Algorithmic Approaches to Training Support Vector Machines: A Survey

Colin Campbell,
Department of Engineering Mathematics, Bristol University,
Bristol BS8 1TR, United Kingdom (C.Campbell@bris.ac.uk)

Abstract: Support Vector Machines (SVMs) have become an increasingly popular tool for machine learning tasks involving classification, regression or novelty detection. They exhibit good generalisation performance on many real-life datasets and the approach is well-motivated theoretically. Training involves optimisation of a convex cost function, there are relatively few free parameters to adjust and the architecture does not have to be found by experimentation. In this tutorial we survey methods for training SVMs including model selection strategies for determining the free parameters and new techniques for active selection of training examples.

1. Introduction.

Support Vector Machines (SVMs) have recently been successfully applied to a number of applications ranging from particle identification, face identification and text categorisation to engine knock detection, bioinformatics and database marketing [9]. The approach is systematic and motivated by statistical learning theory [26] and Bayesian arguments. The training task involves optimisation of a convex cost function: there are no false local minima to complicate the learning process. The approach has many other benefits, for example, the model constructed has an explicit dependence on the most informative patterns in the data (the support vectors), hence interpretation is straightforward. In this tutorial we introduce this subject with an emphasis on the issue of training SVMs.

From the perspective of statistical learning theory the motivation for considering binary classifier SVMs comes from theoretical bounds on the generalisation error [26, 6]. Though we do not quote the relevant theorem here we note that it has two important features. Firstly, the upper bound on the generalization error do not depend on the dimension of the space. Secondly, the error bound is minimised by maximising the *margin*, γ, i.e. the minimal distance between the hyperplane separating the two classes and the closest datapoints to the hyperplane (Figure 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:

$$f(\mathbf{x}) = \mathrm{sign}\left(\mathbf{w} \cdot \mathbf{x} + b\right)$$

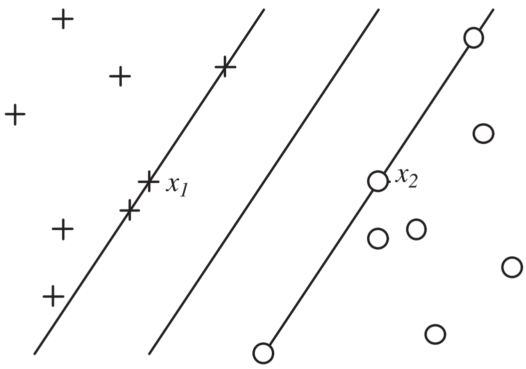

Figure 1: The *margin* is the perpendicular distance between the separating hyperplane and a hyperplane through the closest points (the *support vectors*). $\mathbf{x}_1$ and $\mathbf{x}_2$ are examples of support vectors of opposite sign. The *margin band* is the region between the hyperplanes on both sides of the separating hyperplane.

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 can define a *canonical hyperplane* 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. For the separating hyperplane $\mathbf{w}\cdot\mathbf{x}+b=0$ the normal vector is clearly $\mathbf{w}/\left\|\mathbf{w}\right\|_2$, and hence the margin is given by the projection of $\mathbf{x}_1-\mathbf{x}_2$ onto this vector (see Figure 1). Since $\mathbf{w}\cdot\mathbf{x}_1+b=1$ and $\mathbf{w}\cdot\mathbf{x}_2+b=-1$ this means the margin is $\gamma = 1/\left\|\mathbf{w}\right\|_2$. To maximise the margin the task is therefore:

$$\text{Minimise} \qquad g(\mathbf{w}) = \frac{1}{2}\left\|\mathbf{w}\right\|_2^2$$

subject to the constraints:

$$y_i\left(\mathbf{w}\cdot\mathbf{x}_i + b\right) \geq 1 \qquad \forall i$$

and the learning task can be reduced to minimisation of the primal Lagrangian:

$$L = \frac{1}{2}(\mathbf{w}\cdot\mathbf{w}) - \sum_{i=1}^{m} \alpha_i\left(y_i(\mathbf{w}\cdot\mathbf{x}+b) - 1\right)$$

where α_i are Lagrangian multipliers, hence $\alpha_i \geq 0$. Taking the derivatives with respect to b and $\mathbf{w}$ and resubstituting back in the primal gives the Wolfe dual Lagrangian:

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

which must be maximised with respect to the α_i subject to the constraint:

$$\alpha_i \geq 0 \qquad \sum_{i=1}^{m} \alpha_i y_i = 0 \tag{2}$$

In the dual lagrangian (1) we notice that the datapoints, $\mathbf{x}_i$, only appear inside an inner product. To get a potentially better representation of the data we can map the datapoints into an alternative space, generally called *feature space* (a pre-Hilbert or inner product space) through a replacement:

$$\mathbf{x}_i \cdot \mathbf{x}_j \rightarrow \phi\left(\mathbf{x}_i\right) \cdot \phi(\mathbf{x}_j)$$

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 Hilbert space. With a suitable choice of kernel the data can become separable in feature space despite being non-separable in the original input space. Thus, 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):

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma^2} \tag{3}$$

Many other choices for the kernel are possible e.g.:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d \qquad K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta \mathbf{x}_i \cdot \mathbf{x}_j + b) \tag{4}$$

defining polynomial and feedforward neural network classifiers. Indeed, the class of mathematical objects which can be used as kernels is very general and includes, for example, scores produced by dynamic alignment algorithms [10, 27]. For binary classification with the given choice of kernel the learning task therefore involves maximisation of the Lagrangian:

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

subject to constraints (2). After the optimal values of α_i have been found the decision function is based on the sign of:

$$f(\mathbf{z}) = \sum_{i=1}^{m} y_i \alpha_i K(\mathbf{z}, \mathbf{x}_i) + b \tag{6}$$

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

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

The confidence of a classification is directly related to the magnitude of $f(\mathbf{z})$ [19].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 hypothesis is solely given by those points which are closest to the hyperplane and *they are the most informative patterns in the data.* This framework can be elaborated in many ways, for example:

Multiclass Classification. A number of schemes for multiclass classification have been outlined [15, 28]. One of the simplest schemes is to use a directed acyclic graph (Figure 2 (left)) with the learning task reduced to binary classification at each node [7].

Soft margins and allowing for training errors. An SVM can fit noise present in the training data leading to poor generalisation. The effect of outliers and noise can be reduced by introducing a *soft margin* to remove the effect of outliers [4]. Currently two schemes are possible. In the first (L_1 error norm) the learning task is the same as in (2,5) except for the introduction of the box constraint:

$$0 \leq \alpha_i \leq C \tag{7}$$

while in the second (L_2 error norm) the learning task is the same as (2,5) except for addition of a small positive constant, λ, to the leading diagonal of the kernel matrix:

$$K(\mathbf{x}_i, \mathbf{x}_i) \leftarrow K(\mathbf{x}_i, \mathbf{x}_i) + \lambda \tag{8}$$

C and λ control the trade-off between training error and generalisation ability and are chosen by means of a validation set.

Novelty Detection. For many real-world problems the task is not to classify but to detect novel or abnormal instances. Novelty detection can be performed by modelling the support of a distribution (i.e. finding a function which is 1 where most data lies and 0 elsewhere).One approach [23, 1, 25] is to find a sphere with a minimal radius R and centre a which contains most of the data: novel test points are those which lie outside the boundary of the sphere. During the training process the effect of outliers is reduced by using slack variables ξ_i to allow for some datapoints outside the sphere. Thus the task is to minimise the volume of the sphere and number of datapoints outside i.e. $R^2 + \frac{1}{\nu m} \sum_i \xi_i$ where ν controls the tradeoff between the two terms. For the chosen kernel the learning task then reduces to maximisation of:

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

with respect to α_i and subject to $\sum_{i=1}^{m} \alpha_i = 1$ and $0 \leq \alpha_i \leq 1/\nu m$. A test point $\mathbf{z}$ is novel if:

$$K(\mathbf{z},\mathbf{z}) - 2\sum_{i=1}^{m}\alpha_i K(\mathbf{z},\mathbf{x}_i) + \sum_{i,j=1}^{m}\alpha_i\alpha_j K(\mathbf{x}_i,\mathbf{x}_j) \geq R^2 \tag{9}$$

R^2 is found by using an equality in (9) for a training example for which α_i is not at a bound i.e. $0 < \alpha_i < 1/\nu m$. This approach has also been developed by Schölkopf et al. [20] who give a different QP formulation for estimating the support and provide good experimental evidence in favour of this approach by highlighting abnormal digits in the USPS handwritten character dataset.

Regression. Several approaches to regression [6, 26] are possible but, as for classification, the essential algorithmic task is to minimise a convex function to give a sparse solution. Thus, for example, for ϵ-SV regression [25] we minimise $||\mathbf{w}||_2^2$, as before, to increase flatness or penalise overcomplexity, and use constraints $y_i - \mathbf{w}\cdot\phi(\mathbf{x}_i) - b \leq \epsilon$ and $\mathbf{w}\cdot\phi(\mathbf{x}_i) + b - y_i \leq \epsilon$ allowing for a deviation ϵ between eventual targets y_i and the function $f(x) = \mathbf{w}\cdot\phi(\mathbf{x}) + b$, modelling the data. The learning task can be reduced to the maximisation of dual Lagrangians such as:

$$L = -\epsilon\sum_{i=1}^{m}(\alpha_i^\star + \alpha_i) + \sum_{i=1}^{m} y_i(\alpha_i^\star - \alpha_i) - \frac{1}{2}\sum_{i,j=1}^{m}(\alpha_i^\star - \alpha_i)\left(\alpha_j^\star - \alpha_j\right)K(\mathbf{x}_i,\mathbf{x}_j)$$

subject to the constraint $\sum_{j=1}^{m}\alpha_j^\star = \sum_{j=1}^{m}\alpha_j$, for instance.

2. Algorithmic Approaches to Training SVMs

All these tasks involve optimization of a quadratic Lagrangian and thus techniques from quadratic programming are most applicable including quasi-Newton, conjugate gradient and primal-dual interior point methods. 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 two categories: techniques in which kernel components are evaluated and discarded during learning and *working set* methods in which an evolving subset of data is used. 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 [8]. For binary classification (with no soft margin or bias) this is a simple gradient ascent procedure on (5) in which $\alpha_i \geq 0$ initially and the α_i are subsequently sequentially updated using:

$$\alpha_i \leftarrow \beta_i\theta(\beta_i) \qquad \text{where} \qquad \beta_i = \alpha_i + \eta\left(1 - y_i\sum_{j=1}^{m}\alpha_j y_j K(\mathbf{x}_i,\mathbf{x}_j)\right) \tag{10}$$

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 (6) this method is very easy to implement and can give a quick impression of the performance of SVMs on classification tasks. It is equivalent to Hildreth's method in Optimisation theory and can be generalised to the case of soft margins and inclusion of a bias [14]. 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 optimise 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 maximised. 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* [17] 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 Optimisation (SMO). The limiting case of decomposition is the Sequential Minimial Optimisation (SMO) algorithm of Platt [18] in which only two α_i are optimised at each iteration. The smallest set of parameters which can be optimised 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 optimised and the rest kept fixed then it is possible to derive this 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 optimise and use of an analytical expression to ensure the lagrangian increases monotonically. For the hard margin case the latter is easy to derive from the maximisation of δW with respect to the additive corrections a, b in $\alpha_i \to \alpha_i + a$ and $\alpha_j \to \alpha_j + b$, $(i \neq j)$. For the L_1 soft margin care must be taken to avoid violation of the constraints (7) leading to bounds on these corrections. The SMO algorithm has been refined to improve speed [13] and generalised to cover the above three tasks of classification [18], regression [22] and estimating densities [20]. Due to its decomposition of the learning task and speed it is probably the method of choice for training SVMs.

Model Selection. Apart from the choice of kernel the other indeterminate is the choice of the kernel parameter (e.g. σ in (3)). 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 based on theoretical arguments without use of additional validation data. As a first attempt we can use a theorem stating that the test error bound (E) is reduced as the margin γ is increased $E = R^2/\gamma^2$ where R is the radius of

the smallest ball containing the training data (in general R can be found via a QP task [1, 25]). At the optimum of (3) 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). For RBF kernels $R \simeq 1$ (the data lie on the surface of hypersphere since $\phi(\mathbf{x}) \cdot \phi(\mathbf{x}) = K(\mathbf{x}, \mathbf{x}) = 1$ from (5)). Hence an estimate for σ can be found by sequentially training SVMs on the same dataset at successively larger values of σ, evaluating E from the α_i^0 for each case and choosing that value of σ for which E is minimised. This method [5] 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 [3] is to simply rescale data in kernel space to compensate for uneven distributions. This rescaling is achieved using the eigenvalues and eigenvectors of the covariance matrix $K(\mathbf{x}_i, \mathbf{x}_j)$. A more complex strategy along these lines has also been proposed by Schölkopf et al [21] 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* cross-validation procedure. In this procedure, single elements from the data set are removed, the SVM is trained on the remaining $l-1$ elements and then tested on the removed datapoint. Under the reasonable assumption that the set of support vectors does not change it is possible to derive tight bounds on the generalisation error. Two examples of these model selection rules are the *span-rule* of Chapelle and Vapnik [3] and a rule proposed by Jaakolla and Haussler [12]. Based on recent studies with a limited number of datasets, these model selection strategies appear to work well. However, a comparative study of these different techniques and their application to a wider range of real-life datasets needs to be undertaken to establish if they are fully practical approaches.

Active Selection. So far we have considered learning strategies in which data is acquired passively. However, SVMs construct a hypothesis using a subset of the data containing the most informative patterns and thus they are good candidates for active or selective sampling techniques which seek out these patterns. Suppose the data is initially unlabeled, a good heuristic algorithm would predominantly request the labels for those patterns which will become support vectors: the labels are not required for patterns corresponding to non-support vectors in the eventual hypothesis. Active selection would be particularly important for practical situations in which the process of labelling data is expensive or the dataset is large and unlabelled.
During the process of active selection the information gained from an example depends both on the position (available information) and on its label (unavailable information before querying). Thus we must follow a heuristic strategy to maximise the gain at each step. Firstly we note (Figure 1) that querying a point within the margin band *always* guarantees a gain whatever the label of the point. We do not gain by querying a point outside the band unless the current hypothesis predicts the label incorrectly. The best points to query are indeed those points which are closest to the current hyperplane [2]. Intuitively

this makes sense since these are most likely to be maximally ambiguous with respect to the current hypothesis and hence the best candidates for ensuring that the information received is maximised. Hence a good strategy [2] is to start by requesting the labels for a small initial set of data and then successively querying the labels of points closest to the current hyperplane. Active selection works best if the hypothesis modelling the data is sparse i.e. there are comparatively few support vectors to find (Figure 2(right)).

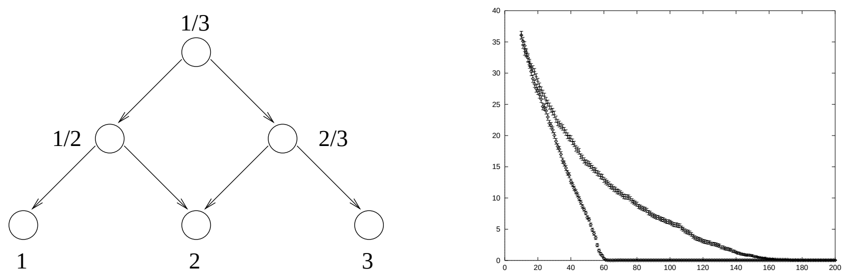

Figure 2: **Left**: a multi-class classification problem can be reduced to a series of binary classification tasks. **Right**: Generalisation error (y-axis) as a percentage versus number of patterns (x-axis) for random selection (top curve) and active selection (bottom curve). Active selection outperforms random selection especially when the hypothesis is sparse (for this dataset 28% of patterns were support vectors).

3. Conclusion

Since the learning task involves optimisation of a quadratic function SVMs provide a unique solution. The approach is general in that they can be applied to a wide range of machine learning tasks (e.g. classification, regression and novelty detection) and can be used to generate many possible learning machine architectures (RBF networks, feedforward neural networks) through the choice of kernel. Kernel substitution of the inner product is, indeed, a powerful idea separate from the concept of the margins and it can be used to define many other types of learning machines [16, 11] distinct from SVMs. Above all SVMs perform well in practice and consequently can be expected to develop as an important tool for future applications of machine learning.

References

[1] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2, p. 121-167, 1998.

[2] C. Campbell, N. Cristianini and A. Smola. Instance selection using support vector machines, submitted to *Machine Learning,* 1999.

[3] O. Chapelle and V. Vapnik. Model selection for support vector machines, to appear in *Advances in Neural Information Processing Systems* 12, ed. S.A. Solla, T.K. Leen and K.-R. Muller, MIT Press, 2000.

[4] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning* 20, p. 273-297, 1995.

[5] N. Cristianini, C. Campbell and J. Shawe-Taylor. Dynamically adapting kernels in support vector machines, *Advances in Neural Information Processing Systems* 11, ed. M. Kearns, S. A. Solla, and D. Cohn, MIT Press, p. 204-210, 1999.

[6] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other Kernel-based Learning Methods*, Cambridge University Press, to appear January 2000.

[7] J.Platt, N. Cristianini and J. Shawe-Taylor. Large Margin DAGS for Multiclass Classification, *Advances in Neural Information Processing Systems,* 12 ed. S.A. Solla, T.K. Leen and K.-R. Muller, MIT Press, 2000.

[8] T.-T. Friess, N. Cristianini and C. Campbell. The kernel adatron algorithm: a fast and simple learning procedure for support vector machines. *15th Intl. Conf. Machine Learning*, Morgan Kaufman Publishers, p. 188-196, 1998.

[9] Cf: http://www.clopinet.com/isabelle/Projects/SVM/applist.html

[10] D. Haussler. Convolution Kernels on Discrete Structures, UC Santa Cruz Technical Report UCS-CRL-99-10, 1999.

[11] R. Herbrich, T. Graepel and C. Campbell. Bayesian learning in reproducing kernel Hilbert spaces, submitted to *Machine Learning,* 1999.

[12] T. Jaakolla and D. Haussler. Probabilistic kernel regression models, in *Proceedings of the 1999 Conference on AI and Statistics,* 1999.

[13] S. Keerthi, S. Shevade, C. Bhattacharyya and K. Murthy. Improvements to Platt's SMO algorithm for SVM classifier design. Tech Report, Dept. of CSA, Banglore, India, 1999.

[14] D. Luenberger. *Linear and Nonlinear Programming.* Addison-Wesley, 1984.

[15] E. Mayoraz and E. Alpaydin. Support Vector Machines for Multiclass Classification, *Proceedings of the International Workshop on Artifical Neural Networks (IWANN99)*, IDIAP Technical Report 98-06, 1999.

[16] S. Mika, G. Ratsch, J. Weston, B. Schölkopf and K.-R. Muller. Fisher Discriminant Analysis with Kernels. *Proceedings of IEEE Neural Networks for Signal Processing Workshop* 1999, 8 pages, 1999.

[17] E. Osuna and F. Girosi. Reducing the Run-time Complexity in Support Vector Machines, in B. Schölkopf, C.Burges and A. Smola (ed.), *Advances in Kernel Methods: Support Vector Learning,* MIT press, Cambridge, MA, p. 271-284, 1999.

[18] J. Platt. Fast training of SVMs using sequential minimal optimisation. in B. Schölkopf, C.Burges and A. Smola (ed.), *Advances in Kernel Methods: Support Vector Learning*, MIT press, Cambridge, MA, p. 185-208, 1999.

[19] J. Shawe-Taylor. Confidence estimates of classification accuracy on new examples. In S. Ben-David (ed.), EuroCOLT97, *Lecture Notes in Artificial Intelligence*, 1208, p.260-271, 1997.

[20] 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.

[21] B. Schölkopf, J. Shawe-Taylor, A. Smola and R. Williamson. Kernel-dependent support vector error bounds, *Ninth International Conference on Artificial Neural Networks*, IEE Conference Publications No. 470, p. 304 - 309, 1999.

[22] A. Smola and B. Schölkopf. A tutorial on support vector regression. NeuroColt2 TR 1998-03, 1998.

[23] D. Tax and R. Duin. Data domain description by Support Vectors, in *Proceedings of ESANN99*, ed. M Verleysen, D. Facto Press, Brussels, p. 251-256, 1999.

[24] V. Vapnik and O. Chapelle. Bounds on error expectation for SVMs, submitted to *Neural Computation*, 1999

[25] V. Vapnik. *The Nature of Statistical Learning Theory,* Springer, N.Y., 1995.

[26] V. Vapnik. *Statistical Learning Theory.* Wiley, 1998.

[27] C. Watkins. Dynamic Alignment Kernels, Technical Report, UL Royal Holloway, CSD-TR-98-11, 1999.

[28] J. Weston and C. Watkins. Multi-Class Support Vector Machines, in *Proceedings of ESANN99,* ed. M Verleysen, D. Facto Press, Brussels, p. 219-224, 1999.