que l'accès à des machines spécialisées doit permettre de combler le handicap de mécanismes neuronaux éventuellement plus lents, il est encore plus important de comprendre que l'opportunité de travailler dans le cadre des réseaux de neurones peut induire de nouvelles idées de processus d'optimisation. C'est ce que nous avons essayé de démontrer.

Enfin la notion même d'optimisation pour les réseaux de neurones n'a pas nécessairement la même connotation que celle que l'on entend en mathématique appliquée. Les problèmes d'optimisation sont rangés en classes de complexité. Les problèmes faciles au sens de la complexité se résolvent en un nombre d'étapes qui croit polynomialement avec la taille du problème, les problèmes difficiles sont ceux dont la solution rigoureuse demande un nombre exponentiel d'étapes. Les réseaux de neurones sont des systèmes pour lesquels on n'est pas trop exigeant quant à la qualité de la solution. Pour sortir d'une pièce par exemple il est nécessaire de passer entre les chambranles de la porte et non de trouver la meilleure trajectoire. Parmi les problèmes difficiles il existe une sous-classe de problèmes (dits problèmes NP) pour lesquels on peut décider en un temps polynomial s'il existe une solution de qualité donnée même si la solution rigoureuse nécessite un nombre exponentiel d'étapes. Si donc on se contente de solutions approchées la distinction entre problèmes faciles et problèmes NP disparait et il est vrai que la difficulté pour un réseau de neurones de résoudre le problème du voyageur de commerce qui est un problème de type NP est exactement la même que celle de résoudre le problème du plus court chemin qui est un problème polynomial.

Références :

(Chen 90)
Chen K. : "A simple learning algorithm for the traveling salesman problem". Preprint. Brookhaven National Laboratory (N.Y.) 1989.

(Durbin 87)
Durbin,R. & Willshaw,D. : Nature, **326**, 1987, 689-691.

(Hopfield 82)
Hopfield,J.J. : Proceedings of the National Academy of Sciences (USA), **79**, 1982, 2554-2558.

(Hopfield 86)
Hopfield,J.J. & Tank,D.W. : Science, **233**, 1986, 625-633.

(Jonhson 84)
Johnson,D.S. & Papadimitriou,C.H. : The traveling salesman problem. Lawler,E.L. et al. eds. Wiley (N.Y.) 1984, Chap.5.

(Kirkpatrick 83)
Kirkpatrick,S.,Gelatt,C.D. & Vecchi,M.P. : Science, **220**, 1983, 671-680.

(Kohonen 84)
Kohonen,T. : Self-organization and Associative Memory. Springer (Berlin), 1984.

(Lin 73)
Lin,S. & Kernighan,B.W. : Operational Research, **21**, 1973, 498-516.

(Peretto 89)
Peretto P. : Artificial Neural Networks. Presses Polytechniques Romandes (Lausanne) 1989, pp. 117-130.

(Rujàn 88)
Rujàn,P. : Zeitschrifft für Physik, **B 73**, 1988, 391-416.

Machines de Boltzmann synchrones et vision artificielle

Synchronous Boltzmann machines and artificial vision

Robert Azencott

Résumé : Les machines de Boltzamnn asynchrones, introduites par Hinton-Sejnovski sont à énergies quadratiques et interactions par paires. On leur reproche d'apprendre lentement. Nous présentons les règles d'apprentissage pour les machines de Boltzmann synchrones (bien plus rapides) avec énergies très générales et interactions par cliques d'ordre arbitraire. Nous esquissons les architectures de réseaux de Boltzmann synchrones dédiés à l'extraction et l'identification de contours lisses sur des images digitalisées.

Mots clés : Machines de Boltzmann, réseaux de neurones stochastiques, vision artificielle, apprentissage synchrone.

Abstract : Asynchronous Boltzamnn machines, introduced by Hinton-Sejnovski, have quadratic energies and pairwise interactions. They are reputedly slow learners. We present *learning rules for the much faster synchronous* Boltzmann machines with general energies and high order clique interactions. We sketch the architecture of synchronous Boltzmann networks dedicated to the extraction and identification of smooth contour lines in digitalized pictures.

Keywords : Boltzmann machines, stochastic neural networks, artificial vision, synchronous learning.

DIAMS-LMENS, Ecole Normale Supérieure
45 rue d'Ulm - 75005 Paris, France
Université Paris-Sud
Laboratoire de Statistiques Appliquées, Orsay, France

1. INTRODUCTION

In the spirit of the asynchronous Boltzmann machines introduced by Hinton and Sejnowski [10], we present here a very general class of stochastic neural networks, where local interactions are non quadratic and involve cliques of neurons of arbitrary order.

We focus our attention on totally synchronous dynamics, as in Azencott [3] [5], since they are of course much faster than the asynchronous ones. We have derived synchronous rigorous learning rules, implementable on a dual pair of networks (neurons and cliques). These rules interestingly involve *delayed* correlations between current score and transitional clique activity. Moreover we impose the use of a non vanishing temperature for these networks since stabilization time grows exponentially at low temperature.

Many low level artificial vision tasks can be modelized by such synchronous Boltzmann modules, borrowing intuition and knowhow from the asynchronous Gibbs field models (D. and S. Geman [8] [9], Azencott [1] [2], Geman-Graffigne [9], Chalmond [7]...). We describe here the architectures of such modules, dedicated to the extraction and identification of smoothed chains of contours. Currently, at DIAM-ENS, we are conducting a team effort to explore these techniques on massively parallel machines (Connection Machine at ETCA, Paris), in collaboration with A. Doutriaux (San Diego), J. Lacaille and L. Younès (DIAM-ENS). Simultaneously, in collaboration with P. Garda (CNRS and IEF Orsay), we seek to evaluate and build specialized hardware dedicated to the implementation of synchronous Boltzmann machines. This work is currently supported by a French research grant (DRET).

2. STOCHASTIC NEURAL NETWORKS WITH HIGH ORDER INTERACTIONS

Let S be a finite set of "formal neurons". The state x_s of neuron s takes values in a finite set A. Call $\Omega = A^S$ the set of *configurations* $x = (x_s)_{s\in S}$ of the network.

Fix a finite family K of subsets of S, called the set of *cliques* in the network. The *activity* of any clique $C \in K$ will be measured by an (arbitrary) *interaction potential* $J_C(x) = J_C(x_C)$ where x_C is the clique configuration. To each clique C is associated a *numerical weight* w_C. Call $w = (w_C)_{C\in K}$ the weight vector of the network.

Let $U(x) = \sum_{C\in K} w_C J_C(x)$ be the weighted activity of the cliques.

The *neighborhood* Ns of neuron s is the set of all $t \in S$ with $t \neq s$ and such that there is some clique $C \in K$ containing both s and t. The *action potential* at site s is given by

$$V_s(x) = -\sum_{C\ni s} w_C J_C(x) = V_s(x_s, x_{Ns}) \qquad (2.1)$$

where the sum extends to all cliques containing s.

Fix a "temperature" $T > 0$, and define the local stochastic update rule for neuron s by

$$P(x_s = b \mid x_{S-s}) = \frac{e^{V_s(b,x_{Ns})/T}}{\sum_{a\in A} e^{V_s(a,x_{Ns})/T}}. \qquad (2.2)$$

If we update only one neuron at a time, with a periodic refreshment of all neurons, the long run asynchronous stochastic equilibrium is the Gibbs probability proportional to $e^{-U(x)/T}$.

However, we focus essentially here on the *much faster synchronous stochastic dynamics*, where *all neurons are simultaneously updated* according to rule (2.1). In this case, the *long run synchronous equilibrium distribution* $Q_T(x)$ exists, but *cannot be computed explicitly in general*, unless all cliques are of cardinal ≤ 2, (*cf.* Azencott [3]).

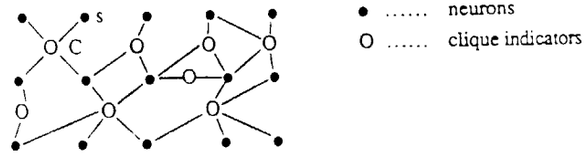
Let us point out that if all cliques are of cardinal 2, with $J_C(x) = J_{\{st\}}(x) = -x_s x_t$ and $x_s, x_t \in \{0,1\}$, we fall back on the more traditional description of a single network S with synaptic weights $w_C = w_{st}$ and quadratic asynchronous energy. But even in this case, the synchronous energy is not quadratic (*cf.* Azencott [3]). Indeed we have shown that for quadratic pairwise interactions $J_{\{s,t\}}(x) = -x_s x_t$, the synchronous equilibrium probability distribution is proportional to $e^{-K_T(x)/T}$, where

$$\begin{cases} K_T(x) = -\sum_s T \log\left[1 + e^{\frac{1}{T} V_s(x)}\right] \\ V_s(x) = \sum_t w_{st} x_t \end{cases}$$

The synchronous dynamics with clique interactions can be locally implemented on a pair (S,K) of dual networks where S is the set of neurons, and K the set of "clique cells". A clique C computes its state y_C by $y_C = J_C(x)$ after reading all neurons in C, and a neuron s computes its action potential $V_s(x)$, after reading all cliques C containing s, by

$$V_s(x) = -\sum_{C\ni s} w_C y_C .$$

Thus the only links in (S,K) lie between pairs (s,C) such that the neuron s belongs to the clique C, as shown in Figure 1.



Example of a small network with cliques of various cardinals

Figure 1

3. SYNCHRONOUS LEARNING RULES

Following the paradigm proposed by Hinton-Sejnowski-Ackley, fix the input units $I \subset S$ and the response units $R \subset S$. Call $\Omega_R = A^R$ the sets of input and output configurations. We want to use the synchronous machine as an adaptative black box to emulate a given mapping $F : \Omega_I \to \Omega_R$.

For a given input $\alpha \in \Omega_I$, the random output $F(\alpha)$ of the machine will be read as the configuration of the response units once synchronous stochastic equilibrium has been reached. We assume that the environment delivers random inputs having a *fixed (unknown) a priori distribution* and has thus provided a representative *training set* of inputs.

To measure the performance of the machine we introduce an (arbitrary) loss function $L(\beta,\gamma)$ on pairs of outputs, verifying $L \geq 0$ and $L(\beta,\beta) \equiv 0$. Then each configuration $x \in \Omega$ has a score

$$\lambda(x) = L\left[F(x_I), x_R\right] \tag{3.1}$$

and the global performance of the machine is measured by its expected score

$$\sigma(w) = \lim_{n\to\infty} E[\lambda(X^n)] \tag{3.2}$$

computed at (synchronous) stochastic equilibrium. Here X^n is the random configuration after the n^{th} synchronous update.

We now seek to optimize the network by selecting clique weights w *minimizing the expected score* $\sigma(w)$. We have computed a theoretical expression for the gradient $(\partial\sigma(w))/(\partial w)$, and interpreted it at the machine level to obtain implementable very general synchronous learning rules, which we now describe (*cf.* Azencott [5] [6]).

For each pair x, y of configurations, define the transition activity $\tau_C(x,y)$ of the clique C by

$$\tau_C(x,y) = \sum_{s\in C} J_C\,(x_{C-s}, y_s)\,. \tag{3.3}$$

We may compute (locally) the expected value of $\tau_C(x,y)$, given x, by

$$\bar{\tau}_C(x) = \sum_{s\in C}\sum_{a\in A} J_C(x_{C-s}, a)\, P(y_s = a \mid x_{S-s})$$

and we call $u_C(x,y) = \tau_C(x,y) - \bar{\tau}_C(x)$ the *centered transition activity of the clique* C.

Fix an integer $k \geq 1$. Then we introduce the k-*delayed learning rule* for which the weight updates Δw_C are given by

$$\Delta w_C = \eta \sum_{i=1}^{k} \mathrm{cor}\left[\lambda(X^n), u_C(X^{n-i}, X^{n-i+1})\right] \tag{3.4}$$

where the small gain η has to decrease slowly with the number of weight updates.

Here n is large enough so that X^n has approximately reached stochastic equilibrium, and correlations can be estimated empirically by time averages. The ideal learning rules involves theoretically large delays k. However in concrete situations, k can be kept small.

For instance in the case when all cliques have cardinal ≤ 2, we have proved that $k = 1$ is the correct delay value. Actually, in the simplest case where interactions are quadratic and involve only pairs of neurons, the synchronous learning rule becomes (Azencott [3]), for a natural choice of the score function

$$\Delta w_{st} = \eta\left[(q_{st} + q_{ts}) - (\hat{q}_{st} + \hat{q}_{ts})\right] \tag{3.5}$$

where q_{st}, $\hat{q}_{st}$ are respectively the empirical frequencies of the event $\{X^n_s = X^{n+1}_t = 1\}$ in the clamped case and the unclamped case. Recall that in the clamped case, the machine has its output units x_R forced to coincide with the correct answer $F(x_I)$.

Extensive testing of the simplest delayed rule (3.5) has been carried out on a Cray 2 and a Connection Machine by J. Lacaille, A. Doutriaux, L. Younès. This learning rule functions quite well at suitable fixed temperatures and all our learning experiments have been satisfactorily completed with a *few hundred passes* of the training sets, after a few trials to adjust the learning parameters. Moreover, so far, it has always been possible to reduce the stabilization time (necessary to reach stochastic equilibrium) to less than 50 synchronous updates in actual classification, and to less than 10 synchronous updates during learning.

During learning, temperatures were either kept fixed or slowly modified to reach an optimal *non zero* limit level. We have derived (and used) a schedule of temperatured updates

$$\Delta T \text{ proportional to } \left[-\frac{1}{T}\sum_{s,t} w_{st}\,\Delta w_{st}\right].$$

4. SYNCHRONOUS BOLTZMANN MACHINES AND ARTIFICIAL VISION

Numerous examples of the use of asynchronous Gibbs fields to model and implement low-level artificial vision tasks have appeared since the initial impetus provided by D. and S. Geman [8]. Part of this expertise can be transfered to the design of high order interaction, synchronous, Boltzmann machines with general energies, specialized to perform low an medium level vision tasks.

4.1. First example : Identification of smooth contour lines in a standard gray-level digitalized picture

The *input layer* [INL 1] of the network is a 16×16 rectangular array of pixels, with an 8bit coding of pixel intensities. The *output layer* [OUTL] describes a finite family of piecewise smooth, continuous contour lines. The first hidden layer [COL 2] has a column of nine $\{0,1\}$ neurons "above" each pixel site s. The neurons θ^k_s $k = \{0,1\ldots,7\}$ code the plausible contour orientations $\{0,\pi/4,\ldots,7\pi/4\}$ at site s and the ninth c_s indicates the presence/absence of contour. Each of the θ^k is totally linked to the 5×5 window centered at s in [INL 1]. The eight neurons $\theta^0_s \ldots \theta^7_s$ are all pairwise connected, and all connected to c_s. Horizontal links are placed between θ^k_s and θ^l_t whenever s and t are immediate neighbours.

Of course all the weights defined so far and below are *translation invariants*.

The layer [LINK 3] provides "link" neurons, λ_{st} placed above the midpoint of each clique pair {s,t} of immediate neighbours in [INL]. Each of the λ_{st} takes the values {-1,0,1} indicating if pixels s and t are linked by a contour line, and its orientation. Each of the λ_{st} is linked to c_s , c_t and all the θ^k_s , θ^k_t $k = 0 \dots 7$ in layer [COL 2]. The layer [LINK 3] is locally analyzed by a layer of cliques [CL 4], which includes one clique C per group of four pixels. Each such clique C contains all the λ_{st} which lie in a 4×4 window. The action potential $J_C(\lambda)$ gives a high score to desirable link patterns and a low score to undesirable ones. A deterministic explicit parallel algorithm [RC 5] recodes the link configuration λ into a finite family of ordered contour chains of linked pixels, of type $[s_1\ s_2 \dots s_p]$ with variable length p and provides "gap neurons" ρ_{st} linking "last pixel" s of one chain and "initial pixel" t of another whenever $|s - t|$ is small.

A standard smoothing method associates to each point s of a contour chain a descriptor $\delta_S = (v_S, K_S, \varepsilon_S)$ regrouping local speed, curvature, and smoothing error at s .

Gap neurons ρ_{st} are linked to descriptors δ_s and δ_t and all ρ_{su} with $|s - u|$ small.

This [GAP 6] layer finalizes the contour chaining. It is now possible to select, using the smooth speeds v_s , a new layer [LINK 7] of link neurons $\tilde{\lambda}_{st}$ which we link to [COL 2] and [LINK 3] by obvious feedback vertical synapses.

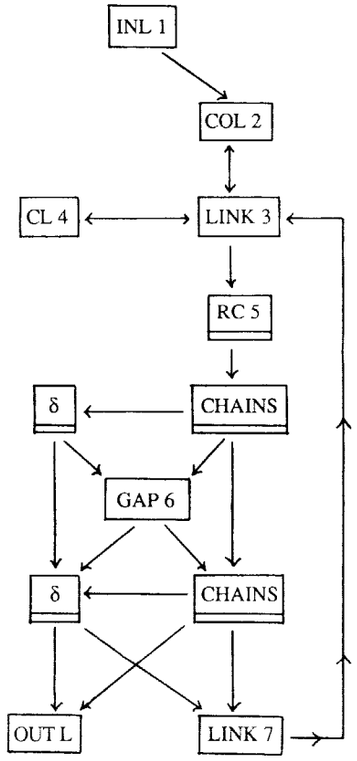


Figure 2 : Architecture of a module for smooth contour chains extraction

Underlined boxes are computed from incoming data by explicit deterministic nets (parallel algorithms). Layers COL 2, LINK 3, GAP 6 have internal synapses.

4.2. Modular network training

The previous network contains essentially two synchronous Boltzmann modules that require training : the block [INL 1] [COL 2] [LINK 3] [CL 4] and the [GAP 6] layer. Moreover the feedback weights [LINK 7] → [LINK 3] have to be learned. The number of input neurons is 2048 one-bit units, the analyzing layers [COL 2] [LINK 3] represent about 3300 one-bit units, the clique layer [CL 4] includes 64 cliques. The [GAP 6] layer involves about 800 one-bit units. However, the connectivity is highly local : 43 synapses per cell in [COL 2], 8 synapses per cell in [CL 4], 21 synapses per cell in [LINK 3]. Moreover, the rotation and translation invariance of weights limits the number of unknown weights to about 120 parameters for the first 4 layers. Since reasonable initial guesses for these weights are easily provided using the the extended experience available in contour detection by relaxation methods, we see that actual training of such a network must be intrinsically quite fast, if massive parallelization is available. Hence such a module, once roughly trained on standard images, should exhibit interesting *fast adaptative behaviour* to deal with noisy environments or specific families of images.

It is also fairly clear that actual training of such a network should and can be done in *stages*, using the strongly hierarchical and modular structure, and the fact that "good answers" at intermediary levels are easily computable on standard images. Thus a preliminary training of [COL 2] can precede the joint training of [COL 2] [CL 4] [LINK 3]. Similarly [GAP 6] can be trained separately, before the final tuning of the global architecture.

Various versions of this architecture are currently studied by J. Lacaille at DIAM-ENS, and actual simulation results will be published elsewhere. We point out that multiscale analysis in the lower layers of this network are quite feasible, and in fact useful to handle noisy or blurred contours. Note also that coarser grain analysis in [COL 2] and [LINK 3] makes sense if limits are imposed on the density of one-bit units by the computing resources actually available.

Finally for actual implementation of such networks with specialized hardware, the possibility of regrouping several units into single multivalued neurons is also quite possible.

4.3. Boat outline classification on infrared images

This task has been undertaken in collaboration with A. Doutriaux (CESTA Toulon and U. San Diego) and L. Younès (DIAM-ENS) who in particular handled all simulations on the ETCA Connection machine, with the technical support of P. Clermont.

Figure 3

A typical boat outline after elementary thresholding (on infrared images)

The images studied here were infra-red pictures of single boats at sea, digitalized on 64 gray-levels per pixel, with standard 512×512 pixels. The goal was to classify these boats in 4 categories, defined *a priori*.

The original examples were submitted to deformations (by oblique affinities and symmetries) as well as to artificial noise to increase the size of the global set available to 275 examples, out of which 25 examples were kept as a control set and 250 used as a training set.

We decided to extract, by standard contour analysis algorithms, a description of each boat outline as a chained sequence of small straight line segments ; typically the number of segments for an extracted chain was in the range [75-100].

The chains of segments were recoded in the following way : first they were *locally smoothed* by standard least square fitting, and then *reparametrized so that the total length of each smooth outline became equal to one*. Then they were *resegmented* into 50 arcs of length 1/50 . For each of these arcs, 3 characteristics were computed : the mean orientation $\overline{\theta}$, the mean curvature ρ , and the local smoothing error $\overline{\varepsilon}$.

The 150 data $(\overline{\theta}_i,\overline{\rho}_i,\overline{\varepsilon}_i)$ $i = 1 \ldots 50$ were used to activate three input layers $(\theta_i,\rho_i,\varepsilon_i)$ $i = 1 \ldots 50$ of $\{0,1\}$ neurons using adaptative thresholding.

The five hidden layers included three "local analysis" layers $(\hat{\theta}_i,\hat{\rho}_i,\hat{\varepsilon}_i)$ $i = 1 \ldots 50$ and two "joint- evaluation layers" (α_i) $i = 1 \ldots 50$ and (β_i) $i = 1 \ldots 50$.

The output layer had only four neurons (four classes only).

Connections were essentially local ; each $\hat{\theta}_i$ was connected to the inputs θ_i , θ_{i-1} , θ_{i+1} and to β_i . Similar connections for $\hat{\rho}_i$, $\hat{\varepsilon}_i$.

Each α_i was connected to β_i , $\theta_i,\rho_i,\varepsilon_i$ and to the four output units. Each β_i was connected to α_i , $\hat{\theta}_i,\hat{\rho}_i,\hat{\varepsilon}_i$ and to the four output units.

Thus we had a synchronous neural net of 402 neurons connected to a coding layer of 150 neurons. The number of weights (including all thresholds) was small : about 2000 weights.

The synchronous learning was completed in about 500 weight iterations. On a 16 K-connection machine, the total CPU time for learning was 2 hours.

Performances of this simple network were quite encouraging : on the training set (250 examples), the percentage of correct recognition was 97 %. On the global set (275 examples), the percentage of correct recognition was 95 %.

REFERENCES

[1] R. AZENCOTT - Markov fields and low-level vision tasks, Proc. Int. Cong. App. Math., ICIAM, Paris (1987).

[2] - Gibbs fields, simulated annealing, and low-level vision tasks, Proc. Cong. Pattern Recognition, AFCET-INRIA, Antibes (1987).

[3] - Synchronous Boltzmann machines : learning rules, Proc. Congress "Neural networks", Les Arcs (1989), to appear Springer-Verlag, NATO series (1990).

[4] - Parameter estimation for synchronous Markov fields, to appear (1990).

[5] - General Boltzmann machines with multiple interactions, to appear in IEEE PAMI (1990).

[6] - Boltzmann Machines, Gibbs fields and artificial vision, to appear (1990), Addison-Wesley.

[7] B. CHALMOND - Image restoration using an estimated Markov model, IEEE PAMI (1988).

[8] D. and S. GEMAN - Gibbs fields, simulated annealing, and Bayesian reconstruction of images, IEEE, PAMI (1984).

[9] S. GEMAN and C. GRAFFIGNE - Gibbs fields and image segmentations, Proc. Int. Cong. Math. (1987).

[10] G. HINTON, T. SEJNOWSKI, D.H. ACKLEY - Boltzmann machines : constraint satisfaction networks that learn (technical report) : Carnegie Mellon University (1984).