2012 19th International Symposium on Temporal Representation and Reasoning

A Review on Temporal Reasoning using Support Vector Machines

Renata C. B. Madeo, Clodoaldo A. M. Lima, Sarajane M. Peres
School of Arts, Sciences and Humanities
University of São Paulo
São Paulo, Brazil
{*renata.si, c.lima, sarajane*}*@usp.br*

***Abstract*—Recently, Support Vector Machines have presented promissing results to various machine learning tasks, such as classification and regression. These good results have motivated its application to several complex problems, including temporal information analysis. In this context, some studies attempt to extract temporal features from data and submit these features in a vector representation to traditional Support Vector Machines. However, Support Vector Machines and its traditional variations do not consider temporal dependency among data. Thus, some approaches adapt Support Vector Machines internal mechanism in order to integrate some processing of temporal characteristics, attempting to make them able to interpret the temporal information inherent on data. This paper presents a review on studies covering this last approach for dealing with temporal information: incorporating temporal reasoning into Support Vector Machines and its variations.**

***Keywords*-Support Vector Machine; temporal reasoning; machine learning;**

I. Introduction

In the last decade, there has been a great increase the use of Support Vector Machines (SVM) in various applications. The growing interest in this technique is justified by its good performance, presented in different studies applied to complex problems, including problems with temporal data.

Traditional SVM approaches aim at processing *independent and identically distributed* (iid) data, ignoring temporal dependencies among instances. Therefore, for employing traditional SVM to temporal problems, it is necessary to extract temporal features, incorporate these features into a vector representation, and process them just as they were iid data. Although this naive approach has provided some good results, it ignores important properties of the data.

We have adopted a taxonomy discussed in [1] in order to organize different SVM approaches regarding temporal data analysis. First, as aforementioned, problems related to temporal analysis can be treated without considering, directly, temporal aspects of the data. Although this approach does not take advantage of temporal dependencies within data, it is still very common because of its simplicity. If we consider the temporal aspects, time can be externally or internally processed with respect to SVM model. In order to treat time externally, we can build implict or explicit temporal data representations, which consist of, respectively, preprocessing data in order to incorporate temporal dynamics into each datapoint to be submitted to traditional SVM models; or developing mathematical models considering time, and use a traditional SVM to estimate some parameters for these models. Otherwise, regarding approaches that deal with temporal aspects in a internal way, we can adapt SVM model to incorporate temporal reasoning, building more complex strategies that are able to interpret and take advantage of the temporal dependencies among the data.

In this paper we present a review on studies that incorporate temporal reasoning into SVM, considering mainly approaches applied in researches executed in the last five years. This paper is organized as follows: Section II and Section III briefly presents traditional SVM and some of its variation; Section IV describes approaches that incorporate temporal reasoning into SVM; finally, our final considerations are delineated in Section V.

II. Support Vector Machines

SVM are based on performing a non-linear mapping on input vectors from their original input space to a high-dimensional feature space; and optimize an hyperplane capable of separating data in this high-dimensional feature space. This section describes SVM formulation for classification and regression problems.

A. Classification Problem

Considering a training set with N samples, defined by $\{\boldsymbol{x_i}, y_i\}_{i=1}^{N}$, with input $\boldsymbol{x_i} \in \mathbb{R}^m$ and output $y_i \in \{-1, +1\}$. SVM aims at finding an optimal hyperplane which separates the datapoints in the feature space. Such hyperplane is given by

$$f(x) = \langle \boldsymbol{w} \cdot \varphi(\boldsymbol{x_i}) \rangle + b,$$

where $\boldsymbol{w}$ is the optimal set of weights, $\varphi(\boldsymbol{x_i})$ represents a nonlinear mapping on $\boldsymbol{x_i}$, b is the optimal bias, and $\langle \cdot \rangle$ is a dot product.

In order to optimize separating hyperplane, SVM aims at maximizing the functional margin, i.e., the distance between the hyperplane and the closest datapoints from the hyperplane. As shown in [2], maximizing the margin corresponds to minimizing the set of weights $\boldsymbol{w}$. The same author also states that, in order to achieve a good generalization, it

1530-1311/12 $26.00 © 2012 IEEE
DOI 10.1109/TIME.2012.15

is necessary to minimize the Vapnik-Chervonenkis (VC) dimension, which measures the capacity of the family of functions realized by a learning machine. Thus, minimizing $\boldsymbol{w}$ also corresponds to minimizing VC dimension and finding an optimal hyperplane provides a SVM with the smallest VC dimension necessary to solve the classification problem.

For problems which are nonlinearly separable in the feature space, we have to consider a soft margin optimisation, which assumes slack variables ξ_i. Such variable denotes the training error for the ith sample, $i = 1, \cdots, N$, in order to find an hyperplane allowing some classification errors that are minimized into the optimisation problem. There are two ways to implement this soft-margin optimisation using a 2-norm or 1-norm [3]. Considering 2-norm soft margin, it is possible to find the optimal hyperplane by minimizing:

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

where C is a regularization factor and $\xi_i = |y_i - f(x_i)|$, subject to

$$y_i(\langle \boldsymbol{w} \cdot \varphi(\boldsymbol{x_i}) \rangle + b) \geq 1 - \xi_i, \quad i = 1, ..., N.$$

From this optimization problem, we can apply the Lagrangian method obtaining

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

where $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise, subject to

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

Already considering 1-norm soft margin, it is possible to optimize a hyperplane by minimizing

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

subject to

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

From this optimization problem, we can apply the Lagrangian method obtaining

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

subject to

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

An α_i is assigned to each input vector. After training, all non-zero α_i are called Support Vectors (SV).

In the models obtained with Lagrangian method, the terms $\varphi(x_i)$ and $\varphi(x_j)$ always appear multiplied. This fact allows us to perform an implicit nonlinear mapping to a high-dimensional feature space through kernels. This approach is based on Cover Theorem, which states that a feature space with non-linearly separable data can be mapped with high probability into a input space where the data is linearly separable, provided that the mapping is non-linear and the feature space dimension is high enough [2]. Most common kernel functions can be seen in Table I.

Table I
MOST COMMON KERNEL FUNCTIONS [2].

Kernel	Function
Polynomial	$(\langle \boldsymbol{x_i} \cdot \boldsymbol{x_j} \rangle + 1)^p$
Radial Basis Function	$\exp\left(\frac{\|\boldsymbol{x_i} - \boldsymbol{x_j}\|^2}{2\delta^2}\right)$
Two Layer Perceptron	$\tanh\left(\delta(\langle \boldsymbol{x_i} \cdot \boldsymbol{x_j} \rangle) + k\right)$

B. Regression Problem

Also, SVM may be adapted for regression tasks. The technique is called Support Vector Regression (SVR) and considers slack variables ξ_i decomposed in ξ_i and $\hat{\xi}_i$, which represents, respectively, errors above and below real output. In the same way as in classification problems, the formulation of the regression problem depends on the loss function.

The most common loss function is the ϵ-insensitive loss function, which is formulated as

$$L_\epsilon = \begin{cases} 0, & \text{if } |y_i - f(\boldsymbol{x_i})| \leq \epsilon \\ y_i - f(\boldsymbol{x_i}) - \epsilon, & \text{if } |y_i - f(\boldsymbol{x_i})| > \epsilon, \end{cases} \tag{1}$$

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.

In this case, 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

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

However, there are other loss functions, such as quadratic loss function and Huber loss function [4]. The quadratic loss function is defined as

$$L_{quad} = (f(\boldsymbol{x_i}) - y_i)^2,$$

which leads us to the problem formulation

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

subject to

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

The Huber loss function is defined as

$$L_{Huber} = \begin{cases} \frac{1}{2}(f(\boldsymbol{x_i}) - y_i)^2, & \text{if } |f(\boldsymbol{x_i} - y_i)| > \mu \\ \mu|f(\boldsymbol{x_i} - y_i)| - \frac{\mu^2}{2}, & \text{if } |f(\boldsymbol{x_i} - y_i)| \leq \mu. \end{cases}$$

leading to a problem formulation similar to (2), as described in [4].

III. Least-Squares Support Vector Machines

In [5], [6], a least squares type of SVM was introduced by changing the problem formulation so as to yield a linear set of equations in the dual space. This is done by taking a least squares cost function, with equality instead of inequality constraints. This modification enable the classifier to be solved through a set of linear equations instead of quadratic programming.

In LS-SVM, the classification problem is formulated as

$$\min \mathcal{J}_2(\boldsymbol{w}, b, e) = \frac{1}{2}\langle \boldsymbol{w} \cdot \boldsymbol{w} \rangle + \frac{C}{2}\sum_{k=1}^{N}\xi_i^2,$$

subject to the equality constraints

$$y_k[\langle \boldsymbol{w} \cdot \varphi(\boldsymbol{x_i}) \rangle + b] = 1 - \xi_i, \quad i = 1, \cdots, N.$$

Applying Lagrangian method, we obtain

$$\begin{aligned} \max \mathcal{L}_2(\boldsymbol{w}, b, e; \alpha) &= J(\boldsymbol{w}, b, \xi) \\ &- \sum_{i=1}^{N}\alpha_i\{y_i[\langle \boldsymbol{w} \cdot \varphi(\boldsymbol{x_i}) \rangle + b] - 1 + \xi_i\}, \end{aligned}$$

with the following optimality conditions:

$$\begin{aligned} &\frac{\partial \mathcal{L}}{\partial w} = 0 \rightarrow \boldsymbol{w} - \sum_{i=1}^{N}\alpha_k y_i \varphi \boldsymbol{x_i} = 0, \\ &\frac{\partial \mathcal{L}}{\partial b} = 0 \rightarrow \sum_{i=1}^{N}\alpha_i y_i = 0, \\ &\frac{\partial \mathcal{L}}{\partial \xi_i} = 0 \rightarrow \alpha_i = C\xi_i \quad i = 1, \cdots, N, \\ &\frac{\partial \mathcal{L}}{\partial \alpha_i} = 0 \rightarrow y_i[\langle \boldsymbol{w} \cdot \varphi(\boldsymbol{x_i}) \rangle + b] - 1 + \xi_i = 0, \quad i = 1, \cdots, N. \end{aligned}$$

This set of equations can be written as the solution of a set of linear equations [5]. Thus, it is easier to calculate a solution for LS-SVM than for SVM. However, LS-SVM has a disadvantage: the number of SV is proportional to the errors at the datapoints, losing sparsity provided by SVM.

Also, as in SVM, LS-SVM can be adapted for regression tasks [7].

IV. Analysed Methods

This section presents a systematic organization of the papers selected by our review, covering the topic *temporal reasoning using SVM*. The section is organized as follows: Section IV-A presents Recurrent LS-SVM; Section IV-B presents Support Vector Echo-State Machines; Profile-Dependent Support Vector Machines are described in Section IV-C; and Section IV-D and Section IV-E presents modified kernels for dealing with temporal data, considering respectively recurrent kernels and sequential kernel.

A. Recurrent Least-Squares Support Vector Machines

Recurrent LS-SVM (RLS-SVM) is mostly applied to time series forecasting [7]. The idea is to consider as data a series of input data u_k and a series of output data y_k and an autonomous recurrent model such as

$$\hat{y}_k = f(\hat{y}_{i-1}, \hat{y}_{i-2}, ..., \hat{y}_{i-p}),$$

where $\hat{y}_i$ denotes an estimated output at the instant i, f is a nonlinear mapping, and the value p defines model order. Thus, we have that the sequence of previously estimated outputs are the input for the forecasting model.

Then, from SVM theory, we can consider $[\hat{y}_{i-1}, \hat{y}_{i-2}, \cdots, \hat{y}_{i-p}]$ as input[1] for the model. Also, the model is formulated in terms of error variables, i.e, RLS-SVM training depends on the error between estimated output and actual output [7]. Thus, the training is defined as the following optimization problem:

$$\min \mathcal{J}(\boldsymbol{w}, b, \boldsymbol{e}) = \frac{1}{2}\langle \boldsymbol{w} \cdot \boldsymbol{w} \rangle + \frac{C}{2}\sum_{i=p+1}^{n+p}\xi_i^2, \tag{3}$$

subject to:

$$y_k - e_{k-1} = \langle \boldsymbol{w} \cdot \varphi(\boldsymbol{z_{i-1|i-p}}) \rangle + b, \quad i = p+1, \cdots, n+p,$$

where n is the lenght of the time series, $\xi_i = y_i - \hat{y}_i$ represents the error, and $\boldsymbol{z_{i-1|i-p}} = [\hat{y}_{i-1}, \hat{y}_{i-2}, \cdots, \hat{y}_{i-p}] - [\xi_{i-1}, \xi_{i-2}, \cdots, \xi_{i-p}]$. Then, applying Lagrangian method, we obtain

$$\begin{aligned} \max \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}; \boldsymbol{\alpha}) &= J(\boldsymbol{w}, \boldsymbol{\xi}, b) \\ &+ \sum_{i=1}^{n}\alpha_{i-p} \times \left[y_i - \xi_{i-1} - \langle \boldsymbol{w} \cdot \varphi(\boldsymbol{z_{i-1|i-p}}) \rangle - b\right], \end{aligned}$$

subject to (4), (5), (6), and (7).

As $\boldsymbol{w}$ can be defined in terms of $\boldsymbol{\alpha}$, it is possible to discard the constraint (4), however, even discarding it, it is still computionally expensive to find a solution considering all the other constraints, specially (6). Thus, it is possible to consider the case when $C \rightarrow \infty$, which corresponds to ignoring the first term in (3), i.e., aiming only at minimizing errors, subject to (5) and (7).

[1] In our notation, $[y_{k-1}, y_{k-2}, \cdots, y_{k-p}]$ is, therefore, equivalent to $\boldsymbol{x_i}$ in the SVM model input.

$$
\begin{cases}
\dfrac{\partial \mathcal{L}}{\partial w} = \boldsymbol{w} - \displaystyle\sum_{i=p+1}^{n+p} \alpha_{i-p} \varphi(\boldsymbol{z_{i-1|i-p}}) = 0 & (4) \\
\dfrac{\partial \mathcal{L}}{\partial b} = \displaystyle\sum_{i=p+1}^{n+p} \alpha_{i-p} = 0 & (5) \\
\dfrac{\partial \mathcal{L}}{\partial \xi_i} = C\xi_i - \alpha_{i-p} - \displaystyle\sum_{j=1}^{p} \alpha_{i-p+j} \dfrac{\partial}{\partial \xi_{i-j}} \left[\langle \boldsymbol{w} \cdot \varphi(\boldsymbol{z_{i-1|i-p}}) \rangle \right] = 0, \quad i = p+1, \cdots, n & (6) \\
\dfrac{\partial \mathcal{L}}{\partial \alpha_{i-p}} = y_i - \xi_i - \langle \boldsymbol{w} \cdot \varphi(\boldsymbol{z_{i-1|i-p}}) \rangle - b = 0, \quad i = p+1, \cdots, n & (7)
\end{cases}
$$

Some particularities of this approach must be highlighted: first, it works only for unidimensional time series, since it uses previous estimated outputs to provide an estimation for the next; second, this approach considers $C \to \infty$, which means it does not control VC-dimension and does not aim at maximizing the margin. In order to overcome these problems, [8] proposes a multidimensional RLS-SVM and [9] proposes some methods for controlling the parameter C. Also, in [10], RLS-SVM is used with a mixed kernel in order to obtain improve accuracy.

1) Regularized Recurrent Least-Squares Support Vector Machines: As aforementioned, the RLS-SVM ignores the regularization terms C, due to the computational cost of calculating the derivative in the constraint (6).

However, [9] argues that this simplification is not really necessary. There are two other options: the former option is called *regularized partially RLS-SVM* by [8]. It consists on disconsidering the summation term in (6) and considering the optimization problem regarding to (3) as a whole. The other option consists in considering all effects of parameter C and the equations (3) and (6) by deriving the summation term in (6) into a set of nonlinear equation which must be solved nummerically [9].

2) Multidimensional Recurrent Least-Squares Support Vector Machines: The RLS-SVM was developed to deal with unidimensional data, which is useful for some applications, such as forecasting unidimensional time series [7]. However, most applications may use multidimensional data.

A multidimensional RLS-SVM can be defined by dividing a multidimensional regression problem into a series of unidimensional problems [8].

Consider a time series given by $\{\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_n}\}$, $x_i \in \mathbb{R}^m$, and a problem with input $x_{i-\tau}$ and output $y_i \in \mathbb{R}^m$ corresponding to an i-th element of the time series, where τ is a time delay. In order to tackle this problem, the authors in [8] propose to convert a m-dimensional problem to m unidimensional problems considering a weight matrix with m weight vectors; and m bias. Incorporating this approach in (3), the problem can be formulated as

$$
\min \mathcal{L}(\boldsymbol{w_i}, \boldsymbol{b_i}, \boldsymbol{e_{k,i}}) = \frac{1}{2} \sum_{i=1}^{m} \langle \boldsymbol{w_i} \cdot \boldsymbol{w_i} \rangle + \frac{1}{2} C \sum_{k=m+1}^{n} \sum_{i=1}^{m} e_{k,i}^2,
$$

subject to

$$
\begin{cases}
y_k^{(1)} - \xi_{k,1} = \langle \boldsymbol{w_1} \cdot \varphi_1(y_{k-1} - \xi_{k-1}) \rangle + b_1, & (8) \\
y_k^{(2)} - \xi_{k,2} = \langle \boldsymbol{w_2} \cdot \varphi_2(y_{k-1} - \xi_{k-1}) \rangle + b_2, \\
\vdots \\
y_k^{(m)} - \xi_{k,m} = \langle \boldsymbol{w_m} \cdot \varphi_m(y_{k-1} - \xi_{k-1}) \rangle + b_m,
\end{cases}
$$

where $k = m+1$, $m = 2, \ldots, n+m-1$ and $\xi_k = y_k - \hat{y}_k$.

This technique may also be applied to unidimensional time series, as long as this time series is reconstructed through Phase Space Reconstruction generating a multidimensional time series.

3) Recurrent Least-Squares Support Vector Machines with Mixed Kernel: RLS-SVM with a mixed kernel consists using a RLS-SVM with a kernel which combines Radial-Basis Function (RBF) and Polynomial Kernel (Poly), weighted by the parameter ρ, as

$$
K_{mix}\left(\boldsymbol{x_i}, \boldsymbol{x_j}\right) = \rho K_{RBF}\left(\boldsymbol{x_i}, \boldsymbol{x_j}\right) + (1-\rho) K_{Poly}\left(\boldsymbol{x_i}, \boldsymbol{x_j}\right).
$$

In this context, the objective is to combine the strenghts of each approach: RBF is a local function, having stronger learning ability and weaker generalization ability; polynomial kernel is a global function, having stronger generalization ability and weaker learning ability [10]. Also, in the same paper is proposed the use of a Genetic Algorithm in order to optimize the parameters of the mixed kernel, i.e., the parameters of RBF, polynomial kernel and ρ.

In order to evaluate the approach, the authors applied it to forecast a time series based on Rossler function. For these data, the use of RLS-SVM with the mixed kernel presented better results than RBF kernel.

B. Support Vector Echo-State Machines

Another strategy for enabling SVM to perform a temporal analysis is called Support Vector Echo-State Machines (SVESM) and uses Reservoir Computing [4]. Reservoir Computing (RC) is a research field which comprises techniques and methods for building Recurrent Neural Networks (RNN) using a "reservoir". A reservoir is a large neural network, with randomly and sparsely connected neurons. Each neuron have an internal representation called states,

which keep some information about the previous states, meaning that the reservoir has memory. Within the reservoir, input weights and internal weights are defined randomly and are not trained.

The SVESM is based on Echo-State Networks (ESN) that is a RNN which uses a reservoir to perform an accurate single-step prediction and then iterates in order to obtain multiple-step predictions [4]. The idea is that only the output weights must be trained, i.e., the reservoir performs a nonlinear mapping and a output weights are trained to perform a regression from reservoir state to the desired output. Its equation can be written as

$$x(i+1) = \text{tansig}\left(\boldsymbol{W_x} \cdot \boldsymbol{x_i} + \boldsymbol{W_{in}} \cdot u_i + \boldsymbol{v_{i+1}}\right)$$
$$y(i+1) = \langle \boldsymbol{w} \cdot \boldsymbol{x_i} \rangle,$$

where $tansig$ denotes the hyperbolic tangent function, $\boldsymbol{x_i}$ denotes the state variables in the reservoir, u_i and y_i are the input and output of the ESN, and $\boldsymbol{v_{i+1}}$ is an optional noise vector. $\boldsymbol{W_x}$, $\boldsymbol{W_{in}}$ and $\boldsymbol{w}$ are the internal connections, input and output weights, respectively.

As aforementioned, the reservoir provides a nonlinear mapping from input space to a state space. The idea of SVESM is using the "reservoir trick" instead of the "kernel trick" to perform the nonlinear mapping [4]. Thus, input data is preprocessed by the reservoir and the reservoir states are used as input to a SVR with linear kernel, which performs the regression in order to obtain the desired output.

This study also presents three types of loss function which can be used in SVESM: quadratic loss function, ϵ-sensitive loss function, and Huber loss function, as described in Section II-B.

Finally, the study presents some simulations in order to evaluate SVESM performance. The method outperforms classical methods – such as ESN, Multilayer Perceptron [11], SVM and Self Organizing Maps [12]; except in the prediction of the benchmark Mackey-Glass time series without noise and outliers, in which case ESN performs much better than any other method. However, considering Mackey-Glass time series with noise and outliers, SVESM shows more robustness and outperforms ESN.

C. Profile-Dependent Support Vector Machines

Profile-Dependent SVM (PD-SVM) method arises from a common practice for classification problems with unbalanced data [13]. In this kind of problem, it is possible to define different penalization factors for different classes, aiming at preventing false positives by favoring classes with less samples. The same principle may be applied to time series, since more recent time series' samples may contain more relevant information than older samples.

In PD-SVM, the overall penalization factor C from SVM is adjusted by a time-dependent weighting factor based on a confident function, i.e., a fuction that measures the relevance (or confidence) of certain sample. It is possible to use this penalization factor to implement an exponential memory decay based on the confidence of past samples [13], as

$$c_i = \lambda^{t_n - t_i}, \quad \lambda \in [0, 1],$$

where c_i is a weighting factor of parameter C for the sample at time t_i, t_i is the time for the i-th sample and t_n is the current time.

PD-SVM is also used for classification problems [13]. In this case, a weighting factor may be incorporated, depending on the problem domain, in order to improve accuracy and reduce false detections.

In [13], the classification problem consists in classifying the level of Cyclosporine A in the blood of a pacient. There are three classes: below 150ng/mL, between 150 and 400ng/mL, and above 400ng/mL. As this problem contains temporal aspects, the weighting factor incorporates memory decay, aiming at favoring last samples, and also increases the penalization factor C near the detection border in order to reduce false detections, such suggested in

$$C_{150,t+1} = \frac{1}{150}[k_2 C_t + C_1 c_{t-1} + k_0] \tag{9}$$
$$C_{400,t+1} = \frac{1}{400}[k_2 C_t + C_1 c_{t-1} + k_0],$$

where k_i represent additional penalization factor, which can be fixed *a priori* or computed in an adaptive way [13].

D. Recursive kernels

Recursive kernels operate on two discrete time series $\{\boldsymbol{x}(1), \boldsymbol{x}(2), \cdots, \boldsymbol{x}(n)\}$ and $\{\boldsymbol{y}(1), \boldsymbol{y}(2), \cdots, \boldsymbol{y}(n)\}$[2] [14]. These kernels are associated to an infinite recurrent network, that is, a recurrent network with a hidden layer modeled as a continuous function[3]. A recursive kernel κ at a time t is given by

$$\kappa_t(x, y) = \langle \Phi(x(t), \Phi(x(t-1), \Phi(\cdots))) \cdot \Phi(y(t), \Phi(y(t-1), \Phi(\cdots))) \rangle.$$

where $\Phi(\cdot, \cdot)$ is a nonlinear mapping, $x(t)$ is the current input, and $\Phi(x(t-1), \Phi(\cdots))$ corrresponds to the infinite-dimension state vector, i.e., a nonlinear mapping applied recursively to all past elements of the time series.

In [14], some examples of recursive kernels are presented. These examples consider two parameters: σ for scaling the infinite-dimension state vector, and σ_i for scaling the current input. The stability of these kernels usually depends on the choice of these parameters, as described in [14]. The list of kernels includes:

[2]In this section, x and y are used to designated two time series that are submitted to the kernel function. As this section covers only kernel function and do not cover SVM formulation, it is not necessary to refer to SVM output. Thus, y does not corresponds to an output; it corresponds to one of the inputs submitted to the kernel function.

[3]For further details about infinite recurrent network and its relation to recursive kernels, see [14].

- Recursive linear kernel:

$$\kappa_t(x,y) = \sigma_i^2 \sum_{j=0}^{\infty} \langle \sigma^{2j} x(t-j) \cdot y(t-j) \rangle.$$

- Recursive polynomial kernel:

$$\kappa_t(x,y) = \left[\langle \sigma_i^2 x(t) \cdot y(t) \rangle + \sigma^2 \kappa_{t-1}(x,y) \right]^q.$$

- Recursive gaussian (also known as Radial Basis Function – RBF) kernel:

$$\kappa_t(x,y) = \exp\left(-\frac{\|x(t) - y(t)\|^2}{2\sigma_i^2} \right) \exp\left(\frac{\kappa_{t-1}(x,y) - 1}{\sigma^2} \right).$$

- Recursive arcsine kernel:

$$\kappa_t(x,y) = \frac{2}{\pi} \arcsin\left(\frac{2\left(\sigma_i^2 \langle x(t) \cdot y(t) \rangle + \sigma^2 \kappa_{t-1}(x,y) + \sigma_b^2\right)}{\sqrt{g_t(x) g_t(y)}} \right),$$

with

$$g_t(x) = 1 + 2\left(\sigma_i^2 \|x(t)\|^2 + \sigma^2 \kappa_{t-1}(x,x) + \sigma_b^2\right).$$

It is important to note that SVM with recursive kernel works just as regular SVM. The differences are: the kernel function considering recursion; and the input data which composes the training set must include τ elements of the past of the time series, where τ corresponds to the depth recursion. Some experiments in [14] show the application of recursive kernels to regression and classification problems.

As regression problem, a benchmark in time series processing called NARMA (nonlinear auto regressive moving average) was considered. For comparing results, a windowed approach using SVM with RBF kernel, considering 27 frames as the size of the window, and an ESN were applied to the problem. Both recursive kernels applied (RBF and arcsine with recursion depth of 50 frames) provided better results than classical approaches when all parameters were correctly set, with recursive RBF kernel performing slightly better than the recursive arcsine kernel.

As classification problem, the authors in [14] considered a phoneme recognition problem, aiming at recognizing 39 symbols representing phonemes. Each frame of speech was represented by a 39-dimensional feature vector obtained using Mel frequency cepstral coefficients analysis. In this case, recursive RBF and recursive arcsine kernel were tested against a windowed approach using SVM with RBF kernel considering a window of 9 frames. The recursion depth of the recursive kernels varied according to the experiment, from 5 to 15 frames. The best results in [14] were obtained using the recursive RBF kernel on subsampled data by a factor of 5, i.e., the time series was "speeded up" by a factor of 5, which corresponds to slowing down the effective timescale of the classifier [14].

E. Sequence kernels

In this review, we found also found another type of modified kernel for dealing with temporal aspects of data called sequence kernels, which are capable of dealing with sequences of vectors. One of these sequence kernels is proposed by [15] and used by [16]. This sequence kernel is based on Dynamic Time Warping (DTW) and it is called *polynomial DTW kernel*.

DTW is a distance measure used for calculating distance between two sequences of vectors. According to [15], symmetric DTW consists in considering a local distance between two vectors and a global distance, which is calculated using local distances and, indeed measures the distance between the two sequences of vectors.

Also, it is possible that these two sequences of vector do not have the same length. In this case, it is necessary to perform an alignment between the sequences. Considering A and B as sequences composed of vectors $a(i)$ and $b(i)$, this alignment may be denoted by $\phi_a(i)$ and $\phi_b(i)$ for $i = \{1, \cdots, I\}$, where I is the length of the alignment, and consists in linking each vector in $a(k)$ to a vector in $b(k)$. A DTW alignment distance is given by summiting local distances between the vectors under analysis:

$$D_{align}(A,B) = \frac{1}{I} \sum_{i=1}^{I} d(a_{\phi_a(i)}, b_{\phi_b(k)}).$$

Then, global distance is obtained by calculating the global distance to each possible alignment and considering

$$D(A,B) = \min D_{align}(A,B). \tag{10}$$

In order to build a kernel, DTW global distance has to be converted to a dot product. For this conversion, a technique called Spherical Normalization is used. It consists on projecting the sampled vectors a and b on a unit hypersphere, as described in [15], through[4]

$$\hat{a} = \frac{1}{\sqrt{(a^2 + \alpha^2)}} \begin{bmatrix} a \\ \alpha \end{bmatrix}. \tag{11}$$

By definition, the smallest curve between the two projected vectors $\hat{a}$ and $\hat{b}$ in a hypersphere is given by the angle $\theta_{\hat{a},\hat{b}}$ between these points [15]. Considering $\hat{a} \cdot \hat{b} = \cos \theta_{\hat{a},\hat{b}}$, local DTW distance may be defined as the dot product:

$$d_S(\hat{a},\hat{b}) = \arccos(\hat{a} \cdot \hat{b}). \tag{12}$$

Since it is possible to calculate DTW considering sequences mapped into a hypersphere, it is possible to perform DTW to get the global distance using (10), using (12) as local distance. Then, a linear DTW kernel can be defined through the equation which reconverts DTW global distance to a dot product:

$$K_{linearDTW}(\hat{A},\hat{B}) = \cos D_S(\hat{A} \cdot \hat{B}).$$

[4]Equation 11 is also applied to vector b.

From this kernel, it is also possible to define the polynomial DTW kernel as:

$$K_{polyDTW}(\hat{A}, \hat{B}) = \cos^n D_S(\hat{A} \cdot \hat{B}).$$

Both studies [15] and [16] presents experiments showing that polynomial DTW kernel can promote improvements on accuracy for speech recognition tasks. The study [15] shows that polynomial DTW kernel performs better than traditional DTW and Hidden Markov Models (HMM) for recognizing speech collecting from both normal and dysarthric speakers. For normal speakers, SVM with polynomial DTW kernel performs better when there is a small number of training samples per word; for more than 5 training samples per word, SVM$_{DTW}$ performs as well as HMM and better then DTW. For dysarthric speakers, SVM$_{DTW}$ performs better than HMM and DTW for any number of training samples per word. In [16], SVM with polynomial DTW kernel performs better than Multilayer Perceptron neural network and Elman network for recognizing speech from dysarthric speakers. However, its performance is comparable to SVM with RBF kernel: one method outperforms the other depending on speaker intelligibility.

Also, in [15], three other examples of sequence kernels are cited: the Fisher kernel [17], the Generalized Discriminant kernel [18] and the pair HMM kernel [19]. The Fisher kernel incorporates a generative model into a discriminant classifier, by generating a kernel using the Fisher Score on the choosen generative model. The Generalized Linear Discriminant Sequence (GLDS) kernel is a sequence kernel based on the Generalized linear Discriminant Scoring and on a classifier. Finally, the pair HMM kernel is based on converting the matching function of pair HMM into a dot product for using it as a kernel.

V. Final Considerations

This paper presented a review of research which combines SVM and some kind of temporal reasoning to provide strategies for data analysis that join the advantages of SVM with the ability to explore temporal dependencies among data. Although most strategies have been applied to time-series forecasting [7], [9], [8], [10], [13], [4], there are also the possibility to apply it for temporal data classification and regression [13], [14], [15], [16], [17], [18], [19]. The main contribuition of each strategy presented here can be summarize as follows:

- RLS-SVM [7]: modifies SVM in order to incorporate recurrence, including the modification of the equations that define SVM operation to consider the error for previous output in SVM training;
- Regularized RLS-SVM [9]: introduces a way to balance errors minimization with margin maximization, which also corresponds to controlling VC-dimension, dealing with a negative aspect of RLS-SVM;
- Multidimensional RLS-SVM [8]: converts a multidimensional problem into m unidimensional problems, allowing multidimensional time series forecasting;
- RLS-SVM with Mixed Kernel [10]: combines kernels with different characteristics, in order to provide more efficient non-linear mappings to RLS-SVM;
- SVESM [4]: uses a reservoir to treat the temporal information, performing an explicit non-linear mapping instead of the implicit mapping performed by kernels;
- PD-SVM [13]: assigns greater weights to more recent data, in a context of sequential events, creating a new viewpoint to analyze a dataset;
- Recursive Kernels [14]: transforms the time series under analysis in a set of series organized in a training dataset, performing the analysis in a recursive way; it can be applied to classificaton tasks using a windowed approach to build the training dataset;
- Sequential Kernels [15], [16], [17], [18], [19]: enables the implict treatment of sequences of vectors, e.g., multidimensional time series.

Considering the objective of each strategy described here, note that techniques related to RLS-SVM aim mainly at time series forecasting.This phenomenon occurs because RLS-SVM techniques work with output in the same domain as inputs, since delayed outputs are used to compose the input vector. The same does not occur with PD-SVM, SVESM and modified kernels, since they incorporate temporal reasoning in data processing, but do not use delayed outputs as input; thus, these techniques can be used to provide outputs from any domain.

Also, it is possible discuss about how the strategies modify the SVM operation. PD-SVM only balances the penalization factor C using a time-dependent weighting factor, without altering any other aspect of SVM training. In SVESM, the SVM operation also does not change, since the main difference is that the non-linear mapping, originally performed by kernels in tradition SVM approachs, is performed by a reservoir. Modified kernels also do not modify SVM operation, since only the kernel is affected. On the other hand, RLS-SVM and its variations modify SVM formulation and operation in order to process recurrence.

As we have shown in this review, incorporating temporal reasoning to SVM has provided interesting and useful strategies for temporal data analysis. In this context, we hope that this review supplies a basis on the subject, aiming at supporting the development of others studies which can provide further insights on the related area. Some initiatives that will deserve our attention in our future efforts of reaseach are:

- a practical study around the approach discussed here, applying them on benchmarks classification, regression or prediction problems, in order to support further analysis about the advantages and limitations of each

approach;
- an specific study about the feasibility of the approaches discussed here in a gesture analysis problem (area of interest to the present paper's authors).

From these future works, the authors aim to achieve enough expertise to propose some directions about when each approach is more useful or feasible, and, finally, to contribute with the improvement of the relation between SVM, or even more general Machine Learning approaches, with temporal data processing.

Acknowledgement

The authors thank São Paulo Research Foundation (FAPESP), Brazil, for its financial support through process number 2011/04608-8. Also, the authors thank PhD Alexandre Ferreira Ramos, for the interesting insights about some mathematical derivations used during the writing of this paper.

REFERENCES

[1] J.-C. Chappelier and A. Grumbach, "Time in neural networks," *SIGART Bulletin*, vol. 5, no. 3, pp. 3–11, jul. 1994.

[2] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 1999.

[3] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge, UK: Cambridge University Press, mar. 2000.

[4] Z. Shi and M. Han, "Support vector echo-state machine for chaotic time-series prediction," *IEEE Trans. on Neural Networks*, vol. 18, no. 2, pp. 359 –372, mar. 2007.

[5] J. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Processing Letters*, vol. 9, pp. 293–300, jun. 1999.

[6] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, *Least squares support vector machines*. World Scientific Pub., 2003.

[7] J. A. K. Suykens and J. Vandewalle, "Recurrent least squares support vector machines," *IEEE Trans. on Circuits and Systems-I*, vol. 47, pp. 1109–1114, jul. 2000.

[8] J. Sun, C. Zheng, Y. Zhou, Y. Bai, and J. Luo, "Nonlinear noise reduction of chaotic time series based on multidimensional recurrent ls-svm," *Neurocomputing*, vol. 71, pp. 3675–3679, oct. 2008.

[9] H. Qu, Y. Oussar, G. Dreyfus, and W. Xu, "Regularized recurrent least squares support vector machines," in *International Joint Conference on Bioinformatics, Systems Biology and Intelligent Computing*. IEEE Computer Society, aug. 2009, pp. 508–511.

[10] J. Xie, "Time series prediction based on recurrent ls-svm with mixed kernel," in *Asia-Pacific Conference on Information Processing*, vol. 1, jul. 2009, pp. 113 –116.

[11] F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan, 1962.

[12] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464 –1480, sep 1990.

[13] G. Camps-Valls, E. Soria-Olivas, J. Perez-Ruixo, F. Perez-Cruz, A. Artes-Rodriguez, and N. Jimenez-Torres, "Therapeutic drug monitoring of kidney transplant recipients using profiled support vector machines," *IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 37, no. 3, pp. 359 –372, may 2007.

[14] M. Hermans and B. Schrauwen, "Recurrent kernel machines: Computing with infinite echo state networks," *Neural Computation*, vol. 24, pp. 104–133, jan. 2012.

[15] V. Wan and J. Carmichael, "Polynomial dynamic time warping kernel support vector machines for dysarthric speech recognition with sparse training data," in *Annual Conference of the International Speech Communication Association*, sep. 2005, pp. 3321–3324.

[16] F. Rudzicz, "Phonological features in discriminative classification of dysarthric speech," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, apr. 2009, pp. 4605 –4608.

[17] T. S. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," in *Conference on Advances in Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, nov. 1999, pp. 487–493.

[18] W. M. Campbell, "Generalized linear discriminant sequence kernels for speaker recognition," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, may 2002, pp. I–161 –I–164.

[19] C. Watkins, "Dynamic alignment kernels," University of London, Tech. Rep., 1999.