PhD Dissertation

International Doctorate School in Information and Communication Technologies

DISI - University of Trento

Local Approaches for Fast, Scalable and Accurate Learning with Kernels

Nicola Segata

Advisor:

Prof. Enrico Blanzieri

Università degli Studi di Trento

December 2009

Abstract

The present thesis deals with the fundamental machine learning issues of increasing the accuracy of learning systems and their computational performances. The key concept which is exploited throughout the thesis, is the tunable trade-off between local and global approaches to learning, integrating the effective setting of Instance Based Learning with the sound foundations of Statistical Learning Theory. Four are the main contributions of the thesis in this context: (i) a theoretical analysis and empirical evaluation of the Local SVM approach, (ii) a family of operators on kernels to obtain Quasi-Local kernels, (iii) the framework of Local Kernel Machines, and (iv) a local maximal margin approach to noise reduction. In our analysis of Local SVM, we derive a new learning bound starting from the theory of Local Learning Algorithms, and we showed that Local SVM statistically significantly overcomes the classification accuracy of SVM in a number of scenarios. The novel family of operators on kernels integrates local feature-space information into any kernel obtaining Quasi-Local *kernels, mixing the effect of the input kernel with a kernel which is local in the feature space of the input one. With Local Kernel Machine we show that locality can be exploited to obtain fast and scalable kernel machines, whereas existing fast approximated SVM solvers try to globally smooth the decision functions. Fast Local Kernel SVM (FaLK-SVM) trains a set of local SVMs on redundant neighbourhoods in the training set selecting at testing time the most appropriate model for each query point. Theoretically supported by a recent result relating consistency and localizability, our approach divides the separation function in solutions of local optimization problems that can be handled very efficiently. For this novel approach, we derive a fast local model selection strategy, theoretical learning bounds and favourable complexity bounds. Local Kernel Machines can also be applied to the problem of detecting and removing noisy examples from datasets in order to enhance the generalization ability of Instance Based Learning and for data-cleansing. The local maximal margin principle provides a more robust alternative to the majority rule on which almost all the existing noise reduction techniques are based and a scalable version of the approach extends the feasibility of the noise reduction task to large datasets such as genomic-scale biological data. Extensive evaluations of the proposed techniques are carried out on more than 100 datasets with up to 3 millions examples, and statistically significantly showed that Quasi-Local kernels are more accurate than the corresponding input kernels using SVM, and that Local Kernel Machines can improve the generalization ability of accurate and approximated SVM solvers and of traditional noise-reduction techniques with much faster training and testing times and better scalability performances.*

Keywords
Locality, Kernel Methods, Scalable Learning, Noise Reduction, Instance-Based Learning

Acknowledgments

I would like to thank my advisor, Prof. Enrico Blanzieri for his constant advice and infinite patience in supporting and guide my PhD work. I really enjoyed the discussions about new ideas and approaches in machine learning we have had together and I think they are invaluable factors of my formation.

I would like to thank Prof. Pádraig Cunningham for the opportunity he gave me to visit his research group in Dublin and Dr. Sarah Jane Delany for her support and suggestions. Pádraig and Sarah Jane are the promoters of the work I did in noise reduction, and the results reported in Chapter 7 of this thesis are the consequence of their advices.

I would like to thank the members of my PhD dissertation committee, Prof. Marco Gori and Prof. Chih-Jen Lin, for their very useful comments about my work.

I would like to dedicate my thesis to Cinzia and my parents.

I believe that learning has just started . . .

V. Vapnik - 2008

Contents

List of Tables

List of Figures

List of abbreviations

Notation	Meaning
$\mathcal{X}$	Training set
$\mathbf{x}$, $\mathbf{x}_i$, $\mathbf{p}$, $\mathbf{q}$, $\mathbf{t}$, $\mathbf{c}$	Examples in a Hilbert Space
N	Dataset size
$\mathcal{H}$	Hilbert Space
NN	Nearest Neighbor classifier
kNN	k Nearest Neighbor classifier
k	Neighbourhood size
SVM	Support Vector Machine classifier (soft-margin L1 norm).
kNNSVM	Local SVM classifier
K	Positive-definite kernel function
IBL	Instance-Based Learning
CBR	Case-Based Reasoning
LLA	Local Learning Algorithm
LKM	Local Kernel Machine
NR	Noise Reduction
RBF	Gaussian Radial Basis Function (kernel)
LIN	Linear (kernel)
POL	Polynomial (kernel)
HPOL	Homogenous polynomial (kernel)
IPOL	Inhomogenous polynomial (kernel)
SIG	Sigmoidal (kernel)
QL	Quasi-Local (kernel)
K^{lin}	Linear kernel
K^{rbf}	Gaussian Radial Basis Function kernel
K^{pol}	Polynomial kernel
K^{hpol}	Homogenous polynomial kernel
K^{ipol}	Inhomogenous polynomial kernel
K^{sig}	Sigmoidal kernel

PD	Positive Definite (kernel)
LibSVM	Software library for SVM
CVM	Core Vector Machine
BVM	Ball Vector Machines
LASVM	Online and active support vector machine method
USVM	non-convex SVM optimization method
CPSP	Cutting-Plane Subspace Pursuit
ENN	Edited nearest neighbor for noise reduction
RENN	Repeated edited nearest neighbor for noise reduction
BBNR	Blame Based noise reduction
AkNN	All kNN noise reduction
AkNNc	All kNN noise reduction with conservative majority rule
FaLKM-lib	The software library for LKM
FkNN	FaLKM-lib implementation of kNN
FkNNSVM	FaLKM-lib implementation of kNNSVM
FaLK-SVM	FaLKM-lib implementation of fast and scalable LKM
FaLK-SVM-train	FaLKM-lib module for the training of FaLK-SVM
FaLK-SVM-test	FaLKM-lib module for the testing of FaLK-SVM
FaLK-SVMc	FaLKM-lib implementation of a variant of FaLK-SVM (faster)
FaLK-SVMl	FaLKM-lib implementation of FaLK-SVM with local model selection
FkNNSVM-nr	FaLKM-lib implementation of kNNSVM for noise reduction
FaLKNR	FaLKM-lib implementation of NR with fast and scalable LKM

Chapter 1

Introduction

The development of intelligent systems able to learn from data in order to understand a phenomenon, to predict information associated with new data or to support decisions, is crucial in a wide range of fields such as bioinformatics, computer vision, artificial intelligence, medicine, and natural language processing. The area of computer science devoted to this task is called machine learning and it is deeply related with statistics and probability theory.

The present thesis deals with some of the fundamental aspects of machine learning: improving the accuracy and computational performances of learning systems.

In the next Section we introduce the context of local and global approaches for learning that is central throughout the present thesis. Then, we furnish some examples about the motivations for further enhance the classification capabilities and performances of learning systems (Section 1.2), and we roughly describe the solutions we propose (Section 1.3), before highlighting the main innovative aspects of our work (Section 1.4).

1.1 The Context

Locality, intended as the properties associated with examples that are close with respect to a metric function in a Hilbert space, plays a crucial role in a number of machine learning techniques. Probably, the approach that is more based on the idea of locality is the well-known nearest neighbour (NN) algorithm which simply learns the unknown information (the label in the case of supervised classification) of a query example relying on the closest example for which enough information (the label) is known, defining the proximity function according to a metric defined in the input space. The k-nearest neighbour algorithm (kNN) is the generalization of the NN algorithm for classification and it is based on the majority rule which assigns to the query example the most frequent label in the k-neighbourhood of the query example. It is interesting to notice that the majority rule and thus kNN is effective until the locality assumption is not

violated; as the value of k increases, in fact, the outcome of the majority rule approaches the label of the class with the higher cardinality (namely the mode) thus giving poor results. Locality thus permits to learn with very simple decision rules and it is the key factor for the success of machine learning fields like Instance-Based Learning (IBL) [4], Case-Based Reasoning (CBR) [1] and Lazy Learning (LL) [3]. Especially in absence of high-level information about the task at hand, locality is clearly a concept on which not only computational systems relies: when one of us has to decide what to do in an unknown scenario he probably tries to simulate the behaviour of people in the same situation or the behaviour he adopted in similar scenarios.

Maximal margin approaches for learning have been proposed in [24, 51] and gained a broad success: they are considered among the state-of-the-art methods especially for classification problems widely tackled with the Support Vector Machine (SVM) [51] classifier. The assumption of Statistical Learning Theory (SLT [193]) that the best linear separation between examples of different class is the hyperplane maximizing the margin between classes seems however to overcome the locality assumption: the maximal margin hyperplane is in fact influenced by the global distribution of examples near the separation between the classes and not by the local distributions of points in subregions of the space, and this may indicate that global strategies minimizing some loss functions, including some regularisation criteria, are better than local ones. Moreover, the theoretical work regarding the Vapnik-Chervonenkis (VC) dimension [195] states that the simpler the class of separating decision functions, the lower the bound on the generalization error; since introducing locality in the decision function causes the VC dimension to have an infinite value, it seems better to avoid locality in this context.

It is thus evident the dualism between effective, intuitive and practical approaches based on locality and sound, well theoretically-founded and bounded-error approaches based on global optimization procedures. This dualism is the base for the SVM using the Gaussian radial basis function kernel (RBF) which is a non-linear local kernel projecting implicitly the examples of the input space in a transformed feature-space with infinite dimensionality. Despite the fact that the RBF kernel violates the VC theory [195], SVM with RBF kernel showed very high accuracy performances and it is considered the best general-purpose kernel except for very high-dimensional data. So the combination of a global learning system (linear SVM) with a non-linear function projecting the examples in a space based on locality information (the RBF kernel) proved to be effective and can be motivated by a very recent results presented in [210] saying that, roughly speaking, "consistency implies local behaviour". Apart from local kernels, little research has been performed for integrating local and global characteristics in kernel-based methods with the exception of Local SVM [19, 212, 20, 164] which is a representative of the more general theory of Local Learning Algorithms (LLA) [27, 194] that, however, suffer from computational problems allowing the application of the approach on small datasets only. From the computational viewpoint, locality is considered to be an issue to avoid for fast and scalable learning because it does not allow to approximate and smooth the decision functions.

Locality is inherently not robust to noise. This is the reason for the accuracy advantages of kNN over NN for noisy problems and for the success of pre-processing procedure, called noise reduction (NR) techniques, trying to remove noisy examples from the training set. Surprisingly enough, almost all the NR techniques reported in literature are based on a strict notion of

locality (NN or kNN with small k) probably because they make use of the majority rule or some variants of it. Setting a different trade-off between local and global approaches for NR has been not investigated yet.

1.2 The Problem

The problem we tackle regards the increasing of classification accuracy capabilities and computational performances of learning systems. The motivations for research in this directions are multiple, and a couple of examples can highlight them.

Consider the machine learning task of detecting spam email messages: although the accuracy of such systems is considered rather high, we almost daily deal in our email clients with false positives. Even worse is the case of false negatives because we are not typically aware of those authenticate mails that are considered spam by the filter and automatically deleted; the potential cost associated with false negatives in this case (e.g. missing good job offers or other important communications) is a clear reason to continue improving the accuracy of machine learning techniques. Other examples can be taken from bioinformatics in which machine learning predictions are typically validated with biological experiments and so the more accurate the prediction, the less time and smaller costs are spent for biological experiments.

Computational performance is another hot topic in machine learning. Real-time applications like automatic video surveillance, network intrusion detection or intelligent robot actions require very fast learning algorithms that have to produce accurate predictions within seconds or even milliseconds. When the available hardware power is limited (e.g. embedded systems) the computational efficiency is even more crucial. But also in situations in which the most modern hardware is available, scalability of learning systems is a key factor for the applicability of machine learning techniques. There are, for example, learning tasks in systems biology that must deal with a small fraction of the available genomic data simply because using all the available information would take years of computation to complete the given task.

1.3 The Solution

The strategies we developed for improving accuracy and computational performances have in common the fact that we operate on the trade-off between local and global approaches to machine learning (introduced above) within the framework of kernel methods and IBL. Our theoretical and empirical analysis of the Local SVM approach [164] that we expand in the first part of the thesis, motivates the effectiveness of the approach and the feasibility of the new research directions we detail in the following. The empirical analysis shows, in fact, that Local SVM statistically significantly overcomes SVM at least for non local kernels, and the theoretical bound we derive starting from LLA confirms the possibility of improving the SVM generalization ability.

With our novel family of kernels, called Quasi-Local (QL) kernels [167], we mix the possibly global properties of any existing input kernel with a kernel which is local in the feature-space of the input one by means of a set of operators. This can be used to add another and different level

of locality to a local kernel (allowing the level of locality to varies locally) or to balance the long-range extrapolation ability of global kernels. The operators we use to produce QL kernels accept two parameters that regulate the width of the exponential influence of points in the locality-dependent component and the balancing between the feature-space local component and the input kernel. We addressed the choice of these parameters with a data-dependent strategy. Experiments carried on with SVM applying the operators on traditional kernel functions on a total of 43 datasets with different characteristics and application domains, achieve very good results supported by statistical significance. It is important to underline that the QL kernels can be used in any kind of kernel method.

With our Local Kernel Machines (LKMs) for classification, described in [165, 163] and called FaLK-SVM, we show that locality can be decisive also for developing fast and scalable kernel methods. In our approach a set of local SVMs are trained on redundant neighbourhoods in the training, and at testing time we select the most appropriate model for each query point. Under the assumption, and consistently with [210], that the decision function estimated using only the neighbourhood of a query point and the global decision function are very similar in the subregion of the query point, LKM divides the separation function in solutions of local optimization problems that can be handled very efficiently. The introduction of a fast local model selection further speedups the learning process. Learning complexity bounds for LKMs are derived, and the empirical evaluation of the approach showed that it is much faster and more accurate than the accurate and approximated state-of-the-art SVM solvers at least for non high-dimensional datasets.

In order to enhance the accuracies and computational performances of IBL approaches, we developed a family of noise reduction techniques based on the local application of the maximal margin principle providing a more robust alternative to the majority rule on which almost all the existing noise reduction techniques are based. The algorithms, called FkNNSVM-nr [169] and FaLKNR [168], are developed within the framework of LKM. Roughly speaking, FkNNSVM-nr trains for each training example an SVM on its neighbourhood and if the SVM classification for the central example disagrees with its actual class there is evidence in favour of removing it from the training set. The empirical evaluation showed that FkNNSVM-nr overcomes state-of-the-art noise reduction techniques in particular for real datasets and for artificial datasets perturbed by Gaussian noise and in presence of uneven class densities. FaLKNR is a modification of FkNNSVM-nr that introduces a set of optimizations in order to scale the approach to very large datasets and the empirical results showed that the accuracies and computational performances are much better than traditional noise reduction techniques.

The algorithms we developed in the framework of LKM are FkNN, FkNNSVM, FaLK-SVM, FkNNSVM-nr and FaLKNR and are contained in the Fast Local Kernel Machine Library (FaLKM-lib, see Appendix A) [163], freely available with source code for research and educational purposes at `http://disi.unitn.it/~segata/FaLKM-lib`.

1.4 Innovative Aspects

The empirical and theoretical analysis of the Local SVM approach we carried on at the beginning of this thesis, completes the approach with an extensive evaluation which has not been performed before. The main outcome of the experiments is that Local SVM is very often significantly more accurate than SVM and this should encourage the application of Local SVM to specific problems in which the accuracy performances are crucial. The additional analysis of the behaviour of Local SVM with RBF kernel and adaptive kernel width highlights that there are highly non-linear problems in which Local SVM substantially overcomes SVM also using a local kernel function.

The combination of the input space information of a kernel with its feature-space information is a research direction that has not been previously investigated. In this context we theoretically define the class of QL kernels formally proving some properties (discussing if they are positive-definite (PD), universal, what happens if the input kernel is a local kernel, how they can be constructed, how the parameters can be estimated, the relation of SVM with QL kernels and SVM with RBF and variable kernel width) and showing intuitively their behaviour. The extensive empirical evaluation showed that SVM with traditional kernels are less accurate than SVM with QL kernels based on the same input kernels and this is supported by solid statistical significance. The fact that SVM with the tested input kernels (linear, polynomial, RBF and sigmoidal) are considered to be the state-of-the-art for general classification problems can give an idea about the potential impact of QL kernels in machine learning. We also provide automatic strategies to select the parameters of QL kernels and show that they are applicable to reasonably large datasets. Although we focus here on the innovative aspects of QL kernels concerning the classification tasks with SVM, they can be applied on all kernel methods because the operators act on the kernel functions only.

LKMs and in particular the FaLK-SVM classifier show that locality can be a key factor for developing fast and scalable kernel methods. This is somehow in contrast with existing approaches for speeding-up kernel methods that are based on approximations of the decision function and consequently on a trade-off between locality and scalability. With FaLK-SVM we are not trying to lower the number of support vectors or basis functions in order to maintain the optimization procedure into a reasonable level of complexity, but, instead, we are constructing and training models on local subsets of examples. Mechanisms for allowing enough redundancy in covering all the training set with local models are introduced as well as techniques for speeding-up the neighbourhoods retrieval. Excellent empirical results have been obtained on very large datasets (up to 3 million examples) in which traditional SVM software like LibSVM is not applicable. In particular FaLK-SVM showed to be orders of magnitude faster than LibSVM and other approximated SVM solvers allowing at the same time higher accuracy capabilities with respect to LibSVM which is in turn more accurate than any approximated solver. Although the case of high-dimensional datasets is not experimentally considered and can be problematic due to the well-known "curse of dimensionality", for non-high dimensional datasets our approach substantially overcome state-of-the-art classifiers both in terms of accuracy and computational performances. Novel strategies for model selection and formal bounds for generalization ability

and scaling performances complete the approach for the practical application and theoretical comparison.

Also the LKMs developed for preprocessing training sets in order to enhance the accuracy of IBL algorithms showed improved performances over state-of-the-art techniques. The basic idea consists in using local maximal margin models in order to predict the class of the central examples (or a set of central examples if scalability to large datasets is needed) removing the central examples if their predictions are not in accordance (with tunable probability thresholds) to the assigned class. Two are in this case the innovations with respect to existent NR techniques. The first is that we introduce a much more powerful principle than the majority rule used by traditional NR algorithms to locally detect noisy examples, the second is that we are able to tune the level of locality permitting a trade-off between local and global behaviour that is more effective than a complete local behaviour when noisy features are present (e.g. Gaussian feature noise). The extensive experimental results show statistically significant improvements of our novel approach on the accuracies induced to NN and kNN using the preprocessed training sets with respect to state-of-the-art NR techniques and, from the computational viewpoint, FaLKNR is much more efficient than existent approaches. It is important to underline that, although not discussed in depth here, NR can give benefits also to bioinformatics and medical tasks and the scalability of FaLKNR permits the application of our approach to large amounts of data possibly reaching the genomic scale. In addition, removing the noisy examples can help supervised learning to work with smaller models thus improving the computational performances.

It is important to underline that the approaches we propose in this work have a much larger application area than the supervised classification tasks on which we mainly focus. QL kernels can be applied to every kernel-based techniques without any modification, whereas a wide range of analyses can be performed locally using the computationally efficient framework of Local Kernel Machines. More generally, in fact, LKM can be though as a way to switch local learning techniques (LLA, IBL, CBR,...) from the inefficient *lazy* learning setting to the *eager* learning setting with its advantage especially in terms of computational prediction performances.

1.5 Structure of the Thesis

The present work is organized as follows. Chapter 2 reviews the state-of-the-art regarding locality in machine learning (IBL, CBR, LLA, locality in kernel methods), scalable maximal margin classifiers and algorithms for noise reduction. Chapter 3 defines the formal tools necessary for the following chapters, including kNN, SVM, kernel functions and local kernels, Local SVM and structures for fast neighbourhood operations. Chapter 4 analyses the original Local SVM approach and its algorithm called FkNNSVM with new theoretical tools and an extensive empirical evaluation. In Chapter 5 we introduce, analyse, and empirically test the operators for obtaining Quasi-Local kernels from any kernel function. Chapter 6 is devoted to the theory of Local Kernel Machines (LKM) with particular focus on fast and scalable classification with the FaLK-SVM classifier which is theoretically and empirically analysed from various viewpoints. In Chapter 7 we detail the specialization of LKM for noise reduction with FkNNSVM-nr and its variant for large datasets called FaLKNR. Finally we draw some conclusions (Chapter 8) and

discuss future research directions enabled by this thesis. Appendix A presents the developed FaLKM-lib software library detailing its modules and a concise manual.

Chapter 2

State of the Art

This chapter reviews the main research areas connected with our work. The first section (Section 2.1) focuses on the concept of locality and on how it has been exploited in machine learning and kernel methods. The following two sections review the state-of-the-art regarding the approaches to scale-up kernel methods for large datasets (Section 2.2), and to remove noise from data especially as a preprocessing step for Instance-Based Learning (Section 2.3); these two sections furnish an overview of the competitors of our fast and scalable local approach for kernel methods (Chapter 6) and our techniques for noise reduction with local kernel machines (Chapter 7).

2.1 Locality in Machine Learning

Locality is central to research fields such as Instance-Based Learning and Case-Based Reasoning (Section 2.1.1) and Local Learning Algorithms (Section 2.1.2) because they all rely on the idea of basing the generalization step on the neighbourhood of the testing point. For kernel methods, instead, the approaches for including locality in the learning process are mainly based on developing kernel functions that take into consideration the distances between examples (Section 2.1.3).

2.1.1 Instance-Based Learning and Case-Based Reasoning

Instance-Based Learning (IBL) [4] denotes a class of methods based on a local approximation of the target discriminant function around the testing instances. IBL methods do not construct a general, explicit and global estimation of the target function during training, but they postpone all the computation to the time an example has to be effectively classified. The training phase of IBL methods simply consists in storing the training examples in order to use them when testing examples are available. Advantages of IBL are the possibility of learning with simple

local decision functions (like the majority rule of k-Nearest Neighbors), the potentially use of all training examples in the generalization process, the simplicity of implementing algorithms, the generalization ability for very complex and highly non-linear and noise-free problems. On the other hand, various limitation can be identified: IBL methods are usually inefficient at testing time, not robust to noisy examples and to noisy or irrelevant features, the distance metric used to retrieve the nearest examples is often very crucial, and they usually need a large training set to achieve satisfactory accuracy performances.

The most popular IBL methods are the k-Nearest Neighbor classifier (kNN, see Chapter 3.1) and its variants, the Locally Weighted Regression approach [47, 48] and the Radial Basis Function networks method [31].

Case-Based Reasoning (CBR) is an instance-based approach in which the examples (called cases) are entities that can be much more complex than points in a n dimensional Euclidean space. More generally, CBR can be seen as an Artificial Intelligence procedure to solve new problems based on the solution of similar past problems. CBR has been characterised by Aamodt and Plaza [1] as a four-step process: (i) *retrieve*, in which relevant (similar) examples for the current problem are identified, (ii) *reuse*, in which the previous solutions are mapped to the target problem, (iii) *revise*, in which the new solution is tested and, if necessary, modified and (iv) *retain*, in which the new solution is stored in the case base to be available for future similar problems.

2.1.2 Local Learning Algorithms

Local Learning Algorithms (LLAs) are a class of learning approaches introduced by Vapnik and Bottou [27, 194] that can be seen as representatives of the IBL approach. Instead of estimating a decision function which is optimal (with respect to some criteria) for all possible unseen testing examples, the idea underlying LLAs consists in estimating the optimal decision function for each single testing point. The value of the function is estimated in a small sub-region of the input space around the query point. For a local learning algorithm, the points in the proximity of the query point have an higher influence in the training of the local model. The approach is particularly effective for not uniformly distributed datasets, because the characteristics of the learning process can be locally adjusted. A proper choice of the locality parameter can in fact reduce the generalization error with respect to a global classifier as formalised by the Local Risk Minimization principle [194, 193]. Notice that there are various ways of specifying the degree of locality for LLAs as discussed for instance by Atkeson et al. [8]. Examples of LLAs are the well known k-Nearest Neighbours (kNN) classifier, the Radial Basis Function networks [31], and the Local SVM classifier [19, 212] (see also Chapter 3.4).

Despite their theoretical and practical appeal, LLAs seem not to have been studied in depth in the last few years. This is probably due to the fact that LLAs, as formulated by Bottou and Vapnik [27], falls in the class of *lazy learning* (or *memory-based learning*) that have great overhead on the testing phase, as opposed to *eager learning* in which the function estimation is performed during training improving the computational performances of the testing phase.

2.1.3 Locality in Kernel Methods

In kernel methods, locality has been introduced with two meanings: i) as local relationship between the features, i.e. local feature dependence, adding prior information reflecting it, ii) as distance proximity between points, i.e. local points dependence, enhancing the kernel values for points that are close to each other and/or penalizing the points that are far from each other. The first meaning has been exploited by locality-improved kernels, the second by local kernels, kernels based on distance measures and *local SVM* . Both approaches are described below. We also review some less general approaches for including locality in kernel methods and how locality can be exploited for performing dimensionality reduction.

Locality-improved kernels

Locality-improved kernels were introduced in Schölkopf and Smola [157] and they take into account prior knowledge of the local structure in data such as local correlation between pixels in images. The way the prior information is integrated into the kernel depends on the specific task but, in general, the kernel increases similarity and correlation of selected features that are considered locally related. Locality-improved kernels were successfully applied on image processing [160] and on bioinformatic tasks [214, 73].

Local kernels

A kernel is local if when the distance between a test point and a training point tends to infinity, the value of the kernel is constant and independent of the test point [13, 171]; if this condition is not respected the kernel is said to be *global*. A popular local kernel is the Gaussian radial basis function (RBF) kernel that tends to zero for points whose distance is high with respect to a width parameter that regulates the degree of locality. On the other hand, distant points influence the value of global kernels (e.g. linear, polynomial and sigmoidal kernels). Local kernels and in particular the RBF kernel show very good classification capability but they can suffer from the curse of dimensionality problem [14] and they can fail with datasets that require non-linear long-range extrapolation. In this case, even if the tuning of the width parameter allows for the contribution of distant points, global kernel reflecting a particular conformation of the separating surface are generally preferred and permits better accuracies. An attempt to mix the good characteristics of local and global kernels is reported in [171] where RBF and polynomial kernels are considered for SVM regression.

Kernels based on distance measures

Local kernels are often kernels that are based on distance functions. The use of distance functions for defining kernels can be important to capture the local charactheristics of the data and introduce it in the learning process.

An interesting kernel family is the class of stationary kernels [77] which is composed by kernels that are translational invariant. Additionally, if a stationary kernel depends only on the Euclidean distance between the examples, namely on the norm of the vector between them, we

have an isotropic stationary kernel. The RBF kernel, belongs to this subclass. Other isotropic stationary PD kernels are the exponential kernel, the rational kernel, the Beta kernel, the uniform kernel, the triangular kernel, the multiquadratic kernel, the inverse multiquadratic kernel, the thin plate splines kernel and the KMOD kernel. Another distance measure that can be used to design kernels, derives from the spectral angle mapper (SAM) which is a scale-invariant and nonadditive distance metric used in remote sensing problems mainly for measuring the spectral difference between examples since it is robust to differences in spectral energy [125, 69]. It consists in determining the angle between two vectors and can be used to define SAM based kernels. Notice that, in principle, SAM can be embedded in all isotropic stationary kernels. The formulation of the listed kernel can be found in the next chapter.

Local SVM

Local SVM is a kernel-based maximal margin LLA and was independently proposed by Blanzieri and Melgani [19, 20] and by Zhang et al. [212] and applied respectively to remote sensing and visual recognition tasks. Other successful applications of the approach are detailed in [164] for general real datasets and in [17] for spam filtering. The main idea of local SVM is to build at prediction time a example-specific maximal marginal hyperplane based on the set of k-neighbours. In [19] it is also proved that the local SVM has chance to have a better bound on generalization with respect to SVM. However, local SVM suffers from the high computational cost of the testing phase that comprises for each example the selection of the k nearest neighbours and the computation of the maximal separating hyperplane, and from the problem of tuning the k parameter. The algorithm for Local SVM is called kNNSVM and will be presented in Section 3.4; and implementation of kNNSVM, called FkNNSVM introducing some strategies to speed-up the approach is available in the FaLKM-lib [163].

Other ways to include locality in classification

Locality in the learning process can be included using strategies based on the work of Amari and Wu [5] that modify the Riemannian geometry induced by the kernel in the input space introducing a quasi-conformal transformation on the kernel metric with a positive scalar function. Particular choices of such scalar functions permitted to Wu and Amari [206] to increase the margin of the separating hyperplane through a two steps SVM training under the empirical assumption that the support vectors (detected during the first step with a preliminary SVM training) are located mainly in proximity of the hyperplane. In the bioinformatics field, a different particular choice of the positive scalar function for the quasi-conformal transformation permitted to reach high accuracy in classification of tissue examples from their microarray gene expression levels through a kNN based scheme [207]. In [142], instead, the positive scalar function is chosen in order to contract the spatial resolution around relevant examples and the opposite for irrelevant examples. In this way a new space is constructed in which the distance between related examples is increased while the distance between non-related examples is decreased. In the context of image retrieval this space is used to estimate the distance between query and database images. Recently, another way of modifying kernel functions based on training set

data has been proposed by Min et al. [130]. The idea is to introduce training example label information in the kernel function decreasing the feature-space distance between examples with the same label and increasing it for examples with different labels. In this case the problem is the application of the kernel in the prediction phase since, for definition, the labels are unknown. The authors estimate the kernel used for testing points with singular value decomposition and linear mapping techniques, achieving good accuracy in detecting protein remote homology. An extreme and interesting version of the idea of modifying the kernel function depending on the training data is described in [126] where the authors developed, in the framework of Tikhonov regularisation theory, a method able to automatically and univocally determine the kernel. Locality has been also used as the key factor to combine multiple kernel functions using a non-stationary (i.e. non- global) fashion as detailed, for example, by Lewis et al. [110].

Locality for dimensionality reduction with kernel methods

Apart for classification, there are many kernel-based subspace analysis techniques like dimensionality reduction, manifold learning and feature selection techniques which are gaining importance in the last few years and are intrinsically related with the concept of locality. Some of the most popular techniques in this area are Locally Linear Embedding (LLE) by Roweis and Saul [152] which has a kernel-based version [58] and it is equivalent to kernel principal component analysis (kernel PCA) by Schölkopf et al. [161] for a particular kernel choice and kernel Local Discriminant Embedding (kernel LDE) by Chen et al. [40]. Other non naturally local techniques, have their local counterparts: Fisher Discriminative Analysis (FDA) [70] and its kernel-based version [129] has a local version in Local Fisher Discriminative Analysis (LFDA) [181], whereas Generalized Discriminant Analysis (GDA) [11] is the base for locally linear discriminant analysis (LLDA) [183]. Global techniques such as ISOMAP [186, 43] can adopt their kernel version using a local kernel to include locality. Other approaches are based on developing and learning kernels subject to local constraints, as for example in [83]. An interesting discussion on local and global approaches for non-linear dimensionality reduction fall beyond the kernel methods field and it is addressed by De Silva and Tenenbaum [57].

2.2 Fast and Scalable Learning with Kernels

In the last few years, the need for fast and scalable kernel-based classifiers led to the development of several methods, although more work seems to have been done for linear classifiers as discussed in 2.2.1. The state-of-the-art for large-scale maximal margin learning that can use non-linear kernel functions is represented by the approaches introduced in Section 2.2.2.

2.2.1 Fast Approaches for Linear SVM

Recently a lot of work has been performed in order to develop very fast and scalable solvers applicable to *linear* SVM only. Keerthi and DeCoste [97] modified the Finite Newton method of Mangasarian[122] introducing robust conjugate gradient techniques and other heuristics, Joachims [92] developed an alternative formulation of the SVM optimization problem exploiting

a different form of sparsity and Lin et al.[112] uses logistic regression with Trust Region Newton Methods. Variants of coordinate descent methods for linear SVM are developed by Chang et al. [38] in the primal and by Hsieh et al. [86] in the dual. A different gradient approach was developed by Smola et al. [172]. Other approaches are based on Stochastic Gradient Descent (SGD) like those developed by Shalev- Shwartz et al. [170] and by Bordes et al. [22] which work in the primal, whereas Collins et al. [49] apply SGD in the dual. Although SGD methods can be theoretically used for non-linear SVM the performances are analysed for the linear case only. LIBLINEAR by Fan et al. [67] is a fast software package implementing some of the cited works. The common idea of all proposed methods is that the advantage of having a method that uses a huge number of training points overcomes the disadvantage of approximating the decision function with a linear model. This is effective, as explicitly noticed in almost all the cited works, when the dimensionality is very large and thus the problem is very sparse. This is, for example, the typical situation of text document classification. However, when the needed decision function is highly non-linear and the intrinsic dimensionality of the space is relatively small, the linear SVM approach cannot compete with SVM using non-linear kernels in terms of generalization accuracies. Apart from the generalization ability also the computational performances can be compromised in this cases, because the algorithm cannot find a good decision function and so convergence problems can occur.

2.2.2 Fast Approaches for Non-Linear SVM

One of the first large-scale maximal margin learning that can use non-linear kernel functions is represented by Core Vector Machines (CVM) by Tsang et al. [190] in which, reformulating the SVM approach as a minimum enclosing ball problem, the authors proved that it is possible to obtain approximated optimal solution in competitive training times by using the core sets. Good results have been achieved using non-linear kernels although it has been pointed out [117] that the choice of the stopping criteria is crucial in the trade-off between computational efficiency and generalization accuracy. Ball Vector Machines (BVM) by Tsang et al. [189] are a modification of CVM in which the minimality of the enclosing balls is not required, because the radius of the ball is fixed. The resulting classifier improves the computational performances. Another approach based on an online setting of the SVM optimization problem, called LASVM, has been proposed by Bordes et al. [23, 21], and it is an algorithm that converges to the SVM solution. It has been shown that competitive accuracies can be achieved also after a single pass over the training set. The approach can be seen as a SVM solver that includes a support vector removal step. In addition, several strategies for active training-points selection can further improve computational and generalization performances. Formulating the optimization problem in the primal, Keerthi et al. [96] proposed a method, called SpSVM, that bounds the number of basis functions considered and consequently the computational complexity. Increasing the cardinality of the basis function set allows the method to converge to the SVM solution. A greedy strategy guides the choice of the basis functions to be included in the working set. Collobert et al. [50] showed that softening the convex setting of maximal margin classifiers using a non-convex loss function can bring computational advantage over the corresponding standard convex problem.

The non-convex problem is solved using the *concave-convex procedure* [209], obtaining the USVM method. Recently the Cutting-Plane Subspace Pursuit (CPSP) by Joachims and Yu [94] based on cutting-plane training [93] has been proposed; it permits to learn maximal-margin decision functions in the feature space using arbitrary basis vectors instead of the support vectors only. This can results in sparser solutions increasing the testing and training computational performances especially for high-dimensional datasets. Although not always considered a method for large-scale learning, LibSVM by Chang and Lin [36] demonstrated to be competitive with approximated approaches from the computational viewpoint. LibSVM is a SVM solver implementing a SMO-type decomposition method proposed by [68] integrating it with caching and shrinking [91].

2.3 Noise Reduction for Instance-Based Learning

Noise reduction for IBL and CBR is included in the more general framework of editing techniques that can have many different objectives as discussed, for example, by Wilson and Martinez [204] and by Brighton and Mellish [29]. According to them, editing techniques can be categorised as competence preservation or competence enhancement techniques. Competence preservation techniques aim to reduce the size of the training set as much as possible without significantly affecting the generalisation accuracy thus achieving a reduction in the storage requirements and increasing the speed of execution. The main goal of competence enhancement techniques is to increase the generalisation accuracy primarily by removing noisy or corrupt training examples.

Obviously, some strategies aim to tackle both objectives at the same time and for this reason are called hybrid techniques [29]. Editing strategies normally operate in one of two ways; *incremental* which involves adding selected examples from the training set to an initially empty edited set, and *decremental* which involves contracting the training set by removing selected examples.

2.3.1 Competence Preservation Methods

The objective of competence preservation consists in reducing the cardinality of the training as much as possible without however affecting the generalisation ability of classifiers trained using the edited training set. Competence preservation was studied almost simultaneously with the introduction of nearest neighbour classifiers mostly because of the limited power of early computational systems.

The first contribution was Hart's *Condensed Nearest Neighbour Rule* (CNN) [82] which incrementally populates the edited set with those training examples that are misclassified by the edited set. Improvements over the CNN rule, primarily developed to overcome its limitations in the presence of noise, are the *Reduced Nearest Neighbour Rule* (RNN) by Gates [76] and the *Selective Nearest Neighbour Rule* (SNN) by Ritter et al. [150]. RNN is a decremental technique which removes an example from the edited set where its removal does not cause any other training example to be misclassified while SNN imposes the rule that every training example must be closer to an example of the same class in the edited set than to any training example of another

class.

CNN (using 1NN) is included as a special case in the *Generalized Condensed Nearest Neighbour Rule* (GCNN) [44] which relaxes the criterion for correct classification by a factor of the minimum distance between heterogeneous examples in the training set. Another variation on the CNN rule for text categorisation is reported by Hao et al. [81] which orders the training examples for rule consideration based on a metric calculated from the document's textual feature weights. Recently, the novel *Fast Condensed Nearest Neighbour Rule* (FCNN) has been introduced by Angiulli [6]. FCNN offers advantages over other CNN variations as it is an order-independent algorithm, it exploits the triangle inequality to reduce computational effort and it is scalable on large multidimensional datasets.

A different approach based on prototypes is proposed by Chang [37] in which the nearest two training examples belonging to the same class are merged using a weighting policy into a new example. A limitation of this approach is that the new training examples are synthetically constructed eliminating the original examples and this prohibits, for example, case-based explanation.

More recent approaches to case-base editing in the CBR paradigm use the competence properties of the training examples or cases to determine which ones to include in the edited set. Measuring and using case competence to guide case-base maintenance was first introduced by Smyth and Keane [173] who introduced two important competence properties, the *reachability* and *coverage* sets for a case in a case-base. The *reachability set* of a case t, which is the set of all cases that can correctly classify t and the *coverage set* which is the set of all examples that t can correctly classify. An example of using case competence to guide editing is the *Footprint Deletion* policy by [173] which is based on the notion of a competence footprint, a subset of training examples providing the same competence as the entire set. The same group also proposes a family of competence-guided methods [124] based on different combinations of four features; an ordering policy, an addition rule, a deletion rule and a competence update policy. Brighton and Mellish [29] also used the competence properties of cases in their *Iterative Case Filtering* (ICF) algorithm which is a decremental algorithm that contracts the training set by removing those cases c, where the number of other cases that can correctly classify c is higher that the number of cases that c can correctly classify. Most competence-based editing techniques can include a preprocessing step for noise removal thus becoming hybrid methods.

Salamó and Golobardes [154] propose techniques based on the theory of Rough Sets [140] which reduce the case-base by analysing the lower and upper approximations to sets of training examples that are indistinguishable with regard to a specific subset of features. Successive refinements from the same authors incorporate their rough sets measures into Smyth and Keane[173]'s competence model and then apply various policies for removing cases [155, 153]. Similar approaches have been proposed by Caballero et al. [32] who creates the edited training data from the lower and upper set approximations and Cao et al. [34] who couples rough sets theory with fuzzy decision tree induction.

Mitra et al[131] present an incremental density-based approach to editing large datasets which uses a nearest neighbour density estimate of the underlying training data to select which examples to keep. The density based approach is further developed by Huang and Chow [89]

introducing the concept of entropy while a successful application of density-based reduction for text categorization is detailed by Li and Hu [111].

2.3.2 Competence Enhancement Methods

The objective of competence enhancement methods is to remove noisy, mislabeled and borderline examples that are likely to cause misclassification thus allowing kNN classifiers to build smoother decision surfaces. In its pure form, competence enhancement will retain all the correctly labelled examples far from the decision boundary thus precluding significant storage reduction. Competence enhancement techniques start with Wilson's *Edited Nearest Neighbour* algorithm (ENN) [205]. It is a decremental strategy that simply removes from the training set those examples that do not agree with the majority of their k nearest neighbours.

Tomek [188] proposed two improvements to ENN; *Repeated Edited Nearest Neighbour* (RENN) and *All-kNN* (AkNN). Both RENN and AkNN make multiple passes over the training set repeating ENN. RENN just repeats the ENN algorithm until no further eliminations can be made from the edited set while AkNN repeats ENN for each example using incrementing values of k each time and removing the example if its label is not the predominant one at least for one value of k. It is worth noting that for $k = 1$, ENN and AkNN are equivalent and for $k > 1$ AkNN is more aggressive than ENN.

A slightly different approach is introduced by Koplowitz and Brown [100] which considers the relabeling of some examples instead of their removal. This idea is expanded on by Jiang and Zhou [90] who use an ensemble of neural networks to determine the label for the examples that are to be relabeled. Another modification of ENN and RENN proposed by Sánchez et al. [156] entails substituting the k nearest neighbours with the k nearest centroid neighbours (k-NCN) where the neighbourhood of an example is defined not only based on distances from an example but also on the symmetrical distribution of examples around it.

The detecting of mislabeled examples in high-dimensional spaces with small training set sizes (the typical characteristics of microarray data in bioinformatics) is addressed by Malossini et al. [120] based on a leave-one-out perturbation matrix and a measure of the stability of the label of an example with respect to label changes of other examples.

In the context of editing training data for spam filtering systems, Delany and Cunningham [60] advocate putting the emphasis on examples that cause misclassifications rather than the examples that are themselves misclassified. The method which is called *Blame Based Noise Reduction* (BBNR) enhances the competence properties of *coverage* and *reachability* with the new concept of a *liability set*. Roughly speaking this set, which is defined for each training example t in a leave-one-out classification of the training set, contains any other misclassified training examples (of a different class than t) where t contributed to the misclassification by being returned as one of the k nearest neighbours.

2.3.3 Hybrid Methods

Instance Based (IB) *Learning Algorithms* (IBn), presented by Aha at al. [4], can be considered the first hybrid approaches to editing. IB2 is an online learning method, similar to CNN, that

works by adding to an initially empty set those examples that are not correctly classified by the edited set. Within this setting a newly available example that is not added to the edited set does not need to be stored. On the other hand, since noisy and mislabeled examples are very likely to be misclassified, they are almost always maintained in the edited set. In order to overcome this weakness, IB3 adds a "wait and see" policy which records how well examples are classified and only keeps those that classify correctly to a statistically significant degree.

Some variations of the IBn algorithms are *Typical Instance Based Learning* algorithm (TIBL) by Zhang [213] which tries to keep examples near the centre of clusters rather than on decision boundaries, *Model Class Selection* techniques (MCS) [30] which checks the class-consistency of an example with respect to the examples it classifies, and methods based on *Encoding Length Heuristic* (ELH) [33].

Another hybrid method proposed by Lowe [119] is based on *Variable-Kernel Similarity Metric* (VSM) *Learning*. In this case an example is removed if its neighbourhood is classified by the VSM classifier as belonging to the same class. In this way examples internal to clusters are removed but as there is no requirement that the removed example has the same class as its neighbours, this technique also removes "noisy" examples.

Wilson [203] introduced a family of *Reduction Techniques* (RT1, RT2 and RT3) which were then enhanced in [204] under the name of *Decremental Reduction Optimization Procedures* (DROP1-DROP5) and *Decremental Encoding Length* (DEL). DROP1 is very similar to RNN with the only difference that the misclassifications are checked in the edited set instead of the training set. DROP2 fixes the order of presentation of examples as those furthest from their nearest unlike neighbour (i.e. nearest example of a different class) to remove examples furthest from the class borders first. DROP2 also uses the original training set when checking for misclassification to avoid some problems that can occur with DROP1 such as removing entire clusters. In order to make DROP2 more robust to noise, DROP3 introduces an explicit noise reduction preprocessing stage with a rule very similar to ENN. In DROP4 this noise reduction phase is made more conservative by only removing an example if it is misclassified by its neighbourhood and if its removal does not hurt the classification of other examples. DROP5 is a modification of DROP2 using the opposite ordering function for the presentation of examples which acts as a noise reduction pass and finally DEL is a version of DROP3 using ELH as the deletion rule.

Recently a new case-base mining framework has been introduced by Pan et al. [137]. The framework includes a case-base mining algorithm which is based on a theoretical foundation. The *Kernel-based Greedy Case-base Mining* (KGCM) algorithm first maps the examples to a new feature-space through a kernel transformation, performs a Fisher Discriminant Analysis (FDA) based feature-extraction method to help remove noise and extract the highly predictive features and finally considers the diversity of the selected cases in terms of the coverage of future problems.

2.3.4 Identifying State-of-the-Art for Noise Reduction

The main editing techniques developed before 2000 have been extensively evaluated by Wilson and Martinez [204]. The overall result of their analysis is that DROP3 has the best mix of

generalisation accuracy and storage reduction. However, looking at generalisation capability only, they conclude that their DROP3 method has somewhat lower accuracy that the group of methods including ENN, RENN and AkNN. In particular, among these last three methods, AkNN has "the highest accuracy and lowest storage requirements in the presence of noise" [204]. The comparisons of ICF with DROP3 done by Brighton and Mellish [29] highlights that they have similar performance but, considering the accuracy results only, it is clear that ENN outperforms both in the majority of the datasets.

k-NCN seems to be more accurate than AkNN and ENN as shown by Sánchez et al. [156], but the analysis is performed on five datasets only and does not include an assessment of statistical significance. Moreover k-NCN substitutes real examples with synthetic ones preventing CBR explanation. Without considering the competence preserving methods as our objective is competence enhancement, the remaining approaches (including the neural network ensemble approach presented by Jiang and Zhou [90] and KGCM [137]) do not provide any comparison with ENN, RENN or AkNN and the reproduction of these techniques is non trivial as they are embedded in complex frameworks. The approach proposed by Malossini et al. [120] is conceived for very high dimensional datasets with very few examples and thus it is not suitable for general real datasets.

Taking this into consideration, we can conclude that AkNN, despite its simplicity, still represents the state-of-the-art for competence enhancement, and that RENN and ENN can give comparable performances.

Chapter 3

Preliminaries

In this chapter, we introduce the theory and the tools our novel techniques are based on, or are deeply related to, and that are necessary to understand the following chapters.

In particular, we briefly introduce the k-Nearest Neighbors (kNN) classifier (Section 3.1), the Support Vector Machine (SVM) classifier (Section 3.2), some aspects of kernel function for kernel machines (Section 3.3), the Local Support Vector Machine classifier (Section 3.4) and the Cover Trees (CT) data structure to efficiently handling nearest neighbour operations (Section 3.5).

In this chapter and throughout all the thesis, we consider a classification problem with examples $(\mathbf{x}_i, y_i) \in \mathcal{H} \times \{-1, +1\}$ for $i = 1, \ldots, N$ and $\mathcal{X} = \{\mathbf{x}_i | i = 1, \ldots, N\}$, where $\mathcal{H}$ is a generic Hilbert space.

3.1 The k-Nearest Neighbors

The k-Nearest Neighbors classifier (kNN) is an IBL approach to classification. Intuitively, it locally estimates the decision function with the majority rule using the examples in the k-neighbourhood of the query example. All the computation is delayed until query examples are available and for this reason is considered a lazy learning approach.

Given an example $\mathbf{x}' \in \mathcal{H}$, it is possible to order the entire set of training examples $\mathcal{X}$ with respect to $\mathbf{x}'$. This corresponds to defining a function $r_{\mathbf{x}'} : \{1, \ldots, N\} \to \{1, \ldots, N\}$ that recursively reorders the indexes of the N examples in $\mathcal{X}$:

$$\begin{cases} r_{\mathbf{x}'}(1) = \underset{i=1,\ldots,N}{\operatorname{argmin}} \|\mathbf{x}_i - \mathbf{x}'\| \\ r_{\mathbf{x}'}(j) = \underset{i=1,\ldots,N}{\operatorname{argmin}} \|\mathbf{x}_i - \mathbf{x}'\| \qquad i \neq r_{\mathbf{x}'}(1), \ldots, r_{\mathbf{x}'}(j-1) \text{ for } j = 2, \ldots, N \end{cases}$$

In this way, $\mathbf{x}_{r_{\mathbf{x}'}(j)}$ is the example of the set $\mathcal{X}$ in the j-th position in terms of distance from $\mathbf{x}'$, namely the j-th nearest neighbour, $\|\mathbf{x}_{r_{\mathbf{x}'}(j)} - \mathbf{x}'\|$ is its distance from $\mathbf{x}'$ and $y_{r_{\mathbf{x}'}(j)}$ is its class with $y_{r_{\mathbf{x}'}(j)} \in \{+1, -1\}$. In other terms:

$$j < k \Rightarrow \left\|\mathbf{x}_{r_{\mathbf{x}'}(j)} - \mathbf{x}'\right\| \leq \left\|\mathbf{x}_{r_{\mathbf{x}'}(k)} - \mathbf{x}'\right\|.$$

Given the above definition, the majority decision rule of kNN for binary classification problems is defined by

$$k\text{NN}(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{k} y_{r_{\mathbf{x}}(i)}\right).$$

For problems with more than two classes, kNN can be easily generalized modifying the majority rule such that it selects the class with the highest number of representatives in the k-neighbourhood instead of taking the sign of the summation.

It is well known that the generalization error of the 1NN classifier is bounded by twice the Bayes error [52] as $N \to \infty$. kNN can lower the generalization error for $k > 1$ and, in particular, kNN is bounded by the Bayes error [62] if the following constraints are satisfied: $k \to \infty$, $N \to \infty$ and $k/N \to 0$.

3.2 Support Vector Machines

Support Vector Machines (SVMs) [24, 51] are classifiers with sound foundations in statistical learning theory [193] that became very popular in the last decade. The decision rule is

$$\text{SVM}(\mathbf{x}) = \text{sign}(\langle w, \Phi(\mathbf{x})\rangle_{\mathcal{F}} + b)$$

where $\Phi(\mathbf{x}) : \mathcal{H} \to \mathcal{F}$ is a mapping in a transformed Hilbert feature space, called $\mathcal{F}$, with inner product $\langle\cdot,\cdot\rangle_{\mathcal{F}}$. The parameters $w \in \mathcal{F}$ and $b \in \mathbb{R}$ are such that they minimize an upper bound on the expected risk while minimizing the empirical risk. The minimization of the complexity term is achieved by the minimization of the quantity $\frac{1}{2}\cdot\|w\|^2$, which is equivalent to the maximization of the margin between the classes. In the optimization problem, the violation of the margin is prevented by the following set of constraints:

$$y_i\left(\langle w, \Phi(\mathbf{x}_i)\rangle_{\mathcal{F}} + b\right) \geq 1. \tag{3.1}$$

If a linear separation cannot be found in the input or feature space, the soft-margin variant of SVM permits the violation of the margin and the presence of misclassified training examples. This is possible introducing slack variables ξ_i:

$$y_i\left(\langle w, \Phi(\mathbf{x}_i)\rangle_{\mathcal{F}} + b\right) \geq 1 - \xi_i \qquad \xi_i \geq 0,\ i = 1, \ldots, N. \tag{3.2}$$

For soft-margin SVM the optimization problem (with linear penalizing of ξ_i, called L1-norm)

becomes $\frac{1}{2} \cdot \|w\|^2 + C \sum_i \xi_i$ subject to (3.2).

Reformulating such an optimization problem with Lagrange multipliers α_i $(i = 1, \ldots, N)$, and introducing a positive definite kernel (PD) function[1] $K(\cdot, \cdot)$ that substitutes the scalar product in the feature space $\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \rangle_{\mathcal{F}}$ (kernel functions are defined and analysed in the next section) the decision rule can be expressed as:

$$\mathrm{SVM}(\mathbf{x}) = \mathrm{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \right).$$

Throughout the thesis, we denote with SVM the soft-margin L1-norm SVM, unless differently specified.

The kernel trick avoids the explicit definition of the feature-space $\mathcal{F}$ and of the mapping Φ [157]. The maximal separating hyperplane defined by SVM has been shown to have important generalisation properties and nice bounds on the VC dimension [193].

Multiple schemas has been proposed in order to apply the maximal margin principle of SVM on multiple class problems. The most popular are the one-against-all method [25] which builds a number of binary decision functions equal to the number of classes N_{cl}, the one-against-one method [99, 102] which builds $N_{cl} \cdot (N_{cl} - 1)/2$ binary decision functions using voting in the prediction phase, and the Directed Acyclic Graph SVM (DAGSVM) [145] which is a modification of the one-against-all method. The study carried on by [88] shows that the more effective strategies are the one-against-one and DAGSVM approaches.

In their original formulation, SVMs are not able to give probability estimates for query examples. In order to obtain the probability estimate that an example $\mathbf{x}_i$ has positive class label, i.e. $\widehat{p}^{\,\mathrm{SVM}}(y = +1|\mathbf{x}) = 1 - \widehat{p}^{\,\mathrm{SVM}}(y = -1|\mathbf{x})$, Platt [144] proposed the following approximation refined by [114]:

$$\widehat{p}^{\,\mathrm{SVM}}(y = +1|\mathbf{x}) = \frac{1}{1 + \exp(A \cdot \mathrm{SVM}(\mathbf{x}) + B)}$$

where A and B are parameters that can be estimated by minimizing the negative log-likelihood using the training set and the associated decision values (using for example cross validation).

Accurate SVM solvers scales as the cube of the number of support vectors when the regularisation parameter C is large, when instead C is small, the solution of the regularisation problem is almost quadratic in the number of support vectors as discussed for example by [26]. Since the number of support vectors grows linearly with the dataset size[2] [178, 179], accurate SVM solvers scales between N^2 and N^3, so $\mathcal{O}(N^3)$. The prediction can be performed with linear complexity with respect to the number of support vectors and so $\mathcal{O}(N)$.

[1] For convention we refer to kernel functions with the capital letter K and to the number of nearest neighbours with the lower-case letter k.

[2] In particular we have that $N_{SV}/N \to 2\mathcal{B}_K$ where $\mathcal{B}_K$ is the smallest classification error achievable with SVM using the kernel K.

3.3 Kernel Functions

The class of functions that correspond to a dot product in some dot product space coincides with the class of positive definite (PD) kernels. A PD kernel is a function $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ that gives rise to a PD Gram matrix which is a real symmetric matrix $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ for all $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$ such that $\sum_{i,j} (c_i \cdot c_j \cdot K_{ij}) \geq 0$ for every $c_i, c_j \in \mathbb{R}$ (i.e. K is positive-definite). The basic kernel is the linear kernel $K^{lin}(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$ which is simply the dot product in the input space thus adopting the identity as mapping function and forcing the perfect congruence between input and feature-spaces. A kernel can also be defined directly with a real-valued function f as $k(x, x') = f(x)f(x')$. For a comprehensive discussion of theory of Reproducing Kernel Hilbert Spaces the reader can refer to [157, 54].

Popular kernels are the linear (LIN) kernel, the radial basis function (RBF) kernel, the general polynomial kernel (POL), the homogeneous (HPOL) and inhomogeneous (IPOL) polynomial kernels and the sigmoidal (SIG) kernel. Their definition are:

$$K^{lin}(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle \tag{3.3}$$

$$K^{rbf}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{\sigma}\right), \quad \sigma > 0 \tag{3.4}$$

$$K^{pol}(\mathbf{x}, \mathbf{x}') = (\gamma_{pol}\langle \mathbf{x}, \mathbf{x}' \rangle + r_{pol})^d, \quad d > 1, \;\; \gamma_{pol}, r_{pol} > 0 \tag{3.5}$$

$$K^{hpol}(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle^d, \quad d > 1 \tag{3.6}$$

$$K^{ipol}(\mathbf{x}, \mathbf{x}') = (\langle \mathbf{x}, \mathbf{x}' \rangle + 1)^d, \quad d > 1 \tag{3.7}$$

$$K^{sig}(\mathbf{x}, \mathbf{x}') = \tanh(\gamma^{sig} \cdot \langle \mathbf{x}, \mathbf{x}' \rangle + r^{sig}). \tag{3.8}$$

It is known that the linear, polynomial and radial basis function kernels are valid kernels since they are PD. It has been shown, however, that the sigmoidal kernel is not PD [157]; nevertheless it has been successfully applied in a wide range of domains as discussed in [158]. In [113] is showed that the sigmoidal kernel can be conditionally positive definite (CPD) for certain parameters and for specific inputs. Since CPD kernels can be safely used for SVM classification [159], the sigmoidal kernel is suitable for SVM only on a subset of the parameters and input space. In this work we use the sigmoidal kernel being aware of its theoretical limitations, which can be reflected in non-optimal solutions and convergence problems in the maximal margin optimization.

Parameter selection for K^{rbf}

For the radial basis function kernel K^{rbf} it is reasonable to set the parameter σ with the double of the squared median of the distribution of $\|\mathbf{x}_i - \mathbf{x}_j\|$, namely the Euclidean distances between every pair of examples $\mathbf{x}_i$ [190, 167]. In fact, with this choice of the kernel width, the distances are weighted with a value that is likely to be in same order of magnitude. More precisely, denoting with $q_h[\|\mathbf{x} - \mathbf{x}'\|^{\mathcal{Z}}]$ the h percentile of the distribution of the distance in the $\mathcal{Z}$ space between every pair of points $\mathbf{x}$, $\mathbf{x}'$ in the training set, σ can be chosen as $\sigma_h = 2 \cdot q_h^2[\|\mathbf{x} - \mathbf{x}'\|^{\mathcal{H}}]$. Reasonable choices for h can be 10, 50 (i.e. the median) or 90 that should be in the same order of magnitude of the median, and 1 which emphasises the local behaviour.

Universal kernels

Universal kernels [176, 180, 127] are kernels that permit to the associated feature map to approximate arbitrarily well any continuous function in the feature space. The notion of universality is important also because it is deeply related to the notion of consistency. Formally, the definition of universality of a kernel is:

Definition 1 (Universal Kernels [176])**.** *A continuous kernel function K is universal if the space of all functions induced by K is dense in $C(\mathcal{X})$, i.e. for every function $f \in C(\mathcal{X})$ and every $\epsilon > 0$ there exists a function g induced by K with*

$$\|f - g\|_\infty \leq \epsilon,$$

where $C(\mathcal{X})$ is the space of continuous functions $f : \mathcal{X} \to \mathbb{R}$.

Since they are able to approximate any function in the feature space, universal kernels are at least theoretically able to approximate arbitrarily well the Bayes decision functions, and thus *optimally* approximate the Bayes decision in probability.

Examples of universal kernels are the RBF kernel and the exponential kernel (see next subsection) [176].

3.3.1 Local and Global Kernels

Kernel functions can be divided in two classes: local and global kernels [171]. Following [13] we define the locality of a kernel as:

Definition 2 (Local kernel)**.** *A PD kernel K is a* local kernel *if, considering a test point $\mathbf{x}$ and a training point $\mathbf{x}_i$, we have that*

$$\lim_{\|\mathbf{x}-\mathbf{x}_i\|\to\infty} K(\mathbf{x}, \mathbf{x}_i) \to c_i \tag{3.9}$$

with c_i constant and not depending on $\mathbf{x}$. If a kernel is not local, it is considered to be global*.*

This definition captures the intuition that, in a local kernel, only the points that are enough close to each other influence the kernel value. This does not directly implicate that the higher

peak of the kernel value is in correspondence of points in the same position, although the most popular local kernel functions have this additional characteristic. In contrast, in a global kernel function, all the points are able to influence the kernel value regardless of their proximity.

It is simple to show that, among the five kernels listed above, the only local kernel is K^{rbf} since for $\|\mathbf{x} - \mathbf{x}_i\| \to \infty$ we have that $K^{rbf}(\mathbf{x}, \mathbf{x}_i) \to 0$ (i.e. a constant that does not depend on $\mathbf{x}$), whereas K^{lin}, K^{pol}, K^{hpol}, K^{ipol} and K^{sig} are global.

Isotropic stationary radial kernels.

An interesting kernel family is the class of stationary kernels [77] which is composed by kernels that are translational invariant, i.e. $K(\mathbf{x}, \mathbf{x}') = K_s(\mathbf{x} - \mathbf{x}')$. Additionally, if a stationary kernel depends only on the Euclidean distance between the examples, namely on the norm of the vector between them, we have an isotropic stationary kernel $K(\mathbf{x}, \mathbf{x}') = K_i(\|\mathbf{x} - \mathbf{x}'\|)$. The already introduced RBF kernel belongs to this subclass. Other isotropic stationary PD kernels are the following:

$$
\begin{aligned}
\text{exponential kernel} \qquad & K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|}{\sigma}\right) \\
\text{rational kernel} \qquad & K(\mathbf{x}, \mathbf{x}') = 1 - \frac{\|\mathbf{x} - \mathbf{x}'\|^2}{\|\mathbf{x} - \mathbf{x}'\|^2 + \theta} \\
\text{Beta kernel} \qquad & K(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|^\beta,\ 0 < \beta \leq 2 \\
\text{uniform kernel} \qquad & K(\mathbf{x}, \mathbf{x}') = 1/2\, I(\sqrt{\|\mathbf{x} - \mathbf{x}'\|} \leq h) \\
\text{triangular kernel} \qquad & K(\mathbf{x}, \mathbf{x}') = \frac{h - \sqrt{\|\mathbf{x} - \mathbf{x}'\|}}{h} I(\sqrt{\|\mathbf{x} - \mathbf{x}'\|} \leq h) \\
\text{Laplacian kernel} \qquad & K(\mathbf{x}, \mathbf{x}') = \lambda^n 2^{-n} \exp\left(-\lambda \|\mathbf{x} - \mathbf{x}'\|\right) \\
\text{multiquadratic kernel} \qquad & K(\mathbf{x}, \mathbf{x}') = (\|\mathbf{x} - \mathbf{x}'\|^2 + c^2)^{1/2} \\
\text{thin plate splines kernel} \qquad & K(\mathbf{x}, \mathbf{x}') = (\|\mathbf{x} - \mathbf{x}'\|^2 + c^2)^{2n+1} \\
\text{KMOD kernel} \qquad & K(\mathbf{x}, \mathbf{x}') = c\left[\exp\left(\frac{\gamma}{\|\mathbf{x} - \mathbf{x}'\|^2 + \sigma^2}\right) - 1\right].
\end{aligned}
$$

Another distance measure that can be used to design kernels, derives from the spectral angle mapper (SAM) which is a scale-invariant and nonadditive distance metric used in remote sensing

problems mainly for measuring the spectral difference between examples since it is robust to differences in spectral energy [125, 69]. It consists in determining the angle θ between two vectors as

$$\theta(\mathbf{x}, \mathbf{x}') = \arccos\left(\frac{\langle \mathbf{x}, \mathbf{x}' \rangle}{\|\mathbf{x}\|\|\mathbf{x}'\|}\right).$$

The $\theta(\mathbf{x}, \mathbf{x}')$ angle can be used to define SAM based kernels; for example [69] proposed $K^{SAM} = \exp[-\gamma\,\theta(\mathbf{x}, \mathbf{x}')]$. Notice that, in principle, we can use SAM to obtain a distance between examples and embed it in all isotropic stationary kernels.

3.3.2 Building Kernels from Kernels

We summarize the main properties of kernels through which we can construct kernels starting from other kernels. The idea is that a kernel can be built using other kernels as building blocks.

Proposition 1. *Let K_1, K_2 two PD kernels, K_3 a PD kernel on $\mathbb{R}^p \times \mathbb{R}^p$, c a real constant, ψ an $\mathbb{R}^p$-valued function, and $pol_d^+ = \left\{ \sum_{i=1}^{d} \alpha_i \mathbf{x}^i | d \in \mathbb{N}, \alpha_1, \ldots, \alpha_n \in \mathbb{R}^+ \right\}$ any polynomial with positive coefficients and degree d; then the following are PD kernels:*

1. $K(\mathbf{x}, \mathbf{x}') = K_1(\mathbf{x}, \mathbf{x}') + K_2(\mathbf{x}, \mathbf{x}')$
2. $K(\mathbf{x}, \mathbf{x}') = c \cdot K_2(\mathbf{x}, \mathbf{x}')$
3. $K(\mathbf{x}, \mathbf{x}') = K_1(\mathbf{x}, \mathbf{x}') \cdot K_2(\mathbf{x}, \mathbf{x}')$
4. $K(\mathbf{x}, \mathbf{x}') = pol_d^+(K_1(\mathbf{x}, \mathbf{x}'))$
5. $K(\mathbf{x}, \mathbf{x}') = \exp(K_1(\mathbf{x}, \mathbf{x}'))$
6. $K(\mathbf{x}, \mathbf{x}') = K_3(\psi(\mathbf{x}), \psi(\mathbf{x}'))$

Proof. The proof of these properties can be found in [54, 77]. □

With this small set of properties it is possible to derive a wide spectrum of kernels; an example is the polynomial kernel that can be simply obtained with point 4. of Proposition 1 starting from the linear kernel.

Building kernels by mixing and combining kernels

Mixture of kernels can be obtained applying properties 1. and 2. of Proposition 1 and was introduced in kernel methods mainly for two reason: (i) for handling heterogeneity in data integrating two kernels reflecting different prior information [139, 125] and (ii) for combining the good characteristics of global kernels (like extrapolation abilities) with those of local kernels

(like the interpolation abilities) as discussed in [171]. In both cases the mixture consists in the convex combination of two kernels K_1, K_2:

$$K(\mathbf{x}, \mathbf{x}') = \rho K_1(\mathbf{x}, \mathbf{x}') + (1 - \rho) K_2(\mathbf{x}, \mathbf{x}') \qquad 0 \leq \rho \leq 1$$

The generalization of the mixture of two kernels is the unweighted and weighted summations of kernels, defined respectively as:

$$K(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{n} K_i(\mathbf{x}, \mathbf{x}') \tag{3.10}$$

$$K(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{n} \rho_i \cdot K_i(\mathbf{x}, \mathbf{x}') \qquad n \in \mathbb{N}. \tag{3.11}$$

In the case of weighted combination of kernels the problem is the estimation of the weights. Considerable work has been done for this purpose. Apart from the classical model selection techniques such as cross validation, some of the more relevant approaches to determine the coefficients of the linear combination of kernel (sometimes even in a more general framework) are those based on semi-definite programming (Multiple Kernel Learning) [104, 103], on the so-called hyperkernels [135], on boosting [53], on the computation of the regularisation path [9], on gradient descent [28], on hierarchical Bayesian models [78] and on von Neumann Entropy [121]. Apart from linear combination of kernel, little work has been done on other kernel combination schemes. An example is proposed in [56] in which starting from the notion of average and differences of kernels, three new kernel combination methods are introduced: the absolute value, the squared quantity and the squared matrix methods.

3.4 Local Support Vector Machines

The method [19, 20] combines locality and search for a large margin separating surface by partitioning the entire transformed feature-space through a set of local maximal margin hyperplanes. It can be seen as a modification of the SVM approach in order to obtain a local learning algorithm [27] able to locally adjust the capacity of the training systems. The local learning approach is particularly effective for uneven distributions of training set examples in the input space. Although kNN is the simplest local learning algorithm, its decision rule based on majority voting overlooks the geometric configuration of the neighbourhood. For this reason the adoption of a maximal margin principle for neighbourhood partitioning can result in a good compromise between capacity and number of training examples [192]. In [212] the authors independently developed a slightly different approach for Local SVM; their version is based on a "crude" and approximated distance metric used to compute a first approximation of the neighbourhood of the testing point and on DAGSVM [145].

In our setting, in order to classify a given example $\mathbf{x}'$ of the input space, we need first to find its k nearest neighbours in the transformed feature-space $\mathcal{F}$ and, then, to search for an optimal

separating hyperplane only over these k nearest neighbours. In practice, this means that an SVM is built over the neighbourhood of each test example $\mathbf{x}'$. Accordingly, the constraints in (3.1) become:

$$y_{r_{\mathbf{x}}(i)} \left(w \cdot \Phi(\mathbf{x}_{r_{\mathbf{x}}(i)}) + b \right) \geq 1 - \xi_{r_{\mathbf{x}}(i)}, \text{ with } i = 1, \ldots, k$$

where $r_{\mathbf{x}'} : \{1, \ldots, N\} \rightarrow \{1, \ldots, N\}$ is a function that reorders the indexes of the training examples defined as:

$$\begin{cases} r_{\mathbf{x}'}(1) = \underset{i=1,\ldots,N}{\operatorname{argmin}} \|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}')\|^2 \\ r_{\mathbf{x}'}(j) = \underset{i=1,\ldots,N}{\operatorname{argmin}} \|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}')\|^2 \qquad i \neq r_{\mathbf{x}'}(1), \ldots, r_{\mathbf{x}'}(j-1) \text{ for } j = 2, \ldots, N \end{cases} \tag{3.12}$$

In this way, $\mathbf{x}_{r_{\mathbf{x}'}(j)}$ is the example of the set $\mathcal{X}$ in the j-th position in terms of distance from $\mathbf{x}'$ and the thus

$$j < k \Rightarrow \|\Phi(\mathbf{x}_{r_{\mathbf{x}'}(j)}) - \Phi(\mathbf{x}')\| \leq \|\Phi(\mathbf{x}_{r_{\mathbf{x}'}(k)}) - \Phi(\mathbf{x}')\|$$

because of the monotonicity of the quadratic operator. The computation is expressed in terms of kernels as:

$$\begin{aligned} & ||\Phi(\mathbf{x}) - \Phi(\mathbf{x}')||^2 = \\ = \;& \Phi^2(\mathbf{x}) + \Phi^2(\mathbf{x}') - 2 \cdot \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}') = \\ = \;& \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}) \rangle_{\mathcal{F}} + \langle \Phi(\mathbf{x}'), \Phi(\mathbf{x}') \rangle_{\mathcal{F}} - 2 \cdot \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle_{\mathcal{F}} = \\ = \;& K(\mathbf{x}, \mathbf{x}) + K(\mathbf{x}', \mathbf{x}') - 2 \cdot K(\mathbf{x}, \mathbf{x}'). \end{aligned} \tag{3.13}$$

If the kernel is the RBF kernel or any polynomial kernels with degree 1, the ordering function is equivalent to using the Euclidean metric. For some non-linear kernels (other than the RBF kernel) the ordering function can be quite different to that produced using the Euclidean metric.

The decision rule associated with the method for an example $\mathbf{x}$ and a training set $\mathcal{X}$ is:

$$k\text{NNSVM}(\mathbf{x}; \mathcal{X}) = \text{sign} \left(\sum_{i=1}^{k} \alpha_{r_{\mathbf{x}}(i)} y_{r_{\mathbf{x}}(i)} K(\mathbf{x}_{r_{\mathbf{x}}(i)}, \mathbf{x}) + b \right). \tag{3.14}$$

Figure 3.1 shows an example of the application of kNNSVM on a toy dataset with $k = 15$ and RBF kernel.

For $k = N$, the kNNSVM method is the usual SVM because all local models actually uses all training examples whereas, for $k = 2$, the method implemented with the LIN or RBF kernel corresponds to the standard 1NN classifier as exemplified in Figure 3.2.

Notice that in situations where the neighbourhood contains only one class the local SVM does not find any separation and so considers all the neighbourhood to belong to the predominant class thus simulating the behaviour of the majority rule.

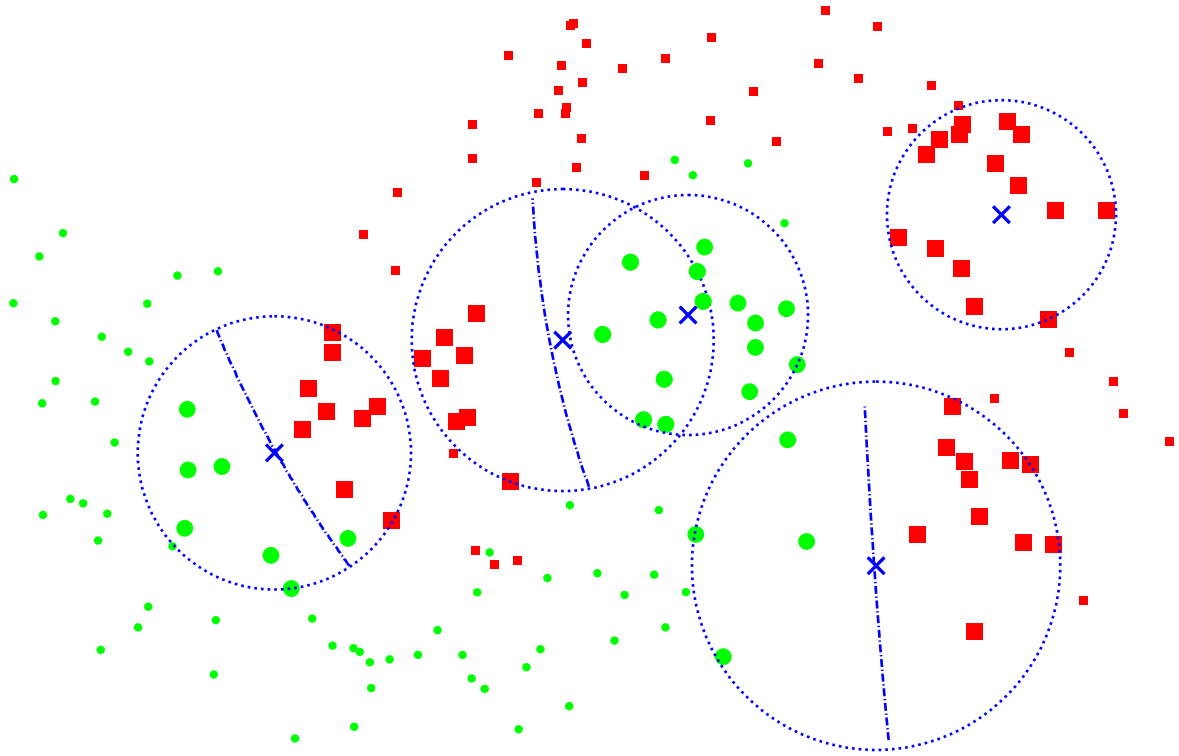


Figure 3.1: The decision functions of kNNSVM with $k = 15$, $C = 10$, K^{rbf} with $\gamma = 10$ for five query points (the blue crosses). The examples in the 15-neighbourhood used for training the five local SVM models are magnified.

Considering kNNSVM as a local SVM classifier built in the feature-space, the method has been shown to potentially have a favourable bound on the expectation of the probability of test error with respect to SVM [20].

We carried out some empirical comparison of kNNSVM with SVM in [164] and an extensive evaluation on a total of 34 datasets in the next chapter, obtaining favourable results. The computational complexity and novel learning bounds are also discussed in the next chapter.

The probability output for this method can be obtained using the local SVM probability estimation as follows:

$$\widehat{p}^{\,kNNSVM}(y = +1|\mathbf{x}; \mathcal{X}) = \frac{1}{1 + \exp(A \cdot kNNSVM(\mathbf{x}; \mathcal{X}) + B)}$$

The generalization of kNNSVM for multi-class classification can occur locally, i.e. solving the local multi-class SVM problem, or globally, i.e. applying the binary kNNSVM classifier on multiple global binary problems. In [164] the adopted strategy for multi-class classification with kNNSVM is the one-against-one strategy applied on the local problems. The choice of the one-against-one approach gave good results in comparison with SVM adopting globally the

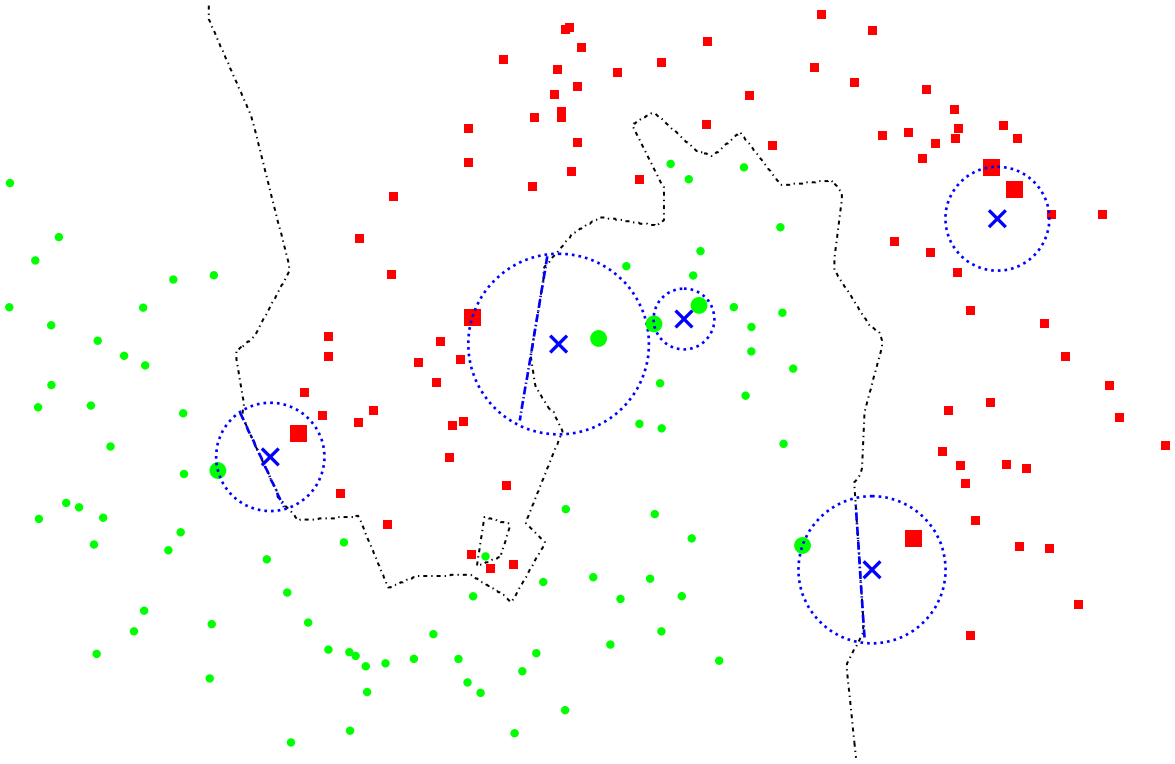


Figure 3.2: The decision functions of kNNSVM with $k = 2$ is equivalent, for each query point, to the NN decision function (the dotted black line). The pairs of examples responsible of the five local hyperplanes are magnified.

same strategy, but no specific empirical studies have been performed yet to understand which is the most appropriate strategy for multi-class classification with Local SVM.

An implementation of kNNSVM, called FkNNSVM, is available in FaLKM-lib [163] which is described in Appendix A.

3.5 Cover Trees for Neighborhood Operations

A Cover Tree is a data structure introduced by Beygelzimer et al. [15] for performing fast and efficient non-approximated nearest neighbour operations. Cover Trees can be applied in general metric spaces without assumptions on the structure and thus also in Hilbert Spaces calculating the distances by means of kernel functions using the kernel trick.

In more detail, a Cover Tree can be viewed as a subgraph of a navigating net [101] and it is a leveled tree in which each level (indexed by a decreasing integer i) is a cover (i.e. is representative) for the level beneath it. Every node of a Cover Tree T is associated with a point of a dataset S. Denoting with C_i the set of points associated with nodes in T at level i, with $b > 1$ a constant, and with $dist(\cdot, \cdot)$ the distance function defining the metric of the space, the

Algorithm 1 Find-Nearest(cover tree T, query point p)

1: set $Q_\infty = C_\infty$ where C_∞ is the root level of T
2: **for** i from ∞ down to $-\infty$ **do**
3: set $Q = \{\text{Children}(q) : q \in Q_i\}$
4: form cover set $Q_{i-1} = \{q \in Q : d(p,q) \leq d(p,Q) + 2^i\}$
5: **end for**
6: **return** $\text{argmin}_{q \in Q_{-\infty}} d(p,q)$

Algorithm 2 Insert(point p, cover set Q_i, level i)

1: $Q = \{\text{Children(q)} : q \in Q_i\}$
2: **if** $d(p,Q) > 2^i$ **then**
3: **return** "parent found" - TRUE
4: **else**
5: $Q_{i-1} = \{q \in Q : d(p,q) \leq 2^i\}$
6: found = Insert(p, Q_{i-1}, $i-1$)
7: **if** found and $d(p,Q_i) \leq 2^i$ **then**
8: pick a single $q \in Q$, such that $d(p,q) \leq 2^i$
9: insert p into $Children(p)$
10: **return** "finished" - FALSE
11: **else**
12: **return** found
13: **end if**
14: **end if**

invariants of a Cover Tree are:

Nesting $C_i \subset C_{i-1}$

Covering tree For every $\mathbf{p} \in C_{i-1}$ there exists a $\mathbf{q} \in C_i$ such that $dist(\mathbf{p},\mathbf{q}) < b^i$ and the node in level i associated with $\mathbf{q}$ is a parent of the node in level $i-1$ associated with $\mathbf{p}$.

Separation For all distinct $\mathbf{p},\mathbf{q} \in C_i$, $dist(\mathbf{p},\mathbf{q}) > b^i$.

Intuitively, the nesting invariant means that once a point appears in a level, it is present for every lower level. A covering tree implies that every node has a parent in the higher level such that the distance between the respective points is less than b^i, while separation assures that the distance between every pair of points associated to the nodes of a level i is higher than b^i. In addition, the root of the tree (i.e. the example in C_∞) is a randomly chosen example.

Denoting with $d(p,Q)$ the distance between the point p and its nearest point in the set Q, Algorithm 1 and Algorithm 2 present the pseudo-code for the insertion an query operations using $b = 2$ as reported in [15] (the original insertion algorithm presents a bug corrected here).

Cover Trees have state-of-the-art performance for exact nearest neighbour operations for general metrics in low-dimensional spaces both in terms of computational complexity and space

requirements. As theoretically proved by Beygelzimer et al. [15], the space required by the Cover Tree data-structure is linear in the dataset size ($\mathcal{O}(n)$), the computational time of single point insertions, deletions and exact nearest neighbour queries is logarithmic ($\mathcal{O}(\log n)$) while the Cover Tree can be built in $\mathcal{O}(n \log n)$. Other related approaches that seems however to guarantee lower performances in practice are R-Trees by Guttman [80], Ball Trees by Omohundro [134, 116] or k-d trees by Wess et al. [201].

Chapter 4

Theoretical and Empirical Analysis of Local SVM

Local Support Vector Machines have been independently introduced by Blanzieri and Melgani [19] and by Zhang et al. [212] and successfully applied respectively to remote sensing and visual recognition tasks. The approach of [19] has been also applied for spam filtering [18]. However, no extensive empirical evaluation has been performed yet in order to understand if the generalization ability of Local SVM is competitive with state-of-the-art classifiers like SVM. The first purpose of this chapter thus consists in assessing this question using various datasets, with different characteristics and application domains, and using different kernel functions and experimental protocols.

From the theoretical viewpoint, it has been shown in [20] that Local SVM can lower the Radius/Margin bound of SVM for some choices of the locality parameter and thus guarantee high classification accuracies. In this chapter we give another complementary theoretical analysis of Local SVM, based on the framework of Local Learning Algorithms [27, 194] (introduced in Chapter 2.1.2), deriving a generalization bound for Local SVM that highlights the possibility of obtaining a lower misclassification risk with respect to SVM. The computational complexity of the approach is also discussed.

The chapter is organized as follows. Section 4.1 introduce the generalization bound for kNNSVM (the algorithm for Local SVM, see Chapter 3.4), whereas Section 4.2 focuses on the computational performances of the algorithm. Section 4.3 details the three experiments we carried out for assessing the classification accuracies of kNNSVM with respect to SVM. In particular, the first experiment, Section 4.3.1, is devoted to compare kNNSVM with SVM on 25 binary-class problems using three different kernel functions, the second experiment, Sec-

tion 4.3.2, analyses the case of multi-class and high-dimensional problems, the third experiment, Section 4.3.3, further discusses the differences between kNNSVM and SVM with RBF kernel on artificially generated data.

We partially presented the results of this chapter in [164] for the empirical part and in [166] for the theoretical part.

4.1 A Generalization Bound for kNNSVM

The class of LLA introduced by [27] (see Chapter 2.1.2), and to which kNNSVM belongs, can be theoretically analysed using the framework based on the local risk minimization [194, 193]. Starting from this theory, we derive here a generalization bound for kNNSVM.

We need to recall the bound for the local risk minimization, which is a generalization of the global risk minimization theory.

Theorem 1 (Vapnik 2000 [193])**.** *For a testing point $\mathbf{x}'$ and with probability $1-\eta$ simultaneously for all bounded functions $A \leq L(y, f(\mathbf{x}, \alpha)) \leq B$, $\alpha \in \Lambda$ (where Λ is a set of parameters), and all locality functions $0 \leq T(\mathbf{x}, \mathbf{x}_0, \beta) \leq 1$, $\beta \in (0, \infty)$, the following inequality holds true:*

$$R^{LLA}(\alpha, \beta, \mathbf{x}') \leq \frac{\frac{1}{N}\sum_{i=1}^{N} L(y_i, f(\mathbf{x}_i, \alpha))T(\mathbf{x}_i, \mathbf{x}', \beta) + (B-A)\gamma(N, h^{\Sigma})}{\left|\frac{1}{N}\sum_{i=1}^{N} T(\mathbf{x}_i, \mathbf{x}', \beta) - \gamma(N, h^{\beta})\right|},$$

where

$$\gamma(N, h) = \sqrt{\frac{h \ln(2N/h + 1) - \ln \eta/2}{N}},$$

and h^{Σ} is the VC dimension of the set of functions $L(y_i, f(\mathbf{x}_i, \alpha))T(\mathbf{x}_i, \mathbf{x}', \beta), \alpha \in \Lambda, \beta \in (0, \infty)$ and h^{β} is the VC dimension of $T(\mathbf{x}_i, \mathbf{x}', \beta)$

For kNNSVM, we consider the following loss function

$$L(y_i, f(\mathbf{x}_i, \alpha)) = \begin{cases} 0 & \text{if } y_i = f(\mathbf{x}_i, \alpha) \\ 1 & \text{if } y_i \neq f(\mathbf{x}_i, \alpha) \end{cases}$$

Notice that the locality function $T(\mathbf{x}, \mathbf{x}_0, \beta)$ can be an "hard-threshold" locality function, i.e. a function giving 1 to $\mathbf{x}$ if its distance from $\mathbf{x}'$ is lower than a fixed distance depending on β and 0 otherwise, or a "soft-threshold" locality function that assign a positive weight to $\mathbf{x}$ depending on its distance from $\mathbf{x}'$. A "soft-threshold" locality function can for example penalise the distance of an example $\mathbf{x}$ from $\mathbf{x}'$ with a negative exponential as:

$$T(\mathbf{x}_i, \mathbf{x}', k) = \exp(-\beta \|\mathbf{x} - \mathbf{x}_0\|).$$

For kNNSVM we retrieve the neighbourhood of $\mathbf{x}'$ and thus we use an "hard-threshold"

locality function that we can easily define using the r function defined in Chapter 3.1 for kNN:

$$T(\mathbf{x}_i, \mathbf{x}', k) = \begin{cases} 1 & \text{if } \exists j \leq k \text{ s.t. } i = r_{\mathbf{x}'}(j) \\ 0 & \text{otherwise} \end{cases}$$

It is straightforward to show that for each query point $\mathbf{x}'$ the sum of the locality function for each training point $\mathbf{x}_i \in \mathcal{X}$ is k:

$$\sum_{i=1}^{N} T(\mathbf{x}_i, \mathbf{x}', \beta) = k.$$

Moreover $T(\mathbf{x}_i, \mathbf{x}', k)$ has VC dimension equal to 2; it is, in fact, a function that can build hyperspheres centered in $\mathbf{x}'$ with diameters equal to the distances of the points from $\mathbf{x}'$ and can thus shatter any set of two points with different class, but cannot shatter three points with the nearest and furthest points having a class different from the third point. Notice that the VC dimension of $T(\mathbf{x}_i, \mathbf{x}', k)$ is 2 regardless of the dimensions of the space.

Noticing that, in our case,

$$\sum_{i=1}^{N} L(y_i, f(\mathbf{x}_i, \alpha)) T(\mathbf{x}_i, \mathbf{x}', \beta) = \sum_{i=1}^{k} L(y_i, f(\mathbf{x}_i, \alpha)),$$

we can obtain:

$$R^{k\text{NNSVM}}(\alpha, k, \mathbf{x}') \leq \frac{\frac{1}{N} k \cdot \nu_{\mathbf{x}'} + \gamma(N, h^{\Sigma})}{\left| \frac{1}{N} k - \gamma(N, 2) \right|} \tag{4.1}$$

where $\nu_{\mathbf{x}'}$ is the ratio of misclassified training points in the k-neighbourhood of $\mathbf{x}'$. Notice that, given η, $\gamma(N, 2)$ is a constant and thus the only parameters (in addition to N and k) are $\gamma(N, h^{\Sigma})$ and $\nu_{\mathbf{x}'}$ which is obtained after the training of the SVM model.

The possibility of obtaining a lower bound on test misclassification probability with local approaches acting with the locality parameter, as stated in [194, 193] for LLA, it is even more evident for kNNSVM looking at Eq. 4.1. In fact, although choosing a $k < N$ is not sufficient to lower the bound, as the model training becomes more and more local the misclassification training rate $\nu_{\mathbf{x}'}$ is very likely to decrease as well. Moreover, also the complexity of the classifier (and thus h^{Σ}) can decrease when the neighbourhood decreases, because simpler decision functions can be used when fewer points are considered.

Taking this into consideration, it is necessary to consider the trade-off between the degree of locality k, the function of the empirical error with respect to k and the complexity of the local classifier needed with respect to k, in order to find a minimum of the expected risk which is lower than the $k = N$ case. Multiple strategies can be used to tune this trade-off, especially if prior or high-level information are available for a specific problem; since in this work we aim to be as general as possible, the expected risk is estimated for the computational experiments using cross-validation based approaches.

4.2 Computational Complexity Bounds for kNNSVM

For each query point, kNNSVM needs to retrieve its k-neighbourhood in the training set, train a local SVM on the k points, and predict the class of the query points with the trained local model. Since the computation is delayed until the testing points are available, kNNSVM is a lazy learning approach that avoids the training phase.

If we use a *brute-force* approach for kNN we need to compute the distances between the query example and all the training examples, to sort the examples by distance and to select the k examples with the smallest distances. Using a sorting algorithm like *quicksort* we can retrieve the neighbourhood of a query example with a computational complexity of $\mathcal{O}(N+N\cdot\log N+k) = \mathcal{O}(N\cdot\log N)$ in average. We can avoid the sorting of all N distances because we need only the k smallest distances and thus, using a partial sorting algorithm we can lower the computational complexity of the k-neighbourhood retrieval to $\mathcal{O}(N + N\cdot\log k + k) = \mathcal{O}(N\cdot\log k)$.

Recalling that SVM has a complexity of $\mathcal{O}(N^3)$ for training and $\mathcal{O}(N)$ for testing (see Chapter 3.2), the overall complexity of the testing phase of kNNSVM using a brute-force approach for kNN is:

$$\mathcal{O}(N\cdot\log k + k^3 + k) = \mathcal{O}(N\cdot\log k + k^3).$$

For small values of k the term is dominated by N, otherwise k^3 is the limiting factor.

Using the Cover Trees (Chapter 3.5), it is possible to lower the testing complexity of kNNSVM. In particular, if we build the Cover Tree during training, we can retrieve the nearest neighbour in $\mathcal{O}(\log N)$ and the k-neighbourhood in $\mathcal{O}(k\log N)$. The algorithm implementing kNNSVM using Cover Tree, called FkNNSVM, has a training complexity of $\mathcal{O}(N\log N)$ (the complexity of building the Cover Tree) and a overall testing complexity of $\mathcal{O}(k\cdot\log N + k^3)$. Throughout this thesis, we use the FkNNSVM implementation of kNNSVM available in the FaLKM-lib [163] and described in Appendix A that make use of Cover Trees and thus has a testing complexity which is logarithmic in N.

This analysis highlights that kNNSVM is computationally inefficient at testing time. The problem can be alleviated by the use of metric trees like Cover Trees, that permits to the testing module to scale logarithmically in the number of training points, but, if the value of the locality parameter k is not very low, the training of the local SVM model at prediction remains an important overhead.

4.3 Empirical Analysis of kNNSVM

In this section we carry out a comparison between SVM (using LibSVM) and kNNSVM (using FkNNSVM) on a total of 34 classification problems (binary-class problems, multi-class problems and artificial problems) with different kernel functions. The objective is to assess if kNNSVM has better generalization performances than SVM and, in case, which are the situations in which the difference is more significant. Some results regarding kNNSVM and SVM can also be found in the following chapters in which we compare SVM and kNNSVM with our novel techniques; however the adopted experimental protocol can be different due to the need of making it consistent with

dataset name	# of features	# of points	class balancing	dataset name	# of features	# of points	class balancing
sonar	60	208	53%/47%	fourclass	2	862	64%/36%
heart	13	270	56%/44%	tic-tac-toe	9	958	65%/35%
mushrooms	112	300	53%/47%	mam	5	961	54%/46%
haberman	3	306	74%/26%	numer	24	1000	70%/30%
liver	6	345	58%/42%	splice	60	1000	52%/48%
ionosphere	34	351	64%/36%	spambase	57	1000	57%/43%
vote	15	435	61%/39%	vehicle	21	1243	76%/24%
musk1	166	476	57%/43%	cmc	7	1473	57%/43%
hill-valley	100	606	51%/49%	ijcnn1	22	1500	68%/32%
breast	10	683	65%/35%	a1a	123	1605	76%/24%
australian	14	690	56%/44%	chess	35	2130	52%/48%
transfusion	4	748	76%/24%	astro	4	3089	65%/35%
diabetes	8	768	65%/35%				

Table 4.1: The 25 binary-class datasets of the first empirical experiment.

the protocol used for the techniques to which it is compared, and thus the results can be a little bit different.

4.3.1 Experiment 1: kNNSVM on Binary-Class Datasets

In this experiment we compare SVM (using LibSVM) with kNNSVM (using FkNNSVM) on 25 non-large datasets, with the objective of studying the generalization performances of kNNSVM with respect to SVM.

Experimental protocol

The datasets are listed in Table 4.1; they are retrieved from the UCI [7] and Statlog [128] repositories, with cardinality between 200 and 3100 points (some datasets have been randomly sub-sampled), dimensionality lower than 200, not very unbalanced, and they are all scaled in the $[0, 1]$ interval. The comparison is carried out using three different kernel functions (the linear, the RBF and the homogeneous polynomial kernels), in a 10-fold cross validation (CV) setting. Internal to each training fold the model selection is performed with a nested 10-fold CV choosing the parameters in the following ranges. The regularisation parameter C is chosen for all methods in the set $\{2^{-2}, 2^{-1}, \ldots, 2^{9}, 2^{10}\}$, the width parameter σ of the RBF kernel in $\{2^{-5}, 2^{-4}, \ldots, 2^{2}, 2^{3}\}$, the degree of the polynomial kernel in $\{1, 2, 3\}$. The neighbourhood parameter k for FkNNSVM is selected by the cross-validation procedure in the set $\{2^1, 2^2, \ldots, 2^9, 2^{10}, |\mathcal{X}|\}$ where $|\mathcal{X}|$ is the cardinality of the training set[1].

[1] for dataset with less than 1024 points some k value are of course not tested.

dataset	K^{lin}		K^{rbf}		K^{hpol}	
	LibSVM	FkNNSVM	LibSVM	FkNNSVM	LibSVM	FkNNSVM
sonar	74.52	**89.36**	87.83	86.90	83.16	87.40
heart	**84.81**	**84.81**	82.22	81.11	**84.81**	**84.81**
mushrooms	97.99	**98.67**	98.33	98.33	98.32	98.60
haberman	73.20	**75.82**	73.20	75.16	72.89	74.18
liver	68.71	73.64	**74.24**	73.96	71.90	73.94
ionosphere	88.04	93.75	93.72	**94.59**	88.88	93.75
vote	94.95	96.32	96.32	**96.33**	94.95	96.32
musk1	86.55	89.44	94.54	**94.96**	93.07	91.17
hill-valley	63.70	64.86	**66.00**	65.18	63.70	64.86
breast	**96.78**	96.49	**96.78**	96.49	**96.78**	96.35
australian	**85.50**	84.78	84.78	**85.50**	84.20	84.92
transfusion	76.21	**79.81**	77.40	78.74	76.47	**79.81**
diabetes	76.54	76.81	76.54	**78.24**	76.68	77.07
fourclass	77.39	**100.00**	**100.00**	**100.00**	78.66	**100.00**
tic-tac-toe	98.33	**100.00**	99.68	**100.00**	**100.00**	**100.00**
mam	82.10	**82.95**	82.63	82.73	81.27	82.85
numer	**77.00**	76.30	75.90	75.70	76.50	76.00
splice	80.41	80.41	**86.70**	86.30	86.60	86.60
spambase	89.80	**90.60**	**90.60**	90.50	89.80	**90.60**
vehicle	82.71	82.78	84.16	84.64	**84.80**	84.71
cmc	59.26	62.46	65.45	**67.72**	64.16	63.61
ijcnn1	85.53	93.93	**93.94**	93.47	92.73	93.60
a1a	**83.43**	82.87	81.94	82.06	**83.43**	82.87
chess	96.57	97.84	98.45	**98.50**	98.03	98.08
astro	95.34	96.96	96.73	96.92	96.89	**97.05**
mean rank	4.96	3.12	3.2	2.84	4.04	2.84

Table 4.2: 10-fold cross validation accuracy results for the 25 datasets of the first experiment. The best results for each dataset are highlighted in bold (taking into account all decimal values).

Results and discussion

Table 4.2 reports the accuracy results of LibSVM and FkNNSVM for each kernel and dataset; also the mean rank of each technique across all datasets is reported. Looking at the mean ranks, FkNNSVM with RBF and HPOL kernels are the methods that achieve the best accuracy results. FkNNSVM with linear local models performs slightly worse than FkNNSVM with RBF and HPOL kernels but much better than global linear SVM and better than SVM with the HPOL and RBF kernels. FkNNSVM with LIN and RBF kernel are the approaches that achieve the highest number (9) of best results on each dataset.

Performing the Wilcoxon Signed Rank Test [202, 61] to detect statistical differences between LibSVM and FkNNSVM on the same kernel, we have that, using $\alpha = 0.05$, FkNNSVM is signif-

dataset name	# of features	# of points	data source	dataset name	# of features	# of points	data source
iris	60	208	UCI [7]	vehicle4	2	862	Statlog [98]
wine	13	270	UCI [7]	vowel	9	958	UCI [7]
leukemia	112	300	[79]	glass	5	961	UCI [7]
bioinf	3	306	[87]				

Table 4.3: The datasets used for the second experiment. Number of classes, training set cardinality, sources and number of features are reported.

icantly better than LibSVM for the linear and polynomial kernels, whereas for the RBF kernel no significant differences are detected, although the mean rank of FkNNSVM with RBF kernel is lower than LibSVM with RBF kernel.

We can thus conclude that kNNSVM is significantly better than SVM on binary-class datasets if both methods use non-local kernels, whereas if a local kernel like the RBF is used, the difference is still in favour of kNNSVM but it is not statistically relevant. This could be due to the fact that SVM with RBF kernel is already very accurate and relevant improvements over it are very difficult. We may also argue that locality is already included in the RBF kernel and thus, at least for non large datasets, the adoption of a local method is somehow equivalent.

4.3.2 Experiment 2: kNNSVM on Multi-Class and High-Dimensional Data

Here we test the performances of kNNSVM (using FkNNSVM) in comparison with the performances of SVM (using LibSVM) on 6 multi-class datasets and one high-dimensional dataset, in order to understand if the results highlighted by the previous experiment on binary-class datasets are confirmed also in these cases.

Experimental protocol

The datasets used in this experiment are listed in Table 4.3 with the corresponding sources. They are multi-class problems with a number of classes ranging from 3 to 11 except for the leukemia dataset which is a binary classification problem with high-dimensionality.

We evaluate the performances using the 10-fold CV classification accuracies considering the linear kernel (LIN), the radial basis function kernel (RBF), the homogeneous polynomial kernel (HPOL) and the inhomogeneous polynomial kernel (IPOL). The folds were randomly chosen during preprocessing. The model selection (on each fold) was performed with 10-fold CV splitting randomly the data at each application. The regularisation parameter C of SVM is chosen in $\{1, 5, 10, 25, 50, 75, 100, 150, 300, 500\}$, σ of the RBF kernel among $\{2^{-10}, 2^{-9}, \ldots, 2^{9}, 2^{10}\}$ and the degree of the polynomial kernels is bounded to 5. The dimension of the neighbourhood for the kNNSVM classifier, i.e. k, is chosen among the first 5 odd natural numbers followed by the ones obtained with a base-2 exponential increment from 9 and the cardinality of the training set, namely in $\{1, 3, 5, 7, 9, 11, 15, 23, 39, 71, 135, 263, 519, |\mathcal{X}|\}$. For the multi-class datasets we adopt the one-against-one strategy for SVM and the same strategy,

dataset	linear kernel K^{lin}				Gaussian RBF kernel K^{rbf}			
	LibSVM	FkNNSVM	diff	ttest	LibSVM	FkNNSVM	diff	ttest
iris	0.967	0.960	−0.007		0.947	0.960	+0.013	
wine	0.966	0.983	+0.017		**0.994**	0.989	−0.006	
leukemia	**0.950**	0.925	−0.025		0.708	0.925	+0.217	√
svmguide2	0.816	**0.859**	+0.043	√	0.836	0.844	+0.008	
vehicle	0.799	**0.861**	+0.061	√	0.849	0.840	−0.008	
vowel	0.837	**0.998**	+0.161	√	0.992	0.998	+0.006	
glass	0.622	0.692	+0.071	√	0.687	0.674	−0.013	

dataset	homog. polynomial kernel K^{hpol}				inhomog. polynomial kernel K^{ipol}			
	LibSVM	FkNNSVM	diff	ttest	LibSVM	FkNNSVM	diff	ttest
iris	**0.973**	0.960	−0.013		0.973	0.967	−0.007	
wine	0.966	0.989	+0.023	√	0.966	0.994	+0.028	√
leukemia	0.950	0.925	−0.025		0.950	0.925	−0.025	
svmguide2	0.816	0.841	+0.026		0.826	0.857	+0.031	√
vehicle	0.837	0.857	+0.020	√	0.847	0.848	+0.001	
vowel	0.979	0.998	+0.019	√	0.989	0.998	+0.009	√
glass	0.720	**0.720**	+0.001		0.701	0.706	+0.006	

Table 4.4: Accuracy results of LibSVM and FkNNSVM on the seven datasets of the second experiment for the four kernel functions analysed. The accuracy differences between LibSVM and FkNNSVM and the significance of the difference (using ttests) are reported. The best achieved accuracy results for each dataset are in bold. In case of multiple best results the simpler method is considered (with SVM simpler than kNNSVM and LIN kernel simpler than RBF, HPOL and IPOL kernels).

applied locally as discussed in Chapter 3.4, for kNNSVM. To assess the statistical significance of the differences between SVM and kNNSVM we use the two-tailed paired t-test ($\alpha = 0.05$) on the two sets of fold accuracies.

Results and discussion

The 10-fold CV accuracy results for the four kernels are reported in Table 4.4 with the accuracy differences between LibSVM and FkNNSVM on the same dataset and with the same kernel, and the t-tests assessing the significance of the differences.

kNNSVM performs substantially better than SVM in a considerable number of datasets without cases of significant accuracy losses. Considering all kernels, kNNSVM improves the SVM performances in 19 cases (68%) and the improvements are significant in 11 cases (39%) while for the 9 cases in which it reduces the accuracies of SVM the differences are never significant. For

kNNSVM with the LIN kernel we have 5 datasets in which kNNSVM achieves better 10-fold CV accuracies (4 significant), and 10 for the polynomial kernels (3 significant both for the HPOL kernel and the IPOL kernel). In the case of RBF kernel, we have 4 improvements but only one is significant; this is at least partially due to the fact that SVM with RBF kernel has already a high classification accuracy.

The results of this experiment substantially confirm the ones carried out on binary-class datasets in the previous experiment: kNNSVM performs significantly better than SVM if non-local kernels are used, whereas for the RBF kernel kNNSVM maintains an accuracy advantage on SVM but this is not supported by statistical significance.

We further discuss SVM and kNNSVM with RBF kernel in the next section.

4.3.3 Experiment 3: kNNSVM with RBF kernel on artificial highly non-linear datasets

Since the two experiments on a total of 32 datasets stated that kNNSVM is more accurate than SVM for linear, polynomial and RBF kernels, but the difference in the case of the RBF kernel is not statistically significant, we further investigate in this experiment the behaviour of kNNSVM and SVM with the RBF kernel using two artificial datasets.

The 2-spirals dataset.

The first toy dataset is based on the two spiral problem, a recurrent artificial benchmark problem in machine learning, see for example [149, 182]. The dataset is shown in Figure 4.1. The two classes are defined with the following function:

$$\begin{cases} x^{(1)}(t) = c \cdot t^d \cdot \sin(t) \\ x^{(2)}(t) = c \cdot t^d \cdot \cos(t) \end{cases}$$

with $d = 2.5$, $t \in [0, 10\pi]$ and using $c = 1/500$ for the first class ($y_i = +1$) and $c = -1/500$ for the second class ($y_i = -1$). The points are exampled with intervals of $\pi/30$ on the t parameter.

Figures 4.2 and 4.3 show the application of SVM and kNNSVM with RBF kernel on the 2-spirals dataset using $C = 1$ and different values for σ. In the second row of Figure 4.3, kNNSVM is applied locally varying the value of the σ parameter (using the 10-th percentile of the distances in the neighbourhoods).

Although no noise is added to the data, SVM with RBF kernel exhibits problems of under- and over-fitting whereas kNNSVM is able to find a separating function very close to the optimal one. For SVM, the under-fitting problems of large σ values are evident in the zoomed dataset with $\sigma = 1/50$ (second plot in the first row of Figure 4.2) while the over-fitting problems of low σ values are highlighted using $\sigma = 1/10000$ (first plot in the second row of Figure 4.2). Intermediate values of σ are not resolutive because, even if it is not clear from Figure 4.2, also $\sigma = 1/10000$ gives under-fitting problems in the central region; the perfect training set separation, in fact, is achievable only choosing for σ a value lower than $1/77750$.

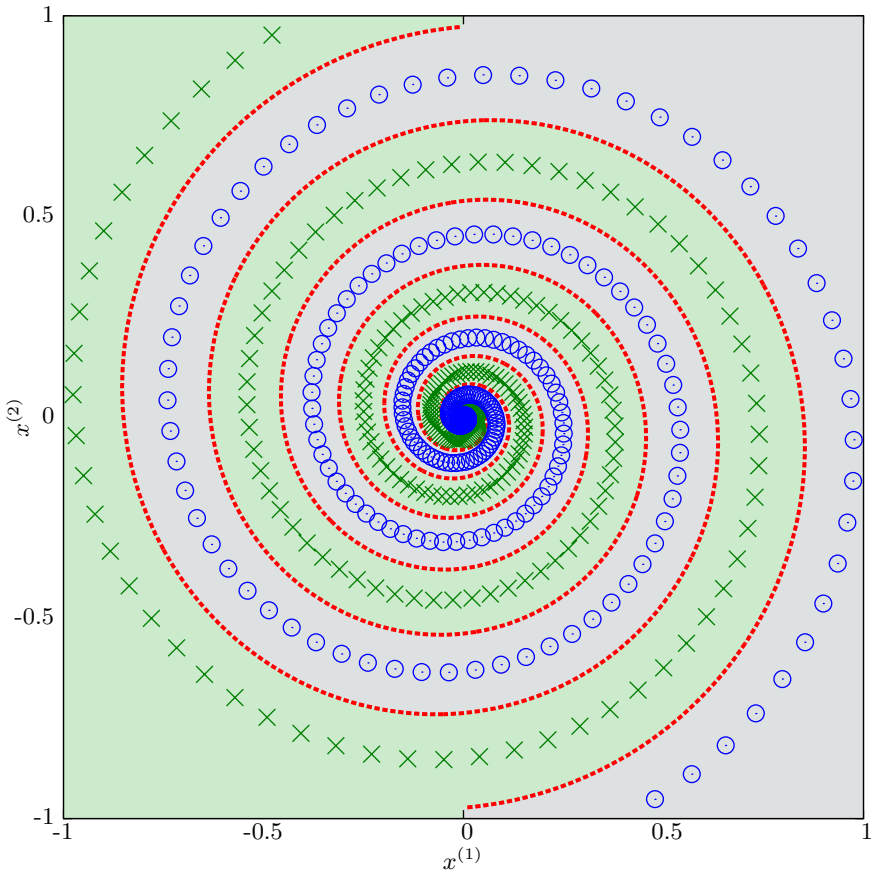
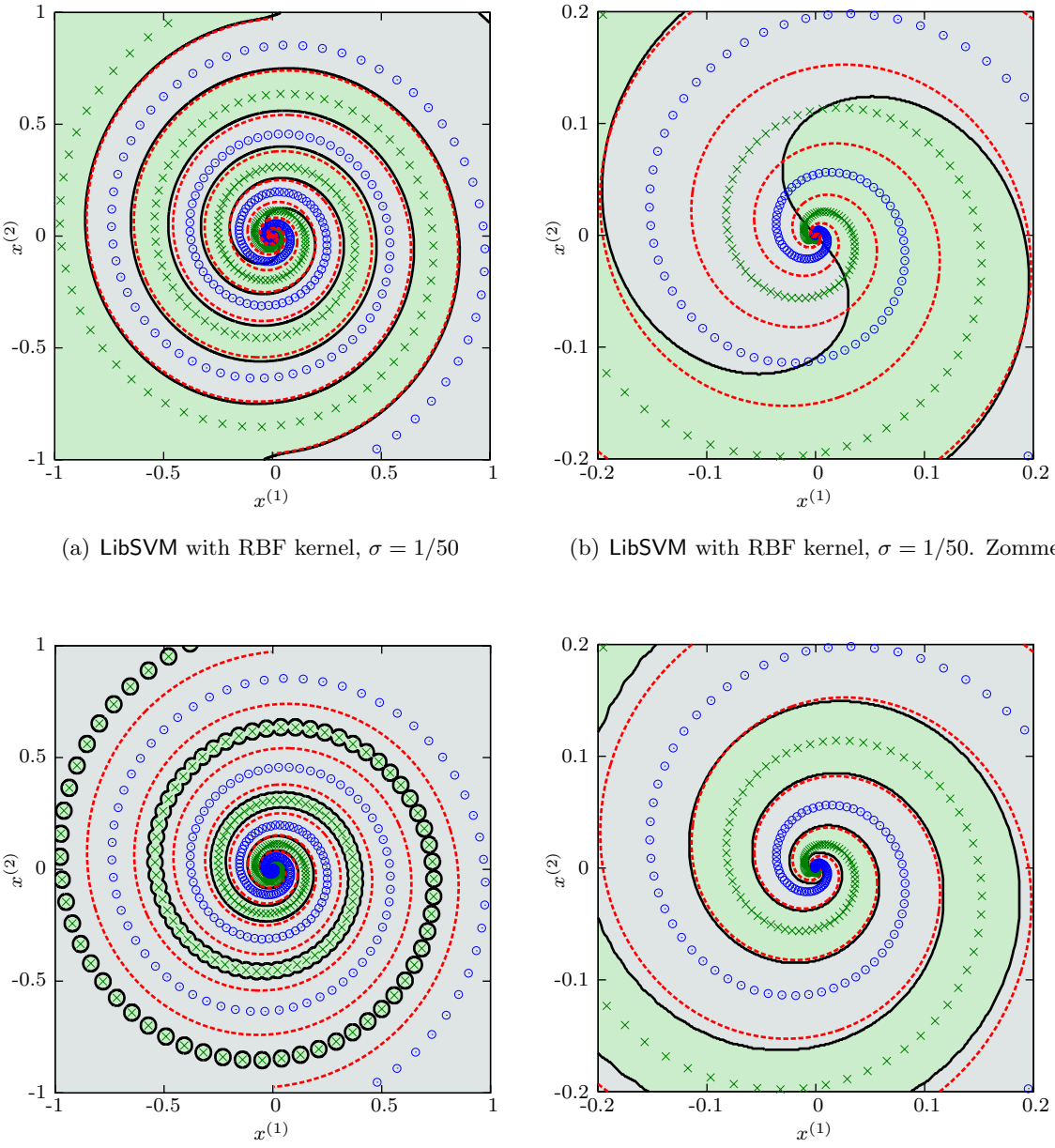


Figure 4.1: The 2-spirals artificial dataset. The two classes are represented by green crosses and blue circles, the dotted red line denotes the perfect separation.

On the contrary, kNNSVM (Figure 4.3) does not show evident over or under-fitting problems with the same σ that causes under-fitting of the central region with SVM. With the local setting of σ, kNNSVM reaches the perfect separation of the training set.

SVM is not able to find a good separation function for the 2-spirals dataset because, even using the non-linear and local RBF kernel, the feature-space mapping must make a compromise between the extreme non-linear requirements of the dataset in the central region and the simpler shapes of decision functions needed by the dataset in the peripheral regions. In other words, SVM is not able to locally adjust the parameters of the feature-space mapping. kNNSVM, on the contrary, tends to deal, in each local model, with subset of the data that are much more homogeneous and thus non very complex decision functions a required locally. If the σ parameter can be chosen locally, the adaptivity of kNNSVM to the different charactheristics of the data in different subregions is even enhanced.

We can conclude that, in this binary-class two-dimensional dataset that requires a highly non-linear decision function, kNNSVM performs substantially better than SVM. We may say that this is mainly due to the ability of kNNSVM with RBF kernel to be locally adaptive to



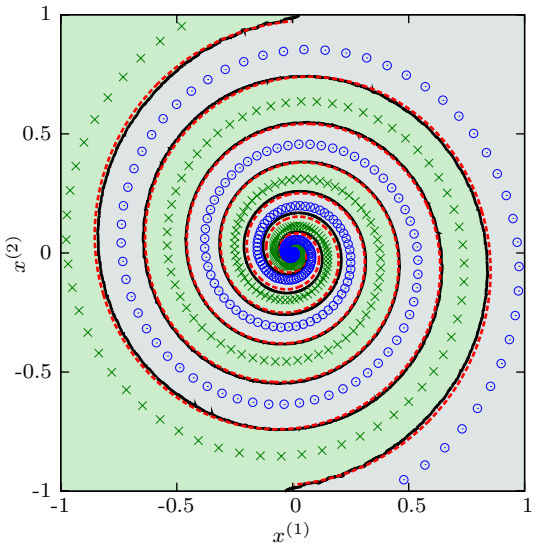
(a) LibSVM with RBF kernel, $\sigma = 1/50$

(b) LibSVM with RBF kernel, $\sigma = 1/50$. Zommed.

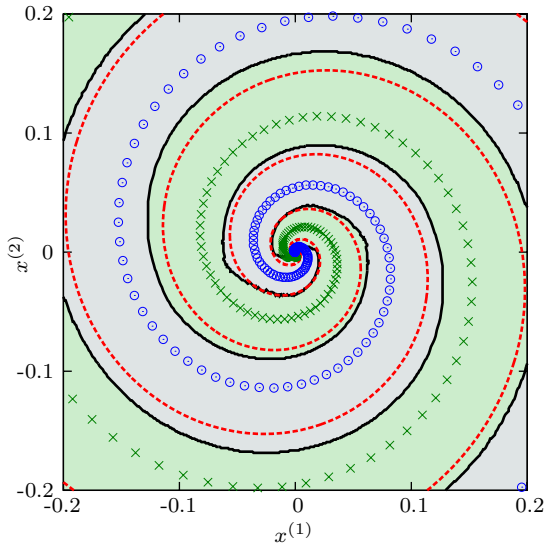
(c) LibSVM with RBF kernel, $\sigma = 1/10000$

(d) LibSVM with RBF kernel, $\sigma = 1/10000$. Zommed.

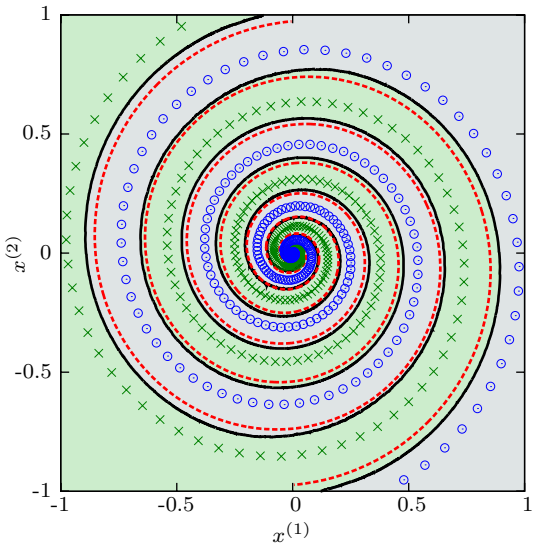
Figure 4.2: The decision function of LibSVM with RBF kernel on the 2-spirals dataset (the dotted line denotes the optimal separation). In the first row we have LibSVM with RBF kernel with $\sigma = 1/50$, in the second with $\sigma = 1/10000$. The right columns report the same classifier on the same dataset but reducing the resolution to the $[-0.2, 0.2]$ interval on both axes.



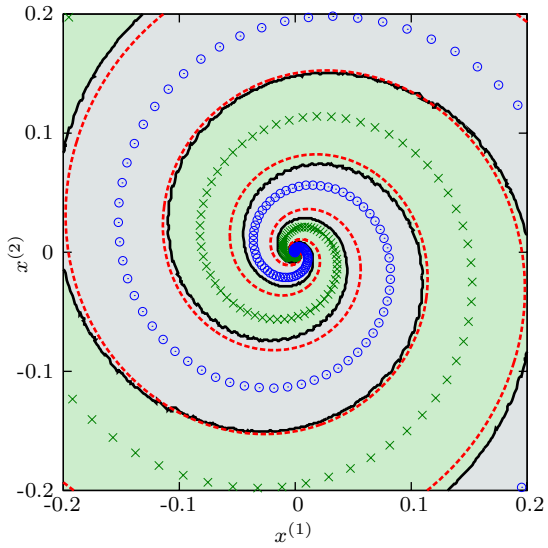
(a) FkNNSVM with RBF kernel, $\sigma = 1/50$



(b) FkNNSVM with RBF kernel, $\sigma = 1/50$. Zommed.



(c) FkNNSVM with RBF kernel, σ locally estimated



(d) FkNNSVM with RBF kernel, σ locally estimated. Zommed.

Figure 4.3: The decision function of FkNNSVM with RBF kernel on the 2-spirals dataset (the dotted line denotes the optimal separation). The first row reports FkNNSVM with $k = 100$ and $\sigma = 1/50$, the second reports FkNNSVM with $k = 100$ and σ locally set with the 0.1 percentile of the distribution of the distances between the k examples that are nearest to the testing one.

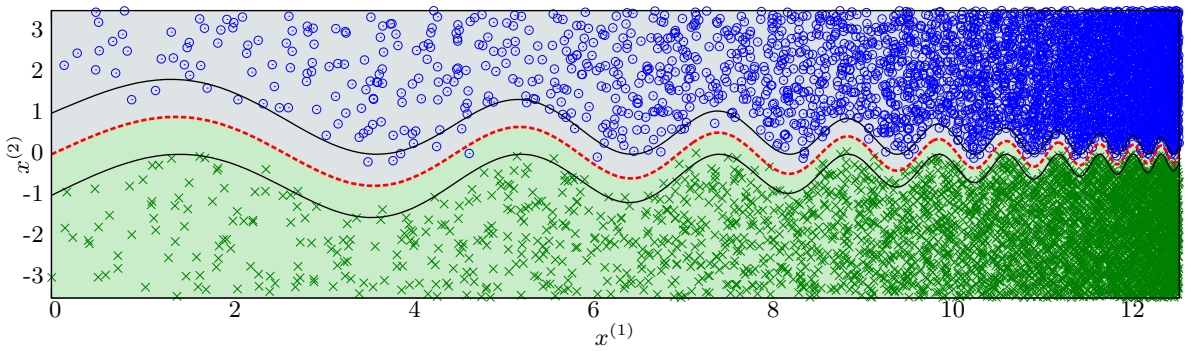


Figure 4.4: The decsin dataset. The two classes are represented by green crosses and blue circles, the black lines denote the limit of the points of the two classes without noise, the red dotted line denotes the optimal separation between the two classes.

dataset characteristics (especially if the σ parameter is estimated locally) whereas for SVM the locality is globally regulated by the σ parameter and this may cause under- or over-fitting problems.

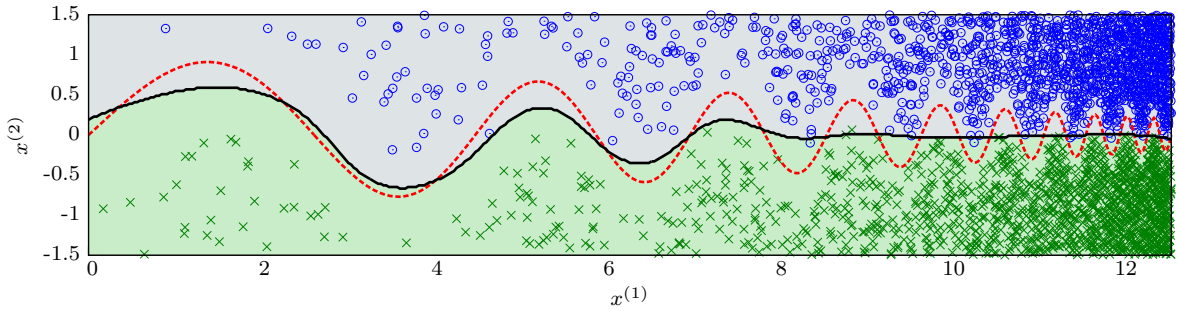
The decsin dataset

The second toy dataset (represented in Figure 4.4) is a two-feature binary-class dataset built starting from the models reported by [138] and [2] and with the following parametric function:

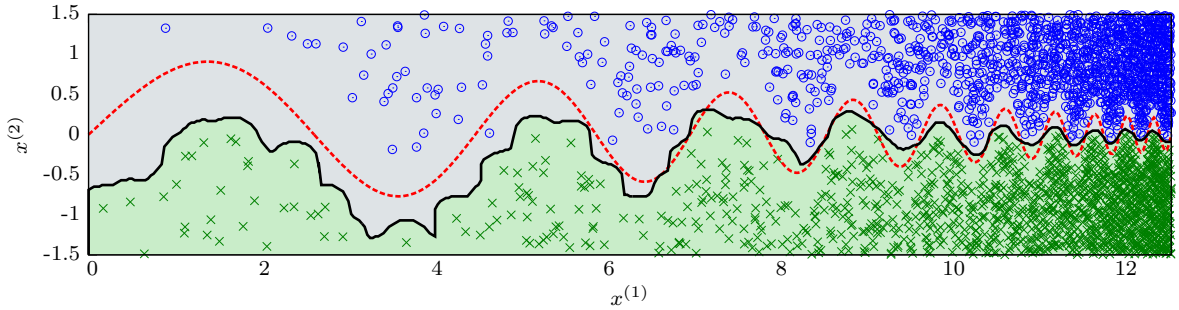
$$\begin{cases} u(t) = \dfrac{t}{1 + c \cdot t} \\ v(t) = \dfrac{\sin(t)}{1 + c \cdot t} \end{cases} \qquad c = \frac{1}{5 \cdot \pi}, \quad t \in [0, 20\pi]$$

considering $y_i = +1$ if $x_i^{(1)} = u(t)$ and $x_i^{(2)} > v(t)$, and $y_i = -1$ if $x_i^{(1)} = u(t)$ and $x_i^{(2)} < v(t)$ where $x_i^{(j)}$ denotes the j-th component of the vector $x_i = (u(t), v(t))$. The points are defined with a minimum distance of $\dfrac{1}{1 + c \cdot t}$ from $v(t)$, increase the resolution as $\dfrac{1}{1 + c \cdot t}$ on both axes and are modified by a Gaussian noise with zero mean and variance of $\dfrac{0.25}{1 + c \cdot t}$.

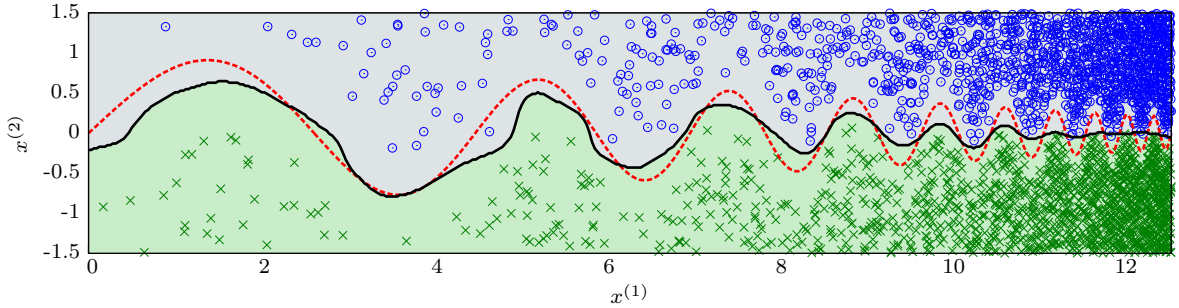
The application of SVM and kNNSVM with the RBF kernel using the local choice of σ on the decsin dataset is shown in Figure 4.5. Similarly to the case of the 2-spirals dataset of the previous section, we can notice that SVM has problems of under- or over-fitting depending on the σ parameter. In fact, if the σ parameter is too high ($\sigma = 1$, first row of Figure 4.5) the separating hyperplane is close to the optimal separation in the leftmost region of the dataset, but it reduces to a straight line in the rightmost region clearly under-fitting the data. Conversely, if the width parameter is too low ($\sigma = 1/50$, second row of Figure 4.5) there are problems of over-fitting in the leftmost region. An intermediate value of the width parameter ($\sigma = 1/10$, third row of Figure 4.5) reaches an unsatisfactory compromise because, even if the central region of



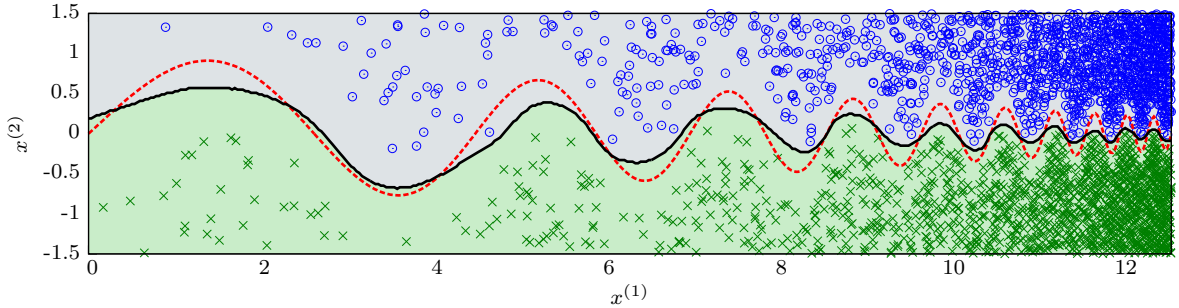
(a) SVM with RBF kernel, $\sigma = 1$



(b) SVM with RBF kernel, $\sigma = 1/50$



(c) SVM with RBF kernel, $\sigma = 1/10$



(d) kNNSVM with RBF kernel, $k = 100$, σ locally chosen

Figure 4.5: The behaviour of LibSVM and FkNNSVM with RBF kernel on the decsin dataset (reported here on the $[-1.5, 1.5]$ interval on the y axis).

the dataset is correctly separated, there are both problems of over-fitting (in the leftmost region) and under-fitting (in the rightmost region). Acting on the C parameter of SVM is not resolutive because in all the three cases the number of misclassified points is very low. kNNSVM with the local choice of σ and setting C=1 and k=100 (last row) has instead a decision function close to the optimal separation in every region of the dataset.

The experiment on the decsin dataset confirms the analysis done on the 2-spirals dataset, and thus the observation that SVM, even with the RBF kernel, cannot handle very complex decision functions with local variation of the kernel parameters.

So, even if the classification performances of kNNSVM with RBF kernel was not particularly positive for the benchmark datasets of Section 4.3.1 and Section 4.3.2, we showed here that there are cased in which it can have substantial advantages with respect to SVM with RBF kernel. From the experiments on the 2-spirals and decsin datasets, we can observe that kNNSVM is able to locally modulate the level of locality of the models permitting better results than SVM on highly non-linear datasets. The ability of locally adapt to the data characteristics of kNNSVM is due both to the local nature of the trained SVM models and to the possibility of locally estimate the width of the RBF kernel.

If we look to the bound on the risk of kNNSVM (Eq. 4.1) is is clear that for kNNSVM we are able to maintain low the complexity of the decision functions and the local training misclassification rate, differently from SVM. The intuitive and graphically evident assertion that the decision functions found by kNNSVM on the 2-spirals and decsin datasets are much better than the SVM ones, is thus also theoretically confirmed by the bound derived at the beginning of this chapter.

4.4 Conclusions

In this chapter we deepen the study of the Local SVM approach for classification, introduced by Blanzieri and Melgani [19], with the analysis of its theoretical and empirical performances with respect to the state-of-the-art kernel method for classification represented by SVM.

From the theoretical viewpoint, the new insight regards the formalisation of a bound on the generalization risks, derived from the theory of LLA [27, 194] and the local risk minimization. The bound highlights the idea that kNNSVM can lower the generalization error controlling the VC dimension of the local decision functions and the local training set misclassification rate. Local models have the advantages, in fact, that they can learn with simpler classes of decision functions (so with functions having low VC dimensions) obtaining lower empirical risk (so lower local training misclassifications). These facts theoretically enable the possibility for kNNSVM of obtaining higher classification performances with respect to SVM.

The empirical evaluation we carried out on a total of 34 datasets confirmed that kNNSVM effectively overcomes SVM on a number of cases. In particular, for binary-class datasets, kNNSVM performs statistically significantly better than SVM using the linear and polynomials kernel. The same conclusions can be drawn for multi-class datasets and one high-dimensional dataset. The advantage of kNNSVM on SVM using RBF, although detected, is not very evident and in fact

it is not statistically significant. For this reasons we further compared the behaviour of SVM and kNNSVM on two artificial problems, finding that kNNSVM performs substantially better than SVM even using a local kernel like the RBF kernel when the dataset requires highly-non linear decision functions and the characteristics of the data distribution vary locally.

The analyses of this chapter open various research directions. From one side, it seems that kNNSVM introduces an higher level of locality than SVM with RBF kernel and this is beneficial for various real world datasets as well as highly non-linear artificial problem, and thus one may think to directly introduce in kernel functions higher or different levels of locality (for example an RBF kernel that can automatically adapt its width locally). We develop this intuition with the so called Quasi-Local kernels introduced in the next chapter. From the other side, the computational complexity analysis points out that kNNSVM is not suitable for large datasets; various modifications can be introduced in order to improve the computational performances of kNNSVM (especially the prediction phase) and make it competitive or even faster than SVM. Chapter 6 details the work we carried out in this direction.

Chapter 5

Quasi-Local Kernels

In this chapter we present a family of operators that transform an arbitrary input kernel into a kernel which has a component that is local and universal in the feature-space of the input kernel. The resulting family of kernels, opportunely tuned, maintains the behaviour of the original kernel for non-local regions, while the values of the kernel increase for pairs of points that fall in a local region. In this way we aim to take advantage of both locality information and of the long-range extrapolation ability of global kernels. The strategy can alleviate the curse of dimensionality problems of the local kernels and regulate the compromise between interpolation and generalization capability.

The operators systematically map the input kernel functions into kernels that maintain the positive definite property and exploit the locality in the feature-space which is a generalization of the standard locality concept and it is central in the notion of quasi-local (QL) kernels. In such a way we are able to introduce the power of local learning techniques in the standard kernel methods framework modifying only the kernel functions and thus overcoming the computational limitation of the original formulation of local SVM. In particular, if the operators are applied on a local kernel, it turns out that the new kernel has a conceptually different notion of locality, basically similar to a local kernel with variable kernel width. We give a practical way of estimating the optimal additional parameters introduced in the resulting kernel functions starting from the optimized input kernel and the penalty parameter of SVM.

The chapter is organized as follows. In the next section we present the new family of operators that produces QL kernels and we analyse them from different viewpoints. The artificial example presented in Section 5.2 illustrates intuitively how the QL kernels work. In section 5.3 we propose a first experiment on 23 datasets with the double purpose of investigating the classification performance and identifying the most suitable QL operators. The most promising QL kernels are applied in the experiment of section 5.4 to 20 large classification datasets. Finally, in section 5.6, we draw some conclusions. The content of this chapter is also available in [167].

5.1 Operators that Transform Kernels into Quasi-Local Kernels

In this section we define the operators we use to integrate the locality information into existing kernels obtaining QL kernels. We first introduce the framework of operators on kernel, then the QL operators discussing their properties, definition, intuitive meaning and strategies to select their parameters.

5.1.1 Operators on Kernels

An operator on kernels, generically denoted as $\mathcal{O}$, is a function that accepts a kernel as input and transforms it into another kernel, i.e. $\mathcal{O}$ is an operator on kernels if $\mathcal{O}\,K$ is a kernel (supposing that K is a kernel). More formally:

Definition 3 (Operators on kernels)**.** *Denoting with l_p a (possibly empty) list of parameters that can be real constants and real-valued functions and with l_K a (possibly empty) a-priori fixed-length list of PD kernels, $\mathcal{O}_{l_p}$ is an operator on kernels if $K(\mathbf{x},\mathbf{x}') = (\mathcal{O}_{l_p}\, l_K)(\mathbf{x},\mathbf{x}')$ with $\mathbf{x},\mathbf{x}' \in \mathcal{X}$ is positive definite for every choice of PD kernels in l_K.*

An example of operator with an empty list of kernels that we can define is $(\mathcal{O}^{mul}_f)(\mathbf{x},\mathbf{x}') := f(\mathbf{x})f(\mathbf{x}')$ which is a PD kernel for every real-valued function f. Also the identity function can be thought of as an operator on kernel such that $(\mathcal{I}\,K)(\mathbf{x},\mathbf{x}') = K(\mathbf{x},\mathbf{x}')$.

The properties of Proposition 1, introduced in Section 3.3.2, can be translated (using the same assumptions) in the operator formalism as:

1. $(\mathcal{O}^{+}[K_1,K_2])(\mathbf{x},\mathbf{x}') := K_1(\mathbf{x},\mathbf{x}') + K_2(\mathbf{x},\mathbf{x}')$
2. $(\mathcal{O}^{\times}_c K_1)(\mathbf{x},\mathbf{x}') := c \cdot K_1(\mathbf{x},\mathbf{x}')$
3. $(\mathcal{O}^{\times}[K_1,K_2])(\mathbf{x},\mathbf{x}') := K_1(\mathbf{x},\mathbf{x}') \cdot K_2(\mathbf{x},\mathbf{x}')$
4. $(\mathcal{O}^{p}_\alpha K_1)(\mathbf{x},\mathbf{x}') := pol^{+}_\alpha(K_1(\mathbf{x},\mathbf{x}'))$
5. $(\mathcal{O}^{e} K_1)(\mathbf{x},\mathbf{x}') := \exp(K_1(\mathbf{x},\mathbf{x}'))$
6. $(\mathcal{O}^{f}_\psi K_1)(\mathbf{x},\mathbf{x}') := K_3(\psi(\mathbf{x}),\psi(\mathbf{x}'))$

The operators can be applied on kernels produced by other operators. For example, applying the kernel trick for distances, the RBF kernel can be defined through the introduced operators starting only from the linear kernel K^{lin}:

$$
\begin{aligned}
K^{rbf} &= \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{\sigma}\right) = \\
&= \exp\left(-\frac{\langle\mathbf{x},\mathbf{x}\rangle + \langle\mathbf{x}',\mathbf{x}'\rangle - 2\langle\mathbf{x},\mathbf{x}'\rangle}{\sigma}\right) = \\
&= \exp\left(-\frac{\langle\mathbf{x},\mathbf{x}\rangle}{\sigma}\right)\exp\left(-\frac{\langle\mathbf{x}',\mathbf{x}'\rangle}{\sigma}\right)\exp\left(\frac{2\langle\mathbf{x},\mathbf{x}'\rangle}{\sigma}\right) \\
&= (\mathcal{O}^{\times}[\mathcal{O}^{mul}_f, \mathcal{O}^{e}\,\mathcal{O}^{\times}_{2/\sigma}\,K^{lin}])(\mathbf{x},\mathbf{x}') \qquad \text{with } f(\mathbf{x}) = \exp\left(\frac{-\langle\mathbf{x},\mathbf{x}\rangle}{\sigma}\right).
\end{aligned}
$$

Using the operators we can thus prove the PD property of a kernel rewriting it starting from known PD kernels applying only operators on kernels.

In this chapter we focus on a particular class of operators, introduced below, producing the so-called Quasi-Local kernels.

5.1.2 Operators for Quasi-Local Kernels

Our operators produce kernels that we call *quasi-local* kernels, combining the input kernel with another kernel based on the distance in the feature-space of the input kernel. The formal definition of quasi-locality will be discussed in Section 5.1.6 but basically the class of QL kernels comprises those kernels that combine an input kernel with a kernel which is local in the feature-space of the input kernel. In the case of a global kernel as input of the operators, the intuitive effect of the *quasi-locality* of the resulting kernels is that they are not, in general, local in the sense of Definition 2 in Chapter 3 but at the same time the kernel score is significantly increased for examples that are close in the feature-space of the input kernel. In this way the kernel can take advantage from both the locality in the feature-space and the long-range extrapolation ability of the global input kernel.

We first construct a kernel to capture the locality information of any kernel function; such a family of kernels takes inspiration from the RBF kernel, substituting the Euclidean distance with the distance in the feature-space.

$$K^{exp}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\Phi(\mathbf{x}) - \Phi(\mathbf{x}')||^2}{\sigma}\right) \quad \sigma > 0$$

where Φ is a mapping between the input space $\mathcal{H}$ and the feature-space $\mathcal{F}$. The feature-space distance $||\Phi(\mathbf{x}) - \Phi(\mathbf{x}')||$ is dependent on the choice of kernel (see Eq. (3.13)):

$$||\Phi(\mathbf{x}) - \Phi(\mathbf{x}')||^2 = K(\mathbf{x}, \mathbf{x}) + K(\mathbf{x}', \mathbf{x}') - 2 \cdot K(\mathbf{x}, \mathbf{x}').$$

The K^{exp} kernel can be obtained with the first operator, named $\mathcal{E}_\sigma$, that accepts a positive parameter σ applied on a kernel K producing $\mathcal{E}_\sigma\, K = K^{exp}$. Explicitly, the $\mathcal{E}_\sigma$ operator is defined as:

$$(\mathcal{E}_\sigma\, K)(\mathbf{x}, \mathbf{x}') = \exp\left(\frac{-K(\mathbf{x}, \mathbf{x}) - K(\mathbf{x}', \mathbf{x}') + 2K(\mathbf{x}, \mathbf{x}')}{\sigma}\right) \quad \sigma > 0. \tag{5.1}$$

Notation 1. *In this chapter σ denotes the parameter of the $\mathcal{E}$ operator and not the width of the RBF kernel as introduced in Chapter 3. For this reason, in this chapter, the RBF kernel is defined as $K^{rbf}(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma_{rbf}\|\mathbf{x} - \mathbf{x}'\|^2\right)$*

Note that $\mathcal{E}_\sigma\, K^{lin} = K^{rbf}$ so as a special case we have the RBF kernel. However, the kernels obtained with $\mathcal{E}_\sigma$ consider only the distance in the feature-space without including explicitly the input kernel. For this reason, we will see that $\mathcal{E}_\sigma\, K$ is formally not a QL kernel.

In order to overcome the limitation of $\mathcal{E}_\sigma$ which completely drops the global information, the idea is to weight the input kernel with the local information to obtain a real QL kernel. So we

include explicitly the input kernel in the output of the following operator:

$$(\mathcal{P}_\sigma K)(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}, \mathbf{x}') \cdot (\mathcal{E}_\sigma K)(\mathbf{x}, \mathbf{x}') \qquad \sigma > 0. \tag{5.2}$$

Observing that the $\mathcal{E}_\sigma K$ kernel can assume values only between 0 and 1 (since it is an exponential with negative exponent) and that the higher the distance in the feature-space between examples the lower the value of the $\mathcal{E}_\sigma K$ kernel, the idea of $\mathcal{P}_\sigma$ is to exponentially penalize the basic kernel K with respect to the feature-space distance between $\mathbf{x}$ and $\mathbf{x}'$.

An opposite possibility is to amplify the values of input kernels in the cases in which the examples contain local information. This can be done simply by adding the $\mathcal{E}_\sigma K$ kernel to the input one.

$$(\mathcal{S}_\sigma K)(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}, \mathbf{x}') + (\mathcal{E}_\sigma K)(\mathbf{x}, \mathbf{x}') \qquad \sigma > 0. \tag{5.3}$$

However, since $\mathcal{E}_\sigma$ gives kernels that can assume at most the value of 1 while the input kernel in the general case does not have an upper bound, it is reasonable to weight the $\mathcal{E}_\sigma$ operator with a constant reflecting the order of magnitude of the values that the input kernel can assume in the training set. We call this parameter η and the new operator is:

$$(\mathcal{S}_{\sigma,\eta} K)(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}, \mathbf{x}') + \eta \cdot (\mathcal{E}_\sigma K)(\mathbf{x}, \mathbf{x}') \quad \sigma > 0, \eta \geq 0. \tag{5.4}$$

A different formulation of the $\mathcal{P}_\sigma$ operator that maintains the product form but adopts the idea of amplifying the local information is:

$$(\mathcal{PS}_\sigma K)(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}, \mathbf{x}') \left[1 + (\mathcal{E}_\sigma K)(\mathbf{x}, \mathbf{x}')\right] \quad \sigma > 0, \eta \geq 0. \tag{5.5}$$

Also in this case the parameter η that controls the weight of the $\mathcal{E}_\sigma K$ kernel is introduced:

$$(\mathcal{PS}_{\sigma,\eta} K)(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}, \mathbf{x}') \left[1 + \eta \cdot (\mathcal{E}_\sigma K)(\mathbf{x}, \mathbf{x}')\right] \quad \sigma > 0, \eta \geq 0. \tag{5.6}$$

The QL kernels produced by the operators defined in Eq. 5.2 5.3, 5.4, 5.5, 5.6 are more complicated then the corresponding input kernels, since it is necessary to evaluate $K(\mathbf{x}, \mathbf{x})$, $K(\mathbf{x}', \mathbf{x}')$, $K(\mathbf{x}, \mathbf{x}')$ and to perform a couple of addition/multiplication operations and an exponentiation instead of the evaluation of $K(\mathbf{x}, \mathbf{x}')$ only. However, this is a constant computational overhead in the kernel evaluation phase, that does not affect the complexity of the SVM algorithm either in the training or in the testing phase. Moreover, it is possible to implement a variant of the dot product that computes $\langle \mathbf{x}, \mathbf{x} \rangle$, $\langle \mathbf{x}', \mathbf{x}' \rangle$, $\langle \mathbf{x}, \mathbf{x}' \rangle$ with only one traversing of $\mathbf{x}$ and $\mathbf{x}'$ vectors, or precompute and store $\langle \mathbf{x}, \mathbf{x} \rangle$ for each example in order to enhance the computational performances of the operators.

Intuitively all the kernels produces by $\mathcal{S}_\sigma$, $\mathcal{S}_{\sigma,\eta}$, $\mathcal{PS}_\sigma$ and $\mathcal{S}_{\sigma,\eta}$ (Eq. 5.2 5.3, 5.4, 5.5, 5.6) are QL since they combine the original kernel with the locality information in its feature-space. We will formalise this in Section 5.1.6, while in the following subsection we will prove that the operators preserve the PD property of the input kernel.

5.1.3 The Operators for Quasi-Local Kernels Preserve the PD Property of the Input Kernels

The introduced operators preserve the PD property of the kernels on which they are applied, as stated in the following theorem.

Theorem 1. *If K is a PD kernel, then $\mathcal{O}\,K$ with $\mathcal{O} \in \{\mathcal{E}_\sigma, \mathcal{P}_\sigma, \mathcal{S}_\sigma, \mathcal{S}_{\sigma,\eta}, \mathcal{PS}_\sigma, \mathcal{PS}_{\sigma,\eta}\}$ is a PD kernel.*

Proof. It is straightforward to see that, for a PD kernel K, all the kernels resulting from the introduced operators can be obtained using properties 1. and 3. of Proposition 1 of Chapter 3, provided that $\mathcal{E}_\sigma\,K$ is a PD kernel. So the only thing that remains to prove is that $\mathcal{E}_\sigma\,K$ is PD. Decomposing the definition of $(\mathcal{E}_\sigma\,K)(\mathbf{x},\mathbf{x}')$ into three exponential functions we obtain:

$$(\mathcal{E}_\sigma\,K)(\mathbf{x},\mathbf{x}') = \exp\left(\frac{2K(\mathbf{x},\mathbf{x}')}{\sigma}\right)\exp\left(\frac{-K(\mathbf{x},\mathbf{x})}{\sigma}\right)\exp\left(\frac{-K(\mathbf{x}',\mathbf{x}')}{\sigma}\right)$$

that can be written as:

$$(\mathcal{E}_\sigma\,K)(\mathbf{x},\mathbf{x}') = (\mathcal{O}^e\,2K/\sigma)(\mathbf{x},\mathbf{x}')\cdot f(\mathbf{x})f(\mathbf{x}')$$

where $\mathcal{O}^e\,2K/\sigma$ is the exponentiation of the $2K/\sigma$ kernel, and f is a real valued function such that $f(\mathbf{x}) = \exp(-K(\mathbf{x},\mathbf{x})/\sigma)$. The first term is the exponentiation of a kernel multiplied by a non-negative constant and, since the kernel exponentiation can be seen as the limit of the series expansion of the exponential function which is the infinite sum of polynomial kernels, for property 4. of Proposition 1 of Chapter 3 we conclude that $\mathcal{O}^e\,2K/\sigma$ is a PD kernel. Moreover, recalling from the definition of PD kernels, that the product $f(\mathbf{x})f(\mathbf{x}')$ is a PD kernel for all the real-valued functions f defined in the input space [54] we conclude that $\mathcal{E}_\sigma\,K$ is a PD kernel. □

Obviously, if the input of $\mathcal{E}_\sigma$ is not a PD kernel, also the resulting function cannot be, in the general case, a PD kernel since the exponentiation operator maintains the PD property only for PD kernels. So, in the case of the sigmoidal kernel as input kernel, the resulting kernel is still not ensured to be PD.

5.1.4 Properties of the Operators

In order to understand how the operators modify the original feature-space of the input kernel we study the distances in the feature-space of the quasi-local kernels. The new feature-space introduced by kernels produced by the operators is denoted with $\mathcal{F}_\mathcal{O}$, the corresponding mapping function with $\Phi_\mathcal{O}$ and the distance between two input points mapped in $\mathcal{F}_\mathcal{O}$ with $dist_{\mathcal{F}_\mathcal{O}}(\mathbf{x},\mathbf{x}') = m(\Phi_\mathcal{O}(\mathbf{x}),\Phi_\mathcal{O}(\mathbf{x}'))$ where m is a metric in $\mathcal{F}_\mathcal{O}$. Applying the kernel trick for distances, we can express the squared distances in $\mathcal{F}_\mathcal{O}$ as:

$$dist^2_{\mathcal{F}_\mathcal{O}}(\mathbf{x},\mathbf{x}') = \|\Phi_\mathcal{O}(\mathbf{x}) - \Phi_\mathcal{O}(\mathbf{x}')\|^2 = (\mathcal{O}\,K)(\mathbf{x},\mathbf{x}) + (\mathcal{O}\,K)(\mathbf{x}',\mathbf{x}') - 2(\mathcal{O}\,K)(\mathbf{x},\mathbf{x}'). \quad (5.7)$$

For $\mathcal{O} = \mathcal{E}_\sigma$, since it is clear that $dist_{\mathcal{F}}(\mathbf{x}, \mathbf{x}) = 0$ for every $\mathbf{x}$, we can derive $dist_{\mathcal{F}_{\mathcal{E}_\sigma}}$ as follows:

$$\begin{aligned} dist^2_{\mathcal{F}_{\mathcal{E}_\sigma}}(\mathbf{x}, \mathbf{x}') &= \exp\left(-\frac{dist^2_{\mathcal{F}}(\mathbf{x}, \mathbf{x})}{\sigma}\right) + \exp\left(-\frac{dist^2_{\mathcal{F}}(\mathbf{x}', \mathbf{x}')}{\sigma}\right) + \\ &\quad -2\exp\left(-\frac{dist^2_{\mathcal{F}}(\mathbf{x}, \mathbf{x}')}{\sigma}\right) = \qquad (5.8) \\ &= 2\left[1 - exp\left(-\frac{dist^2_{\mathcal{F}}(\mathbf{x}, \mathbf{x}')}{\sigma}\right)\right]. \end{aligned}$$

Note that $dist^2_{\mathcal{F}_{\mathcal{E}_\sigma}}(\mathbf{x}, \mathbf{x}') \leq 2$ for every pair of examples, and so the distances in $\mathcal{F}_{\mathcal{E}_\sigma}$ are bounded even if they are not bounded in $\mathcal{F}$.

Substituting $\mathcal{P}_\sigma$, $\mathcal{S}_{\sigma,\eta}$ and $\mathcal{PS}_{\sigma,\eta}$ in Eq. 5.7, an taking into account Eq. 5.8, the distances in $\mathcal{F}_{\mathcal{O}}$ for the quasi-local kernels are:

$$dist^2_{\mathcal{F}_{\mathcal{P}_\sigma}}(\mathbf{x}, \mathbf{x}') = dist^2_{\mathcal{F}}(\mathbf{x}, \mathbf{x}') + K(\mathbf{x}, \mathbf{x}')\, dist^2_{\mathcal{F}_{\mathcal{E}_\sigma}}(\mathbf{x}, \mathbf{x}'); \qquad (5.9)$$

$$dist^2_{\mathcal{F}_{\mathcal{S}_{\sigma,\eta}}}(\mathbf{x}, \mathbf{x}') = dist^2_{\mathcal{F}}(\mathbf{x}, \mathbf{x}') + \eta \cdot dist^2_{\mathcal{F}_{\mathcal{E}_\sigma}}(\mathbf{x}, \mathbf{x}'); \qquad (5.10)$$

$$\begin{aligned} dist^2_{\mathcal{F}_{\mathcal{PS}_{\sigma,\eta}}}(\mathbf{x}, \mathbf{x}') &= (1+\eta)\, dist^2_{\mathcal{F}}(\mathbf{x}, \mathbf{x}') + \eta \cdot K(\mathbf{x}, \mathbf{x}')\, dist^2_{\mathcal{F}_{\mathcal{E}_\sigma}}(\mathbf{x}, \mathbf{x}') = \qquad (5.11) \\ &= dist^2_{\mathcal{F}}(\mathbf{x}, \mathbf{x}') + \eta \cdot dist^2_{\mathcal{F}_{\mathcal{P}_\sigma}}(\mathbf{x}, \mathbf{x}'). \qquad (5.12) \end{aligned}$$

We can notice that the distances in $\mathcal{F}_{\mathcal{E}_\sigma}$ and in $\mathcal{F}_{\mathcal{S}_{\sigma,\eta}}$ do not contain explicitly the kernel function but they are based only on the distances in $\mathcal{F}$. So we can further analyse the behaviour of the distances in $\mathcal{F}_{\mathcal{E}_\sigma}$ and $\mathcal{F}_{\mathcal{S}_{\sigma,\eta}}$ with the following proposition.

Proposition 2. *The operators $\mathcal{E}_\sigma$ and $\mathcal{S}_{\sigma,\eta}$ preserve the ordering on distances in $\mathcal{F}$. Formally*

$$dist_{\mathcal{F}}(\mathbf{x}, \mathbf{x}') < dist_{\mathcal{F}}(\mathbf{x}, \mathbf{x}'') \Rightarrow dist_{\mathcal{F}_{\mathcal{O}}}(\mathbf{x}, \mathbf{x}') < dist_{\mathcal{F}_{\mathcal{O}}}(\mathbf{x}, \mathbf{x}'')$$

for $\mathcal{O} \in \{\mathcal{E}_\sigma, \mathcal{S}_{\sigma,\eta}\}$ and for every example $\mathbf{x}, \mathbf{x}', \mathbf{x}''$.

Proof. It follows directly from the observations that $dist_{\mathcal{F}_{\mathcal{E}_\sigma}}(\mathbf{x}, \mathbf{x}')$ and $dist_{\mathcal{F}_{\mathcal{S}_{\sigma,\eta}}}(\mathbf{x}, \mathbf{x}')$ are defined with strictly increasing monotonic functions, Eq. 5.9 and Eq. 5.10 respectively, and that $dist_{\mathcal{F}}$ is always non-negative. □

This means that $\mathcal{E}_\sigma\, K$ kernel determines the same neighbourhoods as K and that the $\mathcal{E}_\sigma\, K$ exploits the locality information weighting the influence of the neighbours of a point in the feature-space of K maintaining the property that points at distance d in the feature-space of K

influence the $\mathcal{E}_\sigma K$ kernel score more than any other more distant points. In other words $\mathcal{E}_\sigma K$ modifies the influence of the points using the features space distances but the ordering on the weights is the same of the ordering on distances in the input space.

The $\mathcal{E}_\sigma K$ kernel has also an interesting property regarding the class of universal kernels (see Chapter 3.3). Roughly speaking, universal kernels, introduced in [176] and further discussed in [177, 180, 127], are kernels that permit to optimally approximate the Bayes decision rule or, equivalently, to learn an arbitrary continuous function uniformly on any compact subset of the input space. Applying Proposition 8 and Corollary 10 in [176], it turns out that $\mathcal{E}_\sigma K$ is universal in the feature-space of K. Intuitively this happens because $\mathcal{E}_\sigma K$ builds a K^{rbf} kernel, which is universal, in the feature-space of K. This means that, regardless of the universality of the input kernel, the $\mathcal{E}_\sigma$ always finds a space on which the resulting kernel is universal.

5.1.5 Connections between $\mathcal{E}_\sigma\ K^{rbf}$ and K^{rbf} with Variable Kernel Width

Since K^{rbf} is a local kernel, a question that naturally arises concerns the behaviour of $\mathcal{E}_\sigma\ K^{rbf}$, i.e. the quasi-local transformation of a local kernel. In particular, the point is to understand if K^{rbf} and $\mathcal{E}_\sigma\ K^{rbf}$ exploit the same notion of locality. If it is the case, this would mean that $\mathcal{E}_\sigma\ K^{rbf}$ and K^{rbf} are basically equivalent and identify the same features space, possibly under certain parameter settings. This question is addressed by the following Proposition.

Proposition 3. *There not exist two constant $\sigma, \gamma^{rbf} \in \mathbb{R}$ with $\sigma > 0$ and $\gamma \geq 0$, such that, for every $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ with $\mathcal{X}$ with at least 3 distinct points, the following holds:*

$$K^{rbf}(\mathbf{x}, \mathbf{x}') = (\mathcal{E}_\sigma\ K^{rbf})(\mathbf{x}, \mathbf{x}') \tag{5.13}$$

Proof. Suppose, by contradiction, that there exist $\sigma, \gamma^{rbf} \in \mathbb{R}$ such that, for every $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, Eq. 5.13 holds. It can be rewritten as:

$$\exp(-\gamma^{rbf} \cdot \|\mathbf{x} - \mathbf{x}'\|^2) =$$

$$= \exp\left(-\frac{\exp(-\gamma^{rbf} \cdot \|\mathbf{x} - \mathbf{x}\|^2) + \exp(-\gamma^{rbf} \cdot \|\mathbf{x}' - \mathbf{x}'\|^2) - 2 \cdot \exp(-\gamma^{rbf} \cdot \|\mathbf{x} - \mathbf{x}'\|^2)}{\sigma}\right)$$

Since $\exp(-\gamma^{rbf} \cdot \|\mathbf{x} - \mathbf{x}\|^2) = 1$, we can obtain:

$$-\gamma^{rbf} \cdot \|\mathbf{x} - \mathbf{x}'\|^2 = \frac{-2 + 2 \cdot \exp(-\gamma^{rbf} \cdot \|\mathbf{x} - \mathbf{x}'\|^2)}{\sigma},$$

from which we have

$$\exp(-\gamma^{rbf} \cdot \|\mathbf{x} - \mathbf{x}'\|^2) = 1 - \frac{\gamma^{rbf}\sigma}{2} \cdot \|\mathbf{x} - \mathbf{x}'\|^2$$

that can be written as:

$$K^{rbf}(\mathbf{x}, \mathbf{x}') = 1 - \frac{\gamma^{rbf}\sigma}{2} \cdot \|\mathbf{x} - \mathbf{x}'\|^2.$$

Since, with respect to the square of the Euclidean distance $\|\mathbf{x}-\mathbf{x}'\|^2$, $K^{rbf}(\mathbf{x}, \mathbf{x}')$ is a negative exponential function, whereas $1 - \|\mathbf{x} - \mathbf{x}'\|^2 \cdot \frac{\gamma^{rbf}\sigma}{2}$ is a non-increasing linear function, the two

function can have no more than 2 points in common. Because σ and γ^{rbf} are constant, while $\|\mathbf{x}-\mathbf{x}'\|^2$ is not constant, it is straightforward to conclude that $K^{rbf}(\mathbf{x},\mathbf{x}') \neq 1-\|\mathbf{x}-\mathbf{x}'\|^2 \cdot \frac{\gamma^{rbf}\sigma}{2}$ at least for some $\mathbf{x},\mathbf{x}' \in \mathcal{X}$. In this way we get a contradiction thus proving the proposition. □

From this proposition we can conclude that $\mathcal{E}_\sigma\ K^{rbf}$ cannot be emulated by K^{rbf} and thus it introduces an higher degree of locality. Intuitively an increased level of locality can be introduced locally adjusting the local parameters. In the specific case of K^{rbf} this intuition can be applied permitting to the width parameter $(1/\gamma^{rbf})$ to be locally adaptive, as proposed for example in [39]. The following proposition demonstrate that $\mathcal{E}_\sigma\ K^{rbf}$ is equivalent to K^{rbf} with variable kernel width.

Proposition 4. *There exists a real-valued function $f(\sigma,\gamma^{rbf},\|\mathbf{x}-\mathbf{x}'\|)$ such that the following holds for each $\mathbf{x},\mathbf{x}' \in \mathcal{X}$:*

$$\exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{f(\sigma,\gamma^{rbf},\|\mathbf{x}-\mathbf{x}'\|)}\right) = (\mathcal{E}_\sigma\ K^{rbf})(\mathbf{x},\mathbf{x}') \tag{5.14}$$

Proof. We can easily find such function f isolating it from Eq. 5.14:

$$\exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{f(\sigma,\gamma^{rbf},\|\mathbf{x}-\mathbf{x}'\|)}\right) = \exp\left(\frac{-2+2\cdot\exp(-\gamma^{rbf}\cdot\|\mathbf{x}-\mathbf{x}'\|^2)}{\sigma}\right)$$

obtaining:

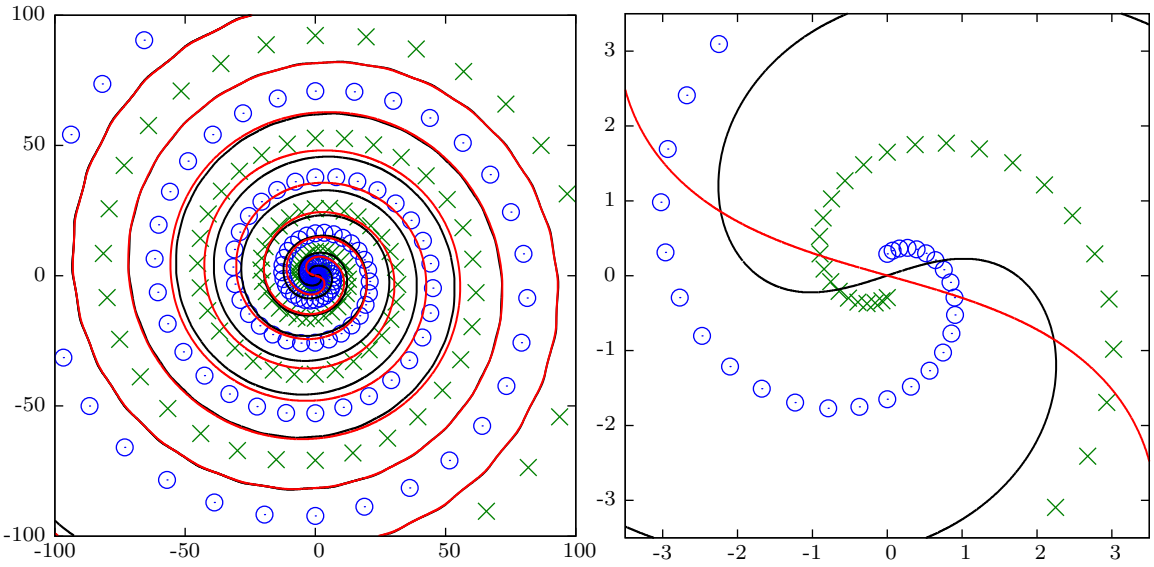
$$f(\sigma,\gamma^{rbf},\|\mathbf{x}-\mathbf{x}'\|) = \frac{\sigma}{2}\cdot\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{1-\exp(-\gamma^{rbf}\cdot\|\mathbf{x}-\mathbf{x}'\|^2)}. \tag{5.15}$$

□

We thus found the function regulating the variable K^{rbf} width. It can be shown that Eq. 5.15 has always positive derivative, meaning that it always grows as the distance between examples grows. This causes the kernel width to be lower for close points and higher for distant points, thus permitting to alleviate the tradeoff between over- and under-fitting on which a uniform kernel width is based. The variable kernel width is particularly crucial in presence of data with uneven densities.

We illustrate these considerations with the application of K^{rbf} and $\mathcal{E}_\sigma\ K^{rbf}$ on the 2-spirals artificial dataset[1] shown in Figure 5.1. Both K^{rbf} and $\mathcal{E}_\sigma\ K^{rbf}$ are applied with the best parameters obtained with a grid search 20-fold CV on $C,\gamma^{rbf},\sigma \in \{2^{-10},2^{-9},\ldots,2^{9},2^{10}\}$. The best training accuracy of K^{rbf} is 0.823 whereas $\mathcal{E}_\sigma\ K^{rbf}$ reaches 0.907, meaning that the quasi-local kernel approach is able to find a better decision function. This is evident also graphically, in fact, while in the peripheral regions of the datasets (see Figure 5.1(a)) both classifiers find a good decision function, whereas in the central region (see Figure 5.1(b)) K^{rbf} starts to clearly underfit the data.

[1] a scaled version of this dataset has been presented in Chapter 4.



(a) The whole 2-spirals dataset (b) The 2-spirals dataset zoomed.

Figure 5.1: The behaviour of SVM with K^{rbf} (the red line) and $\mathcal{E}_\sigma\ K^{rbf}$ (the black line) on the 2-spirals problem (a scaled version of the dataset presented in Chapter 4) where the examples of the two classes are denoted by green crosses and blue circles. The model parameters are obtained with a 20-fold CV grid search. The best training set accuracy is 0.823 for K^{rbf} and 0.907 for $\mathcal{E}_\sigma\ K^{rbf}$.

5.1.6 Formal Definition of Quasi-Local Kernels

In this section, we formally introduce the notion of quasi-local kernels, and we show that kernels produced by the $\mathcal{S}_\sigma$, $\mathcal{S}_{\sigma,\eta}$, $\mathcal{PS}_\sigma$ and $\mathcal{S}_{\sigma,\eta}$ are quasi-local kernels. Firstly, we introduce the concept of locality with respect to a function:

Definition 4. *Given a PD kernel K with implicit mapping function $\Phi : \mathbb{R}^p \mapsto \mathcal{F}$ (namely $K(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$), and a function $\Psi : \mathbb{R}^p \mapsto \mathcal{F}_\Psi$, K* is local with respect to Ψ *if there exists a function $\Omega : \mathcal{F}_\Psi \mapsto \mathcal{F}$ such that the following holds:*

1. $\langle \Phi(\mathbf{x}), \Phi(\mathbf{x}_i) \rangle = \langle \Omega(\Psi(\mathbf{x})), \Omega(\Psi(\mathbf{x}_i)) \rangle$ *for all* $\mathbf{x}, \mathbf{x}_i \in \mathbb{R}^p$
2. $\lim_{\|u - v_i\|_{\mathcal{F}_\Psi} \to \infty} \langle \Omega(u), \Omega(v_i) \rangle = c_i$ *with* $u = \Psi(\mathbf{x})$, $v_i = \Psi(\mathbf{x}_i)$ *for some* $\mathbf{x}, \mathbf{x}_i \in \mathbb{R}^p$ *and* c_i *constant and not depending on* u.

In other terms, the notion of locality referred to examples in input space (Definition 2, Chapter 3), is modified here in order to consider the locality in any space accessible from the input space through a corresponding mapping function. Notice that, as particular cases, we

have that every local kernel is local with respect to the identity function and with respect to its own implicit mapping function.

With the next theorem we see that the $\mathcal{E}_\sigma$ formally respect the idea of producing kernels that are local with respect to the feature-space of the input kernel.

Theorem 2. *If K is a PD kernel with the implicit mapping function $\Phi : \mathbb{R}^p \mapsto \mathcal{F}$, then $\mathcal{E}_\sigma\, K$ is local with respect to Φ.*

Proof. We have already shown that $\mathcal{E}_\sigma\, K$ is a PD kernel given that K is a PD kernel (see Theorem 1). It remains to show that $\mathcal{E}_\sigma\, K$ is local with respect to Φ.

First we need to show that (Definition 4 point 1), denoted with $\Phi' : \mathbb{R}^p \mapsto \mathcal{F}'$ the implicit mapping function of $\mathcal{E}_\sigma\, K$, there exists a function $\Omega : \mathcal{F} \mapsto \mathcal{F}'$ such that $\Phi'(\mathbf{x}) = \Omega(\Phi(\mathbf{x}))$. Taking as $\Omega : \mathcal{F} \mapsto \mathcal{F}'$ the implicit mapping of the kernel $\exp\left(-\frac{\|u-v_i\|}{\sigma}\right)$ with $u = \Phi(\mathbf{x})$, $v_i = \Phi(\mathbf{x}_i)$ with $\mathbf{x}, \mathbf{x}_i \in \mathbb{R}^p$ we have

$$\langle \Omega(u), \Omega(v_i) \rangle = \exp\left(-\frac{\|u - v_i\|}{\sigma}\right). \tag{5.16}$$

Using the hypothesis on u and v_i it becomes:

$$\exp\left(-\frac{\|\Phi(\mathbf{x}) - \Phi(\mathbf{x}_i)\|}{\sigma}\right) = \langle \Omega(\Phi(\mathbf{x})), \Omega(\Phi(\mathbf{x}_i)) \rangle. \tag{5.17}$$

The implicit mapping function of $\mathcal{E}_\sigma\, K$ is Φ' and so

$$\langle \Phi'(\mathbf{x}), \Phi'(\mathbf{x}_i) \rangle = (\mathcal{E}_\sigma\, K)(\mathbf{x}, \mathbf{x}_i) \tag{5.18}$$

Moreover since $(\mathcal{E}_\sigma\, K)(\mathbf{x}, \mathbf{x}_i) = \exp\left(-\frac{\|\Phi(\mathbf{x}) - \Phi(\mathbf{x}_i)\|}{\sigma}\right)$ for definition of $\mathcal{E}_\sigma$ (see Eq. 5.1), substituting Eq. 5.17 into Eq. 5.18 we conclude that

$$\langle \Phi'(\mathbf{x}), \Phi'(\mathbf{x}_i) \rangle = \langle \Omega(\Phi(\mathbf{x})), \Omega(\Phi(\mathbf{x}_i)) \rangle.$$

Second, we need to show that (Definition 4 point 2) $\langle \Omega(u), \Omega(v_i) \rangle \to c_i$ with c_i constant for $\|\Omega(u) - \Omega(v_i)\| \to \infty$. From the Eq. 5.16, it is clear that, as the distance between $u = \Phi(\mathbf{x})$ and $v_i = \Phi(\mathbf{x}_i)$ tend to infinity, the kernel value is equal to the constant 0 regardless of $\mathbf{x}$. □

Now we can define the quasi-locality property of a kernel.

Definition 5 (Quasi-local kernel)**.** *A PD kernel K is a quasi-local kernel if $K = f(K^{inp}, K^{loc})$ where K^{inp} is a PD kernel with implicit mapping function $\Phi : \mathbb{R}^p \mapsto \mathcal{F}$, K^{loc} is a PD kernel which is local with respect to Φ and f is a function involving legal and non trivial operations on PD kernels.*

For legal operations on kernels we mean operations preserving the PD property. For non trivial operations we intend operations that always maintain the influence of all the input kernels in the output kernel; more precisely a function $f(K_1, K_2)$ does not introduce trivial operations

if there exists two kernels K' and K'' such that $f(K', K_2) \neq f(K_1, K_2)$ and $f(K_1, K'') \neq f(K_1, K_2)$. Notice that the K^{inp} kernel of the definition corresponds to the input kernel of the operator that produces the quasi-local kernel K.

Theorem 3. *If K is a PD kernel, then $\mathcal{S}_\sigma\, K$, $\mathcal{S}_{\sigma,\eta}\, K$, $\mathcal{PS}_\sigma\, K$ and $\mathcal{S}_{\sigma,\eta}\, K$ are quasi-local kernels.*

Proof. Theorem 2 already states that $\mathcal{E}_\sigma\, K$ is a PD kernel which is local with respect to the implicit mapping function Φ of the kernel K which is PD for hypothesis. It is easy to see that all the kernels resulting from the introduced operators can be obtained using properties 1. and 3. of Proposition 1 starting from the two PD kernels K and $\mathcal{E}_\sigma\, K$, and thus $\mathcal{S}_\sigma\, K$, $\mathcal{S}_{\sigma,\eta}\, K$, $\mathcal{PS}_\sigma\, K$ and $\mathcal{S}_{\sigma,\eta}\, K$ are PD kernels obtained with legal operations. Moreover, the properties 1. and 3. of Proposition 1 introduce multiplications and sums between kernels and between kernels and constant. The sums introduced by the operators are always non trivial because they always consider positive addends, and so it is for the multiplications because they never consider null factors (the introduced constants are non null for definition). □

Both quasi-local kernels and kNNSVM classifiers are based on the notion of locality in the feature-space. However, two main theoretical differences can be found between them. The first is that in kNNSVM locality is included directly, considering only the points that are close to the testing point, while for the quasi-local kernels the information of the input kernel is balanced with the local information. The second consideration is that kNNSVM has a variable but hard boundary between the local and non local points, while $\mathcal{S}_{\sigma,\eta}$ and $\mathcal{PS}_{\sigma,\eta}$ produce kernels whose locality decreases exponentially but in a continuous way.

5.1.7 Parameter Choice and Empirical Risk Minimization for QL Kernels

There are two parameters for the operators on kernels through which we obtain the quasi-local kernels: σ, which is present in $\mathcal{E}_\sigma$ and consequently in all the operators, and η, which is present in $\mathcal{S}_{\sigma,\eta}$ and $\mathcal{PS}_{\sigma,\eta}$ ($\mathcal{S}_\sigma$ and $\mathcal{PS}_\sigma$ can be seen as special cases of $\mathcal{S}_{\sigma,\eta}$ and $\mathcal{PS}_{\sigma,\eta}$ with $\eta = 1$).

The role of these two parameters will be better illustrated in the next section. Here we propose a strategy for choosing their values. The idea is that a quasi-local operator is applied on an already optimized kernel in order to further enhance the classification capability introducing locality. Notice that, in general, it would be possible to estimate the input kernel parameters, the SVM penalty parameter C and the operator parameters at the same time, but this is in contrast with the above idea. Ideally the operators can accept a kernel matrix without knowledge about the kernel function from which it is generated. So the approach we adopt here is to apply the operators on a kernel for which the parameters are already set, thus requiring only one parameter optimization (for $\mathcal{E}_\sigma$, $\mathcal{P}_\sigma$ and $\mathcal{PS}_\sigma$) or two (for $\mathcal{S}_{\sigma,\eta}$ and $\mathcal{PS}_{\sigma,\eta}$). Moreover, we provide some data-dependent estimations of σ η permitting the reduction of the number of parameters values that need to be optimized (3 for η and 4 for σ).

The dataset-dependent estimation of σ take inspiration from the γ^{rbf} estimation, since σ and γ^{rbf} play a similar role of controlling the width of the kernel. However, differently from the K^{rbf} kernel, the $\mathcal{E}_\sigma$ operator uses distances in the feature-space $\mathcal{F}$ (except for the special case $K = K^{lin}$). More precisely, remembering that the data-dependent values of γ^{rbf} are

obtained with $\gamma_h^{rbf} = 1/(2 \cdot q_h^2[\|\mathbf{x} - \mathbf{x}'\|^{\mathbb{R}^p}])$ where $q_h[\|\mathbf{x} - \mathbf{x}'\|^{\mathcal{Z}}]$ denotes the h percentile of the distribution of the distance in the $\mathcal{Z}$ space between every pair of points $\mathbf{x}$, $\mathbf{x}'$, the σ parameter can be estimated using $\sigma_h = 2 \cdot q_h^2[\|\mathbf{x} - \mathbf{x}'\|^{\mathcal{F}}]$ with $h \in \{1, 10, 50, 90\}$ as for the γ^{rbf} case. For η we adopt $\eta_h = q_h[\|\mathbf{x} - \mathbf{x}'\|^{\mathcal{F}}]$ with $h \in \{10, 50, 90\}$.

We thus have a total of 12 quasi-local parameter configurations, meaning that the model selection for quasi-local kernels in this scenario requires no more than 12 cross-validation runs to choose the best parameters. Notice that, comparing the cross-validation best values of the input kernel and quasi-local kernels, we implicitly test also the $\eta = 0$ case. Since $\mathcal{S}_{\sigma,\eta} K$ and $\mathcal{PS}_{\sigma,\eta} K$ with $\eta = 0$ are equivalent to K, $\mathcal{S}_{\sigma,\eta} K$ and $\mathcal{PS}_{\sigma,\eta} K$ have the possibility to reduce to K as a special case. In our empirical evaluation we will highlight the cases in which $\eta = 0$ is selected.

5.2 Intuitive Behaviour of Quasi-Local Kernels

The operators on kernels defined in the previous section aim to modify the behaviour of an input kernel K in order to produce a kernel more sensitive to local information in the feature-space, maintaining however the original behaviour for regions in which the locality is not important. In addition the η and σ parameters control the balance between the input kernel K and its local reformulation $\mathcal{E}_\sigma K$, in other words the effects of the local information.

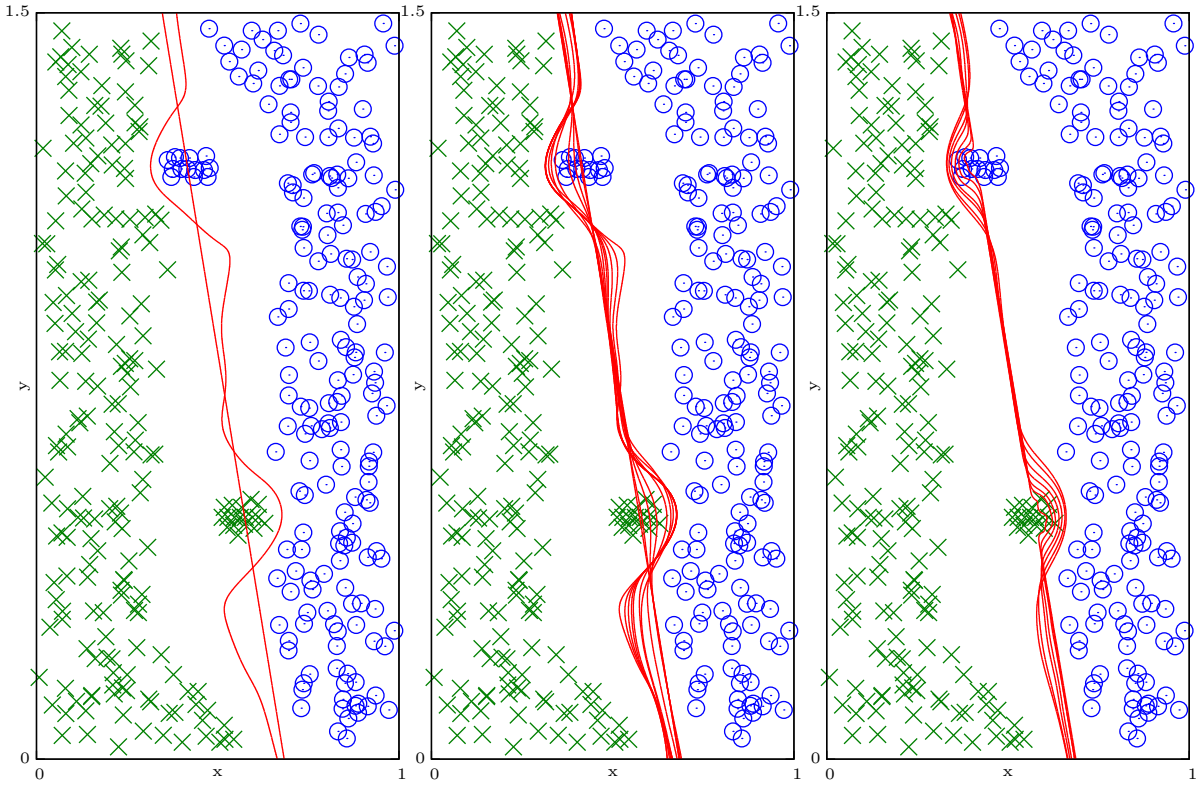
These intuitions are highlighted in Figure 5.2 with an example that illustrates the effects of the $\mathcal{S}_{\sigma,\eta}$ operator on the linear kernel K^{lin} using a two-feature hand-built artificial dataset. Notice that this example is not limited to the combination of K^{lin} and K^{rbf}, because it represents the intuition of what happens in the feature-space of the original kernel applying the $\mathcal{S}_{\sigma,\eta}$ operator. The transformed kernel is:

$$(\mathcal{S}_{\sigma,\eta}\, K^{lin})(\mathbf{x}, \mathbf{x}') = K^{lin}(\mathbf{x}, \mathbf{x}') + \eta \cdot (\mathcal{E}_\sigma\, K^{lin})(\mathbf{x}, \mathbf{x}') = K^{lin}(\mathbf{x}, \mathbf{x}') + \eta \cdot K^{rbf}(\mathbf{x}, \mathbf{x}') \qquad (5.19)$$

with $\gamma^{rbf} = 1/\sigma$. So the $\mathcal{S}_{\sigma,\eta}$ operator on the K^{lin} kernel gives a linear combination of K^{lin} and K^{rbf}. Figure 5.2(a) show the separate behaviours of the global term K^{lin} alone and of the local term $\mathcal{E}_\sigma\, K^{lin} = K^{rbf}$ alone. Figure 5.2(b) illustrates what happens when the local and the global terms are combined with different values of η and a fixed σ. Figure 5.2(c) shows the behaviour of the separating hyperplane with a fixed balancing factor η but varying the σ parameter.

The η parameter regulates the influence on the separating hyperplane of the local term of the quasi-local kernel. In fact, in Figure 5.2(b), we see that all the planes lie between the input kernel (K^{lin}, obtained with $\eta \to 0$ from $\mathcal{S}_{\sigma,\eta}\, K^{lin}$) and the local reformulation of the same kernel (obtained with $\eta = 10^6$ from $\mathcal{S}_{\sigma,\eta}\, K^{lin}$ which behaves as K^{rbf} since the high value of η partially hides the effect of the global term). Moreover, since σ is low, the modifications induced by different values of η are global, influencing all the regions of the separating hyperplane.

We can observe in Figure 5.2(c), on the other hand, that σ regulates the magnitude of the distortion from the linear hyperplane for the region containing points close to the plane itself. The σ parameter in the $\mathcal{E}_\sigma\, K^{lin}$ term of $\mathcal{S}_{\sigma,\eta}\, K^{lin}$ has a similar role to the k parameter in the local SVM approach (i.e. it regulates the range of the locality), even though k defines



(a) K^{lin} and K^{rbf} (b) $\mathcal{S}_{\sigma,\eta}\, K^{lin}$ varying η (c) $\mathcal{S}_{\sigma,\eta}\, K^{lin}$ varying σ

Figure 5.2: The separating hyperplanes for a two-feature hand-built artificial datasets defined by the application of the SVM (all with $C = 3$) with (a) linear kernel K^{lin} and RBF kernel K^{rbf} (with $\gamma^{rbf} = 150$), (b) the $\mathcal{S}_{\sigma,\eta}\, K^{lin}$ quasi-local kernel with fixed σ ($\sigma = 1/150 = 1/\gamma^{rbf}$) and variable η ($\eta = 10^6, 50, 10, 1, 0.5, 0.1, 0.05, 0.03, 0.01, 0.005, 0.001, 0.000001$), and (c) the $\mathcal{S}_{\sigma,\eta}\, K^{lin}$ quasi-local kernel with fixed η ($\eta = 0.05$) and variable σ ($\sigma = 1/5000, 1/2000, 1/1000, 1/500, 1/300, 1/150, 1/100$).

an hard boundary between local and non local points instead of a negative exponential one. It is important to emphasize that in the central region of the dataset the separating hyperplane remains linear, highlighting that the kernel resulting from the $\mathcal{S}_{\sigma,\eta}$ operator differs from the input kernel only where the information is local.

The example simply illustrates the intuition behind the proposed family of quasi-local kernels, and in particular how the input kernel behaviour in the feature-space is maintained for the regions in which the information is not local, so it is not important here to analyse the classification accuracy. However, kernels that are sensitive to important local information but retain properties of global input kernels, can also be obtained from very elaborated and well tuned kernels defined on high-dimensionality input spaces. In the following two sections we investigate the accuracy performances of the quasi-local kernels in a number of real datasets using a data-dependent method of choosing η and σ parameters.

dataset name	brief description	# of classes	training set size	# of features
leukemia	Cancer classification, originally from [79]	2	38	7129
iris	A well known pattern recognition dataset	3	150	4
wine	wine recognition from chemical data, preproc. as [65]	3	178	13
sonar	discrimination between different sonar signals	2	208	60
glass	types of glass classification	6	214	9
heart	heart disease prediction, originally from Statlog [98]	2	270	13
liver	liver disorders pred. from alcohol consumption data	2	345	6
ionosphere	classification of radar signals from the ionosphere	2	351	34
bioinf	(*svmguide2*) bioinformatics data originally from [87]	3	391	20
vowel	automatic recognition of British English vowels	11	528	10
breast	Wisconsin breast cancer data	2	683	10
australian	Australian credit approval, orig. from Statlog [98]	2	690	14
diabetes	Pima indians diabetes data	2	768	8
vehicle4	vehicle recognition [65], originally from Statlog [98]	4	846	18
fourclass	a binary class problem from multi-class problem [85]	2	862	2
splice	primate splice-junction gene sequences data	2	1000	60
numer	German credit risk data, originally from Statlog [98]	2	1000	24
vehicle	(*svmguide3*) vehicle data originally from [87]	2	1243	21
a1a	Adult dataset preprocessed as done by [143]	2	1605	123
DNA	DNA problem preprocessed as done in [88]	3	2000	180
segment	image segmentation data originally from Statlog [98]	7	2310	19
w1a	web page classification, originally from [143]	2	2477	300
astro	(*svmguide1*) astroparticle application from [87]	2	3089	4

Table 5.1: The 23 datasets for Experiment 1 ordered by training set size.

5.3 Experiment 1

The goal of the first experiment is to compare the accuracy of SVM (using LibSVM) with quasi-local kernels against SVM with traditional kernels and kNNSVM (using a preliminary version of FkNNSVM). The evaluation is carried out on 23 non-large datasets.

5.3.1 Experimental Protocol

Table 5.1 lists the 23 datasets from different sources and scientific fields used in this experiment; we took all the freely available datasets from the LibSVM repository [36] with training set with no more than 3500 examples. Some datasets are multi-class and the number of features ranges from 2 to 7129.

The reference input kernels for the quasi-local operators considered are the linear kernel K^{lin}, the polynomial kernel K^{pol}, the radial basis function kernel K^{rbf} and the sigmoidal kernel K^{sig}. The quasi-local kernels we tested are those resulting from the application of the $\mathcal{E}_\sigma$, $\mathcal{P}_\sigma$, $\mathcal{S}_{\sigma,\eta}$, $\mathcal{PS}_{\sigma,\eta}$ operators on the reference input kernels. We also evaluated the accuracy of the

kNNSVM classifier with the same reference input kernels.

The methods are evaluated using 10-fold cross validation. The assessment of statistical significant difference between two methods on the same dataset is performed with the two-tailed paired t-test ($\alpha = 0.05$) on the two sets of fold accuracies. Although the count of positive and negative significative difference can be used to establish if a method performs better than another on multiple datasets, it has been shown [61] that the Wilcoxon signed-ranks test [202] is a theoretically safer (with respect to parametric tests it does not assume "normal distributions or homogeneity of variance") and empirically stronger test. For this reason the final assessment of statistical significance difference on the 23 datasets is performed with the Wilcoxon signed-ranks test (in case of ties, the rank is calculated as the average rank between them).

The model selection is performed on each fold with a inner 5-fold cross validation as follows. For all the methods tested, the regularisation parameter C is chosen in $\{2^{-5}, 2^{-4}, \ldots, 2^{9}, 2^{10}\}$. For the polynomial kernel we adopt the widely used homogeneous polynomial kernel ($\gamma^{pol} = 1$, $r^{pol} = 0$), selecting a degree non higher than 5. The choice of γ^{rbf} for the RBF kernel is done adopting γ_h^{rbf} where h is chosen among $\{1, 10, 50, 90\}$ as described in 3.3. For the sigmoidal kernel, r^{sig} is set to 0, whereas γ^{sig} is chosen among $\{2^{-7}, 2^{-6}, \ldots, 2^{-1}, 2^{0}\}$. For the quasi-local kernels we use the C and kernel parameters found by the model selection described above for each input kernel, whereas σ is chosen using $\sigma_h^{\mathcal{F}}$ with $h \in \{1, 10, 50, 90\}$ and η using $\eta_{.h}^{\mathcal{F}}$ with $h \in \{10, 50, 90\}$ and (implicitly) $\eta = 0$ as described in Section 5.1.7 through a 5-fold CV on the same folds used for model selection on the input kernels. Finally, the value of k for kNNSVM is automatically chosen on the training set between $\mathcal{K} = \{1, 3, 5, 7, 9, 11, 15, 23, 39, 71, 135, 263, 519, 1031\}$ (the first 5 odd natural numbers followed by the ones obtained with a base-2 exponential increment from 9) as described in section 5.1.7.

We used LibSVM [36] version 2.84 for SVM training and testing, extending it with a object-oriented architecture for kernel calculation and specification. For the quasi-local kernels we store the values of $\langle \mathbf{x}, \mathbf{x} \rangle$ for each example in order to obtain the quasi-local kernel value computing only one scalar product, i.e. $\langle \mathbf{x}, \mathbf{x}' \rangle$, instead of three. The kNNSVM implementation is a preliminary version (that does not use Cover Trees) of FkNNSVM available in the FaLKM-lib [163] and described in Appendix A. The experiments are carried out on multiple Intel® Xeon™ CPU 3.20GHz systems, setting the kernel cache dimension to 1024Mb and interrupting the experiments that are not terminated after 72 hours.

5.3.2 Results

Table 5.2 reports the 10-fold cross validation accuracy of SVM with the four considered input kernels, SVM with the quasi-local kernels obtained applying the $\mathcal{E}_\sigma$, $\mathcal{P}_\sigma$, $\mathcal{S}_{\sigma,\eta}$, $\mathcal{PS}_{\sigma,\eta}$ operators and kNNSVM. Some kNNSVM results are missing due to the computational effort of the method, corresponding to the cases in which kNNSVM does not terminates within 72 hours. The $^+$ and $^-$ denotes quasi-local kernel and kNNSVM results that are significatively better (and worse) than the corresponding input kernels according to the two-tailed paired t-test ($\alpha = 0.05$). The total number of datasets in which quasi-local kernels and kNNSVM perform better (and worse) than the corresponding input kernels are reported, with the number of significative differences

dataset	$K = K^{lin}$						$K = K^{rbf}$					
	K	$\mathcal{E}_\sigma K$	$\mathcal{P}_\sigma K$	$\mathcal{S}_{\sigma,\eta} K$	$\mathcal{PS}_{\sigma,\eta} K$	FkNNSVM	K	$\mathcal{E}_\sigma K$	$\mathcal{P}_\sigma K$	$\mathcal{S}_{\sigma,\eta} K$	$\mathcal{PS}_{\sigma,\eta} K$	FkNNSVM
leukemia	**.947**	$.763^{-}$	**.947**	**.947**	**.947**	.921	**.947**	.895	.921	**.947**	**.947**	.895
iris	.967	.960	.960	**.973**	**.973**	.960	.960	.953	.960	**.973**	.967	**.973**
wine	.972	.972	.978	.978	.978	.966	.972	.972	.978	.978	.972	.966
sonar	.745	.755	$.880^{+}$	$.870^{+}$	$.894^{+}$	$.899^{+}$	.894	.899	**.904**	.894	.894	$.865^{-}$
glass	.640	.692	$.710^{+}$	.687	$.701^{+}$	$.710^{+}$	.682	.668	.678	$.715^{+}$	$.706^{+}$	$.734^{+}$
heart	**.833**	.822	.811	.826	.826	.811	.819	.822	.807	.822	.819	.815
liver	.687	.722	.713	$.733^{+}$	.722	$.733^{+}$	.719	.733	.725	.725	.728	.725
ionosphere	.883	$.943^{+}$	$.943^{+}$	.929	$.949^{+}$	$.940^{+}$	.940	**.954**	**.954**	.946	.952	.943
bioinf	.818	.841	.821	.834	.826	.841	.831	.836	.841	.831	.839	.854
vowel	.848	$.989^{+}$	$.989^{+}$	$.992^{+}$	$.991^{+}$	$\mathbf{.996}^{+}$	.992	.994	.994	**.996**	**.996**	**.996**
breast	.958	.966	.965	.966	.962	.971	.969	.969	.968	.971	.972	.971
australian	.848	.848	.839	.846	.848	.864	.843	.851	.843	.852	.848	.851
diabetes	.766	.766	.754	.772	.775	**.779**	.772	.763	.770	.766	.762	.768
vehicle4	.800	$.849^{+}$	$.853^{+}$	$.855^{+}$	$.859^{+}$	$\mathbf{.866}^{+}$	.857	.856	.849	.849	<u>.857</u>	.853
fourclass	.774	$.987^{+}$	$.922^{+}$	$.988^{+}$	$.950^{+}$	$\mathbf{1.00}^{+}$	**1.00**	.998	.999	**1.00**	**1.00**	.999
splice	.800	.774	$.872^{+}$	$.848^{+}$	$.884^{+}$	-	**.885**	.882	.882	.881	.880	-
numer	.769	.747	$.698^{-}$	.765	.770	-	.760	.757	$.750^{-}$	.761	.765	.757
vehicle	.829	.846	.841	$.847^{+}$	.848	.828	.843	.849	.838	.845	.846	.840
a1a	**.833**	$.800^{-}$	$.802^{-}$	.831	.832	-	.828	.831	.827	.831	.827	-
DNA	.952	$.558^{-}$	$.960^{+}$	.953	$.959^{+}$	$.936^{-}$	.958	.960	**.962**	.959	.959	-
segment	.959	$.971^{+}$	$.972^{+}$	$.975^{+}$	.971	$.975^{+}$	.972	.972	**.976**	.973	.975	-
w1a	.981	$.973^{+}$	.979	**.981**	.981	.979	.981	**.981**	.981	.980	.980	-
astro	.955	$.967^{+}$	$.968^{+}$	$.967^{+}$	$.969^{+}$	$\mathbf{.971}^{+}$	.966	.967	.968	$.969^{+}$	$.969^{+}$	-
# pos. diff.		12(7)	15(10)	18(9)	19(9)	13(10)		11(0)	10(0)	15(2)	13(2)	9(1)
# neg. diff.		8(2)	7(2)	4(0)	2(0)	7(1)		9(0)	11(1)	4(0)	4(0)	8(1)
Wsr test			✓	✓	✓	✓				✓	✓	

- + and - denote QL kernel and FkNNSVM results significatively better (or worse) than SVM with input kernels according to the two-tailed paired t-test ($\alpha = 0.05$);
- **# pos. diff.** and **# neg. diff.** denote, for each QL kernel and FkNNSVM methods, the number of datasets in which they perform better (or worse) than the corresponding input kernels. In parenthesis are reported the statistically significative differences;
- **Wsr test** marks the QL kernel improvements over the corresponding input kernels according to the Wilcoxon signed-ranks tests ($\alpha = 0.05$);
- <u>underlined</u> are the cases in which, for $\mathcal{S}_{\sigma,\eta}$ and $\mathcal{PS}_{\sigma,\eta}$, the lowest empirical risk is achieved with $\eta = 0$ for all folds;
- in **bold**, are highlighted the best 10-fold CV accuracies of a specific dataset among all methods and kernels (considering also the kernel of Figure 5.3).

Table 5.2: Exp. 1. 10-fold CV accuracy of SVM with LIN and RBF input kernels, corresponding QL kernels and of FkNNSVM.

dataset	$K = K^{pol}$						$K = K^{sig}$					
	K	$\mathcal{E}_\sigma K$	$\mathcal{P}_\sigma K$	$\mathcal{S}_{\sigma,\eta} K$	$\mathcal{PS}_{\sigma,\eta} K$	FkNNSVM	K	$\mathcal{E}_\sigma K$	$\mathcal{P}_\sigma K$	$\mathcal{S}_{\sigma,\eta} K$	$\mathcal{PS}_{\sigma,\eta} K$	FkNNSVM
leukemia	**.947**	.711$^-$	.763$^+$	**.947**	**.947**	**.947**	.711	.711	.711	.658	.658	.789
iris	**.973**	.960	.960	.947	.947	.960	.960	.967	.953	.960	.960	.960
wine	.961	.961	.978	.966	.961	.966	.972	.983	**.989**	.972	.978	.966
sonar	.851	.716$^-$	.861	.846	.846	.885	.750	.899$^+$	.880$^+$	.894$^+$	.870$^+$	.885$^+$
glass	.701	.701	.701	.706	.696	.720	.626	.678$^+$	.664	.682$^+$	.696$^+$	**.738**$^+$
heart	.819	.785	.796	**.833**	.822	.811	.830	.811	.815	**.833**	.830	.819
liver	.725	.690	.716	.728	.722	.730	.672	.733$^+$	.716	**.739**$^+$	.704	.722$^+$
ionosphere	.900	.766$^-$	.926	.923	.926	.934	.872	.943$^+$	.946$^+$	.949$^+$	**.954**$^+$	.943$^+$
bioinf	.821	.770	.818	.818	.824	**.857**$^+$	.829	.795	.836	.831	.839	.852
vowel	.973	.987	.987	.991$^+$	.992$^+$	.994$^+$	.799	.991$^+$	.991$^+$	.991$^+$	.992$^+$	**.996**$^+$
breast	.963	.962	.960	.958	.960	.956	.975	.975	**.978**	.972	.969	.958$^-$
australian	.851	.854	.843	.851	.851	.852	.849	.849	.854	.851	.848	**.868**$^+$
diabetes	.767	.760	**.779**	.766	.763	.766	.758	.768	.773	.759	.767	**.779**$^+$
vehicle4	.846	.818	.833	<u>.846</u>	.839	-	.787	.852$^+$	.839$^+$	.851$^+$	.830$^+$	-
fourclass	.799	.997$^+$	.964$^+$	.995$^+$	.959$^+$	.998$^+$	.776	**1.00**$^+$	.911$^+$	**1.00**$^+$	.818$^+$	.999$^+$
splice	.862	.828	.878	.862	.876	-	.805	.876$^+$	.865$^+$	.867$^+$	.841$^+$	-
numer	.766	.741	.723	.767	**.771**	-	.766	.734$^-$	.751	<u>.766</u>	.755	.758
vehicle	.850	.797$^-$	.844	**.851**	<u>.850</u>	-	.822	.845	.846$^+$	.846$^+$	.826	-
a1a	.828	.814	.809	.827	.830	-	**.833**	.828	.826	.822	.822	-
DNA	.958	.910	.958	.958	.958	-	.949	.960$^+$	.959$^+$	.956$^+$	.956	-
segment	.970	.966	.973	.972	.972	-	.947	.974$^+$	.972$^+$	.972$^+$	.975$^+$	-
w1a	.980	.974$^-$	.981	.980	.980	-	.981	.980	.980	.979	.979	-
astro	.968	.967	.965	.965	.968	.969	.954	.967$^+$	.968$^+$	**.971**$^+$	.970$^+$	-
# pos. diff.		6(1)	10(2)	10(2)	10(2)	10(3)		15(11)	17(10)	16(12)	15(9)	10(8)
# neg. diff.		15(5)	12(0)	7(0)	8(0)	4(0)		5(1)	5(0)	4(0)	6(0)	4(1)
Wsr test								✓	✓	✓	✓	✓

- + and - denote QL kernel and FkNNSVM results significatively better (or worse) than SVM with input kernels according to the two-tailed paired t-test ($\alpha = 0.05$);
- **# pos. diff.** and **# neg. diff.** denote, for each QL kernel and FkNNSVM methods, the number of datasets in which they perform better (or worse) than the corresponding input kernels. In parenthesis are reported the statistically significative differences;
- **Wsr test** marks the QL kernel improvements over the corresponding input kernels according to the Wilcoxon signed-ranks tests ($\alpha = 0.05$);
- <u>underlined</u> are the cases in which, for $\mathcal{S}_{\sigma,\eta}$ and $\mathcal{PS}_{\sigma,\eta}$, the lowest empirical risk is achieved with $\eta = 0$ for all folds;
- in **bold**, are highlighted the best 10-fold CV accuracies of a specific dataset among all methods and kernels (considering also the kernel of Figure 5.2).

Table 5.3: Exp. 1. 10-fold CV accuracy of SVM with POL and SIG input kernels, corresponding QL kernels and of FkNNSVM.

in parenthesis. The last row reports the Wilcoxon signed-ranks tests to assess the significant improvements of quasi-local kernels over corresponding input kernels on all the datasets (for kNNSVM only on the datasets for which the results are present). The cases in which, for $\mathcal{S}_{\sigma,\eta}$, $\mathcal{PS}_{\sigma,\eta}$, the model selection chose $\eta = 0$ for all the 10 folds thus giving the same results of the input kernels, are underlined. In bold, are highlighted the best 10-fold cross validation accuracy achieved for a specific dataset among all the methods and kernels.

5.3.3 Discussion

The kNNSVM results basically confirm the earlier assessment [164], although the model selection is performed here differently; kNNSVM is able to improve, according to the Wilcoxon signed-rank test, the classification generalization accuracy of SVM with the K^{lin} kernel (10 two-tailed paired t-test significant improvements, 1 deteriorations) and K^{sig} kernel (8 two-tailed paired t-test significant improvements, 1 deteriorations). Instead we do not have evidence of improved generalization accuracy on the benchmark datasets for the K^{pol} kernel and K^{rbf} kernel, although we showed in [164] that, for K^{rbf}, there are scenarios in which kNNSVM can be particularly indicated.

$\mathcal{E}_\sigma\, K$ seems to perform significantly better than K for K^{sig} and for K^{lin} (although without statistical evidence), whereas there are no overall improvement for K^{rbf}, and for K^{pol} the accuracies are deteriorated. These results are probably due to the choice of not re-performing model selection for $\mathcal{E}_\sigma\, K$ in particular for the C parameter. In fact $\mathcal{E}_\sigma$ is the only operator that does not contain the input kernel explicitly in the resulting one, and thus the optimal parameters can be very different. This is confirmed by the fact that $\mathcal{E}_\sigma\, K^{lin}$ is equivalent to K^{rbf} but their results, as reported in Table 5.2, appears to be very different.

The results of $\mathcal{P}_\sigma\, K$ are slightly better than $\mathcal{E}_\sigma\, K$. According to the Wilcoxon signed-rank test, it is better than K for $K = K^{sig}$ and $K = K^{rbf}$, but not for K^{rbf} and K^{pol}. In total, the kernels obtained with $\mathcal{P}_\sigma$ achieve the best accuracies for 8 datasets, meaning that this operator is able to reach very good results but the improvements are not systematic for all the input kernels. It is possible to notice that the classification results of $\mathcal{P}_\sigma\, K$ are very similar to the kNNSVM ones (both improve significantly over SVM with the K^{lin} and K^{sig} but not for K^{rbf} and K^{pol}).

The best results are clearly achieved by the $\mathcal{S}_{\sigma,\eta}$ and $\mathcal{PS}_{\sigma,\eta}$ operators without significative differences between them. According to the Wilcoxon signed-rank test they significantly improve the generalization accuracy for K^{lin}, K^{rbf} and K^{sig}. Moreover, they are the only operators that never cause significant 10-fold CV losses according to the statistical two-tailed paired t-test, while the number of improvements are impressive at least for K^{lin} and K^{sig}. The only kernel that seems not to take a decisive advantage from the two operators is K^{pol} that, together with the results noticed for kNNSVM with the same input kernel, lead us to argue that, at least for non-large datasets, locality is not a crucial point for the polynomial kernels. Comparing $\mathcal{S}_{\sigma,\eta}\, K$ and $\mathcal{PS}_{\sigma,\eta}$ with kNNSVM we can notice that the operator approach performs better in terms of 10-fold CV accuracies (especially for K^{rbf}).

We do not discuss directly the computational performances of the operators in this experi-

dataset name	brief description	# of classes	training set size	testing set size	# of features
optdigit	optical recognition of handwritten digits	10	3823	1797	64
blocks	segmented page blocks classification	5	4107	1368	10
satimage	Landsat satellite data, from Statlog [98]	6	4435	2000	36
musk2	musks/non-musks molecule prediction, v.2	2	4949	1649	166
isolet	spoken letter prediction	26	6238	1559	617
usps	handwritten text recognition	10	7291	2007	256
magic	high energy gamma particles detection	2	14265	4755	10
letter	letter recognition, orig. from Statlog [98]	26	15000	5000	16
news20	newsgroup classification, preproc. as [105]	20	15935	3993	62061
protein	protein classification task	3	17766	6621	357
rcv1	two class version of Reuters Corpus V. I	2	20242	677399	47236
mnist1	handwritten digits, preproc. as [115]	10	21000	49000	780
a9a	Adult dataset preprocessed as [143]	2	32561	16281	123
shuttle	the shuttle dataset, orig. from Statlog [98]	7	43500	14500	9
w8a	web page classification, orig. from [143]	2	49749	14951	300
ijcnn1	IJCNN 2001 challenge, preproc. as [115]	2	49990	91701	22
connect4	connect4 result prediction (binary enc.)	3	50668	16889	126
acoustic	vehicle classification from acoustic sensors	3	78823	19705	50
seismic	vehicle classification from seismic sensors	3	78823	19705	50
cov-type	forest cover type prediction, orig. from [16]	2	100000	481012	54

Table 5.4: The 20 datasets for Experiment 2 ordered by training set size.

ment. However, we can notice that they are much faster, as expected, than kNNSVM since, in total, 25 kNNSVM results are missing due to computational difficulties (the computation does not finish within 72 hours) whereas SVM with input and quasi-local kernels always terminate in a reasonable time.

5.4 Experiment 2

The second experiment applies the SVM with the quasi-local kernels that, in the exploratory Experiment 1, seem to achieve better accuracy values, i.e. $\mathcal{S}_{\sigma,\eta}\,K$ and $\mathcal{PS}_{\sigma,\eta}\,K$. The aim of this experiment is to verify if these kernels are able to improve the input kernel classification accuracy in a considerable number of large datasets without worsening dramatically the computational performances.

5.4.1 Experimental Procedure

The 20 datasets used in the second experiment are summarized in Table 5.4; they are all the datasets with more than 3500 examples available on the LibSVM repository [36] (except the mushrooms dataset for which perfect classification is already easily achievable for all the input

kernels) and the UCI datasets for classification with only numerical values, available test labels, and more than 3500 training examples. The datasets are quite large and for this reason kernels resulting from the four chosen operators with the four input kernels are simply trained on the training set and tested on the testing set. If no separate testing sets are provided we use 75% of available data (randomly selected) for training and the remaining 25% for testing, apart for the cov-type from which we randomly selected 100000 examples leaving the remaining 481012 in the testing set for computational reasons. We normalized the data in the range $[0, 1]$. With this approach the t-tests are not suitable, and the best way to assess statistical significance is the Wilcoxon signed rank test as detailed in [202].

The model selection is performed with 10-fold CV with the same approach of Experiment 1 and with the same ranges of parameter values. We do not test the kNNSVM classifier because of the computational weight of the method.

5.4.2 Results

Table 5.5 shows the generalization accuracy results of the input kernels K and of the quasi-local kernels $\mathcal{S}_{\sigma,\eta}K$ and $\mathcal{PS}_{\sigma,\eta}K$ on all the 20 datasets listed in Table 5.4. We report the number of datasets in which quasi-local kernels perform better (or worse) than the corresponding input kernels, the Wilcoxon signed rank test to asses the statistical significance of differences between them, and the average rank of each method. The cases for which model selection for quasi-local kernels chooses $\eta = 0$ thus obtaining the same model of the SVM with the input kernel are underlined. In bold are highlighted the best generalization accuracies achieved for each dataset. Notice that the results regarding the cov-type dataset with the K^{pol} kernel are missing because of its excessive computational weight (especially for high degrees of the kernel) causes the model selection to take more than 72 hours to be completed.

The training and testing times, expressed in seconds, are reported in Table 5.6 and Table 5.7 respectively. We point out the number of times SVM with quasi local kernels are faster and slower than the corresponding input kernels and (in parenthesis) the number of times SVM with quasi local kernels are three times faster and slower than the corresponding input kernels (these big variations are highlighted in bold and italic).

5.4.3 Discussion

Quasi-local kernels perform better than the corresponding input kernels in terms of generalization accuracy with statistical significance as reported in Table 5.5, for all the input kernels taken into account. The number and the magnitude of the improvements are particularly large for the K^{lin} and K^{sig} input kernels. This because they are global kernels that in general (a part for K^{lin} in presence of a high-dimensional problems) are not able to achieve very high accuracy results, and thus the addition of the local information is almost always crucial. We can notice that, for these large datasets, the operators are able to improve the generalization accuracies also for he K^{pol} kernel differently from Experiment 1. Looking at the average ranks of all the methods, we can see that the methods achieving the best results are $\mathcal{PS}_{\sigma,\eta}\,K^{lin}$, $\mathcal{PS}_{\sigma,\eta}\,K^{rbf}$ and $\mathcal{S}_{\sigma,\eta}\,K^{rbf}$. On the other hand, apart K^{rbf} whose average rank is near the mean position

dataset	$K = K^{lin}$			$K = K^{rbf}$			$K = K^{pol}$			$K = K^{sig}$		
	K	$\mathcal{S}_{\sigma,\eta}K$	$\mathcal{PS}_{\sigma,\eta}K$	K	$\mathcal{S}_{\sigma,\eta}K$	$\mathcal{PS}_{\sigma,\eta}K$	K	$\mathcal{S}_{\sigma,\eta}K$	$\mathcal{PS}_{\sigma,\eta}K$	K	$\mathcal{S}_{\sigma,\eta}K$	$\mathcal{PS}_{\sigma,\eta}K$
optdigit	.9672	.9777	**.9855**	.9816	.9816	.9816	.9750	.9750	.9822	.9666	.9839	.9800
blocks	.9678	.9715	.9715	.9635	.9635	.9635	.9671	.9722	**.9759**	.9591	.9686	.9686
satimage	.8580	.9150	.9180	.9190	**.9230**	.9210	.8880	.9120	.9105	.8570	.9215	.9135
musk2	.9551	.9982	.9982	.9970	.9976	**.9988**	.9970	.9970	.9964	.9260	.9951	.9715
isolet	.9596	.9609	.9673	.9666	.9679	.9679	.9628	.9628	**.9686**	.8012	.8518	.8621
usps	.9357	.9472	.9522	.9527	.9547	**.9557**	.9397	.9422	.9452	.9243	.9532	.9507
magic	.7868	.8694	.8751	.8755	.8763	.8755	**.8776**	.8763	**.8776**	.7865	.8700	.8574
letter	.8512	.9774	.9766	.9748	.9776	**.9778**	.9556	.9692	.9708	.8516	.9768	.9740
news20	.8550	.8550	.8550	.8257	.8257	.8257	.7626	.7744	.7626	**.8610**	**.8610**	**.8610**
protein	.6865	.6868	**.7041**	.7026	.7005	.6987	.6919	.6926	.6919	.6865	.6981	.6874
rcv1	.9605	.9570	**.9637**	.9455	.9426	.9405	.9545	.9478	.9545	.9604	.9542	.9622
mnist1	.9367	.9525	.9747	.9735	.9749	**.9754**	.9708	.9710	.9733	.9044	.9547	.9530
a9a	.8498	**.8511**	.8498	.8502	.8509	.8502	.8477	.8479	.8477	.8498	.8498	.8496
shuttle	.9794	**.9993**	.9992	.9990	.9992	.9992	.9987	**.9993**	.9988	.9757	.9991	.9988
w8a	.9868	.9944	**.9945**	.9910	.9914	.9919	.9924	.9944	.9924	.9858	.9909	.9886
ijcnn1	.9218	.9787	.9748	.9758	.9814	**.9824**	.9676	.9665	.9676	.9203	.9786	.9631
connect4	.7591	.8421	.8600	**.8623**	.8623	.8623	.8441	.8441	.8588	.7476	.8074	.7939
acoustic	.7024	.7997	.8001	.7987	.7999	**.8004**	.7984	.7986	.7993	.7020	.7988	.7845
seismic	.7281	.7694	.7697	.7698	.7698	.7698	.7658	.7658	.7658	.6976	**.7701**	.7486
cov-type	.7629	.9098	.9121	.9077	**.9202**	.9187	-	-	-	.6286	.8732	.8629
# pos. diff.		18	18		13	11		14	10		17	18
# neg. diff.		1	0		2	2		3	1		1	1
Wsr test		✓	✓		✓	✓		✓	✓		✓	✓
avg rank	9.70	5.35	3.70	5.50	3.88	3.88	8.10	7.05	6.55	10.60	5.75	7.95

- **# pos. diff.** and **# neg. diff.** denote, for each QL kernel, the number of datasets in which they perform better (or worse) than the corresponding input kernels;
- **Wsr test** reports if the Wilcoxon signed-ranks test states that the improvements of QL kernels over corresponding input kernels are significant ($\alpha = 0.05$);
- underlined are the cases in which, for $\mathcal{S}_{\sigma,\eta}$ and $\mathcal{PS}_{\sigma,\eta}$, the lowest empirical risk is achieved with $\eta = 0$;
- in **bold**, are highlighted the best generalization accuracies achieved for a specific dataset among all methods and kernels;
- missing values correspond to cases for which model selection was not completed because for some parameter the training of a single fold takes more than 72 hours.
- **avg rank** reports the average rank of the methods.

Table 5.5: Experiment 2. Generalization accuracy of SVM with the input and quasi-local kernels.

dataset	$K = K^{lin}$			$K = K^{rbf}$			$K = K^{pol}$			$K = K^{sig}$		
	K	$\mathcal{S}_{\sigma,\eta}K$	$\mathcal{PS}_{\sigma,\eta}K$	K	$\mathcal{S}_{\sigma,\eta}K$	$\mathcal{PS}_{\sigma,\eta}K$	K	$\mathcal{S}_{\sigma,\eta}K$	$\mathcal{PS}_{\sigma,\eta}K$	K	$\mathcal{S}_{\sigma,\eta}K$	$\mathcal{PS}_{\sigma,\eta}K$
optdigit	1	1	1	2	2	2	1	1	1	1	3	3
blocks	1	1	1	1	1	1	2	1	1	1	2	2
satimage	1	2	2	3	3	4	9	**2**	**2**	2	6	4
musk2	16	**5**	**4**	4	4	4	5	7	4	8	7	8
isolet	27	32	53	50	61	70	31	32	59	123	71	72
usps	9	14	11	11	15	17	7	10	20	12	39	13
magic	164	129	138	99	146	154	3867	2360	6666	63	99	99
letter	12	12	14	14	24	28	13	12	23	21	46	37
news20	284	325	407	359	436	435	561	639	668	271	430	472
protein	354	377	431	440	499	532	410	460	552	424	611	590
rcv1	108	124	189	165	196	208	269	296	337	129	214	219
mnist1	93	99	131	124	153	174	94	101	155	213	185	203
a9a	220	290	460	196	263	270	219	280	*970*	259	590	547
shuttle	79	**6**	**5**	6	6	6	27	**4**	**8**	101	**30**	69
w8a	1292	**84**	**81**	59	90	95	40	*131*	53	55	157	124
ijcnn1	189	109	102	95	85	93	298	267	634	290	282	388
connect4	1219	1380	2419	1608	2469	3225	2913	3255	2985	1403	2074	2194
acoustic	7794	10972	10120	3143	4180	5398	2661	3459	3741	4544	8296	5689
seismic	10045	19704	21466	4160	5909	7144	5670	7190	8340	3958	8992	6164
cov-type	19106	36914	*97532*	15506	22319	24694				2745	6283	5546
# pos. diff.		5(3)	5(3)		1(0)	1(0)		6(2)	4(2)		5(1)	3(0)
# neg. diff.		12(0)	13(1)		14(0)	15(0)		12(1)	14(1)		15(1)	16(0)

- **# pos. diff.** and **# neg. diff.** denote, for each QL kernel, the number of datasets in which they are faster (or slower) than the corresponding input kernels. In parenthesis are reported the differences greater than 3 times;
- in **bold**, are the cases in which the QL kernels are at least three times faster than the corresponding input kernel;
- in *italic*, are the cases in which the QL kernels are at least three times slower than the corresponding in put kernel;
- missing values correspond to cases for which model selection was not completed because for some parameter the training of a single fold takes more than 72 hours.

Table 5.6: Experiment 2. Training times (in seconds) of SVM with the input and quasi-local kernels.

dataset	$K = K^{lin}$			$K = K^{rbf}$			$K = K^{pol}$			$K = K^{sig}$		
	K	$\mathcal{S}_{\sigma,\eta}K$	$\mathcal{PS}_{\sigma,\eta}K$	K	$\mathcal{S}_{\sigma,\eta}K$	$\mathcal{PS}_{\sigma,\eta}K$	K	$\mathcal{S}_{\sigma,\eta}K$	$\mathcal{PS}_{\sigma,\eta}K$	K	$\mathcal{S}_{\sigma,\eta}K$	$\mathcal{PS}_{\sigma,\eta}K$
optdigit	1	1	1	1	1	2	1	1	1	1	3	2
blocks	1	1	1	1	1	1	1	1	1	1	1	1
satimage	1	1	2	2	3	2	1	1	2	2	4	4
musk2	1	2	1	1	1	1	1	1	2	3	3	2
isolet	20	22	25	26	28	29	21	21	26	40	35	35
usps	4	8	6	6	6	8	4	5	7	5	15	7
magic	7	6	7	7	6	7	4	8	7	15	25	28
letter	15	17	19	19	25	26	10	19	22	27	45	42
news20	56	64	72	69	77	77	71	81	82	60	90	94
protein	107	117	125	125	142	150	119	133	151	130	189	188
rcv1	3355	3893	5932	5194	6195	6558	8142	9038	10284	4055	6827	6978
mnist1	419	451	533	529	617	667	368	398	552	820	763	787
a9a	64	89	93	84	107	107	64	89	98	117	238	238
shuttle	14	**1**	**1**	1	1	1	1	1	1	34	**6**	20
w8a	2	*18*	*20*	12	20	22	4	*26*	11	6	*34*	*24*
ijcnn1	239	156	103	166	162	160	28	50	26	456	443	700
connect4	280	273	235	237	274	306	187	222	291	371	559	596
acoustic	589	542	540	534	618	627	461	586	580	854	1134	1256
seismic	546	566	567	561	662	673	477	608	670	791	1195	1279
cov-type	6813	4619	4129	4643	5076	4926				7982	17135	18469
# pos. diff.		6(1)	5(1)		2(0)	1(0)		0(0)	1(0)		4(1)	4(0)
# neg. diff.		11(1)	11(1)		13(0)	14(0)		13(1)	15(0)		14(1)	15(1)

- **# pos. diff.** and **# neg. diff.** denote, for each quasi-local kernel, the number of datasets in which they are faster (or slower) than the corresponding input kernels. In parenthesis are reported the differences greater than 3 times;
- in **bold**, are the cases in which the quasi-local kernels are at least three times faster than the corresponding input kernel;
- in *italic*, are the cases in which the quasi-local kernels are at least three times slower than the corresponding in put kernel;
- missing values correspond to cases for which model selection was not completed because for some parameter the training of a single fold takes more than 72 hours.

Table 5.7: Experiment 2. Testing times (in seconds) of SVM with the input and quasi-local kernels.

(6), the other three input kernels have the worst average ranks. Looking at the best result for each dataset (bold values in Table 5.5), we can notice that $\mathcal{PS}_{\sigma,\eta}\,K^{rbf}$ is the kernel that permits the highest number of best generalization accuracies (about for one third of datasets), whereas the input kernels rarely achieve the best results. Compared to $\mathcal{S}_{\sigma,\eta}$, $\mathcal{PS}_{\sigma,\eta}$ seems to be a more "extreme" approach in the sense that it achieves the best results more frequently but at the same time there are more cases in which $\eta = 0$ is selected meaning that the input kernel has an higher training set accuracy. For this reason we can hypothesize that $\mathcal{PS}_{\sigma,\eta}$ introduces an higher level of locality than $\mathcal{S}_{\sigma,\eta}$. From the above considerations, we can conclude that the $\mathcal{S}_{\sigma,\eta}$ and $\mathcal{PS}_{\sigma,\eta}$ operators are able to significantly improve the generalization ability of traditional kernels, and, in particular, the kernels that show the best accuracies and can be thus indicated as good candidate kernels for general classification problems, are $\mathcal{PS}_{\sigma,\eta}\,K^{lin}$, $\mathcal{PS}_{\sigma,\eta}\,K^{rbf}$ and $\mathcal{S}_{\sigma,\eta}\,K^{rbf}$.

Observing the computational performances of quasi-local kernels in Table 5.6 and Table 5.7, we can notice that both the training and testing times are slightly higher than input kernels. This is not surprising as the quasi-local transformation introduce inevitably and systematically a considerable overhead in kernel computation. However, there is a consistent number of cases in which quasi-local kernels are faster than the corresponding input kernel. This is due to the fact that quasi-local kernels can have more discriminative power and thus they can execute the SVM margin maximization with a smaller number of optimization steps. In general, from the results, we can conclude that the quasi-local kernels are very rarely more than three times slower in comparison with the input kernels, and in few cases they are more than three times faster. This means that although they introduce a certain overhead on kernel computation, the SVM performances are not dramatically deteriorated by the quasi-local transformation of kernel functions.

5.5 Other Families of Operators

Other operators on kernels can be developed with different objectives. In this section we give some details about operators on kernels that are data-dependent, based on isotropic stationary kernels different from the RBF kernel and based on the spectral angle mapper (SAM). We also discuss the possibility of recursively apply the QL operators. Although we believe that these research directions can have a potential high impact, we still not performed in-depth analyses and empirical experiments. This section thus present the basis of the future works we intend to carry on regarding the framework of operators on kernels.

Data-dependent operators. A family of operators can be developed starting from the quasi-conformal transformation proposed by Amari and Wu [5] and briefly discussed in Chapter 2.1.3. Other operators can adjust the width parameter of RBF-based kernels a function of some properties of the feature-space. For example, assuming analogously to [206] to perform a primary SVM training with the original kernel, we can modify the width parameter according to a gradient induced in the feature-space by the proximity of the input examples to the separating hyperplane. Notice that, in the case of quasi-local operators, this approach can be applied to

the σ parameter but also to the η parameter of $\mathcal{S}_{\sigma,\eta}$ and $\mathcal{PS}_{\sigma,\eta}$. Intuitively, in this way, the margin is further enlarged thus minimizing the radius/margin based bounds. Issues that need to be addressed in this context concern the preservation of the PD property of resulting kernels with respect to the introduced gradient function, and the problem of defining the distances of examples from the separating hyperplane obtained with the primary SVM training.

Isotropic stationary operators. In principle, all isotropic stationary kernels are suitable to be applied on the distance between examples in the feature-space similarly to what we have shown for the $\mathcal{E}_\sigma$ operator. So we can define the class of feature-space isotropic stationary operators on kernels. The $\mathcal{E}_\sigma$ operator is a particular case, since it produces isotropic stationary kernels that are also local (i.e. as the distance between points increase, the kernel value tends to a constant). Conversely we can think also to operators that increase the importance of long range extrapolation of the input kernel in the feature-space.

SAM based operators. In the feature-space induced by a kernel K, SAM (see 2.1.3) can be written as:

$$\theta(\Phi(\mathbf{x}),\Phi(\mathbf{x}')) = \arccos\left(\frac{\langle\Phi(\mathbf{x}),\Phi(\mathbf{x}')\rangle}{\sqrt{\langle\Phi(\mathbf{x}),\Phi(\mathbf{x})\rangle}\sqrt{\langle\Phi(\mathbf{x}'),\Phi(\mathbf{x}')\rangle}}\right) \tag{5.20}$$

$$= \arccos\left(\frac{k(\mathbf{x},\mathbf{x}')}{\sqrt{k(\mathbf{x},\mathbf{x})}\sqrt{k(\mathbf{x}',\mathbf{x}')}}\right). \tag{5.21}$$

So with $\theta(\Phi(\mathbf{x}),\Phi(\mathbf{x}'))$ we can define a family of operators on kernels based on the SAM value calculated in the feature-space. We can also compute SAM-based feature-space distances and embed them in the class of isotropic stationary kernels. Moreover, operators combining both Euclidean distance and SAM values in the input and/or in feature-space are another family of operators that seems very interesting.

Recursive or iterative operators. Intuitively, every family of operators on kernels we discussed so far can be applied recursively. It means that the operators can be applied on kernels that have been produced by previous applications of the operators. Preliminary experiments showed us that the recursive application of $\mathcal{S}_{\sigma,\eta}$ and $\mathcal{PS}_{\sigma,\eta}$ starting from linear, RBF and polynomial kernels leads in the majority of the cases to a classification accuracy gain after some recursive applications of the operators, while the accuracy tends to decrease for a high number of recursive steps. In other words we have the intuition that some classes of operators could take advantage from recursive or iterative applications in terms of classification accuracy and that it is possible to move from a grid-search approach to estimate SVM and kernel parameters to another scenario in which the parameters are roughly chosen and the emphasis is put on the number of recursive applications of some operators.

Obviously recursion and iteration can cause serious computational drawbacks, and thus particular attention should be dedicated to the control of kernel evaluation complexity. In some

cases it could be possible to find non-recursive and/or approximate versions of possibly infinite recursive application of operators. Another possibility is the modification of operator parameters at each recursion step.

5.6 Conclusions

In this chapter, we have presented a novel family of operators on kernels that add locality information to the input kernel. The resulting kernels are called quasi-local kernels since they balance the global information of the original kernel (if it is a non-local kernel) with the local kernel with respect to the distance in the feature-space. The intuition is that the resulting kernels are able to maintain the original kernel behaviour for regions in which the information is not local, adapting instead the separating hyperplane following the local distribution of the data. We formally characterize the class of quasi-local kernels, showing that they are assured to be positive-definite. Moreover, we showed that the $\mathcal{E}_\sigma$ operator, on which the quasi-local kernels are based, defines the same neighbourhoods as the input kernel, that, applied to the K^{rbf} its behaviour is equivalent to a K^{rbf} with variable kernel width and we detailed a data-dependent strategy to choose the operator parameters.

The empirical evaluation on a total of 43 datasets carried out transforming the optimized input kernel performing a reduced model selection (no more than 12 parameter choices), showed that the quasi-local kernel are able to significantly improve the classification accuracies of the input kernels. In particular, the kernel

$$(\mathcal{S}_{\sigma,\eta}\, K)(\mathbf{x},\mathbf{x}') = K(\mathbf{x},\mathbf{x}') + \eta \cdot \exp\left(\frac{-K(\mathbf{x},\mathbf{x}) - K(\mathbf{x}',\mathbf{x}') + 2K(\mathbf{x},\mathbf{x}')}{\sigma}\right)$$

and the kernel

$$(\mathcal{P}_{\sigma,\eta}\, K)(\mathbf{x},\mathbf{x}') = K(\mathbf{x},\mathbf{x}') \cdot \left(1 + \eta \cdot \exp\left(\frac{-K(\mathbf{x},\mathbf{x}) - K(\mathbf{x}',\mathbf{x}') + 2K(\mathbf{x},\mathbf{x}')}{\sigma}\right)\right)$$

showed solid statistical evidence of improved generalization capability over input kernels especially for large datasets. Considering the K^{lin}, K^{rbf}, K^{pol} and K^{sig} input kernels, the present work suggests that the best classification accuracies are achieved by $\mathcal{P}_{\sigma,\eta}\, K^{rbf}$, $\mathcal{P}_{\sigma,\eta}\, K^{lin}$ and $\mathcal{S}_{\sigma,\eta}\, K^{rbf}$. We also showed that the computational performances of quasi-local kernels are not dramatically deteriorated with respect to the corresponding input kernels.

Generally speaking, the idea highlighted in this work is that, especially for large and complex problems, the true class boundary reflects a global behaviour that can be estimated using a proper kernel function but is very likely to have local adaptations and modifications. These local anomalies can be detected and introduced in the learning process mainly relying on the example distribution of the subregions. Combining global and high-level information with local and data-dependent analysis can be seen as a strategy that aims to "attack complex worlds" which is, according to a recent interview with prof. Vapnik[2], the main challenge machine learning

[2]Vladimir Vapnik, "Learning Has Just Started", interview available at `http://www.learningtheory.org`.

still has to address.

In this chapter we have thus showed how locality can be included in kernel methods acting directly on the kernel function and we demonstrated that this is beneficial for improving the classification capabilities of SVM. Although the computational performances are only slightly worsened if the QL kernels are used, it is not clear if locality can be exploited when scalability and computational performances are crucial. The next chapter deals with this question.

Chapter 6

Fast and Scalable Local Kernel Machines

Efficiently tackling large amount of data is one of the challenges of current research in kernel methods. Although most of the recently proposed techniques are based on different approaches, their common assumption is that scalability can be obtained by limiting or reducing the complexity of the decision function. In fact, very fast training algorithms have been developed for linear SVM [97, 49, 38, 22, 67], and they are effective for problems in which the linear separation is a good choice such as high-dimensional data. Other approaches permit the non-linear feature space setting, but they limit the complexity working with a reduced number of examples or a small set of support vectors [107], using active and online example selection [23, 21] or bounding the number of basis functions [96, 94] (see Chapter 2.2 for details).

In other words, in the works mentioned above, computational efficiency is sought bounding some aspects of the optimization problem. The result is an *approximation* of the optimal separation and a *smoothing* of the decision function which is more influenced by the global distribution of the examples than by the local behaviour of the unknown target function in each specific sub-region. The emerging approach is thus to trade locality for scalability permitting, with a potentially higher level of under-fitting, to achieve a fast convergence to an approximated solution of the optimization problem.

We show here that locality is not necessary related to computational inefficiency, but, instead, it can be the key factor to obtain very fast kernel methods without the need to smooth locally the global decision function. In our proposed approach, the model is formed by a set of accurate local models in fixed cardinality sub-regions of the training set and the prediction module uses for each query point the more appropriate local model. In this setting, we are not approximating with some level of inaccuracy the original SVM optimization problem, but we are considering separately different parts of the decision function with the potential advantage

of better capturing the local separation. So, instead of locally under-fit the decision function by globally smoothing it like approximated SVM solvers do, we search for decision functions that are locally-calculated and they are very similar (or even better) in terms of accuracy to the global decision function in the proximity of each testing point. This approach is theoretically supported also by the recent result obtained by Zakai and Ritov [210] that showed how, roughly speaking, "consistency implies local behaviour".

In this chapter we present **Fa**st **L**ocal **K**ernel **S**upport **V**ector **M**achine (FaLK-SVM), that precomputes a set of local SVMs covering with adjustable redundancy all the training set and uses for prediction a model which is the nearest (in terms of neighbourhood rank in feature space) to each testing point. FaLK-SVM is obtained introducing various strategies, detailed below, to speed-up the Local SVM approach (see [19], Chapter 3.4 and Chapter 4). The scalability is obtained approximating the Local SVM approach but, differently from the global approaches for fast kernel methods that approximate the decision function of global SVM by smoothing it, we soften the local assumption of Local SVM that the query point must be the central example of the neighbourhood on which the local SVM is trained; in this way we are able to use the same local SVM model for more than one testing point and it is also possible to precompute the local models during training. The locality of the approach is regulated by the neighbourhood size k and the method uses all the training points. Starting from the theory of Local Learning Algorithms [27, 194] (reviewed in Chapter 2.1.2) we derive generalization bounds for FaLK-SVM, and we analyse the computational complexity stating that, under reasonable assumptions, the training of our technique scales as $N \log N$ and the testing as $\log N$ where N is the training set size. We also introduce a procedure for local model selection in order to speed-up the selection of the parameters and better capturing local properties of the data. The empirical evaluation (with datasets with up to 3 million examples) shows that FaLK-SVM outperforms accurate and approximated SVM solvers both in term of generalization accuracy and computational performances. An attempt to computationally improve the Local SVM approach of [212] has been proposed by [42] where the idea is to train multiple SVMs on clusters found by a variant of k-means that take into consideration the class balancing of the clusters. However, the method does not follow directly the idea of kNNSVM, the main difference being that it can build only local linear models and the size of the clusters is not fixed (the variant of k-means does not have constrains on the cardinalities). The achieved computational performances are better than their formulation of Local SVM, but much worse than global SVM.

The effectiveness and efficiency of our approach is directly related to the role that locality plays in the learning problem. It is well known, for example, that for very high-dimensional problems such as text and document classification, the linear kernel performs better than non-linear kernels which are hard to tune and can be subject to the "curse of dimensionality" [14]. On the other hand, there are problems [16, 191] which inherently require non-linear approaches to be tackled. This is due to the combination of an intrinsic dimensionality which is low with respect to the training set size and of a decision function which is not simple to learn. In general, locality plays a more important role as the number of training examples increases because the ratio between training set cardinality and the dimensionality is more favourable and the local characteristics are more evident. Other signals for the need of a non-linear kernel

are the detection of uneven distributions in the datasets (typical of real-world problems), the monotonic increasing of accuracy with respect to training size also for already large amount of data and the inclusion of a high fraction of training examples in the support vector set. A representative of this class of problems is the Forest CoverType dataset [16] which is a large real dataset (more than half a million examples) with bounded dimensionality (54 features) that needs as many examples as possible to increase accuracy. We already in a very preliminary study [165] that our approach on this dataset is more accurate than SVM and much faster than both accurate and approximated SVM solvers.

The present contribution can be seen from multiple viewpoints. (i) FaLK-SVM is primarily a modification of the Local SVM approach [19, 212] that showed excellent classification performances but have dramatic computational problems, obtaining a scalable Local SVM classifier asymptotically much faster than SVM. (ii) The approach is also an enhancement of the local learning algorithms because the learning process is not delayed to the prediction phase (*lazy learning*) but the construction of the local models occurs during training (*eager learning*). (iii) From a practical viewpoint, FaLK-SVM is a novel kernel method which outperforms accurate and approximated SVM solvers for non high-dimensional datasets. (iv) For complex classification problems that require an high fraction of support vectors (SV), we exploit locality to avoid the need of bounding the number of total SV as existing approximated SVM solvers do for computational reasons. (v) More generally, our approach can also be seen as a framework for localizing and make scalable any kernel method, classifier and regressor and in general every data analysis that can be applied on sub-regions of the entire dataset.

The analysis concerning LLAs, Local SVM, fast large margin classifiers and Cover Trees that are all related with our work can be found in Chapter 2 in Chapter 3. We introduce FaLK-SVM in Section 6.1 and we analyse its learning bounds, complexity bounds, implementation, local model selection procedure and intuitive interpretation. Section 6.2 details the empirical evaluation with respect to accurate and approximated approaches. The proposed FaLK-SVM classifier and related tools are freely available with source code for research and educational purposes as part of the Fast Local Kernel Machine Library (FaLKM-lib) [163] (see Appendix A). The content of this chapter is also reported in [166].

6.1 FaLK-SVM: a Fast and Scalable Local Kernel Machine

In this section we introduce our novel technique detailing the way to precompute the local models during training (Section 6.1.1) and the strategies to reduce the number of local models (Section 6.1.2). We then describe the prediction mechanism in Section 6.1.2 and our approach for fast local model selection in Section 6.1.3. We then derive learning bounds for the approach in Section 6.1.4 before discussing the computational complexity in Section 6.1.5 and some details about the implementation (Section 6.1.6).

6.1.1 Precomputing the Local Models during Training Phase

For the local approach we are proposing here, we need to generalize the decision rule of kNNSVM (reported in Chapter 3.4) to the case in which the local model is trained on a set of points belonging to the k-neighbourhood of a point distinct, in the general case, from the query point. A modified decision function for a query point $\mathbf{q} \in \mathcal{H}$ and another (possibly different) point $\mathbf{t} \in \mathcal{H}$ is:

$$k\text{NNSVM}_{\mathbf{t}}(\mathbf{q}) = \text{sign}\left(\sum_{i=1}^{k} \alpha_{r_{\mathbf{t}}(i)} y_{r_{\mathbf{t}}(i)} K(\mathbf{x}_{r_{\mathbf{t}}(i)}, \mathbf{q}) + b\right) \tag{6.1}$$

where $r_{\mathbf{t}}(i)$ is the kNNSVM ordering function and $\alpha_{r_{\mathbf{t}}(i)}$ and b come from the training of an SVM on the k-neighbourhood of $\mathbf{t}$ in the feature space. In the following we will refer to $k\text{NNSVM}_{\mathbf{t}}(\mathbf{q})$ as being centered in $\mathbf{t}$, to $\mathbf{t}$ as the center of the model, and, if $\mathbf{t} \in \mathcal{X}$, to $V_{\mathbf{t}}$ as the Voronoi cell induced by $\mathbf{t}$ in $\mathcal{X}$, formally:

$$V_{\mathbf{t}} = \{\mathbf{p} \in \mathcal{H} \text{ s.t. } \|\mathbf{p} - \mathbf{t}\| \leq \|\mathbf{p} - \mathbf{x}\|, \ \forall \mathbf{x} \in \mathcal{X} \text{ with } t \neq \mathbf{x}\}.$$

The original decision function of kNNSVM corresponds to the case in which $\mathbf{t} = \mathbf{x}$, and thus $k\text{NNSVM}_q(\mathbf{q}) = k\text{NNSVM}(\mathbf{q})$.

kNNSVM requires that the training of an SVM on the k-neighbourhood of the query point must be performed in the prediction step. This approach is convenient only for problems in which there are very few points to test and the training of the local model is very fast, which are conditions that rarely holds in real-world classification problems, so we need to speed-up the prediction phase. The first modification of kNNSVM consists in predicting the label of a test point $\mathbf{q}$ using the local SVM model built on the k-neighbourhood of its nearest neighbour in $\mathcal{X}$. Formally, this can be written as:

$$k\text{NNSVM}_{\mathbf{t}}(\mathbf{q}) \text{ with } \mathbf{t} = \mathbf{x}_{r_{\mathbf{q}}(1)} \tag{6.2}$$

Notice that in situations where the k-neighbourhood contains only one class the local model does not find any separation and so it can adopt the majority rule for improving the computational performances.

With this formulation the local learning can switch from the *lazy learning* [3] setting of the original formulation of kNNSVM to the *eager learning* setting with clear advantages in terms of prediction step complexity. This is possible by computing the set of local SVM models for each $\mathbf{x} \in \mathcal{X}$ during the training phase obtaining the following set: $\mathcal{S} = \{(\mathbf{t}, k\text{NNSVM}_{\mathbf{t}}) \mid \mathbf{t} \in \mathcal{X}\}$, delaying to the testing phase only the retrieval (and the application) of the precomputed $k\text{NNSVM}_{\mathbf{t}}$ model such that $\mathbf{t} = \mathbf{x}_{r_{\mathbf{q}}(1)}$ for each query point $\mathbf{q}$.

This approximation slightly modifies also the approach of kNNSVM and of local learning algorithms. Instead of estimating the decision function for a *given* test example $\mathbf{q}$ and thus for a specific point in the input metric space, we are estimating a decision function for *each* Voronoi cell $V_{\mathbf{x}}$ induced by the training set in the input metric space. In this way, the construction of the models in the training phase requires the estimation of N local decision functions. The

model that is effectively used for the prediction of a test point $\mathbf{q}$ is simply the model build for the Voronoi region in which $\mathbf{q}$ lies ($V_{\mathbf{h}}$ with $\mathbf{h} = \mathbf{x}_{r_{\mathbf{q}}(1)}$) and can thus retrieved performing a nearest neighbour search for $\mathbf{q}$ in $\mathcal{X}$.

6.1.2 Reducing the Number of Local Models that Need to Be Trained

The pre-computation of the local models during the training phase introduced above, makes the prediction step much more computationally efficient, but a considerable overhead is added to the training phase. In fact, the training of an SVM for each training point can be slower than the training of a unique global SVM (especially for non small k values), so we introduce another modification of the method which aims to dramatically reduce the number of SVMs that need to be pre-computed. The idea is that we can relax the constraint that a query point $\mathbf{x}'$ is always evaluated using the model trained around its nearest training point (i.e. for the Voronoi region $V_{\mathbf{x}_h}$ with $h = r_{\mathbf{x}'(1)}$). The decision function of this approach can thus be

$$\mathsf{FastLSVM}(\mathbf{x}) = k\text{NNSVM}_{f(\mathbf{x})}(\mathbf{x}) \tag{6.3}$$

where $f : \mathcal{H} \mapsto \mathcal{C} \subseteq \mathcal{X}$ is a function mapping each unseen example $\mathbf{x}$ to a unique training example which is, accordingly to Eq. 6.1, the center of the local model that is used to evaluate $\mathbf{x}$.

Notice that if $f(\cdot) = \mathbf{x}_{r_{\cdot}(1)}$, we have that $\mathcal{C} = \mathcal{X}$ and that $\mathsf{FastLSVM}(\mathbf{x})$ is equivalent to the kNNSVM formulation of Eq. 6.2, and this can happen if we use *all* the examples in the training set as centers for local SVM models. In the general case, however, we select only a proper subset $\mathcal{C} \subset \mathcal{X}$ of points to be used as centers of kNNSVM models. In this case, if $\mathbf{x}_{r_{\mathbf{x}}(1)} \in \mathcal{C}$ then $f(\mathbf{x})$ can be defined as $f(\mathbf{x}) = r_{\mathbf{x}}(1)$, but if $\mathbf{x}_{r_{\mathbf{x}}(1)} \notin \mathcal{C}$ then $f(x)$ must be defined in a way such that the principle of locality is preserved and the retrieval of the model is fast at prediction time.

Two aspects needs to be addressed now: the strategy to select the subset $\mathcal{C}$ of $\mathcal{X}$, and the formulation of the function f associating each query example with an example in $\mathcal{C}$.

Selecting the centers of the local models

The approach we developed for selecting the set $\mathcal{C}$ of the centers of the local models is based on the idea that each training point must be in the k'-neighbourhood of at least one center with k' being a fixed parameter with $k' \leq k$. From a slightly different viewpoint, we need to cover the entire training set with a set of hyper-spheres whose centers will be the examples in $\mathcal{C}$ and each hyper-sphere contains exactly k' points. We can formalise this idea with the concept of k'-neighbourhood covering set:

Definition 1. *Given $k' \in \mathbb{N}$, a k'-neighbourhood covering set of centers $\mathcal{C} \subseteq \mathcal{X}$ is a subset of the training set such that the following holds:*

$$\bigcup_{\mathbf{c} \in \mathcal{C}} \left\{ \mathbf{x}_{r_{\mathbf{c}}(i)} \mid i = 1, \ldots, k' \right\} = \mathcal{X}.$$

Definition 1 means that the union of the sets of the k'-nearest neighbours of $\mathcal{C}$ corresponds to the whole training set. Theoretically, for a fixed k', the minimization of the number of local SVMs that we need to train can be obtained computing the SVMs centered on the points contained in the *minimal* k'-neighbourhood covering set of centers.

Definition 2. *The* Minimal k'-neighbourhood covering set of centers *is a k'-neighbourhood covering set $\mathcal{C} \subseteq \mathcal{X}$ which have the minimal cardinality.*

This problem is a special case of the *Set Cover Problem* [75, 95, 123] known as the *Minimum Sphere Set Covering Problem* (MSSC) [41] although in its original formulation one specifies the radius of the spheres rather than their cardinality in terms of points they contain. It is easy to show that MSSC is NP-hard but some efficient approximated results are available based on greedy approaches [45, 197], integer and linear programming [200].

In our case, however, we do not need the minimality of the constraints of the k'-neighbourhood covering set of centers to be strictly satisfied, because training some more local SVMs is acceptable instead of solving an NP-hard problem. Notice that, if we want less overlap between the k-neighbourhood we can act on the k' parameter increasing it. Moreover, our problem is somehow different from the MSSC problem because we define the hypersphere dimension using the cardinality of the examples it contains instead of the radius, and we require that the centers of the hyperspheres correspond to training points.

The heuristic procedure we developed can be seen as a modification of the greedy approach for the MSSC problem [45, 197]. The first neighbourhood is selected randomly choosing its center in $\mathcal{X}$, the following neighbourhoods are retrieved selecting the centers that are still not members of other neighbourhoods and are as far as possible from the already selected centers. The selection of the farthest example, still not included in the neighbourhoods, as the center of the next neighbourhood, is the counterpart of the selection of the set of points having the minimum overlapping with the already covered set of points used by the greedy approach to the MSSC (and Set Cover) problem.

For detailing the greedy approach we adopt, we need the concepts of minimum and maximum distance among a set of points A defined respectively as:

$$d(A) = \min \|\mathbf{x} - \mathbf{x}'\| \text{ with } \mathbf{x}, \mathbf{x}' \in A \text{ and } \mathbf{x} \neq \mathbf{x}'$$

and

$$D(A) = \max \|\mathbf{x} - \mathbf{x}'\| \text{ with } \mathbf{x}, \mathbf{x}' \in A.$$

In particular, the minimum distance between points in $\mathcal{X}$ is $m = d(\mathcal{X})$ and the maximum is $M = D(\mathcal{X})$. Our intention is to identify a system of subsets $S_i \subseteq \mathcal{X}$ with decreasing minimum distances $d(S_i)$; we can in this way define an ordering on the sets $\ldots \subset S_{i+1} \subset S_i \subset S_{i-1} \subset \ldots$ such that $\ldots > d(S_{i+1}) > d(S_i) > d(S_{i-1}) > \ldots$. With this strategy we can now choose the centers of the local models first in the set S_{i+1}, then in the set S_i and so on thus selecting first the centers that are assured to be distant at least $d(S_{i+1})$, then at least $d(S_i) < d(S_{i+1})$ and so on. More in detail, we require that in the ith set $S_i \subseteq \mathcal{X}$ the two nearest points are farther than

b^i with $b > 1$, i.e. they are subject to the constraint $d(S_i) > b^i$ with $b > 1$. The bound on the minimum distance $d(S_i)$ thus varies as powers of b depending on the set S_i.

The maximum i index of S_i is named *top* and the minimum is named *bot*, and they are univocally defined as those indexes satisfying $b^{top-1} \leq M < b^{top}$ and $b^{bot} < m \leq b^{bot+1}$. The S_i are recursively defined as:

$$\begin{cases} S_{top} &= \{\text{choose}(\mathcal{X})\} \\ S_i &= S_{i+1} \cup \underset{S \in \mathcal{X} \setminus S_{i+1}}{\text{argmax}} \left(|S| \text{ s.t. } d(S_{i+1} \cup S) > b^i\right) \qquad \text{for } i = top-1, \ldots, bot \end{cases}, \tag{6.4}$$

where choose(A) is a function that selects only one element of the non-empty set A. An example of choose() for our case can be the following definition that selects the example with the minimum index:

$$\text{choose}(A) = \mathbf{x}_i \text{ with } i = min(z \in \mathbb{N} | \mathbf{x}_z \in A).$$

Notice that, since S_i contains S_{i+1} we have that

$$S_{top} = \{\text{choose}(\mathcal{X})\} \subseteq S_{top-1} \subseteq \ldots \subseteq S_{bot+1} \subseteq S_{bot} = \mathcal{X} \tag{6.5}$$

and

$$d(S_{top}) = M > d(S_{top-1}) > \ldots > d(S_{bot+1}) > d(S_{bot}) = m.$$

We can now formalise the selection of the centers from $\mathcal{X}$ using the S_i sets. The first center $\mathbf{c}_1$ is simply the (only) example in S_{top}. The next center $\mathbf{c}_2$ is chosen among the non-empty S_l sets obtained removing from S_i the first center $\mathbf{c}_1$ and the points in its k'-neighbourhood; in particular $\mathbf{c}_2$ is chosen from the non-empty S_l with highest l. The general case for the $\mathbf{c}_j$ center is similar, with the only difference being that we remove from the S_i sets all the centers $\mathbf{c}_t$ with $t < j$ and their k'-neighbourhood. More formally:

$$\begin{cases} \mathbf{c}_1 &= \text{choose}(S_{top}) \\ \mathbf{c}_j &= \text{choose}(S_l) \text{ with } l = \max(m \in \mathbb{N} | S_m \setminus \mathcal{X}_{\mathbf{c}_{j-1}} \neq \emptyset) \end{cases}, \tag{6.6}$$

where

$$\mathcal{X}_{\mathbf{c}_{j-1}} = \bigcup_{l=1}^{j} \left\{ \mathbf{x}_{r_{\mathbf{c}_l}(h)} \, | h = 1, \ldots, k' \right\}.$$

is the union of all the neighbourhoods of the centers already included in $\mathcal{C}$.

We can briefly show that the $\mathcal{C}$ set found with Eq. 6.6 is a k'-neighbourhood covering set of centers. The iterative procedure for selecting the centers in $\mathcal{C}$ terminates when the choose() function cannot select a point from S_l because all S_j with $j = bot, \ldots, top$ are empty. Since for the lowest set S_{bot} we always have that $S_{bot} = \mathcal{X}$, this can happens only when $\mathcal{X}_{\mathbf{c}_{i-1}} = \mathcal{X}$. Noticing that $\mathcal{X}_{\mathbf{c}_i}$ in this situation is equivalent to the constraint of Definition 1, we can conclude that $\mathcal{C}$ is a k'-neighbourhood covering set of centers.

Computationally, the selection of the centers from the S_j sets with Eq. 6.6 can be performed

efficiently once the S_j are identified. More problematic is the construction of the nested set of S_j sets. We can however notice that the S_j sets have some points in common with the levels of Cover Trees. Firstly from Eq. 6.4 we can easily see that for each S_j set with $j < top$ all the points in it are at least distant as b^j because $d(S_j) > b^j$; this is equivalent to the separation invariant of Cover Trees reported in Section 3.5. Secondly, always from Eq. 6.4 we can conclude that each S_j is contained in every S_t set with $t < j$ as also explicated in Eq.6.5; this is equivalent to the nesting invariant of Cover Trees. The only constraint of our strategy to identify the S_j sets that is not respected by Cover Trees is the maximality of the set added to each S_j set to obtain S_{j+1}. However, the procedure to insert a new point in a Cover Tree is based on adding it to the highest possible level, and this is an (efficient) approximation of the maximality constraint we have in Eq. 6.4. Taking all these facts into consideration, we chose to use the levels of Cover Tree as the S_j sets from which we select the centers as reported in Eq. 6.6.

With this approach it is no longer required that a local SVM is trained for each training example, but we need to train only $|\mathcal{C}|$ SVMs centered on each $c \in \mathcal{C}$ obtaining the following models:

$$k\text{NNSVM}_{\mathbf{c}}(\mathbf{x}), \quad \forall \mathbf{c} \in \mathcal{C}.$$

Moreover if a neighbourhood contains only points belonging to one class the local model is the majority rule (specifically, unanimity) and the training of the SVM is avoided.

Figure 6.1 graphically shows the result of adopting the approach described above on an artificial simple dataset. This example has only the purpose to intuitively show how the approach works because the approach is developed for large datasets and for non-extreme values of the neighbourhood parameters.

Selecting the local models for testing points

Once the set of centers $\mathcal{C}$ is defined and the corresponding local models are trained, we need to select the proper model to use for predicting the label of a test point. A simple strategy we can adopt consists in selecting the model whose center $\mathbf{c} \in \mathcal{C}$ is the nearest center with respect to the testing example. Using the general definition of FastLSVM of Eq. 6.3 with $f(x) = r^{\mathcal{C}}_{\mathbf{x}}(1)$ where $r^{\mathcal{C}}$ corresponds to the reordering function defined in Eq. 3.12 performed on the $\mathcal{C}$ set instead of $\mathcal{X}$, the method, called FaLK-SVMc, is defined as:

$$\text{FaLK-SVMc}(\mathbf{x}) = k\text{NNSVM}_{\mathbf{c}}(\mathbf{x}) \text{ where } \mathbf{c} = \mathbf{x}_{r^{\mathcal{C}}_{\mathbf{x}}(1)} \tag{6.7}$$

FaLK-SVMc is convenient from the computational viewpoint, because it performs the nearest neighbour search on $\mathcal{C}$ only. However, it does not assure that the testing point is evaluated with the model for which it is the nearest in terms of neighbourhood. For example, a testing point $\mathbf{q}$ can be closer to $\mathbf{c}_1$ than $\mathbf{c}_2$ using the Euclidean distance, but at the same time we can have that $\mathbf{c}_1$ is the i-th nearest neighbour of $\mathbf{q}$ and $\mathbf{c}_2$ is the j-th nearest neighbour of $\mathbf{q}$ with $j < i$. In order to overcome the problem of FaLK-SVMc we propose to use, for a testing point $\mathbf{q}$, the model centered in the training point which is the nearest in terms of the ranking based on the neighbourhood cardinality to its training nearest neighbour. We can do this defining a function

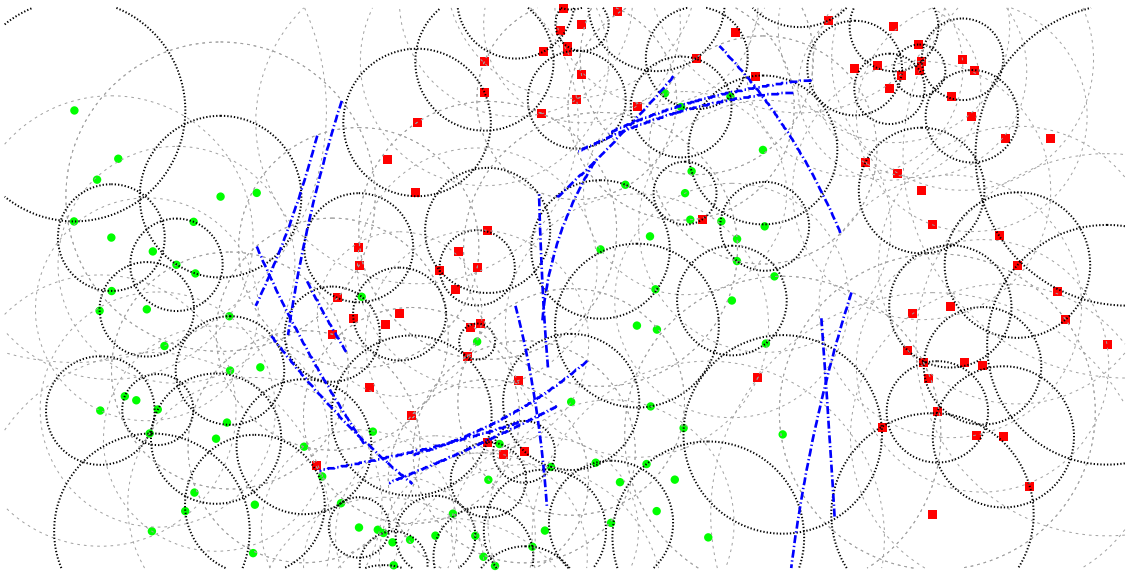


Figure 6.1: Graphical representation of FaLK-SVM using local models with $k' = 4$, $k = 15$, and local SVM with RBF kernel. The bold dotted circles highlights the k'-neighbourhoods covering all the training set (with some unavoidable redundancy), the thin dotted circles denotes the k-neighbourhoods on which the local models are trained. The local SVM (with RBF kernel) decision functions are drawn in blue. Notice that, due both to the adoption of the k'-neighbourhood cover set and to the fact that only a fraction of the neighbourhoods need to be trained, we have only 17 local decision functions for 185 points.

$cnt : \mathcal{X} \mapsto \mathcal{C}$ in the following way:

$$\begin{aligned} cnt(\mathbf{x}_i) &= \text{choose}(\{\mathbf{c}_z \in \mathcal{C} | \mathbf{x}_i = \mathbf{x}_{r_{\mathbf{c}_z}(h)}\}) \\ \text{where } h &= \min\left(t \in \{1, \ldots, k'\} | \mathbf{x}_{r_{\mathbf{c}_j}(t)} = \mathbf{x}_i \text{ and } \mathbf{c}_j \in \mathcal{C}\right). \end{aligned} \tag{6.8}$$

The cnt function finds, for each example $\mathbf{x}$, the minimum value h such that $\mathbf{x}$ is in the h-neighbourhood of at least one center $\mathbf{c} \in \mathcal{C}$; then, among the centers having $\mathbf{x}$ in their h-neighbourhoods, it select the center with the minimum index. In this way each training point is univocally assigned to a center and so the decision function of this approximation of Local SVM derivable from FastLSVM of Eq. 6.3 with $f(\mathbf{x}) = cnt(\mathbf{x})$, and called FaLK-SVM, is simply:

$$\text{FaLK-SVM}(\mathbf{x}) = k\text{NNSVM}_{cnt(\mathbf{t})}(\mathbf{x}) \text{ where } \mathbf{t} = \mathbf{x}_{r_{\mathbf{x}}(1)} \tag{6.9}$$

The association between training points and centers defined by Eq. 6.8 can be efficiently precomputed during the training phase, delaying to the testing phase only the retrieval of the nearest neighbour of the testing point and the evaluation of the local SVM model.

FaLK-SVM can be generalized for multiclass problems in the same way of kNNSVM, but in this chapter we focus on binary problems in order to better evaluate the approach.

6.1.3 FaLK-SVM with Local Model Selection: FaLK-SVMl

In this section we present a procedure for the model selection of local kernel machines and a variant of FaLK-SVM, called FaLK-SVMl, that implements it. In fact, with the local setting of the classification problem we are discussing in this thesis, it is also possible to efficiently tackle the complexity of the model selection phase. Basically, since the classifier trains a set of local models, we can perform the model selection in a grid-search setting on a subset of the local neighbourhoods. In this way we can efficiently estimate the global parameters of FaLK-SVM without considering all the training points during model selection. In addition to the localization of the model selection we could, in principle, localize the setting of the parameters as well, but the efficiency decreases and the over-fitting risk increases, so this option is not introduced in FaLK-SVMl.

In general, when training a kernel machine, it is crucial to choose a proper kernel, to carefully tune the kernel parameters and, for SVM, to set the soft margin regularisation constant C. Model selection is very often performed estimating the empirical error with different parameter choices and, in particular, one of the most effective and practical approaches for doing this is based on κ-fold cross-validation[1] with a grid search on parameter space. Given the following loss function for the two-class classification case

$$L(y, \mathrm{SVM}(\mathbf{x})) = \begin{cases} 0 & \text{if } y = \mathrm{SVM}(\mathbf{x}) \\ 1, & \text{if } y \neq \mathrm{SVM}(\mathbf{x}) \end{cases},$$

and partitioning the entire training set $\mathcal{X}$ in κ disjoint subsets (folds) $\mathcal{X}_f$ with $f = 1, \ldots, N$ with the same cardinality[2], the κ-fold cross validation (CV) procedure consists in finding the parameters of the classifier trained on $\mathcal{X} \setminus \mathcal{X}_f$ that permits to achieve the lowest error on the predicted $\mathcal{X}_f$ labels, averaged on each fold f. Although the effectiveness on testing accuracies of this last approach is very high, its main drawback concerns the computational overhead added to the training phase. In fact, the computational complexity of a κ-fold cross-validation (CV) run on a single parameter choice is in the order of κ times the training time; if we have p parameters to set and c possible choices for each parameter, the κ-fold cross-validation with grid selection is $\kappa \cdot c^p$ times slower than a single training of the classifier.

As a first step for defining the local models selection approach for FaLK-SVM, we define the local setting of model selection for kNNSVM.

Definition 3 (Local κ-fold CV model selection for kNNSVM)**.** *The procedure applies the κ-fold CV model selection on the k-neighbourhood of the query point.*

However, since the local model is used by kNNSVM only for the central point, the model selection should be performed in order to make the local models predictive especially for the very internal points. The idea thus consists in selecting the κ validation sets exclusively from the k'

[1] Although κ can be confused with the neighbourhood size k or with the kernel function K, κ is always used for denoting κ-fold CV, so the context should be sufficient to avoid ambiguity.

[2] Without loss of generality, we assume $|\mathcal{X}| \mod \kappa = 0$.

most internal points, taking as each training fold the union of the remaining k'-neighbourhood points and of the $k - k'$ most external points of the k-neighbourhood.

Definition 4 (Local k'-internal κ-fold CV model selection for kNNSVM)**.**
The procedure applies the κ-fold CV model selection on the k'-neighbourhood of the query point in the training set adding to each training fold the points in the k-neighbourhood that are not in the k'-neighbourhood with $k > k'$.

For FaLK-SVM we can apply the local k'-internal κ-fold CV for kNNSVM model selection on a randomly chosen training example and use the found parameters for all the local models. In order to be robust to possibly "outlier" neighbourhoods we perform the local k'-internal κ-fold CV on more than one k-neighbourhood choosing the parameters that minimize the average local k'-internal κ-fold CV error among the k-neighbourhoods.

Definition 5 (Local k'-internal κ-fold CV model selection for FaLK-SVM)**.**
The procedure applies the local k'-internal κ-fold CV for kNNSVM model selection on the k-neighbourhoods of $1 \leq m \leq |\mathcal{C}|$ randomly chosen centers selecting the parameters that minimize the average error rate among the m applications.

The variant of FaLK-SVM that adopts the local k'-internal κ-fold CV described in Definition 5 is named FaLK-SVMI.

In principle, FaLK-SVM can also estimate the parameters for each model independently, simply applying the local k'-internal κ-fold CV for kNNSVM model selection on each k-neighbourhood retrieved in the training set and using the found parameter setting for the training of the local SVM on the corresponing neighbourhood. In this way it is possible to better capture local properties of the data, but, on the other hand, the resulting approach can be very inefficient (a local model selection based on κ-fold cross-validation is required for *each* local model) and can cause over-fitting. For these reasons, FaLK-SVMI performs a local model selection with the objective to find the best global parameters.

In FaLK-SVMI, a particular strategy is devoted to the estimation of the σ parameter for the RBF kernel. As already introduced in Chapter 3.3, good choices for σ are based on the median (or other percentiles) of the distribution of distances. In our setting we can thus efficiently estimate σ for each local model without CV based model selection. With standard SVM, it has already shown by [39] that SVM with RBF and variable kernel width has good potentialities for classification. Notice that the local k'-internal κ-fold CV for FaLK-SVM model selection can still be used for σ in order to select which percentile of the distribution of the local distances is the optimal one.

6.1.4 Generalization Bounds for kNNSVM and FaLK-SVM

We already saw in Chapter 4.1 that the class of LLAs introduced by Bottou and Vapnik [27] and their framework based on the local risk minimization [194, 193], allow us to derive a bound on the risk of misclassification of kNNSVM. FaLK-SVM is not a LLA as intended by Bottou and Vapnik [27]. In fact, they consider only the learning approaches that compute the local function

for each specific testing point thus delaying the neighbourhood retrieval and model training until the testing points are available. However, we show here that rigourous generalization bounds for FaLK-SVM can be derived starting from the LLA ones.

We briefly need to recall the central theorem of local risk minimization.

Theorem 4 (Vapnik 2000 [193])**.** *For a testing point* $\mathbf{x}'$ *and with probability* $1-\eta$ *simultaneously for all bounded functions* $A \leq L(y, f(\mathbf{x}, \alpha)) \leq B$, $\alpha \in \Lambda$ *(where* Λ *is a set of parameters), and all locality functions* $0 \leq T(\mathbf{x}, \mathbf{x}_0, \beta) \leq 1$, $\beta \in (0, \infty)$*, the following inequality holds true:*

$$R^{LLA}(\alpha, \beta, \mathbf{x}') \leq \frac{\frac{1}{N}\sum_{i=1}^{N} L(y_i, f(\mathbf{x}_i, \alpha))T(\mathbf{x}_i, \mathbf{x}', \beta) + (B-A)\gamma(N, h^{\Sigma})}{\left|\frac{1}{N}\sum_{i=1}^{N} T(\mathbf{x}_i, \mathbf{x}', \beta) - \gamma(N, h^{\beta})\right|},$$

where

$$\gamma(N, h) = \sqrt{\frac{h\ln(2N/h+1) - \ln\eta/2}{N}},$$

and h^{Σ} *is the VC dimension of the set of functions* $L(y_i, f(\mathbf{x}_i, \alpha))T(\mathbf{x}_i, \mathbf{x}', \beta), \alpha \in \Lambda, \beta \in (0, \infty)$ *and* h^{β} *is the VC dimension of* $T(\mathbf{x}_i, \mathbf{x}', \beta)$

Starting from this theorem, in Chapter 4.1 we derived the risk for kNNSVM:

$$R^{k\text{NNSVM}}(\alpha, k, \mathbf{x}') \leq \frac{\frac{1}{N}k \cdot \nu_{\mathbf{x}'} + \gamma(N, h^{\Sigma})}{|\frac{1}{N}k - \gamma(N, 2)|} \tag{6.10}$$

where $\nu_{\mathbf{x}'}$ is the ratio of misclassified training points in the k-neighbourhood of $\mathbf{x}'$.

Now we show how it is possible to derive for FaLK-SVM a learning bound starting from the kNNSVM bound. First we need the following lemma.

Lemma 5. *Given a distribution* $f(\mathbf{x})$*, if a set* $\mathcal{X}$ *with* $|\mathcal{X}| = N$ *and a point* $\mathbf{x}$ *are i.i.d. drawn, the expectation for the query point* $\mathbf{x}$ *to lie in the Voronoi region of* $\mathbf{x}_i \in \mathcal{X}$ *is the same for each* $i = 1, \ldots, N$*. Formally:*

$$E_{\mathcal{X}}[P(\mathbf{x} \in V_{\mathbf{x}_i})] = 1/N \text{ for } i = 1, \ldots, N$$

Proof. Consider the following function returning 1 if the query point lies in the i-th Voronoi cell defined by the N points in the training set:

$$\widehat{v}_i^N(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{x} \in V_{\mathbf{x}_i} \text{ given } |\mathcal{X}| = N \\ 0, & \text{otherwise.} \end{cases}$$

With this function, we can re-write the expectation for the query point to lie in the Voronoi

region of $\mathbf{x}_i \in \mathcal{X}$ as:

$$E_{\mathcal{X}}[P(\mathbf{x} \in V_{\mathbf{x}_i})] = E_{\mathcal{X}}\left[\int_{V_{\mathbf{x}_i}} f(\mathbf{x})\, d\mathbf{x}\right] = E_{\mathcal{X}}\left[\int_{\mathbf{x}} \widehat{v}_i^N f(\mathbf{x})\, d\mathbf{x}\right] = E_{\mathcal{X},\mathbf{x}}[\widehat{v}_i^N(\mathbf{x})]$$

For the i.i.d. hypothesis on $f(\mathbf{x})$ we can write $E_{\mathcal{X},\mathbf{x}}[\widehat{v}_i^N(\mathbf{x})]$ as:

$$\begin{aligned}
E_{\mathcal{X},\mathbf{x}}[\widehat{v}_i^N(\mathbf{x})] &= \int_{\mathcal{X}}\int_{\mathbf{x}} \widehat{v}_i^N(\mathbf{x}) \cdot f(\mathbf{x}_1) \cdot f(\mathbf{x}_2) \cdot \ldots \cdot f(\mathbf{x}_{N-1}) \cdot f(\mathbf{x}_N) f(\mathbf{x})\, d\mathcal{X}\, d\mathbf{x} && (6.11)\\
&= \int_{\mathcal{X}} f(\mathbf{x}_1) \cdot f(\mathbf{x}_2) \cdot \ldots \cdot f(\mathbf{x}_{N-1}) \cdot f(\mathbf{x}_N) \int_{\mathbf{x}} \widehat{v}_i^N(\mathbf{x}) \cdot f(\mathbf{x})\, d\mathcal{X}\, d\mathbf{x} && (6.12)\\
&= \int_{\mathcal{X}} f(\mathbf{x}_1) \cdot f(\mathbf{x}_2) \cdot \ldots \cdot f(\mathbf{x}_{N-1}) \cdot f(\mathbf{x}_N)\, d\mathcal{X} \int_{\mathbf{x}} \widehat{v}_i^N(\mathbf{x}) \cdot f(\mathbf{x})\, d\mathbf{x} && (6.13)\\
&= \int_{\mathbf{x}} \widehat{v}_i^N(\mathbf{x}) \cdot f(\mathbf{x})\, d\mathbf{x} && (6.14)\\
&= E_{\mathbf{x}}[\widehat{v}_i^N(\mathbf{x})]. && (6.15)
\end{aligned}$$

Since the expectation for a test point of lying in a Voronoi cell $E_{\mathbf{x}}[\widehat{v}_i^N(\mathbf{x})]$ is independent from the random sampling of the training point, it must be the same for each Voronoi cell, so $E_{\mathbf{x}}[\widehat{v}_1^N(\mathbf{x})] = E_{\mathbf{x}}[\widehat{v}_2^N(\mathbf{x})] = \ldots = E_{\mathbf{x}}[\widehat{v}_{N-1}^N(\mathbf{x})] = E_{\mathbf{x}}[\widehat{v}_N^N(\mathbf{x})]$ and, since $\sum_{i=1}^{N} E_{\mathbf{x}}\left[\widehat{v}_i^N(\mathbf{x})\right] = E_{\mathbf{x}}\left[\sum_{i=1}^{N} \widehat{v}_i^N(\mathbf{x})\right] = E_{\mathbf{x}}[1] = 1$, the hypothesis follows directly. □

FaLK-SVM precomputes local models to be used for testing points lying in sub-regions (kNN Voronoi cells) of the training set. The risk of FaLK-SVM can be defined using the risk of kNNSVM, supposing that $\mathbf{x}' \in V_{\mathbf{x}_i}$, as:

$$R^{\text{FaLK-SVM}}(\alpha, k, \mathbf{x}') = R^{k\text{NNSVM}}(\alpha, k, r_{\mathbf{x}'}(1)) + \lambda_i = R^{k\text{NNSVM}}(\alpha, k, \mathbf{x}_i) + \lambda_i \qquad (6.16)$$

where λ_i is due to the approximation introduced using for the training of the model that is used to predict the label of the query point $\mathbf{x}$' on the k-neighbourhood of $r_{\mathbf{x}'}(1)$ instead of $\mathbf{x}'$ itself. If we consider $k' = 1$, the approximation is due to the fact that $\{r_{\mathbf{c}}(i) |\, i = 1, \ldots, k\}$ and $\{r_{\mathbf{x}'}(i) |\, i = 1, \ldots, k\}$ can be slightly different; however, considering a non-trivial value for k, the differences between the two sets are possible only for the very peripheral points of the neighbourhoods which are those that influence less the shape of the decision function in the central region. We will empirically show that λ_i is a small penalizing constant that still permits to achieve lower risks than SVM also using k' values higher than 1.

From Eq. 6.16, and using Theorem 5, we can generalize the bound for FaLK-SVM for each

possible testing point, thus switching from a traditional LLA setting to a (local) eager setting:

$$
\begin{aligned}
E_{\mathcal{X}}[R^{\text{FaLK-SVM}}(\alpha,k)] &= \sum_{i=1}^{N} R^{\text{FaLK-SVM}}(\alpha,k,\mathbf{x}') \cdot E_{\mathcal{X}}[P(\mathbf{x}' \in V_{\mathbf{x}_i})] && (6.17)\\
&= \frac{1}{N}\sum_{i=1}^{N} R^{\text{FaLK-SVM}}(\alpha,k,\mathbf{x}') && (6.18)\\
&= \frac{1}{N}\sum_{i=1}^{N} (R^{k\text{NNSVM}}(\alpha,k,\mathbf{x}_i) + \lambda_i) && (6.19)\\
&\leq \frac{1}{N}\sum_{i=1}^{N} \frac{\frac{1}{N}k \cdot \nu_{\mathbf{x}_i} + \gamma(N,h^{\Sigma})}{|\frac{1}{N}k - \gamma(N,2)|} + \overline{\lambda} && (6.20)\\
&= \frac{\frac{k}{N}\sum_{i=1}^{N} \nu_{\mathbf{x}_i} + \gamma(N,h^{\Sigma})}{|k - N\gamma(N,2)|} + \overline{\lambda} && (6.21)
\end{aligned}
$$

where $\overline{\lambda} = \frac{1}{N}\sum_i \lambda_i$ is the term due to the use of the kNNSVM risk for FaLK-SVM as discussed above.

Note that $\nu_{\mathbf{x}_i}$ can vary dramatically with respect to i. Some local models can in fact be very simple or even trivial (all local neighbourhood belongs to the same class), whereas other can be extremely noisy.

6.1.5 Computational Complexity Analysis

We analyse here the computational performances of FaLK-SVM from the theoretical complexity viewpoint. The training phase of FaLK-SVM can be subdivided in four steps:

- the building of the Cover Tree that scales as $\mathcal{O}(N \log N)$;
- the retrieval of the local models that scales as $\mathcal{O}(|\mathcal{C}| \cdot k \log N)$;
- the univocal assignment of each point to a k'-neighbourhood that scales as $\mathcal{O}(N)$;
- the training of the local SVM models that scales as $\mathcal{O}(|\mathcal{C}| \cdot k^3)$.

The overall training time, considering the worst case in which $k' = 1$ so $|\mathcal{C}| = N$, thus scales as:

$$\mathcal{O}(N \log N + \mathcal{C} \cdot k \log N + N + \mathcal{C} \cdot k^3) = \mathcal{O}(kN \cdot \max(\log N, k^2))$$

that, considering a reasonably low and fixed value for k as happens in practice for large datasets, is sub-quadratic, and in particular $\mathcal{O}(N \log N)$, in the number of training points.

For the testing phase of FaLK-SVM we can distinguish two steps (for each testing point):

- the retrieval of the nearest training point that scales as $\mathcal{O}(\log N)$;

- the prediction of the testing label using the selected local SVM model that scales as $\mathcal{O}(k)$.

The testing can thus be performed in $\mathcal{O}(\max(\log N, k))$, so it is logarithmic in n. FaLK-SVMc is even faster because it scales as $\mathcal{O}(\max(\log |\mathcal{C}|, k)) \leq \mathcal{O}(\max(\log N, k))$.

FaLK-SVM is thus asymptotically faster than SVM (also considering the worst case in which SVM scales quadratically and $k' = 1$) and all the classifiers taking more than $\mathcal{O}(N \log N)$ for training and $\mathcal{O}(\log N)$ for testing. Moreover, notice that FaLK-SVM can be very easily parallelised differently from SVM whose parallelization, although possible [211, 64], is rather critical; for FaLK-SVM is sufficient that, every time the points for a model are retrieved, the training of the local SVM is performed on a different processor. In this way the time complexity of FaLK-SVM can be further lowered to $\mathcal{O}(N \cdot \max(k \log N, k^3/N_{proc}))$.

Another advantage of FaLK-SVM over SVM is space complexity. Since FaLK-SVM performs SVM training on small subregions (assuming a reasonable low k), there are no problems of fitting the kernel matrix into main memory. The overall required space is, in fact, $\mathcal{O}(N + k^2)$, i.e. linear in N, that is much lower than SVM space complexity of $\mathcal{O}(N^2)$ which forces, for large datasets, the discarding of some kernel values thus increasing SVM time complexity due to the need of recomputing them. Analysing the space required to store the trained model in secondary storage devices (e.g. hard disks), we can notice that FaLK-SVM needs to save in the model file the entire set of local models; although we store the models with pointers to the training set points, we need to maintain the whole training set in the model file (or give as input for the testing module both the model file and the original training set). FaLK-SVM, in other words, needs to store the training set also in the model file, differently from SVM that needs to store only the support vectors (whose number however grows linearly with N).

Curse of dimensionality. Although not explicitly considered here, Cover Trees have a constant in the complexity bounds depending on the so-called doubling constant [46, 101] which is a robust estimation of the intrinsic dimensionality of the data. Notice that the intrinsic dimensionality of a dataset can be much lower than the dimensionality intended simply as the number of features. Regardless of the doubling constant, FaLK-SVM maintains the derived complexity bounds[3] with respect to N, but the overhead introduced for building the Cover Tree and retrieving the k-neighbourhoods can be very high. This drawback, due to the well-known problem of the *curse of dimensionality* that affects also SVM with local kernels [14], is not however crucial here, as we are considering non-linear classification problems that are not high-dimensional. In fact, apart from computational problems, high-dimensional problems are typically tackled by approaches not related with the concept of locality (eg. linear SVM instead of SVM with a RBF kernel).

6.1.6 Implementation and Availability

FaLK-SVM is freely available as part of the Fast Local Kernel Machine Library [163], see Appendix A. The pseudo-code for the training phase is reported in Algorithm 3 and for the

[3]The high intrinsic dimensionality can cause the need for an high value of $|\mathcal{C}|$, but in the bound we already considered the worst case in which $k' = 1$ and thus $|\mathcal{C}| = N$.

Algorithm 3 FaLK-SVM-train (training set **x**[], training size **n**, neighbourhood size **k**, assignment neighbourhood size **k'**)

```
1: models[] ⇐ null  //the set of models
2: modelPtrs[] ⇐ null  //the set pointers to the models
3: c ⇐ 0  //the counter for the centers of the models
4: indexes[] ⇐ {1, ..., N}  //the indexes for centers selection
5: Randomize indexes  //randomize the indexes
6: for i ⇐ 1 to N do
7:     index ⇐ indexes[i]  //get the i-th index
8:     if modelPtrs[index] = null then  //if the point has not been assigned to a model...
9:         localPoints[] ⇐ get ordered kNN of x[i]  //...retrieve its k-neighbourhood ...
10:        models[c] ⇐ SVMtrain on localPoints[]  //...train a local SVM...
11:        modelPtrs[index] ⇐ models[c]  //...and assign the center to the trained model.
12:        for j = 1 to k' do  //Assign the model to the k'<k nearest neighbours of the center
13:            ind ⇐ get index of localPoints[j]
14:            if modelPtrs[ind] = null then  //assign the points in the k'-neighbourhood ...
15:                modelPtrs[ind] ⇐ models[c]  //...to the c-th model
16:            end if
17:        end for
18:        c ⇐ c+1
19:    end if
20: end for
21: return models, modelPtrs
```

Algorithm 4 FaLK-SVM-test (training set **x**[], points-to-model pointers **modelPtrs**, Local SVM models **models**, query point **q**)

```
1: Set p = get NN of q in x
2: Set nnIndex = get index of p
3: return label = SVMpredict q with modelPtrs[nnIndex]
```

testing phase in Algorithm 4 (both without explicitly using Cover Trees and without minimizing t of Eq. 6.8 for clearness).

6.2 Empirical Analysis

The empirical analysis is organized into three experiments performed with different objectives and using different datasets. The first experiment (Section 6.2.1) has the objective of assessing the generalization performances of FaLK-SVM with respect to SVM (using LibSVM) and to kNNSVM (using FkNNSVM) and thus assessing if FaLK-SVM is more accurate than SVM and if it is a good approximation of kNNSVM. For this experiment we use 25 non-large datasets. The second experiment (Section 6.2.2) focuses on comparing the classification accuracies and the computational performances of FaLK-SVM (and its variants FaLK-SVMc and FaLK-SVMl) with respect to SVM (using LibSVM) on large datasets. For this experiment we use 8 datasets

dataset name	# of features	# of points	class balancing	dataset name	# of features	# of points	class balancing
sonar	60	208	53%/47%	fourclass	2	862	64%/36%
heart	13	270	56%/44%	tic-tac-toe	9	958	65%/35%
mushrooms	112	300	53%/47%	mam	5	961	54%/46%
haberman	3	306	74%/26%	numer	24	1000	70%/30%
liver	6	345	58%/42%	splice	60	1000	52%/48%
ionosphere	34	351	64%/36%	spambase	57	1000	57%/43%
vote	15	435	61%/39%	vehicle	21	1243	76%/24%
musk1	166	476	57%/43%	cmc	7	1473	57%/43%
hill-valley	100	606	51%/49%	ijcnn1	22	1500	68%/32%
breast	10	683	65%/35%	a1a	123	1605	76%/24%
australian	14	690	56%/44%	chess	35	2130	52%/48%
transfusion	4	748	76%/24%	astro	4	3089	65%/35%
diabetes	8	768	65%/35%				

Table 6.1: The 25 binary-class datasets of the first empirical experiment.

with training set cardinalities ranging from about 50k examples to more than 1 million. The third experiment (Section 6.2.3) is performed in order to understand (i) if FaLK-SVM has better scalability and accuracy performances than LibSVM and a number of approximated SVM solvers (CVM, BVM, LASVM, CPSP and USVM) and (ii) which are the computational and accuracy differences between FaLK-SVM, FaLK-SVMc and FaLK-SVMl. For this last experiment we use 4 datasets with increasing training set size up to 3 million examples. The experiments, if not differently specified, are carried out on an AMD Athlon$^{\text{TM}}$ 64 X2 Dual Core Processor 5000+, 2600MHz, with 3.56Gb of RAM with Linux operating system.

6.2.1 Experiment 1: Comparison of FaLK-SVM with LibSVM and FkNNSVM

In this evaluation we compare SVM (using LibSVM), kNNSVM (using FkNNSVM) and FaLK-SVM on 25 non-large datasets, with the objective of studying the generalization performances of kNNSVM with respect to SVM and the level of approximation introduced by FaLK-SVM to the FkNNSVM algorithm.

Experimental protocol

The datasets are listed in Table 6.1; they are retrieved from the UCI [7] and STATLOG [128] repositories, with cardinality between 200 and 3100 points (some datasets have been randomly sub-sampled), dimensionality lower than 200, not very unbalanced, and they are all scaled in the $[0, 1]$ interval. The comparison is carried out using three different kernel functions (linear, RBF and homogeneous polynomial), in a 10-fold CV experimental setting. Internal to each training fold the model selection is performed with a nested 10-fold CV choosing the parameters in the following ranges. The regularisation parameter C is chosen for all methods in the set

$\{2^{-2}, 2^{-1}, \ldots, 2^{9}, 2^{10}\}$, the width parameter σ of the RBF kernel in $\{2^{-5}, 2^{-4}, \ldots, 2^{2}, 2^{3}\}$, the degree of the polynomial kernels in $\{1, 2, 3\}$. The neighbourhood parameter k for FkNNSVM and FaLK-SVM is selected by the cross-validation procedure in the set $\{2^1, 2^2, \ldots, 2^9, 2^{10}, |\mathcal{X}|\}$ where $|\mathcal{X}|$ is the cardinality of the training set[4], while the k' parameter of FaLK-SVM is fixed to $k/2$ which is a value that penalizes accuracy for scalability because we want to test a value that can permit good computational results for large and very large datasets.

Results and discussion

Table 6.2 reports the accuracy results of all tested methods and kernels. In addition to the mean ranks reported in the figure, the Wilcoxon Signed Rank Test [202, 61] with $\alpha = 0.05$ applied to detect statistical differences between pairs of methods using the same kernel, highlights that FkNNSVM is significantly better than LibSVM for the linear and polynomial kernels, whereas for the RBF kernel no significant differences are detected, although the mean rank of FkNNSVM with RBF kernel is lower than LibSVM with RBF kernel. Applied to FaLK-SVM, the Wilcoxon Signed Rank Test detects a significant difference with respect to LibSVM only for the linear kernel. If we perform the Friedman test [72], the null hypothesis is violated, but, according to the Nemenyi post-hoc test [132] the only method that is statistically different from the others is SVM with linear kernel.

The observation that FkNNSVM is significantly better than SVM if a non-local kernel is used, is a confirmation of what we already noticed in Chapter 4 and in [164]. Using the RBF kernel no significant differences are detected, although the mean rank of FkNNSVM with RBF kernel is lower than LibSVM with RBF kernel. This is mainly due to the fact that SVM with RBF kernel is already very accurate and significant improvements over it are very difficult. We may also say that locality is already included in the RBF kernel and thus, at least for non-large datasets, the adoption of a local method is somehow equivalent. Regarding FaLK-SVM, significant differences with respect to LibSVM are detected only for the linear kernel. Although FaLK-SVM does not achieve the accuracy results of FkNNSVM, if we look to the mean ranks, we can conclude that the approximation on the kNNSVM approach introduced in FaLK-SVM still permits to achieve slightly better results than SVM also on non-large datasets, confirming our preliminary analysis detailed in [165]. These results also indicates that the $\overline{\lambda}$ constant introduced in the risk of FaLK-SVM (Section 6.1.4), due to the approximations introduced to the kNNSVM approach, is small enough to assure higher generalization accuracies with respect to SVM.

The overall outcome of this experiment is that FaLK-SVM is a good approximation of FkNNSVM that maintains a little advantage over SVM and it is particularly effective with the RBF kernel with respect to linear and polynomial kernels. Notice that the experiment is carried out using small datasets in which locality is very likely to play a marginal role differently large datasets in which it can be crucial.

[4]For dataset with less than 1024 points some k values are of course not tested.

dataset	LibSVM			FkNNSVM			FaLK-SVM		
	K^{lin}	K^{rbf}	K^{hpol}	K^{lin}	K^{rbf}	K^{hpol}	K^{lin}	K^{rbf}	K^{hpol}
sonar	74.52	87.83	83.16	**89.36**	86.90	87.40	84.55	87.88	84.05
heart	**84.81**	82.22	**84.81**	**84.81**	81.11	**84.81**	83.70	81.85	83.70
mushrooms	97.99	98.33	98.32	98.67	98.33	98.6	**99.00**	**99.00**	**99.00**
haberman	73.20	73.20	72.89	**75.82**	75.16	74.18	73.25	73.20	73.87
liver	68.71	**74.24**	71.90	73.64	73.96	73.94	70.73	71.92	71.92
ionosphere	88.04	93.72	88.88	93.75	**94.59**	93.75	86.91	94.01	89.18
vote	94.95	96.32	94.95	96.32	**96.33**	96.32	94.94	96.32	94.94
musk1	86.55	94.54	93.07	89.44	**94.96**	91.17	87.18	93.90	92.43
hill-valley	63.70	**66.00**	63.70	64.86	65.18	64.86	65.17	64.03	65.00
breast	**96.78**	**96.78**	**96.78**	96.49	96.49	96.35	96.19	96.49	96.19
australian	85.50	84.78	84.20	84.78	**85.50**	84.92	85.07	85.07	84.78
transfusion	76.21	77.40	76.47	79.81	78.74	79.81	79.67	78.87	**79.94**
diabetes	76.54	76.54	76.68	76.81	**78.24**	77.07	75.90	76.68	75.12
fourclass	77.39	**100.00**	78.66	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
tic-tac-toe	98.33	99.68	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
mam	82.10	82.63	81.27	**82.95**	82.73	82.85	81.80	82.63	80.97
numer	**77.00**	75.90	76.50	76.30	75.70	76.00	76.70	74.70	75.90
splice	80.41	**86.70**	86.60	80.41	86.30	86.60	78.30	86.20	86.60
spambase	89.80	90.60	89.80	90.60	90.50	90.60	**90.70**	90.60	**90.70**
vehicle	82.71	84.16	84.80	82.78	84.64	84.71	83.27	84.72	**85.04**
cmc	59.26	65.45	64.16	62.46	**67.72**	63.61	63.61	65.31	64.36
ijcnn1	85.53	93.94	92.73	93.93	93.47	93.60	92.80	**94.47**	93.20
a1a	**83.43**	81.94	**83.43**	82.87	82.06	82.87	82.87	82.06	82.87
chess	96.57	98.45	98.03	97.84	**98.50**	98.08	97.32	98.45	98.08
astro	95.34	96.73	96.89	96.96	96.92	**97.05**	96.96	96.67	96.86
mean rank	7.04	4.60	5.80	4.38	3.86	4.02	5.72	4.56	5.02

Table 6.2: 10-fold CV accuracy results of LibSVM, FkNNSVM and FaLK-SVM with four kernel functions on the 25 dataset of the first experiment. The best results for each dataset are highlighted in bold (taking into account all decimal values).

6.2.2 Experiment 2: FaLK-SVM, FaLK-SVMc and FaLK-SVMl vs. LibSVM and FkNN on Large Datasets

In this experiment we apply FaLK-SVM, FaLK-SVMc, FaLK-SVMl, LibSVM on 8 large datasets comparing the computational and generalization performances using the RBF kernel, because linear or polynomial kernels have very low accuracy results on the considered problems. We also add to the comparison the kNN classifier (implemented with Cover Trees and called FkNN) using the Euclidean distance.

dataset name	# of feat.	train. points	testing points	class balancing	original source
ijcnn1	22	49990	91701	90%/10%	LibSVM rep. [36]
cov-type	54	100000	481010	51%/49%	LibSVM rep. [36] orig. from [16]
census-inc	41	199523	99762	94%/6%	UCI rep. [7]
cod-rna	8	364651	121549	67%/33%	Uzilov et al.[191]
intr-det	40	1026588	311029	79%/21%	UCI KDD rep. [84]
2-spirals	2	100000	100000	50%/50%	Synthetic [165], Chapter 4.3.3
ndcc	5	100000	100000	61%/39%	Synthetic [187]
checker-b	2	300000	100000	50%/50%	Synthetic (e.g. see [190])

Table 6.3: The 8 large datasets of the second empirical experiment.

Experimental protocol

The datasets considered in this experiment are listed in Table 6.3 with the corresponding sources and are all scaled in the $[0, 1]$ interval. They range from a training set cardinality of about 50k points to more than a million, whereas the dimensionality is not high (always under 60) with separated testing sets. In order to select the parameters a 10-fold CV procedure is performed in the training set (apart from FaLK-SVMl) choosing the values in the following sets: $C \in \{2^{-2}, 2^{-1}, \ldots, 2^{9}, 2^{10}\}$, $\sigma \in \{2^{-15}, 2^{-14}, \ldots, 2^{4}, 2^{5}\}$, k for FaLK-SVM in $\{250, 500, 1000, 2000, 4000, 8000\}$ with $k' = k/2$, and k for FkNNSVM in $\{1, 3, 5, 9, 15, 21,\ 31, 51, 71, 101, 151\}$. FaLK-SVM does not necessarily test all values for k because if the maximum empirical accuracy is found for a specific value of k, for example $k = 500$, and for the following value, in this case $k = 1000$, the maximum is lower, the remaining higher values of k are not tested. Due to the computational resources necessary for performing model selection, especially for LibSVM, we performed the cross-validation runs on a Linux-based TORQUE cluster with 20 nodes. For FaLK-SVMl the local model selection is performed on 10 local models, $C \in \{2^0, 2^2, 2^4, 2^6\}$, $k \in \{500, 1000, 2000, 4000\}$, σ locally estimated with the $1st$, $10th$, $50th$ or $90th$ percentile of the distribution of the distances.

Results and discussion

Table 6.4 reports the generalization accuracies of the analysed classifiers. Looking at the mean ranks, we can see that FaLK-SVM is the more accurate (it achieves the best results in half of the datasets), followed by FaLK-SVMl. LibSVM and FaLK-SVMc seem to perform very similar but little worse than FaLK-SVM and FaLK-SVMl. Not surprisingly, FkNN performs poorly in almost all the datasets, except for the intr-det dataset in which it achieves the best result. According to the Wilcoxon Signed Rank Test [202, 61] FaLK-SVM is significantly more accurate than LibSVM, whereas, excluding FkNN, no other significant differences are detected. Apart for the intr-det datasets that have slightly different distribution in the training and testing sets (some types of network attacks are present in the test set only), the best empirical accuracies are always very

dataset	FkNN		LibSVM		FaLK-SVM		FaLK-SVMc		FaLK-SVMl
	10f-CV	test	10f-CV	test	10f-CV	test	10f-CV	test	test
ijcnn1	97.37	96.64	98.99	97.98	99.04	**98.04**	98.96	97.98	98.03
cov-type	91.73	91.99	92.60	92.83	92.68	**92.89**	92.44	92.60	92.84
census-inc	94.53	94.52	95.14	**95.13**	95.07	95.07	95.00	94.99	94.99
cod-rna	95.88	96.25	97.18	97.17	97.19	97.23	97.06	97.09	**97.29**
intr-det	99.74	**92.04**	99.89	91.77	99.74	91.97	99.69	92.01	91.91
2-spirals	88.43	88.43	85.18	85.29	88.42	**88.47**	88.29	88.45	88.30
ndcc	85.47	84.99	86.66	86.21	86.63	**86.29**	86.33	85.93	86.24
checker-b	94.31	94.08	94.46	94.21	94.46	94.21	94.45	94.19	**94.23**
test acc. mean rank		4.25		3.25		1.63		3.38	2.50

Table 6.4: Empirical (using 10-fold CV) and generalization accuracies of FkNN, LibSVM, FaLK-SVM, FaLK-SVMc and FaLK-SVMl on the 8 large datasets. The best generalization accuracy for each dataset is highlighted in bold. The last line reports the mean rank of each method among the 8 datasets.

similar to the generalization accuracies meaning that all techniques avoid over-fitting.

The training times are reported in Table 6.5 together with the speed-ups of FaLK-SVM, FaLK-SVMc and FaLK-SVMl with respect to LibSVM. We can notice that the speed-ups achieved by FaLK-SVM and FaLK-SVMc are always greater than 4.7 and in the majority of the cases they are at least one order of magnitude faster than LibSVM. FaLK-SVMc turns out to be generally faster than FaLK-SVM; although the two classifiers implements the same training algorithm, this happens because the model selection chooses for FaLK-SVMc a lower value of k with respect to FaLK-SVM. This is reasonable because FaLK-SVMc is less accurate than FaLK-SVM in choosing the nearest model for a testing point, and this causes an higher value of the $\overline{\lambda}$ constant that increases the risk of FaLK-SVMc with respect to FaLK-SVM (see Eq. 6.16 and Eq. 6.17). So using a lower k (and thus a lower k') tends to have more models in the proximity of the testing point making the choice less problematic. FaLK-SVMl is sometimes slower than LibSVM, but we have to consider that FaLK-SVMl includes model selection, whereas for the other methods the time needed by model selection is not considered in the training time, so, practically speaking, FaLK-SVMl is the fastest method if the optimal parameters are not *a priori* known.

The testing times required by the analysed methods are reported in Table 6.6. As expected FaLK-SVMc is the fastest among all methods with speed-up over LibSVM ranging from more than 2 to almost 200. FaLK-SVM and FaLK-SVMl are also generally faster than LibSVM with only one case in which the testing time is about two times slower.

This experiment showed that for 8 non high-dimensional datasets, our approach outperforms a state-of-the-art accurate SVM solver both in terms of generalization accuracies and computational performances. Although we have an additional parameter to tune (k), FaLK-SVM and FaLK-SVMc are faster enough to maintain the performance advantages over LibSVM also for model selection (we choose k in a small set of values). Moreover, with FaLK-SVMl we addressed

dataset	LibSVM	FaLK-SVM		FaLK-SVMc		FaLK-SVMl	
	training time (s)	training time (s)	speed-up on LibSVM	training time (s)	speed-up on LibSVM	train. time with l.m.s.	speed-up on LibSVM
ijcnn1	102	**15**	6.8	**15**	6.8	1850	0.1
cov-type	8362	88	95.0	**38**	220.1	1214	6.9
census-inc	13541	6047	4.7	**2391**	5.7	10271	1.3
cod-rna	9777	395	24.8	**225**	43.5	579	16.9
intr-det	5262	286	18.4	**284**	18.5	450	11.7
2-spirals	4043	188	21.5	**81**	49.9	3442	1.2
ndcc	1487	302	4.9	**92**	16.2	4609	0.3
checker-b	6047	**334**	18.1	366	16.5	1374	4.4

Table 6.5: Training times of LibSVM, FaLK-SVM, FaLK-SVMc and FaLK-SVMl and the speed-ups of the three local methods with respect to LibSVM. The best training time for each dataset is highlighted in bold.

the problem of model selection with a local approach to set the parameters; FaLK-SVMl showed to overcome LibSVM for the generalization accuracy and the time it needs to perform both local model selection and training is at least comparable (faster in 7 cases on a total of 8) with the time needed by LibSVM to perform the training only.

dataset	LibSVM	FaLK-SVM		FaLK-SVMc		FaLK-SVMl	
	testing time (s)	testing time (s)	speed-up on LibSVM	testing time (s)	speed-up on LibSVM	testing time (s)	speed-up on LibSVM
ijcnn1	43	32	1.3	**5**	8.6	36	1.2
cov-type	2795	202	13.8	**73**	38.3	191	14.6
census-inc	597	1347	0.4	**58**	10.3	1328	0.4
cod-rna	396	261	1.5	**58**	6.8	259	1.5
intr-det	192	146	1.3	**76**	2.5	149	1.3
2-spirals	957	10	95.7	**5**	191.4	18	53.2
ndcc	148	61	2.4	**7**	21.1	61	2.4
checker-b	167	10	16.7	**7**	23.9	**7**	23.9

Table 6.6: Testing times of LibSVM, FaLK-SVM, FaLK-SVMc and FaLK-SVMl and the speed-ups of the three local methods with respect to LibSVM. The best testing time for each dataset is highlighted in bold.

6.2.3 Experiment 3: Comparison of Scalability Performances of FaLK-SVM, FaLK-SVMc, FaLK-SVMl, LibSVM and Approximated SVM Solvers

In this experiment we test the scalability performances of our techniques (FaLK-SVM, FaLK-SVMc, FaLK-SVMl) on training sets with increasing sizes using the RBF kernel against LibSVM

and the approximated SVM solvers called CVM, LASVM, USVM, BVM, CPSP and presented in Chapter 2.2. Although we apply all the classifiers with the same protocol on the same datasets, we report, for clearness, the results in two parts: the comparison of FaLK-SVM with LibSVM and the approximated SVM solvers are in Section 6.2.3, the comparison of FaLK-SVM with its variants FaLK-SVMc and FaLK-SVMl are in Section 6.2.3.

Experimental protocol

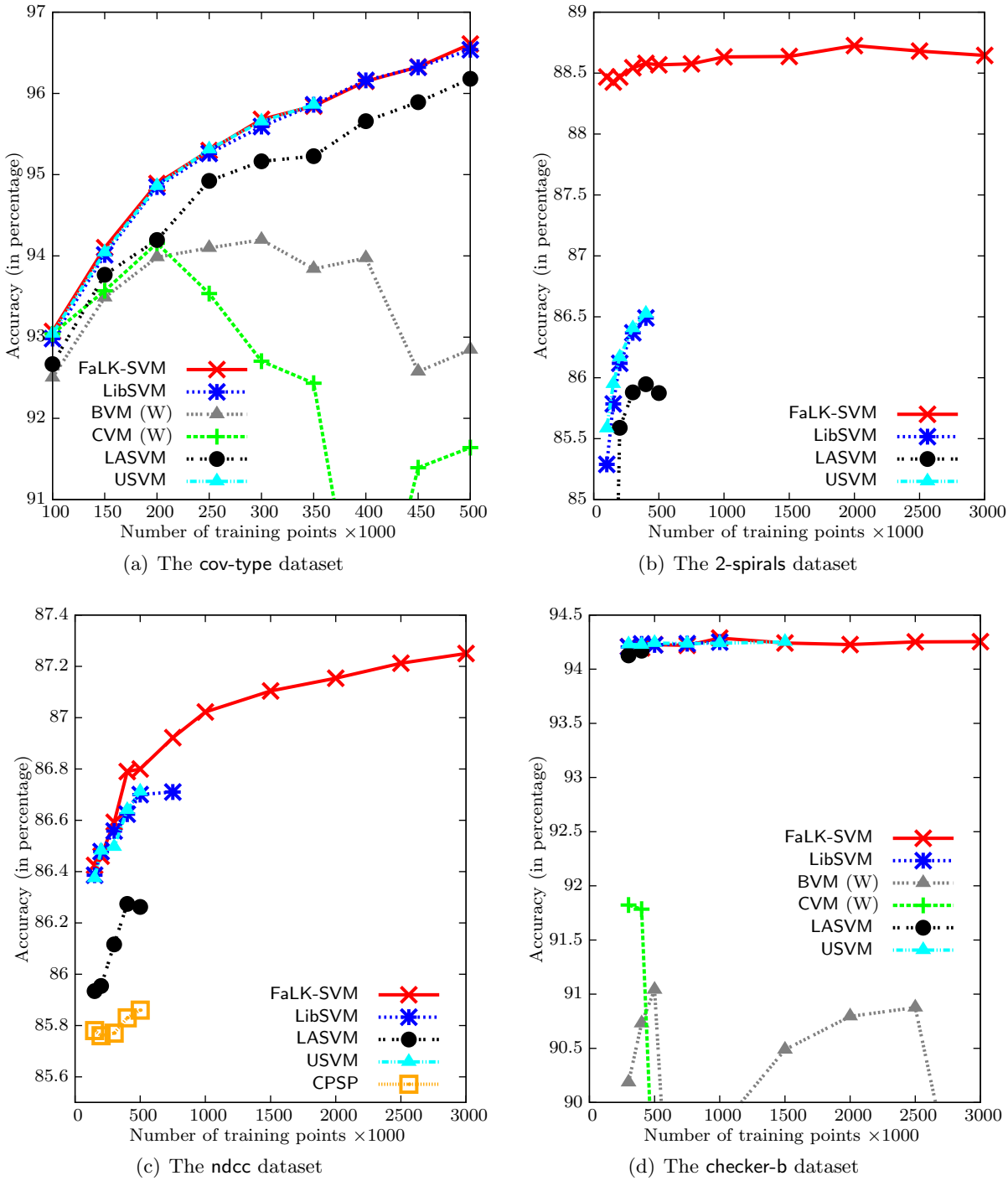
We consider here the datasets of the previous experiment listed in Table 6.3 for which we can further enlarge the training set size. This is possible for four datasets: the cov-type dataset (full training set of 500k points) and the three artificial datasets named 2-spirals, ndcc and checker-b (up to 3 million points). For cov-type the testing set is reduced to 50k examples (the other examples are added to the training set) so the accuracy results are not directly comparable to the previous experiment.

The model selection for all the classifiers (with the exception of FaLK-SVMl that performs internally a local model selection) is performed on the smallest training set only, using the chosen parameter for all the higher training set sizes. This is necessary, especially for LibSVM and approximated SVM solvers, for computational reasons. For LibSVM, BVM, CVM, USVM (with the convex concave procedure) and CPSP, we performed cross validation for C and σ using the same setting of the previous experiment. The default threshold value ϵ for the stopping criteria are maintained: 10^{-3} for LibSVM, FaLK-SVM, LASVM and 10^{-1} for CPSP while CVM and BVM automatically choose the value based on the data at each application. We set the same size of the kernel cache (100M) for all the methods. The maximum number of core vectors for CVM and BVM is 50000 (the default value), the maximum number of basis vectors for CPSP is set to 1000. We also tested FaLK-SVMl using the same setting of the previous experiment. Each algorithm is tested for training set sizes requiring no more than 100000 seconds (more than 27 hours) for training.

Since the authors of BVM [190] and CVM [189] declared the Linux implementation of their techniques deprecated (see the authors reply to [117] available on BVM webpage), we use the Windows executables on a Intel™ Pentium™ D Dual Core CPU 3.40GHz with 2Gb of RAM running Windows XP instead of the AMD Athlon™ 64 X2 Dual Core Processor 5000+, 2600MHz, with 3.56Gb of RAM with Linux operating system used for all the other classifiers. Because of the use of different operating systems and hardware for BVM and CVM, the two techniques are not directly comparable to the other methods. However, since preliminary results showed that the Linux version of BVM on the AMD Athlon™ machine and the Windows version of BVM on the Intel™ Pentium™ machine are very similar in terms of computational time, we present the computational results of all the methods in the same figures.

Results and discussion: FaLK-SVM vs LibSVM and approximated SVM solvers

Figure 6.2 shows the generalization accuracies of the methods at increasing training set sizes. Some methods do not appear in the figures due to low generalization results or computational difficulties that cause abnormal terminations of the algorithms, and some accuracy results for



(a) The cov-type dataset

(b) The 2-spirals dataset

(c) The ndcc dataset

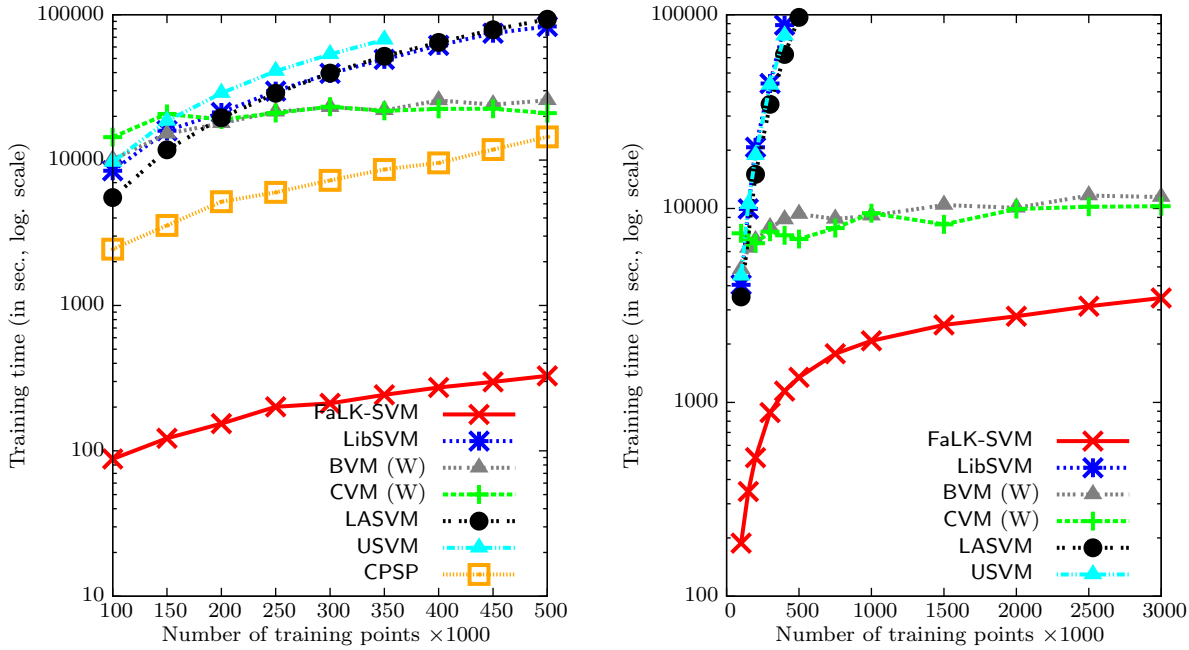
(d) The checker-b dataset

Figure 6.2: Generalization accuracies obtained using FaLK-SVM, LibSVM, BVM (in Windows), CVM (in Windows), LASVM, USVM and CPSP on the cov-type, 2-spirals, ndcc and checker-b datasets with increasing training set sizes. Some accuracies are missing due to the excessive computational requirements (more than 100000 seconds for training) of the corresponding method for large training set sizes.

large training set sizes are not present due to excessive computational time required for training (more than 100000 seconds). We can observe that it is very important to use as many points as possible in order to increase the accuracies for the cov-type and ndcc datasets. The same consideration can be done for the 2-spirals data, although FaLK-SVM already starts from very high accuracies and the increment is limited, while for the checker-b dataset the increment of the accuracies is negligible for almost all the methods. For the checker-b dataset, the enlarging of the training set is not motivated from the accuracy viewpoint, but we still use it as a benchmark for the computational performances.

Comparing the generalization accuracies of Figure 6.2 among the tested methods, we can see that FaLK-SVM is almost always on top for the four datasets. The only methods that seem to give results comparable with FaLK-SVM ones (apart from the 2-spirals dataset) are LibSVM and USVM and they are able, in few cases, to slightly improve the FaLK-SVM results (LibSVM for 2 training set sizes for cov-type and checker-b, USVM for 2 training set sizes for cov-type and checker-b and 1 for ndcc). The results of the online and active learning approach of LASVM are slightly lower than FaLK-SVM, LibSVM and USVM. CPSP gives acceptable results in only one case, and for the 2-spirals and checker-b datasets it suffers from numerical problems maybe due to the scaling of the features in the $[0, 1]$ interval. Enlarging the maximum number of basis functions for CPSP gives higher accuracies but the computational time needed to build the models is too high. The results we achieve here for LibSVM and LASVM on the cov-type datasets are a little higher than the results in [23] (about 1% better both for 100k and $500k$ training set sizes), and we believe that this is due to the model selection approach we used here that is performed with an exhaustive cross-validation grid search for C and σ. As we can notice in Figure 6.2, we experienced stability problems for CVM and BVM, even if we used the Windows binaries as suggested by the authors.

The training computational performances shown in Figure 6.3 highlight that FaLK-SVM is always much faster than the alternative techniques that are competitive from the accuracy viewpoint. In fact, although CVM and BVM show good scalability performances and in two times they overcome the performances of FaLK-SVM, we noticed from Figure 6.2 that their generalization abilities are poor. The scaling behaviours of LibSVM, LASVM and USVM are very similar (among the three methods LibSVM is the fastest for ndcc, LASVM is the fastest for 2-spirals and USVM is the fastest for checker-b) but substantially worse than FaLK-SVM one (FaLK-SVM is always at least one order of magnitude faster with speedups increasing with the training set sizes). The methods that achieve acceptable accuracy results on smaller training set size (i.e. LibSVM, LASVM, USVM) are not applicable when the number of training examples increases sensibly because of poor computational scalability performances; this is evident for the 2-spirals, ndcc and checker-b datasets in which the training times of LibSVM, LASVM, USVM exceed 100000 seconds as soon as the training set cardinality approaches one million (the only exception is USVM that is applicable on 1.5 training examples of the checker-b dataset). On the contrary, FaLK-SVM tackles datasets of 3 millions examples in the order of minutes or few hours. An experiment comparing LibSVM and LASVM on the cov-type dataset with conclusions similar to ours is reported by [23] in which however LASVM is about a third faster than LibSVM whereas here LibSVM slightly overcomes LASVM; this is probably due to the fact that for LASVM the



(a) The cov-type dataset (b) The 2-spirals dataset

(c) The ndcc dataset (d) The checker-b dataset

Figure 6.3: Training times of FaLK-SVM, LibSVM, BVM (in Windows), CVM (in Windows), LASVM, USVM and CPSP on the cov-type, 2-spirals, ndcc and checker-b datasets with increasing training set sizes. The times (in seconds) are reported in logarithmic scale.

only available implementation is the original one by [23] whereas LibSVM is frequently updated and improved. Finally, CPSP performs slightly better than LibSVM, LASVM and USVM.

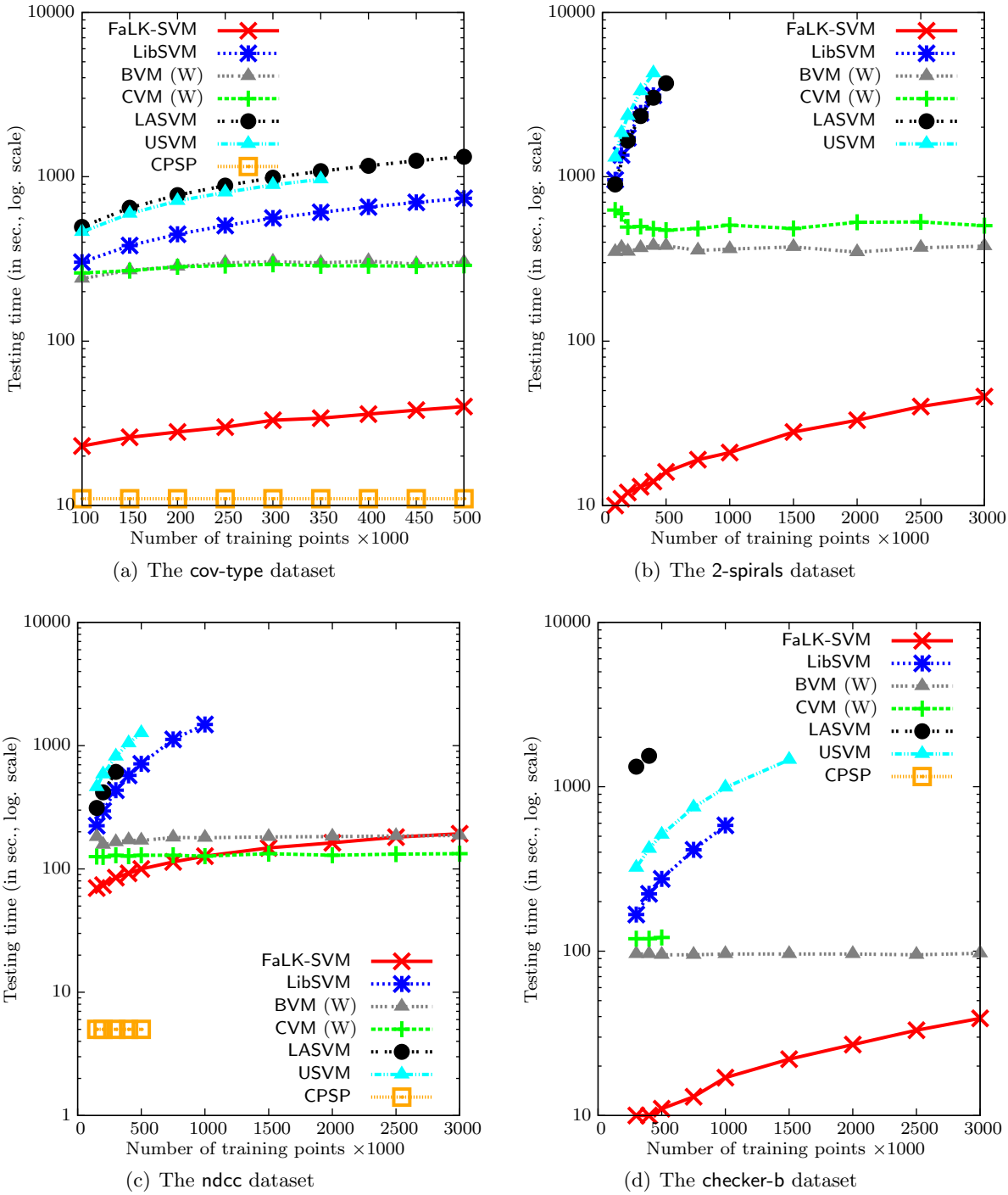
The computational performances of the prediction phase are reported in Figure 6.4. Also in this case the performances of FaLK-SVM are excellent: only CPSP and CVM are faster in 2 datasets than FaLK-SVM, but their corresponding generalization accuracies are low. As expected, CPSP achieves very fast predictions because it limits the number of basis function to 1000 and thus for each testing points no more than 1000 kernel functions are computed. LibSVM, LASVM and USVM achieve similar results also in testing performances and, apart from small training sets for the ndcc dataset, they are at least one order of magnitude slower than FaLK-SVM and the difference grows for large training set sizes.

The overall conclusion we can draw about the scalability of the proposed techniques is that, at least for these 4 non high-dimensional datasets, FaLK-SVM is substantially better than the state-of-the-art kernel methods for classification, and this is achieved without affecting the accuracy performances that showed to be always at least as good as the best alternative technique. Apart for LibSVM, we have to say that the available code of the other tested techniques has not been recently updated and for this reason it is possible to argue that higher performances with more optimized implementations of the tested approaches could be reached. It is also necessary to underline that LASVM, USVM, CPSP, BVM and CVM have been prevalently tested in literature on datasets with high dimensionality or, apart for cov-type, on datasets not requiring highly non-linear decision functions. The tested approximated non-linear SVM solvers are thus indicated for data in which the linear kernel is not the optimal choice, but, at the same time, the decision function can be accurately reconstructed with a reduced amount of information (number of examples, support vectors or basis functions).

Results and Discussion: Comparison between FaLK-SVM, FaLK-SVMc and FaLK-SVMl

Figure 6.5 reports the comparison of the generalization accuracies of FaLK-SVM, FaLK-SVMc and FaLK-SVMl at increasing training set size. The computational performances for the training phase are reported in Figure 6.6, and for the testing phase in Figure 6.7.

From the accuracy viewpoint, we can notice that FaLK-SVM is almost always slightly more accurate than FaLK-SVMc as we expected. FaLK-SVMl, apart from checker-b, is less accurate than FaLK-SVM for the smaller training set sizes, and this is due to the fact that FaLK-SVM performs a full grid search for model selection whereas FaLK-SVMl adopts the very fast local model selection approach. However, FaLK-SVMl is more competitive with respect to FaLK-SVM as the training set sizes increases and this is reasonable because FaLK-SVM uses for all the training set sizes the parameters found for the smaller training sets, and it is not assured that the best cross-validated parameters are the same for sub-sampled sets with different cardinality. For example, as the number of training points increases, the radius of the local neighbourhoods decreases if we maintain the same k and k' values, and the original value for the width parameter of the RBF kernel can no longer be the optimal one. For this reason, in the case of cov-type and ndcc datasets, FaLK-SVMl achieves higher accuracies than FaLK-SVM for the largest training



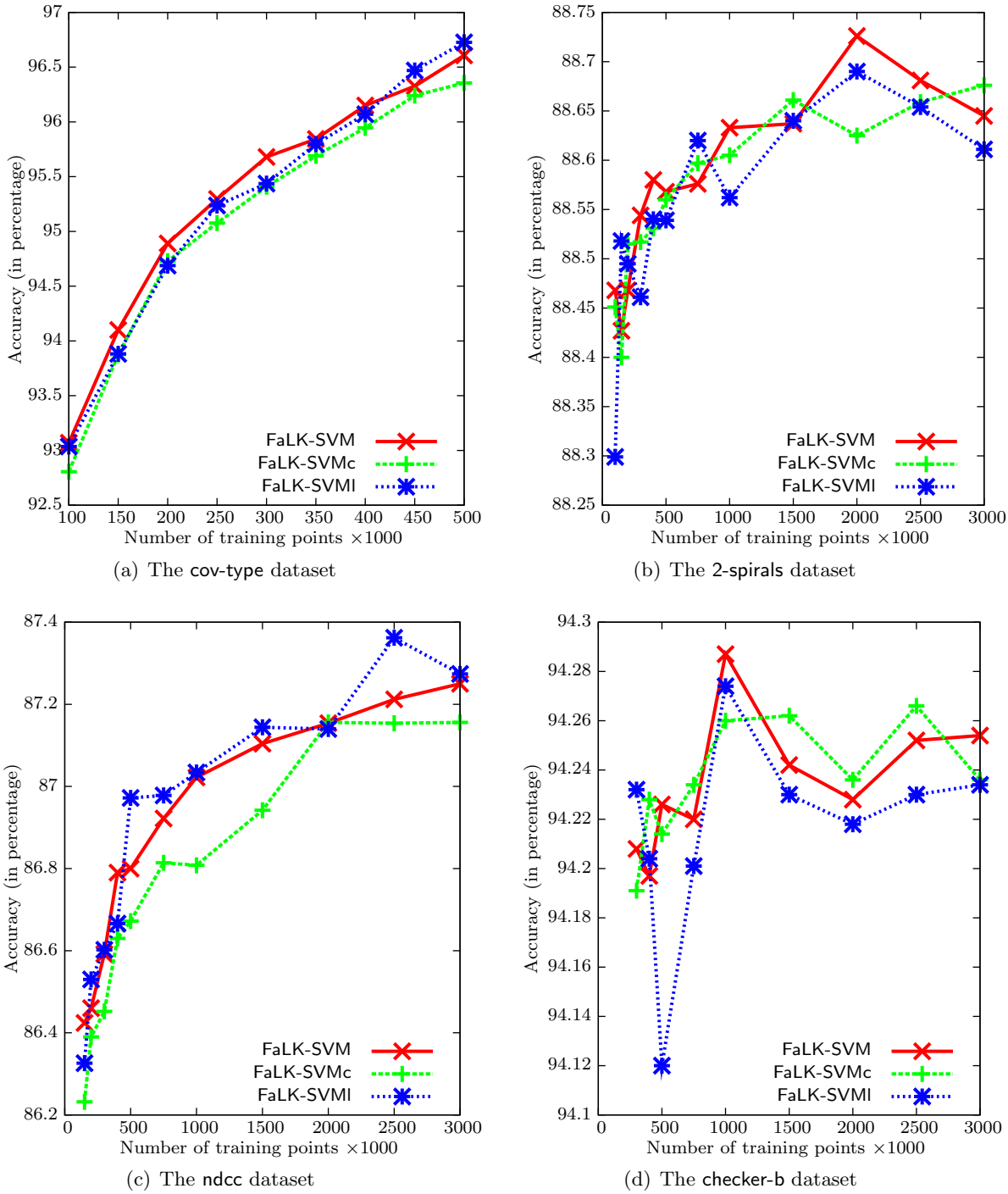
(a) The cov-type dataset

(b) The 2-spirals dataset

(c) The ndcc dataset

(d) The checker-b dataset

Figure 6.4: Testing times of FaLK-SVM, LibSVM, BVM (in Windows), CVM (in Windows), LASVM, USVM and CPSP on the cov-type, 2-spirals, ndcc and checker-b datasets with increasing training set sizes. The times (in seconds) are reported in logarithmic scale. Some testing times are missing due to the excessive computational requirements (more than 100000 seconds for training) of the corresponding method for large training set sizes.



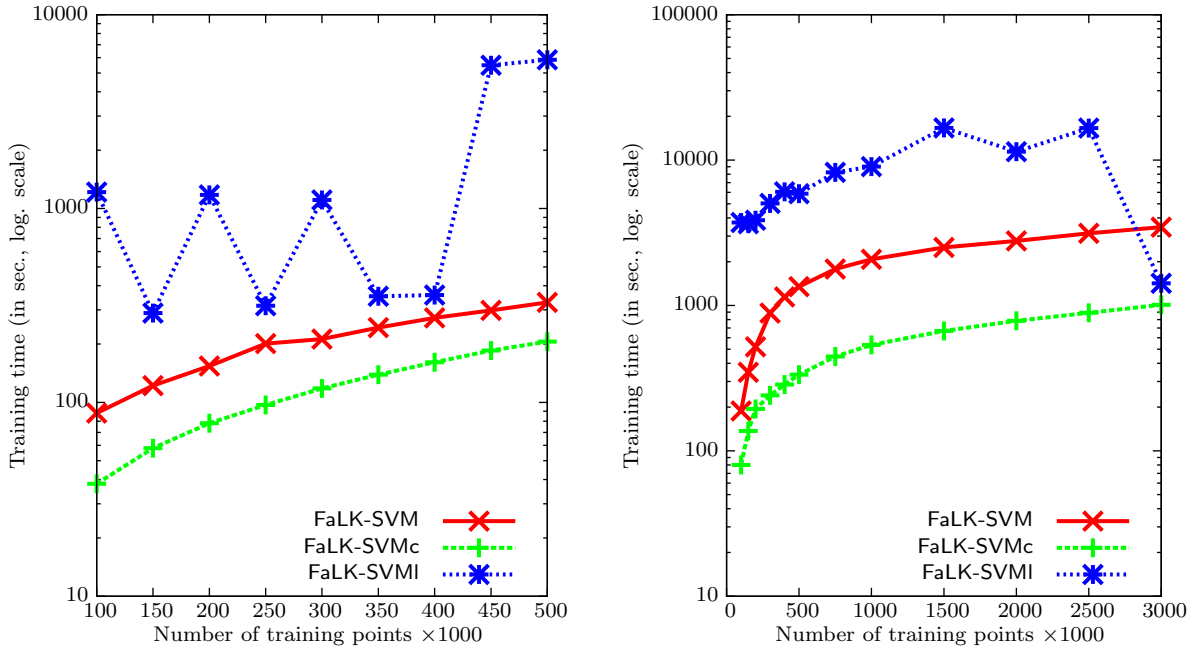
(a) The cov-type dataset

(b) The 2-spirals dataset

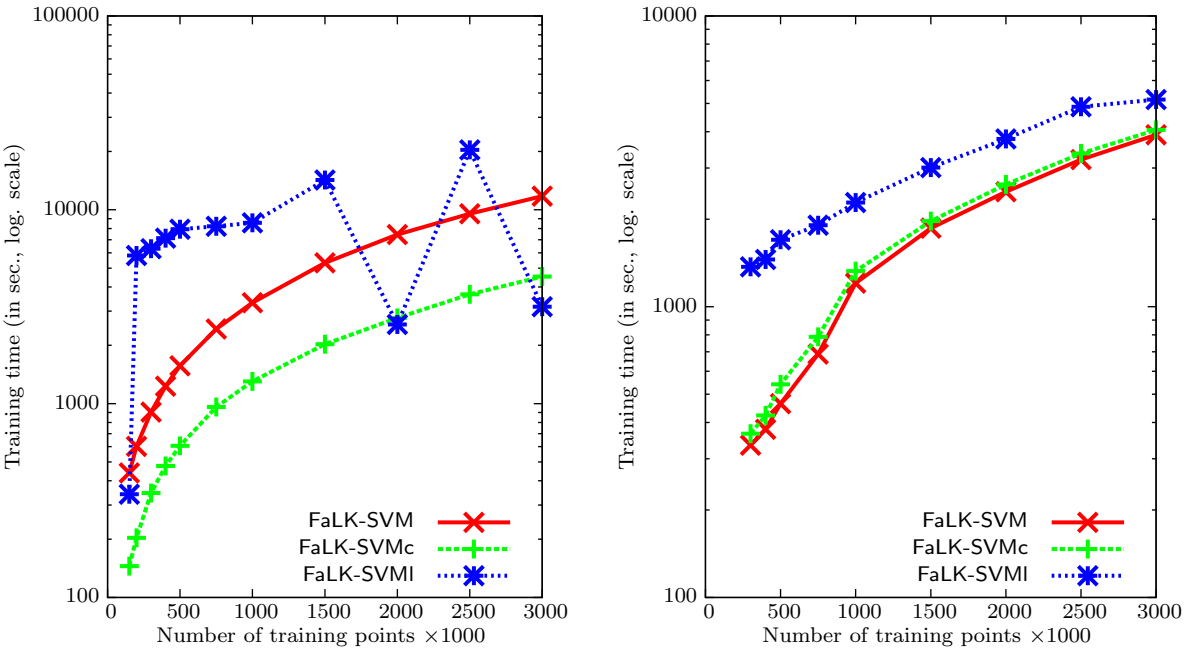
(c) The ndcc dataset

(d) The checker-b dataset

Figure 6.5: Generalization accuracies obtained using FaLK-SVM, FaLK-SVMc and FaLK-SVMl on the cov-type, 2-spirals, ndcc and checker-b datasets with increasing training set sizes.

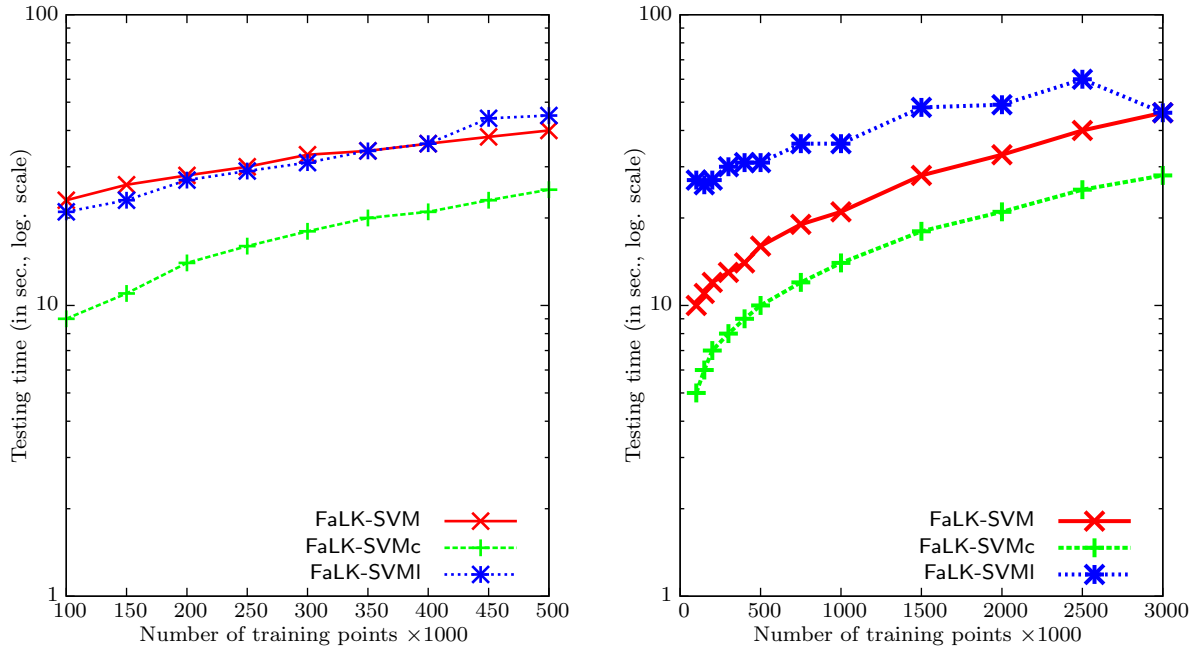


(a) The cov-type dataset (b) The 2-spirals dataset

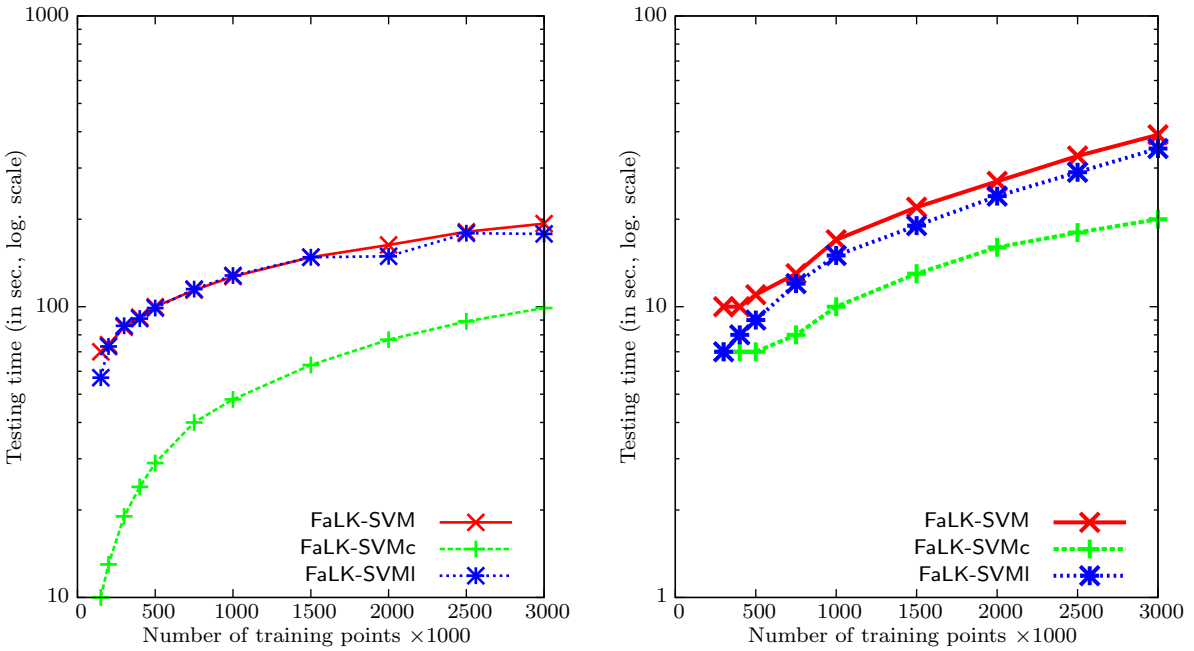


(c) The ndcc dataset (d) The checker-b dataset

Figure 6.6: Training times of FaLK-SVM, FaLK-SVMc, and FaLK-SVMl on the cov-type, 2-spirals, ndcc and checker-b datasets with increasing training set sizes. The times (in seconds) are reported in logarithmic scale.



(a) The cov-type dataset (b) The 2-spirals dataset



(c) The ndcc dataset (d) The checker-b dataset

Figure 6.7: Testing times of FaLK-SVM, FaLK-SVMc, and FaLK-SVMl on the cov-type, 2-spirals, ndcc and checker-b datasets with increasing training set sizes. The times (in seconds) are reported in logarithmic scale.

sets.

The training computational performances of Figure 6.6 confirm (as already discussed in Section 6.2.2) that, although FaLK-SVM and FaLK-SVMc make use of the same training algorithm, the model selection procedure selects lower values of k for FaLK-SVMc, thus assuring faster training times than FaLK-SVM. The speed-ups of FaLK-SVMc with respect to FaLK-SVM are however never higher than one order of magnitude. For FaLK-SVMl we can notice a somehow irregular behaviour for increasing dimensions of the training set and this is due to the different values of the neighbourhood, kernel and regularisation parameters it chooses during the internal fast local model selection phase. Although in some cases FaLK-SVMl is significantly slower than FaLK-SVM, if we consider that the training times for FaLK-SVMl includes the model selection procedure whereas for FaLK-SVM we consider only the training with the optimal parameters, we can conclude that FaLK-SVMl is a good choice for huge training sets on which traditional model selection becomes intractable.

The testing times reported in Figure 6.6 confirm that FaLK-SVMc is always faster than FaLK-SVM and FaLK-SVMl. In particular, we can notice that FaLK-SVMc at least halves the testing time of FaLK-SVM. FaLK-SVMl is computationally very similar to FaLK-SVM; this is not surprising because the only difference between FaLK-SVM and FaLK-SVMl regards the model selection but both classifiers need, during testing, to perform a nearest neighbour search of the query points among all training examples, differently from FaLK-SVMc that performs the nearest neighbour search only among the centers of the local models.

We can conclude that FaLK-SVM, FaLK-SVMc and FaLK-SVMl achieve similar accuracy and computational results. When the model selection for FaLK-SVM and FaLK-SVMc become computationally intractable, FaLK-SVMl is an option to efficiently perform model selection and thus obtain a lower overall training time. When very low testing times are required, FaLK-SVMc is preferable to FaLK-SVM at the price of a slightly lower generalization accuracy.

6.3 Conclusions

In this work, we have introduced a new local kernel-based classifier, called FaLK-SVM, that is scalable for large non high-dimensional data. The approach is developed starting from the theory of local learning algorithms and in particular from the Local SVM classifier, called kNNSVM. Various strategies are introduced to overcome the computational problems of kNNSVM and to switch from a completely lazy-learning setting to a eager learning setting in which the predictions can be performed efficiently. Learning and complexity bounds for FaLK-SVM are detailed and they are favorable if compared with the SVM ones. FaLK-SVM has, in fact, a training time complexity which is sub-quadratic in the training set size, and a prediction time complexity which is logarithmic. A novel approach for model selection, again based on locality, is introduce obtaining the FaLK-SVMl classifier which substantially unburden the model selection strategies based on cross-validation. Another variant of the algorithm for the prediction phase, permits to FaLK-SVMc to further speed-up FaLK-SVM during prediction. We thus showed that locality can be used to develop computationally efficient classifiers.

We carried out an extensive empirical evaluation of the introduced approaches showing that,

for large classification problems requiring non linear decision functions our FaLK-SVM algorithm is much faster and accurate than traditional and approximated SVM solvers. In facts, from the generalization performances viewpoint, FaLK-SVM achieves very good accuracy results because it considers all the points without locally under-fitting the data and, from the computational performances viewpoint, FaLK-SVM is very fast and scalable because the cardinality of the local problems can be maintained low. A variant further enhancing testing speed at the price of a little accuracy loss, called FaLK-SVMc, is presented as well as a variant integrating a very fast local model selection procedure, called FaLK-SVMl.

In general, we have showed that locality can be the key not only for obtaining accurate classifiers, but also for effectively speeding-up kernel-based algorithms differently from the assumption of most state-of-the-art fast SVM solvers.

Local kernel machines can of course be applied for task different from classification. Regression is a related area that can very likely take advantage from the framework introduced in this chapter. Another possibility, discussed and developed in the next chapter, concerns the application of local kernel machines as a preprocessing step in order to remove or at least reduce, the noise present in the data.

Chapter 7

Noise Reduction with Local Kernel Machines

The problem of noise in machine learning has been addressed more by developing algorithms that are noise tolerant than by explicitly removing noise. Examples are soft-margin SVM [51], early stopping for artificial neural networks [35] and the post-pruning of decision trees [146]. Nevertheless there are a number of circumstances where explicitly removing noise can have merit. It is difficult to make IBL algorithms such as kNN classifiers or CBR noise tolerant so noise reduction can be important for improving generalisation accuracy in IBL. A further motivation for noise reduction in CBR is explanation – a capability that is perceived to be one of the advantages of CBR [106, 55]. Since case-based explanation will invoke individual cases as part of the explanation process it is important that noisy cases can be eliminated if possible. Even if noise reduction will not improve the classification accuracy of learning algorithms that have been developed to be noise tolerant, researchers have argued that noise reduction as a preprocessing step can simplify resulting models, an objective that is desirable in many circumstances [118], and that it has been investigated also as a way of computationally unburden maximal margin classifiers [10, 175]

Generally speaking, the random (i.e. not systematic) noise affecting machine learning datasets is mainly of two types: attribute (or feature) noise and class (or mislabeling) noise. The first is almost inevitably present in the data because of errors and approximations on observing and measuring the attributes of the examples. The latter is due to errors in the process of assigning labels to the examples. Moreover other sources of generalization accuracy problems that cannot be strictly considered noise are outlier examples (i.e. correct examples representing some atypical examples) and contradictory examples (i.e. examples with the same attribute values but different labels). A noise reduction algorithm must deal consistently with all these issues in order to be successfully applied for real problems.

In kNN and CBR the problem of noise reduction has traditionally been considered part of the larger problem of case-base maintenance. Since large training sets can influence the response time of lazy learners an extensive literature is dedicated to the development of data reduction techniques that preserve training set competence (see Section 2.3.1). While the problem of noise in kNN can be mitigated by increasing the neighbourhood size and using a majority decision rule there has also been a lot of research on competence enhancing techniques that preprocess the training data to remove noisy examples. Such competence enhancing techniques are the subject of this work.

In this chapter we present a novel technique for competence enhancing in the context of kNN-based classifiers and a variant of the approach in order to efficiently tackle large and very large datasets. The first technique, called kNNSVM-nr, is based on Local Support Vector Machines [19] introduced in Section 3.4. By extending kNNSVM with a probabilistic output we apply it on the training set to remove noisy, corrupted and mislabeled examples. This is done by building a local model in the neighbourhood of each training example and the example is removed if the probability associated with the correct classification is below a threshold. In other words we remove those examples that, with respect to the maximal separating hyperplanes built on the feature-space projections of their neighbourhoods, are too close to or on the wrong side of the decision boundary. From another viewpoint we simply augment the majority rule criterion used by most competence enhanced techniques (see section 2.3.2) with the kernel-space maximal margin principle. The second technique we present here is an extension of kNNSVM-nr, called FaLKNR, that introduces some of the approaches we adopted for the fast local kernel machines described in the previous chapter.

It is well-known that the classification error of the NN classifier is bounded by twice the Bayes error as the number of training examples N goes to infinity. This bound can be lowered to the Bayes error using the k-NN classifier with an high k (it is required that $k \to \infty$, $N \to \infty$, $k/N \to 0$), or using the 1NN classifier on an edited training set such that it guarantees the perfect training set classification [62] (see also Chapter 3.1). This suggests that, for practical problems, if we are interested mainly in generalization accuracies, two aspects are crucial: the ability to detect and remove noisy examples in order to theoretically approach the Bayes error with the 1NN classifier, and the possibility to use as much data as possible in order to approximate the $N \to \infty$ condition. We tackle the first aspect with kNNSVM-nr which, accordingly to the empirical evaluation reported in this chapter, outperforms existing noise reduction techniques mainly due by bringing local class boundaries into consideration. The second aspect, namely the developing of a noise reduction technique that can be applied to large and very large datasets, is tackled with FaLKNR that it is applicable to datasets with cardinalities in the order of millions of examples.

In the evaluation we present in this chapter we compare the performance of our Local SVM-based strategies against three state-of-the-art noise reduction techniques from the literature (see section 2.3.4). FkNNSVM comes out on top against these techniques on a range of 15 real world datasets and on six spam filtering datasets. It also performs very well on artificial datasets where we consider feature noise, label noise and unbalanced class distributions. FaLKNR even enhances the noise reduction capabilities of FkNNSVM on 9 large datasets (up to 500k examples) with

better computational performances than traditional approaches.

The developed techniques have been presented in [169] (kNNSVM-nr) and [166] (FaLKNR) and are available (kNNSVM-nr as the updated and faster FkNNSVM-nr) as part of the Fast Local Kernel Machine Library (FaLKM-lib) [163] freely available at `http://disi.unitn.it/~segata/FaLKM-lib` and described in Appendix A. The original implementation of kNNSVM-nr used in this chapter is available at `http://disi.unitn.it/~segata/LSVM-nr/LSVM-nr.html`.

The chapter is organized as follows. In the next section we elaborate on the motivations for noise reduction before introducing FkNNSVM-nr in Section 7.2 that is empirical evaluated in section 7.3 on a number of real and artificial datasets. Section 7.4 details FaLKNR, the fast and scalable version of FkNNSVM-nr, empirically validated in Section 7.5. The chapter with conclusions and some reflections on promising directions for future work.

7.1 Motivation

The local nature of CBR and IBL entails a vulnerability to noise in training data. Thus they have a dependency on individual training examples that other supervised learning techniques do not have. Other techniques have been developed to be noise tolerant by incorporating into the induction process mechanisms that attempt to avoid over-fitting to noise in the training set. Examples of this include early stopping for artificial neural networks [35], the post-pruning of decision trees [146] and using soft-margin Support Vector Machines which relax the constraints on the margin maximisation [51]. However, instance based techniques such as kNN that rely on specific retrieved instances for induction are affected by noise. These techniques generally lack the induction step that other noise tolerant techniques can adapt. The dependance on the specific retrieved instances can be reduced by retrieving more instances (i.e. kNN, with $k > 1$ is more noise tolerant than NN) but accuracy will not always increase with larger values of k. At some point a large k will result in a neighbourhood that crosses the decision surface and accuracy will drop.

An additional motivation for noise reduction in IBL associated with this dependency on individual training examples is case-based explanation. A learning system that can provide good explanations for its predictions can increase user confidence and trust and give the user a sense of control over the system [151]. Case-based explanations are generally based on a strategy of presenting similar past examples to support and justify the predictions made [55, 133]. If specific cases are to be invoked as explanations then noisy cases need to be identified and removed from the case-base.

Despite the importance of noise reduction for IBL and CBR, little work has been done on making competence enhancement techniques applicable to large data collections in order to better approach the theoretical Bayes error rate on unseen examples with the simple NN classifier.

Finally there are specific application areas where noise reduction is important. It is generally accepted that inductive learning systems in the medical domain are dependent on the quality of the data [141] and there has been significant research into data cleansing in bioinformatics [120, 74, 118, 184, 185]. Although instance based techniques such as k-NN are not generally used for

classification in much of this research, noise reduction is an important element in the process as it can result in the simplification of the models created. Lorena and Carvalho [118], for example, found that preprocessing the training data to remove noise resulted in simplifications in induced SVM classifiers and higher comprehensiveness in induced decision tree classifiers. Similar results are reported for SVM [10, 175]. Both in data cleansing in bioinformatics and as a preprocessing step for different classifiers, the scalability issue is often crucial.

7.2 Local Support Vector Machines for Noise Reduction

We recall the probability output for kNNSVM that can be obtained using the local SVM probability estimation as follows:

$$\widehat{p}^{\,k\text{NNSVM}}(y=+1|\mathbf{x};\mathcal{X}) = \frac{1}{1+\exp(A\cdot k\text{NNSVM}(\mathbf{x};\mathcal{X})+B)} \tag{7.1}$$

Local learning algorithms can be applied in the training set with a leave-one-out strategy to detect the examples that would not be correctly predicted by their neighbourhood. The noise reduction techniques for CBR and IBL proposed in the literature so far use strategies in the spirit of case-based local learning. Here, using the kNNSVM approach, we can apply the maximal margin principle to the neighbourhood of each training example to verify if the actual label of the central example is correctly predicted. What is theoretically appealing about Local SVM for noise removal, is its compromise between the discrimination ability of SVM with respect to the majority voting and the *local* application of the maximal margin principle which is crucial since the final classification is performed with an inherently local nearest neighbour strategy.

The set $\mathcal{X}' \subseteq \mathcal{X}$ of training examples without the noisy examples detected by kNNSVM is thus defined as follows:

$$\mathcal{X}' = \left\{\mathbf{x}_i \in \mathcal{X} \;\middle|\; k\text{NNSVM}(\mathbf{x}_i;\mathcal{X}\setminus\mathbf{x}_i) = y_i\right\}.$$

Notice that we remove the example $\mathbf{x}$ (the point we want to assess if it is noise or not) from the training set on which its neighbourhood is retrieved, because we want to avoid that the local decision rule is overfitted by $\mathbf{x}$ itself.

Although kNNSVM is a local learning algorithm, its decision rule (the maximal margin separation) can be very different from the kNN decision rule (majority rule) which will be used in the final classifier. For this reason and, more generally, in order to be able to adapt to different types and levels of noise, it is desirable to have the possibility to tune the aggressiveness of the removing policy. This can be achieved using the probabilistic output of kNNSVM, obtaining the kNNSVM-nr method, as follows:

$$\mathcal{X}' = \left\{\mathbf{x}_i \in \mathcal{X} \;\middle|\; \widehat{p}^{\,k\text{NNSVM}}(y=y_i|\mathbf{x}_i;\mathcal{X}\setminus\mathbf{x}_i) > \gamma\right\}.$$

The γ threshold can be manually tuned to modify the amount of noise to be removed and the probability level associated with non-noisy examples. Intuitively, we expect that for very low

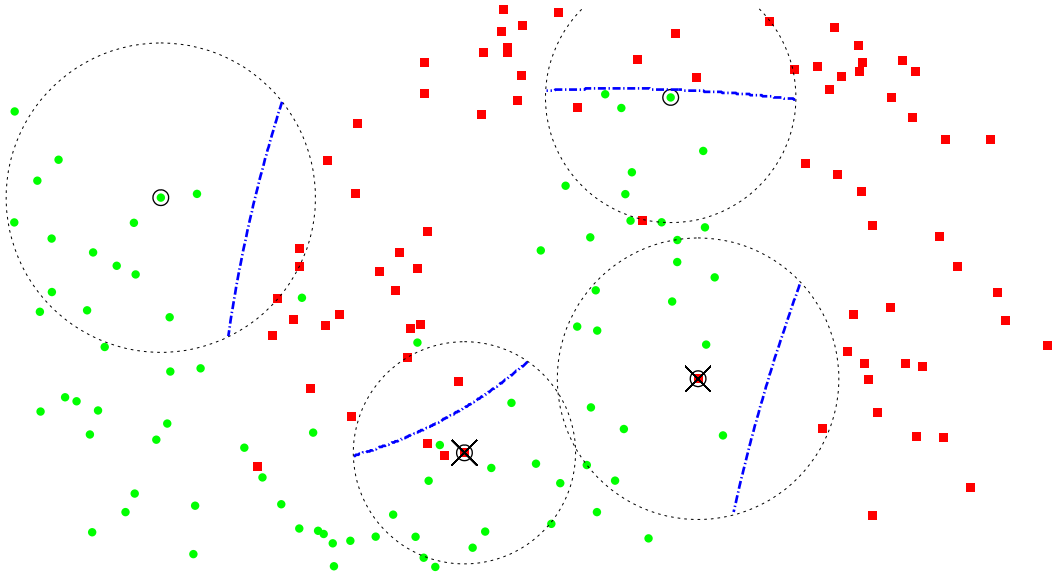


Figure 7.1: The application of kNNSVM-nr with $\gamma = 0.5$, neighbourhood size $k = 15$, regularisation parameter $C = 10$, with the RBF kernel with $\sigma = 0.1$ on a toy dataset. We choose to apply the technique only to a subset of four points for graphical clearness.

values of $\widehat{p}^{\,kNNSVM}(y = y_i|\mathbf{x}_i; X \setminus \mathbf{x}_i)$, $\mathbf{x}_i$ corresponds to a mislabeled example, while for values near 0.5, $\mathbf{x}_i$ could be a noisy example or an example close to the decision surface. High values of γ can be used to maintain in the training set only examples for which kNNSVM is highly confident in their labels, theoretically enhancing the separation between the classes. The locality of the approach is regulated by the k parameter and can be enhanced by using a local kernel such as the RBF kernel [77] or by applying a quasi-local kernel operator to a generic kernel as described in [167]. Figure 7.1 graphically shows how kNNSVM-nr intuitively works.

Although not empirically tested and discussed in this work, the same framework can be used to perform competence preservation (or redundancy reduction) by simply changing the comparison operator:

$$\mathcal{X}' = \left\{ \mathbf{x}_i \in \mathcal{X} \mid \widehat{p}^{\,\mathrm{kNNSVM}}(y = y_i|\mathbf{x}_i; \mathcal{X} \setminus \mathbf{x}_i) < \gamma \right\}.$$

The idea, in this case, is to remove the examples that are very likely to be correctly classified maintaining in the training set only the examples that are close to decision boundary. A further quite straightforward modification would allow the integration of competence preservation and competence enhancement:

$$\mathcal{X}' = \left\{ \mathbf{x}_i \in \mathcal{X} \mid \gamma' < \widehat{p}^{\,\mathrm{kNNSVM}}(y = y_i|\mathbf{x}_i; \mathcal{X} \setminus \mathbf{x}_i) < \gamma'' \right\}.$$

7.2.1 Computational Aspects of kNNSVM-nr

Brute-force approaches for kNN need to compute the distances between the query example and all the training examples, to sort the examples by distance and to select the k examples with the smallest distances. Using a sorting algorithm like *quicksort* we obtain a computational complexity of $\mathcal{O}(n + n \cdot \log n + k) = \mathcal{O}(n \cdot \log n)$ in average. ENN requires a k-nearest neighbour retrieval and a majority rule evaluation for each training example thus scaling as $\mathcal{O}(n^2 \cdot \log n + n \cdot k) = \mathcal{O}(n^2 \cdot \log n)$. If we assume that RENN performs a limited number of recursive applications of ENN (as occurs in practice), it has the same complexity bound as ENN, whereas the complexity of AkNN is k times the complexity of ENN and thus equal to $\mathcal{O}(k \cdot n^2 \cdot \log n)$.

Although modern accurate SVM solvers like LibSVM [36] have in practice a computational time that grows almost quadratically with the number of training examples, we consider here the theoretical computational complexity of SVM training which is in the order of $\mathcal{O}(N^3)$ and of SVM prediction which is in the order of $\mathcal{O}(N)$ as discussed for example by [26] (se also Chapter 3.2). Recalling that kNNSVM-nr trains an SVM on the neighbourhood examples of each training example, its computational bound is $\mathcal{O}(N^2 \cdot \log N + N \cdot k^3 + N \cdot k) = \mathcal{O}(N^2 \cdot \log N + N \cdot k^3)$. We can see then that, although kNNSVM-nr is slower than ENN (as the SVM training and prediction is more complex than the evaluation of the majority rule), the computational time of both methods is dominated by the retrieval of the neighbourhood examples and, if the k parameter is not very high, kNNSVM-nr is competitive with RENN and AkNN. Moreover, the local SVMs trained by kNNSVM-nr generally have a rather small size (k) and thus the kernel matrix generated by the SVM solver can fit in main memory and thus the k^3 scaling factor for SVM is a very loose bound. In addition, for local SVM models that include examples of one class only, training is avoided because the decision rule is equivalent to the majority rule.

Various approaches can be considered to reduce the computational times of the discussed noise reduction techniques. The first is the adoption specific data-structures to support nearest neighbour operations; we recently implemented an extension of kNNSVM-nr, called FkNNSVM-nr that uses the Cover Trees (see 3.5) similarly to FkNNSVM and included in FaLKM-lib [163] (see Appendix A). The computational complexity of FkNNSVM-nr is $\mathcal{O}(N \cdot k \log N + N \cdot k^3)$, which is lower than kNNSVM-nr, but still not scalable enough for the application to huge datasets. For this reason we will introduce in Section 7.4 some approaches similar to those introduced in the previous chapter for FaLK-SVM to further lower the computational complexity.

7.3 Evaluation of kNNSVM-nr

As stated in Chapter 2.3 the state-of-the-art noise reduction strategies are RENN, AkNN and, at least for spam filtering, BBNR; we thus choose to benchmark kNNSVM-nr against these three approaches. Although multiple evaluation strategies for editing techniques for IBL and CBR can be considered [204], we focus here on analysing the change in generalisation accuracy which is arguably the more important aspect in a noise reduction context. However, for completeness, we also present figures for the reduction in the training set for each technique. We have decided to focus the empirical evaluation on the binary class case, although the proposed approach can

be generalized to the multi-class case. This is done primarily as no studies have been performed yet to present the most appropriate strategy for multi-class local SVM classifiers. In addition we wanted to avoid the evaluation being affected by the differences in the multi-class classification strategies adopted by SVM and kNN-based noise reduction techniques.

The model selection is performed as follows. For RENN, AkNN and BBNR the k parameter is chosen as the one giving the best kNN 20-fold cross validation accuracy among the following set of possibilities: $\{1, 3, 5, 7, 10, 20, 40, 80, 160, 320, 640, 1280\}$ (for datasets with less than 3840 examples, the values higher than $|\mathcal{X}|/3$ are not considered). Preliminary results indicated that this choice permits much more accuracy gain with the edited training set compared with the alternative of fixing k to 1 or 3 as usually done in literature. For kNNSVM-nr we use the RBF kernel and, as with the other techniques, we select the value of k (in the same set of values used for RENN, AkNN and BBNR) and the other parameters (the regularisation parameter $C \in \{2^0, 2^1, \ldots, 2^9, 2^{10}\}$ and the kernel width $\sigma \in \{2^{-10}, 2^{-9}, \ldots, 2^4, 2^5\}$) giving the best 20-fold cross classification accuracy of the associated kNNSVM classifier. To select the noise threshold γ for kNNSVM-nr we perform 20-fold cross validation editing on the training set. The parameters found for kNNSVM-nr are used also to perform classification directly with kNNSVM using the RBF kernel. The generalisation accuracy reported are results on the test set using 1NN, 3NN and kNNSVM classifiers. If a separate test set is not available we randomly remove 1/4 of the training set examples and use them for testing.

kNNSVM is implemented using the LibSVM library [36] for training and evaluating the local SVM models and are available at `http://www.disi.unitn.it/~segata/LSVM-nr/LSVM-nr.html`. An updated and revised version of kNNSVM noise reduction (called FkNNSVM) can be obtained as part of the Fast Local Kernel Machine Library [163, FaLKM-lib] freely available for research and education purposes at `http://disi.unitn.it/~segata/FaLKM-lib`. We implemented also RENN and AkNN, while for BBNR we used the *jColibrì* 2.0 framework [12, 63].

7.3.1 Evaluation on 15 Real Datasets

We consider 15 binary-class datasets with no more than 5000 examples and no more than 300 features from the UCI repository [7] and the LibSVM website [36]. The datasets have only numerical feature values, generally balanced class cardinalities and are scaled to the $[0, 1]$ interval. The characteristics of the 15 datasets are reported in Table 7.1.

Table 7.2 reports the classification accuracies of the 1NN and 3NN algorithms applied to the unedited training sets and to the training sets edited with RENN, AkNN, BBNR and kNNSVM-nr. The reported mean rank of each technique, computed by averaging the ranks for each dataset, can give an indication of which technique performs best. The assessment of the significance of the differences in generalization accuracies is performed using the Friedman test [71, 72]; if the Friedman test ($\alpha = 0.05$) succeeds in rejecting the null hypothesis of no differences between the techniques, the Bonferroni-Dunn post test [66] is used to identify those techniques that perform statistically better than the control technique (i.e. 1NN and 3NN on the unedited training set). In addition we also use the Wilcoxon Signed Rank Test [202] ($\alpha = 0.05$) to test if each noise reduction technique causes the 1NN and 3NN classifiers to achieve statistically significant

dataset name	brief description	src	tr. set card.	te. set card.	class balancing	# of feat.
a3a	Adult dataset preprocessed as [143]	[36]	3185	29376	24%/76%	123
astro	astroparticle application (Uppsala University)	[7]	3089	4000	65%/35%	4
australian	australian credit approval, from Statlog	[36]	517	173	44%/56%	14
breast	Wisconsin breast cancer data	[7]	512	171	64%/37%	10
cmc	contraceptive method choice data	[7]	1104	369	43%/57%	8
diabetes	Pima indians diabetes data	[36]	576	192	66%/34%	8
let_MN	Statlog letter recognition data (M and N)	[36]	1212	363	50%/50%	16
mam	mammographic cancer screening mass data	[7]	720	241	46%/54%	5
musk1	musks/non-musks molecule prediction, v. 2	[7]	4948	1650	85%/16%	166
numer	German numeric credit risk, orig. from Statlog	[36]	750	250	70%/30%	24
digit_06	handwritten digits recognition (0 and 6)	[7]	1500	699	54%/48%	16
digit_12	handwritten digits recognition (1 and 2)	[7]	1559	728	50%/50%	16
spambase	spam filtering data	[7]	3450	1151	40%/60%	57
splice	primate splice-junction gene sequences data	[36]	1000	2175	53%/48%	60
w1a	web page classification, from [143]	[36]	2477	47272	97%/3%	300

Table 7.1: The 15 datasets used in the experiments of Section 7.3.1. For each dataset a brief description, the source (LibSVM repository [36] or UCI repository [7]), the training set cardinality, the testing set cardinality, the class balancing and the number of features are reported.

improvements over no editing, in line with [61]. In Table 7.2 we also report the generalization accuracy of FkNNSVM which is the algorithm for performing classification directly with the Local SVM approach.

Table 7.3 reports the training set reductions achieved by each of the noise reduction techniques. The table also includes, for each technique, the proportion of examples removed by the technique and also removed by all other techniques and the proportion of examples that are only removed by the technique and not removed by any other technique.

Focusing on the induced kNN generalization accuracies which is the purpose of the present study, it is clear from Table 7.2 that kNNSVM-nr is the most effective editing technique. This can be seen by considering the number of times kNNSVM-nr allows the kNN classifier to achieve the best results (9 times for 1NN and 10 times for 3NN) and the corresponding average ranks that are much lower than the other techniques. The claim is supported by statistical evidence as the Friedman test and the Bonferroni-Dunn post test reported in Table 7.2 confirms a statistically significance difference for the 1NN case. For the 3NN case although the Friedman test rejects the null hypothesis, the Bonferroni-Dunn post test does not confirm the differences. Using the Wilcoxon signed-ranks test, however, a significant difference between kNNSVM-nr and no editing is found both for 1NN and 3NN. It is reasonable, however, that the difference between using kNN on unedited and edited training sets is higher for $k = 1$ because higher values of k permits some level of noise-tolerance. Notice that RENN, AkNN and BBNR are not significantly better than the unedited training set neither for 1NN and 3NN. Comparing directly the techniques in a pairwise setting using the Wilcoxon Signed Rank Test, we see that kNNSVM-nr is statistically

dataset	1NN test set accuracy					3NN test set accuracy					FkNNSVM test
	uned.	RENN	AkNN	BBNR	kNNSVM-nr	uned.	RENN	AkNN	BBNR	kNNSVM-nr	set accuracy
a3a	78.23	81.94	**82.66**	78.00	82.62	81.04	81.93	**82.67**	81.07	82.62	81.23
astro	93.93	94.75	**95.03**	92.28	94.98	94.93	94.93	95.30	94.40	**95.35**	95.83
australian	82.66	**84.97**	84.39	64.74	**84.97**	82.08	**85.55**	84.39	72.25	84.39	84.97
breast	94.74	97.08	**97.66**	95.32	**97.66**	**98.25**	97.66	97.66	97.66	**98.25**	97.08
cmc	53.39	60.70	57.99	53.39	**63.14**	56.91	**60.98**	58.81	56.64	60.43	63.14
diabetes	66.67	66.15	67.19	58.33	**70.31**	63.54	66.67	65.63	58.33	**68.23**	69.79
let_MN	99.72	99.72	99.72	99.72	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	100.00
mam	75.10	81.33	**82.16**	64.73	80.50	79.25	80.91	81.33	67.22	**81.74**	82.16
musk2	96.24	96.24	95.76	**96.55**	96.42	96.48	96.12	96.00	**96.91**	96.36	99.64
numer	68.80	71.60	70.00	65.60	**72.40**	72.00	71.20	71.20	69.60	**73.60**	72.40
digit_06	**98.71**	**98.71**	**98.71**	**98.71**	**98.71**	**98.71**	**98.71**	**98.71**	**98.71**	**98.71**	98.28
digit_12	**97.80**	97.66	97.66	**97.80**	97.66	**98.08**	97.94	97.94	**98.08**	97.94	98.49
spambase	90.18	88.97	89.40	90.18	**91.23**	90.01	88.71	88.71	89.92	**90.62**	93.48
splice	70.62	48.00	60.97	71.54	**73.84**	72.18	48.00	57.56	76.05	**77.29**	89.28
w1a	95.09	97.13	**97.48**	95.36	97.34	97.34	97.13	97.31	97.11	**97.38**	96.83
average rank	3.6	3.2	2.7	3.7	1.8	3.0	3.2	3.2	3.8	1.9	
Friedman test	Null hypotheses rejected (p-value=.002)					Null hypotheses rejected (p-value=.006)					
BD post-test w.r.t. uned.		×	×	×	✓		×	×	×	×	
WSRT w.r.t. uned.		×	×	×	✓		×	×	×	✓	

Table 7.2: 1NN and 3NN generalisation accuracies for the unedited training set and for the edited training sets and the FkNNSVM generalization accuracies. The best 1NN and 3NN classification accuracies for each dataset are highlighted in bold. We also report the average ranks of the generalization accuracies of 1NN and 3NN using the different noise reduction techniques among all the datasets, the Friedman test which reject the null hypothesis (that all the methods perform equally) if the p-values is lower than $\alpha = 0.05$. Where the Friedman test rejects the null hypothesis, the Bonferroni-Dunn (BD) post test is used to test if one method is statistically better than the others using a control classifier (the unedited training set) and the Wilcoxon Signed Rank Test (WSRT) which shows if the methods are statistically better than the unedited training sets with pairwise comparisons.

dataset	training set reduction				proportion of removed examples that are removed by all the other methods				proportion of removed examples that are not removed by other methods			
	RENN	AkNN	BBNR	kNNSVM-nr	RENN	AkNN	BBNR	kNNSVM-nr	RENN	AkNN	BBNR	kNNSVM-nr
a3a	19.9%	34.1%	45.3%	30.7%	0.317	0.185	0.138	0.204	0.009	0.174	0.751	0.124
astro	4.5%	6.1%	8.8%	2.5%	0.022	0.016	0.011	0.041	0.086	0.079	0.779	0.014
australian	14.3%	27.9%	63.4%	72.5%	0.311	0.160	0.070	0.061	0.000	0.007	0.302	0.192
breast	3.9%	5.5%	7.4%	5.3%	0.100	0.071	0.053	0.074	0.050	0.071	0.553	0.148
cmc	38.5%	50.0%	22.1%	26.0%	0.054	0.042	0.057	0.105	0.019	0.042	0.525	0.066
diabetes	31.3%	41.0%	40.6%	28.8%	0.083	0.072	0.064	0.090	0.150	0.091	0.650	0.030
let_MN	0.4%	0.4%	0.2%	1.2%	0.000	0.000	0.000	0.000	0.200	0.200	1.000	0.857
mam	19.7%	36.1%	39.6%	19.3%	0.028	0.015	0.014	0.029	0.007	0.170	0.726	0.007
musk2	4.6%	6.3%	5.4%	2.6%	0.108	0.082	0.113	0.163	0.079	0.147	0.477	0.136
numer	35.9%	44.7%	41.3%	37.2%	0.309	0.248	0.268	0.297	0.056	0.072	0.397	0.075
digit_06	0.2%	0.2%	0.1%	0.1%	0.000	0.000	0.000	0.000	0.333	0.333	1.000	0.000
digit_12	0.5%	0.6%	0.4%	0.4%	0.000	0.000	0.000	0.000	0.000	0.111	0.875	0.333
spambase	11.2%	10.1%	7.0%	4.1%	0.148	0.164	0.233	0.413	0.067	0.003	0.483	0.105
splice	52.7%	45.6%	28.8%	42.2%	0.040	0.048	0.076	0.052	0.256	0.011	0.194	0.296
w1a	2.8%	5.7%	3.7%	0.4%	0.014	0.007	0.011	0.100	0.145	0.486	0.835	0.000

Table 7.3: The training set reductions achieved with RENN, AkNN, BBNR and kNNSVM-nr are reported in the first four columns. In order to explore the overlap in behaviour between the techniques, the following columns report the proportion of the examples removed by a given method and also by all the other methods, and the proportion of examples removed by a method that are not removed by the other methods.

dataset	computational time (sec)			dataset	computational time (sec)		
	RENN	AkNN	kNNSVM-nr		RENN	AkNN	kNNSVM-nr
a3a	23	7	503	astro	20	5	233
australian	1	1	46	breast	2	1	13
cmc	2	1	238	diabetes	1	1	17
let_MN	4	1	4	mam	1	1	66
musk2	257	73	4103	numer	2	1	48
digit_06	4	2	9	digit_12	3	2	72
spambase	29	9	144	splice	3	2	636
w1a	11	4	89				

Table 7.4: The computational performance of the noise reduction preprocessing step techniques presented in Table 7.2 (times for BBNR are not shown because it is implemented with a different framework and the times are not directly comparable).

significantly better than BBNR for both 1NN and 3NN classifiers, better than RENN for the 1NN classifier and better than AkNN for the 3NN classifier (these results are not reported in the tables).

In addition to the very positive accuracy results achieved by kNNSVM-nr, it is interesting to note that RENN, in contrast to the experiments detailed by [204], achieves rather good results with respect to the unedited datasets. This is probably due to the model selection approach we adopted to determine k whereas in [204] k is a-priori set to 3. Consistent with the literature starting from its introduction by [188], AkNN appears slightly better than RENN. BBNR, on the other hand, has the poorest set of results, damaging generalisation accuracy in many cases. We believe that this is due to the fact that BBNR was designed for use in spam filtering so in the next subsection we analyse its performance in this context.

Although the purpose of kNNSVM-nr is to enhance the classification accuracy of kNN classifiers, it is interesting to compare the results of kNN with edited and unedited training sets to the FkNNSVM classifier which is the Local SVM algorithm for classification presented in Section 3.4. Using the Wilcoxon Signed Rank Test, we have that FkNNSVM accuracies are significantly higher than the ones achieved with 1NN and 3NN on the unedited training set and on the training sets edited with RENN, AkNN and BBNR, but no statistical differences are detected with respect to 1NN and 3NN applied on the training sets edited using kNNSVM-nr. If we compare the FkNNSVM results with the best achieved result using 1NN or 3NN on edited or unedited datasets, we see that FkNNSVM performs better in 5 cases, worse in 7 cases and ties in 4 cases. In general, FkNNSVM seems to achieve slightly higher accuracy results than kNN also using editing (notice for example the results for splice and musk2 datasets) if we compare it to the single techniques, although the differences with kNN using kNNSVM-nr are not supported by statistical significance. Notice however that there are cases in which the editing with kNNSVM-nr achieve better results than using kNNSVM directly for classification and this is important as it has been shown that kNNSVM performs at least as good as SVM (see Chapter 4).

From the analysis of the training set reduction rates reported in Table 7.3 we see that

kNNSVM-nr is generally more conservative than RENN, AkNN and BBNR. This is however not the reason why kNNSVM-nr can induce higher kNN accuracies because for the two cases in which kNNSVM-nr removes more examples than RENN, AkNN and BBNR (the australian and let_MN datasets) the corresponding 1NN accuracies on the edited training sets are the highest. Among the other techniques it emerges that AkNN removes more examples than RENN, while the aggressiveness of BBNR varies substantially with the datasets. From the second set of columns of Table 7.3 we can see that there are few examples that are removed by all the techniques and thus these methods appear to work in different ways. kNNSVM-nr has generally a higher fraction of removed examples that are also removed by all the other techniques suggesting that it focuses only on the more harmful examples as its low reduction rates also suggest. The same behaviour can be observed looking at the fraction of examples removed only by kNNSVM-nr (the last four columns of the Table); there are in fact cases in which the examples removed by kNNSVM-nr overlap considerably with those removed by some of the other techniques, but the induced 1NN accuracies are higher (consider for example the case of digit_06 and diabetes). Very often, instead, examples removed by BBNR are not removed by the other approaches meaning that BBNR effectively focus on different types of examples (the examples that cause misclassifications rather than the examples that are themselves misclassified), but this damages the kNN classification when the rates of reduction are high.

Table 7.4 reports the computational performances of kNNSVM-nr, RENN and AkNN. As expected kNNSVM-nr is computationally slower than RENN and AkNN, because of the training of N local SVMs. For datasets that are not very large such as the ones presented in Table 7.2 the computational time of kNNSVM-nr is still acceptable, but it seems that some strategies for speeding up the kNNSVM approach as discussed in Chapter 6 are necessary to apply the strategy to very large datasets.

The overall conclusion that we can draw about kNNSVM-nr after the evaluation on real datasets, is that it yields kNN accuracies that are higher than kNN accuracies using the unedited training sets and the training sets edited with RENN, AkNN and BBNR, and these differences are statistically significant. The kNN classification accuracies after the kNNSVM-nr step are comparable to that with kNNSVM used directly from classification. However in situations where the instance-based characteristics of kNN are required classification using kNNSVM will not be appropriate. kNNSVM-nr is computationally slower than RENN and AkNN (as the training of an SVM is slower than the computation of the majority rule), but the introduced overhead is still acceptable for non-large datasets without using particular strategies (already available) to speed-up the approach. The reduction rates of training sets edited with kNNSVM-nr are generally smaller than the editing with RENN, AkNN and BBNR, but this cannot be considered a drawback in this context since our focus here is on competence enhancement.

7.3.2 Evaluation for Case-Based Spam Filtering

We further test these noise reduction techniques in the context of spam filtering. Notice that the kNNSVM classifier has been successfully applied for spam classification by [17]. In addition to the spambase dataset already introduced, we use five datasets (spam_1-spam_5) from the work

dataset	NN test set accuracy					training set reduction			
	uned.	RENN	AkNN	BBNR	kNNSVM-nr	RENN	AkNN	BBNR	kNNSVM-nr
spam_1	**94.8**	92.4	92.8	**94.8**	94.0	6.1%	4.8%	0.1%	1.7%
spam_2	**96.4**	92.8	92.8	**96.4**	**96.4**	5.9%	6.7%	6.5%	3.7%
spam_3	**97.2**	**97.2**	**97.2**	96.8	**97.2**	1.6%	3.1%	0.7%	0.9%
spam_4	**97.2**	95.6	95.6	**97.2**	96.4	2.4%	2.5%	1.6%	1.7%
spam_5	**96.4**	94.8	95.2	**96.4**	**96.4**	4.4%	4.0%	0.1%	0.7%
spambase	90.0	88.7	88.7	89.9	**90.6**	11.2%	10.1%	7.0%	5.8%

Table 7.5: Generalization NN accuracies and training set reductions for spam filtering obtained with the unedited training set and the training sets edited with the noise reduction techniques under consideration.

on spam filtering by [59][1].

The results are reported in Table 7.5. Apart for spambase, the editing techniques are not able to improve the generalisation accuracies of the unedited datasets. This is probably due to the fact that very little noise is present in the unedited datasets. However, it is interesting to note that BBNR degrades the accuracy only in one case, while RENN and AkNN do a fair deal of damage. The results are consistent with the experiments performed in [60] in which more noise is present and BBNR succeeds in improving classification performance in that case. We believe that noise reduction in spam filtering is unusual because the classes are not well separated since some spam messages have been made to look very like legitimate email. RENN and AkNN do a lot of damage in this situation as they remove considerably more training data than either BBNR or kNNSVM-nr and thus damage generalisation accuracy. BBNR and kNNSVM-nr delete a lot less and thus have better performance. This characteristic of the kNNSVM-nr strategy proves advantageous again in Section 7.3.5 where we look at noise reduction in the presence of unbalanced class densities.

7.3.3 Data with Gaussian Feature Noise

The objective here is to model a scenario where noise results from errors in observing and measuring the descriptive features of the examples – in the next section we cover a scenario where the errors are in the class labels assigned to the examples. In order to study the behaviour of kNNSVM-nr in the presence of "feature" noise we designed two artificial datasets: the 4×4 checkerboard dataset (cb) and the sinusoid dataset (sin). We modify the examples in the two datasets (both training and the test sets) applying Gaussian noise with zero mean and different variance levels ($\sigma^2 = 0.1, 0.2, 0.3, 0.4, 0.5$ for cb and $\sigma^2 = 0.075, 0.1, 0.125, 0.15, 0.175, 0.2, 0.225, 0.25$ for sin). The cb data is based on an artificial data model from [108] and the sin dataset is based on a model by [138]. A subset of the noise configurations of the training datasets are shown in Figure 7.2.

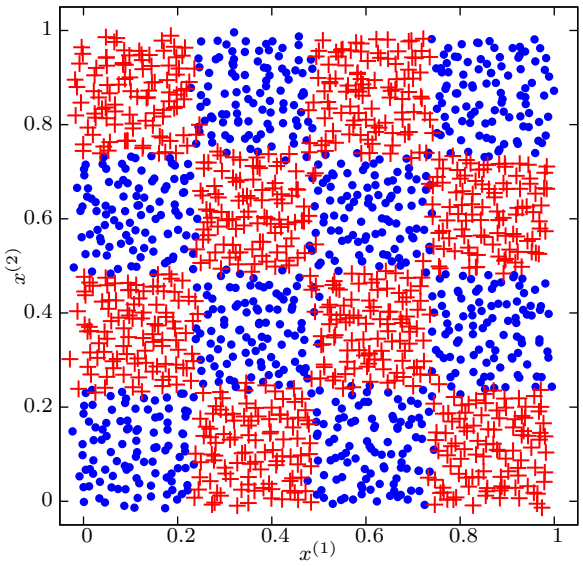
[1]These datasets are available at `http://www.comp.dit.ie/sjdelany/dataset.htm`.

data	σ^2	NN test set accuracy					training set reduction			
		uned.	RENN	AkNN	BBNR	kNNSVM-nr	RENN	AkNN	BBNR	kNNSVM-nr
cb	0.01	94.31	94.13	95.56	93.94	**96.19**	8.2%	17.1%	18.1%	22.8%
cb	0.02	86.94	90.00	89.19	84.69	**90.88**	12.3%	23.0%	31.7%	33.8%
cb	0.03	81.56	86.13	85.50	80.19	**86.81**	16.0%	27.5%	45.6%	44.8%
cb	0.04	76.94	81.94	81.81	72.63	**82.31**	19.3%	35.1%	41.1%	15.8%
cb	0.05	70.75	75.31	75.63	68.81	**75.94**	25.6%	34.0%	34.8%	20.7%
sin	0.075	98.07	98.27	98.33	97.73	**98.80**	2.1%	3.2%	3.6%	8.3%
sin	0.1	92.80	94.07	**94.60**	91.13	94.27	5.2%	10.3%	30.0%	5.5%
sin	0.125	86.60	89.13	90.53	78.73	**90.73**	11.3%	20.8%	46.5%	31.0%
sin	0.15	80.80	85.60	85.87	72.80	**86.13**	12.4%	25.5%	45.9%	36.7%
sin	0.175	74.87	81.53	82.20	66.33	**82.73**	18.3%	33.8%	47.9%	18.0%
sin	0.2	73.20	79.33	79.67	66.80	**80.87**	20.0%	33.5%	37.0%	19.3%
sin	0.225	69.73	73.87	77.07	63.20	**77.73**	33.3%	47.8%	35.8%	54.8%
sin	0.25	66.80	72.93	73.27	61.53	**73.87**	31.8%	50.6%	41.8%	35.4%

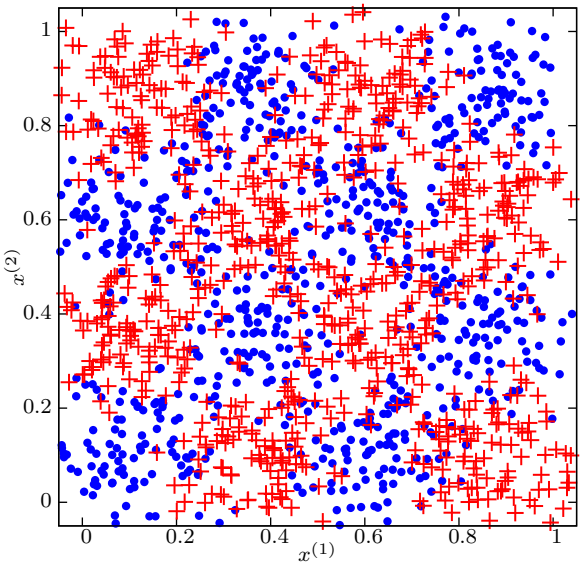
Table 7.6: NN testing accuracies and training set reductions achieved by the noise reduction techniques on cb and sin datasets with examples modified by increasing Gaussian feature noise levels.

Table 7.6 reports the generalisation accuracies and the training set reductions associated with the different noise reduction techniques using a 1NN classifier. Apart from BBNR, all the noise reduction techniques improve on the classification accuracies achievable with the unedited training set (about 5% for significant noise levels), meaning that they are all effective for Gaussian noise reduction. Moreover, our kNNSVM-nr outperforms RENN and AkNN in almost all the considered cases. The superiority of kNNSVM-nr in this context derives from its class discrimination capability introduced by the maximal margin principle which is tolerant to noise. In other words, a noisy example lying in the wrong class region, is more likely to be detected by kNNSVM-nr than by the other techniques based on the neighbourhood majority rule, because kNNSVM-nr is able to estimate the separating hyperplane between classes and thus assess if the example is on the right side or not.

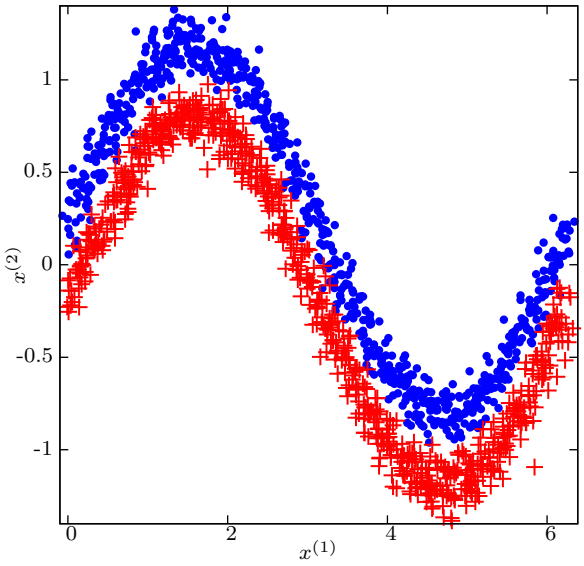
Looking at the training set reduction rates, we can observe that, as expected, RENN and AkNN remove more examples as the variance of the noise increases. For kNNSVM-nr, instead, the reduction rates are less correlated with the Gaussian noise level; this is probably due to the different values chosen by model selection for kNNSVM-nr and in particular to the C regularisation parameter which is the key SVM parameter controlling the estimation of the separating hyperplane with noisy data. Moreover, with little noise, kNNSVM-nr tries to enlarge the class separation thus removing more examples.



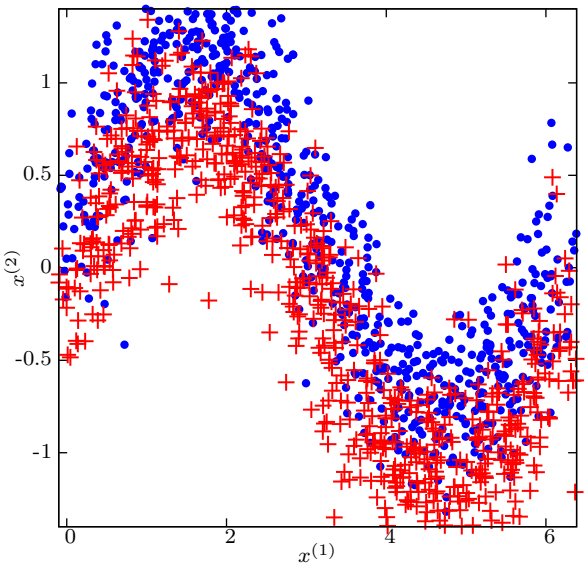
(a) cb dataset, $\sigma^2 = 0.01$



(b) cb dataset, $\sigma^2 = 0.05$



(c) sin dataset, $\sigma^2 = 0.075$



(d) sin dataset, $\sigma^2 = 0.25$

Figure 7.2: The cb and sin datasets with a subset of the different levels of Gaussian noise considered.

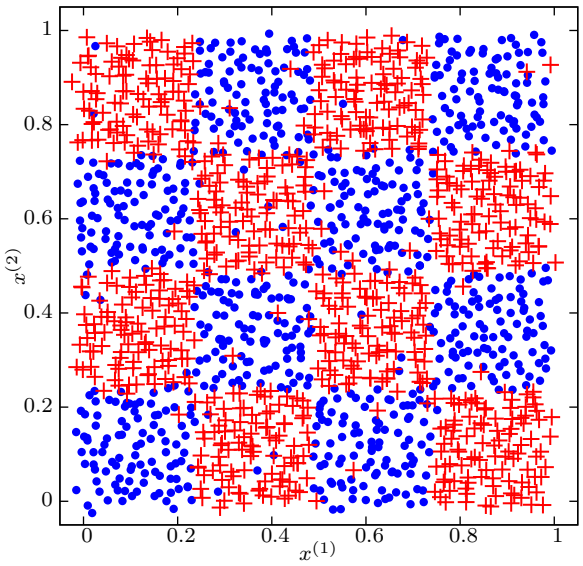
data	mislab.	NN test set accuracy					training set reduction			
	prob.	uned.	RENN	AkNN	BBNR	kNNSVM-nr	RENN	AkNN	BBNR	kNNSVM-nr
cb	0.025	89.81	92.25	92.44	87.00	**92.94**	9.3%	21.4%	27.8%	29.7%
cb	0.05	86.00	89.75	90.00	74.38	**91.38**	11.5%	24.3%	44.4%	35.6%
cb	0.1	78.06	84.75	**84.88**	68.50	84.56	17.7%	35.4%	49.0%	40.0%
cb	0.15	71.81	80.00	**80.25**	64.31	80.00	24.1%	42.0%	45.1%	51.4%
cb	0.2	66.56	**78.25**	76.81	65.06	77.75	30.8%	46.6%	47.9%	43.3%
cb	0.25	61.81	70.00	**70.88**	59.56	70.75	33.1%	59.6%	24.0%	24.5%
sin	0.075	86.60	92.87	**93.00**	71.46	92.93	7.5%	14.7%	58.5%	17.1%
sin	0.1	79.93	**86.93**	86.80	64.26	86.67	11.5%	21.6%	54.3%	11.4%
sin	0.125	74.40	83.13	83.93	59.66	**84.00**	17.7%	32.5%	51.9%	29.9%
sin	0.15	68.13	**78.00**	77.00	56.07	77.73	18.8%	38.7%	58.4%	24.7%
sin	0.175	62.13	75.00	74.80	54.87	**75.60**	25.7%	51.3%	42.8%	35.9%
sin	0.2	56.53	**70.27**	69.47	55.80	69.80	31.9%	62.1%	32.3%	46.3%
sin	0.225	54.47	63.13	63.07	54.73	**63.33**	37.5%	70.3%	30.3%	37.0%
sin	0.25	54.80	58.53	60.73	56.20	**61.33**	47.0%	80.1%	22.1%	58.0%

Table 7.7: NN testing accuracies and training set reduction achieved by the noise reduction techniques on the cb and sin datasets with examples modified by increasing levels of example mislabeling probability.

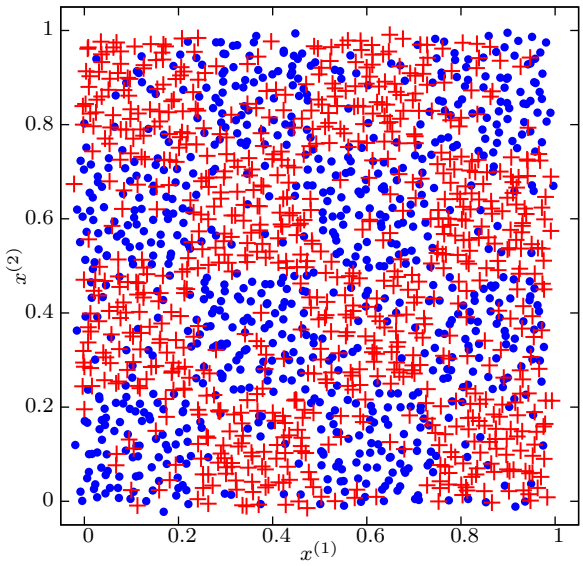
7.3.4 Data with Mislabeled Examples

In this subsection we consider noise that manifests itself as random errors in example labeling (class noise). While the Gaussian feature noise considered in the last section affects the class boundaries, this kind of noise can show up through out the data distribution as can be seen in Figure 7.3. We use the same artificial datasets as previously but with a minimum amount of Gaussian noise and an increasing probability of example mislabeling. Some of the versions of the datasets used in this experiment are shown in Figure 7.3.

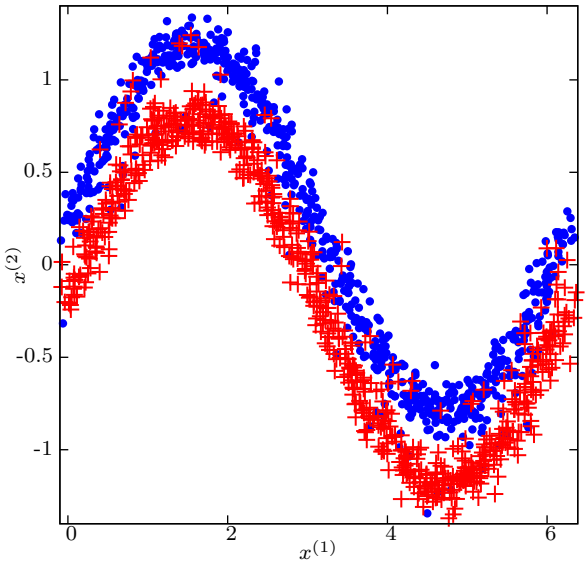
It is clear from the results shown in Table 7.7 that RENN, AkNN and the kNNSVM-nr strategy all produce significant improvements in accuracy, improvements of more than 10% in some cases. For this reason we can conclude that the label noise is more likely to be corrected than feature noise. The differences in improvements due to RENN, AkNN and kNNSVM-nr are minimal and it is not possible to establish which is best. It is not surprising that kNNSVM-nr strategy does not dominate here as its awareness of the decision surface is useful only in the vicinity of class boundaries and many of the noisy examples in this situation are far from the boundaries. In this context the majority rule is effective and kNNSVM-nr does well as it uses this principle since a local SVM model with very unbalanced data classifies all the neighbourhood with the dominant class. The fact than some mislabeled examples are located near to the class boundaries can explain the fact that kNNSVM-nr achieves the best results more frequently than the other approaches (6 times against 4 times of RENN and AkNN) – however this difference is not statistically significant.



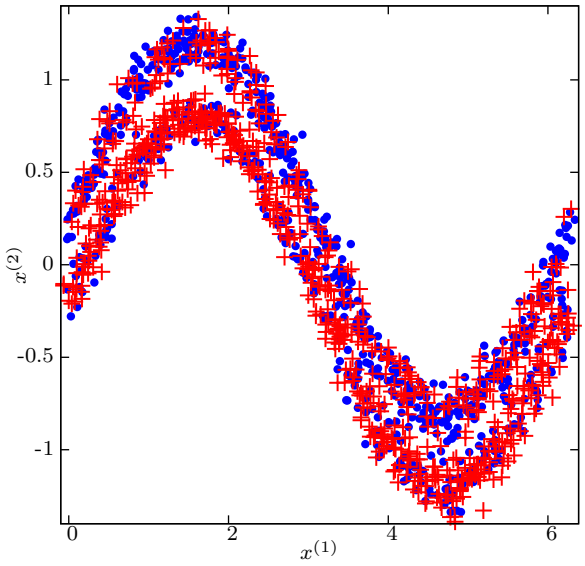
(a) cb dataset, $\sigma^2 = 0.01$, mis. prob.= 0.025



(b) cb dataset, $\sigma^2 = 0.01$, mis. prob.= 0.2



(c) sin dataset, $\sigma^2 = 0.075$, mis. prob.= 0.05



(d) sin dataset, $\sigma^2 = 0.075$, mis. prob.= 0.35

Figure 7.3: The cb and sin datasets with a subset of different example mislabeling probabilities considered.

dataset	NN test set accuracy				training set reduction		
	uned.	RENN	AkNN	kNNSVM-nr	RENN	AkNN	kNNSVM-nr
musk2	96.24	96.24	95.76	**96.42**	4.6%	4.6%	2.6%
musk2 unbal.	**96.12**	94.24	94.91	95.33	1.9%	1.9%	1.4%
astro	93.93	94.75	**95.03**	94.98	4.5%	4.5%	2.5%
astro unbal.	88.23	86.98	87.75	**89.20**	2.9%	2.9%	2.8%

Table 7.8: Generalization accuracies of the NN classifier using the unedited training sets and the noise reduction techniques on the musk2 and astro datasets in the original version and in the unbalanced class densities version.

7.3.5 Data with Unbalanced Class Densities

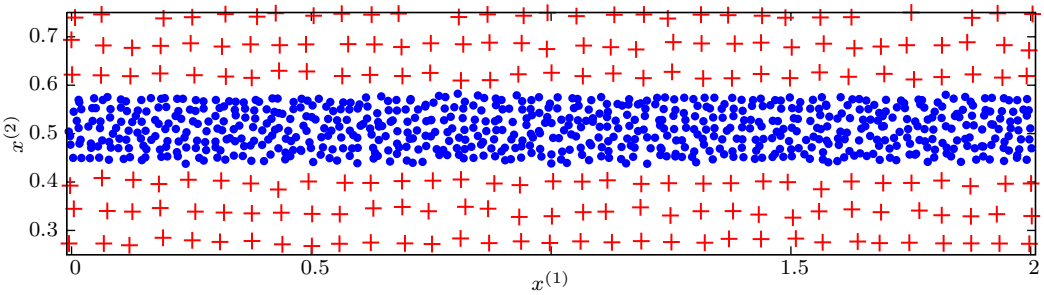
One drawback of the techniques considered here is that unbalanced class densities can have a significant impact on the effectiveness of noise reduction [111]. The problem is that there may be a tendency to remove good examples (i.e. not noise) from the minority class. Because all the techniques considered here are influenced by data density we conducted an evaluation to look at the risk of removing good examples from the minority class. We also looked at the impact of these noise reduction techniques on generalisation accuracy in the presence of unbalanced data.

We built an artificial dataset called den which contains no noise but the examples in different classes have different densities. The dataset is shown in Figure 7.4(a); it is created with a uniform 2-dimensional network of examples with a distance of 0.02 on each dimension for the central class and a distance of 0.06 on each dimension for the peripheral class, and applying Gaussian noise with $\sigma^2 = 0.005$ to all the examples.

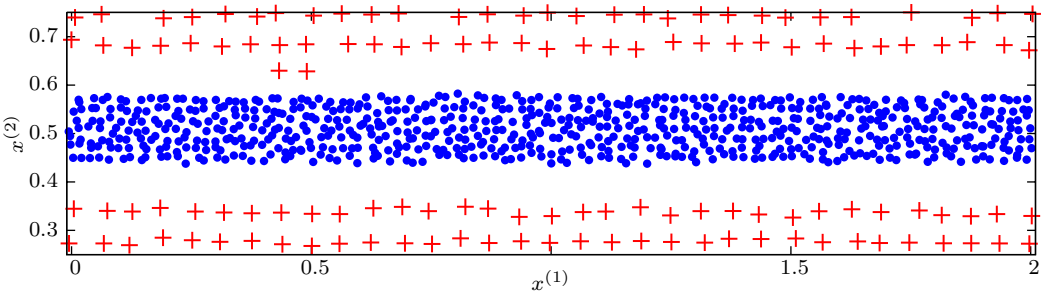
Figure 7.4(b) shows the behaviour of the RENN algorithm which removes almost all the examples of the external class that are closest to the internal class. Although the separation between classes is enlarged, this is achieved by removing only examples of the less dense class and it is clear that the generalisation capability of the edited set is extremely deteriorated. This behaviour is not caused by model selection problems as it will happen across a range of k values because the majority class will always *out vote* the minority class. The AkNN results shown in Figure 7.4(c) are very similar to those for RENN. This is not surprising because the same considerations discussed for RENN hold for AkNN as well.

The application of kNNSVM-nr on the den dataset is shown in Figure 7.4(d). We can observe that only 3 examples are incorrectly removed, meaning that the local SVM is able to correctly separate the classes in the neighbourhood of a borderline example even in the presence of uneven class densities. While the kNNSVM-nr strategy is performing well here it has been proposed for example by [136] to modify the penalty parameter of SVM for unbalanced data to further increase the generalisation accuracy. In fact, by increasing the penalty score associated with the peripheral class, the kNNSVM-nr performance can be improved so that it does not delete any examples of the minority class.

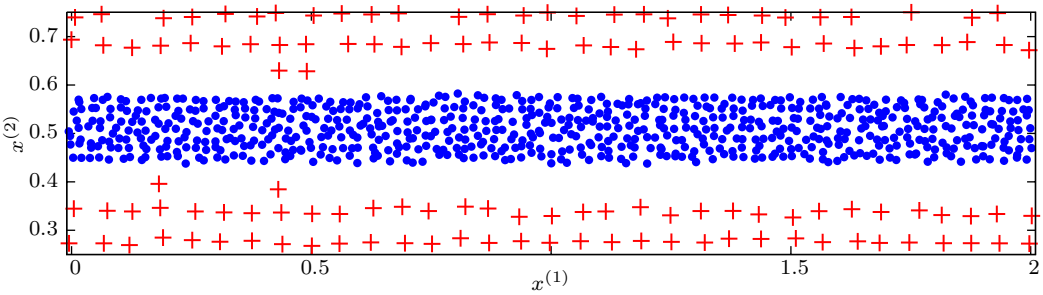
In order to understand the behaviour of the noise reduction techniques on real data with



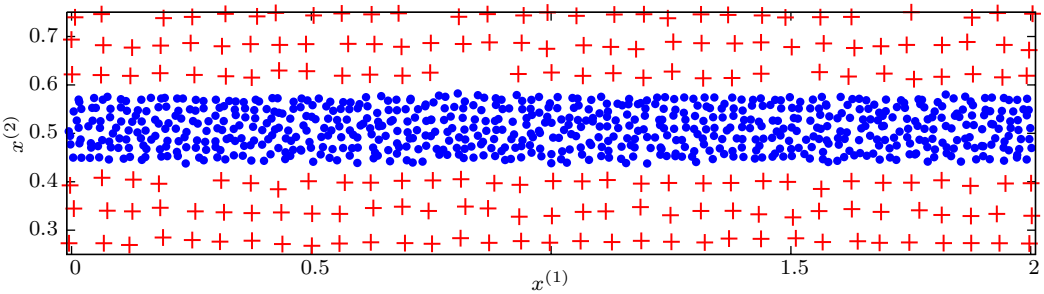
(a) The unedited den dataset.



(b) The den dataset preprocessed with RENN.



(c) The den dataset preprocessed with AkNN.



(d) The den dataset preprocessed with kNNSVM-nr.

Figure 7.4: The unedited den dataset and the noise reduction preprocessed versions.

different class densities, we selected from the datasets of section 7.3.1 two datasets with a considerable number of examples and on which RENN and AkNN perform similar to the kNNSVM-nr strategy. The datasets are musk2 and astro, and we modified them by randomly removing 75% of examples of the already less populated class thus obtaining two datasets with unbalanced class densities. The results of the noise reduction techniques (for kNNSVM-nr the class penalties are not modified) are shown in Table 7.8. While the three techniques achieve very similar test classification results with the original datasets, kNNSVM-nr is clearly better than RENN and AkNN for the unbalanced versions. The results confirm the robustness of kNNSVM-nr for unbalanced class densities.

7.4 Fast and Scalable Noise Reduction with Local Kernel Machines

The kNNSVM-nr method, described in the previous sections, showed an excellent ability of removing noisy examples compared to state-of-the-art noise reduction techniques in a number of different scenarios. The only drawback of kNNSVM-nr concerns the computational performances that make problematic its application for large and very large datasets. For this reason, we introduce here the **Fa**st **L**ocal **K**ernel Machine **N**oise **R**eduction (FaLKNR), which is scalable for large datasets and it is developed starting from the kNNSVM-nr method. Various modifications and optimization strategies, based on those introduced for fast and scalable local kernel machines in Chapter 6, are introduced to make it suitable for large datasets and CBR systems.

7.4.1 The Formulation of FaLKNR

We formulated kNNSVM-nr starting from the probabilistic output of kNNSVM. Similarly, we can formulate FaLKNR simply using the FaLK-SVM classifier we introduced in the previous chapter, applying it in the training set and removing the examples whose labels are not in accordance with the corresponding predictions. The edited training set produced by FaLKNR is thus:

$$\mathcal{X}' = \{\mathbf{x}_i \in \mathcal{X} | \text{FaLK-SVM}(\mathbf{x}_i) = y_i\}. \tag{7.2}$$

Notice that, since the focus of FaLKNR is on the scalability performances, we do not convert the output of the local SVM to a probability. In fact, we want to avoid the tuning of the threshold parameter γ of kNNSVM-nr (in the case of datasets with two classes it is conceptually equivalent to set the threshold to 0.5) and we want to make the SVM model construction faster (the probabilistic approach following [144] and [114] requires a cross-validation step). Moreover, always for computational reasons, the local SVM models avoid to exclude from the training process the examples whose label will be predicted. For this reason, FaLKNR can be more conservative than kNNSVM-nr in removing noisy examples.

The behaviour of FaLKNR is shown in Figure 7.5 for a subset of 4 centers for clearness, and in Figure 7.6 on the entire training set.

Since computational efficiency is our main objective here, particular attention must be put on

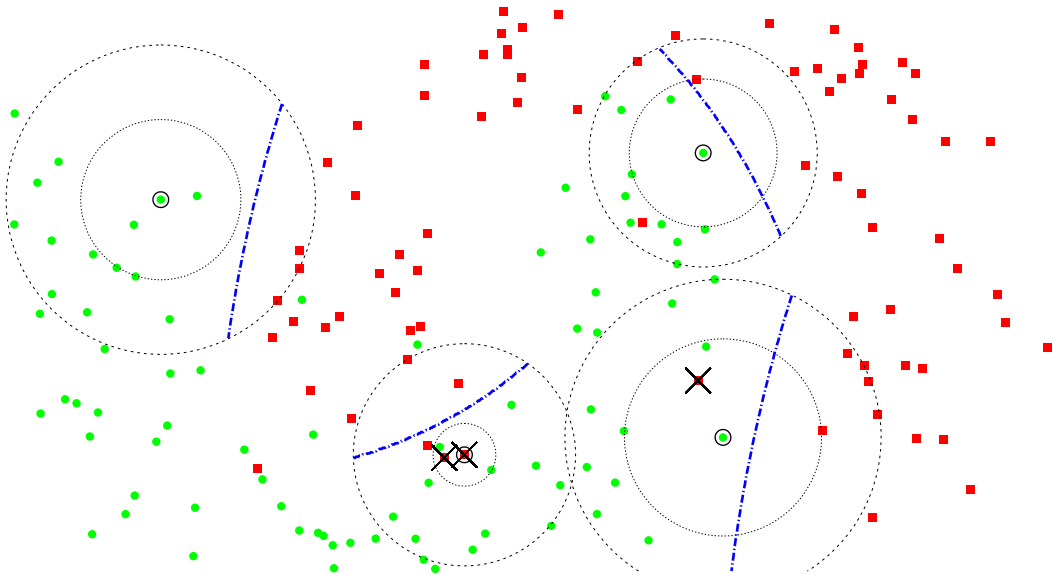


Figure 7.5: The application of FaLKNR on a toy dataset, with neighbourhood size $k = 15$, assignment neighbourhood size $k' = 4$, regularisation parameter $C = 10$, with the RBF kernel with $\sigma = 0.1$ on a toy dataset. Only the models corresponding to 4 centers are reported for clearness.

the model selection strategy. FaLKNR uses the approach we detailed in Chapter 6.1.3 for efficient local model selection. In our experiments we will use the RBF kernel which is a general purpose kernel that has demonstrated very high classification accuracies for SVM. In FaLKNR we set the width parameter σ of RBF kernel to be the double of the squared median of the histogram of the distances in the local model. More formally, $\sigma = 2 \cdot m^2[\|x - x'\|^{\mathbb{R}^p_k}]$ where $m[\|x - x'\|]$ is the median of the distance distribution the k points of the local models. As already discussed in Chapter 3.3, this procedure is motivated by the fact that the obtained σ value is of the same order of magnitude as the distances that it weights. In this way the kernel width is adaptive to the possibly different characteristics of different sub-regions of the training set. For non-low values of k, σ is computed on a random subset of points for computational reasons. The local k'-internal κ-fold CV model selection for FaLK-SVM defined in 6.1.3 for FaLK-SVMl, is used also for FaLKNR in order to choose the regularisation parameter C. In particular C is selected in the set $\{1, 10, 100\}$, using $\kappa = 10$ and a subset of 10 k-neighbourhoods.

7.4.2 Computational Complexity of FaLKNR

Hypothesizing the worst scaling behaviour for the training of each local SVM model to be $\mathcal{O}(k^3)$, and remembering that the nearest neighbour operations with Cover Trees can be done in $\log(N)$, FaLKNR requires $\mathcal{O}(N \log N)$ for building the Cover Tree, $\mathcal{O}(|\mathcal{C}| \cdot \log N \cdot k)$ for retrieving the local models, $\mathcal{O}(|\mathcal{C}| \cdot k^3)$ for training the local SVMs, and $\mathcal{O}(k \cdot N)$ for predicting if each training point is a noisy point or not. This means that the overall complexity of FaLKNR is

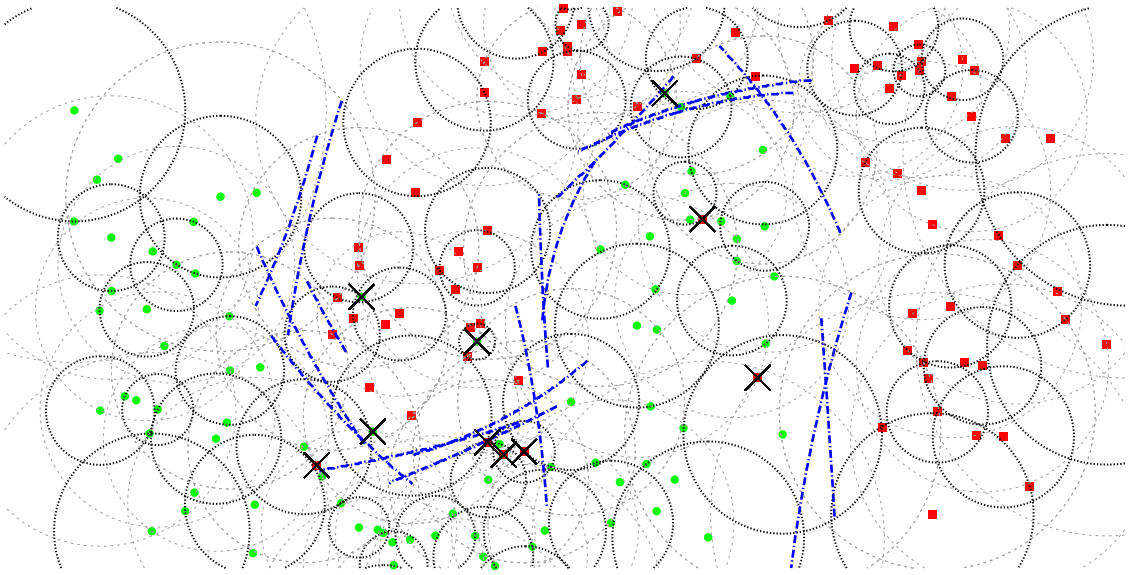


Figure 7.6: The application of FaLKNR on a toy dataset, with neighbourhood size $k = 15$, assignment neighbourhood size $k' = 4$, regularisation parameter $C = 10$, with the RBF kernel with $\sigma = 0.1$ on a toy dataset.

$\mathcal{O}(N \log N + |\mathcal{C}| \cdot \log N \cdot k + |\mathcal{C}| \cdot k^3 + k \cdot N)$, which is, assuming a fixed and reasonably low value for k, sub-quadratic (in particular $\mathcal{O}(N \log(N))$) even considering the worst case in which $k' = 1$ and thus $|\mathcal{C}| = N$. Moreover, FaLKNR can be very easily parallelized, because the training (and testing) of the local SVMs can occur in parallel on different processors.

kNNSVM-nr has a complexity of $\mathcal{O}(N^2 \log N + N \cdot k^3)$. The only work that, as far as we know, is focused on computational performances for noise reduction ([6]) has a complexity of $\mathcal{O}(N^2)$. ENN, using a brute-force nearest neighbour approach, scales like $\mathcal{O}(N^2 \log k)$ but, using Cover Trees, its complexity can be lowered to $\mathcal{O}(N \log N + k \cdot N \log N)$, which is thus of the same complexity class of FaLKNR with respect to N. RENN and AkNN have the same complexity as ENN, with the addition of a small constant (for RENN the number of recursive applications, for AkNN the neighbourhood size k).

As for the computational space requirements, since FaLKNR performs SVM training on small subregions (assuming a reasonable low k), there are no problems with fitting the kernel matrix into main memory. This results in an overall space requirement of $\mathcal{O}(N + |\mathcal{C}| \cdot k^2)$, i.e. linear in N.

7.5 Empirical Evaluation of FaLKNR

We compare FaLKNR to ENN, RENN and AkNN the state-of-the-art methods for competence enhancing as discussed in Chapter 2.3.4. The comparison is made on the basis of nearest neighbour (NN) generalisation accuracies. We implemented FaLKNR using our Cover Trees

name	# training examples	# testing examples	# features	# classes	source
ijcnn1	49990	91701	22	2	LibSVM Rep. [36]
connect4	50669	16888	41	3	UCI Rep. [7]
seismic	78823	19705	50	3	LibSVM Rep. [36]
acoustic	78823	19705	50	3	LibSVM Rep. [36]
2-spirals	100000	100000	2	2	Segata et al. [165]
census-inc	199523	99762	41	2	UCI Rep. [7]
poker_hand	300000	725010	10	2	UCI Rep. [7]
cod-rna	364651	121549	8	2	Uzilov et al. [191]
cov-type	571012	10000	54	2	LibSVM Rep. [36]

Table 7.9: The datasets used for the empirical evaluation.

implementation and LibSVM [36] for local SVM training and prediction; the source code of FaLKNR is freely available as a module of the Fast Local Kernel Machine Library (FaLKM-lib) [163]. The Cover Trees are used to implement ENN and AkNN as well. Although it is not computationally efficient, RENN can be realised by simply recursively applying ENN until no examples are removed. kNNSVM-nr is not considered because is not scalable for large datasets[2]. The experiments are carried out on an AMD Athlon™ 64 X2 Dual Core Processor 5000+, 2600MHz, with 3.56Gb of RAM.

7.5.1 Experimental Procedure

The k and k' parameters of FaLKNR are set to 1000 and 250 respectively. There are no particular strategies to select such values, but we intuitively considered them a good compromise between local and global behaviours (for k) and between generalisation accuracies and computational performance (for k'). The other parameters are chosen or estimated as detailed in Section 7.4.1. In the case of ENN, RENN and AkNN we fixed $k = 3$ as done, among others, by Wilson and Martinez [204]. Notice that, choosing an odd number for k, ties in the majority rule are avoided[3]. However, for AkNN, the $k = 2$ case is considered and thus the number of ties in the majority rule can be large. Two versions of AkNN are thus taken into account: in AkNN an example is removed in the case of a tie, while in AkNNc (more conservative) the example is not removed in the case of a tie.

For the evaluation we used the datasets with less than 60 features and more than 45000 training examples available on the LibSVM [36] and UCI [7] repositories, an artificial dataset described in [165] and the bioinformatics dataset provided in [191]. If no separate testing sets are available we randomly chose one quarter of the data for testing, apart for the cov-type dataset

[2] kNNSVM-nr on the smallest dataset we present here takes more than 10 hours without considering model selection.

[3] Ties in the majority rule can still happens even with $k = 3$ if multiple points are at the same distance from the query point at the k-th position. However in the datasets considered here the number of points at the same position is negligible and the dimensionality is low, and thus ties with odd k values are extremely rare.

dataset	NN accuracies (in %)					
	unedited	FaLKNR	ENN	RENN	AkNN	AkNNc
ijcnn1	96.6	**96.7**	96.3	*96.0*	*96.0*	96.2
connect4	*66.2*	**69.8**	69.3	68.3	69.3	69.4
seismic	*65.3*	**73.3**	71.9	72.6	72.2	71.8
acoustic	*67.4*	**75.3**	73.7	74.2	74.0	73.8
2-spirals	*83.2*	**88.6**	87.6	88.1	87.9	87.7
census-inc	*92.6*	**94.5**	94.2	94.3	94.4	94.3
poker_hand	*56.6*	**60.7**	57.8	58.3	58.3	58.0
cod-rna	**96.3**	95.8	*94.0*	*94.0*	94.3	94.3
cov-type	**95.8**	95.4	95.2	*95.0*	95.1	95.2

Table 7.10: NN accuracies using the analysed techniques to edit the training sets. In bold and italics are highlighted the best and worst results.

for which we selected 10000 testing points (this because for this dataset it is necessary to have almost all the points for good classification results) and for poker_hand for which we added 275000 testing examples to the training set in order to make it larger. The datasets are listed in Table 7.9 and are all scaled in the range $[0, 1]$ (apart for 2-spirals which is in the $[-2, 2]$ range).

7.5.2 Results and Discussion

Table 7.10 reports the NN generalisation accuracies obtained using the original (unedited) training set and the training sets edited with the analysed techniques. FaLKNR improves on the accuracy achieved with the unedited training sets for 7 of the 9 datasets and in a number of cases the improvements are considerable. ENN, RENN, AkNN and AkNNc are also able to improve the NN generalisation accuracy in the majority of the datasets, but their improvements are *always* lower than the FaLKNR ones. If we use the Wilcoxon signed-ranks test to assess the significance of this table of results [61], the improvements due to FaLKNR are statistically significant ($\alpha = 0.05$) with respect to all the other analysed techniques and with respect to the unedited training set.

As reported in Table 7.11, the total computational times for FaLKNR (including local model selection and the local SVM training/prediction) are between 39 seconds for ijcnn1 and about 38 minutes for poker_hand (2230 seconds). In the last column of Table 7.11 we report the speedups of FaLKNR with respect to ENN (implemented using Cover Trees). We chose ENN for this comparison because it is in any case faster than RENN, AkNN and AkNNc, and thus the speedups of FaLKNR with respect to RENN, AkNN and AkNNc are higher than reported in the table. The speedups are always higher than 1 except for the 2-spirals dataset (97 seconds for FaLKNR, 44 for ENN). These favourable computational results are due to the fact that it is faster to perform $|\mathcal{C}|$ retrievals of the $k = 1000$ nearest neighbours than it is to perform N retrievals of $k = 3$ nearest neighbours. This advantage is maintained when training $|\mathcal{C}|$ local SVMs, confirming that the training (and the prediction) of SVMs with 1000 points is extremely

dataset	Computational times (in seconds)		computational speedup
	ENN	FaLKNR	FaLKNR w.r.t. ENN
ijcnn1	61	39	1.6
connect4	1244	455	2.7
seismic	3025	950	3.2
acoustic	2641	331	8.0
2-spirals	44	97	0.5
census-inc	6965	771	9.0
poker_hand	16904	2230	7.6
cod-rna	3340	550	6.1
cov-type	1538	993	1.5

Table 7.11: Computational times of FaLKNR and ENN (the fastest among ENN, RENN and AkNN) and speedups of FaLKNR with respect to ENN.

fast. The only dataset in which this does not hold is the 2-spirals dataset because it is a very complex classification problem and thus the local SVM models are rather slow to train and because it has only two features and thus the nearest neighbour operations of ENN are very efficient.

Although our objective here is competence enhancement, it is interesting to look at the size of the edited training sets reported in Figure 7.7. The correlation between the unedited training set NN accuracies and the size of the edited training sets is evident, and this is an indirect confirmation that the tested techniques do home in on noisy examples. FaLKNR is the method that removes less examples in almost all the datasets. One may thus argue that the reason why FaLKNR outperforms the other techniques in improving NN accuracies is related to the fact that it is less aggressive in removing examples. However, we can notice that the difference in training set reduction between AkNN and AkNNc is consistent, but AkNNc does not permit better NN accuracies. This let us conclude that the advantages of FaLKNR over other techniques is not simply due to its more conservative policy.

We also include comparisons with RENN as it is the most popular noise reduction technique used in the literature. Moreover we include BBNR in the evaluation because as it has only been applied for the spam filtering task, it is of interest to test its performance in general classification problems.

7.6 Conclusions

In this chapter, we presented a novel noise reduction technique, called kNNSVM-nr, based on the probabilistic output of the Local Support Vector Machine classifier trained on the neighbourhood of each training set example. The evaluation shows that this approach is able to improve with statistical significance the generalisation accuracy of 1NN and 3NN classifiers on a number of real datasets and on artificial datasets with increasing levels of noise in both features and labels. We selected AkNN, RENN and BBNR as the alternative noise reduction techniques against which

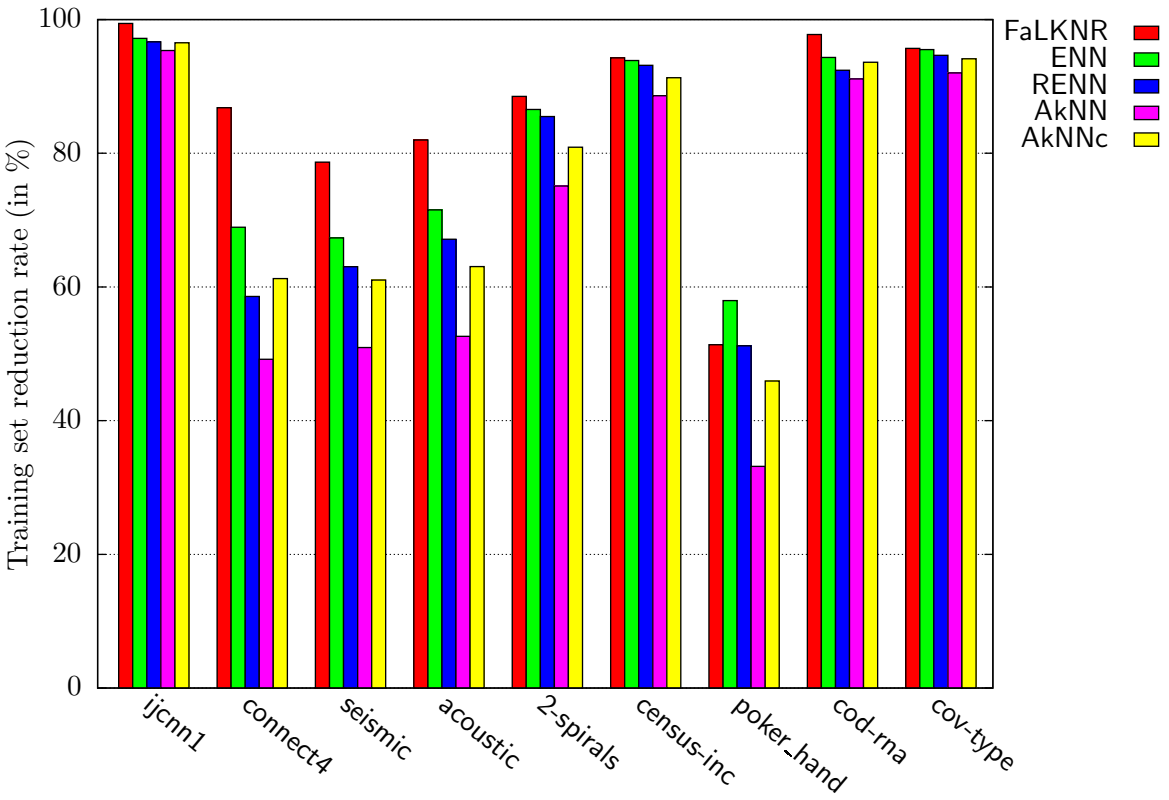


Figure 7.7: Percentage sizes of the training sets edited with the analysed techniques.

we would evaluate our new strategy. We selected AkNN and RENN because, while there are other strategies that achieve better reduction in training set size, these are most effective at improving generalisation accuracy [203]. We chose BBNR because we are interested in spam filtering, the application area where that technique originates and because we were curious about why its good performance there is not reproduced in other application areas. kNNSVM-nr has shown to be more effective that AkNN and RENN for general datasets, for Gaussian noise, for data with different class densities and, together with BBNR, in the specific field of spam filtering.

We have also presented FaLKNR, a scalable noise reduction technique for large and very large problems based on kNNSVM-nr. It includes a number of optimizations to achieve a theoretical complexity bound of $\mathcal{O}(n \log(n))$ for non high-dimensional data using strategies very similar to those introduced in Chapter 6. This makes it possible to apply the method on datasets with more than 500000 samples. Our empirical evaluation carried out in comparison with the state-of-the-art noise reduction techniques represented by ENN, AkNN and RENN, demonstrated that FaLKNR is the fastest and permits the highest NN accuracy improvements.

Chapter 8

Conclusion

In this work we have showed that locality can be a crucial property in machine learning in order to obtain learning systems with higher performance both in terms of prediction accuracies and in terms of computational complexity and scalability.

We have started our work with a theoretical analysis and an empirical evaluation of the Local SVM approach. The bound we have derived for Local SVM shows that the approach can lower the generalization error of SVM. The experimental validation have confirmed the statistically significant improvements achievable with Local SVM with respect to SVM using non-local kernel. If a local kernel is used, instead, the accuracy improvements of Local SVM on small datasets are not evident, but we have highlighted that, in presence of highly non-linear datasets, it performs substantially better than SVM. These conclusions reported in detail in Chapter 4 enabled us the research direction we successfully developed in Chapter 5, Chapter 6 and Chapter 7.

We have developed Quasi-Local kernels that are a new class of kernel functions in which it is possible to regulate the balancing between possibly global input kernels and the local kernel defined in the feature-space of the input kernel. Quasi-Local kernels are positive-definite (PD) given that the input kernels are PD, they are universal and, if the input kernel is local, can simulate a local kernel with locally tunable parameters. Efficient and effective methods for setting the width of the feature-space local component and its balancing with the global ones have been proposed. Theoretical advantages of Quasi-Local kernels include the ability of better capturing the decision function in data with uneven distribution, with variable spatial resolution, with low a-priori knowledge of the shape of the separation and with both long-range extrapolation characteristics (a general global shape of the separation) and local characteristics (local adaptation of the separation).

We have introduced a general framework for Local Kernel Machines taking inspiration from the idea of Local SVM and Local Learning Algorithms, focusing non only on the accuracy capa-

bilities of the system but also on the computational performances. In our approach to the local learning we switch from the lazy learning setting (on which Local SVM, LLA, IBL and CBR are based) to the more efficient eager learning. In FaLK-SVM a set of local SVMs are trained on redundant neighbourhoods in the training set selecting at testing time the most appropriate model for the query points. Supported by the recent result relating consistency and localizability [210], our approach is not a way of approximating the accurate SVM decision function, but, under the assumption that the decision function estimated using only the neighbourhood of a query point and the global decision function are very similar in the subregion of the query point, it divides the separation function in solutions of local optimization problems that can be handled very efficiently. We are in fact able to consider all the points in the local neighbourhoods without any computational limitation on the total number of SVs which is the major problem for the application of SVM optimization (and of approximated SVM solvers) on large and very large datasets. Instead of trading locality for scalability smoothing the decision function such that it can be described with a lower number of SV, in FaLK-SVM locality is exploited to obtain accurate, fast and scalable prediction systems. Of course the advantages of locality decreases as the intrinsic dimensionality of the input space increases due to the "curse of dimensionality". The introduction of a fast local model selection further speedups the learning process. Learning and complexity bounds are derived for the novel approach that have a much larger application domain than the classification task.

We have further specialized the Local Kernel Machine approach to be applied in the context of noise reduction. In fact, it is possible to make use of the probability prediction of local SVM on the label of its central example to consider the possibility of removing it from the training set in order to have a less noisy dataset with accuracy advantages for IBL, CBR and kNN-based approaches for learning, computational performances gain for supervised learning in general, and cleansed data in biological and medical context. FkNNSVM-nr focuses on relatively small datasets on which the computational performances are not problematic, while FaLKNR make use of some strategies (similar to the ones used for FaLK-SVM) for applying the techniques to large datasets.

We have carried out an extensive experimental evaluation of all our novel techniques with respect to the state-of-the-art in various application fields both assessing the classification capabilities and the computational performances. A total of about 100 different datasets with up to 3 millions training examples and non high-dimensionality have been used for the experiments. The classification accuracies achieved with our local techniques are always very satisfactory improving over the state-of-the-art approaches: Quasi-Local kernels proved to be more accurate than corresponding input kernels (considering linear, polynomial, radial basis function and sigmoidal kernels) for the SVM method, FaLK-SVM showed to be statistically better than LibSVM and a number of approximated SVM solvers (using the RBF kernel) in lowering the generalization error, preprocessing with FkNNSVM-nr and FaLKNR permitted to kNN to achieve higher classification accuracies with respect to existing noise reduction techniques (AkNN, RENN, BBNR). From the computational viewpoint we have highlighted that Quasi-Local kernels do not add considerable overhead to SVM optimization, that FaLK-SVM is at least one order of magnitude faster than LibSVM and approximated solvers with comparable accuracies both for training and

for testing and the computational advantage increases with the sizes of training sets (tests performed on datasets with up to 3 millions examples), and that FaLKNR is faster than the other noise reduction techniques (tests performed on datasets with up to half a million examples).

We have thus showed that the combination of IBL and maximal margin approaches gives advantages over existing state-of-the-art techniques supported by theoretical and empirical evidence. From another viewpoint, we have presented effective techniques to tune the trade-off between local and global approaches that permits to achieve excellent results. Although we mainly focused on classification tasks, the introduced approaches are much more general permitting to apply them on any kernel methods (for Quasi-Local kernels) or any learning systems that is localizable (for Local Kernel Machines).

8.1 Availability and Applicability of the Proposed Methods

Quasi-Local kernels can be integrated in any machine learning software tool simply including the computation of the Quasi-Local kernel functions, or precomputing the kernel matrix if the dataset is not very large. So Quasi-Local kernels can be integrated in modern kernel based software like LibSVM [36], SVM-light [91], Shogun toolbox [174], and the approximated SVM solvers presented in Section 2.2 for supervised, unsupervised and semi-supervised learning, but also on software packages implementing other kernel methods.

The Local Kernel Machines algorithms we described and analysed in the thesis are available in the FaLKM-lib [163] software library. Specifically, FaLKM-lib is implemented in C++ and contains the fast implementation of kernel kNN, called FkNN, using the Cover Tree data-structure, the kNNSVM algorithm called FkNNSVM always implemented with Cover Trees (see Chapter 3), the noise reduction technique based on a probabilistic version of kNNSVM called FkNNSVM-nr (see Chapter 7), the fast and scalable version of kNNSVM called FaLK-SVM (subdivided in the two modules: FaLK-SVM-train and FaLK-SVM-predict) with its variants FaLK-SVMc and FaLK-SVMl (see Chapter 6), and the fast and scalable noise reduction technique called FaLKNR (see Chapter 7). The library contains tools common to all the modules for model selection, local model selection, automatic tuning of kernel parameters and performance evaluation. FaLKM-lib is an easy-to-use tool and can be used by people with very limited background in machine learning; the input format is the same sparse encoding of LibSVM and SVM-light and can be thus applied on the same datasets without preprocessing work.

8.2 Outline of Future Works

Multiple are the research directions enabled by the present work and that are discussed in the conclusions of Chapter 5, Chapter 6 and Chapter 7.

For Quasi-Local kernel interesting research directions include other families of operators like operators that are data- and distribution-dependent, Quasi-Local kernel with other isotropic stationary kernels instead of the RBF kernel, operators based on spectral angle mapper and operators that make the tuning of the kernel parameters simpler. Moreover it very interesting

to study the recursive and/or iterative application of Quasi-Local operators.

Concerning Local Kernel Machines, possible developments include a dimensionality reduction preprocessing step in order to attack also high-dimensional problems, the application of local classifiers different from SVM, and a distributed parallel version. In addition other data-structures for improving nearest neighbour operations can be investigated also considering approximated neighbourhood retrieval. It is also worth investigating a hierarchical application of the approach possibly with different clustering techniques.

Since the Local SVM for editing can be applied for redundancy reduction as well, we aim to develop and evaluate a modified version of FkNNSVM-nr for the competence preservation where the main objective is storage minimization preserving classification accuracies. Moreover, for large and noisy datasets, the noise reduction approach can be used in a two-stage SVM strategy in which FkNNSVM-nr is used before the global SVM training as already proposed in [10] and [174] in which the authors use traditional noise reduction methods. The purpose of local maximal margin noise reduction, in this case, is to remove the examples that are very likely to be considered bounded support vectors in training a global SVM in order to enlarge the class separation. In this way the optimization problem converges faster and the linear dependency between the number of support vectors and the training set cardinality is broken, and so the global SVM kernel matrix has a better chance of fitting into memory and thus dramatically speeding up the SVM training and testing phase.

Another research direction consists in applying the introduced techniques to specific research fields and problems. In bioinformatics, preprocessing biological experimental data with the proposed techniques seems to be very promising also because they can be applied to datasets reaching the genomic level. Regarding classification there are multiple problems in bioinformatics involving (potential) huge amounts of data with reduced dimensionality: analyses based on predicted secondary structure formation free energy changes [191], on measures of RNA secondary structure conservation and thermodynamic stability [199], on predictions of consensus structures of aligned sequences [198] are only some examples. Quasi-Local kernels and Local Kernel Machines can be applied also to specific problems requiring ad-hoc kernel functions like the spectrum kernel [109], the weighted degree kernels [147, 148, 162] and the tree kernels for structured data [196, 208]. These kernels do not consider feature-space locality and their classification performances can thus be increased with the discussed approaches.

Bibliography

[1] Agnar Aamodt and Enric Plaza. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *Artificial Intelligence Communication*, 7(1):39–59, 1994.

[2] Peyman Adibi and Reza Safabakhsh. Joint Entropy Maximization in the Kernel-Based Linear Manifold Topographic Map. In *International Joint Conference on Neural Networks (IJCNN 07)*, pages 1133–1138, 2007.

[3] David W. Aha. *Lazy Learning*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[4] David W. Aha, Dennis Kibler, and Marc K. Albert. Instance-Based Learning Algorithms. *Machine Learning*, 6(1):37–66, 1991.

[5] Shun-Ichi Amari and Si Wu. Improving Support Vector Machine Classifiers by Modifying Kernel Functions. *Neural Networks*, 12(6):783–789, 1999.

[6] Fabrizio Angiulli. Fast Nearest Neighbor Condensation for Large Data Sets Classification. *IEEE Transactions on Knowledge and Data Engineering*, 19(11):1450–1464, 2007.

[7] Arthur Asuncion and David Newman. UCI machine learning repository, 2007.

[8] Christopher G. Atkeson, Andrew W. Moore, and Stefan Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1-5):11–73, 1997.

[9] Francis R. Bach, Romain Thibaux, and Michael I. Jordan. Computing Regularization Paths for Learning Multiple Kernels. *Advances in Neural Information Processing Systems*, pages 1–8, 2005.

[10] Gökhan H. Bakır, Léon Bottou, and Jason Weston. Breaking SVM complexity with cross-training. *Advances in neural information processing systems*, 17:81–88, 2005.

[11] G. Baudat and Fatiha Anouar. Generalized Discriminant Analysis Using a Kernel Approach. *Neural Computation*, 12(10):2385–2404, 2000.

[12] Juan José Bello-Tomás, Pedro A. González-Calero, and Belén Díaz-Agudo. JColibri: An Object-Oriented Framework for Building CBR Systems. In *Advances in Case-Based Reasoning, 7th European Conference, (ECCBR 2004)*, pages 32–46, 2004.

[13] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. The Curse of Dimensionality for Local Kernel Machines. Technical Report 1258, Departement dinformatique et recherche operationnelle, Universite de Montreal, 2005.

[14] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. The Curse of Highly Variable Functions for Local Kernel Machines. *Advences on Neural Information Process Systems*, 18:107–114, 2006.

[15] Alina Beygelzimer, Sham Kakade, and John Langford. Cover Trees for Nearest Neighbor. In *Proceedings of the 23rd international conference on Machine learning (ICML 06)*, pages 97–104, New York, NY, USA, 2006. ACM.

[16] Jock A Blackard and Denis J Dean. Comparative Accuracies of Artificial Neural Networks and Discriminant Analysis in Predicting Forest Cover Types from Cartographic Variables. *Computers and Electronics in Agriculture*, 24:131–151, 1999.

[17] Enrico Blanzieri and Anton Bryl. Evaluation of the highest probability SVM nearest neighbor classifier with variable relative error cost. In *CEAS 2007*, Mountain View, California, 2007.

[18] Enrico Blanzieri and Anton Bryl. Instance-Based Spam Filtering Using SVM Nearest Neighbor Classifier. In David Wilson and Geoff Sutcliffe, editors, *FLAIRS Conference*, pages 441–442. AAAI Press, 2007.

[19] Enrico Blanzieri and Farid Melgani. An adaptive SVM nearest neighbor classifier for remotely sensed imagery. In *IEEE International Conference on Geoscience and Remote Sensing Symposium, 2006. (IGARSS 2006)*, pages 3931–3934, 2006.

[20] Enrico Blanzieri and Farid Melgani. Nearest Neighbor Classification of Remote Sensing Images With the Maximal Margin Principle. *IEEE Transactions on Geoscience and Remote Sensing*, 46(6):1804–1811, 2008.

[21] Antoine Bordes and Léon Bottou. The Huller: a Simple and Efficient Online SVM. In *Machine Learning: ECML 2005*, Lecture Notes in Artificial Intelligence, LNAI 3720, pages 505–512. Springer Verlag, 2005.

[22] Antoine Bordes, Léon Bottou, and Patrick Gallinari. SGD-QN: Careful Quasi-Newton Stochastic Gradient Descent. *Journal of Machine Learning Research*, 10:1737–1754, July 2009.

[23] Antoine Bordes, Seyda Ertekin, Jason Weston, and Léon Bottou. Fast Kernel Classifiers with Online and Active Learning. *Journal of Machine Learning Research*, 6:1579–1619, 2005.

[24] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A Training Algorithm for Optimal Margin Classifiers. In *Fifth annual workshop on Computational learning theory (COLT 92)*, pages 144–152, New York, NY, USA, 1992. ACM.

[25] Léon Bottou, Corinna Cortes, JS Denker, H. Drucker, I. Guyon, LD Jackel, Y. LeCun, UA Muller, E. Sackinger, P. Simard, and Vladimir N. Vapnik. Comparison of Classifier Methods: a Case Study in Handwritten Digit Recognition. In *12th IAPR International Conference on Pattern Recognition*, volume 2, 1994.

[26] Léon Bottou and Chih-Jen Lin. Support vector machine solvers. In Léon Bottou, Olivier Chapelle, Dennis DeCoste, and Jason Weston, editors, *Large Scale Kernel Machines*, pages 301–320. MIT Press, Cambridge, MA., 2007.

[27] Léon Bottou and Vladimir N. Vapnik. Local Learning Algorithms. *Neural computation*, 4(6):888–900, 1992.

[28] Olivier Bousquet and Daniel J.L. Herrmann. On the Complexity of Learning the Kernel Matrix. *Advances in Neural Information Processing Systems*, 15:399–406, 2003.

[29] Henry Brighton and Chris Mellish. Advances in Instance Selection for Instance-Based Learning Algorithms. *Data Mining and Knowledge Discovery*, 6(2):153–172, 2002.

[30] Carla E. Brodley. Addressing the Selective Superiority Problem: Automatic Algorithm/-Model Class Selection. In *10th International Machine Learning Conference (ICML)*, pages 17–24. Amherst, MA, 1993.

[31] David Broomhead and David Lowe. Multivariable Functional Interpolation and Adaptive Networks. *Complex Systems*, 2:321–355, 1988.

[32] Yaile Caballero, Simone Joseph, Yuniesky Lezcano, Rafael Bello, Maria M. Garcia, and Yaimara Pizano. Using Rough Sets to Edit Training Set in k-NN Method. In *fifth International Conference on Intelligent Systems Design and Applications, (ISDA 05)*, pages 456–463, Washington, DC, USA, 2005. IEEE Computer Society.

[33] Mike Cameron-Jones. Instance Selection by Encoding Length Heuristic with Random Mutation Hill Climbing. In *8th Australian Joint Conference on Artificial Intelligence*, pages 99–106, 1995.

[34] Guoqing Cao, Simon C. K. Shiu, and Xizhao Wang. A fuzzy-rough approach for case base maintenance. In *4th International Conference on Case-Based Reasoning (ICCBR 01)*, pages 118–130, London, UK, 2001. Springer-Verlag.

[35] Zehra Cataltepe, Yaser S. Abu-Mostafa, and Malik Magdon-Ismail. No Free Lunch for Early Stopping. *Neural Computation*, 11(4):995–1009, 1999.

[36] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a Library for Support Vector Machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[37] Chin-Liang Chang. Finding Prototypes for Nearest Neighbor Classifiers. *IEEE Transactions on Computers*, C-23(11):1179–1184, 1974.

[38] Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. Coordinate Descent Method for Large-scale L2-loss Linear Support Vector Machines. *Journal of Machine Learning Research*, 9:1369–1398, 2008.

[39] Qun Chang, Qingcai Chen, and Xiaolong Wang. Scaling gaussian rbf kernel width to improve svm classification. In *International Conference on Neural Networks and Brain (ICNN&B '05)*, volume 1, pages 19–22, 2005.

[40] Hwann-Tzong Chen, Huang-Wei Chang, and Tyng-Luh Liu. Local Discriminant Embedding and Its Variants. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 05)*, volume 2, pages 846–853, Washington, DC, USA, 2005. IEEE Computer Society.

[41] Long Chen. New Analysis of the Sphere Covering Problems and Optimal Polytope Approximation of Convex Bodies. *Journal of Approximation Theory*, 133(1):134, 2005.

[42] Haibin Cheng, Pang-Ning Tan, and Rong Jin. Localized Support Vector Machine and Its Efficient Algorithm. *SIAM International Conference on Data Mining*, 2007.

[43] Heeyoul Choi and Seungjin Choi. Robust Kernel Isomap. *Pattern Recognition*, 40(3):853–862, 2007.

[44] Chien-Hsing Chou, Bo-Han Kuo, and Fu Chang. The Generalized Condensed Nearest Neighbor Rule as A Data Reduction Method. In *18th International Conference on Pattern Recognition (ICPR 06)*, pages 556–559, Washington, DC, USA, 2006. IEEE Computer Society.

[45] Vasek Chvatal. A Greedy Heuristic for the Set-Covering Problem. *Mathematics of operations research*, pages 233–235, 1979.

[46] Kenneth L. Clarkson. Nearest Neighbor Queries in Metric Spaces. In *Twenty-ninth annual ACM symposium on Theory of computing (STOC 97)*, pages 609–617, New York, NY, USA, 1997. ACM.

[47] William S. Cleveland. Robust Locally Weighted Regression and Smoothing Scatterplots. *Journal of the American Statistical Association*, 74:829–836, 1979.

[48] William S. Cleveland and Susan J. Devlin. Locally weighted regression: an approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83:596–610, 1988.

[49] Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras, and Peter L. Bartlett. Exponentiated Gradient Algorithms for Conditional Random Fields and Max-Margin Markov Networks. *Journal of Machine Learning Research*, 9:1775–1822, 2008.

[50] Ronan Collobert, Fabian Sinz, Jason Weston, and Léon Bottou. Trading Convexity for Scalability. In *23rd International Conference on Machine Learning (ICML 06)*, pages 201–208, New York, NY, USA, 2006. ACM.

[51] Corinna Cortes and Vladimir N. Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995.

[52] Thomas M. Cover and Peter E. Hart. Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.

[53] Koby Crammer, Joseph Keshet, and Yoram Singer. Kernel Design using Boosting. *Advances in Neural Information Processing Systems*, 15:1–8, 2002.

[54] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press New York, NY, USA, 1999.

[55] Pádraig Cunningham, Dónal Doyle, and John Loughrey. An Evaluation of the Usefulness of Case-Based Explanation. *Case-Based Reasoning Research and Development: 5th International Conference on Case-Based Reasoning*, pages 122–130, 2003.

[56] Isaac Martn de Diego, Javier M. Moguerza, and Alberto Munoz. Combining Kernel Information for Support Vector Classification. *fifth International Workshop on Multiple Classifier Systems (MCS 04)*, pages 102–111, 2004.

[57] Vin De Silva and Joshua B. Tenenbaum. Global versus Local Methods in Nonlinear Dimensionality Reduction. In *Advances in Neural Information Processing Systems*, volume 15, pages 705–712, 2003.

[58] D. DeCoste. Visualizing Mercer Kernel Feature Spaces via Kernelized Locally-Linear Embeddings. In *8th International Conference on Neural Information Processing*, 2001.

[59] Sarah Jane Delany and Derek Bridge. Textual Case-Based Reasoning for Spam Filtering: A Comparison of Feature-Based and Feature-Free Approaches. *Artificial Intelligence Review*, 26(1-2):75–87, 2006.

[60] Sarah Jane Delany and Pádraig Cunningham. An Analysis of Case-Based Editing in a Spam Filtering System. In P. Funk and P. González-Calero, editors, *Advances in Case-Based Reasoning, 7th European Conference on Case-based Reasoning (ECCBR 04)*, volume 3155 of *LNAI*, pages 128–141. Springer, 2004.

[61] Janez Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[62] Pierre A. Devijver and Josef Kittler. *Pattern Recognition: a Statistical Approach*. Prentice Hall, Englewood Cliffs, London, 1982.

[63] Belén Díaz-Agudo, Pedro A. González-Calero, Juan A. Recio-García, and Antonio A. Sánchez-Ruiz-Granados. Building cbr systems with jcolibri. *Special Issue on Experimental Software and Toolkits of the Journal Science of Computer Programming*, 69(1-3):68–75, 2007.

[64] Jian-xiong Dong. Fast SVM Training Algorithm with Decomposition on Very Large Data Sets. *IEEE Transaction Pattern Analysis and Machine Intelligence*, 27(4):603–618, 2005. Senior Member-Krzyzak, Adam and Fellow-Suen, Ching Y.

[65] Marco F. Duarte and Yu Hen Hu. Vehicle Classification in Distributed Sensor Networks. *Journal of Parallel and Distributed Computing*, 64(7):826–838, 2004.

[66] Olive Jean Dunn. Multiple Comparisons among Means. *Journal of the American Statistical Association*, pages 52–64, 1961.

[67] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

[68] Rong-En Fan, Pai-Hsuen Chen, and Chih-Jen Lin. Working Set Selection Using Second Order Information for Training Support Vector Machines. *The Journal of Machine Learning Research*, 6:1889–1918, 2005.

[69] Mathieu Fauvel, Jocelyn Chanussot, and Jon Atli Benediktsson. Evaluation of Kernels for Multiclass Classification of Hyperspectral Remote Sensing Data. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 06)*, 2:813–816, 2006.

[70] Ronald A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[71] Milton Friedman. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, pages 675–701, 1937.

[72] Milton Friedman. A Comparison of Alternative Tests of Significance for the Problem of m Rankings. *The Annals of Mathematical Statistics*, pages 86–92, 1940.

[73] Yan Fu, Qiang Yang, Ruixiang Sun, Dequan Li, Rong Zeng, Charles X. Ling, and Wen Gao. Exploiting the kernel trick to correlate fragment ions for peptide identification via tandem mass spectrometry. *Bioinformatics*, 20(12):1948–1954, 2004.

[74] Dragan Gamberger, Nada Lavrac, and Saso Dzeroski. Noise Detection and Elimination in Data Preprocessing: Experiments in Medical Domains. *Applied Artificial Intelligence*, pages 205–223, 2000.

[75] Michael R Garey and David S Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. WH Freeman & Co. New York, NY, USA, 1979.

[76] Geoffrey W. Gates. The Reduced Nearest Neighbor Rule. *IEEE Transactions on Information Theory*, 18(3):431–433, 1972.

[77] Mark G. Genton. Classes of Kernels for Machine Learning: a Statistics Perspective. *The Journal of Machine Learning Research*, 2:299–312, 2002.

[78] Mark Girolami and Simon Rogers. Hierarchic Bayesian Models for Kernel Learning. In *22nd International Conference on Machine Learning (ICML 05)*, pages 241–248, New York, NY, USA, 2005. ACM.

[79] TR Golub, DK Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, JP Mesirov, H. Coller, ML Loh, JR Downing, MA Caligiuri, et al. Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. *Science*, 286(5439):531, 1999.

[80] Antonin Guttman. R-Trees: a Dynamic Index Structure for Spatial Searching. *ACM Sigmod Record*, 14(2):47–57, 1984.

[81] Xiulan Hao, Chenghong Zhang, Hexiang Xu, Xiaopeng Tao, Shuyun Wang, and Yunfa Hu. An Improved Condensing Algorithm. In *Seventh IEEE/ACIS International Conference on Computer and Information Science (ICIS 2008)*, pages 316–321, Washington, DC, USA, 2008. IEEE Computer Society.

[82] Peter E. Hart. The Condensed Nearest Neighbor Rule. *IEEE Transactions on Information Theory*, 14(3):515–516, 1968.

[83] Xiaofei He, Shuicheng Yan, Yuxiao Hu, and Hong-Jiang Zhang. Learning a Locality Preserving Subspace for Visual Recognition. In *Ninth IEEE International Conference on Computer Vision (ICCV 03)*, page 385, Washington, DC, USA, 2003. IEEE Computer Society.

[84] Seth Hettich and Stephen D. Bay. The UCI KDD Archive. Irvine, CA: University of California. *Department of Information and Computer Science*, 1999. `kdd.ics.uci.edu`.

[85] T. K. Ho and E. M. Kleinberg. Building Projectable Classifiers of Arbitrary Complexity. In *13th International Conference on Pattern Recognition (ICPR 96)*, page 880, Washington, DC, USA, 1996. IEEE Computer Society.

[86] Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and S. Sundararajan. A Dual Coordinate Descent Method for Large-Scale Linear SVM. In *Proceedings of the 25th international conference on Machine learning)ICML 08)*, pages 408–415, New York, NY, USA, 2008. ACM.

[87] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A Practical Guide to Support Vector Classification. Technical report, Department of Computer Science, National Taiwan University, 2003.

[88] Chih-Wei Hsu and Chih-Jen Lin. A Comparison of Methods for Multiclass Support Vector Machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

[89] D. Huang and Tommy W. S. Chow. Enhancing Density-Based Data Reduction Using Entropy. *Neural Computation*, 18(2):470–495, 2006.

[90] Yuan Jiang and Zhi-Hua Zhou. Editing Training Data for knn Classifiers with Neural Network Ensemble. In F. Yin, J. Wang, and C. Guo, editors, *Advances in Neural Networks (ISNN 2004)*, volume 3173 of *LNCS*, pages 356–361. Springer, 2004.

[91] Thorsten Joachims. Making Large-Scale Support Vector Machine Learning Practical. *Advances in kernel methods: support vector learning*, pages 169–184, 1999.

[92] Thorsten Joachims. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–226. ACM New York, NY, USA, 2006.

[93] Thorsten Joachims, Thomas Finley, and Chun-Nam Yu. Cutting-Plane Training of Structural SVMs. *Machine Learning*, pages 1–33, 2009.

[94] Thorsten Joachims and Chun-Nam Yu. Sparse Kernel SVMs via Cutting-Plane Training. *Machine Learning*, 2009.

[95] Michael J Kearns and Umesh V Vazirani. *An Introduction to Computational Learning Theory*. MIT Press Cambridge, MA, USA, 1994.

[96] Sathiya Keerthi, Olivier Chapelle, and Dennis DeCoste. Building Support Vector Machines with Reduced Classifier Complexity. *Journal of Machine Learning Research*, 7:1493–1515, 2006.

[97] Sathiya Keerthi and Dennis DeCoste. A Modified Finite Newton Method for Fast Solution of Large Scale Linear SVMs. *Journal of Machine Learning Research*, 6:341–361, 2005.

[98] R.D. King, C. Feng, and A. Sutherland. STATLOG: Comparison of Classification Algorithms on Large Real-World Problems. *Applied Artificial Intelligence*, 9(3):289–333, 1995.

[99] S. Knerr, L. Personnaz, G. Dreyfus, J. Fogelman, A. Agresti, MA Ajiz, A. Jennings, F. Alizadeh, F. Alizadeh, J.P.A. Haeberly, et al. Single-Layer Learning Revisited: a Stepwise Procedure for Building and Training a Neural Network. *Optimization Methods and Software*, 1:23–34, 1990.

[100] Jack Koplowitz and Thomas A. Brown. On the Relation of Performance to Editing in Nearest Neighbor Rules. *Pattern Recognition*, 13(3):251–255, 1981.

[101] Robert Krauthgamer and James R. Lee. Navigating nets: simple algorithms for proximity search. In *Fifteenth annual ACM-SIAM symposium on Discrete algorithms (SODA 04)*, pages 798–807, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[102] U. Kressel et al. Pairwise Classification and Support Vector Machines. *Advances in kernel methods: support vector learning*, pages 255–268, 1999.

[103] Gert R. G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the Kernel Matrix with Semidefinite Programming. *The Journal of Machine Learning Research*, 5:27–72, 2004.

[104] Gert R. G. Lanckriet, Tijl De Bie, Nello Cristianini, Michael I. Jordan, and William Stafford Noble. A Statistical Framework for Genomic Data Fusion. *Bioinformatics*, 20(16):2626–2635, 2004.

[105] Ken Lang. NewsWeeder: Learning to Filter Netnews. In *12th International Conference on Machine Learning*, 1995.

[106] David B. Leake. CBR in Context: the Present and Future. *Case-based reasoning: Experiences, lessons, and future directions*, pages 3–30, 1996.

[107] Yuh-jye Lee and Olvi L Mangasarian. RSVM: Reduced Support Vector Machines. In *First SIAM International Conference on Data Mining*, 2001.

[108] Yuh-Jye Lee and Olvi L. Mangasarian. SSVM: A Smooth Support Vector Machine for Classification. *omputational Optimization and Applications*, 20(1):5–22, 2001.

[109] Christina Leslie, Eleazar Eskin, and William Stafford Noble. The Spectrum Kernel: a String Kernel for SVM Protein Classification. In *Pacific Symposium on Biocomputing*, volume 7, pages 566–575, 2002.

[110] Darrin P. Lewis, Tony Jebara, and William Stafford Noble. Nonstationary Kernel Combination. In *Proceedings of the 23rd international conference on Machine learning (ICML 06)*, pages 553–560, New York, NY, USA, 2006. ACM.

[111] Rong-Lu Li and Jun-Fa Hu. Noise Reduction to Text Categorization based on Density for KNN. In *International Conference on Machine Learning and Cybernetics, 2003*, volume 5, pages 3119–3124, 2003.

[112] Chih-Jen Lin, Ruby C. Weng, and S. Sathiya Keerthi. Trust Region Newton Methods for Large-Scale Logistic Regression. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 561–568, New York, NY, USA, 2007. ACM.

[113] Hsuan-Tien Lin and Chih-Jen Lin. A Study on Sigmoid Kernels for SVM and the Training of non-PSD Kernels by SMO-type Methods. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, 2003.

[114] Hsuan-Tien Lin, Chih-Jen Lin, and Ruby C. Weng. A Note on Platt's Probabilistic Outputs for Support Vector Machines. *Machine Learning*, 68(3):267–276, October 2007.

[115] Kuan-Ming Lin and Chih-Jen Lin. A Study on Reduced Support Vector Machines. *IEEE Transactions on Neural Networks*, 14(6):1449–1459, 2003.

[116] Ting Liu, Andrew W. Moore, Alexander Gray, and Ke Yang. An Investigation of Practical Approximate Nearest Neighbor Algorithms. In Lawrence K Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, volume 17pages, pages 825–832, Cambridge, MA, 2005. MIT Press.

[117] Gaëlle Loosli and Stéphane Canu. Comments on the "Core Vector Machines: Fast SVM Training on Very Large Data Sets". *Journal of Machine Learning Research*, 8:291–301, 2007.

[118] Ana C. Lorena and Andr C. Carvalho. Evaluation of Noise Reduction Techniques in the Splice Junction Recognition Problem. *Genetics and Molecular Biology*, 27:665–672, 2004.

[119] David G. Lowe. Similarity Metric Learning for a Variable-Kernel Classifier. *Neural Computation*, 7(1):72–85, 1995.

[120] Andrea Malossini, Enrico Blanzieri, and Raymond T. Ng. Detecting Potential Labeling Errors in Microarrays by Data Perturbation. *Bioinformatics*, 22(17):2114–2121, September 2006.

[121] Andrea Malossini, Nicola Segata, and Enrico Blanzieri. Kernel Integration using von Neumann Entropy. Technical Report DISI-09-050 1666, University of Trento, Trento, Italy, 2009. Submitted to a journal.

[122] Olvi L. Mangasarian. A finite Newton method for classification. *Optimization Methods and Software*, 17(5):913–929, 2002.

[123] Mario Marchand and John Shawe-Taylor. The Set Covering Machine. *The Journal of Machine Learning Research*, 3:723–746, 2003.

[124] Elizabeth McKenna and Barry Smyth. Competence-Guided Case-Base Editing Techniques. In *5th European Workshop on Advances in Case-Based Reasoning (ECCBR 00)*, pages 186–197, London, UK, 2000. Springer-Verlag.

[125] Gregoire Mercier and Mark Lennon. Support Vector Machines for Hyperspectral Image Classification with Spectral-Based Kernels. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS 03)*, volume 1, pages 288–290, July 2003.

[126] Stefano Merler and Giuseppe Jurman. Terminated Ramp-Support Vector Machines: a Nonparametric Data Dependent Kernel. *Neural Networks*, 19(10):1597–1611, 2006.

[127] Charles A. Micchelli, Yuesheng Xu, and Haizhang Zhang. Universal Kernels. *Journal of Machine Learning Research*, 7:2651–2667, 2006.

[128] Donald Michie, David J. Spiegelhalter, Charles C. Taylor, and John Campbell, editors. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, Upper Saddle River, NJ, USA, 1994.

[129] Sebastian Mika, Gunnar Rätsch, Jason Weston, Bernard Scholkopf, and Klaus.R. Muller. Fisher Discriminant Analysis with Kernels. In *Neural Networks for Signal Processing IX, 1999*, pages 41–48, 1999.

[130] Renqiang Min, Anthony Bonner, and Zhaolei Zhang. Modifying Kernels Using Label Information Improves SVM Classification Performance. In *Sixth International Conference on Machine Learning and Applications (ICMLA 07)*, pages 13–18, Washington, DC, USA, 2007. IEEE Computer Society.

[131] Pabitra Mitra, C. A. Murthy, and Sankar K. Pal. Density-Based Multiscale Data Condensation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(6):734–747, 2002.

[132] Paul Nemenyi. *Distribution-Free Multiple Comparisons*. PhD thesis, Princeton, 1963.

[133] Conor Nugent, Dónal Doyle, and Pádraig Cunningham. Gaining Insight through Case-Based Explanation. *Journal of Intelligent Information Systems*, 32(3), 2009.

[134] Stephen M. Omohundro. Efficient Algorithms with Neural Network Behavior. *Complex Systems*, 1:273–347, 1987.

[135] Cheng Soon Ong, Alexander J. Smola, and Robert C. Williamson. Learning the Kernel with Hyperkernels. *The Journal of Machine Learning Research*, 6:1043–1071, 2005.

[136] Edgar Osuna, Robert Freund, and Federico Girosi. Support Vector Machines: Training and Applications. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1997.

[137] Rong Pan, Qiang Yang, and Sinno Jialin Pan. Mining Competent Case Bases for Case-Based Reasoning. *Artificial Intelligence*, 171(16-17), 2007.

[138] Jae Heon Park, Kwang Hyuk Im, Chung-Kwan Shin, and Sang Chan Park. MBNR: Case-Based Reasoning with Local Feature Weighting by Neural Network. *Applied Intelligence*, 21(3):265–276, 2004.

[139] Paul Pavlidis, Jason Weston, Jinsong Cai, and William Noble Grundy. Gene Functional Classification from Heterogeneous Data. In *Fifth Annual International Conference on Computational Biology (RECOMB 01)*, pages 249–255, New York, NY, USA, 2001. ACM.

[140] Z. Pawlak. *Rough Sets: Theoretical Aspects of Reasoning about Data*. Kluwer Academic Publishers Norwell, MA, USA, 1992.

[141] Mykola Pechenizkiy, Alexey Tsymbal, Seppo Puuronen, and Oleksandr Pechenizkiy. Class Noise and Supervised Learning in Medical Domains: The Effect of Feature Extraction. In *19th IEEE Symposium on Computer-Based Medical Systems (CBMS 06)*, pages 708–713, Washington, DC, USA, 2006. IEEE Computer Society.

[142] Jing Peng, Douglas R. Heisterkamp, and H. K. Dai. Adaptive Quasiconformal Kernel Nearest Neighbor Classification. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 26(5):656–661, 2004.

[143] John C. Platt. Fast Training of Support Vector Machines using Sequential Minimal Optimization. *MIT Press Cambridge, MA, USA*, pages 185–208, 1999.

[144] John C. Platt. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In Peter J. Bartlett, Bernhard Schölkopf, Dale Schuurmans, and Alex J. Smola, editors, *Advances in Large Margin Classifiers*, pages 61–74, Boston, 1999. MIT Press.

[145] John C. Platt, Nello Cristianini, and John Shawe-Taylor. Large Margin DAGs for Multiclass Classification. *Advances in neural information processing systems*, 12(3):547–553, 2000.

[146] Ross J. Quinlan. The effect of noise on concept learning. In R.S. Michalski, J.G. Carboneel, and T.M. Mitchell, editors, *Machine Learning: an Artificial Intelligence Approach*, volume 2, pages 149–166. Morgan Kaufmann, 1986.

[147] Gunnar Rätsch and Sören Sonnenburg. Accurate Splice Site Detection for Caenorhabditis elegans. *Kernel Methods in Computational Biology*, pages 277–298, 2004.

[148] Gunnar Rätsch, Sören Sonnenburg, and Bernard Schölkopf. RASE: Recognition of Alternatively Spliced Exons in C.elegans. *Bioinformatics*, 21(1):369–377, 2005.

[149] Sandro Ridella, Stefano Rovetta, and Rodolfo Zunino. Circular Backpropagation Networks for Classification. *IEEE Transactions on Neural Networks*, 8:84–97, 1997.

[150] G. Ritter, H. Woodruff, S. Lowry, and T. Isenhour. An Algorithm for a Selective Nearest Neighbor Decision Rule. *IEEE Transactions on Information Theory*, 21(6):665–669, 1975.

[151] Thomas Roth-Berghofer. Explanations and Case-Based Reasoning: Foundational Issues. In p Funk and P.A. González-Calero, editors, *Advances in Case-Based Reasoning, 7th European Conference on Case-based Reasoning (ECCBR 04)*, volume 3155. Springer, 2004.

[152] Sam T. Roweis and Lawrence K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326, 2000.

[153] M. Salamó and E. Golobardes. Global, Local and Mixed Rough Sets Case Base Maintenance Techniques. In *sixth Catalan Conference on Artificial Intelligence*, pages 127–134. IOS Press, 2004.

[154] Maria Salamó and Elisabet Golobardes. Rough Sets Reduction Techniques for Case-Based Reasoning. In David W. Aha and Ian Watson, editors, *Case-Based Reasoning Research and Development, 4th International Conference on Case-Based Reasoning, (ICCBR 01)*, volume 2080 of *LNCS*, pages 467–482. Springer, 2001.

[155] Maria Salamó and Elisabet Golobardes. Deleting and Building Sort Out Techniques for Case Base Maintenance. In Susan Craw and Alun D. Preece, editors, *Advances in Case-Based Reasoning, 6th European Conference on Case-Based Reasoning, (ECCBR 02)*, volume 2416 of *LNCS*, pages 365–379. Springer, 2002.

[156] J. S. Sánchez, R. Barandela, A. I. Marqués, R. Alejo, and J. Badenas. Analysis of New Techniques to Obtain Quality Training Sets. *Pattern Recognition Letters*, 24(7):1015–1022, April 2003.

[157] Bernard Schölkopf and Alexander J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT Press, 2002.

[158] Bernhard Schölkopf. *Support Vector Learning*. PhD thesis, Berlin, Techn. Univ., 1997.

[159] Bernhard Schölkopf. The Kernel Trick for Distances. *Advances in Neural Information Processing Systems*, pages 301–307, 2001.

[160] Bernhard Schölkopf, Patrice Simard, Alexander J. Smola, and Vladimir N. Vapnik. Prior knowledge in support vector kernels. In *NIPS '97: Proceedings of the 1997 conference on Advances in neural information processing systems 10*, pages 640–646, Cambridge, MA, USA, 1998. MIT Press.

[161] Bernhard Schölkopf, Alexander J. Smola, and Klaus R. Müller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, 10(5):1299–1319, 1998.

[162] Sebastian J. Schultheiss, Wolfgang Busch, Jan U. Lohmann, Olivier Kohlbacher, and Gunnar Ratsch. KIRMES: Kernel-Based Identification of Regulatory Modules in Euchromatic Sequences. *Bioinformatics, to appear*, 2009.

[163] Nicola Segata. FaLKM-lib v1.0: a Library for Fast Local Kernel Machines. Technical report, University of Trento, Trento, Italy, 2009. Software available at `http://disi.unitn.it/~segata/FaLKM-lib/`.

[164] Nicola Segata and Enrico Blanzieri. Empirical Assessment of Classification Accuracy of Local SVM. In *The 18th Annual Belgian-Dutch Conference on Machine Learning (Benelearn 2009)*, pages 47–55, Tilburg, Belgium, 2009.

[165] Nicola Segata and Enrico Blanzieri. Fast Local Support Vector Machines for Large Datasets. In Petra Perner, editor, *Machine Learning and Data Mining in Pattern Recognition: 6th International Conference (MLDM 09).*, Lecture Notes in Artificial Intelligence, pages 295–310, Leipzig, Germany, 2009. Springer.

[166] Nicola Segata and Enrico Blanzieri. Fast and Scalable Local Kernel Machines. Technical report, University of Trento, Trento, Italy, 2009. Submitted to a journal.

[167] Nicola Segata and Enrico Blanzieri. Operators for Transforming Kernels into Quasi-Local Kernels that Improve SVM Accuracy. Technical report, University of Trento, Trento, Italy, 2009. Submitted to a journal.

[168] Nicola Segata, Enrico Blanzieri, and Pádraig Cunningham. A scalable noise reduction technique for large case-based systems. In L Ginty and D.C Wilson, editors, *Case-Based Reasoning Research and Development: 8th International Conference on Case-Based Reasoning (ICCBR09)*, volume 09 of *Lecture Notes in Artificial Intelligence*, pages 755–758, Seattle, WA, USA, 2009. Springer.

[169] Nicola Segata, Enrico Blanzieri, Sarah Jane Delany, and Pádraig Cunningham. Noise reduction for instance-based learning with a local maximal margin approach. *Journal of Intelligent Information Systems*, Published, August 2009.

[170] Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal Estimated Sub-Gradient SOlver for SVM. In *24th international conference on Machine learning (ICML 07)*, pages 807–814, New York, NY, USA, 2007. ACM.

[171] Guido F. Smits and Elizabeth M. Jordaan. Improved SVM regression using mixtures of kernels. In *International Joint Conference on Neural Networks (IJCNN'02)*, volume 3, 2002.

[172] Alexander J. Smola, SVN Vishwanathan, and Quoc V. Le. Bundle methods for machine learning. *Advances in neural information processing systems*, 20:1377–1384, 2008.

[173] Barry Smyth and Mark T. Keane. Remembering to Forget: A Competence Preserving Case Deletion Policy for CBR System. In C. Mellish, editor, *14th International Joint Conference on Artificial Intelligence, (IJCAI 95)*, pages 337–382. Morgan Kaufmann, 1995.

[174] Sören Sonnenburg, Gunnar Rätsch, Christin Schäfer, and Bernhard Schölkopf. Large Scale Multiple Kernel Learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.

[175] Bharath K. Sriperumbudur and Gert Lanckriet. Nearest Neighbor Prototyping for Sparse and Scalable Support Vector Machines. Technical report, Tech. rep., Dept. of ECE, UCSD, 2007.

[176] Ingo Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, 2:67–93, 2002.

[177] Ingo Steinwart. Support Vector Machines are Universally Consistent. *Journal of Complexity*, 18(3):768–791, 2002.

[178] Ingo Steinwart. Sparseness of Support Vector Machines. *Journal of Machine Learning Research*, 4:1071–1105, 2003.

[179] Ingo Steinwart. Sparseness of Support Vector Machines-Some Asymptotically Sharp Bounds. In Sebastian Thrun, Lawrence Saul, and Bernard Sch\"{o}lkopf, editors, *Advances in Neural Information Processing Systems 16*, Cambridge, MA, 2004. The MIT Press.

[180] Ingo Steinwart. Consistency of support vector machines and other regularized kernel classifiers. *IEEE Transaction on Information Theory*, 51(1):128–142, 2005.

[181] Masashi Sugiyama. Local Fisher Discriminant Analysis for Supervised Dimensionality Reduction. In *23rd international conference on Machine learning (ICML 06)*, pages 905–912, New York, NY, USA, 2006. ACM.

[182] Johan A K Suykens and Joos P Vandewalle. Least Squares Support Vector Machine Classi ers. *Neural Processing Letters*, 9:293–300, 1999.

[183] Kim Tae and Josef Kittler. Locally Linear Discriminant Analysis for Multimodally Distributed Classes for Face Recognition with a Single Model Image. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 27(3):318–327, 2005.

[184] Sheng Tang and Si-Ping Chen. Data Cleansing Based on Mathematic Morphology. In *2nd International Conference on Bioinformatics and Biomedical Engineering (ICBBE 08)*, pages 755–758, 2008.

[185] Sheng Tang and Si-Ping Chen. An effective data preprocessing mechanism of ultrasound image recognition. In *2nd International Conference on Bioinformatics and Biomedical Engineering (ICBBE 08)*, pages 2708–2711, 2008.

[186] Joshua B. Tenenbaum, Vin De Silva, and John C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323, December 2000.

[187] Michael E. Thompson. NDCC: Normally Distributed Clustered Datasets on Cubes, 2006. `www.cs.wisc.edu/dmi/svm/ndcc/`.

[188] Ivan Tomek. An Experiment with the Edited Nearest-Neighbor Rule. *IEEE Transactions on Systems, Man and Cybernetics*, 6(6):448–452, 1976.

[189] Ivor W. Tsang, Andras Kocsor, and James T. Kwok. Simpler Core Vector Machines with Enclosing Balls. In *24th international conference on Machine learning (ICML 07)*, pages 911–918, New York, NY, USA, 2007. ACM.

[190] Ivor W. Tsang, James T. Kwok, and Pak-Ming Cheung. Core Vector Machines: Fast SVM Training on Very Large Data Sets. *Journal of Machine Learning Research*, 6:363–392, 2005.

[191] Andrew V. Uzilov, Joshua M. Keegan, and David H. Mathews. Detection of Non-Coding RNAs on the Basis of Predicted Secondary Structure Formation Free Energy Change. *BMC bioinformatics*, 7(1):173, 2006.

[192] Vladimir N. Vapnik. Principles of Risk Minimization for Learning Theory. *Advances in Neural Information Processing Systems*, 4:831–838, 1993.

[193] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2000.

[194] Vladimir N. Vapnik and Léon Bottou. Local Algorithms for Pattern Recognition and Dependencies Estimation. *Neural Computation*, 5(6):893–909, 1993.

[195] Vladimir N. Vapnik and Alexey J. Chervonenkis. On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Theory of Probability and its Applications*, 16:264–280, 1971.

[196] Jean-Philippe Vert. A Tree Kernel to Analyse Phylogenetic Profiles. *Bioinformatics*, 18(Suppl 1):S276–S284, 2002.

[197] Jigang Wang, Predrag Neskovic, and N. Leon Cooper. A Minimum Sphere Covering Approach to Pattern Classification. *International Conference on Pattern Recognition*, 3:433–436, 2006.

[198] Stefan Washietl and Ivo L. Hofacker. Consensus Folding of Aligned Sequences as a New Measure for the Detection of Functional RNAs by Comparative Genomics. *Journal of molecular biology*, 342(1):19–30, 2004.

[199] Stefan Washietl, Ivo L. Hofacker, and Peter F. Stadler. Fast and Reliable Prediction of Noncoding RNAs. *Proceedings of the National Academy of Sciences*, 102(7):2454–2459, 2005.

[200] Xun-Kai Wei and Ying-Hong Li. Linear Programming Minimum Sphere Set Covering for Extreme Learning Machines. *Neurocomputing*, 71(4–6):570–575, 2008.

[201] Stefan Wess, Klaus-Dieter Althoff, and Guido Derwand. Using k-d Trees to Improve the Retrieval Step in Case-Based Reasoning. In *First European Workshop on Topics in Case-Based Reasoning (EWCBR 93)*, pages 167–181, London, UK, 1994. Springer-Verlag.

[202] Frank Wilcoxon. Individual Comparisons by Ranking Methods. *Biometrics*, pages 80–83, 1945.

[203] D. Randall Wilson and Tony R. Martinez. Instance Pruning Techniques. In *14th International Conference on Machine Learning (ICML 97)*, pages 403–411, 1997.

[204] D. Randall Wilson and Tony R. Martinez. Reduction Techniques for Instance-BasedLearning Algorithms. *Machine Learning*, 38(3):257–286, 2000.

[205] Dennis L. Wilson. Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. *IEEE Transactions on Systems, Man and Cybernetics*, 2(3):408–421, 1972.

[206] Si Wu and Shun-Ichi Amari. Conformal Transformation of Kernel Functions: A Data-Dependent Way to Improve Support Vector Machine Classifiers. *Neural Processing Letters*, 15(1):59–67, 2002.

[207] Huilin Xiong, Ya Zhang, and Xue-Wen Chen. Data-Dependent Kernel Machines for Microarray Data Classification. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(4):583–595, 2007.

[208] Yoshihiro Yamanishi, Francis Bach, and Jean-Philippe Vert. Glycan Classification with Tree Kernels. *Bioinformatics*, 23(10):1211–1216, 2007.

[209] Alan L. Yuille and Anand Rangarajan. The Concave-Convex Procedure. *Neural Computation*, 15(4):915–936, 2003.

[210] Alon Zakai and Yaacov Ritov. Consistency and Localizability. *Journal of Machine Learning Research*, 10:827–856, 2009.

[211] Luca Zanni, Thomas Serafini, and Gaetano Zanghirati. Parallel Software for Training Large Scale Support Vector Machines on Multiprocessor Systems. *Journal of Machine Learning Research*, 7:1467–1492, 2006.

[212] Hao Zhang, Alexander C. Berg, Michael Maire, and Jitendra Malik. SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition. *Proc of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:2126–2136, 2006.

[213] Jianping Zhang. Selecting Typical Instances in Instance-Based Learning. In *9th international workshop on Machine learning (ML 92)*, pages 470–479, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[214] Alexander Zien, Gunnar Rätsch, Sebastian Mika, Bernhard Schölkopf, Thomas Lengauer, and Klaus.R. Muller. Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics*, 16(9):799–807, 2000.

Appendix A

The FaLKM-lib Software Library

FaLKM-lib [163] is a library for fast local kernel machine implemented in C++. It contains modules for classification, regression and noise reduction. All the neighbourhood operations are implemented with our implementation of Cover Trees [15] (see Chapter 3.5) for computational reasons, whereas the training and testing of the local SVM is performed using the LibSVM [36] code (version 2.88).

FkNN is a fast kernel-space implementation of kNN (see Chapter 3.1) both for classification and regression. FkNNSVM is the implementation of the Local SVM approach [19] for classification (and regression) as described in Chapter 3.4 and Chapter 4. FaLK-SVM-train and FaLK-SVM-test are the training and prediction methods for the fast and scalable approach for local support vector classification [166] (FaLK-SVM) as detailed in Chapter 6 which is applicable to very large datasets. The modules for noise reduction are FkNNSVM-nr (an updated version of kNNSVM-nr presented in [169] and in the first part of Chapter 2.3) and FaLKNR (presented in [168] and in the second part of Chapter 2.3). The library contains tools for model selection, local model selection and automatic tuning of kernel parameters. FaLKM-lib code is freely available for research and educational purposes at `http://disi.unitn.it/~segata/FaLKM-lib`.

All modules accept inputs and generate output accordingly to the LibSVM and SVM-light [91] file format which is defined as:

```
<line> .=. <target> <feat>:<val> <feat>:<val> ... <feat>:<val> # <info>
<target> .=. +1 | -1 | 0 | <float>
<feat> .=. <integer> | "qid"
<val> .=. <float>
<info> .=. <string>
```

In the following we briefly describe the modules of the library, listing the available options and some examples.

A.1 FkNN

FkNN takes the training set and the testing set as input and writes into the output file the class predictions of the testing examples. If the cross validation option -v is enabled FkNN accepts the training set only. FkNN prints into the standard output the accuracy (and FMeasure, precision and recall) results for classification and the mean squared error for regression.

FkNN is called in the following way:

```
FkNN [options] training_dataset_file [test_dataset_file output_label_file]
```

the available options are:

```
-K neighbourhood size (default 1)
-T classification (0, default) or regression (1)
-C conservative variant for the majority rule, (def.  0 = disabled, 1 enabled)
-R relaxation parameter for cover trees (default 1)
-t kernel_type : set type of kernel function (default 0)
        0 -- linear: u'*v
        1 -- polynomial: (gamma*u'*v + coef0)^degree
        2 -- radial basis function: exp(-gamma*|u-v|^2) [eq. to -t 0!!]
-d degree : set degree in kernel function (default 3)
-r coef0 : set coef0 in kernel function (default 1)
-s knn strategy : (default 0 = majority rule)
-g gamma : set gamma in kernel function (default = 1 for polynomial kernels)
-v n : n-fold cross validation mode
-S silent mode (default 0)
```

A.1.1 Examples

The 1NN results reported in [168] are obtained in the following way:

```
./FkNN -t 0 -K 1 -R 3 training_set.tr testing_set.te output_file.out
    > accuracy_results.nn
```

The 10-fold CV for a neighbourhood size of 10 using the metric induced by an inhomogeneous polynomial kernel of degree 3 can be obtained as follows:

```
./FkNN -t 1 -g 1 -d 3 -r 1 -K 5 -v 10 training_set.tr
```

A.2 FkNNSVM

FkNNSVM is the implementation of the kNNSVM algorithm [19] introduced in Chapter 3.4 and used for the experiments of Chapter 4, based on our implementation of Cover Trees [15] and on LibSVM [36] for training and testing the local SVM models. It integrates the Cover Trees (see Chapter 3.5) in order to achieve the computational complexity of $\mathcal{O}(k \log N + k^3)$ for each

testing point in addition to construction of the Cover Tree $\mathcal{O}(N \log N)$ (see Chapter 4.2). The crucial parameters for FkNNSVM are the selection of the classification (`-Z 0`) or the regression (`-Z 3`) option, the neighbourhood size `-K`, the SVM regularisation parameter `-c`, the kernel `-t` and its parameters. The option `-g` sets the value of the inverse of the RBF kernel width; setting for `-g` negative values in $[0, 1]$ indicates that the width parameter is automatically chosen using the strategy detailed in Chapter 3.3 using as percentile the absolute value of `-g` multiplied by 100. With the `-wi` is is possible to set different penalising constant for different class in order to improve the accuracy performances for unbalanced datasets.

The syntax for launching FkNNSVM is very similar to the syntax of FkNN:

```
FkNNSVM [options] training_dataset_file [test_dataset_file output_label_file]
```

the available options are:

```
-Z local SVM type (default 0 = C-SVM, 3 for epsilon-SVR)
-R relaxation parameter for cover trees (default 1)
-K neighbourhood size k
-t kernel_type : set type of kernel function (default 2)
    0 -- linear: u'*v
    1 -- polynomial: (gamma*u'*v + coef0)^degree
    2 -- radial basis function: exp(-gamma*|u-v|^2)
-c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)
-wi weight : set the parameter C of class i to weight*C, for C-SVC (default 1)
-p epsilon : set the epsilon in loss function of epsilon-SVR (default 0.1)
-d degree : set degree in kernel function (default 3)
-g gamma :  set gamma in kernel func., negative values for estimation based on
            histogram of distances (def. = -0.5 for RBF, 1 for pol. kernels)
-m cachesize : set cache memory size in MB (default 100)
-r coef0 : set coef0 in kernel function
-S silent mode (default 0)
```

A.2.1 Examples

The following setting is for applying the local SVM approach with neighbourhood size of 200, regularisation parameter C equals to 10, with the RBF kernel function whose width is computed on every local model as the median of the histogram of distances in the local neighbourhood:

```
./FkNNSVM -t 2 -c 10 -g -0.5 -K 200 training_set.tr test_set.te output_file.out
```

A.3 FaLK-SVM

FaLK-SVM is the implementation of the classification approach described in Chapter 6 and presented in [165, 166]. Roughly speaking, it is a modification of FkNNSVM in order to make it scalable for large and very large datasets. FaLK-SVM is subdivided in a training module, called

FaLK-SVM-train, that trains the model and in a prediction module, called FaLK-SVM-test, that makes the prediction on unseen examples using the trained model.

A.3.1 FaLK-SVM-train

FaLK-SVM-train takes the training set and the name of the output file on which the trained model will be stored (unless the model selection is enabled with `-v` option). The option `-P`, which takes values between 0 and 1, specifies the assignment neighbourhood size k' regulating the level of redundancy in covering the training set with local models (see Chapter 6 for details). FaLK-SVM-train can also enable the local model selection option with `-L 1` obtaining the FaLK-SVMl classifier introduced in Chapter6.1.3. FaLK-SVMl needs also the options for setting the number of local models that will be used for the procedure (with `-M`), the possible values for k with `-N` and the grid of SVM and kernel parameters to be considered: the regularisation parameter among the values specified with `-C`, the RBF kernel width and the degree of the polynomial kernels among the values specified with `-G` (possibly using negative values as detailed for FkNNSVM). It is also possible to train the model for obtaining probability estimates rather than predicted class labels with `-b 1`. The other options correspond to the FkNNSVM ones.

Specifically, the command line syntax is:

```
FaLK-SVM-train [options] input_dataset_file [model_file]
```

and the available options are:

```
-t kernel_type : set type of kernel function (default 2)
    0 -- linear: u'*v
    1 -- polynomial: (gamma*u'*v + coef0)^degree
    2 -- radial basis function: exp(-gamma*|u-v|^2)
-L local model selection (default 0)
    0 -- no local model selection
    1 -- local model selection on C K and kernel parameters on -M local models
         (the ranges are specified with -C -G -N)
-R relaxation parameter for cover trees (default 1)
-K neighbourhood size k
-P assignment neighbourhood size as fraction of K such that: K'=K*P (def. 0.5)
-d degree : set degree in kernel function (default 3)
-r coef0 : set coef0 in kernel function (default 0)
-c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)
-g gamma :  set gamma in kernel func., negative values for estimation based on
            histogram of distances (def. = -0.5 for RBF, 1 for pol. kernels)
-C c values for local grid model selection separated by ':'
-G G values for local grid model selection separated by ':'
-N K values for local grid model selection separated by ':'
-M number of local models used for local model selection
-m cachesize : set cache memory size in MB (default 100)
```

```
-wi weight: set the parameter C of class i to weight*C, (default 1)
-b probability_estimates : whether to train local models for probability
                           estimates, 0 or 1 (default 0)
-v n: n-fold cross validation mode
-S silent mode (default 0)
```

A.3.2 FaLK-SVM-test

FaLK-SVM-test uses the model trained with FaLK-SVM-train to predict the label of the query examples. It is possible to specify to use the FaLK-SVMc variant (with `-T 1`) instead of the default FaLK-SVM approach for prediction, in order to increase the computational performances. The probability output of FaLK-SVM can be enabled with `-b 1` (it is necessary that the model has been trained be FaLK-SVM-train with the same option).

FaLK-SVM is launched as follows:

```
FaLK-SVM-predict [options] test_file model_file output_label_file
```

and the available options are:

```
-T local model retrieval strategy (default 0)
    0 -- nn with all points
    1 -- nn with centers only
    2 -- knn with centers only
-b probability_estimates : whether to train local models for probability
                           estimates, 0 or 1 (default 0)
-K k for knn with centers only (default 3)
-M performance measure
-S silent mode (default 0)
```

A.3.3 Examples

The following example show how to train a model and predict the testing labels using FaLK-SVM with a neighbourhood size of 1000, assignment neighbourhood size 500 ($0.5\cdot$ neighbourhood size k'), regularisation parameter C equals to 10 and with the RBF kernel function whose width is computed on every local model as the 10th percentile of the histogram of distances in the local neighbourhood:

```
./FaLK-SVM-train -t 2 -c 10 -g -0.1 -K 1000 -P 0.5 training_set.tr model_file.m
./FaLK-SVM-predict testing_file.te model_file.m predictions.p
```

For estimating the empirical error using cross-validation with the same settings the command line is

```
./FaLK-SVM-train -t 2 -c 10 -g -0.1 -K 1000 -P 0.5 -v 10 training_set.tr
```

It is possible to use the local model selection and directly train the model using the FaLK-SVMl classifier. Based on the previous example, if we want to perform local model selection (using 10 local models) choosing C among $\{1, 10, 100, 1000\}$, g (the inverse of the width of the RBF kernel) among $\{-0.1, -0.5, -0.9\}$ (i.e. the 10th, 50th and 90th percentile of the distances) and the neighbour hood size k among $\{250, 500, 1000, 2000\}$ the proper command line is:

```
./FaLK-SVM-train -t 2 -L 1 -C 1.0:10.0:100.0:1000.0 -G -0.1:-0.5:-0.9
    -N 250:500:1000:2000 -P 0.5 -M 10 training_set.tr model_file.m
./FaLK-SVM-predict testing_file.te model_file.m predictions.p
```

The same model can be evaluated faster (but probably with less accuracy) using only the centers of the models to retrieve the local SVM (i.e. FaLK-SVMc):

```
./FaLK-SVM-predict -T 1 testing_file.te model_file.m predictions.p
```

A.4 FkNNSVM-nr

FkNNSVM-nr is the updated version of the noise reduction technique we presented in Chapter 7 and in [169] (available at `http://disi.unitn.it/~segata/LSVM-nr/LSVM-nr.html`). It takes the dataset and return the dataset without the noisy examples. The principal options are the neighbourhood size `-K`, the regularisation parameter `-c`, the kernel `-t` and its parameters `-d`, `-g` and `-r`. All these options are specified in the same way of FkNNSVM. For FkNNSVM-nr it is also possible to perform local model selection enabling it with `-L 1` and specifying the grid of parameters with the `-C`, `-G`, `-M` and `-N` options in the same way of FaLK-SVM.

The command line of FkNNSVM-nr is:

```
FkNNSVM-nr [options] input_dataset_file output_edited_dataset_file
```

and the available options are:

```
-R relaxation parameter for cover trees (default 1)
-K neighbourhood size k
-L local model selection (default 0)
    0 -- no local model selection
    1 -- local model selection on C K and kernel parameters on -M local models
         (the ranges are specified with -C -G -N)
-t kernel_type : set type of kernel function (default 2)
        0 -- linear: u'*v
        1 -- polynomial: (gamma*u'*v + coef0)^degree
        2 -- radial basis function: exp(-gamma*|u-v|^2)
-C c values for local grid model selection separated by ':'
-G G values for local grid model selection separated by ':'
-N K values for local grid model selection separated by ':'
-M number of local models used for local model selection
```

```
-m cachesize : set cache memory size in MB (default 100)
-wi weight: set the parameter C of class i to weight*C, for C-SVC (default 1)
-T balancing threshold for selecting local models for local model selection
-c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)
-d degree : set degree in kernel function (default 3)
-g gamma :  set gamma in kernel func., negative values for estimation based on
            histogram of distances (def. = -0.5 for RBF, 1 for pol. kernels)
-r coef0 : set coef0 in kernel function
```

A.4.1 Examples

The following example show hot to apply FkNNSVM-nr with the linear kernel performing the local model selection on a total of 20 local models choosing the regularisation parameter c in $\{0.01, 1.0, 100.0, 10000.0\}$ and the neighbourhood size k in $\{25, 50, 100, 200\}$.

```
./FkNNSVM-nr -t 0 -L 1 -C 0.01:1.0:100.0:10000.0 -N 25:50:100:200 -T 0.4 -M 20
    -R 1 training_set.tr edited_training_set.tr > output.txt
```

In the next example, we use the same setting of the previous one, but the kernel function is the RBF. Consequently the local model selection procedure has also to chose the width of the RBF kernel among $\{-0.1, -0.5, -0.9\}$ (i.e. the 10th, 50th and 90th percentile of the distances).

```
./FkNNSVM-nr -t 2 -L 1 -C 0.01:1.0:100.0:10000.0 -G -0.1:-0.5:-0.9 -N 25:50:100
    -T 0.4 -M 20 -R 1 training_set.tr edited_training_set.tr > output.txt
```

A.5 FaLKNR

FaLKNR is the fast and scalable noise reduction technique we presented in the second part of Chapter 7 and in [168] that integrates into the approach of FkNNSVM-nr the scalability performances of FaLK-SVM. Its options are basically the same of FkNNSVM-nr: the neighbourhood size `-K`, the kernel `-t`, the regularisation parameter `-c`, the kernel parameters `-d`, `-g` and `-r`, and the local model selection options `-L 1`, `-C`, `-G`, `-N`, `-M`. The `-P` option specifies the size of the assignment neighbourhood (the k' parameter of LKM) as $k' = k \cdot \texttt{P}$.

The command line for FaLKNR is:

```
FaLKNR [options] input_dataset_file edited_dataset_file
```

the available options are:

```
-K neighbourhood size k
-P assignment neighbourhood size as fraction of K such that: K'=K*P (def. 0.5)
-t kernel_type : set type of kernel function (default 2)
        0 -- linear: u'*v
        1 -- polynomial: (gamma*u'*v + coef0)^degree
        2 -- radial basis function: exp(-gamma*|u-v|^2)
```

```
-L local model selection (default 0)
    0 -- no local model selection
    1 -- local model selection on C K and kernel parameters on -M local models
         (the ranges are specified with -C -G -N)
-R relaxation parameter for cover trees (default 3)
-d degree : set degree in kernel function (default 3)
-g gamma :  set gamma in kernel func., negative values for estimation based on
            histogram of distances (def. = -0.5 for RBF, 1 for pol. kernels)
-r coef0 : set coef0 in kernel function (default 0)
-c cost : set the parameter C (default 1)
-C c values for local grid model selection separated by ':'
-G G values for local grid model selection separated by ':'
-N K values for local grid model selection separated by ':'
-M number of local models used for local model selection
-m cachesize : set cache memory size in MB (default 100)
-wi weight: set the parameter C of class i to weight*C, for C-SVC (default 1)
-S silent mode (default 0)
```

A.5.1 Examples

The results for FaLKNR detailed in [168] and in Chapter 7 are obtained performing local model selection on C, estimating the g parameter (inverse of RBF kernel width) based on the median of the histogram of distances in the local neighbourhood, the neighbourhood size fixed to 1000 and the assignment neighbourhood size to 250 ($0.25 * k$). The command for this setting is:

```
./FaLKNR -t 2 -L 1 -R 3 -K 1000 -P 0.25 -C 1.0:10.0:100.0 -g -0.5 -M 10
    training_set.tr edited_training_set.tr
```

A.6 Other names for the FaLKM-lib modules

Throughout all the present thesis, we refer to the implementations of the classifiers we developed with the names used in the FaLKM-lib library. However, our published papers do not always follow this convention, and the reader who also read the corresponding papers may be misleaded. For this reason, we report in Table A.1, the names we use for the techniques implemented in FaLKM-lib in other papers.

Name in FaLKM-lib	Name in other papers	Papers with the same name
FkNN	kNN in [168, 169]	[166]
FkNNSVM	kNNSVM (prel. ver.) in [165, 164, 169, 168]	[166]
FaLK-SVM	FastLSVM (prel. ver.) in [165]	[166]
FkNNSVM-nr	LSVM noise reduction (prel. ver.) in [169], [168]	
FaLKNR		[168]

Table A.1: Table of different names used in other papers for the FaLKM-lib modules.

Appendix B

Candidate's List of Publication

Here is the list of publications of the candidate relative to its PhD period.

Papers related to the arguments of the thesis

Nicola Segata and Enrico Blanzieri. Fast Local Support Vector Machines for Large Datasets. In Petra Perner, editor, *Machine Learning and Data Mining in Pattern Recognition: 6th International Conference (MLDM 09).* ***Best Paper Award***, Lecture Notes in Artificial Intelligence, pages 295–310, Leipzig, Germany, 2009. Springer. `http://www.springerlink.com/content/g1r24g05137g3518/`

Abstract: Local SVM is a classification approach that combines instance-based learning and statistical machine learning. It builds an SVM on the feature space neighborhood of the query point in the training set and uses it to predict its class. There is both empirical and theoretical evidence that Local SVM can improve over SVM and kNN in terms of classification accuracy, but the computational cost of the method permits the application only on small datasets. Here we propose FastLSVM, a classifier based on Local SVM that decreases the number of SVMs that must be built in order to be suitable for large datasets. FastLSVM precomputes a set of local SVMs in the training set and assigns to each model all the points lying in the central neighborhood of the k points on which it is trained. The prediction is performed applying to the query point the model corresponding to its nearest neighbor in the training set. The empirical evaluation we provide points out that FastLSVM is a good approximation of Local SVM and its computational performances on big datasets (a large artificial problem with 100000 samples and a very large real problem with more than 500000 samples) dramatically ameliorate performances of SVM and its fast existing approximations improving also the generalization accuracies.

Nicola Segata, Enrico Blanzieri, Sarah Jane Delany, and Pádraig Cunningham. Noise reduction for instance-based learning with a local maximal margin approach. *Journal of Intelligent Information Systems*, Online First, August 2009. `http://www.springerlink.com/content/y79tn63873805081/`

Abstract: To some extent the problem of noise reduction in machine learning has been finessed by the development of learning techniques that are noise-tolerant. However, it is difficult to make instance-based learning noise tolerant and noise reduction still plays an important role in k-nearest neighbour classification. There are also other motivations for noise reduction, for instance the elimination of noise may result in simpler models or data cleansing may be an end in itself. In this paper we present a novel approach to noise reduction based on local Support Vector Machines (LSVM) which brings the benefits of maximal margin classifiers to bear on noise reduction. This provides a more robust alternative to the majority rule on which almost all the existing noise reduction techniques are based. Roughly speaking, for each training example an SVM is trained on its neighbourhood and if the SVM classification for the central example disagrees with its actual class there is evidence in favour of removing it from the training set. We provide an empirical evaluation on 15 real datasets showing improved classification accuracy when using training data edited with our method as well as specific experiments regarding the spam filtering application domain. We present a further evaluation on two artificial datasets where we analyse two different types of noise (Gaussian feature noise and mislabelling noise) and the influence of different class densities. The conclusion is that LSVM noise reduction is significantly better than the other analysed algorithms for real datasets and for artificial datasets perturbed by Gaussian noise and in presence of uneven class densities.

Nicola Segata, Enrico Blanzieri, and Pádraig Cunningham. A scalable noise reduction technique for large case-based systems. In L Ginty and D.C Wilson, editors, *Case-Based Reasoning Research and Development: 8th International Conference on Case-Based Reasoning (ICCBR09)*, volume 09 of *Lecture Notes in Artificial Intelligence*, pages 755–758, Seattle, WA, USA, 2009. Springer. `http://www.springerlink.com/content/0m22615370411122/`

Abstract: Because case-based reasoning (CBR) is instance-based, it is vulnerable to noisy data. Other learning techniques such as support vector machines (SVMs) and decision trees have been developed to be noise-tolerant so a certain level of noise in the data can be condoned. By contrast, noisy data can have a big impact in CBR because inference is normally based on a small number of cases. So far, research on noise reduction has been based on a majority-rule strategy, cases that are out of line with their neighbors are removed. We depart from that strategy and use local SVMs to identify noisy cases. This is more powerful than a majority-rule strategy because it explicitly considers the decision boundary in the noise reduction process. In this paper we provide details on how such a local SVM strategy for noise reduction can be made scale to very large datasets (¿ 500,000 training samples). The technique is evaluated on nine very large datasets and shows excellent performance when compared with alternative techniques.

Nicola Segata and Enrico Blanzieri. Empirical Assessment of Classification Accuracy of Local SVM. In *The 18th Annual Belgian-Dutch Conference on Machine Learning (Benelearn 2009)*, pages 47–55, Tilburg, Belgium, 2009. `http://benelearn09.uvt.nl/Proceedings_Benelearn_09.pdf`

Abstract: The combination of maximal margin classifiers and k-nearest neighbors rule constructing an SVM on the neighborhood of the test sample in the feature space (called kNNSVM), was presented as a novel promising classifier. Since no extensive validation was performed yet, we test here kNNSVM on 13 widely used datasets obtaining statistically significant better classification results with respect to SVM for linear and polynomial kernels. For RBF kernels the advantages seems not to be substantial, but we present two toy datasets in which kNNSVM performs much better than SVM with RBF kernel. The empirical results suggest to use kNNSVM for specific problems in which high classification accuracies are crucial and motivates further refinements of the approach.

Nicola Segata and Enrico Blanzieri. Operators for Transforming Kernels into Quasi-Local Kernels that Improve SVM Accuracy. Technical report, University of Trento, Trento, Italy, 2009. `http://eprints.biblio.unitn.it/archive/00001359/` Currently under submission to

a machine learning journal.

Abstract: In the field of statistical machine learning, the integration of kernel methods with local information has been proposed through locality-improved kernels for Support Vector Machines (SVM) that make use of prior information, local kernels and local SVM that apply the SVM approach only on the subset of points close to the testing one. Here we propose a novel family of operators on kernels able to integrate the local information into any kernel without prior information obtaining quasi-local kernels. The quasi-local kernels maintain the possibly global properties of the input kernel and they increase the kernel value as the points get closer in the feature space of the input kernel. The operators combine the input kernel with a locality-dependent term, and accept two parameters that regulate the width of the exponential influence of points in the locality-dependent term and the balancing between the two terms. Experiments carried out with data-dependent systematic selection of the parameters of the operators (i.e. without the need for model selection phase on the obtained kernels) on a total of 33 datasets with different characteristics and application domains, achieve very good results.

Nicola Segata and Enrico Blanzieri. Fast and Scalable Local Kernel Machines. Currently under submission to a machine learning journal.

Abstract: A computational efficient approach for local learning with kernel methods is presented in this work. The **Fa**st **L**ocal **K**ernel **S**upport **V**ector **M**achine (FaLK-SVM) trains a set of local SVMs on redundant neighbourhoods in the training set and the most appropriate model for each query point is selected at testing time according to a nearest neighbour based strategy. Supported by a recent result by [210] relating consistency and localizability, our approach guarantees high generalization ability partitioning the separation function in local optimization problems that can be handled very efficiently. The introduction of a fast local model selection further speeds-up the learning process. Learning and complexity bounds are derived for FaLK-SVM, and the empirical evaluation of the approach (with datasets up to 3 million points) showed that it is much faster and more accurate and scalable than state-of-the-art accurate and approximated SVM solvers at least for non high-dimensional datasets. More generally, we show that locality can be an important factor to sensibly speed-up learning approaches and kernel methods, differently from other recent techniques that tend to disregard local information in order to achieve scalability.

Andrea Malossini, Nicola Segata and Enrico Blanzieri. Kernel Integration using von Neumann Entropy. Technical report, University of Trento, Trento, Italy, 2009. `http://eprints.biblio.unitn.it/archive/00001666/`

Abstract: Kernel methods provide a computational framework to integrate heterogeneous biological data from different sources for a wide range of learning algorithms by designing a kernel for each different information source and combining them in a unique kernel through simple mathematical operations. We develop here a novel technique for weighting kernels based on their von Neumann entropy. This permits to assess the kernel quality without using label information, and to integrate kernels before the beginning of the learning process. Moreover, we carry out a comparison with the unweighted kernel summation and a popular technique based on semi-definite programming on kernel integration benchmark data sets. Finally, we empirically study the variation of the performance of a support vector machine classifier considering pairs of kernels combined in different ratios, and we show how, surprisingly, the unweighted sum of kernels seems to lead to the same performance than a more complex weighting schema.

Nicola Segata. FaLKM-lib v1.0: a Library for Fast Local Kernel Machines. Technical report, University of Trento, Trento, Italy, 2009. Software available at `http://disi.unitn.it/~segata/FaLKM-lib/`. `http://eprints.biblio.unitn.it/archive/00001613/`

Abstract: FaLKM-lib v1.0 is a library for fast local kernel machine implemented in C++. It contains a fast implementation of kernel k-nearest neighbors (kNN) using the Cover Tree data-structure called FkNN, the local support vector machine (LSVM) algorithm called FkNNSVM, a noise reduction technique based on a probabilistic version of LSVM called FkNNSVM-nr, a fast and scalable version of LSVM called

FaLK-SVM (subdivided in the two modules: FaLK-SVM-train and FaLK-SVM-predict) and a fast and scalable noise reduction technique called FaLKNR. The library contains tools for model selection, local model selection and automatic tuning of kernel parameters. This document introduces the formulation of the algorithms in the software library; for a comprehensive discussion on the implemented techniques please refer to the papers cited in this document and for the use of the software refer to the README file included in the package. FaLKM-lib v1.0 code is freely available for research and educational purposes at `http://disi.unitn.it/~segata/FaLKM-lib`.

Papers related to systems biology and process algebras for computational biology

Here we report the candidate's publications related to systems biology and process algebras for computational biology that are not related with the thesis work but have been investigated during the first year of the PhD program.

Nicola Segata and Enrico Blanzieri. Stochastic π-Calculus Modelling of Multisite Phosphorylation Based Signaling: The PHO Pathway in Saccharomyces Cerevisiae. *Transactions on Computational Systems Biology X.*, pages 163-196, 2008. `http://www.springerlink.com/content/w34628427770167g/`

Abstract: We propose a stochastic π-calculus modelling approach able to handle the complexity of post-translational signalling and to overcome some limitations of the ordinary differential equations based methods. The model we developed is customizable without a priori assumptions to every multisite phosphorylation regulation. We applied it to the multisite phosphorylation of the Pho4 transcription factor that plays a crucial role in the phosphate starvation signalling in Saccharomyces cerevisiae, using available in vitro experiments for the model tuning and validation. The in silico simulation of the sub-path with the stochastic π-calculus allows quantitative analyses of the kinetic characteristics of the Pho4 phosphorylation, the different phosphorylation dynamics for each site (possibly combined) and the variation of the kinase activity as the reaction goes to completion. One of the predictions indicates that the Pho80-Pho85 kinase activity on the Pho4 substrate is nearly distributive and not semi-processive as previously found analysing only the phosphoform concentrations in vitro. Thanks to the compositionality property of process algebras, we also developed the whole PHO pathway model that gives new suggestions and confirmations about its general behaviour. The potentialities of process calculi-based in silico simulations for biological systems are highlighted and discussed.

Nicola Segata, Enrico Blanzieri, Corrado Priami. Towards the integration of computational systems biology and high-throughput data: supporting differential analysis of microarray gene expression data. *Journal of Integrative Bioinformatics 5(1):87*, 2008. `http://journal.imbio.de/article.php?aid=87`

Abstract: The paradigmatic shift occurred in biology that led first to high-throughput experimental techniques and later to computational systems biology must be applied also to the analysis paradigm of the relation between local models and data to obtain an effective prediction tool. In this work we introduce a unifying notational framework for systems biology models and high-throughput data in order to allow new integrations on the systemic scale like the use of in silico predictions to support the mining of gene expression datasets. Using the framework, we propose two applications concerning the use of system level models to support the differential analysis of microarray expression data. We tested the potentialities of the approach with a specific microarray experiment on the phosphate system in Saccharomyces cerevisiae and a computational model of the PHO pathway that supports the systems biology concepts.