IMPERIAL COLLEGE LONDON

E4.34
SO17
ISE4.29

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2004

MSc and EEE/ISE PART IV: M.Eng. and ACGI

ANALYSIS OF NEURAL NETWORK MODELS This course is assessed by coursework and a short oral exam.

Wednesday, 19 May 10:30 am

Time allowed: 0:15 hours

There are ZERO questions on this paper. A description follows.

Answer ZERO questions.

Any special instructions for invigilators and information for candidates are on page 1.

Examiners responsible First Marker(s) : P. De Wilde
Second Marker(s) : E.H. Mamdani

Neural Network Models Exam Coursework 2003-2004 Deadline April 26th 2004 at 10.00 am *

This exam is based on the paper *Pruning Error Minimization in Least Squares Support Vector Machines*, Bas de Kruif and Theo de Vries, IEEE Transactions on Neural Networks, Vol. 14, No. 3, May 2003, p. 696–702. It is available from the library electronic journal website. The paper is about function approximation (regression) based on a set of data points. Some data points are deleted, this is called pruning in the paper.

Read section I. Note that this paper will use support vector machines for function approximation only, not for data classification. Do not look up any references. (The challenge at the end of this exam is the only point where you will need to look up a further paper.) Look at figure 1. It explains the procedure well, but the terminology may be unfamiliar to you. By "optimize" is meant the fitting of a function to the data points. The box "Error norm violated?" means that the loop will stop if the approximation is sufficiently good. The pruning is the removal of data points. One pass through the loop is called an iteration.

Read section II A. Do not worry if you don't understand what a radial basis function or a spline is. Equations (3) and (4) define how the function is fitted to the data. Note the term $\frac{1}{2}w^T w$ added to the error, and the multiplier $\gamma \geq 0$ of the sum of squares. This technique is called regularization. Explain in words what the purpose of regularization is (1/20). Backpropagation is another technique that can be used for classification as well as for function fitting. Write down equations similar to (1)-(4) for a backpropagation network (2/20).

The constrained optimization is solved using Lagrange multipliers. Deriving equations (6) will make you more familiar with the notation (no marks for this). Write out the matrix equation (7) for N=3, without using matrices (no marks).

*Post it in the appropriate locked box in the undergraduate office, level 6.

The application of the kernel to the data points x_k and x_l is denoted $K(x_k, x_l)$. We will use two kernels in this coursework exam. One is the radial basis function kernel

$$K_r(x_k, x_l) = \exp\left(-\frac{1}{2\sigma^2}(x_k - x_l)^2\right),$$

the other is the sigmoid kernel

$$K_s(x_k, x_l) = \tanh(0.1 x_k x_l + 1).$$

You can see from equation (8) that all you need to know for approximating the data with a function are the Lagrange multipliers α_k and the bias b from (7), and the functional form of the kernel. Think about how you will program this. To solve the linear system (7), you will use a linear equation solver.

Now read section II B about the combination of pruning and regularization. It is claimed at the start of the section that γ trades off small approximation errors for a smooth function. Explain (1/20). The authors are going to compare two ways of pruning: omit a point and regularize ($\gamma > 0$), or regularize, and then omit the point by minimizing (3) with $\gamma = 0$. Make sure you understand the correspondence between pruning a data point, and omitting (i.e. making equal to zero) the corresponding Lagrange multiplier. The main part of your exam is to compare these two procedures, for two different kernels.

Generate one set of 15 training samples $\{x_k, y_k\}$, where $0 < x_k < 1$. Draw the samples from the sinc function, with noise added. Make one sample point dependent on your CID number, so that your simulations are different from those of your friends. (The least significant digits of your CID number differ more between students than the more significant digits.) Choose the radial basis function kernel K_r to start with. Decide on a value for σ. Finding the best value for σ will not get you extra marks. For these training samples, and this kernel, compare the two methods from the paper: regularization and pruning, and regularization followed by pruning with $\gamma = 0$. Prune one or two points. Show and discuss your simulations (5/20). Now do the same for the sigmoid kernel K_s (5/20). The first method in the paper depends on the choice of *gamma*. Try a few different values. You do not have to justify your choice for σ, but you do have comment on your choice of γ.

Ignore the sentence "the distribution of the input samples determines the amount of information a specific sample contains" in section II B. You will be better able to comment on the results of your simulations if you also read section III of the paper. The iterations mentioned are the successive pruning steps from figure 1. Do not derive any equations, such as (14). Below (18),

the authors state that it is best to remove the sample with the smallest $\frac{\alpha_j}{[A^{-1}]_{jj}}$. What do you think of this (1/20)? The appendices in the paper are not relevant for you.

The Challenge. If you want to get top marks, you have to do this challenge. However, you will get better marks for a good report without the challenge than for a mediocre report with the challenge solved.

Look up Mercer's original paper. The correct citation for the journal is Philosophical Transactions of the Royal Society of London, Series A, Vol. 209 (1909), p. 415–446. It is available from the library electronic journal web site. Find conditions so that the kernel $\kappa(s,t) = \tanh(p\,st + q)$ is of positive type (5/20). In this kernel, p and q are constants. It is not necessary to read the whole paper.

There is no simple condition on p and q alone that guarantees the kernel to be of positive type, the condition will involve s and t as well. Try to find the strongest necessary conditions you can. You will get marks for the way you tackle this problem as well as for the results. Remember that $\int_a^b \kappa(s,t)\delta(s-a_0)ds = \kappa(a_0,t)$ for $a < a_0 < b$, with δ the Dirac delta function.

You could organize your work as follows.

day 1 Read the paper, looking up anything you don't understand in your lecture notes. Answer the questions that can be answered without programming. Plan what you are going to program. Find the Matlab routines for generating random numbers and solving linear systems. Other packages also contain such routines e.g. Numerical Recipes in C and GNU Scientific Library (GSL).

day 2 Do the programming, and debug your program.

day 3 Run the simulations, and collect the results in a form that you can present in your report.

day 4 Write the report. It should be maximum six pages (single sided) A4, in a font not smaller than 10 point. You will not get marks for anything exceeding six pages, even if it is appendices. Font size in tables and figures should be at least 10 point, or the tables and figures will not be marked. Describe the problem, and how you have solved it. Describe your simulations, but do not give programme listings. Do not give references to the literature. Make sure you do and answer everything that is asked for in the coursework. Do not bind the report, but staple the pages together. Mention your name, and indicate for what degree (e.g. MEng Elec. Eng., MEng ISE, MSc) you are studying.

day 5 Check the consistency and quality of your work. Make last minute changes if necessary. If you feel confident and have the time, tackle the challenge. Resist the temptation to spend more than five 8-hour days of intensive effort on your coursework. You will not be compensated for it in marks. Just as an exam paper requires a concentrated effort over a few hours, this coursework requires a concentrated effort over a few days.

Do not forget to attend on the "exam" day. This day will be advertised in your exam schedule. Bring a copy of your report with you, and your college security card. I will ask you one or two questions based on what you have written in your report, to make sure that you have written it yourself. No preparation is necessary.

Good luck.

Dr. P. De Wilde

Pruning Error Minimization in Least Squares Support Vector Machines

Bas J. de Kruif and Theo J. A. de Vries

Abstract—**The support vector machine (SVM) is a method for classification and for function approximation. This method commonly makes use of an ϵ-insensitive cost function, meaning that errors smaller than ϵ remain unpunished. As an alternative, a least squares support vector machine (LSSVM) uses a quadratic cost function. When the LSSVM method is used for function approximation, a nonsparse solution is obtained. The sparseness is imposed by pruning, i.e., recursively solving the approximation problem and subsequently omitting data that has a small error in the previous pass. However, omitting data with a small approximation error in the previous pass does not reliably predict what the error will be after the sample has been omitted. In this paper, a procedure is introduced that selects from a data set the training sample that will introduce the smallest approximation error when it will be omitted. It is shown that this pruning scheme outperforms the standard one.**

Index Terms—**Function approximation, pruning, regression, support vector machine (SVM).**

I. Introduction

THE SUPPORT vector machine (SVM) has been introduced by Vapnik [1] as a method for classification and for function approximation. In this paper, we will be concerned with function approximation only. The SVM makes it possible to deal with high-dimensional input spaces, because it is not liable to the curse of dimensionality [2]; the parameterization of the approximator depends on the complexity of the function only. The SVM is typically based on an ϵ-insensitive cost function, meaning that approximation errors smaller than ϵ will not increase the cost function value. This results in a quadratic convex optimization problem. Due to the inequality constraints contained in this method, the solution that is obtained is sparse.

Instead of using an ϵ-insensitive cost function, a quadratic cost function can be used. This approach results in so-called least squares support vector machines (LSSVMs), which were introduced by Suykens [3] and are closely related to regularization networks [4]. With the quadratic cost function, the optimization problem reduces to finding the solution of a set of linear equations. This is computationally attractive, however, the obtained solution is not sparse. Sparseness is imposed by *pruning*, i.e., recursively solving the approximation problem and subsequently omitting data that has a small error in the previous pass. See Fig. 1.

This two-step approach of LSSVM gives the user control over the approximation process, as it is clear what error is introduced by imposing sparseness. This makes the appraisal between the number of support vectors and the pruning error explicit. To have a clear measure on how the approximation error on the training set is influenced by the pruning, the method used in [5] in which an SVM is used to select a sparse set of support vectors to approximate the function, is not used.

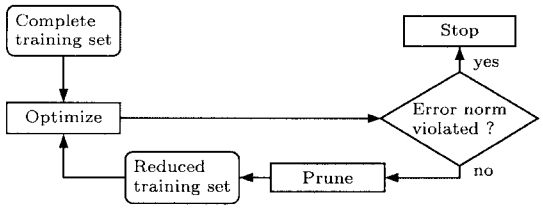

Fig. 1. Obtaining a sparse solution.

The selection of data to be omitted during pruning is one of the determining factors of the function approximation process. The standard procedure for this in LSSVM, omitting the sample with the smallest approximation error in the previous pass, seems sensible, as LSSVM has shown to work well [3]. However, the choice for this selection procedure only accounts for the absolute error and does not incorporate the location of the samples. In this paper it will be shown to be suboptimal. In addition, an alternative procedure will be proposed that selects from a data set the sample that will introduce the smallest approximation error when it is omitted in the next pass of the approximation. An example illustrates the differences between these methods.

This paper is organized as follows. In Section II, function approximation by means of LSSVM is reviewed and the suboptimality of the standard pruning scheme is illustrated. An alternative pruning procedure is proposed in Section III. This procedure is tested on an example function and compared to the standard scheme in Section IV. The conclusion is given in Section V.

II. LSSVM for Function Approximation

This section summarizes known theory concerning LSSVM for function approximation and is based on [1], [3]. First, the general function approximation problem is outlined. Next, regularization and pruning are treated.

A. Function Approximation

Consider a given set of training samples $\{x_k, y_k\}_{k=1\ldots N}$, in which x_k is the input vector and y_k is the corresponding target value for sample k. The goal of function approximation is to find the underlying relation between the input and the target value. Once this relation is found, the outputs for inputs that are not contained in the training set can be approximated.

Manuscript received January 21, 2001; revised December 11, 2002. This work was supported by Tecnotion B.V., Almelo, The Netherlands and by Imotec B.V, Hengelo, The Netherlands.

The authors are with the Drebbel Institute for Mechatronics, University of Twente, NL-7500 AE Enschede, The Netherlands (e-mail: b.j.dekruif@utwente.nl; t.j.a.devries@utwente.nl).

Digital Object Identifier 10.1109/TNN.2003.810597

With a SVM, the relation underlying the data set is represented as a function of the following form:

$$\hat{y}(x) = w^T \phi(x) + b. \tag{1}$$

In here, ϕ is a mapping of the vector x to some (probably high-dimensional) feature space, b is the bias and w is a weight vector of the same dimension as the feature space. The mapping $\phi(x)$ is commonly nonlinear and makes it possible to approximate nonlinear functions. Mappings that are often used result in an approximation by a radial basis function, by polynomial functions, or by splines [5], [6].

The approximation error for sample k is defined as follows:

$$e_k = y_k - \hat{y}(x_k) \tag{2}$$

and for the given data we search for those weights that give the smallest summed quadratic error of the training samples in case of LSSVM. Because this can easily lead to overfitting, ridge regression (a form of regularization) is used to smoothen the approximation. The minimization of the error together with the regularization is given as

$$\min_{w,b} \mathcal{I}(w, e) = \frac{1}{2} w^T w + \gamma \frac{1}{2} \sum_{k=0}^{N} e_k^2 \tag{3}$$

with equality constraint

$$y_k = w^T \phi(x_k) + b + e_k. \tag{4}$$

Here γ is the regularization parameter.

This problem can be solved using optimization theory [7]. Instead of minimizing the primary objective [(3)], a dual objective, the so-called Lagrangian, can be formed of which the saddle point is the optimum. The Lagrangian for this problem is given as

$$\mathcal{L}(w, b, e; \alpha) = \mathcal{I}(w, e) - \sum_{k=0}^{N} \alpha_k (w^T \phi(x_k) + b + e_k - y_k). \tag{5}$$

In this equation, the α_k's are called the Lagrangian multipliers. The saddle point can be found by setting the derivatives equal to zero

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w} &= 0 \rightarrow w = \sum_{k=1}^{N} \alpha_k \phi(x_k) \\
\frac{\partial \mathcal{L}}{\partial b} &= 0 \rightarrow \sum_{k=1}^{N} \alpha_k = 0 \\
\frac{\partial \mathcal{L}}{\partial e_k} &= 0 \rightarrow \alpha_k = \gamma e_k \\
\frac{\partial \mathcal{L}}{\partial \alpha_k} &= 0 \rightarrow w^T \phi(x_k) + b + e_k - y_k = 0.
\end{aligned} \tag{6}$$

Elimination of e_k and w through substitution results in the following set of linear equations:

$$\left[\begin{array}{c|c} 0 & \vec{1}^T \\ \hline \vec{1} & \Omega + \gamma^{-1} I \end{array}\right] \left[\begin{array}{c} b \\ \hline \alpha \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline y \end{array}\right]. \tag{7}$$

In this equation, $\vec{1}$ is a column vector filled with ones, α is the vector with the multipliers and y is a vector with the target values. The elements of matrix Ω equal $\Omega_{kl} = \langle \phi(x_k), \phi(x_l) \rangle = K(x_k, x_l)$. The innerproduct is defined as $\langle \phi(x_k), \phi(x_l) \rangle = \phi(x_k)^T \phi(x_l)$. To calculate the elements in this matrix, the mapping $\phi(x)$ from input space to feature space does not have to be made explicitly; it can be calculated in the input space. The matrix Ω is symmetric positive definite, because otherwise it would not fulfill Mercers conditions on innerproducts in an arbitrary space [8] if different input samples are used. The solution of this set of equations results in a vector of Lagrangian multipliers α and a bias b.

The output of the approximator can be calculated for new input values of x with α and b. The output is given as

$$\begin{aligned}
\hat{y}(x) &= \langle w, \phi(x) \rangle + b \\
&= \left\langle \sum_{k=1}^{N} \alpha_k \phi(x_k), \phi(x) \right\rangle + b \\
&= \sum_{k=1}^{N} \alpha_k \langle \phi(x_k), \phi(x) \rangle + b \\
&= \sum_{k=1}^{N} \alpha_k K(x_k, x) + b.
\end{aligned} \tag{8}$$

B. Pruning and Regularization

In (3) a parameter γ is present that trades off small approximation errors versus a smooth function. This is a form of regularization that is known as ridge regression [9]. The goal of regularization is to stabilize the final approximation by means of some nonnegative function that embeds prior information about the solution [10]. Information that is commonly assumed, is the smoothness of the function. This assumption will smoothen the output of the network and thereby make the solution less sensitive to the current realization of the noise. This will in general increase the generalization ability.

Next to using a regularization to increase the generalization ability, pruning also commonly improves the generalization [11]. Pruning is the omission of free parameters in a network. An overview of pruning techniques is given in [12]. Pruning is necessary if LSSVM is used, because in contrast with standard SVMs as proposed by Vapnik [1], which are based on an ϵ-insensitivity cost function, the α's that appear in LSSVM are not sparse. This implies that all the training points in the data set with their Lagrangian multiplier are needed to calculate the output of a new input, which is clearly unattractive. Therefore, sparseness is imposed by pruning.

Two schemes for combining pruning and regularization are given in Fig. 2 by the dashed lines. The first line represents the combined pruning and regularization scheme. In this scheme, a parameter is omitted and the resulting weights are recalculated with the regularization. In the case of LSSVM it means that the γ has a nonzero value while pruning. The second line in Fig. 2 expresses the scheme were there will be first regularization and afterwards pruning. In this scheme, the regularized data is assumed noiseless and the goal of the pruning is solely to downsize the number of parameters. In the case of LSSVM, it means that the γ is set to zero after the regularization.

The advantage of the first scheme is that there is more design freedom. During the pruning, different regularization method can be applied. How this design freedom can be used to increase performance is difficult because the combination of the regularization and the pruning determines the final result. The stopping

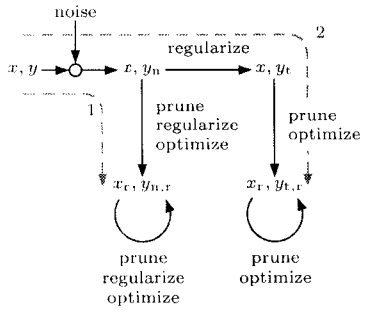

Fig. 2. Possible ways to combine pruning and regularization.

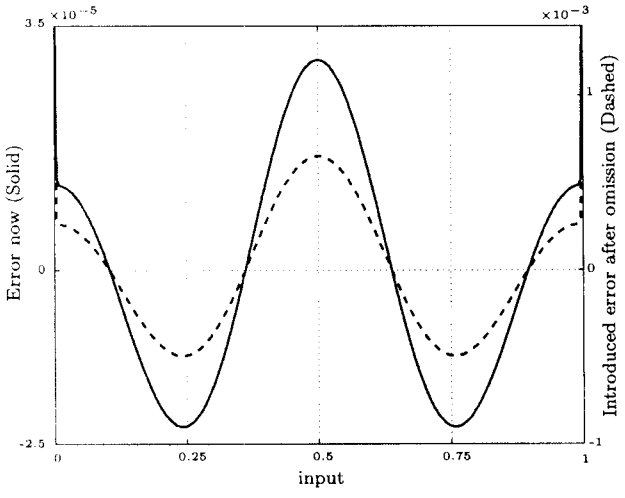

Fig. 3. Present error and the error it will introduce after it is omitted. The input samples are evenly distributed on the input space. Note the difference in scales.

of this method can be tested on an evaluation set. If the error on this set is increasing, the pruning should be stopped.

The advantage of the second scheme is that the approximation error that is introduced by pruning on the regularized data is clear. The user can specify an approximation error that is allowed between the pruned approximation and the full regularized approximation and the pruning can continue until this approximation error norm is violated. Another stopping criterion can be that the pruning will continue until the error on a separate data set, the evaluation set, starts increasing. The regularized solution can make use of complex regularization functions to smoothen the data.

Henceforward the second pruning scheme will be used, because this scheme clearly relates the error introduced due to the pruning.

The intuitive motivation for pruning the sample with the smallest absolute approximation error seems to be that this sample appears to have the smallest information content. However, this is only half the story, as also the distribution of the input samples determines the amount of information a specific sample contains. The influence of the distribution on the value of the present approximation error and the error after the omission of the sample will be illustrated by an example. The function that we want to approximate is a (noiseless) sinc-function. In Fig. 3, the present approximation error and

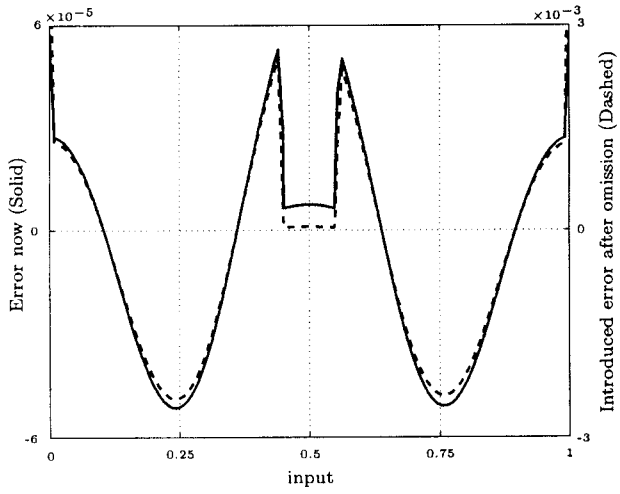

Fig. 4. Present error and the error it will introduce after it is omitted. The density of samples is higher around 0.5. Note the difference in scales.

the introduced approximation error are given if the support vector is excluded which is located at the input in the case that the samples are evenly distributed. So, the approximation error at $x = 0.5$ would be about $0.8 \cdot 10^{-3}$ if the support vector at $x = 0.5$ is omitted, while this approximation error was ca. $3 \cdot 10^{-5}$ before it was omitted. In this case the introduced error is proportional to the present error. However, if the data is not distributed evenly throughout the input space, the introduced error is not proportional to the present error, as is illustrated in Fig. 4. In this figure the number of samples is increased around 0.5. This shows that the difference in error due the omission of the support vector can not be determined solely on basis of the magnitude of the present error.

By selecting the training sample that will *introduce* the smallest error *after* omitting, the increase of the approximation error due to pruning will be minimal. The remainder of this paper is devoted to a procedure that accomplishes this.

III. Minimal Introduced Error

The approximation error of LSSVM is minimized if one selects for pruning that training point that will introduce the smallest additional approximation error *after* being omitted from the data set for the next iteration. The estimation of the output at iteration m of training sample j can be calculated using (8)

$$\hat{y}^m(x_j) = \sum_{\substack{k=0 \\ k \neq j}}^{N} \alpha_k^m K(x_k, x_j) + \alpha_j^m K(x_j, x_j) + b^m. \quad (9)$$

The multiplier for sample k at iteration m is denoted as α_k^m, while b^m denotes the bias at iteration m. If sample j is removed from the training set, the output of its input x_j in iteration $m+1$ is given as

$$\hat{y}^{m+1}(x_j) = \sum_{\substack{k=0 \\ k \neq j}}^{N} \alpha_k^{m+1} K(x_k, x_j) + b^{m+1}. \quad (10)$$

Subtracting these outputs gives the introduced error at sample j when sample j is omitted, as follows:

$$d(x_j) = \sum_{\substack{k=0 \\ k \neq j}}^{N} \left((\alpha_k^m - \alpha_k^{m+1}) K(x_k, x_j) \right) + \alpha_j^m K(x_j, x_j) + b^m - b^{m+1}. \quad (11)$$

This equation shows clearly that the introduced error does not just depend on α_j^m, but more factors determine the introduced error.

The omitting of a training point is equal to setting its Lagrangian multiplier to zero, because it will no longer have any influence on the output of the approximator.

In (3) a regularization parameter γ is present to weight the importance of the accurate approximation and the smoothness of the function. If this γ is set to zero, the error of the approximation does not influence the criterion, so only the weight is minimized. This will result in a weight of zero. Hence, the value of the Lagrangian multiplier of a selective training point can be made zero by applying regularization with $\gamma = 0$ only on that sample.

The regularization of only one element of the solution can be done by adding a regularization parameter on the corresponding diagonal term. Applying this for element j gives the following set of linear equations:

$$\left[\begin{array}{c|ccc} 0 & \vec{1}^T & 1 & \vec{1}^T \\ \hline \vec{1} & \Omega_{1,1} & \omega_1 & \Omega_{1,2} \\ 1 & \omega_1^T & \kappa + \lambda & \omega_2^T \\ \vec{1} & \Omega_{2,1} & \omega_2 & \Omega_{2,2} \end{array}\right] \left[\begin{array}{c} b \\ \hline \alpha_{1\ldots j-1} \\ \alpha_j \\ \alpha_{j+1\ldots N} \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline y_{1\ldots j-1} \\ y_j \\ y_{j+1\ldots N} \end{array}\right]. \quad (12)$$

If $\lambda = 0$ this equation is the same as (7). By setting $\lambda \to \infty$ the value of α_j is forced to become zero.

This set of equations is of the form $Ax = c$ in which A is the matrix on the left-hand side, x is the solution containing the Lagrangian multipliers and the bias and c is the vector with the targets. We want to determine the difference in the solution if λ goes from zero to infinity, which is equivalent to omitting the corresponding sample.

Starting at $\lambda = 0$

$$Ax_1 = c \qquad x_1 = A^{-1}c. \quad (13)$$

Setting $\lambda \to \infty$, A and its inverse are updated as [13]

$$A \leftarrow A + uu^T$$
$$A^{-1} \leftarrow A^{-1} - \frac{A^{-1}uu^TA^{-1}}{1 + u^TA^{-1}u}. \quad (14)$$

In this equation the vector u is defined as $u = [0, \vec{0}^T, \sqrt{\lambda}, \vec{0}^T]^T$. To find the difference in the Lagrangian multipliers, the solution of x before and after the update are subtracted from each other

$$x_2 = \left(A^{-1} - \frac{A^{-1}uu^TA^{-1}}{1 + u^TA^{-1}u} \right) c$$
$$x_1 - x_2 = \left(\frac{A^{-1}uu^TA^{-1}}{1 + u^TA^{-1}u} \right) c$$
$$\Delta x = \left(\frac{\lambda A^{-1}e_je_j^TA^{-1}}{1 + \lambda e_j^TA^{-1}e_j} \right) c. \quad (15)$$

In this equation e_j is a column vector of size $N + 1$ filled with zeros except element $j + 1$ which is equal to one. By taking the limit of Δx with $\lambda \to \infty$ we get

$$\lim_{\lambda \to \infty} \Delta x = \frac{A^{-1}e_je_j^TA^{-1}}{e_j^TA^{-1}e_j} c. \quad (16)$$

In (11) it is not only the difference in the multipliers and the bias that determine the introduced error, but a weighted sum of these differences that determine the error

$$d(x_j) = \omega_1^T \Delta\alpha_{1\ldots j-1} + \omega_2^T \Delta\alpha_{j+1\ldots N} + \kappa\alpha_j + \Delta b. \quad (17)$$

The weights in (17) are equal to the row $j + 1$ of the matrix A. Thus, by multiplying the difference of the solutions in multipliers and bias by the original matrix, the error after omitting sample j is found.

$$d(x_j) = \left[\frac{AA^{-1}e_je_j^TA^{-1}}{e_j^TA^{-1}e_j} c \right]_j = \left[\frac{e_je_j^T\alpha}{e_j^TA^{-1}e_j} \right]_j = \frac{\alpha_j}{[A^{-1}]_{jj}}. \quad (18)$$

Instead of throwing the sample out with the smallest absolute value of α_j, the sample with the smallest absolute value of α_j divided by the diagonal element j, j of the inverse of A should be thrown out to obtain the smallest introduced error.

In the case of $\gamma \neq \infty$ a similar reasoning can be performed. Instead of the unregularized matrix A of (13) the regularized matrix A_γ is used

$$A_\gamma = A + \gamma^{-1}I. \quad (19)$$

Recalculation of (13) until (18) give the introduced error

$$d(x_j) = \left[\frac{AA_\gamma^{-1}e_je_j^TA_\gamma^{-1}}{e_j^TA_\gamma^{-1}e_j} c \right]_j. \quad (20)$$

Because the regularization parameter γ is not infinite anymore, the omission of a sample also introduces an error at other samples. This directly follows from $AA_\gamma^{-1} \neq I$

The pruning rule that is found here is closely related to optimal brain surgeon (OBS) of Hassibi *et al.* [14]. The OBS methodology finds that the weight that can be omitted to be the weight that minimizes

$$\min_q \frac{w_q}{[H^{-1}]_{qq}} \quad (21)$$

in which w_q represents the weight q and H represent the Hessian of the error surface with respect to the weights. If the quadratic error increase is used, as done with OBS, the Hessian that is found in our case equals A^TA. The difference can be explained by the fact that in our case the absolute error is used.

IV. Example

The case that will be considered is the learning of the nonlinear state dependent effects of a linear motor. These effects, friction and cogging, act on the input of the linear part of the plant and can be measured considerable well. The motor has

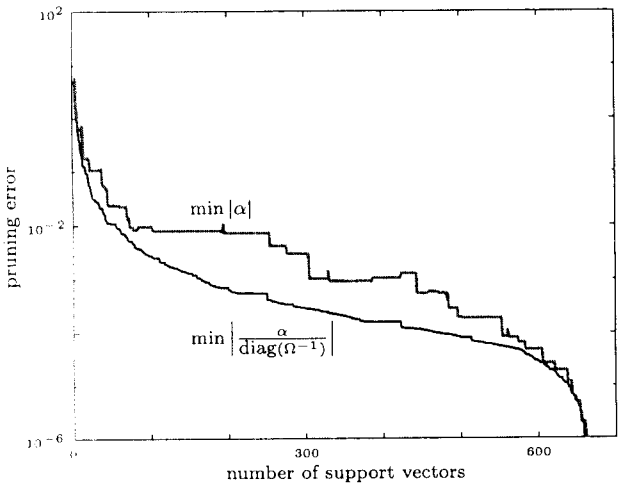

Fig. 5. Sum squared error introduced due to the pruning of both methods.

performed several movements and the magnitude of these effects are measured as well as the states on which they depend. After the measurement of these values, the signals are filtered such that the noise is assumed to be omitted. This corresponds to scheme 1 of Fig. 2. The mapping that is used in this example is a mapping that results in an approximation by first-order splines. This gives a linear interpolation between the remaining training samples. There is no need of regularization, because the samples are assumed to be noise-free after the filtering. Therefore, the γ is set to infinity.

We are interested in the error that is introduced by the pruning between the target values and the approximation. This error gives no direct information on the error between the approximation and the true value.

From the complete nonsparse solution one sample is thrown out and the α's for the remaining samples are recalculated. This is repeated until no samples are left, to illustrate the growing of the error between the pruned and the unpruned approximation. The result of the pruning is given in Fig. 5. The error depicted in this figure is the error between the training points and the approximation. The 2-norm is used in this figure; the infinity-norm gave similar results. It can be observed that the minimal introduced error gives a better result than the minimal error method. This increase in difference can be explained by the fact that the minimal introduced error searches for the sample that will introduce the smallest error. If the diagonal elements have a large variance, the use of the introduced error instead of the present error will significantly alter the outcome. The difference in required support vectors if the allowed pruning error is between $5\ 10^{-5}$ and $5\ 10^{-3}$ is approximate 150.

V. Conclusion

In this paper, a new procedure is proposed to determine which sample can be omitted when LSSVMs are used. Instead of omitting the sample that gives the smallest error *now*, the sample that will *introduce* the smallest error is chosen.

In an example, it is shown that the minimal introduce error procedure outperforms the minimal error method. If the input data is independently identically distributed (i.i.d.) over the input space the minimal introduced error procedure is slightly better. If the data is not i.i.d. the method is better by far.

The calculations of the diagonal elements of a inverse is a computational intensive. This means that the computational load has increased using this method.

Appendix
Calculate the SVs

The set of equations that has to be solved is

$$\left[\begin{array}{c|c} 0 & \vec{1}^T \\ \hline \vec{1} & \Omega + \gamma^{-1} I \end{array}\right] \left[\begin{array}{c} b \\ \hline \alpha \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline y \end{array}\right]. \tag{22}$$

This set of linear equations can be solved fast. But this set of equations has to be solved repeatedly with only minor changes, namely the omission of one training sample, making the total calculation time large. The calculations have to be done all over again if a sample is omitted.

However, the set of α_k's can be calculated and updated as follows.

```
1) Decompose the matrix Ω into LL^T using
the Cholesky decomposition.
2) Calculate α and the bias.
3) Determine the training point that will
introduces the smallest error.
4) Downdate the matrix L and its inverse.
5) Goto 2 if the approximation is good
enough to omit another data point.
```

The Cholesky decomposition decomposes a symmetric positive definite (SPD) matrix into the form $\Omega = LL^T$, in which L is a lower triangle matrix. An algorithm to implement it can be found in [15].

A. Calculate α and the Bias

The submatrix Ω is SPD which makes it fast to solve a system $\Omega x = b$. This can be rewritten as $LL^T x = b$ and this can be solved in two steps $Ly = b$ $L^T x = y$. Because L is a lower triangle matrix, no pivoting is required for solving these equations.

The vectors of ones and zeros make the complete matrix on the left-hand side no longer SPD. The block matrix inversion lemma can be used to calculate the Lagrangian multipliers using the SPD property of the matrix Ω [13].

The inverse of the block matrix is given

$$\begin{bmatrix} A & D \\ C & B \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} + E\Delta^{-1}F & -E\Delta^{-1} \\ -\Delta^{-1}F & \Delta^{-1} \end{bmatrix} \tag{23}$$

with $\Delta = B - CA^{-1}D$, $E = A^{-1}D$ and $F = CA^{-1}$. The solution for α and b are

$$\begin{aligned} \alpha &= \left(\Omega^{-1} - \Omega^{-1}\vec{1}\left(\vec{1}^T\Omega^{-1}\vec{1}\right)^{-1}\vec{1}^T\Omega^{-1}\right) y \\ b &= \left(\left(\vec{1}^T\Omega^{-1}\vec{1}\right)^{-1}\vec{1}^T\Omega^{-1}\right) y. \end{aligned} \tag{24}$$

In [16], an algorithm is given to calculate it for classification problems, but this algorithm can be altered to use for function approximation.

B. Determine the Training Point

To determine which training point might be omitted, the diagonal of the inverse should be determined next to the α. Because the matrix $\Omega = LL^T$, the inverse of Ω equals $\Omega^{-1} = (LL^T)^{-1} = L^{-1^T}L^{-1}$. To calculate the inverse of Ω, it is sufficient to calculate the inverse of L. This inverse can be calculated with [17]

$$\begin{aligned} s_{ii} &= \frac{1}{l_{ii}} \\ s_{ij} &= \frac{-1}{l_{ii}} \left(\sum_{k=1}^{i-1} l_{ik} s_{kj} \right). \end{aligned} \tag{25}$$

In this equation l is an element of the matrix L and the s is an element of the inverse of L. The diagonal elements of the inverse of Ω are given by

$$o_{ii} = \sum_{k=1}^{i} s_{ik}^2. \tag{26}$$

In this o is an element of the inverse of Ω. The complete inversion of L only has to be calculated the first time, afterwards only a part of the inverse matrix has to be changed.

By using the diagonal elements of the matrix Ω instead of the complete matrix, a small error is introduced. Because the α is divided by this value and only the smallest of this division is important, the influence of this error is small.

C. Downdate

After it is determined which training sample will introduce the smallest error, this sample should be omitted from the training set. The removal of a training sample means it's removal from the target set and the removal of the corresponding row/column in the matrix Ω. This requires the decomposition matrix L to be updated. The updating of L can be done without the complete recalculation of L.

If the original matrix and its decomposition are given by

$$\Omega = \left[\begin{array}{cc|c} A & \alpha & B \\ \alpha^T & a & \beta^T \\ \hline B^T & \beta & C \end{array}\right] \quad L = \left[\begin{array}{cc|c} R & 0 & 0 \\ \rho^T & r & 0 \\ \hline P & \pi & N \end{array}\right]. \tag{27}$$

Then, from $\Omega = LL^T$, the following relations are obtained:

$$\begin{aligned} RR^T &= A & r^2 + \rho^T\rho &= a \\ RP^T &= B & P\rho + \pi r &= \beta \\ R\rho &= \alpha & NN^T + \pi\pi^T + PP^T &= C. \end{aligned} \tag{28}$$

If the row $[\alpha^T \quad a \quad \beta^T]$ and the corresponding column are deleted from the matrix Ω the new matrix and its decomposition are given by

$$\Omega_{\text{new}} = \left[\begin{array}{c|c} A & B \\ \hline B^T & C \end{array}\right] \qquad L_{\text{new}} = \left[\begin{array}{c|c} R & 0 \\ \hline P & Q \end{array}\right] \tag{29}$$

and the following relations should hold true

$$\begin{aligned} A &= RR^T \\ B &= RP^T \\ C &= PP^T + QQ^T. \end{aligned} \tag{30}$$

The relations before and after the update show that the submatrices R and P do not change by omission of a row/column. The matrix Q satisfies $QQ^T = NN^T + \pi\pi^T$. This can be calculated by a Cholesky update [18].

The inverse of an lower triangle matrix is given as [13]

$$\begin{bmatrix} A & 0 \\ C & B \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} & 0 \\ -B^{-1}CA^{-1} & B^{-1} \end{bmatrix}. \tag{31}$$

It was argued that only the lower right corner of the matrix L changed due to the omission of a sample. This corresponds to the submatrix B in the equation above. Therefore only those parts of the inverse should be updated in which B is present.

ACKNOWLEDGMENT

The involvement in this research of Tecnotion B.V., Almelo, The Netherlands and Imotec B.V, Hengelo, The Netherlands, is gratefully acknowledged.

REFERENCES

[1] V. N. Vapnik, *The Nature of Statistical Learning Theory*, 2nd ed. New York: Springer-Verlag, 2000.
[2] M. Brown and C. Harris, *Neurofuzzy Adaptive Modeling and Control*. Chichester, U.K.: Prentice-Hall, 1994.
[3] J. A. K. Suykens, J. de Brabanter, L. Lukas, and J. Vandewalle, "Weighted least squares support vector machines: Robustness and sparse approximation," Neurocomput..
[4] A. J. Smola, "Learning with Kernels," Ph.D. dissertation, GMD, Birlinghoven, Germany, 1998.
[5] W. C. Chan, C. W. Chan, K. C. Cheung, and C. J. Harris, "On the modeling of nonlinear dynamic systems using support vector neural networks," *Engineering Applications of Artificial Intelligence*, vol. 14, pp. 105–113, 2001.
[6] B. J. de Kruif and T. J. A. de Vries, "On using a support vector machine in learning feed-forward control," in *Proc. Int. Conf. Advanced Intelligent Mechatronics*, Como, Italy, July 2001, pp. 272–277.
[7] M. Aoki, *Introduction to Optimization Techniques, Fundamentals and Applications of Nonlinear Programming*. New York: Macmillan, 1971, Macmillan Series in Applied Computer Sciences.
[8] J. Mercer, "Functions of positive and negative type and their connection with the theory of integral equations," *Trans. London Philosophical Soc.*, vol. A, no. 209, pp. 415–446, 1909.
[9] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, Feb. 1970.
[10] S. Haykin, *Neural Networks, A Comprehensive Foundation*. Upper Saddle River, NJ: Prentice-Hall, 1994.
[11] L. Prechelt, "Connection pruning with static and adaptive pruning," *Neurocomputing*, vol. 16, pp. 49–61, 1997.
[12] R. Reed, "Pruning algorithms – A survey," *IEEE Trans. Neural Networks*, vol. 4, pp. 740–747, Sept. 1993.
[13] T. Kailath, *Linear Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1980.

[14] B. Hassibi, D. G. Stork, and G. J. Wolff, "Optimal brain surgeon and general network pruning," in *Proc. IEEE Int. Conf. Neural Networks*, San Francisco, CA, 1992, pp. 293–299.
[15] G. H. Golub and C. F. van Loan, *Matrix computations*, 3rd ed. Baltimore, MD: Johns Hopkins Univ. Press, 1996.
[16] J. A. K. Suykens, P. Van Dooren, B. De Moor, and J. Vandewalle, "Least squares support vector machine classifiers: A large scale algorithm," in *Proc. European Conf. Circuit Theory Design*, 1999, pp. 839–842.
[17] T. Seaks, "Syminv: An algorithm for the inversion of a positive definite matrix by the cholesky decomposition," *Econometrica*, vol. 40, no. 5, pp 961–962, Sept. 1972.
[18] G. W. Stewart, *Matrix Algorithms*. Philadelphia, PA: SIAM, 1998, vol. 1. Basic Decompositions.

XVI. *Functions of Positive and Negative Type, and their Connection with the Theory of Integral Equations.*

By J. MERCER, *B.A., Trinity College, Cambridge.*

Communicated by Prof. A. R. FORSYTH, *Sc.D., LL.D., F.R.S.*

Received December 21, 1908—Read May 13, 1909.

Introduction.

THE present memoir is the outcome of an attempt to obtain the conditions under which a given symmetric and continuous function $\kappa(s, t)$ is definite, in the sense of HILBERT.* At an early stage, however, it was found that the class of definite functions was too restricted to allow the determination of necessary and sufficient conditions in terms of the determinants of § 10. The discovery that this could be done for functions of positive or negative type, and the fact that almost all the theorems which are true of definite functions are, with slight modification, true of these, led finally to the abandonment of the original plan in favour of a discussion of the properties of functions belonging to the wider classes.

The first part of the memoir is devoted to the definition of various terms employed, and to the re-statement of the consequences which follow from HILBERT'S theorem.

In the second part, keeping the theory of quadratic forms in view, the necessary and sufficient conditions, already alluded to, are obtained. These conditions are then applied to obtain certain general properties of functions of positive and negative type.

Part III. is chiefly devoted to the investigation of a particular class of functions of positive type. In addition, it includes a theorem which shows that, in general, from each function of positive type it is possible to deduce an infinite number of others of that type.

Lastly, in the fourth part, it is proved that when $\kappa(s, t)$ is of positive or negative type it may be expanded as a series of products of normal functions, and that this series converges both absolutely and uniformly.

* 'Gött. Nachr.' (1904), Heft I.

PART I.—DEFINITIONS AND DEDUCTIONS FROM HILBERT'S THEOREM.

§ 1. Let $\kappa(s,t)$ be a continuous symmetric function of the variables s, t which is defined in the closed square $a \leq s \leq b$, $a \leq t \leq b$; and let Θ be the class of all functions which are continuous in the closed interval (a, b). When the function θ ranges through the class Θ, there are three possible ways in which the double integral

$$\int_a^b \int_a^b \kappa(s,t)\,\theta(s)\,\theta(t)\,ds\,dt$$

may behave :—

(i) There may be two members of Θ, say θ_1 and θ_2, such that

$$\int_a^b \int_a^b \kappa(s,t)\,\theta_1(s)\,\theta_1(t)\,ds\,dt, \qquad \int_a^b \int_a^b \kappa(s,t)\,\theta_2(s)\,\theta_2(t)\,ds\,dt$$

have opposite signs ;

(ii) Each function θ may be such that

$$\int_a^b \int_a^b \kappa(s,t)\,\theta(s)\,\theta(t)\,ds\,dt \geq 0\,;$$

(iii) Each function θ may be such that

$$\int_a^b \int_a^b \kappa(s,t)\,\theta(s)\,\theta(t)\,ds\,dt \leq 0.$$

This suggests a classification of continuous symmetric functions defined in the closed square. We shall speak of those which have the property (i) as functions of *ambiguous type*, whilst the others will be said to be of *positive* or *negative type*, according as they satisfy (ii) or (iii).

§ 2. From the point of view of integral equations this classification is of considerable importance. HILBERT has proved* that

$$\int_a^b \int_a^b \kappa(s,t)\,\theta(s)\,\theta(t)\,ds\,dt = \sum_{n=1} \frac{1}{\lambda_n}\left[\int_a^b \psi_n(s)\,\theta(s)\,ds\right]^2,$$

where $\psi_1(s)$, $\psi_2(s)$, ..., $\psi_n(s)$, ..., are a complete system of normal functions relating to the characteristic function $\kappa(s,t)$ of the integral equation

$$f(s) = \phi(s) - \lambda \int_a^b \kappa(s,t)\,\phi(t)\,dt,$$

and $\lambda_1, \lambda_2, \ldots, \lambda_n, \ldots$, respectively, are the corresponding singular values. It follows at once from this that, when the singular values are all positive, $\kappa(s,t)$ is of positive

* 'Gött. Nachr.' (1904), pp. 69–70. See also SCHMIDT, 'Math. Ann.,' Band 63, pp. 452, 453. We shall refer to the result given above as HILBERT'S *theorem*. The theorem stated by HILBERT on p. 70 of the paper referred to can be deduced by writing $\theta(s) = x(s) + y(s)$ in the equation written above.