Optimization of the SVM Kernels using an Empirical Error Minimization Scheme.

N.E. Ayat [1,2], M. Cheriet [1], and C.Y. Suen [2]

[1] LIVIA, ÉTS, 1100, rue Notre Dame Ouest, Montreal, H3C 1K3, Canada.
[2] CENPARMI, Concordia University, 1455 de Maisonneuve Blvd West, Montreal, H3G 1M8, Canada.
Emails: ayat@livia.etsmtl.ca, cheriet@gpa.etsmtl.ca, suen@cenparmi.concordia.ca

Abstract. We address the problem of optimizing kernel parameters in Support Vector Machine modelling, especially when the number of parameters is greater than one as in polynomial kernels and KMOD, our newly introduced kernel. The present work is an extended experimental study of the framework proposed by Chapelle et al. for optimizing SVM kernels using an analytic upper bound of the error. However, our optimization scheme minimizes an empirical error estimate using a Quasi-Newton technique. The method has shown to reduce the number of support vectors along the optimization process. In order to assess our contribution, the approach is further used for adapting KMOD, RBF and polynomial kernels on synthetic data and NIST digit image database. The method has shown satisfactory results with much faster convergence in comparison with the simple gradient descent method.
Furthermore, we also experimented two more optimization schemes based respectively on the maximization of the margin and on the minimization of an approximated VC dimension estimate. While both of the objective functions are minimized, the error is not. The corresponding experimental results we carried out show this shortcoming.

1 Introduction

Any learning machine embeds hyper-parameters that may deteriorate its performance if not well chosen. These parameters have a regularization effect on the optimization of the objective function. For any classification task, picking the best values for these parameters is a non trivial model selection problem that needs either an exhaustive search over the space of hyper-parameters or optimized procedures that explore only a finite subset of the possible values. The same consideration apply to the Support Vector Machine in that kernel parameters and tradeoff parameter (C) are hyper-parameters that one needs to find their optimal values. Until now, many practitioners select these parameters empirically by trying a finite number of values and keeping those that provide the least test error. Our primary interest through this work is to optimize the SVM classifier for any kernel function used. More specifically, we seek an automatic method for model selection that optimizes the kernel profile dependently of the

data. This allows better performance for the classifier and rigorous comparison among different kernel results as well.

In the literature, there exist many published works that consider the problem of hyper-parameters adaptation in Neural Networks. The concern herein is to automatically fix those parameters involved in the training process which are not, however, explicitly linked to the resulting decision function. Much fewer, however, dealt with the SVM classifiers. Recently, Chapelle et al. in [13] proposed a gradient descent framework for optimizing kernels by minimizing an analytic upper bound on the generalization error. This bound equals to the ratio of the data enclosing sphere radius over the separation margin value. Whereas the computation of the margin is somewhat straightforward given the trained model, estimating the radius of the enclosing sphere needs a quadratic optimization procedure that may be difficult to proceed especially for large scale problems handling tens of thousands of data entries and hundreds of support vectors. This represents a serious handicap to the application of the aforementioned scheme on relatively large databases such as NIST digit image database. One more drawback in [13], is that the gradient descent method requires several iterations, namely hundreds, before convergence would be ensured. This is due to the fact the gradient direction is not necessarily the best search direction for the algorithm to reach a minimum. Clearly, a gradient descent method will throw away a lot of time before multiple SVMs are optimized for a 10-class problem.

Instead, an alternative method that we investigate consists of optimizing the SVM hyper-parameters by minimizing an empirical estimate of the error through a validation set. The present work is an experimental study of the method. In addition, we make use of a Quasi-Newton variant to adapt the hyper-parameters for its convergence efficiency. This optimization algorithm proceeds in two steps: computing the search direction and computing the step along this direction. Furthermore, in order to compute the error estimate, we considered probabilistic outputs for the SVM by fitting a two-parameter logistic function to the classifier output [10]. This provides a posterior probability measure that allows good estimation of the probability of error over the validation data points. Through the use of such a function, the estimated error on the validation set is also smoothed, so one can compute its gradient for the optimization.

The experimental results we obtained on a toy classification problem show a faster convergence for our method. As well, we find out that the presented scheme minimizes the number of support vectors which is a pessimistic upper bound of the generalization error. We have compared the method to two minimization procedures based on the maximization of the margin and on the minimization of an estimate of the VC dimension [16,20]. The results have shown these methods are ineffective to reduce the generalization error.

In section 2, we present the standard Support Vector Machine formulation. Also, we introduce some of the state of the art kernels usually used for classification. A brief recall of the properties of our previously introduced kernel–KMOD will also be found. In section 3, a detailed description of the optimization method is given. The posterior probability mapping algorithm is described too. Section

3.3 presents an experimental study of the method on synthetic two-class data. In section 4, a real-life ten-class problem is considered. The strategy of learning, optimization and classification on NIST digit images are well detailed as well. Finally, a summary of the work is presented in section 5.

2 SVMs for classification: background

The SVM is a structural risk minimization based classifier which finds an optimum decision region. Let us have a data set $\{x_i, y_i\}, i = 1, \ldots, l$, where $y_i \in \{-1, 1\}$ represents the label of an arbitrary example $x_i \in \Re^d$; d being the dimension of the input space. Also, let us define a linear decision surface by the equation $f(x) = w.x+b = 0$. The original formulation of support vector machine algorithm seeks a linear decision surface that maximizes the margin between positive and negative examples. This can be achieved through the minimization of $\|w\|^2$ which solution is

$$w = \sum_{i=1}^{l} \alpha_i y_i x_i \tag{1}$$

with the constraint $\sum_i \alpha_i y_i = 0$. The parameters α_i are found by maximizing the dual objective [19,17]

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i.x_j. \tag{2}$$

However, in real-life classification problems, the algorithm as stated above is unable to achieve perfect separation between two classes especially in case of noisy data. Cortes et al. in [6] slightly modified the model by adding an heuristic that accounts for accepting misclassified examples while penalizing them. Mathematically, this does not imply any major modification except that α_i must be upper bounded as $0 \leq \alpha_i \leq C$, where C is a penalization parameter also called trade-off parameter. An infinite value for C yields a classifier that seeks a well separated data. In another work, Boser et al. [5] added an important feature that allows a different insight into support vector theory. In fact, they enable these models to produce complex nonlinear boundaries in the original space. The technique consists of projecting the data into higher order spaces of possibly infinite dimension through a mapping function ϕ. This function, however, must keep the inner product formulation in Eq. 2 still useful. Furthermore, the explicit analytical form of functions ϕ must not be known. Only the expression of their pairwise inner product $k(x, y) = \phi(x).\phi(y)$ in the augmented space has to be defined. This dot product defines a particular subset of kernels called Mercer kernels [7] (see Table 1). The dual objective of Eq. 2 become then

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j).$$

Kernels	Formula
linear	$k(x,y) = x.y$
sigmoid	$k(x,y) = \tanh(ax.y + b)$
polynomial	$k(x,y) = (1 + x.y)^d$
RBF	$k(x,y) = \exp(-a\|x-y\|^2)$
exponential RBF	$k(x,y) = \exp(-a\|x-y\|)$

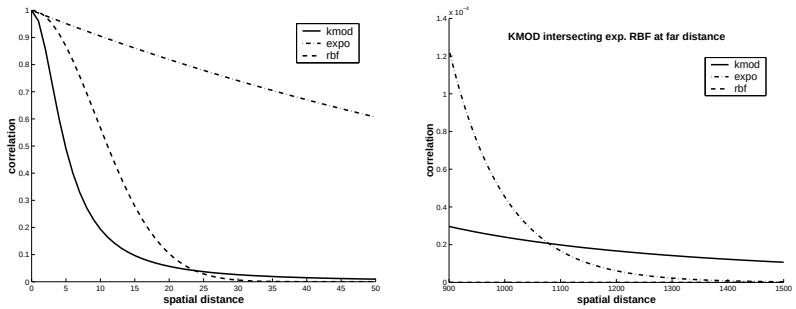
Table 1. *Common kernels*

Recently, we introduced a new two-parameter kernel namely a Kernel with Moderate Decreasing-KMOD [2,3] whose analytic expression is

$$kmod(x,y) = a[\exp(\frac{\gamma}{\|x-y\|^2 + \sigma^2}) - 1]; \tag{3}$$

where a is a normalization constant equal to $\frac{1}{\exp(\frac{\gamma}{\sigma^2})-1}$. The parameters γ and σ are two positive reals that control the decreasing behavior of the kernel. In particular, σ is a scale space parameter that defines a gate surface around zero whereas γ controls the decreasing speed around zero. The -1 bias ensures the kernel converges toward 0 at infinity (see Figure 1). This kernel has a particular behavior that allows a varying speed of decay around zero and a moderate decrease toward infinity. We also undertook the existing duality between similarity in feature space and spatial behavior of kernels. We show that KMOD allows at once good discrimination between patterns while maintaining the closeness information of far apart points from vanishing. Although empirical, some of the experiments we perform using KMOD were encouraging (see Table 2). On the Breast Cancer Database of UCI, for example, KMOD achieves the best results yet established.

In order to assess KMOD as well as any SVM kernel, we undertook the problem of automatically selecting the best kernel parameters so practitioners



$\langle a \rangle$ $\langle b \rangle$

Fig. 1. $\langle a \rangle$ *Correlation in feature space vs. spatial distance in input space;*$\langle b \rangle$ *KMOD preserving the far points closeness information*

classifier	KMOD svm	RBF svm	RBF network	Adaboost	Adaboost Reg
error	25.4±4.4	26.0±4.7	+ 27.6±4.7	+ 30.4±4.7	26.5±4.5

Table 2. *Performance of KMOD (column 1) versus state of the art classifiers on Breast Cancer database [15,3]. The plus sign beside the values indicates wether or not the error is significantly different from KMOD's one*

can compare and choose suitable kernels given different applications data. Next, we address a systematic method for model selection in Support Vector Machines.

3 Model selection

Let us assume any kernel k depends on one or several parameters encoded into a vector $\underline{\theta} = (\theta_1, ..., \theta n)$. Support Vector Machines consider a class of functions parameterized by $\underline{\alpha}$, b and $\underline{\theta}$:

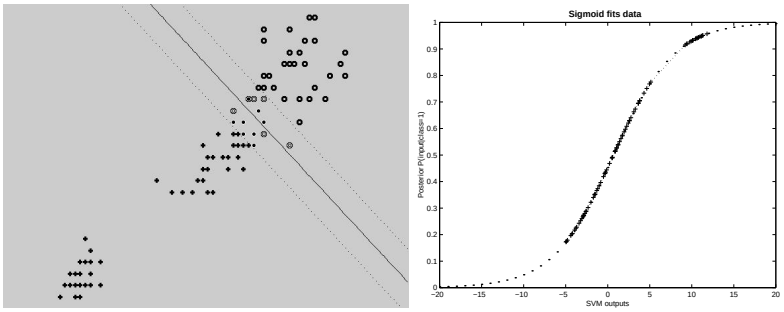
$$f_{\underline{\alpha},b,\underline{\theta}} = \sum_i \alpha_i y_i k_\theta(x, x_i) + b, \tag{4}$$

where $\underline{\alpha}$ is the vector that contain the multipliers α_i and y_i represents the target of support vector x_i (multipliers of non support vectors are zero). Optimizing the SVM hyper-parameters is a model selection problem that needs adapting multiple parameter values at the same time. The parameters to tune are those that embed any kernel function as the σ parameter in an RBF kernel or the couple (γ, σ) in case of KMOD kernel (see Table 1). In addition, another parameter the optimization may consider is the trade-off parameter C which may have a strong effect on the SVM behavior for hard classification tasks. Usually, model selection is done by minimizing an estimate of the generalization error or by default a known upper bound of that error. As in Neural Network area, empirical estimation of the generalization error suggests the use of either a validation error estimate which procedure requires a reduction of the amount of data used for learning; or a leave-one-out error estimate (extreme case of K-fold cross validation) which gives a precise estimate of the true error [4]. However, leave-one-out error estimation is time consuming and may be avoided if enough data could be retained for the validation set. The larger the validation set is, tighter is the variance of the estimated error. This estimate is given by

$$T = \frac{1}{N} \sum_i \Psi(-y_i f(x_i)) \tag{5}$$

where Ψ is a step function (Heaviside function) and N being the size of the validation set. Using a gradient descent approach assumes the error estimate in Eq. 5 to be differentiable. Unfortunately the step function is not. To circumvent this drawback it is possible to use a contracting function of the form $\Psi(x) = \frac{1}{1+\exp(-Ax+B)}$ [10,13]. A very nice way to choose the values of constants A and B is to estimate posterior probabilities [10]; whereby a smooth approximation of the test error is obtained. Next, we describe the method.

3.1 Probability estimation: a two-parameter identification procedure



⟨a⟩ ⟨b⟩

Fig. 2. ⟨a⟩ Separation frontier; ⟨b⟩ Posterior probability versus SVM output value.

Recently, many researchers have considered the problem of probabilities estimation for SVM classifiers. The methods they proposed are of varying levels of complexity. Sollich for example, proposed in [18] a bayesian framework to tackle two of the outstanding challenges in SVM classification: how to obtain predictive class probabilities rather than the conventional deterministic class label predictions and even how to tune hyper-parameters. This very attractive method interprets Support Vector Machines as maximum a posteriori solutions to inference problems with Gaussian process priors. Earlier, Wahba proposed in [8] to use a logistic function of the form

$$P(y = 1|x) = \frac{1}{1 + \exp(-f(x))}$$

where $f(x)$ is the SVM output (without threshold) and $y = \pm 1$ represents the target of the data example x. Platt in [10] used a slightly modified logistic function given by

$$P(y = 1|f) = \frac{1}{1 + \exp(Af + B)}. \tag{6}$$

The two-parameter contracting function he proposed allows to map the SVM output values to the corresponding posteriors. The method is easy to implement and requires a non-linear optimization of the couple of parameters (A, B) such a way the negative log-likelihood

$$\sum_i t_i \log(p_i) + (1 - t_i) \log(1 - p_i)$$

through the validation data points is minimized; where $p_i = \frac{1}{1+\exp(Af_i+B)}$ is the inferred posterior probability and $t_i = \frac{y_i+1}{2}$; with $t_i = 1$ if the input vector

x_i belongs to class C_1 and $t_i = 0$ if x_i belongs to class C_2. For the rest of the paper, we shall refer to y_i as the bipolar target $(+1, -1)$ and to t_i as the binary target $(1, 0)$. The constant B in Eq. 6 allows the threshold probability of 0.5 to correspond to a non zero value for SVM output. Using a contracting function such in Eq. 6 allows to estimate the probability of error on the data and circumvent the need to use the Heaviside function in Eq. 5. It follows that the gradient of the error could now be approximated. We tried out the above mentioned algorithm using a Newton method to adapt the sigmoid in Eq. 6 [10]. Figure 2 shows respectively the separation frontier and the inferred contracting function for a linear SVM trained on iris3v12 data set. Below, we shall integrate this two-parameter identification procedure into the overall optimization process.

3.2 Kernel optimization: a Quasi-Newton scheme

Different approaches for non-linear optimization have shown varying levels of efficiency. Among these methods (gradient descent, conjugate gradient, Newton methods, ...), Quasi-Newton approach has shown faster convergence and stable optimization for Neural Networks. Even though a gradient descent procedure is sufficient to find satisfactory weights values for an MLP or an RBF network, adapting possible hyper-parameters requires an estimation of the gradient after each training process, and then a downhill step toward a local minima could be done [12]. This procedure is time consuming and needs many training process before a feasible solution could be found. In fact, this algorithm has two more drawbacks that must be pointed out. A downhill direction toward a minima is not guaranteed and the algorithm is very sensitive to noise so any discontinuity can lead to an arbitrary step along the error surface. The Quasi-Newton procedure circumvents these disadvantages by ensuring a downhill direction of search through the use of second order information, and computes a feasible amplitude for the step along the search direction. Practically, it proceeds in two steps. First, the search direction must be chosen. This step needs to approximate the corresponding parameters Hessian matrix. Second, a line search minimization along the search direction finds the best amplitude for the computed step. These two features may speed up drastically the convergence of the optimization process. We shall describe below the method to optimize the SVM kernels.

We can formulate the error probability of observing either target value for a given data example x_i as

$$E_i = P(y_i \neq z_i) = p_i^{1-t_i}(1 - p_i)^{t_i}$$

where $z_i = sign(f_i)$, $f_i = f(x_i)$ is the corresponding SVM output value and p_i is the estimated posterior probability. For a validation set of size N, the average estimate of the error over the validation data set could be written as

$$E = \frac{1}{N}\sum_{i=1}^{N} E_i = \frac{1}{N}\sum_{i=1}^{N} p_i^{1-t_i}(1 - p_i)^{t_i}.$$

To approximate the gradient of the error we shall assume the current vector of kernel parameter values is sufficiently close to a local minimum. As the derivative of the error w.r.t $\underline{\theta}$ at the minima vanishes, it follows that we can approximate this gradient as

$$\frac{\partial E}{\partial \underline{\theta}} = \frac{\partial E}{\partial \underline{\alpha}} \frac{\partial \underline{\alpha}}{\partial \underline{\theta}} \tag{7}$$

where $\underline{\alpha} = (\alpha_1, ..., \alpha_k)$ represents the vector of multiplicative parameters and k equals to the number of support vectors. In the other hand, the components $\frac{\partial E}{\partial \alpha_j}$ could be expanded as

$$\frac{\partial E}{\partial \alpha_j} = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial E_i}{\partial p_i} \frac{\partial p_i}{\partial f_i} \frac{\partial f_i}{\partial \alpha_j} \tag{8}$$

where

$$\frac{\partial E_i}{\partial p_i} = -p_i^{1-t_i} t_i (1-p_i)^{t_i - 1} + (1-t_i)(1-p_i)^{t_i} p_i^{-t_i} = \begin{cases} +1 \ if \ t_i = 0 \\ -1 \ if \ t_i = 1 \end{cases} \tag{9}$$

The gradient of the posterior probability w.r.t. to the raw svm output is given by

$$\frac{\partial p_i}{\partial f_i} = -A p_i^2 \exp(A f_i + B) \tag{10}$$

and the gradient of the raw SVM output w.r.t. α_j is given by

$$\frac{\partial f_i}{\partial \alpha_j} = y_i K_\theta(x_j, x_i) \tag{11}$$

where $y_i = \pm 1$ is the bipolar target for the data point x_i.

Once the derivative of the error w.r.t. the multipliers vector $\underline{\alpha}$ is computed, the next step consists of estimating the derivative of $\underline{\alpha}$ w.r.t. hyper-parameters $\underline{\theta}$. Notice that we may include the SVM bias b in the vector $\underline{\alpha}$ such that $\underline{\alpha} = (\alpha_1, ..., \alpha_k, b)$. Moreover, it is shown that

$$\frac{\partial \underline{\alpha}}{\partial \theta} = -H^{-1} \frac{\partial H}{\partial \theta} \underline{\alpha}^T \tag{12}$$

where $H = \begin{pmatrix} K^Y & Y \\ Y^T & 0 \end{pmatrix}$ and the components $K_{ij}^Y = y_i y_j K(x_i, x_j)$ [13]. The vector Y is the target vector corresponding to the support vectors set. Y^T is its transpose. H is a $(\sharp SV + 1) \times (\sharp SV + 1)$ matrix, $\sharp SV$ being the number of support vectors. Next, we shall refer to the matrix H as the kernel's Hessian. This matrix is different from the Quasi-Newton related Hessian. We shall refer to the latter as H'.

Herein we give the optimization algorithm using a Quasi-Newton scheme:

1. Initialize $\underline{\theta}$ to some value.
2. Train the SVM with fixed $\underline{\theta}$.
3. Infer the parameters A and B for the given SVM model and validation set (non-linear optimization procedure).

4. Estimate the probability of error.
5. Calculate the gradient of that error $\frac{\partial E(\underline{\alpha},\underline{\theta})}{\partial \underline{\theta}}$.
6. Calculate the Hessian H'.
7. Update $\underline{\theta}$ using: $\Delta\underline{\theta} = -\lambda H' \frac{\partial E(\underline{\alpha},\underline{\theta})}{\partial \underline{\theta}}$.

where λ is the amplitude of the step along the search direction and H' is a $n \times n$ matrix; n being the dimension of vector $\underline{\theta}$. The used line search minimization algorithm is a variant of the Golden Section Search described in [14].

Besides, it is also possible to choose other criterions for the optimization process. For example, one may consider the minimization of an analytic upper bound of the capacity h (also called VC dimension [19]) given by

$$h < R^2 \|w\|^2, \tag{13}$$

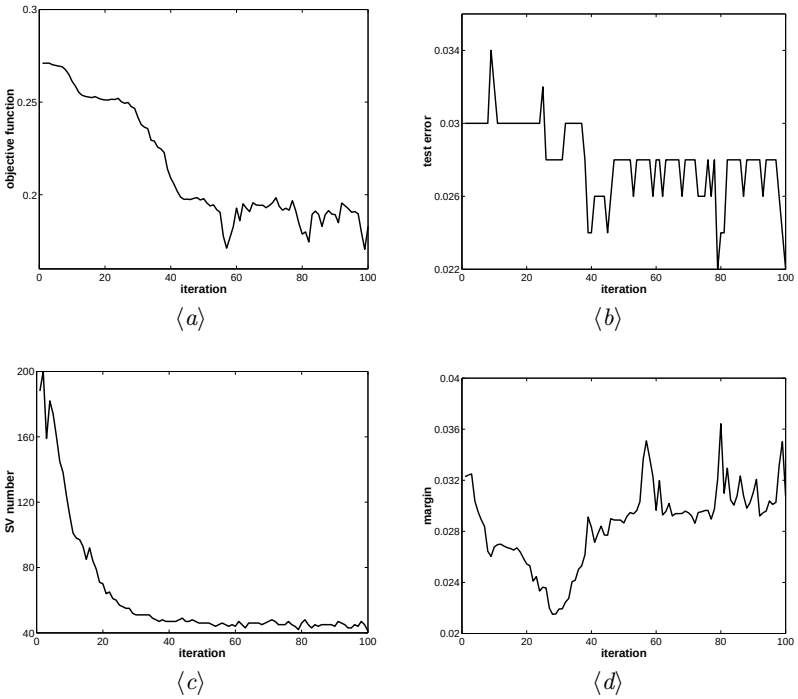
where R is the radius of the smallest sphere enclosing the training data in feature space, and $\|w\|$ is determined by the support vector algorithm and is computed in feature space (see Eq. 1). This criterion has previously been used by Scholkopf in [16] to choose among different degrees for a polynomial kernel. Chapelle et al. in [13] also minimized this criterion as well as the span bound using a gradient descent method.

3.3 Experiments

The core of the used SVM solver is based on the shrinking algorithm implemented in SVM^{light} and detailed in [9]. To invert the kernel Hessian H in Eq. 12, we used an LU decomposition method. This procedure has a computational complexity of $O(n^3)$; n being the dimension of the square matrix H. In addition, some numerical problems could be encountered if H is badly conditioned, i.e. some of the matrix eigenvalues are very close to zero. In fact, since H is guaranteed to be only semi-definite positive, it may happen that the Hessian is singular. One solution consists of adding a very small constant to the diagonal elements of H. Another way consists of using a variant of the Cholesky decomposition algorithm adapted to semi-definite matrices [21]. This problem was mainly remarked for spread kernels, i.e. very small values for σ in case of KMOD kernel and a in case of RBF kernel (Table 1).

For the experiments, we first considered synthetic two-class data. So we produced 2000 data points from two overlapping spherical gaussians with known Bayes error (0.017). Each data point has two attributes and the classes are balanced. For the optimization purpose, we retain a subset of 1000 examples for the training, 500 examples for validation and 500 examples for testing. We did simulation using KMOD kernel with an initial vector of values for (γ, σ) equal to $(\frac{1}{2.72}, \frac{1}{2.72})$ and an initial value for C equal to 1000 (the optimization of C was carried out as in [13]). In order to evaluate the convergence behavior of the empirical error minimization method, we first run the simulation using a gradient-descent method. The estimate of the error to minimize is computed on the validation data examples after each adaptation of the hyper-parameters.

Before, the SVM model is trained on the training data. Along the optimization process, we recorded the estimate of the error, the margin value, the number of support vectors and the test error. The latter is the ratio of the number of misclassified examples over the number of whole test examples. We report in Figure 3 the obtained plots. Figure 3 ⟨a⟩ shows that the algorithm minimizes the estimate of the probability of error even though some noise could be observed. For example, around the iteration 60 we remark a small increase of the error estimate. This confirms the relative sensibility of gradient descent to noisy error surface. In fact, performing a step assumes the trained model i.e. support vectors and multipliers α_i to remain unchanged along the search direction path. However, large gradient values through the error surface invalids this hypothesis so a downhill descent will not be ensured. The test error plot of Figure 3 ⟨b⟩ consolidates the adopted objective function. Notice the existing correspondence between the objective minimums and the test error ones. Notice also the existing noise on the test error curve. Despite this relative instability, the error is minimized after 100 iterations of the optimization procedure to 0.022. In Figure 3 ⟨c⟩ we report the variation of the number of support vectors. The curve shows that



⟨a⟩ ⟨b⟩ ⟨c⟩ ⟨d⟩

Fig. 3. Optimizing hyper-parameters using a gradient descent method and KMOD kernel: ⟨a⟩ Variation of the objective function; ⟨b⟩ Variation of the test error; ⟨c⟩ Variation of support vectors number; ⟨d⟩ Variation of the margin.

the optimization criterion reduces considerably the number of support vectors. This number was reduced from 187 to 42 support vectors. This is a key feature since it is known that

$$\frac{\sharp SV}{l} \tag{14}$$

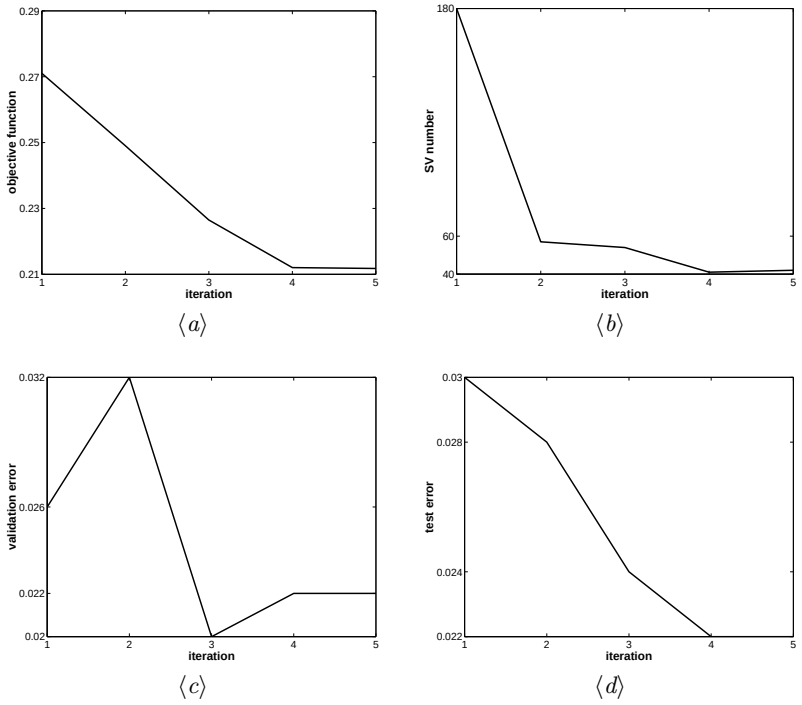
is a pessimistic bound on the true generalization error that one should minimize to ensure good performance (where $\sharp SV$ is the number of support vectors and l is the number of training data). The curve in Figure 3 ⟨d⟩ shows the variation of the margin along the optimization procedure. Notice the relative increase of its values for last iterations.

We also experimented the quasi-Newton method on the same classification task. We report in Figure 4 the corresponding plots which were obtained for an initial $C = 1000$. The curves show that the algorithm is able to reduce the error while converging in very few iterations. Indeed, the procedure converges after only 5 iterations. The plots ⟨a⟩ and ⟨d⟩ of the same figure show that the objective function is minimized as well as the test error after four iterations. The final test error is 0.022. The same value achieved by the gradient descent procedure. Notice the reduction of the support vectors number after just one iteration of the optimization procedure (Figure 4 ⟨b⟩). The algorithm stabilizes at approximately 40 support vectors. Figure 4 ⟨c⟩ shows the variation of the classification error on the validation examples. Moreover, notice that the objective to minimize never increases during the optimization.

Besides, we chose to minimize separately two more criterions. The first one is the VC dimension bound in which we approximate the radius square R^2 with $\max_{i \in supportvectors}(\Phi(x_i) - \Phi(o))^2$. The second criterion is the margin value. We report in Figures 5 and 6 the related variation of the objective function, the test error and the support vectors number for both of the criterions. While the objective functions are minimized, none of the criterions minimizes the test error. In fact, the number of support vectors is even increased from 180 to 700 approximately, which increases dramatically Vapnik's bound (given in Eq. 14). One may conclude the margin maximization is not suitable for our purpose. This is expectable since the data distribution is not takin into account through the use of a precise estimate of the radius R of the enclosing sphere. In the other hand, the approximation of the radius we considered does not reduce the error as well. This lead us to the conclusion that the VC dimension approximate is a very pessimistic upper bound of the SVM capacity h and is an ineffective criterion for the optimization. On the other hand, the empirical estimate through a validation data set lowers the error with an impressive reduction of support vectors number too. The classification duration is , thus, greatly decreased.

4 Hand-written digits recognition

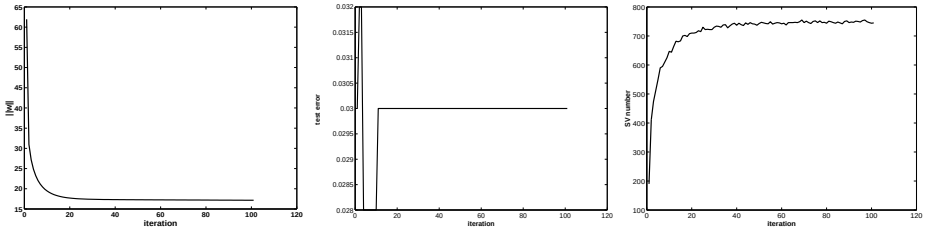
Support Vector Machine is a binary classifier which is useful for two-class data only. However, $k - class$ pattern recognition problems (where one has $k \geq 3$) such as the digit recognition task could be solved using a voting scheme method



$\langle a \rangle$ $\langle b \rangle$ $\langle c \rangle$ $\langle d \rangle$

Fig. 4. Optimizing hyper-parameters using a Quasi-Newton method and KMOD kernel: ⟨a⟩ Variation of the objective function; ⟨b⟩ Variation of support vectors number; ⟨c⟩ Variation of the validation error; ⟨d⟩ Variation of the test error;

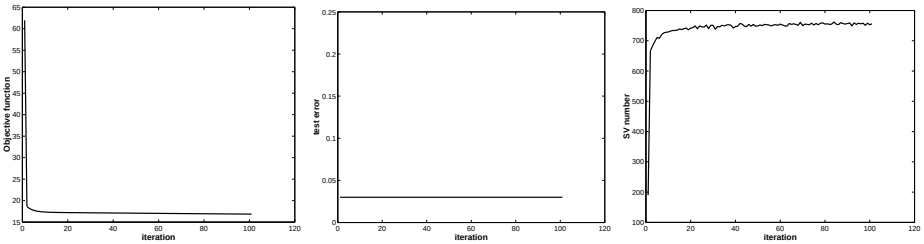
based on combining many binary decision functions. One possible approach is to consider a collection of k binary classification problems. k classifiers can then be constructed, one for each class. The i^{th} classifier constructs a hyper-plane



$\langle a \rangle$ $\langle b \rangle$ $\langle c \rangle$

Fig. 5. Minimizing $\|w\|$ using a gradient descent method and KMOD kernel: ⟨a⟩ Variation of the objective function; ⟨b⟩ Variation of the test error; ⟨c⟩ Variation of support vectors number.

between class i and the $k-1$ other classes. A majority vote across the classifiers is then applied to classify a new example. Alternatively, $\frac{k(k-1)}{2}$ hyper-planes can be constructed, separating the classes from each other and similarly an appropriate voting scheme could be used. Clearly, a digit recognition system using this strategy requires building 45 different models, one for each pair of classes. This scheme was already used to solve multi-class recognition problems with linear decision functions as in the Ho-Kashyap classifier. It is commonly referred to as "Pairwise strategy" in contrast to the well-known "One Against Others strategy" [11,3]. In order to test the optimization method we used NIST digit image database along with different kernel models, namely KMOD, RBF and polynomial kernels. We considered only the pairwise strategy of learning. For that, we proceed 45 training processes to build the entire pairs' models. During classification, an appropriate combination scheme consists of finding the class k for which all the pairs' models (k, j) with $0 \leq j \leq 9$ have a positive output. The example to classify is rejected if no class k was found. The optimization of kernel parameters is done on each pair model as the scheme already described (cf. §3.2). The result is that each SVM will have its own kernel parameter values. Thus, kernel profiles will vary dependently of the pair of classes to separate. This is a strong feature that improves obviously the resulting decision frontiers. We used a subset of 18,000 images from hsf_123 part for training, 2,000 supplementary images were used for validation and 10,000 images from hsf_7 part for testing. From each image we extract 272 features that well characterize local and morphological shapes. Those values are presented to every SVM [1]. In order to assess the method, we decided to start the optimization from the best kernel parameters that we have previously obtained empirically in [3]. The duration of training varies dependently of the used kernel, its parameter values, the trade-off parameter C and obviously the size of data. On the average, the entire training took about 70 hours on a SUN-ULRA-SPARC 500 MHz, 256Mo. We counted 2 to 9 iterations for the Quasi Newton to converge on each pair model. For all experiments, we considered an initial value for C equal to 1000. Table 3 shows



$\langle a \rangle$ $\langle b \rangle$ $\langle c \rangle$

Fig. 6. Minimizing the approximated VC dimension using a gradient descent method and KMOD kernel: ⟨a⟩ Variation of the objective function; ⟨b⟩ Variation of the test error; ⟨c⟩ Variation of support vectors number

the obtained testing rates before and after optimization was done. Notice the increase of recognition rate near 0.5% for the optimization strategy. Remark also the considerable reduction of the support vectors number for the optimized system, which shrinks the complexity of the model and ensures lower bound for the generalization error. Moreover, it is worth mentioning that KMOD does slightly better than other kernels in general. In order to assess the significance of the results we did a z-normal test between KMOD and the performance of the other kernels. This test does not take into account the variability throughout different testing sets. We assume that 10,000 testing examples are sufficient to pass over this condition. The plus sign beside the values in the table indicates whether or not the performance is significantly different from that of KMOD. Finally, recall that KMOD is significantly better than RBF and polynomial kernel of degree 3 (a statistical test at 0.95 confidence level was carried out).

5 Summary

In a multi-class classification problem, the distribution of the data can vary widely from one class to another. It is thus very important to fit the inferred decision frontiers to the classes; i.e. one must select the appropriate model for each classification task with respect to the difficulty of separating the data. This is particularly true for the case of SVM classifier that embeds hyper-parameters for which optimal values must be found. We proposed an empirical error based optimization that uses a Quasi-Newton method to adapt the parameters. We have shown experimentally that the criterion we optimize minimizes the number of support vectors. Moreover, the Quasi-Newton approach proved to converge much faster than the simple gradient descent. It also prevents any divergence of the optimization for badly conditioned Hessian matrices. We also proposed an optimization scheme applied to a multi-class task problem. For this purpose, we adopted a pairwise strategy of learning. On NIST database, the optimization improves our previously obtained recognition rate by 0.5%. Finally, it would be interesting to test the optimization method on more difficult data so one can study and compare the behavior of different kernels.

Acknowledgments: This research was supported by the NSERC of Canada and the FCAR program of the Ministry of Education of Quebec.

Kernel	**Optimized system**		**Not optimized system**	
	recog. rate	**SV**	**recog. rate**	**SV**
KMOD	98.98	198	98.56	527
polynomial (d=4)	98.81	203	98.40	513
polynomial (d=3)	+ 98.25	385	+ 97.88	677
RBF	+ 98.51	222	+ 98.03	540

Table 3. *Testing recognition rates and average of support vectors (per model) on NIST database using the Quasi-Newton method*

References

1. N.E. Ayat, M. Cheriet, and C.Y. Suen. Un système neuro-flou pour la reconnaissance de montants numériques de chèques arabes. In *CIFED*, pages 171–180, Lyon,France, Jul. 2000.
2. N.E. Ayat, M. Cheriet, and C.Y. Suen. Kmod- a new support vector machine kernel for pattern recognition. application to digit image recognition. In *ICDAR*, pages 1215–1219, Seattle,USA, Sept. 2001.
3. N.E. Ayat, M. Cheriet, and C.Y. Suen. Kmod-a two parameter svm kernel for pattern recognition. to appear in icpr 2002. quebec city, canada, 2002, 2002.
4. Y. Bengio. Gradient-based optimization of hyper-parameters. *Neural Computation*, 12(8):1889–1900, 2000.
5. B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Fifth Annual Workshop on Computational Learning Theory*, Pittsburg, 1992.
6. C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
7. Courant and R. Hilbert. *Methods of Mathematical Physics*. Interscience, 1953.
8. G.Wahba. The bias-variance trade-off and the randomized gacv. *Advances in Neural Information Processing Systems*, 11(5), November 1999.
9. T. Joachims. Making large-scale svm learning practical. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11. 1999.
10. J.Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3), October 1999.
11. U. Kreβel. Pairwise classification and support vector machines. In B. Schlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter Chap.15, pages 255–268. 1999.
12. J. Larsen, C. Svarer, L.N. Andersen, and L.K. Hansen. Adaptive regularization in neural network modeling. In *Neural Networks: Tricks of the Trade*, pages 113–132, 1996.
13. O.Chapelle and V.Vapnik. Choosing multiple parameters for support vector machines. *Advances in Neural Information Processing Systems*, 03(5), March 2001.
14. W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C, the art of scientific computing*. Cambridge University Press, second edition, 1992.
15. G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for adaboost. *Machine Learning*, 43(3):287–320, 2001.
16. B. Scholkopf. *Support vector learning*. PhD thesis, Universität Berlin, Berlin, Germany, 1997.
17. B. Scholkopf, C. Burges, and A. Smola. Introduction to support vector learning. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 1. 1999.
18. P. Sollich. Bayesian methods for support vector machines: Evidence and predictive class probabilities. *Machine Learning*, 46(1/3):21, 2002.
19. V. Vapnik. *The Nature of Statistical Learning Theory*. NY, USA, 1995.
20. V. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5), September 1999.
21. D.S. Watkins. *Fundamentals of Matrix Computations*. Wiley, New York, 1991.