Learning recurrent neural models with minimal complexity from neural tuning data

Martin Giese

Dept. of Cognitive Neurology, University of Tübingen, Germany
martin.giese@tuebingen.mpg.de

Abstract

A learning algorithm for the estimation of the structure of nonlinear recurrent neural models from neural tuning data is presented. The proposed method combines support vector regression with additional constraints that result from a stability analysis of the dynamics of the fitted network model. The optimal solution can be determined from a single convex optimization problem that can be solved with semidefinite programming techniques. The method successfully estimates the feed-forward and the recurrent connectivity structure of neural field models using as data only examples of stable stationary solutions of the neural dynamics. The class of neural models that can be learned is quite general. The only *a priori* assumptions are the translation invariance and the smoothness of the feed-forward and recurrent spatial connectivity profile. The efficiency of the method is illustrated by comparing it with estimates based on radial basis function networks and support vector regression.

1 Introduction

A variety of cortical functions can be modeled using relatively simple nonlinear recurrent neural networks (e.g. [2]). The structure of such models is often predefined by the modeler leaving only a small number of parameters unspecified, which can be adjusted either by hand or automatically by applying parameter optimization techniques (e.g. [6]). This approach seems suitable when the basic structure of

[1] The author thanks S. Mukherjee for helpful discussions. This work is supported by the Deutsche Volkswagen Stiftung.

the model can, in advance, be sufficiently constrained based on neurophysiological data. Otherwise, rather heuristic assumptions about the structure of the network connectivity have to be made by the modeler.

A theoretically more satisfying approach determines the network structure directly from the data using a relatively general ansatz for the neural nonlinear network dynamics. In a number of studies techniques from nonlinear system identification, like Wiener and Volterra series have been applied to estimate the parameters of such general dynamic neural models (e.g. [7]). These approaches require typically a large amount of neural data including a sufficient number of transients in order to fit the higher-order kernels. This requirement cannot always be met since gathering large amounts of neural data is quite costly. In cases where no data on the transient relaxations of the neural activity is available modelers often assume that the the measured ”quasistationary” activity within an appropriately chosen time window corresponds to the stable stationary activation states in their neural models.

We propose here a method for the automatic identification of recurrent neural models from neural activity patterns that correspond to stable stationary solutions. By exploiting principles from regularization theory and *structural risk minimization* the amount of required neural data can be kept quite small.

2 Basic model

Our algorithm approximates the data using a spatially continuous neural network (neural field) with the form [1]:

$$\tau\, \dot{u}(x,t) + u(x,t) = \int w(x-x')f(u(x',t))\,\mathrm{d}x' + \int b(x-x')s(x')\,\mathrm{d}x' \qquad (1)$$

The continuous variable x defines the dimension that is encoded by the neurons, e.g. motion direction or orientation. The variable $u(x,t)$ signifies the membrane potential of the model neurons as a function of time. $s(x)$ is an external stimulus that is assumed to be known and constant in time. The nonlinear function $f(u)$ is a monotonic threshold function, typically a linear or a step threshold. The network structure is defined by the continuous functions $w(x)$ and $b(x)$ that define the form of the lateral connectivity and the feed-forward connections. The only assumption about these functions is that they are smooth.

3 Learning problem

The neural data is obtained for Q different stimuli $s^{(q)}(x)$, $1 \leq q \leq Q$. The neural data is given by pairs $[x_l, u_l^{(q)}]$. x_l signifies the parameter value which the neuron encodes, e.g. a certain preferred orientation. The activity value $u_l^{(q)}$ is the activity of the neuron that encodes the value x_l of the stimulus $s^{(q)}(x)$, i.e. $u_l^{(q)} = u(x_l)|s^{(q)}(x)$. Goal of the learning algorithm is to determine the functions $w(x)$ and $b(x)$ in a way that ensures that following three constraints are fulfilled: (1) Good approximation of the neural data by the stationary solutions of the neural model. (2) Smoothness (minimal complexity) of the fitted functions $w(x)$ and $b(x)$. (3) The neural data is approximated by *stable* solutions of the fitted neural dynamics.

4 Algorithm

The algorithm was derived by combining support vector regression [11] with methods from robust control theory *(linear matrix inequalities)* [3]. To simplify the notation in the following we will use the abbreviation $v^{(q)}(x) = f(u^{(q)}(x))$. We assume that the full functional form of this thresholded activation distribution is known. In practice it can be reconstructed from the neural data, e.g. by interpolation or nonlinear regression.

The functions $w(x)$ and $b(x)$ are represented as convolutions of two parameter functions with the Gaussian kernel $g(x)$ (cf. [11]):

$$w(x) = \int \omega(x - x')g(x')\,\mathrm{d}x' \tag{2}$$

$$b(x) = \int \beta(x - x')g(x')\,\mathrm{d}x' \tag{3}$$

To ensure that the functional form of the estimated feed-forward and lateral connections has minimal complexity we apply the *structural risk minimization principle* by minimizing a bound for the VC dimension of these functions. Following derivations in [11] this is possible the minimization of the following error functional:

$$E[\omega, \beta] = \frac{1}{2}\int \omega(x)^2\,\mathrm{d}x + \frac{1}{2}\int \beta(x)^2\,\mathrm{d}x + C\sum_{l,q}(\xi_l^{(q)} + \xi_l^{(q)*}) \tag{4}$$

with the positive constant C subject to the constraints (with $1 \leq l \leq L$ and $1 \leq q \leq Q$):

$$u_l^{(q)} - \int\int \omega(x_l - x' - x'')v^{(q)}(x')g(x'')\,\mathrm{d}x'\mathrm{d}x'' \tag{5}$$
$$- \int\int \beta(x_l - x' - x'')s^{(q)}(x')g(x'')\,\mathrm{d}x'\mathrm{d}x'' + \xi_l^{(q)} + \epsilon \geq 0$$

$$-u_l^{(q)} + \int\int \omega(x_l - x' - x'')v^{(q)}(x')g(x'')\,\mathrm{d}x'\mathrm{d}x'' \tag{6}$$
$$+ \int\int \beta(x_l - x' - x'')s^{(q)}(x')g(x'')\,\mathrm{d}x'\mathrm{d}x'' + \xi_l^{(q)*} + \epsilon \geq 0$$

$$\xi_l^{(q)} \geq 0 \tag{7}$$

$$\xi_l^{(q)*} \geq 0 \tag{8}$$

Using methods from variational calculus this optimization problem can be transformed in *dual form* [11] with solutions of the form

$$\omega(x) = \sum_{l,q}(\alpha_l^{(q)*} - \alpha_l^{(q)})\int g(x_l - x' - x)v^{(q)}(x')\,\mathrm{d}x' \tag{9}$$

$$\beta(x) = \sum_{l,q}(\alpha_l^{(q)*} - \alpha_l^{(q)})\int g(x_l - x' - x)s^{(q)}(x')\,\mathrm{d}x' \tag{10}$$

where the coefficients $\alpha_l^{(q)}$ and $\alpha_l^{(q)*}$ are Lagrange multipliers. The dual optimization problem is a quadratic programming problem that can be written compactly by defining the vectors $\boldsymbol{\alpha}^{(*)} = [\alpha_1^{(1)(*)}, ..., \alpha_L^{(Q)(*)}]^T$ and $\mathbf{u} = [u_1^{(1)}, ..., u_L^{(Q)}]^T$. The following function has to be maximized over the vectors $\boldsymbol{\alpha}^{(*)}$

$$L(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*) = -\frac{1}{2}(\boldsymbol{\alpha}^* - \boldsymbol{\alpha})^T(\mathbf{K}_v + \mathbf{K}_s)(\boldsymbol{\alpha}^* - \boldsymbol{\alpha}) + \mathbf{u}^T(\boldsymbol{\alpha}^* - \boldsymbol{\alpha}) - \epsilon \mathbf{1}^T(\boldsymbol{\alpha}^* + \boldsymbol{\alpha}) \quad (11)$$

with the box constraints:

$$0 \leq \alpha_l^{(q)}, \alpha_l^{(q)*} < C \quad (12)$$

The elements of the $(LQ) \times (LQ)$ matrices $\mathbf{K}_v$ and $\mathbf{K}_s$ are given by

$$(\mathbf{K}_v)_{ll'}^{qq'} = \iiint g(x_l - x' - x) g(x_{l'} - x'' - x) v^{(q)}(x') v^{(q')}(x'')\, \mathrm{d}x\, \mathrm{d}x' \mathrm{d}x''$$
$$(\mathbf{K}_s)_{ll'}^{qq'} = \iiint g(x_l - x' - x) g(x_{l'} - x'' - x) s^{(q)}(x') s^{(q')}(x'')\, \mathrm{d}x\, \mathrm{d}x' \mathrm{d}x''$$

where rows and columns of the matrices run through all combinations of the index pairs (l, q) and (l', q'). A number of additional constraints follows from the requirement that the data should correspond to stable states of the network dynamics. For each stimulus $s^{(q)}(x)$ the stability of the solution can be verified by linearizing the dynamics (1) around the measured neural activity distribution $u^{(q)}(x)$. An approximation for the spatially continuous activity dynamics can be obtained by sampling the field discretely in the points x_r. From this the stability matrix of the linearized dynamics is derived as:

$$\mathbf{A}^{(q)} = -\mathbf{I} + \mathbf{W}\mathbf{D}^{(q)} \quad (13)$$

with $W_{mn} = w(x_m - x_n)$ and $D_{nn}^{(q)} = f'(u^{(q)}(x_r))\Delta x_r$, where Δx_r is the discretization interval along the x dimension. Exploiting equations (2), (3), and (9) it can be shown that this stability matrix is an affine function of the parameters $\alpha_l^{(q)}$ and $\alpha_l^{(q)*}$. For the stability of the dynamics it must be required that the matrix $\mathbf{A}$ has no eigen values with non-negative real part. An equivalent stability condition that is more suitable for an elegant algorithmic implementation is given by the Lyapunov inequality

$$\mathbf{A}^{(q)T}\mathbf{P} + \mathbf{P}\mathbf{A}^{(q)} \leq \mathbf{0} \quad (14)$$

where $\mathbf{P}$ is a positive definite symmetric matrix that determines the speed of the relaxation of perturbations of the stationary solution. This matrix is preassigned to be a multiple of the unit matrix. $\mathbf{X} \leq \mathbf{0}$ means that the matrix $\mathbf{X}$ is negative semi-definite. Obviously, the last equation is also affine in the parameters $\alpha_l^{(q)}$ and $\alpha_l^{(q)*}$. This allows to reformulate the stability conditions as set of linear matrix inequalities of the form:

$$\mathbf{A}_0^{(q)} + \sum_{l,q} (\alpha_l^{(q)*} - \alpha_l^{(q)}) \mathbf{A}_n^{(q)} \geq \mathbf{0} \tag{15}$$

Finally, by application of *Schur complements* [3], the quadratic programming problem (11) and 12) can be converted into a semi-definite programming problem

$$\begin{gathered} \text{minimize} \quad E \quad \text{subject to} \\ \mathbf{B}_0 + E\mathbf{B}_E + \sum_{l,q} (\alpha_l^{(q)*} \mathbf{B}_l^{(q)*} + \alpha_l^{(q)} \mathbf{B}_l^{(q)}) \geq \mathbf{0} \end{gathered} \tag{16}$$

with the new additional scalar parameter E. This together with equations (15) and (16) define a common semi-definite programming problem that, if feasible, defines a solution that fulfills all three constraints formulated in the last section. Semi-definite programming problems are convex, i.e. have a unique solution. In addition, they can be solved efficiently with interior point methods [3,10].

5 Results

The method was evaluated with simulated neural data. A neural field with gaussian feed-forward kernel $b(x)$ and a Mexican hat-shaped lateral interaction function $w(x)$ was simulated. The stable stationary solutions for multiple input signals $s(x)$ (composed from one or multiple delta pulses along the x axes) were used as training data. Figure 1 shows an example for the model fits obtained with three different methods: (1) *standard regularization* by modeling the functions $b(x)$ and $w(x)$ with radial basis function networks (dotted line), (2) *support vector regression* (dashed-dotted line), and (3) the proposed algorithm with *dynamic stability control* (dashed line). Only the proposed method fits a model with stable solutions that are close to the data. All three methods identify models that lead to very small equation errors for the approximation of equation (1). However, the solutions of the methods without dynamic stability control drift away from the data during the iteration of the fitted dynamics. This occurs in particular when the original neural dynamics is close to instability.

6 Discussion

A new learning algorithm was proposed that estimates automatically the structure of a nonlinear recurrent neural network model (neural field) from a sparse set of neural data. The stable solutions of the fitted model approximate the neural data. Only very general *a priori assumptions* about the structure of the model have to made, including the shape of the nonlinearity and smoothness and translation invariance of the connections. Many existing dynamic neural models in neuroscience and technical applications fulfill these assumptions. Neural field models have been automatically fitted before (e.g. [4,5]). The specific property of the proposed method is that it permits an automatic estimation of such models without the knowledge of transients of the neural signals. For a variety of applications in neuroscience such transients are not easily available.

The theoretically interesting aspect of the proposed method is that it combines *structural risk minimization* with constraints resulting from a stability analysis of learned dynamical model. Constraints expressed in terms of *linear matrix inequalities* have been used for the training of recurrent neural networks without exploiting regularization (e.g. [8]). In addition, standard regularization networks have been combined with stability constraints in the context control applications [9]. The proposed approach seems very general and might be extended for other applications that require learning of regularized dynamical models with capacity control.

References

[1] S Amari. Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological Cybernetics*, 27:77–87, 1977.

[2] R Ben-Yishai, R L Bar-Or, and H Sompolinsky. Theory of orientation tuning in visual cortex. *Proceedings of the National Academy of Sciences (USA)*, 92(9):3844–3848, 1995.

[3] S Boyd, L El Ghaoui, E Feron, and V Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. SIAM Studies in Applied Mathematics, Philadelphia, 1994.

[4] M A Giese. *Dynamic Neural Field Theory for Motion Perception*. Kluwer, Boston, 1999.

[5] C Igel, W Erlhagen, and D Jancke. Optimization of dynamic neural fields. *Neurocomputing*, 36:225–233, 2001.

[6] D K Lee, L Itti, C Koch, and J Braun. Attention activates winner-takes -all competition among visual filters. *Nature Neuroscience*, 2:375–381, 1999.

[7] V Marmarelis. Wiener analysis of nonlinear feedback in sensory systems. *Annals of Biomedical Engineering*, 19:345–382, 1991.

[8] J Steil and H Ritter. Recurrent learning of input-output stable behaviour in function space: A case study with the Roessler attractor. In *Proceedings of the ICANN '99*, pages 761–766. IEE, 1999.

[9] J A K Suykens, J Vandewalle, and B De Moor. Optimal control by least squares support vector machines. *Neural Networks*, 14:23–35, 2001.

[10] K C Toh, M J Todd, and R H Tütüncü. SDPT3 — a matlab software package for semidefinite programming. *Optimization Methods and Software*, 11:545–581, 1999.

[11] V N Vapnik. *Statistical Learning Theorey*. John Wiley, New York, 1998.

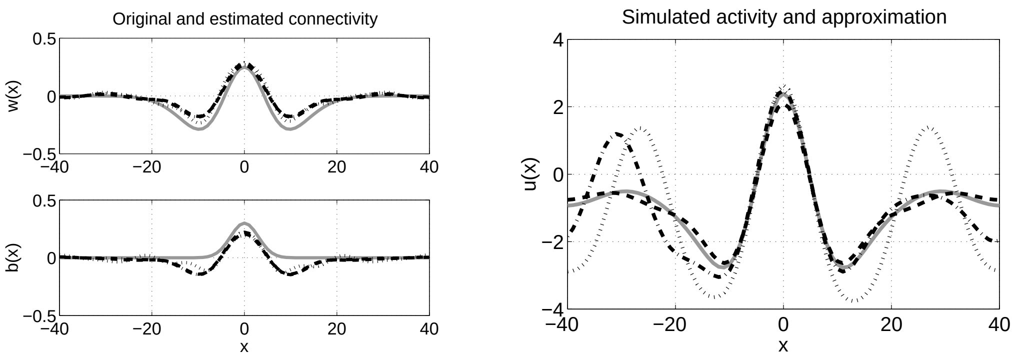

Fig. 1: Estimation results obtained with different methods for simulated data. The left panel shows the estimated kernels describing the feed-forward and feedback connectivity. Right panel shows the stable activity distributions $u(x)$ obtained by simulation of the network using the data as initial condition. The original data is plotted in grey. The dashed curves indicate the estimation results obtained with the proposed method. The results obtained using a standard radial basis function network (dotted lines) and with support vector regression [11] (dashed-dotted lines) are shown for comparison.