An Application of Linear Response Learning

Martijn Leisink* and Bert Kappen†
Department of Biophysics
University of Nijmegen, Geert Grooteplein 21
NL 6525 EZ Nijmegen, The Netherlands
{martijn,bert}@mbfys.kun.nl

December 14, 1999

Abstract

Linear response is an approximation method for Boltzmann machines based on mean field theory. It is known that in the absence of hidden units this method can learn the network quite accurately with the costs of only one matrix inversion. We show that adding a flat distribution to the target can decrease the classification error. We apply linear response learning to a real world data set of digit recognition. We show that the this method can compete with other known methods. An advantage of linear response is the fast learning.

1 Introduction

Boltzmann machines are networks of stochastic binary variables (neurons). All neurons s_i are linked to each other with symmetric weights $w_{ij} = w_{ji}$. Due to this symmetry the probability distribution is given by the Boltzmann-Gibbs distribution which is a known function of the weights and thresholds of the network [1].

Since the exact computation of the statistics is intractable, one has to make an approximation. A well known approach to deal with this intractability was given by Peterson and Anderson [2] and is called 'mean field'. This method can be seen as a first order expansion around a tractable network, for which usually a decoupled network is used, but other structures are possible [3]. Kappen and Rodríguez present in [4] a nice way to obtain a better approximation for the correlations in the network which is still based on the mean field approximation. Additionally, in the absence of hidden units, their so called 'linear response method' allows fast approximate learning.

In this paper we learn ten Boltzmann machines with linear response and use them to solve an existing problem of digit recognition. We compare the performance with other known methods.

*Foundation for Neural Networks
†Real World Computing Partnership

2 Theory

Let us partition the neurons of a Boltzmann machine in a set of v visible units and h hidden units ($v + h = n$). Let α and β label the 2^v visible and 2^h hidden states of the network, respectively. Thus, every state $\vec{s}$ is uniquely described by a tuple $\alpha\beta$. The probability for a state $\vec{s}$ given the weights w_{ij} and thresholds θ_i of a Boltzmann machine is given by

$$\log p\left(\vec{s}\right) = -E\left(\vec{s}\right) - \log Z \tag{1}$$

where the energy is given by

$$-E\left(\vec{s}\right) = \frac{1}{2}\sum_{ij} w_{ij} s_i s_j + \sum_i \theta_i s_i \tag{2}$$

and $\log Z$ is a normalization constant. Learning consists of adjusting the weights and thresholds in such a way that the Boltzmann distribution on the visible units $p_\alpha = \sum_\beta p_{\alpha\beta}$ approximates a target distribution q_α as closely as possible.

A suitable measure for the difference between the distributions p_α and q_α is the Kullback divergence [5]

$$K = \sum_\alpha q_\alpha \log \frac{q_\alpha}{p_\alpha} \tag{3}$$

It is easy to show that $K \geq 0$ for all distributions p_α and q_α and $K = 0$ iff $p_\alpha = q_\alpha$ for all α.

Therefore, learning consists of minimizing K with respect to w_{ij} and θ_i which can be done by gradient descent. The learning rule is given by [1]

$$\begin{aligned} \Delta\theta_i &= -\eta \frac{\partial K}{\partial \theta_i} = \eta\left(\langle s_i\rangle_c - \langle s_i\rangle\right) \\ \Delta w_{ij} &= -\eta \frac{\partial K}{\partial w_{ij}} = \eta\left(\langle s_i s_j\rangle_c - \langle s_i s_j\rangle\right) \qquad i \neq j \end{aligned} \tag{4}$$

where the parameter η is the learning rate. The brackets $\langle\cdot\rangle$ and $\langle\cdot\rangle_c$ denote the 'free' and 'clamped' expectation values, respectively. The 'free' expectation values are averages over all patterns α and the probability distribution p_α. The 'clamped' expectation values are obtained by clamping the visible units in a state α and taking the expectation value with respect to q_α:

$$\begin{aligned} \langle s_i\rangle_c &= \sum_{\alpha\beta} s_i^{\alpha\beta} q_\alpha p_{\beta|\alpha} \\ \langle s_i s_j\rangle_c &= \sum_{\alpha\beta} s_i^{\alpha\beta} s_j^{\alpha\beta} q_\alpha p_{\beta|\alpha} \end{aligned} \tag{5}$$

$s_i^{\alpha\beta}$ is the value of neuron i when the network is in state $\alpha\beta$. $p_{\beta|\alpha}$ is the conditional probability to observe hidden state β given that the visible state is α. Note that in equations 4 and 5, i and j run over both visible and hidden units.

Thus, the learning rule contains clamped and free expectation values of the Boltzmann distribution. The computation of these expectations is intractable, because one has to sum over exponentially many terms to compute the averages.

For the special case of a Boltzmann machine without hidden units, there exist a powerful method to approximately learn the network in a very short time. A detailed explaination of the method can be found in [4]. Here we only describe the method. Define

$$m_i = \langle s_i \rangle_c \tag{6}$$

$$\chi_{ij} = \langle s_i s_j \rangle_c - \langle s_i \rangle_c \langle s_j \rangle_c \tag{7}$$

Due to the absence of the hidden units, the averages in the equations above can be obtained directly from the data set. The linear response learning rule approximates the weights and thresholds by

$$w_{ij} = \frac{\delta_{ij}}{1 - m_i^2} - \left(\chi^{-1}\right)_{ij} \tag{8}$$

$$\theta_i = \tanh^{-1} m_i - \sum_i w_{ij} m_j \tag{9}$$

A consequence of the linear response method is the introduction of so called 'diagonal weights', w_{ii}. Although such weights are not present in the definition of the exact Boltzmann machine, they turn out to play an important role within linear response theory. This can be explained by viewing the diagonal weights as the Onsager reaction term from statistical physics [6] or the TAP correction [7], which is a well known correction to the approximated correlations. In fact, one can show that these diagonal weights are equal to the TAP correction upto the approximation order.

3 Results

We demonstrate the quality of the above linear response method for Boltzmann Machine learning on a digit recognition problem. The data consists of 70,000 examples of handwritten digits (zero to nine) known as the MNIST database[1]. The original black and white images from the NIST data base were rescaled to a 20x20 image preserving their aspect ratio. The images were centered in a 28x28 image by computing the center of mass of the pixels and translating the image to the center. The data set is divided into a training set of 60,000 samples and a test set of 10,000 samples. In figure 3 a few samples from the data set are shown.

Our approach is to model each of the digits with a separate Boltzmann machine using the linear response method. We thus obtain ten different Boltzmann distributions over $n = 28 \cdot 28 = 784$ neurons given by

$$\log P\left(\vec{s}|W^\alpha\right) = -E\left(\vec{s}|W^\alpha\right) - \log Z\left(W^\alpha\right), \qquad \alpha \in \{0, \ldots, 9\} \tag{10}$$

where $W^\alpha = \left(w_{ij}^\alpha, \theta_i^\alpha\right)$ are the weights and thresholds for digit α. We then test the performance of these models on a classification task using the 10,000 test patterns. We classify each pattern to the model α with the highest probability for that pattern. The normalization $\log Z(W^\alpha)$ is intractable and since it depends on α, it affects the classification. We use its mean field approximation given by [4, 8]

$$\begin{aligned} -\log Z = & \frac{1}{2} \sum_{ij} w_{ij} m_i m_j + \sum_i \theta_i m_i \\ & + \frac{1+m_i}{2} \log \frac{1+m_i}{2} + \frac{1-m_i}{2} \log \frac{1-m_i}{2} \end{aligned} \tag{11}$$

[1] The MNIST database can be obtained from `http://www.research.att.com/~yann/ocr/`

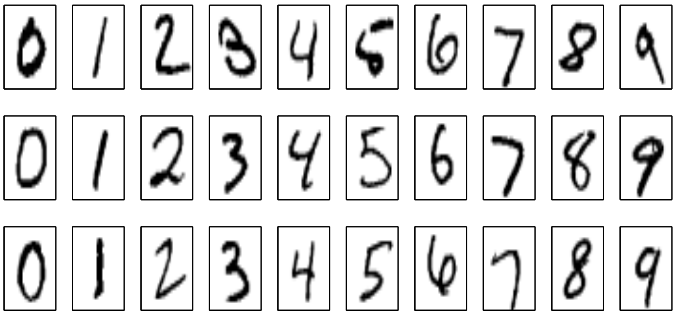

Figure 1: A few samples of the MNIST data set. Each digit is a gray scale image of 28x28 pixels.

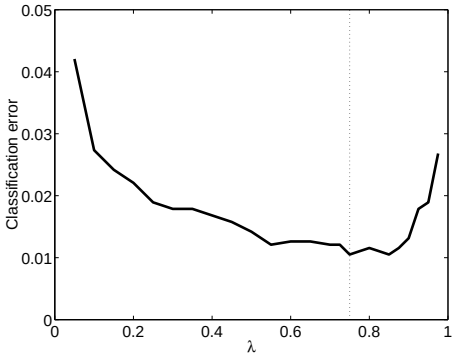

Figure 2: Classification error of two learned Boltzmann machines for the digits three and five for various values of λ. Based on this graph we choose $\lambda_{\text{opt}} = 0.75$. The error was obtained for a test set of 1902 samples.

For our data, the correlation matrix χ_{ij} in equation 7 is (close to) singular. Due to the inversion in equation 8, this results in very large weights and we should question the validity of the mean field approximation, which is based on a small weight expansion. We propose to solve this problem by adding a flat distribution to the training data:

$$q_\alpha \to \lambda q_\alpha + (1-\lambda)\frac{1}{2^n} \tag{12}$$

$$\langle s_i \rangle_c \to \lambda \langle s_i \rangle_c \tag{13}$$

$$\langle s_i s_j \rangle_c \to \lambda \langle s_i s_j \rangle_c + (1-\lambda)\delta_{ij} \tag{14}$$

In figure 3 we show the result of the Boltzmann machine classifier as a function of λ. For this case the classification task was reduced to the recognition of two digits: 'three' and 'five'. We see that the classification error depends strongly on the value of λ. Based on the figure, we choose $\lambda_{\text{opt}} = 0.75$.

A lot of work was done earlier on this data set. Table 3 summarizes the obtained results by linear response and other methods. Although linear response is certainly not the best of all methods, it fits well between other (not specialized) methods like ordinary multi-layer neural networks. The good performing network 'LeNet', for instance, is a neural network which is specially designed for digit recognition, whereas linear response is a general method

Method	*Error %*
Linear response	**3.52**
Linear classifier (1-layer NN)	12.0
Linear classifier (1-layer NN) [deskewing]	8.4
Pairwise linear classifier	7.6
K-nearest-neighbors, Euclidean	5.0
K-nearest-neighbors, Euclidean, deskewed	2.4
40 PCA + quadratic classifier	3.3
1000 RBF + linear classifier	3.6
SVM deg 4 polynomial	1.1
Reduced Set SVM deg 5 polynomial	1.0
Virtual SVM deg 9 poly [distortions]	0.8
2-layer NN, 300 hidden units	4.7
2-layer NN, 300 HU, [distortions]	3.6
2-layer NN, 1000 hidden units	4.5
2-layer NN, 1000 HU, [distortions]	3.8
3-layer NN, 300+100 hidden units	3.05
3-layer NN, 300+100 HU [distortions]	2.5
3-layer NN, 500+150 hidden units	2.95
3-layer NN, 500+150 HU [distortions]	2.45
LeNet-1 [with 16x16 input]	1.7
LeNet-4	1.1
LeNet-4 with K-NN instead of last layer	1.1
LeNet-4 with local learning instead of ll	1.1
LeNet-5, [no distortions]	0.95
LeNet-5, [huge distortions]	0.85
LeNet-5, [distortions]	0.8
Boosted LeNet-4, [distortions]	0.7

Table 1: Results of various methods on this data set as reported in [9].

for learning probability distributions. Support vector machines have a good performance, since they are meant to be used as a classifier. Distortion is a method to 'enlarge' the data set by adding small variations (translation, scaling, etc) of existing data samples. This could also be done for linear response, probably resulting in a better classification. Table 3 shows the confusion matrix for the digit recognition problem. A lot of digits were wrongly classified as an 'eight'. Digits like zero, one, four and six where easy to classify.

An important advantage of linear response is the fast training procedure. On a pentium computer the total learning phase took only slightly more than half an hour. Since the statistics of the data set are the only need for the algorithm, one sweep through the training set is enough. This is in sharp contrast with, for instance, the 'LeNet' approach as in table 3, where a training time of several days is reported. Moreover, once learned, the classification of a digit is done in a few milliseconds.

	0	1	2	3	4	5	6	7	8	9
0	964	0	3	0	0	2	4	1	6	0
1	0	1117	9	2	1	0	2	0	4	0
2	1	0	990	5	3	0	1	5	27	0
3	0	0	3	971	0	4	0	6	21	5
4	0	0	5	0	967	0	2	0	3	5
5	1	0	2	14	0	849	4	1	20	1
6	5	1	1	0	3	11	933	0	4	0
7	0	6	17	0	5	1	0	971	5	23
8	6	0	6	15	0	4	0	4	935	4
9	1	3	6	12	13	2	0	8	13	951

Table 2: Confusion matrix for the digit recognition. The test set consists of 10,000 samples. Vertical: The presented digit. Horizontal: The prediction of the network. Note that not all digits occur even often in the test set.

4 Discussion

In this paper we applied the linear response learning method to the real world application of digit recognition. One might argue that a Boltzmann machine is particularly useful for modelling probability distributions and that therefore a classification task is not an appropriate problem to assess the quality of linear response learning. We agree with that statement. We chose, however, to solve this problem, since it shows that the method is already quite accurate compared to, for instance, multilayer neural networks, although classification is the goal instead of the probability distribution over digits.

We conclude that linear response is a quite accurate method compared to other neural networks that are not specially designed for the classification of digits. In addition, the method is extremely fast: the total learning phase was done in half an hour for ten networks of 784 neurons.

References

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

[2] C. Peterson and J. Anderson. A mean field theory learning algorithm for neural networks. *Complex systems*, 1:995–1019, 1987.

[3] D. Barber and W. Wiegerinck. Tractable undirected approximations for graphical models. In L. Niklasson, M. Bodén, and T. Ziemke, editors, ICANN *98*, volume 1, pages 93–98, ISBN 3 540 76263 9, 1998. Springer-Verlag.

[4] H.J. Kappen and F.B. Rodríquez. Efficient learning in Boltzmann machines using linear response theory. *Neural Computation*, 10(5):1137–1156, 1998.

[5] S. Kullback. *Information theory and statistics.* Wiley, New York, 1959.

[6] L. Onsager. Electric moments of molecules in liquid. 6 1936.

[7] D.J. Thouless, P.W. Andersson, and R.G. Palmer. Solution of 'solvable model of a spin glass'. *Philisophical Magazine*, 35(3):593–601, 1977.

[8] H.J. Kappen and F.B. Rodríquez. Boltzmann machine learning using mean field theory and linear response correction. In M.I. Jordan, M.J. Kearns, and S.A. Solla, editors, NIPS, 1998.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the* IEEE, 1988.