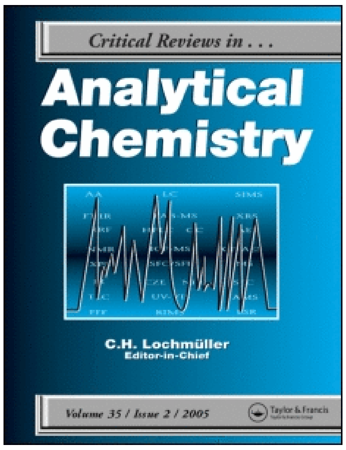

Critical Reviews in Analytical Chemistry
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713400837

Support Vector Machines: A Recent Method for Classification in Chemometrics

Yun Xu[a]; Simeone Zomer[a]; Richard G. Brereton[a]
[a] Centre for Chemometrics, School of Chemistry, University of Bristol, Cantocks Close, Bristol, UK

To cite this Article Xu, Yun , Zomer, Simeone and Brereton, Richard G.(2006) 'Support Vector Machines: A Recent Method for Classification in Chemometrics', Critical Reviews in Analytical Chemistry, 36: 3, 177 — 188
To link to this Article: DOI: 10.1080/10408340600969486
URL: http://dx.doi.org/10.1080/10408340600969486

Critical Reviews in Analytical Chemistry, 36:177–188, 2006
Copyright © Taylor and Francis Group, LLC
ISSN: 1040-8347 print / 1547-6510 online
DOI: 10.1080/10408340600969486

Support Vector Machines: A Recent Method for Classification in Chemometrics

Yun Xu, Simeone Zomer, and Richard G. Brereton
Centre for Chemometrics, School of Chemistry, University of Bristol, Cantocks Close, Bristol, UK

Support Vector Machines (SVMs) are a new generation of classification method. Derived from well principled Statistical Learning theory, this method attempts to produce boundaries between classes by both minimising the empirical error from the training set and also controlling the complexity of the decision boundary, which can be non-linear. SVMs use a kernel matrix to transform a non-linear separation problem in input space to a linear separation problem in feature space. Common kernels include the Radial Basis Function, Polynomial and Sigmoidal Functions. In many simulated studies and real applications, SVMs show superior generalisation performance compared to traditional classification methods. SVMs also provide several useful statistics that can be used for both model selection and feature selection because these statistics are the upper bounds of the generalisation performance estimation of Leave-One-Out Cross-Validation. SVMs can be employed for multiclass problems in addition to the traditional two class application. Various approaches include one-class classifiers, one-against-one, one-against-all and DAG (Directed Acyclic Graph) trees. Methods for feature selection include RFE (Recursive Feature Elimination) and Gradient Descent based approaches.

Keywords support Vector Machines, classification, pattern recognition, feature selection

INTRODUCTION

SVMs (Support Vector Machines) are a relatively young classification technique originally proposed by *Vapnik* (1) that have become increasingly popular after their introduction in the late 1990s particularly within the Machine Learning community. After their introduction, SVM applications have been successfully developed in several areas, including bioinformatics (2), which is probably the most rapidly growing discipline in terms of new methodologies due to the recent explosion of data volumes, econometrics (3) and biometrics (4). More recently, SVMs have been proposed for the analysis of chemical data (5) and have attracted the attention of the chemometrics community, both as a classification technique, and also because their use has been successfully extended to solve calibration problems (6). There are an increasing number of articles focussing on the comparison of SVMs with more traditional chemometrics approaches (7–11).

There are many features of SVMs as discussed below but a key one is that they are boundary methods, that is they try to determine samples on the boundaries between classes, and as such differ from most traditional chemometric methods that use data from classes as a whole for modelling. They also can cope with non-linear situations in which the boundaries are very complex. In chemometrics with the increasing interest in biological problems especially, classes that are not effectively separated linearly are increasingly common, and SVMs have a major role alongside the more traditional multivariate approaches, for this reason. In biology data are generally far more complex than in traditional analytical chemistry, and with the increasing interest of applications of chemometrics to biological problems such as in metabolomics and proteomics, where there is a wealth of data e.g., from mass spectrometric analysis of samples, conventional linear methods are often proving inadequate.

FRAMEWORK FROM STATISTICAL LEARNING THEORY

One feature that makes SVMs attractive is the fact that the technique is well principled in Statistical Learning Theory. In classification, the goal is to generate a function that learns from the available data and that can be used to develop rules to classify new samples. A crucial problem involves controlling complexity: ideally, the classifier should be able to capture all variations in the data relating to class membership without modelling noise. SVMs involve drawing a boundary between groups of samples that fall into different classes. These boundaries may be quite complex: according to *Vapnik* (1), the issue of optimal complexity for the boundaries formed by a classifier can be formally

Address correspondence to Richard G. Brereton, Centre for Chemometrics, School of Chemistry, University of Bristol, Cantocks Close, Bristol BS8 1TS, UK. E-mail: R.G.Brereton@bristol.ac.uk stated as a principle named *structural risk minimisation (SRM)*:

$$R_e \leq R_{emp} + \sqrt{\frac{d_{VC}\left(\log\left(\frac{2N}{d_{VC}}\right)+1\right) - \log\frac{\eta}{4}}{N}} \quad [1]$$

where N is the number of samples used to train the classifier or the size of the training set, d_{VC} is the Vapnik-Chervonenkis dimension that roughly relates to the complexity of the boundaries that the classifier is able to draw between the classes, R_{emp} is the empirical risk that relates to the significance attached to misclassification as observed during the model building, R_e is the expected risk that relates to the true error, and finally η is 1− the probability that the upper bound defined for R holds. While R_{emp} relates (by means of a loss function) to the error measured on the training samples, R_e relates to the "true" underlying error. Accordingly, the best classifier is that one that minimises the upper bound on R_e as defined by the right hand side term of Eq. [1].

SRM is an alternative to empirical risk minimisation (ERM) based on the minimisation of solely R_{emp} and formalises the following common assumptions. First, as the size of the training set increases R_{emp} becomes a better estimate of R_e because the square root term decreases, i.e., more data result in a model whose prediction ability more accurately corresponds to the prediction ability of the population as a whole. Second, using a more complex classifier, roughly relating to d_{VC}, may better model the training data hence reducing R_{emp}, but simultaneously increases the difference between R_{emp} and R_e via the square root term. Essentially, the SRM principle suggests that the complexity of the classifier should be matched to the amount of data available and it shows that it is crucial to control complexity in order to avoid overfitting. A pictorial example for a binary classification problem is given in Figure 1, where two classes of samples are illustrated and proposed boundaries between these classes are drawn. Real boundaries are illustrated in (***a***) which will be unknown prior to obtaining data. Controlling complexity is essential to avoid overfitting in (***b***) and underfitting in (***c***). A good trade-off that is reasonably accurate on the basis of the samples available is (***d***). With fewer samples, a boundary of comparable complexity in (***e***) can be misleading and a simpler solution in (***f***) seems more adequate, hence the more the samples the safer it is to find complex boundaries between classes, but, in converse, if the boundary actually is very complex, one cannot obtain a safe classifier without a large training set.

The SRM principle is, however, mainly of theoretical interest because it solely delivers an upper bound on R_e and because d_{VC} can rarely be computed precisely. Nevertheless, as it will be shown later, SVMs directly embody this formulation by modulating complexity with a trade-off parameter that pulls in different directions the two terms of the upper bound on R_e, on the right end side of Eq. [1]. Also SVMs can provide several useful statistics to estimate the expected generalised performance of a model as these statistics have a close relationship to the common cross-validation error bounds, that are regularly employed in chemometrics.

MATHEMATICAL DERIVATION

In this article, we follow the convention of the machine learning community and all the vectors are column vectors, including sample vectors.

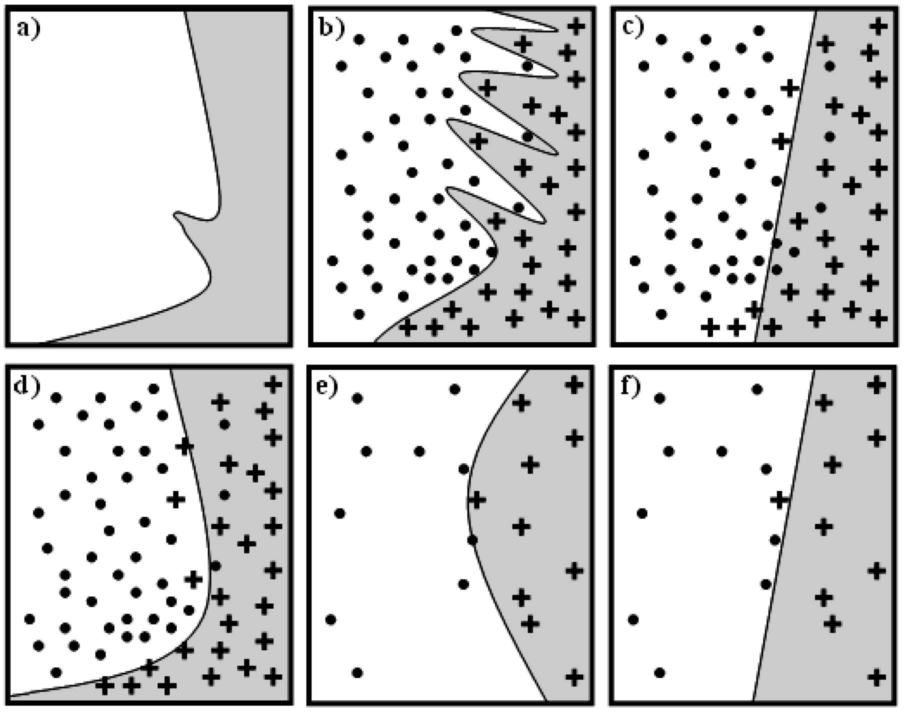

FIG. 1. Relationship between model complexity and number of samples.

The description of the SVM algorithm is in three parts. First, the basic mathematical derivation for linearly separable classes, second, the extension to the non-linearly separable case with the use of kernel functions and third the presentation of the generalised solution with the incorporation of the trade-off parameter to control complexity. The description in this section will be restricted to a two class problem, extensions will be described later.

Linearly Separable Classes

Consider a binary classification problem where samples $\boldsymbol{x}$ have been obtained that have membership of two classes given by $y = \pm 1$. These samples are used to determine a decision function to separate two classes, which in its simplest form will be a linear decision boundary:

$$y(\boldsymbol{x}) = \operatorname{sgn}\left(\boldsymbol{w}_c^T \boldsymbol{x} + b_c\right) \quad [2]$$

where ($\boldsymbol{w}_c$ b_c) are the related weight and bias parameters that must be determined from the training set. Hence, the classification function corresponds to a *hyperplane*, i.e., a line if $\boldsymbol{x}$ is 2-dimensional, a plane if $\boldsymbol{x}$ is 3-dimensional, a n-dimensional hyperplane if $\boldsymbol{x}$ is n-dimensional, that separates the classes in an optimal manner.

A generic hyperplane ($\boldsymbol{w}$, b) is defined by coordinates $\boldsymbol{x}$ satisfying the condition:

$$\boldsymbol{w}^T \boldsymbol{x} + b = 0 \quad [3]$$

which divides the data-space into two regions opposite in sign. The hyperplane *separates* the classes with no error if any sample $\boldsymbol{x_i}$ is projected in the region of the data space with sign equal to the respective class membership y_i or at least on the boundary, i.e. the hyperplane satisfies the condition:

$$y_i \cdot (\boldsymbol{w}^T \boldsymbol{x_i} + b) \geq 1 \quad [4]$$

for all the samples. In addition, the set of samples is said to be *optimally separated* with no error if the distance between the closest samples to the hyperplane is maximal. This specification is necessary because generally one can find an infinite number of hyper-planes ($\boldsymbol{w}$, b) that satisfy Eq. [4], as illustrated in Figure 2.

To simplify the problem of the determination of the optimal separating hyperplane ($\boldsymbol{w}_c$, b_c) and without loss of generality, it is possible to refer to the canonical hyper-plane, where the samples $\boldsymbol{x}_i$ closest to the boundary satisfy the condition:

$$\min_i \|\boldsymbol{w}^T \boldsymbol{x_i} + b\| = 1 \quad [5]$$

For such samples, this distance equals the inverse of the norm of the weight $\frac{1}{\|\boldsymbol{w}\|}$. The optimal hyper-plane is equally spaced from the most similar samples in two classes. The *margin* ρ, i.e., the sum of the distances of the hyper-plane from both the classes, equals $\frac{2}{\|\boldsymbol{w}\|}$.

Hence, the hyperplane that optimally separates the data is the one that minimises the quantity $\frac{1}{2}\|\boldsymbol{w}\|^2$, subject to the constraints of Eq. [4]. This optimisation task can be expressed by the structure error function:

$$\varphi(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{l} \alpha_i (y_i[\boldsymbol{w}^T \boldsymbol{x_i} + b] - 1) \quad [6]$$

where α_i are Lagrange multipliers subjected to the condition $\alpha_i \geq 0$. The value of φ has to be minimised with respect to $\boldsymbol{w}$ and b and maximised with respect to the α_i. Its optimisation can be expressed accordingly as follows (dual form):

$$\max_\alpha L_D(\boldsymbol{\alpha}) = \max_\alpha [\min_{\boldsymbol{w},b} \varphi(\boldsymbol{w}, b, \boldsymbol{\alpha})] \quad [7]$$

The minimum of φ with respect to $\boldsymbol{w}$ and b is given by:

$$\frac{\partial \varphi}{\partial b} = 0 \Rightarrow \sum_{i=1}^{l} \alpha_i y_i = 0 \quad [8]$$

$$\frac{\partial \varphi}{\partial \boldsymbol{w}} = \boldsymbol{0} \Rightarrow \boldsymbol{w} - \sum_{i=1}^{l} \alpha_i y_i \boldsymbol{x_i} = \boldsymbol{0} \quad [9]$$

Hence, from Eqs. [6, 8, 9], the formulation in Eq. [7] becomes

$$L_D(\boldsymbol{\alpha}) = \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i y_i \left(\boldsymbol{x}_i^T \boldsymbol{x_j}\right) y_j \alpha_j - \sum_{i=1}^{l} \alpha_i \quad [10]$$

The optimisation task is that of minimising L_D with respect to α, a vector consisting of Lagrange multipliers, satisfying the

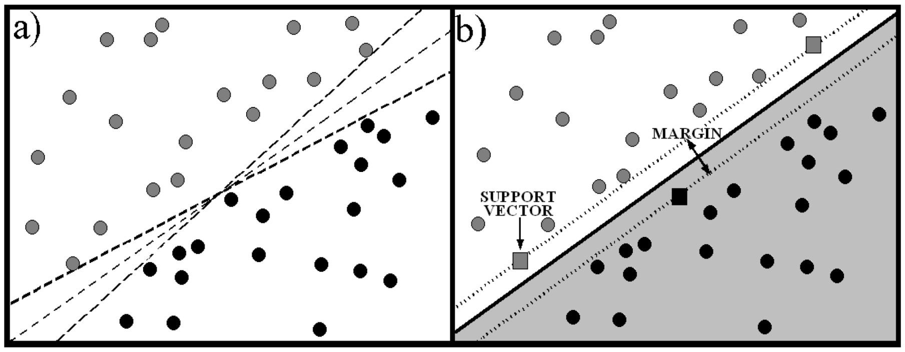

FIG. 2. (a) Possible separating hyperplanes and (b) optimal separating hyperplane (right) maximising the margin between two classes, the closest samples being indicated as support vectors (square marks).

constraints:

$$\alpha_i \geq 0 \tag{11}$$

$$\sum_{j=1}^{l} \alpha_i y_i = 0 \tag{12}$$

Finally, the optimal $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_l)$ allows determination of the weight vector $\boldsymbol{w}_c$ of the optimal separating hyperplane $(\boldsymbol{w}_c, b_c)$ from Eq. [9]:

$$\boldsymbol{w_c} = \sum_{i=1}^{l} \alpha_i y_i \boldsymbol{x_i} \tag{13}$$

while the offset b_c can be calculated from any pair of samples of opposite class satisfying the conditions $\alpha_r, \alpha_s > 0$.

Some additional considerations are as follows.

- The optimisation problem described in Eq. [6] must also satisfy the additional Karush-Kuhn-Tucker condition (12):

$$\alpha_i(y_i[\boldsymbol{w}^T \boldsymbol{x_i} + b] - 1) = 0 \tag{14}$$

  This implies that only the samples satisfying Eq. [5] that are closest to the boundary have $\alpha_i > 0$. These samples that lie on the margin are named *support vectors* (SVs) because they alone determine the solution. The other samples could be removed from the training set without changing the solution. By substituting Eq. [13] into Eq. [2], the classifier can be directly expressed as a function of the support vectors $\boldsymbol{s}_i$ as follows:

$$y(\boldsymbol{x}) = \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i y_i \boldsymbol{s}_i^T \boldsymbol{x} + b_c\right) \tag{15}$$

- The optimisation of L_D in Eq. [10] is a quadratic programming problem, which can be generally written in the form:

$$\min_{\alpha} \left[\frac{1}{2}\alpha' \boldsymbol{H} \alpha + z'\alpha\right] \tag{16}$$

  where $\boldsymbol{H}$ has elements $h_{i,j} = y_i(\boldsymbol{x_i^T x_j})y_j$ and $\boldsymbol{z}$ is a vector of -1s, as can be seen by comparing Eqs. [10] and [16]. Equation [16] can be optimised via quadratic programming and if the matrix $\boldsymbol{H}$ is positive semi-definite (p.s.d.) the solution is unique and has a global optimum. Since the matrix $\boldsymbol{H}$ is essentially an inner product matrix $(\boldsymbol{X}'\boldsymbol{X})$ which is always p.s.d., the learning procedure of SVMs is always reproducible.
- The samples in Eq. [10] are in the form of a scalar product $(\boldsymbol{x}_i^T \boldsymbol{x_j})$. This explains why the approach is particularly fast and suitable when dealing with samples having many variables. Last but not least, this opens the way to treat some of the more complicated non-linearly separable cases using the so called *kernel trick*.

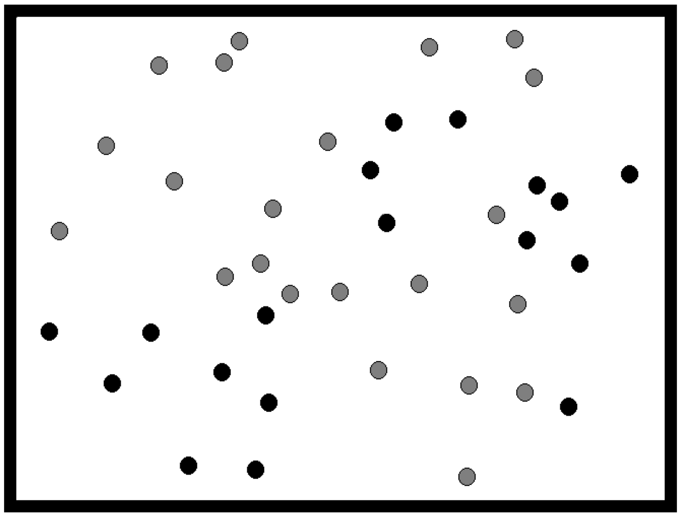
FIG. 3. Two classes that are non-linearly separable.

Kernels

Determining a class boundary in the form of a separating hyperplane is adequate for simpler cases where the classes are nearly or completely linearly separable as it appears in Figure 1(f) and Figure 2. However, this is a situation where arguably many other methods would return satisfactory results and SVMs would not appeal very much due to their relatively complex formulation. An example of a more complex case is presented in Figure 3. There are still two classes, but it is not possible to determine a linear boundary (i.e., a hyperplane) that separates the groups. However, at first glance, samples do not appear randomly distributed: the class represented by the black circles clusters mostly on the right and on the bottom left.

SVMs handle this situation by adding an extra step to the optimisation procedure described above. The optimal separating hyper-plane is not defined in the original input space where the two classes are not separable, but in a new higher dimensional (feature) space where the samples are projected by means of a feature function $\boldsymbol{\Phi}(\boldsymbol{x})$. The back-projection of the optimal separating hyper-plane from this new feature space to the original input space will then result in a non-linear boundary of given complexity that better suits the distribution, providing the feature space is correctly defined. The overall procedure is exemplified in Figure 4: two classes are not linearly separable in the original 2-dimensional space, but their mappings by means of $\boldsymbol{\Phi}(\boldsymbol{x})$ allows the determination of a plane in 3- dimensions that optimally separates them by maximising the margin. Finally the back-projection of this plane into the original 2-dimensional space generates a non-linear boundary of given complexity.

In most situations, the set of functions $\boldsymbol{\Phi}(\boldsymbol{x})$ that is used to map the data is of very high dimensionality, which means that many more dimensions are generally added rather than only one, but it is consequently possible to find boundaries to suit a variety of complex distributions. Mathematically, this is done by reformulating the optimisation task of Eq. [10] by replacing the

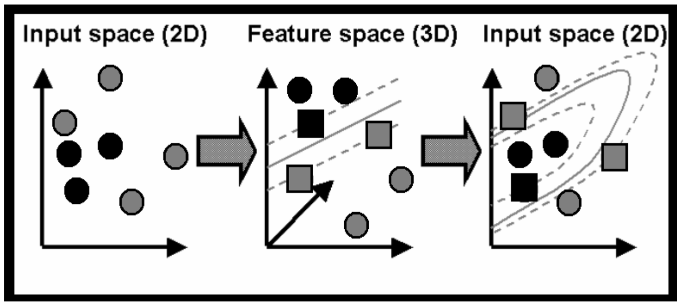

FIG. 4. Creation of the boundary for a non separable case. SVMs project data into a higher-dimensional space, where the optimal separating hyperplane searched. The support vectors are indicated by squares.

scalar product of input vectors ($x_i^T \cdot x_j$) with the scalar product $\langle \Phi(x_i), \Phi(x_j) \rangle$ of the respective feature functions:

$$L_D(\alpha) = \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i y_i \langle \Phi(x_i), \Phi(x_j) \rangle y_j \alpha_j - \sum_{i=1}^{l} \alpha_i \quad [17]$$

However, the feature function $\Phi(x)$ cannot be chosen freely. In fact SVMs restrict the choice to a small family of functions that satisfy the following condition:

$$K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle \quad [18]$$

This introduces one of the most important attributes of SVMs: the scalar product of the feature function $\Phi(x)$ applied to two generic training samples x_i and x_j has an equivalent in the input space where $K(x_i, x_j)$ operates, hence the optimisation task of Eq. [17] can be re-written as:

$$L_D(\alpha) = \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i y_i K(x_i, x_j) y_j \alpha_j - \sum_{i=1}^{l} \alpha_i \quad [19]$$

The optimisation task still involves a quadratic convex programming problem hence being particularly easy to handle, but most importantly, by means of $K(x_i, x_j)$ rather than $\Phi(x)$, it is possible to proceed with the transformation illustrated in Figure 4 omitting the intermediate step of creating the feature space and working only in the original dataspace where $K(x_i, x_j)$ is defined. This powerful attribute is known as the *kernel trick* and it is what makes SVMs effective in addressing more complex tasks. Analogous to Eq. [15], the classification function can be re-written as:

$$y(x) = \text{sgn}\left(\sum_{i=1}^{N_{SVs}} \alpha_i y_i K(s_i, x) + b_c \right) \quad [20]$$

that is still explicitly expressed in a dependence on the SVs.

Only certain kernels that can be employed (as they also must satisfy some additional conditions (12)). Some of the most common are as follows.

- Radial basis function (RBF)

$$K(x_i, x_j) = \exp \frac{-\|x_i - x_j\|^2}{2\sigma^2} \quad [21]$$

- Polynomial function (PF)

$$K(x_i, x_j) = \left(\alpha x_i^T x_j + b\right)^c \quad [22]$$

- Sigmoidal function (SF)

$$K(x_i, x_j) = \tanh\left(\alpha x_i^T x_j + b\right) \quad [23]$$

Each kernel has a set of parameters that must be tuned. The RBF is particularly widespread because it requires only one tuneable parameter (the radial width σ). In chemometrics, recently Üstün et al. proposed also a Pearson VII function with the aim of incorporating the features of both the RBF and PF and applied it to multivariate calibration (13). The application of the kernel operator on each pair of samples in the dataset returns a square and symmetric matrix known as kernel matrix. As an example, Figure 5 shows a possible boundary for the samples in Figure 3, using a RBF kernel with $\sigma = 2$.

Controlling Complexity

Intuitively, because the kernel trick allows SVMs to define complex boundaries, the risk of overfitting is particularly high. Nevertheless, SVMs are equipped with an additional parameter that allows a control on complexity and relates SVMs directly back to the SRM principle described earlier. To introduce this parameter it is easiest to recall the example of the simplest case where the optimal separating hyperplane is determined in the original data-space, without projecting the samples into a higher dimensional feature space. If these cannot be perfectly separated

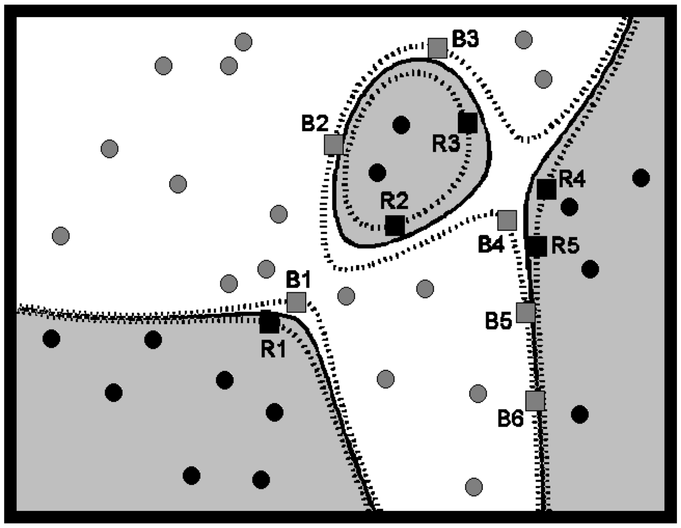

FIG. 5. SVM solution for the case in Figure 3, using a RBF kernel with $\sigma = 2$. The continous line represents the class boundary enclosed by the margin represented with dotted lines.

by the hyperplane, one may also allow deviations $\xi_i > 0$ for individual samples $\boldsymbol{x_i}$. This deviation ξ_i represents the distance of sample $\boldsymbol{x_i}$ from the margin of the pertaining class. Mathematically, the optimisation task in Eq. [6] requires simultaneously maximising the margin $\frac{1}{2}\|\boldsymbol{w}\|^2$ and minimising the empirical error, given by the sum of the allowed deviations $\sum_{i=1}^{l}\xi_i$, hence becoming

$$\varphi(\boldsymbol{w},\xi) = \frac{1}{2}\cdot\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{l}\xi_i \qquad [24]$$

subject to the constraint of Eq. [4] modified as:

$$y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b) \geq 1 - \xi_i \qquad [25]$$

It should be noted that that the margin errors ξ_i become training errors only when $\xi_i > 1$. The additional parameter that allows controlling complexity is the *penalty error* C, as it determines which one of the two criteria of Eq. [24] is emphasised during the optimisation (either $\frac{1}{2}\|\boldsymbol{w}\|^2$ or $\sum_{i=1}^{l}\xi_i$). Lower penalty error values emphasise the first term, allowing higher deviations ξ_i, hence the emphasis will be on margin maximisation rather than training error minimisation. In contrast, higher penalty error values will emphasise the second term, hence allowing smaller deviations ξ_i and minimising the training error. The link to the SRM principle is also due to the fact that the margin defines an upper bound to the VC dimension of the classifier: the larger the margin, the lower is VC dimension (14). Hence, Coffers the opportunity to pursue the optimal trade-off of Eq. [1] because it allows finding the optimal compromise between the two terms on the right hand-side of the equation.

The control on complexity by means of $C$is particularly useful when using kernels to search for non-linear boundaries. The same considerations discussed above are valid with the difference that the generalised hyperplane is searched in the feature space of the function used to project the samples, as it appears in the central diagram of Figure 3. The solution to the optimisation task is identical to the separable case with the exception that the Lagrange multipliers $\boldsymbol{\alpha}$, are now bounded by the penalty error as

$$0 \leq \alpha_i \leq C \qquad [26]$$

In principle, the trade-off between margin maximisation and training error expressed by C should reflect the knowledge of the level of the noise in the data. In practice, the optimal value should be chosen within the range that affects the solution contributing to modify the class boundaries. According to Eq. [26] the lower limit of this interval should be where all the α_i are bounded by C, while the upper limit should be where none are bounded. A detailed explanation on how to optimally tune the SVM parameters (the penalty error together with the parameters of the kernel used) with examples in chemistry is reported by Belousov et al. (15).

To visualise the effect of C on the class boundaries it is possible to refer again to the example of Figure 5, which represents a case where the Lagrange multipliers are left unbounded. Table 1 lists the SVs and the values of the related α coefficients. The boundary is determined by eleven SVs, 6 for class B (in grey) and 5 for class R (in black). The SVs correspond to the samples within the margin space, their number depends on how the parameters are tuned and provide a statistic that indicates the level of separation between classes (in most well defined cases, where classes are better separated, the number of SVs is low). The value of the α_i represents the relative influence of each SV on the boundary. For example B4 has a much higher value for α (12979.89) and in fact one can notice that it is located in a region where samples of the opposite class are found. Roughly speaking, B4 pushes the boundary into a region that is not characteristic of its class, leading to a solution that is likely to be overfitted. Choosing a penalty error that bounds the values of α_i results in smoother boundaries which primarily influence this region. Figure 6 represents the solutions obtained with four values of C. It is possible to notice that as C decreases the margin broadens, the number of SVs determining the boundary (represented by square marks) increases because more samples fall within the margin and the boundary itself becomes smoother. In this case, direct visual inspection might suggest the optimal solution to be found in the range for the penalty error between 1000 and 100. In fact, it appears the while (a) slightly overfits the data distribution, (d) slightly underfits it.

TABLE 1
Support vectors and Lagrange multipliers for the solution in Figure 5 (the samples being marked in the figure)

SVs	α
B1	8149.09
B2	1213.87
B3	479.63
B4	12979.89
B5	279.24
B6	191.34
R1	5410.47
R2	7570.79
R3	1747.22
R4	1502.78
R5	7061.81

MULTI-CLASS SVMs

SVMs were originally developed for separating two classes. However in many cases there are more than two classes in a dataset, so various strategies have been developed to extend SVMs to solve multiclass problems. One main strategy is to divide the multiclass problem into several binary class problems and combine the outputs of all the "sub"-binary classifiers to give the final prediction of the membership of a sample. There are

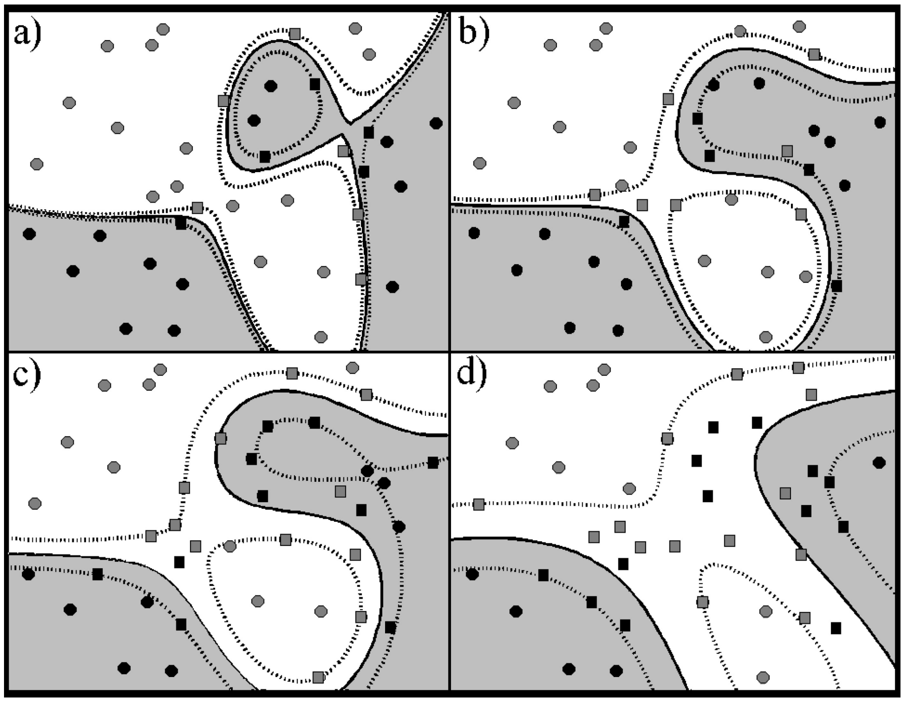

FIG. 6. Boundaries obtained with for the binary classification problem of Figure 3 using a RBF kernel with $\sigma = 2$ and different values of the penalty error. (a) $C = 10000$; (b) $C = 1000$; (c) $C = 100$; (d) $C = 10$. As C decreases, the boundaries become smoother.

three main methods based on these approaches. They are called "one-against-all" (16), "one-against-one" (17, 18) and "DAG (Directed Acyclic Graph) tree" (19).

One-Against-All

One-against-all is the first method reported for extending binary SVMs to solve the multiclass problem. Given k classes under consideration, k binary SVM models are constructed, samples either being considered as part of the class or outside it. The cth ($c = 1, \ldots, k$) SVM is trained with all of the samples in cth class being labelled by $+1$ and all other samples being labelled by -1. Hence k SVM decision functions can be obtained, for each model. Instead of using a sign function, the numerical outputs of the decision functions are considered, as described below. The membership y of an unknown sample $\boldsymbol{x}$ is determined by finding the class for which the corresponding decision function is a maximum:

$$y = \underset{c=1,\ldots,k}{\arg\max}\left(\left(\sum_i \alpha_i y_i K(\boldsymbol{s}_i, \boldsymbol{x})\right)^c + b^c\right) \qquad [27a]$$

where $(\sum_i \alpha_i y_i K(\boldsymbol{s}_i, \boldsymbol{x}))^c + b^c$ is the cth decision function, see Eq. [20]

One-Against-One

Given k classes, the one-against-one approach constructs $k\cdot(k-1)/2$ binary SVM classifiers, each classifier only separating two classes in k classes to be separated. Rather than using a decision function, a sample is simply classified into one of two competing binary classes each time, and membership is determined by a voting scheme, the class with the largest vote is assigned to the sample being tested. There is no generally accepted criterion in the rare event that a sample is equally assigned to more than one class, although probably this sample is ambiguous, which is common in many other soft modelling methods in chemometrics.

DAG Tree

In DAG tree SVM, the training phase is the same as one-against-one and $k \cdot (k-1)/2$ binary SVM classifiers are built. But when testing an unknown sample, it uses a rooted binary directed acyclic graph with $k\cdot(k-1)/2$ internal nodes and k leaves. Each node involves a binary one-against-one SVM model of ith and jth classes. A test sample, starts from the root node and the binary decision function is evaluated depending on the output value of the decision function. A simple example DAG tree of a 4- class decision problem is shown in Figure 7. The computational time for a DAG tree is usually shorter than one-against-one because it is not necessary to go through every binary classifier to test an unknown sample. There is no universal guidance as to which two classes are tested in each node, but for most problems the method is invariant to the arrangement of the nodes. A possible strategy, for example, would be to choose the two most different classes for the first decision in the tree and so on.

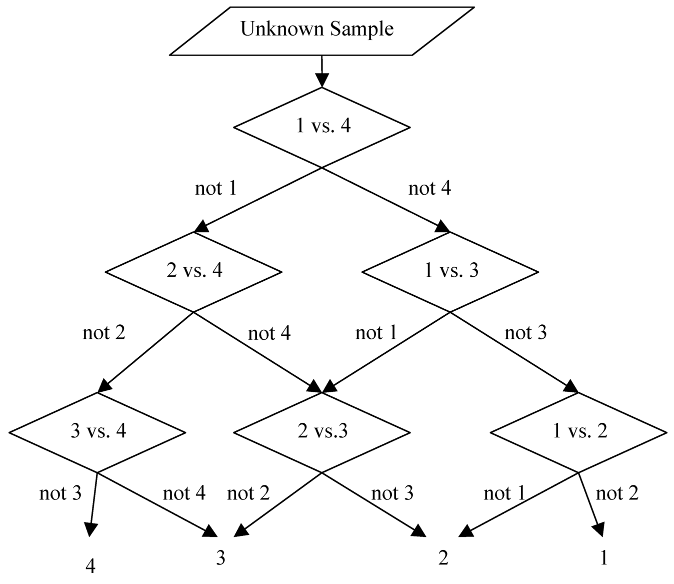

FIG. 7. DAG tree for Multi-class SVMs where there are 4 classes.

Comparison of Approaches

A comprehensive comparison has been conducted by Hsu et al. (20). The results suggest that all three methods provide very similar results. None has overall significant advantages over others in terms of prediction accuracy. In some special cases, one-against-all gave the worst predictions. In addition, one-against-one and the DAG tree appeared to have advantages due to the fact that time needed for training is shorter, a possible explanation could be although more binary classifiers are needed for these two methods, the time required for building each classifier is much less as they only work on a much smaller optimisation problem.

Distante et al. used polynomial SVMs with the "one-against-all" extension to separate six different chemical liquids (water, acetone, hexanal and 3 mixtures) measured by a chemical sensor array (21). SVMs showed much lower prediction error using leave-one-out cross-validation (LOO-CV) at 4.5% compared to feed-forward-neural networks with error back propagation (40%) and RBF neural networks (49%). Brudzewski et al. used linear and RBF SVMs with the "one-against-one" extension to separate 4 types of Polish milk products (22). The samples were measured by 7 tin oxide-based gas sensors and the measurements were used for classification. Perfect cross-validation prediction was achieved.

ONE-CLASS SVMs

Another extension of SVMs is the so called "one-class SVM". It is designed to solve a type of problem referred to as "one-class classification" (23), "concept learning" (24) or "distribution estimation" (25) in different contexts. Rather than separate two or more classes, one-class SVMs are built on a single class that is usually referred to "target class." This type of approach has the ability to tell whether the unknown testing sample belongs to the target class or not (usually being referred as the outlier class). The main advantage of one-class classifiers is that they are "naturally" robust against outliers. For binary/multi-class classification, it is generally assumed that the unknown sample under test belongs to one of the classes to be separated. Even the tested sample does not belong to any of the classes that the classifier was built to separate the classifier can still assign the sample to one of these classes and hence results in erroneous conclusion. For one-class classification a test sample is allowed to belong to none of known classes and can be identified as an "outlier." One-class classification is also more suitable for unbalanced situations, e.g., where one class has been sampled extensively but another class is very under sampled. In such cases, the decision boundary separating these two classes is not stable, but it is possible to draw a stable boundary around the well sampled class. One-class SVMs differ from the methods above in that they are built around one class, the aim being to draw a boundary around the target class, rather than produce a boundary between two classes.

There are two main one-class SVM algorithms, one is called "Support Vector Domain Description" (26) and another is called "ν-Support Vector Classifier" (ν-SVC) (25). These two methods use different structure error function (see Eq. [6]).

For SVDD, the strategy is to find a hypersphere defined by centre point $\boldsymbol{a}$ and radius R which covers most target class regions with a minimum of R. The structure error function is defined as:

$$\varphi(R, \alpha, \xi) = R^2 + C \cdot \sum_i \xi_i \text{ with the constraint}$$
$$\|\boldsymbol{x}_i - \alpha\|^2 \leq R^2 + \xi_i \quad \text{and} \quad \xi_i \geq 0 \qquad [27]$$

where ξ_i is the slack variable modelling the training error (soft margin) and the tolerance of the training error is controlled by the penalty term C.

The ν-SVC method is very much like a binary classifier but instead of separating the positive class from the negative class, all the training samples are assigned to the positive class and the second class is defined as the origin ($\mathbf{0}$) in feature space so that samples belong to the same class as the training samples belong to can be identified as the positive class and all other samples can be identified as the negative class. Hence the problem is defined as finding a hyperplane $\boldsymbol{w}$ that separates all the training samples from the origin with the maximum margin in feature space. There is no bias term b in ν-SVC as the samples are separated from the origin, and the bias term becomes the margin ρ. The structure error function is defined as:

$$\varphi(\boldsymbol{w}, \rho, \xi) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \rho + \frac{1}{\nu \cdot n}\sum_i \xi_i \text{ with constraint } \boldsymbol{w}' \cdot \boldsymbol{x}_i \geq \rho - \xi_i. \qquad [28]$$

where ξ_i is the same as above, n is the number of samples, ρ is the shortest distance from the origin to $\boldsymbol{w}$ and the free parameter ν plays the same role as the C in SVDD. Tax proved that if all samples $\boldsymbol{x}$ have unit norms (sum of squares of all the variables equals 1), both SVDD and ν-SVC give the identical solutions (27). Also if an RBF kernel has been used these two one-class SVMs are the same because for an RBF kernel, all samples have unit norms in feature space (since $K(\boldsymbol{x}_i, \boldsymbol{x}_i) = \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_i)\rangle = 1$).

The decision boundary of a one-class SVM with a linear kernel is a rigid hypersphere and may not always fit real datasets well. However, like binary SVMs, one-class SVMs can also use various kernel functions to obtain flexible decision boundaries. An RBF kernel is a particularly suitable kernel for one-class classification because the boundaries given by a Gaussian with different σ values are very similar to Parzen windows local density estimation (28) with different window width h. Hence it can fit irregularly shaped data. However, if σ is sufficiently large, the decision boundary would be a rigid hypersphere just like linear kernel function. A simulated example of one-class SVM using RBF kernel with different values of σ is given in Figure 8.

The application of one-class classifiers are extensively reported in the fields such as digital character recognition (29), instrument fault detection (30) and forged bank notes detection (31). But it is still not common in chemometrics at the moment. Xu and Brereton applied linear and RBF kernel SVDDs to 3 public gene expression profile data sets and compared the performances of SVDD with other 5 different one-class classifiers namely nearest neighbour, Gaussian model, Parzen windows, PCA data description and k-means data description (32), SVDD appeared to be a "strict" classifier, resulting in relatively high false negative errors but with low false positive errors compared to others. The performances of one-class classifiers are generally comparable to those of multi class classifiers.

MODEL AND FEATURE SELECTION

No matter what type of SVM to be used, there are always one or more parameters that need to be tuned to achieve the optimum performance. Feature selection is also important task. In multivariate data analysis, it is rare that all the variables are relevant to the objective of the study, identifying the most relevant variables can help improving prediction accuracy of the model and also provides insight into the data structure.

SVMs provide several useful statistics that can guide model selection and/or variable selection. They are the weight vector $\boldsymbol{w}$, the norm of the weight vector $||\boldsymbol{w}||^2$, the radius R (the radius of the smallest hypersphere that contains all the training samples in feature space as defined using the SVDD) and "span estimate" S_i (the shortest distance from one SV $\boldsymbol{s}_i$ to the spans of other SVs in feature space, a tighter boundary compared to R). The equations of calculating these statistics are given below (the equation for calculating the weight vector $\boldsymbol{w}$ has already been given in Eq. [13]):

$$\|\boldsymbol{w}\|^2 = \sum_i \sum_j \alpha_i \alpha_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

$$R^2 = \max_\beta \left(\sum_i \beta_i K(\mathrm{x}_i, \mathrm{x}_i) - \sum_i \sum_j \beta_i \beta_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \right) \quad [29]$$

$$\boldsymbol{K}_{sv} = \begin{bmatrix} \boldsymbol{K} & 1 \\ 1' & 0 \end{bmatrix} \quad [30]$$

$S_i^2 = \dfrac{1}{(\boldsymbol{K}_{sv}^{-1})_{ii}}$ where $(\boldsymbol{K}_{sv}^{-1})_{ii}$ is the element in the ith column and ith row of matrix $\boldsymbol{K}_{sv}^{-1}$ and

$K_{sv} = \begin{bmatrix} K & 1 \\ 1^T & 0 \end{bmatrix} \cdot \boldsymbol{K}$ is the kernel matrix and $\mathbf{1}$ is a column vector with 1s.

The works of Vapnik et al. (33, 34) have shown that the generalisation performance from bounds on the leave-one-out cross-validation (LOO-CV) error L (in terms of misclassification rate) can be estimated from the SVM statistics directly because the following two inequalities hold:

$$L \leq 4R^2 \|\boldsymbol{w}\|^2$$

and

$$L \leq \sum_i \alpha_i S_i^2$$

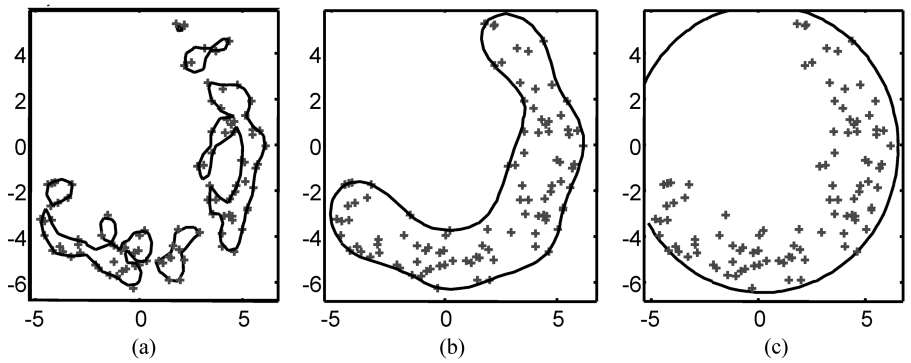

FIG. 8. One-Class SVMs: (a) $\sigma = 2$, (b) $\sigma = 5$ and (c) $\sigma = 100$.

Hence calculating $R^2\|\boldsymbol{w}\|^2$ and/or $\sum_i \alpha_i \cdot S_i^2$ on the training set with various setting of kernel parameters or different set of features can help finding out the optimum SVM model parameters or most relevant variables without going through the actual LOO-CV (35, 36).

Model Selection

There are numerous possible models as represented by the kernel functions and their tuneable parameters, so it is important to have a strategy for choosing the most appropriate model.

A common used strategy is to use a grid search (37–39) on the training set to find the optimum combination of kernel parameters that maximise the generalisation performance of SVM model (e.g., minimise $R^2\|\boldsymbol{w}\|^2$, $\sum_i \alpha_i S_i^2$ or Cross-Validation error as appropriate). The range of each kernel parameters as well as the step length has to be specified, and a contour plot can be obtained. The region with the minimum error suggests the best combination of the kernel parameters. For difficult problems a two-step grid search might be needed. The first search is conducted in a large range with large step length to identify possible optimum region(s) and then use a second fine search with small step length in those possible optimum regions is performed.

Feature Selection

All the SVM statistics described above can be used for feature selection as well. Variables can be ranked by their contribution to the SVM statistics and variables with high ranks can be considered as the most important ones. However, feature selection is more complicated than model selection as the search space is much larger. Hence an effective searching heuristic is required.

Guyon et al. proposed a SVM feature selection method called SVM-RFE (40) (Recursive Feature Elimination) based on backwards sequential selection method using the weight vector $\boldsymbol{w}$ as the ranking criterion. SVM-RFE is conducted in an iterative way. In each step, a SVM model is built on the training set and the weight vector $\boldsymbol{w}$ is obtained. The variable with the lowest absolute value of the weight is considered as the least important and removed from dataset The SVM model is built again on the data set without the least important variable to obtain a new weight vector $\boldsymbol{w}$ and then the second least important variable is removed from the dataset. This procedure is repeated until only r (as specified beforehand) variables remain. An alternative implementation involves continuing until $r = 1$, this gives a rank list of all the variables which represents their relative importance. The authors applied the SVM-RFE algorithm to two gene expression profile data sets and showed that by selecting a subset of important genes can significantly improve the accuracy of the class prediction. Prados et al. applied SVM-RFE to SELDI (Surface Enhanced Laser Desorption/Ionisation)-TOF Mass spectra data to identify potential biomarkers for early cerebral accident diagnosis (41). Compared to two other ranking criteria, the features identified by SVM-RFE can yield the best prediction accuracy and in fact the only acceptable accuracy when subjected to a blind test. West-Nielsen et al. applied linear SVM and SVM-RFE (also with a linear kernel) to proteomic mass spectra data sets attempting to separate normal samples and spiked samples (42). It appeared that the prediction accuracy of SVM built on 10 out of 22,572 variables is just slightly worse than the SVM built on the data set with all the variables (97% comparing to 100%, in term of prediction accuracy of LOO-CV). This suggests that SVM-RFE can retain most relevant information while SVM model itself relatively robust to the existence of redundant information.

Rakotomamonjy (43) proposed a more general SVM feature selection framework. Two ranking strategies were introduced namely zero-order ranking, i.e., using a ranking criterion directly and first-order ranking, taking into account the changed SVM statistics when each variable has been removed in turn. As an example, the first-order $\|\boldsymbol{w}\|^2$ ranking criterion is $|\|\boldsymbol{w}\|^2 - \|\boldsymbol{w}_j\|^2|$ where $\boldsymbol{w}_j$ is the weight vector when the jth variable has been removed. The zero-order criteria represent the direct contributions of each variable to the model while first-order criteria represent the sensitivity of the model to each variable as reflected by size of the change in the model when it has been removed.

Using either zero-order or first-order criteria, the variable obtained the lowest ranking criterion value is considered as the least important one. The author also used backwards sequential selection and ranked variables using zero and first-order values of $R^2\|\boldsymbol{w}\|^2$ and $\sum_i \alpha_i S_i^2$. The performances of these ranking criteria were compared to each other on both simulated and real data. The results suggested that no criterion is universally superior to others but first order $\|\boldsymbol{w}\|^2$ criterion performed consistently well. The author suggested that due to the simplicity and good performance of $\|\boldsymbol{w}\|^2$ on every testing data set, it might be the best choice for practical application. Camps-Valls et al. employed a RBF kernel SVM model and using first-order $\|\boldsymbol{w}\|^2$ criterion on a molecules database to select important variables to predict the efficacy of antisense oligonucleotide.[44] The results suggested that the prediction accuracy has been significantly improved comparing to previously reported results by using the Pearson correlation coefficient.

Weston et al. (45) proposed another SVM feature selection method based on optimising $R^2\|\boldsymbol{w}\|^2$. In their work, instead of using backwards sequential selection they considered variable selection as an optimisation problem: given a binary vector $\boldsymbol{\nu}$ consisting of a random sequence of 1s and 0s of length m (no. of variables) $\boldsymbol{x}$ is replaced with $\boldsymbol{x} \cdot \boldsymbol{\nu}$ and $\boldsymbol{\nu}$ is optimised to minimise the objective function $R^2\|\boldsymbol{w}\|^2$ via gradient descent. In the optimisation $\boldsymbol{\nu}$ was unconstrained and approximated by real values. The q variables (where q is much smaller than m) with the lowest values in $\boldsymbol{\nu}$ are set to 0 and only non-zero elements are retained. The optimisation is repeated until only r nonzero elements in $\boldsymbol{\nu}$ remain. The authors compared this method with some common used feature selection criteria such as Fisher-scores, Kolmorogov–Smirnov scores and Pearson correlation coefficients on various simulated and real data sets and this method consistently showed superior performance.

CONCLUSIONS

SVMs are a technique that is attractive for a variety of reasons. Computationally, the algorithm is efficient because it involves optimising a quadratic convex programming function that is easy to handle and training vectors feed the algorithm in the form of scalar product, making the approach suitable when vectors account for many variables. The so called *kernel trick* allows determining complex boundaries without changing the nature of the optimisation task. Additionally, because samples feed the algorithm in the form of scalar products, the method is particularly robust when dealing with high dimensional data. Another attractive feature is that SVMs are well principled and allow controlling complexity within a framework provided by Statistical Learning theory, as they embody the structural risk minimisation principle. Finally, the number of parameters that need tuning is relatively limited (e.g., the penalty error C and the radial width σ for the RBF kernel).

SVMs have as yet rarely been employed in analytical chemistry for classification purposes. A problem is that there is not currently available easy to use commercial software, so only investigators with a good experience of programming e.g., via Matlab, can effectively employ SVMs. It is usually necessary to optimise the kernel, which requires some experience and appreciation of the theory behind SVMs. Most current chemometrics software is based around linear approaches such as PCA and PLS, which, while very well established represent the state of the art in chemometric methodology of 10 or 20 years in the past. These traditional methods are very effective under certain circumstances, especially in traditional analytical chemistry (e.g., spectroscopy in process analysis) where the problems are linear and well understood, and SVMs may well be unnecessary and introduce unwanted complexity to such well established processes. However with a new generation of problems emerging, especially in the interface of biology and analytical chemistry, new methods are required to tackle complex non-linear problems. In addition to their attractions in terms of handling complex boundaries, while reducing the risk of over-fitting if correctly applied, SVMs differ to, for example, most neural networks, in that they are reproducible and so it is possible to return to a problem and, knowing the optimisation parameters, obtain identical results which makes issues of validation, traceability and use in controlled environments feasible. We are likely to hear more about SVMs in the future.

REFERENCES

1. V. N. Vapnik, *The Nature of Statistical Learning Theory, 2nd edition* (Springer, New York, 2000).
2. T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler, Support vector machine classification and validation of cancer tissue samples using microarray data. *Bioinformatics* 16 (2000):906–914.
3. F. E. H. Tay and L. J. Cao, Modified support vector machines in financial time series forecasting. *Neurocomputing* 48 (2002):847–861.
4. G. Guo, S. Z. Li, and K. L. Chan, Face recognition by support vector machines. *Image and Visual Computing* 19 (2001): 631–638.
5. A. I. Belousov, S. A. Verzakov, and J. Von Frese, A flexible classification approach with optimal generalisation performance: support vector machines. *Chemometrics and Intelligent Laboratory Systems* 64 (2002):15–25.
6. U. Thissen, B. Üstün, W. J. Melssen, and L. M. C. Buydens, Comparing support vector machines to PLS for spectral regression applications. *Analytical Chemistry* 76 (2004):3099–3105.
7. S. R. Amendolia, G. Cossu, M. L. Ganadu, B. Golisio, G. L. Masala, and G. M. Mura, A comparative study of K-nearest neighbour, support vector machine and multi-layer perceptron for Thalassemia screening. *Chemometrics and Intelligent Laboratory Systems* 69 (2003):13–20.
8. U. Thissen, M. Pepers, B. Ustun, W.J. Melssen and L.M.C. Buydens, Comparing support vector machines to PLS for spectral regression applications. *Chemometrics and Intelligent Laboratory Systems* 73 (2004):169–179.
9. T. Czekaj, W. Wu, and B. Walczak, A modification of the ICOMP criterion for estimation of optimum complexity of PCR models. *Journal of Chemometrics* 19 (2005):341–354.
10. S. Zomer, C. Guillo, R. G. Brereton, and M. Hanna-Brown, Toxicological classification of urine samples using pattern recognition techniques and capillary electrophoresis. *Analytical and Bioanalytical Chemistry* 378 (2004):2008–2020.
11. S. Zomer, M. Sánchez, R. G. Brereton, and J. L. Pérez Pavón, Active learning support vector machines for optimal sample selection in classification. *Journal of Chemometrics* 18 (2004):294–305.
12. C. J. C. Burges, A tutorial on support vector machines for Pattern recognition. *Data Mining and Knowledge Discovery* 2 (1998):121–167.
13. B. Üstün, W. J. Melssen, and L. M. C. Buydens, Facilitating the application of Support Vector Regression by using a universal Pearson VII function based kernel. *Chemometrics and Intelligent Laboratory Systems* 81 (2006):29–41.
14. S. R. Gunn, Support vector machines for classification and regression. Technical Report, School of Electronics and Computer Science, University of Southampton, England, (1998). ⟨http://www.ecs.soton.ac.uk/∼srg/publications/pdf/SVM.pdf⟩.
15. A. I. Belousov, S. A. Verzakov, and J. Von Frese, Applicational aspects of support vector machines. *Journal of Chemometrics* 16 (2002):482–489.
16. L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, L. D. Jackel, Y. Le Cun, U. A. Muller, E. Sackinger, P. Simard, and V. Vapnik, In *International Conference on Pattern Recognition*, (Jerusalem, Israel, 1994), 77.
17. J. Friedman, Another approach to polychotomous classification, Technical report, Department of statistics, Stanford University, 1996 ⟨http://www-stat.standford.edu/reports/friedman/poly.ps.Z⟩.
18. U. Kreßel, *Advances in Kernel Methods – Support Vector Learning* (MIT Press, Cambridge, MA, 1999).
19. J. C. Platt, N. Cristianini, and J. Shawe-Taylor, *Advances in Neural Information Processing Systems* (MIT Press, Cambridge, MA, 2000).
20. C. W. Hsu, and C. J. Lin, Comparison of methods for multiclass support vector machines. *IEEE Transaction on Neural Networks* 13 (2002):415–425.

21. C. Distante, N. Ancona, and P. Siciliano, Analysis of peaches ripeness by an electronic nose and near-infrared spectroscopy. *Sensors and Actuators B-Chemical* 88 (2003):30–39.
22. K. Brudzewski, S. Osowski, and T. Markiewicz, Classification of milk by means of an electronic nose and SVM neural network. *Sensors and Actuators B-Chemical* 98 (2004):291–298.
23. M. Moya, M. Koch, and L. Hostetler, in *Proceedings word congress on neural networks* (International Neural Network Society, Portland, OR, 1993), 797.
24. N. Japkowicz, C. Myers, and M. Gluck, In *the Fourteenth International Joint Conference on Artificial Intelligence* (Montreal, Canada, 1995).
25. B. Schölkopf, R. C. Williamson, A. Smola, J. Shawe-Taylor, *Advances in Neural Information Processing Systems* (MIT Press, Cambridge, MA, 2000).
26. D. M. J. Tax, and R. P. W. Duin, Support vector domain description, *Pattern Recognition Letters* 20 (1999):1191–1199.
27. D. M. J. Tax, *One-Class Classification*, Ph.D thesis, Delft University of Technology, 2001.
28. R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification, 2nd Edition (*John Wiley & Sons, New York, 2001).
29. L. M. Manevitz, and M. Yousef, One-class SVMs for document classification. *Journal of Machine Learning Research* 2 (2002):139–154.
30. T. Sarmiento, S. J. Hong, and G. S. May, In *IEEE/SEMI Advanced Semiconductor Manufacturing Conference* (Munich, Germany, 2005).
31. C. He, M. Girolami, and G. Ross, Employing Optimised Combinations of One-Class Classifiers for Automated Currency Validation, *Pattern Recognition* 37 (2004):1085–1096.
32. Y. Xu and R. G. Brereton, Diagnostic pattern recognition on gene-expression profile data by using one-class classification. *Journal of Chemical Information and Modeling* 45 (2005):1392–1401.
33. V. Vapnik, *Statistics Learning Theory* (Wiley, New York, 1998)
34. V. Vapnik and O. Chapelle, Bounds on error expectation for support vector machines. *Neural Computing* 12 (2000):2013–2036.
35. O. Chapelle and V. Vapnik, *Advances in Neural Information Processing Systems Volume 12* (MIT Press, Cambridge, MA, 2000), 230.
36. O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, Choosing multiple parameters for support vector machings. *Machine Learning* 46 (2002):131–159.
37. J. M. Kriegl, T. Arnhold, B. Beck, and T. Fox, A support vector machine approach to classify human cytochrome P450 3A4 inhibitors. *Journal of Computer Aided Molecular Design* 19 (2005):189–201.
38. J. A. Fernandez Pierna, V. Baeten, A. Michotte Renier, R. P. Cogdill, and P. Dardenne, Classification of modified starches by Fourier Transform infrared spectroscopy using support vector machines. *Journal of Chemometrics* 18 (2004):341–349.
39. S. Zomer, R. G. Brereton, J. F. Carter, and C. Eckers, Support vector machines for the discrimination of analytical chemical data: application to the determination of tablet production by pyrolysis-gas chromatography-mass spectrometry. *Analyst* 129 (2004):175–281.
40. I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, Gene selection for cancer classification using support vector machines. *Machine Learning*, 46 (2002):389–442.
41. J. Prados, A. Kalousis, J. Sanchez, L. Allard, O. Carrette, and M. Hilario, Mining mass-spectra for diagnosis and biomarker discovery of cerebral accidents. *Proteomics* 4 (2004):2320–2332.
42. M. West-Nielsen, E. V. Hogdall, E. Marchiori, C. K. Hogdall, C. Schou, and N. H. Heegaard, Sample handling for mass spectrometric proteomic investigations of human sera, *Analytical Chemistry* 77 (2005):5114–5123.
43. A. Rokotomamonjy, Variable selection using svm based criteria. *Journal of Machine Learning Research* 3 (2003):1357–1370.
44. G. Camps-Valls, A. M. Chalk, A. J. Serrano-López, J. D. Martín-Guerrero, and E. L. Sonnhammer, Profiled support vector machines for antisense oligonucleotide efficacy prediction, *BMC Bioinformatics*, 5 (2004):135–143.
45. J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik, *Advances in Neural Information Processing Systems, Volume 13* (MIT Press, Cambridge, MA, 2001), 668.