Internal Use Only

TR-I-0130

Shift- invariant Deterministic Boltzmann Machines for Phoneme Recognition

シフト不変決定論的ボルツマン マシン による音声認識

Jean-Claude DANG, Shin'ichi TAMURA, and Hidefumi SAWAI
ジャン クロード ダン, 田村 震一, 沢井 秀文

1989 . 12

内容梗概

決定論的ボルツマンマシン (DBM) とは従来の統計論的ボルツマンマシンよりも学習がはるかに速いニューラルネットワークであり、統計論的ボルツマンマシンから導かれるものである。本論文では、決定論的ボルツマンマシン (DBM) の理論について簡単に説明した後、決定論的ボルツマンマシンの音声認識への適用について述べる。音素の時間軸方向の伸縮を考慮しない静的な DBM は、bdg の認識に於て、平均98.6 % (最高99.1 %) の認識率を達成した。また、全子音では97%の認識率を達成した。音素の時間軸方向の伸縮を考慮した動的な DBM では、静的な DBMに較べて数パーセントの音素認識率の低下が認められるものの、状態フィードバックの動的な構造の効果が確認された。

ATR 自動翻訳電話研究所
ATR Interpreting Telephony Research Laboratories

CONTENTS

Shift-invariant Deterministic Boltzmann Machines for Phoneme Recognition

Abstract The Deterministic Boltzmann Machine (DBM) is a form of neural network that learns much faster than the original stochastic Boltzmann Machine that it is derived from. In this paper we overview briefly the theory of DBMs, and describe their application to speech recognition. In a static phoneme configuration task the DBM obtained an average recognition rate of 98.6 % (best: 99.1%) for the "bdg" task, and 97 % for an all-consonant task. In a dynamic recognition task (including time-shifts), rates are less good by a few percent, but a state-feedback dynamic architecture provided some improvement.

Introduction

Boltzmann Machines are neural networks of stochastic binary units. They use a learning algorithm which derives from a probabilistic model [1]. Studies have shown that their performance in pattern recognition tasks can be very good, but at the cost of a very large amount of computing time [2]. This large computational requirement results from the necessity to simulate a stochastic process and perform simulated annealing. Such a drawback makes the Bolztmann Machines impractical for speech recognition tasks [3,4].

A variant of the Boltzmann Machine algorithm, based on a mean field theory approximation has been proposed by Peterson and Anderson [5], and also studied by Hinton [6]. In this new model the network is composed of *analog* units which are activated according to a *deterministic* rule. These characteristics and the fact that simulated annealing is not needed make the so-called Deterministic Boltzmann Machines (DBM's) learn much faster than the original stochastic Boltzmann Machines (SBM's).

This paper is split into four main sections. In the first one which is a synthesis of [5] and [6] we expose the theory of DBM's. In the second section we describe experiments where DBM's are applied to static phoneme recognition. The next section deals with dynamic recognition of phoneme segments with DBM's. We propose two architectures and evaluate their performance on the recognition of the phonemes *b* , *d* , *g* . Finally we discuss the DBM algorithm with respect to the shift-invariance property.

1. Overview of DBM theory

1.1 Network

DBM's are neural networks of analog units connected by symmetric links. The links between units can be arbitrary but with the restriction that no unit should be connected to itself. With respect to the network dynamics DBM's are similar to analog networks described by Hopfield [7] and Cohen-Grossberg [8]. These networks are characterized by the existence of an energy quantity which is minimized by the network during its activation.

More specifically in the case of the DBM's the so-called free energy is defined by :

$$F = - \sum_{i<j} w_{ij} o_i o_j + \sum_i [o_i \log o_i + (1 - o_i) \log (1 - o_i)] \quad (1)$$

where o_i is the activity value of unit i
and w_{ij} is the weight value between units i and j

This name of 'free energy' comes from a similarity with the systems of statistical mechanics. The first term in formula (1) corresponds to a quadratic energy while the second term corresponds to an entropy. This second term is needed to have continuous values for the units activation values.

Derivation of F with respect to o_i yields :

$$\frac{\partial F}{\partial o_i} = - \sum_{j \neq i} w_{ij} o_j + \log \left(\frac{o_i}{1 - o_i} \right) \quad (2)$$

Solving (2) for $\frac{\partial F}{\partial o_i} = 0$, we get :

$$o_i = \sigma \left(\sum_{j \neq i} w_{ij} o_j \right) \quad (3)$$

where σ is the logistic activation function :

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \quad (4)$$

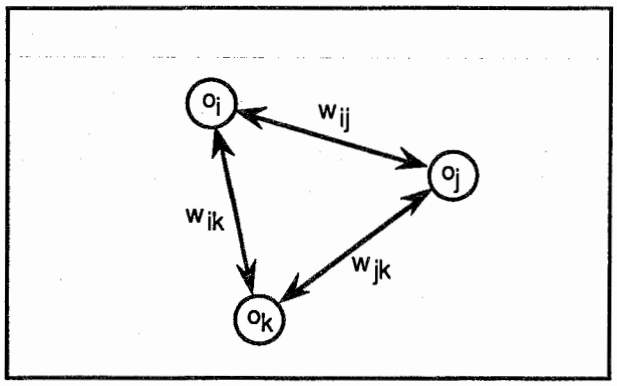

Fig. 1 Symmetric connections

Note that the derivation is possible thanks to the existence of the logarithmic term in the expression for F .

Equation (3) gives the relative minimum of F with respect to o_i for o_j , $j \neq i$ kept constant. In the version of the algorithm with asynchronous update[1], each unit is activated sequentially according to (3). F decreases with each activation and after enough iterations we expect to reach a stable point where all the partial derivatives $\frac{\partial F}{\partial o_i}$ are zero, i.e. a local minimum of F. This is different from the stochastic Bolztmann Machine where a *global* minimum is searched by simulated annealing.

It appears that DBM's should be faster than SBM's for at least two reasons. First the deterministic activation function (3) eliminates the need for a costly stochastic simulation : we don't have to run the network for a large number of times in order get probability estimates. Moreover simulated annealing is not necessary since we only look for a local minimum of F.

1.2 Learning algorithm

So far the network we have described is no more than an analog network of the Hopfield type. The true originality of DBM's lies in their learning algorithm which is adapted from the original SBM's. Peterson and Anderson [5] show how this algorithm can be derived from the SBM's using an approximation method well-known to physicists as

[1] It is also psoosible to activate the units in parallel. We didn't choose this solution because it seemed more likely subject to oscillatory behavior (see [5] for details on parallel update).

mean field theory. Hinton [6] shows how DBM's can be viewed in a rigorous way as exact systems and not only as approximations of SBM's. We present here the learning algorithm in an intuitive manner, and the reader should refer to the above mentioned papers for more theoretical justifications.

For the purpose of pattern recognition, the units of the network are separated, into input, output and hidden units. The input units will contain the pattern to be recognized while the output units will give the response of the network. The hidden units act as extra feature detectors to catch high order relationships between input-output pairs.

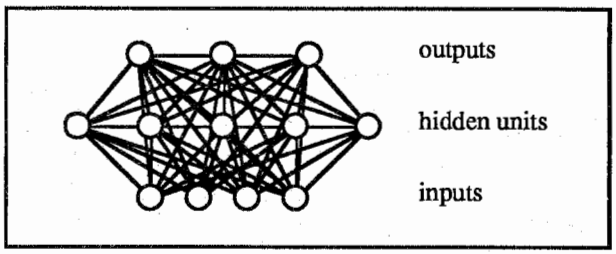

Fig. 2 Network

1.2.1 Minus and Plus phases

Learning each input pattern in a DBM consist of two phases, so-called 'Plus' and 'Minus' phase. These two phases are quite similar in principle and follow the same order of operation.

For each phase the units of the network are divided into two groups : one group of 'free' units which will be activated and will change their value and one group of 'clamped' units whose value will remain constant throughout the activation process. The clamped units are a way to impose a constraint to the network, and their value is determined by the outside world. In the case of pattern recognition the input units are always clamped.

- Minus phase

In the Minus phase, only the input units are set with one input pattern and they are clamped, while the output units and the hidden units are free. The free units are first initialized with some initial values (a neutral value of 0.5 for each unit seems best), and they are activated until a minimum F_-^* is reached. This phase is the counterpart of the forward pass for back-propagation networks. Given an input, we look at the spontaneous response of the network.

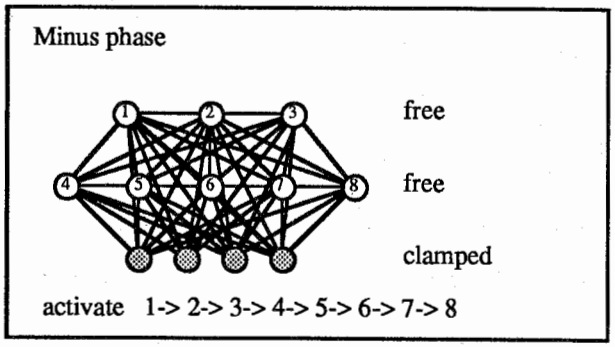

Fig. 3 Minus phase

- Plus phase

In the Plus phase, the input units are set with the input pattern, and the output units are set with the output which the network has to learn. These units are clamped while the hidden units are free. For initial conditions they are initialized with the values of the hidden units from the minimum of the minus phase F_-^* . Then they are activated until a minimum F_+^* is reached. The reason for using the state of the hidden units in F_-^* as initial values is that we expect F_-^* and F_+^* to be quite close. Starting from a point close to the local minimum allows the network to reach it faster.

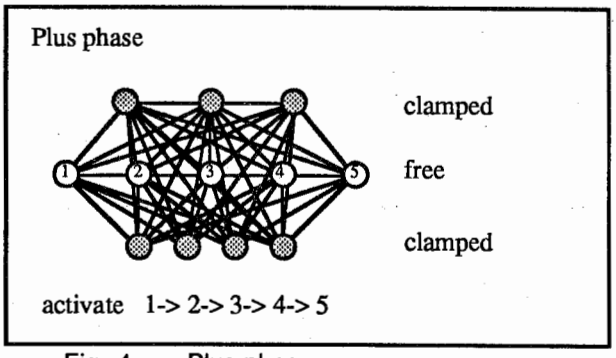

Fig. 4 Plus phase

The Plus phase can be compared to the backward pass of the back-propagation algorithm. The important difference is that in the Plus phase, the actual output value is given to the network instead of the error signal in the case of back-propagation. Thus the teaching information is more direct than for back-propagation. This property will hopefully allow the network to learn faster.

1.2.2 Weight update

After performing the Plus and Minus phases, the weights are updated according to the following law :

$$\Delta w_{ij} = \varepsilon \left(- \frac{\partial F^*}{\partial w_{ij}}(F_+^*) + \frac{\partial F^*}{\partial w_{ij}}(F_-^*) \right) \qquad (5)$$

where ε is a small positive constant (step size)

Eq. (5) has a simple intuitive explanation. The first term in equation (5) lowers the energy of F_+^* , the state which we would like to teach to the network. On the other hand, the second term in Eq. (5) raises the energy of F_-^*. In the end, we expect F_+^* to become lower than F_-^* , so that hopefully F_+^* will become the response of the network (Figs. 5, 6).

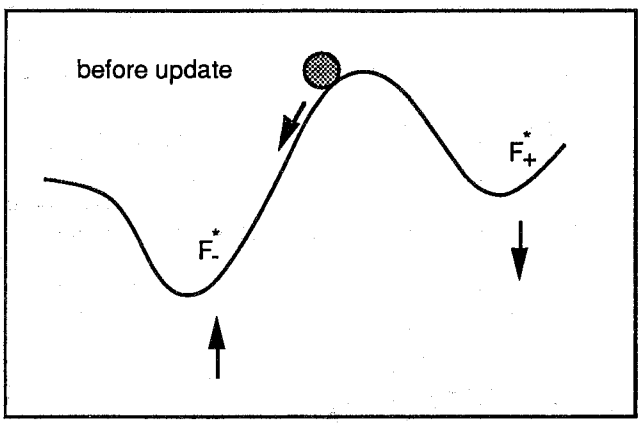

Fig. 5 Before weight update

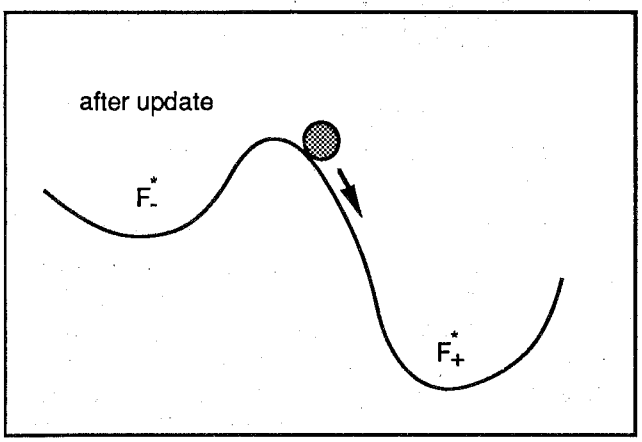

Fig. 6 After weight update

The partial derivative $\frac{\partial F^*}{\partial w_{ij}}(F_+^*)$ (resp. $\frac{\partial F^*}{\partial w_{ij}}(F_-^*)$) is the variation of F at the minimum F_+^* (resp. F_-^*) over a small change in w_{ij}. We have for the expression of $\frac{\partial F^*}{\partial w_{ij}}$:

$$\frac{\partial F^*}{\partial w_{ij}}(F^*) = \frac{\partial F}{\partial w_{ij}}(F^*) \Big|_{o_k} + \sum_k \frac{\partial F}{\partial o_k}(F^*) \Big|_{w_{pq}} \qquad (6)$$

The second term in eq. (6) is zero because F^* is a minimum of F. Thus in a first-order approximation, a variation of the weights changes the value of F at the minimum, without changing its position.

Deriving Eq. (1) with respect to o_i , we get :

$$\frac{\partial F}{\partial w_{ij}}(F^*) = - o_i^* \, o_j^* \qquad (7)$$

From (5) and (7) :

$$\Delta w_{ij} = \varepsilon (o_i^+ \, o_j^+ - o_i^- \, o_j^-) \qquad (8)$$

where o_i^+ and o_i^- are the values of the units for the minima F_+^* and F_-^* respectively.

1.2.3 Learning a data set

To learn a sample set, each sample data is presented to the network, the Plus and Minus phases are performed, and the weights updated . The whole training set is presented repeatedly as many times as necessary. A convenient estimate of the performance of the network on the training data is the mean square error between the outputs of the network and the correct values. So we considered that the training was over when the mean square error fell below a given threshold.

2. Static phoneme recognition

We applied the theory of DBM's to phoneme recognition. In a first step we evaluated the model on speaker dependent, static recognition problems.

2.1 Data set

We used the same training and testing sets as in the TDNN's experiments of Waibel et al. [9]. The speech segments were extracted from ATR's large vocabulary database of 5240 common Japanese words uttered in isolation by one male native speaker (a professional announcer). All utterances were recorded in a soundproof booth and digitized at a 12 kHz sampling rate. Frames consisting of 16 melscale coefficients were then computed at intervals of 10 ms. The tokens were 15 frames long (150 ms) and centered around the hand-labeled vowel onset.

2.2 Network architecture

For all our experiments, we used a fully connected architecture (see Fig. 7). Note in particular that the output units are directly connected to the input units. The bias unit is an input unit which is always set to 1. Its effect is to introduce an offset in the activation function of the units.

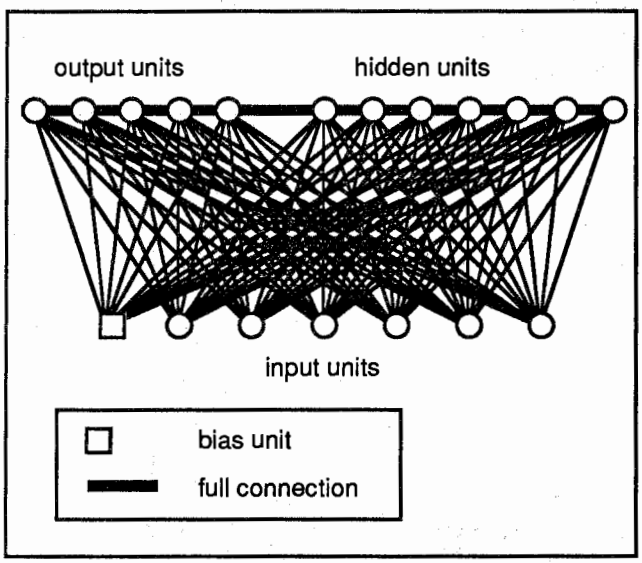

Fig. 7 Network for static phoneme recognition

2.3 Experiments

2.3.1 b , d , g

Our first task was the recognition of the phonemes *b* , *d* , *g* . The results have been reported in [10]. Recognition rate on test data was 98.6% on average, with a maximum of 99.1%. However the training time was quite long due to the large number of hidden units (100 hidden units).

We have since found out that it is indeed possible to reduce the number of hidden units (to 10 - 20) without loss of performance. Learning becomes very fast (several minutes of CPU time on an Alliant parallel computer with eight processors). The same average recognition rate of 98.6% was observed.

2.3.2 All consonants

After these very encouraging results, we trained a DBM for the discrimination of all 18 consonants. The network was just an expanded version of our *b* , *d* , *g* network, which means that all the units where connected. It had 241 inputs (15 x 16 melscale coefficients + 1 bias unit), 18 outputs (one for each category) and 10 hidden units.

The results of our experiment are shown in Table 1. All the training samples were correctly learned. On average, the recognition rate on test data was 97.0%.

We didn't encounter any problem of convergence with the DBM algorithm. About 90 presentations of the training set were needed for training, and it took 5 hours of cpu time with an Alliant.

phoneme	# tokens	# correct	% correct
b	227	226	99.6
d	179	164	91.6
g	252	231	91.7
p	15	13	86.7
t	440	426	96.8
k	1164	1132	97.3
m	481	461	95.8
n	265	249	94.0
N	488	483	99.0
s	538	534	99.3
sh	177	177	100.0
h	207	204	98.6
z	115	114	99.1
ch	71	64	90.1
ts	177	163	92.1
r	722	713	98.8
w	81	79	97.5
y	174	169	97.1
total	5773	5602	97.0

Table 1 Recognition results on testing data for all consonants task

3. Shift-invariant phoneme recognition

The very good results which we obtained on static phoneme recognition show that DBM's are powerful classifiers and that they could be used in speech recognition systems. However it would be very desirable to have an architecture which is invariant under translations in time in order to achieve reliable phoneme spotting.

We have tried two architectures to deal with this problem. the first one uses shifted samples to train a network. The second one improves this method by using delay links. The task was the recognition of *b* , *d* , *g* on the same data as the static experiment.

3.1 Shifted network

The most straightforward way to achieve shift-invariance is to train a static network on shifted samples, using a small temporal window as input (see Fig. 8).

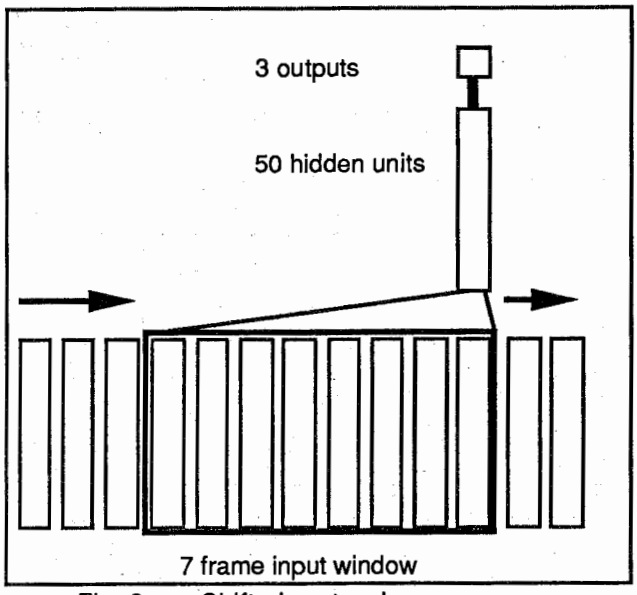

Fig. 8 Shifted network

Training on shifted segments results in a larger data set. This approach was used by McDermott et al. in their LVQ-based system [11].

The network of our experiment had 113 inputs (7 x 16 melscale coefficients + one bias unit), 50 hidden units and 3 outputs. Contrary to the static case, we found out that convergence was very slow. Training took about 500 iterations (compared to 40 for the static network). in 5 hours of cpu time.

Table 2 shows the recognition results on training and testing data. The performance on testing data is much lower than in the static case (98.6%). The network seems to have a lot of difficulty to learn this task. Contrary to the results that McDermott reports for LVQ2, shifting the samples didn't improve the performance of the network.

data set	training	testing
% recognition rate	99.7	95.1

Table 2 Recognition results for *b* , *d* , *g* with shifted network

3.2 State-feedback network

We tried an architecture described by Prager et al. in their pioneer work on SBM's applied to speech recognition. This architecture is a improvement of the preceding one in that it uses delay links (see Fig. 9). These links are used to give as input to the network at time t the state of both input and output units of the network at time t-1 . This realizes a kind of state feedback. The network has two types of information at its disposal : one is given by the input window and the other by the state of the previous time step. Thus it benefits from some kind of memory which will help it to make its decision, and accumulates information over time.

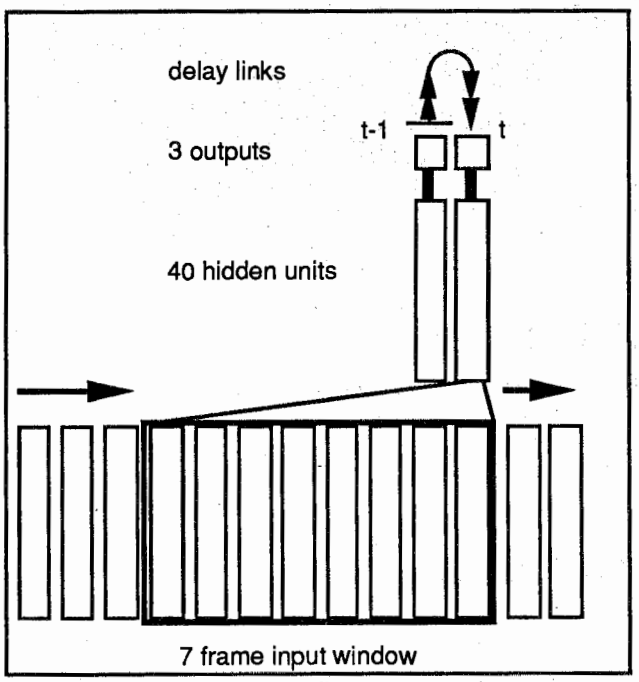

Fig. 9 State-feedback network

We trained the network depicted on Fig. 9 for the recognition of b , d , g. Training time was substantially lower than for the shifted network : about 2 hours and a half. It required only 240 iterations to converge.

Table 3 shows the recognition results. As expected, the introduction of delay links improved the recognition performance. Still the rate of 97.0% is lower than in the static case.

data set	training	testing
% recognition rate	99.5	97.4

Table 3 Recognition results for *b , d , g* with state-feedback network

4. Discussion

Our experiments have shown that DBM's can achieve very high performance on static recognition. But on dynamic tasks it decreased by several percents.

How can we explain this degradation of performance ? A possible reason is that we give to the network conflicting information during training. When the network is shifted over a sample, the desired output value is kept constant, with a value of one for the correct class and zero for all other outputs. But it may well happen that two samples from different classes have nevertheless similar beginnings or endings. Thus we may state that a given segment is in one case 'b' and in another case 'd'. This amounts to have overlapping categories. The characteristics of the DBM algorithm make it more difficult to deal with such overlapping categories than for example LVQ2, where several reference vectors are available for each class.

Another point which may account for the lower performance of the state-feedback is that the ideal response of such a network should be undetermined at the beginning of the sample where no information is available, and it should progressively become more pronounced at the end where all the sample has been input and all the relevant clues have been detected. So that giving a constant teaching output signal is not the best method to teach this kind of network.

5. Conclusion

We showed that DBM's can be very powerful pattern recognizers. We got very high performance on static phoneme recognition of 98.6% for the discrimination of *b , d , g*, and 97.0% for all consonants. For a shifted network recognition performance fell to 95.1% . This result could be improved to 97.4% with the introduction of a state-feedback architecture, but still lower than in the static case.

The preceding facts suggest that other dynamic phoneme recognition methods using DBMs should be studied in order to use DBM's pattern recognition ability to the full extent.

References

[1] Hinton, G.E. and T.J. Sejnowski. Learning and relearning in Boltzmann machines. *In:* Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations, eds. D.E. Rumelhart, J.L. McClelland, and the PDP group. Cambridge, MA: MIT Press. (1986)

[2] Kohonen, T. , G. Barna and R. Chrisley. Statistical Pattern Recognition with Neural Networks: Benchmarking studies. IEEE, Proc. of ICNN, Vol. 1, 61-68 (July 1988)

[3] Lippmann, R.P. Review of Neural Networks for Speech Recognition. *Neural Computation* 1,1-38 (1989).

[4] Prager, R.W. and F. Fallside. A comparison of the Boltzmann machine and the back-propagation network as recognizers of static speech patterns. *Computer Speech and Language* **22**, 179-183 (1987)

[5] Peterson, C. and J.R. Anderson. A Mean Field Theory Algorithm for Neural networks. *Complex Systems* 1,995-1019 (1987).

[6] Hinton, G.E. Deterministic Boltzmann Learning Performs Steepest Descent in Weight-Space. *Neural Computation* 1,143-150 (1989).

[7] Hopfield, J.J. Neurons with Graded Responses Have Collective Computational Properties like Those of Two-state Neurons. *Proceedings of the National Academy of Sciences U.S.A.* **81**, 3088-3092

[8] Cohen, M.A., and S. Grossberg. Absolute Stability Of Global Pattern Formation and Parallel Memory Storage by Competitive Neural Networks. IEEE Trans. SMC, Vol. SMC-13, No. 5, (September/October 1983)

[9] Waibel, A., T. Hanazawa, G. Hinton, K. Shikano and K. Lang. Phoneme Recognition Using Time Delay Neural Networks. Technical Report TR-1-006, ATR Interpreting Telephony Research Laboratories, October 1987 (also in ASSP, **37**, 3, March 1989).

[10] Dang, J.C. and H. Sawai. Deterministic Boltzmann Machines for Phoneme Recognition. *Proceedings of Fall Meeting of ASJ* 1989.

[11] McDermott, E. and S. Katagiri. LVQ2 for Phoneme Recognition. Proceedings of Spring Meeting of ASJ 1988.