Overview on Support Vector Machines applied to Temporal Modeling

Conference Paper · January 2012

CITATION
1

READS
340

3 authors:

Renata C. B. Madeo

19 PUBLICATIONS **124** CITATIONS

SEE PROFILE

Sarajane Marques Peres
Utrecht University

92 PUBLICATIONS **262** CITATIONS

SEE PROFILE

Clodoaldo A M Lima
University of São Paulo

68 PUBLICATIONS **467** CITATIONS

SEE PROFILE

Overview on Support Vector Machines applied to Temporal Modeling

Renata C. B. Madeo, Sarajane M. Peres, Clodoaldo A. M. Lima

[1]School of Arts, Sciences, and Humanities – University of São Paulo (USP)
São Paulo – SP – Brazil

{renata.si, sarajane, c.lima}@usp.br

Abstract. *Support Vector Machines have presented good results in Machine Learning tasks, mainly in classification and regression; such results have been observed on several complex problems, including temporal information processing. Considering that traditional Support Vector Machines do not natively deal with temporal dependency among data, some approaches aim at extracting features that capture temporal aspects from data, providing a vector representation. This paper presents an overview on Support Vector Machines applied to temporal modeling.*

1. Introduction

Support Vector Machine (SVM) is a technique of Machine Learning that has raised an increasing interest, mainly for complex applications. Initially, SVM was developed for binary classification tasks [Vapnik 1995]. Later it was adapted for regression tasks, leading to Support Vector Regression (SVR) [Vapnik et al. 1997], and multiclass classification [Hsu and Lin 2002]. SVM consists in mapping the data in the input space into a high dimensional feature space through a nonlinear mapping such that the binary classification problem are indeed linearly separable or linearly approximately separable, and then finds an optimal separating hyperplane that maximizes the margin between two classes in the high-dimensional feature space.

However, theoretically, the traditional SVM is not able to handle data with temporal dependencies, since it considers data as independent and identically distributed. Thus, in order to tackle this problem with SVM, we need to apply some strategy for representing and processing temporal information.

In fact, problems that involve temporal aspects could be treated in a way which does not consider, directly, such aspects. This approach is very common because of its simplicity, but it does not take advantage of temporal dependencies within data, which can represent important information for the data analysis in this type of problems. Thus, in order to provide more adequated approaches, it is most useful considering the temporal aspects in the stratety to solve the problems.

In this context and based on [Chappelier and Grumbach 1994], we propose a classification of SVM models regarding time integration, considering time aspects in a internal or external way, with respect to the analysis data performed by a SVM model.

In order to deal with time internally to the SVM model, it is necessary to adapt SVM model to incorporate temporal reasoning. Some techniques for incorporating temporal reasoning into SVM includes Recurrent Least-Squares

SVM [Suykens and Vandewalle 2000, Qu et al. 2009, Sun et al. 2008, Xie 2009], Support Vector Echo-State Machines [Shi and Han 2007], Profile-Dependent SVM [Camps-Valls et al. 2007], Recursive Kernels [Hermans and Schrauwen 2012], and Sequential Kernels [Wan and Carmichael 2005, Rudzicz 2009, Jaakkola and Haussler 1999, Campbell 2002, Watkins 1999].

In order to treat time externally, there are other two possible classifications: time can be represented implicitly or explicitly. An implicit way to represent time consists in preprocessing data in order to incorporate temporal dynamics into each datapoint, as applied in [Niu and Wang 2009, Wang and Ren 2006, Peng et al. 2011, Schmidhuber et al. 2007]. Explicitly, it is possible to develop a mathematical model considering time, and use a traditional SVM to estimate some parameters for this model, as applied in [Chen and Zhong 2009, Li et al. 2008].

Initiatives in SVM with time aspects treatment have been developed by different researches, independentely, and they are not correlated neither and not standardized. Thus, the present paper has the objective to integrate efforts to provide a unified taxonomy and formalism for such initiatives, facilitating and motivating the development of research in this area. In the present overview we are discussing the last approach: SVM models working with time aspects using an externally way. The other approach (working with time aspects in an internal way) is treated in a previous and specific review[1].

In order to organize the present overview, this paper is divided in: Section 2 presents the basis on SVM; Section 3 describes some strategies using SVM and temporal modeling, focusing mainly on researches from the last five years; in Section 4 are discussed some aspects regarding to the topic addressed in this paper; and finally, Section 5 presents our final considerations.

2. Support Vector Machines

SVM is based on performing a nonlinear mapping on input vectors from their original feature space to a high-dimensional feature space, and optimizing a hyperplane capable of separating data in this high-dimensional feature space [Vapnik 1995].

One of the main differences between SVM and other machine learning techniques is that it uses the structural risk minimization principle. This principle relates to Vapnik-Chervonenkis dimension, also known as VC dimension, that measures the capacity or expressive power of a set of classification functions given by a learning machine. Structural risk minimization depends on finding a learning machine such that a reduction in VC dimension occurs at expense of the smallest possible increase in training error [Haykin 1999].

Consider a training set with N samples, defined by $\{\boldsymbol{x_i}, y_i\}_{i=1}^{N}$, where $\boldsymbol{x_i}$ is an input and y_i is an output, and $y_i \in \{-1, +1\}$. SVM aims at finding an optimal classes separation hyperplane, which is given by $f(\boldsymbol{x_i}) = \boldsymbol{w}\varphi(\boldsymbol{x_i}) + b$, where $\boldsymbol{w}$ is the optimal set of weights, b is the optimal bias, and φ is the nonlinear mapping applied to input vectors. SVM optimizes the hyperplane maximizing the distance between this hyperplane and its

[1]Our review about techniques to incorporate temporal reasoning into SVM entitled "A Review on Temporal Reasoning using Support Vector Machines" has been accepted for publication in the 19th International Symposium on Temporal Representation and Reasoning (TIME 2012), to be presented in September 2012.

closest datapoints ($\boldsymbol{x_i}$), which corresponds to minimizing $\boldsymbol{w}$ using, for example:

$$\min \phi(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \gamma \sum_{i=1}^{N} \xi_i, \tag{1}$$

where γ is a regularization factor, and ξ is an error factor, subject to

$$\begin{aligned} & y_i(\boldsymbol{w}^T\varphi(\boldsymbol{x_i}) + b) \geq 1 - \xi, \quad i = 1, ..., N, \\ & \xi_i \geq 0, \quad i = 1, \cdots, N. \end{aligned}$$

Applying Lagrangian method in (1), we obtain

$$\max \mathcal{L}_1(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \varphi(\boldsymbol{x_i}) \cdot \varphi(\boldsymbol{x_j}) \rangle, \tag{2}$$

subject to

$$\begin{aligned} & \sum_{i=1}^{N} \alpha_i y_i = 0, \\ & \gamma \geq \alpha_i \geq 0 \quad \text{for } i = 1, 2, \cdots, N, \end{aligned}$$

where $\boldsymbol{\alpha}$ are Lagrangian multipliers. Solving the problem in (2), it is possible to solve the problem in (1), since $\boldsymbol{w}$ can be defined in terms of $\boldsymbol{\alpha}$, as in [Haykin 1999]. In (2), a kernel function $K(\boldsymbol{x_i}, \boldsymbol{x_j})$ can be used to represent the dot product $\langle \varphi(\boldsymbol{x_i}) \cdot \varphi(\boldsymbol{x_j}) \rangle$, performing an implicit nonlinear mapping. Some functions which can be used as kernel are discussed in [Haykin 1999].

Also, we can adapt SVM to perform regression tasks through a technique called Support Vector Regression (SVR) [Vapnik et al. 1997]. SVR considers slack variables ξ_i decomposed in ξ_i and $\hat{\xi}_i$, which represents, respectively, errors above and below real output. The most common loss function used for SVR is the ϵ-insensitive loss function, which is formulated as

$$L_\epsilon = \begin{cases} 0, & \text{if } |y_i - f(\boldsymbol{x_i})| \leq \epsilon \\ y_i - f(\boldsymbol{x_i}) - \epsilon, & \text{if } |y_i - f(\boldsymbol{x_i})| > \epsilon, \end{cases} \tag{3}$$

where ϵ is a parameter defined by the user which states the maximum deviation that should be accepted by the algorithm, that is, errors below ϵ are not considered errors.

The formulation of the regression problem depends on the loss function. Thus, for ϵ-insensitive loss function, the problem is formulated as minimizing

$$\min \phi(\boldsymbol{w}) = \frac{1}{2}\langle \boldsymbol{w} \cdot \boldsymbol{w} \rangle + C \sum_{i=1}^{N} \left(\xi_i + \hat{\xi}_i \right),$$

subject to the restrictions

$$y_i - \langle \boldsymbol{w} \cdot \boldsymbol{x_i} \rangle - b \leq \epsilon + \xi_i$$
$$\langle \boldsymbol{w} \cdot \boldsymbol{x_i} \rangle + b \leq \epsilon + \hat{\xi}_i$$
$$\xi_i, \hat{\xi}_i \geq 0.$$

3. Support Vector Machine and Temporal Modeling

This section summarizes researches on SVM applied to temporal modeling, and it is divided in: temporal preprocessing on data for posterior analysis by a traditional SVM (Sections 3.1 and 3.2); temporal mathematical modeling for posterior estimation of some parameters through SVM (Section 3.3).

3.1. Methods based on Phase Space Reconstruction

One approach for temporal analysis consists in extracting temporal features from the data, and composing a representation vector with these features. Nevertheless, it may be difficult to extract temporal features for temporal problems such as unidimensional time-series analysis. In this kind of problem, the data is composed by one unidimensional datapoint at each time, thus temporal features would have to be extracted as relations between datapoints at different times.

One method for incorporating time-delay into unidimensional data is Phase Space Reconstruction (PSR), which converts a scalar that represents an unidimensional datapoint at a time t into a m-dimensional vector, incorporating the temporal dynamics. According to [Packard et al. 1980] and [Fraser and Swinney 1986], this mapping creates an embedded matrix $\boldsymbol{x}_i = (x_i, x_{i+\tau}, \cdots, x_{i+(m-1)\tau}), \quad i = 1, 2, \cdots, N - (m-1)\tau, \boldsymbol{x_i} \in \mathbb{R}^m$, where x_i is the i^{th} datapoint in the original time series, $\boldsymbol{x}_i$ is the i^{th} reconstructed datapoint, m is a embedding dimension, and τ is a time delay; in this approach, $\boldsymbol{x_i}$ compose the new data to be processed by SVM. Figure 1 shows a graphical representation of how PSR converts an unidimensional time series into a embedding matrix.

For classification or regression tasks, it is possible to use the embedded matrix as input for a SVM, instead of the original unidimensional time-series. Also, it is possible to extract time-domain and frequency-domain features from each m-dimensional vector in order to represent each datapoint [Phinyomark et al. 2009].

It is interesting to note that one of the most common windowing techniques corresponds to PSR with $\tau = 1$, usually using an empirical method for finding the value for m. However, it is possible to select more adequate values for m and τ. Some approaches to calculate them are mentioned here.

3.1.1. Mutual Information and Cao Method.

In [Peng et al. 2011], Mutual Information (MI) is employed for defining time-delay, and Cao Method (CM) for selecting the embedding dimension. MI measures general dependence between two variables. Thus, defining time-delay through MI can guarantee the

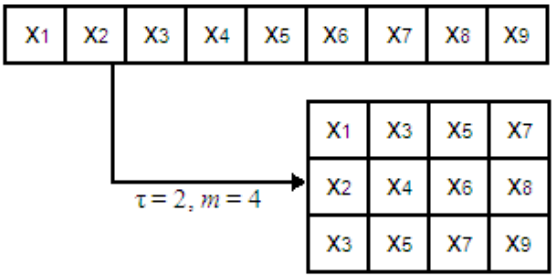

Figure 1. Graphical representation of Phase Space Reconstruction.

maximum independence between variables [Fraser and Swinney 1986]. In [Cao 1997], CM is based on false neighbors method. The latter consists in determining the embedding dimension as the lowest dimension which does not contain false neighbors, i.e., datapoints that are close in original phase space are also close in reconstructed space. However, determining how far the neighbors should be in reconstructed phase space to be considered false neighbors depends on parameters that must be set by the user. In order to overcome this need for setting parameters, CM computes coefficients based on distance relations between neighbors points in the current dimension (l) and the previous dimension ($l+1$), and finds the embedding dimension ($m = l$) when the variation within such coefficients becomes sufficiently small. MI and CM were used for preprocessing data before applying SVM in a classification problem, aiming at evaluate deterioration degree of tools in metal cutting machining [Peng et al. 2011].

3.1.2. Correlation and Lyapunov Exponents.

A method based on Lyapunov Exponents (LE) also can be used to obtain embedding dimension. The approaches in [Niu and Wang 2009] and [Wang and Ren 2006] consists in: reconstructing m-dimensional phase spaces with time series; choosing a minimal value for τ which marks the correlation among phase spaces; and calculating the maximal estimated value of LE for each value of m. The ideal value for the embedding dimension is the minimum value for which the variation within maximal LE through iterations becomes sufficiently small. LE and was applied to time-series forecasting problems, in order to preprocess data before applying SVR in two applications: one regarding power load in an area of Inner Mongolia region [Niu and Wang 2009]; and other regarding defects in three manufacturing lines of glass panels for CRT TV [Wang and Ren 2006]. These studies use methods considering correlation aiming at defining the time delay parameter.

In a different way, [Li et al. 2009] represents some features by calculating the Hurst Index and using PSR to calculate fractal measures from data, instead of reconstructing a time-series incorporating time-delay. These features are input to a SVM.

3.2. Evoke: Combining Long Short-Term Memory with Support Vector Machines

Evoke is a framework for supervised sequence learning that uses SVR proposed by [Schmidhuber et al. 2007]. It is based on Evolino, which consists of the same framework, but using linear regression (LR) instead of SVR (or SVM, since this framework can

be used also for classification purposes). Both use a procedure based on two-phases: first, a Recurrent Neural Network (RNN) is randomly initialized, it processes input sequences from training set and its output is used as input to train linear regression or SVR; second, after defining the framework output weights[2], the RNN is effectively trained through an evolutionary algorithm using prediction error or residual error (from SVR or SVM models) as fitness measure.

This strategy can be formulated by:

$$y(k) = w_0 + \sum_{i=1}^{k} \sum_{j=0}^{l_i} w_{ij} K(\phi(t), \phi^i(j)) \tag{4}$$

$$\varphi(k) = f(u(k), u(k-1), \cdots, u(0)), \tag{5}$$

where $K(\cdot, \cdot)$ is a predefined kernel function, and the weights w_{ij} correspond to k training sequences ϕ^i, each of length l_i, and are computed with the SVR/SVM, $\varphi(k)$ is the output of the RNN given by $f(\cdot)$, which is a function on the entire input history $(u(k), u(k-1), \cdots, u(0))$, and $u(k)$ is the input data at time k. In Evoke, $f(u(k), u(k-1), \cdots, u(0))$ corresponds to the input $\boldsymbol{x_i}$ to the SVR. Figure 2 illustrates Evoke architecture.

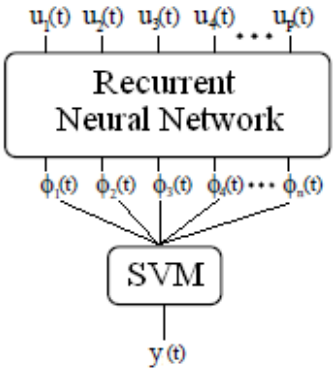

Figure 2. Evoke architecture: a RNN for pre-processing input and a SVM for mapping the pre-processed input into the desired output (adapted from [Schmidhuber et al. 2007]).

In [Schmidhuber et al. 2007], Evolino and Evoke are presented as frameworks that could be instantiate using any kind of RNN. Thus, the authors in [Schmidhuber et al. 2007] highlight the possibility of use Long Short-Term Memory (LSTM), which is a RNN where neurons are represented by memory cells. A memory cell is composed by a linear unit which holds the state of the cell, and three gates: an input gate, which can be closed so that the input does not affect the internal state of the cell; an output gate, which can be closed so that the cell output does not affect other parts of the network; and a forget gate, which enables a state to discard some activity when it is not useful anymore. As described in [Schmidhuber et al. 2007], a state depends on a previous state, on the activation of each gate, and on four weight vectors (gates' weights) – three for defining the activation of each gate and one for the external inputs. Finally, LSTM

[2]Actually, these output weights are the LR or SVR weights.

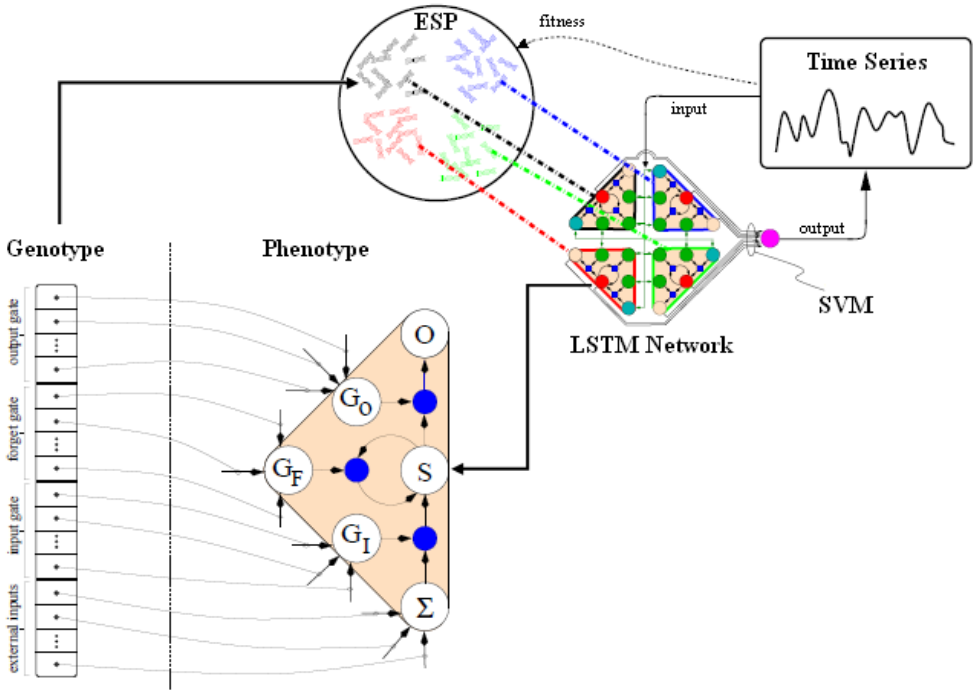

Figure 3. LSTM are trained with Enforced Subpopulations (ESP) evolutionary algorithm. Each LSTM is represented as a chromosome (or genotype) in a subpopulation, which encodes the weights of a memory cell. Each subpopulation represents one specific neuron, so that the evolution occurs for each memory cell separatedly. The weights leading out of the state (S) and output (O) units are not encoded in the genotype, but are instead computed by SVR (or SVM) (adapted from [Schmidhuber et al. 2007]).

training consists of adapting, by using an evolutionary algorithm, these sets of weights which define neuron state through time, defining the function $f(\cdot)$ in (4).

In the instatiation of Evoke described here, Enforced SubPopulations (ESP) is used in order to train the gates' weights. The weights for each neuron are organized as a chromosome, where each gene corresponds to a weight corresponding to external inputs (S), input gate (G_I), forget gate (G_F), and output gate (G_O). [Schmidhuber et al. 2007] highlights that ESP creates separate subpopulations, so that evolution occurs in a separatedly for each neuron. Therefore, a subpopulation is created for each memory cell of LSTM. Also, ESP usually uses crossover to recombine neurons, but, as local search is useful in this case, a Cauchy-distribution is used to add noise to previous individuals to produce all new individuals. Based on [Schmidhuber et al. 2007], we integrate some figures to provide a unified view about the RNN and its evolutive process training. The integrated view – comprising an example of LSTM with four memory cell, and illustrating the concept of subpopulations and the chromosome used to represent each memory cell – is presented on Figure 3.

The process includes: (i) initialization, by creating subpopulations with n neuron cromossomes each (all cromossomes encapsulate a weight vector for that memory cell); (ii) evaluation, by selecting neurons from each subpopulation, combining them to create a RNN, the RNN is evaluated using a fitness measure, and a cumulative fitness is calcu-

lated for each neuron; (iii) reproduction, by ranking the neurons in each subpopulation by fitness, considering the top quarter as parents, making copies of these parents (children), adding noise using Cauchy distribution to these children, and replacing lowest-ranking half by these children. Evaluation and reproduction steps are repeated until a good RNN is found.

An approach similar to Evoke is presented by [Bayro-Corrochano and Arana-Daniel 2010] – the Recurrent Clifford SVM. It employs Evoke using the Clifford SVM (CSVM) – a SVM based on Clifford Geometric Algebra that is capable to process multiple inputs and multiple outputs – instead of SVR, traditionally used in such framework.

3.3. Discrete Time-Delay Systems Modeling

The studies [Chen and Zhong 2009] and [Li et al. 2008] present approaches based on modeling dynamical problems as discrete time-delay systems. Such systems are usually defined by a function which relates[3]: the state of the system $\boldsymbol{x(k)}$, a delayed state of the system $\boldsymbol{x(k-\tau)}$, an input $\boldsymbol{u(k)}$, and the output $y(k)$. Also, these systems usually contain a linear part and a nonlinear part.

In [Chen and Zhong 2009], the described system aims at modeling a problem involving fault detection. The system is modeled by

$$\begin{aligned} x(k+1) &= A\boldsymbol{x(k)} + A_\tau \boldsymbol{x(k-\tau)} + B\boldsymbol{u(k)} + B_f f(\boldsymbol{x(k)}, \boldsymbol{u(k)}, k) \\ y(k) &= Cx(k), \end{aligned} \tag{6}$$

where A, A_τ, B, B_f and C are matrices that represent the linear part of the system, and f is the unknown fault function, which is a nonlinear mapping. Also in [Chen and Zhong 2009], the authors convert this the system modeling presented in (6) into

$$\begin{aligned} \hat{x}(k+1) &= A\boldsymbol{x(k)} + A_\tau \boldsymbol{x(k-\tau)} + B\boldsymbol{u(k)} + L(\boldsymbol{x(k)} - \hat{\boldsymbol{x}}\boldsymbol{(k)}) + B_f \boldsymbol{v(k)} \\ \hat{y}(k) &= C\hat{x}(k), \end{aligned}$$

where $\hat{x}$ is the estimated state of the system, $\hat{y}$ is the estimated output, L is the gain for the linear part of the time-delay system, and $\boldsymbol{v(k)}$ is a control vector, which models the nonlinear part of the system. Thus, it is possible to use SVM (in [Chen and Zhong 2009], a variation of SVM called Least-Squares SVM [Suykens and Vandewalle 1999]) in order to estimate the control vector $\boldsymbol{v(k)}$ using $x(k)$, $\boldsymbol{u(k)}$ and k as LS-SVM inputs.

In a similar approach, the authors in [Li et al. 2008] model an uncertain discrete time-delay system with input saturation as

$$\begin{aligned} x(k+1) &= (A + \Delta A)\boldsymbol{x(k)} + (A_\tau + \Delta A_\tau)\boldsymbol{x(k-\tau)} + B\ \mathrm{sat}(\boldsymbol{u(k)}), \\ y(k) &= Cx(k), \end{aligned} \tag{7}$$

[3]The notation used in this section is slightly different from the rest of the paper in order to be coherent with the notation used in dynamic systems area.

where ΔA and ΔA_τ represents uncertainty regarding A and A_τ, respectively, and $\text{sat}(\boldsymbol{u(k)}$ is a saturation function that bounds the input formulated as

$$\text{sat}(\boldsymbol{u(k)}) = \begin{cases} u_i(k) = b_{inf} & \text{if } u_i(k) < b_{inf} \\ u_i(k) = u_i(k) & \text{if } b_{inf} \leq u_i(k) \leq b_{sup} \\ u_i(k) = b_{sup} & \text{if } u_i(k) > b_{sup}. \end{cases}$$

In this study is proposed a modified Sliding Mode Control (SMC) for solving this class of systems, since traditional SMC suffers with input saturation, i.e., the system may be unstable. Actually, SMC depends on a control law that defines SMC operation; a sliding surface that is an adequate switching function matrix; and a reaching law, which ensures that the system enters in sliding motion within a finite time, as described in [Ma 2009]. Thus, in [Li et al. 2008], the sliding surface is given by:

$$s = Cx, \tag{8}$$

where C is the switching function matrix; and the reaching law is given by

$$s(k+1) = s(k) - Qts(k) - Eto, \tag{9}$$

where t is the sampling time, and E and Q are diagonal matrices with positive components. Traditionally, o is a sign function $\text{sgn}(s(k))$, but in [Li et al. 2008] o is the output of a LS-SVM whose input is $s(k)$. Based on (7), (8), and (9), it is possible to define an overall control law which enforces the systems trajectory onto the sliding surface after a finite period of time. In [Li et al. 2008] the use of a LS-SVM for learning a nonlinear component in the reaching law is proposed, allowing a better response for systems with input saturation. Results show that the sign function employed in the reaching law of traditional SMC causes chattering, while the LS-SVM method makes the sliding surface smoother.

4. Discussion

In Section 3, we presented two approaches of temporal modeling using SVM: (i) preprocessing data in order to incorporate time delay, and then using SVM on these data; (ii) building a mathematical model which considers temporal aspects, and using SVM for estimating specific parameters of the model.

The first approach was discussed using PSR or RNN for preprocessing data. Using PSR, the challenge lies on finding optimal values for embedding dimension and time delay parameters, and some alternatives were cited in Section 3.1. Regarding such alternatives, it is interesting highlighted that:

- About methods for finding time-delay: Correlation only measures linear dependences, while Mutual Information measures a general dependence, corresponding to the gain of information of using two variable together; thus, the latter provides a better criterion for defining the time-delay parameter [Fraser and Swinney 1986].

- About the task of finding embedding dimension: Lyapunov Exponents demands a largest amount of computation among the discussed methods; False Neighbors Method requires user choices about some parameters to indicate when there is a false neighbor; Cao Method adapts the False Neighbor Method eliminating the need such choices, improving the accuracy [Cao 1997].

Using RNN, we have cited the Evoke framework [Schmidhuber et al. 2007]. Experiments in [Schmidhuber et al. 2007] shows that Evoke and Evolino usually presents a better performance than the gradient-descendent version of LSTM (G-LSTM) and is even able to perform tasks that G-LSTM is not able to perform, since these methods overcome problems with local minima using an evolutionary approach for learning. However, reported results also indicate that: Evolino usually outperforms Evoke; and, G-LSTM outperforms Evolino only in complex tasks that require larger LSTM networks (such as 100,000 weights).

Mathematical models that considers temporal aspects are used by [Chen and Zhong 2009] and [Li et al. 2008]. These approaches require some prior knowledge about the system, such as the system linear variables, and need some non-linear estimation method. SVM is used by these authors in order to estimate non-linear variables of the system, presenting better results than classical methods, as described in [Chen and Zhong 2009] and [Li et al. 2008].

5. Conclusion

In this paper we presented a brief review of researches which considers temporal aspects of a problem through two different ways: internal ways were cover in initiatives that preprocessing data with Phase Space Reconstruction, using Mutual Information, Autocorrelation Method based on Correlation Measures, Lyapunov Exponents, False Neighbors Method and Cao Method, or preprocessing data using Recurrent Neural Networks, covered by the Evolino strategy applied on Evoke framework; external ways were cover in initiative that build temporal mathematical models. Internal ways apply SVM or SVR on vector representations created in the data preprocessing; external ways use SVM to estimate non-linear parameters of the mathematical models. The applications solved by each initiative were briefly mentioned.

The object of this review was to present recent researches on the topic in a correlated and standardized way, providing insights on modeling temporal problems using SVM. Moreover, the approaches discussed in this review compose a type of strategy that do not modify the traditional SVM techniques, offering interesting alternatives to efficiently apply previous and classical implementation of SVM in temporal and complex problems.

Acknowledgments.

The authors thank São Paulo Research Foundation (FAPESP), Brazil, for its support through process number 2011/04608-8.

References

Bayro-Corrochano, E. and Arana-Daniel, N. (2010). Clifford support vector machines for classification, regression, and recurrence. *IEEE Trans. on Neural Networks*, 21(11):1731 –1746.

Campbell, W. M. (2002). Generalized linear discriminant sequence kernels for speaker recognition. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 1, pages I–161 –I–164.

Camps-Valls, G., Soria-Olivas, E., Perez-Ruixo, J., Perez-Cruz, F., Artes-Rodriguez, A., and Jimenez-Torres, N. (2007). Therapeutic drug monitoring of kidney transplant recipients using profiled support vector machines. *IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37(3):359 –372.

Cao, L. (1997). Practical method for determining the minimum embedding dimension of a scalar time series. *Physica D: Nonlinear Phenomena*, 110(1–2):43–50.

Chappelier, J.-C. and Grumbach, A. (1994). Time in neural networks. *SIGART Bulletin*, 5(3):3–11.

Chen, Y. and Zhong, Z. (2009). Application of ls-svm and deconvolution method in fault diagnosis for discrete time-delay systems. In *Int. Conf. on Natural Computation*, volume 1, pages 473 –477.

Fraser, A. M. and Swinney, H. L. (1986). Independent coordinates for strange attractors from mutual information. *Physical Review A*, 33:1134–1140.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Prentice Hall, Upper Saddle River, NJ, 2nd edition.

Hermans, M. and Schrauwen, B. (2012). Recurrent kernel machines: Computing with infinite echo state networks. *Neural Computation*, 24:104–133.

Hsu, C.-W. and Lin, C.-J. (2002). A comparison of methods for multiclass support vector machines. *IEEE Trans. on Neural Networks*, 13:415–425.

Jaakkola, T. S. and Haussler, D. (1999). Exploiting generative models in discriminative classifiers. In *Conf. on Advances in Neural Information Processing Systems*, pages 487–493, Cambridge, MA, USA. MIT Press.

Li, J., Zhang, Y., and Pan, H. (2008). Chattering-free ls-svm sliding mode control for a class of uncertain discrete time-delay systems with input saturation. In *Int. Conf. on Intelligent Computation Technology and Automation*, pages 322–326, Washington, DC, USA. IEEE Computer Society.

Li, Q., Zhao, J., and Zhao, Y. (2009). Detection of ventricular fibrillation by support vector machine algorithm. In *Int. Asia Conf. on Informatics in Control, Automation and Robotics*, pages 287 –290.

Ma, J. (2009). Formation flying of spacecrafts for monitoring and inspection. Master's thesis, Lulea University of Technology.

Niu, D. and Wang, Y. (2009). A new model to short-term power load forecasting combining chaotic time series and svm. In *Asian Conf. on Intelligent Information and Database Systems*, pages 420–425. IEEE Computer Society.

Packard, N. H., Crutchfield, J. P., Farmer, J. D., and Shaw, R. S. (1980). Geometry from a time series. *Physical Review Letters*, 45:712–716.

Peng, N., Zhengqiang, L., Yanchun, L., Xinyu, L., and Hongyao, X. (2011). Study on identification method of tool wear based on singular spectrum analysis and support vector machine. In *Int. Conf. on Digital Manufacturing and Automation*, pages 1164 –1167.

Phinyomark, A., Limsakul, C., and Phukpattaranont, P. (2009). A novel feature extraction for robust emg pattern recognition. *Journal of Computing*, 1:71–80.

Qu, H., Oussar, Y., Dreyfus, G., and Xu, W. (2009). Regularized recurrent least squares support vector machines. In *Int. Joint Conf. on Bioinformatics, Systems Biology and Intelligent Computing (IJCBS)*, pages 508–511. IEEE Computer Society.

Rudzicz, F. (2009). Phonological features in discriminative classification of dysarthric speech. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pages 4605 –4608.

Schmidhuber, J., Wierstra, D., Gagliolo, M., and Gomez, F. (2007). Training recurrent networks by evolino. *Neural Comput.*, 19:757–779.

Shi, Z. and Han, M. (2007). Support vector echo-state machine for chaotic time-series prediction. *IEEE Trans. on Neural Networks*, 18(2):359 –372.

Sun, J., Zheng, C., Zhou, Y., Bai, Y., and Luo, J. (2008). Nonlinear noise reduction of chaotic time series based on multidimensional recurrent ls-svm. *Neurocomputing*, 71:3675–3679.

Suykens, J. and Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural Processing Letters*, 9:293–300.

Suykens, J. and Vandewalle, J. (2000). Recurrent least squares support vector machines. *IEEE Trans. on Circuits and Systems-I*, 47:1109–1114.

Vapnik, V. (1995). *The nature of statistical learning theory*. Springer-Verlag.

Vapnik, V., Golowich, S. E., and Smola, A. J. (1997). Support vector method for function approximation, regression estimation and signal processing. In Mozer, M., Jordan, M. I., and Petsche, T., editors, *Advances in Neural Information Processing Systems 9 — Proceedings of the 1996 Neural Information Processing Systems Conference (NIPS 1996)*, pages 281–287. MIT Press, Cambridge, MA, USA.

Wan, V. and Carmichael, J. (2005). Polynomial dynamic time warping kernel support vector machines for dysarthric speech recognition with sparse training data. In *Annual Conf. of the Int. Speech Communication Association*, pages 3321–3324.

Wang, J. and Ren, G. (2006). Load forecasting based on chaotic support vector machine with incorporated intelligence algorithm. In *Int. Conf. on Innovative Computing, Information and Control*, volume 3, pages 435 –439.

Watkins, C. (1999). Dynamic alignment kernels. Technical report, University of London.

Xie, J. (2009). Time series prediction based on recurrent ls-svm with mixed kernel. In *Asia-Pacific Conf. on Information Processing*, volume 1, pages 113 –116.