Simple Learning Algorithms for Training Support Vector Machines

Colin Campbell
Dept. of Engineering Mathematics,
University of Bristol, BS8 1TR,
United Kingdom
c.campbell@bris.ac.uk

Nello Cristianini
Dept. of Engineering Mathematics
University of Bristol, BS8 1TR,
United Kingdom
nello.cristianini@bris.ac.uk

Abstract

Support Vector Machines (SVMs) have proven to be highly effective for learning many real world datasets but have failed to establish themselves as common machine learning tools. This is partly due to the fact that they are not easy to implement, and their standard implementation requires the use of optimization packages. In this paper we present simple iterative algorithms for training support vector machines which are easy to implement and guaranteed to converge to the optimal solution. Furthermore we provide a technique for automatically finding the kernel parameter and best learning rate. Extensive experiments with real datasets are provided showing that these algorithms compare well with standard implementations of SVMs in terms of generalisation accuracy and computational cost, while being significantly simpler to implement.

1 Introduction

Since their introduction by Vapnik and coworkers [38, 7], Support Vector Machines (SVMs) have been successfully applied to a number of real world problems such as handwritten character and digit recognition [27, 6, 17, 38], face detection [22] text categorisation [36] and object detection in machine vision [25]. They manifest an impressive resistance to overfitting, a feature which can be explained using VC theory [37, 38], and their training is performed by maximising a convex functional, which means that there is a unique solution that can always be found in polynomial time. For simple binary classification tasks they work by mapping the training points into a high-dimensional feature space where a separating hyperplane can be found which has a maximal distance from the two classes of labelled points. This minimizes the effective VC dimension of the system enforcing good generalization [37, 38]. The task of finding the maximal margin hyperplane is reduced to a quadratic programming (QP) problem which can be solved using optimization routines.

Despite a number of practical successes, SVMs have not yet become established as a standard tool in machine learning, whereas systems such as neural networks and decision trees became widely used within a few years of their introduction. The reason for this is the difficulty of implementing such systems since solution of a complex quadratic programming problem is required. Thus the editors of [30] note: "Despite the fact that the perceptron was invented in the sixties, interest in feed-forward neural networks only took off in the eighties, due largely to a new training algorithm. Backpropagation is conceptually simple and, perhaps more important, easy to implement. We believe that research into Support Vector Machines has been similarly hampered by the fact that training requires solving a quadratic programming problem which is a notoriously difficult business". Furthermore, standard QP programming routines have substantial memory resource requirements and large datasets require additional techniques such as chunking (breaking the QP problem into a series of simpler QP tasks) [23].

In this paper we address these problems by proposing extremely simple training algorithms for SVMs which are sufficiently fast for practical application. Indeed, for most datasets investigated learning times scale sub-quadratically with the number of patterns, and the amount of memory required is extremely small since the algorithms are inherently online and chunking is not necessary. These algorithms can be derived from first principles and come with theoretical guarantees of fast convergence to the optimal solution. Furthermore, we show that it is possible to perform automatic model selection without the use of a validation set.

Other recent results [24] similarly address the problem of speed (time complexity) and implementational simplicity. Such systems are extremely interesting, and we will compare these algorithms with the procedures outlined here at the end of this paper. Furthermore, other algorithms have recently been proposed which merge perceptron-like rules with kernel methods, as proposed here, though using very different types of architecture[12]

The paper is organized as follows. In section 2 we present an overview of Support Vector Machines, in section 3 we introduce the new algorithms and in section 4 we provide an extensive experimental study for real and artificial datasets.

2 Support Vector Machines

Support Vector machines implement complex decision rules by using a non-linear function ϕ to map training points to a high-dimensional *feature space* where the labelled points are separable. A separating hyperplane is found which maximizes the distance between itself and the nearest training points (this distance is called the *margin*). The hyperplane is, in fact, represented as a linear combination of the training points. Theoretical results exist from VC theory [38, 31], which guarantee that the solution found will have high predictive power, in the sense that it minimizes an upper bound on the test error (a survey covering the generalization power of SV machines can be found in [3]).

Let $S = \{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}$ be a sample of points $x_i \in \mathcal{X}$ labelled by $y_i \in \{-1, +1\}$. Consider a hyperplane defined by (w, θ), where w is a weight vector and θ a bias. Let $S = (X, Y)$ be a labeled sample of inputs from $\mathcal{X}$ that has empty intersection with the hyperplane, so that:

$$\gamma = \min_{x \in X} |\langle x, w \rangle + \theta| > 0$$

This distance is the *margin* of the hyperplane w with respect to the sample S. We also say that the hyperplane is in canonical form with respect to the sample if:

$$\min_{x \in X} |\langle x, w \rangle + \theta| = 1$$

It is possible to prove that for canonical hyperplanes:

$$\gamma = 1/||w||_2$$

The following theorem holds:

Theorem[31]: Suppose inputs are drawn independently according to a distribution whose support is contained in a ball in $\Re^n$ centered at the origin, of radius R. If we succeed in correctly classifying m such inputs by a canonical hyperplane with $||w|| = 1/\gamma$ and $|\theta| \leq R$, then with confidence $1 - \delta$ the generalization error will be bounded from above by

$$\epsilon(m, \gamma) = \frac{1}{m} \left(k \log \left(\frac{8em}{k} \right) \log (32m) + \log \left(\frac{8m}{\delta} \right) \right)$$

where $k = \lfloor 577R^2/\gamma^2 \rfloor$.

This quantity which upper bounds the generalization error does not depend on the dimension of the input space, and this is why SVMs can use high dimensional spaces without overfitting the data. Two main ideas (data-dependent representations and kernels) make it possible to efficiently deal with high dimensional feature spaces. The first is based on the identity:

$$\sum_{i=1}^{N} w_i \phi_i(x) + \theta = \sum_{k=1}^{m} \alpha_k \phi(x_k) \phi(x) + \theta$$

which provides an alternative, data-dependent, representation of the hypothesis itself, and the other is the use of kernels:

$$K(x', x) = \sum_i \phi_i(x') \phi_i(x)$$

which give the dot product of the images of two vectors in the feature space [1]. Appropriate Kernels implicitly describing this mapping must satisfy Mercer's condition [38], i.e. for any $g(x)$ for which:

$$\int g(x)^2 dx < \infty$$

then:

$$\int K(x,x')g(x)g(x')dxdx' \geq 0$$

A common choice are Radial Basis Functions (RBF) such as Gaussians:

$$K(x,x') = e^{-||x-x'||^2/2\sigma^2}$$

or polynomial kernels:

$$K(x,x') = (\langle x,x' \rangle + 1)^d$$

which always satisfy such conditions. The tunable parameter in the kernel controls model complexity and it is usually chosen by means of a validation set. After mapping to feature space the next step is to find the maximal margin hyperplane. A separating hyperplane can be written:

$$(\mathbf{w} \cdot \mathbf{x}) + \theta = 0$$

For points labelled with $y_i = \pm 1$ maximising the expression $y_i(\mathbf{w} \cdot \mathbf{x} + \theta)/||\mathbf{w}||_2$ will maximise the margin. Implicitly fixing the scale of the numerator with:

$$y_i(\mathbf{w} \cdot \mathbf{x} + \theta) \geq 1$$

the task is therefore to minimise $||w||^2$. Thus the maximal margin can be found by maximising the Lagrangian:

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

The derivatives with respect to θ and $\mathbf{w}$ give:

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \qquad\qquad \mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$$

Substituting the latter equation in the Lagrangian gives the dual representation of the Lagrangian:

$$L = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{1}$$

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

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

The α_i are Lagrange multipliers, one for each training point. When the maximal margin hyperplane is found, only points which lie closest to the hyperplane have $\alpha_i > 0$ and these points are called *support vectors.* All other points have $\alpha_i = 0$. This means that the representation of the hypothesis is given only by those points which lie closest to the hyperplane and they are the most informative patterns in the data. Their number can also be used to give an independent bound on the reliability of the hypothesis [3]. The resulting decision function can be written as:

$$f(x) = \mathrm{sign}\left(\sum_{i \in \mathtt{SV}} y_i \alpha_i^o K(x, x_i) + \theta\right)$$

where α_i^o is the solution of the constrained maximization problem and $\mathtt{SV}$ represents the indexes of the support vectors.

When the data are not linearly separable in the feature space, a more general setting can be used, where a certain number of training points are allowed to lie close to the decision boundary or even to be misclassified. For noisy datsets it is also beneficial to allow some misclassified points. This approach, using a *soft margin*, gives rise to a slightly different optimization problem, and it is possible to prove that this is equivalent to placing an upper bound on the maximum size of the parameters α_i mentioned above. The problem is again to maximize Lagrangian (1) but with the constraints:

$$0 \leq \alpha_i \leq C$$

where C controls the trade-off between training error and generalisation ability and its value is chosen by means of a validation set. C and the kernel parameter are the only two parameters which have to be hand-tuned while training SVMs.

Such a scheme has found to be very resistent to overfitting in many classification problems [27, 7, 38]. However, training these systems is non-trivial and computationally expensive requiring the use of optimization packages. In the next section we will present some simple training algorithms for SVMs which are guaranteed to deliver the optimal solution and which can be easily implemented.

3 Simple Training Algorithms for SV Machines

The most obvious way of maximizing a concave Lagrangian under linear constraints is gradient ascent. The lagrangian to be maximised is:

$$L(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j) - \lambda \sum_i \alpha_i y_i$$

where the last term implements the constraint condition $\sum_i \alpha_i y_i = 0$. We will maximise this Lagrangian using stochastic gradient ascent based on the derivative of the Lagrangian, thus:

$$\delta\alpha_k = \eta \frac{\partial L}{\partial \alpha_k} = \eta \left(1 - y_k \sum_j \alpha_j y_j K(x_k, x_j) - \lambda y_k\right) \tag{2}$$

Furthermore, we will enforce the constraints $\alpha_i \geq 0$ by setting $\alpha_i \to 0$ for those α_i's which would become negative. Let us consider the change in this Lagrangian due to an updating $\alpha_k \to \alpha_k + \delta\alpha_k$ for a particular pattern k:

$$\begin{aligned}
\delta L &= L(\alpha_k + \delta\alpha_k) - L(\alpha_k) && (3)\\
&= \delta\alpha_k \left(1 - y_k \sum_j \alpha_j y_j K(x_k, x_j) - \lambda y_k\right) - \frac{1}{2}(\delta\alpha_k)^2 K(x_k, x_k) && (4)\\
&= \left[\frac{1}{\eta} - \frac{K(x_k, x_k)}{2}\right](\delta\alpha_k)^2 && (5)
\end{aligned}$$

Then it is apparent that $\delta L > 0$ provided:

$$2 > \eta K(x_k, x_k) > 0$$

For a Gaussian kernel:

$$K\left(x_i, x_j\right) = \exp\left(-\frac{\left|\left|x_i - x_j\right|\right|^2}{2\sigma^2}\right)$$

$K(x_k, x_k) = 1$ and consequently the algorithm converges to the maximum of L provided:

$$2 > \eta > 0$$

On the other hand for polynomial kernels of the form:

$$K(x_i, x_j) = ((x_i \cdot x_j) + 1)^d$$

the bound depends on the training data and it is determined from the L_2 norm for each pattern:

$$\frac{2}{(||x_k|| + 1)^d} > \eta_k > 0$$

If we substitute (2) in (5) we obtain:

$$\delta L = \left(\eta - \frac{K(x_k, x_k)}{2}\eta^2\right)\left(1 - y_k \sum_j \alpha_j y_j K(x_k, x_j) - \lambda y_k\right)^2$$

hence, if α_k is not set to zero during the current α_k-update, then we can optimise δL with respect to η giving:

$$\eta = \frac{1}{K(x_k, x_k)}$$

In general, the optimal value for the learning rate η^o is pattern dependent. For example, for polynomial kernels:

$$\eta_k^o = \frac{1}{(||x_k|| + 1)^d}$$

However, for Gaussian kernels:

$$\delta L = \left(\eta - \frac{1}{2}\eta^2\right)\left(1 - y_k \sum_j \alpha_j y_j K(x_k, x_j) - \lambda y_k\right)^2$$

and consequently the optimum occurs when $\eta^o = 1$. For the KA algorithm (described in the next section) certain α_k will be set to zero during the learning phase and hence the appropriate η is the value satisfying $\alpha_k + \eta(1 - y_k \sum_j \alpha_j y_j K(x_j, x_k)) = 0$ rather than η^o. Consequently, if the large majority of patterns are support vectors the optimal choice for η is indeed 1. This is illustrated in Fig. 1 for learning the mirror symmetry problem (see section 4.1). This Figure shows the number of epochs required to achieve a margin of 0.99 (vertical axis) versus η (horizontal axis) with the same data. The graph clearly illustrates the fact that convergence is only achieved on the range $2 > \eta > 0$ (oscillations occur outside this region) and the best choice for η is $\eta = 1$. If the target concept is sparse and many α_k are set to zero then the best choice for η is less than 1. This is illustrated in Fig. 2 for learning sonar classification data (see section 4.2).

3.1 The KA (Kernel-Adatron) Algorithm

In this section we present an algorithm which uses the above gradient ascent routine to maximise the margin in Feature Space. The algorithm we will consider is conceptually similar to a perceptron-like algorithm called the *Adatron* [2]. Since the proposed algorithm recasts the Adatron in the Feature Space of Support Vector Machines we have therefore called it the *Kernel-Adatron*, or KA algorithm. Dropping the condition $\sum_{i=1}^m \alpha_i y_i = 0$ is equivalent to forcing the hyperplane to pass through the origin in Feature Space, a choice that can be motivated by the treatment of *semi-parametric* models provided in [28]. Given that Feature Space is high-dimensional, this is not a particular restriction for many problems and, indeed, generalisation can be little affected as we illustrate with some examples in section 4. Thus we will begin by outlining the KA algorithm for training an SVM without a bias using gradient ascent in α-space (this algorithm was developed jointly with Thilo Friess [11]):

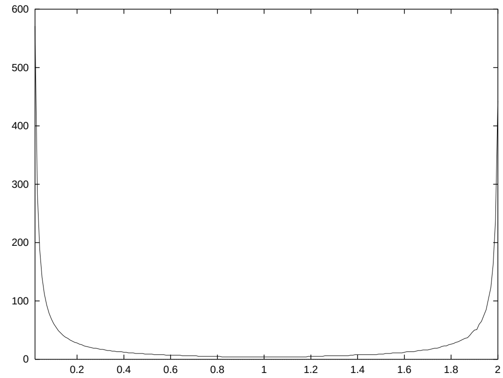

Figure 1: The number of epochs (y-axis) required to reach a margin of 0.99 versus η (x-axis) for the mirror symmetry problem with 30 inputs and $m = 200$ training examples (Gaussian kernels with $\sigma = 7.0$).

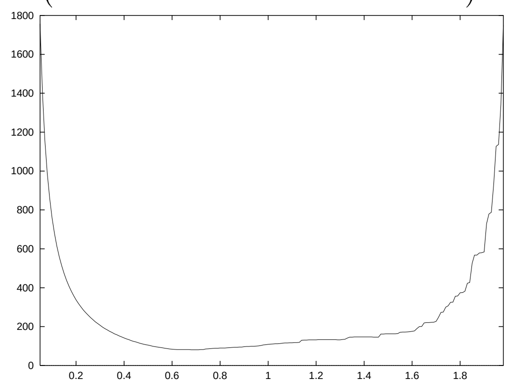

Figure 2: The number of epochs (y-axis) required to reach a margin of 0.99 versus η (x-axis) for the sonar classification problem (Gaussian kernels with $\sigma = 1.0$).

KA algorithm without bias

1. Initialise $\alpha_i^0 = 0$.

2. For $i = 1, m$ execute steps **3** and**4** below.

3. For labelled points (x_i, y_i) calculate:

$$z_i = \sum_{j=1}^{m} \alpha_j y_j K(x_i, x_j)$$

4. Calculate $\delta\alpha_i^t = \eta(1 - z_i y_i)$:

4.1. If $(\alpha_i^t + \delta\alpha_i^t) \leq 0$ then $\alpha_i^t = 0$

4.2. If $(\alpha_i^t + \delta\alpha_i^t) > 0$ then $\alpha_i^t \leftarrow \alpha_i^t + \delta\alpha_i^t$

5. If a maximum number of iterations is exceeded or the margin:

$$\gamma = \frac{1}{2}\left(\min_{\{i|y_i=+1\}}(z_i) - \max_{\{i|y_i=-1\}}(z_i)\right)$$

is approximately 1 then **stop**, otherwise return to **2** for the next epoch t.

To reduce the learning time it is best to evaluate the kernel matrix $K(x_i, x_j)$ during initialisation in step **1**. From the discussion at the start of this section we notice that an increase in L occurs irrespective of the choice of λ except when $\delta\alpha_i = 0$. Furthermore, the final λ-value is the bias since, from the stationarity condition $\delta\alpha_k = 0$ when the maximum has been found:

$$1 - y_k \sum_j \alpha_j y_j K(x_k, x_j) - \lambda y_k = y_k(y_k - \sum_j \alpha_j y_j K(x_k, x_j) - \lambda) = 0$$

consequently the bias can be found by a subprocess involving iterative adjustment of the λ based on the gradient of L with respect to λ:

$$\lambda^t = \lambda^{t-1} - \nu \sum_i \alpha_i^{t-1} y_i$$

at each epoch t. How do we find a good value of the parameter ν to ensure fast convergence? Given that the Lagrangian is concave with a unique solution satisfying $\sum_i \alpha_i y_i = 0$ we can quickly find λ using a variable ν parameter derived from the secant method, for example. Thus if t labels the epoch and $\omega^t = \sum_j \alpha_j^t y_j$, the algorithm is as follows:

KA algorithm with bias

1. Initialise $\alpha_i^0 = 0$.

2. For $t = 1, t_{\max}$ execute steps **3** to **8** below.

3. If $t = 0$ then

$\lambda^0 = \mu$

elseif $t = 1$ then
$\lambda^1 = -\mu$
else
$\lambda^t = \lambda^{t-1} - \omega^{t-1}\left(\frac{\lambda^{t-1}-\lambda^{t-2}}{\omega^{t-1}-\omega^{t-2}}\right)$
endif (see comment below).

4. For $i = 1, m$ execute steps **5** and **6** below.

5. For labelled points (x_i, y_i) calculate:

$$z_i = \sum_{j=1}^{m} \alpha_j y_j K(x_i, x_j)$$

6. Calculate $\delta\alpha_i^t = \eta(1 - z_i y_i - \lambda^t y_i)$:

6.1. If $(\alpha_i^t + \delta\alpha_i^t) \leq 0$ then $\alpha_i^t = 0$

6.2 If $(\alpha_i^t + \delta\alpha_i^t) > 0$ then $\alpha_i^t \leftarrow \alpha_i^t + \delta\alpha_i^t$

7. Calculate $\omega^t = \sum_j \alpha_j^t y_j$

8. If a maximum number of iterations is exceeded or the margin:

$$\gamma = \frac{1}{2}\left(\min_{\{i|y_i=+1\}}(z_i) - \max_{\{i|y_i=-1\}}(z_i)\right)$$

is approximately 1 then **stop**, otherwise return to **2** for the next t.

The decision function is then:

$$f(x) = sign\left(\sum_{i\in SV} y_i\overline{\alpha}_i K(x, x_i) + \overline{\lambda}\right)$$

where $\overline{\alpha}_i$ and $\overline{\lambda}$ are the last values of α_i^t and λ^t found during the final iteration $t = t_{\max}$. In step **3** an error trap is needed to avoid divergence if $\omega^{t-1} - \omega^{t-2} = 0$. Thus the magnitude of $\epsilon = \omega^{t-1} - \omega^{t-2}$ should be lower bounded by a small number or ν upper bounded. Convergence of the secant method is not affected by the choice of μ and hence this parameter can be set at will (e.g. $\mu = 0.1$). The method is robust and gives exactly the same solution as that found by standard Quadratic Programming routines.

3.2 Soft Margins

The soft margin variation of Support Vector Machines, designed to tolerate training errors, is equivalent to solving the same optimization problem with the additional constraint that all the Lagrange multipliers lie in a box of side C, where C is usually chosen by means of a validation set. This can be done by simply adding the line:

$$\texttt{if}\ \ (\alpha_i + \delta\alpha_i) > C\ \texttt{then}\ \ \alpha_i = C$$

immediately following the updating rule in the above algorithm. Given the concavity of the functional, the algorithm is still guaranteed to converge to the solution. When a soft margin is used the margin, γ is evaluated using those patterns for which $0 < \alpha_i < C$ i.e.:

$$\gamma = \frac{1}{2}\left(\min_{\{i|y_i=+1,\alpha_i<C\}}(z_i) - \max_{\{i|y_i=-1,\alpha_i<C\}}(z_i)\right)$$

3.3 Theoretical Analysis of KA

The Kernel-Adatron algorithm is theoretically guaranteed to converge in a finite number of steps to the maximal margin, provided that the data are linearly separable in feature space with a margin $\gamma > 0$. This can be easily shown in two steps, by noting that: (1) all the fixed points of KA are Kuhn-Tucker points and vice versa, and (2) KA always converges to a (unique) fixed point.

The first observation is trivially verified by substituting the Kuhn-Tucker conditions for a maximal margin:

$$\begin{aligned} \alpha_i &> 0 \Leftrightarrow \gamma_i = 1 \\ \alpha_i &= 0 \Leftrightarrow \gamma_i > 1 \end{aligned}$$

into the KA updating rule from which it follows that the optimal margin is a fixed point. Conversely, by imposing $\delta\alpha_i = 0 \, \forall i$ the Kuhn-Tucker conditions are obtained.

Convergence to a fixed point is apparent from the concavity of L, or alternatively by noting firstly that L is upper bounded by the quantity:

$$L(\alpha^o) = \frac{1}{2\gamma^2}$$

where α^o are the lagrange multipliers of the optimal hyperplane [38] and, secondly, that the lagrangian increases monotonically at each update, provided a suitable value of η is chosen.

The rate of convergence of the Adatron was also studied in by Anlauf and Biehl [2] and found to be exponentially fast in the number of epochs. In section 4 we give experimental results suggesting that this is also the case for KA.

3.4 Adaptive Kernels

The choice of the kernel-parameter determines the nature of the nonlinear mapping to the high dimensional feature space, which in the case of gaussian kernels is infinite-dimensional. In is well known from experimentation that, provided the margin is maximized, the kernel-parameter σ controls the model complexity, in the sense that a too rich a kernel-space will let the machine overfit, and too poor a kernel-space will not be sufficient to separate the points in feature space or provide an efficient solution (see Fig. 3).

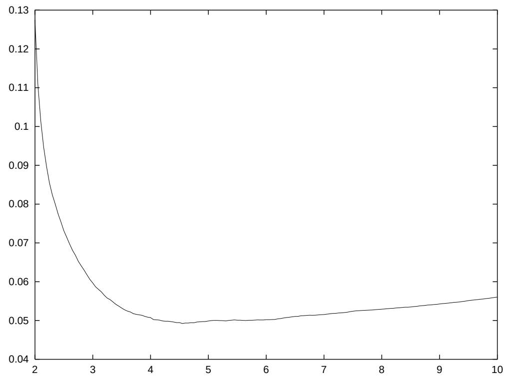

Figure 3: Generalisation error (y-axis) as a function of σ (x-axis) for the mirror symmetry problem (for Gaussian kernels with zero training error and maximal margin, $m = 200$, $n = 30$ and averaged over 10^5 unseen examples including repeats)

The following theorem, which we prove elsewhere [9], states that the margin of the optimal hyperplane is a smooth function of the kernel parameter and so is the upper bound on the generalisation error.

Theorem [8, 9] **:** The *margin* γ of SV machines depends continuously on the kernel parameter σ.

This means that, when the margin is optimal, small variations in the kernel parameter will produce small variations in the margin (and in the bound on the generalisation error), or $\gamma_\sigma \approx \gamma_{\sigma+\delta\sigma}$. After updating the σ, the system will still be in a sub-optimal position. Given that Kernel-Adatron can be regarded as a gradient ascent algorithm that maximizes the Kuhn-Tucker Lagrangian, little computational effort will be needed to bring the system back to a maximal margin position, starting from such a sub-optimal state.

For example, for Gaussian kernels this suggests the following strategy:

Kernel Selection Procedure

1. Initialize σ to a very small value

2. Start the KA training procedure described in section 2.

3. If the margin is maximized, then

3.1. Compute the VC bound (or observe the validation error)

3.2. Increase the kernel parameter: $\sigma \leftarrow \sigma + \delta\sigma$ else goto step **2**

4. Stop when a predetermined value of σ is reached.

Experimental results presented elsewhere [9, 8] indicate that this procedure is fast and gives a good estimate for the kernel parameter.

4 Experimental Results

We have evaluated the performance of the KA algorithm on a number of standard classification datasets, both artificial and real. The artificial datasets include mirror symmetry [19], n-parity [19] and the two-spirals problem [11]. The real world datasets include a sonar classification problem [14], the Wisconsin breast cancer dataset [35] and a database of handwritten digits collected by the US Postal Service [17]. As examples of the improvements with generalisation ability which can be achieved with a soft margin we will also describe experiments with the ionosphere and Pima Indians diabetes datasets from the UCI Repository [4]. Though we have successfully used other kernels with KA we will only describe experiments using Gaussian kernels in this section. We will predominantly use the KA algorithm with bias though we add some comments on the simpler version without bias.

4.1 Artificial datasets: mirror symmetry, n-parity and the two-spirals problem

In the mirror symmetry problem [19] the output y is a 1 if the input pattern x (with components from {-1, +1}) is exactly symmetrical about its centre, otherwise the output is a -1. For randomly constructed input strings the output would be a -1 with a high probability. Consequently the labels ± 1 are selected with a 50% probability and the first half of the input string is randomly constructed from components in {-1, +1} (both selected with a 50% probability) and the second half of the string is symmetrical or random depending on the target value given. Generalisation was evaluated using a test set drawn from the same distribution (eliminating any instances for which the input string is identical to a member of the training set). A plot of the generalisation error versus σ was given earlier (Fig. 3) and in Fig. 4 and Fig. 5 we show the evolution of the margin and generalisation error versus number of epochs for learning with Gaussian kernels with $\sigma = 5.0$.

For neural networks, mirror symmetry is a non-linearly separable problem though a solution only requiring two hidden nodes is known [19]. Similar hard tasks for a neural network are n-parity and the two-spirals problem where the labelled points of each class are heavily intermeshed. As we report elsewhere [11] it is straightforward to solve these problems using the KA algorithm. For example, for n-parity a solution was found in 1 epoch for $n = 3$ to $n = 6$ though it took several epochs to maximise the margin [11].

4.2 Real-life datasets: Sonar classification, Wisconsin breast cancer dataset and the USPS dataset.

Sonar Classification: The sonar classification problem of Gorman and Sejnowski [14] consists of 208 instances each with 60 attributes (excluding the labels) representing returns from a roughly cylindrical rock or a metal cylinder. This dataset is equally divided into training and test sets. For the aspect-angle dependent dataset these authors trained a standard back-propagation neural network with 60 inputs and 2 output nodes. Experiments were performed with up to 24 hidden nodes and each neural network was trained for 300 epochs through the training set. The best result was achieved with 12 hidden nodes and a generalisation performance of 90.4%.

For the KA algorithm a plot of σ against generalisation error gives a best generalisation performance of 92.3% by comparison. Figure 6 illustrates the margin evolution for Gaussian kernels with $\sigma = 1.0$ and a learning rate $\eta = 1.0$. We also plot the generalisation error versus number of epochs (Fig. 7) and evolution of the bias (Fig. 8) in this instance. Interestingly, the bias does not appear to be of benefit for this problem since it was possible to achieve a generalisation performance of 95.2% without the bias [11].
The Wisconsin breast cancer dataset: The Wisconsin breast cancer

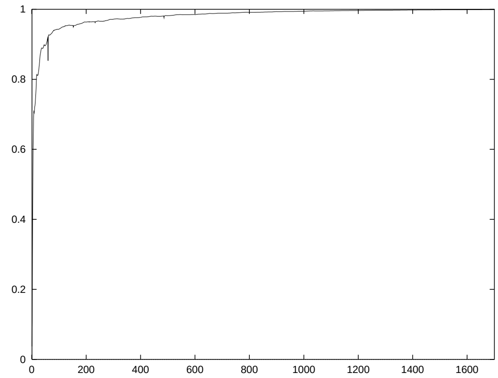

Figure 4: Margin evolution (y-axis) against number of epochs (x-axis) for the mirror symmetry problem. Gaussian kernels were used with $\sigma = 5.0$, $\eta = 1.0$ and $\mu = 5.0$.

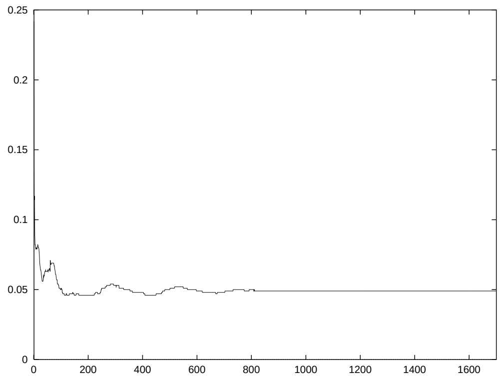

Figure 5: Generalisation error (y-axis) against number of epochs (x-axis) for the mirror symmetry problem. Gaussian kernels were used with $\sigma = 5.0$, $\eta = 1.0$ and $\mu = 5.0$.

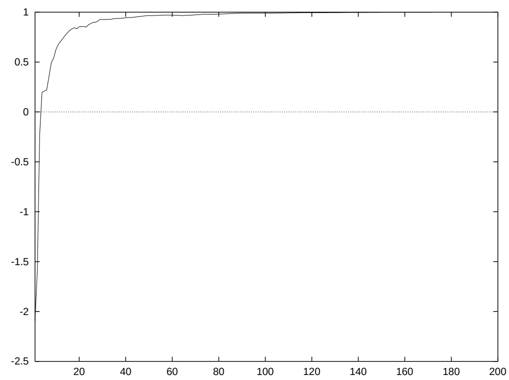

Figure 6: Margin evolution (y-axis) against number of epochs (x-axis) for the sonar classification problem. Gaussian kernels were used with $\sigma = 1.0$, $\eta = 1.0$ and $\mu = 5.0$.

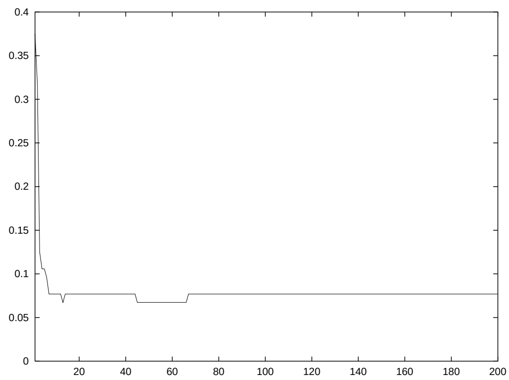

Figure 7: Generalisation error (y-axis) against number of epochs (x-axis) for the sonar classification problem. Gaussian kernels were used with $\sigma = 1.0$, $\eta = 1.0$ and $\mu = 5.0$.

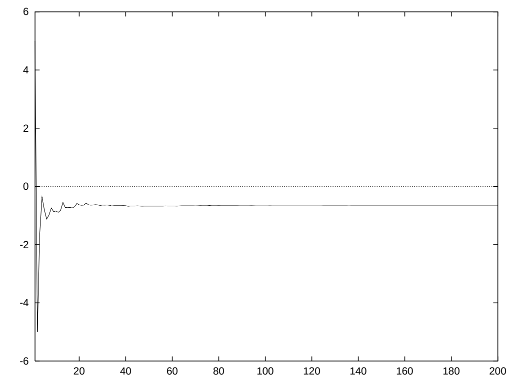

Figure 8: Bias (y-axis) against number of epochs (x-axis) for the sonar classification problem. Gaussian kernels were used with $\sigma = 1.0$, $\eta = 1.0$ and $\mu = 5.0$.

dataset contains 699 patterns with 10 attributes for a binary classification task (the tumour is malignant or benign). This dataset has been extensively studied by other authors. CART gives a generalisation of 94.2%, an RBF neural network gave 95.9%, a linear discriminant method gave 96.0% and a multi-layered neural network (trained via Back-Propagation) 96.6% (all the results have been obtained using 10-fold cross-validation [35]). Our optimal test performance was 98.5% by comparison though our pre-processing and handling of the missing 16 missing values may be different from the methods used in previous studies. For purposes of illustration we plot the evolution of the margin, generalisation error and bias for one split of the dataset in Figures 9, 10 and 11 respectively.
The USPS dataset: The benchmarking of SVMs has traditionally been performed using the database of handwritten digits for US Postal Codes [17, 29]. This dataset consists of a training set of 7,291 examples and a test set of 2,007. Each digit is given by a 16×16 vector with components which lie in the range -1 to 1. In this experiment we have performed two-class classification i.e. separating a particular digit from the others. To find suitable values for σ the training set was split into a smaller training set of 6,000 examples and a validation set of 1,291. The best value of σ was found by evaluating performance on the validation set across the range $(1, 10)$. The full training set of 7,291 was then used with the selected value of σ to train the system to classify each digit.

The results are shown in the table below where the last column shows the best value of σ found from the validation study. The other columns show the number of errors on the test set of 2,007 examples for the KA algorithm (without bias), an RBF neural network and a Support Vector Machine (SVM/KA with bias) [29].

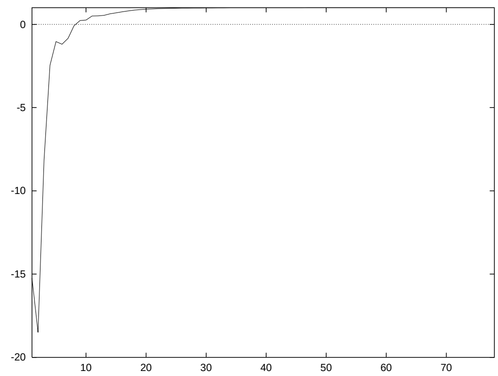

Figure 9: Margin evolution (y-axis) against number of epochs (x-axis) for the Wisconsin breast cancer dataset. Gaussian kernels were used with $\sigma = 3.0$, $\eta = 1.0$ and $\mu = 50.0$.

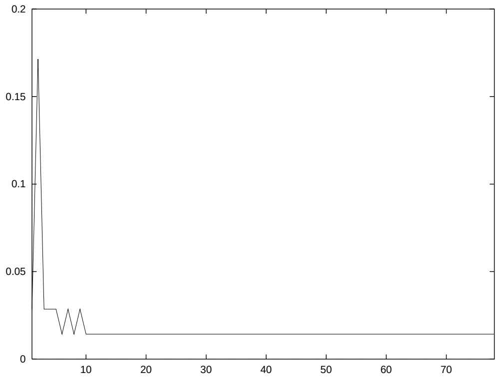

Figure 10: Generalisation error (y-axis) against number of epochs (x-axis) for the Wisconsin breast cancer dataset. Gaussian kernels were used with $\sigma = 3.0$, $\eta = 1.0$ and $\mu = 50.0$.

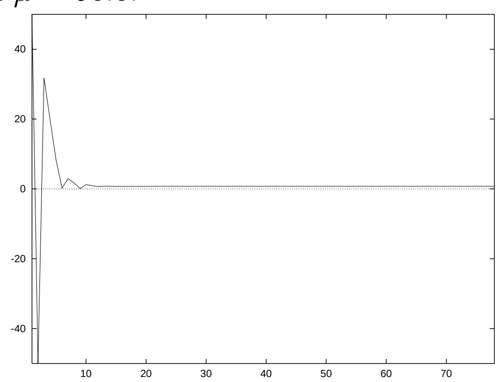

Figure 11: Bias (y-axis) against number of epochs (x-axis) for the Wisconsin breast cancer dataset. Gaussian kernels were used with $\sigma = 3.0$, $\eta = 1.0$ and $\mu = 50.0$.

Digit	RBF	SVM	KA	σ
0	20	16	13	1.8
1	16	8	10	1.6
2	43	25	21	2.4
3	38	19	24	2.0
4	46	29	26	4.0
5	31	23	19	1.8
6	15	14	15	2.4
7	18	12	11	2.8
8	37	25	26	3.2
9	26	16	14	1.6

The absence of a bias does not appear to make a difference for this particular dataset and generally SVMs with Gaussian kernels compare favourably with an RBF network. We also note that chunking was not required because the online nature of KA means only one pattern is considered at a time. This is an advantage over QP algorithms which have substantial memory requirements and which therefore require chunking for large datasets.

4.3 Use of a Soft Margin: the Ionosphere and Pima Indians Diabetes Datasets

The ionosphere dataset [32] from the UCI Repository [4] consists of 200 training and 150 test examples of radar returns from the ionosphere. This dataset contains some noise and consequently generalisation benefits from use of a soft margin. To study the effect of the soft margin we determined the generalisation error using the KA algorithm (with bias) using Gaussian kernels on the 2D range $\sigma = \{0.5, 2.5\}$ and $C = \{1.0, 10.0\}$. Best performance was achieved with a non-zero training error of 1% with a corresponding generalisation performance of 96.0% (Figure 12). This exceeds the hard margin ($C = \infty$) performance of 92.0% and compares favourably with the performance for k-nearest neighbours (92.1%) and Quinlan's C4.5 (94.1%). However, a comparable performance of 96.0% has been achieved using a multi-layered neural network trained with the Back Propagation algorithm [32].

The Pima Indians diabetes dataset from the UCI dataset [33, 4] similarly illustrates the benefit of a soft margin for noisy datasets. For a training set

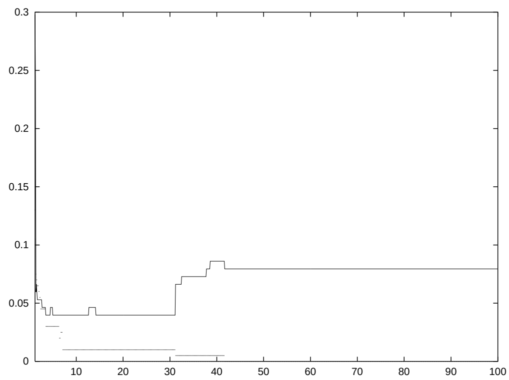

Figure 12: Generalisation error (y-axis) against soft margin parameter C (x-axis) for the ionosphere dataset. Gaussian kernels were used with $\sigma = 1.5$, $\eta = 1.0$ and $\mu = 0.2$. The lower (dotted) curve shows the training error (y-axis) against C (x-axis).

of 609 and a test set of 161 examples, a minimum in the generalisation error (0.248) was achieved with $C = 1.02$ and $\sigma = 11.0$ and a training error of 0.244. This is an even more impressive improvement over the hard margin result of 0.335 for the generalisation error.

4.4 Learning times

To estimate how learning times scale with m we note that each epoch requires of the order of m^2 operations. This is evident because there is an outer loop in m as each α_i is updated and an inner loop in m as z_i is determined. To study the scaling with m we therefore approximate the number of operations with the number of epochs necessary to reach a margin of 0.999 (and changes in the bias of less than 10^{-5}) multiplied by m^2. With each incrementation of the sample size m by 1, performance was averaged over 200 sets of patterns randomly chosen from the data and used as the training set. We have determined the best fit power law from these plots for the datasets outlined above and the scaling exponents for m are given in the table below:

Dataset	Mir. Sym.	Sonar	Cancer	Diab.	Ion.	Digits (av.)
Exponent	1.7	1.8	2.0	2.1	3.0	1.8

Learning times are affected by factors such as the complexity of the task or amount of noise in the data. For comparatively noise-free datasets such as sonar classification, digit classification or mirror symmetry (which is noiseless) learning times scale sub-quadratically with m. On the other hand, for noisy datasets (particularly the ionosphere dataset) scaling is more than quadratic. This scaling behaviour is similar to that of the Sequential Minimal Optimization (SMO) algorithm of Platt [24] which can scale sub-quadratically with m for readily separable tasks. Both KA and SMO appear to scale much better than QP routines such as projected conjugate gradient (PCG) [13] which scales with exponents beyond 2.5 for many datasets [24]. Thus the KA algorithm is expected to be most efficient for large datasets.

5 Conclusion

The implementational complexity of SVMs is one of the factors that has hampered their uptake as a standard machine learning tool, despite their clear effectiveness in dealing with overfitting. In this paper we have explicitly addressed this problem by proposing simple learning procedures which require virtually no implementational effort and which are computationally efficient. Despite their simplicity, our algorithms are guaranteed to converge to the optimal solution, and theoretical and experimental results suggest that they scale well with the number of patterns in the data. Furthermore, our algorithms have paved the way for an automatic model selection method, which can optimize the kernel parameter with little additional computational cost[8]. We note that the version of the algorithm without bias often exhibits good generalisation, at least for the case of Gaussian kernels. Though this may not be true in general, it does suggest further investigation of the role of the bias parameter would be useful.

The KA algorithm can also be readily generalised to handle regression problems. In this case [39] two sets of lagrange multipliers α_i and $\alpha_i^\star$ are introduced and the KA algorithm generalises with similar gradient ascent corrections for α_i and $\alpha_i^\star$ and the requirement $\alpha_i \to 0$ and $\alpha_i^\star \to 0$ if the positivity of these variables would be violated.

Other algorithms have been proposed for simplifying the training of SVMs. Among these, it is interesting to mention the ingenious algorithm SMO, by John Platt [24], which can be regarded as the extreme limit of chunking. For SMO it is possible to find an analytic solution to the smallest possible QP problem. The time complexity of this system is also subquadratic, and the space complexity (memory requirement) linear. From an implementational viewpoint SMO is not as simple as KA but it also avoids the use of QP packages. The availability of simple and fast training algorithms has been the key to the uptake of other learning systems in the past, and its lack has been pointed out as one of the causes for the slow uptake of SVMs. We hope that systems based on online stochastic gradient ascent techniques, such as KA, will contribute to filling this gap.

Acknowledgements: The authors wish to thank Thilo Friess for earlier collaboration on this topic and discussions. We would also like to thank the EPSRC for financial support under research grant GR/K70366.

References

[1] Aizerman, M., Braverman, E., and Rozonoer, L. (1964). Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning, *Automations and Remote Control*, **25**:821-837.

[2] Anlauf, J.K., and Biehl, M. (1989). The Adatron - An Adaptive Perceptron Algorithm. *Europhysics Letters* **10**:687-692.

[3] Bartlett P., Shawe-Taylor J., (1998). Generalization Performance of Support Vector Machines and Other Pattern Classifiers. 'Advances in Kernel Methods - Support Vector Learning', Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola (eds.), MIT Press, Cambridge, USA.

[4] Blake, C., Keogh, E. and Merz, C.J. (1998). UCI Repository of machine learning databases [http://www.ics.uci.edu/ mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

[5] Boser, B., Guyon, I., Vapnik, V. (1992). A training algorithm for optimal margin classifiers. *Fifth Annual Workshop on Computational Learning Theory.* ACM Press.

[6] Cortes, C., and Vapnik, V. (1995). Support Vector networks, *Machine Learning* 20:273-297.

[7] Cortes, C. (1995). *Prediction of Generalization Ability in Learning Machines.* PhD Thesis, Department of Computer Science, University of Rochester.

[8] Cristianini, N., Campbell, C., and Shawe-Taylor, J., Dynamically Adapting the Kernels of Support Vector Machines, to appear in Advances in Neural Information Processing Systems 12, The MIT Press, Cambridge, Mass., 1999.

[9] Cristianini, N., Campbell, C., and Shawe-Taylor, J., Automatic Model Selection for Support Vector Machines, *in preparation.*

[10] Cristianini, N., Shawe-Taylor, J., Sykacek, P., (1998). Bayesian Classifiers are Large Margin Hyperplanes in a Hilbert Space, in Shavlik, J., ed., *Machine Learning: Proceedings of the Fifteenth International Conference*, Morgan Kaufmann Publishers, San Francisco, CA.

[11] FrießT-T., Cristianini, N. & Campbell, C., (1998). The Kernel Adatron: a Fast and Simple Learning Procedure for Support Vector Machines, *Proceedings of the Fifteenth International Conference on Machine Learning*, Morgan Kaufmann, Madison, Wisconsin.

[12] Freund, Y. & Schapire, R. (1998). Large margin classification using the perceptron algorithm. In Proceedings of Computational Learning Theory (COLT98).

[13] Gill, P.E., Murray, W. and Wright, M.H., *Practical Optimization.* Academic Press, 1981.

[14] Gorman R.P. & Sejnowski, T.J. (1988) Analysis of Hidden Units in a Layered Network trained to classify sonar targets. *Neural Networks* **1**:75-89.

[15] Guyon, I., Matic, N., & Vapnik, V. (1996). Discovering Informative Patterns and Data Cleaning, *Advances in Knowledge Discovery and Data Mining* ed by U.M.Fayyad, G. Piatelsky-Shapiro, P. Smyth and R. Uthurusamy AAAI Press/ MIT Press.

[16] Kinzel, W.,(1990) Statistical Mechanics of the Perceptron with Maximal Stability. *Lecture Notes in Physics* (Springer-Verlag) **368**:175-188.

[17] LeCun, Y., Jackel, L. D., Bottou, L., Brunot, A., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Muller, U. A., Sackinger, E., Simard, P. and Vapnik, V., (1995). Comparison of learning algorithms for handwritten digit recognition, *International Conference on Artificial Neural Networks*, Fogelman, F. and Gallinari, P. (Ed.), pp. 53-60.

[18] Krauth W. and Mezard. M. (1987) Learning Algorithms with Optimal Stability in Neural Networks. *J.Phys.* **A20**:L745-L752.

[19] Minsky M.L. & Papert, S.A. (1969) *Perceptrons*, MIT Press: Cambridge.

[20] Opper, M. (1988). Learning Times of Neural Networks: Exact Solution for a Perceptron Algorithm. *Physical Review* **A38**:3824-3826

[21] Opper, M. (1989). Learning in Neural Networks: Solvable Dynamics. *Europhysics Letters*, **8**:389-392.

[22] Osuna E., Freund R., Girosi F., (1997) "Training Support Vector Machines: An Application to Face Detection", Proc. Computer Vision and Pattern Recognition '97, 130-136.

[23] Osuna E., Freund R., Girosi F., (1997) "An improved training algorithm for support vector machines", In Principe, J. et al. (ed.) *Neural Networks for Signal Processing VII - Proceedings of the 1997 IEEE Workshop*, IEEE, New York, p. 276-285.

[24] Platt, J.C., Fast training of Support Vector Machines Using Sequential Minimal Optimization, in *Advances in Kernels Methods: Support Vector Learning*, ed. Schoelkopf, B., Burges, C. and Smola, A., MIT Press, Cambridge, Mass. 1998.

[25] Papageorgiou, C., Oren, M. and Poggio, T., A General Framework for Object Detection, International Conference on Computer Vision, 1998.

[26] Schapire. R., Freund, Y., Bartlett, P., & Sun Lee, W. (1997). Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods, *Proceedings of Fourteenth International Conference on Machine Learning.*

[27] Schoelkopf, B., (1997). *Support Vector Learning.* PhD Thesis. R. Oldenbourg Verlag, Munich.

[28] Smola A., FrießT-T., & Schoelkopf, B., Semiparametric Support Vector and Linear Programming Machines, to appear in Adavances in Neural Information Processing Systems 12, The MIT Press, Cambridge, Mass. 1999.

[29] Schoelkopf, B., Sung, K., Burges, C., Girosi, F., Niyogi, P., Poggio, T., and Vapnik, V., Comparing Support Vector Machines with Gaussian Kernels to Radial Basis Function Classifiers, M.I.T. Preprint (A.I. Laboratory), A.I. Memo No. 1599.

[30] Schoelkopf, B., Burges, C. and Smola, A. (ed.) , Advances in Kernel Methods: Support Vector Learning, MIT Press, Cambridge, Mass., 1999.

[31] Shawe-Taylor, J., Bartlett, P., Williamson, R. & Anthony, M. (1996). Structural Risk Minimization over Data-Dependent Hierarchies NeuroCOLT Technical Report NC-TR-96-053 (`ftp://ftp.dcs.rhbnc.ac.uk /pub/neurocolt/tech_reports`).

[32] Sigillito, V., Wing, S., Hutton, L. & Baker, K. (1989). Classification of radar returns from the ionosphere using neural networks. John Hopkins APL Technical Digest 10, 262-266.

[33] Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C. and Johannes, R.S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In Proceedings on the Symposium on Computer Applications and Medical Care. IEEE Computer Society Press, p. 261-265.

[34] `Sonar dataset:`
`http://www.boltz.cs.cmu.edu/benchmarks/sonar.html`

[35] Ster, B., & Dobnikar, A. (1996) Neural networks in medical diagnosis: comparison with other methods. In A. Bulsari et al. (ed.) *Proceedings of the International Conference EANN'96*, p. 427-430.

[36] Joachims, T., (1998) Text Categorisation with Support Vector Machines. In European Conference on Machine Learning (ECML)

[37] Vapnik, V. (1982) *Estimation of Dependencies Based on Empirical Data*, Springer-Verlag, Berlin.

[38] Vapnik, V. (1995) *The Nature of Statistical Learning Theory*, Springer Verlag.

[39] Vapnik, V., Golowich, S. & Smola, A. 1997. Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing. In Advances in Neural Information Processing Systems, Vol. 9. MIT Press, Cambridge, Mass., p. 281-287.

[40] Watkin, T., Rau, A. & Biehl, M. (1993). The Statistical Mechanics of Learning a Rule, *Rev. Mod. Phys.* **65**:499-556.