The Intrinsic Recurrent Support Vector Machine

Daniel Schneegaß[1,2], Anton Maximilian Schaefer[1,3], and Thomas Martinetz[2]

1- Siemens AG, Corporate Technology, Learning Systems,
Otto-Hahn-Ring 6, D-81739 Munich, Germany

2- University of Luebeck, Institute for Neuro- and Bioinformatics,
Ratzeburger Allee 160, D-23538 Luebeck, Germany

3- University of Osnabrueck, Neuroinformatics Group,
Albrechtstraße 28, D-49069 Osnabrueck, Germany

Abstract. In this work, we present a new model for a Recurrent Support Vector Machine. We call it intrinsic because the complete recurrence is directly incorporated within the considered optimisation problem. This approach offers the advantage that the model straightforwardly develops an algorithmic solution. We test the algorithm on several simple time series. The results are promising and can be seen as a starting point for further research. By inventing better and more efficient methods and algorithms, we expect that Recurrent Support Vector Machines could become an alternative to handle and simulate dynamical systems.

1 Introduction

The Support Vector Machine (SVM) [1] is a sophisticated and widely used tool for classification and regression tasks. During the last years many fast and robust methods to calculate the Support Vector solution were invented. In particular we point out the work of Platt [2] and extensions [3]. So far the Sequential Minimal Optimisation algorithm is considered to be the fastest batch learning method in practice, for classification as well as for regression tasks.

There have already been some approaches for Recurrent SVMs in the past. Most remarkably are the works of Suykens and Vandewalle [4] as well as of Schmidhuber et al. [5]. Schmidhuber's model can be seen as a recurrent version of an SVM in the sense that it is a hybrid learning machine composed of a Recurrent Neural Network (RNN) and an SVM. The RNN provides the features for the SVM. Its parameters are determined by artificial evolution [5] whereas the SV solution is calculated by solving the corresponding Quadratic Programming problem. This so-called Evoke algorithm achieves outstanding results.

Our approach substantially differs from Schmidhuber's work. We do not use any other model than an SVM. For learning methods we restrict ourselves to Quadratic Programming and numerical optimisation techniques such as Gradient Descent. Most important, our Recurrent SVM itself contains the full recurrence properties and identifies the underlying dynamical system.

2 Support Vector Machine For Classification

In SVMs, for a given data set $X \subset D \subset \mathbb{R}^n$ and its labels $Y \in \{-1, 1\}^n$, one determines a vector $\mathbf{w}$ and a scalar b which realise an optimal linear classification in the sense of the maximal margin solution using the function $f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$. Apart from its norm there is a unique vector of the form

$$\mathbf{w}^* \quad = \quad \arg \max_{\|\mathbf{w}\|=1, b} \min_i y_i(\mathbf{w}^T\mathbf{x}_i + b),$$

whose class separating hyperplane holds the greatest possible distance to its closest input vectors. It is easy to see that the optimisation problem given as

$$\begin{aligned} \frac{1}{2}\mathbf{w}^T\mathbf{w} \quad &\mapsto \quad \min_{\mathbf{w}} \\ \text{with} \quad y_i\left(\mathbf{w}^T\mathbf{x} + b\right) \quad &\geq \quad 1 \quad \forall\, i \end{aligned}$$

leads exactly to the desired solution [1]. Using the Lagrange formalism it turns out that the solution of the weight vector can be written as $\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i$, where α_i are the Lagrange coefficients and l is the number of training samples. Here every strictly positive α_i represents an active constraint and consequently a Support Vector, which is classified most critically. Hence it is possible to write the classifier finally as

$$f(\mathbf{x}) \quad = \quad \text{sign}\left(\sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b\right).$$

In order to deal with non-linear problems the inner product $\mathbf{x}^T\mathbf{z}$ with $\mathbf{z} \in D$ is usually substituted by a symmetric and positive definite kernel $K(\mathbf{x}, \mathbf{z}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle$ which implicitly transforms the input space into an appropriate feature space.

In the case of data sets which are linearly non-separable within the feature space it is usually necessary to tolerate slight errors by introducing so-called slack variables ξ_i with a regularisation parameter $C > 0$ and modifying the optimisation problem to e. g.

$$\begin{aligned} \mathbf{w}^T\mathbf{w} + C\|\xi\|_2 \quad &\mapsto \quad \min_{\mathbf{w}} \\ \text{with} \quad y_i\left(\mathbf{w}^T\mathbf{x} + b\right) \quad &\geq \quad 1 - \xi_i \quad \forall i. \end{aligned}$$

This error model leads to a slightly modified kernel $K'(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j) + \frac{\delta_{i,j}}{C}$ where δ defines the Kronecker symbol. The kernel is essentially the same as the one which is used in kernelized Ridge Regression [1]. Hence, using this error model, any hard margin Support Vector method can be straightforwardly extended to an appropriate soft margin method. For details we recommend the respective literature, e. g. [1].

3 The Intrinsic Recurrent Support Vector Machine

In dynamical systems, classification or regression does not only depend on the current observation, but also on past time information. Typically, in RNNs, an additional internal state is introduced, representing the memory of the past trajectory. Due to the limited space, we refer to [6] for a good introduction into RNNs and dynamical systems.

Analogous to RNNs the Intrinsic Recurrent Support Vector Machine (IRSVM) needs some kind of internal memory to represent the current state $\mathbf{s}$ of the system. We implement this by introducing an additional weight vector $\hat{\mathbf{w}}$ to the presented standard and non-recurrent SVM, so that f is replaced by $f(\mathbf{x}, \mathbf{s}) = \text{sign}(\mathbf{w}^T\mathbf{x} + \hat{\mathbf{w}}^T\mathbf{s} + b)$, where $\mathbf{x}$ is the current observation and $\mathbf{s}$ the internal state. Further, to develop the internal state in time, we use two operators, e. g. matrices, A and B such that the new internal state is represented by $\mathbf{s}' = A\mathbf{s} + B\mathbf{x}$. Subsequently, we derive the IRSVM using the Lagrange formalism, which is the standard procedure to handle and understand certain types of SVMs.

We start with an extended primal optimisation problem following the 2-norm regularisation as described in section 2 as follows.

$$\begin{aligned} \rho\left(\mathbf{w}, \hat{\mathbf{w}}, A, B\right) = \frac{1}{2}\left(\mathbf{w}^T\mathbf{w} + \hat{\mathbf{w}}^T\hat{\mathbf{w}} + \text{Tr}\left(AA^T\right) + \text{Tr}\left(BB^T\right)\right) &\quad\mapsto\quad \min_{\mathbf{w},\hat{\mathbf{w}},A,B} \\ \text{with}\quad y_i(\mathbf{w}^T\mathbf{x}_i + \hat{\mathbf{w}}^T\mathbf{s}_i + b) &\quad\geq\quad 1 \quad \forall\, i \\ \frac{A\mathbf{s}_{i-1} + B\mathbf{x}_{i-1}}{\|A\mathbf{s}_{i-1} + B\mathbf{x}_{i-1}\|} &\quad=\quad \mathbf{s}_i \quad \forall\, i \end{aligned} \tag{1}$$

Note, that instead of the proposed ρ any other penaliser can be used. In particular we want to mention $2\rho' = \mathbf{w}^T\mathbf{w} + \hat{\mathbf{w}}^T\hat{\mathbf{w}}$, where A and B are unpenalised. The normalisation of $\mathbf{s}_i$ is a simple way to avoid divergence. Therefore no special constraints have to be fulfilled by A and B. Below we write $\mathbf{s}_i$ as a function of A and B to avoid the explicit form of eq. 1. We obtain the Lagrange functional

$$\begin{aligned} L \quad = \quad & \frac{1}{2}\left(\mathbf{w}^T\mathbf{w} + \hat{\mathbf{w}}^T\hat{\mathbf{w}} + \text{Tr}\left(AA^T\right) + \text{Tr}\left(BB^T\right)\right) \\ & - \sum_{i=1}^{T} \alpha_i \left(y_i\left(\mathbf{w}^T\mathbf{x}_i + \hat{\mathbf{w}}^T\mathbf{s}_i\left(A, B\right) + b\right) - 1\right) \end{aligned} \tag{2}$$

with the Lagrange coefficients α. Its partial derivatives are

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{T} \alpha_i y_i \mathbf{x}_i = 0, \qquad \frac{\partial L}{\partial \hat{\mathbf{w}}} = \hat{\mathbf{w}} - \sum_{i=1}^{T} \alpha_i y_i \mathbf{s}_i\left(A, B\right) = 0,$$

and $\frac{\partial L}{\partial b} = \sum_{i=1}^{T} \alpha_i y_i = 0$ for the bias. Therefrom, by replacing the inner product $\mathbf{x}^T\mathbf{z}$ with a symmetric and positive definite kernel function, we derive the

kernelized versions

$$\mathbf{w}^T\mathbf{x} = \sum_{i=1}^{T} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) \tag{3}$$

$$\hat{\mathbf{w}}^T\mathbf{s} = \sum_{i=1}^{T} \alpha_i y_i K'\left(\mathbf{s}_i\left(A,B\right),\mathbf{s}\right) \tag{4}$$

where $K(\mathbf{x}_i,\mathbf{x}_j) = \Phi(\mathbf{x}_i)^T\Phi(\mathbf{x}_j)$ and $K'(\mathbf{s}_i,\mathbf{s}_j) = \Phi'(\mathbf{s}_i)^T\Phi'(\mathbf{s}_j)$ with Φ and Φ' transforming $\mathbf{x}$ and $\mathbf{s}$ implicitly into two appropriate feature spaces. As in standard SVMs the kernel replaces the usual inner product. By substituting eq. 3 and 4 into the Lagrangian we obtain

$$\begin{aligned} L = & \frac{1}{2}\left(\mathbf{w}^T\mathbf{w} + \hat{\mathbf{w}}^T\hat{\mathbf{w}} + Tr\left(AA^T\right) + Tr\left(BB^T\right)\right) \\ & - \sum_{i=1}^{T}\alpha_i\left(y_i\left(\sum_{j=1}^{T}\alpha_j y_j K\left(\mathbf{x}_j,\mathbf{x}_i\right)\right.\right. \\ & \left.\left. + \sum_{j=1}^{T}\alpha_j y_j K'\left(\mathbf{s}_j\left(A,B\right),\mathbf{s}_i\left(A,B\right)\right) + b\right) - 1\right) \end{aligned}$$

and get the further partial derivatives

$$\frac{\partial L}{\partial A} = A - \sum_{i=1}^{T}\alpha_i y_i \sum_{j=1}^{T}\alpha_j y_j \frac{\partial}{\partial A} K'\left(\mathbf{s}_j\left(A,B\right),\mathbf{s}_i\left(A,B\right)\right) = 0 \tag{5}$$

$$\frac{\partial L}{\partial B} = B - \sum_{i=1}^{T}\alpha_i y_i \sum_{j=1}^{T}\alpha_j y_j \frac{\partial}{\partial B} K'\left(\mathbf{s}_j\left(A,B\right),\mathbf{s}_i\left(A,B\right)\right) = 0 \tag{6}$$

which is equal to

$$A = \frac{\partial}{\partial A}\left\|\hat{\mathbf{w}}\left(A,B\right)\right\|^2, \qquad B = \frac{\partial}{\partial B}\left\|\hat{\mathbf{w}}\left(A,B\right)\right\|^2.$$

Alternatively, if we take ρ' as a penaliser, this system of equations reduces to

$$\left\|\hat{\mathbf{w}}\left(A,B\right)\right\|^2 \mapsto \min_{A,B}$$

with modifiable A and B. However, independent of the penaliser used we finally obtain

$$f(\mathbf{x},\mathbf{s}) = \sum_{i=1}^{T}\alpha_i y_i\left(K(\mathbf{x}_i,\mathbf{x}) + K'\left(\mathbf{s}_i\left(A,B\right),\mathbf{s}\right)\right) \tag{7}$$

as a classifier where the subsequent internal state $\mathbf{s}'$ is given as

$$\mathbf{s}' = A\mathbf{s} + B\mathbf{x}. \tag{8}$$

In our derivation we restricted ourselves to classification. Apparently the regression case can be derived in a quite similar way.

4 Initial Algorithm and First Results

For simplicity we keep the description of the proposed algorithm (alg. 1) in a very general form and leave out the details of the numerical techniques we used. By combining Quadratic Programming with Gradient Descent we are able to construct a simple concrete algorithm which solves the subproblems given by the dual form [1] of eq. 2 (wrt. $\mathbf{w}$ and $\hat{\mathbf{w}}$) as well as eq. 5 and 6 (wrt. A and B) alternately. Thereby the SV solution is always calculated using the LibSVM toolbox [7] while the adaptation of A and B is obtained by a one-step Gradient Descent modification in order to solve eq. 5 and 6. The development of more sophisticated and efficient optimisation techniques for this model is an important goal of future work.

Algorithm 1 The Intrinsic Recurrent Support Vector Machine Algorithm

Require: given set $\{X, Y\}$
Ensure: calculates maximal separating hyperplane within feature space incorporating an internal dynamics
  set $\mathbf{s}_0$ appropriately, A and B randomly, and $\forall i : \alpha_i = 0$
  while the desired precision is not reached **do**
    calculate $\forall i : \mathbf{s}_i$ according to eq. 1
    solve the Support Vector Classification problem given by the training set $\{\hat{X}, Y\}$ with $\hat{\mathbf{x}}_i = (\mathbf{x}_i, \mathbf{s}_i)$ using $K_{MSV}\left(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j\right) = K(\mathbf{x}_i, \mathbf{x}_j) + K'\left(\mathbf{s}_i, \mathbf{s}_j\right)$
    find the common solution of eq. 5 and 6 wrt. A and B
  end while
  set f with appropriate bias b as described in [1] according to eq. 7 and 8

We tested our method on five simple time series of the following two types:

- **Summing**
  The task is to classify, observing $\mathbf{x}_i$, whether $\mathbf{x}_{i-r}^1 + \cdots + \mathbf{x}_{i-r}^n + \cdots + \mathbf{x}_i^1 + \cdots + \mathbf{x}_i^n \geq c$. In this example the classification apparently depends directly on r predecessor states. We define Simple Summing ($r = 1$), Medium Summing ($r = 2$) and Hard Summing ($r = 3$).

- **Superimposed Sine Prediction**
  The task is to classify, observing $x_i = \sum_{j=1}^{r} \sin(a_j t_i + d_j)$ with $t_i = i$, whether $x_{i+1} = \sum_{j=1}^{r} \sin(a_j t_{i+1} + d_j) \geq c$. We define Sine Prediction ($r = 1$) and Two Superimposed Sine Prediction ($r = 2$).

Due to the linear separability within the input space we applied the standard inner product, the linear kernel. Fig. 1 shows results of the classification performance on the test data, each averaged over a hundred trials, dependent on the dimensionality of the input space for a constant regularisation setting ($C = 15$). As expected, the performance increases with higher dimensionality of the internal state, but ultimately decreases if the dimensionality gets inadequately high for the regarded problems. This observation is an indicator for overfitting. Hence the determination of the optimal dimensionality, i.e., the amount of information transport needed, can be seen as a further regularisation technique.

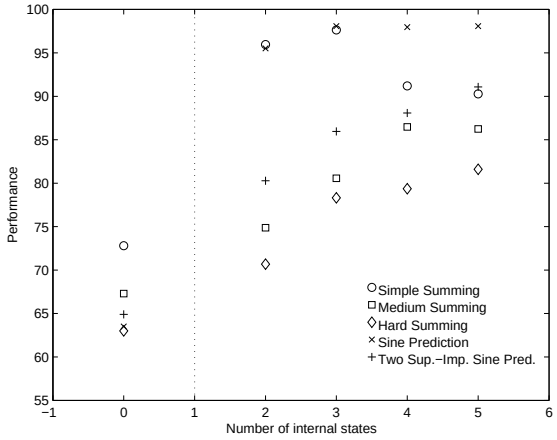

Fig. 1: Performance for different benchmarks and internal state dimensionalities.

5 Conclusion

In this paper we presented the IRSVM as a new model for a Recurrent SVM. Our main aim was to point out, that recurrence and the capability to describe dynamical systems can be realised by SVMs itself. However, the results we achieved so far are quite promising and leave potential for future research. Our focus will be on the advancement of the existing optimisation techniques, so that we do not necessarily depend on Gradient Descend. We will further analyse the convergence properties of our model to set it on mathematical foundations.

References

[1] Nello Cristianini and John Shawe-Taylor. *Support Vector Machines And Other Kernel-based Learning Methods*. Cambridge University Press, Cambridge, 2000.

[2] John C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in kernel methods: support vector learning*, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

[3] S. Sathiya Keerthi, Shirish Krishnaj Shevade, Chiranjib Bhattacharyya, and K. R. K. Murthy. Improvements to platt's smo algorithm for svm classifier design. *Neural Computation*, 13(3):637–649, 2001.

[4] J. A. K. Suykens and J. Vandewalle. Recurrent least squares support vector machines. *IEEE Transactions on Circuits Systems*, 47:1109–1114, 2000.

[5] Juergen Schmidhuber, Matteo Gagliolo, Daan Wierstra, and Faustino Gomez. Evolino for recurrent support vector machines. In *Proc. of the European Symposium on Artificial Neural Networks*, pages 593–598, 2006.

[6] H. G. Zimmermann and R. Neuneier. *A Field Guide to Dynamical Recurrent Networks*, chapter Neural Network Architectures for the Modeling of Dynamical Systems, pages 311–350. 2001.

[7] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/˜cjlin/libsvm.