Journal of the Korean Data & Information Science Society
2016, **27**(3), 827–833

http://dx.doi.org/10.7465/jkdi.2016.27.3.827
한국데이터정보과학회지

Deep LS-SVM for regression[†]

Changha Hwang[1] · Jooyong Shim[2]

[1]Department of Applied Statistics, Dankook University
[2]Department of Statistics, Inje University

Received 7 April 2016, revised 21 April 2016, accepted 25 April 2016

Abstract

In this paper, we propose a deep least squares support vector machine (LS-SVM) for regression problems, which consists of the input layer and the hidden layer. In the hidden layer, LS-SVMs are trained with the original input variables and the perturbed responses. For the final output, the main LS-SVM is trained with the outputs from LS-SVMs of the hidden layer as input variables and the original responses. In contrast to the multilayer neural network (MNN), LS-SVMs in the deep LS-SVM are trained to minimize the penalized objective function. Thus, the learning dynamics of the deep LS-SVM are entirely different from MNN in which all weights and biases are trained to minimize one final error function. When compared to MNN approaches, the deep LS-SVM does not make use of any combination weights, but trains all LS-SVMs in the architecture. Experimental results from real datasets illustrate that the deep LS-SVM significantly outperforms state of the art machine learning methods on regression problems.

Keywords: Deep learning, hidden layer, least squares support vector machine, multilayer neural network, penalized objective function.

1. Introduction

Vapnik (1995) and his group at AT&T Bell Laboratories firstly developed the support vector machine (SVM), which has been successfully applied to a number of real world problems such as the classification and regression. Despite of lots of successful applications of SVM in regression and classification problems, a quadratic programming problem is required to train SVM, which is computationally formidable in many cases. Suykens and Vanderwalle (1999) introduced the least squares SVM (LS-SVM) which is a least squares version of SVM. LS-SVM has been proved to be a very appealing and promising method. Introductions and overviews of recent developments of SVM and LS-SVM can be found in Smola and Scholkopf (1998), Suykens *et al.* (2001), Hwang (2014, 2015, 2016), Seok (2015), and Shim and Seok

[†] This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (NRF-2015R1D1A1A01056582). This work was supported by the Ministry of Education of the Republic of Korea and the National Research Foundation of Korea (NRF-2015S1A3A2046715).
[1] Professor, Department of Applied Statistics, Dankook University, Yongin 16890, Korea.
[2] Corresponding author: Adjunct professor, Department of Statistics, Institute of Statistical Information, Inje University, Kimhae 50834, Korea. E-mail: ds1631@hanmail.net

(2014). SVM and LS-SVM are in principle shallow architectures and thus tend to perform worse for the complicated data because of its simple structure (Bengio and Le Cun, 2007). However, deep architectures with neural networks have been shown to achieve state-of-the-art performances in many pattern recognition problems.

As a result, several researchers have studied whether kernel learning can be modified for deep learning. For example, Cho and Saul (2009) firstly studied the kernel method for deep learning by optimizing an arccosine kernel which mimics the massive random projections of an infinite neural network. However, the method did not allow easily tunable parameters beyond the first layer. Zhuang *et al.* (2011) proposed to tune a combination of kernels but had trouble optimizing the network beyond two layers. For classification problems, Li *et al.* (2014) proposed deep twin support vector machine, which combines twin support vector machine with deep learning ideas. Wiering and Schomaker (2014) described a simple method for constructing and training multilayer SVMs.

In this paper, we present the deep LS-SVM that combines ideas from deep neural network architectures with those of LS-SVM. We can easily make the association between our approach and multilayer neural network (MNN) by replacing each LS-SVM with each individual neuron. However, in the deep LS-SVM, LS-SVMs of the hidden layer and the main LS-SVM are trained to minimize the penalized objective function.

The remainder of paper is organized as follows. In Section 2 we give a brief review LS-SVM. In Section 3 we propose the deep LS-SVM which consists of input layer and hidden layer. In Section 4 we illustrate the performance of the proposed method through five real data sets. Section 5 contains the conclusions.

2. LS-SVM

Given the training data set $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ with each input vector $\boldsymbol{x}_i \in R^d$ and the response $y_i \in R$, and a test data point by $\boldsymbol{x}_t$, we consider the nonlinear regression function given as the form of $f(\boldsymbol{x}) = \boldsymbol{w}'\boldsymbol{\phi}(\boldsymbol{x}) + b$, where b is a bias. Here $\boldsymbol{\phi} : R^d \to R^{d_f}$ is the nonlinear feature mapping function which maps the input space to the higher dimensional feature space, where the dimension d_f defined in an implicit way.

The optimization problem of LS-SVM is defined as follows:

$$\min \frac{1}{2}\boldsymbol{w}'\boldsymbol{w} + \frac{C}{2}\sum_{i=1}^n e_i^2 \tag{2.1}$$
$$\text{subject to } e_i = y_i - \boldsymbol{w}'\boldsymbol{\phi}(\boldsymbol{x}_i) - b, i = 1, \cdots, n,$$

where $C > 0$ is a penalty parameter which controls the tradeoff between the goodness-of-fit on the data and $\boldsymbol{w}'\boldsymbol{w}$. From (2.1) the Lagrangian function can be constructed as follows:

$$L = \frac{1}{2}\boldsymbol{w}'\boldsymbol{w} + \frac{C}{2}\sum_{i=1}^n e_i^2 - \sum_{i=1}^n \alpha_i(e_i - y_i + \boldsymbol{w}'\boldsymbol{\phi}(\boldsymbol{x}_i) + b), \tag{2.2}$$

where α_i's are the Lagrangian multipliers.

From the optimality conditions we have,

$$\begin{aligned}
&\frac{\partial L}{\partial \boldsymbol{w}} = 0 \ \rightarrow \ \boldsymbol{w} - \sum_{i=1}^{n} \alpha_i \boldsymbol{\phi}(\boldsymbol{x}_i) = \mathbf{0} \\
&\frac{\partial L}{\partial b} = 0 \ \rightarrow \ \sum_{i=1}^{n} \alpha_i = 0 \\
&\frac{\partial L}{\partial e_i} = 0 \ \rightarrow \ Ce_i - \alpha_i = 0, \ i = 1, \cdots, n \\
&\frac{\partial L}{\partial \alpha_i} = 0 \ \rightarrow \ e_i - y_i + \boldsymbol{w}'\boldsymbol{\phi}(\boldsymbol{x}_i) + b = 0, \ i = 1, \cdots, n,
\end{aligned}$$

which are equivalent to the linear equations as follows:

$$y_i = \sum_{j=1}^{n} K(\boldsymbol{x}_i, \boldsymbol{x}_j)\alpha_i + \alpha_i/C + b, i = 1, \cdots, n, \text{ and } \sum_{i=1}^{n} \alpha_i = 0, \tag{2.3}$$

where $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\phi}(\boldsymbol{x}_i)'\boldsymbol{\phi}(\boldsymbol{x}_j)$, which is obtained by the application of Mercer's conditions (1909).

From the linear equation (2.3) the optimal Lagrange multipliers and the estimated bias, $\widehat{\alpha}_i$'s and $\widehat{b}$ can be obtained. Then the predicted regression function given $\boldsymbol{x}_t \in R^p$ is obtained as

$$\widehat{f}(\boldsymbol{x}_t) = \boldsymbol{k}_t\widehat{\boldsymbol{\alpha}} + \widehat{b} = \boldsymbol{H}_t\boldsymbol{y}, \tag{2.4}$$

where $\boldsymbol{k}_t = (K(\boldsymbol{x}_t, \boldsymbol{x}_1), \cdots, K(\boldsymbol{x}_t, \boldsymbol{x}_n))'$, $\widehat{\boldsymbol{\alpha}} = (\alpha_1, \cdots, \alpha_n)'$, $\boldsymbol{H}_t = (\boldsymbol{k}_t, 1)\boldsymbol{H}_0$ with

$$\boldsymbol{H}_0 = \begin{pmatrix} (\boldsymbol{K} + \boldsymbol{I}/C)^{-1} - (\boldsymbol{K} + \boldsymbol{I}/C)^{-1}\mathbf{1}(\mathbf{1}'(\boldsymbol{K} + \boldsymbol{I}/C)^{-1}\mathbf{1})\mathbf{1}'(\boldsymbol{K} + \boldsymbol{I}/C)^{-1} \\ (\mathbf{1}'(\boldsymbol{K} + \boldsymbol{I}/C)^{-1}\mathbf{1})^{-1}\mathbf{1}'(\boldsymbol{K} + \boldsymbol{I}/C)^{-1} \end{pmatrix}$$

and the $n \times n$ kernel matrix $\boldsymbol{K}$ with elements $K_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

The performance of LS-SVM is affected by the penalty parameter and the kernel parameters. To select the optimal values of those parameters of LS-SVM, we use the leave-one-out cross validation (LOOCV) function as follows:

$$CV(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{f}_i^{(-i)}(\boldsymbol{\theta}))^2, \tag{2.5}$$

where $\boldsymbol{\theta}$ is a candidate set of the penalty and the kernel parameters and $\widehat{f}_i^{(-i)}(\boldsymbol{\theta})$ is the predicted value of $f(\boldsymbol{x}_i)$ obtained from data without the ith observation. Since for each candidate set of the penalty and the kernel parameters, $\widehat{f}_i^{(-i)}(\boldsymbol{\theta})$ for $i = 1, \cdots, n$, should be evaluated, selecting parameters using LOOCV function is computationally formidable. By using leaving-out-one lemma (Wahba, 1990) and the first order of Taylor expansion, the ordinary cross validation function is obtained as follows:

$$OCV(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \widehat{f}_i(\boldsymbol{\theta})}{1 - \dfrac{\partial \widehat{f}_i}{\partial y_i}}\right)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \widehat{f}_i(\boldsymbol{\theta})}{1 - h_{ii}(\boldsymbol{\theta})}\right)^2, \tag{2.6}$$

where $h_{ii}(\boldsymbol{\theta})$ for $i = 1, \cdots, n$, is the ith diagonal element of the hat matrix $\boldsymbol{H}$ such that $\widehat{\boldsymbol{f}} = (f(\boldsymbol{x}_1), \cdots, f(\boldsymbol{x}_n))' = \boldsymbol{H}\boldsymbol{y}$. By averaging $(1 - h_{ii}(\boldsymbol{\theta}))$ in (2.6) by $(1 - trace(\boldsymbol{H})/n)$, the generalized cross validation (GCV) function is obtained as follows:

$$GCV(\boldsymbol{\theta}) = \frac{n \sum_{i=1}^{n} (y_i - \widehat{f}_i(\boldsymbol{\theta}))^2}{(n - trace(\boldsymbol{H}))^2}. \tag{2.7}$$

3. Deep LS-SVM

Given the training data set $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ with each input vector $\boldsymbol{x}_i \in R^d$ and the response $y_i \in R$, and a test data point by $\boldsymbol{x}_t$, we consider the nonlinear regression. We consider a two-layer deep LS-SVM as shown in Figure 3.1.

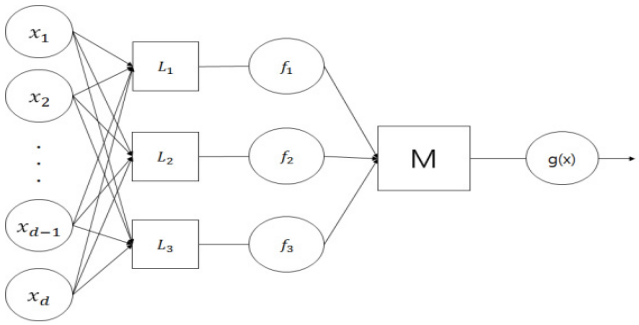

Figure 3.1 Architecture of a two-layer deep LS-SVM with three LS-SVMs of the hidden layer

The deep LS-SVM has two layer architecture, which consists of an input layer of d input variables and d_L LS-SVMs (in figure $d_L = 3$) of the hidden layer. In the hidden layer, each LS-SVM is trained to produce one output $f_\ell(\boldsymbol{x})$ for an input vector $\boldsymbol{x}$. For computing the hidden layer representation $f_\ell(\boldsymbol{x}_i)$ we use as follows:

$$f_\ell(\boldsymbol{x}_i) = \sum_{j=1}^{n} K_1(\boldsymbol{x}_i, \boldsymbol{x}_j)\alpha_{j\ell} + b_\ell, \ \ell = 1, \cdots, d_L \tag{3.1}$$

where $\alpha_{j\ell}$ is Lagrange multiplier, b_ℓ is a bias and $K_1(\cdot, \cdot)$ is a kernel function used for the ℓth LS-SVM of the hidden layer, respectively. To produce different outputs of each hidden layer LS-SVM, the initial reponses are constructed as follows:

$$y_{\ell i} = y_i + \gamma_{\ell i}, \ i = 1, \cdots, n, \ \ell = 1, \cdots, d_L, \tag{3.2}$$

where $\gamma_{\ell i}$ is a some random value in $(-\gamma, \gamma)$ with a metaparameter $\gamma > 0$ as in Wiering and Schomaker (2014).

Here $\boldsymbol{f}_\ell = (f_\ell(\boldsymbol{x}_1), \cdots, f_\ell(\boldsymbol{x}_n))'$ can be written as the linear combination of the response vector of the ℓth LS-SVM of the hidden layer as follows:

$$\boldsymbol{f}_\ell = (f_\ell(\boldsymbol{x}_1), \cdots, f_\ell(\boldsymbol{x}_n))' = \boldsymbol{H}_\ell \boldsymbol{y}_\ell \tag{3.3}$$

where $\boldsymbol{H}_\ell = (\boldsymbol{K}_1, \boldsymbol{1})\boldsymbol{H}_0$ with the $n \times n$ kernel matrix $\boldsymbol{K}_1$ and $\boldsymbol{H}_0$ in (2.4), and $\boldsymbol{y}_\ell = (y_{\ell 1}, \cdots, y_{\ell n})'$.

Finally, there is a main LS-SVM, which learns to estimate the regression function $g(\boldsymbol{x})$ for an input vector $\boldsymbol{x}$. For computing the output of the whole system, the main LS-SVM maps the produced hidden layer outputs to the final output as follows:

$$g(\boldsymbol{x}_t) = \sum_{j=1}^{n} K_2(\boldsymbol{f}(\boldsymbol{x}_t), \boldsymbol{f}(\boldsymbol{x}_j))\alpha_j + b, \tag{3.4}$$

where $\boldsymbol{f}(\boldsymbol{x}_t) = (f_1(\boldsymbol{x}_t), \cdots, f_{d_L}(\boldsymbol{x}_t))'$, $\boldsymbol{f}(\boldsymbol{x}_j) = (f_1(\boldsymbol{x}_j), \cdots, f_{d_L}(\boldsymbol{x}_j))'$ is $d_L \times 1$ input vector for the main LS-SVM and $K_2(\cdot, \cdot)$ is a kernel function used in the main LS-SVM.

The objective function of the main LS-SVM for Lagrange multipliers and bias can be written as follows:

$$J(\boldsymbol{\alpha}, b) = \frac{1}{2}\sum_{ij,=1}^{n} \alpha_i\alpha_j K_2(\boldsymbol{f}(\boldsymbol{x}_i|\theta),\ \boldsymbol{f}(\boldsymbol{x}_j|\theta)) + \frac{1}{2C}\sum_{i=1}^{n}\alpha_i^2 - \sum_{i=1}^{n}\alpha_i y_i + b\sum_{i=1}^{n}\alpha_i \tag{3.5}$$

The optimal Lagrange multipliers and the estimated bias can be obtained from the linear equations as follows:

$$\begin{pmatrix} \boldsymbol{K}_2 + \boldsymbol{I}/C & \boldsymbol{1}_n \\ \boldsymbol{1}_n' & 0 \end{pmatrix}\begin{pmatrix} \boldsymbol{\alpha} \\ b \end{pmatrix} = \begin{pmatrix} \boldsymbol{y} \\ 0 \end{pmatrix}, \tag{3.6}$$

where K_2 is $n \times n$ kernel matrix with element $K_2(\boldsymbol{f}(\boldsymbol{x}_i|\theta), \boldsymbol{f}(\boldsymbol{x}_j|\theta))$ and $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_n)'$.

In this paper we use the radial basis function (RBF) kernels in both layers of the deep LS-SVM. For the main LS-SVM and LS-SVMs of the hidden layer, the RBF kernels are defined respectively as follows:

$$K_2(\boldsymbol{f}(\boldsymbol{x}_i), \boldsymbol{f}(\boldsymbol{x}_j)) = \exp\left(-\frac{1}{\sigma_2}\sum_{\ell=1}^{d_L}(f_\ell(\boldsymbol{x}_i) - f_\ell(\boldsymbol{x}_j))^2\right)$$

$$K_1(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{1}{\sigma_1}\sum_{k=1}^{d}(x_{ik} - x_{jk})^2\right),$$

where $\sigma_2 > 0$ and $\sigma_1 > 0$ are kernel parameters of RBF kernels.

The deep LS-SVM updates the input variables of the main LS-SVM (outputs of LS-SVMs of the hidden layer) with a backpropagation algorithm of Rumelhart *et al.* (1986) as follows:

$$f_\ell^{(new)}(\boldsymbol{x}_i) = f_\ell(\boldsymbol{x}_i) - \eta\frac{\partial J}{\partial f_\ell(\boldsymbol{x}_i)},$$

where $\eta > 0$ is a learning rate and

$$\frac{\partial J}{\partial f_\ell(\boldsymbol{x}_i)} = -\frac{\alpha_i}{\sigma_2}\sum_{j=1}^{n}\alpha_j(f_\ell(\boldsymbol{x}_i) - f_\ell(\boldsymbol{x}_j))K_2(f_\ell(\boldsymbol{x}_i), f_\ell(\boldsymbol{x}_j)). \tag{3.7}$$

The updated response vector for the ℓth LS-SVM of the hidden layer is obtained as follows:

$$\boldsymbol{y}_\ell^{(new)} = \boldsymbol{H}_\ell^{-1}\boldsymbol{f}_\ell^{(new)}, \tag{3.8}$$

where $\boldsymbol{f}_\ell^{(new)} = (f_\ell^{(new)}(\boldsymbol{x}_1), \cdots, f_\ell^{(new)}(\boldsymbol{x}_n))'$ and $\boldsymbol{H}_\ell$ is the hat matrix obtained in (3.3).

Thus the learning algorithm of the deep LS-SVM is given as follows:

(i) Train LS-SVMs of the hidden layer with $\{(\boldsymbol{x}_i, y_{\ell i})\}_{i=1}^n$, $\ell = 1, \cdots, d_L$, using the prespecified values of the penalty parameters and the kernel parameters, where $y_{\ell i}$ is the initially perturbed response constructed by (3.2).

(ii) Train the main LS-SVM with $\{((f_1(\boldsymbol{x}_i), \cdots, f_{d_L}(\boldsymbol{x}_i)), y_i)\}_{i=1}^n$ using the prespecified values of the penalty parameters and the kernel parameters.

(iii) Use backpropagation algorithm to update $\boldsymbol{y}_\ell$, $\ell = 1, \cdots, d_L$, for LS-SVMs of the hidden layer by (3.7) and (3.8).

(iv) Train LS-SVMs of the hidden layer with $\{(\boldsymbol{x}_i, f_\ell(\boldsymbol{x}_i)\}_{i=1}^n$, $\ell = 1, \cdots, d_L$.

(v) Train the main LS-SVM with $\{((f_1(\boldsymbol{x}_i), \cdots, f_{d_L}(\boldsymbol{x}_i)), y_i)\}_{i=1}^n$.

(vi) Reiterate (iii)$\sim$(v) until the maximal number of epochs or $\sum_{i=1}^n (y_i - g(\boldsymbol{x}_i))^2$ <tol.

4. Numerical Studies

We performed experiments on regression problems with 5 regression datasets taken from Bilkent University Function Approximation Repository (`http://funapp.cs.bilkent.edu.tr/datasets`) and UCI Machine Learning Depository (`http://archive.ics.uci.edu/ml`) to compare the deep LS-SVM to the standard LS-SVM and also to MNN. Baseball dataset, Diabetes dataset and Machine-CPU dataset were from taken Bilkent University Function Approximation Repository, and Concrete strength dataset and GPS trajectories dataset were taken from UCI Machine Learning Depository. The size of dataset ranges from 43 to 337, and the number of input variables is between 2 and 16. The responses and input variables of each dataset were normalized using the min-max approach. Each dataset is split into 67% training dataset and 33% test dataset. We replicate the above process 100 times. The initially perturbed responses are constructed as follows:

$$y_{\ell i} = y_i + \gamma_{i\ell},\ i = 1, \cdots, n,\ \ell = 1, \cdots, d,$$

where $\gamma_{i\ell}$ is generated from a uniform distribution $U(\gamma, -\gamma)$ with γ is the standard deviation of y_i's.

For the deep LS-SVM, we set the number of LS-SVMs of the hidden layer equal to the number of input variables of each dataset. And for MNN we set the number of hidden layers equal to one and the number nodes in the hidden layer equal to the number of input variables of each dataset. In the standard LS-SVM the optimal values of the penalty parameter and the bandwidth parameter are obtained from training dataset by GCV function (2.7). In the deep LS-SVM, 10-fold cross validation method is used to obtain the optimal values of the penalty parameters and the bandwidth parameters, the learning rate, the maximal number of epochs, tolerance in the output layer using the greed search.

Table 4.1 shows the averages and standard errors of 100 mean squared errors and standard errors of LS-SVM, the deep LS-SVM, and MNN on 100 test datasets of 5 datasets. From the table we can see that the deep LS-SVM significantly outperforms the other methods on 4 datasets ($p < 0.0001$) and only performs worse than the standard LS-SVM on the Concrete dataset.

The average gain over all datasets is 21.0% error reduction to the standar LS-SVM, and 46.4% to MNN.

Table 4.1 The averages of mean squared errors by LS-SVM, the deep LS-SVM and MNN on regression datasets (standard error in parenthesis)

Dataset	size	LS-SVM	Deep LS-SVM	MNN
Baseball	337×16	0.0199 (0.0004)	**0.0133** (0.0003)	0.0193 (0.0005)
Concrete	93×7	**0.0031** (0.0003)	0.0038 (0.0001)	0.0080 (0.0007)
Diabetes	43×2	0.0362 (0.0029)	**0.0342** (0.0016)	0.0543 (0.0065)
GPS	163×7	0.0079 (0.0009)	**0.0073** (0.0006)	0.0154 (0.0012)
Machine	139×7	0.0037 (0.0005)	**0.0007** (0.0003)	0.0017 (0.0003)

5. Conclusions

Through the examples we showed that the deep LS-SVM shows the good results, which is simple modeling of the regression problems. In future work, we study the deep LS-SVM for the classification problems in which the initial response vectors for LS-SVMs of the hidden layer are to be reconsidered.

References

Bengio, Y. and Le Cun, Y. (2007). Scaling learning algorithms towards AI. In *Large Scale Kernel Machines*, edited by Bottou, L., Chapelle, O., De Coste, D., and Weston, J., MIT Press, Cambridge.

Cho, Y. and Saul, S. K. (2009). Kernel methods for deep learning. *Advances in Neural Information Processing Systems*, **22**, 342-350.

Hwang, C. (2014). Support vector quantile regression for autoregressive data. *Journal of the Korean Data & Information Science Society*, **25**, 1539-1547.

Hwang, C. (2015). Partially linear support vector orthogonal quantile regression with measurement errors. *Journal of the Korean Data & Information Science Society*, **26**, 209-216.

Hwang, C. (2016). Multioutput LS-SVR based residual MCUSUM control chart for autocorrelated process. *Journal of the Korean Data & Information Science Society*, **27**, 523-530.

Li, D., Tian, Y. and Xu, H. (2014). Deep twin support vector machine. In *Proceedings of IEEE International Conference on Data Mining Workshop*, 65-73, IEEE, Shenzhen, China.

Mercer, J. (1909). Functions of positive and negative type and their connection with theory of integral equations. *Philosophical Transactions of Royal Society A,* **209**, 415-446.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986). Learning internal representations by error propagation. *Nature*, **323**, 533-536.

Seok, K. (2015). Semisupervised support vector quantile regression. *Journal of the Korean Data & Information Science Society*, **26**, 517-524.

Shim, J. and Seok, K. (2014). A transductive least squares support vector machine with the difference convex algorithm. *Journal of the Korean Data & Information Science Society*, **25**, 455-464.

Suykens, J. A. K. and Vanderwalle, J. (1999). Least square support vector machine classifier. *Neural Processing Letters*, **9**, 293-300.

Suykens, J. A. K., Vandewalle, J. and DeMoor, B. (2001). Optimal control by least squares support vector machines. *Neural Networks*, **14**, 23-35.

Vapnik, V. N. (1995). *The nature of statistical learning theory*, Springer, New York.

Wahba, G. (1990). *Spline models for observational data*, CMMS-NSF Regional Conference Series in Applied Mathematics, **59**, SIAM, Philadelphia.

Wiering, M. A. and Schomaker, L. R. B. (2014). Multi-layer support vector machines. In *Regularization, Optimization, Kernels, and Support Vector Machines*, edited by Suykens, Signoretto and Argyriou, Chapman & Hall/CRC, Boca Raton.

Zhuang, Z., Tsang, I. W. and Choi, S. C. H. (2011). Two-layer multiple kernel learning. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, 909-917.