Boltzmann Machines and the EM algorithm

Piërre van de Laar[1] and Bert Kappen
Laboratory for Medical and Biophysics
University of Nijmegen
Postbus 9101
6500 HB Nijmegen
the Netherlands

Abstract

In this paper we formulate the Expectation Maximization (EM) algorithm for Boltzmann Machines and we prove that the Kullback distance is a Lyaponov function for the EM algorithm. As a result the EM algorithm yields the same solutions as the original learning rule of Ackley, Hinton and Sejnowski. We give an example of the EM algorithm applied to a special class of Boltzmann Machines (BM). This class of BM's includes feedforward networks, radial basis networks and unsupervised clustering and probability density estimation networks. For this Boltzmann Machine the EM algorithm gives a significant speed up compared to standard methods such as (conjugate) gradient descent.

[1] email: pierre@mbfys.kun.nl

1 Introduction

Boltzmann Machines are an interesting class of Neural Networks, because they have explicit parallelism of neuron and weight dynamics. They describe a general class of networks of which feedforward networks and clustering networks are special cases. However, their practical use has been limited because of the excessive time required for training. Learning can be accelerated using a mean field approximation (Peterson and Hartman, 1989), this works best in the context of optimization, but has limited applicability in the context of probability estimation. Recently, one of us (Kappen, 1994) has described a large class of BM for which closed form expressions for the learning rules can be obtained. As a result, no time-consuming Glauber dynamics is required. This class of BM's includes multilayered feedforward networks, radial basis networks and unsupervised clustering and probability density estimation networks. The execution time of these learning rules is fast enough to use BM's for practical applications. In this paper we go one step further. We investigate the applicability of the EM algorithm for Boltzmann Machines.

After formulating the EM algorithm for Boltzmann Machines, we prove in Section 2 that the Kullback distance is a Lyaponov function for the EM algorithm. As a result the EM algorithm yields the same solutions as the original learning rule of Ackley, Hinton and Sejnowski (Ackley et al., 1985). In Section 2.1 we briefly indicate how the `em` algorithm[2] defined by Amari, Kurata and Nagaoka (Amari et al., 1992) is related to the EM algorithm. And in Section 3 we give an example of the EM algorithm applied to a special class of Boltzmann Machines (Kappen, 1993). This class of BM includes feedforward, radial basis networks and unsupervised clustering and probability densities estimation networks.

2 The EM Algorithm

When a Boltzmann Machine is trained, the observed data $\vec{x}_V$, with probability $q(\vec{x}_V)$, is clamped on the visible units and the weights are adjusted to make the best approximation of the observed data. Let $\vec{x}$ and $\vec{x}_H$ be the total and hidden activity of the Boltzmann Machine respectively.

Let $p(\vec{x}|\vec{w})$ be a probability distribution depending on parameters $\vec{w} \in \Omega$, the parameter space. The probability distribution on the visible units is equal to:

$$p(\vec{x}_V|\vec{w}) = \sum_{\vec{x}_H} p(\vec{x}|\vec{w})$$

Where the sum becomes an integral for continue-valued hidden neurons.

The EM algorithm (Boyles, 1983; Dempster et al., 1977; Little and Rubin, 1987; Redner and Walker, 1984; Wu, 1983) starts by defining the function Q:

$$Q(\vec{w}|\vec{w}') = \sum_{\vec{x}} q(\vec{x}_V) p(\vec{x}_H|\vec{x}_V\,\vec{w}') \log\left(p(\vec{x}|\vec{w})\right)$$

where the summation is over all visible and hidden states.

The EM algorithm is an iterative algorithm consisting of an E- and an M-step. Let $\vec{w}^t$ denote the present value of $\vec{w}$ and let $\vec{w}^0$ be the randomly chosen starting weights. The EM iteration ($\vec{w}^t \rightarrow \vec{w}^{t+1}$) can now be defined as follows:
- The expectation step is equal to calculating $Q(\vec{w}|\vec{w}^t)$.

[2] The `e` stands for `e`-geodesic projection and the `m` stands for `m`-geodesic projection in the `em` algorithm defined by Amari, Kurata and Nagaoka

- The maximalization step is to choose $\vec{w}^{t+1}$ to be a value of $\vec{w} \in \Omega$ which maximizes $Q(\vec{w}|\vec{w}^t)$ with respect to $\vec{w}$:

$$\vec{w}^{t+1} = \underset{\vec{w} \in \Omega}{\arg\max}\ Q(\vec{w}|\vec{w}^t) \tag{1}$$

The EM algorithm converges to a maximum value of

$$L(\vec{w}) = \sum_{\vec{x}_V} q(\vec{x}_V) \log\left(p(\vec{x}_V|\vec{w})\right)$$

or

$$F(\tilde{p}(\vec{x}_H), \vec{w}) = \sum_{\vec{x}} q(\vec{x}_V)\tilde{p}(\vec{x}_H) \left(\log\left(p(\vec{x}|\vec{w}) - \log\left(\tilde{p}(\vec{x}_H)\right)\right)\right)$$

See (Dempster et al., 1977; Neal and Hinton, 1994)

When the probability, defined by the Boltzmann Machine, is continuous in $\vec{w}$, then all the limit points of any instance $\{\vec{w}^t\}$ of an EM algorithm are stationary points of L and $L(\vec{w}^t)$ converges monotonically to $L^* = L(\vec{w}^*)$ for some stationary point $\vec{w}^*$. See Theorem 2 of Wu (1983).

Note that

$$L(\vec{w}) = -d(p, q) + C(q)$$

with

$$C(q) = \sum_{\vec{x}_V} q(\vec{x}_V) \log\left(q(\vec{x}_V)\right)$$

and

$$d(p, q) = \sum_{\vec{x}_V} q(\vec{x}_V) \log\left(\frac{q(\vec{x}_V)}{p(\vec{x}_V|\vec{w})}\right)$$

where $d(p, q)$ is the Kullback distance between p and q. So the EM algorithm maximizes $L(\vec{w})$ and at the same time minimizes the Kullback distance. As a result the EM algorithm will converge to the same set of solutions as the standard Boltzmann Machine learning rule. (See also (Hinton and Sejnowski, 1986))

2.1 The `em` algorithm

Let $S = \{q(\vec{x})\}$ be the set of all the probability distributions over the state space X, consisting of 2^n states $\vec{x}$. Let $B = \{p(\vec{x}|\vec{w})\}$ be the set of all probability distributions realizable by a Boltzmann Machine. Given $q(\vec{x}_V)$, let E_q be the set of probability distributions in S, that have the same marginal distribution $q(\vec{x}_V)$ on the visible neurons, that is,

$$E_q = \left\{ q(\vec{x}) \,\middle|\, \sum_{\vec{x}_H} q(\vec{x}) = q(\vec{x}_V) \right\}$$

Let $D(p, q)$ be the divergence between p and q defined by

$$D(p, q) = \sum_{\vec{x}} q(\vec{x}) \log\left(\frac{q(\vec{x})}{p(\vec{x}|\vec{w})}\right)$$

Amari, Kurata and Nagaoka (1992) (See also (Csiszár and Tusnády, 1984; Byrne, 1992)) derived, using information geometry, the following learning algorithm

for Boltzmann Machines. The optimal solution for learning a given marginal distribution $q(\vec{x}_V)$ with a BM is in the framework of Amari, Kurata and Nagaoka (1992)

$$p_{\text{opt}} = \begin{array}{c}\text{argmin} \\ p \in B\end{array} \quad \begin{array}{c}\min \\ q \in E_q\end{array} \; D(p,q)$$

Note that this is a dual optimization, where both $p \in B$ and $q \in E_q$ must be found. The `em` algorithm consists of a sequence of orthogonal projections onto E_q and B alternatively. Specifially Amari, Kurata and Nagaoka (1992) derived the following iterative `em` algorithm for training a Boltzmann Machine to a given $q(\vec{x}_V)$. For $t =$ 1,2,...

$$\text{- } \mathtt{e}\text{-step : Put } q^t(\vec{x}) = \prod_{E_q} p(\vec{x}|\vec{w}^t) = q(\vec{x}_V)p(\vec{x}_H|\vec{x}_V\,\vec{w}^t) \tag{2}$$

$$\text{- } \mathtt{m}\text{-step : Put } p(\vec{x}|\vec{w}^{t+1}) = \prod_{B} q^t(\vec{x}) = \begin{array}{c}\arg\min \\ p \in B\end{array} \; D(q^t,p)$$

where $\vec{w}^t$ and q^t are the weights of the Boltzmann Machine and the 'given' probabilty $q(\vec{x})$ in iteration step t, respectively, and where $\prod_X$ denotes an orthogonal projection onto X. The last equality in Equation (2) is true in most cases. However, counter examples can be constructed (Amari, 1994).

It can be proven (Amari et al., 1992) that the monotonic relation

$$D[q_{t+1}, p_{t+1}] \leq D[q_t, p_t]$$

holds, where the equality holds only for the fixed points of the projections.

Substitution of the `e`-step into the `m`-step gives

$$\begin{aligned}
\vec{w}^{t+1} &= \begin{array}{c}\arg\max \\ \vec{w} \in \Omega\end{array} \; \sum_{\vec{x}} q(\vec{x}_V)p(\vec{x}_H|\vec{x}_V\,\vec{w}^t) \log\left(\frac{p(\vec{x}|\vec{w})}{q(\vec{x}_V)p(\vec{x}_H|\vec{x}_V\,\vec{w}^t)}\right) \\
&= \begin{array}{c}\arg\max \\ \vec{w} \in \Omega\end{array} \; Q(\vec{w}|\vec{w}^t) + f(\vec{w}^t) \\
&= \begin{array}{c}\arg\max \\ \vec{w} \in \Omega\end{array} \; Q(\vec{w}|\vec{w}^t)
\end{aligned}$$

Where $f(\vec{w}^t)$ is independent of $\vec{w}$. Thus the `em` algorithm is identical to the EM algorithm (see Equation (1)).

3 An example of the EM algorithm applied to Boltzmann Machines

Kappen (Kappen, 1994) proposed a Boltzmann Machine with the architecture as given in Figure 1. Between the units $\vec{s}$ are inhibitory connections, such that the total activity in the hidden layer is 1. This allows to formulate efficient expressions for the learning rules of this network. For this architecture the probability for state $(\vec{x}\vec{y}\vec{s_j})$ is

$$p(\vec{x}\vec{y}\vec{s_j}) = \frac{1}{Z}\exp\left(-\beta||\vec{w_j} - \vec{x}||^2 + \sum_{k=0}^{m} v_{jk}y_k\right) \tag{3}$$

with

$$Z = \left(\frac{\pi}{\beta}\right)^{\frac{n}{2}} \sum_{j=1}^{h} \exp\left(v_{j0}\right) \prod_{k=1}^{m} 2\cosh\left(v_{jk}\right)$$

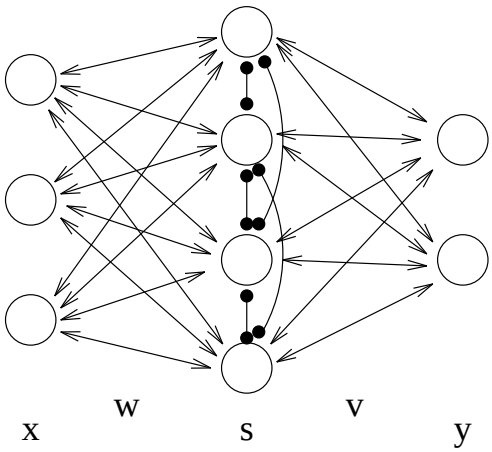

Figure 1: The architecture of a Radial Basis Boltzmann Machine.
Some visible units are continuous neurons ($\forall_{i\in[1,n]} : x_i \in \Re$). Some visible units are two-valued ($\forall_{k\in[1,m]} : y_k = \pm 1$). There is a threshold neuron $y_0 \equiv 1$. The hidden units are two-valued ($\forall_{j\in[1,h]} : s_j = 0$ or 1). Between the units $\vec{s}$ are inhibitory connections, such that the total activity in the hidden layer is 1. Thus the hidden state can be denoted as $\vec{s_j}$ with $(\vec{s_j})_{j'} = \delta_{jj'}$. There are adaptive connections Between the units $\vec{x}$ and $\vec{s}$ are adaptive connections. Between the units $\vec{s}$ and $\vec{y}$ are adaptive connections. The local field contribution from $\vec{x}$ to $\vec{s_j}$ is equal to $-\beta||\vec{w_j} - \vec{x}||^2$.

The EM algorithm defines

$$Q(\vec{w}\vec{v}|\vec{w}'\vec{v}') = \sum_{\vec{x}\vec{y}\vec{s_j}} q(\vec{x}\vec{y}) p(\vec{s_j}|\vec{x}\vec{y}\vec{w}'\vec{v}') \log\left(p(\vec{x}\vec{y}\vec{s_j}|\vec{w}\vec{v})\right)$$

Old and new weights are indicated with and without prime, respectively. In order to maximize, we compute

$$\frac{\partial Q(\vec{w}\vec{v}|\vec{w}'\vec{v}')}{\partial \alpha} = 0 \tag{4}$$

with $\alpha \in \{\vec{w}, \vec{v}\}$ and solve this equation for the new weights in terms of the old weights. The results are (see the Appendix):

$$w_{ij} = \frac{<x_i \vec{s_j}>'_{\text{clamped}}}{<\vec{s_j}>'_{\text{clamped}}} \tag{5}$$

$$v_{jk} = \operatorname{arctanh}\left(\frac{<y_k \vec{s_j}>'_{\text{clamped}}}{<\vec{s_j}>'_{\text{clamped}}}\right) \tag{6}$$

$$v_{j0} = \log\left(\frac{<\vec{s_j}>'_{\text{clamped}}}{\prod_{k=1}^{m} 2\cosh\left(v_{jk}\right)}\right) \tag{7}$$

where

$$<f(\vec{x}\vec{y})\vec{s_j}>'_{\text{clamped}} = \sum_{\vec{x}\vec{y}} q(\vec{x}\vec{y}) f(\vec{x}\vec{y}) p(\vec{s_j}|\vec{x}\vec{y}\vec{w}'\vec{v}')$$

Table 1: CPU time to find solution

method	**time** (in sec)
gradient descent (with momentum)	17342.0
conjugate gradient descent	1040.8
EM algorithm	229.2

which can be simply computed from the training set.

The fact that for this architecture the EM algorithm leads to explicit expressions of new weights in terms of old weights is fortunate, and results in an extremely fast algorithm. In Table 1 we compare the time to find the solution between this EM algorithm, gradient descent and conjugate gradient descent.

The Boltzmann Machine had to learn a perfectly symmetric problem consisting of 240 datapoints. The solution had to be found for $\beta \in [0.08, 100.0]$ for 100 different values of β.

All methods found the same solutions, only in the calculating time there was a difference.

4 Discussion

We have shown how the EM algorithm can be applied to Boltzmann Machines. We have applied the EM algorithm to a subclass of Boltzmann Machines, containing Radial Basis networks and clustering algorithms as special cases. We have shown the superiority of the EM algorithm to other learning algorithms for this class of Boltzmann Machines, with some computer simulations.

In principle, the EM algorithm can be applied to any type of Boltzmann Machine architecture. Whether similar speed up will result as in the above case depends on whether the Equation (4) can be explicitly solved.

5 Appendix

We want the EM algorithm to give us an iterative algorithm, which expresses the new weigths as a function of the old weights. First we calculate the fixed-point equations of the EM algorithm, then we try to manipulate the found formulas to get the desired form.

The derivatives of Q for the Boltzmann Machine proposed by Kappen (Kappen, 1993) are

$$\frac{\partial Q(\vec{w}\vec{v}|\vec{w}'\vec{v}')}{\partial \alpha} = \sum_{\vec{x}\vec{y}\vec{s_{j'}}} \frac{q(\vec{x}\vec{y})p(\vec{s_{j'}}|\vec{x}\vec{y}\vec{w}'\vec{v}')}{p(\vec{x}\vec{y}\vec{s_{j'}}|\vec{w}\vec{v})} \frac{\partial p(\vec{x}\vec{y}\vec{s_{j'}}|\vec{w}\vec{v})}{\partial \alpha} = 0$$

which gives using Equation (3)

$$\begin{aligned} \frac{\partial Q(\vec{w}\vec{v}|\vec{w}'\vec{v}')}{\partial w_{ij}} &= -\sum_{\vec{x}\vec{y}} q(\vec{x}\vec{y})p(\vec{s_j}|\vec{x}\vec{y}\vec{w}'\vec{v}')2\beta(w_{ij} - x_i) \\ &= 2\beta \left(<x_i \vec{s_j}>'_{\text{clamped}} - w_{ij} <\vec{s_j}>'_{\text{clamped}} \right) = 0 \end{aligned} \tag{8}$$

$$\begin{aligned} \frac{\partial Q(\vec{w}\vec{v}|\vec{w}'\vec{v}')}{\partial v_{jk}} &= \sum_{\vec{x}\vec{y}} q(\vec{x}\vec{y}) \left(y_k p(\vec{s_j}|\vec{x}\vec{y}\vec{w}'\vec{v}') - \tanh(v_{jk}) p(\vec{s_j}|\vec{w}\vec{v}) \right) \\ &= <y_k \vec{s_j}>'_{\text{clamped}} - \tanh(v_{jk}) p(\vec{s_j}|\vec{w}\vec{v}) = 0 \end{aligned} \tag{9}$$

and

$$\frac{\partial Q(\vec{w}\vec{v}|\vec{w}'\vec{v}')}{\partial v_{j0}} = \sum_{\vec{x}\vec{y}} q(\vec{x}\vec{y})\left(p(\vec{s_j}|\vec{x}\vec{y}\vec{w}'\vec{v}') - p(\vec{s_j}|\vec{w}\vec{v})\right)$$
$$= <\vec{s_j}>'_{\text{clamped}} - p(\vec{s_j}|\vec{w}\vec{v}) = 0 \qquad (10)$$

Equation (8) yields

$$w_{ij} = \frac{<x_i \vec{s_j}>'_{\text{clamped}}}{<\vec{s_j}>'_{\text{clamped}}}$$

Combination of Equations (9) and (10), leads to

$$v_{jk} = \text{arctanh}\left(\frac{<y_k \vec{s_j}>'_{\text{clamped}}}{<\vec{s_j}>'_{\text{clamped}}}\right)$$

Using

$$p(\vec{s_j}|\vec{w}\vec{v}) = \frac{\exp(v_{j0})\prod_{k=1}^{m} 2\cosh(v_{jk})}{\sum_{j'=1}^{h}\exp(v_{j'0})\prod_{k'=1}^{m} 2\cosh(v_{j'k'})}$$

and

$$\sum_{j=1}^{h} <\vec{s_j}>'_{\text{clamped}} = 1$$

we find that Equation (10) is satisfied when

$$\exp(v_{j0})\prod_{k=1}^{m} 2\cosh(v_{jk}) = \exp(-\tilde{\lambda})<\vec{s_j}>'_{\text{clamped}}$$

where $\tilde{\lambda}$ is the same for all j. So

$$v_{j0} + \tilde{\lambda} = \log\left(\frac{<\vec{s_j}>'_{\text{clamped}}}{\prod_{k=1}^{m} 2\cosh(v_{jk})}\right)$$

where $\tilde{\lambda} = 0$ may be chosen, because of the invariance of this Boltzmann distribution (see Equation (3)) under variation of $\tilde{\lambda}$. Note that v_{jk} is already known as a function of $\vec{w}'$ and $\vec{v}'$.

Acknowledgements

This work is sponsored in part by the Dutch Foundation for Neural Networks (SNN) and the Japanese Real World Computing Program. We would like to thank the NIPS'94 referees for there useful remarks and references.

References

Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169.

Amari, S.-i. (1994). Information geometry of the EM and em algorithms for neural networks. Technical report, Department of Mathematical Engineering and Instrumentation Physics, University of Tokyo.

Amari, S.-i., Kurata, K., and Nagaoka, H. (1992). Information geometry of Boltzmann machines. *IEEE Transactions on Neural Networks*, 3(2):260–271.

Boyles, R. A. (1983). On the convergence of the EM algorithm. *Journal of the Royal Statistical Society*, 45(1):47–50.

Byrne, W. (1992). Alternating minimization and Boltzmann machine learning. *IEEE Transactions on Neural Networks*, 3(4):612–620.

Csiszár, I. and Tusnády, G. (1984). Information geometry and alternating minimization procedures. *Statistics & Decisions, Supplement Issue*, 1:205–237.

Dempster, A. P., Laird, N. M., and Rubin, D. R. (1977). Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society*, 39(1):1–38.

Hinton, G. E. and Sejnowski, T. J. (1986). Learning and relearning in Boltzmann machines. In Rumelhart, D. E. and McClelland, J. L., editors, *Parallel Distributed Processing : explorations in the microstructure of cognition*, chapter 7, pages 282–317. MIT Press, Cambridge.

Kappen, H. J. (1993). Using Boltzmann machines for probability estimation. In Gielen, C. and Kappen, H., editors, *Proceedings of the International Conference on Artificial Neural Networks 1993*, pages 521–526. Springer-Verlag.

Kappen, H. J. (1994). Using Boltzmann machines for probability estimation. In Gelsema, E. and Kanal, L., editors, *Proceedings Pattern Recognition in Practice IV*. Elsevier. Accepted.

Little, R. J. A. and Rubin, D. B. (1987). *Statistical Analysis with Missing Data*. Wiley series in probability and mathematical statistics. John Wiley & Sons, New York.

Neal, R. M. and Hinton, G. E. (1994). A new view of the EM algorithm that justifies incremental and other variants. Submitted to: Biometrika.

Peterson, C. and Hartman, E. (1989). Explorations of the mean field theory learning algorithm. *Neural Networks*, 2(6):475–494.

Redner, R. A. and Walker, H. F. (1984). Mixture densities, maximum likelihood and the EM algorithm. *Society for Industrial and Applied Mathematics Review*, 26(2):195–239.

Wu, C. F. J. (1983). On the convergence properties of the EM algorithm. *The Annals of Statistics*, 11(1):95–103.