ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 109-115

A Stop Criterion for the Boltzmann Machine Learning Algorithm

Berthold Ruf
School of Computer Science, Carleton University, Ottawa, K1S 5B6, Canada,
and Rheinisch-Westfälische Technische Hochschule, Aachen, Germany

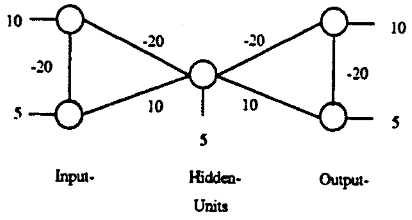
Abstract: Ackley, Hinton and Sejnowski introduced a very interesting and versatile learning algorithm for the Boltzmann machine (BM). However it is difficult to decide when to stop the learning procedure. Experiments have shown that the BM may destroy previously achieved results when the learning process is executed for too long. This paper introduces a new quantity, the conditional divergence, measuring the learning success for the inputs of the data set. To demonstrate its use, some experiments are presented, based on the Encoder Problem.

1 Introduction

The Boltzmann machine (BM), introduced by Ackley, Hinton and Sejnowski in [Ack 84] is one of the most interesting neural networks. This paper first summarizes the basic concepts of the BM and gives in chapter 2 a short description of the learning algorithm, which was also introduced in [Ack 84]. Chapter 3 analyzes the convergence behavior of the algorithm and introduces a new quantity which makes it possible to decide when to stop the learning process.
Let $U = \{u_1, \ldots, u_n\}$ denote the set of units of the BM which are considered to be binary. Thus one can specify a mapping $k : \{u_1, \ldots, u_n\} \rightarrow \{0, 1\}$ which is called a *configuration*. The space of all possible configurations is denoted by $\Re$, clearly $|\Re| = 2^n$. A weight w_{ij} is assigned to the edge between units u_i and u_j (note that $w_{ij} = w_{ji}$). The loop of unit u_i has the weight w_{ii}. An edge is said to be *activated* if the two corresponding units have the value '1'. The *energy* E is defined by:

$$E(k) := -\sum_{i=1}^{n} \sum_{j=1}^{i} w_{ij} k(u_i) k(u_j) \qquad (1)$$

The task of a BM is to find a configuration of minimal energy which is done by the simulated annealing algorithm (see for example [Aar 89]).
Aarts and Korst have shown in [Aar 89] that the BM converges for a fixed temperature T to a stationary probability distribution $q_k(T)$ over the space $\Re$ of all possible configurations as time (i.e. the number of suggested state



k	Input	Output	Hidden	Energy
k_1	10	10	0	-20
k_2	01	01	1	-35
k_3	10	10	1	15
k_4	01	01	0	-10
k_5	11	11	0	10
k_6	11	11	1	25
k_7	00	00	0	0
k_8	00	00	1	-5
k_9	10	01	0	-15
k_{10}	10	01	1	-10

Figure 1: Solution to the 2-1-2 encoder problem

transitions) goes towards infinity. If the BM chooses configurations due to this distribution it is said to be in *equilibrium*. $q_k(T)$ converges for $T \rightarrow 0$ to the uniform distribution over the set of configurations with minimal energy (see again [Aar 89]).

For realizing classification problems on the BM it is necessary to split the set of units U into three disjoint subsets U_I, U_O and U_H for the input, output and hidden units (note that $U_H = \emptyset$ is possible). An inquiry to the BM is made by clamping the input units and executing the simulated annealing algorithm only for the hidden and the output units. The classification problem for a given BM can be described by an *environment* $\mathcal{V} \subset \Re_{IO}$ where $\Re_{IO}$ is the space of all possible input/output configurations.

Example 1.1 The task of the n-k-n encoder problem, introduced in [Ack 84] is simply the following: there are n input and n output units. If the ith input unit is on and all other inputs are off, the same should happen to the outputs. This problem is trivial, if the inputs are directly connected to the outputs. But with a 'bottleneck' of $k < n$ hidden units and edges only from the inputs to the hidden units to the output units, it is interesting to see if the learning algorithm can find a solution, since one can easily show that there exists a solution with $k = \lceil \log n \rceil$. Figure 1 shows a possible solution for the 2-1-2 encoder problem. Fixing 10 (01) at the inputs results in k_1 (k_2) as the global minimum for the energy. If no units are clamped then k_2 is the global minimum.

2 The Learning Algorithm

The task of a learning algorithm is to find a proper weight constellation for realizing a desired input/output behavior specified by a given environment $\mathcal{V}$. This chapter summarizes the learning algorithm introduced in [Ack 84] using a slightly modified syntax.

The algorithm generally distinguishes two phases: In the *plus phase* the input/output units are clamped according to a given probability distribution $q^+_{k_{IO}}$ over the input/output configurations for the environment $\mathcal{V}$. Usually all elements of $\mathcal{V}$ are chosen with equal probability $1/|\mathcal{V}|$.[1] In the *minus phase* all units may change their state and one considers the probability distribution q_k of the BM reached by simulated annealing.[2] Since only the input/output configurations k_{IO} are of interest, the probabilities of configurations with same k_{IO} but different hidden unit configuration k_H have to be summed up. Thus $q^-_{k_{IO}} := \sum_{k_H \in \Re_H} q_{k_{IO,H}}$ where $\Re_H$ denotes the set of all possible hidden unit configurations and $k_{IO,H}$ the configuration with input/output-configuration k_{IO} and hidden unit configuration k_H.
It is now possible to measure the distance between these two distributions by a well known quantity from information theory:

Definition 2.1 *Given an environment $\mathcal{V}$ and the two distributions q^+ and q^- over the input/output configurations. The divergence G is defined as*

$$G := \sum_{k_{IO} \in \Re_{IO}} q^+_{k_{IO}} \ln \frac{q^+_{k_{IO}}}{q^-_{k_{IO}}} \tag{2}$$

G equals zero iff $q^+_{k_{IO}} = q^-_{k_{IO}}$ for all $k_{IO} \in \Re_{IO}$ and is greater than zero otherwise. For a proof see [Aar 89]. Thus the aim of the learning algorithm is the minimization of the divergence. Since $q^-_{k_{IO}}$ only depends on the weights (seen away from T) and $q^+_{k_{IO}}$ is given by the environment, G can be minimized by weight modifications. Let p^+_{ij} denote the activation probability of the edge between unit u_i and u_j when the input/output units are clamped by the environment. p^-_{ij} is the corresponding probability for the case that no units are clamped. In [Ack 84] it is shown that

$$\frac{\partial G}{\partial w_{ij}} = -\frac{1}{T}(p^+_{ij} - p^-_{ij}) \tag{3}$$

G can be decremented by moving in the opposite direction of the gradient vector. This results in the following learning rule for modifying the weights:

$$\Delta w_{ij} = \sigma(p^+_{ij} - p^-_{ij}) \tag{4}$$

where $\sigma > 0$ specifies the amount of weight change. The three steps of plus and minus phases for estimating p^+_{ij} and p^-_{ij} and then changing the weights by (4) are called a *sweep* and executed repeatedly by the algorithm.
When applying the learning algorithm, it is in many cases better to change the

[1] Since it is not possible for the BM to realize that some configurations have probability zero, one adds some noise to the clamped elements from $\mathcal{V}$.

[2] T is assumed to be arbitrary but fixed, thus the dependency of q from the temperature is not considered anymore.

weights by a fixed constant:

$$\Delta w_{ij} = \begin{cases} \beta & \text{if } p_{ij}^+ - p_{ij}^- > 0 \\ 0 & \text{if } p_{ij}^+ - p_{ij}^- = 0 \\ -\beta & \text{if } p_{ij}^+ - p_{ij}^- < 0 \end{cases} \qquad (5)$$

For a motivation of this rule see [Ack 84], [Der 84].

3 The Conditional Divergence

For several reasons it is important to recognize as early as possible when the learning algorithm has converged; i.e. it can make the correct input/output associations: Since the algorithm is very time demanding, it should not execute more sweeps than necessary. It can also happen that the algorithm, once having converged, can again destroy its achieved learning results by "overlearning". This phenomenon can be caused for example by big weights leading to a poor performance of simulated annealing in the plus and minus phase of the algorithm.

The learning algorithm uses several parameters. There is not much known about how to choose them. Usually this is done by trial and error. In order to make systematic experiments about the influence of the parameters on the learning success it is necessary to have a stop criterion as precise as possible. In most cases one simply uses the following stop criterion: after every sweep all input patterns are clamped at the input units and after executing simulated annealing the output units are compared with the desired output. Due to the stochastic nature of simulated annealing this has to be done several times. ([Ack 84],[Low 89],[Pet 87])

In [Aar 89] the following stop criterion is suggested: every edge tests after every sweep, if for a given $\varepsilon > 0$: $|p_{ij}^+ - p_{ij}^-| < \varepsilon$. If this condition is satisfied for some succeeding sweeps, then the corresponding weight will be no longer changed. The algorithm is stopped when all weights have stabilized. The advantage of this stop criterion is that every edge can *locally* decide for how long its activation probabilities have to be estimated. However several experiments have shown that it may happen that despite of a learning success not all edges stabilize, some edges can show an oscillating behavior.

It seems reasonable to use G as a stop criterion. But due to the following reason G is often *not* minimized: for minimizing G it is necessary that q^- converges to the distribution q^+ of the patterns to be learned. Usually the BM is not able to reach this, although it is able to make the correct input/output associations. Example 1.1 showed a good solution for the 2-1-2 encoder. However $G \approx 6.813$ since $q_{k_1} \approx 3 \cdot 10^{-7}$ and $q_{k_2} \approx 1.0$.

This motivates weakening the demand that $G = 0$ and introducing a new quantity. It should be sufficient to consider in definition 2.1 instead of q^- the probability of an output configuration in equilibrium when clamping a pattern of the environment at the inputs. In doing so, the probability of this pattern within the

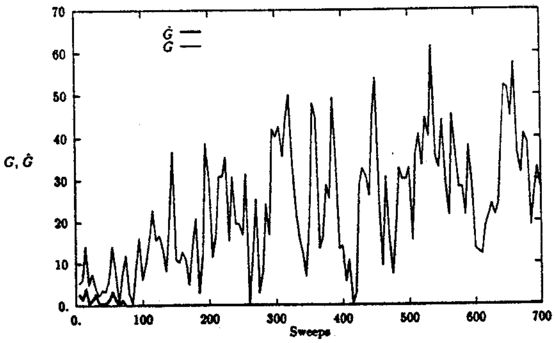


Figure 2: Comparison of G and $\hat{G}$

environment also has to be taken into consideration. This leads to the following definition.

Definition 3.1 *Let $\mathcal{P}$ be the set of input patterns from the environment $\mathcal{V}$, k_I (k_O) an input (output) configuration and $\Re_O$ the set of all possible output configurations. The conditional divergence is then defined as*

$$\hat{G} := \sum_{k_I \in \mathcal{P}} \sum_{k_O \in \Re_O} q_{k_{IO}^+} \ln \frac{q_{k_{IO}}^+}{q_{k_O|k_I}^- q_{k_I}^+} \tag{6}$$

where $q_{k_O|k_I}^-$ denotes the conditional probability for getting k_O in equilibrium at the output units when k_I is fixed at the input units.

In example 1.1 $q_{k_O|k_I}^-$ equals for k_1 and k_2 0.992 respectively 0.999. Since the two input configurations for k_1 and k_2 in $\mathcal{V}$ have equal probabilities of 0.5 one finally gets $\hat{G} \approx 0.0003$.

Example 3.1 During the learning process for the 3-2-3 encoder G and $\hat{G}$ were calculated after every fifth sweep, which is shown in figure 2.[3] As one can see, G is not minimized here. After 70 sweeps $\hat{G}$ is constantly 0. Additional tests have shown that after 70 sweeps the BM is for the first time able to answer all questions correctly.
The big changes of G can be explained as follows: as mentioned above it is very difficult for the BM to reach a uniform distribution among the patterns of $\mathcal{V}$. Thus p^+ often differs from p^- and the weights are changed resulting in changes in G.
Figure 3 shows for the same environment that learning too long may result in a degradation of the performance. One can see that the stabilization, which is reached after about 70 sweeps is again destroyed after approximately 450 sweeps.

[3] The parameters were chosen simular to the ones described in [Ack 84] for the 4-2-4 encoder.

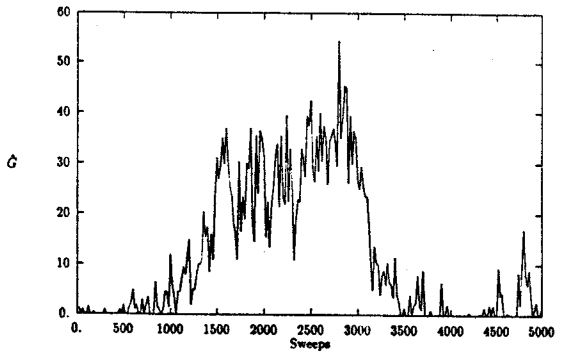


Figure 3: Learning for a long period of time

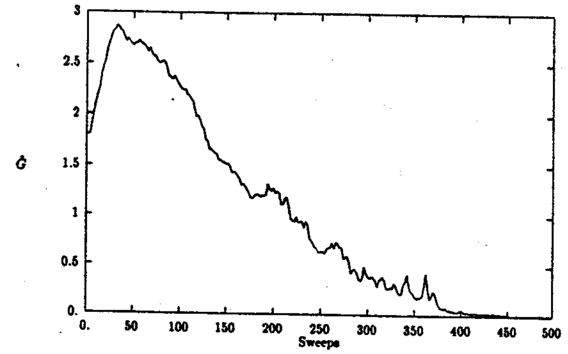


Figure 4: $\hat{G}$ with proportional weight changes

It is obvious how to use $\hat{G}$ as a stop criterion: after every ith sweep $\hat{G} \approx 0$ is tested.[4] If the weights are changed by equation (5), then up to the learning success the weights will change from sweep to sweep considerably. After the algorithm has converged the *conditional* probabilities of the output configurations will not change anymore.

Problems may arise if the weights are changed proportionally by rule (4). For a safe convergence σ should be chosen small, resulting in small changes of the weights and thus of $\hat{G}$. Figure 4 shows for the 3-2-3 encoder why it is difficult to use $\hat{G}$ as a stop criterion here. It should be finally mentioned that the big computational effort for calculating $\hat{G}$, which is exponential in the number of units can be easily reduced by computing the terms of the sum in equation (6) in parallel.

[4] During the calculation of $\hat{G}$ the use of large numbers may be necessary, so due to numerical errors a test for $G = 0$ does not make sense.

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 109-115

References

[Aar 89] Aarts, E. H. L., Korst, J.: *Simulated annealing and Boltzmann machines.* John Wiley & Sons Ltd, 1989.

[Ack 84] Ackley, A., Hinton, E., Sejnowski, T.: *Boltzmann machines: Constraint satisfaction networks that learn.* Technical Report CMU-CS-84-119, Carnegie-Mellon University, Pittsburgh, 1984.

[Ack 85] Ackley, A., Hinton, E., Sejnowski, T.: *A learning algorithm for Boltzmann machines.* Cognitive Science 9, 1985, p. 147-169.

[Der 84] Derthick, M.: *Variations on the Boltzmann machine learning algorithm.* Technical Report CMU-CS-84-120, Carnegie-Mellon University, Pittsburgh, 1984.

[Hin 86] Hinton, G., Sejnowski, T.: *Learning and Relearning in Boltzmann Machines.* from: Rumelhard, David E.: Parallel distributed processing. Volume 1, MIT Press, 1986.

[Laa 87] Laarhoven, P., Aarts, E.: *Simulated Annealing: Theory and applications.* D. Reidel Publishing, Holland, 1987.

[Low 89] Lowton, A. , Harget, A.: *On the performance of a neural network - the Boltzmann machine.* Proc. of the international symposium of applied informatics, 1989, p. 63-66.

[Pet 87] Peterson, C., Anderson, J. R.: *A mean field theory learning algorithm for neural networks.* Complex Systems I, 1987, p. 995-1019.