Alternating Minimization and Boltzmann Machine Learning

William Byrne, *Student Member, IEEE*

Abstract—**Training a Boltzmann machine with hidden units is appropriately treated in information geometry using the information divergence and the technique of alternating minimization. The resulting algorithm is shown to be closely related to gradient descent Boltzmann machine learning rules, and the close relationship of both to the EM algorithm is described. An iterative proportional fitting procedure for training machines without hidden units is described and incorporated into the alternating minimization algorithm.**

I. Introduction

BOLTZMANN machines are neural networks whose behavior can be described statistically in terms of simple interactions between the units of the network [1]. In general, network behavior may be desired which is more complex than can be achieved through these simple interactions. To obtain such behavior, additional units can be added to the network which effectively allow more complicated interactions between the units. The behavior of the original units is described by marginal distributions obtained from the state distribution of the entire network. The training problem is then to find a network such that these marginals closely approximate some given distribution.

Consider the set of all Boltzmann machines with a given number of units and consider all distributions on the machine states which have the desired marginal distributions. If a Boltzmann machine can be found whose distribution belonged to this latter set, the problem is solved. Otherwise, a possible solution is to chose a machine which is somehow close to this set.

This learning problem is described in information geometric terms. The method of alternating minimization is applied to produce a sequence of machines though repeated projections between these sets. Each machine is closer to the set of desirable distributions than its predecessor as measured by the information divergence, which is particularly appropriate for operations on the distributions considered here. It will be shown that existing Boltzmann machine learning rules can be interpreted using this technique of alternating minimization.

II. Boltzmann Machines

A Boltzmann machine is a network of n units which behave according to a stochastic, discrete-time updating rule. The state of the network is described by a binary vector of length n, $x^n = [x_1, \cdots, x_n]$, where $x_j \in \{0,1\}$ specifies whether unit j has value 0 or 1 in the network state. When the network is running, at each time instant one of the units is chosen for updating. Suppose unit i is chosen for updating. It assumes the value 1 with probability

$$p_i = \frac{1}{1+\exp\{-\frac{1}{2T}\Sigma_{1\le j\le n} w_{i,j}x_j\}} \tag{1}$$

and the value 0 with the complementary probability $1-p_i$. The parameters of the updating rule are the temperature, T, and the network connectivities, $\{w_{i,j}\ 1 \le i,j \le n\}$, where $w_{i,j}$ describes the effect of unit j on unit i. Usually, $w_{i,j}$ is specified to be equal to $w_{j,i}$ and the self-connectivities $w_{i,i}$ are 0. When this is so, the connectivities can be arranged in a real-valued, symmetric, zero–diagonal $n \times n$ matrix $W = (w_{i,j})$, and it can be shown that if the units are chosen uniformly for updating, the steady-state probability of finding the network in state x^n is

$$\begin{aligned} B(x^n) &= b\exp\Big\{\frac{1}{T}\sum_{1\le i<j\le n} w_{i,j}x_ix_j\Big\} \\ &= b\exp\Big\{\frac{1}{2T}x^nWx^{n'}\Big\} \end{aligned} \tag{2}$$

$$\frac{1}{b} = \sum_{x^n\in\{0,1\}^n} \exp\Big\{\frac{1}{T}\sum_{1\le i<j\le n} w_{i,j}x_ix_j\Big\}. \tag{3}$$

The constant b is a normalizing constant so that for all matrices W described above, the function B is a valid state distribution. The temperature, T, determines the randomness in the system, as can be seen from (2): as T increases with W fixed, the state distribution becomes uniform. Varying T allows for complex changes in behavior which are exploited in various applications of Boltzmann machines. However, T usually varies slowly with respect to network updating and training, so T will be fixed at 1 for the rest of this work (see [2] for a description of the issues involved in varying temperature).

In the treatment of Boltzmann machines presented here, a machine and the parametric form of its state distribution (i.e., (2)) will be considered identical. The family of Boltzmann machines with n units will be written as

$$\mathcal{B} = \Big\{ B \in \mathcal{P}^n : B(x^n) = b\exp\Big\{\sum_{1\le i<j\le n} w_{i,j}x_ix_j\Big\}\Big\} \tag{4}$$

Manuscript received February 25, 1991; revised November 11, 1991. This work was supported by the Air Force Office of Scientific Research, by the Naval Research Laboratory, and by the National Science Foundation under Grant NSFD CDR-8803012.

The author is with the Department of Electrical Engineering and the Systems Research Center, University of Maryland, College Park, MD 20742.

IEEE Log Number 9105994.

where $\mathcal{P}^n$ is the collection of probability distributions on the set of machine states $\{0,1\}^n$.

III. The Boltzmann Machine Training Problem

The training problem addressed here is that of constructing an n-unit Boltzmann machine whose *visible* units $1, \cdots, v$ behave according to a given distribution, $\hat{P}$, which is defined on the states of the visible units $[x_1, \cdots, x_v] \in \{0,1\}^v$. The probabilities which can be assigned to a set of binary states by a v-unit Boltzmann machine (i.e., those probabilities described by (2)) have a specific form and, in general, $\hat{P}$ may not be one of these distributions. However, by adding *hidden* units to the network, it may be possible to alter the behavior of the original visible units so that it more closely approximates $\hat{P}$ (this is discussed in [1]).

Since the entire network behaves according to the state distribution B, the behavior of the v visible units is described by the marginal distribution, B_v, determined from B:

$$B_v(x^v) = \sum_{y^n \in \{0,1\}^n} B(y^n) I_{x^v}(y^n), \tag{5}$$

where $I_{x^v}([y_1, \cdots, y_n]) = 1$ if $[y_1, \cdots, y_v] = x^v$ and 0 otherwise (the subscript v indicates the marginal distribution over the units $1, \cdots, v$). The objective is to find a machine for which B_v is close to $\hat{P}$.

The similarity between two distributions will be measured using the information divergence:

$$D(P\|Q) = \sum_{z \in \mathcal{Z}} P(z) \log_2 \frac{P(z)}{Q(z)}, \tag{6}$$

where P and Q are arbitrary distributions which assign probabilities to the same set of events $\mathcal{Z}$. Using the information divergence, the training problem can be stated as finding B^{opt} for which

$$D\left(\hat{P}\|B_v^{\text{opt}}\right) \le D\left(\hat{P}\|B_v\right) \quad \forall\, B \in \mathcal{B}. \tag{7}$$

Finding B^{opt} is therefore the same as solving

$$\inf_{B \in \mathcal{B}} D\left(\hat{P}\|B_v\right) \tag{8}$$

in terms of the parameters which specify B.

An Equivalent statement of the Training Problem

The statement of the training problem in (8) is somewhat awkward, in that it involves the marginal distributions of the machines. A simpler reformulation is possible in terms of the family, $\mathcal{D}$, of desirable distributions on the set of machine states $\{0,1\}^n$:

$$\mathcal{D} = \left\{ P \in \mathcal{P}^n : \sum_{y^n \in \{0,1\}^n} P(y^n) I_{x^v}(y^n) = \hat{P}(x^v) \quad \forall\, x^v \{0,1\}^v \right\} \tag{9}$$

whose marginal distribution on the visible units agrees with $\hat{P}$. Finding a machine whose visible units behave according to $\hat{P}$ is then the same as finding a machine which belongs to $\mathcal{D}$. It will be shown below that $\mathcal{D}$ is related to $\mathcal{B}$ in such a way that solving

$$\inf_{B \in \mathcal{B}} D\left(\hat{P}\|B_v\right) \tag{10}$$

is the same as solving

$$\inf_{B \in \mathcal{B}\, \mathcal{P} \in \mathcal{D}} D(P\|B). \tag{11}$$

This is a consequence of the following relationship between Boltzmann machines and the family $\mathcal{D}$:

Property 1: For a fixed $B \in \mathcal{B}$,

$$D\left(\hat{P}\|B_v\right) = \min_{P \in \mathcal{D}} D(P\|B) \tag{12}$$

and there is a $P^* \in \mathcal{D}$ which achieves this minimum.

This property follows immediately from the log-sum inequality [3]. For $P \in \mathcal{D}$ and B fixed,

$$D\left(\hat{P}\|B_v\right) = D(P_v\|B_v) = \sum_{x^v} \left(\sum_{y^n} P(y^n) I_{x^v}(y^n) \right) \cdot \log \frac{\Sigma_{y^n} P(y^n) I_{x^v}(y^n)}{\Sigma_{y^n} B(y^n) I_{x^v}(y^n)} \tag{13}$$

$$\le \sum_{x^n} P(x^n) \log \frac{P(x^n)}{B(x^n)} = D(P\|B) \tag{14}$$

so that $D(\hat{P}\|B_v) \le \inf_{P \in \mathcal{D}} D(P\|B)$. Equality holds in (14) if and only if [3]

$$\frac{B_v(x^v)}{\hat{P}(x^v)} = \frac{B(x^n)}{P(x^n)} \quad \forall\, \{x_1, \cdots, x_n\} \in \{0,1\}^n \tag{15}$$

that is, for $B(x^n)/P(x^n)$ constant when the state of the visible units is specified. This necessary and sufficient condition can be solved in terms of $\hat{P}$ and B to find the unique P^*:

$$P^*(\{x_1, \cdots, x_n\}) = \hat{P}(\{x_1, \cdots, x_v\}) \frac{B(\{x_1, \cdots, x_n\})}{B_v(\{x_1, \cdots, x_v\})}, \tag{16}$$

which achieves the lower bound

$$D(P^*\|B) = D\left(\hat{P}\|B_v\right). \tag{17}$$

By construction, the v marginal of P^* is $\hat{P}$, so P^* is the element of $\mathcal{D}$ closest to B under the information divergence. This P^* is called the I-projection of B on $\mathcal{D}$.

Restatement of the Training Problem: Applying Property 1 to (8) yields

$$\inf_{B \in \mathcal{B}} D\left(\hat{P}\|B_v\right) = \inf_{B \in \mathcal{B}} \inf_{P \in \mathcal{D}} D(P\|B). \tag{18}$$

So the Boltzmann machine training problem of finding the best machine, B^{opt}, is that of solving

$$\inf_{B \in \mathcal{B}\, \mathcal{P} \in \mathcal{D}} D(P\|B) \tag{19}$$

in terms of its parameters $(w_{i,j}^{\text{opt}})$ and b^{opt}.

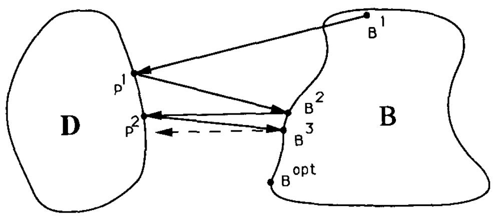


Fig. 1. Alternating minimization applied to Boltzmann machine learning.

IV. Alternating Minimization

Finding B^{opt} requires solving simultaneous nonlinear equations in $(w_{i,j})$, which is in general a difficult problem. However, a suboptimal solution to the training problem can be found using the technique of alternating minimization (Fig. 1) [4], which yields a sequence of distributions $\{B^1, P^1, B^2, P^2, B^3, P^3, \cdots\}$ such that

$$D(P^t \| B^t) = \min_{P \in \mathcal{D}} D(P \| B^t) \tag{20}$$

$$D(P^t \| B^{t+1}) = \min_{B \in \mathcal{B}} D(P^t \| B). \tag{21}$$

The machines which appear in this sequence have the property that

$$\min_{P \in \mathcal{D}} D(P \| B^{t+1}) \leq \min_{P \in \mathcal{D}} D(P \| B^t), \tag{22}$$

which, through Property 1, means that the machines improve with each iteration:

$$D\left(\hat{P} \| B_v^{t+1}\right) \leq D\left(\hat{P} \| B_v^t\right). \tag{23}$$

The nonnegativity of the information divergence [3] implies that this nonincreasing sequence of divergences converges. If the sequence converges to zero, then the resulting distribution, $\lim_{t \to \infty} B^t$, belongs to $\mathcal{D}$; therefore a machine can be found whose behavior is arbitrarily good. It is also possible that the divergences are bounded away from zero, which may occur even when $\mathcal{D}$ and $\mathcal{B}$ intersect. In this case the training algorithm has found a local minimum, which may still be a useful machine. In practice, the procedure is halted when $D\left(\hat{P} \| B_v^{t+1}\right)$ is sufficiently small.

A. Finding P^t from B^t

P^t is computed from B^t, which is available from the previous iteration (the initial machine, B^1, is chosen at random). The distribution P^t is the I-projection of B^t on $\mathcal{D}$, which satisfies

$$D(P^t \| B^t) = \min_{P \in \mathcal{D}} D(P \| B^t) \tag{24}$$

and can be found immediately from B^t in closed form as P^*, defined in (16).

B. Finding B^{t+1} from P^t

The computation of B^{t+1} from P^t is more complicated than finding P^t from B^t. It is necessary to solve

$$\min_{B \in \mathcal{B}} D(P^t \| B) \tag{25}$$

in terms of the connectivities $(w_{i,j})$ which determine B. The derivation of the following property yields a description of the desired B^{t+1} in terms of parameters specified by P^t.

Property 2: For a fixed $\overline{P} \in \mathcal{D}$ there is a $B^* \in \mathcal{B}$ for which

$$D(\overline{P} \| B^*) = \min_{B \in \mathcal{B}} D(\overline{P} \| B). \tag{26}$$

Property 2 follows from the relationship between the moments of an exponential distribution and the family of distributions with the same moments. Let $\mathcal{L}$ be the linear family of distributions with the same moments $(p_{i,j})$ as a given distribution $\overline{P}$:

$$p_{i,j} \equiv \sum_{x^n} \overline{P}(x^n) x_i x_j = \overline{P}(\{x^n : x_i = x_j = 1\}), \quad 1 \leq i < j \leq n \tag{27}$$

$$\mathcal{L} = \left\{ P \in \mathcal{P}^n : \sum_{x^n} P(x^n) x_i x_j = p_{i,j}, \quad 1 \leq i < j \leq n \right\} \tag{28}$$

and let Q be any distribution on $\{0,1\}^n$. The I-projection of Q on $\mathcal{L}$ is defined to be the distribution P^* of $\mathcal{L}$ for which $D(P^* \| Q) \leq D(P \| Q) \; \forall \; P \in \mathcal{L}$. The Pythagorean theorem of information geometry [5] specifies P^* through the relationship

$$D(P \| Q) = D(P \| P^*) + D(P^* \| Q) \quad \forall \, P \in \mathcal{L}, \tag{29}$$

where P^* is constructed so that it belongs to $\mathcal{L}$:

$$P^*(x^n) = c\, Q(x^n) \exp\left\{ \sum_{1 \leq i < j \leq n} \lambda_{i,j} x_i x_j \right\} \tag{30}$$

$$\frac{1}{c} = \sum_{x^n} Q(x^n) \exp\left\{ \sum_{1 \leq i < j \leq n} \lambda_{i,j} x_i x_j \right\} \tag{31}$$

$$\lambda_{i,j} : p_{i,j} = \sum_{x^n} P^*(x^n) x_i x_j, \qquad 1 \leq i < j \leq n. \tag{32}$$

In this application the Pythagorean theorem is easily verified so its proof will not be presented.

To obtain Property 2, pick a Boltzmann machine B with parameters $(w_{i,j})$ and b. Using (30), the I-projection of B on $\mathcal{L}$ is

$$P^*(x^n) = cB(x^n) \exp\left\{ \sum_{1 \leq i < j \leq n} \lambda_{i,j} x_i x_j \right\} \tag{33}$$

$$= cb \exp\left\{ \sum_{1 \leq i < j \leq n} (w_{i,j} + \lambda_{i,j}) x_i x_j \right\} \tag{34}$$

$$= b^* \exp\left\{ \sum_{1\le i<j\le n} w^*_{i,j} x_i x_j \right\}, \tag{35}$$

where $w^*_{i,j} = w_{i,j} + \lambda_{i,j}$. P^* is clearly a Boltzmann machine which belongs to $\mathcal{L}$ by construction, so it is renamed B^*. Substituting B^* for P^* in (29) yields

$$D(P\|B) = D(P\|B^*) + D(B^*\|B) \quad \forall\, P \in \mathcal{L}. \tag{36}$$

Since the given distribution, $\overline{P}$, is certainly an element of $\mathcal{L}$ and $D(B^*\|B) = 0$ iff $B^* = B$ [3],

$$D(\overline{P}\|B^*) < D(\overline{P}\|B) \quad \forall\, B \in \mathcal{B} \quad B \neq B^*. \tag{37}$$

This implies that, for a given $\overline{P}$, the Boltzmann machine B^* with the same moments as $\overline{P}$ achieves $\min_{B\in\mathcal{B}} D(\overline{P}\|B)$ and Property 2 follows.

Property 2 requires that B^{t+1} be chosen as B^*, the element of $\mathcal{B}$ whose moments agree with those of P^t : B^{t+1} should belong to $\mathcal{L}^t$, where

$$p^t_{i,j} \equiv \sum_{x^n} P^t(x^n) x_i x_j, \quad 1 \le i < j \le n \tag{38}$$

$$\mathcal{L}^t = \left\{ P \in \mathcal{P}^n : \sum_{x^n} P(x^n) x_i x_j = p^t_{i,j}, \quad 1 \le i < j \le n \right\}. \tag{39}$$

This is an instance of the duality between the parameters and the moments of exponential distributions; complete knowledge of either determines the distribution (see, e.g., [6]).

Finding B^{t+1} from P^t has a geometric interpretation. Equation (29) shows that $\mathcal{B}$ and $\mathcal{L}^t$ intersect at a single distribution, B^t, and do so at a "right angle," where the information divergence plays the role of the square of the Euclidean distance in the analogy to plane geometry.

There is also a geometric description of finding P^t from B^t. This involves the exponential family, $\mathcal{E}$, of distributions

$$\mathcal{E} = \{ E_\alpha \in \mathcal{P}^n : \log E_\alpha(x^n) = (1-\alpha)\log P^t(x^n) + \alpha \log B^t(x^n) + \log c_\alpha \} \tag{40}$$

that connects the two distributions. For $P \in \mathcal{D}$ and $E_\alpha \in \mathcal{E}$,

$$D(P\|E_\alpha) = D(P\|P^t) + \alpha \sum_{x^n} P(x^n) \log \frac{P^t(x^n)}{B^t(x^n)} - \log c_\alpha \tag{41}$$

$$= D(P\|P^t) + \alpha \sum_{x^v} \hat{P}(x^v) \log \frac{\hat{P}(x^v)}{B^t_v(x^v)} - \log c_\alpha, \tag{42}$$

where the last equality holds because P^t is chosen to satisfy (eq. (16))

$$\frac{P^t([x_1,\cdots,x_n])}{B^t([x_1,\cdots,x_n])} = \frac{\hat{P}([x_1,\cdots,x_v])}{B^t_v([x_1,\cdots,x_v])}. \tag{43}$$

By the same argument,

$$D(P^t\|E_\alpha) = \alpha \sum_{x^v} \hat{P}(x^v) \log \frac{\hat{P}(x^v)}{B^t_v(x^v)} - \log c_\alpha \tag{44}$$

so that

$$D(P\|E_\alpha) = D(P\|P^t) + D(P^t\|E_\alpha) \quad \forall\, P \in \mathcal{D} \quad \forall\, E_\alpha \in \mathcal{E}. \tag{45}$$

From this it is apparent that the intersection of $\mathcal{D}$ and $\mathcal{E}$ contains a single distribution, P^t, and that this intersection occurs at a right angle. These two geometric descriptions are applications of the I-divergence geometry presented in [5] and are also applied to Boltzmann machine learning within a more general information geometry framework in [7].

C. Finding B^{t+1} Through Iterative Proportional Fitting

A useful relationship follows from the Pythagorean theorem. The distribution of a Boltzmann machine can be rewritten as

$$B(x^n) = cU(x^n) \exp\left\{ \sum_{1\le i<j\le n} w_{i,j} x_i x_j \right\}, \tag{46}$$

where U is the uniform distribution on $\{0,1\}^n$. B belongs to a particular linear family of the form given in (28); thus, by comparison to (30), B is the I-projection of U onto this family.[1] Therefore, B^{t+1} is the I-projection of the uniform distribution U onto $\mathcal{L}^t$. Note also that $\mathcal{L}^t$ can be written as an intersection of larger linear families:

$$\mathcal{L}^t = \bigcap_{1\le i<j\le n} \mathcal{L}^t_{i,j} \tag{47}$$

$$\mathcal{L}^t_{i,j} = \left\{ P \in \mathcal{P}^n : \sum_{x^n} P(x^n) x_i x_j = p^t_{i,j} \right\}. \tag{48}$$

To find an I-projection on such an intersection, an iterative proportional fitting procedure (IPFP) [5] can be used to find the projection of U onto $\mathcal{L}^t$ in terms of projections between the $\mathcal{L}^t_{i,j}$. U is projected onto $\mathcal{L}^t_{1,2}$, and the resulting projection is projected onto $\mathcal{L}^t_{1,3}$. This procedure is carried out cyclically, projecting back onto $\mathcal{L}^t_{1,2}$ from $\mathcal{L}^t_{n-1,n}$ (e.g., Fig. 2). The resulting infinite sequence of projections converges to B^{t+1}. Each step in this computation yields a Boltzmann machine which satisfies one of the moment constraints. The computation of each projection is straightforward, since only one moment constraint must be met.

To apply this to computing B^{t+1} from P^t, form an infinite sequence of sets in which the sets $\mathcal{L}^t_{i,j}$ repeat infinitely often:[2]

$$\{\mathcal{L}_1, \mathcal{L}_2, \cdots\} = \{\mathcal{L}^t_{1,2}, \mathcal{L}^t_{1,3}, \cdots, \mathcal{L}^t_{n-1,n}, \mathcal{L}^t_{1,2}, \mathcal{L}^t_{1,3}, \cdots\}, \tag{49}$$

where (i_k, j_k) are such that $\mathcal{L}_k = \mathcal{L}^t_{i_k,j_k}$. Form $\{p_1, p_2, \cdots\}$ from $p^t_{i,j}$ in a corresponding manner. This allows the cyclic projection introduced above to be described as projections

[1] $D(P\|U) = -H(P)$, where H is the entropy function, so B is also the maximum entropy distribution in this family.

[2] The ordering is not important; each set should appear infinitely often.

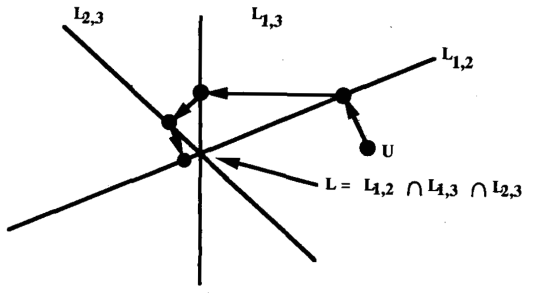


Fig. 2. Computation of B^{t+1} from P^t in a three-unit machine.

from $\mathcal{L}_{k-1}$ to $\mathcal{L}_k$: let $Q_0 = U$ and for $k \geq 1$ let Q_k be the I-projection of Q_{k-1} on $\mathcal{L}_k$. From the expression for the I-projection onto a linear family (eq. (30)),

$$Q_k(x^n) = c_k Q_{k-1}(x^n) \exp\{\lambda_k x_{i_k} x_{j_k}\} \tag{50}$$

$$\lambda_k : p_k = \sum_{x^n} c_k Q_{k-1}(x^n) \exp\{\lambda_k x_{i_k} x_{j_k}\} x_{i_k} x_{j_k} \tag{51}$$

$$\frac{1}{c_k} = \sum_{x^n} Q_{k-1}(x^n) \exp\{\lambda_k x_{i_k} x_{j_k}\}. \tag{52}$$

Evaluating the condition on λ_k yields

$$p_k = \sum_{x^n} \frac{Q_{k-1}(x^n) \exp\{\lambda_k x_{i_k} x_{j_k}\} x_{i_k} x_{j_k}}{\Sigma_{y^n} Q_{k-1}(y^n) \exp\{\lambda_k y_{i_k} y_{j_k}\}} \tag{53}$$

$$= \frac{e^{\lambda_k} \sigma_k}{e^{\lambda_k} \sigma_k + 1 - \sigma_k} \tag{54}$$

which can be solved for λ_k in terms of p_k and Q_{k-1}

$$e^{\lambda_k} = \frac{1-\sigma_k}{\sigma_k} \frac{p_k}{1-p_k} \tag{55}$$

$$\sigma_k \equiv Q_{k-1}(\{x^n : x_{i_k} = x_{j_k} = 1\}). \tag{56}$$

From the above updating procedure, it is clear that each element of the sequence $\{Q_k\}$ is a Boltzmann machine with weights $(w_{i,j}^k)$:

$$Q_k(x^n) = c_k Q_{k-1}(x^n) e^{\lambda_k x_{i_k} x_{j_k}} \tag{57}$$

$$= c_k \cdots c_1 \exp\left\{ \sum_{h=1}^{k} \lambda_h x_{i_h} x_{j_h} \right\} \tag{58}$$

$$= c_k b_{k-1} \exp\left\{ \sum_{1 \leq i < j \leq n} w_{i,j}^{k-1} x_i x_j + \lambda_k x_{i_k} x_{j_k} \right\} \tag{59}$$

$$= b_k \exp\left\{ \sum_{1 \leq i < j \leq n} w_{i,j}^k x_i x_j \right\} \tag{60}$$

with

$$w_{i,j}^k = \begin{cases} w_{i,j}^{k-1} + \lambda_k & \text{if } i = i_k \text{ and } j = j_k \\ w_{i,j}^{k-1} & \text{otherwise} \end{cases} \tag{61}$$

$$b_k = c_k b_{k-1}, \tag{62}$$

where the initial weights, $w_{i,j}^0$, are zero, since Q_0 is the uniform distribution. This sequence converges to B^{t+1}, the I-projection of U on $\mathcal{L}^t$, and the machine which achieves

$$D(P^t \| B^{t+1}) = \min_{B \in \mathcal{B}} D(P^t \| B). \tag{63}$$

It is possible to implement the IPFP in a fairly efficient nondistributed algorithm by exploiting recursive relationships between parameters. Continuing the argument which led to (59) yields

$$Q_k(x^n) = \begin{cases} \frac{1-\sigma_k}{1-p_k} Q_{k-1}(x^n) & \text{if } x_{i_k} = x_{j_k} = 1 \\ \frac{p_k}{\sigma_k} Q_{k-1}(x^n) & \text{otherwise.} \end{cases} \tag{64}$$

The iterations are halted when the σ_k are close enough to the p_{i_k,j_k}^t. This is useful for small networks, but unfortunately the operational cost increases exponentially with the network size.

The alternating minimization training algorithm has been developed as a nested recursion. The desired distributions $\{P^t\}$ are obtained through projection onto $\mathcal{D}$. The IPFP is then used to obtain the machine closest to each of these distributions. The algorithm can be implemented using the Boltzmann machine distributed processing architecture, but before describing this, this training algorithm will be compared with other Boltzmann machine training algorithms.

V. Relationship to Other Boltzmann Machine Training Algorithms

The alternating minimization algorithm developed here is very closely related to other Boltzmann machine learning algorithms. Consider the problem of training a machine which has no hidden units: under the divergence criterion, the training goal is to find the machine B which minimizes $D(P\|B)$ for a given P. This P describes the behavior of all the units in the network. The convexity of the information divergence [3] implies that a machine exists which solves this problem,[3] and a gradient descent algorithm is described in [8] (see also [2]) that finds this machine. This minimization is also solved by the IPFP described earlier. The IPFP can therefore be used to train Boltzmann machines which have no hidden units.

This problem of training a network to an entirely known distribution is only of restricted interest. Boltzmann machines are not powerful enough without hidden units, and if a machine has hidden units, the correct behavior of the hidden units is usually unknown. However, if the distribution which described the proper behavior of the entire network of hidden and visible units were known, the IPFP or gradient descent algorithms could be used to obtain the best network. The alternating minimization algorithm does not solve this problem of finding the globally optimal behavior for the entire network; however it does provide locally improving estimates of the optimal behavior through its projections onto the set of desirable distributions. Each of these projections is then used by the IPFP to find exactly the corresponding best machine. The algorithm can therefore be thought of as having two parts: one

[3]The minimum value of this criterion may be unattainable, but it can be approached arbitrarily closely although some of the weights may diverge to $\pm\infty$.

part provides estimates of the best network behavior and the other part finds a Boltzmann machine whose behavior closely approximates this estimate.

Gradient descent methods are also used to train machines with hidden units [9]. Two modes of network operation are employed to achieve this. While free running, a network updates its states as described previously and has steady-state distribution B. When the network is *clamped*, the visible units are fixed according to the desired distribution, $\hat{P}$, and the remaining units update their states as if they were free running. This results in a new steady-state distribution, denoted B^+. It can be shown that

$$\frac{\delta}{\delta w_{i,j}} D(B^+ \| B) = -(p_{i,j}^+ - p_{i,j}), \qquad 1 \le i < j \le n, \tag{65}$$

where $p_{i,j}^+ = B^+(\{x^n : x_i = x_j = 1\})$, the probability of finding units i and j active simultaneously when the network is clamped ($p_{i,j}$ is found analogously when the network is free running). If B can be made closer to B^+, clearly the free-running behavior of the visible units will be closer to $\hat{P}$. To achieve this, the weights $(w_{i,j})$ of B are updated in the direction of decreasing $D(B^+\|B)$; that is, each connectivity $w_{i,j}$ is incremented by $-\beta \frac{\delta}{\delta w_{i,j}} D(B^+\|B)$ for some small β. This is done repeatedly: the network is clamped and the $p_{i,j}^+$ are observed; the network is allowed to run freely and the $p_{i,j}$ are observed; the weights are then updated to obtain a new machine. The sequence of machines produced by this gradient descent method will be denoted $\{\tilde{B}^t\}$ to distinguish them from the machines $\{B^t\}$ obtained through alternating minimization.

To see the relationship between gradient descent Boltzmann machine learning and the alternating minimization algorithm, note that the clamped distributions can be written as

$$B^+(x^n) = B(x^n \mid x^v)\hat{P}(x^v) \quad \forall\, x^n = \{x_1, \cdots, x_n\} \tag{66}$$

$$= \frac{B(x^n)}{B_v(x^v)} \hat{P}(x^v) \tag{67}$$

$$B^+(x^n) = P^*(x^n), \tag{68}$$

where $P^*(x^n)$ is the I-projection of B on $\mathcal{D}$ as defined in (16). Clamping is therefore the same as projection onto the set of desirable distributions.

If at each iteration in gradient descent learning, the weights are updated repeatedly before the network is clamped again, it follows from the discussion earlier in this section that the resulting $\tilde{B}^{t+1}$ minimizes $D(\tilde{B}^{t\,+}\|B)$. The convexity of the divergence then implies that $\tilde{B}^{t+1} = B^{t+1}$, and under this repeated weight adjustment, gradient descent learning is equivalent to learning through alternating minimization. The equivalence between the two procedures is also suggested by the learning rule, (65). The weights are updated in order to minimize the difference between the moments of the clamped distribution and the free-running distribution and, as described earlier, equality of these moments ensures that the machine closest to P^t has been found.

It is noted in [1] that the gradient descent update rule, (65), does not depend on whether the units involved are hidden or visible. This is explained by restating the problem in terms of projections between distributions on states $\{0, 1\}^n$, where there is no distinction between visible and hidden; this distinction is present only implicitly in the definition of $\mathcal{D}$.

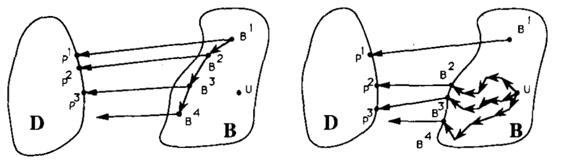


Fig. 3. Comparison between gradient descent (left) and alternating minimization (right).

However, the gradient descent learning algorithm is not usually performed so as to minimize $D(\tilde{B}^{t\,+}\|B)$ and cannot be considered as a form of alternating minimization. It is suggested in [2] that by not performing this minimization, the algorithm combines the best aspects of gradient descent and random search. In a sense the algorithm is a continuing search for better initial conditions which avoids entrapment in local minima (see Fig. 3), although there is no guarantee that it will find or remain in the global minimum. A similar sort of random search should occur by carrying out an inexact version of the alternating minimization algorithm in which, at each iteration, the IPFP halts after only a few steps. The machine B^{t+1} is then somewhat closer to P^t than B^t, but not as close as is usually required. This is investigated in experiments described in a subsequent section (see Fig. 5).

In [4] it is shown that in applications such as this one, alternating minimization leads to the EM algorithm [10], which is a technique for finding maximum likelihood estimates of model parameters when the given information is incomplete. To see this, note that finding B^{t+1} requires minimizing $D(P^t\|B)$, which, by using the definition of P^t in terms of B^t, can be shown to be the same as the following:

$$D(P^t\|B) = \sum_{x^n} P^t(x^n) \log P^t(x^n) - \sum_{[x_1,\cdots,x_n]} \hat{P}(x^v) \frac{B^t(x^n)}{B_v^t(x^v)} \log B(x^n) \tag{69}$$

$$= \sum_{x^n} P^t(x^n) \log P^t(x^n) - \sum_{[x_1,\cdots,x_v]} \hat{P}(x^v) \sum_{[x_{v+1},\cdots,x_n]} B^t(x^n \mid x^v) \log B(x^n) \tag{70}$$

$$= \sum_{x^n} P^t(x^n) \log P^t(x^n) - \sum_{x^v} \hat{P}(x^v) E_{B^t}\,[\log B(x^n) \mid x^v]. \tag{71}$$

Therefore B^{t+1} is found so that it gives a maximum value of $\Sigma_{x^v} \hat{P}(x^v) E_{B^t}[\log B(x^n) \mid x^v]$; this is the combination of expectation and maximization which forms the EM technique. Alternating minimization is therefore equivalent to the EM technique, and gradient descent learning is then either the EM algorithm exactly or an approximation to it.

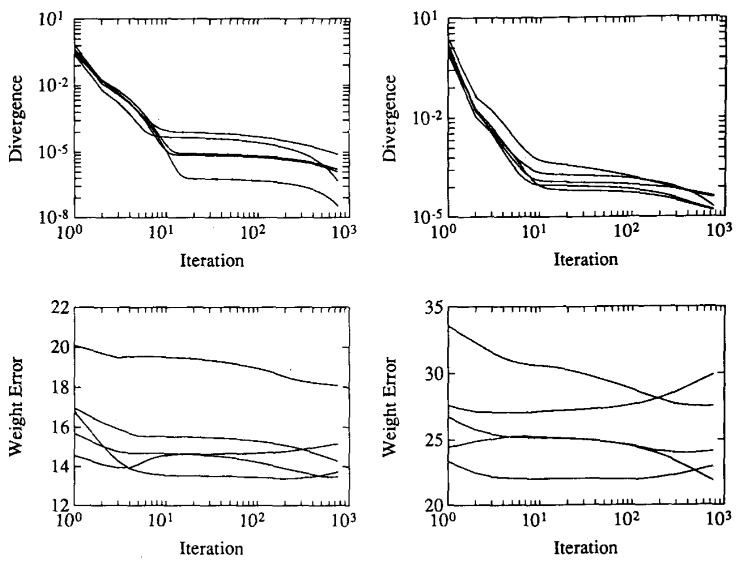


Fig. 4. Convergence evaluations for various network configurations: (left) machines with five hidden units and three visible units; (right) machines with five hidden units and five visible units. The top plots show divergence versus iteration and the bottom plots show the distance to the true weights.

VI. Implementation of the Alternating Minimization Algorithm

The equivalence between clamping and projection onto the set $\mathcal{D}$ implies that the alternating minimization can be implemented using the Boltzmann machine distributed processing architecture. The moments $p_{i,j}^t$ (see (38)) necessary for finding B^{t+1} from B^t can be found by clamping B^t and observing its behavior. Similarly, in performing the IPFP, the σ_k can be found from observing the machine Q_{k-1} (see (56)).

A summary of the training procedure is included here. Two stopping parameters are needed: δ_1 controls the overall procedure and δ_2 controls the IPFP.

Alternating Minimization Training Algorithm Using the IPFP

```
choose $B^1$ at random.
for $t = 1, 2, \cdots$ until $D(P_t \| B_t) \leq \delta_1$ {
    find $p_{i,j}^t$ by clamping $B^t$.
    for $k = 1, 2, \cdots$ until $|\sigma_k - p_{i_k,j_k}^t| \leq \delta_2 \; \forall \; p_{i,j}^t$ {
        find $Q_k$ from $Q_{k-1}$ using (61).
    }
    use the final $Q_k$ as $B^{t+1}$.
}
```

VII. Examples of Training Algorithm Performance

Simulations of the alternating minimization training procedure were performed to verify the convergence properties of the algorithm. The necessary parameters were computed explicitly via their defining equations, rather than through observing the behavior of a Boltzmann machine. In each simulation, a Boltzmann machine is chosen at random and the marginal distribution over its visible units is used as the desired distribution, $\hat{P}$ (a different machine is then used to initialize the algorithm). In this situation there is a best machine and ideally the training algorithm will find it; the problem of choosing a training task appropriate for Boltzmann machines is avoided.

The results presented in Fig. 4 describe training results for two network configurations. Each machine had five hidden units, but the machine described in Fig. 4 (left) had three visible units while the machine in Fig. 4 (right) had five visible units. In each case, the IPFP was halted when the moments agreed within 0.00001, and 750 iterations of the minimization algorithm were performed. The training objective $D(\hat{P} \| B_v^t)$ is plotted at each iteration in each of the ten tests of the algorithm. The l_1 distance between the weights of the machine being trained and the weights $w_{i,j}^{\text{opt}}$ used to generate the distribution $\hat{P}$ is also presented. In none of these cases do the weights of the machine being trained converge to the weights of the machine used to generate the desired distribution $\hat{P}$.

In Fig. 5, the convergence results are presented under two types of updating. In the first, B^{t+1} is found nearly exactly; i.e., the IPFP is performed so that the moments of B^{t+1} and P^t differ by less than 0.00001. In the second type of updating, the IPFP is performed so that each weight is updated ten times, and the resulting machine is taken as B^{t+1}, regardless of how its moments agree with those of P^t. This is of interest as a possible way to reduce computational cost and also sheds light on the differences between the exact minimization algorithm and the random search/gradient descent learning algorithm. The network tested had five hidden units and three visible units and weights were chosen at random to initialize each of the algorithms. Each algorithm has its own characteristic behavior. Measured by the divergence objective function, the exact algorithm tends to improve rapidly initially, and then its learning slows. Presumably, it is exploring a local minimum in the search

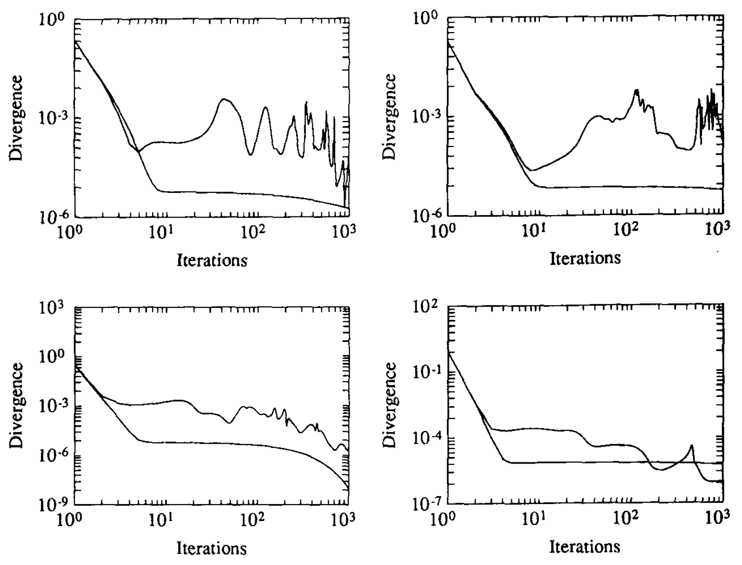


Fig. 5. Exact alternating minimization versus random search applied to a network with five hidden units and three visible units. In the exact method the IPFP is applied to convergence so that the resulting divergences are strictly decreasing. In the random search the IPFP halts after each weight is updated ten times.

space. The inexact algorithm tends to search for a minimum, explores it, and then abandons the search to begin again. Sometimes the inexact algorithm finds a minimum sooner than the exact algorithm reaches that value, but in general, the exact algorithm tends to find slightly better solutions, although at a high computational cost. This behavior should depend on the machine architecture, and knowledge of the training problem could determine which approach works best.

VIII. Discussion

A description of Boltzmann machine learning has been presented using concepts in information geometry. Alternating minimization through information projections and the iterative proportional fitting procedure are applied to the problem and it is shown that gradient descent Boltzmann machine learning rules are a form of alternating minimization.

A simple form of the Boltzmann machine was used to present the alternating minimization procedure. More complicated network behavior than that of (2) is achievable by using a more complicated stochastic updating rule. For example, it is possible to achieve third-order interactions by using second-order update rules [11], i.e. update rules which depend on terms $x_i x_j$ and lead to steady-state distributions of the form

$$B(x^n) = b\exp\left\{\sum w_k x_k + \sum w_{i,k} x_i x_k + \sum w_{i,j,k} x_i x_j x_k\right\}. \tag{72}$$

The alternating minimization algorithm presented here can also be used to train this sort of network. The computation of P^t from B^t is unchanged. B^{t+1} is found from projecting U onto $\mathcal{L}^{t+1}$ using the IPFP, where

$$p_i^t \equiv \sum_{x^n} P^t(x^n) x_i \qquad p_{i,j}^t \equiv \sum_{x^n} P^t(x^n) x_i x_j$$
$$p_{i,j,k}^t \equiv \sum_{x^n} P^t(x^n) x_i x_j x_k \tag{73}$$

$$\mathcal{L}^t \equiv \left\{ P' \in \mathcal{P}^n : \sum_{x^n} P'(x^n) x_i = p_i^t \quad \text{and} \quad \sum_{x^n} P'(x^n) x_i x_j = p_{i,j}^t \quad \text{and} \quad \sum_{x^n} P'(x^n) x_i x_j x_k = p_{i,j,k}^t \; 1 \le i < j < k \le n \right\}. \tag{74}$$

In a similar manner, the learning procedure developed here can be extended to machines whose steady-state distributions result from more complex updating rules.

Acknowledgment

This work is an application of material presented by I. Csiszár in a course taught at the University of Maryland Systems Research Center. The author thanks the reviewer for valuable comments and also thanks S. Shamma for his support.

References

[1] G. E. Hinton and T. J. Sejnowski, "Learning and relearning in Boltzmann machines," in *Parallel Distributed Processing,* D. E. Rumelhart and J. L. McClelland Eds. Cambridge, MA: MIT Press, 1986.
[2] E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines.* New York: Wiley, 1989.
[3] I. Csiszár and J. Körner, *Information Theory: Coding Theorems for Discrete Memoryles Systems.* New York: Academic Press, 1981.
[4] I. Csiszár and G. Tusnády, "Information geometry and alternating minimization procedures," in *Statistics and Decisions, Supplementary*

Issue No. 1, E. J. Dudewicz, D. Plachky, and P. K. Sen, Eds. Munich: Oldenbourg Verlag, 1984, pp. 205–237.

[5] I. Csiszár, "I-divergence geometry of probability distributions and minimization problems," *Annals of Probability,* vol. 3, no. 1, pp. 146–15, 1975.

[6] B. Efron, "The geometry of exponential families," *Annals of Statistics,* vol. 6, no. 2, pp. 362–376, 1978.

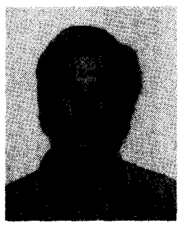
[7] S. Amari, K. Kurata, and H. Nagaoka, "Information geometry of Boltzmann machines," *IEEE Trans. Neural Networks,* vol. 3, pp. 260–271, Mar. 1992.

[8] G. E. Hinton and T. J. Sejnowski, "Analyzing cooperative computation," in *Proc. Fifth Ann. Conf. Cognitive Science Soc.* (Rochester, NY), 1983.

[9] D. Ackley, G. Hinton, and T. Sejnowski, "A learning algorithm for Boltzmann machines," *Cognitive Sci.,* vol. 9, pp. 147–169, 1985.

[10] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Statist. Soc.,* Series B, vol. 39, pp. 1–122, 1977.

[11] T. Sejnowski, "Higher-order Boltzmann machines," in *Neural Networks for Computing,* J. Denker, Ed. New York: American Institute of Physics, 1986, pp. 398–403.



William Byrne (S'79) was born in New York, NY. He received the B.S. degree in electrical engineering from Cornell University in 1982. He is currently a Ph.D. student in the Department of Electrical Engineering at the University of Maryland.

He has worked as an electrical engineer at the National Institutes of Health, Bethesda, MD, and is currently a Graduate Research Assistant in the University of Maryland Systems Research Center and a research engineer at Entropic Research Laboratory Inc., Washington, DC. His research interests include speech recognition and neural networks.