ELSEVIER Neurocomputing 48 (2002) 63–84

NEUROCOMPUTING

www.elsevier.com/locate/neucom

Kernel methods: a survey of current techniques

Colin Campbell

Department of Engineering Mathematics, Bristol University, Bristol BS8 1TR, UK

Received 1 November 2000; accepted 6 June 2001

Abstract

Kernel methods have become an increasingly popular tool for machine learning tasks such as classification, regression or novelty detection. They exhibit good generalization performance on many real-life datasets, there are few free parameters to adjust and the architecture of the learning machine does not need to be found by experimentation. In this tutorial, we survey this subject with a principal focus on the most well-known models based on kernel substitution, namely, support vector machines. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Kernel methods; Machine learning tasks; Architecture of learning machine; Support vector machines

1. Introduction

Support vector machines (SVMs) have been successfully applied to a number of applications ranging from particle identification, face identification and text categorization to engine-knock detection, bioinformatics and database marketing [17]. The approach is systematic and properly motivated by statistical learning theory [58]. Training involves optimization of a convex cost function: there are no local minima to complicate the learning process. The approach has many other benefits, for example, the model constructed has an explicit dependence on a subset of the datapoints (the support vectors), hence interpretation is straightforward and data cleaning [16] could be implemented to improve performance. SVMs are the most well known of a class of algorithms which use the idea of kernel substitution and which we will broadly refer to as *kernel methods*.

In this tutorial, we introduce this subject, describing the application of kernel methods to classification, regression and novelty detection and the different

E-mail address: c.campbell@bris.ac.uk (C. Campbell).

0925-2312/02/$ - see front matter © 2002 Elsevier Science B.V. All rights reserved.
PII: S0925-2312(01)00643-9

optimization techniques that may be used during training. This tutorial is not exhaustive and many alternative kernel-based approaches (e.g. kernel PCA [42], density estimation [64], etc) have not been considered here. More thorough treatments are contained in the books by Cristianini and Shawe-Taylor [11], Vapnik's classic textbook on statistical learning theory [58], recent edited volumes [41,49] and a special issue of *Machine Learning* [9].

2. Learning with support vectors

To introduce the subject we will begin by outlining the application of SVMs to the simplest case of binary classification. From the perspective of statistical learning theory the motivation for considering binary classifier SVMs comes from theoretical bounds on the generalization error [58,11] (the theoretical generalization performance on new data). These generalization bounds have two important features (Appendix A). Firstly, the upper bound on the generalization error does not depend on the dimension of the space. Secondly, the error bound is minimized by maximizing the *margin*, γ, i.e. the minimal distance between the hyperplane separating the two classes and the closest datapoints to the hyperplane (Fig. 1).

Let us consider a binary classification task with datapoints $\mathbf{x}_i$ $(i = 1, \ldots, m)$ having corresponding labels $y_i = \pm 1$ and let the decision function be

$$f(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b). \tag{1}$$

If the dataset is separable then the data will be correctly classified if $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) > 0 \; \forall i$. Clearly this relation is invariant under a positive rescaling of the argument inside the *sign*-function, hence we implicitly define a scale for $(\mathbf{w}, b)$ to give *canonical hyperplanes* such that $\mathbf{w} \cdot \mathbf{x} + b = 1$ for the closest points on one side and $\mathbf{w} \cdot \mathbf{x} + b = -1$ for the closest on the other side. For the separating hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$ the normal vector is clearly $\mathbf{w}/\|\mathbf{w}\|_2$, and hence the margin can be found from the projection of $\mathbf{x}_1 - \mathbf{x}_2$ onto this vector (see Fig. 1). Since $\mathbf{w} \cdot \mathbf{x}_1 + b = 1$ and $\mathbf{w} \cdot \mathbf{x}_2 + b = -1$ this means the margin is $\gamma = 1/\|\mathbf{w}\|_2$. To maximize

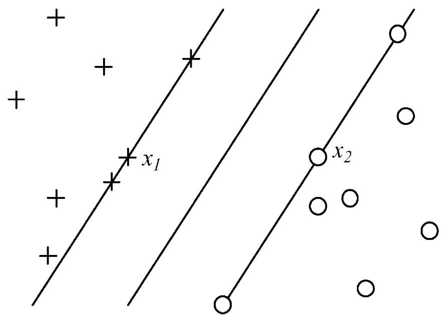

Fig. 1. The perpendicular distance between the separating hyperplane and a hyperplane through the closest points (the support vectors) is called the *margin*. $\mathbf{x}_1$ and $\mathbf{x}_2$ are examples of *support vectors* of opposite sign.

the margin the task is therefore

$$\text{minimize} \quad g(\mathbf{w}) = \tfrac{1}{2}\|\mathbf{w}\|_2^2 \tag{2}$$

subject to the constraints

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geqslant 1 \quad \forall i \tag{3}$$

and the learning task can be reduced to minimization of the primal lagrangian

$$L = \tfrac{1}{2}(\mathbf{w} \cdot \mathbf{w}) - \sum_{i=1}^{m} \alpha_i (y_i(\mathbf{w} \cdot \mathbf{x} + b) - 1), \tag{4}$$

where α_i are Lagrangian multipliers (hence $\alpha_i \geqslant 0$). From Wolfe's theorem [66] we can take the derivatives with respect to b and $\mathbf{w}$ and resubstitute back in the primal to give the Wolfe dual lagrangian

$$W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \tag{5}$$

which must be maximized with respect to the α_i subject to the constraints

$$\alpha_i \geqslant 0 \quad \sum_{i=1}^{m} \alpha_i y_i = 0. \tag{6}$$

So far we have not used the second feature implied by the generalization theorem mentioned above: the bound does not depend on the dimensionality of the space. For the dual lagrangian (5) we notice that the datapoints, $\mathbf{x}_i$, only appear inside an inner product. To get a better representation of the data we can therefore map the datapoints into an alternative higher-dimensional space, called *feature space*, through a replacement

$$\mathbf{x}_i \cdot \mathbf{x}_j \rightarrow \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j). \tag{7}$$

The functional form of the mapping $\phi(\mathbf{x}_i)$ does not need to be known since it is implicitly defined by the choice of *kernel*: $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$ or inner product in feature space (feature space must therefore be a pre-Hilbert or inner product space). With a suitable choice of kernel the data can become separable in feature space despite being non-separable in the original input space: hence kernel substitution provides a route for obtaining non-linear algorithms from algorithms previously restricted to handling linearly separable datasets. Thus, for example, whereas data for n-parity or the two spirals problem is non-separable by a hyperplane in input space it can be separated in the feature space defined by RBF kernels (giving an RBF-type network)

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathrm{e}^{-(\mathbf{x}_i - \mathbf{x}_j)^2/2\sigma^2}. \tag{8}$$

Many other choices for the kernel function are possible, e.g.

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d \quad K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta \mathbf{x}_i \cdot \mathbf{x}_j + b) \tag{9}$$

defining polynomial and feedforward neural network classifiers. Indeed, the class of mathematical objects which can be used as kernels is very general, and includes scores produced by dynamic-alignment algorithms [18,62], for example. Suitable kernels must satisfy a mathematical condition called Mercer's theorem [30] which we describe further in Appendix B (the tanh kernel above only satisfies Mercer's condition for certain values of β and b).

For binary classification with the given choice of kernel the learning task therefore involves maximization of the lagrangian

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

subject to the constraints (6). The associated Karush-Kuhn-Tucker (KKT) conditions are

$$\begin{aligned} y_i(\mathbf{w}\cdot\mathbf{x}_i+b)-1 &\geqslant 0 \quad \forall i \\ \alpha_i &\geqslant 0 \quad \forall i \\ \alpha_i(y_i(\mathbf{w}\cdot\mathbf{x}_i+b)-1) &= 0 \quad \forall i \end{aligned} \tag{11}$$

which are always satisfied when a solution is found. After the optimal values of α_i have been found the decision function is based on the sign of

$$f(\mathbf{z})=\sum_{i=1}^{m}y_i\alpha_i K(\mathbf{z},\mathbf{x}_i)+b. \tag{12}$$

Since the bias, b, does not feature in the above dual formulation it is found from the primal constraints

$$b=-\frac{1}{2}\left[\max_{\{i|y_i=-1\}}\left(\sum_{j=1}^{m}y_j\alpha_j K(\mathbf{x}_i,\mathbf{x}_j)\right)+\min_{\{i|y_i=+1\}}\left(\sum_{j=1}^{m}y_j\alpha_j K(\mathbf{x}_i,\mathbf{x}_j)\right)\right]. \tag{13}$$

We will henceforth refer to such a solution (α_i, b) as a *hypothesis* modelling the data. When the maximal margin hyperplane is found in feature space, only those points which lie closest to the hyperplane have $\alpha_i>0$ and these points are the *support vectors*. All other points have $\alpha_i=0$. This means that the representation of the hypothesis is given solely by those points which are closest to the hyperplane. For some hypotheses most of the datapoints may be support vectors and we refer to these as *dense* hypotheses. If a small fraction of the datapoints are support vectors then we call the hypothesis *sparse*.

Many problems involve multiclass classification and a number of schemes have been outlined [29,63] (with broadly similar performance). One of the simplest schemes is to use a directed acyclic graph (DAG) with the learning task reduced to binary classification at each node [36]. Thus if we consider a 3-class classification problem (Fig. 2) the first node in the DAG decides for classification with label 1 or 3, say, with the next step being classifications 1 against 2 or 2 against 3 depending on the outcome of the first decision. This scheme is adequate for small multi-class classification problems, whereas for larger problems it is possible to generalize the

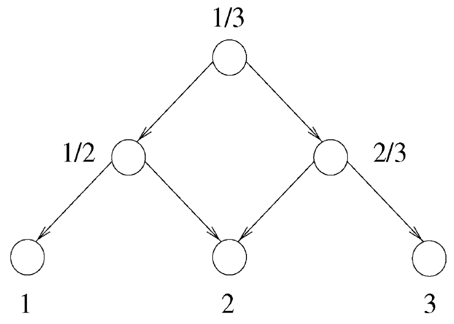

Fig. 2. A multi-class classification problem can be reduced to a series of binary classification tasks.

above binary classification model to maximization of a dual lagrangian for multiple separating hyperplanes [63] or one could simply use a series of one-against-all classifiers.

2.1. Soft margins and allowing for training errors

Most real-life datasets contain noise and an SVM can fit this noise leading to poor generalization. The effect of outliers and noise can be reduced by introducing a *soft margin* [8] and two schemes are currently used (see also Appendix A). In the first (L_1 error norm) the learning task is the same as in (10, 6) except for the introduction of the box constraint

$$0 \leqslant \alpha_i \leqslant C \tag{14}$$

while in the second (L_2 error norm) the learning task is (10, 6) except for addition of a small positive constant to the leading diagonal of the kernel matrix [8,47]

$$K(\mathbf{x}_i, \mathbf{x}_i) \leftarrow K(\mathbf{x}_i, \mathbf{x}_i) + \lambda. \tag{15}$$

C and λ control the trade-off between training error and generalization ability and are chosen by means of a validation set. The effect of these soft margins is illustrated in Fig. 3 for the ionosphere dataset from the UCI Repository [65].

The justification for these soft-margin techniques comes from statistical learning theory but can be readily viewed as relaxation of the *hard margin* constraint (3). Thus for the L_1 error norm (and prior to introducing kernels) we introduce a positive slack variable ξ_i into (3)

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geqslant 1 - \xi_i \tag{16}$$

and the task is now to minimize the sum of errors $\sum_{i=1}^{m} \xi_i$ in addition to $\|\mathbf{w}\|^2$

$$\min \left[\tfrac{1}{2}\mathbf{w} \cdot \mathbf{w} + C \sum_{i=1}^{m} \xi_i \right]. \tag{17}$$

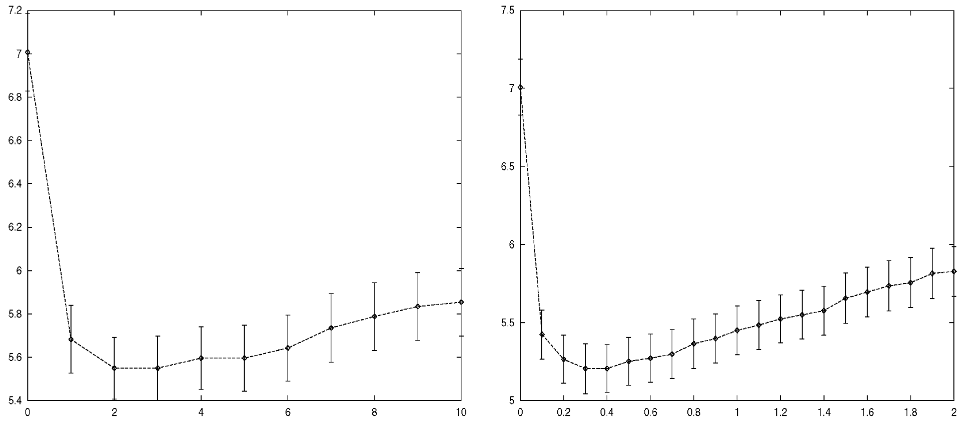

Fig. 3. *Left*: Test error as a percentage (y-axis) versus C (x-axis) and *Right*: test error as a percentage (y-axis) versus λ (x-axis) for soft margin classifiers based on L_1 and L_2 error norms, respectively. The UCI ionosphere dataset was used with RBF kernels ($\sigma = 1.5$) and 100 samplings of the data.

This is readily formulated as a primal objective function

$$L(\mathbf{w}, b, \alpha, \xi) = \frac{1}{2}\mathbf{w} \cdot \mathbf{w} + C\sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} \alpha_i[y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i] - \sum_{i=1}^{m} r_i \xi_i \tag{18}$$

with Lagrange multipliers $\alpha_i \geqslant 0$ and $r_i \geqslant 0$. The derivatives with respect to $\mathbf{w}$, b and ξ give

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i = 0, \tag{19}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{m} \alpha_i y_i = 0, \tag{20}$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - r_i = 0. \tag{21}$$

Resubstituting these back in the primal-objective function we obtain the same dual-objective function, (10), as before. However, $r_i \geqslant 0$ and $C - \alpha_i - r_i = 0$, hence $\alpha_i \leqslant C$ and the constraint $0 \leqslant \alpha_i$ is replaced by $0 \leqslant \alpha_i \leqslant C$. Patterns with values $0 < \alpha_i < C$ will be referred to as *non-bound* and those with $\alpha_i = 0$ or $\alpha_i = C$ will be said to be *at bound*. For an L_1 error norm we find the bias in the decision function (12) from the KKT conditions for the soft margin case. In particular $r_i\xi_i = 0$ and $\alpha_i(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i) = 0$, hence if we select a *non-bound* datapoint i (such that $0 < \alpha_i < C$) we find from $C - \alpha_i - r_i = 0$ that $r_i > 0$ hence $\xi_i = 0$, and since $\alpha_i > 0$ so can determine b from $b = y_i - \mathbf{w} \cdot \mathbf{x}_i$ assuming $y_i = \pm 1$.

The optimal value of C must be found by experimentation using a validation set and it cannot be readily related to the characteristics of the dataset or model. In an alternative approach (ν-SVM [45]) it can be shown that solutions for an L_1-error norm are the same as those obtained from maximizing

$$W(\alpha) = -\tfrac{1}{2}\sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{22}$$

subject to

$$\sum_{i=1}^{m} y_i \alpha_i = 0 \quad \sum_{i=1}^{m} \alpha_i \geqslant \nu \quad 0 \leqslant \alpha_i \leqslant \frac{1}{m}, \tag{23}$$

where ν lies on the range 0 to 1. The fraction of training errors is upper bounded by ν and ν also provides a lower bound on the fraction of points which are support vectors. Hence in this formulation the conceptual meaning of the soft margin parameter is more transparent.

For the L_2 error norm the primal objective function is

$$L(\mathbf{w}, b, \alpha, \xi) = \tfrac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum_{i=1}^{m} \xi_i^2 - \sum_{i=1}^{m} \alpha_i [y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i] - \sum_{i=1}^{m} r_i \xi_i \tag{24}$$

with $\alpha_i \geqslant 0$ and $r_i \geqslant 0$. After obtaining the derivatives with respect to $\mathbf{w}$, b and ξ, substituting for $\mathbf{w}$ and ξ in the primal objective function and noting that the dual objective function is maximal when $r_i = 0$, we obtain the following dual objective function after kernel substitution

$$W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) - \frac{1}{4C}\sum_{i=1}^{m} \alpha_i^2. \tag{25}$$

With $\lambda = 1/2C$ this gives the same dual objective function as for hard margin learning except for the substitution (15). For many real-life datasets there is an imbalance between the amount of data in different classes, or the significance of the data in the two classes can be quite different. For example, for the detection of tumours on MRI scans it may be best to allow a higher number of false positives if this improved the true positive detection rate. The relative balance between the detection rate for different classes can be easily shifted [59] by introducing *asymmetric soft margin parameters*. Thus for binary classification with an L_1 error norm $0 \leqslant \alpha_i \leqslant C_+$ $(y_i = +1)$, and $0 \leqslant \alpha_i \leqslant C_-(y_i = -1)$, while $K(\mathbf{x}_i, \mathbf{x}_i) \leftarrow K(\mathbf{x}_i, \mathbf{x}_i) + \lambda_+$ (if $y_i = +1$) and $K(\mathbf{x}_i, \mathbf{x}_i) \leftarrow K(\mathbf{x}_i, \mathbf{x}_i) + \lambda_-$ (if $y_i = -1$) for the L_2 error norm.

2.2. *A linear programming approach to classification*

Rather than using quadratic programming it is also possible to derive a kernel classifier in which the learning task involves *linear programming* (*LP*) instead. Training the classifier involves the minimization

$$\min \left[\sum_{i=1}^{m} \alpha_i + C\sum_{i=1}^{m} \xi_i \right] \tag{26}$$

subject to

$$y_i\left[\sum_{j=1}^{m}\alpha_i K(x_i,x_j)+b\right]\geqslant 1-\xi_i, \tag{27}$$

where $\alpha_i \geqslant 0$ and $\xi_i \geqslant 0$. By minimizing $\sum_{i=1}^{m}\alpha_i$ we could obtain a solution which is *sparse* i.e. relatively fewer datapoints are used. Furthermore, efficient simplex or column-generation techniques exist for solving linear programming problems so this is a practical alternative to conventional QP SVMs. This linear programming approach evolved independently of the QP approach to SVMs [27] and, as we will see, linear programming approaches to regression and novelty detection are also possible. It is also possible to handle multi-class problems using linear programming [63].

2.3. Model selection

Apart from the choice of kernel, the other indeterminate is the choice of the kernel parameter (e.g. σ in (8)). The kernel parameter can be found using cross-validation if sufficient data is available. However, recent model-selection strategies can give a reasonable estimate for the kernel parameter without use of additional validation data. As a first attempt we can use a theorem stating that the generalization error bound is reduced as the margin γ is increased. This theorem gives the upper bound as $R^2/m\gamma^2$ where R is the radius of the smallest ball containing the training data. At an optimum of (10) it is possible to show that $\gamma^2 = 1/\sum_i \alpha_i^0$ (where α_i^0 are the values of α_i at the optimum). Also for RBF kernels it is frequently the case that $R \simeq 1$ (since the data lies on the surface of hypersphere from $\phi(\mathbf{x})\cdot\phi(\mathbf{x}) = K(\mathbf{x},\mathbf{x}) = 1$) so the bound can be written $\sum_{i=1}^{m}\alpha_i^0/m$. Hence an estimate for σ can be found by sequentially training SVMs on the same dataset at successively larger values of σ, evaluating the bound from the α_i^0 for each case and choosing that value of σ for which the bound is minimized. This method [10] will give a reasonable estimate if the data is spread evenly over the surface of the hypersphere but it is poor if the data lie in a flat ellipsoid, for example, since the radius R would be influenced by the largest deviations. More refined estimates therefore take into account the distribution of the data.

One approach [6] is to theoretically rescale data in feature space to compensate for uneven distributions. A more complex strategy along these lines has also been proposed by Schölkopf et al. [44] which leads to an algorithm which has performed well in practice for a small number of datasets. The most economical way to use the training data is to use a *leave-one-out* procedure [6,21]. As an example, we consider a recent result [22,56] in which the number of leave-one-out errors of an L_1-norm soft margin SVM is bounded by $|\{i\colon (2\alpha_i B^2+\xi_i)\geqslant 1\}|/m$ where α_i are the solutions of the optimization task in (10, 6) and B^2 is an upper bound on $K(x_i,x_i)$ with $K(x_i,x_j)\geqslant 0$ (we can determine ξ_i from $y_i(\sum_j \alpha_j K(x_j,x_i)+b)\geqslant 1-\xi_i$). Thus, for a given value of the kernel parameter, the leave-one-out error is estimated from this quantity (the system is *not* retrained with datapoints left out: the bound is determined using the α_i^0 from the solution of (10, 6)). The kernel parameter is then incremented or decremented in the direction needed to lower the bound. This method has worked well on classification of text [22].

2.4. Novelty detection

For many real-world problems the task is not to classify but to detect novel or abnormal instances. Novelty or abnormality detection has potential applications in many problem domains such as condition monitoring or medical diagnosis. One approach can be viewed as one-class classification in which the task is to model the *support* of a data distribution (rather than having to find a real-valued function for estimating the density of the data itself). Thus, at its simplest level, the objective is to create a binary-valued function which is positive in those regions of input space where the data predominantly lies and negative elsewhere.

One strategy [52] is to find a hypersphere with a minimal radius R and center $\mathbf{a}$ which contains most of the data: novel test points lie outside the boundary of this hypersphere. The technique we now outline was originally suggested by Burges [40,3], intepreted as a novelty detector by Tax and Duin [52] and used by the latter authors for real life applications [53]. The effect of outliers is reduced by using slack variables ξ_i to allow for datapoints outside the sphere and the task is to minimize the volume of the sphere and number of datapoints outside i.e.

$$\min \left[R^2 + \frac{1}{mv} \sum_i \xi_i \right] \tag{28}$$

subject to the constraints

$$(\mathbf{x}_i - \mathbf{a})^{\mathrm{T}}(\mathbf{x}_i - \mathbf{a}) \leqslant R^2 + \xi_i \tag{29}$$

and $\xi_i \geqslant 0$, and where v controls the tradeoff between the two terms. The primal objective function is then

$$\begin{aligned} L(R, \mathbf{a}, \alpha_i, \xi_i) = R^2 + \frac{1}{mv} \sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} \gamma_i \xi_i \\ - \sum_{i=1}^{m} \alpha_i (R^2 + \xi_i - (\mathbf{x}_i \cdot \mathbf{x}_i - 2\mathbf{a} \cdot \mathbf{x}_i + \mathbf{a} \cdot \mathbf{a})) \end{aligned} \tag{30}$$

with $\alpha_i \geqslant 0$ and $\gamma_i \geqslant 0$. After kernel substitution the dual formulation amounts to maximization of

$$W(\alpha) = \sum_{i=1}^{m} \alpha_i K(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j=1}^{m} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{31}$$

with respect to α_i and subject to $\sum_{i=1}^{m} \alpha_i = 1$ and $0 \leqslant \alpha_i \leqslant 1/mv$. If $mv > 1$ then *at bound* examples will occur with $\alpha_i = 1/mv$ and these correspond to outliers in the training process. Having completed the training process a test point $\mathbf{z}$ is declared novel if

$$K(\mathbf{z},\mathbf{z}) - 2\sum_{i=1}^{m} \alpha_i K(\mathbf{z},\mathbf{x}_i) + \sum_{i,j=1}^{m} \alpha_i \alpha_j K(\mathbf{x}_i,\mathbf{x}_j) - R^2 \geqslant 0, \tag{32}$$

where R^2 is first computed by finding an example which is *non-bound* and setting this inequality to an equality.

An alternative approach has been developed by Schölkopf et al. [43]. Suppose we restrict our attention to RBF kernels then the datapoints lie on the surface of a hypersphere in feature space since $\phi(\mathbf{x}) \cdot \phi(\mathbf{x}) = K(\mathbf{x},\mathbf{x}) = 1$ from (8). The objective is to separate off the region containing the datapoints from the surface region containing no data. This is achieved by constructing a hyperplane which is maximally distant from the origin with all datapoints lying on the opposite side from the origin and such that $\mathbf{w} \cdot \mathbf{x}_i + b \geqslant 0$. This approach leads to an alternative quadratic programming problem and the authors [43] report that the technique works well on real-life datasets, including the highlighting of abnormal digits for the USPS handwritten character dataset. Instead of repelling the hyperplane away from the origin a further alternative is to *attract* the hyperplane towards the datapoints in feature space (while maintaining the requirement $\mathbf{w} \cdot \mathbf{x}_i + b \geqslant 0$). This leads to an algorithm for novelty detection based on linear programming [4].

3. Regression

For real-valued outputs the learning task can also be theoretically motivated from statistical learning theory (Appendix A). Instead of (3) we now use constraints $y_i - \mathbf{w} \cdot \mathbf{x}_i - b \leqslant \varepsilon$ and $\mathbf{w} \cdot \mathbf{x}_i + b - y_i \leqslant \varepsilon$ to allow for some deviation ε between the eventual targets y_i and the function $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$, modelling the data. We can visualize this as a band or tube of size $\pm(\theta - \gamma)$ around the hypothesis function $f(\mathbf{x})$ and any points outside this tube can be viewed as training errors. The structure of the tube is defined by an *ε-insensitive* loss function (Fig. 4). As before we minimize $\|\mathbf{w}\|^2$ to penalize overcomplexity. To account for training errors we also introduce slack variables $\xi_i, \hat{\xi}_i$ for the two types of training error. These slack variables are

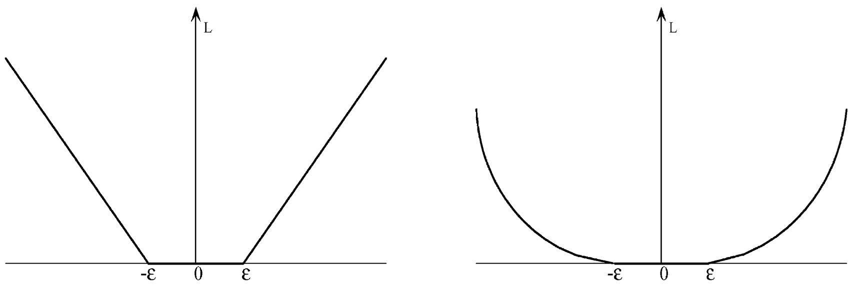

Fig. 4. Left figure: a linear ε-insensitive loss function versus $y_i - \mathbf{w} \cdot \mathbf{x}_i - b$. Right figure: a quadratic ε-insensitive loss function.

zero for points inside the tube and progressively increase for points outside the tube according to the loss function used. This general approach is called ε-SV regression [57] and is the most common approach to SV regression, though not the only one [58]. For a *linear ε-insensitive loss function* the task is therefore to minimize

$$\min \left[\|\mathbf{w}\|^2 + C \sum_{i=1}^{m} (\xi_i + \hat{\xi}_i) \right] \tag{33}$$

subject to

$$\begin{aligned} & y_i - \mathbf{w} \cdot \mathbf{x}_i - b \leqslant \varepsilon + \xi_i, \\ & (\mathbf{w} \cdot \mathbf{x}_i + b) - y_i \leqslant \varepsilon + \hat{\xi}_i, \end{aligned} \tag{34}$$

where the slack variables are both positive $\xi_i, \hat{\xi}_i \geqslant 0$. After kernel substitution the dual objective function is

$$W(\alpha, \hat{\alpha}) = \sum_{i=1}^{m} y_i(\alpha_i - \hat{\alpha}_i) - \varepsilon \sum_{i=1}^{m} (\alpha_i + \hat{\alpha}_i) - \tfrac{1}{2} \sum_{i,j=1}^{m} (\alpha_i - \hat{\alpha}_i)(\alpha_j - \hat{\alpha}_j) K(x_i, x_j) \tag{35}$$

which is maximized subject to

$$\sum_{i=1}^{m} \hat{\alpha}_i = \sum_{i=1}^{m} \alpha_i \tag{36}$$

and

$$0 \leqslant \alpha_i \leqslant C \quad 0 \leqslant \hat{\alpha}_i \leqslant C. \tag{37}$$

Similarly a *quadratic ε-insensitive loss function* gives rise to

$$\min \left[\|\mathbf{w}\|^2 + C \sum_{i=1}^{m} (\xi_i^2 + \hat{\xi}_i^2) \right] \tag{38}$$

subject to (34), giving a dual objective function

$$\begin{aligned} W(\alpha, \hat{\alpha}) = & \sum_{i=1}^{m} y_i(\alpha_i - \hat{\alpha}_i) - \varepsilon \sum_{i=1}^{m} (\alpha_i + \hat{\alpha}_i) \\ & - \frac{1}{2} \sum_{i,j=1}^{m} (\alpha_i - \hat{\alpha}_i)(\alpha_j - \hat{\alpha}_j)(K(\mathbf{x}_i, \mathbf{x}_j) + \delta_{ij}/C) \end{aligned} \tag{39}$$

which is maximized subject to (36). The function modelling the data is then

$$f(\mathbf{z}) = \sum_{i=1}^{m} (\alpha_i - \hat{\alpha}_i) K(\mathbf{x}_i, \mathbf{z}) + b. \tag{40}$$

We still have to compute the bias, b, and we do so by considering KKT conditions for regression. For a linear loss function prior to kernel substitution these are

$$\alpha_i(\varepsilon + \xi_i - y_i + \mathbf{w}\cdot\mathbf{x}_i + b) = 0,$$

$$\hat{\alpha}_i(\varepsilon + \hat{\xi}_i + y_i - \mathbf{w}\cdot\mathbf{x}_i - b) = 0, \tag{41}$$

where $\mathbf{w} = \sum_{j=1}^{m} y_j(\alpha_j - \hat{\alpha}_j)\mathbf{x}_j$, and

$$(C - \alpha_i)\xi_i = 0,$$

$$(C - \hat{\alpha}_i)\hat{\xi}_i = 0. \tag{42}$$

From the latter conditions we see that only when $\alpha_i = C$ or $\hat{\alpha}_i = C$ are the slack variables non-zero: these examples correspond to points outside the ε-insensitive tube. Hence from (41) we can find the bias from a non-bound example with $0 < \alpha_i < C$ using $b = y_i - \mathbf{w}\cdot\mathbf{x}_i - \varepsilon$ and similarly for $0 < \hat{\alpha}_i < C$ we can obtain it from $b = y_i - \mathbf{w}\cdot\mathbf{x}_i + \varepsilon$. Though the bias can be obtained from one such example it is best to compute it using an average over all points on the margin.

Apart from the formulations given here it is possible to define other loss functions giving rise to different dual objective functions. In addition, rather than specifying ε a priori it is possible to specify an upper bound ν $(0 \leqslant \nu \leqslant 1)$ on the fraction of points lying outside the band and then find ε by optimizing over the primal objective function

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\left(\nu m\varepsilon + \sum_{i=1}^{m}|y_i - f(\mathbf{x}_i)|\right) \tag{43}$$

with ε acting as an additional parameter to minimize over [39]. As for classification and novelty detection it is possible to formulate a linear programming approach to regression with [64]

$$\min\left[\sum_{i=1}^{m}\alpha_i + \sum_{i=1}^{m}\alpha_i^* + C\sum_{i=1}^{m}\xi_i + C\sum_{i=1}^{m}\xi_i^*\right] \tag{44}$$

subject to

$$y_i - \varepsilon - \xi_i \leqslant \left(\sum_{j=1}^{m}(\alpha_j^* - \alpha_j)K(x_i, x_j)\right) + b \leqslant y_i + \varepsilon + \xi_i^*. \tag{45}$$

Minimizing the sum of the α_i approximately minimizes the number of support vectors which favours sparse hypotheses with smooth functional approximations of the data. In this approach the kernel does not need to satisfy Mercer's condition [64].

4. Algorithmic approaches

So far the methods we have considered have involved linear or quadratic programming. Linear programming can be implemented using column generation techniques [32] and many packages are available, e.g. CPLEX. Existing LP packages based on simplex or interior point methods can handle problems of moderate size (up to thousands of datapoints). For quadratic programming there are also many applicable techniques including conjugate gradient and primal-dual interior point methods [26]. Certain QP packages are readily applicable such as MINOS and LOQO. These methods can be used to train an SVM rapidly but they have the disadvantage that the kernel matrix is stored in memory. For small datasets this is practical and QP routines are the best choice, but for larger datasets alternative techniques have to be used. These split into three categories: techniques in which kernel components are evaluated and discarded during learning, *working set* methods in which an evolving subset of data is used, and new algorithms that explicitly exploit the structure of the problem. For the first category the most obvious approach is to sequentially update the α_i and this is the approach used by the kernel adatron (KA) algorithm [15]. For binary classification (with no soft margin or bias) this is a simple gradient ascent procedure on (10) in which $\alpha_i \geqslant 0$ initially and the α_i are subsequently sequentially updated using

$$\alpha_i \leftarrow \beta_i \theta(\beta_i), \quad \text{where} \quad \beta_i = \alpha_i + \eta \left(1 - y_i \sum_{j=1}^{m} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) \right) \tag{46}$$

and $\theta(\beta)$ is the heaviside step function. The optimal learning rate η can be readily evaluated: $\eta = 1/K(\mathbf{x}_i, \mathbf{x}_i)$ and a sufficient condition for convergence is $0 < \eta K(\mathbf{x}_i, \mathbf{x}_i) < 2$. With the decision function (12) this method is very easy to implement and can give a quick impression of the performance of SVMs on classification tasks. It can be generalized to the case of soft margins and inclusion of a bias [26,5]. However, it is not as fast as most QP routines, especially on small datasets.

Chunking and decomposition. Rather than sequentially updating the α_i the alternative is to update the α_i in parallel but using only a subset or *chunk* of data at each stage. Thus a QP routine is used to optimize the lagrangian on an initial arbitrary subset of data. The support vectors found are retained and all other datapoints (with $\alpha_i = 0$) discarded. A new working set of data is then derived from these support vectors and additional datapoints which maximally violate the storage constraints. This *chunking* process is then iterated until the margin is maximized. Of course, this procedure may still fail because the dataset is too large or the hypothesis modelling the data is not sparse (most of the α_i are non-zero, say). In this case *decomposition* [34] methods provide a better approach: these algorithms only use a fixed size subset of data with the α_i for the remainder kept fixed.

Decomposition and sequential minimal optimization (*SMO*). The limiting case of decomposition is the sequential minimal optimization (SMO) algorithm of Platt [35] in which only two α_i are optimized at each iteration. The smallest set of parameters which can be optimized with each iteration is plainly two if the constraint $\sum_{i=1}^{m} \alpha_i y_i = 0$ is to hold. Remarkably, if only two parameters are optimized and the rest kept fixed then it is possible to derive an analytical solution which can be executed using few numerical operations. The method therefore consists of a heuristic step for finding the best pair of parameters to optimize and use of an analytic expression to ensure the lagrangian increases monotonically. For the hard margin case the latter is easy to derive from the maximization of δW with respect to the additive corrections a, b in $\alpha_i \rightarrow \alpha_i + a$ and $\alpha_j \rightarrow \alpha_j + b$, $(i \neq j)$. For the L_1 soft margin care must be taken to avoid violation of the constraints (14) leading to bounds on these corrections. The SMO algorithm has been refined to improve speed [24] and generalized to cover the above above-mentioned tasks of classification [35], regression [48] and novelty detection [43]. SVM packages such as SVMTorch [7] and SVMLite [23] also use these *working set methods*. There are also interesting LP variants for these decomposition methods. The fastest LP methods decompose the problem by rows and columns and have been used to solve the largest reported non-linear SVM regression problem with up to 16000 datapoints and a kernel matrix with over a billion elements [2].

Further optimization algorithms. The third approach is to directly approach training from an optimization perspective and create new algorithms. Keerthi et al. [25] have proposed a very effective binary classification algorithm based on the dual geometry of finding the two closest points in the convex hulls. These approaches have been particularly effective for linear SVM problems. The Lagrangian SVM (LSVM) method of Mangasarian and Musicant [28] reformulates the classification problem as an unconstrained optimization task and then solves the problem using an algorithm which only requires the solution of systems of linear equalities. Using a simple program, LSVM can solve linear classification problems for millions of points in minutes. LSVM uses a method based on the Sherman-Morrison-Woodbury formula which only requires solution of systems of linear equalities. The interior-point [13] and semi-smooth support vector methods [14] of Ferris and Munson can be used to solve linear classification problems with up to 60 million data points in 34 dimensions.

5. Further techniques based on kernel representations

So far we have considered methods based on linear and quadratic programming. Here we shall consider further kernel-based approaches which may utilize general non-linear programming and other techniques. In particular, we will consider approaches to two issues: how to improve generalization performance over standard SVMs and how to create hypotheses which are sparse.

Algorithms leading to dense hypotheses. Taking the geometric dual of input space we find datapoints become hyperplanes and separating hyperplanes become points. In this dual space we define *version space* as the set of all hypotheses (points) consistent with the data and this version space is bounded by the hyperplanes representing the data. An SVM solution can be viewed as the center of the largest inscribable hypersphere in version space: the support vectors correspond to those examples with hyperplanes tangentially touching this hypersphere (Fig. 5). If version space is elongated then the center of the largest inscribed hypersphere does not appear to be the best choice. Indeed, a better choice would be the Bayes point which can be approximated by the center of mass of version space [61]. Bayes point machines (BPMs) construct a hypothesis based on this center of ver-

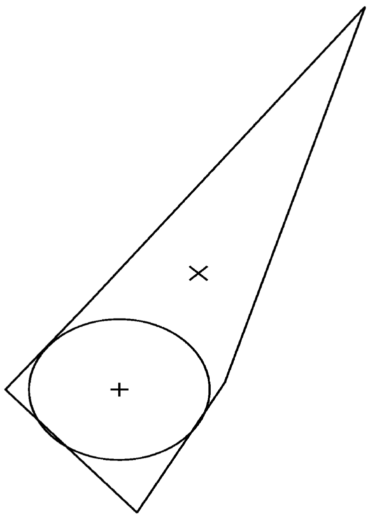

Fig. 5. The center of mass of version space (×) and the center of the largest inscribed sphere (+) in an elongated version space.

sion space and this choice can be justified by theoretical arguments [33,61,38] in addition to having a geometric appeal. In one approach the center of mass is determined using a kernelized billiard algorithm in which version space is traversed uniformly and an estimate of the center of mass is repeatedly updated. For a large majority of datasets version space diverges from sphericality and the BPM outperforms an SVM at statistically significant levels. For artificial examples with very elongated version spaces the generalization error of a BPM can be half that of an SVM [19,20].

Rather than using the center of mass of version space an alternative might be to use a hypothesis that lies towards the center of this space but which is easier to compute. This could be achieved by using repulsive potentials $\Phi(\alpha)$ favouring points towards the center of version space [38]. As an example we could use

$$\min \Phi(\alpha) = \left[\sum_{i=1}^{m} \ln(\alpha_i K(x_i, x_j) + b) \right] \tag{47}$$

subject to

$$\frac{1}{2} \sum_{i=1}^{m} \alpha_i^2 = 1 \tag{48}$$

which is the basis of the *analytic center machine* (ACM) [55]. The gradient and Hessian for (47) can be readily evaluated allowing for computational efficiency and the algorithm appears to perform well in practice. Training involves optimization of a non-linear function and leads to a dense hypothesis.

The ACM algorithm is one example of a broad class of algorithms to which kernel substitution can be applied and which lead to a non-QP non-linear programming task for training. The hypotheses constructed are typically dense in the number of support vectors. For example, for classification, the idea of kernel substitution can be readily applied to the Fisher discriminant [12] and the resulting classifier works well in practice [31]. For regression one can similarly apply kernel substitution to minimization of the standard least squares error function based on the difference between target and the output of the regression function [37,50,51]. This leads to non-linear regression (and classification) functions which perform well on real-life datasets.

Generating sparse hypotheses. The Bayes point machine may exhibit good generalization but it has the disadvantage that the hypothesis is dense. Ideally we would also like to derive kernel classifiers or regression machines which give *sparse* hypotheses using a minimal number of datapoints. The most effective means of obtaining sparse hypotheses remains an object of research but an excellent scheme is the relevance vector machine of Tipping [54]. Using the function $f(z) = \sum_{i=1}^{m} \alpha_i K(z, x_i) + b$ to model the data, a Bayesian prior is defined over the model parameters favouring smooth sparse hypotheses. From Bayes rule a posterior over the weights can be obtained and thence a marginal likelihood or evidence. Iterative maximization of this evidence suggests suitable examples for pruning, creating an eventual hypothesis which is sparse in the number of datapoints used. Experiments show that this approach can sometimes give hypotheses which only use a few percent of the available data [54].

6. Conclusion

The approach we have considered is very general in that it can be applied to a wide range of machine learning tasks and can be used to generate many possible learning machine architectures (RBF networks, feedforward neural networks) through an appropriate choice of kernel. A variety of optimization techniques can be used during the training process which typically involves optimization of a convex function. Above all, kernel methods have been found to work well in practice. The subject is still very much under development but it can be expected to develop as an important tool for machine learning and applications.

Appendix A. Generalization bounds

The generalization bounds mentioned in Section 2 are derived within the framework of probably approximately correct (*pac*) learning. The training and test data are assumed independently and identically (iid) generated from a fixed distribution denoted $\mathscr{D}$. The distribution over input–output mappings will be denoted $(\mathbf{x}, y) \in X \times \{-1, 1\}$ with X assumed to be an inner product space. With these assumptions *pac*-learnability can be defined as follows. Consider a class of possible target concepts C and a learner L using a hypothesis space H to try and learn this concept class. Given a sufficient number, m, of training examples the class C is *pac*-learnable by L if for any target concept $c \in C$, L will with probability $(1-\delta)$ output a hypothesis $h \in H$ with a generalization error $\mathrm{err}_{\mathscr{D}}(h) < \varepsilon(m, H, \delta)$. The *pac* bound $\varepsilon(m, H, \delta)$ is derived using probabilistic arguments [1,60] and bounds the tail of the distribution of the generalization error $\mathrm{err}_{\mathscr{D}}(h)$.

For the case of a thresholding learner L with unit weight vector on an inner product space X and a margin $\gamma \in \mathfrak{R}^+$ the following theorem can be derived if the dataset is linearly separable:

Theorem 1. *Suppose examples are drawn independently according to a distribution whose support is contained in a ball in* $\mathfrak{R}^n$ *centered at the origin, of radius R. If we succeed in correctly classifying m such examples by a canonical hyperplane, then with confidence* $1-\delta$ *the generalization error will be bounded from above by* [46]

$$\varepsilon(m, H, \delta) = \frac{2}{m}\left(\frac{64R^2}{\gamma^2}\log\left(\frac{\gamma e m}{8R^2}\right)\log\left(\frac{32m}{\gamma^2}\right) + \log\left(\frac{4}{\delta}\right)\right) \tag{49}$$

provided $64R^2/\gamma^2 < m$. *This result does not depend on the dimensionality of the space and also states that the bound is reduced by maximizing the margin* γ.

Though this is our main result motivating maximization of the margin for SVMs it does not handle the case of non-separable data or the existence of noise. As pointed out in the main text these instances are handled by introducing an L_1 or L_2 soft margin. The following two bounds do not depend on the training data being linearly separable and cover these two cases [47]:

Theorem 2. *Suppose examples are drawn independently according to a distribution whose support is contained in a ball in* $\mathfrak{R}^n$ *centered at the origin, of radius R. There is a constant c such that with confidence* $1-\delta$ *the generalization error will be bounded from above by*

$$\varepsilon(m, H, \delta) = \frac{c}{m}\left(\frac{R^2 + \|\xi\|_1^2 \log(1/\gamma)}{\gamma^2}\log^2(m) + \log\left(\frac{1}{\delta}\right)\right), \tag{50}$$

where ξ *is the margin slack vector.*

Theorem 3. *Suppose examples are drawn independently according to a distribution whose support is contained in a ball in* $\mathfrak{R}^n$ *centered at the origin, of radius R. There is a constant c such that with confidence* $1-\delta$ *the generalization error will be bounded from above by*

$$\varepsilon(m, H, \delta) = \frac{c}{m}\left(\frac{R^2 + \|\xi\|_2^2}{\gamma^2}\log^2(m) + \log\left(\frac{1}{\delta}\right)\right), \tag{51}$$

where ξ *is the margin slack vector.*

For both these theorems we see that maximizing the margin alone does not necessarily reduce the bound and it is necessary to additionally reduce the norms of the slack variables.

Both these theorems can be adapted to the case of regression. However, in contrast to Theorems 1–3 above it is no longer appropriate to fix the norm of the weight vector since invariance under positive rescaling of the weight vector only holds for a thresholding decision function. For regression the relevant theorem for an L_2 norm on the slack variables is then:

Theorem 4. *Suppose examples are drawn independently according to a distribution whose support is contained in a ball in $\Re^n$ centered at the origin, of radius R. Furthermore fix $\gamma \leqslant \theta$ where θ is a positive real number. There is a constant c such that with probability $1-\delta$ over m random examples, the probability that a hypothesis with weight vector $\mathbf{w}$ has output more than θ away from its true value is bounded above by*

$$\varepsilon(m,H,\delta)=\frac{c}{m}\left(\frac{\|\mathbf{w}\|_2^2R^2+\|\xi\|_2^2}{\gamma^2}\log^2(m)+\log\left(\frac{1}{\delta}\right)\right), \tag{52}$$

where $\xi=\xi(\mathbf{w},\theta,\gamma)$ is the margin slack vector. This theorem motivates the loss functions for regression.

Appendix B. Kernel substitution and Mercer's theorem

In Section 2 we introduced the idea of kernel substitution, equivalent to introducing an implicit mapping of the data into a high-dimensional feature space. Non-linear datasets which are unlearnable by a linear learning machine in input space can then become learnable in feature space. In input space the hypothesis modelling the data is of the form

$$f(\mathbf{x})=\mathbf{w}\cdot\mathbf{x}+b. \tag{53}$$

If the dataset is separable, the separating hyperplane passes through the convex hull defined by the datapoints and hence $\mathbf{w}$ can be expressed as an expansion in terms of the datapoints thus

$$\mathbf{w}=\sum_{i=1}^{m}\alpha_i y_i \mathbf{x}_i. \tag{54}$$

With this expansion the decision function can therefore be written

$$f(\mathbf{x})=\sum_{i=1}^{m}\alpha_i y_i(\mathbf{x}_i\cdot\mathbf{x}_j)+b. \tag{55}$$

The $\mathbf{x}_i$ only appear inside an inner product, justifying kernel substitution and with the choice of kernel implicitly selecting a particular feature space

$$K(\mathbf{x}_i,\mathbf{x}_j)=\phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j). \tag{56}$$

However, only certain choices of kernel are allowable. The requirements on the kernel function are defined by the two theorems below. First we observe that the kernel function is symmetric. In addition we also note from that for a real vector $\mathbf{v}$ we have

$$\mathbf{v}^{\mathrm{T}}\mathbf{K}\mathbf{v} = \left\| \sum_{i=1}^{m} \mathbf{v}_i^{\mathrm{T}} \phi(\mathbf{x}_i) \right\|_2^2 \geqslant 0, \tag{57}$$

where the matrix $\mathbf{K}$ has components $K(\mathbf{x}_i, \mathbf{x}_j)$, $(i = 1, \ldots, m; j = 1, \ldots, m)$. This suggests the following theorem which can be proved:

Theorem 5. *Let $K(\mathbf{x}, \mathbf{y})$ be a real symmetric function on a finite input space, then it is a kernel function if and only if the matrix $\mathbf{K}$ with components $K(\mathbf{x}_i, \mathbf{x}_j)$ is positive semi-definite.*

More generally, for C a compact subset of $\Re^N$ we have:

Theorem 6 (Mercer's theorem). *If $K(\mathbf{x}, \mathbf{y})$ is a continuous symmetric kernel of a positive integral operator T i.e.*

$$(Tf)(\mathbf{y}) = \int_C K(\mathbf{x}, \mathbf{y}) f(\mathbf{x})\, \mathrm{d}\mathbf{x} \tag{58}$$

with

$$\int_{C \times C} K(\mathbf{x}, \mathbf{y}) f(\mathbf{x}) f(\mathbf{y})\, \mathrm{d}\mathbf{x}\, d\mathbf{y} \geqslant 0 \tag{59}$$

for all $f \in L_2(C)$ then it can be expanded in a uniformly convergent series in the eigenfunctions ψ_j and positive eigenvalues λ_j of T, thus

$$K(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{n_{\mathrm{e}}} \lambda_j \psi_j(\mathbf{x}) \psi_j(\mathbf{y}), \tag{60}$$

where n_{e} is the number of positive eigenvalues.

This theorem holds for general compact spaces, and generalizes the requirement to infinite feature spaces. (59) generalizes the semi-positivity condition given in Theorem 5 above.

References

[1] M. Anthony, P. Barlett, Learning in neural networks: theoretical foundations, Cambridge University Press, Cambridge, 1999.

[2] P. Bradley, O. Mangasarian, D. Musicant, Optimization in massive datasets, in: J. Abello, P. Pardalos, M. Resende (Eds.), Handbook of Massive Datasets, Kluwer, Dordrecht, 2001, to appear.

[3] C. Burges, A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery 2 (1998) 121–167.

[4] C. Campbell, K.P. Bennett, A linear programming approach to novelty detection, Advances in Neural Information Processing Systems, vol. 13, MIT Press, Cambridge, MA, 2001, 395–401.

[5] C. Campbell, N. Cristianini, Simple training algorithms for support vector machines, Technical Report, Bristol University, http://lara.enm.bris.ac.uk/cig, 1998.
[6] O. Chapelle, V. Vapnik, Model selection for support vector machines, to appear in Advances in Neural Information Processing Systems, vol. 12, MIT Press, Cambridge, MA, 2000.
[7] R. Collobert, S. Bengio, SVMTorch web page: http://www.idiap.ch/learning/SVMTorch.html.
[8] C. Cortes, V. Vapnik, Support vector networks, Machine Learning 20 (1995) 273–297.
[9] N. Cristianini, C. Campbell, C. Burges (Eds.), Support vector machines and kernel methods, Machine Learning, 2001, to appear.
[10] N. Cristianini, C. Campbell, J. Shawe-Taylor, Dynamically adapting kernels in support vector machines, Advances in Neural Information Processing Systems, vol. 11, MIT Press, Cambridge, MA, 1999, pp. 204–210.
[11] N. Cristianini, J. Shawe-Taylor, An introduction to support vector machines and other kernel-based learning methods, Cambridge University Press, Cambridge, 2000.
[12] R.O. Duda, P.E. Hart, Pattern classification and scene analysis, Wiley, New York, 1973.
[13] M. Ferris, T. Munson, Interior point methods for massive support vector machines, Data Mining Institute Technical Report 00-05, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, 2000.
[14] M. Ferris, T. Munson, Semi-smooth support vector machines, Data Mining Institute Technical Report 00-09, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, 2000.
[15] T.-T. Friess, N. Cristianini, C. Campbell, The kernel adatron algorithm: a fast and simple learning procedure for support vector machines. 15th International Conference on Machine Learning, Morgan Kaufman Publishers, 1998, pp. 188–196.
[16] I. Guyon, N. Matic, V. Vapnik, Discovering informative patterns and data cleaning, in: U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (Eds.), Advances in Knowledge Discovery and Data Mining, MIT Press, Cambridge, MA, 1996, pp. 181–203.
[17] Cf: http://www.clopinet.com/isabelle/Projects/SVM/applist.html.
[18] D. Haussler, Convolution Kernels on Discrete Structures, UC Santa Cruz Technical Report UCS-CRL-99-10, 1999.
[19] R. Herbrich, T. Graepel, C. Campbell, Bayes Point Machines, J. Machine Learning Res. (2001) to appear.
[20] R. Herbrich, T. Graepel, C. Campbell, Robust bayes point machines, Proceedings of ESANN2000, D-Facto Publications, Belgium, 2000, pp. 49–54.
[21] T. Jaakolla, D. Haussler, Probabilistic kernel regression models, Proceedings of the 1999 Conference on AI and Statistics, 1999.
[22] T. Joachims, Estimating the generalization performance of an SVM efficiently, Proceedings of the Seventeenth International Conference on Machine Learning, Morgan Kaufmann, 2000, pp. 431–438.
[23] T. Joachims, Web Page for SVMLight Software: http://wwwais.gmd.de/~thorsten/svm_light.
[24] S. Keerthi, S. Shevade, C. Bhattacharyya, K. Murthy, Improvements to Platt's SMO algorithm for SVM classifier design, Technical Report, Department of CSA, Bangalore, India, 1999.
[25] S. Keerthi, S. Shevade, C. Bhattacharyya, K.A. Murthy, A fast iterative nearest point algorithm for support vector machine classifier design, IEEE Trans. Neural Networks 11 (2000) 124–136.
[26] D. Luenberger, Linear and Nonlinear Programming, Addison-Wesley, Reading, MA, 1984.
[27] O.L. Mangasarian, Linear and non-linear separation of patterns by linear programming, Oper. Res. 13 (1965) 444–452.
[28] O. Mangasarian, D. Musicant, Lagrangian support vector regression, Data mining Institute Technical Report 00-06, June 2000.
[29] E. Mayoraz, E. Alpaydin, Support vector machines for multiclass classification, Proceedings of the International Workshop on Artificial Neural Networks (IWANN99), IDIAP Technical Report 98-06, 1999.
[30] J. Mercer, Functions of positive and negative type and their connection with the theory of integral equations, Philos. Trans. Roy. Soc. London A 209 (1909) 415–446.

[31] S. Mika, G. Ratsch, J. Weston, B. Schölkopf, K.-R. Muller, Fisher discriminant analysis with kernels, Proceedings of IEEE Neural Networks for Signal Processing Workshop, 1999, pp. 41–48.
[32] S. Nash, A. Sofer, Linear and non-linear programming, McGraw-Hill, New York, 1996.
[33] M. Opper, D. Haussler, Generalization performance of bayes optimal classification algorithm for learning a perceptron, Phys. Rev. Lett. 66 (1991) 2677–2680.
[34] E. Osuna, F. Girosi, Reducing the Run-time Complexity in Support Vector Machines, in: B. Schölkopf, C. Burges, A. Smola (Eds.), Advances in Kernel Methods: Support Vector Learning, MIT press, Cambridge, MA, 1999, pp. 271–284.
[35] J. Platt, Fast training of SVMs using sequential minimal optimization, in: B. Schölkopf, C. Burges, A. Smola (Eds.), Advances in Kernel Methods: Support Vector Learning, MIT press, Cambridge, MA, 1999, pp. 185–208.
[36] J. Platt, N. Cristianini, J. Shawe-Taylor, Large margin DAGS for multiclass classification, Advances in Neural Information Processing Systems, vol. 12, MIT Press, Cambridge, MA, 2000.
[37] C. Saunders, A. Gammerman, V. Vovk, Ridge regression learning algorithm in dual variables, Proceedings of the 15th International Conference on Machine Learning (ICML 98), Morgan Kaufmann, Los Altos, CA, 1998, pp. 515–521.
[38] J. Schietse, Towards Bayesian Learning for the Perceptron, Ph.D. Thesis, Faculteit Wetenschappen, Limburgs Universitair Centrum, Belgium, 1996.
[39] B. Schölkopf, P. Bartlett, A. Smola, R. Williamson, Support vector regression with automatic accuracy control, in: L. Niklasson, M. Bóden, T. Ziemke (Eds.), Proceedings of the 8th International Conference on Artificial Neural Networks, Perspectives in Neural Computing, Springer, Berlin, 1998.
[40] B. Schölkopf, C. Burges, V. Vapnik, Extracting support data for a given task, Proceedings: First International Conference on Knowledge Discovery and Data Mining, in: U.M. Fayyad, R. Uthurusamy (Eds.), AAAI Press, Menlo Park, CA, 1995.
[41] B. Schölkopf, C. Burges, A. Smola, Advances in Kernel Methods: Support Vector Machines, MIT Press, Cambridge, MA, 1998.
[42] B. Schölkopf, A. Smola, K.-R. Muller, Kernel Principal Component Analysis, in: B. Schölkopf, C. Burges, A. Smola (Eds.), Advances in Kernel Methods: Support Vector Machines, MIT Press, Cambridge, MA, 1998, pp. 327–352.
[43] B. Schölkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola, R.C. Williamson, Estimating the support of a high-dimensional distribution, Microsoft Research Corporation Technical Report MSR-TR-99-87, 1999.
[44] B. Schölkopf, J. Shawe-Taylor, A. Smola, R. Williamson, Kernel-dependent support vector error bounds, Ninth International Conference on Artificial Neural Networks, No. 470, IEE Conference Publications, 1999, pp. 304–309.
[45] B. Schölkopf, A. Smola, R.C. Williamson, P.L. Bartlett, New support vector algorithms, Neural Computation 12 (2000) 1207–1245.
[46] J. Shawe-Taylor, P.L. Bartlett, R.C. Williamson, M. Anthony, Structural risk minimization over data-dependent hierarchies, IEEE Trans. Inform. Theory 44 (1998) 1926–1940.
[47] J. Shawe-Taylor, N. Cristianini, Margin distribution and soft margin, in: A. Smola, P. Barlett, B. Schölkopf, C. Schuurmans (Eds.), Advances in Large Margin Classifiers, Chapter 2, MIT Press, Cambridge, MA, 1999.
[48] A. Smola, B. Schölkopf, A tutorial on support vector regression, NeuroColt2 TR 1998-03, 1998.
[49] A. Smola, P. Barlett, B. Schölkopf, C. Schuurmans (Eds.), Advances in Large Margin Classifiers, MIT Press, Cambridge, MA, 2001.
[50] J.A.K. Suyhens, J. Vandewalle, Least squares support vector machine classifiers, Neural Process. Lett. 9 (1999) 293–300.
[51] J.A.K. Suyhens, L. Lukas J. Vandewalle, Sparse approximation using least squares support vector machines, Proceedings of the IEEE International Symposium on Circuits and Systems, Geneva, Switzerland (2000) II757–II760.
[52] D. Tax, R. Duin, Data domain description by Support Vectors, in: M. Verleysen (Ed.), Proceedings of ESANN99, D. Facto Press, Brussels, 1999, pp. 251–256.

[53] D. Tax, A. Ypma, R. Duin, Support vector data description applied to machine vibration analysis, in: M. Boasson, J. Kaandorp, J. Tonino, M. Vosselman (Eds.), Proceedings of the 5th Annual Conference of the Advanced School for Computing and Imaging, Heijen, NL, June 15–17, 1999, pp. 398–405.
[54] M. Tipping, The Relevance Vector Machine, Advances in Neural Information Processing Systems, vol. 12, MIT Press, Cambridge, MA, 2000, pp. 652–658.
[55] T. Trafalis, A. Malyscheff, An analytic center machine, Machine Learning, to appear.
[56] V. Vapnik, O. Chapelle, Bounds on error expectation for SVMs, Neural Computation, to appear.
[57] V. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.
[58] V. Vapnik, Statistical Learning Theory, Wiley, New York, 1998.
[59] K. Veropoulos, C. Campbell, N. Cristianini, Controlling the sensitivity of support vector machines, Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), Stockholm, Sweden, 1999.
[60] M. Vidyasagar, A Theory of Learning and Generalization, Springer, Berlin, 1997.
[61] T. Watkin, A. Rau, The statistical mechanics of learning a rule, Rev. Modern Phys. 65 (1993) 499–556.
[62] C. Watkins, Dynamic alignment kernels, Technical Report, UL Royal Holloway, CSD-TR-98-11, 1999.
[63] J. Weston, C. Watkins, Multi-Class Support Vector Machines, in: M. Verleysen (Ed.), Proceedings of ESANN99, D. Facto Press, Brussels, 1999, pp. 219–224.
[64] J. Weston, A. Gammerman, M. Stitson, V. Vapnik, V. Vovk, C. Watkins, Support vector density estimation, in: B. Schölkopf, C. Burges, A. Smola (Eds.), Advances in Kernel Methods: Support Vector Machines, MIT Press, Cambridge, MA, 1998, pp. 293–306.
[65] http://www.ics.uci.edu/~mlearn/MLRepository.html
[66] P. Wolfe, A duality theorem for non-linear programming, Quart. Appl. Math. 19 (1961) 239–244.