Variance Minimization Least Squares Support Vector Machines for Time Series Analysis

Róbert Ormándi

MTA-SZTE, Research Group on Artificial Intelligence,
6720 Szeged, Aradi Vértanúk tere 1., Hungary
ormandi@inf.u-szeged.hu

Abstract

Here we propose a novel machine learning method for time series forecasting which is based on the widely-used Least Squares Support Vector Machine (LS-SVM) approach. The objective function of our method contains a weighted variance minimization part as well. This modification makes the method more efficient in time series forecasting, as this paper will show. The proposed method is a generalization of the well-known LS-SVM algorithm. It has similar advantages like the applicability of the kernel-trick, it has a linear and unique solution, and a short computational time, but can perform better in certain scenarios. The main purpose of this paper is to introduce the novel Variance Minimization Least Squares Support Vector Machine (VMLS-SVM) method and to show its superiority through experimental results using standard benchmark time series prediction datasets.

1 Introduction

The problem of time series forecasting is a challenge in many fields and it has attracted great interest recently, finding applications in domains as diverse as medicine, finance, entertainment, and industry. It is a complex topic, which has actually several points in common with function approximation problems. For this reason it is natural to use regression methods [12, 9] taken from the field of machine learning for time series forecasting. These methods can be used without any modifications and often give a good performance.

These regression methods usually apply the mean squared error (MSE) minimization in some way. For example, the Artificial Neural Network (ANN) approach applies empirical risk minimization, which also minimizes the MSE. In another example, the Support Vector Machine (SVM) approach [2, 3] uses the structural risk minimiza-

tion, which seeks to minimize an upper bound of the generalization error. It contains an empirical risk term and a confidence interval term that depends on the Vapnik-Chervonenkis dimension. In this case the empirical risk minimizes the MSE as well. Although minimizing the MSE implicitly minimizes the variance too, in some cases the variance of the error can be relatively large. Training models that have a small prediction error variance are most preferred in time series prediction, especially when the process that generates the time series is unsteady (or chaotic).

For this reason we propose here a regression method which is called the Variance Minimization Least Squares Support Vector Machines (VMLS-SVM) approach. VMLS-SVM is a generalization of LS-SVM [14], the objective function of LS-SVM being extended with a weighted variance minimization term. With this modification, we can adjust the weight of the variance of the error in the feature space, which can give a smoother fitting in the input space. The source code of our method is freely available as a WEKA classifier at (<http://www.inf.u-szeged.hu/~ormandi/vmlssvm/>)¹.

The main objectives of this paper are, firstly, to give a good theoretical description of the VMLS-SVM approach, and, secondly, to examine the feasibility of applying VMLS-SVM in time series forecasting by comparing it with certain other methods which are widely-used for time series forecasting. We selected some public time series benchmark databases, and used their evaluation method to get comparable results.

This article is structured as follows. The next section provides an overview of related work, including the basic theory of LS-SVM. Section 3 introduces and solves the optimization task of VMLS-SVM. Section 4 contains our experimental data and results. Then we round off the paper with some conclusions and suggestions for future study.

¹In our implementation, we used the LAPACK linear equation solver.

2 Related Work

Times series can be modelled by a large class of models which can be divided into two main subclasses, namely linear models and non-linear models. The linear models (such as ARMA [1]) have been extensively investigated and are quite advanced, but they only work well when the data dependency is linear.

In the 1990s the non-linear models used for time series forecasting were chiefly the ANN [8] and the Radial Basis Function Neural Network (RBF) [11] approaches. Although these networks work well, they have some inherent drawbacks, such as the problem of multiple local minima, the choice of number of hidden units and the danger of overfitting. These problems have made the use of neural networks less popular in the field of time series forecasting.

SVMs, based on the unique theory of the structural risk minimization principle [2], usually achieves a better generalization ability than traditional neural networks, which implement the empirical risk minimization principle. Another advantage of the SVM approach is the applicability of the *kernel-trick*, which makes it more practicable. Yet another key feature of this approach is that training an SVM is equivalent to solving a linearly constrained quadratic programming (QP) problem [2, 3], hence the solution of the optimization task of SVM is always unique and globally optimal, but the computation time is often huge because the size of the problem is $O(n^2)$, where n is the number of training samples. LS-SVM, which is a new kind of SVM approach, does not have this latter problem.

In the next part we will provide an overview of the LS-SVM method.

2.1 LS-SVM

This approach is a reformulation of the principles of SVM, which applies equality instead of inequality constraints. Furthermore, LS-SVM uses the least squares loss function instead of the ϵ -insensitive loss function. In this way, the solution follows from a linear KKT [14] system instead of a computationally hard QP Problem [2].

Now suppose that we are given a training set that has the following form:

$$S = \{(x_1, y_1), \dots, (x_n, y_n)\} \subseteq \mathbb{R}^d \times \mathbb{R}. \quad (1)$$

With LS-SVM, one considers the following optimization problem (primal form):

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + \gamma \frac{1}{2} \sum_{i=1}^n e_i^2 \\ \text{s.t.} \quad & \forall 1 \leq i \leq n : y_i = \mathbf{w}^T \varphi(x_i) + b + e_i, \end{aligned} \quad (2)$$

where $\varphi : \mathbb{R}^d \rightarrow \mathbb{R}^m$ is the feature mapping, γ is a trade-off parameter between generalization and the error minimization weighting, and $\forall 1 \leq i \leq n : e_i \in \mathbb{R}$ are the error variables. Using the dual form of this problem, we get that the optimal solution has to be satisfied the following constraints:

$$\frac{\partial \mathbf{L}(\mathbf{w}, b, e, \alpha)}{\partial \mathbf{w}} = 0 \rightarrow \mathbf{w} = \sum_{i=1}^n \alpha_i \varphi(x_i), \quad (3)$$

$$\frac{\partial \mathbf{L}(\mathbf{w}, b, e, \alpha)}{\partial b} = 0 \rightarrow \sum_{i=1}^n \alpha_i = 0, \quad (4)$$

$$\frac{\partial \mathbf{L}(\mathbf{w}, b, e, \alpha)}{\partial e_i} = 0 \rightarrow \alpha_i = \gamma e_i, \forall 1 \leq i \leq n, \quad (5)$$

$$\begin{aligned} \frac{\partial \mathbf{L}(\mathbf{w}, b, e, \alpha)}{\partial \alpha_j} = 0 \rightarrow \mathbf{w}^T \varphi(x_j) + b + e_j &= y_j, \\ \forall 1 \leq j \leq n. \end{aligned} \quad (6)$$

After the elimination of variables $\mathbf{w}$ and $\mathbf{e}$, we get a system of equations with variables b and α . Using the solutions of the equation system (α_{opt} and b_{opt}), we can write the regression function in the following form: $f(x) = \sum_{i=1}^n \alpha_{opt,i} K(x_i, x) + b_{opt}$.

This formulation is quite simple and the method has all of the advantages of SVM, like the applicability of the kernel-trick, and it has a unique solution. But in the case of LS-SVM the solution comes from solving a linear system of equations, not a quadratic one. Nevertheless, in spite of these advantages LS-SVM has also one slight drawback. While SVM chooses some objects of the training data (the support vectors) that are important in the regression, LS-SVM uses all the training data to produce the result. Sparseness can also be introduced with LS-SVM by applying a pruning method to select the most important objects [6].

In the next section we will define and solve the optimization task of VMLS-SVM and later we will furnish examples which demonstrate the usefulness of our new method.

3 VMLS-SVM

Essentially, VMLS-SVM is an extension of LS-SVM, where the objective function of optimization contains a weighted variance minimization part as well. Hence it is capable of producing a smoother fit than LS-SVM, in terms of the variance of the prediction error. The weight of the variance term can be adjusted via a trade-off parameter. This parameter and the smoothness of fitting plays an important role in time series forecasting. Our new method has all the advantages of LS-SVM, but it is more general.

Now let us describe this method in detail. Suppose that we have a training set S like that defined in Eq. (1).

Next, let us express the optimization problem of VMLS-SVM in the following way:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + \gamma \frac{1}{2} \sum_{i=1}^n e_i^2 + \delta \frac{1}{2} \sum_{i=1}^n \left(e_i - \frac{1}{n} \sum_{j=1}^n e_j \right)^2 \\ \text{s.t. } \quad & \forall 1 \leq i \leq n : y_i = \mathbf{w}^T \varphi(x_i) + b + e_i, \end{aligned} \quad (7)$$

where φ , γ and $\forall 1 \leq i \leq n : e_i$ are the same as those of LS-SVM and the δ parameter is a trade-off between generalization and variance minimization weighting.

As before, one solves the dual problem:

$$\begin{aligned} \mathbf{L}(\mathbf{w}, b, \mathbf{e}, \alpha) = & \frac{1}{2} \|\mathbf{w}\|^2 + \frac{\gamma}{2} \sum_{i=1}^n e_i^2 + \frac{\delta}{2} \sum_{i=1}^n \left(e_i - \frac{1}{n} \sum_{j=1}^n e_j \right)^2 - \sum_{i=1}^n \alpha_i \{ \mathbf{w}^T \varphi(x_i) + b + e_i - y_i \}. \end{aligned} \quad (8)$$

The optimal solution of this function depends on partial derivatives like that in Eqs. (3)-(6). But here the following partial derivative has to be changed:

$$\frac{\partial \mathbf{L}(\mathbf{w}, b, \mathbf{e}, \alpha)}{\partial e_i} = 0 \rightarrow \alpha_i = (\gamma + \delta) e_i - \frac{\delta}{n} \sum_{j=1}^n e_j, \quad (9)$$

where $1 \leq i \leq n$.

From Eq. (4) and (9) we get the following:

$$0 = \sum_{i=1}^n \alpha_i = (\gamma + \delta) \sum_{i=1}^n e_i - \delta \sum_{j=1}^n e_j = \gamma \sum_{i=1}^n e_i. \quad (10)$$

This will be described in the first equation of the set of equations below (see Eq. (12)).

For each $1 \leq j \leq n$, replacing $\mathbf{w}$ in (6) with the rhs of (3), then replacing α_i with the rhs of (9), and some algebraic manipulation, we get:

$$\sum_{i=1}^n \underbrace{\left[(\gamma + \delta) K(x_j, x_i) - \frac{\delta}{n} \sum_{k=1}^n K(x_k, x_j) \right]}_{\Omega_{j,i}} e_i + b + e_j = y_j. \quad (11)$$

This will be described in the j th equation of the set of equations below (see Eq. (12)).

Hence using Eqs. (10) and (11) and eliminating the variables $\mathbf{w}$ and α , we get the following set of equations:

$$\begin{pmatrix} 0 \\ \mathbf{1} \end{pmatrix} \begin{vmatrix} \gamma^T \\ \Omega + \mathbf{I} \end{vmatrix} \begin{pmatrix} b \\ \mathbf{e} \end{pmatrix} = \begin{pmatrix} 0 \\ \mathbf{y} \end{pmatrix}, \quad (12)$$

where $\mathbf{1}^T = (1, \dots, 1) \in \mathbb{R}^{1 \times n}$, $\mathbf{y} = (y_1, \dots, y_n)^T$, $\gamma^T = (\gamma, \dots, \gamma) \in \mathbb{R}^{1 \times n}$ and $\forall 1 \leq i, j \leq n : \Omega_{j,i}$ is defined in Eq. (11).

Now the regression function becomes

$$f(x) = \sum_{i=1}^n \left[(\gamma + \delta) e_{opt,i} - \frac{\delta}{n} \sum_{j=1}^n e_{opt,j} \right] K(x_i, x) + b_{opt}, \quad (13)$$

where $\mathbf{e}_{opt}$ and b_{opt} are the solutions of Eq. (12). In the next part we will present our experimental data and results.

4 Experimental results

Now we would like to explain how our method was applied to time series forecasting and how we chose the parameters of the learning methods during the test phase, and then summarize our experimental results in time series forecasting. In all of our tests we experimented with three type of kernels for the LS-SVM and VMLS-SVM. These were the following: Polynomial Kernel, RBF kernel, Alignment kernel for time series [5].

4.1 How our method is applied to time series forecasting

A time series is a sequence of vectors, $x(t)$, $t = 0, 1, \dots$, where t represents the elapsed time. For the sake of simplicity, we will consider only sequences of scalars here, although each technique can be readily generalised to vector series. For us, time series forecasting means predicting the value of a variable from a series of observations of that variable up to that time. Usually we have to forecast the value of $x(t+k)$ from the following series of observations: $x(t), \dots, x(t-N+1)$. Formally this can be stated as: find a prediction function $f_k : \mathbb{R}^N \rightarrow \mathbb{R}$ to get an estimate of x at time $t+k$, from the N time steps back from time t . Hence $x(t+k) = f_k(x(t), \dots, x(t-N+1))$.

The observations $(x(t), \dots, x(t-N+1))$ constitute a *window* and N is referred to as the *window size*. The technique which produces all the window-value pairs as training samples of a regression method is called the *sliding window technique*. This technique slides a window of length $N+1$ over the full time series and generates the training samples in the following form for every possible value of i : $(x_i, y_i) = (x(t+i-N+1), \dots, x(t+i), x(t+i+k))$, where $0 \leq i, x_i \in \mathbb{R}^N$ and $y_i \in \mathbb{R}$.

Table 1. MSEs of forecasting of various configurations of different methods on the missing values.

Method	Value of δ	Value of γ	Kernel	Window size	MSE
LS-SVM	-	1,000	Poly, E = 12	60	534.33
LS-SVM	-	1,000	RBF, G = 0.001	60	412.27
LS-SVM	-	1,000	Align, $\sigma = 12$	60	432.66
VMLS-SVM	100	0.1	Poly, E = 12	60	501.25
VMLS-SVM	100	0.1	RBF, G = 0.001	60	421.34
VMLS-SVM	100	0.1	Align, $\sigma = 12$	60	437.45
VMLS-SVM	1,000	0.01	Poly, E = 12	80	480.11
VMLS-SVM	1,000	0.01	RBF, G = 0.001	80	287.36
VMLS-SVM	1,000	0.01	Align, $\sigma = 12$	80	312.28
VMLS-SVM	10,000	0.01	Poly, E = 12	80	498.62
VMLS-SVM	10,000	0.01	RBF, G = 0.001	80	386.25
VMLS-SVM	10,000	0.01	Align, $\sigma = 12$	80	408.17

In order to perform time series forecasting using the VMLS-SVM method, we will employ the sliding window technique, which produces an initial training database. Afterwards, our system applies a normalization on the input databases.

4.2 Parameter selection

In the field of machine learning, determining the hyper-parameters of a learning method is important and if they are improperly chosen these parameters can induce a poor performance.

To avoid this, in our first tests on the first two benchmarks, we applied a simulated annealing based [10] optimization method, which optimized the parameters of the underlying learning method. This parameter selection approach can be viewed as a function minimizing method, where the input of the objective function is the parameter of the underlying learner and the value of the function is the aggregated error of the underlying method on a fixed optimization set. The underlying method were trained on a different, but also fixed training set using the input of the objective function as parameters. The applied error measure depends on the database evaluation metric i.e. it is always the same as the error measure of the evaluation process. The stopping criterion of the parameter selection method was determined by visual inspection, i.e. when the error was quite small and it did not decrease, we stopped the process.

Using this optimization technique, we get a sequence of parameter sets, which was provided by the parameter optimization method in the course of learning, revealed the trend of a correct parameter setup. Afterwards, we carried out some manually parameterized tests, using experiments from the automatic parameter selection. These tests were evaluated on the evaluation set, which is completely different from the training and optimization sets.

4.3 Results on CATS benchmark

The CATS benchmark [7] is a public available database for time series forecasting. It was made for a competition organised during the IJCNN'04 conference in Budapest. This artificial time series has 5,000 data points, among which 100 are missing. The missing values are divided in 5 blocks. The goal of the competition was to predict the missing values. Twenty-four teams participated and seventeen of them uploaded acceptable predictions. The evaluation metric was the mean squared error metrics: $MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$, where n is the number of prediction, $\hat{y}_i$ is the i th prediction and y_i is the i th expected value.

The method that was used by the winner [13] of the competition is divided in two parts: the first sub-method provides a short-term prediction and the second sub-method provides a long-term one.

In order to make a comparison, we decided to use the LS-SVM and our VMLS-SVM in the same way. We defined twenty different prediction functions. In this setup the first prediction function (f_1) was used to predict the first missing value, the second prediction function (f_2) was used to predict the second missing value, and so on. Each prediction function used the same parameter setting, which was provided² by the parameter optimization method in the early tests. This optimization process used an optimization set to test the goodness of different parameter setups. This optimization set was generated in the following way: from the end of each training block we split the last 20 values. Afterwards, using the experiments of the automatic parameter optimization, we carried out some tests, using the fixed parameterized prediction method. In this phase, for each prediction function we trained its own LS-SVM or VMLS-SVM learner with the same parameter values on the full training set. The overall results are presented in Table 1.

²Here we changed just the values of δ , γ and the window size.

Table 2. Comparision between the main methods on CATS benchmark.

Method	MSE
VMLS-SVM (RBF)	287
VMLS-SVM (Align)	312
Kalman Smoother	408
Recurrent Neural Networks	441
Competitive Associative Net	502
Weighted Bidirectional Multi-stream Extended Kalman Filter	530
SVCA Model	577
MultiGrid -Based Fuzzy Systems	644
⋮	⋮

In Table 1, Poly means Polynomial kernel and E denotes its parameter, the exponent. RBF means the RBF kernel and G denotes its parameter, the gamma parameter. Similarly, Align means Aligment kernel and its parameter is denoted by σ . Here MSE means the mean squared error, computed on the evaluation set. As can be seen in Table 1, the VMLS-SVM method with the RBF kernel was the most effective for predicting the 100 missing values and VMLS-SVM models are consequently more accurate than LS-SVM models with a similar kernel function.

The weighting of the variance minimizing term helped the VMLS-SVM achieve a better performance on the evaluation set, but where the weight exceeded a threshold, the error on the evaluation set started to increase. It means that the overweighting of the variance term caused overfitting. But in this case the error on the optimization set started to increase, while the training error decreased. Hence using the parameter selection, we could have chosen the best generalization model on the given task, which was about $\delta = 1000$, $\gamma = 0.01$ and Window size= 80.

Table 2 shows a comparison between our method (labeled as VMLS-SVM) and the best results reported for this series in the competition³. This table contains just the best results of our models, using $\delta = 1000$, $\gamma = 0.01$ and Window size= 80 parameter values.

4.4 Results on dataset of NN3 Neural Forecasting Competition

To further investigate the capability of VMLS-SVM, we carried out a comparison with the ϵ -insensitive SVM regression method. Hence we used our method to forecast the reduced subset of the time series of the NN3 Artificial Neural Network and Computational Intelligence Forecasting Competition, which has reported results using an ϵ -insensitive SVM regression approach [4].

The NN3 dataset contains 11 time series. The averaged length of the time series is 124⁴, and there is no domain knowledge about the time series. In [4], the last 18 values of each time series were predicted. The evaluation metric was the symmetric mean absolute percent error (SMAPE = $\frac{1}{n} \sum_{i=1}^n \frac{|y_i - \hat{y}_i|}{(y_i + \hat{y}_i)/2} \cdot (100)$), where n is the number of prediction, $\hat{y}_i$ is the i th prediction and y_i is the i th expected value.

For each time series, we defined 18 different prediction functions. Each prediction function used the same parameter setting, but each time series used a different parameter setting, which was determined by parameter optimization. To perform the parameter optimization, we made three subsets from each time series e.g. the training set, the optimization set and the evaluation set. The each evaluation set contains the last 18 values from the corresponding time series, and each optimization set is made up from the last 18 values of the corresponding time series without the evaluation set. For each set triplet, we made a hyper-parameter optimization for the VMLS-SVM based prediction functions, using the training and optimization sets. We adjusted only the δ and γ parameters during the optimization and used a fixed window size= 12 and RBF kernel with parameter $G = 0.001$. These optimizations determined well generalization parameter sets for each VMLS-SVM predictor. Afterward, we used the correct parameter setups; we trained each VMLS-SVM prediction function on the training and optimization sets and evaluated them on the evaluation set as test set. Table 3 lists the results of comparison between our method and those reported for this series in [4].

The mean SMAPE of our method is 3.6757 and the corresponding value for the ϵ -insensitive SVM regression method is 5.05. Hence we can say that our method using VMLS-SVM methods and parameter optimization achieved a significant better mean performance. As can be seen, our method achieved a lower SMAPE value on each time series except for the first one. This means that the hyper-parameter

³The full version of Table 2 is available at <http://www.inf.u-szeged.hu/~ormandi/vmlssvm>.

⁴The length of each time series is given in Table 3.

Table 3. SMAPEs of each time series using different regression methods.

Time series	Length	SMAPE(VMLS-SVM)	SMAPE(ϵ -insensitive SVR)
1	126	0.5218	0.49
2	126	1.7432	2.5
3	126	8.7458	12.46
4	115	4.6944	5.4
5	126	0.3091	0.7
6	126	1.8892	3.38
7	126	0.9871	1.21
8	115	8.1293	9.35
9	123	1.1947	2.54
10	126	8.4968	12.7
11	126	3.7217	4.82

optimization method can tackle overfitting in most cases; but since this overfitting-detection is only a heuristic that is based on the optimization set, it can fail in a few cases.

We tested our method on a widely-recognised benchmark, the D data series taken from the Santa Fe competition⁵. The results show that our methods can also achieve a better performance than others in this task.

5 Conclusions

In this article we presented a novel extension for LS-SVM based on the weighting the variance of the error. After presenting the basic theory of the method, we tested the method on publicly available benchmarks. The test results show that the proposed method can indeed achieve a higher efficiency on the two different, widely-recognised benchmarks than other methods here.

On the other hand, there are some issues that need to be investigated like a more in-depth exploration of the main idea e.g. whether we can apply such a variance minimization term in the ϵ -insensitive SVM regression. We also would like to test our method on standard regression benchmarks. This is what we plan to do in the near future.

6 Acknowledgment

This work was supported in part by the NKTH grant of National Technology Programme 2005 (project codename RET-07/2005) of the Hungarian government.

References

- [1] G. E. P. Box and G. Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day, Incorporated, 1990.

⁵The description of database and the results can be viewed at <http://www.inf.u-szeged.hu/~ormandi/vmlssvm>.

- [2] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.
- [3] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, March 2000.
- [4] S. F. Crone and S. Pietsch. A naïve support vector regression benchmark for the nn3 forecasting competition. In *IJCNN*, pages 2454–2459, 2007.
- [5] M. Cuturi, J.-P. Vert, O. Birkenes, and T. Matsui. A kernel for time series based on global alignments. *CoRR*, abs/cs/0610033, 2006. informal publication.
- [6] B. J. de Kruif and T. de Vries. Pruning error minimization in least squares support vector machines. *IEEE Transactions on Neural Networks*, 14(3):696–702, 2003.
- [7] A. L. Erkki. Time series prediction competition: The cats benchmark, 2004.
- [8] J. Faraway and C. Chatfield. Time series forecasting with neural networks: A case study, 1995.
- [9] T. V. Gestel, J. Suykens, D. Baestaens, A. Lambrechts, G. Lanckriet, B. Vandaele, B. D. Moor, and J. Vandewalle. Financial time series prediction using least squares support vector machines within the evidence framework, 2001.
- [10] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, Number 4598, 13 May 1983, 220, 4598:671–680, 1983.
- [11] V. M. Rivas, J. J. Merelo, P. A. Castillo, M. G. Arenas, and J. G. Castellano. Evolving rbf neural networks for time-series forecasting with evrbf. *Inf. Sci. Inf. Comput. Sci.*, 165(3-4):207–220, 2004.
- [12] G. Rubio, H. Pomares, L. J. Herrera, and I. Rojas. Kernel methods applied to time series forecasting. In F. Sandoval, A. Prieto, J. Cabestany, and M. G. a, editors, *IWANN*, volume 4507 of *Lecture Notes in Computer Science*, pages 782–789. Springer, 2007.
- [13] S. Särkkä, A. Vehtari, and J. Lampinen. Cats benchmark time series prediction by kalman smoother with cross-validated noise density. *Neurocomputing*, 70(13-15):2331–2341, 2007.
- [14] J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.