Probability Density Estimation Using Entropy Maximization

Gad Miller
David Horn
School of Physics and Astronomy, Tel Aviv University, Tel Aviv 69978, Israel

We propose a method for estimating probability density functions and conditional density functions by training on data produced by such distributions. The algorithm employs new stochastic variables that amount to coding of the input, using a principle of entropy maximization. It is shown to be closely related to the maximum likelihood approach. The encoding step of the algorithm provides an estimate of the probability distribution. The decoding step serves as a generative mode, producing an ensemble of data with the desired distribution. The algorithm is readily implemented by neural networks, using stochastic gradient ascent to achieve entropy maximization.

1 Introduction

The problem of constructing a probability density function $p(x)$ from a given finite number of data points is well known. There exist many approaches in the statistical literature (see Bishop, 1995, for a short review), which are usually divided into parametric and nonparametric methods, as well as mixtures of the two. The parametric approach starts from an assumed family of distributions and tries to find the best fit. The simplest nonparametric approach is binning the data and smoothing the resulting histogram. Examples of mixture methods are maximum likelihood (ML) and expectation-maximization (EM) algorithms. Neural networks can be naturally included in many of these approaches.

The new approach that we propose here is to employ entropy maximization. In this approach we make use of ideas that have recently been applied successfully to blind separation and blind deconvolution problems (Bell & Sejnowski, 1995). While the latter are usually stated within the framework of information maximization, they naturally reduce to entropy maximization (Nadal & Parga, 1994), as pointed out by Roth and Baram (1996), who have developed and employed this method for probability estimation. As will be explained in the next section, this involves, at some stage, the construction of a new, uniformly distributed variable. In our algorithm, this construction will correspond to an encoding step, in which a neural network is trained with the data so as to produce a uniform output. In this process, it creates a representation of the probability distribution that we look for. A second step of the algorithm can be viewed as a decoding step, in which a uniformly distributed input is fed to another network, which is also trained by the original examples, and serves as a generative model; it produces an output distributed according to the probability density function of the data.

The general structure of our algorithm is explained in the next section using the problem of conditional density function estimation. Here the data are presented in pairs (x, y), and the problem is to find $p(y|x)$. Probability density estimation can be viewed as a special simplified case of conditional density function estimation. Numerical examples of probability distributions serve to demonstrate our method. Section 3 discusses the relation of the maximum entropy (ME) approach to the well-known ML one. This is followed in the next two sections by further applications of the ME method. In section 4 we apply it to the case of stochastic variables with deterministic functional relations, and in section 5 we discuss the possible use of ME as a tool for Monte Carlo generation of artificial data.

2 Entropy Maximization

In this section we demonstrate our method on the problem of estimation of conditional density functions. Data are given as samples of pairs of variables (x, y) that are generated by some distribution function $p(x, y)$, and we wish to determine the functional form of $p(y|x)$. We propose to solve this problem by constructing a network with inputs (x, y) and an output v that is bounded in the domain $0 \leq v(x, y) \leq 1$ and is assumed to be one-to-one with respect to y. Moreover, our aim will be to have this new stochastic variable uniformly distributed, $V = U(0, 1)$, and independent of X. As we will show, this can be achieved by requiring entropy maximization. We refer to this stage of our algorithm as the encoding step, replacing the variable y by v.

Let us define the joint entropy of the output variables:

$$H(X, V) = -\int p(x, v) \ln p(x, v) dx dv. \tag{2.1}$$

This can be rewritten in terms of the input variables as

$$\begin{aligned} H(X, V) &= -\int p(x, y) \ln \frac{p(x, y)}{J} dx dy \\ &= -\int p(x, y) \ln p(x, y) dx dy + \int p(x, y) \ln J dx dy \\ &= H(X, Y) + E[\ln J], \end{aligned} \tag{2.2}$$

where J is the absolute value of the Jacobian,

$$J = \left| \frac{\partial(x, v)}{\partial(x, y)} \right| = \left| \frac{\partial v}{\partial y} \right|. \tag{2.3}$$

The principle of our algorithm is entropy maximization; we look for a choice of $v(x, y)$ such that $H(X, V)$ is maximized. The entropy can be decomposed into

$$H(X, V) = H(X) + H(V) - I(X; V), \tag{2.4}$$

where $I(X; V)$ is the mutual information between X and V. $H(V)$ reaches its upper bound, $H(V) = 0$, for a uniformly distributed $v \sim U(0, 1)$ (Cover & Thomas, 1991). $I(X; V)$ is a nonnegative quantity, lower bounded by 0 for the case where X and V are independent. It remains to be shown that there exists a uniformly distributed V that is independent of X. To derive that function, we note that $p(x, v)$ is related to $p(y|x)$ through

$$p(x, v) = \frac{p(x, y)}{J} = \frac{p(y|x)p(x)}{J}. \tag{2.5}$$

If V is to satisfy the uniformity and independence conditions,

$$p(x, v) = p(x)p(v) = p(x) \tag{2.6}$$

holds. Comparing equations 2.5 and 2.6, we obtain $J = p(y|x)$, or,

$$\frac{\partial v}{\partial y}(x, y) = \pm p(y|x). \tag{2.7}$$

Explicit solutions of v are

$$v(x, y) = \begin{cases} \text{either} & \int_{-\infty}^{y} p(y'|x)dy' \\ \text{or} & \int_{y}^{\infty} p(y'|x)dy'. \end{cases} \tag{2.8}$$

In our applications we try to realize the solution with a network with parameters $\boldsymbol{\alpha}$ that maximize the entropy $H(X, V)$. We learn from equation 2.2 that this is equivalent to maximizing $E[\ln J]$. Using the stochastic gradient ascent method, this can be done through a learning algorithm

$$\Delta\boldsymbol{\alpha} \propto \frac{\partial \ln J}{\partial \boldsymbol{\alpha}} = J^{-1}\frac{\partial J}{\partial \boldsymbol{\alpha}} = \left(\frac{\partial v}{\partial y}\right)^{-1} \frac{\partial^2 v}{\partial y \partial \boldsymbol{\alpha}}. \tag{2.9}$$

When training is completed, we can use its result to estimate the desired conditional distribution function by evaluating

$$\left|\frac{\partial v}{\partial y}(x, y)\right| = p(y|x), \tag{2.10}$$

which is the first step of our task. In other words, by performing the encoding step, which is the construction of the uniformly distributed variable V, we have solved the problem of conditional distribution function estimation. A potential pitfall that deserves attention is the one-to-one assumption of the function v, that is, $J \neq 0$ over the domain of y. This, however, seems quite natural given that two conditions hold: the domain of nonzero probability is connected, and $J \neq 0$ over this domain for the initial estimate of $v(t = 0)$. If gradient ascent is performed sufficiently slowly, in a "quasi-static" limit, such that equation 2.2 always holds, it follows from the last integral in equation 2.2 that v remains one-to-one. If it does not, the integral diverges to $-\infty$, contradicting the fact that it should monotonically increase when climbing along the gradient.

Of course, the above discussion holds only in the limit where the number of samples approaches infinity. For any finite sample number, the discrete analog of equation 2.2, developed in the next section in the form of equation 3.4, may increase even as v ceases to be one-to-one, regardless of the step size. In addition, the optimization space may be very complex so as to result in the finding of a local maximum, potentially entailing a poor estimation of the density function. We have not yet answered these questions, which concern the practicality of the ME method.

Next we turn to the decoding step, or the generative step of our algorithm. We wish to construct a network that can generate events distributed according to $p(y|x)$ for a given x. Using the output of the encoding step, $(x, y) \to (x, v)$, we train a second network to reproduce the original inputs $(x, v) \to (x, y)$. The training of this secondary net follows standard supervised learning. Once the second network is trained on the data set, it can be used in a generative mode: given a value of x, one draws randomly a value of v that is uniformly distributed $v \sim U(0, 1)$, and uses it as an input to this network. The output y will be appropriately distributed according to $p(y|x)$. That this can be done follows from the fact that the solution $v(x, y)$ to the encoding phase is a monotonic function of y. To be specific, we may stick to the monotonically increasing solution. It is then invertible; a solution $y = g(x, v)$ is well defined.

Figure 1 presents the general schematic structure of the two steps of our method. Using real variables (x, y), one can invoke naturally the gradient ascent algorithm for learning $0 \leq v(x, y) \leq 1$ on a feedforward neural network. The decoding step may require first performing an unsquashing transformation on v, before employing it as an input, together with x, in a standard backpropagation or any other supervised learning procedure.

The same kind of formalism can be applied to the more complicated problem of higher-dimensional probability distributions, when x and y are replaced by vector variables of arbitrary dimensions. The general solution to the estimation problem of a conditional distribution function is given by

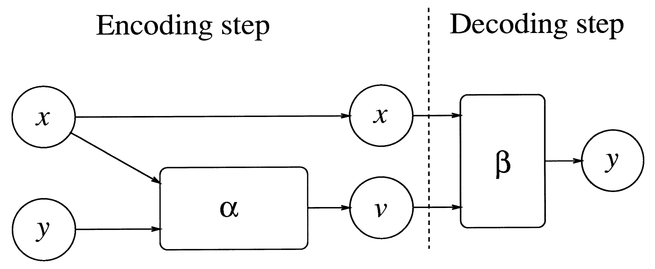

Figure 1: A schematic representation of the encoding and decoding steps.

the relation

$$p(\mathbf{y}|\mathbf{x}) = \left| \frac{\partial \mathbf{v}}{\partial \mathbf{y}}(\mathbf{x}, \mathbf{y}) \right|, \tag{2.11}$$

where the auxiliary vector variable **v** is of the same dimension as **y** which may be different from that of **x**. The proof is outlined in appendix A. **v** is uniformly distributed in a unit cube and is independent of **x**.

It is straightforward to apply this algorithm to the simpler case of probability density function estimation for a single variable **y**. In this case, the variable **x** drops out from the formalism, but the logic of the encoding and decoding steps continues to be the same. Our derivation coincides in this case with the work of Roth and Baram (1996), but our implementation is different, allowing for a general neural network to define the functional dependence of **v** on **y**. We use this problem for an illustration of our method. Figures 2 and 3 depict two probability distributions used to generate samples for training the encoding step of our algorithm. The estimated distributions are shown, as well as histograms that were generated from the same samples as those used for the encoding. Figure 2 is a bimodal distribution tested with a (small) set of 50 samples only. The samples turn out to be asymmetrically generated, due to the small statistics. This is evident in the histogram and reflected in the ME estimation. The network employed in this case used two hidden units only; hence its structure is

$$v = \sigma \left(\sum_i W_i \sigma(w_i y - \theta_i) - \theta_0 \right), \tag{2.12}$$

with $i = 1, 2$. σ is the sigmoid function $\sigma(z) = (1 + e^{-\beta z})^{-1}$. In Figure 3 we look at a triangular distribution, this time tested with 200 samples. Correspondingly we use a more complex network, with four hidden units.

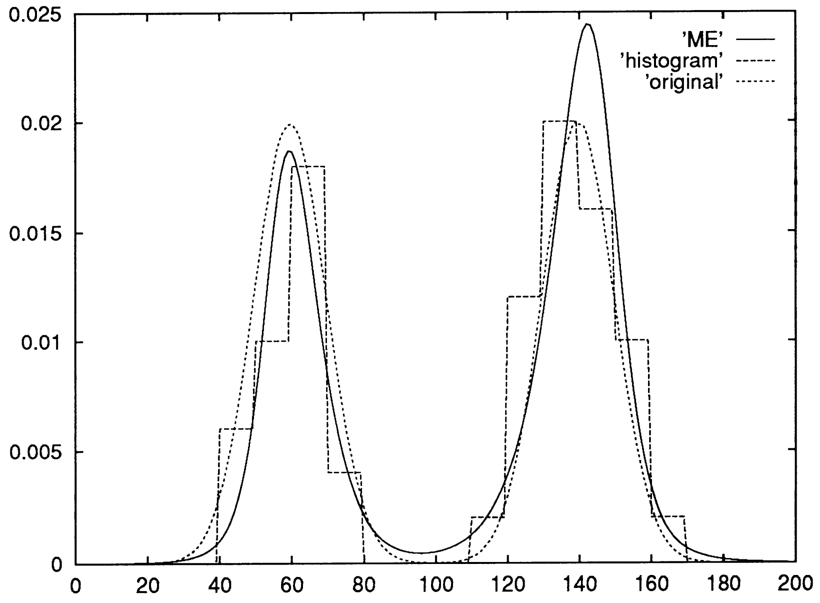

Figure 2: Comparison between the ME estimate (solid line) and the histogram of 50 samples drawn from a bimodal distribution (dotted line). The network subserving the ME encoding step contains two hidden units.

In general, we need a network whose complexity (number of parameters) is less than linear in the number of samples, for example, a square root of the latter. Clearly this is independent of the complexity of the distribution and its dimensionality.

3 Maximum Entropy versus Maximum Likelihood

It is instructive to draw a comparison between the ML method and the proposed ME one. A connection may be expected in view of previous results (Pearlmutter & Parra, 1996; MacKay, 1996; Cardoso, 1997) casting independent component analysis into the ML formalism.

Suppose we have a set $\chi = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\}$ of M statistically independent samples of some random variable $\mathbf{X}$, and that the distribution of $\mathbf{X}$ is assumed to belong to some known class of distributions with probability density functions $f(\mathbf{x}; \boldsymbol{\alpha})$. Let us denote by $\boldsymbol{\alpha}_0$ the true parameters of $\mathbf{X}$, that is, $f(\mathbf{x}; \boldsymbol{\alpha}_0) = p(\mathbf{x})$. According to the ML approach, we maximize the

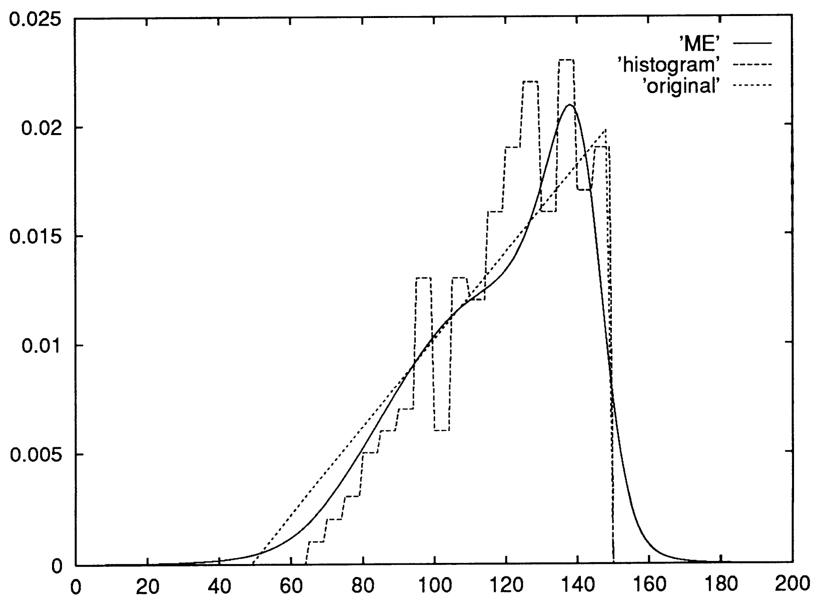

Figure 3: Comparison of the ME estimate and a histogram of 200 samples drawn from a triangular distribution. The larger sample size allows the use of a more complex network with four hidden units.

expression:

$$\mathcal{E} = p(\chi|\boldsymbol{\alpha}) = \prod_{i=1}^{M} p(\mathbf{x}_i|\boldsymbol{\alpha}) = \prod_{i=1}^{M} f(\mathbf{x}_i; \boldsymbol{\alpha}). \tag{3.1}$$

This is equivalent to maximizing

$$\frac{\ln \mathcal{E}}{M} = \frac{\sum_{i=1}^{M} \ln f(\mathbf{x}_i; \boldsymbol{\alpha})}{M} = \hat{E}[\ln f(\mathbf{x}; \boldsymbol{\alpha})], \tag{3.2}$$

where $\hat{E}$ denotes the estimated expectancy on an ensemble of size M. In the limit $M \to \infty$, this tends to

$$E[\ln f(\mathbf{x}; \boldsymbol{\alpha})] = \int f(\mathbf{x}; \boldsymbol{\alpha}_0) \ln f(\mathbf{x}; \boldsymbol{\alpha}) d\mathbf{x}. \tag{3.3}$$

Since the last integral is maximal for $\boldsymbol{\alpha} = \boldsymbol{\alpha}_0$, ML is a consistent criterion, that is, when the number of samples approaches infinity, finding the ML solution yields the correct distribution $f(\mathbf{x}; \boldsymbol{\alpha}_0)$. The ME algorithm can be cast into a similar form. Starting with the auxiliary variable $\mathbf{v}$, which is of the same dimension as the input variable $\mathbf{x}$ that it is to encode, we redefine our problem as searching for a one-to-one function in the family $\{\mathbf{v}(\mathbf{x}) \mid 0 \leq v_i(\mathbf{x}) \leq 1 \quad \forall \mathbf{x} \in \mathbf{R}^n,\ 1 \leq i \leq n\}$ that maximizes a cost function

$$\mathcal{H} = \prod_{i=1}^{M} \left| \frac{\partial \mathbf{v}}{\partial \mathbf{x}}(\mathbf{x}_i; \boldsymbol{\alpha}) \right|, \tag{3.4}$$

which will ensure maximization of the entropy. We then seek an $\boldsymbol{\alpha}_0$ for which $\mathcal{H}$ is maximal. As before, this is equivalent to maximizing

$$\frac{\ln \mathcal{H}}{M} = \frac{\sum_{i=1}^{M} \ln \left| \frac{\partial \mathbf{v}}{\partial \mathbf{x}}(\mathbf{x}_i; \boldsymbol{\alpha}) \right|}{M} = \hat{E}\left[\ln \left| \frac{\partial \mathbf{v}}{\partial \mathbf{x}}(\mathbf{x}; \boldsymbol{\alpha}) \right| \right], \tag{3.5}$$

which, in the limit $M \to \infty$, leads to maximization of

$$E\left[\ln \left| \frac{\partial \mathbf{v}}{\partial \mathbf{x}}(\mathbf{x}; \boldsymbol{\alpha}) \right| \right] = \int p(\mathbf{x}) \ln \left| \frac{\partial \mathbf{v}}{\partial \mathbf{x}}(\mathbf{x}; \boldsymbol{\alpha}) \right| d\mathbf{x}. \tag{3.6}$$

We show in appendix B that $\mathcal{H}$ maximization is obtained when

$$\left| \frac{\partial \mathbf{v}}{\partial \mathbf{x}} \right| = p(\mathbf{x}). \tag{3.7}$$

Thus we find that $\mathbf{v}$ that maximizes $\mathcal{H}$ also maximizes the entropy of the auxiliary variable,

$$H(\mathbf{V}) = H(\mathbf{X}) + E\left[\ln \left| \frac{\partial \mathbf{v}}{\partial \mathbf{x}} \right| \right]. \tag{3.8}$$

This derivation shows the relation between the ME and ML approaches: The space of possible $\left|\frac{\partial \mathbf{v}}{\partial \mathbf{x}}\right|$ serves as the space of candidate functions over which the ME algorithm is operative. These functions are positive definite but are not necessarily properly normalized. The example of equation 2.12 can serve as an illustration. There are regimes in parameter space where v will not reach its upper limit of 1. In other words, our parameter space is not restricted to properly normalized functions. However, as training proceeds, $\left|\frac{\partial \mathbf{v}}{\partial \mathbf{x}}\right|$ turns into an appropriate probability distribution function, coinciding with an ML solution of the network that we employ. Hence ME is a realization of ML.

4 Functional Dependence Among Random Variables

We expect the entropy maximization algorithm to be a useful tool in many applications. In this section we wish to demonstrate how it can come in handy in solving problems where there is a known, or expected, dependence among some of the stochastic variables. We restrict ourselves to estimation in the sense of reproducing the desired random variable rather than finding its density function.

We will start with the single variable case: a relation of the type $Y = g(X)$ between two stochastic variables. We can think of three interesting cases:

1. *g is known, and samples of X are given. An estimate of Y is to be obtained.* This case is the simplest and is solved by the estimation of X, to which one applies $g(X)$.

2. *g is known and samples of Y are given. An estimate of X is sought.* In this case we employ the encoding stage on Y as usual, thus estimating $v(y)$. We then apply a second encoding step, maximizing the entropy of $v(g(w(u)))$, where u is drawn from the usual uniform distribution. The entropy of v is maximal when it is uniformly distributed. The entropy of v is maximal when it is uniformly distributed. By the way v was constructed, this means that $g(w) \sim Y$ from which $w(u) \sim X$.

3. *Samples of both X and Y are given in the form $\{x_i\}$ and $\{y_i\}$ (as opposed to $\{x_i, y_i\}$). An estimate of g is required.* Again we first use the encoding step on Y, thus determining $v(y)$. In the second encoding step, we maximize the entropy of $v(w(x))$ using the samples of X. When a uniform distribution of $v(w)$ is attained, $w \equiv g$.

Things get more interesting when the functional dependence of the random variables becomes more complicated. As an example, suppose that the relation $Y = g(X_1, X_2)$ holds between the three random variables with g known, and also X_1 and X_2 are known to be independent. This can be a model of a transmitted signal, where X_1 is the signal, X_2 is the noise, and Y is the corrupted received signal. Samples of X_1 and of Y are given, and the purpose is to estimate X_2.

We begin by encoding Y, thus obtaining $v(y)$. We then construct a network that has two inputs, x and u, and two outputs, a and b. The x input is taken from the X_1 samples and the u input is drawn uniformly. The outputs have the functional form $a = v(g(x, h(u)))$ and $b = w(x, u)$, with h and w functions realized by the network. When the entropy of (a, b) is maximized, it is uniformly distributed. In particular, a is uniformly distributed, and by the way v was constructed, we then have $g \sim Y$, implying that $h(u) \sim X_2$. Note that b was introduced here only for computational assistance; it is much easier to maximize the entropy of $(a(x, u), b(x, u))$ than that of $a(x, u)$ alone, because of the mismatch between the number of inputs and outputs.

5 Generating Artificial Data of a Given Distribution

As a further example of the use of entropy maximization, we address the question of generating random deviates (or artificial data) for a given distribution. The latter can be given in a nonnormalized fashion, as a complicated (nonnegative) function $f(\mathbf{x})$ over some domain. In that case we can proceed, in analogy with the analysis of section 3, with defining an integral to be maximized:

$$\mathcal{I} = \int f(\mathbf{x}) \ln \left| \frac{\partial \mathbf{v}}{\partial \mathbf{x}}(\mathbf{x}; \boldsymbol{\alpha}) \right| d\mathbf{x}. \tag{5.1}$$

The maximization procedure is carried out by gradient ascent on a neural network that implements the mapping $\mathbf{x} \rightarrow \mathbf{v}$, with $\mathbf{v}$ turning out to be uniformly distributed in a unit cube. In this process, the Jacobian of the transformation takes the shape of the desired distribution. Once this encoding step is completed, this time through numerical evaluation of $\partial\mathcal{I}/\partial\boldsymbol{\alpha}$ rather than through training by data, the decoding step is developed as before. The decoder becomes the desired generator of artificial data. This can be viewed as an alternative to existing methods, such as the rejection method (Press, Flannery, Teukolsky, & Vetterling, 1986). Its generality, as well as the fact that neural networks can serve as universal approximators, makes it a likely candidate for a new tool.

6 Discussion

The principle of maximal entropy, which leads to a uniform distribution of the new random variables, is the cornerstone of the mathematical structure of our model. We have seen that it may be viewed as closely related to the maximum likelihood approach. In a simple problem, like the one depicted in Figure 2, it is very similar to conventional realizations of ML, because its result is close to a sum of two gaussians. This is not the case for more complex problems. Even in the result displayed in Figure 3, the output of the network is quite different from the sum of four gaussians, which might be expected from the hidden layer. The reason lies in the nonlinear character of the output node of our network, which is a squashing function producing the bounded variable v.

We distinguished between two different steps of our model: the encoding and decoding ones. The encoding step produces the desired probability distribution, which can be read off its internal neural structure. The decoding phase is useful as a generative mode, such as the "sleep" phase of the Helmholtz machine (Dayan, Hinton, Neal, & Zemel, 1995). In our model, the two steps are completely separated. In particular, the encoding step, which is the analog of the "wake" phase of the Helmholtz machine, is completely independent of the decoding one. In the Helmholtz machine, the two modes are coupled. This is, of course, part of the elegance of that structure, whose motivation was the idea that the generative mode has to play an inherent role in the brain's capability of pattern recognition. Although in principle one could structure the Helmholtz machine to be closer to our algorithm, in our case the two networks performing encoding and decoding have very different architectures that are unrelated to one another. Our generative step is introduced only as a computational tool. It may be useful as a Monte Carlo generator of artificial data.

The traditional approach of using neural networks in the realm of probability distributions is to endow single neurons with stochastic transfer functions, as in the Boltzmann machine (Ackley, Hinton, & Sejnowski, 1985) and the Helmholtz machine (Dayan et al., 1995). Our model is based on a completely different point of view. Its neural network component is purely deterministic. The probabilistic character comes from interpreting some of the nodes, forming outputs of the encoding step and inputs of the decoding step, as random variables. This type of variable (Roth & Baram, 1996) emerges naturally in the blind separation algorithm of Bell and Sejnowski (1995) and can be viewed as a novel element in the formation of stochastic neural networks.

Appendix A: Entropy Maximization for Vector Variables

We prove here that the method presented in section 2 can be generalized to $\mathbf{x}$ and $\mathbf{y}$ of vector form. First, we notice that the discussion in section 2 is immediately generalizable to a vector $\mathbf{x}$ simply by substituting $\mathbf{x}$ for x. Its results therefore apply for the case $(\mathbf{x}, y)$ with $\mathbf{x}$ a vector and y a scalar. In particular, this means that there exists a function $v^0(\mathbf{x}, y)$ such that the uniformity and independence conditions (see equation 2.6) hold whence $H(\mathbf{x}, v^0) = H(\mathbf{x})$.

Now suppose we have a vector $\mathbf{y}$ (and accordingly a vector $\mathbf{v}$ of the same dimension, each of its components bounded between 0 and 1). By the argument above, it follows that there exists a function $v_1^0(\mathbf{x}, y_1)$ such that

$$H(\mathbf{x}, v_1^0) = H(\mathbf{x}). \tag{A.1}$$

Similarly, defining $\mathbf{x}' \equiv (\mathbf{x}, v_1^0(\mathbf{x}, y_1))$, there exists a function $v_2^0(\mathbf{x}', y_2)$ such that

$$H(\mathbf{x}, v_1^0, v_2^0) = H(\mathbf{x}', v_2^0) = H(\mathbf{x}') = H(\mathbf{x}, v_1^0) = H(\mathbf{x}). \tag{A.2}$$

This procedure may be iterated until finally we obtain a $\mathbf{v}^0$ such that

$$H(\mathbf{x}, \mathbf{v}^0) = H(\mathbf{x}). \tag{A.3}$$

However, as in the scalar case, the relation

$$H(\mathbf{x},\mathbf{v}) = H(\mathbf{x}) + H(\mathbf{v}) - I(\mathbf{x},\mathbf{v}) \le H(\mathbf{x}) \tag{A.4}$$

holds, with equality iff $\mathbf{v}$ is uniformly distributed and independent of $\mathbf{x}$. We deduce that $\mathbf{v}^0$ satisfies these conditions, and from now on refer as $\mathbf{v}^0$ to *any* $\mathbf{v}$ for which an equality in equation A.4 holds. Indeed, the twofold degeneracy of section 2 is now inflated to at least $2^n(n!)^2$-fold degeneracy. However, it is not the uniqueness of $\mathbf{v}^0$ which is of interest to us but rather the uniqueness of $|\partial\mathbf{v}^0/\partial\mathbf{y}|$, as will become clear shortly.

Having the vector analog of equation 2.3, we can continue as in section 2 to obtain

$$p(\mathbf{y}|\mathbf{x}) = \left|\frac{\partial\mathbf{v}^0}{\partial\mathbf{y}}(\mathbf{x},\mathbf{y})\right|. \tag{A.5}$$

Using equation A.5, for any $(\mathbf{x},\mathbf{y})$ such that $p(\mathbf{x},\mathbf{y}) > 0$, we have

$$\left|\frac{\partial\mathbf{v}^0}{\partial\mathbf{y}}(\mathbf{x},\mathbf{y})\right| = p(\mathbf{y}|\mathbf{x}) > 0, \tag{A.6}$$

and $\mathbf{v}^0(\mathbf{x},\mathbf{y})$ is therefore invertible in the domain of nonzero probability to the form $\mathbf{y}^0(\mathbf{x},\mathbf{v})$.

Finally, for a given $\mathbf{x}^0$ choose $\mathbf{v} \sim U^n(0,1)$ and define $\mathbf{g}^0(\mathbf{v}) \equiv \mathbf{y}^0(\mathbf{x}^0,\mathbf{v})$. For the distribution of $\mathbf{g}^0$ we have, using equation A.5,

$$p(\mathbf{g}^0) = \frac{p(\mathbf{v})}{\left|\frac{\partial\mathbf{g}^0}{\partial\mathbf{v}}\right|} = \left|\frac{\partial\mathbf{v}}{\partial\mathbf{g}^0}\right| = \left|\frac{\partial\mathbf{v}}{\partial\mathbf{y}^0}(\mathbf{x}^0,\mathbf{y}^0)\right| = p(\mathbf{y}^0|\mathbf{x}^0), \tag{A.7}$$

which shows that $\mathbf{G}^0 \equiv \mathbf{Y}|\mathbf{X} = \mathbf{x}^0$.

It is evident from the above derivation that in order to achieve a uniform independent distribution, it suffices for every output element v_i to be connected to $\mathbf{x}$ and to $(v_1,\ldots,v_{i-1})$. This triangular formation was advocated by Roth and Baram (1996). It has the advantage of having a smaller number of parameters, but is also more constrained; its degeneracy is much reduced, making the global maximum potentially harder to attain.

Appendix B: Proof of ME Consistency

We wish to prove here that the ME criterion is consistent—that is, when the number of samples approaches infinity, the correct distribution is obtained. As explained in section 3, its consistency relies on the following theorem:

Theorem 1. *Define* $\mathcal{B} = \{\mathbf{g}(\mathbf{x}) \mid \mathbf{g}(\mathbf{x})$ *is one-to-one and* $0 \le g_i(\mathbf{x}) \le 1 \quad \forall\mathbf{x} \in \mathbf{R}^n,\ 1 \le i \le n\}$ *and let* $f(\mathbf{x})$ *be a probability density function. If* $\mathbf{g}(\mathbf{x}) \in \mathcal{B}$ *maximizes* $\int f(\mathbf{x}) \ln\left|\frac{\partial\mathbf{g}}{\partial\mathbf{x}}(\mathbf{x})\right| d\mathbf{x}$ *then* $\left|\frac{\partial\mathbf{g}}{\partial\mathbf{x}}(\mathbf{x})\right| \equiv f(\mathbf{x})$.

Proof. Anyone who has read appendix A will not be surprised at what is coming, since this is just a different way of stating the same thing.
Define

$$\hat{g}_1(x_1) \equiv F_{X_1}(x_1) \equiv \int_{-\infty}^{x_1} dx_1' \int_{-\infty}^{\infty} dx_2' \dots \int_{-\infty}^{\infty} dx_n' f(x_1', x_2', \dots, x_n'), \tag{A.8}$$

$$\begin{aligned}\hat{g}_2(x_1, x_2) &\equiv F_{X_2|X_1}(x_1, x_2)\\ &\equiv \int_{-\infty}^{x_2} dx_2' \int_{-\infty}^{\infty} dx_3' \dots \int_{-\infty}^{\infty} dx_n' f(x_1, x_2', \dots, x_n')/f_{X_1}(x_1),\end{aligned} \tag{A.9}$$

$$\hat{g}_3(x_1, x_2, x_3) \equiv F_{X_3|X_1,X_2}(x_1, x_2, x_3), \tag{A.10}$$

and so forth, where $f_{X_1}(x_1)$ is the marginal probability distribution of X_1.

Then $\partial\hat{g}_i/\partial x_i = f(x_i|x_1, \dots, x_{i-1})$ and for every $i < j$, $\partial\hat{g}_i/\partial x_j = 0$. The matrix $\partial\hat{g}_i/\partial x_j$ is therefore triangular, so that $\partial\hat{\mathbf{g}}/\partial\mathbf{x} = \prod_{i=1}^n \partial\hat{g}_i/\partial x_i = \prod_{i=1}^n f(x_i|x_1, \dots, x_{i-1}) = f(\mathbf{x})$. From the construction of $\hat{g}(\mathbf{x})$, it is clear that $\hat{\mathbf{g}}(\mathbf{x}) \in \mathcal{B}$. This proves the attainability of the limit. To complete the proof, note first that since $\mathbf{g}$ is one-to-one, $\partial\mathbf{g}/\partial\mathbf{x}$ can be assumed to be positive without loss of generality. Thus, using the inequality $\ln x \le x - 1$,

$$\begin{aligned}\int f(\mathbf{x}) \ln \frac{\partial\mathbf{g}}{\partial\mathbf{x}} d\mathbf{x} - \int f(\mathbf{x}) \ln f(\mathbf{x}) d\mathbf{x} &= \int f(\mathbf{x}) \ln \frac{\frac{\partial\mathbf{g}}{\partial\mathbf{x}}}{f(\mathbf{x})} d\mathbf{x}\\ &\le \int f(\mathbf{x}) \left(\frac{\frac{\partial\mathbf{g}}{\partial\mathbf{x}}}{f(\mathbf{x})} - 1 \right) d\mathbf{x} =\end{aligned} \tag{A.11}$$

$$\int \frac{\partial\mathbf{g}}{\partial\mathbf{x}} d\mathbf{x} - \int f(\mathbf{x}) d\mathbf{x} = \int d\mathbf{g} - \int f(\mathbf{x}) d\mathbf{x} \le 1 - 1 = 0, \tag{A.12}$$

with equality holding iff $\partial\mathbf{g}/\partial\mathbf{x} \equiv f$.

Acknowledgments

We thank H. Abramowicz, S. Ackerman, I. Meilijson, B. Pearlmutter, E. Ruppin, and S. Solla for helpful discussions.

References

Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cog. Sci.*, *9*, 147–169.

Bell, A. J., & Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Comp., 7*, 1129–1159.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford: Oxford University Press.

Cardoso, J.-F. (1997). Infomax and maximum likelihood for source separation. *IEEE Letters on Signal Processing, 4*, 112–114.

Cover, T. M., & Thomas, J. A. (1991). *Elements of information theory*. New York: Wiley.

Dayan, P., Hinton, G. E., Neal, R. M., & Zemel, R. S. (1995). The Helmholtz machine. *Neural Comp., 7*, 889–904.

MacKay, D. J. C. (1996). *Maximum likelihood and covariant algorithms for independent component analysis*. Unpublished manuscript.

Nadal, J.-P., & Parga, N. (1994). Nonlinear neurons in the low noise limit: a factorial code maximizes information transfer. *Network, 5*, 565–581.

Pearlmutter, B. A. & Parra, L. C. (1996). A context-sensitive generalization of ICA. *Int. Conf. on Neural Network Processing.* Hong Kong.

Press, W. H., Flannery, B. P., Teukolsky, S. A., & Vetterling, W. T. (1986). *Numerical recipes*. Cambridge: Cambridge University Press.

Roth, Z., & Baram, Y. (1996). Multidimensional density shaping by sigmoids. *IEEE Trans. on Neural Networks, 7*, 1291–1298.

Received May 8, 1997; accepted April 2, 1998.