Empirical Error based Optimization of SVM Kernels: Application to Digit Image Recognition.

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

[1] LIVIA, École de Technologie Supérieure, 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 modeling, 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 optimization method. To assess our method, the approach is further used for adapting KMOD, RBF and polynomial kernels on synthetic data and NIST database. The method shows a much faster convergence with satisfactory results in comparison with the simple gradient descent method.

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 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 where the concern 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 [7] 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 points 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. This represents a serious handicap to the application of the aforementioned scheme on relatively large digit image databases. Instead, an alternative method that we investigate consists on 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. As well, one more drawback in [7], is that the gradient descent method needs 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, this method will throw away a lot of time before multiple SVMs are optimized for a 10-class problem. In our method we make use of a Quasi-Newton variant to adapt the hyper-parameters. The optimization algorithm proceeds in two steps: computing the search direction and computing the step along this direction. Another part of the method consists of using probabilistic outputs for the SVM by fitting a two-parameter sigmoid to the unbiased output of the classifier [5]. This provides a posterior probability that is used to estimate the probability of error on the validation examples. Through the use of such a sigmoid, the true error on the validation set is

smoothed, so we can compute its gradient for the optimization. The experimental results we got on a toy classification problem show a faster convergence for our proposed method. As well, we find out that the presented scheme minimizes a pessimistic bound on the generalization error. In section 2, a detailed description of the optimization method is given. The posterior probability mapping algorithm is described as well. In section 2.3, we give an experimental study on a two-class synthetic data. In section 3, a real-life ten-class problem is further considered through NIST digit image database. Finally, we summarize the work in section 4.

2 Model selection Algorithm

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 $\underline{\theta}$ as $f_{\underline{\alpha},b,\underline{\theta}} = \sum_i \alpha_i y_i K_{\underline{\theta}}(x, x_i) + b$; where the parameters α_i are found by maximizing a quadratic function (maximum margin algorithm) [10] and y_i represents the target of support vectors x_i. 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. However, leave-one-out error estimation is time consuming and may be avoided if enough data could be retained for the validation set. Th larger the validation set is, the smaller is the variance of the estimated error. This estimate is given by

Kernels	**Formula**
linear	$k(x, y) = x.y$
sigmoid	$k(x, y) = \tanh(ax.y + b)$
polynomial	$k(x, y) = (1 + x.y)^d$
KMOD	$k(x, y) = a[\exp(\frac{\gamma}{\|x-y\|^2+\sigma^2}) - 1]$
RBF	$k(x, y) = \exp(-a\|x - y\|^2)$
exponential RBF	$k(x, y) = \exp(-a\|x - y\|)$

Table 1. *Common kernels*

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

where Ψ is the step function and N being the size of the data set.

Using a gradient descent approach assumes the error estimate in Eq. 1 to be differentiable. Unfortunately, the step function is not. To circumvent this drawback it is possible to use a contracting function of the form $\Psi(x) = \frac{1}{1+\exp(-Ax+B)}$ [5, 7]. A very nice way to choose the values of constants A and B is to estimate posterior probabilities [5]; whereby a smooth approximation of the test error is obtained. Next, we describe the method.

2.1 Probability estimation in SVM

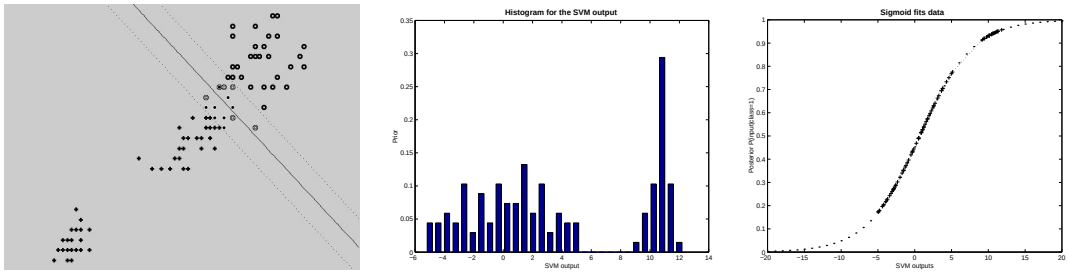
Recently, many researchers have considered the problem of probabilities estimation for SVM classifiers. The methods they proposed are of varying levels of complexity. Sollich for example, proposed in [9] a bayesian framework to tackle two of the outstanding challenges in SVM classification: how to obtain 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 [4] 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 [5] used a slightly modified logistic function given by

$$P(y = 1|f) = \frac{1}{1 + \exp(Af + B)}. \quad (2)$$

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 $t_i = \frac{y_i+1}{2}$ is the binary target coding for the data point (x_i, y_i). During the experiments, we tried out the above mentioned algorithm using a Newton method to adapt the sigmoid in Eq. 2. Figure 1 shows respectively the separation frontier, the distribution of the SVM output values and the inferred contracting function for a linear SVM trained on iris3v12 data set [3]. Below, we shall integrate this two-parameter identification procedure into the overall optimization process.



⟨a⟩ ⟨b⟩ ⟨c⟩

Figure 1. ⟨a⟩ Separation frontier; ⟨b⟩ Distribution of the SVM output values; ⟨c⟩ Posterior probability versus SVM output.

2.2 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 [6]. This procedure is time consuming and needs many training process before a feasible solution could be found. Furthermore, 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. 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.

Let us consider now, a target coding scheme for which $t_i = 1$ if the input vector x_i belongs to class C_1 and $t_i = 0$ if it belongs to class C_2. We can formulate the error probability of observing either target value for a given data example x_i as

$$E_i = P(y_i \neq z_i) = p_i^{1-t_i}(1-p_i)^{t_i} \tag{3}$$

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 could be written as:

$$E = \frac{1}{N}\sum_{i=1}^{N} E_i = \frac{1}{N}\sum_{i=1}^{N} p_i^{1-t_i}(1-p_i)^{t_i} \tag{4}$$

To compute the gradient of the error we shall assume the current vector of kernel parameters to optimize is near a local minimum so the first derivative of the error w.r.t $\underline{\theta}$ at the minima vanishes. It follows that we can approximate the gradient of the error as

$$\frac{\partial E}{\partial \underline{\theta}} = \frac{\partial E}{\partial \underline{\alpha}}\frac{\partial \underline{\alpha}}{\partial \underline{\theta}} \tag{5}$$

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

$$\frac{\partial E}{\partial \alpha_j} = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial E_i}{\partial p_i}\frac{\partial p_i}{\partial f_i}\frac{\partial f_i}{\partial \alpha_j} \tag{6}$$

where

$$\frac{\partial E_i}{\partial p_i} = -p_i^{1-t_i}t_i(1-p_i)^{t_i-1} + (1-t_i)(1-p_i)^{t_i}p_i^{-t_i},$$

$\frac{\partial p_i}{\partial f_i} = -Ap_i^2\exp(Af_i + B)$ and $\frac{\partial f_i}{\partial \alpha_j} = y_i K_\theta(x_j, x_i)$; $y_i = \pm 1$ being the bipolar target of example x_i. Once the error derivative w.r.t. multipliers vector $\underline{\alpha}$ is computed, the next step consists of estimating the derivative of $\underline{\alpha}$ w.r.t. $\underline{\theta}$. Notice that we may include the SVM bias b in the vector $\underline{\alpha}$ as $\underline{\alpha} = (\alpha_1, ..., \alpha_k, b)$. It is shown that

$$\frac{\partial \underline{\alpha}}{\partial \theta} = -H^{-1}\frac{\partial H}{\partial \theta}\underline{\alpha}^T \tag{7}$$

where $H = \begin{pmatrix} K^Y & Y \\ Y^T & 0 \end{pmatrix}$ and the components $K_{ij}^Y = y_i y_j K(x_i, x_j)$ [7] . 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'. We give herein 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 sigmoid parameters A and B (Newton optimization procedure).
4. Estimate the probability of error on the validation set.
5. Calculate the gradient of that error $\frac{\partial E(\underline{\alpha},\underline{\theta})}{\partial \underline{\theta}}$.
6. Calculate the hessian H' over the kernel's parameter space .
7. Update $\underline{\theta}$ using: $\Delta\underline{\theta} = -\lambda H'.\frac{\partial E(\underline{\alpha},\underline{\theta})}{\partial \underline{\theta}}$.

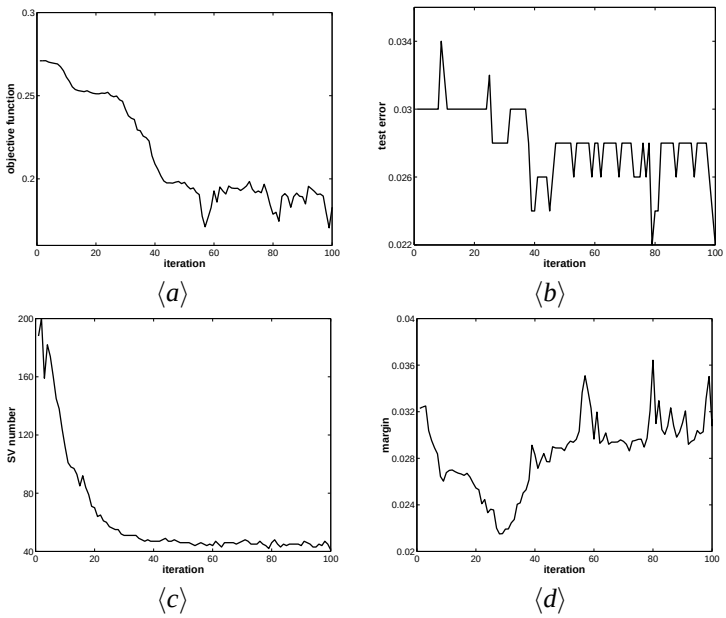
where λ is the amplitude of the step along the search direction and H' is a $n \times n$ matrix; n being the dimension of vector $\underline{\theta}$. The used line search minimization algorithm is a variant of the Golden Section Search described in [8].

2.3 Experiments

For the experiments we considered first a synthetic two-class problem. So we produced 2000 data points, from two overlapping spherical gaussians with a known Bayes error of 0.013. The data points have two attributes and the classes are balanced. We retain 1000 examples for the training set, 500 examples for the validation set and 500 examples for the testing set. 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. In order to evaluate the convergence behavior of the validation error minimization scheme, we first run the simulation using a gradient-descent method. We report in Figure 2 the obtained plots. Figure 2 $\langle$a$\rangle$ shows that the algorithm efficiently minimizes the estimate of the probability of error even though some noise could be observed which in time vanishes. The test error plot of Figure 2 $\langle$b$\rangle$ consolidates our minimization principle in that objective function minima correspond to those of the test error. Figure 2 $\langle$c$\rangle$ shows that our minimization principle minimizes the number of support vectors. This is a key feature cause it is also known that $\frac{\sharp sv}{l}$ 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 examples). We also test the quasi-Newton scheme on the same synthetic data. The plots of Figure 3 were obtained for an initial C=1000. The plots show that the algorithm is capable of reducing the error while converging in few iterations. In our case, the algorithm converges after 5 iterations. The plots $\langle$a$\rangle$ and $\langle$d$\rangle$ of Figure 3 show that the objective function is minimized as well as the test error after only four iterations. One advantage of the presented scheme is that the objective to minimize never increases; which is not always the case of the gradient descent procedure if the search direction does not go downhill w.r.t. error surface.

3 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 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 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". 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. §2.2). 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. 272 features that well characterize local and morphological shapes are extracted from the images and fed to every SVM model [1]. In order to evaluate our method we decided to start optimization from the best kernel parameters that we already obtained empirically [2]. The duration



⟨*a*⟩ ⟨*b*⟩ ⟨*c*⟩ ⟨*d*⟩

Figure 2. Gradient descent method: ⟨a⟩ Variation of objective function; ⟨b⟩ Variation of test error; ⟨c⟩ Variation of support vectors number; ⟨d⟩ Variation of the margin.

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

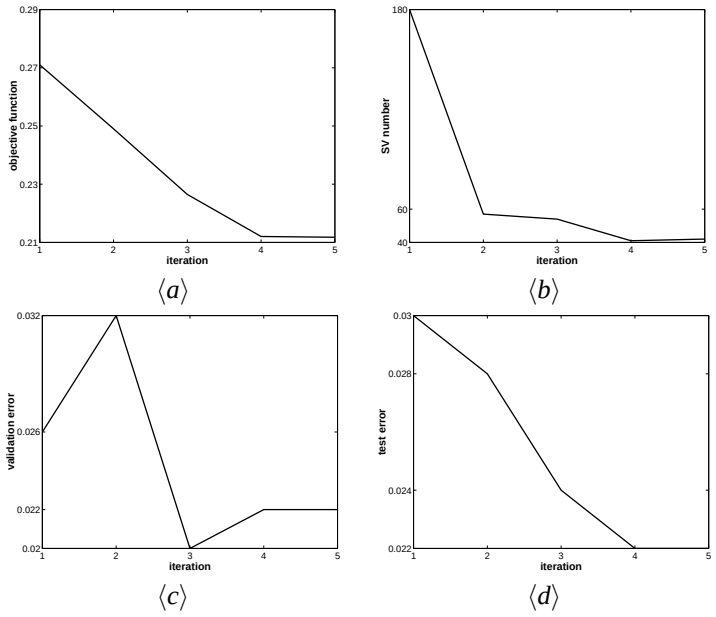
Table 2. *Testing recognition rates and average of support vectors (per model) on NIST database using the Quasi-Newton method*

of training varies dependently of the used kernel, its parameter values, the trade-off 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 a starting value for C equal to 1000. Table 2 shows 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 reduction of the number of support vectors 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 signs 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 at a confidence level of 95%.

4 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. we must select the appropriate model for each classification task with respect to the difficulty to separate the data. This is particularly true for the case of SVM classifier that embeds hyper-parameters for which optimal values must be found.

We propose an empirical error based optimization that uses



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

Figure 3. Quasi Newton method: $\langle$a$\rangle$ Variation of objective function; $\langle$b$\rangle$ Variation of support vectors number; $\langle$c$\rangle$ Variation of validation error; $\langle$d$\rangle$ Variation of test error;

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 convergence faster than the simple gradient descent. It also prevents any divergence of the optimization for badly conditioned Hessian matrix. On the NIST database, the method also improves our already obtained performances for the classification of digit images. A gain of 0.5% was obtained.

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

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 two parameter svm kernel for pattern recognition. In *ICPR*, Quebec city, Canada., 2002.

[3] C.L. Blake and C.J. Merz. UCI repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/mlrepository.html, 1998.

[4] G.Wahba. The bias-variance trade-off and the randomized gacv. *Advances in Neural Information Processing Systems*, 11(5), November 1999.

[5] J.Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3), October 1999.

[6] 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.

[7] O.Chapelle and V.Vapnik. Choosing multiple parameters for support vector machines. *Advances in Neural Information Processing Systems*, 03(5), March 2001.

[8] 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.

[9] P. Sollich. Bayesian methods for support vector machines: Evidence and predictive class probabilities. *Machine Learning*, 46(1/3):21, 2002.

[10] V. Vapnik. *The Nature of Statistical Learning Theory*. NY, USA, 1995.