Learning recurrent neural models with minimal complexity from neural tuning data

Martin Giese

Dept. of Cognitive Neurology
University Clinic Tübingen
72076 Tübingen, Germany
martin.giese@tuebingen.mpg.de

Abstract

We present a method for the estimation of nonlinear recurrent neural models from neural tuning data. Our algorithm makes no specific assumptions about the structure of the recurrent neural network model, except for the smoothness of the lateral interaction profile and feed-forward input kernel. A new learning algorithm is devised that achieves optimal generalization from small amounts of neural data that combines support vector regression with stability constraints resulting from the recurrent network dynamics, resulting in a semi-definite programming problem. The method is illustrated by fitting models for orientation tuning in the visual cortex.

1 Introduction

Many cortical functions can be modeled with nonlinear recurrent neural networks (e.g. [1, 2]). Typically, the structure of such models is predefined by the modeler, leaving only a small number of parameters, unspecified that can be adjusted either by hand, or automatically by a parameter optimization techniques (e.g. [3]). This approach is well justified if the basic network structure can be sufficiently constrained on the basis of neurophysiological data. If this is not the case, the network structure is often chosen based on heuristic considerations. Theoretically more satisfying seem approaches that determine the network structure directly from the data starting from a relatively general nonlinear angst for the network dynamics. Techniques from nonlinear system identification, such as Wiener and Volleyer series, have been repeatedly applied to neural data (e.g. [4, 5]). Such approaches require typically a relatively large amount of neural data including a sufficient number of transients in order to fit the higher-order kernels. This requirement can not always be fulfilled, since gathering large amounts of neural data is costly. Also, detailed data about transient relaxations of the neural dynamics is often not available. In this case modelers typically make the assumption that the measured neural activities in an appropriately chosen time window correspond to stable stationary activation states of the neural model.

We propose here a method for the automatic approximation of neural data with recurrent neural models that avoids some of these problems. Our method assumes a relatively general recurrent neural network structure. By exploiting principles from regularization theory and statistical learning theory the required amount of neural data can be kept relatively small. The method fits dynamic neural models under the assumption that the neural data corresponds to the stable states of the network dynamics. In this way, recurrent neural models can be fitted for data sets for which no time-resolved traces of the neural activity are available.

2 Basic model

Our algorithm approximates the data using a spatially continuous neural network (neural field) with the form [6]:

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

The continuous variable x defines the dimension that is encoded by the neurons, e.g. 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 codes, e.g. a certain preferred orientation. The activity value $u_l^{(q)}$ is the activity of the neuron that codes the value x_l for the stimulus $s^{(q)}(x)$, i.e. $u_l^{(q)} = u(x_l)|s^{(q)}(x)$. Goal of the algorithm is to determine the functions $w(x)$ and $b(x)$ in a way that ensures that the following constraints are fulfilled:

- good approximation of the neural data by the stationary solutions of the neural model

- smoothness and minimal complexity of the fitted functions $w(x)$ and $b(x)$
- The neural data must be approximated by *stable* solutions of the fitted neural dynamics.

4 Algorithm

Our algorithm was derived by combining methods from statistical learning theory [7] and linear matrix inequalities from control theory [8]. To simplify 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. [7]):

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

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

$$\tag{4}$$

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 [7], this is possible the minimization of the following error functional:

$$E[\omega, \beta] = \frac{1}{2}\int \omega(x)^2\,\mathrm{d}x + \frac{1}{2}\int \beta(x)^2\,\mathrm{d}x + C\sum_{l,q}(\xi_l^{(q)} + \xi_l^{(q)*}) \tag{5}$$

with the positive constant C, subject to the constraints:

$$u_l^{(q)} - \int\int \omega(x_l - x' - x'')v^{(q)}(x')g(x'')\,\mathrm{d}x'\mathrm{d}x'' \tag{6}$$

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

$$-u_l^{(q)} + \int\int \omega(x_l - x' - x'')v^{(q)}(x')g(x'')\,\mathrm{d}x'\mathrm{d}x'' \tag{7}$$

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

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

$$\xi_l^{(q)*} \geq 0 \tag{9}$$

Using methods from variational calculus from this can be derived that the solutions of this problem have the form

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

$$\beta(x) = \sum_{l,q}(\alpha_l^{(q)*} - \alpha_l^{(q)})\int g(x_l - x' - x)s^{(q)}(x')\,\mathrm{d}x' \tag{11}$$

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 $\boldsymbol{\alpha}^{(*)}$:

$$L(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*) = -\frac{1}{2}(\boldsymbol{\alpha}^* - \boldsymbol{\alpha})^T(\mathbf{K}_v + \mathbf{K}_s)(\boldsymbol{\alpha}^* - \boldsymbol{\alpha}) + \mathbf{u}^T(\boldsymbol{\alpha}^* - \boldsymbol{\alpha}) - \epsilon \mathbf{1}^T(\boldsymbol{\alpha}^* + \boldsymbol{\alpha}) \tag{12}$$

with the constraints:

$$0 \leq \alpha_l^{(q)}, \alpha_l^{(q)*} < C \tag{13}$$

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

$$(\mathbf{K}_v)_{ll'}^{qq'} = \int\int\int g(x_l - x' - x) g(x_{l'} - x'' - x) v^{(q)}(x') v^{(q')}(x'') \, \mathrm{d}x \, \mathrm{d}x' \mathrm{d}x'' \tag{14}$$

$$(\mathbf{K}_s)_{ll'}^{qq'} = \int\int\int g(x_l - x' - x) g(x_{l'} - x'' - x) s^{(q)}(x') s^{(q')}(x'') \, \mathrm{d}x \, \mathrm{d}x' \mathrm{d}x'' \tag{15}$$

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 $s^{(l)}(x)$ 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:

$$\mathbf{A}^{(q)} = -\mathbf{I} + \mathbf{W}\mathbf{D}^{(q)} \tag{16}$$

with $W_{mn} = w(x_m - x_n)$ and $D_{nn} = f'(u^{(q)}(x_r))\Delta x_r$, where Δx_r is the discretization interval along the x dimension. Exploiting equations (2) and (10) 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

$$\mathbf{A}^{(q)T}\mathbf{P} + \mathbf{P}\mathbf{A}^{(q)} \leq \mathbf{0} \tag{17}$$

where $\mathbf{P}$ is a positive definite symmetric matrix that determines the speed of the relaxation towards the stationary solution. $\mathbf{X} \leq \mathbf{0}$ 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:

$$\mathbf{A}_0^{(q)} + \sum_{l,q}(\alpha_l^{(q)*} - \alpha_l^{(q)})\mathbf{A}_n^{(q)} \geq \mathbf{0} \tag{18}$$

Finally, by application of Schur complements [8], the quadratic programming problem (12) can be converted into a semi-definite programming problem of the form:

$$\begin{aligned} &\text{minimize} \quad E \quad \text{subject to} \\ &\mathbf{B}_0 + E\mathbf{B}_E + \sum_{l,q}(\alpha_l^{(q)*}\mathbf{B}_l^{(q)*} + \alpha_l^{(q)}\mathbf{B}_l^{(q)}) \geq \mathbf{0} \end{aligned} \tag{19}$$

with the new scalar parameter E. This together with equations (18) and (13) 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 [8, 9].

5 Evaluation of the method

The method for approximation of neural responses by recurrent neural models has been tested with data on orientation tuning in area V1 from the literature. It is demonstrated that inclusion of the stability constraint is crucial. Solutions of models that were fitted without the stability constraint approximate the data well, but can have solutions that drift away from the data for slightly different initial conditions or in presence of noise.

References

[1] D C Somers, S B Nelson, and M Sur. An emergent model of orientation selectivity in cat visual cortical sensitive cells. *Journal of Neuroscience*, 15:5448–5465, 1995.

[2] E Salinas and L F Abott. A model of multiplicative neural responses in parietal cortex. *Proceedings of the National Academy of Sciences (USA)*, 93:11956–11961, 1996.

[3] 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.

[4] W von Seelen and K P Hoffmann. Analysis of neural networks in the visual system of the cat using statistical signals. *Biological Cybernetics*, 22:7–20, 1976.

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

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

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

[8] 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.

[9] 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.