Variance Minimization Least Squares Support Vector Machines for Time Series Analysis

Submitted for Blind Review

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 problem, which has actually several points in common with function approximation problems. For this reason it is natural to use regression methods [24, 25, 19, 13, 5] 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 [26, 4, 6] uses the structural risk minimization, 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 too. 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 [22, 23], 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 and a smaller error for time series forecasting. The source code of our method is freely available as a WEKA [29] classifier at (URL was removed through blind reviewing).

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 as well. 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, then for demonstration purposes we present a simple example to show how our method works. Section 4 contains our experimental data and results, and the technique used for data preprocessing and evaluation metrics are also given here. Then we round off the paper with some conclusions and suggestions for future study.

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

\[\text{URL was removed through blind reviewing}\]
as ARMA [3]) have been extensively investigated and are quite advanced, but they only work well when the data dependence is linear.

In the 1990s the non-linear models used for time series forecasting were chiefly the ANN [12] and the Radial Basis Function Neural Network (RBF) [18] 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 [26, 4], 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 [26, 4, 6], hence the solution of the optimization problem (primal form):

\[\text{min}_{w, b, \alpha} \frac{1}{2} ||w||^2 + \frac{\gamma}{2} \sum_{i=1}^{n} e_i^2 \]

subject to \( \forall i \leq n : y_i = w^T \varphi(x_i) + b + e_i, \)

where \( \varphi : \mathbb{R}^d \rightarrow \mathbb{R}^m \) is the feature mapping, \( \gamma \) 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.

In this case the regression function is given by:

\[f(x) = w_{opt}^T \varphi(x) + b_{opt},\]

where \( w_{opt} \) and \( b_{opt} \) are the solutions of the optimization problem defined in (2).

Actually it is easier to solve the dual problem rather than the optimization problem defined in (2). Hence expressed in Lagrangian form,

\[L(w, b, e, \alpha) = \frac{1}{2} ||w||^2 + \gamma \frac{1}{2} \sum_{i=1}^{n} e_i^2 - \sum_{i=1}^{n} \alpha_i (w^T \varphi(x_i) + b + e_i - y_i).\]

The optimal solution of this function can be obtained by demanding that the following conditions be satisfied:

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

\[\frac{\partial L(w, b, e, \alpha)}{\partial b} = 0 \rightarrow \sum_{i=1}^{n} \alpha_i = 0,\]

\[\frac{\partial L(w, b, e, \alpha)}{\partial e_i} = 0 \rightarrow \alpha_i = \gamma e_i,\]

where \( 1 \leq i \leq n, \)

\[\frac{\partial L(w, b, e, \alpha)}{\partial \alpha_j} = 0 \rightarrow w^T \varphi(x_j) + b + e_j = y_j,\]

where \( 1 \leq j \leq n. \)

From this we get the following set of equations, after the elimination of variables \( w \) and \( e: \)

\[
\begin{pmatrix}
0 \\
1^T \\
\Omega + \frac{1}{\gamma}
\end{pmatrix}
\begin{pmatrix}
b \\
\alpha
\end{pmatrix}
= \begin{pmatrix}
0 \\
y
\end{pmatrix},
\]

where \( y = (y_1, \ldots, y_n)^T, \alpha = (\alpha_1, \ldots, \alpha_n)^T, 1^T = (1, \ldots, 1) \in \mathbb{R}^{1 \times n} \) and \( \forall 1 \leq i, j \leq n : \Omega_{i,j} = \varphi(x_i)^T \varphi(x_j) = K(x_i, x_j). \)

In this case, the regression function becomes:

\[f(x) = \sum_{i=1}^{n} \alpha_{opt,i} \varphi(x_i)^T \varphi(x) + b_{opt} = \]

\[= \sum_{i=1}^{n} \alpha_{opt,i} K(x_i, x) + b_{opt}.\]

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 choses 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 [9, 16].

This formulation can also be shown to have a connection with regularization networks [11]. When no bias term is used in the LS-SVM formulation, as proposed in Kernel Ridge regression [21], the expressions in the dual space correspond to Gaussian Processes [28].

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 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 usual 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:

\[
\begin{align*}
\text{minimize}_w & \quad \frac{1}{2} \| w \|^2 + \gamma \frac{1}{2} \sum_{i=1}^{n} e_i^2 + \\
& \quad \delta \frac{1}{2} \sum_{i=1}^{n} \left( e_i - \frac{1}{n} \sum_{j=1}^{n} e_j \right)^2 \\
\text{subject to} & \quad \forall 1 \leq i \leq n : y_i = w^T \varphi(x_i) + b + e_i,
\end{align*}
\]

(11)

where \( \varphi, \gamma \) and \( \forall 1 \leq i \leq n : e_i \) are the same as those of LS-SVM and the \( \delta \) parameter is a trade-off between generalization and variance minimization weighting.

The regression function is the same as before, hence it is defined by Eq. (3).

As before, one solves the dual problem:

\[
L(w, b, e, \alpha) = \frac{1}{2} \| w \|^2 + \gamma \frac{1}{2} \sum_{i=1}^{n} e_i^2 + \\
\delta \frac{1}{2} \sum_{i=1}^{n} \left( e_i - \frac{1}{n} \sum_{j=1}^{n} e_j \right)^2 - \\
\sum_{i=1}^{n} \alpha_i \left( w^T \varphi(x_i) + b + e_i - y_i \right).
\]

(12)

The optimal solution of this function depends on partial derivatives like that in Eqs. (5)-(8). But here the following partial derivative has to be changed:

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

(13)

where \( 1 \leq i \leq n, \) since

\[
\frac{\partial \frac{1}{2} \delta \sum_{k=1}^{n} \left( e_k - \frac{1}{n} \sum_{j=1}^{n} e_j \right)^2}{\partial e_i} = \\
= \delta \sum_{k=1}^{n} \left( e_k - \frac{1}{n} \sum_{j=1}^{n} e_j \right) \frac{\partial \left( e_k - \frac{1}{n} \sum_{j=1}^{n} e_j \right)}{\partial e_i} = \\
= \sum_{k=1}^{n} \left( - \frac{\delta}{n} e_k + \frac{\delta}{n^2} \sum_{j=1}^{n} e_j \right) + \delta e_i - \frac{\delta}{n} \sum_{j=1}^{n} e_j = \\
= \delta e_i - \frac{\delta}{n} \sum_{j=1}^{n} e_j.
\]

(14)

From Eq. (6) and (13) we get the following:

\[
0 = \sum_{i=1}^{n} \alpha_i = (\gamma + \delta) \sum_{i=1}^{n} e_i - \delta \sum_{j=1}^{n} e_j = \gamma \sum_{i=1}^{n} e_i,
\]

(15)

This will be described in the first equation of the set of equations below (see Eq. (18)).

For each \( 1 \leq j \leq n, \) replacing \( w \) in (8) with the rhs of (5), and then replacing \( \alpha_i \) with the rhs of (13), we get:

\[
\sum_{i=1}^{n} \left( (\gamma + \delta) e_i - \frac{\delta}{n} \sum_{k=1}^{n} e_k \right) K(x_i, x_j) + b + e_j = y_j.
\]

(16)

After some algebraic manipulation, we arrive at the following:
\[
\sum_{i=1}^{n} \left( (\gamma + \delta) K(x_j, x_i) - \frac{\delta}{n} \sum_{k=1}^{n} K(x_k, x_j) \right) e_i_{\Omega_{j,i}} + b + e_j = y_j.
\]

(17)

This will be described in the \( j \)th equation of the set of equations below (see Eq. (18)).

Hence using Eqs. (15) and (17) and eliminating the variables \( w \) and \( \alpha \), we get the following set of equations:

\[
\begin{pmatrix}
0 \\
\gamma^T \\
\Omega + 1
\end{pmatrix}
\begin{pmatrix}
b \\
y
\end{pmatrix} = \begin{pmatrix}
0 \\
y
\end{pmatrix},
\]

where \( 1, y \) are the same as in Eq. (9), \( \gamma^T = (\gamma, \ldots, \gamma) \in \mathbb{R}^{1 \times n} \) and \( \forall 1 \leq i, j \leq n : \Omega_{j,i} \) is defined in Eq. (17).

Now the regression function becomes

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

(19)

In the next part we will present a simple example to show how our method works.

3.1 An Illustration of how our method works

To illustrate how the method works, we chose a simple database called CPU Performance\(^2\) from the UCI repository [1] and trained our system using different \( \gamma \) and \( \delta \) parameter values. What it predicts is shown in Fig. 1 and 2 below.

As can be seen in Fig. 1 and 2, when the \( \delta \) parameter is bigger in magnitude than \( \gamma \), the method minimizes the variance of the prediction error with a larger weight, and then we get a smoother fitting. But when we make the weight of the error variance minimizing term very large, this can cause overfitting. Hence we recommend that a parameter selection method be used beforehand to determine the right trade-off parameter value for the given optimization task.

In the following subsection, we will give an overview of how our approach can be applied in time series forecasting.

3.2 How our method is applied to time series forecasting

A time series is a sequence of vectors, \( x(t) \), \( t = 0, 1, \ldots \), 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: \( x(t), \ldots, x(t-N+1) \). Formally this can be stated as: find a function \( f_k : \mathbb{R}^N \rightarrow \mathbb{R} \) to obtain an estimate of \( x \) at time \( t+k \), from the \( N \) time steps back from time \( t \).

Hence

\[
x(t+k) = f_k(x(t), \ldots, x(t-N+1)).
\]

(20)

The observations \( x(t), \ldots, x(t-N+1) \) 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

\[\text{Figure 1. Predicted values for the CPU database between the 10th and 110th instances.}\]

\[\text{Figure 2. Predicted values for the CPU database between the 110th and 210th instances.}\]

\(^2\)We deleted the ‘vendor’ attribute to make the data consistent with our learning approach. This modified database is available in the database repositories of the WEKA distributions.
Table 1. MSEs of forecasting of various configurations of the LS-SVM and VMLS-SVM methods on the 100 missing values.

<table>
<thead>
<tr>
<th>Method</th>
<th>Value of δ</th>
<th>Value of γ</th>
<th>Kernel</th>
<th>Window size</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LS-SVM</td>
<td>-</td>
<td>1.000</td>
<td>Poly, E = 12</td>
<td>60</td>
<td>534.33</td>
</tr>
<tr>
<td>LS-SVM</td>
<td>-</td>
<td>1.000</td>
<td>RBF, G = 0.001</td>
<td>60</td>
<td>412.27</td>
</tr>
<tr>
<td>LS-SVM</td>
<td>-</td>
<td>1.000</td>
<td>Align, σ = 12</td>
<td>60</td>
<td>432.66</td>
</tr>
<tr>
<td>VMLS-SVM</td>
<td>100</td>
<td>0.1</td>
<td>Poly, E = 12</td>
<td>60</td>
<td>501.25</td>
</tr>
<tr>
<td>VMLS-SVM</td>
<td>100</td>
<td>0.1</td>
<td>RBF, G = 0.001</td>
<td>60</td>
<td>421.34</td>
</tr>
<tr>
<td>VMLS-SVM</td>
<td>100</td>
<td>0.1</td>
<td>Align, σ = 12</td>
<td>60</td>
<td>437.45</td>
</tr>
<tr>
<td>VMLS-SVM</td>
<td>1,000</td>
<td>0.01</td>
<td>Poly, E = 12</td>
<td>80</td>
<td>480.11</td>
</tr>
<tr>
<td>VMLS-SVM</td>
<td>1,000</td>
<td>0.01</td>
<td>RBF, G = 0.001</td>
<td>80</td>
<td>287.36</td>
</tr>
<tr>
<td>VMLS-SVM</td>
<td>1,000</td>
<td>0.01</td>
<td>Align, σ = 12</td>
<td>80</td>
<td>312.28</td>
</tr>
<tr>
<td>VMLS-SVM</td>
<td>10,000</td>
<td>0.01</td>
<td>Poly, E = 12</td>
<td>80</td>
<td>498.62</td>
</tr>
<tr>
<td>VMLS-SVM</td>
<td>10,000</td>
<td>0.01</td>
<td>RBF, G = 0.001</td>
<td>80</td>
<td>386.25</td>
</tr>
<tr>
<td>VMLS-SVM</td>
<td>10,000</td>
<td>0.01</td>
<td>Align, σ = 12</td>
<td>80</td>
<td>408.17</td>
</tr>
</tbody>
</table>

in the following form for every possible value of $i$:

$$(x_i, y_i) = (x(t + i − N + 1), \ldots, x(t + i), x(t + i + k)),$$

where $0 \leq i, x_i \in \mathbb{R}^N$ and $y_i \in \mathbb{R}$.

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.

In our tests we experimented with three types of kernels for the LS-SVM and VMLS-SVM. These were the following:

- Polynomial Kernel,
- RBF kernel,
- Alignment kernel for time series [8].

4 Experimental results

Now we would like to explain how we chose the parameters of the learning methods during the test phase and summarize our experimental results in time series forecasting.

4.1 Parameter selection

In the field of machine learning, determining the hyperparameters 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 [15] optimization method, the parameter optimizer which optimized the parameters of the underlying learning method. This parameter selection method can be viewed as a function minimizing method, where the input of objective function is the parameter of the underlying learner. The value of the function is the aggregated error of the underlying method on a fixed optimization set where the underlying method were trained on a different, but also fixed training set using the input of the objective function as parameter. 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 set.

4.2 Results on CATS benchmark

The CATS benchmark [10] 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:

- elements 981 to 1,000,
- elements 1,981 to 2,000,
The goal of the competition was to predict these 100 missing values. Twenty-four teams participated and seventeen of them uploaded acceptable predictions. The evaluation metric was the MSE metrics, computed on the 100 missing values as evaluation set.

The method that was used by the winner [20] 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 based on (20). 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 provided 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.

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 Alignment kernel and its parameter is denoted by \( \sigma \). Here tMSE 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 \( \delta = 1, 000, \gamma = 0.01 \) 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 competition. The Table 2 contains just the best

<table>
<thead>
<tr>
<th>Method</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMLS-SVM (RBF)</td>
<td>287</td>
</tr>
<tr>
<td>VMLS-SVM (Align)</td>
<td>312</td>
</tr>
<tr>
<td>Kalman Smoother</td>
<td>408</td>
</tr>
<tr>
<td>Recurrent Neural Networks</td>
<td>441</td>
</tr>
<tr>
<td>Competitive Associative Net</td>
<td>502</td>
</tr>
<tr>
<td>Weighted Bidirectional Multi-stream Extended Kalman Filter</td>
<td>530</td>
</tr>
<tr>
<td>SVCA Model</td>
<td>577</td>
</tr>
<tr>
<td>MultiGrid -Based Fuzzy Systems</td>
<td>644</td>
</tr>
<tr>
<td>Double Quantization Forecasting Method</td>
<td>653</td>
</tr>
<tr>
<td>Time -reversal Symmetry Method</td>
<td>660</td>
</tr>
<tr>
<td>BYY Harmony Learning Based Mixture of Experts Model</td>
<td>676</td>
</tr>
<tr>
<td>Ensemble Models</td>
<td>725</td>
</tr>
<tr>
<td>Chaotic Neural Networks</td>
<td>928</td>
</tr>
<tr>
<td>Evolvable Block-based Neural Networks</td>
<td>954</td>
</tr>
<tr>
<td>Time -line Hidden Markov Experts</td>
<td>1037</td>
</tr>
<tr>
<td>Fuzzy Inductive Reasoning</td>
<td>1050</td>
</tr>
<tr>
<td>Business Forecasting Approach to Multilayer Perceptron Modelling</td>
<td>1156</td>
</tr>
<tr>
<td>A hierarchical Bayesian Learning Scheme for Autoregressive Neural Networks</td>
<td>1247</td>
</tr>
<tr>
<td>Hybrid Predictor</td>
<td>1425</td>
</tr>
</tbody>
</table>

- elements 2,981 to 3,000,
- elements 3,981 to 4,000,
- elements 4,981 to 5,000.

3Here we changed just the values of \( \delta, \gamma \) and the Window size.
results of our models, using $\delta = 1,000$, $\gamma = 0.01$ and Window size = 80 parameter values.

Figure 3 shows the predicted values of the best setup.

4.3 Results on dataset of NN3 Neural Forecasting Competition

To further investigate the capability of VMLS-SVM, we made a comparison with $\epsilon$-insensitive SVM regression method. Hence we applied our method to forecast the reduced subset of time series of the NN3 Artificial Neural Network and Computational Intelligence Forecasting Competition, which has reported results using an $\epsilon$-insensitive SVM regression approach [7].

The NN3 dataset contains 11 time series. The averaged length of the time series is 124\(^4\), and there is no domain knowledge about the time series. In [7], each the last 18 values of each time series were predicted. The evaluation metric was the [2, 14] symmetric mean absolute percent error (SMAPE):

\[
\text{SMAPE} = \frac{1}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{(y_i + \hat{y}_i)/2},
\] (21)

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 based on Eq. (20). Each prediction function used the same parameter setting, but each times series used different parameter setting, which were determined by parameter optimization. To make 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 $\delta$ and $\gamma$ 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 shows a comparison between our method and the results reported for this series in [7].

The mean SMAPE of our method is 3.6757 and the same value of the $\epsilon$-insensitive SVM regression method is 5.05. Hence we can assess that our method using VMLS-SVM methods and parameter optimization achieved significant higher mean performance. As can be seen our method achieved lower SMAPE on each time series except the first one. This can be interpreted by using of parameter optimization method e.g. the hyper-parameter selection method can identify the overfitting, so it cannot try any parameter setting, but this overfitting detection based on the optimization set, so it is only a heuristic, which can be wrong.
Table 3. SMAPE of each time series using the VMLS-SVM and $\epsilon$-insensitive SVM regression methods.

<table>
<thead>
<tr>
<th>Time series</th>
<th>Length</th>
<th>SMAPE(VMLS-SVM)</th>
<th>SMAPE($\epsilon$-insensitive SVR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>126</td>
<td>0.5218</td>
<td>0.49</td>
</tr>
<tr>
<td>2</td>
<td>126</td>
<td>1.7432</td>
<td>2.5</td>
</tr>
<tr>
<td>3</td>
<td>126</td>
<td>8.7458</td>
<td>12.46</td>
</tr>
<tr>
<td>4</td>
<td>115</td>
<td>4.6944</td>
<td>5.4</td>
</tr>
<tr>
<td>5</td>
<td>126</td>
<td>0.3091</td>
<td>0.7</td>
</tr>
<tr>
<td>6</td>
<td>126</td>
<td>1.8892</td>
<td>3.38</td>
</tr>
<tr>
<td>7</td>
<td>126</td>
<td>0.9871</td>
<td>1.21</td>
</tr>
<tr>
<td>8</td>
<td>115</td>
<td>0.9871</td>
<td>1.21</td>
</tr>
<tr>
<td>9</td>
<td>123</td>
<td>1.1947</td>
<td>2.54</td>
</tr>
<tr>
<td>10</td>
<td>126</td>
<td>8.4968</td>
<td>12.7</td>
</tr>
<tr>
<td>11</td>
<td>126</td>
<td>3.7217</td>
<td>4.82</td>
</tr>
</tbody>
</table>

4.4 Results for the Santa Fe Data Set D

Here we will present the results of our method on a widely-recognised benchmark, the D data series from the Santa Fe competition [27]. The data set consists of artificial data generated from a nine-dimensional periodically driven dissipative dynamical system with an asymmetrical four-well potential and a drift on the parameters.

The main characteristics of this series are:

- a relatively high dimensional dynamics,
- a big data set (100,000 data points),
- there is no background information.

Our evaluation is measured by the Root Mean Squared Error (RMSE) metric and we predict 25 missing values in the same setting as that presented in Section 4.1.

This data set is quite large, which implies that it is a computationally demanding learning task. Hence we trained our system with only one configuration:

- we used only the last 20,000 training samples,
- we used the RBF kernel with $G = 0.001$,
- the window size was 80,
- the value of $\delta$ was 1,000,
- the value of $\gamma$ was 0.01.

These parameters were determined using our experiments with the earlier tested benchmarks.

Table 4 shows the results of our method relative to other methods proposed by Müller et al. [17].

The presented results showed that our methods can achieve better performance than others in this task also. In the next section we give some overview about our work.

Table 4. Comparison between the methods on Santa Fe D data set.

<table>
<thead>
<tr>
<th>Method</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMLS-SVM (RBF)</td>
<td>0.0628</td>
</tr>
<tr>
<td>SVM ($\epsilon$-insensitive)</td>
<td>0.0639</td>
</tr>
<tr>
<td>SVM (Huber)</td>
<td>0.0653</td>
</tr>
<tr>
<td>RBF network</td>
<td>0.0677</td>
</tr>
<tr>
<td>ZH [30]</td>
<td>0.0665</td>
</tr>
</tbody>
</table>

5 Conclusions

In this article we presented a novel extension for LS-SVM based on 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 the standard LS-SVM or other similar methods. The nice results we obtained can probably be attributed to the beneficial features of our novel method, since the data sets we used for expriments are typical and widely-used benchmarks for testing machine learning algorithms for time series prediction.

On the other hand, there are some issues that need to investigated like a theoretical analysis of the precise connection between the two parameters $\gamma$ and $\delta$, a more in depth exploration of the main idea e.g. can we apply such a variance minimization term in the $\epsilon$-insensitive SVM regression. We also would like to test our methods on standard regression benchmark. This is what we plan to do in the near future.
References