Learning Structure with Many-Take-All networks

D.Tax and H.J.Kappen

RWCP*** Novel Functions SNN† Laboratory
Dept. of Medical Physics and Biophysics, University of Nijmegen
Geert Grooteplein 21, NL 6525 EZ Nijmegen, The Netherlands

Abstract. It is shown that by restricting the number of active neurons in a layer of a Boltzmann machine, a sparse distributed coding of the input data can be learned. Unlike Winner-Take-All, this coding reveals the distance structure in the training data and thus introduces proximity in the learned code. Analogous to the normal Radial Basis Boltzmann Machine, the network uses an annealing schedule to avoid local minima. The annealing is terminated when generalization performance deteriorates. It shows symmetry breaking and a critical temperature, depending on the data distribution and the number of winners. The learned structure is independent of the details of the architecture if the number of neurons and the number of active neurons are chosen sufficiently large.

1 Introduction

In biological systems it is assumed that objects in the outside world are encoded by a sparse distributed code where the input data is encoded by several feature vectors, one for each active neuron in the code. The important advantage of this sparse coding is the ability of obtaining knowledge about the underlying structure of received data by the coding of the objects. When two objects share an active neuron in their coding, they share a property encoded by the feature vector belonging to the neuron. By the number of active neurons that overlap, the distances in a high dimensional pattern space of the these objects can be found. Thus the complete topology of places of objects and distances between objects in pattern space can be deduced. Koenderink [4] showed that this can be done in biological systems by fully using the modalities (and the cohesion within the modalities) of the perceptual data the biological brain perceives.

In 1994 Kappen [2] introduced lateral inhibition in a Boltzmann machine to reduce the computational costs of executing this Boltzmann machine. By allowing a restricted number of neurons to become active, the Boltzmann Machine encodes all objects by a sparse distributed code. In this paper we will show that with the use of this architecture, called a Many-Take-All network, the extra distance information from the input data can be encoded by a set of neurons.

*** Real World Computing Partnership
† Foundation for Neural Networks

In Section 2 we will start with a general introduction of a Boltzmann machine with a restricted number of winners. In Section 3 the annealing schedule and symmetry breaking will be explained. We will show in Section 4 that in contrast with the Winner-Take-All network a Many-Take-All network can learn all topological information of data which can easily be inspected by looking at the coding In addition we show that in high dimensional data spaces the Many-Take-All network can offer a more compact representation.

2 The restricted Boltzmann machine

The Boltzmann Machine we will consider consists of a set of neurons $\mathbf{x} = (x_1, ...x_n), x_i \in \mathbf{R}$, and a set of hidden neurons $\mathbf{s} = (s_1, ..., s_h), s_j \in [0,1]$. We call the connections between $\mathbf{x}$ and $\mathbf{s}$ w_{ij}. We will also use thresholds in the hidden layer, which will be called θ_j. By presenting training patterns $\mathbf{x}$ this Boltzmann machine can learn the probabilities associated with these training patterns (the probability of pattern $\mathbf{x}$ will be called $q(\mathbf{x})$). During the training of this Boltzmann machine a partition function has to be calculated. The number of terms in this partition function depends in an exponential manner on the size of the network so it is very time consuming to train this network.

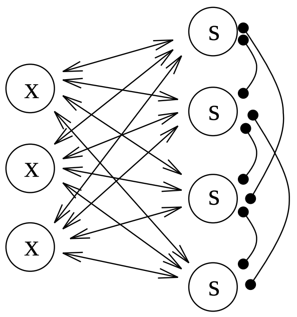

Fig. 1. The architecture of a Radial Basis Boltzmann machine. $\mathbf{x}$ are input neurons and $\mathbf{s}$ are hidden neurons. The connections with circles are inhibitory weights, the connections with arrows are learnable.

We can, as shown by Kappen [2], introduce lateral inhibition in the hidden layer of a Boltzmann machine. Then the number of permissible states in the network and so the number of terms in the partition function will be reduced. This results in a serious reduction of the required training time. We add an threshold $J(2h_0 - 1)$ to each neuron, where $J > 0$ the strength of the lateral inhibition (see Figure 1). If h_0 neurons are permitted to be 'on', then the local field of hidden unit j becomes

$$-\beta \sum_{j=1}^{h} ||\mathbf{w}_j - \mathbf{x}||^2 s_j + \sum_{j=1}^{h_0} \theta_j s_j - J \left(\sum_{j' \neq j}^{h} s_j - 2h_0 + 1 \right) . \qquad (1)$$

In the limit of an infinite strong lateral inhibition the $p(\mathbf{x})$ reduces to:

$$p(\mathbf{x}) = \frac{1}{Z} \sum_{(j_1 \cdots j_{h_0})} \exp \left\{ \sum_{\alpha=1}^{h_0} \left[-\beta ||\mathbf{w}_{j_\alpha} - \mathbf{x}||^2 + \theta_{j_\alpha} \right] \right\} \qquad (2)$$

Here, $\sum_{(j_1 \cdots j_{h_0})}$ means we have a summation over all combinations of h_0 hidden units out of h hidden units (so this summation will consist of $\binom{h}{h_0}$ terms). The factor $\exp(Jh_0^2)$ has disappeared because of the normalization.

Now the learning rules for the Many-Take-All network can be derived by performing a gradient descent on the Kullback divergence K between the target probability density $q(\mathbf{x})$ and the network probability density $p(\mathbf{x})$ by variating the weights w_{ij}. The complexity of the learning rule for this Boltzmann Machine is by the introduction of lateral inhibition diminished from a 2^h dependency to a $\binom{h}{h_0}$.

3 Annealing schedule and symmetry breaking

By applying gradient descent on the Kullback divergence one hopes to reach the global minimum of K. Unfortunately, this learning rule often causes the network to converge not to the global, but to a local minimum. To prevent this and as discussed in Kappen, Nijman [3], an annealing schedule can be used, where β plays the role of inverse temperature. Here one starts with random weights at a high temperature (small β). Then repeatedly the network is trained until the weights are converged and the temperature is lowered (or β is increased). The procedure is repeated until the Kullback divergence on an independent test set of patterns is at a minimum and starts to increase. Then the best modeling of the data is achieved. A check on the test set in contrast to a check on the train set is used to avoid overfitting on the training patterns. When the network is trained in this way, local minima can be avoided effectively, as was shown for $h_0 = 1$ in [2].

When β is increased, the weight vectors will shift, and thus specialize, from the average of the big cluster to the average of a certain sub cluster. These shifts do not occur smoothly, but there will be several symmetry breakings at critical temperatures. This mechanism of symmetry breaking resembles the clustering in statistical mechanics (see for instance Rose et al. in [7]) and is seen before in the RBBM (Kappen, [2]). The critical β for the first symmetry breaking can be calculated.

To do this calculation we will take the average of all patterns in the origin. Thus $w_{ij} = 0$ for small β. At some $\beta > 0$ this solution gets unstable and we will get the first phase transition. We expand the learning rule for small β. Assuming that the weights will stay small and that $h \gg h_0$, we can derive to lowest order in w_{ij}:

$$(I - 2\beta h_0 C_{xx})\mathbf{w}_j = 0 \tag{3}$$

where C_{xx} is the covariance matrix of the training patterns. Thus the critical temperature will be:

$$\beta_c = \frac{1}{2\lambda_m h_0} \tag{4}$$

where λ_m is the largest eigenvalue of C_{xx}. This is in agreement with Rose et al.[7] for $h_0 = 1$.

We see that the number of winners and the variance along the largest principal axis of the training data, determine the value of β_c. The more winners the network has, the faster the first phase transition will occur.

4 Results

4.1 16 clusters in 4 dimensions

The properties of this Many-Take-All network can best be showed by demonstrating an example: a data distribution in four dimensions. The data clusters are placed on the angles of a four dimensional hypercube (so it consists of sixteen clusters). A winner-Take-All network requires $h = 16$ and $h_0 = 1$. In this dataset distances $0 \ldots 4$ between clusters can be distinguished. A minimal code that contains these distances is $h = 8$ and $h_0 = 4$, which is smaller than the Winner-Take-All. After learning, we see that the sparse representation correctly gives the distance distribution between the corresponding data clusters. Furthermore, we observe that pairs of neurons encode cartesian axes in the data space. For instance, neuron 1 and 5 always have opposite value and encode whether the first coordinate of the particular cluster is 0 or 1 (see Table 1).

cluster	s	cluster	s	cluster	s	cluster	s
1	 1111	5	..1. 11.1	9	...1 111.	13	..11 11..
2	1... .111	6	1.1. .1.1	10	1..1 .11.	14	1.11 .1..
3	.1.. 1.11	7	.11. 1..1	11	.1.1 1.1.	15	.111 1...
4	11.. ..11	8	111. ...1	12	11.1 ..1.	16	1111 ....

Table 1. The encoding of a four dimensional cubic distribution by eight hidden units with four winners.

4.2 Scaling of learning properties

The number of distances the network can distinguish with this coding, depends on the number of neurons overlap and thus on the number of winners in the network. In general, with h_0 winners and enough hidden units available, the distances $0, \ldots, h_0$ can be encoded. Unfortunately, the complexity of the learning rule is roughly proportional to $\binom{h}{h_0}\binom{h_0}{2}nh$ (see eq (2)), so the more winners are taken, the more time the learning will require (see Table 2).

dim	Winner-Take-All		Many-Take-All		
	hidden	time (s)	hidden	winners	time (s)
1	2	1	2	1	1
2	4	2	4	2	5
3	8	6	6	3	72
4	16	37	8	4	947
5	32	192	10	5	11511
6	64	1292	12	6	88855

Table 2. Comparison of learning times of Winner-Take-All and Many-Take-All network for cubic distributions.

When more than one winner is introduced a new phenomenon occurs. Although in principle the network can encode $\binom{h}{h_0}$ different clusters, in practice some combinations are not used. This is because the neurons not only encode the clusters but also encode information over the distances between the clusters. This distance constraint makes it for instance impossible that in the first example neurons will fire simultaneously. Although in small and low dimensional problems the number of neurons for Many-Take-All are sometimes higher than Winner-Take-All, in higher dimensions and larger numbers of hidden units, the Many-Take-All becomes more efficient. To encode for example the cubic distribution of clusters in a n-dimensional space, a Winner-Take-All network requires 2^n units, while a Many-Take-All needs $2n$ (see Table 2).

4.3 Hierarchical data structures

We show here how Many-Take-All network can be used to learn structure in hierarchical data sets. Ultrametric sets can be represented as the leafs of a tree, where distance is defined as the number of nodes to a common parent [6]. Clearly, ultrametric structure is not represented in the output coding of Winner-Take-All networks, and Many-Take-All offers a clear advantage. As an example we take an ultrametric data set, consisting of 8 clusters in 4 dimensions.

The results are shown in Table 3 (left) for a network with $h = 6$ and $h_0 = 2$. After learning, the distances in the code display the same hierarchical structure

cluster	coding $h = 6, h_0 = 2$	coding $h = 14, h_0 = 3$
1	110 000	1110 000 0000 000
2	110 000	1101 000 0000 000
3	101 000	1000 110 0000 000
4	101 000	1000 101 0000 000
5	000 110	0000 000 1110 000
6	000 110	0000 000 1101 000
7	000 101	0000 000 1000 110
8	000 101	0000 000 1000 101

Table 3. Coding of an ultrametric data set consisting of eight clusters in four dimensions.

as the original data, although not in great detail, using $h = 6, h_0 = 2$ can distinguish only four different clusters. This result is robust for changes in h and h_0. In Table 3 (right) shows the results on the same data for a network with $h = 14$ and $h_0 = 3$. As can be seen, the fact that we have used a larger network does not affect the basic distance structure of the coding that we obtained, but adds more details. The network with $h = 14, h_0 = 3$ distinguishes all eight clusters and their distances.

References

1. Földiák, P.: Forming sparse representations by local anti-Hebbian learning. Biological Cybernetics **64** (1990) 165-170
2. Kappen, H.: Deterministic Learning Rules for Boltzmann Machines. Neural Networks **8** (1994) 537-548
3. Kappen, H., Nijman, M.: Radial Basis Boltzmann Machines and Learning of Missing Values. Proceedings of the World Congres on Neural Networks **1** (1995) 72-75
4. Koenderink, J.J.: Simultaneous Order in Nervous Nets form a Furnctional Standpoint. Biological Cybernetics **50** (1984) 35-41
5. Koenderink, J.J.: (1984). Geomtrical Structures Determined by the Functional Order in Nervous Nets Biological Cybernetics **50** (1984) 41-50
6. Rammal, R., Toulouse, G., Virasoro, M.A.: Ultrametricity for physicists. Rev.Mod.Phys. **3** (1986) 765-787
7. Rose, K., Gurewitz, E. and Fox, G.: Statistical Mechanics and Phase Transitions in Clustering. Physical Review Letters **65** (1990) 945-948
8. Saund, E.: A Multiple Cause Mixture Model for Unsupervised Learning. Neural Computation **7** (1995) 51-57

This article was processed using the LaTeX macro package with LLNCS style