Ann Oper Res
DOI 10.1007/s10479-015-1956-8

Nearest neighbors methods for support vector machines

S. A. Camelo[1,3] · **M. D. González-Lima**[2] · **A. J. Quiroz**[1]

Abstract A key issue in the practical applicability of the support vector machine methodology is the identification of the support vectors in very large data sets, a problem to which a great deal of attention has been given in the literature. In the present article we propose methods based on sampling and nearest neighbors, that allow for an efficient implementation of an approximate solution to the classification problem and, at least in some problems, will help in identifying a significant fraction of the support vectors in large data sets at low cost. The performance of the proposed method is evaluated in different examples and some of its theoretical properties are discussed.

Keywords Support vector machines · k-Nearest-neighbors · Sampling

1 Introduction

Support vector machines (SVM) is a classification technique based on the supervised learning theory developed by Vapnik (1995), Cortes and Vapnik (1995). SVM has become very popular and it is currently used in many important fields of study such as computational biology (Ben-Hur et al. 2008; Noble 2004), bioinformatics (Byvatov and Schneider 2003), finances (Cao et al. 2009), medicine. For a list of real life applications we refer to the web page http://www.clopinet.com/isabelle/Projects/SVM/applist.html.

✉ M. D. González-Lima
mlima@usb.ve

S. A. Camelo
sa.camelo38@uniandes.edu.co

A. J. Quiroz
aj.quiroz1079@uniandes.edu.co

1 Dpto. de Matemáticas, Universidad de Los Andes, Bogotá, Colombia

2 Dpto. de Matemáticas, Universidad Militar Nueva Granada, Bogotá, Colombia

3 Research and Development, Quantil SAS, Bogotá, Colombia

Published online: 22 August 2015

SVM for binary classification (the one considered in this paper) is based on the following. Given points $\{X_i \in \mathbb{R}^d, i = 1, \ldots, n\}$ belonging to two classes (identified with the corresponding tags $y_i = 1$ or $y_i = -1$), they are linearly separable if there exists an hyperplane that divides them into the two different classes. The dimension d denotes the "attributes" of the data. Among the separating hyperplanes, SVM seeks to find the one that maximizes the separation margin between classes, constrained to respecting the classification of each point. This problem can be modeled, after a normalization, as the optimization problem

$$\begin{aligned} &\underset{w,b}{\text{minimize}} \ \frac{1}{2}\|w\|^2 \\ &\text{subject to } y_i \left(w^t X_i + b\right) \geq 1 \quad \forall i = 1, \ldots, n \end{aligned} \tag{1}$$

Let us denote $w^* \in \mathbb{R}^d$ and $b^* \in \mathbb{R}$ the solution to this problem. The separating hyperplane is the set of points X such that $(w^*)^t X + b^* = 0$. The support vectors are the points from the given dataset that are closer to this hyperplane. This is, the points X_i where at least one of the constraints in the feasible set of (1) is satisfied with equality, i.e. $|(w^*)^t X_i + b^*| = 1$. A new point is said to belong to class 1 or -1 depending on the region where it falls with respect to this optimal hyperplane. It is important to realize that the solution of problem (1) is the same if only the support vectors are considered in the constraints. This points out to the importance of these vectors.

If the data set is linearly nonseparable (this is, there does not exist a solution of problem (1)) then two variants are considered to formulate the optimization problem. On one hand, perturbation variables are included in order to relax the constraints, so that a margin of error in the classification is accepted. Additionally, since the data might be separable by a nonlinear decision surface, such a surface is computed by mapping the input variables into a higher dimensional "feature space" and by working with linear classification in that space. In other words, $x \in \mathbb{R}^d$ is mapped into $\phi(x) = (\phi_1(x), \phi_2(x), \ldots)^t$, where $\{\phi_m\}_{m=1}^{m=l}$ are some real functions and l may be ∞. Thus, the covariate vectors X_i are substituted with the new "feature vectors" $\phi(X_i)$.

It turns out that an explicit description of ϕ above is not required, since the solution of the optimization problem (convex, quadratic) in the nonseparable case can be found by solving its dual problem. In this way, what is really needed is to find a function that preserves, in higher dimensional spaces, the properties of the inner product. These are the kernel functions. Formally, a kernel function K is a real function on $\mathbb{R}^d \times \mathbb{R}^d$, such that, for some transformation ϕ from $\mathbb{R}^d$ into a Hilbert space, we have $K(z, a) = \phi(z)^t \phi(a)$ for every $z, a \in \mathbb{R}^d$.

In terms of the kernel function, the dual problem corresponding to (1), after relaxing the constraints and considering the higher dimensional feature space, is given by

$$\begin{aligned} &\underset{\lambda}{\text{maximize}} \ \sum_{i=1}^{n} \lambda_i - \frac{1}{2}\lambda^t Q \lambda \\ &\text{subject to } y^t \lambda = 0, \\ &\qquad 0 \leq \lambda_i \leq C \quad \text{for } i = 1, \ldots, n \end{aligned} \tag{2}$$

where C is a positive constant and $Q \in \mathbb{R}^{n \times n}$ is a symmetric positive semidefinite matrix with positive diagonal, defined as $Q_{ij} = y_i y_j K(X_i, X_j)$.

One advantage of solving problem (2) is that constraints are simpler than in the original problem, but more importantly, the dimension of the feature space for the classification can be increased without increasing the dimension of the optimization problem to solve.

Let λ^* be a solution of (2). Because of the optimality relations between primal and dual problems, the hyperplane that separates the data in the high dimensional space, determined by the normal w^* and the intersection with the axis, b^*, satisfies $w^* = \sum_{i=1}^{n} \lambda_i^* y_i \phi(X_i)$ and $b^* = 1 - \max_{\{y_j=1\}} (w^*)^t \phi(X_j)$. Hence, the function used to classify a new point X (according to the side of the "hyperplane" where it falls) can be written as

$$f(X) = \text{sign}\left(\sum_{i=1}^{n} y_i \lambda_i^* K\,(X, X_i) + b^* \right).$$

This function is called the decision or generalization function. Observe that in this sum, only the $\lambda_i^* > 0$ are relevant. Because of the strict complementarity optimality conditions, these components corresponds to the X_i support vectors. This is saying that the classification of a new point can be made just by selecting a (hopefully small) group of points from the large original data. Therefore, the main objective for SVM is to find these vectors: the support vectors. The procedure of finding the support vectors is usually referred to as the training process or training the machine. After training, it is customary to qualify the result by using it in order to classify points that are known to be in one class or another. This set of points is called the testing set. The classification error (or error rate) is the percentage of error found by the trained SVM classifying the testing set.

We refer to the book of Cristianini and J. Shawe-Taylor (Cristianini and Shawe-Taylor 2000) for details on the precedent discussion.

Observe that the Hessian of the objective function in (2) has dimension $n \times n$ being n the number of data samples, therefore it is usually very large for real applications. Besides, the entries are rarely zero so the matrix is dense which makes the optimization problem difficult to solve by standard procedures.

Osuna et al. (1997a, b) introduced a decomposition procedure to find the solution of the optimization problem (2) by solving a finite sequence of smaller optimization problems but of the same structure than (2). Their idea is based on the optimality conditions for the SVM problem and the fact that the support vectors are the data elements X_i corresponding to the components of λ_i^* that are not equal to zero. Because there are finite options for these components, their decomposition method seeks to find the optimal partition of the set $\{1, \ldots, n\}$ by interchanging elements from one set to the other of the partition while optimality is not reached. Many state of the art efficient approaches are based on (or a version of) this seminal idea by Osuna et al. Some of these algorithms are SOM (Platt 1998) which gave rise to the broadly used LIBSVM (Fan et al. 2005) (that decomposes the large problem into small problems of size two). Algorithms that decompose the large optimization problem into a sequence of medium size problems are SVM^{light} (Joachims 1998; Lin 2001), GPDT (Serafini et al. 2003; Serafini and Zanni 2005; Zanni 2006), and ASL (Gonzalez-Lima et al. 2011). There are other attempts in the literature that do not use decomposition procedures but an efficient use of the linear algebra for some kernels (for example, Ferris and Munson 2002; Woodsend and Gondzio 2011), or identification procedures of the positive components at solution as the one developed in Jung et al. (2008) in the context of primal-dual interior point methods.

Different from these approaches, in this paper we seek for a reduction of the optimization problem (2) by using statistical information of the data through sampling and k-nearest-neighbors methods. The outline of the paper is the following. In the next section we describe our approach applied to the SVM problem (2). Section 3 includes theoretical results that supports our approach. The following section considers numerical experience solving SVM problems with real life data. Last section includes some conclusions and remarks.

2 A sub-sample and k-nearest-neighbor approach to the SVM problem

A fundamental principle in statistics is that a large enough sample will be, with very high probability, representative of the behavior of the data population from which it is sampled. On the other hand, k-nearest-neighbors methods have found many applications in different statistical contexts, from the two-sample-problem (Schilling 1986), to clustering (Brito et al. 1997) and even in dimensionality reduction (Brito et al. 2002, 2013).

Given a data set $\mathcal{S} = \{X_1, \ldots, X_n\}$ and a vector x, all in $\mathbb{R}^d$, the k-nearest-neighbors of x in $\mathcal{S}$ is the set $\mathcal{NN}_k(x)$, formed by those k points that attain the k smallest values of $\|X_i - x\|$ among the data set. When x is one of the X_i, $\mathcal{NN}_k(X_i)$ is formed by the k points from the data set, different from X_i and closest to it. In what follows, we will ignore the possibility of distance ties, since we will work under the assumption that the data set has been generated from a continuous distribution, in which case distance ties have probability zero of occurring. From the practical implementation viewpoint, very fast procedures have been made available for the identification of k-nearest-neighbors in d dimensional space [see, for instance Friedman et al. (1975)].

The basic idea of the approximate solution procedure proposed in this article is outlined in the following:

Procedure nearest neighbors SVM

(i) Select a large enough random subsample, $\mathcal{T}$, of the data set.
(ii) For the subsample, solve the SVM problem.
(iii) Identify the support vector set, $\mathcal{V}_\mathcal{T}$, corresponding to the subsample $\mathcal{T}$.
(iv) For a previously chosen value of k ($k << n$), find the k-nearest-neighbors of each support vector in $\mathcal{V}_\mathcal{T}$ in the (complete) sample $\mathcal{S}$. Denote $\mathcal{NN}(\mathcal{V})$ this set of nearest neighbors.
(v) Redefine the subsample $\mathcal{T}$ as $\mathcal{T} = \mathcal{V}_\mathcal{T} \bigcup \mathcal{NN}(\mathcal{V})$.
(vi) Solve the SVM problem for the new subsample $\mathcal{T}$. If there is not significant improvement in terms of classification error with respect to the previous definition of $\mathcal{T}$, or if the maximum number of iterations has been reached, end the procedure and return the current support vector set $\mathcal{V}_\mathcal{T}$, the weights vector w and the estimated classification error. Else, go to step (iii).

The rationale of this procedure is as follows: let $\mathcal{V}_\mathcal{S}$ be the (unknown) support vectors for the SVM solution corresponding to the whole sample, $\mathcal{S}$. If the subsample $\mathcal{T}$ is large enough, we are guaranteed to have, with large probability, a sub-sample point in a small neighborhood of radio ρ_n for every vector in $\mathcal{V}_\mathcal{S}$ (this is proved in next section). These neighbors are likely to be support vectors for the solution of the SVM on $\mathcal{T}$. Then, if k is large enough, we will find a large percentage of the support vectors in $\mathcal{V}_\mathcal{S}$ by looking at neighbors of points in $\mathcal{V}_\mathcal{T}$.

The actual methods considered here are a bit more elaborate than the procedure just outlined above. Details will be given in Sect. 4. A preliminary computational evaluation of the methodology proposed here was presented in Zarruk (2012).

Several choices of parameters need to be made in order to give a complete definition of the procedure just outlined. The theoretical results described in next section might be of help in this regard. In particular, we need to define (i) the subsample size, (ii) the particular procedure to be used in each iteration to solve the subset SVM problem, (iii) the choice of k and (iv) the procedure for finding the k-nearest-neighbors in the sample of each point in $\mathcal{V}_\mathcal{T}$.

With respect to this last point, finding the k-nearest-neighbors, this is done in the original space, of dimension d, where the data sample $\mathcal{S}$ live, and not in the space in which $\mathcal{S}$ is

mapped by the use of a kernel in the SVM procedure. An explanation of why this is valid for our purposes, can be found in next section. Although some improvements have appeared later in the literature [see for instance Friedman et al. (1978)], the univariate projection method of Friedman et al. (1975) continues to be a very good compromise in terms of ease of implementation and low complexity when looking for the k-nearest-neighbors of a point in a sample. The number of operations performed to find the k-nearest neighbors of a set of m points with respect to a sample of size n in dimension d is $O(mdk^{1/d}n^{1-1/d})$.

3 Some covering properties of k-nearest-neighbors of subsamples

In this section we will prove a couple of results that serve as motivation for the methods proposed here. If a sample of size n is taken from a distribution with compact support $\mathcal{S}$ in $\mathbb{R}^d$, we have a distinguished subset of the sample and a random subsample of size δn for some positive δ. Then, very likely, the subsample will contain points at distance less than ρ from every point in the distinguished set, for any ρ such that ρ^d is significantly larger than $\ln n/n$. A similar result holds if we view the sample, the distinguished set and the subsample in the image space where the sample is mapped by the transformation ϕ defined by the kernel function used in the solution of the SVM problem. This is what Theorem 1 says. Theorem 2 gives the number k of neighbors of points in the subsample that must be considered, in order to have, very likely, every point of the distinguished set as one of the k-nearest-neighbors of a point in the subsample.

Let us begin with some basic facts, perhaps well known (although we could not find them in the literature), that will simplify our posterior analysis. Denote as above, $K(\cdot,\cdot)$ the kernel used in the SVM procedure and let ϕ be the transformation from $\mathbb{R}^d$ to a higher dimension space, implicitly defined by the kernel. We have the following lemma.

Lemma 1 *Let $\{X_1,\ldots,X_n\}$ be the covariate data set used to train the SVM, contained in the closed ball $\mathcal{B}$ centered at zero with radio R. Assume that the kernel K is a Lipschitz function, with Lipschitz constant C. Then, both $\mathcal{B}$ and $\phi(\mathcal{B})$ are totally bounded sets. Besides, for any $\epsilon>0$, $N_2(\epsilon,\phi(\mathcal{B}))\le N_2\left(\frac{\epsilon^2}{2C},\mathcal{B}\right)$. Here, for any totally bounded set $\mathcal{A}$, $N_2(\epsilon,\mathcal{A})$ denotes the minimum number of ϵ-radius balls needed to cover set $\mathcal{A}$ using the L^2 norm.*

Proof $\mathcal{B}$ is clearly totally bounded. For $m=N_2(\epsilon,\mathcal{B})$ let $\mathcal{C}=\{X_1^*,\ldots,X_m^*\}$ be an $\epsilon-$cover of $\mathcal{B}$ in the L^2 norm. For $X\in\mathcal{B}$, we have, by definition, $X^*\in\mathcal{C}$ such that $\|X-X^*\|\le\epsilon$. Then

$$\|\phi(X)-\phi\left(X^*\right)\|^2=K(X,X)+K\left(X^*,X^*\right)-2K\left(X,X^*\right)$$

but $|K(X,X)-K(X,X^*)|\le C\|(X,X)-(X,X^*)\|=C\|X-X^*\|$ by the Lipschitz condition and the same bound holds for $|K(X^*,X^*)-K(X,X^*)|$. From these bounds the result follows.

Using this lemma we get the following proposition.

Proposition 1 *Suppose the covariates X_i, for $i\le n$ form an i.i.d. sample from a probability distribution P on $\mathbb{R}^d$ with totally bounded support set $\mathcal{S}$. Furthermore, let us suppose that the probability law P is bounded away from zero, in the sense that there exists a positive constant a_1, such that, for every $x\in\mathcal{S}$, and $r>0$ small enough, if $X\sim P$, then $\Pr(\|X-x\|\le r)\ge a_1 v_d r^d$, where v_d is the volume of the unit ball in $\mathbb{R}^d$. Let us also assume that the kernel K is a Lipschitz function, with Lipschitz constant C.*

Then, for every $\phi(x) \in \phi(\mathcal{S})$, $X \sim P$, and $r > 0$ small enough, we have

$$\Pr\left(\|\phi(X) - \phi(x)\| \leq r\right) \geq p := a_1 \nu_d \left(\frac{r^2}{2C}\right)^d. \tag{3}$$

The proof is simple and omitted. It follows from the same argument in the proof of Lemma 1, substituting the role of the ball $\mathcal{B}$ by the support set $\mathcal{S}$.

The assumption $\Pr(\|X - x\| \leq r) \geq a_1 \nu_d r^d$, in Proposition 1, will hold when P has a density bounded away from zero and the boundary of $\mathcal{S}$ is regular enough. For instance, it is easy to see that this assumption holds if $\mathcal{S}$ is a regular polyhedron or a d-dimensional ellipsoid, and P has a density bounded away from zero. For a more general sufficient condition, let us denote by μ the d-dimensional Lebesgue measure. Say that a cube array is regular if the vertices of the cubes fall in a regular grid in $\mathbb{R}^d$ (in a regular grid the coordinates of the points are multiples of a fixed number l). Say that two cubes in a regular array are neighbors if they share a face. The following condition was introduced in Brito et al. (1997). We will say that the support $\mathcal{S}$ is *grid compatible* if there exists a positive number α, such that for small enough $l > 0$, $\mathcal{S}$ can be covered, possibly after translation and rotation, by a regular array $\mathcal{W}$ of cubes of side l, with a subarray $\mathcal{V} \subset \mathcal{W}$ satisfying:

(i) $\forall V \in \mathcal{V}, \mu(V \cap \mathcal{S}) \geq \alpha \mu(V)$,
(ii) $\forall W \in \mathcal{W}$, W has a neighboring cube $V \in \mathcal{V}$, and
(iii) The collection $\mathcal{V}$ is connected, in the sense that from any $V_1 \in \mathcal{V}$ one can reach any $V_2 \in \mathcal{V}$ by always moving between neighboring cubes in $\mathcal{V}$.

It is simple to verify [using properties (i) and (ii) above, (iii) is not needed] that, if $\mathcal{S}$ is grid compatible and P has a density bounded away from zero, then the assumption $\Pr(\|X - x\| \leq r) \geq a_1 \nu_d r^d$ in Proposition 1, will hold for every $x \in \mathcal{S}$ and small enough $r > 0$.

Proposition 1 says that, in an important probabilistic sense, $\phi(\mathcal{S})$ behaves as the original d-dimensional sample. This leads to the following theorems, in which the statement *with high probability* means that the probability of the event considered tends to 1, as $n \to \infty$, and we will interpret $a_n << b_n$ to mean that a_n / b_n goes to zero, as $n \to \infty$ (it is understood that both a_n and b_n are positive quantities).

Theorem 1 *Under the assumptions of* Proposition 1, *let $A = \{x_1, x_2, \ldots, x_m\}$ be a subset of the covariate sample $\{X_1, \ldots, X_n\}$ and $A' = \{x'_1, x'_2, \ldots, x'_m\}$ a subset of the transformed covariate sample, $\{\phi(X_1), \ldots, \phi(X_n)\}$ where $m = \lfloor \alpha n \rfloor$, for some $\alpha \in (0, 1)$. Suppose we take a random subsample $\mathcal{M}$ of the covariate sample of size $n' = \lceil \delta n \rceil$, where δ is a small positive number. Let $\mathcal{M}' = \phi(\mathcal{M})$. Let a_1 and C denote the constants of Proposition 1. If the radii $\rho = \rho(n)$ and $r = r(n)$ satisfy*

$$\rho^d \gg \frac{\ln(\alpha n)}{\delta\, n\, a_1\, \nu_d} \quad \text{and} \quad \left(\frac{r^2}{2C}\right)^d \gg \frac{\ln(\alpha n)}{\delta n a_1 \nu_d} \tag{4}$$

then, with high probability, for every $x_i \in A$ and every $x'_j \in A'$ there are a $X_{i^*} \in \mathcal{M}$ and a $\phi(X_{j^*}) \in \phi(\mathcal{M})$, such that

$$\|x_i - X_{i^*}\| \leq \rho \quad \text{and} \quad \|x'_j - \phi\left(X_{j^*}\right)\| \leq r. \tag{5}$$

Proof We prove only the statement about the $x'_j \in A'$ in (5), the proof of the statement about the $x_i \in A$ is similar and easier. For the moment, fix $x' \in A'$. Let $N_{x',r} = \#\{\phi(X_j) \in \mathcal{M}' : \|x' - \phi(X_j)\| \leq r\}$. Let $\text{Bin}(n', p)$ denote the Binomial distribution, with n' trials and p as in Proposition 1. Then, by that proposition,

$$\Pr\left(N_{x',r}=0\right) \leq \text{Bin}\left(n', p\right)(0) \leq \exp\left(-n'p\right) \leq \exp\left(-\delta n a_1 v_d \left(\frac{r^2}{2C}\right)^d\right).$$

Thus,

$$\Pr\left(N_{x',r}=0 \text{ for some } x' \text{ in } A\right) \leq \exp\left(\ln(\alpha n) - \delta n a_1 v_d \left(\frac{r^2}{2C}\right)^d\right). \tag{6}$$

In order to make this last probability go to zero, it suffices to let the exponent in the right hand side of (6) tend to $-\infty$, and this will happen for a radius r as given in (4).

A value of ρ^d such as $\ln^2(\alpha n)/(\delta n a_1 v_d)$, will satisfy the statement (4). In fact, for this particular choice, the probability of having $\|x_i - X_{i^*}\| \leq \rho$ for every $x_i \in A$ and some X_{i^*} in the subsample (depending on x_i), will be less than n^{-2}, for large enough n. The same comment is valid in the transformed space, if we let $r^{2d} = (2C)^d \ln^2(\alpha n)/(\delta n a_1 v_d)$.

We think of the sets A and A' of Theorem 1 as the set of support vectors of the training data set in the original space and in the transformed space, respectively (below we will discuss whether the assumptions of the theorem may be valid for the support vectors). Since the conditions in (4) can be satisfied by values of ρ and r going to zero, as n goes to infinity, Theorem 1 is saying that a large enough subsample will, very likely, have elements near all of the support vectors, both in the original and transformed sample space.

To connect Theorem 1 with the procedures we propose here, we would like to know how large must be the number of neighbors, k, in order for all the k-nearest-neighbors balls of the points in the subsample $\mathcal{M}$ to have radius larger or equal than the ρ of inequality (4). To give a result in that direction, we will add some assumptions to those of the previous Theorem. For a positive integer k, $(k < n)$ and $x \in \mathcal{S}$, let $\mathcal{NN}_k(x)$ be the set of k-nearest-neighbors of x in the covariate sample $\mathcal{S}$. Let $\tau_k(x)$ denote the largest distance from x to an element of $\mathcal{NN}_k(x)$.

Theorem 2 *In the same setting of* Theorem 1*, add the following assumption: suppose the distribution P of the training covariate data $\{X_1, \ldots, X_n\}$ is bounded above, in the sense that there exists a positive constant a_2, such that, for every $x \in \mathcal{S}$, and $\epsilon > 0$ small enough, if $X \sim P$, then* $\Pr(\|X - x\| \leq \epsilon) \leq a_2 v_d \epsilon^d$*. Then, for*

$$\rho^d = \frac{\ln^2(\alpha n)}{\delta n a_1 v_d} \quad \text{and} \quad k = \frac{2a_2 \ln^2(\alpha n)}{\delta a_1}, \tag{7}$$

with high probability, $\tau_k(x) > \rho$ for every x in the subsample $\mathcal{M}$.

Proof The proof follows the argument of Theorem 2.1 in Brito et al. (1997). For each $x \in \mathcal{M}$, let $N_{x,\rho} = \#\{j \leq n : \|x - X_j\| \leq \rho\}$. Let $R_{\min} = \min_{x \in \mathcal{M}} \tau_k(x)$. By the assumption in the statement of Theorem 2, conditionally on x, $N_{x,\rho}$ has a $\text{Bin}(n-1, p)$ distribution, with $p \leq a_2 v_d \rho^d$. Also

$$\Pr(R_{\min} \leq \rho) = \Pr\left(\bigcup_{x \in \mathcal{M}} \{N_{x,\rho} \geq k\}\right). \tag{8}$$

By our choice of ρ and k, we have, for a fixed $x \in \mathcal{M}$, $\mathbb{E}N_{x,\rho} \leq k/2$. It follows that

$$\Pr\left(N_{x,\rho} \geq k\right) \leq \Pr\left(N_{x,\rho} \geq \mathbb{E}N_{x,\rho} + \frac{a_2 \ln^2(\alpha n)}{\delta a_1}\right).$$

Applying one of the Chernoff–Okamoto inequalities for binomial probabilities [see, for instance, Theorem 1 in Janson (2002)], we get:

$$\Pr\left(N_{x,\rho} \geq k\right) \leq \exp\left(-\frac{(k/2)^2}{2\left(\mathbb{E}\left(N_{x,\rho}\right) + k/6\right)}\right) \leq \exp(-k/8).$$

It follows that

$$\Pr\left(R_{\min} \leq \rho\right) \leq \lceil \delta n \rceil \exp(-k/8) = \lceil \delta n \rceil \left(\frac{1}{\alpha n}\right)^{(a_2 \ln(\alpha n)/(8\delta a_1))}$$

and the last bound will be less than n^{-2} for n large enough, finishing the proof.

Following the same line of reasoning, a result similar to Theorem 2 can be established for the data in the transformed space. We do not include it here for two reasons: (i) that result is not that satisfactory, in the sense that a value of k significantly larger than the one given in Theorem 2 is required (although still converging to zero as a fraction of n) and (ii) the result on the transformed space is not essential to our procedure, since having the k-nearest-neighbors balls of points in the subsample $\mathcal{M}$, for k large enough, containing the support vectors (with high probability) is enough as motivation for the method proposed. Theorems 1 and 2 together say exactly that: With high probability all the k-nearest-neighbors balls of points in the sample $\mathcal{M}$ will have radii larger than ρ, while every point in the subset A will be within distance ρ of a point in $\mathcal{M}$. Thus, when we include the k-nearest-neighbors of the points in the subsample, all the elements of A will be included, with high probability.

Still, it is necessary to discuss whether the support vectors for the SVM problem corresponding to the complete sample can be assumed to satisfy the assumptions of our theorems. The discussion of this issue will not be formal, but just intuitive, since we do not feel that the tools are available yet to settle the issue formally. Suppose X^* is a support vector for the complete training sample. By being a support vector, X^* should be, within a certain neighborhood, the closest point of the sample to the separating surface for the problem. Thus, at least in some directions (towards the separating surface) there should be no other sample points. Still, conditionally on being a support vector, nearby sample points can be allowed in certain directions (away from the separating surface) and for that reason it seems reasonable to believe that, conditionally, the bound $\Pr(\|X - X^*\| \leq r) \geq a_1 v_d r^d$ of Proposition 1 can hold, for an appropriate value of the constant a_1.

4 Implementation issues and performance evaluation

The value of k in the statement of Theorem 2 could in principle be approximated: the constants a_1 and a_2 are estimable by non-parametric density estimation methods, for instance, while δ is given by our choice of subsample size and α, although unknown, could be guessed from solutions obtained previously for similar problems, or estimated from solutions for subsamples or some other method. Still, the value of k thus obtained, being an upper bound, could be too large in terms of computation time savings. Also, since we expect the subsample to contain, very likely, points near most of the support vectors of the solution for the complete data set, it can be expected that with those points or some of their neighbors included as support vectors, the solution obtained for the subsample can achieve a generalization error close to that of the solution for the full training data. For these reasons, we will prefer to use relatively small values of k and work iteratively, augmenting in relatively small steps

the size of the subsample considered, until the classification error on the test data set ceases to improve significantly. In what follows, $\mathcal{D}$ denotes the training sample of size n. We will consider the following two variants for the procedure outlined in Sect. 2.

Algorithm 1

1. Select a random subsample $\mathcal{T}^{(0)}$ of size $\lceil \delta n \rceil$ from the training data set $\mathcal{D}$ of size n.
2. Initialize $\mathcal{S}^{(0)} = \mathcal{D} \backslash \mathcal{T}^{(0)}$ and $j = 0$.
3. Solve the SVM problem for $\mathcal{T}^{(j)}$ and identify the support vectors $\mathcal{V}_{\mathcal{T}}^{(j)}$.
4. For the previously chosen value of k, find the k-nearest-neighbors in $\mathcal{S}^{(j)}$ of each support vector. Denote $\mathcal{N}(\mathcal{V}_{\mathcal{T}}^{(j)})$ this set of nearest neighbors.
5. Redefine the subsample as $\mathcal{T}^{(j+1)} = \mathcal{V}_{\mathcal{T}}^{(j)} \cup \mathcal{N}(\mathcal{V}_{\mathcal{T}}^{(j)})$. Define $\mathcal{S}^{(j+1)} = \mathcal{S}^{(j)} \backslash \mathcal{N}(\mathcal{V}_{\mathcal{T}}^{(j)})$.
6. Solve the SVM problem for $\mathcal{T}^{(j+1)}$. If there is no significant improvement in terms of the classification error, or the allotted time has been exceded, end the procedure and return the current solution. Else, $j \to j + 1$ and go to step (3).

Algorithm 2

1. Select a random subsample $\mathcal{T}^{(0)}$ of size $\lceil \delta n \rceil$ from the training data set $\mathcal{D}$ of size n.
2. Initialize $\mathcal{S}^{(0)} = \mathcal{D} \backslash \mathcal{T}^{(0)}$ and $j = 0$.
3. Solve the SVM problem for $\mathcal{T}^{(j)}$ and identify the support vectors $\mathcal{V}_{\mathcal{T}}^{(j)}$. In case $j > 0$, consider only the new support vectors $\hat{\mathcal{V}}_{\mathcal{T}}^{(j)} = \mathcal{V}_{\mathcal{T}}^{(j)} \cap \mathcal{R}^{(j-1)}$, otherwise, $\hat{\mathcal{V}}_{\mathcal{T}}^{(0)} = \mathcal{V}_{\mathcal{T}}^{(0)}$.
4. For the previously chosen value of k, find the k-nearest-neighbors in $\mathcal{S}^{(j)}$ of each support vector in $\hat{\mathcal{V}}_{\mathcal{T}}^{(j)}$. Denote $\mathcal{N}(\hat{\mathcal{V}}_{\mathcal{T}}^{(j)})$ this set of nearest neighbors.
5. Redefine the subsample as $\mathcal{T}^{(j+1)} = \mathcal{V}_{\mathcal{T}}^{(j)} \cup \mathcal{N}(\hat{\mathcal{V}}_{\mathcal{T}}^{(j)}) \cup \mathcal{R}^{(j)}$, where $\mathcal{R}^{(j)}$ is a random subsample from $\mathcal{S}^{(j)} \backslash \mathcal{N}(\hat{\mathcal{V}}_{\mathcal{T}}^{(j)})$ of size $\lceil \epsilon n \rceil$. Define $\mathcal{S}^{(j+1)} = \mathcal{S}^{(j)} \backslash (\mathcal{N}(\hat{\mathcal{V}}_{\mathcal{T}}^{(j)}) \cup \mathcal{R}^{(j)})$.
6. Solve the SVM problem for $\mathcal{T}^{(j+1)}$. If there is no significant improvement in terms of the classification error, or the allotted time has been exceded, end the procedure and return the current solution. Else, $j \to j + 1$ and go to step (3).

Both algorithms start with a subsample $\mathcal{T}^{(0)}$ and extend its support vector set until an approximate solution to the full problem is found.

Algorithm 1 extends the support vectors by adding their neighbors. Observations that are not support vectors, are discarded after each iteration. Looking for the support vectors in $\mathcal{S}^{(j)}$ instead of in the complete dataset $\mathcal{D}$ is done to avoid resampling of observations that where previously discarded.

Algorithm 2 includes two modifications: first, the subsample is extended using not only neighbors from support vectors, but adding new random points. This is done so that, in case the initial subsample is not representative enough, leaving some regions of the domain without representative points, the subsequent random subsamples $\mathcal{R}^{(j)}$ might enrich the original subsample with points in those regions. Second, at each iteration we only consider neighbors of new support vectors (drawn from the new random subsample $\mathcal{R}^{(j)}$).

4.1 Parameter choice

In preliminary evaluations, we found that $\delta = 0.01$ (that is, subsamples of 1 % of the training data set) is a small initial subsample size for most data sets considered in our experiments, while $\delta = 0.1$ (subsamples of 10 % of the training data) is a medium-to-large value in the sense that with this choice of δ, the procedures are able to find a significant part of the support vectors of the full training data and achieve a better testing classification error. In

the evaluations presented below, we used these two values, 1 and 10 % for the parameter δ. When $\delta = 0.01$ we used $k = 5$ as the number of neighbors, while a smaller value, $k = 3$ was used when δ was 10 %. For the parameter ϵ in Algorithm 2, we used always 10 %. Thus, in what follows, the performance of four procedures is considered: Algorithms 1 and 2 with small sized ($\delta = 1\,\%$) and medium sized ($\delta = 10\,\%$) starting subsamples.

Kernels used in our evaluation were the linear, radial and polynomial of degree two. These are given by the formulas

$$K_l(x, y) = x^T y$$
$$K_r(x, y) = \exp\left(-\gamma \|x - y\|\right)$$
$$K_p(x, y) = \left(x^T y + a\right)^2$$

For each dataset we chose the kernel parameters and the costs C_r, C_p, and C_l for the optimization process (this is, the values of the parameter C presented in the SVM problem (2) when using the radial, polynomial and lineal kernel, respectively) in the following way. Two random 20 % disjoint samples from the training data were taken. A radial SVM was trained with the first of these samples, for $\gamma = 2^n \hat{d}$ and $C_r = 2^m$ and each n and m value in $\{-4, -3, \ldots, 4\}$ that minimize the test error in the second sample.[1] We chose a and C_p through a similar procedure, and fixed $C_l = 1$.

For each dataset, kernel, and method, we choose a subsample of the given size and run the algorithm with the chosen relevant parameters. At each iteration we estimate the generalization error on a test data set disjoint from the training data (and provided in the library used). To lessen the dependence of the results on the randomness of the chosen subsample, we repeat the procedures five times with five different and disjoint initial subsamples, and we report the mean error and mean computing time for each iteration.

4.2 Performance evaluation

We test the performance of Algorithms 1 and 2 on six datasets from the LIBSVM library using the LIBSVM code.[2] We compile with R using the e1071 package[3] on a personal computer, MacBook 2.4 GHz and Intel Core 2 Duo processor with 4 GB 1067 MHz DDR3 memory.

Table 1 shows the size of the training and testing datasets, as well as the number of features for each problem, while Table 2 displays the parameters chosen for each dataset. In the selection of the datasets for our evaluation, we have tried to include problems with widely varying numbers of features, to appreciate the effect of dimensionality on the methods studied.

With the parameters in Table 2 we adjust a SVM to each complete training dataset, and estimate the error rate on the test data, as reference for evaluating the performance of our procedures. Table 3 shows the error rates and computing times in seconds for these full data SVMs.

Figures 1 and 2 show the plots of computing times versus test error rates for each method and size of initial subsample for the procedures considered here. The added horizontal lines indicate the error rate for the full problem. Times are reported as percentages of the computing time for the solution for the full training dataset. Each point in these plots corresponds to

[1] $\hat{d}$ represents the median of the squared euclidean distance between data from different classes in the feature space, as suggested by Wu and Wang (2009).

[2] For more information on the LIBSVM library and the datasets go to www.csie.ntu.edu.tw/~cjlin/libsvm/

[3] For more information on the e1071 package go to cran.r-project.org/web/packages/e1071/.

Table 1 Datasets description

Dataset	TrainSize	TestSize	Features
A9A	32,561	10,000	123
COD-RNA	59,535	20,000	8
IJCNN1	49,990	15,000	22
SVMGUIDE1	3089	4000	21
W7A	24,692	8000	300
COVTYPE	50,000	10,000	54

Table 2 Parameters

Dataset	γ	C_r	a	C_l
A9A	0.00246	4	2	0.03125
COD-RNA	0.0371	32	0.03125	8
IJCNN1	0.0812	32	8	0.5
SVMGUIDE1	0.186	32	0.5	16
W7A	0.000728	1	0.03125	0.03125
COVTYPE	0.0326	16	2	0.25

Table 3 Error rates and computing times for SVM from full training data

	Error rates (%)			Time (s)		
	Linear	Polynomial	Radial	Linear	Polynomial	Radial
A9A	15.33	16.80	15.43	1647.33	12,003.55	556.13
COD-RNA	4.61	3.66	3.48	93.19	1252.77	215.11
IJCNN1	7.75	2.07	1.28	195.98	466.63	104.65
SVMGUIDE1	4.28	3.60	3.13	0.12	0.33	0.31
W7A	0.98	1.53	1.03	211.36	112.45	71.31
COVTYPE	33.40	32.63	33.43	1051.63	6665.07	908.97

an iteration of a procedure. Points to the left, near the added horizontal lines, correspond to better performance. For reasons of space, in each case the results are displayed only for the radial kernel and the kernel, of the other two, with better performance, in terms of testing error, for the complete training data (Table 3).

The figures indicate that in all cases, at least for some of the nearest neighbor methods proposed, a classification error close to the error for the full training dataset is attained in a time significantly smaller than the required for the complete dataset solution. This means that considerable time can be saved by classifying using one of the neighbors methods instead of solving for the entire training set.

From Figs. 1 and 2 it is also clear that using an initial subsample of 10 % of the training data is almost always preferable, in terms of performance, than an initial sample of 1 %. Regarding the algorithms, the figures suggest that Algorithm 2 is more reliable than Algorithm 1, in the sense of reaching in all cases (when $\delta = 0.1$) a solution with error near the error for the full training dataset after a computation time significantly smaller than the solution time for the complete data set. Still, for the radial kernel, and specially for the high dimensional problems,

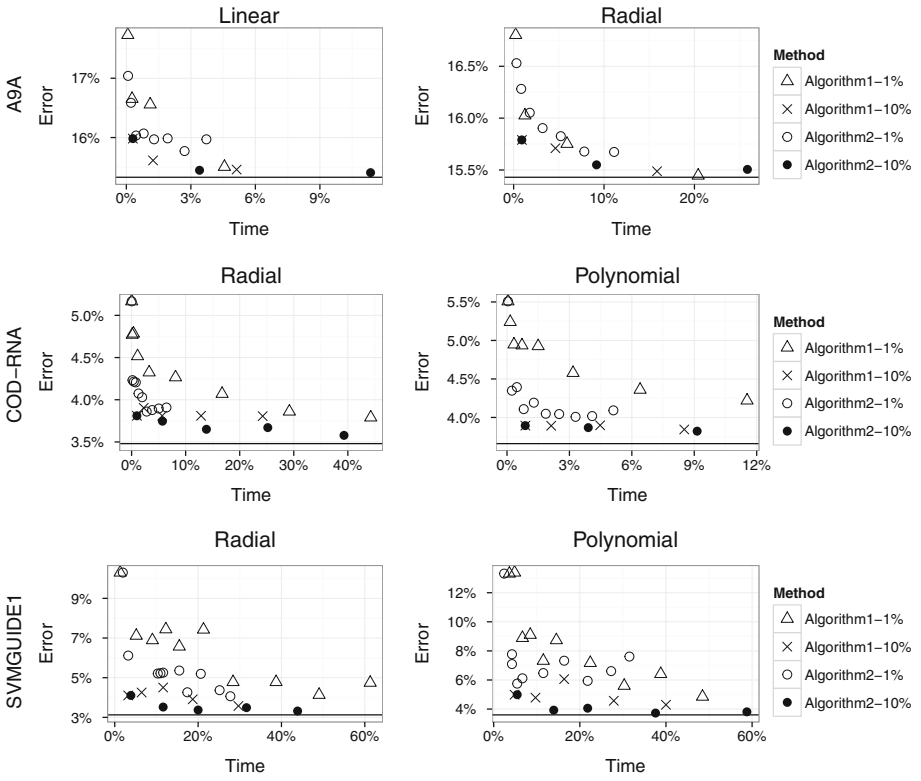

Fig. 1 Test error rate versus computing time, Algorithms 1 and 2

A9A and W7A, Algorithm 1 (again, with $\delta = 0.1$) shows a very good performance, reaching a very good error rate at a low computational cost (sometimes even faster than Algorithm 2).

If we choose the winning combination of Algorithm 2 with a medium-sized subsample ($\delta = 0.1$), the error rate for the full problem is practically always well approximated before 20 % of the time that it takes to solve the full problem, taking less than 5 % of this time in some cases. An advantage of this winning method is that the error rate is decreasing in computing time. This means that whenever the error rate starts fluctuating, it is because the error rate for the full problem has been reached and the algorithm can thus be stopped. This is not always the case for Algorithm 1, which may show some erratic movements in the error rate even when it is still far from the optimum.

It might happen, however, that the available computing power only permits working with small subsamples. In this case, Algorithm 2 is still the best candidate, since Algorithm 1 tends to stagnate around an error rate above the optimal. The explanation is that if the subsample is small, only part of the feature space will be covered and more can be gained from exploring new regions through random subsamples than from seeking neighbors of the first subsample.

For each one of the datasets considered and five different runs of the winning method, Table 4 shows the average error rate for each iteration with standard deviations indicated inside parentheses (the result is displayed for only one kernel). The table also reports the average percentage of support vectors in the complete sample that are included among the support vectors in the current subsample. This percentage increases with each iteration. Also

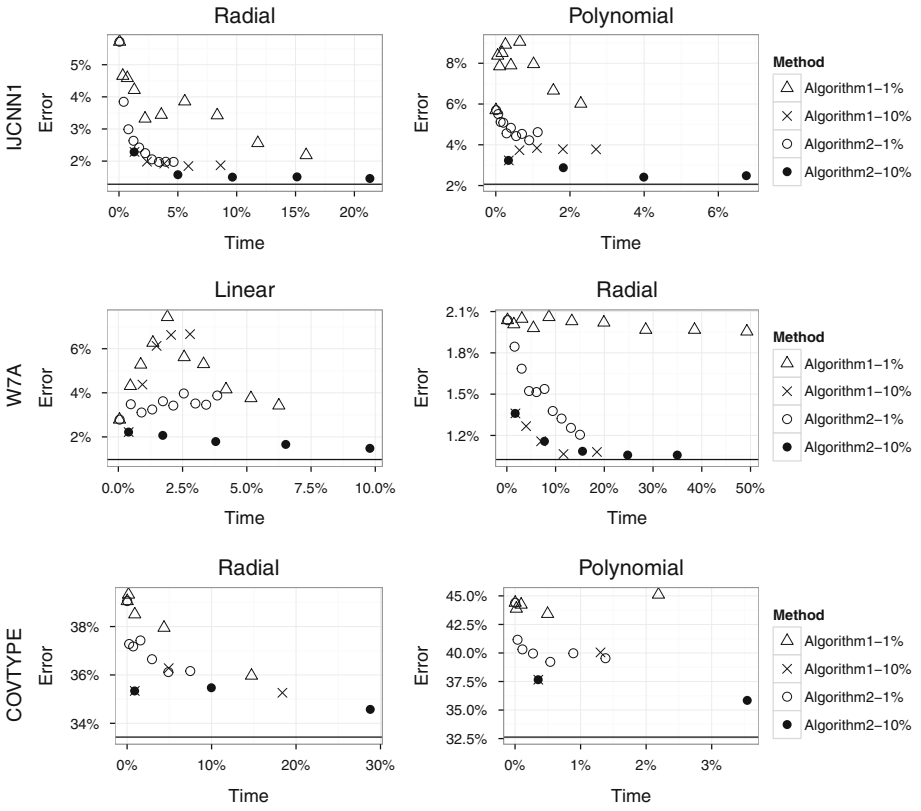

Fig. 2 Test error rate versus computing time, Algorithms 1 and 2, continued

included in the Table is the percentage of support vectors of the complete dataset captured by a random sample of the same size as the current subsample (RSupports in Table 4). The average percentage of support vectors found is always higher for the neighbors method than for the random subsample, indicating that the probability of finding support vectors of the full dataset in the subsamples produced by the neighbors method, is higher than the probability of finding them in a randomly chosen subsample. Similar results, in terms of percentage of support vectors found were obtained for Algorithm 1, and are available from the authors.

5 Concluding remarks

In this paper a novel approach for solving the SVM problem (2), based on subsample and neighbor methods, is proposed. Theoretical support is developed and four algorithmic versions are presented and compared. Computer simulations indicate that Algorithm 2 from previous section has a good performance, achieving error rates that are very close to the full problem classification error in a small proportion of the total time required for the complete dataset problem to be solved. This algorithm can be easily accommodated to the computer power at hand. Even if the problem is too big, the size of the initial subsample and the number of iterations can be adjusted to obtain good performance. It is important, however, to keep the initial subsample as big as possible.

Table 4 Test error rate and support vector percentage for the winning model

COD-RNA—radial					IJCNN1—polynomial				
Iter	Error rate (%)	Supports	RSupports	Time (s)	Iter (%)	Error rate (%)	Supports	RSupports	Time (s)
0	3.81 (0.12)	9.18	9.18	2.108	0	3.23 (0.19)	6.75	6.75	1.592
1	3.75 (0.05)	31.78	22.49	12.286	1	2.88 (0.19)	22.09	16.50	8.494
2	3.65 (0.11)	50.04	35.20	29.778	2	2.41 (0.2)	35.82	27.28	18.614
3	3.67 (0.07)	64.74	48.03	54.194	3	2.48 (0.22)	47.97	38.58	31.516
4	3.58 (0.07)	76.34	60.15	84.524	4	2.3 (0.21)	57.93	49.82	48.518
Total	3.48			215.11	Total	2.07			466.63
W7A—linear					SVMGUIDE1—polynomial				
Iter	Error rate (%)	Supports	RSupports	Time (s)	Iter (%)	Error rate (%)	Supports	RSupports	Time (s)
0	2.22 (0.38)	5.71	5.71	0.84	0	4.99 (0.91)	7.59	7.59	0.018
1	2.07 (0.23)	17.15	12.62	3.654	1	3.92 (0.49)	26.72	19.34	0.046
2	1.79 (0.15)	27.94	19.56	8.01	2	4.06 (0.48)	43.43	29.78	0.072
3	1.66 (0.11)	35.50	28.41	13.772	3	3.73 (0.29)	57.37	43.21	0.124
4	1.48 (0.04)	41.09	37.18	20.692	4	3.81 (0.13)	68.10	54.38	0.194
Total	0.98			211.36	Total	3.60			0.33
COVTYPE—radial					A9A—linear				
Iter	Error rate (%)	Supports	RSupports	Time (s)	Iter (%)	Error rate (%)	Supports	RSupports	Time (s)
0	35.34 (1.27)	8.74	8.74	8.092	0	15.98 (0.17)	8.99	8.99	4.988
1	35.47 (1.45)	35.33	29.11	90.632	1	15.45 (0.19)	34.01	26.99	56.064
2	34.58 (0.55)	57.69	47.96	261.66	2	15.41 (0.18)	54.65	44.23	187.068
Total	33.43			908.97	Total	15.33			1647.33

Because good results can be obtained with limited time, our approach proves to be especially useful for cross-validation: it might be the case that the computer power is enough to run SVM once, but not the number of times necessary to try several cost or kernel parameters and see which ones give the optimal error rate. In that case, a good option would be to use a neighbor algorithm with few iterations.

A last issue that deserves discussion is the fact that our proposal, at least in its present form, is restricted to the Euclidean vector space situation, where all data points have a fixed and finite number of coordinates, and is limited to dimensions where the nearest neighbors identification algorithms available for the Euclidean setting, such as those mentioned in the last paragraph of Sect. 2, can work reasonably fast. In this regard, see above the discussion regarding the performance of our procedure on datasets A9A and W7A, in dimensions 123 and 300, respectively. In many important problems the Euclidean dimension is very large or, simply, the Euclidean model is not valid and more general metric spaces must be considered. Such is usually the case in applications dealing with text, multimedia data and computational biology data, among others. For these situations one of the obstacles to solve in order to adapt the methodology proposed here, would be the existence of an appropriate notion of inner product between data points, so that the SVM paradigm can be applied. Thus, in the infinite dimensional case, only data allowing a Hilbert space representation could be considered. In the case of text data, this issue has been addressed and different notions of inner product and kernels for the SVM approach have been proposed. See the last chapter in Cristianini and Shawe-Taylor (2000) for details.

The other aspect that needs to be addressed for the generalization of the methodology proposed here is that of identifying nearest neighbors of data in general metric spaces at relatively low computational cost. In this regard, a host of methods have been developed in order to solve "proximity search queries in metric spaces", including nearest neighbors identification. A classical reference in this subject is Chavez et al. (2001), where the importance of the idea of "intrinsic dimension" has been stressed. A more recent survey of the varied available approaches for this problem is provided in Patella and Ciaccia (2009). Some of the methods proposed for proximity search in metric spaces are based on clustering of the data set and on the development of appropriate data structures for the search problem. In Uribe et al. (2006) a data structure based on Voronoi diagrams, called evolutionary geometric near-neighbor access tree (EGNAT) is proposed. Probabilistic algorithms based on the estimation of the number of elements in intersections of balls in the metric space are considered in Amato et al. (2003), while an alternative approximate search method, that finds nearest neighbors by "walking" through a neighbor tree structure is proposed in Navarro (2002).

We intend to consider the extension of the methodology proposed here beyond the scope of Euclidean vector spaces in future research.

References

Amato, G., Rabitti, F., Savino, P., & Zezula, P. (2003). Region proximity in metric spaces and its use for approximate similarity search. *ACM Transactions on Information Systems*, *2*(21), 192–227.

Ben-Hur, A., Ong, C.-S., Sonnenburg, S., Scholkopf, B., & Ratsch, G. (2008). Support vector machines and kernels for computational biology. *PLoS Computer Biology, 4*(10). http://svmcompbio.tuebingen.mpg.de/.

Brito, M. R., Chavez, E. L., Quiroz, A. J., & Yukich, J. E. (1997). Connectivity of the mutual k-nearest neighbor graph in outlier detection and clustering. *Statistics and Probability Letters*, *35*, 33–42.

Brito, M. R., Quiroz, A. J., & Yukich, J. E. (2002). Graph theoretic procedures for dimension identification. *Journal of Multivariate Analysis*, *81*, 67–84.

Brito, M. R., Quiroz, A. J., & Yukich, J. E. (2013). Intrinsic dimension identification via graph-theoretic methods. *Journal of Multivariate Analysis*, *116*, 263–277.

Byvatov, E., & Schneider, G. (2003). Support vector machine applications in bioinformatics. *Applied Bioinformatics*, *2*(2), 67–77.

Cao, B., Zhan, D., & Wu. X. (2009). Application of svm in financial research. In *CSO international joint conference on computational sciences and optimization*, Vol. 2, pp. 507–511.

Chavez, E., Navarro, G., Baez-Yates, R., & Marroquin, J. L. (2001). *ACM Computing Surveys*, *3*(33), 273–321.

Cortes, C., & Vapnik, V. (1995). Support vector networks. *Machine Learning*, *20*, 1–25.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge: Cambridge University Press.

Fan, R.-E., Chen, P.-H., & Lin, C.-J. (2005). Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, *6*, 1889–1918.

Ferris, M. C., & Munson, T. S. (2002). Interior point methods for massive support vector machines. *SIAM Journal on Optimization*, *13*(3), 783–804.

Friedman, J. H., Baskett, F., & Shustek, J. (1975). An algorithm for finding nearest neighbors. *IEEE Transactions on Computers*, *C–24*(10), 1000–1006.

Friedman, J. H., Bentley, J. L., & Finkel, R. (1978). An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, *3*(3), 209–226.

Gonzalez-Lima, M. D., Hager, W. W., & Zhang, H. (2011). An affine-scaling interior-point method for continuous knapsack constraints with application to support vector machines. *SIAM Journal on Optimization*, *21*(1), 361–390.

Janson, S. (2002). On concentration of probability. *Contemporary Combinatorics*, *10*, 289–301.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In C. Nédellec & C. Rouveirol (Eds.), *Proceedings of the 10th European conference on machine learning (ECML-98)*, pp. 137–142. Springer, Berlin.

Jung, J. H., Leary, D. P. O., & Tits, A. L. (2008). Adaptive constraint reduction for training support vector machines. *Electronic Transactions on Numerical Analysis*, *31*, 156–177.

Lin, C.-J. (2001). On the convergence of the decomposition method for support vector machines. *IEEE Transactions on Neural Networks*, *12*, 1288–1298.

Navarro, G. (2002). Searching in metric spaces by spatial approximation. *The VLDB Journal*, *11*(1), 28–46.

Noble, W. S. (2004). Support vector machine applications in computational biology. In B. Schoelkopf, K. Tsuda, & J.-P. Vert (Eds.), *Kernel methods in computational biology* (pp. 71–92). Cambridge: MIT Press.

Osuna, E. E., Freund, R., & Girosi, F. (1997a). *Support vector machines: Training and applications*. Technical report A.I. memo no. 1602, CBCL paper no. 144, Massachusetts Institute of Technology, Cambridge, MA, USA.

Osuna, E. E., Freund, R., & Girosi, F. (1997b). Training support vector vector machines: An application to face detection. In *IEEE conference on computer vision and pattern recognition*, pp. 130–136.

Patella, M., & Ciaccia, P. (2009). Approximate similarity search: A multi-faceted problem. *Journal of Discrete Algorithms*, *1*(7), 36–48.

Platt, J. (1998). Fast training of support vector machines using sequential minimal optimization. In B. Scholkopf, C. Burges, & A. Smola (Eds.), *Advances in kernel methods, support vector learning* (pp. 41–65). Cambridge, MA: MIT Press.

Schilling, M. F. (1986). Mutual and shared neighbor probabilities: Finite and infinite dimensional results. *Advances in Applied Probability*, *18*, 388–405.

Serafini, T., Zanghirati, G., & Zanni, L. (2003). Gradient projection methods for quadratic programs and applications in training support vector machines. *Optimization Methods and Software*, *20*, 353–378.

Serafini, T., & Zanni, L. (2005). On the working set selection in gradient-based descomposition techniques for support vector machines. *Optimization Methods and Software*, *20*, 586–593.

Uribe, R., Navarro, G., Barrientos, R. J., Marín, M. (2006). An index data structure for searching in metric space databases. In *Proceeding of international conference on computational science 2006 (ICC 2006). Lecture notes in computer science* (Vol. 3991, pp. 611–617). Springer.

Vapnik, V. (1995). *The nature of statistical learning theory*. New York: Springer.

Woodsend, K., & Gondzio, J. (2011). Exploiting separability in large scale linear support vector machine training. *Computational Optimization and Applications*, *49*, 241–269.

Wu, K.-P., & Wang, S.-D. (2009). Choosing the kernel parameters for support vector machines by the inter-cluster distance in the feature space. *Pattern Recognition*, *42*, 710–717.

Zanni, L. (2006). An improved gradient projection-based decomposition techniques for support vector machines. *Computational Management Science*, *3*, 131–145.

Zarruk, D. (2012) *Resolución del problema de Máquinas de Vectores de Soporte mediante búsqueda de k-vecinos más cercanos*. Undergraduate mathematics thesis, Departamento de Matemáticas, Universidad de los Andes, Bogotá, Colombia.