Choosing Multiple Parameters for Support Vector Machines

Olivier Chapelle, Vladimir Vapnik
AT&T Research Labs
ochapell@ens-lyon.fr
vlad@research.att.com

Olivier Bousquet
École Polytechnique
bousquet@cmapx.polytechnique.fr

Sayan Mukherjee
MIT
sayan@ai.mit.edu

March 16, 2001

Abstract

The problem of automatically tuning multiple parameters for pattern recognition Support Vector Machines (SVMs) is considered. This is done by minimizing some estimates of the generalization error of SVMs using a gradient descent algorithm over the set of parameters. Usual methods for choosing parameters, based on exhaustive search become intractable as soon as the number of parameters exceeds two. Some experimental results assess the feasibility of our approach for a large number of parameters (more than 100) and demonstrate an improvement of generalization performance.

1 Introduction

In the problem of supervised learning, one takes a set of input-output pairs $Z = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$ and attempts to construct a classifier function f that maps input vectors $\mathbf{x} \in \mathcal{X}$ onto labels $y \in \mathcal{Y}$. We are interested here in pattern recognition or classification, that is the case where the set of labels is simply $\mathcal{Y} = \{-1, 1\}$. The goal is to find a $f \in \mathcal{F}$ which minimizes the error $(f(\mathbf{x}) \neq y)$ on future examples. Learning algorithms usually depend on parameters which control the size of the class $\mathcal{F}$ or the way the search is conducted in $\mathcal{F}$. Several techniques exist for performing the selection of these parameters. The idea is to find the parameters that minimize the

generalization error of the algorithm at hand. This error can be estimated either via testing on some data which has not been used for learning (hold-out testing or cross-validation techniques) or via a bound given by theoretical analysis.

Tuning multiple parameters Usually there are multiple parameters to tune at the same time and moreover, the estimates of the error are not explicit functions of these parameters, so that the naive strategy which is exhaustive search in the parameter space becomes intractable since it would correspond to running the algorithm on every possible value of the parameter vector (up to some discretization). We propose here a methodology for automatically tuning multiple parameters for the Support Vector Machines (SVMs) which takes advantage of the specific properties of this algorithm.

The SVM algorithm Support vector machines (SVMs) realize the following idea: map a n-dimensional input vector $\mathbf{x} \in \mathbb{R}^n$ [1] into a high dimensional (possibly infinite dimensional) feature space $\mathcal{H}$ by Φ and construct an optimal separating hyperplane in this space. Different mappings construct different SVMs.

When the training data is separable, the optimal hyperplane is the one with the maximal distance (in $\mathcal{H}$ space) between the hyperplane and the closest image $\Phi(\mathbf{x}_i)$ of the vector $\mathbf{x}_i$ from the training data. For non-separable training data a generalization of this concept is used.

Suppose that the maximal distance is equal to γ and that the images $\Phi(\mathbf{x}_1), ..., \Phi(\mathbf{x}_\ell)$ of the training vectors $\mathbf{x}_1, ..., \mathbf{x}_\ell$ are within a sphere of radius R. Then the following theorem holds true [20].

Theorem 1 *Given a training set* $Z = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$ *of size* ℓ*, a feature space* $\mathcal{H}$ *and a hyperplane* $(\mathbf{w}, b)$*, the margin* $\gamma(\mathbf{w}, b, Z)$ *and the radius* $R(Z)$ *are defined by*

$$\gamma(\mathbf{w}, b, Z) = \min_{(\mathbf{x}_i, y_i) \in Z} \frac{y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i) + b)}{\|\mathbf{w}\|}$$

$$R(Z) = \min_{\mathbf{a}, \mathbf{x}_i} \|\Phi(\mathbf{x}_i) + \mathbf{a}\|$$

The maximum margin algorithm $L_\ell : (\mathcal{X} \times \mathcal{Y})^\ell \to \mathcal{H} \times \mathbb{R}$ *takes as input a training set of size* ℓ *and returns a hyperplane in feature space such that the*

[1] In the rest of this article, we will reference vectors and matrices using bold notation

margin $\gamma(\mathbf{w}, b, Z)$ is maximized. Note that assuming the training set separable means that $\gamma > 0$. Under this assumption, for all probability measures P underlying the data Z, the expectation of the misclassification probability

$$p_{err}(\mathbf{w}, b) = P(sign(\mathbf{w} \cdot \Phi(\mathbf{X}) + b) \neq Y)$$

has the bound

$$E\{p_{err}(L_{\ell-1}(Z))\} \leq \frac{1}{\ell} E\left\{\frac{R^2(Z)}{\gamma^2(L(Z), Z)}\right\}.$$

The expectation is taken over the random draw of a training set Z of size $\ell - 1$ for the left hand side and size ℓ for the right hand side.

This theorem justifies the idea of constructing a hyperplane that separates the data with a large margin: the larger the margin the better the performance of the constructed hyperplane. Note however that according to the theorem the average performance depends on the ratio $E\{R^2/\gamma^2\}$ and not simply on the large margin γ.

Why multiple parameters ? The SVM algorithm usually depends on several parameters. One of them, denoted C, controls the tradeoff between margin maximization and error minimization. Other parameters appear in the non-linear mapping into feature space. They are called *kernel parameters*. For simplicity, we will use a classical trick that allows us to consider C as a kernel parameter, so that all parameters can be treated in a unified framework.

It is widely acknowledged that a key factor in an SVM's performance is the choice of the kernel. However, in practice, very few different types of kernels have been used due to the difficulty of appropriately tuning the parameters. We present here a technique that allows to deal with a large number of parameters and thus allows to use more complex kernels.

Another potential advantage of being able to tune a large number of parameters is the possibility of rescaling the attributes. Indeed, when no a priori knowledge is available about the meaning of each of the attributes, the only choice is to use spherical kernels (i.e. give the same weight to each attribute). But one may expect that there is a better choice for the shape of the kernel since many real-world database contain attributes of very different natures. There may thus exist more appropriate *scaling factors* that give the right weight to the right feature. For example, we will see how to use

radial basis function kernels (RBF) with as many different scaling factors as input dimensions:

$$K(\mathbf{x}, \mathbf{z}) = \exp\left(-\sum_i \frac{(x_i - z_i)^2}{2\sigma_i^2}\right).$$

The usual approach is to consider $\sigma = \sigma_1 = \cdots = \sigma_n$ and to try to pick the best value for σ. However, using the proposed method, we can choose automatically good values for the scaling factors σ_i. Indeed, these factors are precisely parameters of the kernel.

Moreover, we will demonstrate that the problem of *feature selection* can be addressed with the same framework since it corresponds to finding those attributes which can be rescaled with a zero factor without harming the generalization.

We thus see that tuning kernel parameters is something extremely useful and a procedure that allows to do this would be a versatile tool for various tasks such as finding the right shape of the kernel, feature selection, finding the right tradeoff between error and margin, etc. All this gives a rationale for developing such techniques.

Our approach In summary, our goal is not only to find the hyperplane which maximizes the margin but also the values of the mapping parameters that yield best generalization error. To do so, we propose a minimax approach: maximize the margin over the hyperplane coefficients and minimize an estimate of the generalization error over the set of kernel parameters. This last step is performed using a standard gradient descent approach.

What kind of error estimates We will consider several ways of assessing the generalization error.

- Validation error: this procedure requires a reduction of the amount of data used for learning in order to save some of it for validation. Moreover, the estimates have to be smoothed for proper gradient descent.
- Leave-one-out error estimates: this procedure gives an estimate of the expected generalization as an analytic function of the parameters.

We will examine how the accuracy of the estimates influences the whole procedure of finding optimal parameters. In particular we will show that what really matters is how variations of the estimates relate to variations of the test error rather than how their values are related.

Outline The paper is organized as follows. The next section introduces the basics of SVMs. The different possible estimates of their generalization error are described in section 3 and section 4 explains how to smooth theses estimates. Then we introduce in section 5 a framework for minimizing those estimates by gradient descent. Section 6 deals with the computation of gradients of error estimates with respect to kernel parameters. Finally, in section 7 and 8, we present experimental results of the method applied to a variety of databases in different contexts. Section 7 deals with finding the right penalization along with the right radius for a kernel and with finding the right shape of a kernel. In section 8 we present results of applying our method to feature selection.

2 Support Vector Learning

We introduce some standard notations for SVMs; for a complete description, see [19]. Let $\{(\mathbf{x}_i, y_i)\}_{1 \leq i \leq \ell}$ be a set of training examples, $\mathbf{x}_i \in \mathbb{R}^n$ which belong to a class labeled by $y_i \in \{-1, 1\}$. In the SVM methodology, we map these vectors into a feature space using a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ that defines an inner product in this feature space. Here, we consider a kernel $K_{\boldsymbol{\theta}}$ depending on a set of parameters $\boldsymbol{\theta}$. The decision function given by an SVM is:

$$f(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{\ell} \alpha_i^0 y_i K_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{x}) + b\right), \tag{1}$$

where the coefficients α_i^0 are obtained by maximizing the following functional:

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

under the constraints

$$\sum_{i=1}^{\ell} \alpha_i y_i = 0 \quad \text{and} \quad \alpha_i \geq 0, \quad i = 1, ..., \ell.$$

The coefficients α_i^0 define a maximal margin hyperplane in a high-dimensional feature space where the data are mapped through a non-linear function Φ such that $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$.

This formulation of the SVM optimization problem is called the *hard margin* formulation since no training errors are allowed. Every training point

satisfies the inequality $y_i f(\mathbf{x}_i) \geq 1$ and for points $\mathbf{x}_i$ with corresponding $\alpha_i > 0$ an equality is satisfied. These points are called support vectors.

Notice that one may require the separating hyperplane to pass through the origin by choosing a fixed $b = 0$. This variant is called the hard margin SVM *without threshold*. In that case, the optimization problem remains the same as above except that the constraint $\sum \alpha_i y_i = 0$ disappears.

Dealing with non-separability For the non-separable case, one needs to allow training errors which results in the so called *soft margin* SVM algorithm [4]. It can be shown that soft margin SVMs with quadratic penalization of errors can be considered as a special case of the hard margin version with the modified kernel [4, 6]

$$\mathbf{K} \leftarrow \mathbf{K} + \frac{1}{C}\mathbf{I}, \tag{3}$$

where $\mathbf{I}$ is the identity matrix and C a constant penalizing the training errors. In the rest of the paper, we will focus on the hard margin SVM and use (3) whenever we have to deal with non-separable data. Thus C will be considered just as another parameter of the kernel function.

3 Estimating the performance of an SVM

Ideally we would like to choose the value of the kernel parameters that minimize the true risk of the SVM classifier. Unfortunately, since this quantity is not accessible, one has to build estimates or bounds for it. In this section, we present several measures of the expected error rate of an SVM.

3.1 Single validation estimate

If one has enough data available, it is possible to estimate the true error on a validation set. This estimate is unbiased and its variance gets smaller as the size of the validation set increases. If the validation set is $\{(\mathbf{x}'_i, y'_i)\}_{1 \leq i \leq p}$, the estimate is

$$T = \frac{1}{p} \sum_{i=1}^{p} \Psi(-y'_i f(\mathbf{x}'_i)), \tag{4}$$

where Ψ is the step function: $\Psi(x) = 1$ when $x > 0$ and $\Psi(x) = 0$ otherwise.

3.2 Leave-one-out bounds

The *leave-one-out* procedure consists of removing from the training data one element, constructing the decision rule on the basis of the remaining training data and then testing on the removed element. In this fashion one tests all ℓ elements of the training data (using ℓ different decision rules). Let us denote the number of errors in the leave-one-out procedure by $\mathcal{L}(\mathbf{x}_1, y_1, ..., \mathbf{x}_\ell, y_\ell)$. It is known [12] that the the leave-one-out procedure gives an almost unbiased estimate of the expected generalization error:

Lemma 1

$$Ep_{err}^{\ell-1} = \frac{1}{\ell} E(\mathcal{L}(\mathbf{x}_1, y_1, ..., \mathbf{x}_\ell, y_\ell)),$$

where $p_{err}^{\ell-1}$ is the probability of test error for the machine trained on a sample of size $\ell - 1$ and the expectations are taken over the random choice of the sample.

Although this lemma makes the leave-one-out estimator a good choice when estimating the generalization error, it is nevertheless very costly to actually compute since it requires running the training algorithm ℓ times. The strategy is thus to upper bound or approximate this estimator by an easy to compute quantity T having, if possible, an analytical expression.

If we denote by f^0 the classifier obtained when all training examples are present and f^i the one obtained when example i has been removed, we can write:

$$\mathcal{L}(\mathbf{x}_1, y_1, ..., \mathbf{x}_\ell, y_\ell) = \sum_{p=1}^{\ell} \Psi(-y_p f^p(\mathbf{x}_p)), \tag{5}$$

which can also be written as

$$\mathcal{L}(\mathbf{x}_1, y_1, ..., \mathbf{x}_\ell, y_\ell) = \sum_{p=1}^{\ell} \Psi(-y_p f^0(\mathbf{x}_p) + y_p(f^0(\mathbf{x}_p) - f^p(\mathbf{x}_p))).$$

Thus, if U_p is an upper bound for $y_p(f^0(\mathbf{x}_p) - f^p(\mathbf{x}_p))$, we will get the following upper bound on the leave-one-out error:

$$\mathcal{L}(\mathbf{x}_1, y_1, ..., \mathbf{x}_\ell, y_\ell) \leq \sum_{p=1}^{\ell} \Psi(U_p - 1),$$

since for hard margin SVMs, $y_p f^0(\mathbf{x}_p) \geq 1$ and Ψ is monotonically increasing.

3.2.1 Support vector count

Since removing a non-support vector from the training set does not change the solution computed by the machine (i.e. $U_p = f^0(\mathbf{x}_p) - f^p(\mathbf{x}_p) = 0$ for $\mathbf{x}_p$ non-support vector), we can restrict the preceding sum to support vectors and upper bound each term in the sum by 1 which gives the following bound on the number of errors made by the leave-one-out procedure [18]:

$$T = \frac{N_{SV}}{\ell},$$

where N_{SV} denotes the number of support vectors.

3.2.2 Jaakkola-Haussler bound

For SVMs without threshold, analyzing the optimization performed by the SVM algorithm when computing the leave-one-out error, Jaakkola and Haussler [9] proved the inequality:

$$y_p(f^0(\mathbf{x}_p) - f^p(\mathbf{x}_p)) \leq \alpha_p^0 K(\mathbf{x}_p, \mathbf{x}_p) = U_p$$

which leads to the following upper bound:

$$T = \frac{1}{\ell} \sum_{p=1}^{\ell} \Psi(\alpha_p^0 K(\mathbf{x}_p, \mathbf{x}_p) - 1).$$

Note that Wahba et al. [21] proposed an estimate of the number of errors made by the leave-one-out procedure, which in the hard margin SVM case turns out to be

$$T = \sum \alpha_p^0 K(\mathbf{x}_p, \mathbf{x}_p),$$

which can be seen as an upper bound of the Jaakkola-Haussler one since $\Psi(x-1) \leq x$ for $x \geq 0$.

3.2.3 Opper-Winther bound

For hard margin SVMs without threshold, Opper and Winther [14] used a method inspired from linear response theory to prove the following: under the assumption that the set of support vectors does not change when removing the example p, we have

$$y_p(f^0(\mathbf{x}_p) - f^p(\mathbf{x}_p)) = \frac{\alpha_p^0}{(\mathbf{K}_{SV}^{-1})_{pp}},$$

where $\mathbf{K}_{SV}$ is the matrix of dot products between support vectors; leading to the following estimate:

$$T = \frac{1}{\ell} \sum_{p=1}^{\ell} \Psi \left(\frac{\alpha_p^0}{(K_{SV}^{-1})_{pp}} - 1 \right).$$

3.2.4 Radius-margin bound

For SVMs without threshold and with no training errors, Vapnik [19] proposed the following upper bound on the number of errors of the leave-one-out procedure:

$$T = \frac{1}{\ell} \frac{R^2}{\gamma^2}.$$

where R and γ are the radius and the margin as defined in theorem 1.

3.2.5 Span bound

Vapnik and Chapelle [20, 3] derived an estimate using the concept of *span* of support vectors.

Under the assumption that the set of support vectors remains the same during the leave-one-out procedure, the following equality is true:

$$y_p(f^0(\mathbf{x}_p) - f^p(\mathbf{x}_p)) = \alpha_p^0 S_p^2,$$

where S_p is the distance between the point $\Phi(\mathbf{x}_p)$ and the set Λ_p where

$$\Lambda_p = \left\{ \sum_{i \neq p,\ \alpha_i^0 > 0} \lambda_i \Phi(\mathbf{x}_i), \sum_{i \neq p} \lambda_i = 1 \right\}. \tag{6}$$

This gives the exact number of errors made by the leave-one-out procedure under the previous assumption:

$$T = \frac{1}{\ell} \sum_{p=1}^{\ell} \Psi(\alpha_p^0 S_p^2 - 1). \tag{7}$$

The span estimate can be related to other approximations:

Link with Jaakkola-Haussler bound

If we consider SVMs without threshold, the constraint $\sum \lambda_i = 1$ can be removed in the definition of the span. Then we can easily upper bound the value of the span: $S_p^2 \leq K(\mathbf{x}_p, \mathbf{x}_p)$, and thus recover the Jaakkola-Haussler bound.

Link with R^2/γ^2

For each support vector, we have $y_p f^0(\mathbf{x}_p) = 1$. Since for $x \geq 0$, $\Psi(x-1) \leq x$, the number of errors made by the leave-one-out procedure is bounded by:

$$\sum_p \alpha_p^0 S_p^2.$$

It has been shown [20] that the span S_p is bounded by the diameter of the smallest sphere enclosing the training points and since $\sum \alpha_p^0 = 1/\gamma^2$, we finally get

$$T \leq 4\frac{R^2}{\gamma^2}.$$

A similar derivation as the one used in the *span bound* has been proposed in [10], where the leave-one-out error is bounded by $|\{p, 2\alpha_p^0 R^2 > y_p f^0(\mathbf{x}_p)\}|$, with $0 \leq K(\mathbf{x}_i, \mathbf{x}_i) \leq R^2, \ \forall i$.

Link with Opper-Winther

When the support vectors do not change, the hard margin case without threshold gives the same value as the Opper-Winther bound, namely:

$$S_p^2 = \frac{1}{(\mathbf{K}_{SV}^{-1})_{pp}}.$$

4 Smoothing the test error estimates

The estimate of the performance of an SVM through a validation error (4) or the leave-one-out error (5) requires the use of the step function Ψ. However, we would like to use a gradient descent approach to minimize those estimates of the test error. Unfortunately the step function is not differentiable. As already mentioned in section 3.2.5, it is possible to bound $\Psi(x-1)$ by x for $x \geq 0$. This is how the bound R^2/γ^2 is derived from the leave-one-out error. Nevertheless by doing so, large errors count more than one, therefore it might be advantageous instead to use a contracting function of the form $\Psi(x) = (1 + \exp(-Ax + B))^{-1}$ (see figure 1).

However, the choice of the constants A and B is difficult. If A is too small, the estimate is not accurate and A is too large, the resulting estimate is not smooth.

Instead of trying to pick good constants A and B, one can try to get directly a smooth approximation of the test error by estimating posterior

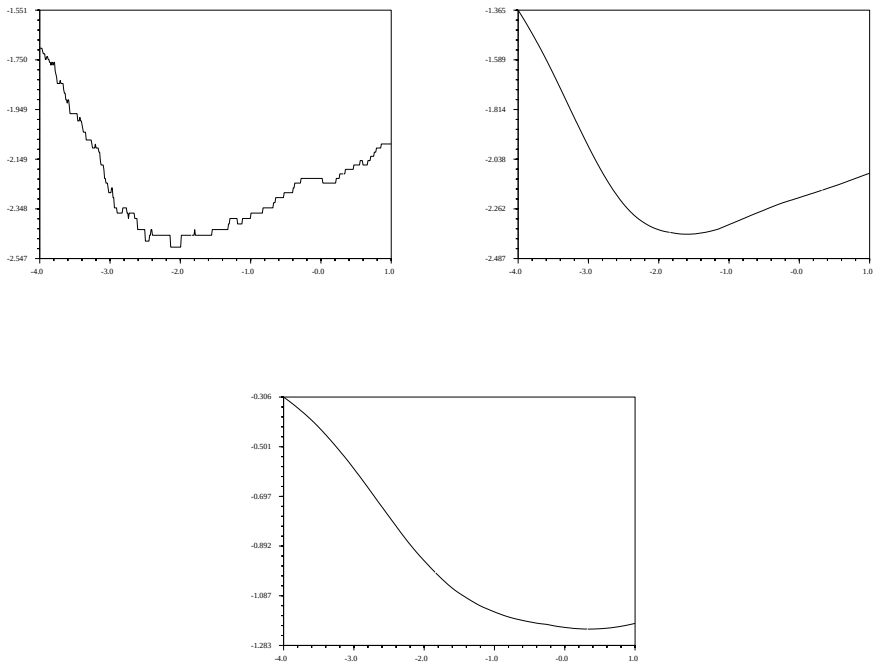

Figure 1: Validation error for different values of the width of an RBF kernel. Top left: with a step function, $\Psi(x) = 1_{x>0}$. Top right: sigmoid function, $\Psi(x) = (1+\exp(-5x))^{-1}$. Bottom: linear function, $\Psi(x) = 1+x$ for $x > -1$, 0 otherwise. Note that on the bottom picture, the minimum is not at the right place

probabilities. Recently, Platt proposed the following estimate of the posterior distribution $P(Y = 1|X = \mathbf{x})$ of an SVM output $f(\mathbf{x})$ [15]:

$$\tilde{P}_{A,B}(\mathbf{x}) = \tilde{P}(Y = 1|X = \mathbf{x}) = \frac{1}{1 + \exp(Af(\mathbf{x}) + B)},$$

where $f(\mathbf{x})$ is the output of the SVM. The constants A and B are found by minimizing the Kullback-Leibler divergence between $\tilde{P}$ and an empirical approximation of P built from a validation set $(\mathbf{x}'_i, y'_i)_{1 \leq i \leq n_v}$:

$$(A^*, B^*) = \arg\max_{A,B} \sum_{i=1}^{n_v} \left(\frac{1 + y'_i}{2} \log(\tilde{P}_{A,B}(\mathbf{x}'_i)) + \frac{1 - y'_i}{2} \log(1 - \tilde{P}_{A,B}(\mathbf{x}'_i)) \right).$$

This optimization is carried out using a second order gradient descent algorithm [15].

According to this estimate the best threshold for our SVM classifier f is such that $f(\mathbf{x}) = \text{sign}(\tilde{P}_{A^*,B^*}(x) - 0.5)$. Note that if $B^* \neq 0$, we obtained a correction compared to the usual SVM threshold.

By definition the generalization error of our classifier is

$$P(Y \neq f(X)) = \int_{\mathbf{x}, f(\mathbf{x})=-1} P(Y = 1|\mathbf{x}) d\mu(\mathbf{x}) + \int_{\mathbf{x}, f(\mathbf{x})=1} (Y = -1|\mathbf{x}) d\mu(\mathbf{x}).$$

This error can be empirically estimated as [2]:

$$\begin{aligned} P(Y \neq f(X)) &\approx \sum_{i, \tilde{P}(\mathbf{x}'_i)<0.5} \tilde{P}(\mathbf{x}'_i) + \sum_{i, \tilde{P}(\mathbf{x}'_i)>0.5} 1 - \tilde{P}(\mathbf{x}'_i) \qquad (8) \\ &= \sum_{i=1}^{n_v} \min\left(\tilde{P}(\mathbf{x}'_i), 1 - \tilde{P}(\mathbf{x}'_i)\right). \qquad (9) \end{aligned}$$

Note that the labels of the validation set are not used directly in this last step but indirectly through the estimation of the constants A and B appearing in the parametric form of $\tilde{P}_{A^*,B^*}$. To have a better understanding of this estimate, let us consider the extreme case where there is no error on the validation set. Then the maximum likelihood algorithm is going to yield $A = -\infty$ and $\tilde{P}_{A^*,B^*}(\mathbf{x})$ will only take binary values. As a consequence, the estimate of the error probability will be zero.

5 Optimizing the kernel parameters

Let's go back to the SVM algorithm. We assume that the kernel k depends on one or several parameters, encoded into a vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)$. We thus consider a class of decision functions parametrized by $\boldsymbol{\alpha}$, b and $\boldsymbol{\theta}$:

$$f_{\boldsymbol{\alpha},b,\boldsymbol{\theta}}(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{\ell} \alpha_i y_i K_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{x}_i) + b\right).$$

We want to choose the values of the parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$ such that W (see equation (2)) is maximized (maximum margin algorithm) and T, the model selection criterion, is minimized (best kernel parameters). More precisely, for $\boldsymbol{\theta}$ fixed, we want to have $\boldsymbol{\alpha}^0 = \arg\max W(\boldsymbol{\alpha})$ and choose $\boldsymbol{\theta}^0$ such that

$$\boldsymbol{\theta}^0 = \arg\min_{\boldsymbol{\theta}} T(\boldsymbol{\alpha}^0, \boldsymbol{\theta}).$$

[2] We note $\tilde{P}(\mathbf{x})$ as an abbreviation for $\tilde{P}_{A^*,B^*}(\mathbf{x})$

When $\boldsymbol{\theta}$ is a one dimensional parameter, one typically tries a finite number of values and picks the one which gives the lowest value of the criterion T. When both T and the SVM solution are continuous with respect to $\boldsymbol{\theta}$, a better approach has been proposed by Cristianini et al. [5]: using an incremental optimization algorithm, one can train an SVM with little effort when $\boldsymbol{\theta}$ is changed by a small amount. However, as soon as $\boldsymbol{\theta}$ has more than one component computing $T(\boldsymbol{\alpha}, \boldsymbol{\theta})$ for every possible value of $\boldsymbol{\theta}$ becomes intractable, and one rather looks for a way to optimize T along a trajectory in the kernel parameter space.

Using the gradient of a model selection criterion to optimize the model parameters has been proposed in [1] and demonstrated in the case of linear regression and time-series prediction. It has also been proposed by [11] to optimize the regularization parameters of a neural network.

Here we propose an algorithm that alternates the SVM optimization with a gradient step is the direction of the gradient of T in the parameter space. This can be achieved by the following iterative procedure:

1. Initialize $\boldsymbol{\theta}$ to some value.
2. Using a standard SVM algorithm, find the maximum of the quadratic form W:
$$\boldsymbol{\alpha}^0(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\alpha}} W(\boldsymbol{\alpha}, \boldsymbol{\theta}).$$
3. Update the parameters $\boldsymbol{\theta}$ such that T is minimized. This is typically achieved by a gradient step (see below).
4. Go to step 2 or stop when the minimum of T is reached.

Solving step 3 requires estimating how T varies with $\boldsymbol{\theta}$. We will thus restrict ourselves to the case where $K_{\boldsymbol{\theta}}$ can be differentiated with respect to $\boldsymbol{\theta}$. Moreover, we will only consider cases where the gradient of T with respect to $\boldsymbol{\theta}$ can be computed (or approximated).

Note that $\boldsymbol{\alpha}^0$ depends implicitly on $\boldsymbol{\theta}$ since $\boldsymbol{\alpha}^0$ is defined as the maximum of W. Then, if we have n kernel parameters $(\theta_1, \ldots, \theta_n)$, the total derivative of $T^0(\cdot) \equiv T(\boldsymbol{\alpha}^0(\cdot), \cdot)$ with respect to θ_p is:

$$\frac{\partial T^0}{\partial \theta_p} = \left.\frac{\partial T^0}{\partial \theta_p}\right|_{\boldsymbol{\alpha}^0 \text{ fixed}} + \frac{\partial T^0}{\partial \boldsymbol{\alpha}^0}\frac{\partial \boldsymbol{\alpha}^0}{\partial \theta_p}.$$

Having computed the gradient $\nabla_{\boldsymbol{\theta}} T(\boldsymbol{\alpha}^0, \boldsymbol{\theta})$, a way of performing step 3 is to make a *gradient step*:

$$\delta\theta_k = -\varepsilon \frac{\partial T(\boldsymbol{\alpha}^0, \boldsymbol{\theta})}{\partial \theta_k},$$

for some small and eventually decreasing ε. The convergence can be improved with the use of second order derivatives (Newton's method):

$$\delta\theta_k = -(\Delta_{\boldsymbol{\theta}} T)^{-1} \frac{\partial T(\boldsymbol{\alpha}^0, \boldsymbol{\theta})}{\partial \theta_k}$$

where the Laplacian operator Δ is defined by

$$(\Delta_{\boldsymbol{\theta}} T) i, j = \frac{\partial^2 T(\boldsymbol{\alpha}^0, \boldsymbol{\theta})}{\partial \theta_i \partial \theta_j}.$$

In this formulation, additional constraints can be imposed through projection of the gradient.

6 Computing the gradient

In this section, we describe the computation of the gradient (with respect to the kernel parameters) of the different estimates of the generalization error. First, for the bound R^2/γ^2 (see Theorem 1), we obtain a formulation of the derivative of the margin (section 6.1) and of the radius (section 6.2). For the validation error (see equation (4)), we show how to calculate the derivative of the hyperplane parameters $\boldsymbol{\alpha}^0$ and b (see section 6.3). Finally, the computation of the derivative of the span bound (7) is presented in section 6.4.

We first begin with a useful lemma.

Lemma 2 *Suppose we are given a $(n \times 1)$ vector $\mathbf{v}_\theta$ and an $(n \times n)$ matrix $\mathbf{P}_\theta$ smoothly depending on a parameter θ. Consider the function:*

$$L(\theta) = \max_{\mathbf{x} \in F} \ \mathbf{x}^T \mathbf{v}_\theta - \frac{1}{2} \mathbf{x}^T \mathbf{P}_\theta \mathbf{x}$$

where

$$F = \{\mathbf{x} : \mathbf{b}^T \mathbf{x} = c, \mathbf{x} \geq 0\}.$$

Let $\bar{\mathbf{x}}$ be the the vector $\mathbf{x}$ where the maximum in $L(\theta)$ is attained. If this minimum is unique then

$$\frac{\partial L(\theta)}{\partial \theta} = \bar{\mathbf{x}}^T \frac{\partial \mathbf{v}_\theta}{\partial \theta} - \frac{1}{2} \bar{\mathbf{x}}^T \frac{\partial \mathbf{P}_\theta}{\partial \theta} \bar{\mathbf{x}}.$$

In other words, it is possible to differentiate L with respect to θ as if $\bar{\mathbf{x}}$ did not depend on θ. Note that this is also true if one (or both) of the constraints in the definition of F are removed.

Proof: We first need to express the equality constraint with a Lagrange multiplier λ and the inequality constraints with Lagrange multipliers γ_i:

$$L(\theta) = \max_{\mathbf{x},\lambda,\boldsymbol{\gamma}} \quad \mathbf{x}^T \mathbf{v}_\theta - \frac{1}{2} \mathbf{x}^T \mathbf{P}_\theta \mathbf{x} - \lambda(\mathbf{b}^T \mathbf{x} - c) + \boldsymbol{\gamma}^T \mathbf{x}. \tag{10}$$

At the maximum, the following conditions are verified:

$$\begin{aligned} &\mathbf{v}_\theta - \mathbf{P}_\theta \bar{\mathbf{x}} = \bar{\lambda} \mathbf{b} - \bar{\boldsymbol{\gamma}}, \\ &\mathbf{b}^T \bar{\mathbf{x}} = c, \\ &\bar{\gamma}_i \ \bar{x}_i = 0, \quad \forall i. \end{aligned}$$

We will not consider here differentiability problems. The interested reader can find details in [2]. The main result is that whenever $\bar{\mathbf{x}}$ is unique, L is differentiable.

We have

$$\frac{\partial L(\theta)}{\partial \theta} = \bar{\mathbf{x}}^T \frac{\partial \mathbf{v}_\theta}{\partial \theta} - \frac{1}{2} \bar{\mathbf{x}}^T \frac{\partial \mathbf{P}_\theta}{\partial \theta} \bar{\mathbf{x}} + \frac{\partial \bar{\mathbf{x}}}{\partial \theta}^T (\mathbf{v}_\theta - \mathbf{P}_\theta \bar{\mathbf{x}}),$$

where the last term can be written as follows,

$$\frac{\partial \bar{\mathbf{x}}}{\partial \theta}^T (\mathbf{v}_\theta - \mathbf{P}_\theta \bar{\mathbf{x}}) = \bar{\lambda} \frac{\partial \bar{\mathbf{x}}}{\partial \theta}^T \mathbf{b} - \frac{\partial \bar{\mathbf{x}}}{\partial \theta}^T \bar{\boldsymbol{\gamma}}.$$

Using the derivatives of the optimality conditions, namely

$$\frac{\partial \bar{\mathbf{x}}}{\partial \theta}^T \mathbf{b} = 0,$$

$$\frac{\partial \bar{\gamma}_i}{\partial \theta} \ \bar{x}_i + \bar{\gamma}_i \frac{\partial \bar{x}_i}{\partial \theta} = 0,$$

and the fact that either $\bar{\gamma}_i = 0$ or $\bar{x}_i = 0$ we get:

$$\frac{\partial \bar{\gamma}_i}{\partial \theta} \ \bar{x}_i = \bar{\gamma}_i \frac{\partial \bar{x}_i}{\partial \theta} = 0,$$

hence

$$\frac{\partial \bar{\mathbf{x}}}{\partial \theta}^T (\mathbf{v}_\theta - \mathbf{P}_\theta \bar{\mathbf{x}}) = 0$$

and the result follows. □

6.1 Computing the derivative of the margin

Note that in feature space, the separating hyperplane $\{\mathbf{x} : \mathbf{w} \cdot \Phi(\mathbf{x}) + b = 0\}$ has the following expansion

$$\mathbf{w} = \sum_{i=1}^{\ell} \alpha_i^0 y_i \Phi(\mathbf{x}_i)$$

and is normalized such that

$$\min_{1 \leq i \leq \ell} y_i (\mathbf{w} \cdot \Phi(\mathbf{x}_i) + b) = 1.$$

It follows from the definition of the margin in Theorem 1 that this latter is $\gamma = 1/\|\mathbf{w}\|$. Thus we write the bound R^2/γ^2 as $R^2 \|\mathbf{w}\|^2$.

The previous lemma enables us to compute the derivative of $\|\mathbf{w}\|^2$. Indeed, it can be shown [19] that

$$\frac{1}{2}\|\mathbf{w}\|^2 = W(\boldsymbol{\alpha}^0),$$

and the lemma can be applied to the standard SVM optimization problem (2), giving

$$\frac{\partial \|\mathbf{w}\|^2}{\partial \theta_p} = -\sum_{i,j=1}^{\ell} \alpha_i^0 \alpha_j^0 y_i y_j \frac{\partial K(\mathbf{x}_i, \mathbf{x}_j)}{\partial \theta_p}$$

6.2 Computing the derivative of the radius

Computing the radius of the smallest sphere enclosing the training points can be achieved by solving the following quadratic problem [19]:

$$R^2 = \max_{\boldsymbol{\beta}} \sum_{i=1}^{\ell} \beta_i K(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j=1}^{\ell} \beta_i \beta_j K(\mathbf{x}_i, \mathbf{x}_j)$$

under constraints

$$\sum_{i=1}^{\ell} \beta_i = 1$$
$$\forall i \quad \beta_i \geq 0$$

We can again use the previous lemma to compute the derivative of the radius:

$$\frac{\partial R^2}{\partial \theta_p} = \sum_{i=1}^{\ell} \beta_i^0 \frac{\partial K(\mathbf{x}_i, \mathbf{x}_i)}{\partial \theta_p} - \sum_{i,j=1}^{\ell} \beta_i \beta_j \frac{\partial K(\mathbf{x}_i, \mathbf{x}_j)}{\partial \theta_p},$$

where $\boldsymbol{\beta}^0$ maximizes the previous quadratic form.

6.3 Computing the derivative of the hyperplane parameters

Let us first compute the derivative of $\boldsymbol{\alpha}^0$ with respect to a parameter $\boldsymbol{\theta}$ of the kernel. For this purpose, we need an analytical formulation for $\boldsymbol{\alpha}^0$. First, we suppose that the points which are not support vectors are removed from the training set. This assumption can be done without any loss of generality since removing a point which is not support vector does not affect the solution. Then, the fact that all the points lie on the margin can be written

$$\underbrace{\begin{pmatrix} \mathbf{K}^Y & \mathbf{Y} \\ \mathbf{Y}^T & 0 \end{pmatrix}}_{\mathbf{H}} \begin{pmatrix} \boldsymbol{\alpha}^0 \\ b \end{pmatrix} = \begin{pmatrix} \mathbf{1} \\ 0 \end{pmatrix},$$

where $\mathbf{K}^Y_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$. If there are n support vectors, $\mathbf{H}$ is a $(n+1) \times (n+1)$ matrix. The parameters of the SVMs can be written as:

$$(\boldsymbol{\alpha}^0, b)^T = \mathbf{H}^{-1}(1 \cdots 1\ 0)^T.$$

We are now able to compute the derivatives of those parameters with respect to a kernel parameter θ_p. Indeed, since the derivative of the inverse of a matrix M depending on a parameter θ_p can be written [3]

$$\frac{\partial \mathbf{M}^{-1}}{\partial \theta_p} = -\mathbf{M}^{-1} \frac{\partial \mathbf{M}}{\partial \theta_p} \mathbf{M}^{-1}, \tag{11}$$

it follows that

$$\frac{\partial (\boldsymbol{\alpha}^0, b)}{\partial \theta_p} = -\mathbf{H}^{-1} \frac{\partial \mathbf{H}}{\partial \theta_p} \mathbf{H}^{-1} (1 \cdots 1\ 0)^T,$$

and finally

$$\boxed{\frac{\partial (\boldsymbol{\alpha}^0, b)}{\partial \theta_p} = -\mathbf{H}^{-1} \frac{\partial \mathbf{H}}{\partial \theta_p} (\boldsymbol{\alpha}^0, b)^T.}$$

We can easily use the result of this calculation to recover the computation $\frac{\partial \|\mathbf{w}\|^2}{\partial \theta_p}$. Indeed, if we denote $\tilde{\boldsymbol{\alpha}} = (\boldsymbol{\alpha}^0, b)$, we have $\|\mathbf{w}\|^2 = (\boldsymbol{\alpha}^0)^T \mathbf{K}^Y \boldsymbol{\alpha}^0 = \tilde{\boldsymbol{\alpha}}^T \mathbf{H} \tilde{\boldsymbol{\alpha}}$ and it turns out that:

$$\begin{aligned} \frac{\partial \|\mathbf{w}\|^2}{\partial \theta_p} &= \tilde{\boldsymbol{\alpha}}^T \frac{\partial \mathbf{H}}{\partial \theta_p} \tilde{\boldsymbol{\alpha}} + 2 \tilde{\boldsymbol{\alpha}} \mathbf{H} \frac{\partial \tilde{\boldsymbol{\alpha}}}{\partial \theta_p} \\ &= \tilde{\boldsymbol{\alpha}}^T \frac{\partial \mathbf{H}}{\partial \theta_p} \tilde{\boldsymbol{\alpha}} - 2 \tilde{\boldsymbol{\alpha}} \mathbf{H} \mathbf{H}^{-1} \frac{\partial \mathbf{H}}{\partial \theta_p} \tilde{\boldsymbol{\alpha}} \end{aligned}$$

[3] This inequality can be easily proved by differentiating $\mathbf{M}\mathbf{M}^{-1} = \mathbf{I}$

$$
\begin{aligned}
&= -\tilde{\boldsymbol{\alpha}}^T \frac{\partial \mathbf{H}}{\partial \theta_p} \tilde{\boldsymbol{\alpha}} \\
&= -(\boldsymbol{\alpha}^0)^T \frac{\partial \mathbf{K}^Y}{\partial \theta_p} \boldsymbol{\alpha}^0.
\end{aligned}
$$

6.4 Computing the derivative of the span-rule

Now, let us consider the span value. Recall that the span of the support vector $\mathbf{x}_p$ is defined as the the distance between the point $\Phi(\mathbf{x}_p)$ and the set Λ_p defined by (6). Then the value of the span can be written as:

$$
S_p^2 = \min_{\boldsymbol{\lambda}} \max_{\mu} \left(\Phi(\mathbf{x}_p) - \sum_{i \neq p} \lambda_i \Phi(\mathbf{x}_i) \right)^2 + 2\mu \left(\sum_{i \neq p} \lambda_i - 1 \right).
$$

Note that we introduced a Lagrange multiplier μ to enforce the constraint $\sum \lambda_i = 1$.

Introducing the extended vector $\tilde{\boldsymbol{\lambda}} = (\boldsymbol{\lambda}^T \mu)^T$ and the extended matrix of the dot products between support vectors

$$
\tilde{\mathbf{K}}_{SV} = \begin{pmatrix} \mathbf{K} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{pmatrix},
$$

the value of the span can be written as:

$$
S_p^2 = \min_{\boldsymbol{\lambda}} \max_{\mu} (K(\mathbf{x}_p, \mathbf{x}_p) - 2\mathbf{v}^T \tilde{\boldsymbol{\lambda}} + \tilde{\boldsymbol{\lambda}}^T \mathbf{H} \tilde{\boldsymbol{\lambda}}),
$$

where $\mathbf{H}$ is the submatrix of $\tilde{\mathbf{K}}_{SV}$ with row and column p removed, and $\mathbf{v}$ is the p-th column of $\tilde{\mathbf{K}}_{SV}$.

From the fact that the optimal value of $\tilde{\boldsymbol{\lambda}}$ is $\mathbf{H}^{-1}\mathbf{v}$, it follows:

$$
\begin{aligned}
S_p^2 &= K(\mathbf{x}_p, \mathbf{x}_p) - \mathbf{v}^T \mathbf{H}^{-1} \mathbf{v} \\
&= 1/(\tilde{\mathbf{K}}_{SV}^{-1})_{pp}. \qquad (12)
\end{aligned}
$$

The last equality comes from the following block matrix identity, known as the "Woodbury" formula [13]

$$
\begin{pmatrix} \mathbf{A}_1 & \mathbf{A}^T \\ \mathbf{A} & \mathbf{A}_2 \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{B}_1 & \mathbf{B}^T \\ \mathbf{B} & \mathbf{B}_2 \end{pmatrix},
$$

where $\mathbf{B}_1 = (\mathbf{A}_1 - \mathbf{A}\mathbf{A}_2^{-1}\mathbf{A}^T)^{-1}$.

The closed form we obtain is particularly attractive since we can compute the value of the span for each support vector just by inverting the matrix $\mathbf{K}_{SV}$.

Combining equation (12) and (11), we get the derivative of the span

$$\frac{\partial S_p^2}{\partial \theta_p} = S_p^4 \left(\tilde{\mathbf{K}}_{SV}^{-1} \frac{\partial \tilde{\mathbf{K}}_{SV}}{\theta_p} \tilde{\mathbf{K}}_{SV}^{-1} \right)_{pp}$$

Thus, the complexity of computing the derivative of the span-rule with respect to a parameter θ_p of the kernel requires only the computation of $\frac{\partial K(\mathbf{x}_i, \mathbf{x}_j)}{\partial \theta_p}$ and the inversion of the matrix $\tilde{\mathbf{K}}_{SV}$. The complexity of these operations is not larger than that of the quadratic optimization problem itself.

There is however a problem in this approach: the value given by the span-rule is not continuous. By changing smoothly the value of the parameters $\boldsymbol{\theta}$, the coefficients α_p change continuously, but the span S_p^2 does not. There is actually a discontinuity for most support vectors when the set of support vectors changes. This can be easily understood from equation (6): suppose that upon changing the value of the parameter from $\boldsymbol{\theta}$ to $\boldsymbol{\theta} + \boldsymbol{\varepsilon}$, a point $\mathbf{x}_m$ is not a support vector anymore, then for all other support vectors $(\mathbf{x}_p)_{p \neq m}$, the set Λ_p is going to be smaller and a discontinuity is likely to appear for the value of $S_p = d(\Phi(\mathbf{x}_p), \Lambda_p)$.

The situation is explained in figure 2: we plotted the value of the span of a support vector $\mathbf{x}_p$ versus the width of an RBF kernel σ. Almost everywhere the span is decreasing, hence a negative derivative, but some jumps appear, corresponding to a change in the set of support vectors. Moreover the span is globally increasing: the value of the derivate does not give us a good indication of the global evolution of the span.

One way to solve is this problem is to try to smooth the behavior of the span. This can be done by imposing the following additional constraint in the definition of Λ_p in equation (6): $|\lambda_i| \leq c\ \alpha_i^0$, where c is a constant. Given this constraint, if a point $\mathbf{x}_m$ is about to leave or has just entered the set of support vectors, it will not have a large influence on the span of the other support vectors, since α_m^0 will be small. The effect of this constraint is to make the set Λ_p become "continuous" when the set of support vectors changes.

However this new constraint prevents us from computing the span as efficiently as in equation (12). A possible solution is to replace the constraint

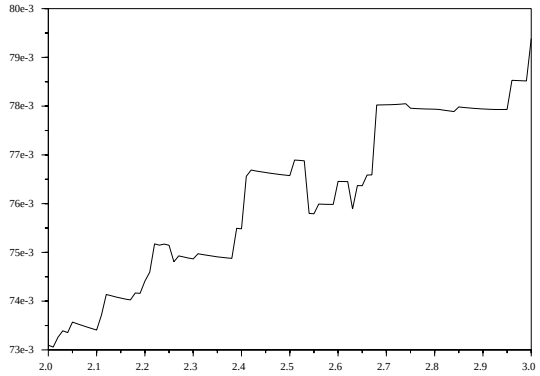

Figure 2: Value of $\sum S_p^2$, the sum of the span of the training points for different values of the width of an RBF kernel varying in the small vicinity

by a regularization term in the computation of the span:

$$S_p^2 = \min_{\boldsymbol{\lambda},\ \sum \lambda_i = 1} \left\| \Phi(\mathbf{x}_p) - \sum_{i \neq p}^{n} \lambda_i \Phi(\mathbf{x}_i) \right\|^2 + \eta \sum_{i \neq p}^{n} \frac{1}{\alpha_i^0} \lambda_i^2$$

With this new definition of the span, equation (12) becomes:

$$S_p^2 = 1/(\tilde{\mathbf{K}}_{SV} + \mathbf{D})_{pp}^{-1} - \mathbf{D}_{pp},$$

where $\mathbf{D}$ is a diagonal matrix with elements $\mathbf{D}_{ii} = \eta/\alpha_i^0$ and $\mathbf{D}_{n+1,n+1} = 0$. As shown on figure 3 , the span is now much smoother and its minimum is still at the right place. In our experiments, we took $\eta = 0.1$.

Note that computing the derivative of this new expression is no more difficult than the previous span expression.

It is interesting to look at the leave-one-out error for SVMs without threshold. In this case, the value of the span with regularization writes:

$$S_p^2 = \min_{\boldsymbol{\lambda}} \left\| \Phi(\mathbf{x}_p) - \sum_{i \neq p} \lambda_i \Phi(\mathbf{x}_i) \right\|^2 + \eta \sum_{i \neq p}^{n} \frac{1}{\alpha_i^0} \lambda_i^2$$

As already pointed out in section 3.2.5, if $\eta = 0$, the value of span is:

$$S_p^2 = \frac{1}{(\mathbf{K}_{SV}^{-1})_{pp}}.$$

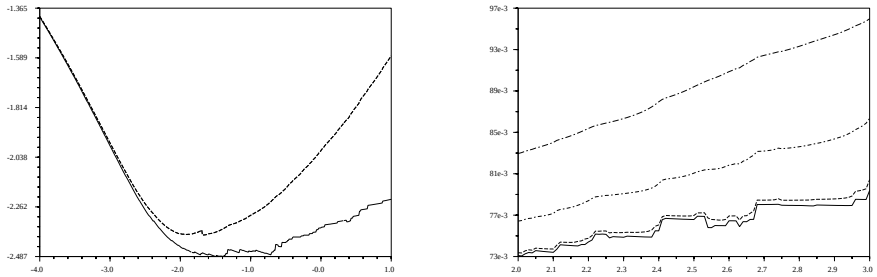

Figure 3: Left: the minima of the span with regularization (dashed line) and without regularization (solid line) are close. Right: detailed behavior of the span for different values of the regularizer, $\eta = 0,\ 0.001,\ 0.01,\ 0.1$

and we recover the Opper-Winther bound.

On the other hand, if $\eta = +\infty$, then $\boldsymbol{\lambda} = 0$ and $S_p^2 = K(\mathbf{x}_p, \mathbf{x}_p)$. In this case, the span bound is identical to the Jaakkola-Haussler one.

In a way, the span bound with regularization is in between the bounds of Opper-Winther and Jaakkola-Haussler.

7 Experiments

Experiments have been carried out to assess the performance and feasibility of our method.

The first set of experiments consists in finding automatically the optimal value of two parameters: the width of an RBF kernel and the constant C in equation (3). The second set of experiments corresponds to the optimization of a large number of scaling factors in the case of handwritten digit recognition. We then show that optimizing scaling factors leads naturally to feature selection and demonstrate the application of the method to the selection of relevant features in several databases.

7.1 Optimization details

The core of the technique we present here is a gradient descent algorithm. We used the optimization toolbox of Matlab to perform it. It includes second order updates to improve the convergence speed. Since we are not interested in the exact value of the parameters minimizing the functional, we used a loose stopping criterion.

	Cross-validation	R^2/γ^2	Span-bound
Breast Cancer	26.04 ± 4.74	26.84 ± 4.71	25.59 ± 4.18
Diabetis	23.53 ± 1.73	23.25 ± 1.7	23.19 ± 1.67
Heart	15.95 ± 3.26	15.92 ± 3.18	16.13 ± 3.11
Thyroid	4.80 ± 2.19	4.62 ± 2.03	4.56 ± 1.97
Titanic	22.42 ± 1.02	22.88 ± 1.23	22.5 ± 0.88

Table 1: Test error found by different algorithms for selecting the SVM parameters C and σ. The first column reports the results from [16]. In the second and last column, the parameters are found by minimizing R^2/γ^2 and the span-bound using a gradient descent algorithm.

7.2 Benchmark databases

In a first set of experiments, we tried to select automatically the width σ of a RBF kernel,

$$K(\mathbf{x}, \mathbf{z}) = \exp\left(-\sum_{i=1}^{n} \frac{(x_i - z_i)^2}{2n\sigma^2}\right)$$

along the constant C penalizing the training error appearing in equation (3).

In order to avoid adding positivity constraints in the optimization problem (for the constant C and the width σ of the RBF kernel), we use the parameterization $\boldsymbol{\theta} = (\log C, \log \sigma)$. Moreover, this turns out to give a more stable optimization. The initial values are $C = 1$ and $\log \sigma = -2$. Each component being normalized by its standard deviation, this corresponds to a rather small value for σ.

We used benchmark databases described in [16]. Those databases, as long as the 100 differents training and test splits are available at
`http://ida.first.gmd.de/~raetsch/data/benchmarks.htm`.

We followed the same experimental setup as in [16]. On each of the first 5 training sets, the kernel parameters are estimated using either 5-fold cross-validation, minimization of R^2/γ^2, or the span-bound. Finally, the kernel parameters are computed as the median of the 5 estimations.

The results are shown in table 1.

It turns out that minimizing R^2/γ^2 or the span estimates yields approximately the same performances as picking-up the parameters which minimize the cross-validation error. This is not very surprising since cross-validation is known to be an accurate method for choosing the hyper-parameters of

	Cross-validation	R^2/γ^2	Span-bound
Breast Cancer	500	14.2	7
Diabetis	500	12.2	9.8
Heart	500	9	6.2
Thyroid	500	3	11.6
Titanic	500	6.8	3.4

Table 2: Average number of SVM trainings on one training set needed to select the parameters C and σ using standard cross-validation or by minimizing R^2/γ^2 or the span-bound.

any learning algorithm.

A more interesting comparison is the computational cost of these methods. Table 2 shows how many SVM trainings in average are needed to select the kernel parameters on each split. The results for cross-validation are the ones reported in [16]. They tried 10 different values for C and σ and performed 5-fold cross-validation. The number of SVM trainings on each of the 5 training set needed by this method is $10 \times 10 \times 5 = 500$.

The gain in complexity is impressive: on average *100* times fewer SVM training iterations are required to find the kernel parameters. The main reason for this gain is that there were two parameters to optimize. Because of computational reasons, exhaustive search by cross-validation can not handle the selection of more than 2 parameters, whereas our method can, as highlighted in the next section.

Discussion As explained in section 3.2, R^2/γ^2 can seem to be a rough upper bound of the span-bound, which is in an accurate estimate of the test error [3]. However in the process of choosing the kernel parameters, what matters is to have a bound whose minimum is close to the optimal kernel parameters. Even if R^2/γ^2 cannot be used to estimate the test error, the previous experiments show that its minimization yields quite good results. The generalization error obtained by minimizing the span-bound (cf table 1) are just slightly better. Since the minimization of the latter is more difficult to implement and to control (more local minima), we recommend in practice to minimize R^2/γ^2. In the experiments of the following section, we will only relate experiments with this bound, but similar results have been obtained with the span-bound.

7.3 Automatic selection of scaling factors

In this experiment, we try to choose the scaling factors for an RBF and polynomial kernel of degree 2. More precisely, we consider kernels of the following form:

$$K(\mathbf{x}, \mathbf{z}) = \exp\left(-\sum_i \frac{(x_i - z_i)^2}{2\sigma_i^2}\right)$$

and

$$K(\mathbf{x}, \mathbf{z}) = \left(1 + \sum_i \frac{x_i z_i}{\sigma_i^2}\right)^2$$

Most of the experiments have been carried out on the USPS handwritten digit recognition database. This database consists of 7291 training examples and 2007 test examples of digit images of size 16x16 pixels. We try to classify digits 0 to 4 against 5 to 9. The training set has been split into 23 subsets of 317 examples and each of this subset has been used successively during the training.

To assess the feasibility of our gradient descent approach for finding kernel parameters, we first used only 16 parameters, each one corresponding to a scaling factor for a squared tile of 16 pixels as shown on figure 4.

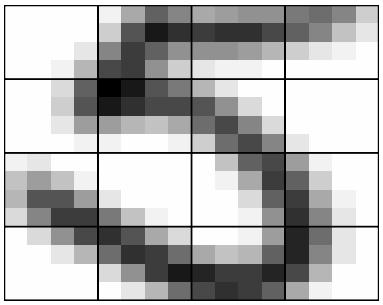

Figure 4: On each of the 16 tiles, the scaling factors of the 16 pixels are identical.

The scaling parameters were initialized to 1. The evolution of the test error and of the bound R^2/γ^2 is plotted versus the number of iterations in the gradient descent procedure in figures 5 (polynomial kernel) and 6 (RBF kernel).

Note that for the polynomial kernel, the test error went down to 9% whereas the best test error with only one scaling parameter is 9.9%. Thus,

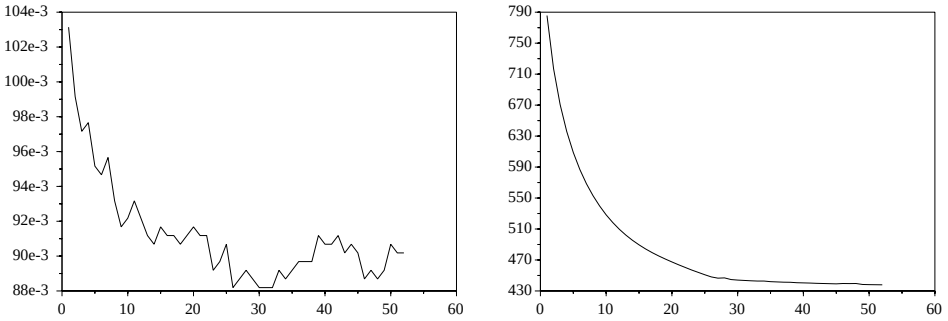

Figure 5: Evolution of the test error (left) and of the bound R^2/γ^2 (right) during the gradient descent optimization with a polynomial kernel

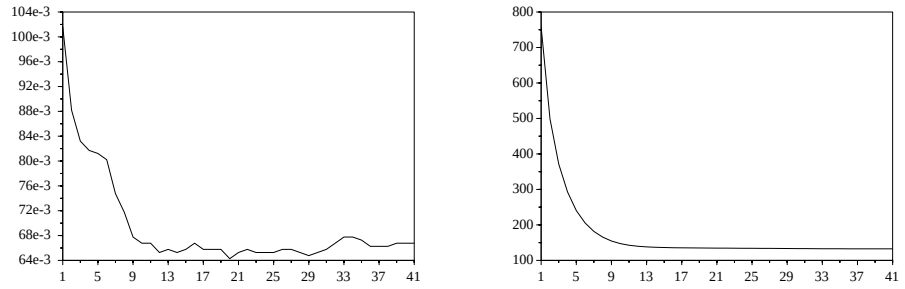

Figure 6: Evolution of the test error (left) and of the bound R^2/γ^2 (right) during the gradient descent optimization with an RBF kernel

by taking several scaling parameters, we managed to make the test error decrease.

It might be interesting to have a look at the value of the scaling coefficients we have have found. For this purpose, we took 256 scaling parameters (one per pixel) and minimized R^2/γ^2 with a polynomial kernel. The map of the scaling coefficient is shown in figure 7.

The result is quite consistent with what one could expect in such a situation: the coefficients near the border of the picture are smaller than those in the middle of the picture, so that these coefficients can be directly interpreted as measures of the relevance of the corresponding features.

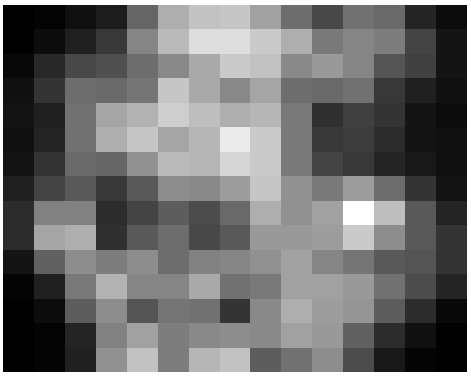

Figure 7: Scaling factors found by the optimization procedure: darker means smaller scaling factor

Discussion This experiment can be considered as a sanity check experiment. Indeed, it proves it is feasible to choose multiple kernel parameters of an SVM and that it does not lead to overfitting. However, the gain in test error was not our main motivation since we did not expect any significant improvement on such a problem where most features play a similar role (taking all scaling factors equal on this database seems a reasonable choice). However as highlighted by figure 7, this method can be a powerful tool to perform feature selection.

8 Feature selection

The motivation for feature selection is three-fold:

1. Improve generalization error
2. Determine the relevant features (for explanatory purposes)
3. Reduce the dimensionality of the input space (for real-time applications)

Finding optimal scaling parameters can lead to feature selection algorithms. Indeed, if one of the input components is useless for the classification problem, its scaling factor is likely to become small. But if a scaling factor becomes small enough, it means that it is possible to remove it without affecting the classification algorithm. This leads to the following idea

for feature selection: keep the features whose scaling factors are the largest. This can also be performed in a principal components space where we scale each principal component by a scaling factor.

We consider two different parametrization of the kernel. The first one correspond to rescaling the data in the input space:

$$K_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) = K(\boldsymbol{\theta}^T \mathbf{x}, \boldsymbol{\theta}^T \mathbf{z})$$

where $\boldsymbol{\theta} \in \mathbb{R}^n$.

The second one corresponds to rescaling in the principal components space:

$$K_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) = K(\boldsymbol{\theta}^T \boldsymbol{\Sigma} \mathbf{x}, \boldsymbol{\theta}^T \boldsymbol{\Sigma} \mathbf{z})$$

where $\boldsymbol{\Sigma}$ is the matrix of principal components.

We compute $\boldsymbol{\theta}$ and $\boldsymbol{\alpha}$ using the following iterative procedure:

1. Initialize $\boldsymbol{\theta} = (1, \ldots, 1)$
2. In the case of principal component scaling, perform principal component analysis to compute the matrix $\boldsymbol{\Sigma}$.
3. Solve the SVM optimization problem
4. Minimize the estimate of the error T with respect to $\boldsymbol{\theta}$ with a gradient step.
5. If a local minimum of T is not reached go to step 3.
6. Discard dimensions corresponding to small elements in $\boldsymbol{\theta}$ and return to step 2.

We demonstrate this idea on two toy problems where we show that feature selection reduces generalization error. We then apply our feature selection algorithm to DNA Micro-array data where it is important to find which genes are relevant in performing the classification. It also seems in these types of algorithms that feature selection improves performances. Lastly, we apply the algorithm to face detection and show that we can greatly reduce the input dimension without sacrificing performance.

8.1 Toy data

We compared several algorithms

- The standard SVM algorithm with no feature selection

- Our feature selection algorithm with the estimate R^2/γ^2 and with the span estimate
- The standard SVM applied after feature selection via a filter method

The three filter methods we used choose the m largest features according to: Pearson correlation coefficients, the Fisher criterion score[4], and the Kolmogorov-Smirnov test[5]. Note that the Pearson coefficients and Fisher criterion cannot model nonlinear dependencies.

In the two following artificial datasets our objective was to assess the ability of the algorithm to select a small number of target features in the presence of irrelevant and redundant features [22].

For the first example, six dimensions of 202 were relevant. The probability of $y = 1$ or -1 was equal. The first three features $\{x_1, x_2, x_3\}$ were drawn as $x_i = yN(i, 1)$ and the second three features $\{x_4, x_5, x_6\}$ were drawn as $x_i = N(0, 1)$ with a probability of 0.7, otherwise the first three were drawn as $x_i = N(0, 1)$ and the second three as $x_i = yN(i - 3, 1)$. The remaining features are noise $x_i = N(0, 20)$, $i = 7, \ldots, 202$.

For the second example, two dimensions of 52 were relevant. The probability of $y = 1$ or -1 was equal. The data are drawn from the following: if $y = -1$ then $\{x_1, x_2\}$ are drawn from $N(\mu_1, \Sigma)$ or $N(\mu_2, \Sigma)$ with equal probability, $\mu_1 = \{-\frac{3}{4}, -3\}$ and $\mu_2 = \{\frac{3}{4}, 3\}$ and $\Sigma = I$, if $y = 1$ then $\{x_1, x_2\}$ are drawn again from two normal distributions with equal probability, with $\mu_1 = \{3, -3\}$ and $\mu_2 = \{-3, 3\}$ and the same Σ as before. The rest of the features are noise $x_i = N(0, 20)$, $i = 3, \ldots, 52$.

In the linear problem the first six features have redundancy and the rest of the features are irrelevant. In the nonlinear problem all but the first two features are irrelevant.

We used a linear kernel for the linear problem and a second order polynomial kernel for the nonlinear problem.

We imposed the feature selection algorithms to keep only the best two features. The results are shown in figure (8) for various training set sizes, taking the average test error on 500 samples over 30 runs of each training set size. The Fisher score (not shown in graphs due to space constraints) performed almost identically to correlation coefficients.

[4] $F(r) = \left|\frac{\mu_r^+ - \mu_r^-}{\sigma_r^+ + \sigma_r^-}\right|$, where $\mu_r^\pm$ is the mean value for the r-th feature in the positive and negative classes and $\sigma_r^\pm$ is the standard deviation

[5] $\mathrm{KS}_{tst}(r) = \sqrt{\ell}\ \sup\left(\hat{P}\{X \le f_r\} - \hat{P}\{X \le f_r, y_r = 1\}\right)$ where f_r denotes the r-th feature from each training example, and $\hat{P}$ is the corresponding empirical distribution.

In both problem, we clearly see that our method outperforms the other classical methods for feature selection. In the nonlinear problem, among the filter methods only the Kolmogorov-Smirnov test improved performance over standard SVMs.

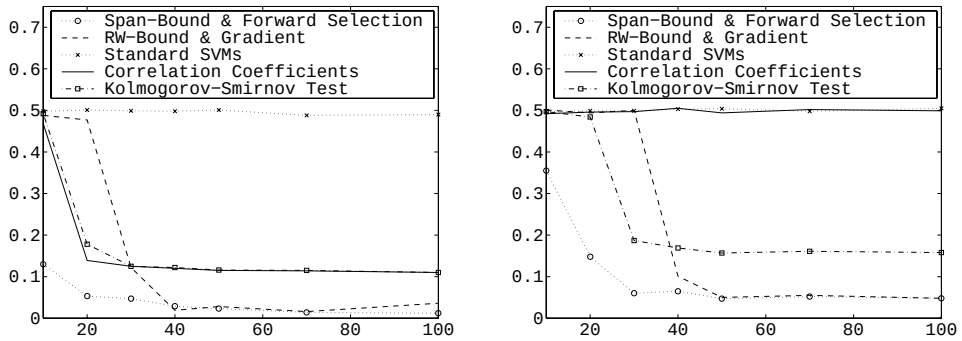

Figure 8: A comparison of feature selection methods on (a) a linear problem and (b) a nonlinear problem both with many irrelevant features. The x-axis is the number of training points, and the y-axis the test error as a fraction of test points.

8.2 DNA Microarray Data

Next, we tested this idea on two leukemia discrimination problems [7] and a problem of predicting treatment outcome for Medulloblastoma [6]. The first problem was to classify myeloid versus lymphoblastic leukemias based on the expression of 7129 genes. The training set consists of 38 examples and the test set 34 examples. Standard linear SVMs achieve 1 error on the test set. Using gradient descent on R^2/γ^2 we achieved 0 error using 30 genes and 1 error using 1 gene. Using the Fisher score to select features resulted in 1 error for both 1 and 30 genes.

The second leukemia classification problem was discriminating B versus T cells for lymphoblastic cells [7]. Standard linear SVMs make 1 error for this problem. Using either the span bound or gradient descent on R^2/γ^2 results in 0 error using 5 genes, whereas the Fisher score get 2 errors using the same number of genes.

The final problem is one of predicting treatment outcome of patients that have Medulloblastoma. Here there are 60 examples each with 7129

[6] The database will be available at : `http://waldo.wi.mit.edu/MPR/data_sets.html`

expression values in the dataset and we use leave-one-out to measure the error rate. A standard SVM with a Gaussian kernel makes 24 errors, while selecting 60 genes using the gradient descent on R^2/γ^2 we achieved an error of 15.

8.3 Face detection

The trainable system for detecting frontal and near-frontal views of faces in gray images presented in [8] gave good results in terms of detection rates. The system used gray values of 19×19 images as inputs to a second-degree polynomial kernel SVM. This choice of kernel lead to more than 40,000 features in the feature space. Searching an image for faces at different scales took several minutes on a PC. To make the system real-time reducing the dimensionality of the input space and the feature space was required. The feature selection in principal components space was used to reduce the dimensionality of the input space [17].

The method was evaluated on the large CMU test set 1 consisting of 479 faces and about 57,000,000 non-face patterns. In Figure 9, we compare the ROC curves obtained for different numbers of selected components.

The results showed that using more than 60 components does not improve the performances of the system [17].

9 Conclusion

We proposed an approach for automatically tuning the kernel parameters of an SVM. This is based on the possibility of computing the gradient of various bounds on the generalization error with respect to these parameters. Different techniques have been proposed to smooth these bounds while preserving their accuracy in predicting the location of the minimum of test error. Using these smoothed gradients we were able to perform gradient descent to search the kernel parameter space, leading to both an improvement of the performance and a reduction of the complexity of the solution (feature selection). Using this method, we chose in the separable case appropriate scaling factors. In the non separable case, this method allows us to choose simultaneously scaling factors and parameter C (see equation 3).

The benefits of this technique are many. First it allows to actually optimize a large number of parameters while previous approaches only could deal with 2 parameters at most. Even in the case of a small number of parameters, it improves the run time by a large amount. Moreover experimental results have demonstrated that an accurate estimate of the error

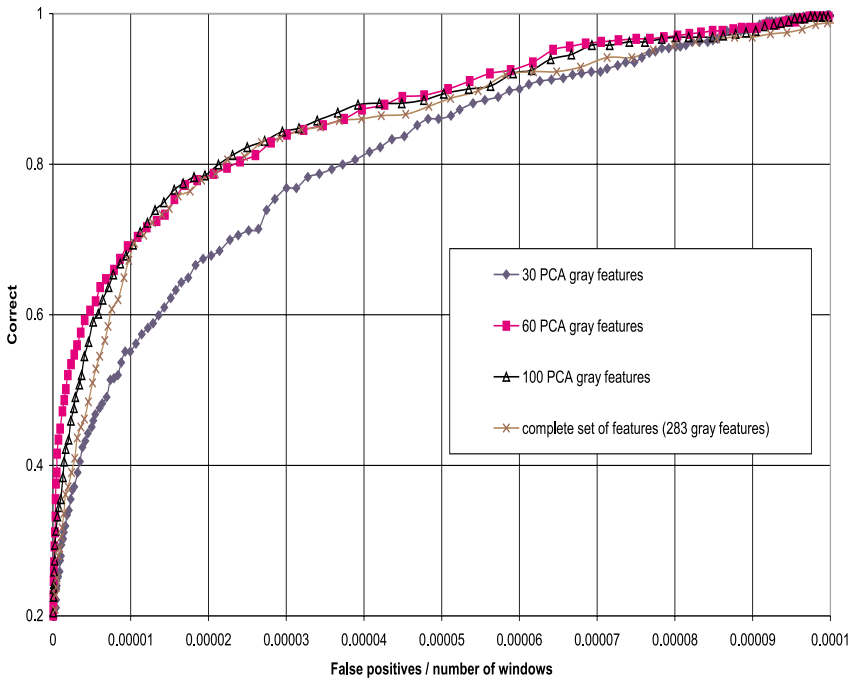

Figure 9: ROC curves for different number of PCA gray features.

is not required and that a simple estimate like R^2/γ^2 has a very good behaviour in terms of finding the right parameters. In a way this renders the technique even more applicable since this estimate is very simple to compute and derive. Finally, this approach avoids holding out some data for validation and thus makes full use of the training set for the optimization of parameters, contrary to cross-validation methods.

This approach and the fact that it has be proven successful in various situation opens new directions of research in the theory and practice of Support Vector Machines. On the practical side, this approach makes possible the use of highly complex and tunable kernels, the tuning of scaling factors for adapting the shape of the kernel to the problem and the selection of relevant features. On the theoretical side, it demonstrates that even when a large number of parameter are simultaneously tuned the overfitting effect remains low.

Of course a lot of work remains to be done in order to properly understand the reasons. Another interesting phenomenon is the fact that the quantitative accuracy of the estimate used for the gradient descent is only

marginally relevant. This raises the question of how to design good estimates for parameter tuning rather than accurate estimates.

Future investigation will focus on trying to understand these phenomena and obtain bounds on the generalization error of the overall algorithm, along with looking for new problems where this approach could be applied as well as new applications.

Acknowledgments

The authors would like to thank Jason Weston and Élodie Nédélec for helpful comments and discussions.

References

[1] Y. Bengio. Gradient-based optimization of hyper-parameters. *Neural Computation*, 12(8), 2000.

[2] J.F. Bonnans and A. Shapiro. *Perturbation Analysis of Optimization Problems*. Springer-Verlag, 2000.

[3] O. Chapelle and V. Vapnik. Model selection for support vector machines. In *Advances in Neural Information Processing Systems*, 1999.

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

[5] N. Cristianini, C. Campbell, and J. Shawe-Taylor. Dynamically adapting kernels in support vector machines. In *Advances in Neural Information Processing Systems*, 1999.

[6] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

[7] T. Golub, D. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, C.D. Bloomfield, and E.S. Lander. Molecular classification of cancer : Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

[8] B. Heisele, T. Poggio, and M. Pontil. Face detection in still gray images. AI Memo 1687, Massachusetts Institute of Technology, 2000.

[9] T. S. Jaakkola and D. Haussler. Probabilistic kernel regression models. In *Proceedings of the 1999 Conference on AI and Statistics*, 1999.

[10] T. Joachims. Estimating the generalization performance of a svm efficiently. In *Proceedings of the International Conference on Machine Learning*. Morgan Kaufman, 2000.

[11] J. Larsen, C. Svarer, L.N. Andersen, and L.K. Hansen. Adaptive regularization in neural network modeling. In G.B. Orr and K.R. Müller, editors, *Neural Networks : Trick of the Trade*. Springer, 1998.

[12] A. Luntz and V. Brailovsky. On estimation of characters obtained in statistical procedure of recognition. *Technicheskaya Kibernetica*, 3, 1969. (in Russian).

[13] H. Lütkepohl. *Handbook of Matrices*. Wiley & Sons, 1996.

[14] M. Opper and O. Winther. Gaussian processes and svm: Mean field and leave-one-out. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 311–326, Cambridge, MA, 2000. MIT Press.

[15] John Platt. Probabilities for support vector machines. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press, Cambridge, MA, 2000.

[16] G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, 2001.

[17] T. Serre, B. Heisele, S. Mukherjee, and T. Poggio. Feature selection for face detection. AI Memo 1697, Massachusetts Institute of Technology, 2000.

[18] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[19] V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

[20] V. Vapnik and O. Chapelle. Bounds on error expectation for support vector machines. *Neural Computation*, 12(9), 2000.

[21] G. Wahba, Y. Lin, and H. Zhang. Generalized approximate cross-validation for support vector machines : another way to look at margin-like quantities. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 297–309. MIT Press, 2000.

[22] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for support vector machines. In *Advances in Neural Information Processing Systems*, 2000.