (especially if C also is relatively large) both bias and net-variance stabilizes at low level and the error tends to be low, but in another data sets (e.g. *P2*) the bias increases inducing a larger error rate.

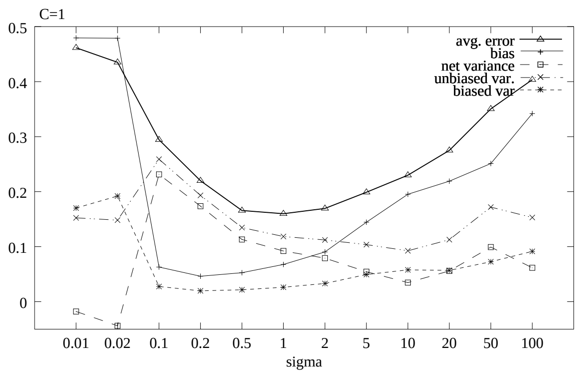
5.2.2.2 Decomposition with respect to the number of base learners

Considering the bias–variance decomposition of the error with respect to the number of base learners, we can observe that the error reduction arises in the first 10 iterations, especially for the reduction of the unbiased variance. The bias and the biased variance remain substantially unchanged for all the iterations (Fig. 5.5)

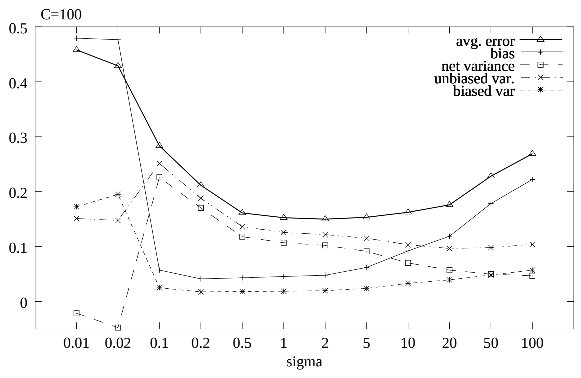
5.2.2.3 Comparison of bias–variance decomposition in single and bagged RBF-SVMs

Here are reported the graphics comparing bias–variance decomposition in single SVMs and bagged ensembles of SVMs. In all graphics of this section the data referred to single SVMs are labeled with crosses, while bagged SVMs are labeled with triangles. The corresponding quantities (e.g. bias, net-variance, etc.) are represented with the same type of line both in single and bagged SVMs.

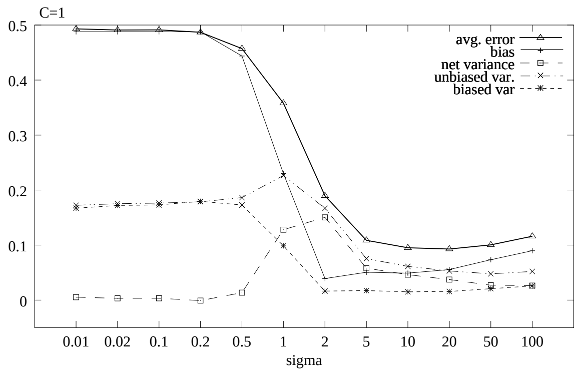
We analyze the relationships between bias-variance decomposition of the error in single and bagged RBF-SVMs for each different region that characterizes the bias-variance decomposition itself. In bagged SVM ensembles are also visible the three different regions



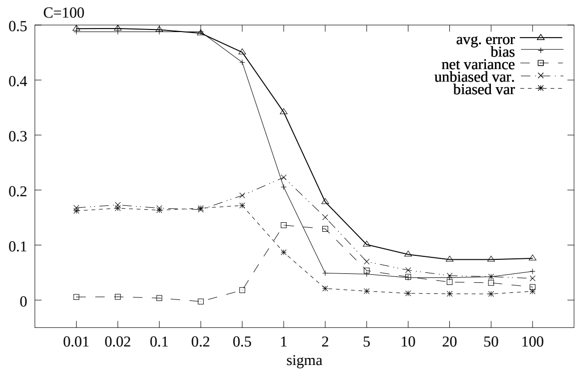
(a)



(b)



(c)



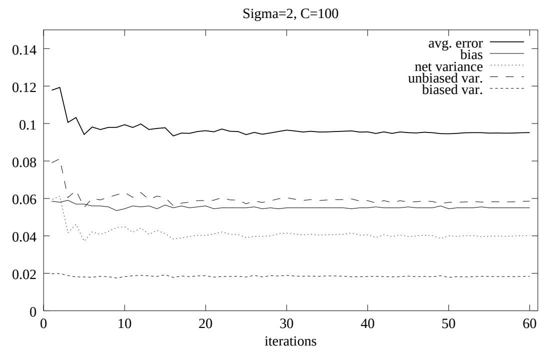
(d)

Figure 5.4: Bias-variance decomposition of error in bias, net variance, unbiased and biased variance in bagged RBF-SVMs, while varying σ and for some fixed values of C. P2 data set: (a) $C = 1$, (b) $C = 100$. Letter-Two data set: (c) $C = 1$, (d) $C = 100$

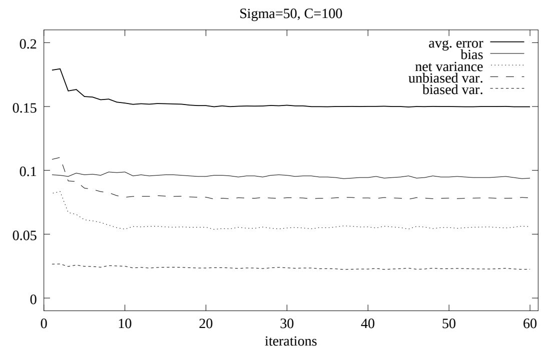
that characterize bias–variance decomposition in single SVMs (Sect. 4.4.1).

High bias region. In this region the error of single and bagged SVMs is about equal, and it is characterized by a very high bias. The net-variance is close to 0, because biased variance is about equal to the unbiased variance. In some cases they are both close to 0. In other cases they are equal but greater than 0 with slightly larger values in single that in in bagged SVMs (Fig. 5.6).

Transition region. In this region the bias goes down very quickly both in single and bagged SVMs. The net-variance maintains the wave-shape also in bagged SVMs, but it is slightly lower. The error drops down at about the same rate in single and bagged SVMs (Fig. 5.6 a and b).



(a)



(b)

Figure 5.5: Bias-variance decomposition of error in bias, net variance, unbiased and biased variance in bagged SVM RBF, with respect to the number of iterations. (a) Grey-Landsat data set (b) Spam data set.

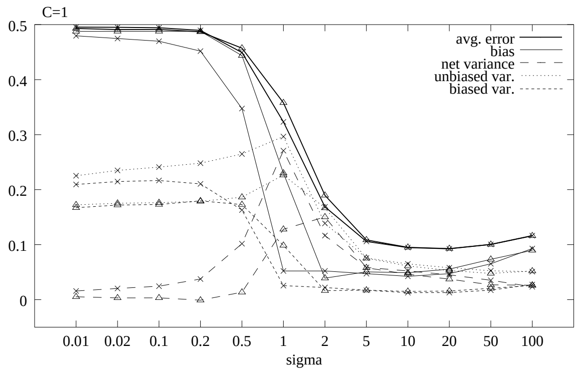
Stabilized region. For relatively large values of σ the net-variance tends to stabilize (Fig. 5.6). In this region the net-variance of the bagged SVMs is equal or less than the net-variance of the single SVMs, while bias remains substantially unchanged in both. With some data sets (Fig. 5.4 a and b) the bias tends to increase with σ, especially with low values of C. As a result, bagged SVMs show equal or lower average error with respect to single SVMs (Fig. 5.6)

5.2.3 Bagged polynomial SVM ensembles

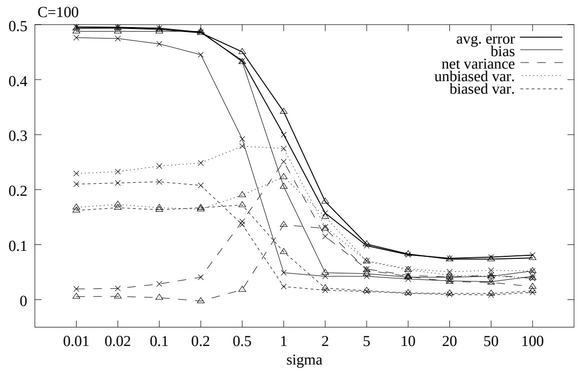
In this section are reported the results of the experiments to evaluate the decomposition of the error in bagged ensembles of polynomial SVMs.

5.2.3.1 Bias–variance decomposition of the error

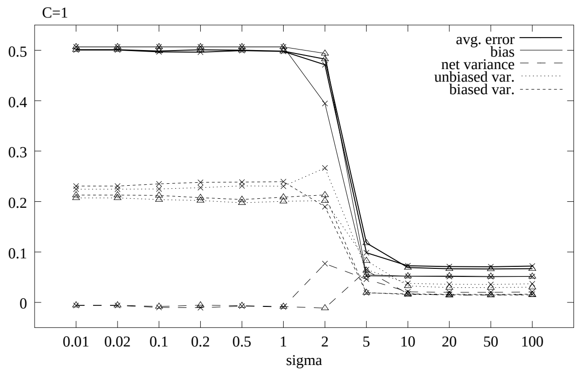
The decomposition of the error is represented with respect to different values of the polynomial degree and for fixed values of C. Also in bagged polynomial ensembles the error shows an "U" shape w.r.t. to the degree (Fig. 5.7), such as in single polynomial SVM (see Sect. 4.3.2). This shape depends both on bias and net-variance. The classical trade-off between bias and variance is sometimes noticeable (Fig. 5.7 b), but in other cases both bias and net-variance increase with the degree (Fig. 5.7 c and d). As a general rule for low



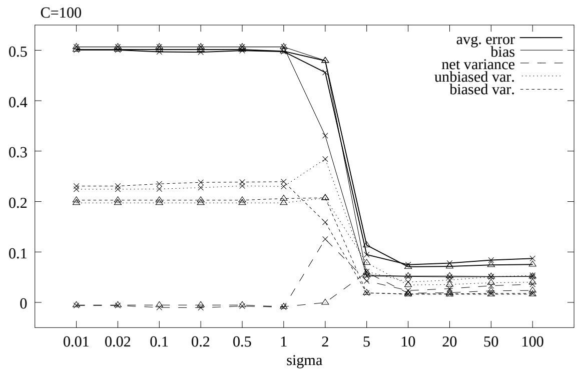
(a)



(b)



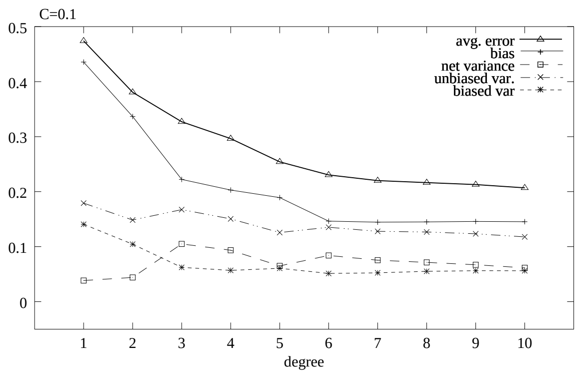
(c)



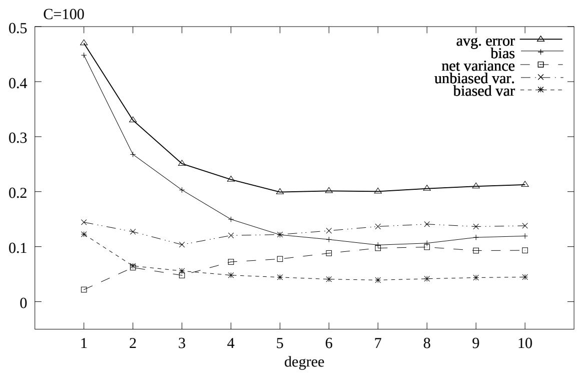
(d)

Figure 5.6: Comparison between bias-variance decomposition between single RBF-SVMs (lines labeled with crosses) and bagged SVM RBF ensembles (lines labeled with triangles), while varying σ and for some fixed values of C. Letter-Two data set: (a) $C = 1$, (b) $C = 100$. Waveform data set: (c) $C = 1$, (d) $C = 100$.

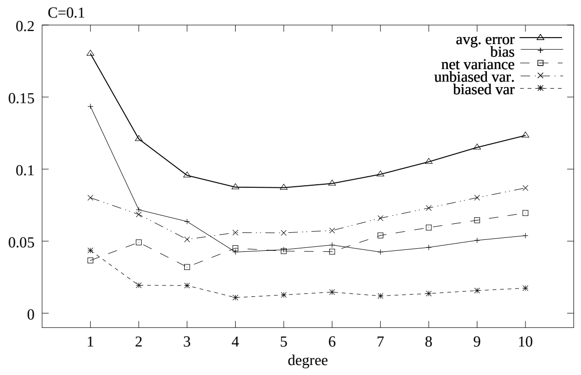
degree polynomial kernel the bias is relatively large and the net variance is low, while the opposite occurs with high degree polynomials (Fig. 5.7 a). The regularization parameter C plays also an important role: large C values tends to decrease the bias also for relatively low degree (Fig. 5.7 d). Of course these results depend also on the specific characteristics of the data sets.



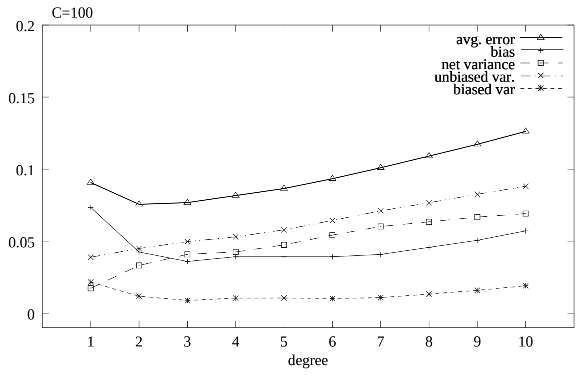
(a)



(b)



(c)

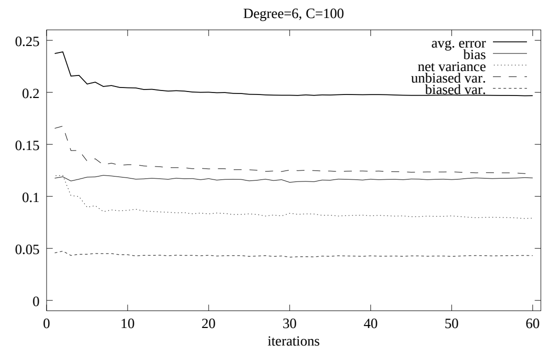


(d)

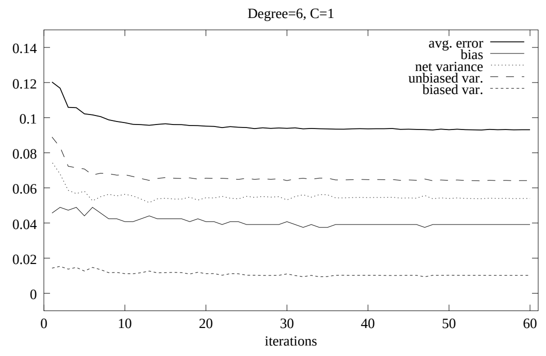
Figure 5.7: Bias-variance decomposition of error in bias, net variance, unbiased and biased variance in bagged polynomial SVM, while varying the degree and for some fixed values of C. P2 data set: (a) $C = 0.1$, (b) $C = 100$. Letter-Two data set: (c) $C = 0.1$, (d) $C = 100$

5.2.3.2 Decomposition with respect to the number of base learners

This section reports data and graphs about the decomposition of bias–variance in bagged SVMs with respect to the number of iterations of bagging, that is the number of base learners used. The error decreases in first 10 iterations, for the reduction of the unbiased variance, while bias and net-variance remain substantially unchanged (Fig. 5.8). This behavior is similar to that shown by bagged ensembles of gaussian SVMs (Fig. 5.5).



(a)



(b)

Figure 5.8: Bias-variance decomposition of error in bias, net variance, unbiased and biased variance in bagged polynomial SVMs, with respect to the number of iterations. (a) P2 data set (b)Letter-Two data set.

5.2.3.3 Comparison of bias–variance decomposition in single and bagged polynomial SVMs

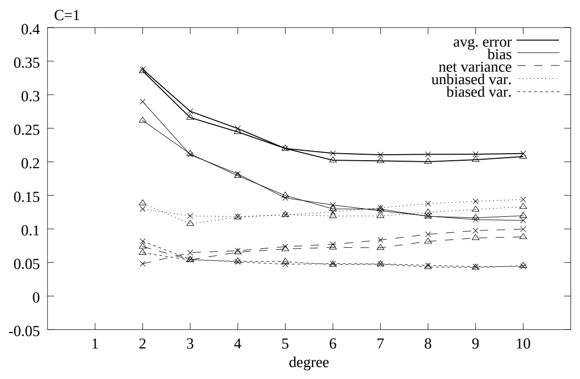
In bagged SVMs, the trend of the error with respect to the degree shows an "U" shape similar to that of single polynomial SVMs(Fig. 5.9). It depends both on bias and unbiased variance. Bias and biased variance are unchanged with respect to single SVMs, while net-variance is slightly reduced (for the reduction of the unbiased variance). As a result we have a slight reduction of the overall error.

5.2.4 Bagged dot-product SVM ensembles

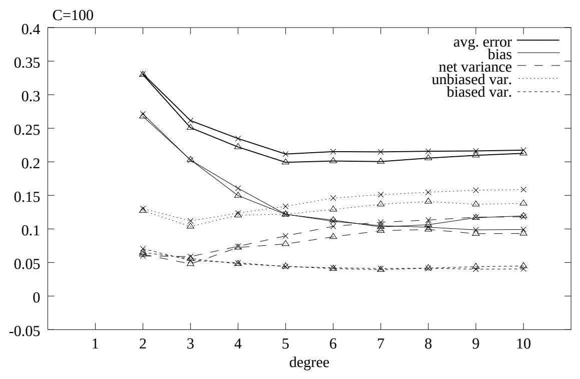
In this section are reported the results of bias–variance analysis in bagged SVMs, using base learners with dot-product kernels.

5.2.4.1 Bias–variance decomposition of the error

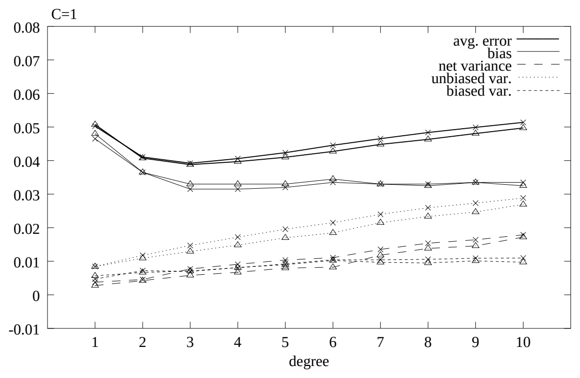
The decomposition of the error is represented with respect to different values of C. The error seems to be relatively independent of C (Fig. 5.10), and no changes are observed both for bias and variance components of the error. In some data sets the bias slightly decreases with C while unbiased variance slightly increases.



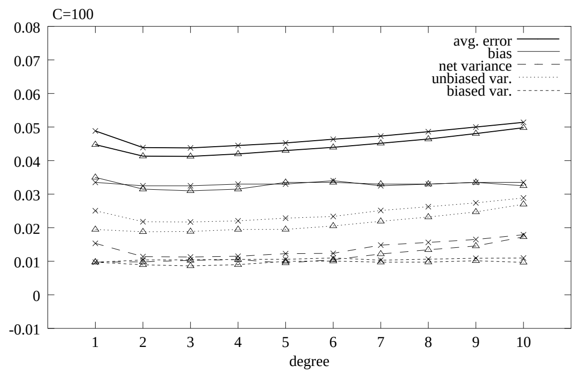
(a)



(b)



(c)



(d)

Figure 5.9: Comparison between bias-variance decomposition between single polynomial SVMs (lines labeled with crosses) and bagged polynomial SVM ensembles (lines labeled with triangles), while varying the degree and for some fixed values of C. P2 data set: (a) $C = 1$, (b) $C = 100$. Grey-Landsat data set: (c) $C = 1$, (d) $C = 100$.

5.2.4.2 Decomposition with respect to the number of base learners

Considering the bias–variance decomposition of the error with respect to the number of base learners, we can observe that the error reduction arises in the first 10-20 iterations, especially for the reduction of the unbiased variance. The bias and the biased variance remain substantially unchanged for all the iterations (Fig. 5.11)

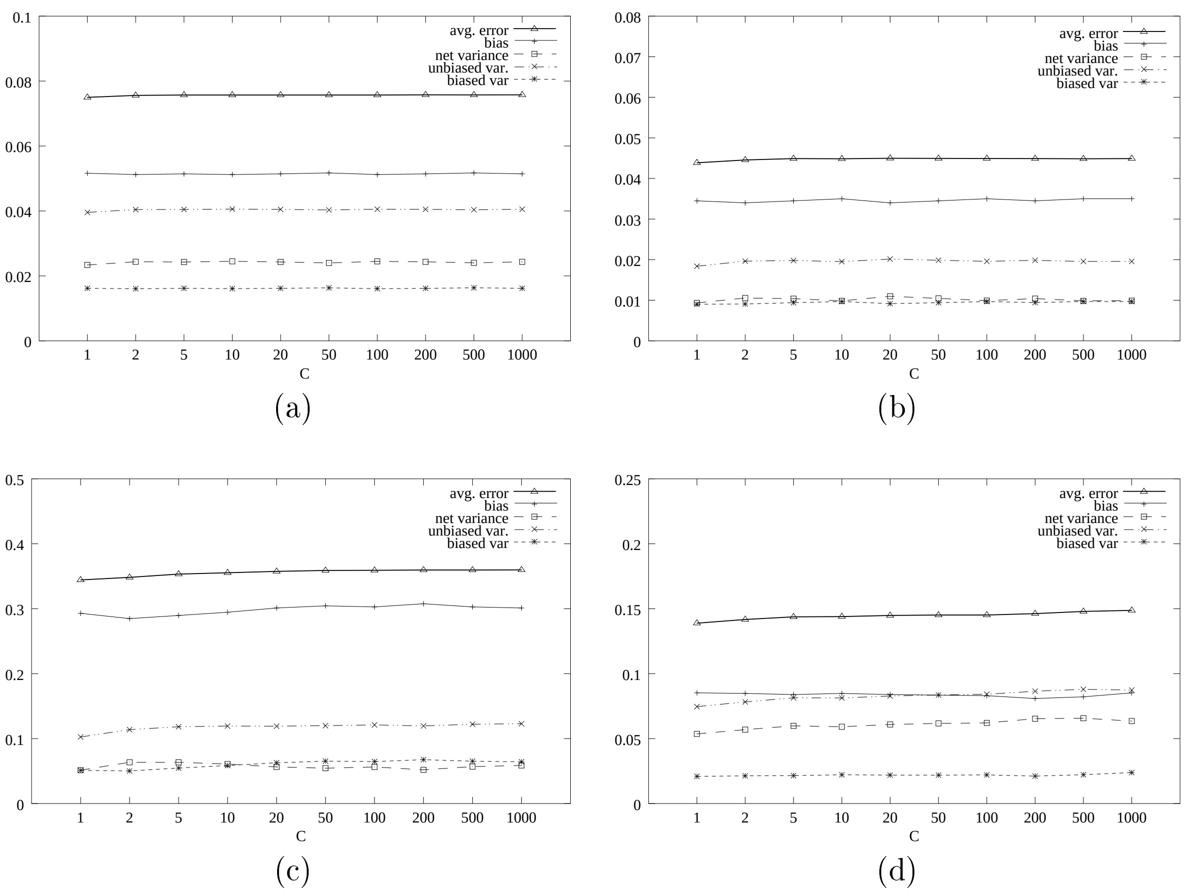
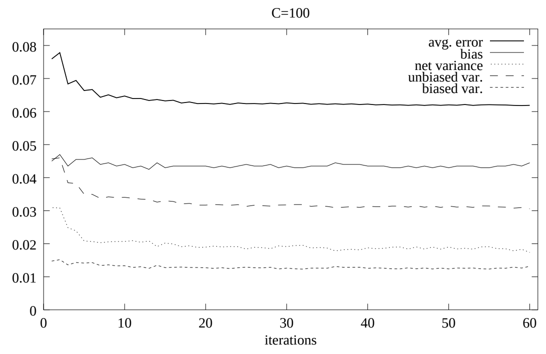


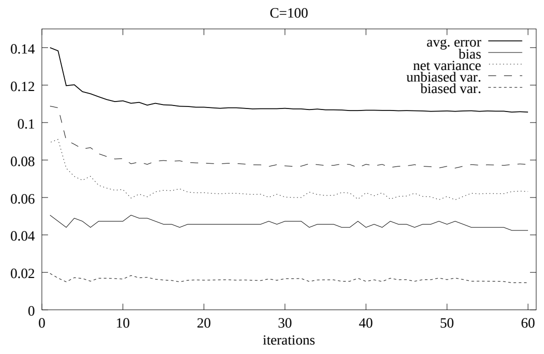
Figure 5.10: Bias-variance decomposition of error in bias, net variance, unbiased and biased variance in bagged dot-product SVM, while varying C. (a) Waveform data set (b) Grey-Landsat (c) Letter-Two with noise (d) Spam

5.2.4.3 Comparison of bias–variance decomposition in single and bagged dot-product SVMs

Fig. 5.12 shows the comparison between bias-variance decomposition between single dot-product SVMs. The reduction of the error in bagged ensembles is due to the reduction on the unbiased variance, while bias is unchanged or slightly increases in bagged dot-product SVMs. The biased variance also remains substantially unchanged. The shape of the error curve is quite independent of the C values, at least for $C \geq 1$.



(a)



(b)

Figure 5.11: Bias-variance decomposition of error in bias, net variance, unbiased and biased variance in bagged dot-product SVMs, with respect to the number of iterations. (a) Grey-Landsat data set (b) Letter-Two data set.

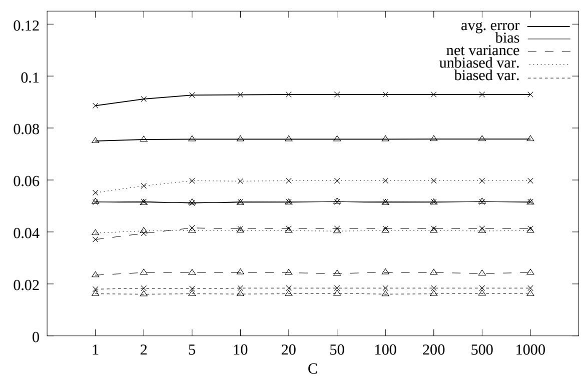
5.2.5 Bias–variance characteristics of bagged SVM ensembles

In Tab. 5.1 are summarized the compared results of bias–variance decomposition between single SVMs and bagged SVM ensembles. E_{SVM} stands for the estimated error of single SVMs, E_{bag} for the estimated error of bagged ensembles of SVMs, % *Error reduction* stands for the percent error reduction of the error between single and bagged ensembles, that is:

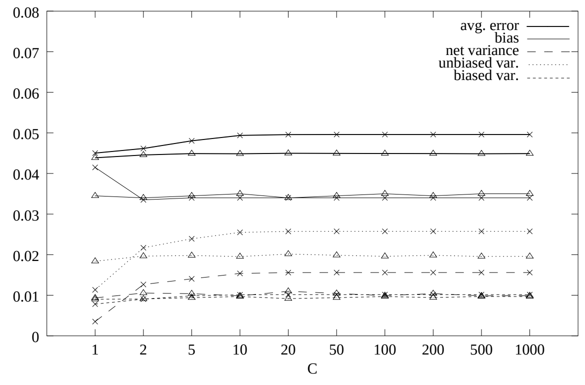
$$\%Error\ reduction = \frac{E_{SVM} - E_{bag}}{E_{SVM}}$$

% *Bias reduction*, % *NetVar reduction* and % *UnbVar reduction* corresponds respectively to the percent bias, net–variance and unbiased variance reduction between single and bagged ensemble of SVMs. The negative signs means that we have a larger error in the bagged ensemble. Note that sometimes the decrement of the net–variance can be larger than 100 %: the net–variance can be negative, when the biased variance is larger than the unbiased variance.

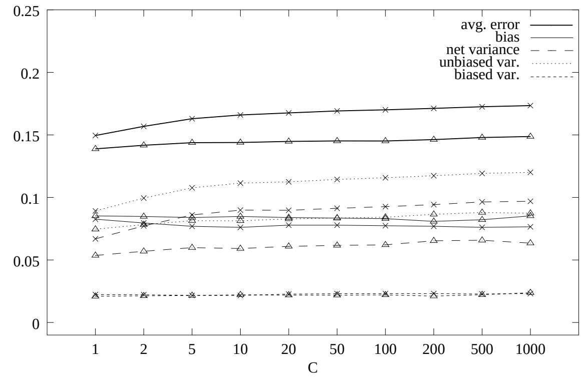
As expected, bagging does not reduce the bias (on the contrary, sometimes bias slightly increases). The net-variance is not eliminated but only partially reduced, and its decrement ranges from 0 to about 40 % with respect to single SVMs. Its reduction is due to the unbiased variance reduction, while biased variance is unchanged. As a result the error decreases, but its decrement it is not so noticeable, as it ranges from 0 to about 15 % with respect to single SVMs, depending on the kernel and the data set. The overall shape of



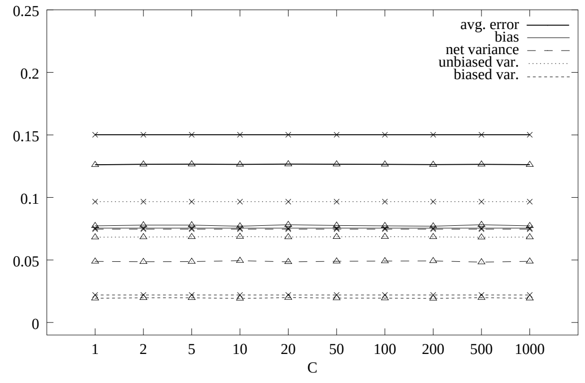
(a)



(b)



(c)



(d)

Figure 5.12: Comparison between bias-variance decomposition between single dot-product SVMs (lines labeled with crosses) and bagged dot-product SVM ensembles (lines labeled with triangles), while varying the values of C. (a) Waveform (b) Grey-Landsat (c) Spam (d) Musk.

the curves of the error, bias and variance are very close to that of single SVMs.

5.3 Bias–variance analysis in random aggregated ensembles of SVMs

This section investigates the effect of random aggregation of SVMs on bias and variance components of the error.

Table 5.1: Comparison of the results between single and bagged SVMs.

	$\mathbf{E_{SVM}}$	$\mathbf{E_{bag}}$	% **Error reduction**	% **Bias reduction**	% **NetVar reduction**	% **UnbVar reduction**
Data set *P2*						
RBF-SVM	0.1517	0.1500	1.14	-2.64	3.18	2.19
Poly-SVM	0.2088	0.1985	4.95	4.85	5.08	5.91
D-prod SVM	0.4715	0.4590	2.65	1.11	34.09	15.28
Data set *Waveform*						
RBF-SVM	0.0707	0.0662	6.30	-1.41	26.03	17.82
Poly-SVM	0.0761	0.0699	8.11	0.36	23.78	17.94
D-prod SVM	0.0886	0.0750	15.37	-0.22	37.00	28.20
Data set *Grey-Landsat*						
RBF-SVM	0.0384	0.0378	1.74	2.94	-7.46	3.94
Poly-SVM	0.0392	0.0388	1.05	-4.76	24.80	12.06
D-prod SVM	0.0450	0.0439	2.58	16.87	-165.72	-62.21
Data set *Letter-Two*						
RBF-SVM	0.0745	0.0736	1.20	-25.00	21.63	12.29
Poly-SVM	0.0745	0.0733	1.55	-15.79	13.92	10.41
D-prod SVM	0.0955	0.0878	8.09	2.22	27.55	23.06
Data set *Letter-Two with added noise*						
RBF-SVM	0.3362	0.3345	0.49	1.75	-5.78	0.40
Poly-SVM	0.3432	0.3429	0.09	-0.58	3.06	0.91
D-prod SVM	0.3486	0.3444	1.21	-0.56	10.23	6.09
Data set *Spam*						
RBF-SVM	0.1292	0.1290	0.14	-0.48	1.57	2.22
Poly-SVM	0.1323	0.1318	0.35	2.11	-5.83	-1.19
D-prod SVM	0.1495	0.1389	7.15	-3.16	19.87	16.38
Data set *Musk*						
RBF-SVM	0.0898	0.0920	-2.36	-6.72	22.91	13.67
Poly-SVM	0.1225	0.1128	7.92	-10.49	38.17	37.26
D-prod SVM	0.1501	0.1261	15.97	-2.41	34.56	29.38

Our aim consists in getting insights into the way random aggregated ensembles learn, in order to characterize learning in terms of the bias–variance components of the error.

In particular, an extended experimental bias–variance decomposition of the error in random aggregated SVM ensembles is performed in order to verify the theoretical property of canceled variance in random aggregation.

The plan of the experiments replicates the previous one we followed for bagged SVM ensembles, using dot-product, polynomial and gaussian kernels.

We evaluated for each kernel the expected error and its decomposition in bias, net-variance, unbiased and biased variance with respect to the learning parameters of the base learners. Then we analyzed the bias–variance decomposition as a function of the number of the base learners employed. Finally we compared bias and variance with respect to the learning parameters in random aggregated SVM ensembles and in the corresponding single SVMs, in order to study the effect of random aggregation on the bias and variance components of the error.

Here are reported only some examples and a summary of the results of bias–variance analysis in random aggregated SVM ensembles. The experiments we performed with random aggregated ensembles of SVMs are detailed in [181].

5.3.1 Experimental setup

In order to estimate the decomposition of the error in bias, unbiased and biased variance with random aggregated ensembles of SVMs, we used a bootstrap approximation of the unknown distribution P, that is, we drew samples of relatively small size from a relatively large training set, according to an uniform probability distribution. From this standpoint we approximated random aggregation by a sort of undersampled bagging, drawing data from the universe population U represented by a comfortable large training set. The bias-variance decomposition of the error is computed with respect to a separate test set significantly larger than the undersampled training sets.

To estimate the decomposition of the error in bias, net-variance, unbiased and biased variance with random aggregated ensembles of SVMs, we performed a bias-variance decomposition of the error on the data sets described in Chap. 4.

We split the data in a separated learning set $\mathcal{D}$ and testing set $\mathcal{T}$. Then we drew with replacement from $\mathcal{D}$ n set of samples $\bar{D}_i$, according to a uniform probability distribution. Each set of samples $\bar{D}_i$ is composed by m samples D_{ij} drawn with replacement from $\mathcal{D}$, using an uniform probability distribution. Each sample D_{ij} is composed by s samples. The D_{ij} samples are in turn collected in n sets $\bar{D}_i = \{D_{ij}\}_{j=1}^{m}$.

We used the n sets $\bar{D}_i$ to train n random aggregated ensembles, repeating this process for all the considered SVM models. In order to properly compare the effect of different choices of the learning parameters on bias–variance decomposition of the error, each SVM model is represented by a different choice of the kernel type and parameters and it is trained with the same sets $\bar{D}_i$, $1 \leq i \leq n$ of samples.

Fig. 5.13 summarizes the experimental procedure we adopted to generate the data and Fig. 5.14 the experimental procedure to evaluate bias–variance decomposition of the error.

```
Procedure Generate_samples
Input arguments:
    - Data set $\mathcal{S}$
    - Number $n$ of set of samples
    - Size of samples $s$
    - Number $m$ of samples collected in each set
Output:
    - Sets $\bar{D}_i = \{D_{ij}\}_{j=1}^{m},\ 1 \leq i \leq n$ of samples
begin procedure
    $[\mathcal{D}, \mathcal{T}] = \texttt{Split}(\mathcal{S})$
    for $i = 1$ to $n$
    begin
        $\bar{D}_i = \emptyset$
        for $j = 1$ to $m$
        begin
            $D_{ij} = \texttt{Draw\_with\_replacement}(\mathcal{D}, s)$
            $\bar{D}_i = \bar{D}_i + D_{ij}$
        end
    end
end procedure.
```

Figure 5.13: Procedure to generate samples to be used for bias–variance analysis in random aggregation

Sets $\bar{D}_i$ of samples are drawn with replacement according to an uniform probability distribution from the training set $\mathcal{D}$ by the procedure `Draw_with_replacement`. This process is repeated n times (procedure `Generate_samples`, Fig. 5.13). Then the procedure `Bias-Variance_analysis` (Fig. 5.14) trains different ensembles of random aggregated SVMs (procedure `Ensemble_train`) using the sets of samples generated by the procedure `Generate_samples`. The bias–variance decomposition of the error is performed on the separated test set $\mathcal{T}$ using the previously trained bagged SVM ensembles (procedure `Perform_BV_analysis`).

We employed gaussian, polynomial and dot-product kernels evaluating 110 different SVM models, considering different combinations of the type of the kernel and learning parameters for each data set. For each model we set $s = 100,\ n = 100,\ m = 60$, training for each data set $110 \times 100 = 11000$ random aggregated ensembles and a total of $110 \times 100 \times 60 = 660000$

```
Procedure Bias–Variance_analysis
Input arguments:
    - Test set $\mathcal{T}$
    - Number of random aggregated ensembles $n$
    - Number of bootstrap replicate $m$
    - Set of learning parameters $\mathcal{A}$
Output:
    - Error, bias, net-variance, unbiased and biased variance $BV = \{bv(\alpha)\}_{\alpha \in \mathcal{A}}$
    of the random aggregated ensemble having base learners with learning parameters $\alpha \in \mathcal{A}$.
begin procedure
    For each $\alpha \in \mathcal{A}$
    begin
        Ensemble_Set($\alpha$) = $\emptyset$
        for $i = 1$ to $n$
        begin
            rand_aggr($\alpha$, $\bar{D}_i$) = Ensemble_train ($\alpha$, $\bar{D}_i$)
            Ensemble_Set($\alpha$) = Ensemble_Set($\alpha$) $\cup$ rand_aggr($\alpha$, $\bar{D}_i$)
        end
        $bv(\alpha)$ = Perform_BV_analysis(Ensemble_Set ($\alpha$), $\mathcal{T}$)
        $BV = BV \cup bv(\alpha)$
    end
end procedure.
```

Figure 5.14: Procedure to perform bias–variance analysis on random aggregated SVM ensembles

different SVMs. Considering all the data sets we trained and tested about 80000 different random aggregated SVM ensembles and a total of about 5 millions of single SVMs.

5.3.2 Random aggregated RBF-SVM ensembles

In this section are reported the results of bias–variance analysis in random aggregated RBF-SVMs.

5.3.2.1 Bias–variance decomposition of the error

The decomposition of the error is represented with respect to different values of σ and for fixed values of C.

Schematically we can observe the following facts:

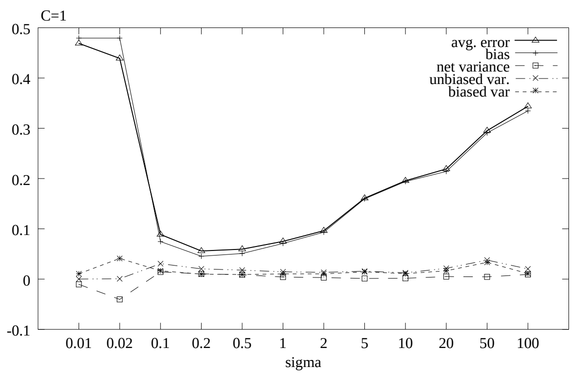
- The shape of the error is mostly determined by the bias and it is in general sigmoid with respect to σ, but sometimes an "U" shape is observed, as the error can increase with σ (Fig. 5.15).
- In all the data sets the net-variance is about 0 for all the values of σ, and the wave-shape of the net-variance is very reduced or completely absent.
- The net-variance is 0 as both biased and unbiased variance are very low, with values close to 0. Only in the Waveform data set unbiased and biased variance are both quite large in the "high bias" region (and partially also in Letter-Two).
- The net-variance is always 0 in the "stabilized region" in all the considered data sets.
- The error is determined almost entirely by the bias: in Fig. 5.15 it is difficult to the distinguish the error and bias curves.

5.3.2.2 Decomposition with respect to the number of base learners

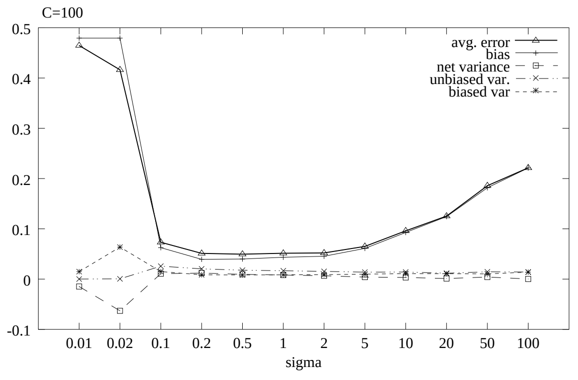
Fig. 5.16 a, b and d refer to bias–variance decomposition of the error with respect to the number of base learners for random aggregated SVMs of the "stabilized region". In these cases we can observe the following facts:

- Most of the decrement of the error occurs within the first iterations (from 10 to 30, depending on the data set).
- The bias remains unchanged during all the iterations
- The decrement of the error is almost entirely due to the decrement of the unbiased variance, and it is larger than in bagged ensembles of SVMs.

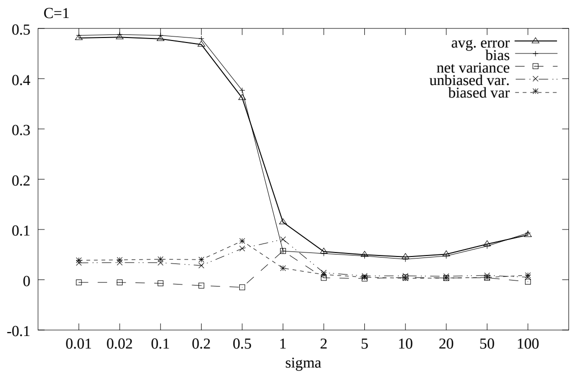
On the contrary Fig. 5.16 c refers to σ values in the the "transition region". Also in this case the bias remains unchanged in average (higher than the bias of SVMs of the "stabilized region"), but oscillates largely, especially during the first 20 iterations. The unbiased variance also oscillates, but tends to decrement with the iterations, lowering the error. The biased variance oscillates in the same way (that is with the same phase) with respect to the bias, but with a lower amplitude, while the unbiased variance and in particular the net-variance oscillates in a specular way (opposite phase) with respect to the bias. This is observed also in the other data sets (except in Letter-Two with noise). I have no explanations for this behaviour.



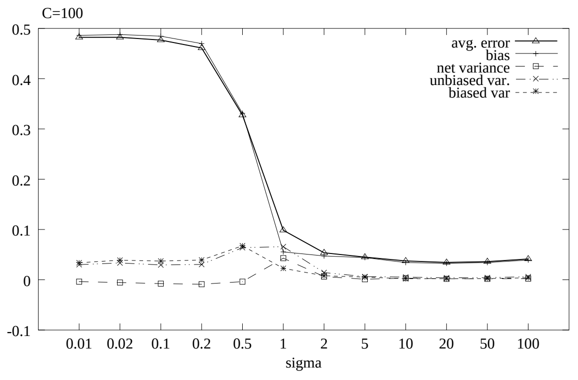
(a)



(b)



(c)



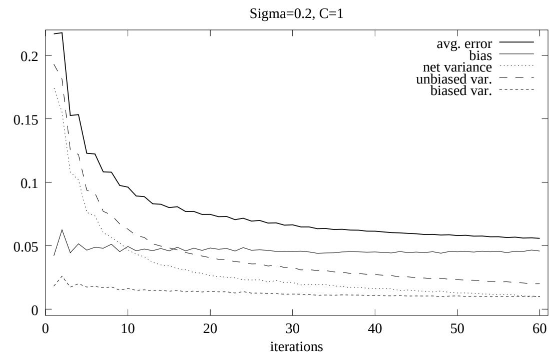
(d)

Figure 5.15: Bias-variance decomposition of error in bias, net variance, unbiased and biased variance in random aggregated gaussian SVMs, while varying σ and for some fixed values of C. P2 data set: (a) $C = 1$, (b) $C = 100$. Letter-Two data set: (c) $C = 1$, (d) $C = 100$

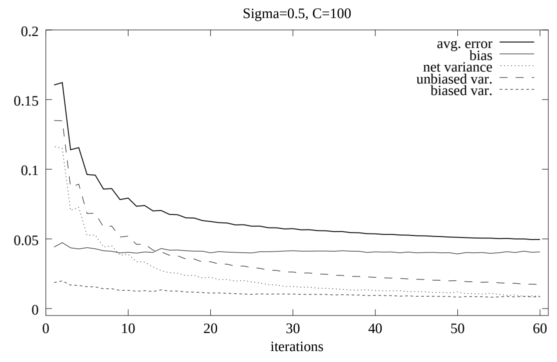
5.3.2.3 Comparison of bias–variance decomposition in single and random aggregated RBF-SVMs

In all the graphics of this section the data referred to single SVMs are labeled with crosses, while random aggregated SVMs are labeled with triangles. The corresponding quantities (e.g. bias, net-variance, etc.) are represented with the same type of line both in single and random aggregated SVMs.

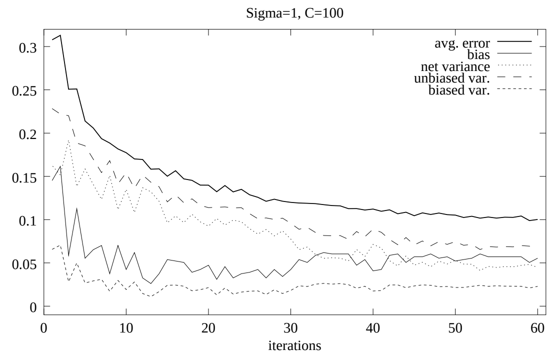
In random aggregated ensembles net-variance is very close to 0. As a consequence, the error is in practice reduced to the bias. As in single SVMs, we can distinguish three main regions with respect to σ:



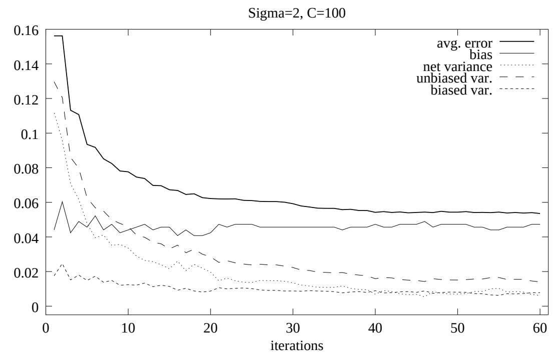
(a)



(b)



(c)



(d)

Figure 5.16: Bias-variance decomposition of error in bias, net variance, unbiased and biased variance in random aggregated SVM RBF, with respect to the number of iterations. P2 dataset: (a) $C = 1$, $\sigma = 0.2$, (b) $C = 100$, $\sigma = 0.5$. Letter-Two data set: (c) $C = 100$, $\sigma = 1$, (d) $C = 100$, $\sigma = 2$

High bias region. In this region the the error of single and random aggregated SVMs is about equal, and it is characterized by a very high bias. The net-variance is close to 0, because biased variance is about equal to the unbiased variance. In most cases they are both close to 0 (Fig. 5.17 a and b). In some cases they are equal but greater than 0 with significantly larger values in single that in random aggragated SVMs (Fig. 5.17 c and d).

Transition region. The bias decreases in the transition region at about the same rate in single and random aggregated SVM ensembles. The net-variance maintains the wave-shape also in random aggregated SVMs, but it is lower. In some data sets (Fig. 5.17 a and b), the net-variance remains low with no significant variations also for small values of σ. For these reasons the error decreases more quickly in random aggregated SVMs, and the error of the ensemble is about equal to the bias.

Stabilized region. The net-variance stabilizes, but at lower values (very close to 0) compared with net-variance of single SVMs. Hence we have a reduction of the error for random aggregated SVM ensembles in this region. Note that the reduction of the error depends heavily on the level of the unbiased variance of dingle SVMs in the stabilized region. If it is sufficiently high, we can achieve substantial reduction of the error in random aggregated SVM ensembles. With some data sets the error increases for large values of σ, mainly for the increment of the bias (Fig. 5.15 a and b).

5.3.3 Random aggregated polynomial SVM ensembles

5.3.3.1 Bias–variance decomposition of the error

The decomposition of the error is represented with respect to different values of the polynomial degree and for fixed values of C.

Schematically we can observe the following facts:

- In all the data sets the net-variance is about 0 for all the values of polynomial degree, as both biased and unbiased variance are very low close to 0. Only in some data sets (e.g. P2), with low values of C we can observe a certain level on unbiased variance, especially with low degree polynomials (Fig. 5.18 a).
- In almost all the considered data sets the error shows an "U" shape with respect to the degree. This shape tends to a flat line if C is relatively large. With the P2 data set the error decreases with the degree (Fig. 5.18).
- The error is determined almost entirely by the bias: its minimum is reached for specific values of the degree of the polynomial and depends on the characteristics of the data set.

5.3.3.2 Decomposition with respect to the number of base learners

This section reports data and graphs about the decomposition of bias–variance in random aggregated SVMs with respect to the number of iterations, that is the number of base learners used.

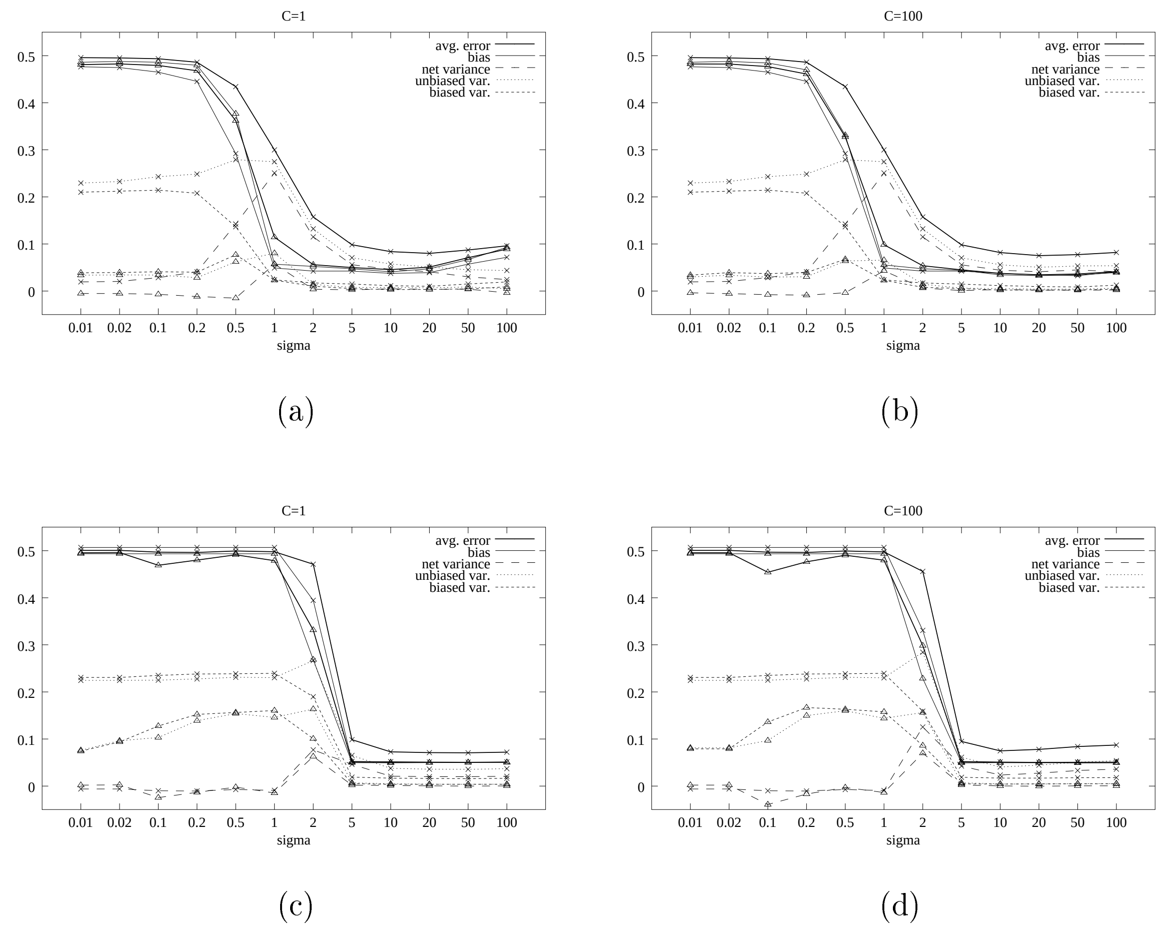
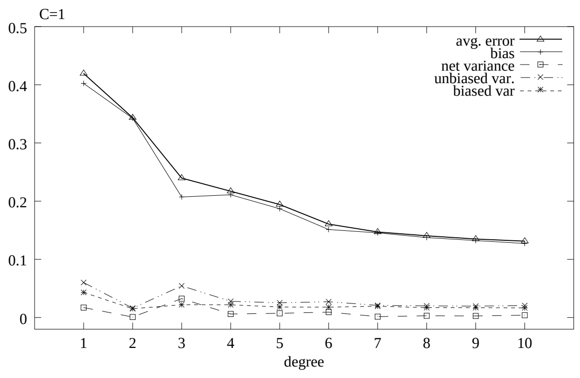
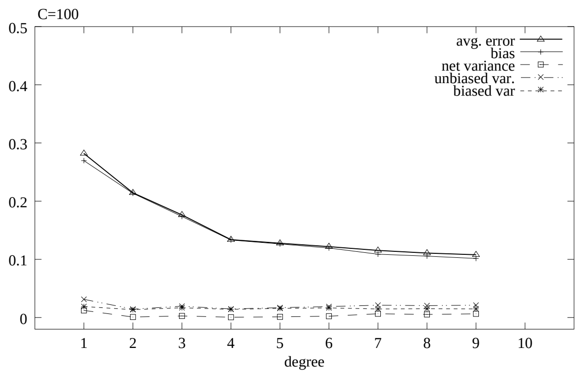


Figure 5.17: Comparison of bias-variance decomposition between single RBF-SVMs (lines labeled with crosses) and random aggregated ensembles of RBF-SVMs (lines labeled with triangles), while varying σ and for some fixed values of C. Letter-Two data set: (a) $C = 1$, (b) $C = 100$. Waveform data set: (c) $C = 1$, (d) $C = 100$.

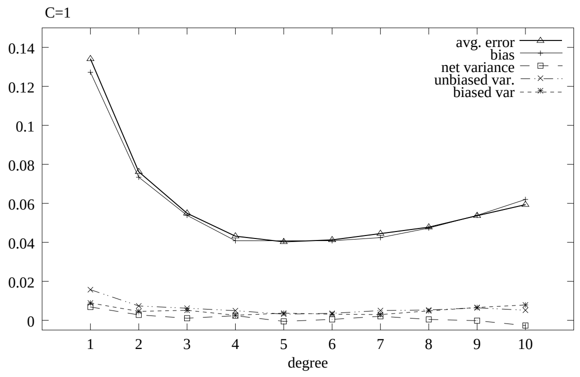
Fig. 5.19 shows that the bias remains constant throughout the iterations. Most of the error decrement is achieved within the first 10-20 iterations, and it is almost entirely due to the decrement of the unbiased variance. The error is reduced to the bias, when the number of iterations is sufficiently large. The biased variance is low and slowly decreases at each iteration, while the unbiased variance continues to decrease at each iteration, but most of its decrement occurs within the first 20 iterations (Fig. 5.19).



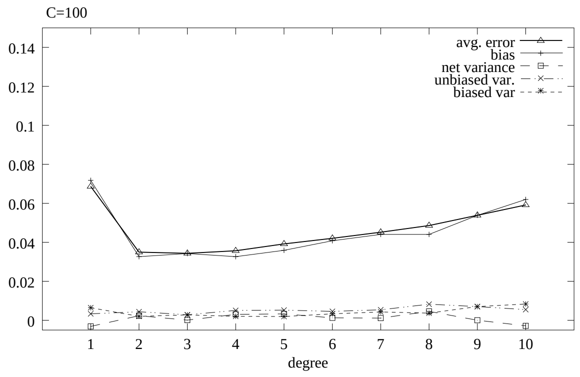
(a)



(b)



(c)



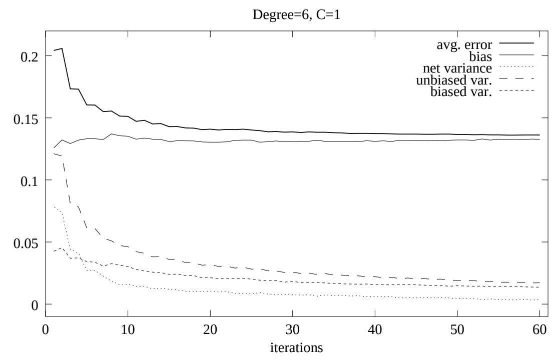
(d)

Figure 5.18: Bias-variance decomposition of error in bias, net variance, unbiased and biased variance in random aggregated polynomial SVM, while varying the degree and for some fixed values of C. P2 data set: (a) $C = 1$, (b) $C = 100$. Letter-Two data set: (c) $C = 1$, (d) $C = 100$

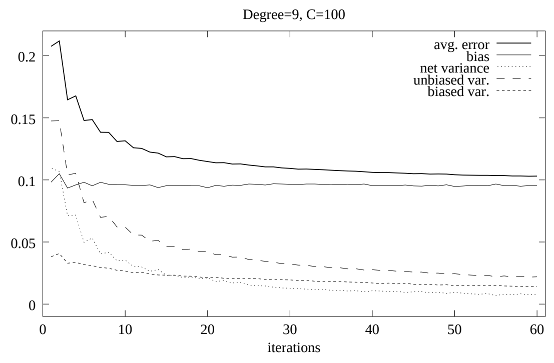
5.3.3.3 Comparison of bias–variance decomposition in single and random aggregated polynomial SVMs

In random aggregated polynomial SVMs the error is due almost entirely to the bias. The bias component is about equal in random aggregated and single SVMs.

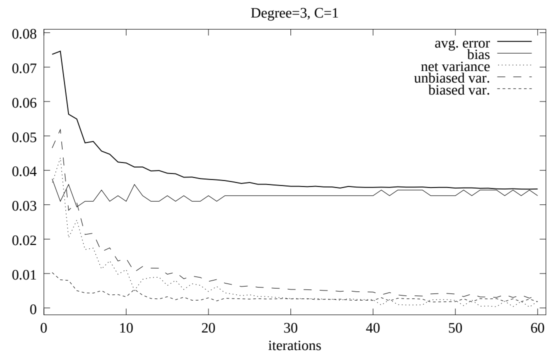
In single SVMs sometimes are observed opposite trends between bias and unbiased variance: the bias decreases, while the unbiased variance increases with the degree (Fig. 5.20 a and b). On the contrary in random aggregated ensembles the net-variance is very close to 0 and the error is due almost entirely to the bias (Fig. 5.20).



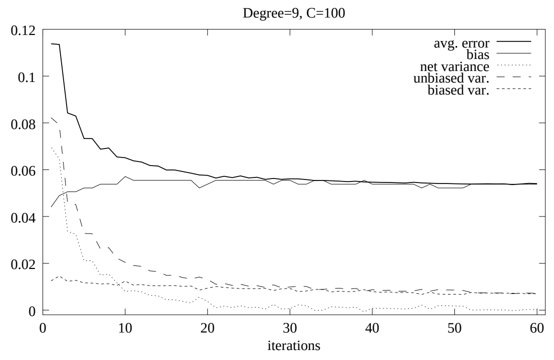
(a)



(b)



(c)

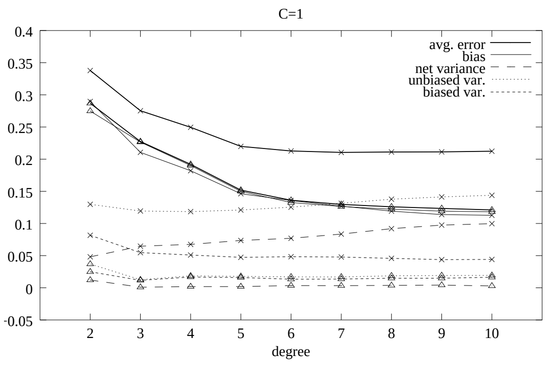


(d)

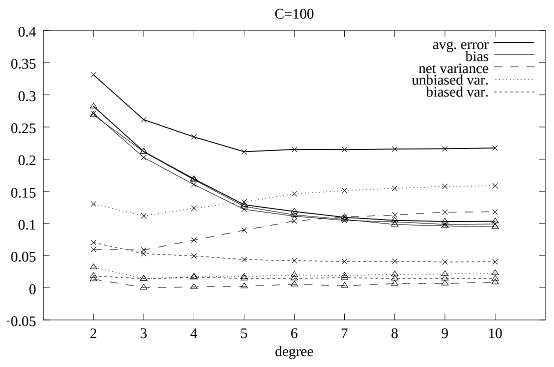
Figure 5.19: Bias-variance decomposition of error in bias, net variance, unbiased and biased variance in random aggregated polynomial SVMs, with respect to the number of iterations. P2 dataset: (a) $C = 1$, degree = 6 (b) $C = 100$, degree = 9. Letter-Two data set: (c) $C = 1$, degree = 3, (d) $C = 100$, degree = 9.

Hence in random aggregated SVMs, the shape of the error with respect to the degree depends on the shape of the bias, and consequently the error curve shape is bias-dependent, while in single SVMs it is variance or bias-variance dependent.

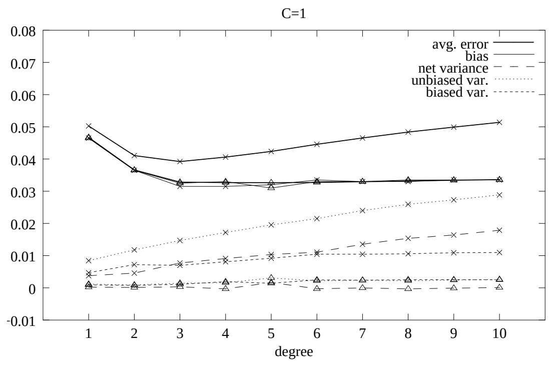
The general shape of the error with respect to the degree resembles an "U" curve, or can be flatted in dependence of the bias trend, especially with relatively large C values.



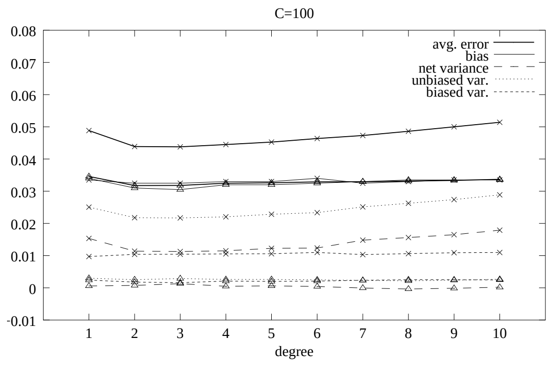
(a)



(b)



(c)



(d)

Figure 5.20: Comparison of bias-variance decomposition between single polynomial SVMs (lines labeled with crosses) and random aggregated polynomial SVM ensembles (lines labeled with triangles), while varying the degree and for some fixed values of C. P2 data set: (a) $C = 1$, (b) $C = 100$. Grey-Landsat data set: (c) $C = 1$, (d) $C = 100$.

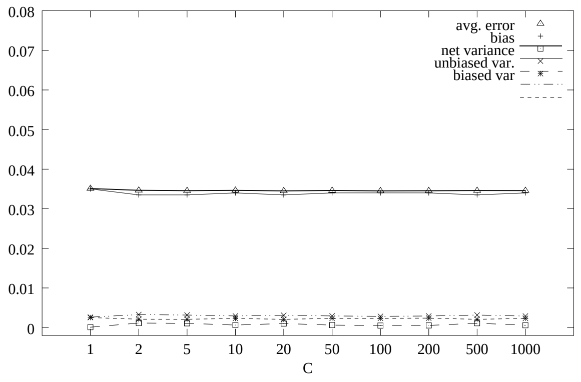
5.3.4 Random aggregated dot-product SVM ensembles

5.3.4.1 Bias–variance decomposition of the error

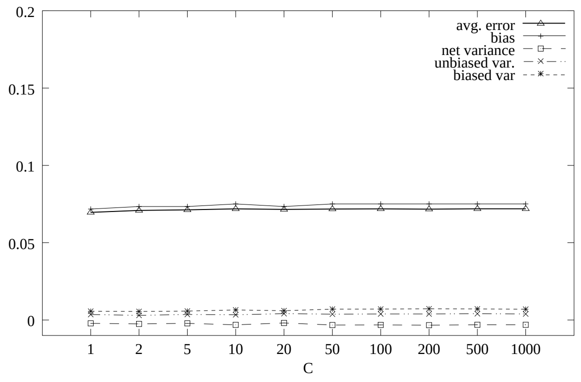
The decomposition of the error is represented with respect to different values of C.

Schematically we can observe the following facts (Fig. 5.21):

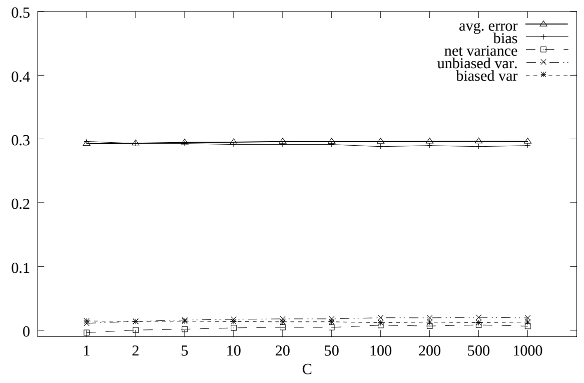
- Net-variance is about 0 for all the values of C, as both biased and unbiased variance are very low close to 0.
- The error, bias and variance seem to be independent of the values of C. Anyway, note that in the experiments we used only values of $C \geq 1$.
- The error is determined almost totally by the bias.



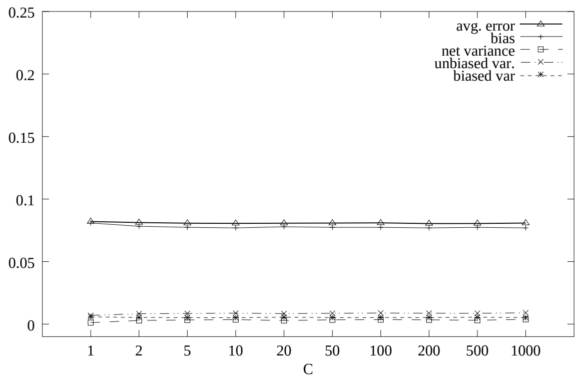
(a)



(b)



(c)



(d)

Figure 5.21: Bias-variance decomposition of error in bias, net variance, unbiased and biased variance in random aggregated dot-product SVM, while varying C. (a) Grey-Landsat data set (b) Letter-Two (c) Letter-Two with noise (d) Spam

5.3.4.2 Decomposition with respect to the number of base learners

Considering the number of iterations, the most important facts with dot-product random aggregated SVM ensembles are the following (Fig. 5.22):

- The bias remains constant
- Most of the error decrement is achieved within the first 10-20 iterations
- Error decrement is due to the decrement of the unbiased variance
- The error is determined almost totally by the bias.
- The biased variance slowly decreases at each iteration

5.3.4.3 Comparison of bias–variance decomposition in single and random aggregated dot-product SVMs

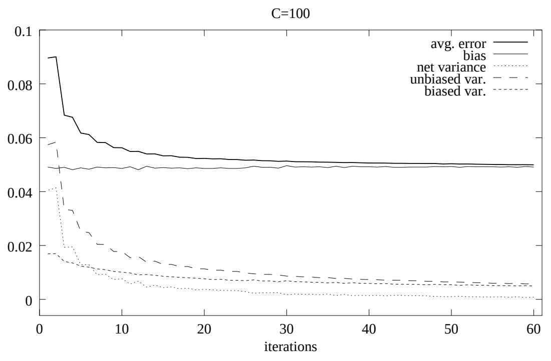
In all cases the error is about equal to the bias, that remains unchanged with respect to the single SVMs. As a consequence the error shape is equal to the shape of the bias and it is independent of the C values, at least for $C \geq 1$. As a result we a have a significant reduction of the error due to decrement of the unbiased variance (Fig. 5.23).

5.3.5 Bias–variance characteristics of random aggregated SVM ensembles

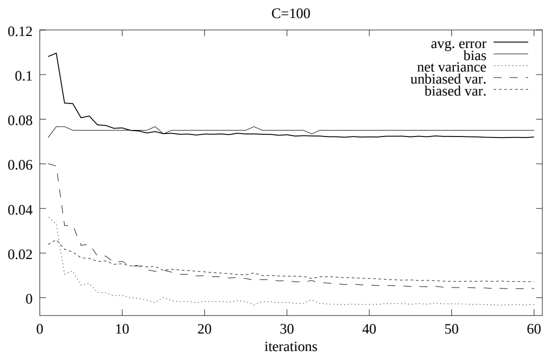
In the following tables are summarized the compared results of bias–variance decomposition between single SVMs and random aggregated SVM ensembles. E_{SVM} stands for the estimated error of single SVMs, E_{agg} for the estimated error of random aggregated ensembles of SVMs, $\%$ *Error reduction* stands for the percent error reduction of the error between single and random aggregated ensembles, that is:

$$\%Error\ reduction = \frac{E_{SVM} - E_{agg}}{E_{SVM}}$$

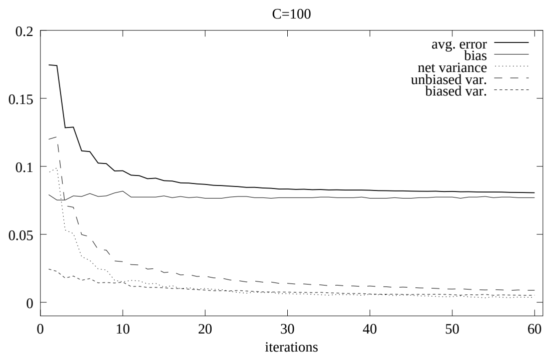
$\%$ *Bias reduction*, $\%$ *NetVar reduction* and $\%$ *UnbVar reduction* corresponds respectively to the percent bias, net–variance and unbiased variance reduction between single and random aggregated ensemble of SVMs. The negative signs means that we have a larger error in the random aggregated ensemble. Note that sometimes the decrement of



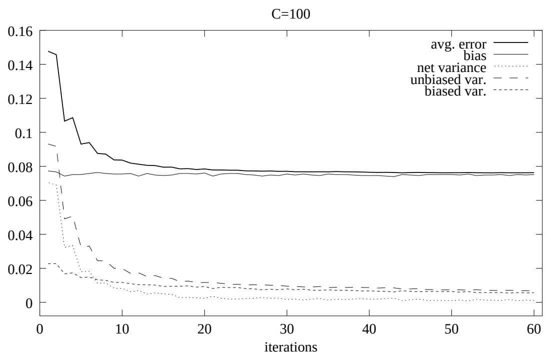
(a)



(b)



(c)



(d)

Figure 5.22: Bias-variance decomposition of error in bias, net variance, unbiased and biased variance in random aggregated dot-product SVM, with respect to the number of iterations. (a) Waveform (b) Letter-Two (c) Spam (d) Musk.

the net–variance can be larger than 100 %: recall that net–variance can be negative (when the biased variance is larger than the unbiased variance).

Random aggregated ensembles of SVMs strongly reduce net-variance. Indeed in all data sets the net-variance is near to 0, with a reduction close to 100 % with respect to single SVMs, confirming the ideal behavior of random aggregating (Sect. 5.1). Unbiased variance reduction is responsible for this fact, as in all data sets its decrement amounts to about 90 % with respect to single SVMs (Tab. 5.2). As expected bias remains substantially unchanged, but with the *P2* data set with polynomial and gaussian kernels we register a

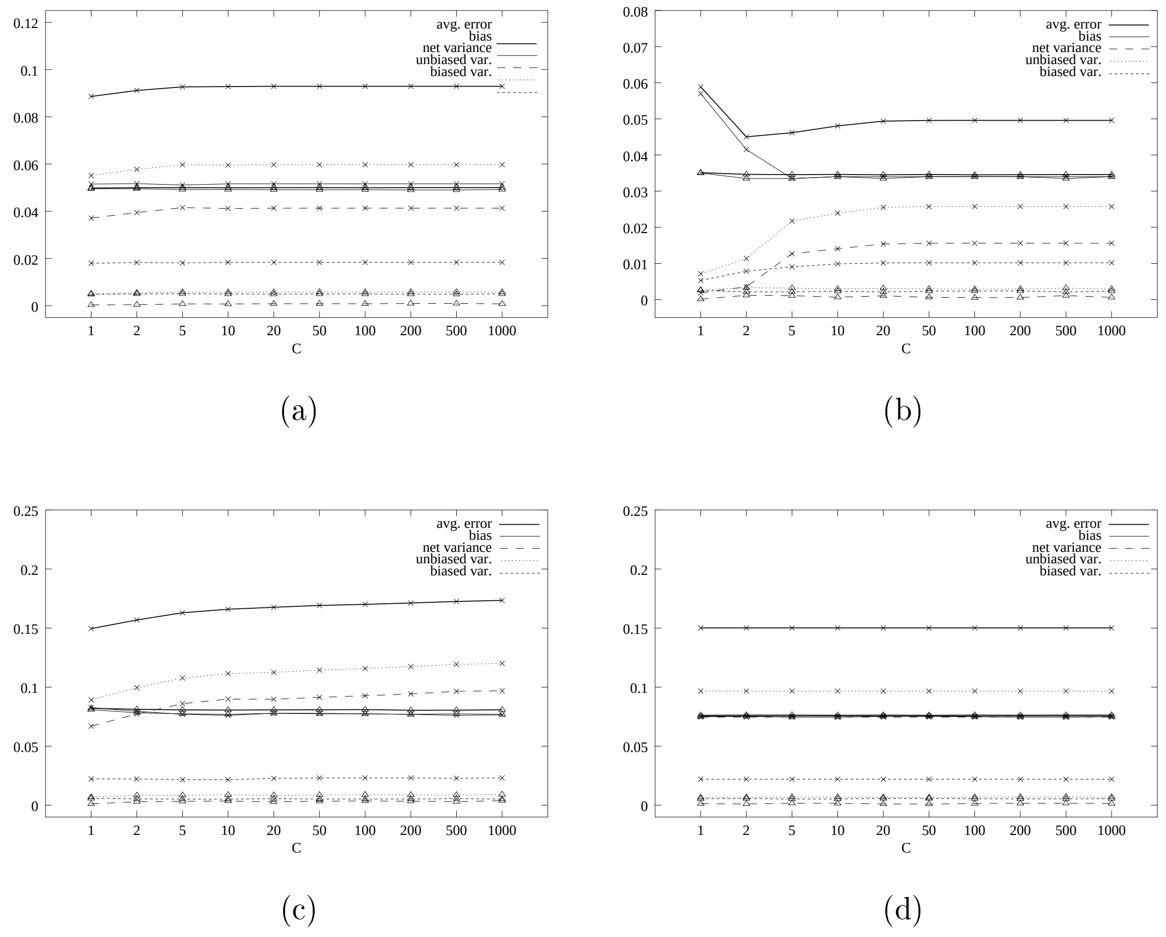


Figure 5.23: Comparison of bias-variance decomposition between single dot-product SVMs (lines labeled with crosses) and random aggregated dot-product SVM ensembles (lines labeled with triangles), while varying the values of C. (a) Waveform (b) Grey-Landsat (c) Spam (d) Musk.

not negligible decrement of the bias. As a result the error decreases from 15 to about 70 % with respect to single SVMs, depending on the kernel and on the characteristics of the data set. The overall shape of the curves of the error resembles that of the bias of single SVMs, with a characteristics sigmoid shape for gaussian kernels (that can also become an "U" shape for certain data sets) with respect to the σ width values (Fig. 5.15 and 5.16), an "U" shape with respect to the degree for polynomial kernels (Fig. 5.18 and 5.19), while it is relatively independent of the C values (at least for sufficiently large values of C) for random aggregated linear SVMs (Fig. 5.21).

Table 5.2: Comparison of the results between single and random aggregated SVMs.

	$\mathbf{E_{SVM}}$	$\mathbf{E_{agg}}$	% **Error reduction**	% **Bias reduction**	% **NetVar reduction**	% **UnbVar reduction**
Data set *P2*						
RBF-SVM	0.1517	0.0495	67.37	24.52	99.04	85.26
Poly-SVM	0.2088	0.1030	50.65	19.56	92.26	83.93
D-prod SVM	0.4715	0.4611	2.21	0.89	142.65	91.08
Data set *Waveform*						
RBF-SVM	0.0707	0.0501	29.08	1.14	100.58	89.63
Poly-SVM	0.0761	0.0497	34.59	3.68	97.12	89.44
D-prod SVM	0.0886	0.0498	43.74	3.84	99.12	90.69
Data set *Grey-Landsat*						
RBF-SVM	0.0384	0.0300	21.87	3.22	99.95	85.42
Poly-SVM	0.0392	0.0317	19.13	3.17	83.79	80.95
D-prod SVM	0.0450	0.0345	23.33	19.27	69.88	72.57
Data set *Letter-Two*						
RBF-SVM	0.0745	0.0345	53.69	0.00	95.32	92.48
Poly-SVM	0.0745	0.0346	53.54	-5.26	95.46	92.71
D-prod SVM	0.0955	0.0696	27.11	2.22	109.73	92.31
Data set *Letter-Two with added noise*						
RBF-SVM	0.3362	0.2770	17.55	2.92	90.26	87.04
Poly-SVM	0.3432	0.2775	19.13	1.75	95.96	89.42
D-prod SVM	0.3486	0.2925	16.07	-1.68	106.4	89.97
Data set *Spam*						
RBF-SVM	0.1292	0.0844	34.67	6.75	99.74	90.05
Poly-SVM	0.1323	0.0814	38.47	22.33	95.22	86.03
D-prod SVM	0.1495	0.0804	46.22	6.90	94.91	90.24
Data set *Musk*						
RBF-SVM	0.0898	0.0754	16.02	0.39	106.70	93.85
Poly-SVM	0.1225	0.0758	38.12	1.53	97.52	94.02
D-prod SVM	0.1501	0.0761	49.28	0.80	98.30	93.03

5.4 Undersampled bagging

While bagging had been successfully applied to different classification and regression problems [8, 44, 5, 102, 186], random aggregating is almost ideal, because in most cases the true distribution P is unknown and we can access only a limited and often small sized data set.

From a theoretical standpoint we need to know the usually unknown true distribution of the data, and we should be able to access the (possibly infinite) universe U of the data

From a different standpoint random aggregating (using a bootstrap approximation of P) can be viewed as a form of undersampled bagging if we consider the universe U as a data set from which undersampled data, that is data sets whose cardinality is much less than the cardinality of U, are randomly drawn with replacement. For instance this is the way by which we approximated random aggregating in our experiments described in Sect. 5.3.

In real problems, if we have very large learning sets, or on-line available learning data, we could use undersampled bagging in order to overcome the space complexity problem raising from learning too large data sets, or to allow on-line learning [34].

Indeed in most data mining problems we have very large data sets, and ordinary learning algorithms cannot directly process the data set as a whole. For instance most of the implementations of the SVM learning algorithm have a $\mathcal{O}(n^2)$ space complexity, where n is the number of examples. If n is relatively large (e.g. $n = 100000$) we need room for 10^{10} elements, a too costly memory requirement for most current computers. In these cases we could use relatively small data sets randomly drawn form the large available data set, using undersampled bagging methods to improve performances.

This situation is very similar to the ideal random aggregation setting: the only difference is that we use only limited data and an uniform probability distribution to draw the data.

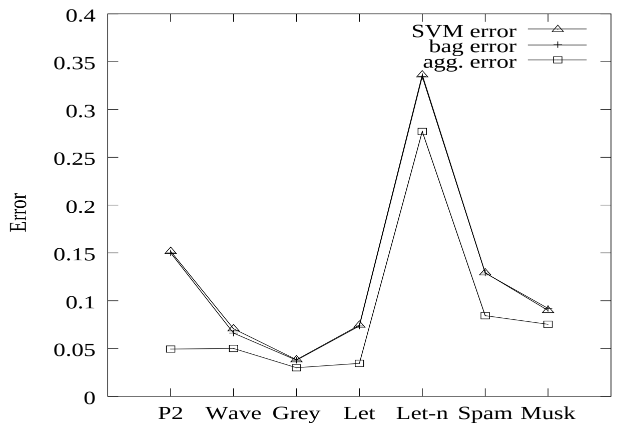
In these cases we could expect a strong decrement of the variance, while bias should remain substantially unchanged. Indeed our experiments (Sect. 5.3) reported a reduction of the net-variance over 90 %, as well as no substantial changes in bias.

Moreover the inherent parallelism of this process should permit to obtain a significant speed up using, for instance, simple cluster of workstations using message passing interface [140].

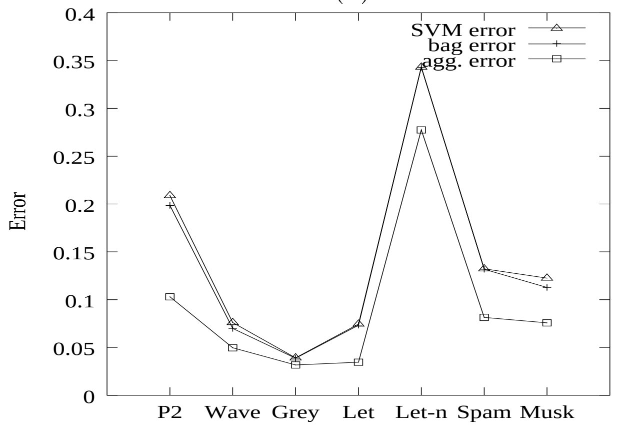
On the other hand we could use this approach for incremental learning strategies, collecting on-line samples in small data sets and aggregating the resulting classifiers. Of course this approach holds if the on-line samples are distributed according to an uniform probability distribution along time.

5.5 Summary of bias–variance analysis results in random aggregated and bagged ensembles of SVMs

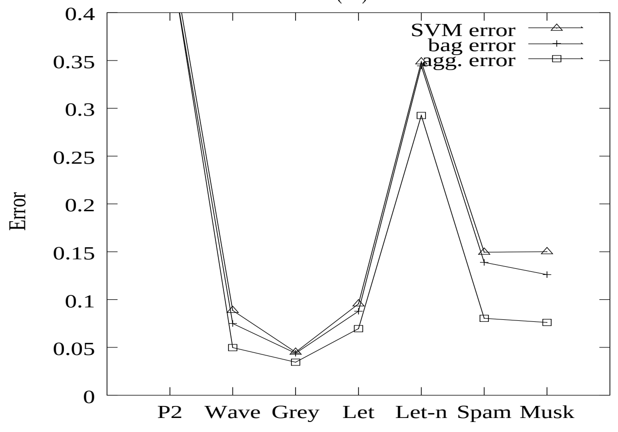
We conducted an extensive experimental analysis of bias–variance decomposition of the error in random aggregated and bagged ensembles of SVMs, involving training an testing of more than 10 millions of SVMs. In both cases we used relatively small data sets (100 examples) bootstrapped from a relatively large data set and reasonably large test sets to perform a reliable evaluation of bias and variance.



(a)

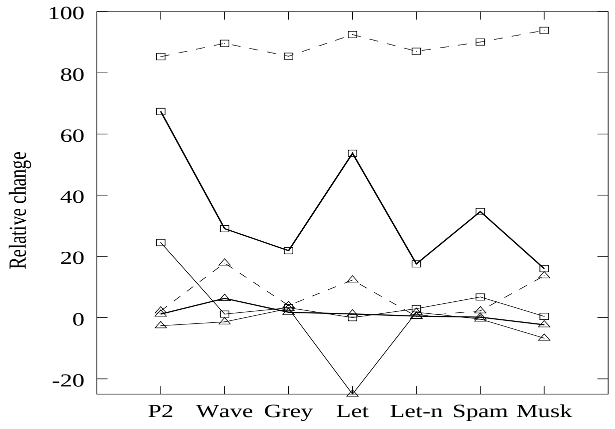


(b)

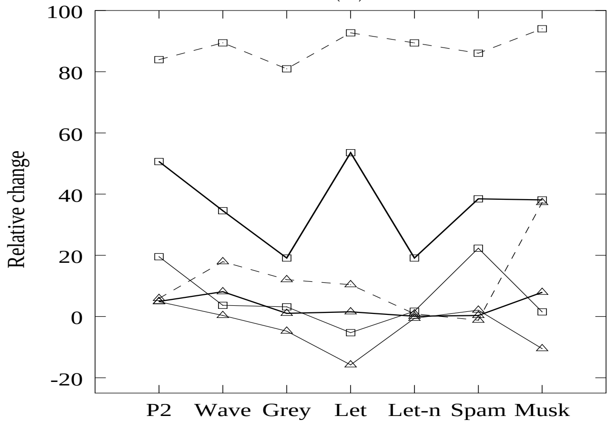


(c)

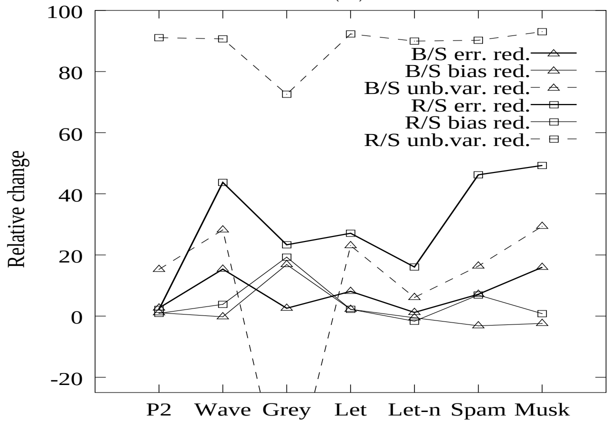
Figure 5.24: Comparison of the error between single SVMs, bagged and random aggregated ensembles of SVMs. Results refers to 7 different data sets. (a) Gaussian kernels (b) Polynomial kernels (c) Dot-product kernels.



(a)



(b)



(c)

Figure 5.25: Comparison of the relative error, bias and unbiased variance reduction between bagged and single SVMs (lines labeled with triangles), and between random aggregated and single SVMs (lines labeled with squares). B/S stands for Bagged versus Single SVMs, and R/S for random aggregated versus Single SVMs. Results refers to 7 different data sets. (a) Gaussian kernels (b) Polynomial kernels (c) Dot-product kernels.

Considering random aggregated ensembles, the most important fact we can observe consists in a very large reduction of the net-variance. It is always reduced close to 0, independently of the type of kernel used (Fig. 5.15, 5.18, 5.21). This behaviour is due primarily to the unbiased variance reduction, while the bias remains unchanged with respect to the single SVMs (Fig. 5.17, 5.20, 5.23).

Comparing bias–variance decomposition of the error between single and random aggregated ensembles of SVMs, we note that the relative error reduction varies from 10 to about 70 %, depending on the data set (Tab. 5.2). This reduction is slightly larger for high values of the C parameter (that reduces the bias of the base learners) and is due primarily to the reduction of the unbiased variance. Indeed in all data sets the relative reduction of the unbiased variance amounts to about 90 %, while bias remains substantially unchanged. The error of the ensemble is reduced to the bias of the single SVMs, because net and unbiased variance are largely reduced and close to 0.

Considering the bias-variance decomposition with respect to the number of base learners, we can observe that most of the decrement of the error occurs within the first iterations (from 10 to 30, depending on the data set), while the bias and the biased variance remains unchanged during all the iterations. The decrement of the error is almost entirely due to the decrement of the net and unbiased variance (Fig. 5.16, 5.19, 5.22).

With bagging also we have a reduction of the error, but not so large as with random aggregated ensembles (Fig. 5.24).

Indeed, unlike random aggregating, net and unbiased variance, although reduced, are not actually dropped to 0 (Fig. 5.4, 5.7, 5.10).

In particular, in our experiments, we obtained a smaller reduction of the average error (from 0 to 20 %) due to a lower decrement of the net-variance (about 35% against a reduction over 90 % with random aggregated ensembles), while bias remains unchanged or slightly increases (Fig. 5.25).

Random aggregating, approximated through undersampled bagging of sufficiently large training sets, shows a behavior very close to that predicted by theory (Sect. 5.1.1 and 5.1.2): eliminated variance and bias unchanged with respect to single base SVMs.

On the other hand experimental results confirm that bagging can be interpreted as an approximation of random aggregating (Sect. 5.1.3), as net-variance is reduced, but not canceled by bootstrap aggregating techniques, while bias remains unchanged or slightly increases.

The generalization error reduction provided by bootstrap aggregating techniques depends critically on the variance component of the error and on the bias proper of the base learner used. Using base learners with low bias and aggregating them through bootstrap replicates of the data can potentially reduce both the bias and variance components of the error.

Undersampled bagging, as an approximation of random aggregating can also provide very significant reduction of the variance and can be in practice applied to data mining problems when learning algorithms cannot comfortably manage very large data sets.

Chapter 6

SVM ensemble methods based on bias–variance analysis

Bias–variance theory provides a way to analyze the behavior of learning algorithms and to explain the properties of ensembles of classifiers [66, 48, 94]. Some ensemble methods increase expressive power of learning algorithms, thereby reducing bias [63, 33]. Other ensemble methods, such as methods based on random selection of input examples and input features [19, 24] reduce mainly the variance component of the error. In addition to providing insights into the behavior of learning algorithms, the analysis of the bias–variance decomposition of the error can identify the situations in which ensemble methods might improve base learner performances. Indeed the decomposition of the error into bias and variance can guide the design of ensemble methods by relating measurable properties of algorithms to the expected performance of ensembles [182]. In particular, bias–variance theory can tell us how to tune the individual base classifiers so as to optimize the overall performance of the ensemble.

The experiments on bias–variance decomposition of the error in SVMs gave us interestingly insights into the way SVMs learn (Chap. 4). Indeed, with single SVMs, we provided a bias–variance characterization of their learning properties, showing and explaining the relationships between kernel and SVMs parameters and their bias–variance characteristics (Sect. 4.4). Moreover bias–variance analysis in random aggregated and bagged ensembles (Chap. 5) showed how ensemble methods based on resampling techniques influence learning characteristics and generalization capabilities of single SVMs.

From a general standpoint, considering different kernels and different parameters of the kernel, we can observe that the minimum of the error, bias and net–variance (and in particular unbiased variance) do not match. For instance, considering RBF-SVM we see that we achieve the minimum of the error, bias and net–variance for different values of σ

(see, for instance, Fig. 4.6). Similar considerations can also be applied to polynomial and dot–product SVMs. Often, modifying parameters of the kernel, if we gain in bias we lose in variance and vice versa, even if this is not a rule.

Moreover, in our experiments, comparing bias–variance decomposition of the error between single and random aggregated ensembles of SVMs, we showed that the relative error reduction varies from 10 to about 70 %, depending on the data set. This reduction is due primarily to the reduction of the unbiased variance(about 90 %), while bias remains substantially unchanged.

With bagging also we have a reduction of the error, but not so large as with random aggregated ensembles. In particular the error with bagged ensembles of SVMs depends mainly on the bias component, but, unlike random aggregating, net and unbiased variance, although reduced, are not actually reduced to 0. In particular, in our experiments, we obtained a smaller reduction of the average error (from 0 to 20 %) due to a lower decrement of the net-variance (about 35% on the average against a reduction over 90 % with random aggregated ensembles), while bias remains unchanged or slightly increases.

Hence in both cases we have significant net-variance reduction (due to the unbiased variance decrement), while bias remains substantially unchanged.

In the light of the results of our extensive analysis for single SVMs and ensembles of SVMs, we propose two possible ways of applying bias–variance analysis to develop SVM-based ensemble methods.

The first approach tries to apply bias–variance analysis to enhance both accuracy and diversity of the base learners. The second research direction consists in bootstrap aggregating low bias base learners in order to lower both bias and variance. Regarding the first approach, only some very general research lines are depicted. About the second direction, a specific new method that we named *Lobag*, that is *Lo*w bias *bag*ged SVMs, is introduced, considering also different variants. Lobag applies bias–variance analysis to direct the tuning of Support Vector Machines to optimize the performances of bagged ensembles. Specifically, since bagging is primarily a variance-reduction method, and since the overall error is (to a first approximation) the sum of bias and variance, this suggests that SVMs should be tuned to minimize bias before being combined by bagging.

Numerical experiments show that Lobag compares favorably with bagging, and some preliminary results show that Lobag can be successfully applied to gene expression data analysis.

6.1 Heterogeneous Ensembles of SVMs

The analysis of bias–variance decomposition of the error in SVMs shows that the minimum of the overall error, bias, net–variance, unbiased and biased variance occurs often in different SVM models. These different behaviors of different SVM models could be in principle exploited to produce diversity in ensembles of SVMs. Although the diversity of base learner itself does not assure the error of the ensemble will be reduced [121], the combination of accuracy and diversity in most cases does [43]. As a consequence, we could select different SVM models as base learners by evaluating their accuracy and diversity through the bias-variance decomposition of the error.

For instance, our results show that the "optimal region" (low average loss region) is quite large in RBF-SVMs (Fig. 4.5). This means that C and σ do not need to be tuned extremely carefully. From this point of view, we can avoid time-consuming model selection by combining RBF-SVMs trained with different σ values all chosen from within the "optimal region." For instance, if we know that the error curve looks like the one depicted in Fig. 4.23, we could try to fit a sigmoid-like curve using only few values to estimate where the stabilized region is located. Then we could train an heterogeneous ensemble of SVMs with different σ parameters (located in the low bias region) and average them according to their estimated accuracy.

A high-level algorithm for *Heterogeneous Ensembles of SVMs* could include the following steps:

1. Individuate the "optimal region" through bias–variance analysis of the error
2. Select the SVMs with parameters chosen from within the optimal region defined by bias-variance analysis.
3. Combine the selected SVMs by majority or weighted voting according to their estimated accuracy.

We could use different methods or heuristics to find the "optimal region" (see Sect. 4.3.1.3) and we have to define also the criterion used to select the SVM models inside the "optimal region". The combination could be performed using also other approaches, such as minimum, maximum, average and OWA aggregating operators [105] or Behavior-Knowledge space method [87], Fuzzy aggregation rules [190], Decision templates [118] or Meta-learning techniques [150]. Bagging and boosting [63] methods can also be combined with this approach to further improve diversity and accuracy of the base learners.

If we apply bootstrap aggregating methods to the previous approach, exploiting also the fact that the most important learning parameter in gaussian kernels is represented by the spread σ (Sect. 4.3.1), we obtain the following *σ-Heterogeneous Ensembles of bagged SVMs*:

1. Apply bias-variance analysis to SVMs, in order to individuate the low bias region with respect to kernel parameter σ
2. Select a subset of values for σ, chosen from within the optimal region defined by bias-variance analysis (for instance n values).
3. For each σ value, selected in the previous step, train a bagged ensembles, for a total of n bagged ensembles
4. Combine the n ensembles by majority or weighted voting according to their estimated accuracy.

In Sect. 4.3.1 we showed that with gaussian kernels the optimal region with respect to σ is quite large, and also that σ is the most relevant parameter affecting the performances of gaussian SVMs. Hence we could select different σ values in order to improve diversity in the ensemble, while maintaining a high accuracy. We could also apply explicit measures of diversity [121] to select appropriate subsets of σ values. Then the variance of the set of σ-*heterogeneous* SVMs is lowered using bagging. Training multiple bagged SVM ensembles is computationally feasible, as in our experiments we showed that usually the error stabilizes within the first $20 - 30$ iterations (Sect. 5.2.2). These results were confirmed also in other experimental applications of bagged SVMs, for instance in bioinformatics [186].

This approach presents several open problems. Even if we discuss this point in Chapter 4, we need to choose an appropriate criterion to define the "optimal region": for instance, optimal in the sense of minimum overall error or minimum bias? Moreover, we the need to define the relationships between diversity and accuracy in selecting the "optimal" subset of σ values. Other questions are which diversity measure should be more appropriate in this context and whether the combination in the last step has to be performed at base learner or ensemble level.

Another more general approach, Breiman's random forests [19] "inspired", could use randomness at different levels to improve performances of ensemble methods. For instance, besides random selection of input samples, we could consider random selection of features, or also other types of randomness. In this context bias–variance analysis could select "appropriate" subsets of learning parameters, while randomness at different levels could be used to reduce the variance and/or the bias components of the error.

6.2 Bagged Ensemble of Selected Low-Bias SVMs

In chapter 5 we showed that random aggregating removes all variance, leaving only bias and noise. Hence, if bagging is a good approximation to random aggregating, it will also remove most of the variance. As a consequence, to minimize the overall error, bagging should be applied to base learners with minimum bias.

6.2.1 Parameters controlling bias in SVMs

We propose to tune SVMs to minimize the bias and then apply bagging to reduce (if not eliminate) variance, resulting in an ensemble with very low error. The key challenge, then, is to find a reasonable way of tuning SVMs to minimize their bias. The bias of SVMs is typically controlled by two parameters. First, recall that the objective function for (soft margin) SVMs has the form: $\|\mathbf{w}\|^2 + C \sum_i \xi_i$, where $\mathbf{w}$ is the vector of weights computed by the SVM and the ξ_i are the margin slacks, which are non-zero for data points that are not sufficiently separated by the decision boundary. The parameter C controls the tradeoff between fitting the data (achieved by driving the ξ_i's to zero) and maximizing the margin (achieved by driving $\|\mathbf{w}\|$ to zero). Setting C large should tend to minimize bias.

The second parameter that controls bias arises only in SVMs that employ parameterized kernels such as the polynomial kernel (where the parameter is the degree d of the polynomial) and RBF kernels (where the parameter is the width σ of the gaussian kernel). In Chap. 4 we showed that in gaussian and polynomial SVMs bias depends critically on these parameters.

6.2.2 Aggregating low bias base learners by bootstrap replicates

Bagging is an ensemble method effective for unstable learners. Under the bootstrap assumption, it reduces only variance. From bias-variance decomposition we know that unbiased variance reduces the error, while biased variance increases the error.

In theory, the bagged ensemble having a base learner with the minimum estimated bias will be the one with the minimum estimated generalization error, as the variance of the single base learner will be eliminated by the bagged ensemble, and the estimated generalization error will be reduced to the estimated bias of the single base learner. Indeed the bias (without noise) is $B(x) = L(t, y_m)$, where L is the loss function, t is the target and the main prediction $y_m = \arg\min_y E_D[L(y_D, y)]$, for a classification problem is the most voted class, that is the class selected by the bagged ensemble.

Hence bagging should be applied to low-bias classifiers, because the biased variance will be small, while bagging is essentially a variance reduction method, especially if well-tuned low bias base classifiers are used.

Summarizing, we can schematically consider the following observations:

- We know that bagging lowers net–variance (in particular unbiased variance) but not bias.
- From Domingos bias-variance decomposition we know that unbiased variance reduces the error, while biased variance increases the error. Hence bagging should be applied to low-bias classifiers, because the biased variance will be small.
- For single SVMs, the minimum of the error and the minimum of the bias are often achieved for different values of the tuning parameters C, d, and σ.
- SVMs are strong, low-biased learners, but this property depends on the proper selection of the kernel and its parameters.
- If we can identify low-biased base learners with no negligible unbiased variance, bagging can lower the error.
- Bias–variance analysis can identify SVMs with low bias.

We could try to exploit the low bias of a base learner to build a bagged ensemble that combines the reduced variance peculiar to bagging with low bias in order to reduce the generalization error. This is the key idea of *Lobag*, **Lo**w bias **bag**ged ensembles, that is bagged ensembles of low bias learning machines:

1. Estimate bias-variance decomposition of the error for different SVM models
2. Select the SVM model with the lowest bias
3. Perform bagging using as base learner the SVM with the estimated lowest bias.

From this algorithmic scheme, a major problem is the selection of a base learner with minimum estimated bias for a given data set. That is, given a learning set $\mathcal{D}$ and a parametric learning algorithm $\mathcal{L}(\cdot, \alpha)$ that generates a model $f_\alpha = \mathcal{L}(\cdot, \alpha)$, with α representing the parameters of the learning algorithm $\mathcal{L}$, we need to find:

$$f_B^* = \arg\min_\alpha B(f_\alpha, \mathcal{D}) \tag{6.1}$$

where $B()$ represents the bias of the model f_α estimated using the learning data set $\mathcal{D}$. This in turn requires an efficient way to estimate the bias–variance decomposition of the error.

6.2.3 Measuring Bias and Variance

To estimate bias and variance, we could use cross-validation in conjunction with bootstrap, or out-of-bag estimates (especially if we have small training sets), or hold-out techniques in conjunction with bootstrap techniques if we have sufficiently large training sets.

We propose to apply out-of-bag procedures [19] to estimate the bias and variance of SVMs trained with various parameter settings (see also Sect. 3.2.2). The procedure works as follows. First, we construct B bootstrap replicates of the available training data set $\mathcal{D}$ (e. g., $B = 200$): $D_1, \ldots, D_B$. Then we apply a learning algorithm $\mathcal{L}$ to each replicate S_b to obtain an hypothesis $f_b = \mathcal{L}(D_b)$. For each bootstrap replicate D_b, let $T_b = \mathcal{D} \backslash D_b$ be the ("out-of-bag") data points that do not appear in D_b. We apply hypothesis f_b to the examples in T_b and collect the results.

Consider a particular training example $(\mathbf{x}, t)$. On the average, this point will be in 63.2% of the bootstrap replicates D_b and hence in about 36.8% of the out-of-bag sets T_b. Let K be the number of times that $(\mathbf{x}, t)$ was out-of-bag; K will be approximately $0.368B$. The optimal prediction at $\mathbf{x}$ is just t. The main prediction y_m is the class that is most frequently predicted among the K predictions for $\mathbf{x}$. Hence, the bias is 0 if $y_m = t$ and 1 otherwise. The variance $V(\mathbf{x})$ is the fraction of times that $f_b(\mathbf{x}) \neq y_m$. Once the bias and variance have been computed for each individual point $\mathbf{x}$, they can be aggregated to give B, V_u, V_b, and V_n for the entire data set $\mathcal{D}$.

6.2.4 Selecting low-bias base learners.

Considering the second step of the Lobag algorithm (Sect. 6.2.2), that is the selection of the low bias SVM model, depending on the type of kernel and parameters considered, and on the way the bias is estimated for the different SVM models, different variants can be provided:

1. Selecting the RBF-SVM with the lowest bias with respect to the C and σ parameters.
2. Selecting the polynomial-SVM with the lowest bias with respect to the C and degree parameters.
3. Selecting the dot–prod-SVM with the lowest bias with respect to the C parameter.
4. Selecting the SVM with the lowest bias with respect both to the kernel and kernel parameters.

Note that here we propose SVMs as base learners, but other low bias base learners could in principle be used (for instance MLPs), as long as an analysis of their bias-variance characteristics suggests to apply them with bootstrap aggregating techniques. Of course, we cannot expect a high error reduction if the bias–variance analysis shows that the base learner has a high bias and a low unbiased variance.

A problem uncovered in this work is the estimate of the noise in real data sets. A straightforward approach simply consists in disregarding it, but in this way we could overestimate the bias. Some heuristic are proposed in [94], but the problem remains substantially unresolved.

6.2.5 Previous related work

Lobag can be interpreted as a variant of bagging: it estimates the bias of the SVM classifiers, selects low-bias classifiers, and then combines them by bootstrap aggregating.

Previous work with other classifiers is consistent with this approach. For example, several studies have reported that bagged ensembles of decision trees often give better results when the trees are not pruned [8, 41]. Unpruned trees have low bias and high variance. Similarly, studies with neural networks have found that they should be trained with lower weight decay and/or larger numbers of epochs before bagging to maximize accuracy of the bagged ensemble [5].

Unlike most learning algorithms, support vector machines have a built-in mechanism for variance reduction: from among all possible linear separators, they seek the maximum margin classifier. Hence, one might expect that bagging would not be very effective with SVMs. Previous work has produced varying results. On several real-world problems, bagged SVM ensembles are reported to give improvements over single SVMs [102, 186]. But for face detection, Buciu et al. [23] report negative results for bagged SVMs.

A few other authors have explored methods for tuning SVMs in ensembles. Collobert et al. [34] proposed solving very large scale classification problems by using meta-learning techniques combined with bagging. Derbeko et al. [40] applied an optimization technique from mathematical finance to reduce the variance of SVMs.

6.3 The lobag algorithm

The *Lobag* algorithm [183] accepts the following inputs: (a) a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, with $\mathbf{x}_i \in \mathbb{R}$ and $y_i \in \{-1, 1\}$, (b) a learning algorithm $\mathcal{L}(\cdot, \alpha)$, with tuning parameters α, and (c) a set $\mathcal{A}$ of possible settings of the α parameters to try. *Lobag* estimates the bias of each parameter setting $\alpha \in \mathcal{A}$, chooses the setting that minimizes the estimated bias, and applies the standard bagging algorithm to construct a bag of classifiers using $\mathcal{L}(\cdot, \alpha)$ with the chosen α value.

Unlike bagging, lobag selects the hypothesis with the estimated lowest bias to build the bootstrap aggregated classifier. As a consequence the core of the algorithm consists in evaluating bias–variance decomposition of the error varying the learning parameters α.

The remainder of this section provides the pseudo-code for *Lobag*.

6.3.1 The *Bias–variance decomposition* procedure

This procedure estimates the bias–variance decomposition of the error for a given learning algorithm $\mathcal{L}$ and learning parameters α.

The learning algorithm $\mathcal{L}$ returns a hypothesis $f_\alpha = \mathcal{L}(\mathcal{D}, \alpha)$ using a learning set $\mathcal{D}$, and it is applied to multiple bootstrap replicates D_b of the learning set $\mathcal{D}$ in order to generate a set $F_\alpha = \{f^b_\alpha\}^B_{b=1}$ of hypotheses f^b_α. The procedure returns the models F_α and the estimate of their loss and bias. For each learning parameter it calls `Evaluate_BV`, a procedure that provides an out-of-bag estimate of the bias–variance decomposition.

```
**Procedure** $[V, \mathcal{F}]$ **BV_decomposition** $(\mathcal{L}, \mathcal{A}, \mathcal{D}, B)$
Input:
    - Learning algorithm $\mathcal{L}$
    - Set of algorithm parameters $\mathcal{A}$
    - Data set $\mathcal{D}$
    - Number of bootstrap samples $B$
Output:
    - Set $V$ of triplets $(\alpha, loss, bias)$, where $loss$ and $bias$ are the estimated loss and bias of the model trained through the learning algorithm $\mathcal{L}$ with algorithm parameters $\alpha$.
    - Set of ensembles $\mathcal{F} = \{F_\alpha\}_{\alpha \in \mathcal{A}}$ with $F_\alpha = \{f^b_\alpha\}^B_{b=1}$
begin procedure
    $V = \emptyset$
    $\mathcal{F} = \emptyset$
    for each $\alpha \in \mathcal{A}$
    begin
        $F_\alpha = \emptyset$
        $\mathcal{T}_\alpha = \emptyset$
        for each $b$ from 1 to $B$
        begin
            $D_b = $ Bootstrap_replicate$(\mathcal{D})$
            $f^b_\alpha = \mathcal{L}(D_b, \alpha)$
            $T_b = \mathcal{D} \setminus D_b$
            $F_\alpha = F_\alpha \cup f^b_\alpha$
            $\mathcal{T}_\alpha = \mathcal{T}_\alpha \cup T_b$
        end
        $\mathcal{F} = \mathcal{F} \cup F_\alpha$
        $[loss,\ bias,\ variance]$ = Evaluate_BV $(F_\alpha, \mathcal{T}_\alpha)$
        $V = V \cup (\alpha, loss, bias)$
    end
end procedure.
```

The following procedure `Evaluate_BV` provides an out-of-bag estimate of the bias–variance decomposition of the error for a given model. The function $||z||$ is equal to 1 if z is true, and 0 otherwise. $E_{\mathbf{x}}[Q(\mathbf{x})]$ represents the expected value of $Q(\mathbf{x})$ with respect to the random variable $\mathbf{x}$.

```
Procedure $[ls, bs, var]$ Evaluate_BV $(F, \mathcal{T})$
Input:
    - Set $F = \{f_b\}_{b=1}^B$ of models trained on bootstrapped data
    - Set $\mathcal{T} = \{T_b\}_{b=1}^B$ of out-of-bag data sets
Output:
    - Out-of-bag estimate of the loss $ls$ of model $F$
    - Out-of-bag estimate of the bias $bs$ of model $F$
    - Out-of-bag estimate of the net variance $var$
        of model $F$
begin procedure
    for each $\mathbf{x} \in \cup_b T_b$
    begin
        $K = |\{T_b | \mathbf{x} \in T_b, 1 \le b \le B\}|$
        $p_1(\mathbf{x}) = \frac{1}{K} \sum_{b=1}^B ||(\mathbf{x} \in T_b)$ and $(f_b(\mathbf{x}) = 1)||$
        $p_{-1}(\mathbf{x}) = \frac{1}{K} \sum_{b=1}^B ||(\mathbf{x} \in T_b)$ and $(f_b(\mathbf{x}) = -1)||$
        $y_m = \arg\max(p_1, p_{-1})$
        $B(\mathbf{x}) = \left|\frac{y_m - t}{2}\right|$
        $V_u(\mathbf{x}) = \frac{1}{K} \sum_{b=1}^B ||(\mathbf{x} \in T_b)$ and $(B(\mathbf{x}) = 0)$
                                and $(y_m \ne f_b(\mathbf{x}))||$
        $V_b(\mathbf{x}) = \frac{1}{K} \sum_{b=1}^B ||(\mathbf{x} \in T_b)$ and $(B(\mathbf{x}) = 1)$
                                and $(y_m \ne f_b(\mathbf{x}))||$
        $V_n(\mathbf{x}) = V_u(\mathbf{x}) - V_b(\mathbf{x})$
        $Err(\mathbf{x}) = B(\mathbf{x}) + V_n(\mathbf{x})$
    end
    $ls = E_{\mathbf{x}}[Err(\mathbf{x})]$
    $bs = E_{\mathbf{x}}[B(\mathbf{x})]$
    $var = E_{\mathbf{x}}[V_n(\mathbf{x})]$
end procedure.
```

Even if the bias–variance decomposition of the error error could be, in principle, evaluated using other methods, such as multiple hold-out sets or cross-validation, the out-of-bag estimation is cheaper and allows us to exploit all the available data without separating the learning set in a training and a validation data set. Moreover the bias estimated for a single learning machine corresponds to the estimated error of the bagged ensemble having the same learning machine as base learner.

6.3.2 The *Model selection* procedure

After the estimate of the bias–variance decomposition of the error for different models, we need to select the model with the lowest bias. This procedure chooses in a straightforward way the learning parameters corresponding to the model with the lowest estimated bias and loss.

```
Procedure [$\alpha_B, \alpha_L, B_{min}, L_{min}, B_{L_{min}}$] Select_model ($V$)
Input:
    - Set $V$ of triplets $(\alpha, loss, bias)$, where $loss$ and $bias$ are the estimated loss and bias of
the model trained through the learning algorithm $\mathcal{L}$ with algorithm parameters $\alpha$.
Output:
    - Learning parameter $\alpha_B$ corresponding to the model with the estimated minimum bias
    - Learning parameter $\alpha_L$ corresponding to the model with the estimated minimum loss
    - Minimum $B_{min}$ of the bias values collected in $V$
    - Minimum $L_{min}$ of the loss values collected in $V$
    - Bias $B_{L_{min}}$ corresponding to the minimum loss $L_{min}$
begin procedure
    $L_{min} = \min_{v \in V} v.loss$
    $B_{min} = \min_{v \in V} v.bias$
    $\alpha_L = v.\alpha$ s.t. $v.loss = L_{min}$
    $\alpha_B = v.\alpha$ s.t. $v.bias = B_{min}$
    $B_{L_{min}} = v.bias$ s.t. $v.loss = L_{min}$
end procedure.
```

6.3.3 The overall *Lobag* algorithm

Using the procedure `BV_decomposition` we can implement a version of the Lobag algorithm that exhaustively explores a given set of learning parameters in order to build a low bias bagged ensemble.

Using out-of-bag estimate of the bias–variance decomposition of the error, the procedure `Select_model` selects the model with the minimum bias and/or minimum loss and returns the parameter values α_B and α_L that correspond respectively to the model with minimum bias and minimum loss. Then the Lobag and bagged ensembles are chosen through the procedure `Select_ensemble`: the Lobag ensemble has base learners with the minimum estimated bias, while the bagged ensemble has base learners with the minimum estimated loss.

Algorithm Lobag exhaustive

```
Input:
    - Learning algorithm $\mathcal{L}$
    - Set of algorithm parameters $\mathcal{A}$
    - Data set $\mathcal{D}$
    - Number of bootstrap samples $B$
Output:
    - Selected Lobag ensemble : $F_{Lob} = \{f^b_{\alpha_B}\}^B_{b=1}$
    - Selected bagged ensemble : $F_{Bag} = \{f^b_{\alpha_L}\}^B_{b=1}$
    - Oob error of the Lobag ensemble : $B_{min}$
    - Oob error of the bagged ensemble : $B_{L_{min}}$
    - Oob error of the single model : $L_{min}$
begin algorithm
    $V = \emptyset$
    $\mathcal{F} = \emptyset$
    $[V, \mathcal{F}]$ = BV_decomposition $(\mathcal{L}, \mathcal{A}, \mathcal{D}, B)$
    $[\alpha_B, \alpha_L, B_{min}, L_{min}, B_{L_{min}}]$ = Select_model $(V)$
    $F_{Lob} = \{f^b_{\alpha_B}\}^B_{b=1}$ = Select_ensemble $(\mathcal{F}, \alpha_B)$
    $F_{Bag} = \{f^b_{\alpha_L}\}^B_{b=1}$ = Select_ensemble $(\mathcal{F}, \alpha_L)$
end algorithm.
```

In order to speed up the computation, we could design variants of the exhaustive Lobag algorithm. For example, we could apply multidimensional search methods, such as the Powell's method [149], to select the tuning values that minimize bias.

Lobag presents several limitations. Such as classical bagging it is only an approximation of random aggregating: there is no guarantee of canceled net-variance. Moreover if variance is small, we cannot expect a significant decrement of the error. For data sets where the minimum of the bias and loss are achieved for the same learning parameters, lobag cannot improve bagging.

6.3.4 *Multiple hold-out Lobag* algorithm

This procedure shows how to apply lobag in a multiple-hold-out experimental setting, that is when multiple random splits of the data in a separated training and test set are provided, in order to reduce the effect of a particular split of the data on the evaluation of the generalization performance of learning machines.

```
Algorithm Multiple hold-out Lobag
Input:
    - Learning algorithm $\mathcal{L}$
    - Set of algorithm parameters $\mathcal{A}$
    - Data set $\mathcal{S}$
    - Number of bootstrap samples $B$
    - Number of splits $n$
Output:
    - Oob estimate of the error of the Lobag ensemble : $B_{min}$
    - Oob estimate of the error of the bagged ensemble : $B_{L_{min}}$
    - Oob estimate of the error of the single model : $L_{min}$
    - Hold-out estimate of the error of the Lobag ensemble : $L_{lobag}$
    - Hold-out estimate of the error of the bagged ensemble : $L_{bag}$
    - Hold-out estimate of the error of the single model : $L_{single}$
begin algorithm
    for each $i$ from 1 to $n$
    begin
        $V_i = \emptyset$
        $\mathcal{F}_i = \emptyset$
        $[D_i, T_i] = \texttt{Split}(\mathcal{S})$
        $[V_i, \mathcal{F}_i] = \texttt{BV\_decomposition}\ (\mathcal{L}, \mathcal{A}, D_i, B)$
    end
    for each $\alpha \in \mathcal{A}$
    begin
        $loss = \frac{1}{n} \sum_{i=1}^{n} (v.loss | v \in V_i,\ v.\alpha = \alpha)$
        $bias = \frac{1}{n} \sum_{i=1}^{n} (v.bias | v \in V_i,\ v.\alpha = \alpha)$
        $V = V \cup (\alpha, loss, bias)$
    end
    $[\alpha_B, \alpha_L, B_{min}, L_{min}, B_{L_{min}}] = \texttt{Select\_model}\ (V)$
    for each $i$ from 1 to $n$
    begin
        $F^i_{Lob} = \{f^{i,b}_{\alpha_B}\}_{b=1}^{B} = \texttt{Select\_ensemble}\ (\mathcal{F}_i,\ \alpha_B)$
        $F^i_{Bag} = \{f^{i,b}_{\alpha_L}\}_{b=1}^{B} = \texttt{Select\_ensemble}\ (\mathcal{F}_i,\ \alpha_L)$
    end
    $L_{single} = \texttt{Calc\_avg\_loss}(\{f^i_{\alpha_L}\}_{i=1}^{n}, \{T_i\}_{i=1}^{n})$
    $L_{bag} = \texttt{Calc\_avg\_loss}(\{F^i_{Bag}\}_{i=1}^{n}, \{T_i\}_{i=1}^{n})$
    $L_{lobag} = \texttt{Calc\_avg\_loss}(\{F^i_{Lob}\}_{i=1}^{n}, \{T_i\}_{i=1}^{n})$
end algorithm
```

The algorithm generates the lobag ensemble using multiple splits of a given data set $\mathcal{S}$ (procedure `Split`) in a separated learning D_i and test T_i sets. On each learning set D_i it is performed an out-of-bag estimate of the bias–variance decomposition of the error (procedure `BV_decomposition`, Sect. 6.3.1). The bias and the loss for each model are evaluated averaging the estimated bias and loss over each training set D_i. Then the SVMs with parameter α_B that corresponds to the minimum estimated bias are selected as base learners for the Lobag ensemble (procedure `Select_model`). Lobag and bagged ensembles are built through the procedure `Select_ensemble`, and the loss L_{lobag} of the Lobag ensemble is estimated averaging the error of the n ensembles $\{F^i_{Lob}\}^n_{i=1}$ over the test sets $\{T_i\}^n_{i=1}$, where $F^i_{Lob} = \{f^{i,b}_{\alpha_B}\}^B_{b=1}$, and the $f^{i,b}_{\alpha_B}$ is the SVM trained on the b^{th} bootstrap sample obtained from the i^{th} training set D_i using the learning parameter α_B. The algorithm provides an hold-out estimate of the generalization error of the lobag and bagged ensembles, averaging between the resulting loss on the different test sets T_i. The procedure `Calc_avg_loss` simply returns the average of the loss of the ensemble tested on different test sets:

```
Procedure $[Err]$ Calc_avg_loss $(\{f_i\}^n_{i=1}, \{T_i\}^n_{i=1})$
Input arguments:
    - Set $(\{f_i\}^n_{i=1}$ of the models trained on the different $\mathcal{S}\backslash T_i$ learning sets
    - Set $\{T_i\}^n_{i=1}$ of the multiple hold-out test sets $T_i$
Output:
    - Estimated average loss $Err$
begin procedure
    $Err = 0$
    for each $i$ from 1 to $n$
    begin
        $e = f_i(T_i)$
        $Err = Err + e$
    end
    $Err = Err/n$
end procedure.
```

6.3.5 ***Cross-validated Lobag*** algorithm

This procedure applies the lobag algorithm in the experimental framework of cross-validation.

Algorithm Cross-validated Lobag
Input:

```
    - Learning algorithm $\mathcal{L}$
    - Set of algorithm parameters $\mathcal{A}$
    - Data set $\mathcal{D}$
    - Number of bootstrap samples $B$
    - Number of folds $n$
Output:
    - Oob estimate of the error of the Lobag ensemble : $B_{min}$
    - Oob estimate of the error of the bagged ensemble : $B_{L_{min}}$
    - Oob estimate of the error of the single model : $L_{min}$
    - Cross-validated estimate of the error of the Lobag ensemble : $L_{lobag}$
    - Cross-validated estimate of the error of the bagged ensemble : $L_{bag}$
    - Cross-validated estimate of the error of the single model : $L_{single}$
begin algorithm
    $\{D_i\}_{i=1}^k =$ Generate_folds $(\mathcal{D}, k)$
    for each $i$ from 1 to $n$
    begin
        $V_i = \emptyset$
        $\mathcal{F}_i = \emptyset$
        $[V_i, \mathcal{F}_i] =$ BV_decomposition $(\mathcal{L}, \mathcal{A}, \mathcal{D} \backslash D_i, B)$
    end
    for each $\alpha \in \mathcal{A}$
    begin
        $loss = \frac{1}{n} \sum_{i=1}^{n}$ (loss of the element $v \in V_i$ s.t. $v.\alpha = \alpha$)
        $bias = \frac{1}{n} \sum_{i=1}^{n}$ (bias of the element $v \in V_i$ s.t. $v.\alpha = \alpha$)
        $V = V \cup (\alpha, loss, bias)$
    end
    $[\alpha_B, \alpha_L, B_{min}, L_{min}, B_{L_{min}}] =$ Select_model $(V)$
    for each $i$ from 1 to $n$
    begin
        $F_{Lob}^i = \{f_{\alpha_B}^{i,b}\}_{b=1}^B =$ Select_ensemble $(\mathcal{F}_i,\ \alpha_B)$
        $F_{Bag}^i = \{f_{\alpha_L}^{i,b}\}_{b=1}^B =$ Select_ensemble $(\mathcal{F}_i,\ \alpha_L)$
    end
    $L_{single} =$ Calc_avg_loss$(\{f_{\alpha_L}^i\}_{i=1}^n, \{D_i\}_{i=1}^n)$
    $L_{bag} =$ Calc_avg_loss$(\{F_{Bag}^i\}_{i=1}^n, \{D_i\}_{i=1}^n)$
    $L_{lobag} =$ Calc_avg_loss$(\{F_{Lob}^i\}_{i=1}^n, \{D_i\}_{i=1}^n)$
end algorithm
```

The selection of the lobag ensemble is performed through a cross-validated out-of-bag estimate of the bias–variance decomposition of the error. The data set is divided in k separated folds through the procedure `Generate_folds`. The oob estimate of the bias–

variance decomposition of the error is performed on each fold, and the overall estimate of bias and loss are computed averaging over the different folds. The algorithm provides also a cross-validated estimate of the generalization error of the resulting lobag and bagged ensembles (procedure `Calc_avg_loss`).

6.3.6 A ***heterogeneous Lobag*** approach

Using cross-validation or multiple hold-out techniques to evaluate the error, instead of using the "best" low bias model, obtained averaging the bias over all the folds or the different splits of the data, we could select each time the model with the lowest bias for each fold/split. In this way we could in principle to obtain different models, each one well-tuned for a specific fold/split.

Then we could combine them by majority or weighted voting, or we could combine them by multiple bootstrap aggregating in order to lower the variance. According to this second approach, we could bag each model selected on each different fold/split combining the different ensembles by majority or weighted voting. We could also introduce a second-level meta-learner in order to combine the base learners and the ensembles. This general approach could introduce diversity in the ensemble, while preserving at the same time the accuracy of the different "heterogeneous" base learners.

6.4 Experiments with lobag

We performed numerical experiments on different data sets to test the Lobag ensemble method using SVMs as base learners. We compared the results with single SVMs and classical bagged SVM ensembles.

6.4.1 Experimental setup

We employed 7 different two-class data sets, both synthetic and "real". We selected two synthetic data sets (*P2* and a two-class version of *Waveform*) and 5 "real" data sets (*Grey-Landsat*, *Letter*, reduced to the two-class problem of discriminating between the letters B and R, *Letter* with added 20% noise, *Spam*, and *Musk*). Most of them are from the UCI repository [135].

We applied two different experimental settings, using the same data sets, in order to compare lobag, classical bagging and single SVMs.

At first, we employed small $\mathcal{D}$ training sets and large test $\mathcal{T}$ sets in order to obtain a reliable

Table 6.1: Results of the experiments using pairs of train $\mathcal{D}$ and test $\mathcal{T}$ sets. E_{lobag}, E_{bag} and E_{SVM} stand respectively for estimated error of lobag, bagged and single SVMs on the test set $\mathcal{T}$. The three last columns show the confidence level according to the Mc Nemar test. **L/B**, **L/S** and **B/S** stand respectively for the comparison Lobag/Bagging, Lobag/Single SVM and Bagging/Single SVM. If the confidence level is equal to 1, no significant difference is registered.

Kernel type	$\mathbf{E_{lobag}}$	$\mathbf{E_{bag}}$	$\mathbf{E_{single}}$	**Confidence level**		
				L/B	**L/S**	**B/S**
	Data set *P2*					
Polyn.	0.1735	0.2008	0.2097	0.001	0.001	0.001
Gauss.	0.1375	0.1530	0.1703	0.001	0.001	0.001
	Data set *Waveform*					
Linear	0.0740	0.0726	0.0939	1	0.001	0.001
Polyn.	0.0693	0.0707	0.0724	1	0.1	0.1
Gauss.	0.0601	0.0652	0.0692	0.001	0.001	0.001
	Data set *Grey-Landsat*					
Linear	0.0540	0.0540	0.0650	1	0.001	0.001
Polyn.	0.0400	0.0440	0.0480	1	0.1	1
Gauss.	0.0435	0.0470	0.0475	0.1	0.1	1
	Data set *Letter-Two*					
Linear	0.0881	0.0929	0.1011	1	0.025	0.05
Polyn.	0.0701	0.0717	0.0831	1	0.05	0.1
Gauss.	0.0668	0.0717	0.0799	1	1	1
	Data set *Letter-Two with added noise*					
Linear	0.3535	0.3518	0.3747	1	1	0.1
Polyn.	0.3404	0.3715	0.3993	1	0.05	0.1
Gauss.	0.3338	0.3764	0.3829	0.05	0.025	1
	Data set *Spam*					
Linear	0.1408	0.1352	0.1760	0.05	0.001	0.001
Polyn.	0.0960	0.1034	0.1069	0.1	0.025	1
Gauss.	0.1130	0.1256	0.1282	0.005	0.001	1
	Data set *Musk*					
Linear	0.1291	0.1291	0.1458	1	0.001	0.001
Polyn.	0.1018	0.1157	0.1154	0.001	0.001	1
Gauss.	0.0985	0.1036	0.0936	0.05	1	0.05

estimate of the generalization error: the number of examples for $\mathcal{D}$ was set to 100, while the size of $\mathcal{T}$ ranged from a few thousands for the "real" data sets to ten thousands for synthetic data sets. Then we applied the Lobag algorithm described in Sect. 6.3, setting the number of samples bootstrapped from $\mathcal{D}$ to 100, and performing an out-of-bag estimate of the bias–variance decomposition of the error. The selected lobag, bagged and single SVMs were finally tested on the separated test set $\mathcal{T}$.

Then, using a different experimental set-up, we divided the data into a separated training $\mathcal{D}$ and test $\mathcal{T}$ sets. We then drew 30 data sets D_i from $\mathcal{D}$, each consisting of 100 examples drawn uniformly with replacement. Then we applied the lobag algorithm described in Sect. 6.3 to each of the D_i, setting the number of examples bootstrapped from each D_i to 100, and averaging both the out-of-bag estimation of the error and the error estimated on the separated test sets $\mathcal{T}$.

We developed new C++ classes and applications using the *NEURObjects* library [185] to implement the lobag algorithm and to analyze the results.

6.4.2 Results

Table 6.1 shows the results of the experiments with small $\mathcal{D}$ training sets and large $\mathcal{T}$ test sets. We measured 20 outcomes for each method: 7 data sets, and 3 kernels (gaussian, polynomial, and dot-product) applied to each data set except P2 for which we did not apply the dot-product kernel (because it was obviously inappropriate). For each pair of methods, we applied McNemar test [42] to determine whether there was a significant difference in predictive accuracy on the test set.

On nearly all the data sets, both bagging and Lobag outperform the single SVMs independently of the kernel used. The null hypothesis that Lobag has the same error rate as a single SVM is rejected at or below the 0.1 significance level in 17 of the 20 cases. Similarly, the null hypothesis that bagging has the same error rate as a single SVM is rejected at or below the 0.1 level in 13 of the 20 cases.

Most importantly, Lobag generally outperforms standard bagging. Lobag is statistically significantly better than bagging in 9 of the 20 cases, and significantly inferior only once.

These experiments are also shown graphically in Fig. 6.1. In this figure, each pair of points (joined by a line) corresponds to one of the 20 cases. The x coordinate of the point is the error rate of Lobag, the y coordinate is the error rate of either a single SVM (for the "star" shapes) or of standard bagging (for the "+" shapes). The line $y = x$ is plotted as well. Points above the line correspond to cases where Lobag had a smaller error rate. In most cases, the "star" is above the "+", which indicates that bagging had lower error than a single SVM.

Tab 6.2 summarizes the results of the comparison between bagging, lobag and single SVMs, according to the second experimental set-up (Sect. 6.4.1), using directly the out-of-bag estimate of the generalization error, averaged over the 30 different splits of the data. On

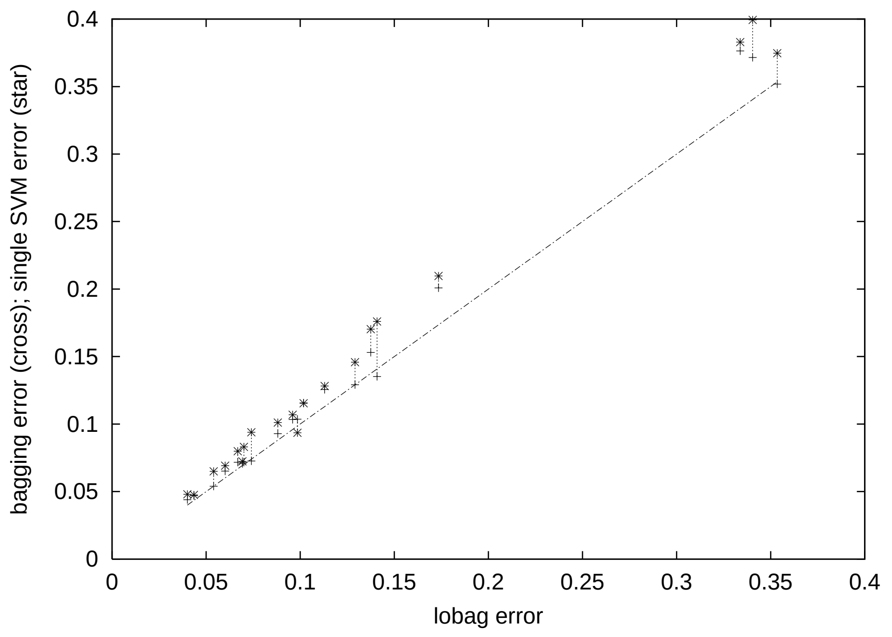


Figure 6.1: Graphical comparison of Lobag, bagging, and single SVM.

all the data sets both bagging and lobag outperform the single SVM, independently of the kernel used. The null hypothesis (no difference between the considered classifiers), is rejected at a 0.01 confidence level according to the resampled paired t test.

Moreover Lobag compares favorably to bagging. The average relative error reduction with respect to single SVMs is about 23 % for lobag and 18 % for bagged ensembles of SVMs. Using SVMs with gaussian kernels as base learners the difference of accuracy between lobag and bagging is significant at 0.01 confidence level on all the 7 data sets. We achieve the same results with polynomial kernels, except for the Grey-Landsat data set, where the difference is significant only at 0.05 level. With linear kernels there is no significant statistical difference in the data sets Waveform, Grey-Landsat and Musk. Using the separated test sets to evaluate the generalization error, the differences between bagging, lobag and also single SVMs become less significant, but also in this case lobag slightly tends to outperform bagging.

The outcomes of the second experimental approach confirm the results of the first one, even if they must be considered with caution, as the resampled t test suffers of a relatively large type I error, and consequently it can incorrectly detect a difference when no difference exists [42].

The results show that despite the ability of SVMs to manage the bias–variance tradeoff,

Table 6.2: Comparison of the results between lobag, bagging and single SVMs. E_{lobag}, E_{bag} and E_{SVM} stand respectively for average error of lobag, bagging and single SVMs. r.e.r. stands for relative error reduction between lobag and single SVMs and between bagging and single SVMs.

Kernel type	$\mathbf{E_{lobag}}$	$\mathbf{E_{bag}}$	$\mathbf{E_{SVM}}$	**r.e.r L/S**	**r.e.r B/S**
	Data set *P2*				
Polynomial	0.1593 ± 0.0293	0.1753 ± 0.0323	0.2161 ± 0.0321	26.28	18.88
Gaussian	0.1313 ± 0.0337	0.1400 ± 0.0367	0.1887 ± 0.0282	30.41	25.80
	Data set *Waveform*				
Linear	0.0713 ± 0.0312	0.0716 ± 0.0318	0.0956 ± 0.0307	25.41	25.10
Polynomial	0.0520 ± 0.0210	0.0597 ± 0.0214	0.0695 ± 0.0200	25.17	14.10
Gaussian	0.0496 ± 0.0193	0.0553 ± 0.0204	0.0668 ± 0.0198	25.74	17.21
	Data set *Grey-Landsat*				
Linear	0.0483 ± 0.0252	0.0487 ± 0.0252	0.0570 ± 0.0261	15.26	14.56
Polynomial	0.0413 ± 0.0252	0.0430 ± 0.0257	0.0472 ± 0.0257	12.50	8.89
Gaussian	0.0360 ± 0.0209	0.0390 ± 0.0229	0.0449 ± 0.0221	19.82	13.14
	Data set *Letter-Two*				
Linear	0.0890 ± 0.0302	0.0930 ± 0.0310	0.1183 ± 0.0281	24.76	21.38
Polynomial	0.0616 ± 0.0221	0.0656 ± 0.0247	0.0914 ± 0.0233	32.60	28.22
Gaussian	0.0553 ± 0.0213	0.0597 ± 0.0238	0.0875 ± 0.0244	36.80	31.77
	Data set *Letter-Two with added noise*				
Linear	0.2880 ± 0.0586	0.2993 ± 0.0604	0.3362 ± 0.0519	14.34	10.97
Polynomial	0.2576 ± 0.0549	0.2756 ± 0.0633	0.3122 ± 0.0502	17.49	11.72
Gaussian	0.2580 ± 0.0560	0.2706 ± 0.0607	0.3064 ± 0.0512	15.79	11.68
	Data set *Spam*				
Linear	0.1273 ± 0.0374	0.1353 ± 0.0400	0.1704 ± 0.0423	25.29	20.59
Polynomial	0.1073 ± 0.0379	0.1163 ± 0.0400	0.1407 ± 0.0369	23.74	17.34
Gaussian	0.1120 ± 0.0352	0.1190 ± 0.0380	0.1392 ± 0.0375	19.54	14.51
	Data set *Musk*				
Linear	0.1250 ± 0.0447	0.1250 ± 0.0447	0.1612 ± 0.0446	22.45	22.45
Polynomial	0.0960 ± 0.0331	0.1070 ± 0.0364	0.1295 ± 0.0357	25.87	17.37
Gaussian	0.0756 ± 0.0252	0.0793 ± 0.0253	0.0948 ± 0.0247	20.25	16.35

SVM performance can generally be improved by bagging, at least for small training sets. Furthermore, the best way to tune the SVM parameters is to adjust them to minimize bias and then allow bagging to reduce variance.

6.5 Application of lobag to DNA microarray data analysis

As an application of Lobag to "real world" problems, we consider a challenging classification problem in functional bioinformatics. In particular we applied Lobag to the analysis of DNA microarray data, in order to preliminary evaluate the effectiveness of the proposed ensemble method to small-sized and high-dimensional data, characterized also by a large biological variability.

DNA hybridization microarrays [57, 127] supply information about gene expression through measurements of mRNA levels of a large amount of genes in a cell. After extracting mRNA samples from the cells, preparing and marking the targets with fluorescent dyes, hybridizing with the probes printed on the microarrays and scanning the microarrays with a laser beam, the obtained TIFF images are processed with image analysis computer programs to translate the images into sets of fluorescent intensities proportional to the mRNA levels of the analyzed samples. After preprocessing and normalization stages, gene expression data of different cells or different experimental/functional conditions are collected in matrices for numerical processing: each row corresponds to the gene expression levels of a specific gene relative to all the examples, and each column corresponds to the expression data of all the considered genes relative to a specific cell example. Typically thousands of genes are used and analyzed for each microarray experiment.

Several supervised methods have been applied to the analysis of cDNA microarrays and high density oligonucleotide chips. These methods include decision trees, Fisher linear discriminant, multi-layer perceptrons (MLP), nearest-neighbors classifiers, linear discriminant analysis, Parzen windows and others [22, 53, 75, 101, 146]. In particular Support Vector Machines are well suited to manage and classify high dimensional data [188], as microarray data usually are, and have been recently applied to the classification of normal and malignant tissues using dot-product (linear) kernels [67], or polynomial and gaussian kernels in order to classify normal and tumoural tissues [179]. These types of kernels have also been successfully applied to the separation of functional classes of yeast genes using microarray expression data [22].

Furthermore, ensembles of learning machines are well-suited for gene expression data analysis, as they can reduce the variance due to the low cardinality of the available training sets, and the bias due to specific characteristics of the learning algorithm [43]. Indeed, in recent works, combinations of binary classifiers (one-versus-all and all-pairs) and Error Correcting Output Coding (ECOC) ensembles of MLP, as well as ensemble methods based on resampling techniques, such as bagging and boosting, have been applied to the analysis of DNA microarray data [194, 158, 54, 178, 186].

6.5.1 Data set and experimental set-up.

We used DNA microarray data available on-line. In particular we used the *GCM* data set obtained from the Whitehead Institute, Massachusetts Institute of Technology Center for Genome Research [158]. It is constituted of 300 human normal and tumor tissue specimens spanning 14 different malignant classes. In particular it contains 190 tumoral samples pertaining to 14 different classes, plus other 20 poorly differentiated tumor samples and 90 normal samples.

We grouped together the 14 different tumor classes and the poorly differentiated tumor samples to reduce the multi-class classification problem to a dichotomy in order to separate normal from malignant tissues. The 300 samples sequentially hybridized to oligonucleotide microarrays contain a total of 16063 probe sets (genes or ESTs) and we performed a stratified random splitting of these data in a training and test set of equal size. We preprocessed raw data using thresholding, filtering and normalization methods as suggested in [158]. Performances of Lobag ensembles of SVMs were compared with a standard bagging approach and with single SVMs, using subsets of genes selected through a simple feature-filtering method.

6.5.2 Gene selection.

We used a simple filter method, that is a gene selection method applied before and independently of the induction algorithm, originally proposed in [75]. The mean gene expression value across all the positive (μ_+) and negative (μ_-) examples are computed separately for each gene, together with their corresponding standard deviations (σ_+ and σ_-). Then the following statistic (a sort of signal-to-noise ratio) c_i is computed:

$$c_i = \frac{\mu_+ - \mu_-}{\sigma_+ + \sigma_-} \tag{6.2}$$

The larger is the distance between the mean values with respect to the sum of the spread of the corresponding values, more related is the gene to the discrimination of the positive and negative classes. Then the genes are ranked according to their c_i value, and the first and last m genes are selected. The main problem of this approach is the underlying independence assumption of the expression patterns of each gene: indeed it fails in detecting the role of coordinately expressed genes in carcinogenic processes. Eq. 6.2 can also be used to compute the weights for weighted gene voting [75], a minor variant of diagonal linear discriminant analysis [54].

With the *GCM* data set we applied a permutation test to automatically select a set of marker genes. It is a gene-specific variant of the neighborhood analysis proposed in [75]:

1. Calculate for each gene the signal-to-noise ratio (eq. 6.2)
2. Perform a gene-specific random permutation test:
   (a) Generate n random permutations of the class labels computing each time the signal-to-noise ratio for each gene.
   (b) Select a p significance level (e.g. $0 < p < 0.1$)
   (c) If the randomized signal-to-noise ratio is larger than the actual one in less than $p \cdot n$ random permutations, select that gene as significant for discrimination at p significance level.

This is a simple method to estimate the significance of the matching of a given phenotype to a particular set of marker genes: its time complexity is $\mathcal{O}(nd)$, where n is the number of examples and d the number of features (genes). Moreover the permutation test is distribution independent: no assumptions about the functional form of the gene distribution are supposed.

6.5.3 Results.

Using the above gene-specific neighborhood analysis, we selected 592 genes correlated with tumoral examples ($p = 0.01$) (set A) and about 3000 genes correlated with normal examples ($p = 0.01$) (set B). Then we used the genes of set A and the 592 genes with highest signal-to-noise ratio values of set B to assemble a selected set composed by 1184 genes. The results of the classifications with single SVMs, with and without gene selection are summarized in Tab. 6.3.

Table 6.3: *GCM* data set: results with single SVMs

Kernel type and parameters	**Err.all genes**	**Err.sel. genes**	**Relative err.red.**
Dot-product, C=20	0.2600	0.2279	12.31 %
Polynomial, deg=6 C=5	0.7000	0.2275	—-
Polynomial, deg=2 C=10	0.6900	0.2282	—-
Gaussian, σ=2 C=50	0.3000	0.2185	27.33 %

There is a significant increment in accuracy using only a selected subset of genes for classification. According to the McNemar test [44], in all cases there is a statistical significant difference at 0.05 confidence level between SVMs trained with and without feature selection. Polynomial kernels without feature selection fail to classify normal from malignant tissues.

Tab. 6.4 summarizes the results of bagged SVMs on the *GCM* data set. Even if not always

Table 6.4: *GCM* data set: compared results of single and bagged SVMs

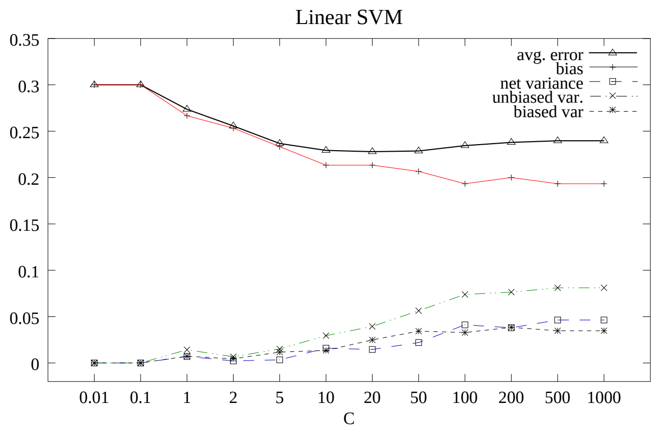
Kernel type and parameters	**Error SVMs**	**Error bagged**	**Relative err.red.**
Dot-product, C=10	0.2293	0.2200	4.06 %
Dot-product, C=20	0.2279	0.2133	6.41 %
Polynomial, degree=6 C=5	0.2275	0.2000	12.09 %
Polynomial, degree=2 C=10	0.2282	0.2133	6.53 %
Gaussian, sigma=2 C=50	0.2185	0.2067	5.40 %
Gaussian, sigma=10 C=200	0.2233	0.2067	7.44 %

there is a statistical significant difference (according to Mc Nemar test) between single and bagged SVMs, in all cases bagged ensembles of SVMs outperform single SVMs. The degree of enhancement depends heavily on the possibility to reduce the variance component of the error, as bagging is mainly a variance-reduction ensemble method.

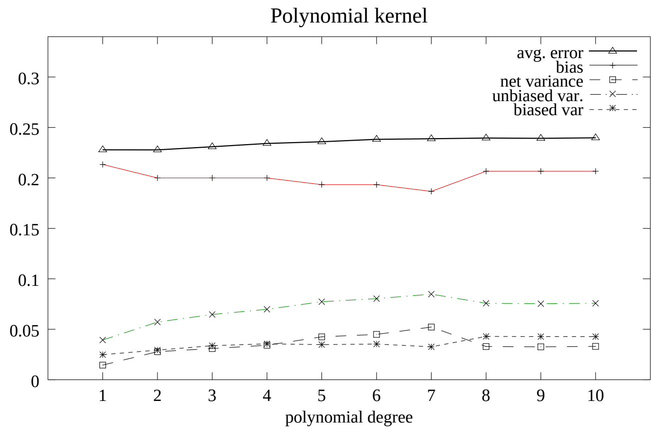
Indeed, performing a bias–variance analysis of the error of single SVMs on the *GCM* data set, we note that bias largely overrides the variance components of the error, and in this case we cannot expect a very large reduction of the error with bagging (Fig. 6.2). Nonetheless we can see that both with linear SVMs (Fig. 6.2 a), polynomial (Fig. 6.2 b), and gaussian (Fig. 6.2 c) SVMs, the minimum of the estimated error and the estimated bias are achieved for different learning parameters, showing that in this case Lobag could improve the performance, even if we cannot expect a large reduction of the overall error, as the bias largely dominates the variance component of the error (Fig. 6.2).

Indeed with Lobag the error is lowered, both with respect to single and bagged SVMs (Tab. 6.5). As expected, both bagged and lobag ensembles of SVMs outperform single SVMs, but with lobag the reduction of the error is significant at 0.05 confidence level, according to Mc Nemar's test, for all the applied kernels, while for bagging it is significant only for the polynomial kernel. Moreover Lobag always outperforms bagging, even if the error reduction is significant only if linear or polynomial kernels are used. Summarizing, Lobag achieves significant enhancements with respect to single SVMs in analyzing DNA microarray data, and also lowers the error with respect to classical bagging.

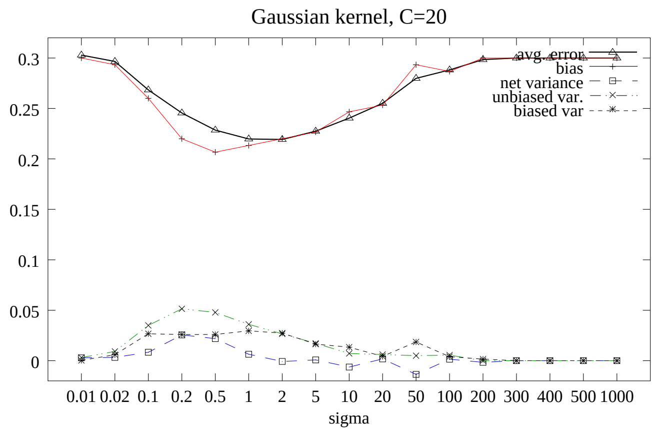
Even if these results seem quite encouraging, they must be considered only as preliminary, and we need more experiments, using different data sets and using more reliable cross-



(a)



(b)



(c)

Figure 6.2: *GCM* data set: bias-variance decomposition of the error in bias, net-variance, unbiased and biased variance, while varying the regularization parameter with linear SVMs (a), the degree with polynomial kernels (b), and the kernel parameter σ with gaussian SVMs (c).

validated estimates of the error, in order to evaluate more carefully the applicability of the lobag method to DNA microarray data analysis. Moreover we need also to assess the quality of the classifiers using for instance ROC curves or appropriate quality measures as shown, for instance, in [76].

Table 6.5: *GCM* data set: compared results of single, bagged and Lobag SVMs on gene expression data. An asterisk in the last three columns points out that a statistical significant difference is registered ($p = 0.05$) according to the Mc Nemar test.

Kernel type	**Error SVMs**	**Error bagged**	**Error Lobag**	**Err.red. SVM$-$ > bag**	**Err.red. SVM$-$ > Lobag**	**Err.red. bag$-$ > Lobag**
Dot-product	0.2279	0.2133	0.1933	6.41 %	15.18 % $*$	9.38 % $*$
Polynomial	0.2275	0.2000	0.1867	12.09 % $*$	17.93 % $*$	6.65 % $*$
Gaussian	0.2185	0.2067	0.1933	5.40 %	11.53 % $*$	6.48 %

Conclusions

Cosa volevo dire, non lo so,
però ho ragione, e i fatti,
mi cosano.
Palmiro Cangini

Research on ensemble methods focused on the combination/aggregation of learning machines, while the specific characteristics of the base learners that build them up have been only partially considered. On the contrary we started from the learning properties and the behavior of the learning algorithms used to generate the base predictors, in order to build around them ensemble methods well-tuned to their learning characteristics.

To this purpose we showed that bias–variance theory provides a way to analyze the behavior of learning algorithms and to explain the properties of ensembles of classifiers. Moreover we showed that the analysis of the bias–variance decomposition of the error can identify the situations in which ensemble methods might improve base learner performances.

We conducted an extended bias–variance analysis of the error in single SVMs (Chap. 4), bagged and random aggregated ensembles of SVMs (Chap. 5), involving training and testing of over 10 million of SVMs, in order to gain insights into the way single and ensembles of SVMs learn. To this purpose we developed procedures to measure bias and variance in classification problems according to Domingos bias–variance theory.

In particular we performed an analysis of bias and variance in single SVMs, considering gaussian, polynomial, and dot–product kernels. The relationships between parameters of the kernel and bias, net–variance, unbiased and biased variance were studied, discovering regular patterns and specific trends. We provided a characterization of bias–variance decomposition of the error, showing that in gaussian kernels we can individuate at least three different regions with respect to the σ (spread) parameter, while in polynomial kernels the U shape of the error can be determined by the combined effects of bias and unbiased variance. The analysis also revealed that the expected trade-off between bias and variance holds only for dot product kernels, while other kernels showed more complex relationships. We discovered that the minimum of bias, variance and overall error are often achieved for different values of the regularization and kernel parameters, as a result of a different learning behavior of the trained SVM.

According to Breiman's theoretical results, we showed that bagging can be interpreted as an approximation of random aggregating, that is a process by which base learners, trained on samples drawn accordingly to an unknown probability distribution from the entire universe population, are aggregated through majority voting or averaging their outputs. These experiments showed that the theoretical property of a very large variance reduction holds for random aggregating, while for bagging we registered a smaller reduction, but not a total elimination as in random aggregating.

Bias–variance analysis in random aggregated SVM ensembles suggested also to aggregate ensembles of SVMs for very large scale data mining problems using undersampled bagging. Unfortunately, time was not sufficient to follow this promising research line.

On the basis of the information supplied by bias-variance analysis we proposed two research lines for designing ensembles of SVMs. The first one applies bias-variance analysis to construct a heterogeneous, diverse set of low-bias classifiers. The second presents an ensemble method, that is Lobag, that selects low bias base learners (well tuned - low bias SVMs) and then combines them through bagging. The key issue of the bias–variance evaluation is performed through an efficient out-of-bag estimate of the bias–variance decomposition of the error. This approach affects both bias, through the selection of low bias base learners, and variance, through bootstrap aggregation of the selected low bias base learners. Numerical experiments showed that low bias bagged ensembles of SVMs compare favorably both to single and bagged SVM ensembles, and preliminary experiments with DNA microarray data suggested that this approach might be effective with high-dimensional low sized data, as gene expression data usually are.

Open questions, related to some topics only partially developed in this thesis, delineate possible future works and developments.

In our research planning, we pursued to execute a bias–variance analysis for ensemble methods based on resampling techniques. However we performed only a bias–variance analysis in bagged SVMs, but we plan to perform the same analysis in boosted ensembles of SVMs, in order to gain insights into the behavior of boosted SVMs with "strong" well-tuned SVMs, comparing them with "weak" not-optimally-tuned SVMs.

We showed that bias–variance analysis is an effective tool to design new ensemble methods tuned to specific bias-variance characteristics of base learners. In particular "strong" base learners such as SVMs work well with lobag. We expect that this will be true for base learners that exhibit relatively large variance and low bias, especially with relatively small data sets. Hence we plan to experiment with other low bias base learners (e.g. Multi Layer Perceptrons) in order to gain insights into their learning behavior and to evaluate if we can apply them with Lobag or to evaluate if we can design other base learner specific ensemble methods.

In order to speed-up the computation, we plan to implement variants of the basic Lobag algorithm. For instance we could apply multidimensional search methods, such as the Powell method, to select the tuning values that minimize bias.

In our experiments we did not consider noise, but it is present in most real data sets. As a result noise is embodied into bias, and bias itself is overestimated. Even if the evaluation of noise in real data sets is an open problem, we plan to evaluate the role of noise in synthetic and real data sets, in order to develop variants of lobag specific for noisy data.

The peculiar characteristics of Lobag and the preliminary results relative to its application to DNA microarray data, encourage us to continue along this research line (Sect. 6.5). In particular we plan to perform an extended experimental analysis with high-dimensional low-sized gene expression data, evaluating Lobag with respect to single SVMs (largely applied in bioinformatics) and to other ensemble methods (bagging and boosting, for instance), assessing carefully the quality and the reliability of the classifiers

We provided only high-level algorithmic schemes for heterogeneous ensembles of SVMs (Sect. 6.1). We plan to design and implement these algorithms, possibly integrating this approach with an explicit evaluation of the diversity of the base learners, using measures and approaches similar to those proposed by Kuncheva [121].

In our experiments with bagged and random aggregated ensembles of SVMs we used relatively small and fixed sized bootstrap samples. A natural development of these experiments could be to explicitly consider the cardinality of the data, setting-up a series of experiments with increasing number of examples for each randomly drawn data set, in order to evaluate the effect of the sample size on bias, variance and instability of base learners.

Experiments with random aggregated ensembles of SVMs showed that we could use undersampled bagging with large data sets in order to obtain large reduction of the unbiased variance, without significant increment in bias (Sect. 5.4). We plan to develop this approach, also in relation with the above research on the effect of the cardinality of the data in random aggregating. The main goal of this research line is the development of ensemble methods for very large data mining problems.

In our experiments we did not explicitly consider the characteristics of the data. Nonetheless, as we expected and our experiments suggested, different data characteristics influence bias–variance patterns in learning machines. To this purpose we plan to explicitly analyze the relationships between bias–variance decomposition of the error and data characteristics, using data complexity measures based on geometrical and topological characteristics of the data [126, 84].

Bibliography

[1] D. Aha and R. Bankert. Cloud classification using error-correcting output codes. In *Artificial Intelligence Applications: Natural Science, Agriculture and Environmental Science*, volume 11, pages 13–28. 1997.

[2] E.L. Allwein, R.E. Schapire, and Y. Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.

[3] E. Alpaydin and E. Mayoraz. Learning error-correcting output codes from data. In *ICANN'99*, pages 743–748, Edinburgh, UK, 1999.

[4] R. Anand, G. Mehrotra, C.K. Mohan, and S. Ranka. Efficient classification for multiclass problems using modular neural networks. *IEEE Transactions on Neural Networks*, 6:117–124, 1995.

[5] T. Andersen, M. Rimer, and T. R. Martinez. Optimal artificial neural network architecture selection for voting. In *Proceedings of the IEEE International Joint Conference on Neural Networks IJCNN'01*, pages 790–795. IEEE, 2001.

[6] G. Bakiri and T.G. Dietterich. Achieving high accuracy text-to-speech with machine learning. In *Data mining in speech synthesis*. 1999.

[7] R. Battiti and A.M. Colla. Democracy in neural nets: Voting schemes for classification. *Neural Networks*, 7:691–707, 1994.

[8] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning*, 36(1/2):525–536, 1999.

[9] J. Benediktsson, J. Sveinsson, O. Ersoy, and P. Swain. Parallel consensual neural networks. *IEEE Transactions on Neural Networks*, 8:54–65, 1997.

[10] J. Benediktsson and P. Swain. Consensus theoretic classification methods. *IEEE Transactions on Systems, Man and Cybernetics*, 22:688–704, 1992.

[11] A. Berger. Error correcting output coding for text classification. In *IJCAI'99: Workshop on machine learning for information filtering*, 1999.

[12] C. M. Bishop. *Neural Networks for Pattern Recognition.* Clarendon Press, Oxford, 1995.

[13] A. Blum and R.L. Rivest. Training a 3-node neural network is NP-complete. In *Proc. of the 1988 Workshop ob Computational Learning Learning Theory*, pages 9–18, San Francisco, CA, 1988. Morgan Kaufmann.

[14] R.C. Bose and D.K. Ray-Chauduri. On a class of error correcting binary group codes. *Information and Control*, (3):68–79, 1960.

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

[16] L. Breiman. Bias, variance and arcing classifiers. Technical Report TR 460, Statistics Department, University of California, Berkeley, CA, 1996.

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

[18] L. Breiman. Prediction games and arcing classifiers. *Neural Computation*, 11(7):1493–1517, 1999.

[19] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[20] M. Breukelen van, R.P.W. Duin, D. Tax, and J.E. Hartog den. Combining classifiers fir the recognition of handwritten digits. In *Ist IAPR TC1 Workshop on Statistical Techniques in Pattern Recognition*, pages 13–18, Prague, Czech republic, 1997.

[21] G.J. Briem, J.A. Benediktsson, and J.R. Sveinsson. Boosting. Bagging and Consensus Based Classification of Multisource Remote Sensing Data. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems. Second International Workshop, MCS 2001, Cambridge, UK*, volume 2096 of *Lecture Notes in Computer Science*, pages 279–288. Springer-Verlag, 2001.

[22] M. Brown et al. Knowledge-base analysis of microarray gene expression data by using support vector machines. *PNAS*, 97(1):262–267, 2000.

[23] I. Buciu, C. Kotropoulos, and I. Pitas. Combining Support Vector Machines for Accurate Face Detection. In *Proc. of ICIP'01*, volume 1, pages 1054–1057, 2001.

[24] P. Buhlmann and B. Yu. Analyzing bagging. *Annals of Statistics*, 30:927–961, 2002.

[25] P. Chan and S. Stolfo. Meta-learning for multistrategy and parallel learning. In *Proc. 2^{th} International Workshop on Multistrategy Learning*, pages 150–165, 1993.

[26] P. Chan and S. Stolfo. A comparative evaluation of voting and meta-learning on partitioned data. In *Proc.* 12th *ICML*, pages 90–98, 1995.

[27] D.. Chen. *Statistical estimates for Kleinberg's method of Stochastic Discrimination.* PhD thesis, The State University of New York, Buffalo, USA, 1998.

[28] K.J. Cherkauker. Human expert-level performance on a scientific image analysis task by a system using combined artificial neural networks. In Chan P., editor, *Working notes of the AAAI Workshop on Integrating Multiple Learned Models*, pages 15–21. 1996.

[29] S. Cho and J. Kim. Combining multiple neural networks by fuzzy integral and robust classification. *IEEE Transactions on Systems, Man and Cybernetics*, 25:380–384, 1995.

[30] S. Cho and J. Kim. Multiple network fusion using fuzzy logic. *IEEE Transactions on Neural Networks*, 6:497–501, 1995.

[31] S. Cohen and N. Intrator. A Hybrid Projection Based and Radial Basis Function Architecture. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems. First International Workshop, MCS 2000, Cagliari, Italy*, volume 1857 of *Lecture Notes in Computer Science*, pages 147–156. Springer-Verlag, 2000.

[32] S. Cohen and N. Intrator. Automatic Model Selection in a Hybrid Perceptron/Radial Network. In *Multiple Classifier Systems. Second International Workshop, MCS 2001, Cambridge, UK*, volume 2096 of *Lecture Notes in Computer Science*, pages 349–358. Springer-Verlag, 2001.

[33] S. Cohen and N. Intrator. Hybrid Projection-based and Radial Basis Function Architecture: Initial Values and Global Optimisation. *Pattern Analysis and Applications*, 5(2):113–120, 2002.

[34] R. Collobert, S. Bengio, and Y. Bengio. A Parallel Mixture of SVMs for Very Large Scale Problems. *Neural Computation*, 14(5):1105–1114, 2002.

[35] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[36] K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 35–46, 2000.

[37] N. Cristianini and J. Shawe-Taylor. *An introduction to Support Vector Machines and other kernel-based learning methods.* Cambridge University Press, Cambridge, UK, 2000.

[38] N.C. de Condorcet. *Essai sur l' application de l' analyse à la probabilité des decisions rendues à la pluralité des voix.* Imprimerie Royale, Paris, 1785.

[39] A. Demiriz, K.P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1-3):225–254, 2002.

[40] P. Derbeko, R. El-Yaniv, and R. Meir. Variance Optimized Bagging. In *Machine Learning: ECML 2002*, volume 2430 of *Lecture Notes in Computer Science*, pages 60–71. Springer-Verlag, 2002.

[41] T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–158, 2000.

[42] T.G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, (7):1895–1924, 1998.

[43] T.G. Dietterich. Ensemble methods in machine learning. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems. First International Workshop, MCS 2000, Cagliari, Italy*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, 2000.

[44] T.G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. *Machine Learning*, 40(2):139–158, 2000.

[45] T.G. Dietterich and G. Bakiri. Error - correcting output codes: A general method for improving multiclass inductive learning programs. In *Proceedings of AAAI-91*, pages 572–577. AAAI Press / MIT Press, 1991.

[46] T.G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, (2):263–286, 1995.

[47] P. Domingos. A unified bias–variance decomposition. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA, 2000.

[48] P. Domingos. A Unified Bias-Variance Decomposition and its Applications. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 231–238, Stanford, CA, 2000. Morgan Kaufmann.

[49] P. Domingos. A Unified Bias-Variance Decomposition for Zero-One and Squared Loss. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 564–569, Austin, TX, 2000. AAAI Press.

[50] H. Drucker and C. Cortes. Boosting decision trees. In *Advances in Neural Information Processing Systems*, volume 8. 1996.

[51] H. Drucker, C. Cortes, L. Jackel, Y. LeCun, and V. Vapnik. Boosting and other ensemble methods. *Neural Computation*, 6(6):1289–1301, 1994.

[52] R.O. Duda and P.E. Hart. *Pattern classification and scene analysis.* Wiley & Sons, New York, 1973.

[53] S. Dudoit, J. Fridlyand, and T. Speed. Comparison of Discrimination Methods for the Classification of Tumors Using Gene Expression Data. Technical Report 576, Department of Statistics, University of California, Berkeley, 2000.

[54] S. Dudoit, J. Fridlyand, and T. Speed. Comparison of discrimination methods for the classification of tumors using gene expression data. *JASA*, 97(457):77–87, 2002.

[55] R.P.W. Duin and D.M.J. Tax. Experiments with Classifier Combination Rules. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems. First International Workshop, MCS 2000, Cagliari, Italy*, volume 1857 of *Lecture Notes in Computer Science*, pages 16–29. Springer-Verlag, 2000.

[56] B. Efron and R. Tibshirani. *An introduction to the Bootstrap.* Chapman and Hall, New York, 1993.

[57] M. Eisen and P. Brown. DNA arrays for analysis of gene expression. *Methods Enzymol.*, 303:179–205, 1999.

[58] S.E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 524–532. Morgan Kauffman, San Mateo, CA, 1990.

[59] U.M. Fayyad, C. Reina, and P.S. Bradley. Initialization of iterative refinement clustering algorithms. In *Proc.* 14^{th} *ICML*, pages 194–198, 1998.

[60] E. Filippi, M. Costa, and E. Pasero. Multi-layer perceptron ensembles for increased performance and fault-tolerance in pattern recognition tasks. In *IEEE International Conference on Neural Networks*, pages 2901–2906, Orlando, Florida, 1994.

[61] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[62] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and Systems Sciences*, 55(1):119–139, 1997.

[63] Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, pages 148–156. Morgan Kauffman, 1996.

[64] J. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 39(5), 2001.

[65] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 38(2):337–374, 2000.

[66] J.H. Friedman. On bias, variance, 0/1 loss and the curse of dimensionality. *Data Mining and Knowledge Discovery*, 1:55–77, 1997.

[67] T.S. Furey, N. Cristianini, N. Duffy, D. Bednarski, M. Schummer, and D. Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10):906–914, 2000.

[68] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias-variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

[69] R. Ghani. Using error correcting output codes for text classification. In *ICML 2000: Proceedings of the 17th International Conference on Machine Learning*, pages 303–310, San Francisco, US, 2000. Morgan Kaufmann Publishers.

[70] J. Ghosh. Multiclassifier systems: Back to the future. In *Multiple Classifier Systems. Third International Workshop, MCS2002, Cagliari, Italy*, volume 2364 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, 2002.

[71] G. Giacinto and F. Roli. Dynamic Classifier Fusion. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems. First International Workshop, MCS 2000, Cagliari, Italy*, volume 1857 of *Lecture Notes in Computer Science*, pages 177–189. Springer-Verlag, 2000.

[72] G. Giacinto and F. Roli. An approach to the automatic design of multiple classifier systems. *Pattern Recognition Letters*, 22(1):25–33, 2001.

[73] G. Giacinto and F. Roli. Dynamic classifier selection based on multiple classifier behaviour. *Pattern Recognition*, 34(9):179–181, 2001.

[74] G. Giacinto, F. Roli, and G. Fumera. Selection of classifiers based on multiple classifier behaviour. In *SSPR/SPR*, pages 87–93, 2000.

[75] T.R. Golub et al. Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. *Science*, 286:531–537, 1999.

[76] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene Selection for Cancer Classification using Support Vector Machines. *Machine Learning*, 46(1/3):389–422, 2002.

[77] L. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[78] T. Hastie and R. Tibshirani. *Generalized Additive Models.* Chapman and Hall, London, 1990.

[79] T. Hastie and R. Tibshirani. Classification by pairwise coupling. *The Annals of Statistics*, 26(1):451–471, 1998.

[80] T. Heskes. Bias/Variance Decompostion for Likelihood-Based Estimators. *Neural Computation*, 10:1425–1433, 1998.

[81] T.K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[82] T.K. Ho. Complexity of Classification Problems ans Comparative Advantages of Combined Classifiers. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems. First International Workshop, MCS 2000, Cagliari, Italy*, volume 1857 of *Lecture Notes in Computer Science*, pages 97–106. Springer-Verlag, 2000.

[83] T.K. Ho. Data Complexity Analysis for Classifiers Combination. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems. Second International Workshop, MCS 2001, Cambridge, UK*, volume 2096 of *Lecture Notes in Computer Science*, pages 53–67, Berlin, 2001. Springer-Verlag.

[84] T.K. Ho and M. Basu. Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):289–300, 2002.

[85] T.K. Ho, J.J. Hull, and S.N. Srihari. Decision combination in multiple classifiers. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(4):405–410, 1997.

[86] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4:251–257, 1991.

[87] Y.S. Huang and Suen. C.Y. Combination of multiple experts for the recognition of unconstrained handwritten numerals. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17:90–94, 1995.

[88] L. Hyafil and R.L. Rivest. Constructing optimal binary decision tree is np-complete. *Information Processing Letters*, 5(1):15–17, 1976.

[89] S. Impedovo and A. Salzo. A New Evaluation Method for Expert Combination in Multi-expert System Designing. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems. First International Workshop, MCS 2000, Cagliari, Italy*, volume 1857 of *Lecture Notes in Computer Science*, pages 230–239. Springer-Verlag, 2000.

[90] R.A. Jacobs. Methods for combining experts probability assessment. *Neural Computation*, 7:867–888, 1995.

[91] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, and G.E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):125–130, 1991.

[92] A. Jain, R. Duin, and J. Mao. Statistical pattern recognition: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:4–37, 2000.

[93] G. James. *Majority vote classifiers: theory and applications.* PhD thesis, Department of Statistics - Stanford University, Stanford, CA, 1998.

[94] G. James. Variance and bias for general loss function. *Machine Learning*, 2003. (in press).

[95] C. Ji and S. Ma. Combinination of weak classifiers. *IEEE Trans. Neural Networks*, 8(1):32–42, 1997.

[96] T. Joachims. Making large scale SVM learning practical. In Smola A. Scholkopf B., Burges C., editor, *Advances in Kernel Methods - Support Vector Learning*, pages 169–184. MIT Press, Cambridge, MA, 1999.

[97] M. Jordan and R. Jacobs. Hierarchies of adaptive experts. In *Advances in Neural Information Processing Systems*, volume 4, pages 985–992. Morgan Kauffman, San Mateo, CA, 1992.

[98] M.I. Jordan and R.A. Jacobs. Hierarchical mixture of experts and the em algorithm. *Neural Computation*, 6:181–214, 1994.

[99] H. Kargupta, B. Park, D. Hershberger, and E. Johnson. Collective data mining:a new perspective toward distributed data mining. In H. Kargupta and P. Chan, editors, *Advances in in Distributed and Parallel Knowledge Discovery.* MIT/AAAI Press, 1999.

[100] J.M. Keller, P. Gader, H. Tahani, J. Chiang, and M. Mohamed. Advances in fuzzy integratiopn for pattern recognition. *Fuzzy Sets and Systems*, 65:273–283, 1994.

[101] J. Khan et al. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Medicine*, 7(6):673–679, 2001.

[102] H.C. Kim, S. Pang, H.M. Je, D. Kim, and S.Y. Bang. Pattern Classification Using Support Vector Machine Ensemble. In *Proc. of ICPR'02*, volume 2, pages 20160–20163. IEEE, 2002.

[103] F. Kimura and M. Shridar. Handwritten Numerical Recognition Based on Multiple Algorithms. *Pattern Recognition*, 24(10):969–983, 1991.

[104] J. Kittler. Combining classifiers: a theoretical framework. *Pattern Analysis and Applications*, (1):18–27, 1998.

[105] J. Kittler, M. Hatef, R.P.W. Duin, and Matas J. On combining classifiers. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[106] J. Kittler and F. Roli. *Multiple Classifier Systems, First International Workshop, MCS 2000, Cagliari, Italy*, volume 1857 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2000.

[107] J. Kittler and F. Roli. *Multiple Classifier Systems, Second International Workshop, MCS2001, Cambridge, UK*, volume 2096 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2001.

[108] E.M. Kleinberg. On the Algorithmic Implementation of Stochastic Discrimination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[109] E.M. Kleinberg. Stochastic Discrimination. *Annals of Mathematics and Artificial Intelligence*, pages 207–239, 1990.

[110] E.M. Kleinberg. An overtraining-resistant stochastic modeling method for pattern recognition. *Annals of Statistics*, 4(6):2319–2349, 1996.

[111] E.M. Kleinberg. A Mathematically Rigorous Foundation for Supervised Learning. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems. First International Workshop, MCS 2000, Cagliari, Italy*, volume 1857 of *Lecture Notes in Computer Science*, pages 67–76. Springer-Verlag, 2000.

[112] R. Kohavi and D.H. Wolpert. Bias plus variance decomposition for zero-one loss functions. In *Proc. of the Thirteenth International Conference on Machine Learning, The Seventeenth International Conference on Machine Learning*, pages 275–283, Bari, Italy, 1996. Morgan Kaufmann.

[113] J. Kolen and Pollack J. Back propagation is sensitive to initial conditions. In *Advances in Neural Information Processing Systems*, volume 3, pages 860–867. Morgan Kauffman, San Francisco, CA, 1991.

[114] E. Kong and T.G. Dietterich. Error - correcting output coding correct bias and variance. In *The XII International Conference on Machine Learning*, pages 313–321, San Francisco, CA, 1995. Morgan Kauffman.

[115] A. Krogh and J. Vedelsby. Neural networks ensembles, cross validation and active learning. In D.S. Touretzky, G. Tesauro, and T.K. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 107–115. MIT Press, Cambridge, MA, 1995.

[116] L.I. Kuncheva. Genetic algorithm for feature selection for parallel classifiers. *Information Processing Letters*, 46:163–168, 1993.

[117] L.I. Kuncheva. An application of OWA operators to the aggragation of multiple classification decisions. In *The Ordered Weighted Averaging operators. Theory and Applciations*, pages 330–343. Kluwer Academic Publisher, USA, 1997.

[118] L.I. Kuncheva, J.C. Bezdek, and R.P.W. Duin. Decision templates for multiple classifier fusion: an experimental comparison. *Pattern Recognition*, 34(2):299–314, 2001.

[119] L.I. Kuncheva, F. Roli, G.L. Marcialis, and C.A. Shipp. Complexity of Data Subsets Generated by the Random Subspace Method: An Experimental Investigation. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems. Second International Workshop, MCS 2001, Cambridge, UK*, volume 2096 of *Lecture Notes in Computer Science*, pages 349–358. Springer-Verlag, 2001.

[120] L.I. Kuncheva and C.J. Whitaker. Feature Subsets for Classifier Combination: An Enumerative Experiment. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems. Second International Workshop, MCS 2001, Cambridge, UK*, volume 2096 of *Lecture Notes in Computer Science*, pages 228–237. Springer-Verlag, 2001.

[121] L.I. Kuncheva and C.J. Whitaker. Measures of diversity in classifier ensembles. *Machine Learning*, 51:181–207, 2003.

[122] L. Lam. Classifier combinations: Implementations and theoretical issues. In *Multiple Classifier Systems. First International Workshop, MCS 2000, Cagliari, Italy*, volume 1857 of *Lecture Notes in Computer Science*, pages 77–86. Springer-Verlag, 2000.

[123] L. Lam and C. Sue. Optimal combination of pattern classifiers. *Pattern Recognition Letters*, 16:945–954, 1995.

[124] L. Lam and C. Sue. Application of majority voting to pattern recognition: an analysis of its behavior and performance. *IEEE Transactions on Systems, Man and Cybernetics*, 27(5):553–568, 1997.

[125] W.B. Langdon and B.F. Buxton. Genetic programming for improved receiver operating characteristics. In J. Kittler and F. Roli, editors, *Second International Conference on Multiple Classifier System*, volume 2096 of *LNCS*, pages 68–77, Cambridge, 2001. Springer Verlag.

[126] M. Li and P Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications.* Springer-Verlag, Berlin, 1993.

[127] D.J. Lockhart and E.A. Winzeler. Genomics, gene expression and DNA arrays. *Nature*, 405:827–836, 2000.

[128] R. Maclin and D. Opitz. An empirical evaluation of bagging and boosting. In *Fourteenth National Conference on Artificial Intelligence*, pages 546–551, Providence, USA, 1997. AAAI-Press.

[129] L. Mason, P. Bartlett, and J. Baxter. Improved generalization through explicit optimization of margins. *Machine Learning*, 2000.

[130] F. Masulli and G. Valentini. Comparing decomposition methods for classification. In R.J. Howlett and L.C. Jain, editors, *KES'2000, Fourth International Conference on Knowledge-Based Intelligent Engineering Systems & Allied Technologies*, pages 788–791, Piscataway, NJ, 2000. IEEE.

[131] F. Masulli and G. Valentini. Effectiveness of error correcting output codes in multiclass learning problems. In *Lecture Notes in Computer Science*, volume 1857, pages 107–116. Springer-Verlag, Berlin, Heidelberg, 2000.

[132] F. Masulli and G. Valentini. Dependence among Codeword Bits Errors in ECOC Learning Machines: an Experimental Analysis. In *Lecture Notes in Computer Science*, volume 2096, pages 158–167. Springer-Verlag, Berlin, 2001.

[133] F. Masulli and G. Valentini. Quantitative Evaluation of Dependence among Outputs in ECOC Classifiers Using Mutual Information Based Measures. In K. Marko and P. Webos, editors, *Proceedings of the International Joint Conference on Neural Networks IJCNN'01*, volume 2, pages 784–789, Piscataway, NJ, USA, 2001. IEEE.

[134] E. Mayoraz and M. Moreira. On the decomposition of polychotomies into dichotomies. In *The XIV International Conference on Machine Learning*, pages 219–226, Nashville, TN, July 1997.

[135] C.J. Merz and P.M. Murphy. UCI repository of machine learning databases, 1998. www.ics.uci.edu/mlearn/MLRepository.html.

[136] D. Modha and Spangler W.S. Clustering hypertext with application to web searching. In *Proc. of the ACM Hypertext 2000 Conference*, San Antonio, TX, 2000.

[137] M. Moreira and E. Mayoraz. Improved pairwise coupling classifiers with correcting classifiers. In C. Nedellec and C. Rouveirol, editors, *Lecture Notes in Artificial Intelligence, Vol. 1398*, pages 160–171, Berlin, Heidelberg, New York, 1998.

[138] D.W. Opitz and J.W. Shavlik. Actively searching for an effective neural network ensemble. *Connection Science*, 8(3/4):337–353, 1996.

[139] N.C. Oza and K. Tumer. Input Decimation Ensembles: Decorrelation through Dimensionality Reduction. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems. Second International Workshop, MCS 2001, Cambridge, UK*, volume 2096 of *Lecture Notes in Computer Science*, pages 238–247. Springer-Verlag, 2001.

[140] P.S. Pacheco. *Parallel Programming with MPI.* Morgan Kauffman, San Francisco, CA, 1997.

[141] H.S. Park and S.W. Lee. Off-line recognition of large sets handwritten characters with multiple Hidden-Markov models. *Pattern Recognition*, 29(2):231–244, 1996.

[142] J. Park and I.W. Sandberg. Approximation and radial basis function networks. *Neural Computation*, 5(2):305–316, 1993.

[143] B. Parmanto, P. Munro, and H. Doyle. Improving committe diagnosis with resampling techniques. In D.S. Touretzky, M. Mozer, and M. Hesselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 882–888. MIT Press, Cambridge, MA, 1996.

[144] B. Parmanto, P. Munro, and H. Doyle. Reducing variance of committee predition with resampling techniques. *Connection Science*, 8(3/4):405–416, 1996.

[145] D. Partridge and W.B. Yates. Engineering multiversion neural-net systems. *Neural Computation*, 8:869–893, 1996.

[146] P. Pavlidis, J. Weston, J. Cai, and W.N. Grundy. Gene functional classification from heterogenous data. In *Fifth International Conference on Computational Molecular Biology*, 2001.

[147] M.P. Perrone and L.N. Cooper. When networks disagree: ensemble methods for hybrid neural networks. In Mammone R.J., editor, *Artificial Neural Networks for Speech and Vision*, pages 126–142. Chapman & Hall, London, 1993.

[148] W.W. Peterson and E.J.Jr. Weldon. *Error correcting codes.* MIT Press, Cambridge, MA, 1972.

[149] W.H. Press, S.A. Teukolski, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 1992.

[150] A. Prodromidis, P. Chan, and S. Stolfo. Meta-Learning in Distributed Data Mining Systems: Issues and Approaches. In H. Kargupta and P. Chan, editors, *Advances in Distributed Data Mining*, pages 81–113. AAAI Press, 1999.

[151] J.R. Quinlan. *C4.5 Programs for Machine Learning*. Morgan Kauffman, 1993.

[152] G. Ratsch, B. Scholkopf, A.J. Smola, K-R. Muller, T. Onoda, and S. Mika. nu-arc ensemble learning in the presence of outliers. In S. A. Solla, T.K. Leen, and K-R. Muller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, Cambridge, MA, 2000.

[153] Y. Raviv and N. Intrator. Bootstrapping with noise: An effective regularization technique. *Connection Science*, 8(3/4):355–372, 1996.

[154] G. Rogova. Combining the results of several neural neetworks classifiers. *Neural Networks*, 7:777–781, 1994.

[155] F. Roli and G. Giacinto. Analysis of linear and order statistics combiners for fusion of imbalanced classifiers. In *Multiple Classifier Systems. Third International Workshop, MCS2002, Cagliari, Italy*, volume 2364 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.

[156] F. Roli, G. Giacinto, and G. Vernazza. Methods for Designing Multiple Classifier Systems. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems. Second International Workshop, MCS 2001, Cambridge, UK*, volume 2096 of *Lecture Notes in Computer Science*, pages 78–87. Springer-Verlag, 2001.

[157] F. Roli and J. Kittler. *Multiple Classifier Systems, Third International Workshop, MCS2002, Cagliari, Italy*, volume 2364 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2002.

[158] S. Ramaswamy et al. Multiclass cancer diagnosis using tumor gene expression signatures. *PNAS*, 98(26):15149–15154, 2001.

[159] R. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[160] R.E. Schapire. The strenght of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[161] R.E. Schapire. A brief introduction to boosting. In Thomas Dean, editor, *16th International Joint Conference on Artificial Intelligence*, pages 1401–1406. Morgan Kauffman, 1999.

[162] R.E. Schapire, Y. Freund, P. Bartlett, and W. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

[163] R.E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[164] H. Schwenk and Y. Bengio. Training methods for adaptive boosting of neural networks. In *Advances in Neural Information Processing Systems*, volume 10, pages 647–653. 1998.

[165] A. Sharkey, N. Sharkey, and G. Chandroth. Diverse neural net solutions to a fault diagnosis problem. *Neural Computing and Applications*, 4:218–227, 1996.

[166] A Sharkey, N. Sharkey, U. Gerecke, and G. Chandroth. The test and select approach to ensemble combination. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems. First International Workshop, MCS 2000, Cagliari, Italy*, volume 1857 of *Lecture Notes in Computer Science*, pages 30–44. Springer-Verlag, 2000.

[167] A. (editor) Sharkey. *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems.* Springer-Verlag, London, 1999.

[168] M. Skurichina and R.P.W. Duin. Bagging, boosting and the randon subspace method for linear classifiers. *Pattern Analysis and Applications.* (in press).

[169] M. Skurichina and R.P.W. Duin. Bagging for linear classifiers. *Pattern Recognition*, 31(7):909–930, 1998.

[170] M. Skurichina and R.P.W. Duin. Bagging and the Random Subspace Method for Redundant Feature Spaces. In *Multiple Classifier Systems. Second International Workshop, MCS 2001, Cambridge, UK*, volume 2096 of *Lecture Notes in Computer Science*, pages 1–10. Springer-Verlag, 2001.

[171] A Strehl and J. Ghosh. Cluster Ensembles - A Knowledge Reuse Framework for Combining Multiple Partitions. *Journal of Machine Learning Research*, 3:583–617, 2002.

[172] C. Suen and L. Lam. Multiple classifier combination methodologies for different output levels. In *Multiple Classifier Systems. First International Workshop, MCS 2000, Cagliari, Italy*, volume 1857 of *Lecture Notes in Computer Science*, pages 52–66. Springer-Verlag, 2000.

[173] R. Tibshirani. Bias, variance and prediction error for classification rules. Technical report, Department of Preventive Medicine and Biostatistics and Department od Statistics, University of Toronto, Toronto, Canada, 1996.

[174] K. Tumer and J. Ghosh. Error correlation and error reduction in ensemble classifiers. *Connection Science*, 8(3/4):385–404, 1996.

[175] K. Tumer and N.C. Oza. Decimated input ensembles for improved generalization. In *IJCNN-99, The IEEE-INNS-ENNS International Joint Conference on Neural Networks*, 1999.

[176] G. Valentini. Upper bounds on the training error of ECOC-SVM ensembles. Technical Report TR-00-17, DISI - Dipartimento di Informatica e Scienze dell' Informazione - Università di Genova, 2000. ftp://ftp.disi.unige.it/person/ValentiniG/papers/TR-00-17.ps.gz.

[177] G. Valentini. Classification of human malignancies by machine learning methods using DNA microarray gene expression data. In G.M. Papadourakis, editor, *Fourth International Conference Neural Networks and Expert Systems in Medicine and HealthCare*, pages 399–408, Milos island, Greece, 2001. Technological Educational Institute of Crete.

[178] G. Valentini. Gene expression data analysis of human lymphoma using support vector machines and output coding ensembles. *Artificial Intelligence in Medicine*, 26(3):283–306, 2002.

[179] G. Valentini. Supervised gene expression data analysis using Support Vector Machines and Multi-Layer perceptrons. In *Proc. of KES'2002, the Sixth International Conference on Knowledge-Based Intelligent Information & Engineering Systems, special session Machine Learning in Bioinformatics*, Amsterdam, the Netherlands, 2002. IOS Press.

[180] G. Valentini. Bias–variance analysis in bagged svm ensembles: data and graphics. DISI, Dipartimento di Informatica e Scienze dell' Informazione, Università di genova, Italy., 2003. ftp://ftp.disi.unige.it/person/ValentiniG/papers/bv-svm-bagging.ps.gz.

[181] G. Valentini. Bias–variance analysis in random aggregated svm ensembles: data and graphics. DISI, Dipartimento di Informatica e Scienze dell' Informazione, Università di genova, Italy., 2003. ftp://ftp.disi.unige.it/person/ValentiniG/papers/bv-svm-undersampling.ps.gz.

[182] G. Valentini and T.G. Dietterich. Bias–variance analysis and ensembles of SVM. In *Multiple Classifier Systems. Third International Workshop, MCS2002, Cagliari, Italy*, volume 2364 of *Lecture Notes in Computer Science*, pages 222–231. Springer-Verlag, 2002.

[183] G. Valentini and T.G. Dietterich. Low Bias Bagged Support Vector Machines. In *Proc. ICML 2003, The Twentieth International Conference on Machine Learning*, Washington D.C., USA, 2003.

[184] G. Valentini and F. Masulli. Ensembles of learning machines. In *Neural Nets WIRN-02*, volume 2486 of *Lecture Notes in Computer Science*, pages 3–19. Springer-Verlag, 2002.

[185] G. Valentini and F. Masulli. NEURObjects: an object-oriented library for neural network development. *Neurocomputing*, 48(1–4):623–646, 2002.

[186] G. Valentini, M. Muselli, and F. Ruffino. Bagged Ensembles of SVMs for Gene Expression Data Analysis. In *IJCNN2003, The IEEE-INNS-ENNS International Joint Conference on Neural Networks*, Portland, USA, 2003. IEEE.

[187] J. Van Lint. *Coding theory.* Spriger Verlag, Berlin, 1971.

[188] V. N. Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998.

[189] G Wahba. Spline models for observational data. In *SIAM*, Philadelphia, USA, 1990.

[190] D. Wang, J.M. Keller, C.A. Carson, K.K. McAdoo-Edwards, and C.W. Bailey. Use of fuzzy logic inspired features to improve bacterial recognition through classifier fusion. *IEEE Transactions on Systems, Man and Cybernetics*, 28B(4):583–591, 1998.

[191] D.H. Wolpert. Stacked Generalization. *Neural Networks*, 5:241–259, 1992.

[192] K. Woods, W.P. Kegelmeyer, and K. Bowyer. Combination of multiple classifiers using local accuracy estimates. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(4):405–410, 1997.

[193] L Xu, C Krzyzak, and C. Suen. Methods of combining multiple classifiers and their applications to handwritting recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 22(3):418–435, 1992.

[194] C. Yeang et al. Molecular classification of multiple tumor types. In *ISMB 2001, Proceedings of the 9th International Conference on Intelligent Systems for Molecular Biology*, pages 316–322, Copenaghen, Denmark, 2001. Oxford University Press.