ESANN'2009 proceedings, European Symposium on Artificial Neural Networks - Advances in Computational Intelligence and Learning. Bruges (Belgium), 22-24 April 2009, d-side publi., ISBN 2-930307-09-9.

Learning reconstruction and prediction of natural stimuli by a population of spiking neurons

Michael Gutmann[1] and Aapo Hyvärinen[12] *

1 - Dept of Computer Science and HIIT, University of Helsinki
P.O.Box 68, FIN-00014 University of Helsinki, Finland.

2 - Dept of Mathematics and Statistics, University of Helsinki

Abstract. We propose a model for learning representations of time dependent data with a population of spiking neurons. Encoding is based on a standard spiking neuron model, and the spike timings of the neurons represent the stimulus. Learning is based on the sole principle of maximization of representation accuracy: the stimulus can be decoded from the spike timings with minimum error. Since the encoding is causal, we propose two different representation strategies: The spike timings represent the stimulus either in a predictive manner or by reconstructing past input. We apply the model to speech data and discuss differences between the emergent representations.

1 Introduction

How are sensory stimuli represented in the neural system? How is the neural system adapted to the structure in natural stimuli? A theoretical approach to these questions for the early stages of the neural system consists in modeling neural representation by a data representation loop: A neural encoding system transforms the stimuli into neural activity, and a hypothetical decoder indicates how to "read" the input stimuli from the activity.

Previous work has often used rather abstract models for the neural encoding. Linear transforms [1], or transforms that issue from statistical estimation theory [2], or mathematically efficient algorithms for function decomposition such as matching pursuit were used [3, 4]. The transforms were adapted to the stimuli space in order to maximize representation accuracy alone [3, 4], or, additionally, sparseness of the neural response [2] or its temporal coherence [1].

Recently, we have proposed a data representation method where the encoding transform is given by a standard spiking neuron model, and decoding is based on the spike timings alone [5]. This single neuron data representation method was applied to artificial stimuli.

In this paper, we propose data representation by means of a *population* of spiking neurons. The learning principle is representation accuracy: the encoding of the stimulus is such that a hypothetical homunculus can accurately decode the stimulus from the spike timings (see e.g [6] for the concept of the homunculus). Further, we distinguish between decoding as *reconstruction* of the input, and decoding as *prediction*, see Figure 1.

*This work was funded by the Academy of Finland (NEURO program and the Algodan Centre of Excellence)

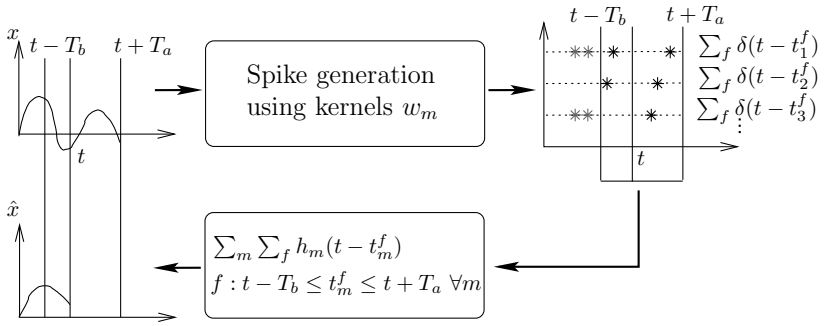


Fig. 1: Proposed data representation method in this paper: The input $\mathbf{x}(t)$ is represented by the spike timings t_m^f of a population of spiking neurons. Spike generation for each neuron m is based on the neuron model SRM_0 [7]. Because of the explicit presence of time and the causality of the spike generation process, we can distinguish between two different decoding strategies: (a) reconstruction of the input at time t from spikes which come later (setting $T_b = 0$ and $T_a = T_d$, where T_d is the reconstruction delay) or (b) prediction of the input at time t from spikes which occur beforehand (setting $T_a = 0$ and $T_b = T_p$, where T_p is the prediction horizon). For both cases, we derive here learning rules for the encoding filters $\mathbf{w}_m$ and the decoding filters $\mathbf{h}_m$. Note: to simplify the presentation, the figure has been drawn for scalar data as e.g. speech.

2 Model for the encoder and decoder

For sake of generality, we present the theory for vector-valued input $\mathbf{x}(t)$ so that the spike timings of the neuron population may represent multiple input channels.

The input, which is time dependent, is encoded into spike timings t_m^f based on the SRM_0 neuron model [7] : The equation for the membrane voltage $u_m(t)$ of neuron m is

$$u_m(t) = \underbrace{\eta_0 \exp\left[-\frac{t-\hat{t}_m}{\tau}\right]}_{u_m^r(t)} + \underbrace{\int_0^{\min\{t,T_w\}} \mathbf{w}_m(s)^T \mathbf{x}(t-s)\mathrm{d}s}_{u_m^i(t)} + u_m^n(t), \qquad (1)$$

where $\hat{t}_m$ is the last spike timing of neuron m before time t, and $\mathbf{w}_m(s)$ the unknown causal encoding filter, to be learned, of length T_w. The remaining constants are the refractory time constant τ of the suppressive refractory voltage $u_m^r(t)$ and the reset amount $\eta_0 < 0$. Convolution and scalar product of the input $\mathbf{x}(t)$ with the encoding filter $\mathbf{w}_m(s)$ defines the voltage $u_m^i(t)$. The term $u_m^n(t)$ models voltage that is not related to the input. Spike timings $\{t_m^f; f = 1, \ldots\}$ are defined by $u_m(t_m^f) = \theta$, where $\theta > 0$ is a fixed threshold. After a spike, the refractory voltage $u_m^r(t)$ leads to a reset and suppression of the voltage.

The approximation $\hat{\mathbf{x}}(t)$ is obtained from the spike timings of a neuron pop-

ESANN'2009 proceedings, European Symposium on Artificial Neural Networks - Advances in Computational Intelligence and Learning. Bruges (Belgium), 22-24 April 2009, d-side publi., ISBN 2-930307-09-9.

ulation of size M as sum of partial approximations $\hat{\mathbf{x}}_m(t)$,

$$\hat{\mathbf{x}}(t) = \sum_{m=1}^{M} \hat{\mathbf{x}}_m(t) \qquad \hat{\mathbf{x}}_m(t) = \sum_{f: t-T_b \leq t_m^f \leq t+T_a} \mathbf{h}_m(t - t_m^f), \tag{2}$$

For the reconstruction-based representation, we set $T_a = T_d$ and $T_b = 0$, which means that a data point is reconstructed from spikes that come later. This introduces a delay T_d in the approximation. For the prediction-based representation, we set $T_a = 0$ and $T_b = T_p$, where T_p is the prediction horizon. Here, a data point is predicted from spikes which occur beforehand. The vector valued decoding filters $\mathbf{h}_m(s)$ are unknown and to be learned. They are acausal and of length T_d for the reconstruction-based representation, while they are causal and of length T_p for the prediction-based representation.

3 Learning rules

Cost functional. We measure the accuracy of the representation via

$$J\left(\mathbf{h}_1(s), \ldots, \mathbf{h}_M(s), \mathbf{w}_1(s), \ldots, \mathbf{w}_M(s)\right) = \frac{1}{2T} \int_0^T ||\hat{\mathbf{x}}(t) - \mathbf{x}(t)||^2 \mathrm{d}t, \tag{3}$$

where the approximation $\hat{\mathbf{x}}(t)$ was defined in Equation (2). Iterative minimization of J provides a learning rule for the vector valued encoding filters $\mathbf{w}_m(s)$ and decoding filters $\mathbf{h}_m(s)$. We work with a stochastic gradient descent algorithm so that finding the functional derivatives $\delta J/\delta \mathbf{w}_m(s)$ and $\delta J/\delta \mathbf{h}_m(s)$ leads to the learning rules.

Learning encoding filters $\mathbf{w}_m$. For $\delta J/\delta \mathbf{w}_m(s)$, we note that there is no coupling among the membrane voltages $u_m(t)$ in Equation (1). Hence, the spike timings t_m^f do not depend on other spike timings t_i^f $(i \neq m)$. That is why the functional derivative $\delta J/\delta \mathbf{w}_m(s)$ can be calculated as in the single neuron case. Generalizing the results from [5] to the vector-valued case, we have

$$\frac{\delta J}{\delta \mathbf{w}_m(s)} = -\frac{1}{T} \sum_f \bar{e}_m(t_m^f) \mathbf{y}_m(s, f), \tag{4}$$

where

$$\bar{e}_m(t_m^f) = \int_{t_m^f - T_a}^{t_m^f + T_b} \mathbf{e}(t)^T \dot{\mathbf{h}}_m(t - t_m^f) \mathrm{d}t \tag{5}$$

for $\mathbf{e}(t) = \hat{\mathbf{x}}(t) - \mathbf{x}(t)$. The term $\mathbf{y}_m(s, f)$ is calculated via

$$\mathbf{y}_m(s, f) = \frac{-\mathbf{x}(t_m^f - s)}{\dot{u}_m(t_m^f)} + \frac{-\eta_0}{\tau \dot{u}_m(t_m^f)} \exp\left[-\frac{t_m^f - t_m^{f-1}}{\tau}\right] \mathbf{y}_m(s, f-1). \tag{6}$$

The initial values in this recursion are $\mathbf{y}_m(s, 0) = \mathbf{0}$. Using the stochastic gradient for Equation (4), we obtain the following online rule: If neuron m emits at t_m^f its f-th spike, update $\mathbf{w}_m(s)$ by

$$\mathbf{w}_m(s) \leftarrow \mathbf{w}_m(s) + \mu_w \bar{e}_m(t_m^f) \mathbf{y}_m(s, f), \tag{7}$$

ESANN'2009 proceedings, European Symposium on Artificial Neural Networks - Advances in Computational Intelligence and Learning. Bruges (Belgium), 22-24 April 2009, d-side publi., ISBN 2-930307-09-9.

where μ_w is the step size.

Learning decoding filters $\mathbf{h}_m$. Straightforward calculation of $\delta J/\delta \mathbf{h}_m(s)$ leads to

$$\frac{\delta J}{\delta \mathbf{h}_m(s)} = \frac{1}{T} \sum_f 1_{[-T_a,T_b]}(s) 1_{[-t_m^f, T-t_m^f]}(s) \mathbf{e}(s + t_m^f), \tag{8}$$

where $1_{[-T_a,T_b]}(s)$ is the indicator function that is one if the argument s is within the interval $[-T_a, T_b]$, and zero else. Using the stochastic gradient, the following least mean square like learning rule is obtained

$$\mathbf{h}_m(s) \leftarrow \mathbf{h}_m(s) - \mu_h \mathbf{e}(s + t_m^f) 1_{[-T_a,T_b]}(s) 1_{[-t_m^f, T-t_m^f]}(s) \tag{9}$$

The step size is given by μ_h, and the update takes place after each spike t_m^f.

4 Simulations with speech data

We learned a reconstruction-based and a prediction-based representation of speech[1] for a population of $M = 15$ neurons. The input $\mathbf{x}(t)$ is here one-dimensional, and denoted by $x(t)$. The accuracy of the representation is measured by the Signal to Noise Ratio (SNR $= 10 \log_{10} ||x||^2/||e||^2$.)

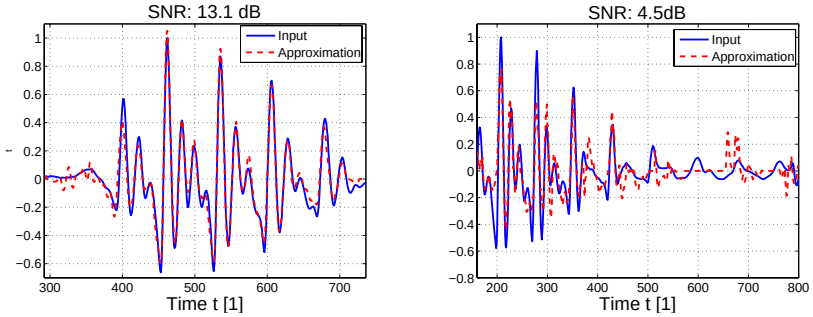
In the reconstruction-based representation scheme, speech segments are represented with an average accuracy of 13.2dB (std 2.17dB) by the population of spiking neurons. In the prediction-based representation scheme, the speech segments are represented with an average accuracy of 4.5dB (std 1.2dB). Figure 2 shows selected examples to illustrate the representation performance for the two cases. It can be seen that speech segments that have small values over a long time interval are hard to represent. Further, the examples show that important parts of the stimuli can be predicted from the spike timings.

Not all the neurons contribute equally to the representation of the stimuli. For each neuron, one can calculate how much the total error increases when the neuron is omitted from the representation. The neurons can be ordered according to this increase of the error.

In Figures 3a to 3c, we show the encoding and decoding filters of the three most contributing neurons for the reconstruction-based representation scheme: For each neuron m, we see that the decoding filter $h_m(s)$ and the time inverted encoding filter $w_m(-s)$ are similarly shaped. This means that the filtering process is here much like matching feature templates to the input, and the firing event of each neuron encodes the presence of the feature in the input.

In Figures 3d to 3f, we show the encoding and decoding filters of the three most contributing neurons for the prediction-based representation scheme. Here, the encoding and decoding filters do not show a similar shape. However, each decoding filter $h_m(s)$ seems to be a continuation of the corresponding $w_m(-s)$. The encoding filters serve here to compute an appropriate guess about the future input, and the guess is expressed by the decoding filters. Neuron 2 in Figure 3d

[1]Data is freely available for download at http://festvox.org/dbs/dbs_kdt.html.



(a) Reconstruction-based representation (b) Prediction-based representation

Fig. 2: Examples for average approximation performance.

detects a fast growth of the input stimulus (much like taking a derivative), and its firing represents further growth of the future input with a subsequent decay to the baseline. Neuron 15 in Figure 3f works similarly, but it is additionally tuned to oscillations in the input. Neuron 3 in Figure 3e detects the end of bumps in the stimulus, and represents the guess that the future input goes down before returning back to positive values.

5 Conclusions

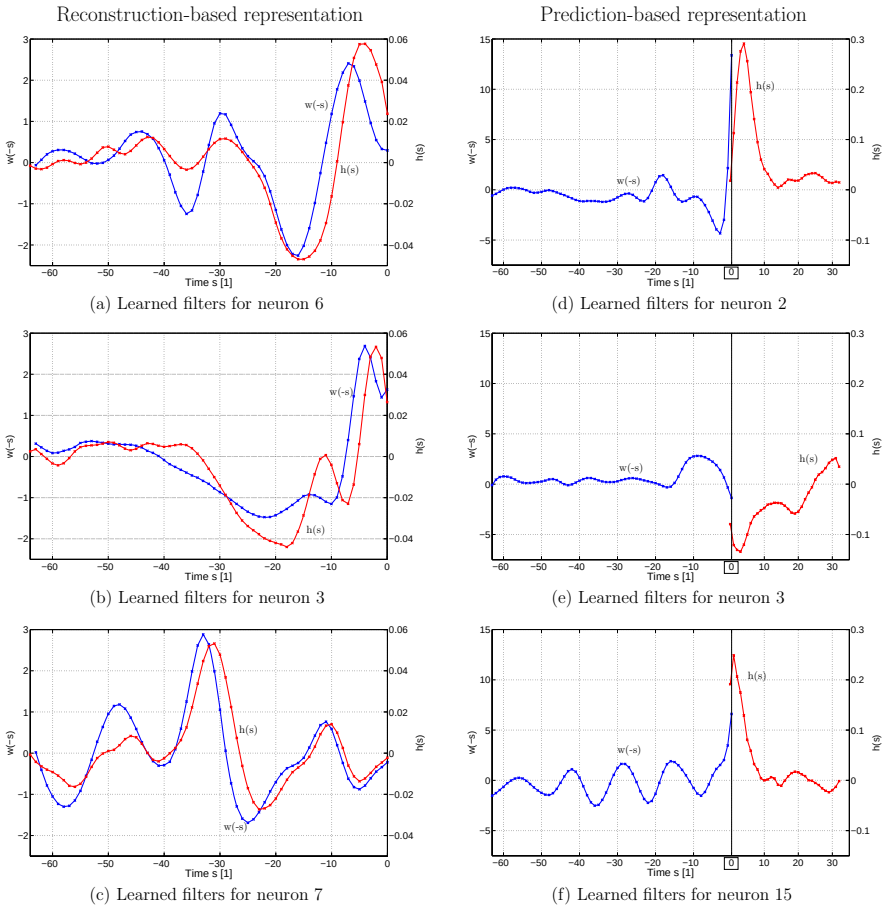
We proposed here a model for the representation of time dependent data with a population of spiking neurons. Specifically, we distinguished between reconstruction and prediction-based representations. We derived learning rules for both representation strategies and applied them to natural speech data. The two representation strategies result in encoder-decoder pairs with distinct properties: For the reconstruction-based representation, the encoder-decoder pairs decompose the input into feature templates while for the prediction-based representation, the emergent encoder-decoder pairs compute a guess about the future input.

Previous attempts to learn spike timings-based representations include [3, 4]. A major difference to their method is the encoding because in our method, encoding is based on neurons that are modeled with a standard spiking neuron model. We note further that because of the causality of the encoding process, we were able to make the difference between reconstruction-based and prediction-based representation strategies, which cannot be done with the approach of [3, 4].

References

[1] J. Hurri and A. Hyvärinen. Simple-cell-like receptive fields maximize temporal coherence in natural video. *Neural Computation*, 15(3):663–691, 2003.

[2] B. Olshausen and D. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.



(a) Learned filters for neuron 6

(d) Learned filters for neuron 2

(b) Learned filters for neuron 3

(e) Learned filters for neuron 3

(c) Learned filters for neuron 7

(f) Learned filters for neuron 15

Fig. 3: (a)-(c): Learned filters of the three most contributing neurons for the reconstruction-based representation. (d)-(f): Likewise for the prediction-based representation. Time inverted encoding filters $w_m(-s)$ are shown in blue, the decoding filters $h_m(s)$ in red. See the text body for a discussion.

[3] L. Perrinet. Finding independent components using spikes: A natural result of hebbian learning in a sparse spike coding scheme. *Natural Computing*, 3(2):159–175, 2004.

[4] E. Smith and M. Lewicki. Efficient auditory coding. *Nature*, 439(7079):978–982, 2006.

[5] M. Gutmann, A. Hyvärinen, and K. Aihara. Learning encoding and decoding filters for data representation with a spiking neuron. In *International Joint Conference on Neural Networks (IJCNN)*, 2008.

[6] F. Rieke, D. Warland, R. de R. van Steveninck, and W. Bialek. *Spikes: Exploring the neural code*. MIT Press, 1997.

[7] W. Gerstner and W. Kistler. *Spiking Neuron Models*. Cambridge University Press, 2002.

Learning reconstruction and prediction of natural stimuli by a population of spiking neurons – Supplementary methods –

Michael Gutmann and Aapo Hyvärinen

Main article in Proc. ESANN2009

Preprocessing and settings for the encoding, decoding, and the learning We used speech samples from the Festvox KED timit database. [1] The speech samples are downsampled from 16kHz to 8kHz using MATLAB's decimate function. We extracted then randomly speech segments of length $T = 800$ time units (100ms), where we excluded segments of silence. Additionally, the segments were multiplied with half a period of a shifted raised cosine window (period: 160 time units) so that the segments start from zero. The segments were then smoothed with a Gaussian kernel (variance: 4 time units) and normalized to maximal value 1.

For learning, we used 7000 speech segments $x(t)$. The step sizes were $\mu_w = 0.01$ and $\mu_h = 0.001$. Delay T_d was 64 time units, T_p valued 32 time units. The number of neurons was $M = 15$. The encoding filters w_m had length $T_w = 64$ time units. For initialization of the learning rules, the initial values of the discretized w_m and h_m were drawn from a Gaussian distribution with standard deviation 0.01 and mean 0. These initial values of the encoding filters w_m are too small to trigger spikes. To have spikes even for $w_m = 0$, we set $u_m^n(t)$ to be a ramp voltage with a randomly chosen slope: For each neuron m the slope was drawn from an uniform distribution on $[\theta/240, \theta/480]$. The threshold θ was set to 4, $\eta_0 = -8$, and τ was 10 time units.

[1] Free download at http://festvox.org/dbs/dbs_kdt.html