Advanced Review

Support vector machines

Alessia Mammone,[1] Marco Turchi[2] and Nello Cristianini[2]*

Support vector machines (SVMs) are a family of machine learning methods, originally introduced for the problem of classification and later generalized to various other situations. They are based on principles of statistical learning theory and convex optimization, and are currently used in various domains of application, including bioinformatics, text categorization, and computer vision. © 2009 John Wiley & Sons, Inc. *WIREs Comp Stat* 2009 1 283–289

Support vector machines (SVMs), introduced by Vapnik and coworkers in the 1990s,[1] are a family of algorithms for learning two-class discriminant functions from a set of training examples. They have been applied for a wide range of areas including text categorization, handwriting recognition, face detection, gene-expression data analysis, protein homology prediction, and many others.[2–8] The method combines ideas from statistical learning theory[9] and convex optimization, to find a suitable boundary in data space to separate the two classes of points. Extensions of the methods have been developed by a vast research community, and include methods for regression, clustering, factor analysis, based on the same design principles.

LINEAR SVMS FOR CLASSIFICATION

As a family of algorithms, SVMs can address different learning tasks; here, we describe the case of two-class discrimination, while further extensions are discussed later on.

Briefly, the aim of support vector classification is to search efficiently for a 'good' separating hyperplane in a high-dimensional feature space, 'good' in sense of some measure of generalization performance. The generalization performance of SVMs is discussed later on, but the most popular bound on error rates relies on the concept of *margin* that is the minimal distance between the hyperplane separating the two classes and the closest data points to the hyperplane. The optimal hyperplane is therefore defined as the one with the maximal margin of separation between the two classes. In the following paragraphs, we will describe how this optimal hyperplane can be found.

*Correspondence to: nello@support-vector.net

[1]Department of Statistics, Probability and Applied Statistics, 'Sapienza' University of Rome, Rome 00185, Italy

[2]Department of Engineering Mathematics, University of Bristol, Bristol BS81TR, UK

DOI: 10.1002/wics.049

The Separable Case: The Maximal Margin Classifier

We will start with the simplest model of SVM, which is also the first introduced,[1] called *maximal margin classifier* or *hard margin SVM*. Although it is not often used in real-world situations because it works only on linearly separable data, it is a building block for more complex SVM and it is easy to understand.

Suppose that we have a linearly separable training sample $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\} \subseteq (X \times Y)^\ell$, where $X \subseteq \mathbb{R}^n$ denotes the input space, $Y = \{-1, +1\}$ the output domain for binary classification, ℓ the total number of data points, and $\mathbf{x}_i$ and y_i are their labels.

Support vector classifiers are based on the class of hyperplanes $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$ with $\mathbf{w}, \mathbf{x} \in \mathbb{R}^n$, $b \in \mathbb{R}$, corresponding to decision functions $f(\mathbf{x}) = \text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle + b)$. The points $\mathbf{x}$ which lie on the hyperplane satisfy $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$, where $\mathbf{w}$ defines a direction perpendicular to the hyperplane, while varying the value of b moves the hyperplane parallel to itself (Figure 1).

We assume that at the closest points to the boundary, the decision function has value ± 1 because we can always reduce to this case by rescaling the vector $\mathbf{w}$ in the decision function. For two generic points, $\mathbf{x}^+$ and $\mathbf{x}^-$ belonging, respectively, to the positive and negative classes closest to the hyperplane, it is true that:

$$\langle \mathbf{w}, \mathbf{x}^+ \rangle + b = 1, \quad (1)$$

$$\langle \mathbf{w}, \mathbf{x}^- \rangle + b = -1, \quad (2)$$

$$\gamma = \left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|}, (\mathbf{x}^+ - \mathbf{x}^-) \right\rangle = \frac{2}{\|\mathbf{w}\|}, \quad (3)$$

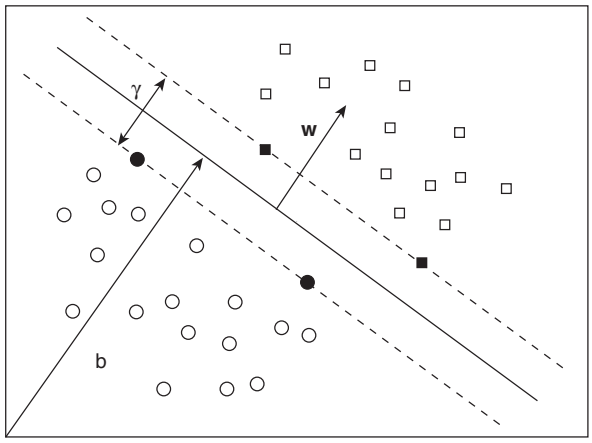

FIGURE 1 | A separating hyperplane for two-dimensional data points; the circles and the squares represent, respectively, data points in classes -1 and $+1$. On the dotted lines, in black, lie the support vectors. The margin γ, b, and $\mathbf{w}$ are also reported.

hence, the margin γ is inversely proportional to the two-norm of $\mathbf{w}$ (in this article, we denote the two-norm with the standard symbol $\|\cdot\|$). Therefore, for the linearly separable case, the support vector algorithm simply looks for the separating hyperplane with largest margin and this can be achieved by minimizing the two-norm of $\mathbf{w}$. This can be formulated by the following optimization problem:

$$\begin{aligned} &\min \ \langle \mathbf{w}, \mathbf{w} \rangle \\ &\text{s.t.} \ \ y_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b) - 1 \geq 0. \end{aligned} \tag{4}$$

This system can be solved by the primal Lagrangian formulation of the problem:

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle - \sum_{i=1}^{\ell} \alpha_i [y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1], \tag{5}$$

where $\alpha_i \geq 0 \quad i = 1, \ldots, \ell$ are the Lagrange multipliers, one for each of the inequality constraints (4).

Convexity. The minimization of a quadratic objective function under linear inequality constraints is a convex optimization problem; this means that we can equivalently solve its corresponding dual problem.

The corresponding dual of Eq. (5) can be obtained by imposing the following optimality conditions:

$$\frac{L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{\ell} y_i \alpha_i \mathbf{x}_i = 0, \tag{6}$$

$$\frac{L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial b} = \sum_{i=1}^{\ell} y_i \alpha_i = 0, \tag{7}$$

and solving the previous equations we obtain:

$$\mathbf{w} = \sum_{i=1}^{\ell} \alpha_i y_i \mathbf{x}_i, \tag{8}$$

$$0 = \sum_{i=1}^{\ell} \alpha_i y_i. \tag{9}$$

Substituting them into Eq. (5), we obtain the optimization problem with linear constraints:

$$\max \ W(\boldsymbol{\alpha}) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \tag{10}$$

$$\text{s.t.} \ \ \sum_{i=1}^{\ell} \alpha_i y_i = 0, \tag{11}$$

where $\alpha_i \geq 0 \quad i = 1, \ldots, \ell$, that is the Lagrangian dual problem associated with Eq. (5).

Since this problem is convex, the Karush–Kuhn–Tucker complementarity conditions are necessary and sufficient[10] to find the optimal solution $(\boldsymbol{\alpha}^*, \mathbf{w}^*, b^*)$ of the primal; the solution must satisfy:

$$\alpha_i^* [y_i(\langle \mathbf{w}^*, \mathbf{x}_i \rangle + b^*) - 1] = 0, \quad i = 1, \ldots, \ell. \tag{12}$$

As the previous equation holds if $\alpha_i^* = 0$ or if $y_i(\langle \mathbf{w}^*, \mathbf{x}_i \rangle + b^*) = 1$, only data points $\mathbf{x}_i$ that lie closest to the hyperplane have a non-zero α_i^*. Such points, called *support vectors* (hence the name of the entire approach) are the critical elements of the training sample, because they contain all the information necessary to reconstruct the hyperplane. If the other training points were removed from the training sample (or moved around, so as not to approach the separating hyperplane), and training was repeated, the same maximal separating hyperplane would be found. This implies that points that are not support vectors can be removed without loss of information about the classification function.

Sparseness. The property that only a subset of α_i^* is non-zero is known as sparseness and has many consequences, both in the implementation and in the analysis of the algorithm.

Dual representation. Notice that reformulating the problem in the dual form, the training data only appear as inner product between vectors. This is a crucial property which allows the procedure to be extended to the nonlinear case, as we will see later on.

The Non-Separable Case: The Soft-Margin Classifier

As the maximal margin algorithm produces a hypothesis that is perfectly consistent with the training data, it cannot be applied to the many real-world datasets, where, because of the noise, a linear separation between classes cannot be found. This problem motivates the development of a more robust version of the algorithm introduced by Cortes and Vapnik[11] that can tolerate noise and outliers in the data without drastically altering the solution.

To this aim, we can allow some misclassifications on the training data by relaxing the constraints in Eq. (4) but only when it is necessary. So we need to introduce a further cost (i.e., an increase in the primal objective function) for doing so. This can be done by introducing positive slack variables ξ_i, in the constraints which then become:

$$y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) - 1 \geq 1 - \xi_i \qquad i = 1, \ldots, \ell \quad (13)$$

$$\xi_i \geq 0 \qquad i = 1, \ldots, \ell. \quad (14)$$

Thus, for an error to occur, the corresponding ξ_i must exceed unity, so $\sum_i \xi_i$ is an upper bound on the number of training errors. A natural way to assign an extra cost for errors is to change the objective function to be minimized from $\langle \mathbf{w}, \mathbf{w}\rangle$ to $\langle \mathbf{w}, \mathbf{w}\rangle + C\sum_{i=1}^{\ell} \xi_i^k$ where C is a parameter to be chosen by the users, depending on how much they want to penalize misclassification errors, meaning, for example, that a larger C corresponds to assign a higher penalty to errors. The optimization problem to be solved is now:

$$\min \ \langle \mathbf{w}, \mathbf{w}\rangle + C\sum_{i=1}^{\ell} \xi_i^k \quad (15)$$

$$\text{s.t.} \ \ y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) \geq 1 - \xi_i, \quad \forall i, \quad (16)$$

which is again a convex problem for any positive integer $k \in \mathbb{N}^+$; for $k = 1, 2$ it is also a quadratic programming problem. Note also that for $k = 0$, the extra cost term added to the objective function to be minimized simply counts the number of data points but does not lead to a convex optimization problem (see Cristianini and Shawe-Taylor[12]) for derivation. Let us write for $k = 2$, the primal Lagrangian of Eq. (16):

$$L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \frac{1}{2}\langle \mathbf{w}, \mathbf{w}\rangle + \frac{C}{2} - \sum_{i=1}^{\ell} \xi_i^2 - \sum_{i=1}^{\ell} \alpha_i [y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) - 1 + \xi_i^2]. \quad (17)$$

The corresponding dual is found by differentiating with respect to $\mathbf{w}, \boldsymbol{\xi}$, and b, and imposing stationarity:

$$\frac{L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{\ell} y_i \alpha_i \mathbf{x}_i = 0, \quad (18)$$

$$\frac{L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha})}{\partial \boldsymbol{\xi}} = C\boldsymbol{\xi} - \boldsymbol{\alpha} = \mathbf{0}, \quad (19)$$

$$\frac{L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha})}{\partial b} = \sum_{i=1}^{\ell} y_i \alpha_i = 0, \quad (20)$$

and substituting the relations obtained into the primal, the dual objective function is the following:

$$L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2C}\langle \boldsymbol{\alpha}, \boldsymbol{\alpha}\rangle - \frac{1}{2}\sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j\rangle. \quad (21)$$

Hence, minimizing this object over $\boldsymbol{\alpha}$ is equivalent to maximizing

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \left(\langle \mathbf{x}_i, \mathbf{x}_j\rangle + \frac{1}{C}\delta_{ij}\right), \quad (22)$$

where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise is the Kronecker δ. Also this problem is convex, and the Karush–Kuhn–Tucker complementary conditions to find the optimal solution $(\boldsymbol{\alpha}^*, \mathbf{w}^*, b^*, \boldsymbol{\xi}^*)$ are:

$$\alpha_i^* [y_i(\langle \mathbf{w}^*, \mathbf{x}_i\rangle + b^*) - 1 + \xi_i^*] = 0, \quad i = 1, \ldots, \ell. \quad (23)$$

Note that in both $k = 1$ and $k = 2$ cases, the soft-margin classifier effectively reduces the impact of individual points on the solution by limiting the size of Lagrange multipliers α_i.

NONLINEAR SVMS

In general, complex real-world applications require a more expressive hypothesis space than simple linear functions (Figure 2). So we need to introduce more complex functions to deal with real-world data.

First, notice that the only way in which the data appear in the optimization problem previously described is in the form of inner product $\langle \mathbf{x}_i, \mathbf{x}_j\rangle$. To get a potentially better representation of the data, we can map the data points into an alternative higher-dimensional space called feature space through a replacement: $\langle \mathbf{x}_i, \mathbf{x}_j\rangle \rightarrow \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j)\rangle$. The functional form of the mapping $\boldsymbol{\phi}(\mathbf{x})$ does not need to be known

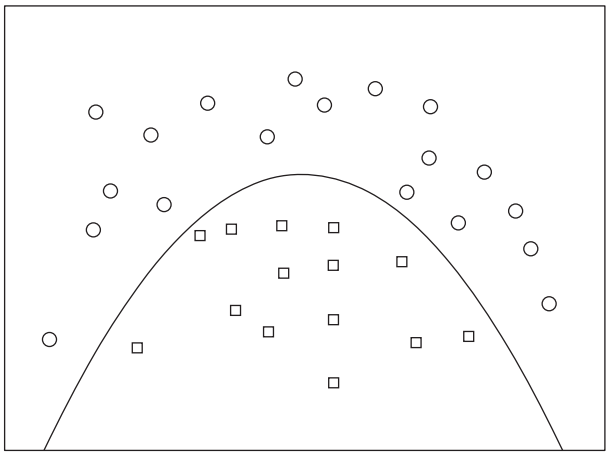

FIGURE 2 | Two-dimensional data points that cannot be linearly separated; the circles and the squares, respectively, represent data points in classes −1 and +1.

because it is implicitly determined by the choice of a *kernel function* that we describe below. By replacing the inner product with an appropriately chosen kernel function, data can become linearly separable in the feature space despite being non-separable in the input space. Ideally, this choice depends on the difficulty in the target function to be learned.

The first step of the kernel approach changes the representation of the data:

$$\mathbf{x} = (x_1, \ldots, x_n) \rightarrow \phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \ldots, \phi_N(\mathbf{x})). \quad (24)$$

This step is equivalent to embedding the input space $X \subseteq \mathbb{R}^n$ into a new feature space that can be defined as

$$F = \{\phi(\mathbf{x}) : \mathbf{x} \in X\}, \quad (25)$$

where $\phi : X \rightarrow F \subseteq \mathbb{R}^N$ is the embedding map and $\mathbf{x}$ a vector containing the feature values. The second step is to detect relations in the feature space, using a robust and efficient linear machine such as the soft-margin SVM.

The dual representation of linear relations highlighted in the previous sections permits the use of the kernel approach, because linear relations can be represented by using inner products $\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$ between all pairs of observed points $\mathbf{x}, \mathbf{z} \in X$ and without explicitly using their coordinates in $\mathbb{R}^N$. The function that returns the inner product between the images of any two data points in the feature space is called *kernel*.

Kernel function. A *kernel* is a function κ that for all $\mathbf{x}, \mathbf{z} \in X$ satisfies

$$\kappa(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle, \quad (26)$$

where ϕ is a mapping $\phi : \mathbf{x} \rightarrow \phi(\mathbf{x}) \in F$. The most important consequence of the dual representation is that the dimension of the feature space does not affect the computation; in fact the feature vector is not represented explicitly, so the number of operations required to compute the inner product in the evaluation of the kernel function is not proportional to the number of features, but rather to the number of data points.

The only information used about the training examples is their so-called *kernel matrix*, defined as the square matrix $\mathbf{K} \in \mathbb{R}^{\ell \times \ell}$ such that $\mathbf{K}_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ for a set of vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\} \subseteq X$ and some kernel function κ.

As we highlighted before, kernel functions are a crucial part of the SVMs method; the following paragraphs will regard the problem of creating such kernel functions.

Making Kernels

First of all, let us determine what properties of a function $\kappa(\mathbf{x}, \mathbf{z})$ are necessary to ensure it is a kernel for some feature space. Clearly, the function must be symmetric:

$$\kappa(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle = \langle \phi(\mathbf{z}), \phi(\mathbf{x}) \rangle = \kappa(\mathbf{z}, \mathbf{x}) \quad (27)$$

and satisfy the Cauchy–Schwarz inequality:

$$\kappa(\mathbf{x}, \mathbf{z})^2 = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle^2 \leq \|\phi(\mathbf{x})\|^2 \|\phi(\mathbf{z})\|^2 \quad (28)$$

$$\leq \langle \phi(\mathbf{x}), \phi(\mathbf{x}) \rangle \langle \phi(\mathbf{z}), \phi(\mathbf{z}) \rangle \leq \kappa(\mathbf{x}, \mathbf{x}) \kappa(\mathbf{z}, \mathbf{z}). \quad (29)$$

These conditions are not sufficient to guarantee that a function $\kappa(\mathbf{x}, \mathbf{z})$ is a kernel, while the following proposition helps us to define another necessary and sufficient condition for a function to be a kernel.

Proposition 1 *Let X be a finite input space with $\kappa(\mathbf{x}, \mathbf{z})$ a symmetric function on X. Then $\kappa(\mathbf{x}, \mathbf{z})$ is a kernel function if and only if the matrix*

$$\mathbf{K} = \kappa(\mathbf{x}_i, \mathbf{x}_j) \qquad i, j = 1, \ldots, \ell \quad (30)$$

is positive semi-definite (has non-negative eigenvalues) for any set of data points $\mathbf{x}_i \in X$.

An equivalent condition to the previous one is given by Mercer's theorem that provides a second characterization of a kernel function, which is most useful when we constructing kernels. More details on this can be found in Shawe-Taylor and Cristianini[13] and Cristianini and Shawe-Taylor.[12]

Operations on Kernel Functions

As we pointed out, the positive semi-definiteness property is the core for the characterization of kernel functions. New functions are kernels if they are finitely positive semi-definite. So it is sufficient to verify that the function is a valid kernel and this demonstrates that there exists a feature space map for which the function computes the corresponding inner product. It is useful to introduce some operations on kernel functions which always give a new positive semi-definite function as result. We will say that the class of kernel functions is *closed* under such operations.

The following two propositions can be viewed as showing that kernels satisfy a number of closure properties, allowing us to create more complex kernels from simpler ones.

Proposition 2 (Closure properties) *Let* κ_1 *and* κ_2 *be kernels over* $X \times X$, $X \subseteq \mathbb{R}^n$, $a \in \mathbb{R}^+$, $f(\cdot)$ *a real-valued function on* X, $\phi : X \longrightarrow \mathbb{R}^N$ *with* κ_3 *a kernel over* $\mathbb{R}^N \times \mathbb{R}^N$, *and* $\mathbf{B}$ *a symmetric positive semi-definite* $n \times n$ *matrix. Then the following functions are kernels:*

1. $\kappa(\mathbf{x}, \mathbf{z}) = \kappa_1(\mathbf{x}, \mathbf{z}) + \kappa_2(\mathbf{x}, \mathbf{z})$
2. $\kappa(\mathbf{x}, \mathbf{z}) = a\kappa_1(\mathbf{x}, \mathbf{z})$
3. $\kappa(\mathbf{x}, \mathbf{z}) = \kappa_1(\mathbf{x}, \mathbf{z})\kappa_2(\mathbf{x}, \mathbf{z})$
4. $\kappa(\mathbf{x}, \mathbf{z}) = f(\mathbf{x})f(\mathbf{z})$
5. $\kappa(\mathbf{x}, \mathbf{z}) = \kappa_3(\boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{z}))$
6. $\kappa(\mathbf{x}, \mathbf{z}) = \mathbf{x}'\mathbf{B}\mathbf{z}$ *with* $\mathbf{x}, \mathbf{z} \in X$

Proposition 3 *Let* $\kappa_1(\mathbf{x}, \mathbf{z})$ *be a kernel over* $X \times X$, *where* $\mathbf{x}, \mathbf{z} \in X$, *and* $p(x)$ *a polynomial with positive coefficients. Then the following functions are also kernels:*

1. *Polynomial kernel:* $\kappa(\mathbf{x}, \mathbf{z}) = p(\kappa_1(\mathbf{x}, \mathbf{z}))$;
2. $\kappa(\mathbf{x}, \mathbf{z}) = \exp(\kappa_1(\mathbf{x}, \mathbf{z}))$;
3. *Gaussian kernel with* $\mathbf{x}, \mathbf{z} \in X$: $\kappa(\mathbf{x}, \mathbf{z}) = \exp\left(-\|\mathbf{x} - \mathbf{z}\|^2/(2\sigma^2)\right)$;

Operating on Non-Vectorial Data

An important advantage of using kernel functions is that they can be defined not only on vectorial data but also on general data types such as strings,[14] graphs,[15] and trees[13], and this allows SVMs and their extensions to operate on very general classes of problems. Kernel methods are also used in biology,[16,17] in probabilistic model[18] and in textual data.[19,20]

GENERALIZATION PERFORMANCE

So far we have highlighted the main characteristics of SVMs but we have no guarantee that they lead to good classification results. In this section, we discuss their generalization performance.

The aim of generalization theory is to describe which factors have to be controlled in the learning machine in order to guarantee good generalization performance. Among the several existing theories, the one introduced by Vapnik and Cervonenkis is the most common to describe SVMs because it motivates them. It follows the style of probably approximately correct (PAC) analysis that attempts to find a bound on the error probability that holds with high confidence. Another possible approach is the Bayesian one that in contrast attempts to choose output values that are most likely based on the observed training values.

It is possible to prove[12] that the generalization error of the hard margin SVM has functional shape:

$$\varepsilon = \widehat{O}\left(\frac{1}{\ell}\left(\frac{R}{\gamma}\right)^2\right), \tag{31}$$

where R is the radius of the ball containing all data points, ℓ the number of data points, and γ the margin.

Another bound on the expected generalization error can be obtained using a leave-one out estimate by removing one of the training points, re-training, and then testing on the removed point; and then repeating

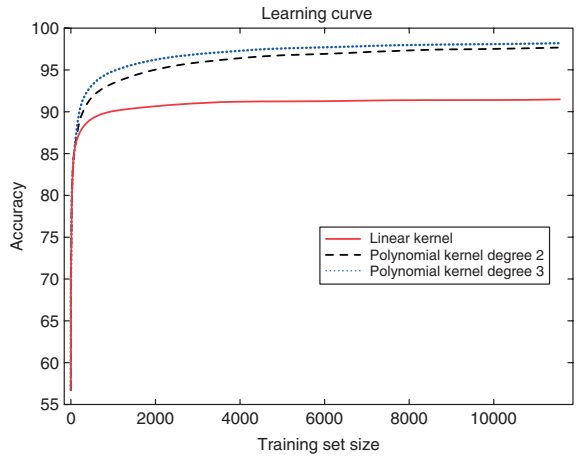

FIGURE 3 | These learning curves have been obtained using SVMs with a linear kernel and two polynomial kernels, respectively, with degrees 2 and 3. Among all experiments, the results relative to digit 8 is reported. This is because it better highlights how different kernel functions can produce different performance.

this, for all training points. From the support vector solution, we know that removing any training points that are not support vectors (the latter include the errors) will have no effect on the hyperplane found. Thus, the worst that can happen is that every support vector will become an error. Taking the expectation over all such training sets therefore gives an upper bound on the actual risk, that is for training sets of size ℓ

$$E(\varepsilon) = \frac{\sharp \mathtt{sv}}{\ell} \tag{32}$$

The bounds described do not depend on the dimension of the input but depend on quantities measured by the training algorithm such as γ and $\sharp \mathtt{sv}$.[12]

Also the choice of kernel is important for the generalization performance of a SVM. Consider, for example, the next experiment where we build three different learning curves using three different kernels. A learning curve shows how the behavior of the generalization error decreases as the number of training examples increases. It represents how fast the performance of a learning machine is improved by learning from examples.

Figure 3 shows the learning curve of SVMs with various kernels using the MNIST data.[21] It reports accuracy of binary classification for digit 8 versus rest of digits, averaged over 40 bootstraps for each training set size. All the experiments are computed using SVMLight[22] with default parameters and $C = 10$ (see Eq. 15). This value was taken from earlier work,[23] no additional model selection was performed.

Given the complexity of the feature space, the polynomial kernel of degree 3 achieves a better separation of the data resulting in a higher accuracy.

EXTENSIONS OF THE METHOD

In the previous sections we have discussed the classification task, but SVMs and kernel methods can be extended to regression, density estimation, clustering (identifying groups of 'similar' data points), novelty detection (detecting new data points that deviate from the normal), ranking (learning a ranking function from ranked examples), and transduction (the problem of completing the labeling of a partially labeled dataset).

In all these cases, SVMs still maintain the main features that characterize the maximal margin algorithm: a nonlinear function is learned by a linear machine in a kernel-induced feature space; the problem to be solved can be reduced to maximizing a convex quadratic form subject to linear constraints having no local maxima and its solution can always be found efficiently; furthermore, the solution is still sparse.

The only step that is data-dependent is the choice of the kernel and the corresponding feature space; this must generally be made by a human user. Recent work[24] shows how it is possible to learn a kernel from a large kernel class in a convex way. The method is based on semi-definite programming hence once more on convex optimization.

CONCLUSION

The availability of reliable classification algorithms, with clear computational and statistical properties, has marked a significant progress in the field of pattern analysis. The previous technology, neural networks, suffered from several practical drawbacks that have been addressed by kernel methods, particularly because of their convexity in the training algorithm, the modularity of the algorithm, and their natural applicability to structured data.

REFERENCES

1. Boser BE, Guyon IM, Vapnik VN. A training algorithm for optimal margin classifiers. In *Proceedings of fifth COLT*. Pittsburgh, PA, 1992, 144–152.
2. Joachims T, Text categorization with support vector machines: learning with many relevant features. In *Proceedings of ECML*. Chemnitz, Germany, 1998, 137–142.
3. Decoste D, Schölkopf B. Training invariant support vector machines. *Mach Learn* 2002, 46(1–3):161–190.
4. Brown MP, Grundy WN, Lin D, Cristianini N, Sugnet CW, et al. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proc Natl Acad Sci U S A* 2000, 97(1):262–267.
5. Lanckriet GR, De Bie T, Cristianini N, Jordan MI, Noble WS. A statistical framework for genomic data fusion. *Bioinformatics* 2004, 20(16):2626–2635.
6. Bradford JR, Westhead DR. Improved prediction of protein-protein binding sites using a support

vector machines approach. *Bioinformatics* 2005, 8(21):1487–1494.

7. Zhang S, Pan Q, Zhang H, Zhang Y, Wang H. Classification of protein quaternary structure with support vector machine. *Bioinformatics* 2003, 19(18):2390–2396.
8. Lei Z, Dai Y. An SVM-based system for predicting protein subnuclear localizations. *BMC Bioinformatics* 2005, 6(1):291–298.
9. Vapnik VN. *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.
10. Fletcher R. *Practical Methods of Optimization*. 2nd ed. New York: John Wiley & Sons; 1987.
11. Cortes C, Vapnik V. Support vector networks. *Mach Learn* 1995, 20(3):273–297.
12. Cristianini N, Shawe-Taylor J. *An Introduction to Support Vector Machines*. Cambridge: Cambridge University Press; 2000. url: www.support-vector.net.
13. Shawe-Taylor J, Cristianini N. *Kernel Methods for Pattern Analysis*. Cambridge: Cambridge University Press; 2004. url: www.kernel-methods.net.
14. Leslie CS, Kuang R. Fast kernels for inexact string matching. In *Proceedings of COLT*, Washington, DC, 2003, 114–128.
15. Kondor RI, Lafferty J. Diffusion kernels on graphs and other discrete structures. In *Proceedings of 19th ICML*, Sydney, Australia, 2002, 315–322.
16. Jaakkola T, Diekhans M, Haussler D. Using the fisher kernel method to detect remote protein homologies. In *Proceedings of seventh ISMB*, Heidelberg, Germany, 1999, 149–158.
17. Kashima H, Tsuda K, Inokuchi A. Kernels for Graphs. In: Schölkopf B, Tsuda K, Vert JP, eds. Kernel methods in computational biology. *Handbook of Computational Geometry for Pattern Recognition*. Cambridge, MA: MIT, 2004.
18. Tsuda K, Kawanabe M, Ratsch G, Sonnenburg S, Mller KR. A new discriminative kernel from probabilistic models. *Neural Comput* 2002, 14(10):2397–2414.
19. Vinokourov A, Shawe-Taylor J, Cristianini N. Inferring a semantic representation of text via cross-language correlation analysis. In *NIPS*, Vancouver, Canada, 2002, 1473–1480.
20. Vert JP, Kanehisa M. Graph-driven features extraction from microarray data using diffusion kernels and kernel CCA. In *NIPS*, Vancouver, Canada, 2003, 1425–1432.
21. Lecun Y, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, Taipei, Taiwan, 1998, 2278–2324.
22. Joachims T. Making large-scale support vector machine learning practical. In: Schölkopf CB, Burges CJ, Smola AJ, eds. *Advances in Kernel Methods: Support Vector Learning*. Cambridge, MA: MIT Press; 1999, 169–184.
23. Schölkopf B, Burges C, Vapnik V. Extracting support data for a given task. In *Proceedings of first International Conference on Knowledge Discovery and Data Mining*; 1995, 252–257.
24. Lanckriet GR, Cristianini N, Bartlett P, El Ghaoui L, Jordan MI. Learning the kernel matrix with semidefinite programming. *J Mach Learn Res* 2004, 5(16): 27–72.
25. Schölkopf B, Smola A. *Learning with Kernels*. Cambridge, MA: MIT Press, 2002.
26. De Bie T, Cristianini N, Rosipal R. Eigenproblems in pattern recognition. In: Bayro-Corrochano E., ed. *Handbook of Computational Geometry for Pattern Recognition, Computer Vision, Neurocomputing and Robotics*. New York: Springer-Verlag; 2004.
27. Vapnik VN. *Statistical Learning Theory*. 2nd ed. New York: Wiley-Interscience; 1999.

FURTHER READING

Comprehensive introductions include the introductory textbook,[12] whereas a general framework of kernels is examined in Shawe-Taylor and Cristianini[13] and Schölkopf and Smola.[25] We refer the reader to De Bie et al.[26] for a tutorial on kernel methods based on eigenvalue problems. Many websites are also available, with free software and pointers to recent publications in the field. In particular, `www.kernel-methods.net` and `www.support-vector.net` contain free material and software, whereas `www.kernel-machines.org` contains updated pointers to all main events in the kernel methods community. VC theory also known as statistical learning theory is extensively described by Vapnik.[27]

