Constructing modular architectures with Boltzmann Machines

Hilbert J. Kappen [1]

Department of Medical Physics and Biophysics, University of Nijmegen, Geert Grooteplein 21, 6525 EZ Nijmegen, The Netherlands

In this paper, I discuss how the Boltzmann Machine framework can be used for modeling hybrid and modular architectures. The advantage is that it provides a firm theoretical basis for the derivation of learning rules in the presence of lateral inhibition, feed-forward and feed-back.

1 Introduction

Boltzmann Machines are stochastic neural networks with symmetric connections. The neurons in the network have a probability of firing, which is proportional to the summed activity that it receives through its synapses with other neurons in the network. The most important property of Boltzmann Machines is that this dynamics leads to an equilibrium distribution, known as the Boltzmann-Gibbs distribution, which is a complex expression in terms of the weights of the network.

The learning rule for the Boltzmann Machine consists of a gradient descent on an error measure. This measure is the distance of the Boltzmann Distribution from a desired distribution on the visible units of the network. Therefore, after each learning step, the equilibrium distribution of the network must be calculated and compared with the desired distribution. This is a very time-consuming procedure, and for this reason, Boltzmann Machines have received little attention.

Recently, it was shown, that by introducing lateral inhibition in the Boltzmann Machine, one can simplify the calculation of the Boltzmann Distribution significantly, from $O(e^n)$ to $O(n^k)$ with n the number of neurons and $k > 0$. Examples of such networks are multi-layered feed-forward networks, radial basis-type networks and clustering algorithms. The basic architecture of these networks is sketched in Fig. 1 Numerical simulations have shown that these learning algorithms have similar time-complexity as learning rules for multi-layered perceptrons and can be accellerated with the same methods (momentum, conjugent gradient) [1, 2]. It is also possible to achieve further speed-up by using the EM algorithm. This accelleration is only possible for some Boltzmann Machines such as for instance the radial basis Boltzmann Machine [3].

An important advantage in studying neural networks in the context of Boltzmann Machines is that it gives a unified framework for different architectures. By applying the lateral inhibition in different ways, very many different types of architectures can be constructed, of which the above are only the simplest. In addition, different networks can be combined as modules in a larger network. The learning rules for such hybrid, modular architectures can be directly derived from one global measure, since the total architecture is still a Boltzmann Machine. This measure summarizes in one number the quality of the learned solution. In addition, this measure guarantees that the learning rules derived from it will converge. This is to be contrasted with other approaches, where one uses the individual learning rules for the networks in the modules, optimizing different criteria, and trains connections between modules, using yet another criterion.

For robust recognition in noisy environments it is generally assumed that feed-back plays an important role. It is assumed that some components of a high dimensional stimulus, which are insufficient for recognition, will excite one or several expectations or hypotheses about the identity of the object in the outside world. These hypotheses will provide information about which other components must be read to force a decision. This procedure has the advantage that not all sensory information has to be processed,

[1]This work was partly sponsored by the Dutch Foundation for Neural Networks (SNN) and the Japanese Real World Computing Program

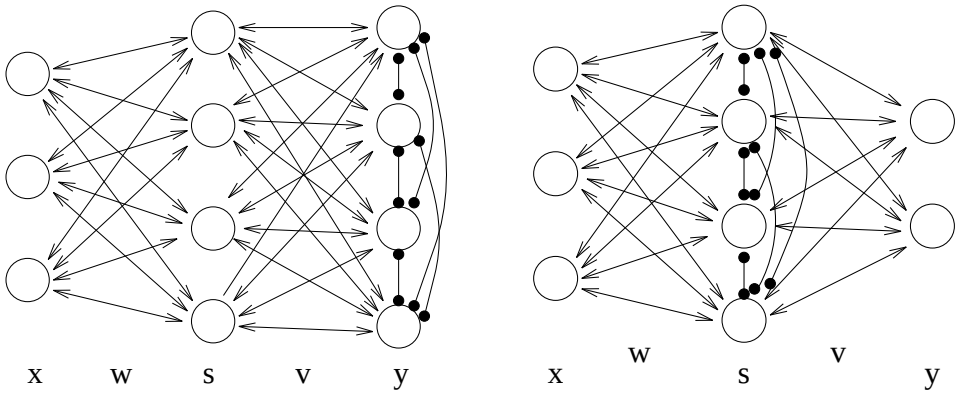

Figure 1: a) The architecture of a Boltzmann Perceptron for feed-forward mapping as proposed by Hopfield. Lateral inhibition in the output layer is an example of how fast learning rules can be obtained for this architecture. b) Radial Basis Boltzmann Machine for probability density estimation. There is no distinction between input and output neurons. The activity of the real-valued neurons are collectively denoted $\vec{x}$ and the activity of the binary-valued neurons are collectively denoted $\vec{y}$. The hidden layer has fully connected lateral inhibition of strength $-J$.

but only the 'relevant' information. Boltzmann Machines can be used to study the interaction between sensory data on the external units and hypotheses in the hidden structure.

The paper is organized as follows. In Section 2, I will introduce the general concept. In Section 3, I will outline how to apply this idea to build modular architectures.

2 Boltzmann Machines with restricted state space

Let a vector $(\vec{x}, \vec{y})$ denote the values of the visible units of the Boltzmann Machine: $\vec{x} = (x_1, \ldots, x_n)$ and $\vec{y} = (y_1, \ldots, y_m)$. The Boltzmann Machine can be used to estimate joint probabilities or conditional probabilities. The latter is also known as feed-forward mappings. For feed-forward mapping, $\vec{x}$ and $\vec{y}$ will denote the values of the input nodes and output nodes, respectively. For joint probability estimation, $\vec{x}$ and $\vec{y}$ have no individual significance and $(\vec{x}, \vec{y})$ is simply a data vector.

Let for each training pattern $(\vec{x}, \vec{y})$, the probability for occurence be $q(\vec{x}, \vec{y})$. Let $\vec{s} = (s_1, \ldots, s_h)$ denote the values of the hidden units of the Boltzmann Machine. Then the total neuron state of the network may be written as $\vec{S} = (\vec{x}, \vec{s}, \vec{y})$ with components $S_I, I = 1, \ldots, n + h + m$. The generic architecture is given in Fig. 2. For given symmetric weights W_{IJ}, the equilibrium distribution of the Boltzmann Machine is given by a Gibbs distribution:

$$p(\vec{S}) = \frac{1}{Z} \exp(\beta \sum_{(IJ)} W_{IJ} S_I S_J) \tag{1}$$

$$Z = \sum_{\vec{S}} \exp(\beta \sum_{(IJ)} W_{IJ} S_I S_J) \tag{2}$$

The observed probabilities on the visible units are:

$$p(\vec{x}, \vec{y}) = \sum_{\vec{s}} p(\vec{x}, \vec{s}, \vec{y})$$

For joint probability estimation (JPE), the Boltzmann Machine minimizes the Kullback divergence

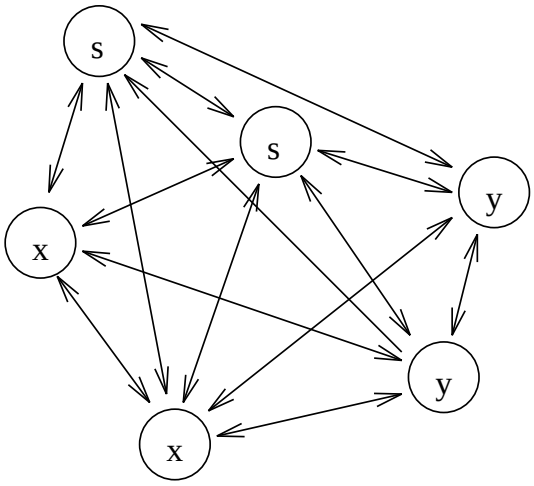

Figure 2: The architecture of a generic Boltzmann Machine with visible units $\vec{x}$ and $\vec{y}$ and hidden units $\vec{s}$. The weights between units are symmetric.

between q and p on the visible units:

$$\begin{aligned} d_{JPE}(p,q) &= \sum_{\vec{x},\vec{y}} q(\vec{x},\vec{y}) \log\left(\frac{q(\vec{x},\vec{y})}{p(\vec{x},\vec{y})}\right). \\ &= \sum_{\vec{x}} q(\vec{x}) \log\left(\frac{q(\vec{x})}{p(\vec{x})}\right) + \sum_{\vec{x},\vec{y}} q(\vec{x},\vec{y}) \log\left(\frac{q(\vec{y}|\vec{x})}{p(\vec{y}|\vec{x})}\right). \end{aligned} \tag{3}$$

The first term attempts to make $p(\vec{x})$ of the Boltzmann Machine equal to the a priori probability $q(\vec{x})$. The second term attempts to make the conditional probability $p(\vec{y}|\vec{x})$ equal to the conditional probability $q(\vec{y}|\vec{x})$, averaged over $\vec{x}$. For feed-forward mappings only the last term is minimized [4]:

$$d_{FF}(p,q) = \sum_{\vec{x},\vec{y}} q(\vec{x},\vec{y}) \log\left(\frac{q(\vec{y}|\vec{x})}{p(\vec{y}|\vec{x})}\right). \tag{4}$$

When a pattern $\vec{x}$ is clamped to the input nodes, the conditional probability $p(\vec{y}|\vec{x})$ should have the desired value $q(\vec{y}|\vec{x})$. The probability to observe $\vec{x}$ is then not modeled by the network.

In both cases the learning rules are given by gradient descent on d [5]. The result for joint probability estimation is:

$$\Delta W_{IJ} = -\eta \frac{\partial d_{JPE}(p,q)}{\partial W_{IJ}} = \eta \left(\sum_{\vec{x},\vec{y}} q(\vec{x},\vec{y}) < S_I S_J >_{\vec{x},\vec{y}} - < S_I S_J > \right) \tag{5}$$

and for feed-forward mapping:

$$\Delta W_{IJ} = -\eta \frac{\partial d_{FF}(p,q)}{\partial W_{IJ}} = \eta \sum_{\vec{x}} q(\vec{x}) \left(\sum_{\vec{y}} q(\vec{y}|\vec{x}) < S_I S_J >_{\vec{x},\vec{y}} - < S_I S_J >_{\vec{x}} \right) \tag{6}$$

where $< S_I S_J >_A$ denotes the two point correlations between neurons S_I and S_J, when the units specified by A are clamped:

$$< S_I S_J >_A = \sum_{\vec{S} \in A} S_I S_J p(\vec{S}) \tag{7}$$

So we see, that for instance for a weight between two hidden units i and j, $< s_i s_j >_{\vec{x},\vec{y}}$, $< s_i s_j >_{\vec{x}}$ and $< s_i s_j >$ contain 2^h, 2^{h+m} and 2^{h+m+n} terms, respectively. Instead of calculating this exponential

number of terms, it is usually faster to *simulate* the network dynamics. Using Glauber dynamics, the network will visit only the most probable states, which make up the largest contributions to the sum in Eq (7). Nevertheless, whether simulated or calculated, the resulting Boltzmann Machine learning rule is extremely slow.

The main idea of this paper is that by introducing lateral inhibition, various specialized and hybrid architectures can be constructed as well as modular architectures. The number of states in A will be reduced from exponential to linear or polynomial in the number of neurons. As a result, fast learning rules can be obtained for these architectures, which makes it possible to study their behaviour on simple sequential machines.

It should be emphasized that the learning rules discussed in this paper are *identical* to the original rule derived by Ackley et al. [5]. The difference is that for the class of architectures discussed, the time-consuming Glauber dynamics can be omitted. The learning rules derived in this paper are well suited for fast prototyping on sequential machines. Larger scale implementations can then be realized in parallel using the original Glauber dynamics.

3 Modular architectures

3.1 Learning rules

In this section we will outline how rules for modular architectures can be derived. From the analysis in [1] we know that the complexity of the learning rules will be proportional to the number of permissible states of the network. Thus, for a modular architecture with M modules and S permissible states per module the complexity of the learning rule is $O(S^M)$.

We will derive learning rules for an arbitrary modular architecture. Let w_{ij}^{α} denote the adaptive connection between neuron i and j in module α. Let $w_{ij}^{\alpha\beta}$ denote the adaptive connection between neuron i in module α and j in module β. Let $s_i^{\alpha} = 0, 1, i = 1, \ldots, H^{\alpha}$ denote the activity of neuron i in module α. The neurons in each module, except the one labeled 'stimulus', are fully interconnected with fixed inhibitory connections such that their total activity $\sum_{i=1}^{H^{\alpha}} s_i^{\alpha} = A^{\alpha}$ is low. Let $\vec{x}$ denote the activity of those neurons whose dynamics is not controlled by the network, but externally. $\vec{x}$ may denote the sensory stimuli or neural representations of the task to be executed. Let $h_i^{\alpha}(\vec{x})$ denote the local field contribution of the external activity to neuron i in module α. The detailed form of $h_i^{\alpha}(\vec{x})$ depends on whether the values of $\vec{x}$ are continuous or discrete, and on whether the learning rules are derived from Eqs. (3) or (4) (see [1]). The Boltzmann distribution for this network is given by

$$p(\vec{s},\vec{x}) = \frac{1}{Z}\exp\{\sum_{\alpha=1}^{M}\sum_{i,j=1}^{H^{\alpha}} s_i^{\alpha}s_j^{\alpha}w_{ij}^{\alpha} + \sum_{\alpha,\beta\neq\alpha}^{M}\sum_{i=1}^{H^{\alpha}}\sum_{j=1}^{H^{\beta}} s_i^{\alpha}s_j^{\beta}w_{ij}^{\alpha\beta} + \sum_{\alpha=1}^{M}\sum_{i=1}^{H^{\alpha}} h_i^{\alpha}(\vec{x})s_i^{\alpha} - J\sum_{\alpha=1}^{M}(\sum_{i=1}^{H^{\alpha}} s_i^{\alpha} - A^{\alpha})^2\}$$

This expression implies that for large J, all states $\vec{s}$ with total activity in module α unequal to A^{α} will have vanishing small probability and can thus be ignored. Z is given by

$$\begin{aligned} Z &= \sum_{\vec{s}}\exp\{\sum_{\alpha=1}^{M}\sum_{i,j=1}^{H^{\alpha}} s_i^{\alpha}s_j^{\alpha}w_{ij}^{\alpha} + \sum_{\alpha,\beta\neq\alpha}^{M}\sum_{i=1}^{H^{\alpha}}\sum_{j=1}^{H^{\beta}} s_i^{\alpha}s_j^{\beta}w_{ij}^{\alpha\beta} + \sum_{\alpha=1}^{M}\sum_{i=1}^{H^{\alpha}} h_i^{\alpha}(\vec{x})s_i^{\alpha} - J\sum_{\alpha=1}^{M}(\sum_{i=1}^{H^{\alpha}} s_i^{\alpha} - A^{\alpha})^2\} \\ &\approx \sum_{\vec{s}^1}\cdots\sum_{\vec{s}^M}\exp\{\sum_{\alpha=1}^{M}\sum_{i,j=1}^{H^{\alpha}} s_i^{\alpha}s_j^{\alpha}w_{ij}^{\alpha} + \sum_{\alpha,\beta\neq\alpha}^{M}\sum_{i=1}^{H^{\alpha}}\sum_{j=1}^{H^{\beta}} s_i^{\alpha}s_j^{\beta}w_{ij}^{\alpha\beta} + \sum_{\alpha=1}^{M}\sum_{i=1}^{H^{\alpha}} h_i^{\alpha}(\vec{x})s_i^{\alpha}\} \end{aligned} \quad (8)$$

where $\vec{s}^{\alpha}, \alpha = 1, \ldots, M$ denote the permissible states in module α. Note that Z contains $\Pi_{\alpha}A^{\alpha}$ terms. Since the learning rule is given by gradient descent on the Kullback divergence, Eqs. (3) or (4), the complexity of the learning rule is also $O(\Pi_{\alpha}A^{\alpha})$.

An interesting class of modular networks is obtained when $A^{\alpha} = 1$ for all α. Let $\vec{s}_i^{\alpha}$ denote the state in module α with neuron i active and all other neurons quiescent. Note that total activity 1 does not mean that each module can be in only one of the states $\vec{s}_i^{\alpha}$. The stochastic nature of the network allows

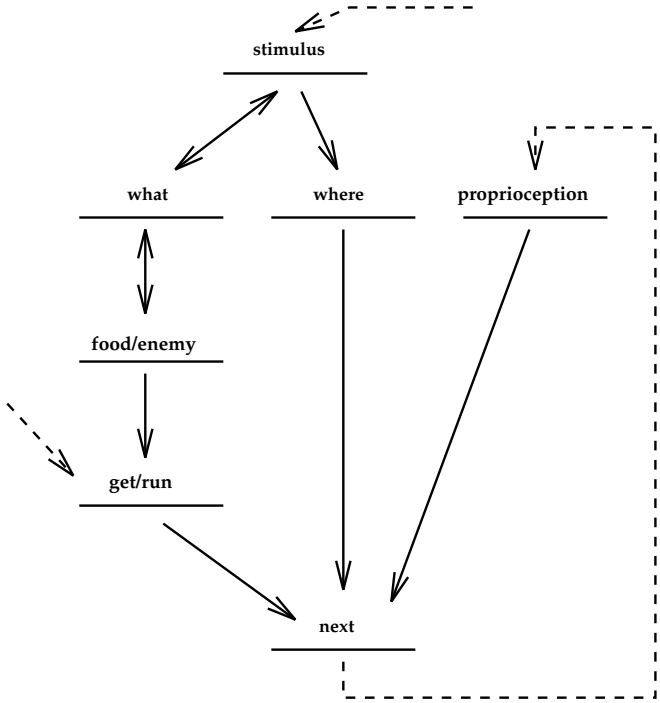

Figure 3: Example of an abstract modular architecture for sensori-motor tasks. The 'stimulus' layer represents direct sensory input (vision or auditory). All other layers are modules with lateral inhibition as described in the text and have low total activity. All weights are denoted by arrows between modules and can be trained by efficient BM learning rules.

equilibrium probabilities where the network can be in each of the states $\vec{s}_i^{\alpha}$ with a finite probability. However, the sum of these probabilities is one, since all other network states are forbidden.

The partition function becomes

$$Z = \sum_{i^1=1}^{h^1} \dots \sum_{i^M=1}^{h^M} \exp\{w_{i^1 i^2}^{12} + w_{i^1 i^3}^{13} + \dots + w_{i^{M-1} i^M}^{M-1\,M} + h_{i^1}^1(\vec{x}) + \dots + h_{i^M}^M(\vec{x})\}$$

Z involves a sum of $\Pi_i^M h_i$ terms. Exact learning rules can now be derived by gradient descent on the Kullback divergence.

3.2 Feed-back, multiple tasks and choice of action

To see how the above formalism can be applied to the study of intelligence, consider the example given in Fig. 3. This architecture is intended to describe goal-oriented behaviour in a *very* simplified way. It is clear that many details in this example are in fact mayor areas of research, such as vision, motor control and planning. The aim is not to solve any of these problems, but merely to illustrate how to provide learning rules for the total system. Each of the permissible states of these modules can represent some significant category. The module labeled 'stimulus' represents the visual stimuli in the outside world. The module labeled 'what' represents a finite number of relevant objects such as different types of food and different enemies. The 'where' module represents the location of the object relative to the animal. Through the layer 'food enemy', the tasks 'get' or 'run' are activated. Given the location of the object ('where'), the ego-location ('proprioception') and the task the 'next' move can be calculated. After the move, the proprioceptive information is changed and a new move is calculated until the goal is reached.

The role of feed-back can be studied in this architecture by observing that in the absence of a task specification, for some visual scenes both 'food' and 'enemy' concepts may be activated. However, signals external to the nervous system may stimulate the hungry animal such that 1) through feed-back the

appropriate representations are disambiguated in the 'what' and 'where' modules and 2) the 'get' action is chosen and executed.

4 Discussion

In this paper, I have indicated how Boltzmann Machines can be used for training of modular architectures. This approach may be helpful for modeling intelligent behaviour, when the components and what they encode, are known to some extend, and a learning algorithm is required to train the interconnectivity of the modules. There are still various problems with this approach.

- Although the lateral inhibition has reduced the complexity of the learning rule significantly, the computing time will still be sizable for large modular architectures. As I mentioned in the introduction, this problem can only be solved through hardware implementations, making full use of the parallellism of the Boltzmann Machine dynamics.
- For 'small' modular architectures, the analytic expressions for the learning rules become very complex. Also, some changes in the architecture, like adding a module, require that the learning rules are rederived from scratch. However, for every modular architecture there exists a straightforward procedure to calculate the learning rules. We are currently considering the possibility of generating the learning rules automatically, using symbolic programming methods.
- Through inhibition we have ensured that only small subsets of neurons are active in any task. This is very much in line with biological findings on sparse coding. However, in the current formulation, all modules are still involved in all tasks. This contributes to the complexity of the learning rules, but is also undesirable from a functional point of view. We are considering mechanisms to implement competition between modules.

If networks are designed for the execution of multiple tasks, a logical consequence is that for each individual task the majority of the neurons will be unspecified, because they are not involved in that task. In [6] we developed learning rules for general Boltzmann Machines that can learn when only a fraction of the visible units are specified.

Acknowledgements

This work was partly sponsored by the Japanese Real World Computing Program and the Dutch Foundation for Neural Networks (SNN).

References

[1] H.J. Kappen. Using boltzmann machines for probability estimation: A general framework for neural network learning. In E.S. Gelsema and L.N. Kanal, editors, *Proc. Pattern Recognition in Practice IV*, pages 299–312. Elsevier, 1994.

[2] H.J. Kappen. Deterministic learning rules for boltzmann machines. *Neural Networks*, 1994. Accepted for publication.

[3] P. van der Laar and H.J. Kappen. The em algorithm for boltzmann machines. Unpublihed, 1994.

[4] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the theory of neural computation*, volume 1 of *Santa Fe Institute*. Addison-Wesley, Redwood City, 1991.

[5] D. Ackley, G. Hinton, and T. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

[6] M. Nijman and H.J. Kappen. Using boltzmann machines to fill in missing values. Unpublished, 1994.