FAST AND SCALABLE LOCAL KERNEL MACHINES

Nicola Segata and Enrico Blanzieri

December 2009

Technical Report # DISI-09-072

Fast and Scalable Local Kernel Machines

Nicola Segata SEGATA@DISI.UNITN.IT
Enrico Blanzieri BLANZIER@DISI.UNITN.IT
Department of Information Engineering and Computer Science
University of Trento
Trento, Italy

Abstract

A computationally efficient approach to local learning with kernel methods is presented. 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 an appropriate model for each query point is selected at testing time according to a proximity strategy. Supported by a recent result by Zakai and Ritov (2009) relating consistency and localizability, our approach guarantees high generalization ability by dividing 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 dismiss local information in order to improve scalability.

Keywords: Locality, Kernel Methods, Local Learning Algorithms, Support Vector Machines, Memory-Based Learning

1. Introduction

Efficiently processing 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 (Keerthi and DeCoste, 2005; Collins et al., 2008; Chang et al., 2008; Bordes et al., 2009; Fan et al., 2008), and indeed they are effective when the linear separation is a good choice such as in high-dimensionality problems. Other approaches permit the non-linear feature space setting, but they limit the complexity by working with a reduced number of examples or a small set of support vectors (Lee and Mangasarian, 2001), using active and online example selection (Bordes et al., 2005; Bordes and Bottou, 2005) or bounding the number of basis functions (Keerthi et al., 2006; Joachims and Yu, 2009).

In the works referenced 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 high 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 trained on 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 separately considering 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 (2009) that showed how, roughly speaking, "consistency implies local behaviour".

In this work 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 the whole 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 Blanzieri and Melgani (2006) and Section 3.3). Scalability is obtained approximating the Local SVM approach but, differently from the global approaches for fast kernel methods which 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 use the same local SVM model for more than one testing point and we can also 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 (LLA) (Bottou and Vapnik, 1992; Vapnik and Bottou, 1993) 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.

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" (Bengio et al., 2005). On the other hand, there are problems (Blackard and Dean, 1999; Uzilov et al., 2006) 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 (Blackard and Dean, 1999) 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 showed in a very preliminary study (Segata and Blanzieri, 2009c) 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 modifies the Local SVM approach (Blanzieri and Melgani, 2006; Zhang et al., 2006) that showed excellent classification performances but had dramatic computational problems, leading to 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 until 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 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 (Segata, 2009, FaLKM-lib).

In the next Section we analyse the work on LLAs, Local SVM and fast large margin classifiers that are all related with our work. Section 3 formally introduces some machine learning tools that we need in order to introduce FaLK-SVM in Section 4 and analyse its learning bounds, complexity bounds, implementation, local model selection procedure and intuitive interpretation. Section 5 details the empirical evaluation with respect to accurate and approximated approaches.

2. Related Work

Locality is often a crucial component of machine learning systems, although we are not aware of approaches exploiting locality for improving the computational performances. We review in this section these areas that are more related with our approach: local learning algorithms, local support vector machines, approximated and scalable SVM solvers.

2.1 Local Learning Algorithms

Local Learning Algorithms (LLAs) are a class of learning approaches introduced by Bottou and Vapnik (1992). 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 uneven 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 formalized by the Local Risk Minimization principle (Vapnik and Bottou, 1993; Vapnik, 2000). Notice that there are various ways of specifying the degree of locality for LLAs as discussed for instance by Atkeson et al. (1997). Examples of LLAs are the well-known k-Nearest Neighbours (kNN) classifier, the Radial Basis Function networks (Broomhead and Lowe, 1988), and the Local SVM classifier (Blanzieri and Melgani, 2006; Zhang et al., 2006) described in Section 2.2.

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 (1992), 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 increasing the computational performances of the testing phase.

2.2 Local Support Vector Machines

Local SVM is a LLA and was independently proposed by Blanzieri and Melgani (2006, 2008) and by Zhang et al. (2006) and applied respectively to remote sensing and visual recognition tasks. Other successful applications of the approach are detailed by Segata and Blanzieri (2009a) for general real datasets, by Blanzieri and Bryl (2007) for spam filtering and by Segata et al. (2009) for noise reduction. The main idea of local SVM, described in details in Section 3.3, is to build at prediction time an example-specific maximal marginal hyperplane based on the set of K-neighbours. In (Blanzieri and Melgani, 2006) 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, (i) the selection of the K nearest neighbours and (ii) the computation of the maximal separating hyperplane on the K examples. An attempt to computationally improve the Local SVM approach of Zhang et al. (2006) has been proposed by Cheng et al. (2007) where the idea is to train multiple SVMs on clusters found by a variant of k-means, called MagKmeans, that introduces in the clustering criterion the requirement that the clusters cannot have unbalanced class cardinalities. 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 (MagKmeans does not have constraints on the cardinalities and the balancing requirement can cause the detection of clusters with high cardinalities). The achieved computational performances are better than their formulation of Local SVM, but worse than global SVM.

2.3 Fast Large Margin Classifiers

The need for fast and scalable kernel-based classifiers led to the development of several methods in the last few years, although considerable attention seems to have been focused especially on linear SVM classifiers. Below, we initially consider the works applicable also to non-linear kernels, successively we review the works on the linear case.

One of the first large-scale maximal margin learning that can use non-linear kernel functions is represented by Core Vector Machines (Tsang et al., 2005, CVM) 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 that the choice of the stopping criteria is crucial for the trade-off between computational efficiency and generalization accuracy. Ball Vector Machines (Tsang et al., 2007, BVM) 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 has been proposed by Bordes et al. (2005, LASVM) and by Bordes and Bottou (2005) 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. (2006, SpSVM) proposed a method that bounds the number of basis functions considered and thus 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. (2006, USVM) 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* (Yuille and Rangarajan, 2003). Recently the Cutting-Plane Subspace Pursuit (Joachims and Yu, 2009, CPSP) based on cutting-plane training (Joachims et al., 2009) 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 (Chang and Lin, 2001) demonstrated to be competitive with approximated approaches from the computational viewpoint. LibSVM is a SVM solver implementing a SMO-type decomposition method proposed by Fan et al. (2005) integrating it with caching and shrinking (Joachims, 1999).

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 (2005) modified the Finite Newton method of Mangasarian (2002) introducing robust conjugate gradient techniques and other heuristics, Joachims (2006) developed an alternative formulation of the SVM optimization problem exploiting a different form of sparsity and Lin et al. (2007) used logistic regression with Trust Region Newton Methods. Variants of coordinate descent methods for linear SVM are developed by Chang et al. (2008) in the primal and by Hsieh et al. (2008) in the dual. A different gradient approach was developed by Smola et al. (2008). Other approaches are based on Stochastic Gradient Descent (SGD) like those developed by Shalev-Shwartz et al. (2007) and by Bordes et al. (2009) which work in the primal, whereas Collins et al. (2008) 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 (Fan et al., 2008) is a fast software package implementing some of the cited works. The common idea of all the 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 accuracy. Apart from the generalization ability also the computational performances can be compromised in these cases, because the algorithm cannot find a good decision function and so convergence problems can occur.

3. Preliminaries

In order to introduce our approach, we need to analyse the formulation of kNN, SVM, kNNSVM and Cover Trees.

Here and in the following of the paper, we consider a binary class classification 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 an Hilbert space with inner product $\langle \cdot, \cdot \rangle$ and norm $\| \cdot \|$. Extensions to multi-class problems will be explicitly discussed.

3.1 The k Nearest Neighbour Algorithm

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

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

In this way, $\mathbf{x}_{r_{\mathbf{x}'}(j)}$ is the example 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. In other terms:

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

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

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

For problems with more than two classes, the decision rule of kNN is the usual majority rule, namely it selects the class with the highest number of representatives in the k-neighbourhood instead of taking the sign of the summation.

3.2 Support Vector Machines

SVMs (Cortes and Vapnik, 1995) are classifiers with sound foundations in statistical learning theory (Vapnik, 2000). 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:

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

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:

$$y_i \left(\langle w, \Phi(\mathbf{x}_i)\rangle_{\mathcal{F}} + b\right) \geq 1 - \xi_i \qquad \xi_i \geq 0,\ i = 1, \ldots, N. \tag{2}$$

For soft-margin SVM the optimization problem with linear penalisation of ξ_i (L1-norm), becomes the minimisation of $\frac{1}{2} \cdot \|w\|^2 + C \sum_i \xi_i$ subject to (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 $\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x})\rangle_{\mathcal{F}}$ the decision rule can be expressed as:

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

Throughout this work, SVM denotes the soft-margin L1-norm SVM.

The kernel trick avoids the explicit definition of the feature space $\mathcal{F}$ and of the mapping Φ (Schölkopf and Smola, 2002). Popular kernels are the linear (LIN) kernel, the radial

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.

basis function (RBF) kernel, and the homogeneous (HPOL) and inhomogeneous (IPOL) polynomial kernels. Their definitions are:

$$\begin{array}{rclrcl} K^{lin}(\mathbf{x},\mathbf{x}') &=& \langle \mathbf{x},\mathbf{x}'\rangle & K^{rbf}(\mathbf{x},\mathbf{x}') &=& \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{\sigma}\right) \\ K^{hpol}(\mathbf{x},\mathbf{x}') &=& \langle \mathbf{x},\mathbf{x}'\rangle^d & K^{ipol}(\mathbf{x},\mathbf{x}') &=& (\langle \mathbf{x},\mathbf{x}'\rangle+1)^d. \end{array}$$

The maximal separating hyperplane defined by SVM has been shown to have important generalisation properties and nice bounds on the VC dimension (Vapnik, 2000).

Multiple methods has been proposed in order to apply the maximal margin principle of SVM on multiple class problems. The more popular are the one-against-all method (Bottou et al., 1994) which builds a number of binary decision functions equal to the number of classes N_{cl}, the one-against-one method (Knerr et al., 1990; Kressel, 1999) which builds $N_{cl} \cdot (N_{cl}-1)/2$ binary decision functions using voting in the prediction phase, and the Directed Acyclic Graph SVM (Platt et al., 2000, DAGSVM) which is a modification of the one-against-all method. The study carried on by Hsu and Lin (2002) shows that the more effective strategies are the one-against-one and DAGSVM approaches.

3.3 Local SVM: the kNNSVM Classifier

We already introduced the idea of Local SVM in Section 2.2, here we detail kNNSVM which is the formulation of Local SVM proposed by Blanzieri and Melgani (2006, 2008). kNNSVM can be seen as a modification of the SVM approach in order to obtain a LLA able to locally adjust the capacity of the training systems.

In order to classify a given example $\mathbf{x}' \in \mathcal{H}$, we need first to retrieve its k-neighbourhood in the transformed feature space $\mathcal{F}$ and, then, to search for an optimal separating hyperplane only over this k-neighbourhood. In practice, this means that an SVM is built over the neighbourhood in $\mathcal{F}$ of each test example $\mathbf{x}'$. Accordingly, the constraints in (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 $r_{\mathbf{x}'} : \{1,\ldots,N\} \to \{1,\ldots,N\}$ is a function that reorders the indexes of the training examples defined as:

$$\begin{cases} r_{\mathbf{x}'}(1) = \underset{i=1,\ldots,N}{\operatorname{argmin}} \|\Phi(\mathbf{x}_i)-\Phi(\mathbf{x}')\|^2_{\mathcal{F}} \\ r_{\mathbf{x}'}(j) = \underset{i=1,\ldots,N}{\operatorname{argmin}} \|\Phi(\mathbf{x}_i)-\Phi(\mathbf{x}')\|^2_{\mathcal{F}} \qquad i \neq r_{\mathbf{x}'}(1),\ldots,r_{\mathbf{x}'}(j-1) \quad \text{for} \quad j=2,\ldots,N \end{cases} \tag{3}$$

In this way, $\mathbf{x}_{r_{\mathbf{x}'}(j)}$ is the example in the j-th position in terms of distance from $\mathbf{x}'$ and thus $j<k \Rightarrow \|\Phi(\mathbf{x}_{r_{\mathbf{x}'}(j)})-\Phi(\mathbf{x}')\|_{\mathcal{F}} \leq \|\Phi(\mathbf{x}_{r_{\mathbf{x}'}(k)})-\Phi(\mathbf{x}')\|_{\mathcal{F}}$ because of the monotonicity of the quadratic operator. The computation is expressed in terms of kernels as:

$$\begin{aligned} ||\Phi(\mathbf{x})-\Phi(\mathbf{x}')||^2_{\mathcal{F}} &= \langle\Phi(\mathbf{x}),\Phi(\mathbf{x})\rangle_{\mathcal{F}}+\langle\Phi(\mathbf{x}'),\Phi(\mathbf{x}')\rangle_{\mathcal{F}}-2\cdot\langle\Phi(\mathbf{x}),\Phi(\mathbf{x}')\rangle_{\mathcal{F}} = && (4)\\ &= K(\mathbf{x},\mathbf{x})+K(\mathbf{x}',\mathbf{x}')-2\cdot K(\mathbf{x},\mathbf{x}'). && (5) \end{aligned}$$

If the kernel is the RBF kernel or any polynomial kernels with degree 1, the ordering function is equivalent to the one defined by the Euclidean metric. In general, 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 $\mathbf{x}$ is:

$$k\text{NNSVM}(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{k} \alpha_{r_{\mathbf{x}}(i)} y_{r_{\mathbf{x}}(i)} K(\mathbf{x}_{r_{\mathbf{x}}(i)}, \mathbf{x}) + b\right). \tag{6}$$

For $k = N$, the kNNSVM method is the usual SVM whereas, for $k = 2$, the method implemented with the LIN or RBF kernel corresponds to the standard 1-NN classifier. 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 similarly to the behaviour of the majority rule.

Considering kNNSVM as a local SVM classifier built in the feature space, the method has been shown to have a potentially favourable bound on the expectation of the probability of test error with respect to SVM (Blanzieri and Melgani, 2008).

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 (Segata and Blanzieri, 2009a) 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 the same strategy on SVM, but no specific empirical studies have been performed yet to identify the most appropriate strategy for multi-class classification with Local SVM.

3.4 Cover Trees

A Cover Tree is a data structure introduced by Beygelzimer et al. (2006) for performing exact nearest-neighbour operations in a fast and efficient way. Cover Trees can be applied in general metric spaces without any other assumption on their 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 (Krauthgamer and Lee, 2004b) 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 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. The covering tree invariant implies that every node has a parent in a higher level such that the distance between the respective points is less than b^i, while separation invariant 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 (called C_∞ and containing only one example) is a randomly chosen example.

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. (2006), 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 $\mathcal{O}(n \log n)$.

4. FaLK-SVM: a Fast and Scalable Local Kernel Machine

In this section we introduce our novel technique. Initially we detail the way to precompute the local models during training (Section 4.1) and the strategies to reduce the number of local models (Section 4.2). We then describe the prediction mechanism in Section 4.2.2 and our approach for fast local model selection in Section 4.3. Successively, we derive learning bounds for the approach in Section 4.4 before discussing the computational complexity in Section 4.5 and some details about the implementation (Section 4.6).

4.1 Precomputing the Local Models during Training Phase

For the local approach we are proposing here, we need to generalize the decision rule of kNNSVM to the case in which the local model is trained on 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:

$$k\text{NNSVM}_{\mathbf{t}}(\mathbf{q}) = \text{sign}\left(\sum_{i=1}^{k} \alpha_{r_{\mathbf{t}}(i)} y_{r_{\mathbf{t}}(i)} K(\mathbf{x}_{r_{\mathbf{t}}(i)}, \mathbf{q}) + b\right) \tag{7}$$

where $r_{\mathbf{t}}(i)$ is the kNNSVM ordering function (see above Section 3.3) 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:

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

The original decision function of kNNSVM corresponds to the case in which $\mathbf{t} = \mathbf{q}$, 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 computationally feasible only for problems with few points to test which is a condition that rarely holds in real-world classification problems. In general, 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:

$$k\text{NNSVM}_{\mathbf{t}}(\mathbf{q}) \text{ with } \mathbf{t} = \mathbf{x}_{r_{\mathbf{q}}(1)} \tag{8}$$

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* (Aha, 1997) setting of the original formulation of kNNSVM to the *eager learning* setting with clear advantages in terms of prediction step complexity. This is possible computing a local SVM model for each $\mathbf{x} \in \mathcal{X}$ during the training phase obtaining the following sets $\{(\mathbf{t}, k\text{NNSVM}_{\mathbf{t}}) \mid \mathbf{t} \in \mathcal{X}\}$ and applying the precomputed $k\text{NNSVM}_{\mathbf{t}}$ model such that $\mathbf{t} = \mathbf{x}_{r_{\mathbf{q}}(1)}$ for each query point $\mathbf{q}$ during the testing phase.

This approximation slightly modifies the approach of kNNSVM as a local learning algorithm. 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 estimate 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 prediction of a test point $\mathbf{q}$ is done using the model built for the Voronoi region in which $\mathbf{q}$ lies ($V_{\mathbf{h}}$ with $\mathbf{h} = r_{\mathbf{q}}(1)$) that can be retrieved by searching for the nearest neighbour of $\mathbf{q}$ in $\mathcal{X}$.

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

The pre-computation of the local models during the training phase introduced above, increases the computational efficiency of the prediction step. However, 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_h with $h = r_{\mathbf{x}'(1)}$). The decision function of this approach is

$$\mathsf{FastLSVM}(\mathbf{x}) = k\text{NNSVM}_{f(\mathbf{x})}(\mathbf{x}) \tag{9}$$

where $f : \mathcal{H} \mapsto \mathcal{C} \subseteq \mathcal{X}$ is a function mapping each unseen example $\mathbf{x}$ to a unique training example $f(\mathbf{x})$ which is, accordingly to Eq. 7, the center of the local model that is used to evaluate $\mathbf{x}$. The set $\mathcal{C}$ is the image of $f(\cdot)$, so $\mathcal{C} = f(\mathcal{H})$.

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. 8, 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 $\mathcal{C}$.

4.2.1 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 and $k' \leq k$. 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 related to the *Set Cover Problem* (Garey and Johnson, 1979; Kearns and Vazirani, 1994; Marchand and Shawe-Taylor, 2003) known as the *Minimum Sphere Set Covering Problem* (MSSC) (Chen, 2005) although in its original formulation one specifies the radius of the spheres rather than their cardinality in terms of points they contain and it is not required that the centers of the hyperspheres correspond to points in the set. It is easy to show that MSSC is NP-hard but some efficient approximated results are available based on greedy approaches (Chvatal, 1979; Wang et al., 2006), integer and linear programming (Wei and Li, 2008).

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.

The heuristic procedure we developed can be seen as a modification of the greedy approach for the MSSC problem (Chvatal, 1979; Wang et al., 2006). The first k'-neighbourhood is selected randomly choosing its center in $\mathcal{X}$, the following k'-neighbourhoods are retrieved selecting the centers that are still not members of other k'-neighbourhoods and are as far as possible from the already selected centers. The selection of the farthest example, still not included in the k'-neighbourhoods, as the center of the next k'-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 between the elements of 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 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.

Let us define precisely the system of sets $\{S_i\}$. 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:

$$\begin{cases} S_{top} &= \{\text{choose}(\mathcal{X})\} \\ S_i &= S_{i+1} \cup \underset{S \in \mathcal{X} \setminus S_{i+1}}{\text{argmax}} \left(|S| \text{ s.t. } d(S_{i+1} \cup S) > b^i\right) \qquad \text{for } i = top-1, \ldots, bot \end{cases}, \tag{10}$$

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

$$S_{top} = \{\text{choose}(\mathcal{X})\} \subseteq S_{top-1} \subseteq \ldots \subseteq S_{bot+1} \subseteq S_{bot} = \mathcal{X} \tag{11}$$

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:

$$\begin{cases} \mathbf{c}_1 &= \text{choose}(S_{top}) \\ \mathbf{c}_j &= \text{choose}(S_l) \text{ with } l = \max(m \in \mathbb{N} | S_m \setminus \mathcal{X}_{\mathbf{c}_{j-1}} \neq \emptyset) \end{cases}, \tag{12}$$

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 k'-neighbourhoods of the centers already included in $\mathcal{C}$.

We can briefly show that the $\mathcal{C}$ set found with Eq. 12 is a k'-neighbourhood covering set of centers. In fact, 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 set S_{bot} we always have that $S_{bot} = \mathcal{X}$, this 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. 12 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 share some characteristics with the levels of Cover Trees. First from Eq. 10 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 3.4. Second, always from Eq. 10 we can conclude that each S_j is contained in every S_t set with $t < j$ as also explicated in Eq.11; 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. 10. 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. 12.

Consequently with the goal of reducing the number of local models, this approach no longer requires 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 1 graphically shows the result of adopting the approach described above on a simple artificial dataset with k and k' chosen for illustrative purposes. In fact, the example just aims to show the intuition behind the approach that is instead developed for large datasets and for non-extreme values of the neighbourhood parameters.

From Figure 1 we can also notice that the level of overlapping between k'-neighbourhoods and thus between k-neighbourhoods depends on the value of k'. If k' is low, a large number of k'-neighbourhoods are required to cover the entire training set, whereas if k' is large fewer k'-neighbourhoods are needed. The k' parameter thus tune the level of redundancy of the local models.

4.2.2 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. 9 with $f(x) = r^{\mathcal{C}}_{\mathbf{x}}(1)$ where $r^{\mathcal{C}}$ corresponds to the reordering function defined in Eq. 3 performed on the $\mathcal{C}$ set instead of $\mathcal{X}$, the method, called FaLK-SVMc, is defined as:

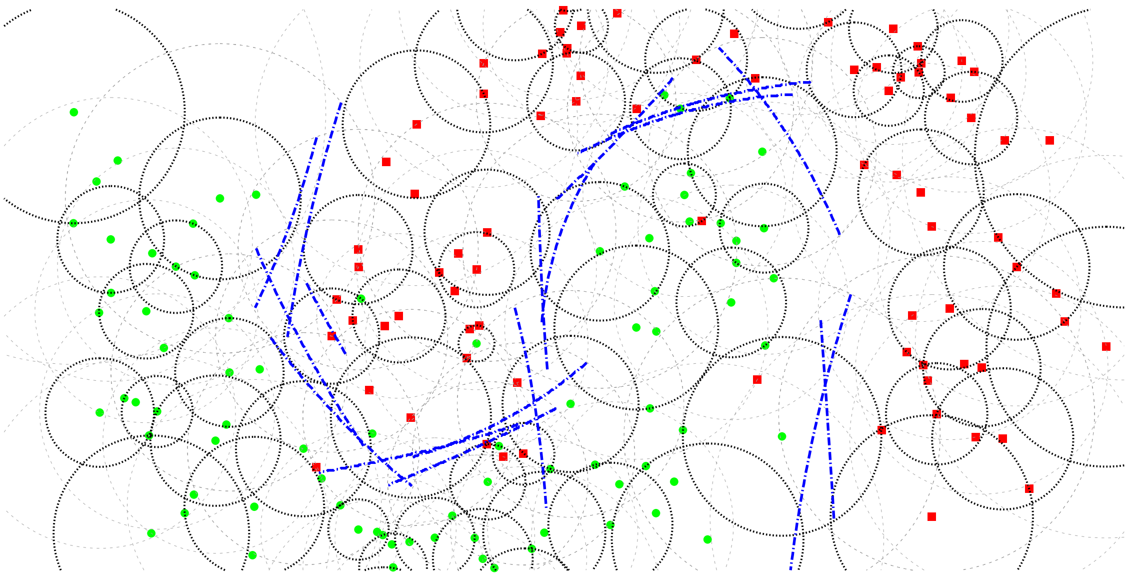


Figure 1: Graphical representation of the proposed approach using local models with $k' = 4$, $k = 15$, and local SVM with RBF kernel. The bold dotted circles highlights the k'-neighbourhoods covering all the training set (with some unavoidable redundancy), the thin dotted circles denotes the k-neighbourhoods on which the local models are trained. Some k-neighbourhoods do not produce an explicit decision function because entirely composed by points of the same class. 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.

$$\mathsf{FaLK\text{-}SVMc}(\mathbf{x}) = k\mathrm{NNSVM}_{\mathbf{c}}(\mathbf{x}) \text{ where } \mathbf{c} = \mathbf{x}_{r^{\mathcal{C}}_{\mathbf{x}}(1)} \tag{13}$$

FaLK-SVMc is satisfactory from the computational viewpoint, for it performs the nearest neighbour search on $\mathcal{C}$ only. However, it does not assure that the testing point is evaluated with the model centered in the point for which the testing point itself is the nearest in terms of neighbour ranking. 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 $\mathbf{q}$ is the i-th nearest neighbour of $\mathbf{c}_1$ in $\mathcal{X}$ and the j-th nearest neighbour of $\mathbf{c}_2$ with $i > j$. This is a problem because using the model centered in $\mathbf{c}_2$ is better in terms of proximity. In order to overcome this issue 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 neighborhood ranking to its training

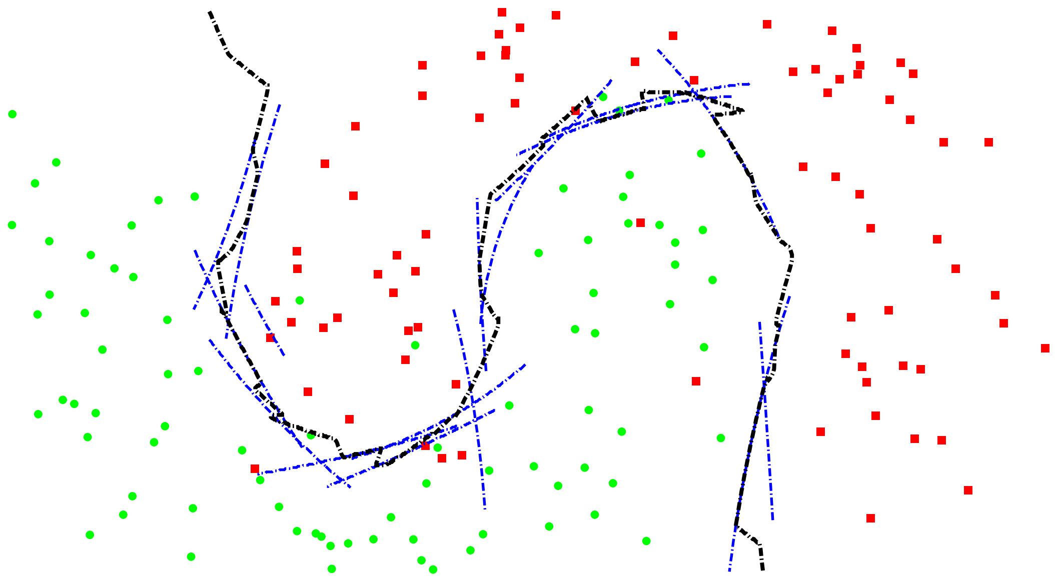


Figure 2: Graphical representation of the global decision function (black dotted line) obtained with the local decision functions (the same of Figure 1) using the described approach that uses for each query point the local decision function of the Voronoi region in which it lies.

nearest neighbour. We can do this defining a function $cnt : \mathcal{X} \mapsto \mathcal{C}$ in the following way:

$$\begin{aligned} &cnt(\mathbf{x}_i) = \text{choose}(\{\mathbf{c}_z \in \mathcal{C} | \mathbf{x}_i = \mathbf{x}_{r_{\mathbf{c}_z}(h)}\}) \\ &\text{where } h = \min\left(t \in \{1, \dots, k'\} \middle| \mathbf{x}_{r_{\mathbf{c}_j}(t)} = \mathbf{x}_i \text{ and } \mathbf{c}_j \in \mathcal{C}\right). \end{aligned} \tag{14}$$

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 selects the center with the minimum index. The existence of h is guaranteed by the k'-neighbourhood covering strategy. 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. 9 with $f(\mathbf{x}) = cnt(\mathbf{x})$, and called FaLK-SVM, is simply:

$$\text{FaLK-SVM}(\mathbf{x}) = k\text{NNSVM}_{cnt(\mathbf{t})}(\mathbf{x}) \text{ where } \mathbf{t} = \mathbf{x}_{r_{\mathbf{x}}(1)} \tag{15}$$

The association between training points and centers defined by Eq. 14 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.

Figure 2 graphically shows the application of the FaLK-SVM($\mathbf{x}$) prediction strategy on a toy dataset; the training phase for the same dataset is illustrated in Figure 1.

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

4.3 FaLK-SVM with Internal Model Selection: FaLK-SVMl

For training a kernel machine, once a proper kernel is chosen, it is crucial 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 values and a popular method is the κ-fold cross-validation[2] 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 training set $\mathcal{X}$ in κ subsets each with the same cardinality[3] (called folds). The κ-fold cross validation (CV) procedure consists in searching for the parameters that minimise the average of the losses on $\mathcal{X}_f$ of the classifier trained on $\mathcal{X} \setminus \mathcal{X}_f$ for $f = 1, \ldots, \kappa$. The effectiveness in terms of testing accuracies of κ-fold CV is high, but it adds a computational overhead to the training phase. In fact, the computational complexity of a κ-fold 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.

The model selection for FaLK-SVM and FaLK-SVMc can be performed using κ-fold CV. The only difference with SVM is that our local kernel machines need to estimate an additional parameter which is the neighborhood size k (which is however usually chosen in a small set of possible values). However, with the local setting of the classification problem we are discussing in this paper, it is also possible to efficiently tackle the complexity of the model selection phase. Basically, since FaLK-SVM trains a set of local models, we can perform the model selection in a grid-search setting on a subset of the neighbourhoods. In this way we can efficiently estimate the global parameters of FaLK-SVM without considering all the training points during model selection. The classifier implementing this approach to model selection is called FaLK-SVMl.

As a first step for defining the model selection approach of FaLK-SVMl, we define a different setting of model selection for kNNSVM.

Definition 3 (Localised 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' most internal points, taking as each corresponding training fold the union of the remaining k'-neighbourhood points and of the $k - k'$ most external points of the k-neighbourhood.

Definition 4 (k'-internal κ-fold CV model selection for kNNSVM)
The procedure applies the localised κ-fold CV model selection on the k'-neighbourhood of the

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

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 k'-internal κ-fold CV for kNNSVM model selection on a randomly chosen training example and use the resulting parameters for all the local models. In order to be robust the procedure is repeated on more than one k-neighbourhood choosing the parameters that minimize the average k'-internal κ-fold CV error among the k-neighbourhoods.

Definition 5 (k'-internal κ-fold CV model selection for FaLK-SVM)
The procedure applies the 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 k'-internal κ-fold CV described in Definition 5 is named FaLK-SVMl.

A specific strategy for setting the RBF kernel width. As already proposed by Tsang et al. (2005) and by Segata and Blanzieri (2009b), good choices for the RBF kernel width σ of SVM are based on the median (or other percentiles) of the distribution of distances. In FaLK-SVMl we can thus efficiently estimate σ for each local model simply calculating the median of the distances in the neighbourhood. This approach has some analogies with standard SVM using a variable RBF kernel width that have good potentialities for classification (Chang et al., 2005). Since other percentiles different from the median can give better accuracy performances, in FaLK-SVMl the percentile can be a value to set using the k'-internal κ-fold CV approach.

4.4 Generalization Bounds for kNNSVM and FaLK-SVM

The class of LLAs introduced by Bottou and Vapnik (1992) includes kNNSVM, and can be theoretically analysed using the framework based on the local risk minimization (Vapnik and Bottou, 1993; Vapnik, 2000). On the other hand, FaLK-SVM is not a LLA as intended by Bottou and Vapnik (1992). In fact, LLAs compute the local function for each specific testing point thus delaying the neighbourhood retrieval and model training until the testing point is available. However, we show here that generalization bounds for FaLK-SVM can be derived starting from the LLA ones.

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

Theorem 6 (Vapnik (2000)) *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:*

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

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, the loss function is simply

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

and the locality function is

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

It is straightforward to show that $\sum_{i=1}^{N} T(\mathbf{x}_i,\mathbf{x}',k) = k$. Moreover $T(\mathbf{x}_i,\mathbf{x}',k)$ has VC dimension equal to 2; it is, in fact, the class of functions corresponding to 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 classes, but cannot shatter three points with the nearest and furthest points having a class different from the third point.

We observe 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:

$$R^{k\text{NNSVM}}(\alpha,k,\mathbf{x}') \le \frac{\frac{1}{N}k\cdot\nu_{\mathbf{x}'} + \gamma(N,h^{\Sigma})}{|\frac{1}{N}k - \gamma(N,2)|} \tag{16}$$

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

The possibility of local approaches to obtain a lower bound on test misclassification probability acting with the locality parameter, as stated in (Vapnik and Bottou, 1993; Vapnik, 2000) for LLA, it is even more evident for kNNSVM considering Eq. 16. In fact, although choosing a $k < N$ is not sufficient to lower the bound, as the model training becomes more and more local k decreases and (very likely) the misclassification training rate $\nu_{\mathbf{x}'}$ decreases 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.

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 7 *Given a distribution* $g(\mathbf{x})$*, if a set* $\mathcal{X}$ *with* $|\mathcal{X}| = N$ *and a point* $\mathbf{x}$ *are i.i.d. drawn from* g*, the expectation for the 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:*

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

Proof Consider the following function returning 1 if the query point lies in the i-th Voronoi cell $V_{\mathbf{x}_i}$ defined by the N points in the set $\mathcal{X}$:

$$v_i^N(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{x} \in V_{\mathbf{x}_i} \\ 0, & \text{otherwise.} \end{cases}$$

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:

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

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

$$\begin{aligned} E_{\mathcal{X},\mathbf{x}}[v_i^N(\mathbf{x})] &= \int_{\mathcal{X}}\int_{\mathbf{x}} v_i^N(\mathbf{x}) \cdot g(\mathbf{x}_1) \cdot g(\mathbf{x}_2) \cdot \ldots \cdot g(\mathbf{x}_{N-1}) \cdot g(\mathbf{x}_N) g(\mathbf{x})\, d\mathcal{X}\, d\mathbf{x} && (17) \\ &= \int_{\mathcal{X}} g(\mathbf{x}_1) \cdot g(\mathbf{x}_2) \cdot \ldots \cdot g(\mathbf{x}_{N-1}) \cdot g(\mathbf{x}_N) \int_{\mathbf{x}} v_i^N(\mathbf{x}) \cdot g(\mathbf{x})\, d\mathcal{X}\, d\mathbf{x} && (18) \\ &= \int_{\mathcal{X}} g(\mathbf{x}_1) \cdot g(\mathbf{x}_2) \cdot \ldots \cdot g(\mathbf{x}_{N-1}) \cdot g(\mathbf{x}_N)\, d\mathcal{X} \int_{\mathbf{x}} v_i^N(\mathbf{x}) \cdot g(\mathbf{x})\, d\mathbf{x} && (19) \\ &= \int_{\mathbf{x}} v_i^N(\mathbf{x}) \cdot g(\mathbf{x})\, d\mathbf{x} && (20) \\ &= E_{\mathbf{x}}[v_i^N(\mathbf{x})]. && (21) \end{aligned}$$

Since the expectation for a test point of lying in a Voronoi cell $E_{\mathbf{x}}[v_i^N(\mathbf{x})]$ is independent from the random sampling of the points in $\mathcal{X}$, it must be the same for each Voronoi cell, so $E_{\mathbf{x}}[v_1^N(\mathbf{x})] = E_{\mathbf{x}}[v_2^N(\mathbf{x})] = \ldots = E_{\mathbf{x}}[v_{N-1}^N(\mathbf{x})] = E_{\mathbf{x}}[v_N^N(\mathbf{x})]$ and, since $\sum_{i=1}^{N} E_{\mathbf{x}}\left[v_i^N(\mathbf{x})\right] = E_{\mathbf{x}}\left[\sum_{i=1}^{N} v_i^N(\mathbf{x})\right] = E_{\mathbf{x}}[1] = 1$, the hypothesis follows directly. ■

FaLK-SVM precomputes local models to be used for testing points lying in sub-regions (k-NN 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:

$$R^{\text{FaLK-SVM}}(\alpha, k, \mathbf{x}') \le R^{k\text{NNSVM}}(\alpha, k, r_{\mathbf{x}'}(1)) + \lambda_i = R^{k\text{NNSVM}}(\alpha, k, \mathbf{x}_i) + \lambda_i \qquad (22)$$

where λ_i is due to the approximation introduced, for the prediction of the label of the query point $\mathbf{x}$', by the use of the k-neighbourhood of $r_{\mathbf{x}'}(1)$ instead of the k-neighbourhood 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 using k' values higher than 1.

From Eq. 22, and using Lemma 7, 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:

$$
\begin{aligned}
R^{\textsf{FaLK-SVM}}(\alpha, k) &= \sum_{i=1}^{N} R^{\textsf{FaLK-SVM}}(\alpha, k, \mathbf{x}') \cdot E_{\mathcal{X}}[P(\mathbf{x}' \in V_{\mathbf{x}_i})] && (23)\\
&= \frac{1}{N}\sum_{i=1}^{N} R^{\textsf{FaLK-SVM}}(\alpha, k, \mathbf{x}') && (24)\\
&= \frac{1}{N}\sum_{i=1}^{N} (R^{k\text{NNSVM}}(\alpha, k, \mathbf{x}_i) + \lambda_i) && (25)\\
&\leq \frac{1}{N}\sum_{i=1}^{N} \frac{\frac{1}{N}k \cdot \nu_{\mathbf{x}_i} + \gamma(N, h^{\Sigma})}{|\frac{1}{N}k - \gamma(N,2)|} + \overline{\lambda} && (26)\\
&= \frac{\frac{k}{N}\sum_{i=1}^{N} \nu_{\mathbf{x}_i} + \gamma(N, h^{\Sigma})}{|k - N\gamma(N,2)|} + \overline{\lambda} && (27)
\end{aligned}
$$

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.

4.5 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 $|\mathcal{C}| = N$, 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, FaLK-SVM can be very easily parallelized differently from SVM whose parallelization, although possible (Zanni et al., 2006; Dong, 2005), 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 $\mathcal{O}(N \cdot \max(k \log N, k^3/N_{proc}))$ where N_{proc} is the number of processors.

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 $\mathcal{O}(N^2)$. For large datasets, FaLK-SVM can still maintain in memory the entire local kernel matrix (if k is not too large), whereas SVM must discard 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 (Clarkson, 1997; Krauthgamer and Lee, 2004a) 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[4] 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 (Bengio et al., 2005), 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).

4.6 Implementation and Availability

FaLK-SVM (and also FkNN and FkNNSVM that are the implementations of kNN and kNNSVM using Cover Trees) is available as part of the Fast Local Kernel Machine Library (Segata, 2009, FaLKM-lib). FaLK-SVM is written in C/C++ and it uses LibSVM

4. 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 1 FaLK-SVM-train (training set **x**[], training size **n**, neighbourhood size **k**, assignment neighbourhood size **k'**)

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

v. 2.88 (Chang and Lin, 2001) for local SVM training and testing whereas we use our own implementation of the Cover Trees data-structure. The pseudo-code for the training phase is reported in Algorithm 1 and for the testing phase in Algorithm 2 (use of Cover Trees and minimization of t in Eq. 14 are omitted for clearness).

Algorithm 2 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 $x$ //retrieve the nearest training point with respect to $q$...
2: Set $nnIndex$ = get index of $p$ //...retrieve its index ...
3: return $label$ = SVMpredict $q$ with $modelPtrs[nnIndex]$ //...and use the corresponding model
   for predict the label of the query point.
```

5. Empirical Analysis

The empirical analysis is organized into three experiments performed with different objectives and using different datasets. Experiment 1 (Section 5.1) 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. Experiment 2 (Section 5.2) 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 with training set cardinalities ranging from about 50k examples to more than 1 million. Experiment 3 (Section 5.3) aims to understand (i) wether 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, unless otherwise specified, are carried out on an AMD Athlon 64 X2 Dual Core Processor 5000+, 2600MHz, with 3.56Gb of RAM with Linux operating system.

5.1 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.

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 1: The 25 binary-class datasets of Experiment 1.

5.1.1 Experimental protocol

The datasets are listed in Table 1; they are retrieved from the UCI (Asuncion and Newman, 2007) and STATLOG (Michie et al., 1994) 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[5], while the k' parameter of FaLK-SVM is fixed to $k/2$ which is a value that privileges scalability over accuracy because we want to test a value that can permit good computational results for large and very large datasets.

5.1.2 Results and discussion

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 2: 10-fold CV accuracy results for the 25 dataset of Experiment 1. The best results for each dataset are highlighted in bold (taking into account all decimal values).

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

Table 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 (Wilcoxon, 1945; Demšar, 2006) 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 (Friedman, 1940) ($\alpha = 0.05$), the null hypothesis is violated, but, according to the Nemenyi post-hoc test (Nemenyi, 1963) ($\alpha = 0.05$) the only method that is statistically significantly 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 (Segata and Blanzieri, 2009a): 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 (Segata and Blanzieri, 2009c). These results also indicates that the $\overline{\lambda}$ constant introduced in the risk of FaLK-SVM (Section 4.4), 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 from large datasets in which it can be crucial.

5.2 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 preliminary experiments showed that the 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.

5.2.1 Experimental protocol

The datasets considered in this experiment are listed in Table 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 one million, whereas the dimensionality is not

dataset name	# of feat.	train. points	testing points	class balancing	original source
ijcnn1	22	49990	91701	90%/10%	LibSVM rep. (Chang and Lin, 2001)
cov-type *	54	100000	481010	51%/49%	LibSVM rep. (Chang and Lin, 2001)
census-inc	41	199523	99762	94%/6%	UCI rep. (Asuncion and Newman, 2007)
cod-rna	8	364651	121549	67%/33%	(Uzilov et al., 2006)
intr-det	40	1026588	311029	79%/21%	UCI KDD rep. (Hettich and Bay, 1999)
2-spirals *	2	100000	100000	50%/50%	Synthetic (Segata and Blanzieri, 2009c)
ndcc *	5	100000	100000	61%/39%	Synthetic (Thompson, 2006)
checker-b *	2	300000	100000	50%/50%	Synthetic (e.g. see Tsang et al., 2005)

Table 3: The 8 large datasets of the second empirical experiment. The datasets whose extensions are used also in Experiment 3 are denoted with *.

high (always under 60) with separated test 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.

5.2.2 Results and discussion

Table 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 (Wilcoxon, 1945; Demšar, 2006) 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 similar to the generalization accuracies meaning that all techniques avoid over-fitting.

Table 5 reports the training times 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

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 4: Empirical (using 10-fold CV) and generalization accuracies of FkNN, LibSVM, FaLK-SVM, FaLK-SVMc and FaLK-SVMl on the 8 large datasets of Experiment 2. 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.

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 5: Training times for Experiment 2 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.

they are at least one order of magnitude bigger than LibSVM. Generally, FaLK-SVMc turns out to be 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. In fact, 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. 22 and Eq. 23). 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.

dataset	LibSVM testing time (s)	FaLK-SVM testing time (s)	FaLK-SVM speed-up on LibSVM	FaLK-SVMc testing time (s)	FaLK-SVMc speed-up on LibSVM	FaLK-SVMl testing time (s)	FaLK-SVMl 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: Testing times for Experiment 2 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.

The testing times required by the analysed methods are reported in Table 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 the problem of model selection with a specific approach to set the parameters; FaLK-SVMl showed to outperform LibSVM in generalization accuracy, and the time it needs for both internal model selection and training is at least comparable (faster in 7 cases on a total of 8) with the time LibSVM needs for the training only.

5.3 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 Section 2.3. 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 in Section 5.3.2, the comparison of FaLK-SVM with its variants FaLK-SVMc and FaLK-SVMl in Section 5.3.3.

5.3.1 Experimental protocol

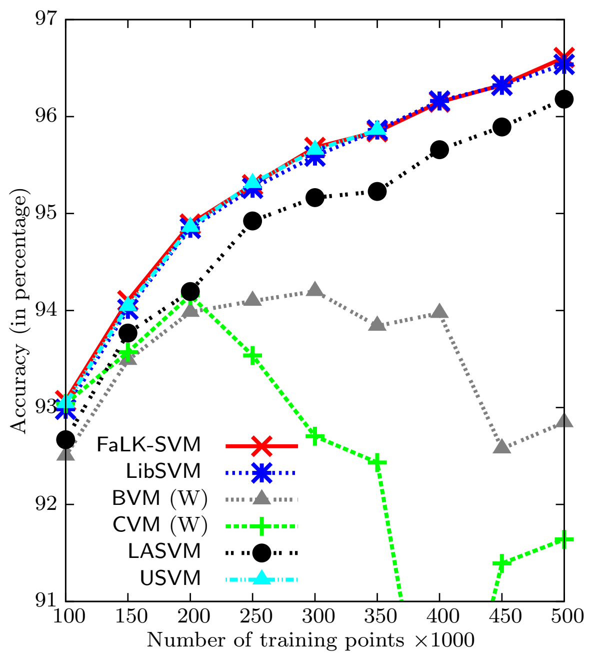
We consider here the datasets of Table 3 for which we can further enlarge the training set size. This is possible for four datasets: the cov-type dataset (full training set of 500k points) and the three artificial datasets named 2-spirals, ndcc and checker-b (up to 3 million points). For cov-type the 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 of ϵ 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.

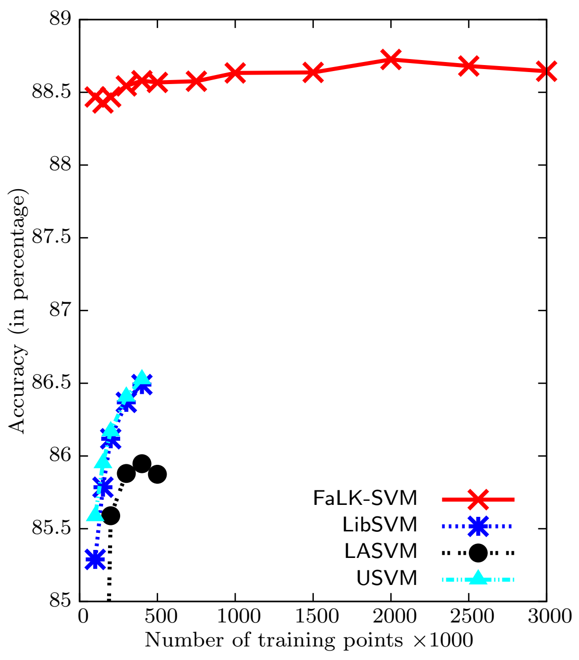
Since the authors of BVM (Tsang et al., 2005) and CVM (Tsang et al., 2007) declared the Linux implementation of their techniques deprecated (see the authors reply to Loosli and Canu (2007) available on BVM webpage), we use the Windows executables on a Intel Pentium D Dual Core CPU 3.40GHz with 2Gb of 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, their running times should not be directly compared to the others. However, the comparison is justified by preliminary tests that showed that the Linux version of BVM on the AMD Athlon machine and the Windows version of BVM on the Intel Pentium machine have similar running times.

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

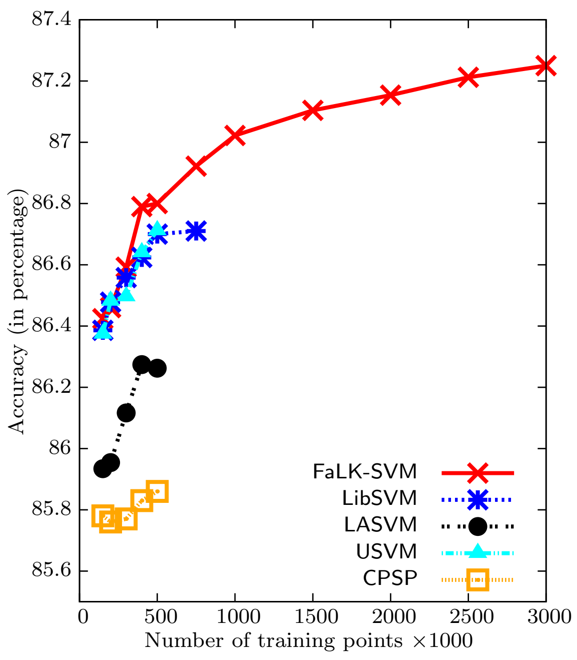
Figure 3 shows the generalization accuracies of the methods at increasing training set sizes. Some methods do not appear in the figures due to low generalization results or computational difficulties that cause abnormal terminations of the algorithms, and some accuracy results for large training set sizes are not present due to the 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.



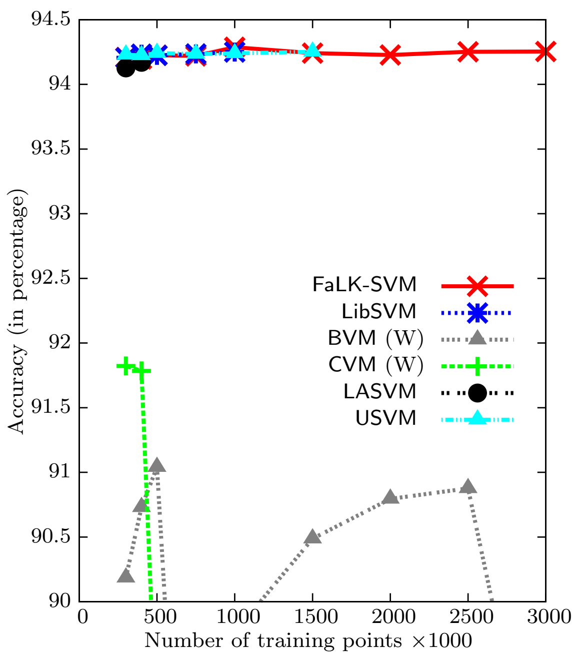
(a) cov-type dataset



(b) 2-spirals dataset



(c) ndcc dataset



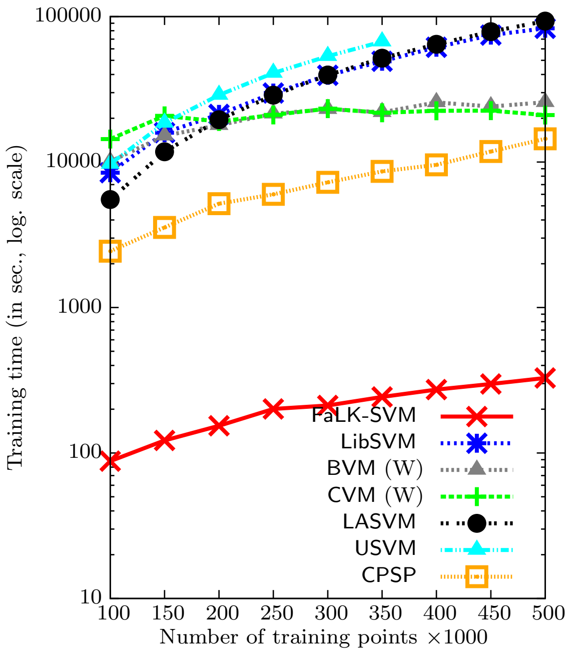
(d) checker-b dataset

Figure 3: Generalization accuracies 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 (Experiment 3). 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.

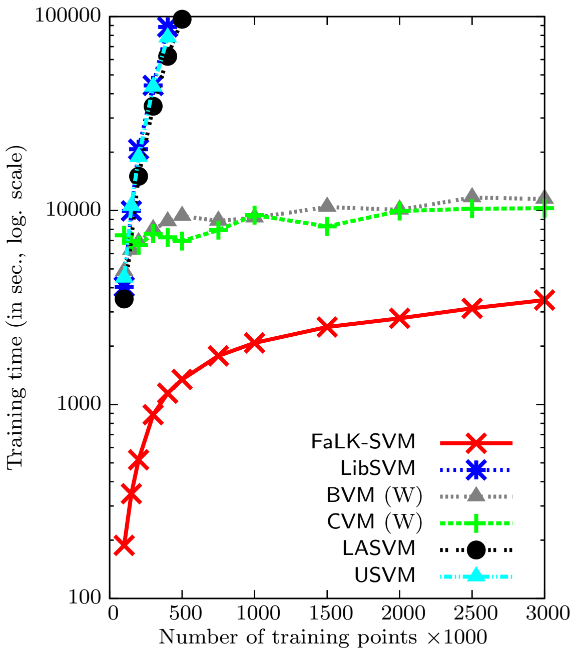
Comparing the generalization accuracies of Figure 3 among the tested methods, we can see that FaLK-SVM is almost always on top for each of the four datasets. The methods that seem to give results comparable with FaLK-SVM (apart from the 2-spirals dataset) are LibSVM and USVM and they are able, in few 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 possibly 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 Bordes et al. (2005) (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 3, we observed 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 4 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 3 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 performance; 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 processes 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 Bordes et al. (2005) 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 only available implementation is the original one by Bordes et al. (2005) whereas LibSVM is frequently updated and improved. Finally, CPSP performs slightly better than LibSVM, LASVM and USVM.

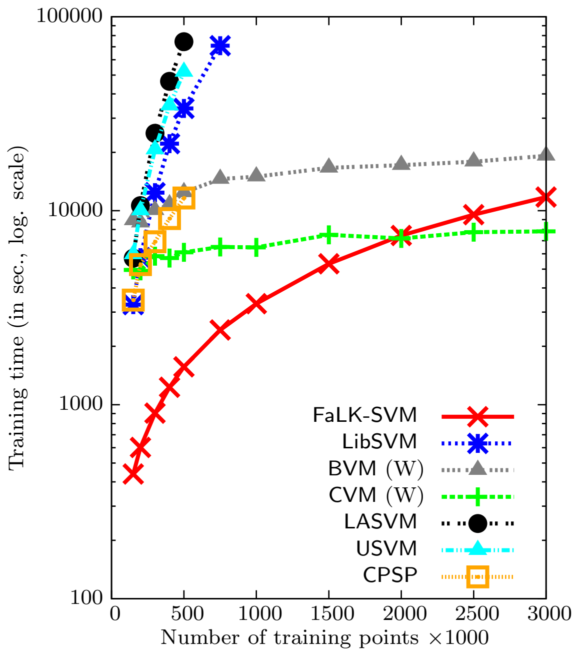
The computational performances of the prediction phase are reported in Figure 5. Also in this case the performance of FaLK-SVM is excellent: only CPSP and CVM are faster in 2 datasets than FaLK-SVM, but their corresponding generalization accuracies are low. As expected, CPSP achieves very fast predictions 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



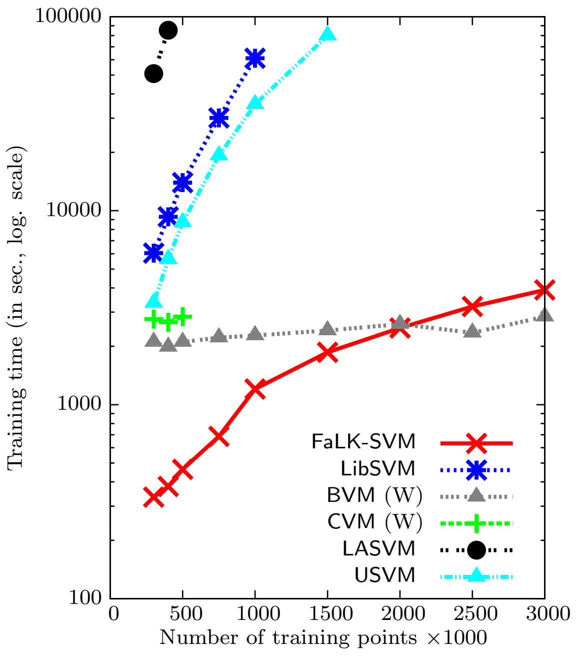
(a) cov-type dataset



(b) 2-spirals dataset

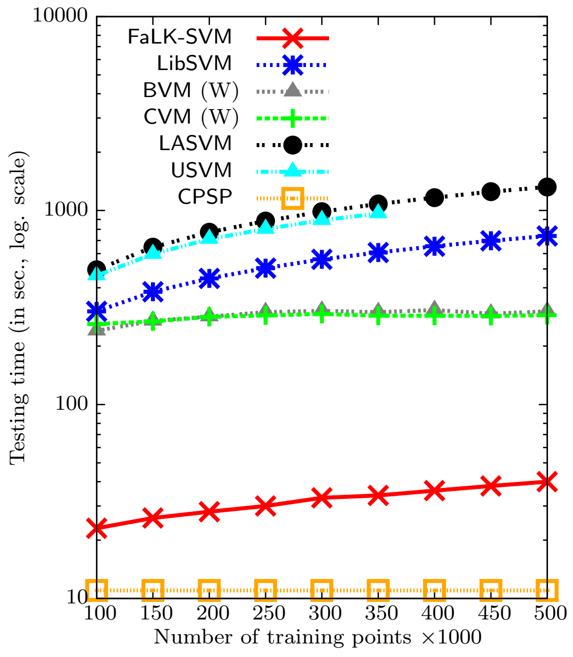


(c) ndcc dataset

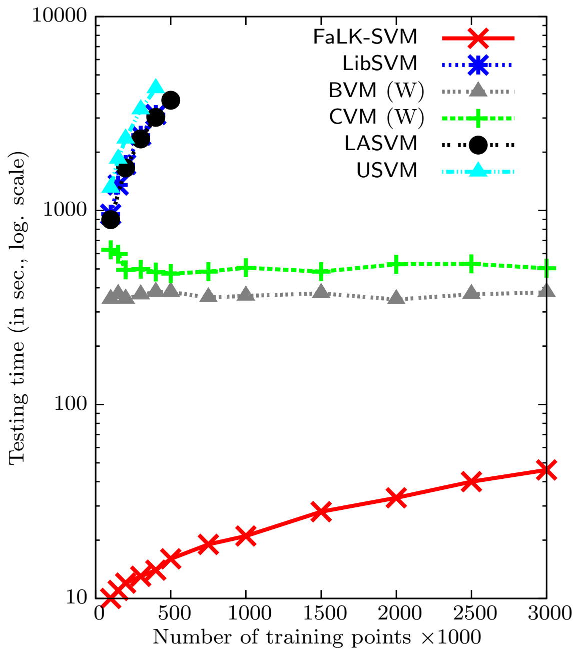


(d) checker-b dataset

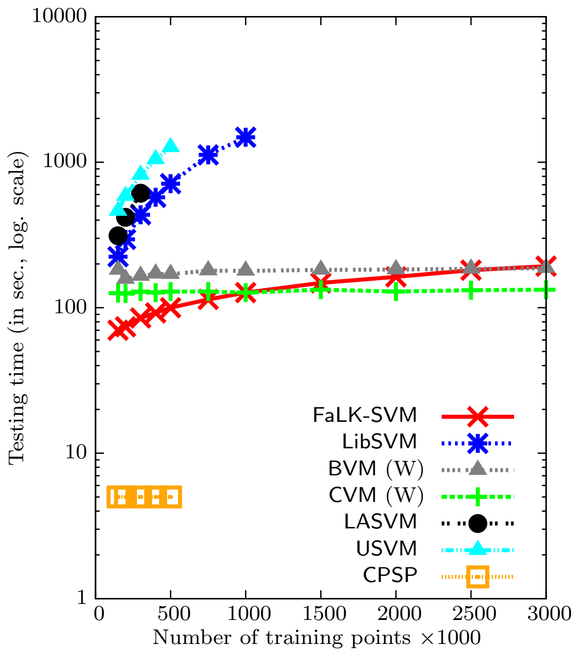
Figure 4: Training times of FaLK-SVM, LibSVM, BVM (in Windows), CVM (in Windows), LASVM, USVM and CPSP on the cov-type, 2-spirals, ndcc and checker-b datasets with increasing training set sizes (Experiment 3). The times (in seconds) are reported in logarithmic scale.



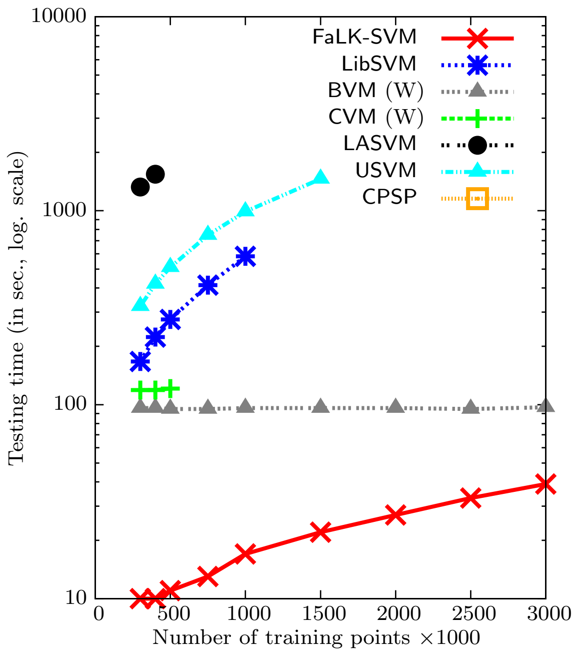
(a) cov-type dataset



(b) 2-spirals dataset



(c) ndcc dataset



(d) checker-b dataset

Figure 5: 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 (Experiment 3). 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.

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.

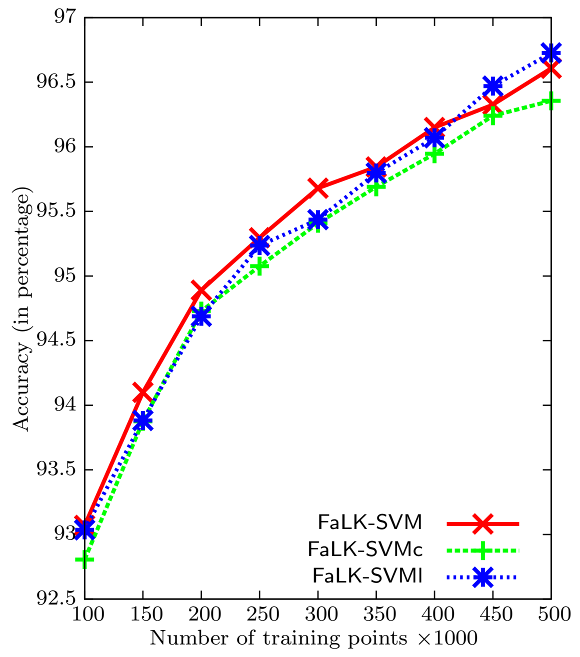
The overall conclusion we can draw about the scalability of the proposed techniques is that, at least for these 4 non high-dimensional datasets, FaLK-SVM is substantially better than the state-of-the-art kernel methods for classification, and this is achieved without affecting the accuracy performances 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 in literature LASVM, USVM, CPSP, BVM and CVM have been prevalently tested on datasets with high dimensionality or, apart for cov-type, on datasets not requiring highly non-linear decision functions. The approximated non-linear SVM solvers we tested could be 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).

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

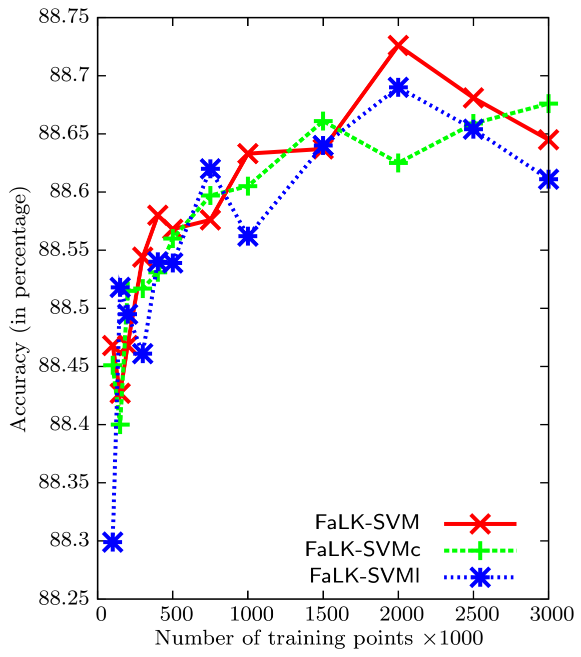
Figure 6 reports the comparison of the generalization accuracies of FaLK-SVM, FaLK-SVMc and FaLK-SVMl at increasing training set size. The computational performances for the training phase are reported in Figure 7, and for the testing phase in Figure 8.

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 rivals FaLK-SVM as the training set sizes increases. This is reasonable because FaLK-SVM uses for all the training set sizes the parameters found for the smaller training sets, and the best cross-validated parameters can differ 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 sets.

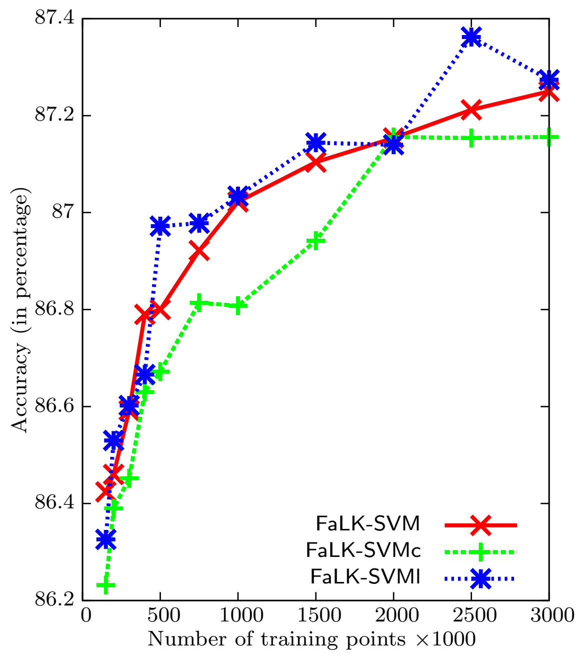
The training computational performances of Figure 7 confirm (as already discussed in Section 5.2.2) that, although FaLK-SVM and FaLK-SVMc make use of the same training algorithm, the model selection procedure selects lower values of k for FaLK-SVMc, thus assuring faster training times than FaLK-SVM. 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. In some cases FaLK-SVMl is significantly slower than FaLK-SVM. However, the training times for FaLK-SVMl include



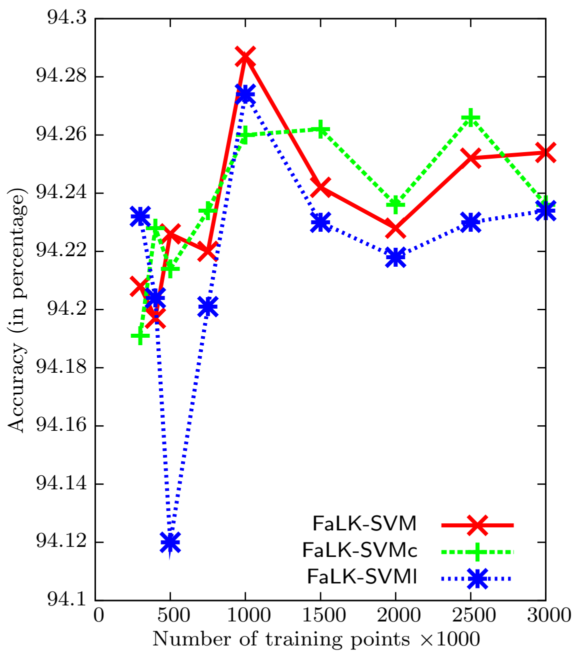
(a) cov-type dataset



(b) 2-spirals dataset

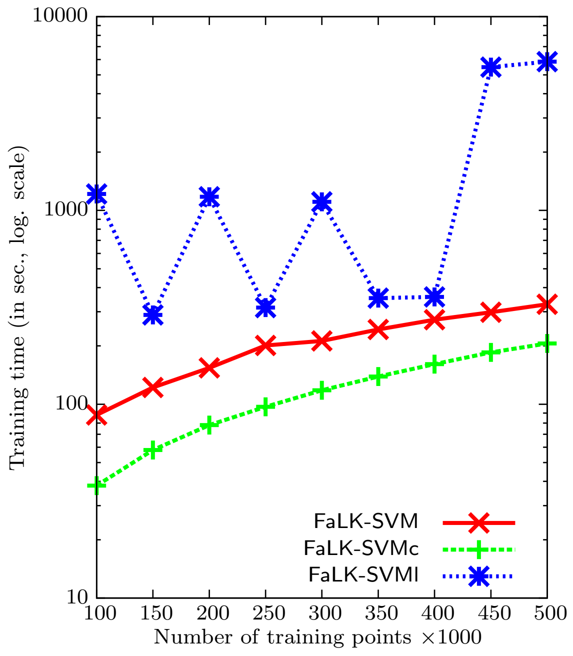


(c) ndcc dataset

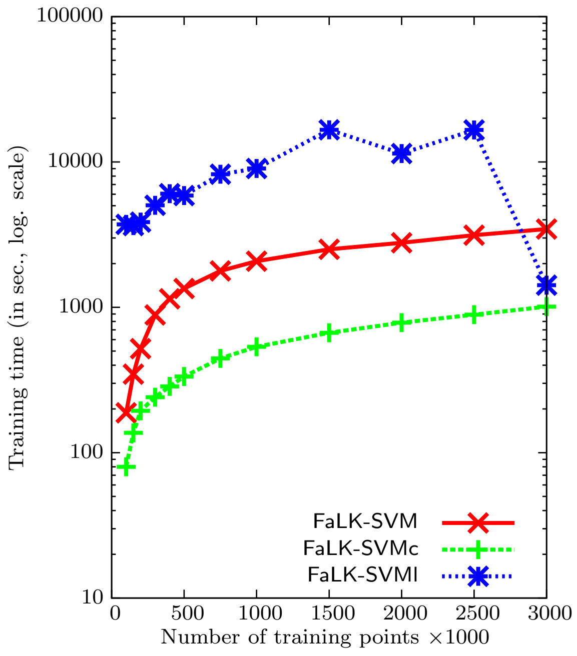


(d) checker-b dataset

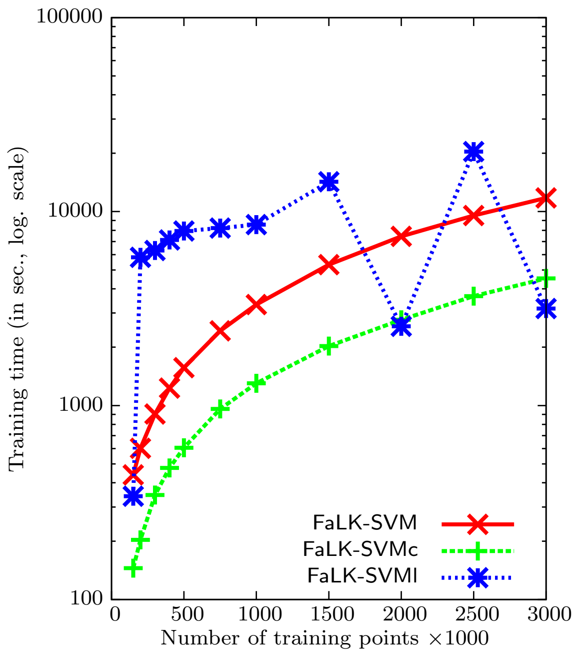
Figure 6: Generalization accuracies of FaLK-SVM, FaLK-SVMc and FaLK-SVMl on the cov-type, 2-spirals, ndcc and checker-b datasets with increasing training set sizes (Experiment 3).



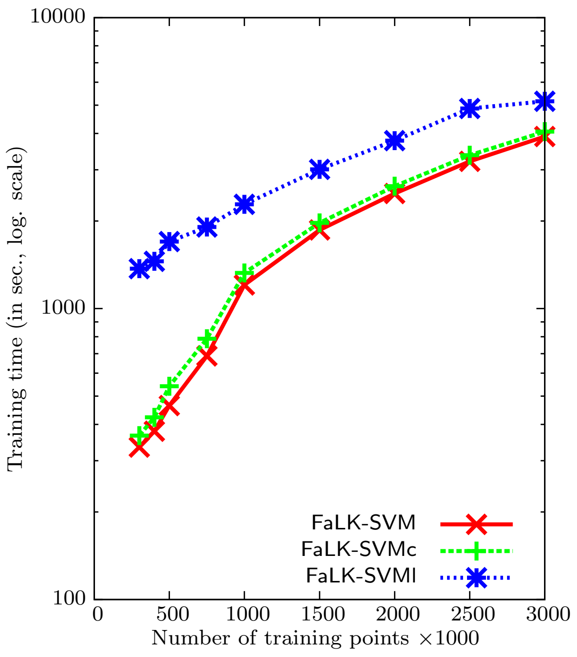
(a) cov-type dataset



(b) 2-spirals dataset

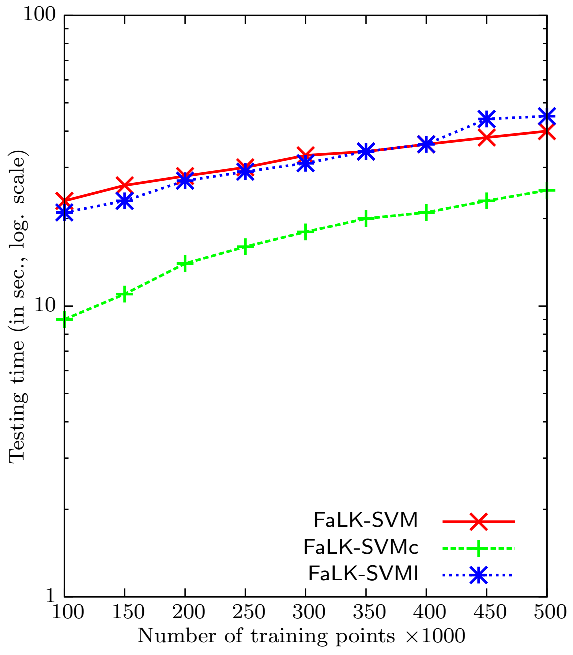


(c) ndcc dataset

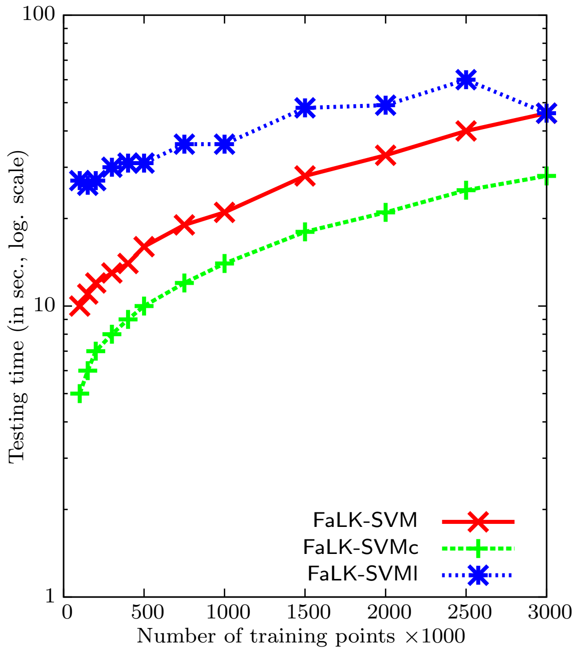


(d) checker-b dataset

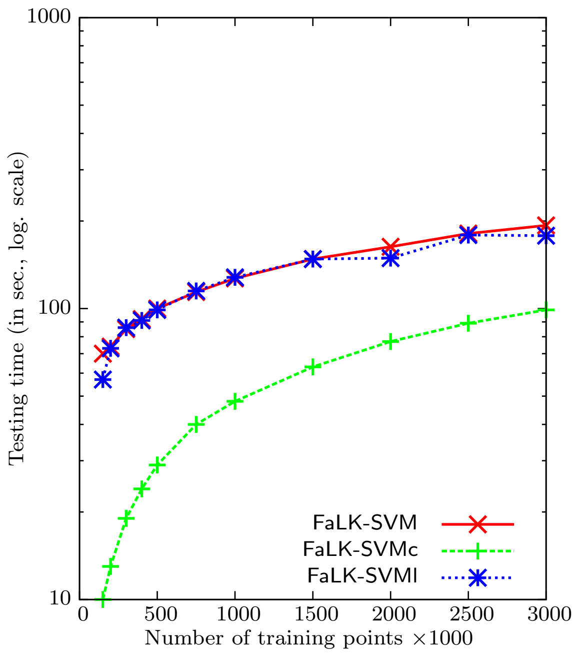
Figure 7: Training times of FaLK-SVM, FaLK-SVMc, and FaLK-SVMl on the cov-type, 2-spirals, ndcc and checker-b datasets with increasing training set sizes (Experiment 3). The times (in seconds) are reported in logarithmic scale.



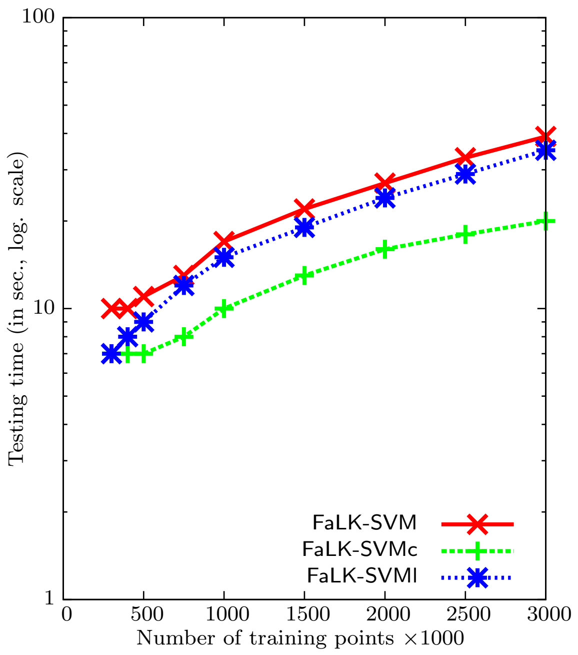
(a) cov-type dataset



(b) 2-spirals dataset



(c) ndcc dataset



(d) checker-b dataset

Figure 8: 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 (Experiment 3). The times (in seconds) are reported in logarithmic scale.

the model selection procedure whereas for FaLK-SVM we consider only the training with the optimal parameters, so 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 7 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. 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 with efficient predictions. Learning and complexity bounds for FaLK-SVM 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 introduced 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 simplify the prediction phase. 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 fact, (i) FaLK-SVM achieves very good accuracy results because it considers all the points without locally under-fitting the data and (ii) FaLK-SVM is very fast and scalable because the cardinality of the local problems can be maintained low. The variant called FaLK-SVMc further enhances testing speed at the price of a little accuracy loss, and the other variant, called FaLK-SVMl, decreases the overall training time.

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.

Further developments of the approach 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.

References

David W. Aha. *Lazy learning.* Kluwer Academic Publishers Norwell, MA, USA, 1997.

Arthur Asuncion and David Newman. UCI Machine Learning Repository, 2007. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Christopher G. Atkeson, Andrew W. Moore, and Stefan Schaal. Locally Weighted Learning. *Artificial Intelligence Review*, 11(1-5):11–73, 1997.

Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. The Curse of Highly Variable Functions for Local Kernel Machines. In *Advances in Neural Information Processing Systems*, volume 18, 2005.

Alina Beygelzimer, Sham Kakade, and John Langford. Cover Trees for Nearest Neighbor. In *Twenty-third International Conference on Machine Learning (ICML 06)*, pages 97–104, New York, NY, USA, 2006. ACM.

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.

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.

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 (IGARSS 06)*, pages 3931–3934, 2006.

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.

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.

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.

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.

Léon Bottou and Vladimir N. Vapnik. Local Learning Algorithms. *Neural computation*, 4 (6):888–900, 1992.

Léon Bottou, Corinna Cortes, John S. Denker, Harris Drucker, Isabelle Guyon, Lawrence D. Jackel, Yann Le Cun, Urs A. Muller, Eduard Säckinger, Patrice Simard, and Vladimir Vapnik. Comparison of Classifier Methods: a Case Study in Handwritten Digit Recognition. In *Twelveth IAPR International Conference on Pattern Recognition, Conference B: Computer Vision & Image Processing*, volume 2, pages 77–82. IEEE, 1994.

David Broomhead and David Lowe. Multivariable Functional Interpolation and Adaptive Networks. *Complex Systems*, 2:321–355, 1988.

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`.

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.

Q. Chang, Q. Chen, and X. Wang. Scaling Gaussian RBF Kernel Width to Improve SVM Classification. *International Conference on Neural Networks and Brain, 2005. (ICNN&B 05)*, 1:19–22, 2005.

Long Chen. New Analysis of the Sphere Covering Problems and Optimal Polytope Approximation of Convex Bodies. *Journal of Approximation Theory*, 133(1):134, 2005.

Haibin Cheng, Pang-Ning Tan, and Rong Jin. Localized Support Vector Machine and Its Efficient Algorithm. *SIAM International Conference on Data Mining*, 2007.

Vasek Chvatal. A Greedy Heuristic for the Set-Covering Problem. *Mathematics of operations research*, pages 233–235, 1979.

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.

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.

Ronan Collobert, Fabian Sinz, Jason Weston, and Léon Bottou. Trading Convexity for Scalability. In *Twenty-third International Conference on Machine Learning (ICML 06)*, pages 201–208, New York, NY, USA, 2006. ACM.

Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Machine Learning*, 20(3): 273–297, 1995.

Janez Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

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.

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.

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.

Milton Friedman. A Comparison of Alternative Tests of Significance for the Problem of m Rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940.

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.

Seth Hettich and Stephen D. Bay. The UCI KDD Archive [http://kdd. ics. uci. edu]. Irvine, CA: University of California. *Department of Information and Computer Science*, 1999.

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 *Twenty-fifth International Conference on Machine Learning (ICML 08)*, pages 408–415, New York, NY, USA, 2008. ACM.

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.

Thorsten Joachims. Making Large-Scale Support Vector Machine Learning Practical. *Advances in kernel methods: support vector learning*, pages 169–184, 1999.

Thorsten Joachims. Training linear SVMs in linear time. In *Twelveth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–226. ACM New York, NY, USA, 2006.

Thorsten Joachims and Chun-Nam Yu. Sparse Kernel SVMs via Cutting-Plane Training. *Machine Learning*, 2009.

Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009.

Michael J Kearns and Umesh V Vazirani. *An introduction to computational learning theory.* MIT Press Cambridge, MA, USA, 1994.

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.

Sathiya Keerthi, Olivier Chapelle, and Dennis DeCoste. Building Support Vector Machines with Reduced Classifier Complexity. *Journal of Machine Learning Research*, 7:1493–1515, 2006.

S. Knerr, L. Personnaz, and G. Dreyfus. Single-Layer Learning Revisited: a Stepwise Procedure for Building and Training a Neural Network. *Optimization Methods and Software*, 1:23–34, 1990.

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, 2004a. Society for Industrial and Applied Mathematics.

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, 2004b. Society for Industrial and Applied Mathematics.

Ulrich H.-G. Kressel. Pairwise Classification and Support Vector Machines. *Advances in Kernel Methods: Support Vector Learning*, pages 255–268, 1999.

Yuh-jye Lee and Olvi L Mangasarian. RSVM: Reduced Support Vector Machines. In *First SIAM International Conference on Data Mining*, 2001.

Chih-Jen Lin, Ruby C. Weng, and S. Sathiya Keerthi. Trust Region Newton Methods for Large-Scale Logistic Regression. In *Twenty-fourth International Conference on Machine learning (ICML 07)*, pages 561–568, New York, NY, USA, 2007. ACM.

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.

Olvi L. Mangasarian. A finite Newton method for classification. *Optimization Methods and Software*, 17(5):913–929, 2002.

Mario Marchand and John Shawe-Taylor. The Set Covering Machine. *The Journal of Machine Learning Research*, 3:723–746, 2003.

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. ISBN 0-13-106360-X.

Paul Nemenyi. *Distribution-Free Multiple Comparisons.* PhD thesis, Princeton, 1963.

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.

Bernard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press Cambridge, MA, USA, 2002.

Nicola Segata. FaLKM-lib v1.0: a Library for Fast Local Kernel Machines. Technical Report DISI-09-025, id 1613, DISI, University of Trento, Italy, 2009. Software available at `http://disi.unitn.it/~segata/FaLKM-lib`.

Nicola Segata and Enrico Blanzieri. Empirical Assessment of Classification Accuracy of Local SVM. In *Eighteenth Annual Belgian-Dutch Conference on Machine Learning (Benelearn 2009)*, pages 47–55, 2009a.

类似的，准的

Nicola Segata and Enrico Blanzieri. Operators for Transforming Kernels into Quasi-Local Kernels that Improve SVM Accuracy. Technical Report DISI-09-042, id 1652, 2009b.

Nicola Segata and Enrico Blanzieri. Fast Local Support Vector Machines for Large Datasets. In *International Conference on Machine Learning and Data Mining (MLDM 2009)*, volume 5632 of *Lecture Notes in Computer Science*. Springer, 2009c.

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*, 2009. In Press.

Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal Estimated Sub-Gradient SOlver for SVM. In *Twenty-fourth International Conference on Machine learning (ICML 07)*, pages 807–814, New York, NY, USA, 2007. ACM.

Alexander J. Smola, SVN Vishwanathan, and Quoc V. Le. Bundle methods for machine learning. *Advances in Neural Information Processing Systems*, 20:1377–1384, 2008.

Michael E. Thompson. NDCC: Normally Distributed Clustered Datasets on Cubes, 2006. www.cs.wisc.edu/dmi/svm/ndcc/.

Ivor W. Tsang, James T. Kwok, and Pak-Ming Cheung. Core Vector Machines: Fast SVM Training on Very Large Data Sets. *The Journal of Machine Learning Research*, 6:363–392, 2005.

Ivor W. Tsang, Andras Kocsor, and James T. Kwok. Simpler Core Vector Machines with Enclosing Balls. In *Twenty-fourth International Conference on Machine Learning (ICML 07)*, pages 911–918, New York, NY, USA, 2007. ACM.

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.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2000.

Vladimir N. Vapnik and Léon Bottou. Local Algorithms for Pattern Recognition and Dependencies Estimation. *Neural Computation*, 5(6):893–909, 1993.

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.

Xun-Kai Wei and Ying-Hong Li. Linear Programming Minimum Sphere Set Covering for Extreme Learning Machines. *Neurocomputing*, 71(4–6):570–575, 2008.

Frank Wilcoxon. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6): 80–83, 1945.

A. L. Yuille and Anand Rangarajan. The Concave-Convex Procedure. *Neural Computation*, 15(4):915–936, 2003.

Alon Zakai and Yaacov Ritov. Consistency and Localizability. *Journal of Machine Learning Research*, 10:827–856, 2009.

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.

Hao Zhang, Alexander C. Berg, Michael Maire, and Jitendra Malik. SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:2126–2136, 2006.