PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 48, NO. 1–2, PP. 17–22 (2004)

A SPARSE LEAST SQUARES SUPPORT VECTOR MACHINE CLASSIFIER

József VALYON

Department of Measurement and Information Systems
Budapest University of Technology and Economics
H–1521 Budapest, Hungary
Phone: +36 1 463-2057, Fax: +36 1 463-4112
e-mail: valyon@mit.bme.hu

Received: January 6, 2004

Abstract

In the last decade Support Vector Machines (SVM) – introduced by Vapnik – have been successfully applied to a large number of problems. Lately a new technique, the Least Squares SVM (LS–SVM) has been introduced, which addresses classification and regression problems by formulating a linear equation set. In comparison to the original SVM, which involves a quadratic programming task, LS–SVM simplifies the required computation, but unfortunately the sparseness of standard SVM is lost. The linear equation set of LS–SVM embodies all available information about the learning process. By applying modifications to this equation set, we present a Least Squares version of the Least Squares Support Vector Machine (LS^2–SVM). The modifications simplify the formulations, speed up the calculations and provide better results, but most importantly it concludes a sparse solution.

Keywords: Support Vector Machines, Least Squares Support Vector Machines, regression, classification, system modelling.

1. Introduction

Among Neural Networks, the main advantage of SVM methods is that they automatically derive a network structure which guarantees an upper bound on the generalization error. This is very important in a large number of real life classification problems.

The Least Squares Support Vector Machine (LS–SVM) is attracting increasing attention, mostly because it has some very promising properties regarding the implementation and the computational issues of teaching. In this case, training means solving a set of linear equations, instead of the quadratic programming problem involved by the standard SVM [1].

While the least squares version incorporates all training data in the network to produce the result, the traditional SVM selects some of them (called support vectors) that have a more significant effect on the classification. Because LS–SVM does not incorporate a support vector selection method, the network size is usually much larger than it would be with a traditional SVM. Sparseness can also be reached with LS–SVM by applying a pruning method [2], but this iterative process requires an equation set – slowly decreasing in size – to be solved in every step, which multiplies the complexity.

The optimal solution should combine the desirable features of these methods. It: (1) should be fast, (2) should lead to a sparse solution, (3) should produce good results. In order to achieve these goals, two new methods are introduced in the sequel. The combination of these methods leads to a sparse LS–SVM solution, which means that a smaller network – based on a subset of the training samples – is accomplished with the speed and simplicity of the least squares solution.

The LS–SVM method is capable of solving both classification and regression problems. The present study concerns classification therefore only this is introduced in the sequel, along with the standard pruning method. Only a brief outline of these methods is presented, a detailed description can be found in refs. [2, 3, 4] and [5].

2. A Brief Overview of the LS–SVM Method

Given the $\{\mathbf{x}_i, d_i\}_{i=1}^{N}$ training data set, where $\mathbf{x}_i \in \Re^p$ represents a p-dimensional input vector with $d_i \in \{-1, +1\}$ labels, our goal is to construct a classifier of form

$$y(\mathbf{x}) = \operatorname{sign}\left[\sum_{j=1}^{h} w_j \varphi_j(\mathbf{x}) + b\right] = \operatorname{sign}\left[\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) + b\right],$$
$$\mathbf{w} = [w_1, w_2, \ldots, w_h]^T, \qquad \boldsymbol{\varphi} = [\varphi_1, \varphi_2, \ldots, \varphi_h]^T. \tag{1}$$

The $\varphi(\cdot) : \Re^p \rightarrow \Re^h$ is a mostly non-linear function, which maps the data into a higher (possibly infinite $-h$) dimensional feature space. The optimization problem can be given by the following equations ($k = 1, \ldots, N$):

$$\min_{\mathbf{w},b,\mathbf{e}} J_p(\mathbf{w}, \mathbf{e}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\frac{1}{2}\sum_{k=1}^{N} e_k^2, \text{ with constraints: } d_k\left[\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_k) + b\right] = 1 - e_k. \tag{2}$$

The first term is responsible to find a smooth solution, while the second one minimizes the training errors (C is the trade–off parameter between the terms). From this, the following Lagrangian can be formed:

$$L(\mathbf{w}, b, \mathbf{e}; \boldsymbol{\alpha}) = J_p(\mathbf{w}, \mathbf{e}) - \sum_{k=1}^{N} \alpha_k \left\{d_k\left[\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_k) + b\right] - 1 + e_k\right\}, \tag{3}$$

where the α_k parameters are the Lagrange multipliers. The solution concludes in a constrained optimization and the following overall solution:

$$\begin{bmatrix} 0 & \mathbf{d}^T \\ \mathbf{d} & \boldsymbol{\Omega} + C^{-1}\mathbf{I} \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \vec{\mathbf{1}} \end{bmatrix}, \qquad \mathbf{d} = [d_1, d_2, \ldots, d_N]^T,$$
$$\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_N]^T, \qquad \vec{\mathbf{1}} = [1, \ldots, 1]^T, \qquad \Omega_{i,j} = d_i d_j K(\mathbf{x}_i, \mathbf{x}_j), \tag{4}$$

where $C \in \Re$ is a positive constant, b is the bias and the result is: $y(\mathbf{x}) = \sum_{k=1}^{N} \alpha_k d_k K(\mathbf{x}, \mathbf{x}_k) + b$. This result can be interpreted as a neural network, which contains N non-linear neurons in its single hidden layer. The number of these nonlinear neurons equals the number of selected support vectors (network size). The result (y) is the weighted sum of the outputs of the middle layer neurons. The weights are the calculated α_k Lagrange multipliers. Although in practice SVMs are rarely formulated as actual networks, this neural interpretation is important, because it provides an easier discussion framework than the purely mathematical approach. This paper uses the neural interpretation throughout the discussions, because the points and statements of this work can be more easily understood this way.

It is important to mention that the α_i weights are proportional to the e_i errors in the training points: $\alpha_i = Ce_i$. The following iterative methods are based on this property of the LS–SVM.

LS–SVM pruning [2], [3]: In most real life situations the LS–SVM networks are unnecessarily large. This drawback can be eliminated by applying a pruning method which eliminates some training samples based on the sorted support vector spectrum [2]. The weighting of the Least Squares SVM reflects the importance of the training samples, therefore by eliminating some vectors, represented by the smallest values from this $|\alpha_i|$ spectrum, the number of neurons can be reduced.

3. The Proposed Method

3.1. Modifying the Equation Set

If the training set consists of N samples, then our original linear equation set will have $(N+1)$ unknowns, the α_i-s, $(N+1)$ equations and $(N+1)^2$ multiplication coefficients. These factors are mainly the $K(\mathbf{x}_i, \mathbf{x}_k)$ kernel matrix elements representing every training input pairs. The cardinality of the training set therefore determines the size of this coefficient matrix. Let's take a closer look at the linear equation set describing the problem (4). To reduce the kernel matrix, columns and/or rows may be omitted. If the kth *column* is left out, then the corresponding α_k weight is also removed, therefore the resulting network will be smaller. If the kth *row* is omitted, then the input–output defined by the $(\mathbf{x}_k, d_k)$ training sample is lost, because the kth equation is removed. This leads to a less constrained, and therefore worst solution. Each column (k) stands for a neuron, with a kernel centered on the corresponding input ($\mathbf{x}_k$). The formulation of this matrix can be generalised as follows:

1. The number of kernels (M) may be less than N, so columns may be represented by M chosen $\mathbf{c}_j$ vectors. $\{c_1, c_2, \ldots, c_M \mid c_i \in \{x_1, x_2, \ldots, x_N\}, M < N\}$
2. The kernel functions may be different from column to column.

The formulation of $\mathbf{\Omega}$ changes as follows:

$$\Omega_{j,k} = d_j d_k K_k\left(\mathbf{x}_j, \mathbf{c}_k\right), \tag{5}$$

and the result will be calculated from $y(\mathbf{x}) = \sum_{k=1}^{M} \alpha_k d_k K_k\left(\mathbf{x}, c_k\right) + b$, where M is the number of kernels (nonlinear neurons) used. The $\mathbf{c}_k$ centers may be selected from the training sample set (from the $\mathbf{x}_k$-s), which is assumed in this paper. A possible selection method is proposed in the next section.

Reducing only one dimension of the kernel matrix is referred to as partial reduction [6]. Using fewer columns than training samples, means less weights (α_k) and, consequently, a sparse solution. It also leads to an overdetermined equation set, which can be solved as a linear least squares problem, consisting of only $(M+1) \times (N+1)$ coefficients.

$$\left[\begin{array}{c|ccc} 0 & & \mathbf{d}^T & \\ \hline & K_1(\mathbf{x}_1,\mathbf{c}_1)+C^{-1} & \cdots & K_M(\mathbf{x}_1,\mathbf{c}_M) \\ & \vdots & \ddots & \vdots \\ \mathbf{d} & K_M(\mathbf{x}_M,\mathbf{c}_1) & \cdots & K_M(\mathbf{x}_M,\mathbf{c}_M)+C^{-1} \\ & \vdots & \ddots & \vdots \\ & K_1(\mathbf{x}_N,\mathbf{c}_1) & \cdots & K_M(\mathbf{x}_N,\mathbf{c}_M) \end{array}\right] \left[\begin{array}{c} b \\ \hline \alpha_1 \\ \vdots \\ \alpha_M \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{array}\right] \tag{6}$$

This equation set is written shortly as $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$, $\mathbf{x}$ and $\mathbf{b}$ are the matrixes in *Eq.* (6) respectively.

There is a slight problem with the regularisation parameter C since it can only be inserted in the first M rows, but it is enough to ensure us M linearly independent rows, so the equation set can be solved. The solution is calculated as

$$\mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{b} \tag{7}$$

The modified matrix $\mathbf{A}$ has $(N+1)$ rows and $(M+1)$ columns. After the matrix multiplications the results are obtained from a reduced equation set, incorporating $\mathbf{A}^T\mathbf{A}$, which is only of size $(M+1) \times (M+1)$. Reducing only the number of columns and not the rows means that the number of neurons is reduced, but all the known constraints are taken into consideration. This is the key concept of keeping the quality, while sparseness is achieved. When traditional iterative pruning is applied to the LS–SVM solution some training points are fully omitted. They do not participate in the next kernel matrix, therefore information embodied in the subset of dropped points are entirely lost!

Since the modified LS–SVM equation set is solved in a least squares sense, we name this method LS^2–SVM.

3.2. B. A Support Vector Selection Method

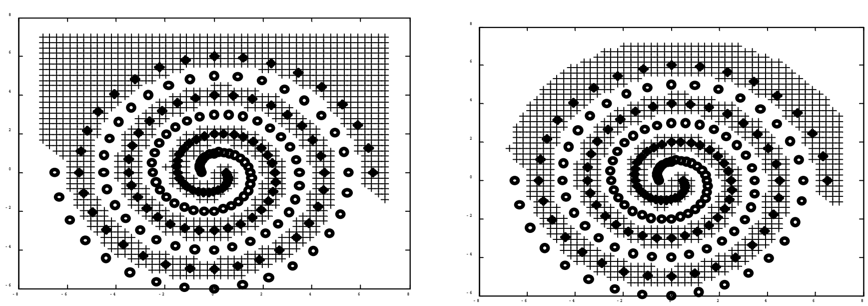
As the kernel matrix is formed from columns we can select a linearly independent subset of column vectors and omit all others. This can be done by finding a ‘basis’ of the coefficient matrix. A slight modification of a common mathematical method – used for bringing the matrix to the reduced row echelon form – can be utilized to find this 'basis'. When searching for basis vectors, the linear dependence of vectors does not mean exact linear dependence, because the method uses an adjustable tolerance value when determining the 'resemblance' of the column vectors. The use of this tolerance value is essential, because none of the columns of $\mathbf{\Omega}$ will likely be exactly dependent, especially if the selection is applied to the regularized $\mathbf{\Omega}+C^{-1}\mathbf{I}$ matrix. The larger the tolerance, the fewer vectors the algorithm will select.

4. Experiments

First, the two spiral benchmark problem is presented. The results for this CMU (Carnegie Melon University) benchmark is plotted in *Fig. 1*. It shows that both methods are perfectly capable of distinguishing between the two input sets.

The next table summarises the results for some UCI benchmarks. In the experiments we split the datasets to a training and a test set as seen in ref. [5].

For simple problems consisting of many samples, the gain is high, because a lot of samples may be pruned (e.g. Bupa liver disorders), while for hard problems, with a small sample set (e.g. Statlog heart disease) the network size cannot be reduced.



(a) (b)

Fig. 1. The classification boundaries obtained for the standard LS–SVM (a) and the LS ^{2}–SVM (b)

Table 1. Results achieved for benchmark problems. Where N_{TR} is the number of training inputs and N_{TS} is the number of test samples. The $N_{\mathrm{LS}^2-\mathrm{SVM}}$ column contains the network size of our sparse solution. The last two columns show the good/miss classification rates for the test sets.

Bench-mark	N_{TR}	N_{TS}	$N_{\mathrm{LS-SVM}}$	$N_{\mathrm{LS}^2-\mathrm{SVM}}$	LS–SVM	LS^2–SVM
Bupa liver disorders	230	115	230	37	67.82/32.18	70.44/29.56
Pima Indians diabetes	512	256	512	379	67.97/32.03	68.36/31.64
Tic–tac–toe endgame	638	320	638	136	97.19/ 2.81	94.37/ 5.63
Statlog heart disease	180	90	180	168	72.23/27.77	70.00/30.00

5. Conclusion

In this paper a sparse LS-SVM was presented. The basic idea is that the number of input vectors chosen to be centers can be reduced, hence the main equation set may be overdetermined. By eliminating variables a pruned solution can be achieved.

References

[1] VAPNIK, V., *The Nature of Statistical Learning Theory*, Springer, New York, 1955.

[2] SUYKENS, J. A. K. – LUKAS, L. – VANDEWALLE, J., Sparse Least Squares Support Vector Machine Classifiers, In: *ESANN'2000 European Symposium on Artificial Neural Networks*, (2000), pp. 37–42.

[3] SUYKENS, J. A. K. – LUKAS, L. – VANDEWALLE, J., Sparse Approximation Using Least Squares Support Vector Machines, In: *IEEE International Symposium on Circuits and Systems ISCAS'2000*.

[4] SUYKENSM, J. A. K., Nonlinear Modeling and Support Vector Machines, *IEEE Instrumentation and Measurement Technology Conference*, Budapest, Hungary, 2001.

[5] SUYKENS, J. A. K. – GESTEL, V. T. – DE BRABANTER, J. – DE MOOR, B. – VANDEWALLE, J., *Least Squares Support Vector Machines*, World Scientific, 2002.

[6] VALYON, J. – HORVATH, G., A Weighted Generalized LS–SVM, Accepted in: *Periodica Polytechnica*.