

Emergence of modularity within one sheet of intrinsically active stochastic neurons

C. Weber

K. Obermayer

Department of Computer Science, FR2-1, Technische Universität Berlin
Franklinstr. 28/29, D-10587 Berlin, Germany. Email: cweber@cs.tu-berlin.de

Abstract

We investigate how modular structure within a neural net of stochastic neurons can emerge from intrinsic dynamics and from unsupervised learning of data. The model consists of an input layer and two hierarchically organized hidden layers and differs from the Helmholtz machine only in that hidden neurons are not assigned to a single one of the layers prior to learning. On the basis of a maximum likelihood framework the task is (i) to infer a hidden code from data using the recognition weights and (ii) to generate given input data from the hidden code using the generative weights. Hidden neurons are fully connected via both weight types allowing to code on different hierarchical levels. The hidden neurons are separated into two groups by their intrinsic parameters. One group, designed to be on the highest hierarchical level, is highly spontaneously active and is thus responsible for the initiation of the hidden code. The other group, designed to be on the lower level, is highly responsive to stimulation by input only and thus likely transmits information between the input and the highest level in both directions. Besides demonstrating the emergence of this hierarchical structure, we show using a different data set that a parallel structure emerges which matches the data. Finally, if the net is trained without any input, then a weak hierarchical structure emerges by the differential intrinsic activity.

1 Introduction

Let us imagine an animal to evolve a sensory organ by a beneficial mutation. If a corresponding mutation would have to take place simultaneously within the cortex, then this mutation would very unlikely be successful. The cortex should instead always make use of improved input. It should act as a uniform and universal information processing tissue.

This two-dimensional sheet of neuronal tissue hosts many functionally distinct areas (e.g. 65 areas in cat [1]) which process information through pathways

which are organized in parallel as well as hierarchically [2]. The earliest manifestations of areas during corticogenesis are regionally restricted molecular patterns (“neurochemical fingerprints” [3]) which appear before the formation of thalamocortical connections [4]. However, these molecules do not ultimately determine the area a neuron will belong to: e.g. in congenitally blind human subjects, auditory signals activate association areas in the occipital cortex [5], areas which in normal subjects are involved in visual location and motion detection experiments.

Since there are ten times more area-to-area connections than areas, the description of cortico-cortical connectivity is more complex and thus unlikely to be fully described genetically. Abstract geometrical models suggest that topological neighborhood plays an important but not exclusive role in determining these connections [1]. We propose here that learning also accounts for these large-scale connections, which is an attractive idea in particular to explain the intense adaptations to disturbances [5].

Recently we have presented a computational model in which neuronal activity plays the major role in learning rules for the connections between areas [6]. Only the areas were genetically defined: already before development the intrinsic functional properties of their neurons differed between different areas. More precisely, hidden neurons were divided into two groups which differed by one parameter of their transfer function. This makes one group respond with stronger activity to a given input than the other group. The first, more active group then learns to process stimulus features which occur more frequently.

Not only the input space can thereby be divided into two groups. Using lateral weights among the hidden neurons, the former model allows hidden neurons also to code on two hierarchical levels. Using hierarchically generated data, some of the hidden neurons from the more active group establish a second hierarchical level by grouping neurons from the other hid-

den units via their lateral weights while their weights to the input neurons decline.

The former model consists of deterministic neurons which are activated by their input through a continuous transfer function. As a drawback for biological plausibility, the hidden neurons have to take track of two activation values continuously, in order to determine their role in information processing on the two possible hierarchical levels.

Here, we use a stochastic model, a Helmholtz machine [7], where neurons are able to become active spontaneously. Activations on different hierarchical levels are instantiated at consecutive time steps, reminiscent of a synfire chain model [8]. However, two consecutive layers are concatenated to one layer.

With this model, all results of the former, deterministic model are reproduced. In addition, it is shown that through spontaneous activity even without data, internal structure within the network can emerge.

2 Theory

The original Helmholtz machine [7] has several hierarchically organized layers. Each receives bottom-up input via recognition weights from the next lower layer and top-down input via generative weights from the next higher layer. Originally, there are no lateral weights within one layer.

The task of each weight set is to “invert” the weights going into the other direction: the generative weights are trained to model as good as possible the data given their hidden representation which has been obtained via the recognition weights. Vice versa, the recognition weights are trained to generate as good as possible a hidden activation pattern given the representation of this on the input units which has been obtained via the generative weights.

An efficient method to train a Helmholtz machine is the wake-sleep algorithm [9] which we use to train our model. It has two phases:

In the *wake phase*, activation originating from data is propagated upwards the hierarchy (using the recognition weights) and then, originating from the hereby inferred hidden code propagated downwards again (using the generative weights). The reconstruction error is used to train the generative weights.

The *sleep phase* is similar but reversed: activation originates from a stochastically generated hidden code at the hierarchically highest layer. It is propagated (using the generative weights) to the input and back (using the recognition weights) to the highest level again. The error of the hidden code reconstruction is used to train the recognition weights.

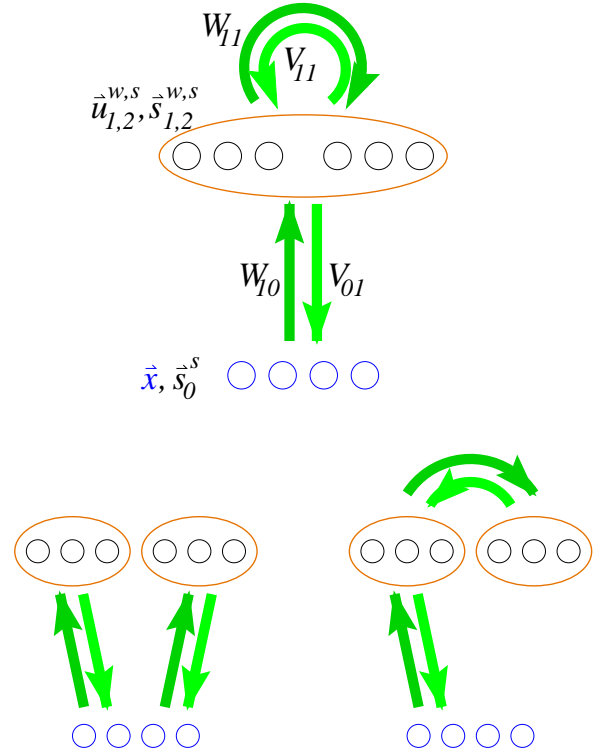


Figure 1: **Top:** general architecture of our model which can cover as special cases a parallel organization (**bottom left**) or a hierarchical organization of the hidden units (**bottom right**). The lateral weights W_{11} and V_{11} of our model allow each hidden neuron to take part in a representation \vec{u}_1 on a lower and \vec{u}_2 on a higher hierarchical level. Dependent on the structure of the data, training will result in one of the two architectures shown at the bottom. In each of them the activations \vec{x} on the input units are represented by hidden unit activations \vec{u} . W are recognition weights, V are generative weights, indexed with the number of the layer of termination and origin.

Concatenation of hierarchical levels

Our model is a Helmholtz machine with two hidden layers. However, from the architecture or the algorithm only, the two hidden layers cannot be distinguished. The two layers are concatenated to one hidden layer (Fig. 1, top). Lateral weights, both recognition W_{11} and generative V_{11} , as well as possibly missing (zero-value) weights to the input allow some neurons to represent the code of other hidden neurons and thus to belong logically to a second hierarchical level.

Recognition of data involves two consecutive steps which distinguish the two hidden layers: activations \vec{u}_1 evoked from data \vec{x} via input weights W_{10} are assigned

to the first (lower) layer. Activations \vec{u}_2 evoked from \vec{u}_1 via lateral weights W_{11} are assigned to the second (higher) layer.

Data generation also distinguishes two hidden layers: a hidden code \vec{s}_2 on the logically higher hidden layer evokes via lateral, generative weights V_{11} a hidden code \vec{s}_1 on the logically lower layer. Reconstructed data \vec{s}_0 is then generated via V_{01} using \vec{s}_1 .

Note that a neuron can be active at both times consecutively. In order to discourage this during recognition, an activity-dependent weight constraint introduces competition between all incoming weights of a hidden neuron. This encourages a hidden neuron to receive input from the input neurons via W_{10} or from other lateral neurons via W_{11} but not both.

Distinguishing the modules

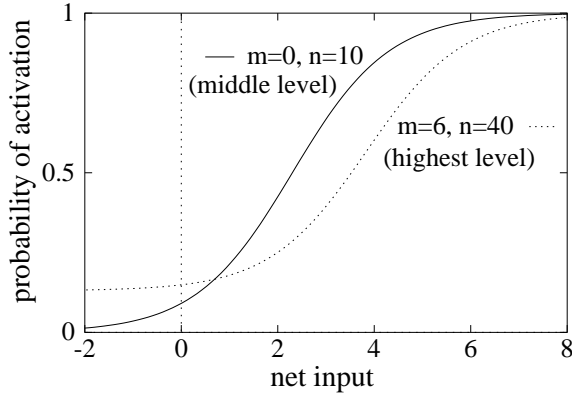


Figure 2: The binary stochastic transfer functions $\vec{f}_b^{w,s}$ with two different sets of parameters n and m which are used in the two subgroups of the hidden neurons. Given an input (x-axis) they represent the probability (y-axis) that a neuron is “ON”. This is also the mean output value used in $\vec{f}_m^{w,s}$. The dotted line shows the transfer function of neurons which are designed for the hierarchically higher level. There is a prominent resting activity ($m = 6$) but a weak gain ($n = 40$, i.e. high sparsity of firing). The function displayed as a solid line is intended for middle layer neurons. There is no resting activity ($m = 0$) but a stronger gain ($n = 10$). In the wake phase, these latter values were used for the function \vec{f}_m^{w} for all neurons.

The idea to distinguish the hidden neurons to form two modules takes advantage of data generation in the sleep phase: higher-level neurons do not receive any input, but lower-level neurons receive input from the spontaneous activity of the higher-level neurons.

Thus, a neuron which tends to be spontaneously active is well suited for the hierarchically highest level. In contrast, a neuron which becomes strongly active from input only is well suited to represent the lower level. Two stochastic transfer functions which display these differential behaviors are depicted in Fig. 2. Underlying are three states a hidden neuron can choose from stochastically: an “ON”-state evoked by its input h_i , an “ON”-state evoked spontaneously with degeneracy m and an “OFF”-state with degeneracy n .

It was shown that degenerate “OFF”-states introduce sparse coding which is used to let feature detectors emerge in a stochastic net [10]. On the other hand, spontaneous (resting) activity was shown to be necessary in attractor nets [11] as well as *in vitro* [12] to maintain robust low frequency firing. Our model functions accommodate both features, “OFF”-states with degeneracy n and spontaneous “ON”-states with degeneracy m . The probability that a hidden neuron i is switched “ON” by its input h_i is:

$$P(u_i = 1) = \frac{e^{h_i} + m}{e^{h_i} + m + n} =: f_m(h_i)$$

The corresponding stochastic transfer function which assigns the two binary values is denoted by f_b . The corresponding mean-field transfer function is f_m .

3 Methods

The detailed processing steps of the wake-sleep algorithm are as follows. Wake phase, inferring the hidden code from a data point \vec{x} :

$$\begin{aligned} \vec{u}_1^w &= \vec{f}_m^w(W_{10}\vec{x}) \\ \vec{u}_2^w &= \vec{f}_m^w(W_{11}\vec{u}_1^w) \end{aligned} \quad (1)$$

Reconstruction of the input:

$$\begin{aligned} \vec{s}_1^w &= V_{11}\vec{u}_2^w \\ \vec{s}_0^w &= V_{01}\vec{u}_1^w \end{aligned} \quad (2)$$

Update of the generative weights:

$$\begin{aligned} \Delta V_{11} &= \varepsilon_{11} (\vec{u}_1^w - \vec{s}_1^w) \cdot (\vec{u}_2^w)^T \\ \Delta V_{01} &= \varepsilon_{01} (\vec{x} - \vec{s}_0^w) \cdot (\vec{u}_1^w)^T \end{aligned} \quad (3)$$

Sleep phase, initiation of the hidden code at the hierarchically highest level:

$$\vec{s}_2^s = \vec{f}_b^s(0) \quad (4)$$

Generation of an input code:

$$\begin{aligned} \vec{s}_1^s &= \vec{f}_b^s(V_{11}\vec{s}_2^s) \\ \vec{s}_0^s &= V_{01}\vec{s}_1^s \end{aligned} \quad (5)$$

Reconstruction of the hidden code:

$$\begin{aligned}\vec{u}_1^s &= \vec{f}_m^s(W_{10}\vec{s}_0^s) \\ \vec{u}_2^s &= \vec{f}_m^s(W_{11}\vec{s}_1^s)\end{aligned}\quad (6)$$

Update of the recognition weights:

$$\begin{aligned}\Delta W_{10} &= \varepsilon_{10}(\vec{s}_1^s - \vec{u}_1^s) \cdot (\vec{s}_0^s)^T \\ \Delta W_{11} &= \varepsilon_{11}(\vec{s}_2^s - \vec{u}_2^s) \cdot (\vec{s}_1^s)^T\end{aligned}\quad (7)$$

The wake-sleep algorithm is not a gradient descent in an energy space and easily gets stuck in local minima. The following weight constraints were applied to improve the solutions found: (i) generative weights V_{01} and V_{11} as well as lateral recognition weights W_{11} were rectified, i.e. negative weights were set to zero. (ii) a soft weight constraint was applied to the recognition weights in order to preserve sparse coding and to enforce competition between coding on the two hierarchical levels. It adds to Eq. 7 and treats positive and negative weights separately. With the use of the Heaviside function $\Theta(x) = 1$, if $x > 0$, otherwise 0, it can be written as:

$$\Delta w_{ij}^{decay} = \lambda^w h_i \Theta(w_{ij}) w_{ij} \sum_{j'} \Theta(w_{ij'}) w_{ij'}^2, \quad (8)$$

where $\vec{h} = \vec{u}_1^w + \vec{u}_2^w + \vec{u}_1^s + \vec{u}_2^s$ is the sum of all activations which have been induced by the recognition weights. The indices j, j' extend over all input and hidden units.

Generation of training data

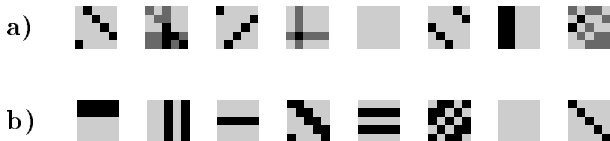


Figure 3: Examples of stimuli \vec{x} used for training. **a)** Lines generated independently of each other, 0° and 45° lines with probability 0.05 and 90° and 135° lines with probability 0.1. **b)** Lines generated by a hierarchical model: lines of one randomly chosen orientation only are generated with probability 0.44.

Artificial data are generated by two different paradigms. First, in a non-hierarchical manner, data consist of discrete, sparsely generated elements. These elements are lines of 4 different orientations on a 5×5 grid of input units resulting in a total number of 20 different elements. For the first experiment each line is chosen with a fixed probability independently of any

other. Thus there is no structure among the code elements. For the purpose of structuring our network into two distinct groups of hidden neurons we form two groups of the elements. One group, horizontal and 45° lines ($-$, $/$), measured in polar coordinates, each are generated with probability 0.05 (thus, $2 \cdot 5 \cdot 0.05 = 0.5$ lines of this group are present on average). The other group, vertical and 135° lines ($|$, \backslash), are generated twice as often, with probability 0.1 each (1 line on average). Fig. 3 a) shows example data.

For the second experiment data are generated in a hierarchical manner [9]. First, one of 4 orientations is chosen at random, representing a decision process on a higher hierarchical level. Then, on the lower level, lines with this orientation are generated with probability 0.44 each. Fig. 3 b) shows example data.

Training

Weights were initialized with small positive random values. On-line learning was performed with $5 \cdot 10^6$ randomly generated data points. For each data point, Eqs. (1) to (8) were computed, where the sleep phase was repeated 8 times in order to obtain an average over the stochastic functions, Eqs. (4,5).

One third of the hidden neurons (upper third in Figs. 4,5,6) were designed to code on the highest hierarchical level (see the caption of Fig. 2 for description), the other two thirds for the lower level. Other parameters were: learning step sizes $\varepsilon_{01} = \varepsilon_{10} = 0.01$, $\varepsilon_{11} = 0.001$, both were decreased linearly to zero in the second half of training. Weight decay parameter $\lambda^w = 0.01$.

4 Results

Areas organize in parallel

After training on parallelly organized data we find that neurons in the upper third (Fig. 4) predominantly represent the more frequent stimuli (lines of 90° and 135°) while neurons in the lower two thirds represent the less frequent stimuli (lines of 0° and 45°). Fig. 4 shows an exemplary result of the trained recognition weights W_{10} , and also lateral recognition weights W_{11} which have been included for consistency with the next experiment and which have not learned a meaningful structure here.

This result is surprising, because we would expect the neurons in the upper third to code for the less frequent stimuli, because they are less susceptible to input. Instead, it seems to be substantial that they are more active spontaneously and when input is low (Fig. 2, $m = 6$, $n = 40$). This is the case in the initial phase of learning when the weights have not

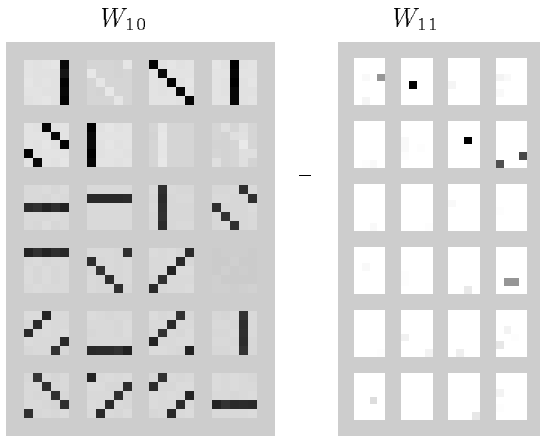


Figure 4: The recognition weight matrix W_{10} (**left**) and the lateral recognition weight matrix W_{11} (**right**) after training on parallelly organized data. Each square of the weight matrices shows the receptive field of one of the 4×6 hidden neurons; **left**, weights from the 5×5 input neurons and **right**, lateral weights. Negative weights are brighter than the background (frame), positive weights are darker (**left**). For lateral weights (**right**), zero weights are white (there are no negative weights).

Areas have organized parallelly: weights W_{10} to the inputs code for 90° and 135° lines in the upper third and predominantly for 0° and 45° lines in the lower two thirds.

organized. The neurons which are more active will now learn faster the most frequent stimuli.

We repeated this simulation ten times with different initial random values which changes the initial weights as well as the randomly generated stimuli. Table 1 shows the scores. Since there are more frequent lines than there are neurons in the upper third, some neurons in the lower two thirds, too, code for frequent lines. Altogether, there are four more neurons than stimulus types and so some receptive fields remain empty.

	$0^\circ, 45^\circ$	$90^\circ, 135^\circ$	none
upper third	10%	70%	20%
lower two thirds	56%	28%	16%

Table 1: The neurons from the two regions and the stimulus classes which they choose to code for in the parallel setting. The percentage states how often a neuron selects a class. Numbers are averaged over ten runs and over all neurons within each region.

Areas organize hierarchically

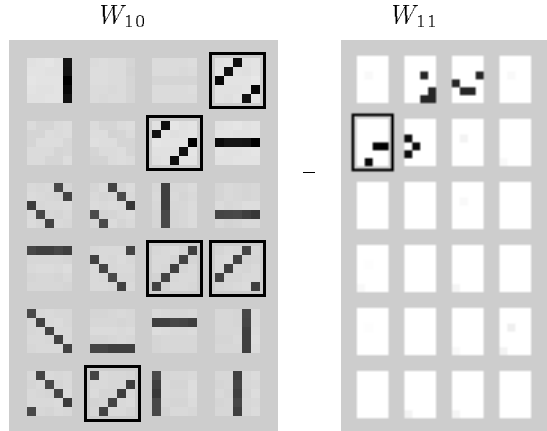


Figure 5: Same as Fig. 4, but trained on hierarchically organized data (same initialization of weights). Areas have organized hierarchically: neurons in the lower two thirds code for the input via W_{10} while four neurons in the upper third each integrate via W_{11} units from the lower two thirds which code stimuli of one direction. Neurons which code for lines of 45° orientation are marked by a frame.

The net shown in Fig. 5 was trained on the hierarchical data set and has self-organized accordingly. Neurons in the upper third are more active spontaneously, four of them have vanishing connections W_{10} to the input. Instead, they code on the higher level by integrating via lateral weights W_{11} neurons from the lower level which code for the same orientation.

These higher-level neurons were never observed to have weights to other neurons in the upper third even if these had matching orientation. This would not be a flaw if only four neurons had been given the proper parameters for the higher level. In some experiments, however, only three neurons in the upper third managed to code on the higher level while the others were connected to the input neurons. Then one neuron within the lower two thirds was superfluous and did not code on the higher level.

Areas organize from intrinsic noise

Without data (\vec{x} set to zero), only intrinsic stochastic neuronal dynamics remains for training. As the generative weights V_{01} towards the input become zero, also the recognition weights W_{10} will decay in order to “invert” the generative model. Prevalently low activation values favor neurons in the upper third (Fig. 6) to develop lateral recognition weights W_{11} because these have stronger resting activity. The resulting structure,

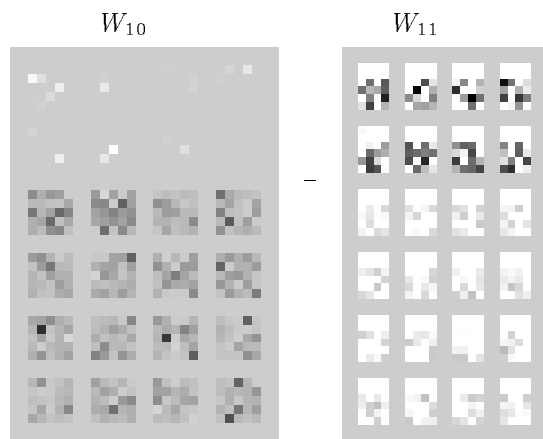


Figure 6: Same as Fig. 4, but trained with zero-valued data. Neurons in the upper third receive input via W_{11} from the units in the lower two thirds. Weights W_{10} to the input neurons are much smaller but still visible in the lower two thirds because of normalization of the brightness scale.

however, is not reflected in the generative weights V_{11} (not shown), because in the wake phase, when these are trained, parameters of both regions do not differ.

5 Discussion

In separate experiments we have perceived a strong influence of developmental constraints with constant parameters n and m across the hidden neurons: in the parallel setting, a hidden area where weights grow faster will look at those input features which occur more often (results not shown). This effect parallels the effect of stronger activity at low input which amplifies only the Hebbian learning term. Likewise, one could initialize weights between certain groups of neurons stronger than others to yield the same effect. This would clearly be a model for the influence of topological neighborhood.

In contrary to models which are based on neighborhood only [1], however, we model activity based and data driven growth and refinement of connections. The results on parallel and hierarchical organization are similar to those obtained by our previous work [6]. They show that simple parameter changes within a neural network can conduct the data flow to structure the network via Hebbian learning to form an adequate representation of the data.

The methods in this work were advanced by the use of stochastic neurons which renders activation dynamics biologically plausible and which also allows for internal structuring without any data.

References

- [1] J.W. Scannell, C. Blakemore, and M.P. Young. Analysis of connectivity in the cat cerebral cortex. *J. Neurosci.*, 15(2):1463–1483, 1995.
- [2] D.J. Felleman and D.C. Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, 1:1–47, 1991.
- [3] S. Geyer, M. Matelli, G. Luppino, A. Schleicher, Y. Jansen, N. Palomero-Gallagher, and K. Zilles. Receptor autoradiographic mapping of the mesial motor and premotor cortex of the macaque monkey. *J. Comp. Neurol.*, 397:231–250, 1998.
- [4] M. J. Donoghue and P. Rakic. Molecular evidence for the early specification of presumptive functional domains in the embryonic primate cerebral cortex. *J. Neurosci.*, 19(14):5967–79, 1999.
- [5] R. Weeks, B. Horwitz, A. Aziz-Sultan, B. Tian, C.M. Wessinger, L.G. Cohen, M. Hallett, and J.P. Rauschecker. A positron emission tomographic study of auditory localization in the congenitally blind. *J. Neurosci.*, 20(7):2664–2672, 2000.
- [6] C. Weber and K. Obermayer. Structured models from structured data: emergence of modular information processing within one sheet of neurons. In *Proceedings IJCNN*, 2000.
- [7] P. Dayan, G. E. Hinton, R. Neal, and R. S. Zemel. The Helmholtz machine. *Neur. Comp.*, 7:1022–1037, 1995.
- [8] M. Abeles. *Corticonics. Neural Circuits of the Cerebral Cortex*. Cambridge University Press, 1991.
- [9] G. E. Hinton, P. Dayan, B. J. Frey, and R. Neal. The wake-sleep algorithm for unsupervised neural networks. *Science*, 268:1158–1161, 1995.
- [10] C. Weber and K. Obermayer. Orientation selective cells emerge in a sparsely coding Boltzmann machine. In C. von der Malsburg, W. von Seelen, J. C. Vorbruggen, and B. Sendhoff, editors, *Proceedings ICANN*, pages 286–291, 1999.
- [11] P.E. Latham, B.J. Richmond, P.G. Nelson, and S. Nirenberg. Intrinsic dynamics in neuronal networks. I. Theory. *J. Neurophysiol.*, 83:808–827, 2000.
- [12] P.E. Latham, B.J. Richmond, S. Nirenberg, and P.G. Nelson. Intrinsic dynamics in neuronal networks. II. Experiment. *J. Neurophysiol.*, 83:828–835, 2000.