Predicting the Nonlinear Dynamics of Biological Neurons using Support Vector Machines with Different Kernels

Thomas Frontzek, Thomas Navin Lal, Rolf Eckmiller

University of Bonn
Department of Computer Science VI
Roemerstr. 164, D 53117 Bonn, F. R. Germany
{frontzek, lal, eckmiller}@nero.uni-bonn.de

Abstract

Based on biological data we examine the ability of Support Vector Machines (SVMs) with gaussian, polynomial and tanh-kernels to learn and predict the nonlinear dynamics of single biological neurons. We show that SVMs for regression learn the dynamics of the pyloric dilator neuron of the australian crayfish, and we determine the optimal SVM parameters with regard to the test error. Compared to conventional RBF networks and MLPs, SVMs with gaussian kernels learned faster and performed a better iterated one-step-ahead prediction with regard to training and test error. From a biological point of view SVMs are especially better in predicting the most important part of the dynamics, where the membranpotential is driven by superimposed synaptic inputs to the threshold for the oscillatory peak.

1 Introduction

Modeling on the basis of biological data extracted via intracellular recording is still a challenging task. A biological neural network well studied by physiologists is the stomatogastric ganglion (STG) of crayfish. This neural network consists of a manageable number of specifiable individual neurons. In the literature the dynamics of single neurons or small parts of the STG is modeled on different levels of detail [4]. On the most detailed level ion channels of the membrane of individual neurons are considered for conductance-based Hodgkin-Huxley-models [3]. So called "neuronal caricatures" are on a more abstract level. Here, the dynamics of a neuron are reduced to a few differential equations. Coupled simple binary neurons build the other end of the modeling hierarchy.

Within our paper we focus on the measured time series of specified individual neurons of the STG to model their dynamics. Former studies show that the Support Vector Machine (SVM) technique is very well suited for learning and predicting nonlinear time series. Applied to different benchmarks SVMs perform better than established regression techniques [5][6]. In contrast to those approaches we apply SVMs to biological data and compare the results generated by SVMs with different kernels and the results of conventional RBF networks and MLPs.

2 The stomatogastric ganglion of crayfish

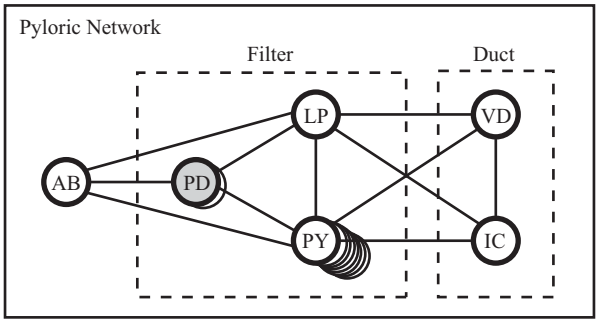

Figure 1: Pyloric network of crayfish. This network consists of 14 cells: 13 motor neurons of five different cell types (two PD, eight PY, LP, VD, IC) and one interneuron (AB). The synaptic connections are illustrated schematically. Within our paper we focus on the activity of the pyloric dilator (PD) neuron.

The stomatogastric ganglion (STG) of crayfish is a model system for biological neural networks. A manageable number of specifiable individual neurons generates several rhythmic motor patterns. The STG consists of the gastric network, which controls the movement of the gastric teeth, and the pyloric network (see fig.1), which controls the movement of the pyloric filter and the cardiopyloric duct.

The pyloric rhythm is driven by the anterior burster (AB) interneuron, which is able to oscillate au-

tonomously. Three phases of the pyloric rhythm can be distinguished: pyloric dilator (PD) neuron, lateral pyloric (LP) neuron, and pyloric (PY) neuron. These neurons are connected with electrical and (mostly inhibitory) chemical synapses.

One can measure the activity of the pyloric network neurons in two different ways: either potentials of single neurons via intracellular recording, or the sum potential of PD, LP and PY via extracellular recording at the dorsal ventricular nerve (dvn). Within our paper we focus on the intracellular measured activity of the PD neuron (see fig.2).

3 Learning and predicting time series with SVMs

Let T be some time series that is generated by an unknown mapping $f : \mathbb{N} \to \mathbb{R}^d$:

$$T = (f(1), f(2), ..., f(n)) =: (x_1, x_2, ..., x_n).$$

Often there is no or little information available about f, so x_{n+1} can only be determined by the information included in T. One common approach is:

1. choose $m, h \in \mathbb{N}$ and a set $L \subseteq \{1, ..., n\}$,
2. build training data $D = \left\{ \left(\left(x_{i-mh}, ..., x_{i-2h}, x_{i-h} \right), x_i \right) \in \mathbb{R}^{md} \times \mathbb{R}^d \mid i \in L \right\}$,
3. choose a class of functions $F \subset \left\{ f : \mathbb{R}^{md} \to \mathbb{R}^d \right\}$,
4. try to find a minimizer $\widehat{f}$ of $I : F \to \mathbb{R}$, $I[f] := \sum_{(x,y)\in D} V(y, f(x))$ with some appropriate loss function V,
5. set $\widetilde{x}_{n+1} := \widehat{f}\left(x_{n+1-mh}, ..., x_{n+1-2h}, x_{n+1-h}\right)$.

Minimizing I using SVMs with the ϵ-*insensitive* loss function:

$$V(y, f(x)) := \begin{cases} 0, & \text{if } |y - f(x)| \le \epsilon, \\ |y - f(x)| - \epsilon, & \text{otherwise.} \end{cases}$$

leads to the quadratic minimization problem:

$$\min_{\alpha, \alpha^* \in A_\lambda} \left\{ \frac{1}{2} \sum_{i,j=1}^{|L|} \left(\alpha_i^* - \alpha_i\right)\left(\alpha_j^* - \alpha_j\right) K(x_i, x_j) - \sum_{i=1}^{|L|} \alpha_i^*(y_i - \epsilon) - \alpha_i(y_i + \epsilon) \right\}$$

with $K : \mathbb{R}^{2d} \to \mathbb{R}$ satisfying the Mercer-condition, $\sum_{i=1}^{|L|}(\alpha_i^* - \alpha_i) = 0$, $A_\lambda := \{\alpha \in \mathbb{R}^{|L|} : 0 \le \alpha_i \le \frac{1}{\lambda}, 1 \le i \le |L|\}$, ($\lambda$ regularization parameter), which is equivalent to:

$$\min_{\beta \in \mathbb{R}^{2|L|}} \left\{ -c^t\beta + \frac{1}{2}\beta^t H \beta \right\}, \quad \text{with } A\beta = b, l \le \beta \le u.$$

With the matrix H, which consists of pairwise scalar products of the input data in the feature space:

$$H = \begin{pmatrix} \vdots & & \vdots & \\ \dots & +K(x_i, x_j) & \dots & -K(x_i, x_j) & \dots \\ \vdots & & \vdots & \\ \dots & -K(x_i, x_j) & \dots & +K(x_i, x_j) & \dots \\ \vdots & & \vdots & \end{pmatrix}$$

And with $c, \beta, A, b, l, u, (x_i, y_i)$ as follows:

$$c = \begin{pmatrix} -y_1 - \epsilon \\ \vdots \\ -y_{|L|} - \epsilon \\ +y_1 - \epsilon \\ \vdots \\ +y_{|L|} - \epsilon \end{pmatrix}, \quad \beta = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_{|L|} \\ \alpha_1^* \\ \vdots \\ \alpha_{|L|}^* \end{pmatrix}, \quad A = \begin{pmatrix} 1 \\ \vdots \\ 1 \\ -1 \\ \vdots \\ -1 \end{pmatrix},$$

$$b = 0, \quad l = 0, \quad u = \frac{1}{\lambda}\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}, \quad (x_i, y_i) \in T.$$

4 Results

4.1 Biological data

We used data from the pyloric network of the australian crayfish *cherax destructor albidus*. The intracellular recordings were done using glass-microelectrodes, filled with $3MKCl$ solution and with resistors ranging from $16M\Omega$ to $25M\Omega$. The data was recorded with an A-D-converter and a video-8-recorder[1].

To reduce the amount of data we replaced every 1000 consecutive datapoints by their average. Then we scaled the results to fit into $[-0.9, 0.9]$. Our training set consisted of 200 patterns (patterns $21 - 220$ of fig.2) and the test set of 100 patterns (patterns $221 - 320$ of fig.2). For our iterated one-step-ahead prediction we wanted to learn the dynamics of the given time series from the last

[1] Recordings were provided by G. Heinzel, H. Boehm, C. Gutzen, Department of Neurobiology, University of Bonn.

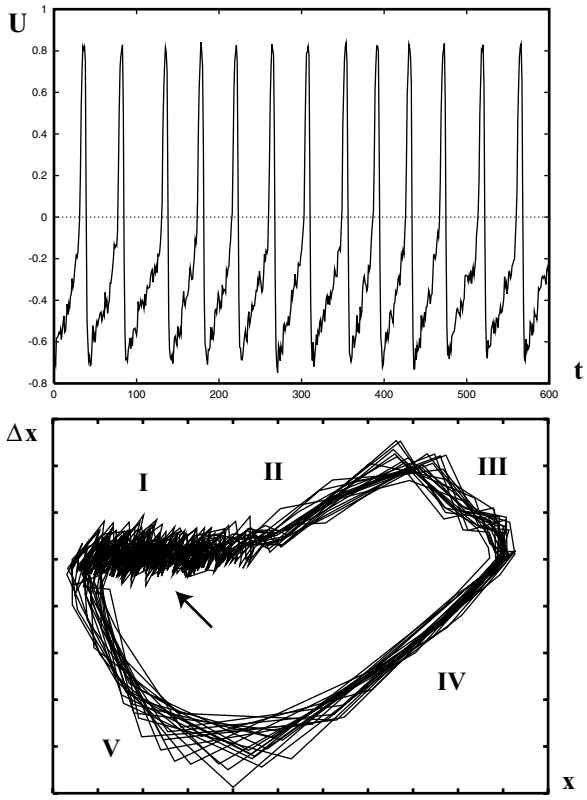

Figure 2: Time series of the pyloric dilator neuron (above), and phase portrait of its dynamics (down). One can distinguish five sections within the phase portrait, marked I to V. Section I is most important, because here the membranpotential is driven by superimposed synaptic inputs to the threshold for the oscillatory peak (sections II to V).

20 data points and we therefore set $m = 20$, $h = 1$, and $L = \{21, 22, ..., 220\}$.

To test the generalization ability of our approximator A (SVM, RBF, MLP), which was trained to learn the data $(x_{21}, x_{22}, \ldots, x_{220})$, we recursivly set

$$\begin{aligned} \widetilde{x}_i &:= x_i, \\ & \quad i = 1..20 \\ \widetilde{x}_{20+i} &:= A(\widetilde{x}_{(20+i)-20}, \widetilde{x}_{(20+i)-19}, \ldots \widetilde{x}_{(20+i)-1}), \\ & \quad i = 1..300 \end{aligned}$$

and then calculated the test error

$$E[A] := \frac{1}{100} \sum_{i=221}^{320} |\widetilde{x}_i - x_i|. \qquad (1)$$

4.2 Simulations with Support Vector Machines

To calculate our SVMs with gaussian kernel

$$K(x, y) = exp\ (\frac{- < x, y >}{\sigma^2})$$

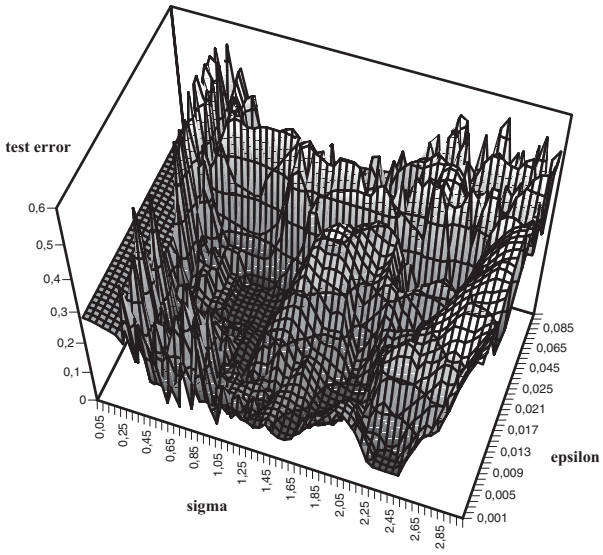

Figure 3: Test error of SVM_{gauss} with gaussian kernel subject to variance σ and exactness ϵ. One can observe a valley of low error values as well as several peaks of very low error values for certain combinations of σ and ϵ.

we adapted the *LOQO* algorithm by Vanderbei [7] which was implemented by Smola and set the regularization parameter to $\lambda = 0.05$.

Varying the variance σ from 0.05 to 100 and the exactness ϵ from 0.001 to 0.8 we performed a large number of simulations. After the learning process we calculated the training and test errors (see equation 1). Fig.3 shows the test error for the most interesting area, which is $\sigma = [0.05, 3.0]$ and $\epsilon = [0.001, 0.01]$.

Regarding the test error the best SVM_{gauss} we found was SVM_{gbest} with $\sigma = 2.85$ and $\epsilon = 0.04$. SVM_{gbest} used 196 support vectors and had a test error of 0.04622. Fig.6a shows the phase portrait of the predicted values of SVM_{gbest}.

We also performed simulations with the *LOQO* algorithm using other than the gaussian kernel. As we expected *tanh*-kernels and polynomial kernels showed worse results. This could be explained by the more global properties of those functions.

Using the *LIBSVM* package from Chang and Lin we verified our results for SVM_{gauss} and obtained results for SVMs with polynomial kernel

$$K(x, y) = (s < x, y > + r)^d.$$

Varying s from 0.3 to 0.5, r from 0.5 to 0.7, and dimension d from 3 to 6 we got discontinuous results: one the one hand stable values for a small area (see fig.4),

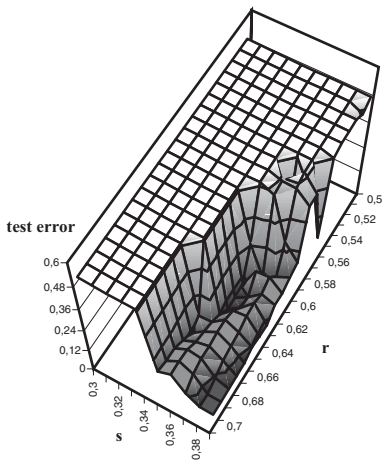

Figure 4: Test error of SVM_{poly} with polynomial kernel subject to s and r, with $d = 3$. Within the area $s = [0.34, 0.39]$ and $r = [0.54, 0.70]$ one can observe two channels of low error values. Outside of the plotted area we obtained discontinuous results.

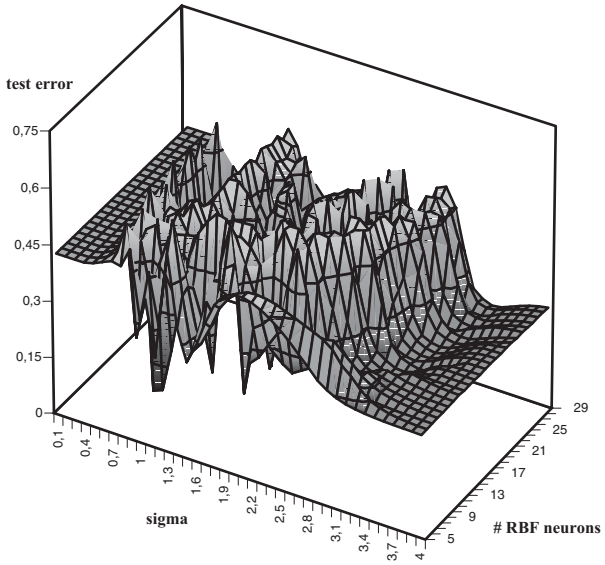

Figure 5: Test error of RBF network subject to variance σ and the number of RBF neurons. The landscape of test error has two plateaus, one with $\sigma < 0.5$ and one with $\sigma > 3.5$. In between one can observe some mountain ranges of high error values as well as several peaks of very low error values.

on the other hand values of infinity for certain combinations of s, r, and d. SVMs with *tanh*-Kernels again showed worse results.

Regarding the test error the best SVM_{poly} we found was SVM_{pbest} with $s = 0.36$, $r = 0.62$ and $d = 3$. SVM_{pbest} used 200 support vectors and had a test error of 0.07311. Fig.6b shows the phase portrait of the predicted values of SVM_{pbest}.

4.3 Simulations with RBF networks and MLPs

To train our RBF networks we first of all determined the centers of the RBF neurons by a *k-means-clustering* algorithm for a given k. Then the weights of the output layer were set randomly in $[-0.0001, 0.0001]$. Finally, these weights were learned using delta rule. We stopped the learning process when the change of error was less than 10^4 for the last 20 epochs, compared to the actual epoch.

Varying the variance σ from 0.1 to 100 and the number of RBF neurons from 5 to 200 we again performed a large number of simulations. After the learning process we calculated the training and test errors (see equation 1). Fig.5 shows the test error for the most interesting area, which is $\sigma = [0.1, 4.0]$ and number of RBF neurons $= [5, 29]$.

Regarding the test error the best RBF network we found was RBF_{best} with 26 RBF neurons and $\sigma = 0.7$. RBF_{best} had a test error of 0.07295. Fig.6c shows the phase portrait of the predicted values of RBF_{best}.

Additionally, we calculated conventional MLPs with one and two hidden layers using standard backpropagation of error algorithm. We stopped the learning process when the change of error was less than 10^4 for the last 20 epochs, compared to the actual epoch. Varying the number of hidden neurons from 2 to 50 (one hidden layer, see fig.7c), the best MLP we found was MLP_{best} with 4 hidden neurons and a test error of 0.11340. Fig.6d shows the phase portrait of the predicted values of MLP_{best}. The best MLP with two hidden layers had a 20-5-7-1-topology and a test error of 0.22592. Generally, MLPs with two hidden layers were not better than MLPs with one hidden layer.

4.4 Discussion

Fig.6 shows the phase portraits of the predicted values of our best SVM_{gauss}, SVM_{poly}, RBF network, and MLP. Regarding the test error and the learning time (see also table 1) SVMs with gaussian kernel clearly outperform the other neural network types.

Compared to the dynamics of the original PD timeseries (see fig.2 (down)), SVMs with gaussian kernel are especially better in predicting the biologically most important part of the dynamics, where the membranpotential is driven by superimposed synaptic inputs to the threshold for the oscillatory peak (section I of fig.6a). The predicted dynamics of SVMs with polynomial kernel shows a slight drift during prediction (fig.6b). RBF networks

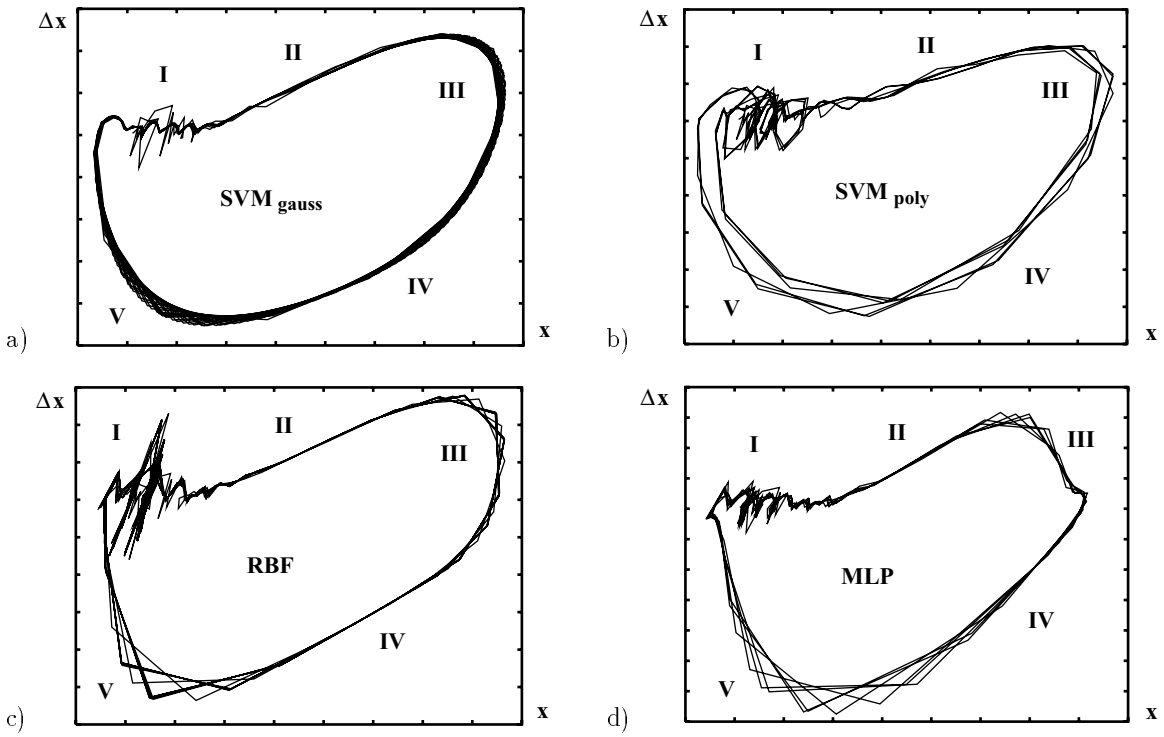

Figure 6: Comparison of the phase portraits of the predicted dynamics using different types of neural networks. SVMs with gausssian kernels (a) were especially better in predicting the most important part of the dynamics (section I), where the membranpotential is driven by superimposed synaptic inputs to the threshold for the oscillatory peak. The structure of the oscillatory peak (sections II to V) is predicted best by MLPs (d).

Table 1: Comparison of lowest errors and fastest (normalized) learning times.

	SVM_{gauss} (# support vectors)	RBF (# RBF neurons)	SVM_{poly} (# support vectors)	MLP (# hidden neurons)
training error	0.02752 (200)	0.17105 (18)	0.16400 (175)	0.19953 (34)
test error	0.04622 (196)	0.07295 (26)	0.07311 (200)	0.11340 (04)
error sum	0.19868 (198)	0.33106 (09)	0.29613 (199)	0.34236 (04)
learning times	1.0/1.0/1.0	16.6/4.7/1.8	0.1/0.1/0.1	>22.5/8.3/6.3

do a bad section-I-prediction with some outliers, but are quite good within sections II to V (fig.6c). MLPs predict the structure of the oscillatory peak (sections II to V) best (fig.6d).

For further comparison we examined the test error subject to the number of support vectors resp. the number of RBF/hidden neurons. In fig.7 we plotted for every number of support vectors resp. RBF/hidden neurons the lowest test error that was achieved by the corresponding network. Best results for SVMs were obtained using a large number of support vectors (compared to the number of training patterns) (fig.7a), whereas best results for RBF networks and MLPs were obtained using small hidden layers (fig.7b,7c). Nevertheless, SVMs with about 115 support vectors and even SVMs with about 25 support vectors had test errors which were as low as those of RBF_{best}.

5 Conclusions

On the basis of the results of the previous sections we conclude:

- SVMs with gaussian and polynomial kernels were able to learn the nonlinear dynamics of biological data. This was demonstrated with the dynamics of the pyloric dilator neuron of the crayfish *cherax destructor albidus*. SVMs with *tanh*-kernels were not able to learn the PD dynamics.

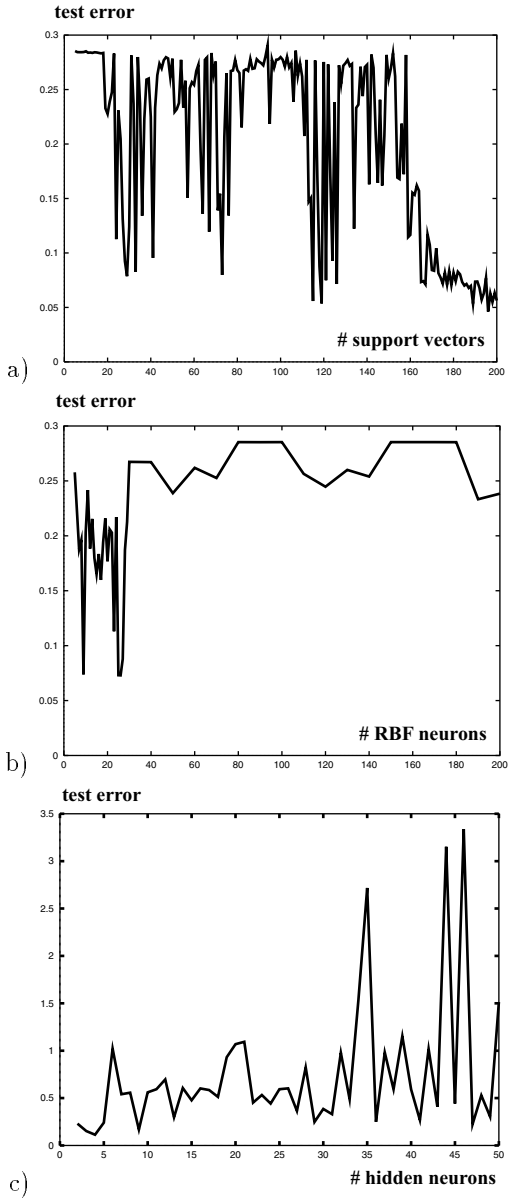

Figure 7: Test error subject to the number of support vectors of SVMs with gaussian kernel (a), to the number of RBF neurons of RBF networks (b), and the number of hidden neurons of MLPs with one hidden layer (c). For every number of support vectors resp. RBF neurons resp. hidden neurons we plotted the lowest test error that was achieved by the corresponding network. Note the different scales for the test error of (a) and (b) compared to (c).

- Compared to conventional RBF networks and MLPs, SVMs with gaussian kernels performed a better iterated one-step-ahead prediction with regard to training and test error. SVMs with gaussian kernels were especially better in predicting the most important part of the dynamics, where the membranpotential is driven by superimposed synaptic inputs to the threshold for the oscillatory peak. The structure of the oscillatory peak was predicted best by MLPs.
- SVMs learned much faster than RBF networks. RBF networks learned faster than MLPs.
- Best results for SVMs were obtained using a large number of support vectors (compared to the number of training patterns), whereas best results for RBF networks and MLPs were obtained using small hidden layers.

Acknowledgments

Parts of this work have been supported by the Federal Ministry for Education, Science, Research, and Technology (BMBF), project LENI. Thanks to Hartmut Boehm and Mikio L. Braun for useful comments.

References

[1] Juergen A. Donath, Thomas Frontzek, and Rolf Eckmiller. Towards predicting neural net control of macro-econometric multi-compartment models. In *Proceedings of ICANN'99, Edinburgh*, volume 1, pages 341–346, 1999.

[2] Thomas Frontzek, T. Navin Lal, and Rolf Eckmiller. Learning and prediction of the nonlinear dynamics of biological neurons with support vector machines. In *Proceedings of ICANN'2001, Vienna*, 2001.

[3] Jorge Golowasch, Michael Casey, L. F. Abbott, and Eve Marder. Network stability from activity-dependent regulation of neuronal conductances. *Neural Computation*, (11):1079–1096, 1999.

[4] Eve Marder and Allen I. Selverston. Modeling the stomatogastric nervous system. In Ronald M. Harris-Warrick, Eve Marder, Allen I. Selverston, and Maurice Moulins, editors, *Dynamic Biological Networks*, pages 161–196. The MIT Press, 1992.

[5] K.-R. Mueller, A. J. Smola, G. Raetsch, B. Schoelkopf, J. Kohlmorgen, and V. Vapnik. Predicting time series with support vector machines. In *Proceedings of ICANN'97, Lausanne*, pages 999–1004, 1997.

[6] Sayan Mukherjee, Edgar Osuna, and Federico Girosi. Nonlinear prediction of chaotic time series using support vector machines. In *Proceedings of IEEE NNSP'97*, pages 511–519, 1997.

[7] Robert J. Vanderbei. *LOQO: An Interior Point Code for Quadratic Programming.* Technical Report SOR94-15. Princeton University, 1994.

[8] Vladimir N. Vapnik. *Statistical Learning Theory.* John Wiley & Sons, 1998.