Reducing the complexity and controlling the network size of LS–SVM solutions, by solving an overdetermined set of equations

József Valyon

Introduction: In case of noisy learning data, the traditional NN –due to it´s construction – often leads to poor generalization and "over–fitting". The SVM [1],[2] (Support Vector Machine) method, introduced by Vapnik, was designed to overcome these problems. The LS–SVM (Least Squares SVM) [3],[4] provides us with the similar advantages, but in this case training means solving a set of linear equations instead of a long and computationally hard quadratic programming problem involved by the standard SVM. This LS method effectively reduces the algorithmic complexity, but for really large problems, comprising a very large number of training samples, even this least-squares solution can become highly memory and time consuming. The least–squares version incorporates all input vectors in the network to produce the result, while the traditional SVM selects, marks some vectors (called support vectors) as ones that are important in the regression. This behavior can also be reached with LS-SVM by applying a pruning method [4], but in order to achieve it, the entire large problem must be solved at least once. This paper describes a new formulation of the LS–SVM, which provides us with control over the size and structure of the network, and at the same time reduces the complexity –time and memory requirements– of the required calculations. The LS–SVM method is capable of solving both classification and regression problems. Our sample problem concerns regression (LS–SVR), therefore we discuss this in the sequel. With minor changes the given algorithm can also be applied to classification.

The LS-SVR method [3],[4]: A training data set $\{\mathbf{x}_i, d_i\}_{i=1}^{N}$ is obtained, where $\mathbf{x}_i \in \mathbb{R}^p$ represents a p-dimensional input vector and $d_i \in \mathbb{R}$ is the scalar target output. Our goal is to approximate an $y = f(\mathbf{x})$ function, which represents the dependence of the output d from the input $\mathbf{x}$. Let´s define the form of this function as formulated below:

$$y = \sum_{j=0}^{m_1} w_j \varphi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}), \quad \mathbf{w} = [w_0, w_1, \dots, w_{m_1}]^T, \quad \boldsymbol{\varphi} = \lfloor \varphi_0(\mathbf{x}), \varphi_1(\mathbf{x}), \dots, \varphi_{m_1}(\mathbf{x}) \rfloor.$$

The $\varphi_0(\mathbf{x})$ basis function is assumed to be 1, therefore w_0 represents the bias b. The solution concludes in a constrained optimization, which leads to the following overall solution:

$$\begin{bmatrix} 0 & \vec{\mathbf{1}}^T \\ \vec{1} & \Omega + C^{-1}\mathbf{I} \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix}, \quad \mathbf{y} = [y_0, y_1, \dots, y_N], \quad \vec{\mathbf{1}} = [1, \dots, 1], \quad \boldsymbol{\alpha} = [\alpha_0, \alpha_1, \dots, \alpha_N],$$

$\Omega_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j)$ (C is a chosen constant, $K(\mathbf{x}_i, \mathbf{x}_j)$ is a kernel function) and the estimation is: $y = \sum_{i=1}^{N} \alpha_i K(\mathbf{x}, \mathbf{x}_j) + b$.

The reduced method: If the training set comprises N samples, then our original linear equation set consists of $(N+1)$ equations, $(N+1)$ unknowns and $(N+1)^2$ coefficients. By selecting some (e.g. $M, M < N$) vectors to be "*support vectors*", the number of variables are reduced, resulting in more equations than unknowns. Our problem becomes overdetermined, which can be solved as a *linear least–squares problem*, consisting only $(M+1)^2$ coefficients. Every variable stands for a neuron –representing it´s weight– and each of the M selected training vectors will become a center of a kernel function. Therefore the selected inputs must be chosen accordingly (e.g. equally distanced, less noisy etc.). The above described method solves a much smaller problem, but still takes all known samples into consideration! The result for a simple noisy $sinc(x)$ regression is plotted on figure 1, where (+) –s are the noisy training samples (every second one –*circled*– is selected into set M), (.) represents the result of the reduced LS–SVM method and (o) stands for the result of the normal solution. It can be seen, that the above described method –when the training set is large enough– leads to almost the same results based on a much smaller equation set, as the original solution. The resulting network is smaller, whilst the algorithmic complexity is reduced.

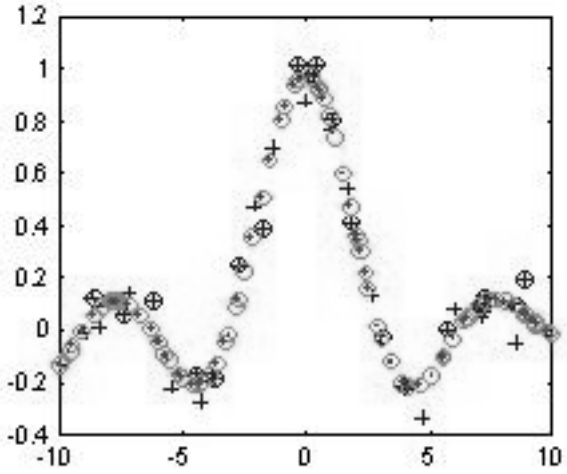

Figure 1: A $sinc(x)$ regression based on the described method ($M = N/2$).

References

[1] S. Haykin: "Neural networks. A comprehensive foundation", Prentice Hall, N. J. 1999

[2] S. Gunn, "Support Vector Machines for Classification and Regression", ISIS Technical Report, 14. May 1998

[3] J. A. K. Suykens and J. Vandewalle, "Least Squares Support Vector Machine Classifiers", Neural Processing Letters, vol. 9, no. 3, Jun. 1999, pp. 293-300.

[4] J. A. K. Suykens, "Nonlinear Modeling and Support Vector Machines", IEEE Instrumentation and Measurement Technology Conference, Budapest, Hungary, May 21–23, 2001.