LETTER Communicated by Donald Geman

A Learning Theorem for Networks at Detailed Stochastic Equilibrium

Javier R. Movellan
Cognitive Science Department, University of California at San Diego, La Jolla, CA 92093, U.S.A.

This article analyzes learning in continuous stochastic neural networks defined by stochastic differential equations (SDE). In particular, it studies gradient descent learning rules to train the equilibrium solutions of these networks. A theorem is given that specifies sufficient conditions for the gradient descent learning rules to be local covariance statistics between two random variables: (1) an evaluator that is the same for all the network parameters and (2) a system variable that is independent of the learning objective. While this article focuses on continuous stochastic neural networks, the theorem applies to any other system with Boltzmann-like equilibrium distributions. The generality of the theorem suggests that instead of suppressing noise present in physical devices, a natural alternative is to use it to simplify the credit assignment problem. In deterministic networks, credit assignment requires an evaluation signal that is different for each node in the network. Surprisingly, when noise is not suppressed, all that is needed is an evaluator that is the same for the entire network and a local Hebbian signal. This modularization of signals greatly simplifies hardware and software implementations. The article shows how the theorem applies to four different learning objectives that span supervised, reinforcement, and unsupervised problems: (1) regression, (2) density estimation, (3) risk minimization, and (4) information maximization. Simulations, implementation issues, and implications for computational neuroscience are discussed.

1 Introduction

This article studies how to train equilibrium solutions of continuous stochastic neural networks. The networks proposed are specified by stochastic differential equations (SDE), an extension of ordinary differential equations that incorporates probabilistic dynamics. The article illustrates how these networks can be optimized so that their equilibrium distributions exhibit desired properties.

From a more general point of view, the article is also a theoretical analysis of learning in systems that exhibit Boltzmann equilibrium distributions, regardless of whether this distribution is obtained using SDE models or

 © 1998 Massachusetts Institute of Technology

other methods (see Neal, 1993, and Gidas, 1986, for a review of methods to generate Boltzmann-type equilibrium distributions). As such, the article generalizes the original Boltzmann machine learning algorithm (Ackley, Hinton, & Sejnowski, 1985) to a very wide variety of architectures and learning criteria.

The article proposes sufficient conditions for the gradients of the cost functions minimized during learning to be local covariance statistics between two random variables: (1) an evaluator variable that is the same for all the network parameters and (2) a local system variable that is independent of the particular cost function being minimized. This factorization avoids backpropagation of error signals specific to each network parameter, greatly simplifying hardware and software implementations. The analysis presented here suggests that probabilistic dynamics may play an integral part of learning in natural nervous systems by simplifying the solution to the credit assignment problem.

Learning with stochastic networks has played an important theoretical role in the neural network literature (Cowan, 1968; Geman & Geman, 1984; Ackley et al., 1985; Smolensky, 1986), but curiously, even though most neural network applications use continuous representations, learning in the continuous stochastic case has seldom been studied. Analyzing the continuous stochastic case is important for the following reasons:

- Many natural signals, like pixel gray-level object positions and orientations, are well described as continuous random processes. Experience shows that some practical applications benefit from the use of a continuous stochastic framework (Isard & Blake, 1996).

- Randomness is essential when modeling natural computation because of the intrinsic variability of natural hardware. It has been proposed that a unified theory of cognition and neural computation should be based on models that are random, continuous, and interactive (McClelland, 1993).

- Current digital technology suppresses structural and thermodynamic noise in hardware devices by creating high-energy barriers between states, thus requiring relatively high power supplies (Andreou, 1994; Landauer, 1992; Mead & Conway, 1980). This approach does not work with supplies on the order of 0.1 volt, characteristic of natural computers. As we move toward low-voltage systems, the computing environment becomes analog and stochastic, the realm of stochastic diffusion models.

- There are existing VLSI implementations of continuous stochastic neural networks, and it is important to have a formal framework to understand the kind of things that we can do with them (Alspector, Jayakumar, & Luna, 1992).

This article focuses on the problem of optimizing networks for tasks where what matters is the stable distribution of responses (e.g., image completion) rather than the paths leading to those responses. For generality, the framework and the results are presented in an abstract manner. For concreteness, we show how to construct and train an SDE version of the continuous Hopfield model (Hopfield, 1984).

Sections 2 and 3 introduce SDEs and discuss analytical solutions for their equilibrium distributions. Section 4 presents a theorem applicable to a general class of models and cost functions. When this theorem applies, gradient descent learning rules take the form of local covariance statistics between evaluators and system variables. Section 5 applies the theorem to derive evaluator variables for the following supervised, reinforcement and unsupervised problems: (1) regression, (2) density estimation, (3) risk minimization, and (4) information maximization. The theorem applies to any dynamical system that exhibits Boltzmann equilibrium distributions; however, the main focus of this article is on the application of the theorem to neural networks specified via SDEs. As such, section 6 derives the system variables for a stochastic version of the continuous Hopfield model. Sections 7 and 8 present simulations and discuss implications of this approach.

2 Introduction to SDEs

The theory of SDEs is a well-known formalism for describing continuous strong Markov processes (Karatzas & Shreve, 1988, p. 81). SDEs are commonly used to model the effects of noise in electric circuits, computer networks, and control systems (Oksendal, 1992; Borkar, 1989). In the cognitive modeling literature, SDEs have been used to model human reaction time distributions (Ratcliff, 1979) and to illustrate the principles of random, graded, and interactive propagation of information (McClelland, 1993). In the neural network literature, SDE models have been used explicitly or implicitly to describe single neuron activity (Gerstein & Mandelbrot, 1964; Ricciardi, 1977; Hanson & Tuckwell, 1983), small pools of neurons (Matsuyama, Shirai & Akizuki, 1974) and noisy neural networks (Zipser, 1991; Ohira & Cowan, 1995).

The solutions of SDE equations are known as "diffusion processes" because they can be thought of as the mathematical description of the motion of small particles in a moving fluid (Karatzas & Shreve, 1988; Oksendal, 1992). Hereafter we will refer to the general class of neural network models specified by SDEs as diffusion networks. From a formal point of view, diffusion networks are continuous hidden Markov models (HMM). In discrete-state HMMs, standard in current automatic speech recognition systems (Rabiner, 1989), the system dynamics are explicitly defined by a matrix of state transition probabilities and a matrix of output probabilities conditional on each hidden state. In diffusion networks, the state and observation probabilities are implicitly defined by a drift function, which is the deterministic kernel

of the system (e.g., as for a deterministic neural network) and a dispersion function, which controls the level of uncertainty in the system. More specifically, diffusion networks are specified by the following SDE

$$dY_t^\lambda = \mu(\lambda, Y_t^\lambda, X)\, dt + \sigma(Y_t^\lambda)\, dW_t\,, \tag{2.1}$$

$$Y_0 = \nu\,, \tag{2.2}$$

where

$\{Y_t^\lambda;\ \lambda \in \mathbb{R}^p\}_{t\in[0,\infty)}$ is a continuous $\mathbb{R}^n$-valued random process representing the state of the n nodes in the network.

X is an $\mathbb{R}^m$ valued random vector representing the input. It has a known probability density f_X fixed by the environment.

ν is an $\mathbb{R}^n$ valued random variable representing the initial conditions.

σ: $\mathbb{R}^n \to \mathbb{R}^n \otimes \mathbb{R}^n$ is a matrix function called the dispersion.

$\lambda \in \mathbb{R}^p$ is a vector of adaptive parameters (e.g., coupling weights between units).

μ: $\mathbb{R}^p \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a function known as the drift. The drift can be interpreted as the deterministic kernel of a neural network with vector parameter λ (see section 6).

$\{W_t\}_{t\in[0,\infty)}$ is an N-dimensional Wiener process, a mathematical model of Brownian motion (Papoulis, 1991, p. 346). The process is assumed independent of $\{Y_0^\lambda\}_{\lambda\in\mathbb{R}^p}$ and X.

To simplify the presentation, we will assume that our formal objects are mathematically well behaved. For example, if we take a partial derivative of a function, we implicitly assume that such a derivative exists.

3 Stochastic Equilibrium

This article focuses on the probability densities of diffusion networks at stochastic equilibrium, the densities in $\mathbb{R}^n$ induced by the random variables

$$\{Y_\infty^\lambda \triangleq \lim_{t\to\infty} Y_t^\lambda\}_{\lambda\in\mathbb{R}^p}, \tag{3.1}$$

where $\triangleq$ stands for "defined." Under conditions to be detailed later, such a limit exists in distribution and is independent of the initial conditions. In this article, we care about the diffusion dynamics only to the extent that they lead to an equilibrium solution; thus we will ease the notation by dropping the time index in the underlying process. For example, Y^λ will stand for Y_∞^λ, and Y_i^λ for its ith component. Moreover, we will denote the space of continuous densities on R^n w.r.t the Lebesgue measure as $\mathcal{D}_n$. Let $f_{Y|X}^\lambda$ represent the

density induced by Y^λ in response to input X. It is well known that this density uniquely satisfies the following equilibrium condition:[1]

$$\nabla_y \cdot J^\lambda(y, x) = 0 , \tag{3.2}$$

$$J^\lambda(y, x) \triangleq f^\lambda_{Y|X}(y \mid x)\, V^\lambda(y, x) , \tag{3.3}$$

$$V^\lambda_i(y, x) \triangleq \mu_i(\lambda, y, x) - \frac{1}{2}\sum_{j=1}^{n} \sigma^2_{i,j}(y) \frac{\partial}{\partial y_j} \log\Big(f^\lambda_{Y|X}(y \mid x)\, \sigma^2_{i,j}(y)\Big), \tag{3.4}$$

$$\sigma^2_{i,j}(y) \triangleq \Big(\sigma(y)^T \sigma(y)\Big)_{i,j} , \tag{3.5}$$

where J^λ: $\mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the equilibrium current, V^λ: $\mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ the equilibrium velocity, $\nabla_y \cdot$ the divergence with respect to y, and ∇_y the gradient with respect to y.

For the purposes of this article, a diffusion network with parameter λ is a deterministic mapping $\mathcal{N}_\lambda$: $\mathbb{R}^m \to \mathcal{D}_n$, from input space into the space of densities on $\mathbb{R}^n$. Equation 3.2 defines the mapping implicitly. Making it explicit is difficult in general; however, there is a special case with a well-known solution, the focus of this article. In particular, let $\beta > 0$ such that $\sigma(y) = \sqrt{\frac{2}{\beta}} I_n$, where I_n is the $n \times n$ identity matrix, and let $\{U$: $\mathbb{R}^p \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}\}$ be a function such that

$$\mu(\lambda, y, x) = -\nabla_y U(\lambda, y, x) . \tag{3.6}$$

In this case it is easy to verify that the Boltzmann density

$$f^\lambda_{Y|X}(y \mid x) = \frac{1}{Z^\lambda(x)} \exp\,[\, -\beta\, U(\lambda, y, x)\,] , \tag{3.7}$$

$$Z^\lambda(x) \triangleq \int_{\mathbb{R}^n} \exp\,[\, -\beta\, U(\lambda, y, x)\,] dy, \tag{3.8}$$

satisfies equation 3.2. For equation 3.7 to be well defined, the integral in equation 3.8 needs to exist. Gidas (1986, p. 190) and Geman and Hwang (1986) specify sufficient conditions on $U(\lambda, y, x)$ for this. Note that in fact equation 3.7 makes the velocity $V^\lambda(y, x)$ zero everywhere, a condition sufficient but not necessary to satisfy equation 3.2 and which is known as detailed balance (Poggio & Girosi, 1994). The function $U(\lambda, y, x)$ is sometimes known as the potential, the energy, or the dissonance. When its sign is changed, it is known as the harmony (Smolensky, 1986).

[1] This condition easily follows from the Kolmogorov forward equation (Oksendal, 1992, p. 127), assuming nonzero equilibrium density everywhere and positive definite dispersion. This simple transformation of the forward Kolmogorov equation ties up nicely with mathematical physics and emphasizes the idea of probability currents moving throughout the state-space.

4 Learning

Equation 3.7 defines a family of mappings $\{\mathcal{N}_\lambda\colon \mathbb{R}^m \to \mathcal{D}_n\}_{\lambda\in\mathbb{R}^p}$, and equation 2.1 tells us how to implement these mappings using diffusion networks. The learning task is that of finding members of the family that are optimal in some specific sense. In general we may care about only the distribution of a subset of $d < n$ system variables, which we call the observables. In such a case, we divide the state random variables Y^λ into observable and hidden components: $Y^\lambda = (O^\lambda, H^\lambda)$.

- $\{O^\lambda\colon \Omega \to \mathbb{R}^d\}_{\lambda\in\mathbb{R}^p}$ is called the observable component.
- $\{H^\lambda\colon \Omega \to \mathbb{R}^{n-d}\}_{\lambda\in\mathbb{R}^p}$ is called the hidden component.

Hidden variables are important to allow the marginal density of observables to be non-Boltzmann. In this section we propose a learning theorem that applies to two different cases: one typical in the neural network literature and the other in the stochastic filtering literature.

Case 1: The joint state process (i.e., the observable process and the hidden processes) is defined by the SDE

$$dY_t^\lambda = -\nabla_y U(\lambda, Y_t^\lambda, X)dt + \sqrt{\frac{2}{\beta}}\, dW_t\ , \tag{4.1}$$

with an input density f_X fixed by the environment. For each input x and parameter vector λ, the joint state equilibrium distribution is Boltzmann, as in equation 3.7.

Case 2: In this case, the hidden process is defined by the SDE

$$dH_t^\lambda = -\nabla_h U(\lambda, H_t^\lambda, X)dt + \sqrt{\frac{2}{\beta}}\, dW_t\ , \tag{4.2}$$

with $U^\lambda\colon \mathbb{R}^{n-d} \times R^m \to \mathbb{R}$. As in case 1, the input density is fixed by the environment. Moreover, the equilibrium densities of observables conditioned on inputs and hidden states[2] $f_{O|H,X}$ do not depend on λ. Note in case 2 that the drift for the hidden process does not depend on the observable process. From a neural net point of view, this says that there is no feedback connection from the observables back into the hidden nodes, a classic constraint in stochastic filtering (Oksendal, 1992, p. 58)

Let $\{(\Omega, \mathcal{F}, m^\lambda)\}_{\lambda\in\mathbb{R}^p}$ be an indexed family of probability spaces where $\Omega = \mathbb{R}^d \times \mathbb{R}^{n-d} \times \mathbb{R}^m$, $\mathcal{F} = \mathcal{B}(\Omega)$, the Borel sigma-algebra of Ω. The probability measures m^λ are defined by densities induced by the random variables

[2] These conditional densities could be optimized, but that is not the focus of this article.

$\{O^\lambda, H^\lambda\}_{\lambda\in\mathbb{R}^p}$ and X. We represent these density functions with the symbol f and appropriate superscript and subscripts.

Next, we define a series of random variables needed in the derivation of the learning theorem. We define these random variables in terms of auxiliary functions whose only role is to facilitate the derivations by making explicit dependencies between different variables.

- Let $\{S_i^\lambda,\ i = 1, \ldots, p\}_{\lambda\in\mathbb{R}^p}$, be a family of random variables named the system covariates and defined by auxiliary functions $\{\tilde{S}_i^\lambda\colon \mathbb{R}^d \times \mathbb{R}^{n-d} \times \mathbb{R}^m \to \mathbb{R}\}_{\lambda\in\mathbb{R}^p}$,

$$S_i^\lambda \triangleq \tilde{S}_i^\lambda \left(O^\lambda, H^\lambda, X\right) , \tag{4.3}$$

$$\tilde{S}_i^\lambda(o, h, x) \triangleq -\frac{\partial U(\lambda, o, h, x)}{\partial \lambda_i}. \tag{4.4}$$

- Let $\{Q^\lambda\colon \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}\}_{\lambda\in\mathbb{R}^p}$ be a family of functions called the state cost and defined by auxiliary function $\tilde{Q}\colon \mathbb{R}^d \times \mathbb{R}^m \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$,

$$Q^\lambda(u, v) \triangleq \tilde{Q}\left(u, v, f_O^\lambda(u) , f_{O|X}^\lambda(u \mid v)\right) . \tag{4.5}$$

- Let $\{R^\lambda\}_{\lambda\in\mathbb{R}^p}$ be a family of random variables named the evaluators, defined by auxiliary functions $\{\tilde{R}^\lambda\colon \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}\}_{\lambda\in\mathbb{R}^p}$,

$$R^\lambda \triangleq \tilde{R}^\lambda (O, X) , \tag{4.6}$$

$$\tilde{R}^\lambda(u, v) \triangleq \frac{\partial \tilde{Q}\left(u, v, f_O^\lambda(u) , f_{O|X}^\lambda(u \mid v)\right)}{\partial f_{O|X}^\lambda(u \mid v)} . \tag{4.7}$$

- Let $\{\tilde{T}^\lambda\colon \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}\}_{\lambda\in\mathbb{R}^p}$ be a family of functions defined as follows:

$$\tilde{T}^\lambda(u, v) \triangleq \frac{\partial \tilde{Q}\left(u, v, f_O^\lambda(u) , f_{O|X}^\lambda(u \mid v)\right)}{\partial f_O^\lambda(u)} . \tag{4.8}$$

- Let $C\colon \mathbb{R}^p \to \mathbb{R}$ be an overall cost function defined as

$$C(\lambda) \triangleq \int_{\mathbb{R}^m} dx\, g(x) \int_{\mathbb{R}^d} do\ Q^\lambda(o, x) , \tag{4.9}$$

where $g\colon \mathbb{R}^m \to \mathbb{R}$ is a function for which equation 4.9 is well defined

and such that

$$\nabla_o \int_{\mathbb{R}^m} dx\, g(x)\, \tilde{T}^\lambda(o, x) = \mathbf{0}\ . \tag{4.10}$$

This condition tells us that the integral in equation 4.10 ought to be a constant with respect to o, (i.e., its gradient with respect to o is the zero vector). The reasons for this condition will become apparent later. Our goal is to find values of λ that minimize $C(\lambda)$ by using gradient-descent approaches. We will show that these gradients are linear combinations of covariance statistics. We refer to this fact as the Boltzmann covariance theorem (BCT).

Boltzmann covariance theorem. *For systems defined as in cases 1 and 2, and satisfying equation 4.10, the gradient of the overall cost with respect to the network parameters is a linear combination of covariance statistics between the evaluator and the system covariates, and it has the following form:*

$$\frac{\partial C(\lambda)}{\partial \lambda_i} = \beta \int_{\mathbb{R}^m} dx\, g(x)\, \mathrm{Cov}^\lambda(R^\lambda, S_i^\lambda \mid X = x)\ , \tag{4.11}$$

where Cov^λ *is a covariance with respect to measure* m^λ.

Proof. The proof consists of two steps. In the first step, we study the gradients of output probabilities. In the second step, we study the gradient of the cost function.

Step 1. We explore two different cases as defined at the beginning of this section.

In case 1, the joint states have Boltzmann density

$$f^\lambda_{O,H|X}(o, h \mid x) = \frac{1}{Z^\lambda(x)} \exp\,[\ -\beta\ U(\lambda, o, h, x)\ ]. \tag{4.12}$$

Thus,

$$\begin{aligned}\frac{\partial f^\lambda_{O,H|X}(o, h \mid x)}{\partial \lambda_i} &= \beta\ f^\lambda_{O,H|X}(o, h \mid x) \\ &\quad [\mathrm{E}^\lambda(S_i^\lambda \mid O^\lambda = o, H^\lambda = h, X = x) \\ &\quad -\ \mathrm{E}^\lambda(S_i^\lambda \mid X = x)]\ ,\end{aligned} \tag{4.13}$$

where E^λ represents expected values with respect to measure m^λ. Therefore,

$$\begin{aligned}\frac{\partial f^\lambda_{O|X}(o \mid x)}{\partial \lambda_i} &= \int_{\mathbb{R}^{n-d}} dh\ \frac{\partial f^\lambda_{O,H|X}(o, h \mid x)}{\partial \lambda_i} \\ &= \beta\ f^\lambda_{O|X}(o \mid x) \\ &\quad [\ \mathrm{E}^\lambda(S_i^\lambda \mid O^\lambda = o, X = x) - \mathrm{E}(S_i^\lambda \mid X = x)\ ].\end{aligned} \tag{4.14}$$

In case 2, the hidden states are Boltzmann and the outputs have a fixed conditional density model $f_{O|H,X}$. Thus,

$$f^{\lambda}_{H|X}(h \mid x) = \frac{1}{Z^{\lambda}(x)} \exp\left[-\beta\, U(\lambda, h, x) \right], \tag{4.15}$$

$$\frac{\partial f^{\lambda}_{H|X}(h \mid x)}{\partial \lambda_i} = \beta\, f^{\lambda}_{H|X}(h \mid x) \left[\mathrm{E}^{\lambda}(S^{\lambda}_i \mid H^{\lambda} = h, X = x) - \mathrm{E}^{\lambda}(S^{\lambda}_i \mid X = x) \right], \tag{4.16}$$

and

$$\frac{\partial f^{\lambda}_{O|X}(o \mid x)}{\partial \lambda_i} = \int_{\mathbb{R}^{n-d}} dh\; f^{\lambda}_{O|H,X}(o \mid h, x)\; \frac{\partial f^{\lambda}_{H|X}(h \mid x)}{\partial \lambda_i}$$

$$= \beta\, f^{\lambda}_{O|X}(o \mid x) \tag{4.17}$$

$$[\mathrm{E}^{\lambda}(S^{\lambda}_i \mid O^{\lambda} = o, X = x) - \mathrm{E}^{\lambda}(S^{\lambda}_i \mid X = x)], \tag{4.18}$$

which is equivalent to equation 4.14. Thus, from now on cases 1 and 2 behave identically.

Step 2. Applying the chain rule on equation 4.9,

$$\nabla_{\lambda} C(\lambda) = \int_{\mathbb{R}^m} dx\, g(x) \int_{\mathbb{R}^d} do\; \tilde{R}^{\lambda}(o, x) \nabla_{\lambda} f^{\lambda}_{O|X}(o \mid x) + \int_{\mathbb{R}^m} dx\, g(x) \int_{\mathbb{R}^d} do\; \tilde{T}^{\lambda}(o, x)\; \nabla_{\lambda} f^{\lambda}_{O}(o), \tag{4.19}$$

where $\tilde{R}^{\lambda}$ and $\tilde{T}^{\lambda}$ are defined in equations 4.7 and 4.8. The last term in equation 4.19 vanishes,

$$\int_{\mathbb{R}^m} dx\, g(x) \int_{\mathbb{R}^d} do\; \tilde{T}^{\lambda}(o, x)\; \nabla_{\lambda} f^{\lambda}_{O}(o) \tag{4.20}$$

$$= \int_{\mathbb{R}^d} do\, \nabla_{\lambda} f^{\lambda}_{O}(o) \int_{\mathbb{R}^m} dx\, g(x)\; \tilde{T}^{\lambda}(o, x) \tag{4.21}$$

$$\propto \int_{\mathbb{R}^d} do\, \nabla_{\lambda} f^{\lambda}_{O}(o) = \nabla_{\lambda} \int_{\mathbb{R}^d} dx f^{\lambda}_{O}(o) = \mathbf{0} \tag{4.22}$$

since $\int_{\mathbb{R}^d} do\; f^{\lambda}_{O}(o) = 1$ and we assume the equation 4.10 holds. Moreover, the components of the second term have the desired covariance form,

$$\frac{\partial C(\lambda)}{\partial \lambda_i} = \int_{\mathbb{R}^m} dx\, g(x) \int_{\mathbb{R}^d} do\; \tilde{R}^{\lambda}(o, x)\; \frac{\partial f^{\lambda}_{O|X}(o \mid x)}{\partial \lambda_i} \tag{4.23}$$

$$= \beta \int_{\mathbb{R}^m} dx\, g(x) \int_{\mathbb{R}^d} do\, \tilde{R}^\lambda(o, x) f^\lambda_{O|X}(o \mid x)$$
$$[\mathrm{E}^\lambda(S^\lambda_i \mid O^\lambda = o, X = x) - \mathrm{E}^\lambda(S^\lambda_i \mid X = x)] \tag{4.24}$$
$$= \beta \int_{\mathbb{R}^m} dx\, g(x) \int_{\mathbb{R}^d} do\, \tilde{R}^\lambda(o, x) f^\lambda_{O|X}(o \mid x)$$
$$\times \int_{\mathbb{R}^{n-d}} dh\, \tilde{S}^\lambda_i(o, h, x) f^\lambda_{O,H|X}(o, h \mid x)$$
$$- \beta \int_{\mathbb{R}^m} dx\, g(x)\, \mathrm{E}^\lambda(R^\lambda \mid X = x) \mathrm{E}^\lambda(S^\lambda_i \mid X = x) \tag{4.25}$$
$$= \beta \int_{\mathbb{R}^m} dx\, g(x)$$
$$[\mathrm{E}^\lambda(R^\lambda S^\lambda_i \mid X = x) - \mathrm{E}^\lambda(R^\lambda \mid X = x) \mathrm{E}^\lambda(S^\lambda_i \mid X = x)]$$
$$= \beta \int_{\mathbb{R}^m} dx\, g(x)\, \mathrm{Cov}^\lambda(R^\lambda, S^\lambda_i \mid X = x)\ . \tag{4.26}$$

Corollary. Let an overall cost function $\grave{C}$ have the form

$$\grave{C}(\lambda) \triangleq \int_{\mathbb{R}^m} dx\, g(x)\ q\,(x, C(\lambda, x))\ , \tag{4.27}$$
$$C(\lambda, x) \triangleq \int_{\mathbb{R}^d} do\, \tilde{Q}\left(o, x, f^\lambda_O\,(o)\,, f^\lambda_{O|X}\,(o \mid x)\right)\ , \tag{4.28}$$

where q: $\mathbb{R}^d \times \mathbb{R} \to \mathbb{R}$ is a well-behaved function and $\tilde{Q}$ is defined in equation 4.5, satisfying equation 4.10. In this case the gradient of the overall cost with respect to the network parameters is also a linear combination of covariance statistics between an evaluator and system random variables

$$\frac{\partial \grave{C}(\lambda)}{\partial \lambda_i} = \beta \int_{\mathbb{R}^m} dx\, g(x)\, \mathrm{Cov}^\lambda(\grave{R}^\lambda, S^\lambda_i \mid X = x)\ , \tag{4.29}$$
$$\grave{R}^\lambda(\omega) \triangleq R^\lambda(\omega) \frac{\partial q\,(x, C(\lambda, X(\omega))}{\partial C(\lambda, X(\omega))}\ ; \omega \in \Omega\,, \tag{4.30}$$

where R^λ is defined in equation 4.6.

Proof. Applying the chain rule,

$$\frac{\partial \grave{C}(\lambda)}{\partial \lambda_i} = \int_{\mathbb{R}^m} dx\, g(x) \frac{\partial q\,(x, C(\lambda, x))}{\partial C(\lambda, x)} \frac{\partial C(\lambda, x)}{\partial \lambda_i}\ . \tag{4.31}$$

Note that C can be expressed as an overall cost function,

$$C(\lambda, u) = \int_{\mathbb{R}^m} dx\, \delta(x - u) \int_{\mathbb{R}^d} do\, \tilde{Q}\left(o, x, f^\lambda_O\,(o)\,, f^\lambda_{O|X}\,(o \mid x)\right)\ , \tag{4.32}$$

where δ is the Dirac delta function. Thus, applying the BCT,

$$\frac{\partial C(\lambda, u)}{\partial \lambda_i} = \beta \int_{\mathbb{R}^m} dx\, \delta(x - u)\, \mathrm{Cov}^{\lambda}(R^{\lambda}, S_i^{\lambda} \mid X = x) \tag{4.33}$$

$$= \beta\, \mathrm{Cov}^{\lambda}(R^{\lambda}, S_i^{\lambda} \mid X = u)\,, \tag{4.34}$$

and

$$\frac{\partial \grave{C}(\lambda)}{\partial \lambda_i} = \beta \int_{\mathbb{R}^m} dx\, g(x) \frac{\partial q\,(x, C(\lambda, x))}{\partial C(\lambda, x)} \mathrm{Cov}^{\lambda}(R^{\lambda}, S_i^{\lambda} \mid X = x)\,. \tag{4.35}$$

Moreover, since the partial derivative in equation 4.35 is a function of λ and x, it can be moved inside the covariance.

5 Evaluators for Common Optimization Problems

An interesting aspect of the BCT is that the evaluators R^{λ} can be derived directly from the cost function without specifying the system being optimized. Moreover, the system covariates S^{λ} can be derived directly from the system dynamics, regardless of the cost function. This modularization of learning signals greatly simplifies software and hardware implementations. In this section, we derive the evaluators for four different learning problems that span supervised, unsupervised, and reinforcement situations: (1) regression, (2) density estimation, (3) risk minimization, and (4) information maximization. In section 6 we derive the system variables for continuous stochastic neural networks.

5.1 Regression. In regression problems, the goal is to learn the expected values of a random vector O conditional on an input vector X with respect to a probability measure P of input and outputs. In other words, the goal of regression is to approximate the function $\zeta(x) = E^P(O \mid X = x)$, from inputs to conditional expectation of the output (Papoulis, 1991, p. 179). Thus, in regression problems we care only about the expected value of the distribution of outputs and disregard its higher-order statistics. Most of the applications on supervised neural network learning, and classical signal filtering can be seen as regression problems. A popular cost function for such problems is the expected Euclidean distance, or sum of squares. To simplify the presentation, we focus on the case with only one observable node ($d = 1$),

$$\grave{C}(\lambda) = \frac{1}{2} \int_{\mathbb{R}^m} dx\; f_X(x) \parallel \zeta(x) - \mathrm{E}^{\lambda}(O^{\lambda} \mid X = x) \parallel^2\,, \tag{5.1}$$

which has the form studied in equation 4.27,

$$C(\lambda, u) = E^{\lambda}(O^{\lambda} \mid X = u)\,, \tag{5.2}$$

$$q\left(x, C(\lambda, u)\right) = \tfrac{1}{2} \| \zeta(x) - C(\lambda, u) \|^2 , \tag{5.3}$$

$$\tilde{Q}\left(o, x, f^{\lambda}_{O}\left(o\right), f^{\lambda}_{O|X}\left(o \mid x\right)\right) = f^{\lambda}_{O|X}(o \mid x)\, o\,. \tag{5.4}$$

Moreover, $C(\lambda, x)$ satisfies equation 4.10 since

$$\frac{\partial \tilde{Q}\left(o, x, f^{\lambda}_{O}\left(o\right), f^{\lambda}_{O|X}\left(o \mid x\right)\right)}{\partial f^{\lambda}_{O}(o)} = 0\,. \tag{5.5}$$

Applying equation 4.29 the evaluator random variable follows,

$$R^{\lambda} = O^{\lambda}\,, \tag{5.6}$$

$$\dot{R}^{\lambda} = -O^{\lambda}\left(\, \zeta(X) - \mathrm{E}^{\lambda}(O^{\lambda} \mid X)\, \right). \tag{5.7}$$

In general, if there is more than one observable node, $d > 1$, the evaluator is as follows:

$$\dot{R}^{\lambda} = -\sum_{j=1}^{d} O_j \left(\, \zeta_j(X) - \mathrm{E}^{\lambda}(O_j \mid X)\, \right). \tag{5.8}$$

Note that this evaluator corresponds to the classic backpropagation delta signal (Rumelhart, Hinton, & Williams, 1986) evaluated at the outputs. In diffusion networks, however, the very same evaluator is sent to all the adaptive parameters (e.g., the weights), with no need of further transformation as we move to hidden layers.

5.2 Density Estimation. In this case the problem is that of approximating an entire mapping from inputs into probability densities on $\mathbb{R}^d$. This is a much harder problem than regression, since we care about the entire density of outputs, not just the expected value. In general, density estimation is important when unimodal uncertainty models are not appropriate (Movellan & McClelland, 1993). A popular cost function for density estimation is the Kullback-Leibler information criterion (KLIC). In our case we need the KLIC between the desired and obtained conditional distributions averaged with respect to the input density (Haykin, 1994, p. 447),

$$C(\lambda) = \int_{\mathbb{R}^m} dx\; f_X(x) \int_{\mathbb{R}^d} do\; p(o \mid x)\; \log \frac{p(o \mid x)}{f^{\lambda}_{O|X}(o \mid x)}\,, \tag{5.9}$$

where $p(o \mid x)$ is the desired conditional output density. In this case,

$$\tilde{Q}\left(o, x, f^{\lambda}_{O}\left(o\right), f^{\lambda}_{O|X}\left(o \mid x\right)\right) = p(o \mid x)\; \log \frac{p(o \mid x)}{f^{\lambda}_{O|X}(o \mid x)}\,. \tag{5.10}$$

The partial derivative of $\tilde{Q}$ with respect to $f^{\lambda}_{O}(o)$ is zero, and thus equation 4.10 holds. The evaluator random variable, defined in equation 4.6, easily follows:

$$R^{\lambda} = -\frac{p(O^{\lambda} \mid X)}{f^{\lambda}_{O|X}(O^{\lambda} \mid X)} . \tag{5.11}$$

This evaluator measures to what extent the desired density is larger than the obtained density (i.e, whether state regions are visited at the desired rate). It requires computing densities of individual states. Good results can be obtained by discretizing the states into a finite number of regions. Although the number of states to keep track of in principle grows exponentially with the number of units, in practice only a few regions with nonnegligible measure are visited. Those are the only ones we need to care for when computing our covariance statistic. Note that this method avoids the two different learning phases of the Boltzmann machine learning algorithm (Ackley et al., 1985). In any case, it is easy to show that the gradient obtained using the BCT can also be expressed as a generalized form of the standard Boltzmann learning algorithm:

$$\begin{aligned}\frac{\partial C(\lambda)}{\partial \lambda_i} &= \beta \int_{\mathbb{R}^m} dx f_X(x)\ \mathrm{Cov}^{\lambda}(R^{\lambda}, S^{\lambda}_i \mid X = x) \\ &= -\beta \int_{\mathbb{R}^m} dx\ f_X(x) \int_{\mathbb{R}^d} do\ p(o \mid x) \\ &\quad [\mathrm{E}^{\lambda}(\ S^{\lambda}_i \mid O^{\lambda} = o, X = x\ ) - \mathrm{E}^{\lambda}(\ S^{\lambda}_i \mid X = x\ )] .\end{aligned} \tag{5.12}$$

where $\mathrm{E}^{\lambda}(\ S^{\lambda}_i \mid O^{\lambda} = o, X = x\ )$ is estimated by clamping the observable and input units, and $\mathrm{E}^{\lambda}(\ S^{\lambda}_i \mid X = x\ )$ is estimated by clamping the input units. For applications of diffusion networks to density estimation problems, see Movellan and McClelland (1993). Methods to accelerate learning are discussed in Stark and McClelland (1994).

5.3 Risk Minimization. The objective in this case is to minimize the expected loss, which in Bayesian decision theory is known as the risk (Duda & Hart, 1973, p. 14),

$$C(\lambda) = \mathrm{E}^{\lambda}[\ \rho^{\lambda}\ ], \tag{5.13}$$

where $\{\rho^{\lambda}\colon \Omega \to \mathbb{R}\}_{\lambda \in \mathbb{R}^p}$ is the loss random variable, defined by an auxiliary loss function $\tilde{\rho}\colon \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}$,

$$\rho^{\lambda} = \tilde{\rho}(O^{\lambda}, X). \tag{5.14}$$

Risk minimization is at the heart of most reinforcement problems. The definition of the loss function, $\tilde{\rho}$, is entirely general (e.g., it may be discrete

or continuous; it may be based on the entire output state or on just a few dimensions of the state). In this case,

$$\tilde{Q}\left(o, x, f_O^\lambda(o), f_{O|X}^\lambda(o \mid x)\right) = f_{O|X}^\lambda(o \mid x)\tilde{\rho}(o, x), \tag{5.15}$$

which satisfies equation 4.10 since the partial derivative of $\tilde{Q}$ with respect to $f_O^\lambda(o)$ is zero. Moreover, the evaluator is the loss itself,

$$R^\lambda = \rho^\lambda. \tag{5.16}$$

5.4 Information Maximization. Information maximization (infomax) is a classic criterion for unsupervised learning problems. Information maximization has been studied by Linsker (1988), and more recently by Nadal and Parga (1994) and by Bell and Sejnowski (1995), among others. It turns out that the infomax criterion satisfies the constraints of the BCT, and thus infomax learning can be performed using covariance statistics. An appropriate cost function for this problem is the negative mutual information between input and outputs, which is defined as follows (Haykin, 1994, p. 451):

$$C(\lambda) = -E^\lambda[\, I^\lambda \,], \tag{5.17}$$

where I^λ: $\Omega \to \mathbb{R}$ is the mutual differential information random variable, defined by auxiliary functions $\{\tilde{I}^\lambda$: $\mathbb{R}^m \times \mathbb{R}^d \to \mathbb{R}\}_{\lambda \in \mathbb{R}^p}$,

$$I^\lambda = \tilde{I}^\lambda(O, X), \tag{5.18}$$

$$\tilde{I}^\lambda(o, x) \triangleq \log \frac{f_{O|X}^\lambda(o \mid x)}{f_O^\lambda(o)}. \tag{5.19}$$

Note that equation 5.17 is not a special case of risk minimization because $\tilde{I}^\lambda$ varies with λ whereas $\tilde{\rho}$ does not. In this case,

$$\tilde{Q}\left(o, x, f_O^\lambda(o), f_{O|X}^\lambda(o \mid x)\right) = f_{O|X}^\lambda(o \mid x)\ \log \frac{f_{O|X}^\lambda(o \mid x)}{f_O^\lambda(o)}, \tag{5.20}$$

which satisfies equation 4.10, since

$$\nabla_o \int_{\mathbb{R}^m} dx\ f_X(x)\ \tilde{T}^\lambda(o, x) = -\nabla_o \int_{\mathbb{R}^m} dx\ \frac{f_X(x) f_{O|X}^\lambda(o \mid x)}{f_O^\lambda(o)} = 0, \tag{5.21}$$

where $\tilde{T}$ is defined in equation 4.8. Therefore, the BCT applies, and the evaluator random variable easily follows,

$$R^\lambda = -I^\lambda(O^\lambda, X) - 1. \tag{5.22}$$

Since constants do not affect covariance statistics, we can drop the -1. Equation 5.22 tells us that the smaller the mutual information in an outcome, the less we like that outcome. This evaluator requires computing mutual information of individual states. Good results can be obtained by discretizing the states into a finite number of regions. Although the number of states to keep track of in principle grows exponentially with the number of units, in practice only a few regions have nonnegligible measure and thus an influence on the covariance.

5.5 Remarks. Since covariances are linear operators, it is possible to optimize a weighted sum of learning criteria by using an evaluator that is a weighted sum of the evaluators for each criterion. Second-order optimization methods (Gill, Murray, & Wright, 1981, p. 105) would also be based on covariance statistics. The reason is that covariances are linear operators that satisfy the conditions of the BCT. Therefore, if the gradient is a linear combination of covariances, the gradient of gradients (the Hessian matrix) is also a linear combination of covariances.

6 System Covariates

Up to now, we have derived evaluators for a variety of cost functions. In this section, we construct diffusion neural networks and derive their system covariates. This completes all that is needed to apply the BCT. There are many ways to construct diffusion neural networks with detailed equilibrium solutions. The one we worked with in our simulations is based on the continuous Hopfield model (Hopfield, 1984) and is constructed as follows. We are given:

- An $n \times n$ symmetric matrix, w, whose elements represent coupling strengths between nodes in the network.
- An $n \times m$ matrix, v, representing the coupling strengths between input lines and the nodes in the network.
- A vector $\alpha \in \mathbb{R}^n$, of activation gains.
- An invertible activation function g: $\mathbb{R} \to \mathbb{R}$.
- A constant β that controls the level of uncertainty in the system.

The adaptive parameter vector λ is the elements of w, v, and α organized as a vector. We then define the following potential function,

$$U(\lambda, y, x) = \sum_{i=1}^{n} \alpha_i \int_{g(0)}^{g(y_i)} g^{-1}(s)ds \; - \frac{1}{2} \sum_{i,j=1}^{n} g(y_i) w_{ij} g(y_j) - \sum_{i=1}^{n} \sum_{j=1}^{m} g(y_i) v_{ij} x_j \;, \tag{6.1}$$

where $g^{-1}(g(y)) \triangleq y$. We then define the drift as the negative gradient of the potential,

$$\begin{aligned}\mu_i(\lambda, y, x) &= -g'(y_i)\frac{\partial U(\lambda, y, x)}{\partial g(y_i)} \\ &= g'(y_i)\left(-\alpha_i y_i + \sum_{j=1}^{n} w_{ij} g(y_j) + \sum_{j=1}^{m} v_{ij} x_j\right), \end{aligned} \tag{6.2}$$

where g' is the derivative of g. The state variables Y_i^λ are commonly interpreted as presynaptic activations and $g(Y_i^\lambda)$ as postsynaptic activation. The network dynamics are described by equation 4.1 defining a stochastic variation of the continuous Hopfield model (Hopfield, 1984). In this case, the Wiener process represents presynaptic noise. The system covariates (see equation 4.3) easily follow:

$$S_i = \begin{cases} g(Y_j)\, g(Y_k) & \text{if } \lambda_i \text{ is } w_{j,k}, \\ g(Y_j)\, X_k & \text{if } \lambda_i \text{ is } v_{j,k}, \\ -\int_{g(0)}^{g(Y_j)} g^{-1}(s)\, ds & \text{if } \lambda_i \text{ is } \alpha_j. \end{cases} \tag{6.3}$$

Note that since the potential function is separable into additive second-order terms, the system covariates are local. Moreover, the S_i for the weight parameters are products of activations, and thus the necessary gradients can be computed with Hebbian-like operations.

7 Simulation: Image Reconstruction

The purpose of this simulation was to test whether the BCT could be used in practice to train the equilibrium distribution of diffusion networks. To do so, we need to substitute the covariances called for by the BCT by estimates of these covariances based on discrete time approximations to SDEs. For this reason, it is unclear whether the learning algorithms would work at all in computer simulations. We chose a problem for which the true population covariances of the continuous system can be obtained analytically and which has potential applications for hardware implementations of diffusion networks (Alspector et al., 1992).

The technique employed for the simulation was a simple forward-Euler approach. Equation 2.1 is replaced with the discrete-time stochastic difference equation,

$$Y_{t+\Delta t}^\lambda = Y_t^\lambda + \mu(\lambda, Y_t^\lambda, X)\Delta t + \sigma\sqrt{\Delta_t} Z_t\,, \tag{7.1}$$

where Δt is a small constant, and for each $\{t = k\Delta t;\ k = 1, 2, \ldots\}$, the random vectors Z_t are independent identically distributed N-dimensional gaussian

with zero mean and identity covariance matrix. Moreover, we fixed $\sigma = I_n$, the $n \times n$ identity matrix. When the sampled points are linearly interpolated, this defines a stochastic process that converges in distribution to the solution of the original SDE as $\Delta t \to 0$ (Gillespie, 1992, p. 193).

7.1 Task and Network Architecture. The task was to do optimal reconstruction[3] of noise-contaminated samples from TULIPS1 (Movellan, 1995), a database of 935 human lip images.[4] Each image consists of 100×75 pixels with gray-level values ranging from 0 to 255. The images were contaminated on a pixel-by-pixel basis with i.i.d. zero mean Gaussian noise and standard deviation ranging from 10 to 80.

The network consisted of 100×75 observable units, one per pixel, an equal number of input units, and no hidden units. Thus, in our notation, $n = m = d = 7500$. There was a one-to-one correspondence of input units, output units, and pixels. Each input unit represented a pixel value in a noisy version of an image. The corresponding output unit represented the clean value of the same pixel. To do the reconstruction, each output unit used a 15×15 receptive field: each output unit received input from the corresponding input unit and from 224 surrounding input units arranged as a square patch. Image borders were treated using a toroidal wrap-around of the input image. All receptive fields were constrained to share the same kernel of weights, effectively performing a convolution operation. Thus, the total number of free parameters was $p = 225$. The activation function was linear $g(y) = y$ since for this case the optimal solution can be shown to be a classical Wiener filter (Jain, 1989), which can be calculated analytically. This allowed us to compare the solution found by the network with the optimal solution.

The BCT works for equilibrium solutions, the limiting density as $t \to \infty$. In practice, the initial state of each output unit was set equal to the inner product between the input weight vector of that unit and the input image being processed. Then we cycled 10 times using equation 7.1, with $\Delta t = 0.1$. After these 10 settling cycles, the system was considered sufficiently closed to stochastic equilibrium. Equilibrium statistics were then calculated by running the network for 10 additional cycles and estimating expected values and covariances based on the states obtained during those 10 cycles.

7.2 Training Sample. Each training sample consisted of a noisy input image, which was used as input (100×75 noisy pixel values), and the corresponding clean image, which was used as a teacher for the output units. Training was performed with respect to the sum of squares criterion

[3] We are using *optimal* in the mean square sense. It is well known that other filters may perform better with respect to other criteria.

[4] Available at http://cogsci.ucsd.edu.

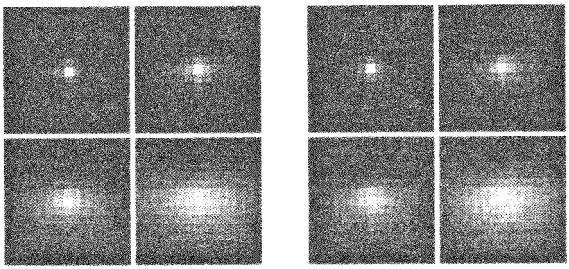

Figure 1: (Left) Weight kernels analytically derived. Each image has 15 × 15 pixels, with each pixel representing a weight. Large weights appear white, small weights dark. (Right) The weight kernels learned using a diffusion network. The four kernels reflect four different noise conditions. From top to bottom and left to right, the standard deviation of noise is 10, 20, 40, and 80, respectively, where each pixel can take values from 0 to 255.

presented in section 5.1. Thus, the goal was for the expected value of each output unit to approximate as closely as possible the value of the clean pixel corresponding to that output unit. Learning was done using Newton's second-order method (Gill et al., 1981, p. 105), with a single pass over the entire image database. As mentioned in section 5.5, computation of the Hessian matrix is also accomplished through the use of covariance statistics. We had to resort to second-order methods because first-order methods were too slow for this task, a problem shared by deterministic linear systems with a large number of correlated inputs.

Figure 1 shows the impulse response of the optimal Wiener filter, analytically obtained, and the weight kernels obtained using the discrete time approximation to the diffusion network. Each image in Figure 1 has 15 × 15 pixels, with each pixel representing a weight. Large weights appear white, small weights dark. The weights were obtained for four different levels of noise in the input images. As the figure shows, the solutions obtained by sampling in discrete time were very close to the analytical solutions. Note how the kernels increase in size as the noise power increases. Note also that the kernels are elongated horizontally, capturing the fact that lips are mostly a horizontal structure and, thus, on average, pixel values correlate more with horizontal than with vertical neighbors. Figure 2 shows example reconstructions of three images performed by the simulated diffusion network. The first column shows the original images, the second column shows the images contaminated with gaussian noise (mean = 0, SD = 80), and the third column is the reconstruction. Each pixel of the reconstruction image is the conditional average activation of an output unit given the noisy input image.

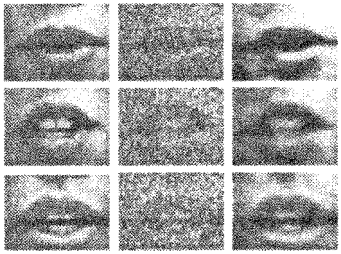

Figure 2: (Left to right) Clean images, contaminated images (SD = 80), and reconstructed images.

8 Discussion

We explored the problem of learning equilibrium solutions in diffusion networks, a stochastic extension of continuous neural networks. A learning theorem is proposed that specifies sufficient conditions for the gradients of cost functions to be computable by simple covariance statistics of a uniform evaluator and a set of local system variables. The conditions proposed by the theorem apply to a variety of cost functions that span common supervised, unsupervised, and reinforcement problems. Although our focus was on training diffusion networks, the learning theorem applies to any system whose equilibrium solution is Boltzmann (see Neal, 1993, and Gidas, 1986, for a review of such systems).

The article suggests an approach to learning that may serve as inspiration for hardware design and computational neuroscience. The approach is consistent with von Neumann's views of the brain as a system in which "error ... is not [seen] as an extraneous and misdirected or misdirecting accident, but as an essential part of the process under consideration" (Neumann, 1956). Instead of suppressing noise present in physical devices, natural computers may use it to simplify the credit assignment problem. In deterministic approaches, like backpropagation, proper credit assignment requires an evaluation signal that is different for each node in the network. Surprisingly, when noise is not suppressed, all that is needed is an evaluator that is the same for the entire network. The covariance between this uniform evaluator and Hebbian signals is sufficient for the proper distribution of credit throughout the entire network.

Acknowledgments

This article emerged through interactions with James McClelland, who sparked my interest in noisy natural computers. Ongoing interactions with Ruth Williams and Paul Mineiro helped formalize my thinking about diffu-

sion networks. Comments from an anonymous reviewer provided a missing link for the BCT. I am grateful to David Zipser for his guidance.

References

Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science*, *9*(2), 147–169.

Alspector, J., Jayakumar, A., & Luna, S. (1992). Experimental evaluation of learning in a neural mycrosystem. In J. Moody, S. Hanson, & R. Lippmann (Eds.), *Advances in neural information processing systems* (Vol. 4, pp. 871–878). San Mateo, CA: Morgan Kaufman.

Andreou, A. (1994). On physical models of neural computation and their analog VLSI implementation. In *Workshop on physics and computation, Phys-comp 94 conference* (pp. 255–264). Los Alamitos, CA: IEEE Computer Society.

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

Borkar, S. V. (1989). *Optimal control of diffusion processes.* New York: Longman.

Cowan, J. D. (1968). Statistical mechanics of nervous nets. In E. R. Caianiello (Ed.), *Neural networks* (p. 181). Berlin: Springer-Verlag.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis.* New York: Wiley.

Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions of Pattern Analysis and Machine Intelligence, PAMI-6,* 721–741.

Geman, S., & Hwang, C.-R. (1986). Diffusions for global optmization. *SIAM J. Control and Optimization, 24,* 1031–1043.

Gerstein, G. L., & Mandelbrot, B. (1964). Random walk models of the spike activity of a single neuron. *Biophysics J., 4,* 41–68.

Gidas, B. (1986). Metropolis-type Monte Carlo simulation algorithms and simulated annealing. In J. L. Snell (Ed.), *Topics in contermporary probability and its applications* (pp. 159–232). Boca Raton, FL: CRC Press.

Gill, E. P., Murray, W., & Wright, M. H. (1981). *Practical optimization.* London: Academic Press.

Gillespie, D. T. (1992). *Markov processes: An introduction for physical scientists.* San Diego: Academic Press.

Hanson, F. B., & Tuckwell, H. C. (1983). Diffusion approximations for neural activity including synaptic reversal potentials. *J. Theoretical Neurobiology, 2,* 127–153.

Haykin, S. (1994). *Neural networks: A comprehensive foundation.* New York: Macmillan.

Hopfield, J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Science, 81,* 3088–3092.

Isard, M., & Blake, A. (1996). Contour tracking by stochastic propagation of conditional density. In *Proc. European Conf. Computer Vision* (Vol. 58, pp. 343–356). Cambridge, UK.

Jain, A. K. (1989). *Fundamentals of digital image processing*. Englewood Cliffs, NJ: Prentice Hall.

Karatzas, I., & Shreve, S. E. (1988). *Brownian motion and stochastic calculus*. New York: Springer-Verlag.

Landauer, R. (1992). Information is physical. In *Proceedings of the 1992 Physics of Computation Workshop* (pp. 1–4). Los Alamitos, CA: IEEE Computer Society.

Linsker, R. (1988). Self-organization in a perceptual network. *IEEE Computer*, *21*, 105–117.

Matsuyama, M., Shirai, K., & Akizuki, K. (1974). On some properties of stochastic information processes in neurons and neuron populations. *Kybernetik*, *15*, 127–145.

McClelland, J. L. (1993). Toward a theory of information processing in graded, random, and interactive networks. In D. E. Meyer & S. Kornblum (Eds.), *Attention and performance XIV: Synergies in experimental psychology, artificial intelligence, and cognitive neuroscience* (pp. 655–688). Cambridge, MA: MIT Press.

Mead, C. A., & Conway, L. (1980). *Introduction to VLSI*. Reading, MA: Addison-Wesley.

Movellan, J. R. (1995). Visual speech recognition with stochastic neural networks. In G. Tesauro, D. Touretzky, & T. Leen (Eds.), *Advances in neural information processing systems*. Cambridge, MA: MIT Press.

Movellan, J. R., & McClelland, J. L. (1993). Learning continuous probability distributions with symmetric diffusion networks. *Cognitive Science*, *17*(4), 463–496.

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

Neal, R. M. (1993). *Probabilistic inference using Markov chain Monte Carlo methods*. (Tech. Rep. No. CRG-TR-93-1). Toronto: Department of Computer Science, University of Toronto.

Neumann, J. (1956). Probabilistic logics and the synthesis of reliable organisms from unreliable components. In C. E. Shannon & J. McCarthy (Eds.), *Automata studies* (pp. 43–98). Princeton: Princeton University Press.

Ohira, T., & Cowan, J. D. (1995). Stochastic single neurons. *Neural Computation, 7*, 518–528.

Oksendal, B. (1992). *Stochastic differential equations*. Berlin: Springer-Verlag.

Papoulis, A. (1991). *Probability, random variables, and stochastic processes*. New York: McGraw-Hill.

Poggio, T., & Girosi, F. (1994). *Continuous stochastic cellular automata that have a stationary distribution and no detailed balance* (Tech. Rep. A.I. Memo No. 1168). Cambridge, MA: MIT, AI Lab.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257–286.

Ratcliff, R. (1979). Group reaction time distributions and an analysis of distribution statistics. *Psychological Bulletin*, *86*, 446–461.

Ricciardi, L. M. (1977). *Diffusion processes and related topics in biology: Lecture notes in biomathematics*. Berlin: Springer-Verlag.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, & the PDP Research Group (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition* (Vol. 1, pp. 318–362). Cambridge, MA: MIT Press.

Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition* (Vol. 1, pp. 194–281). Cambridge, MA: MIT Press.

Stark, C., & McClelland, J. L. (1994). Tractable learning of probability distributions using the contrastive Hebbian algorithm. In *Proceedings of CogSci94* (pp. 818–823). Hillsdale, NJ: Erlbaum.

Zipser, D. (1991). Recurrent network model of the neural mechanism of short-term active memory. *Neural Computation, 3,* 179.

Received June 21, 1996; accepted September 9, 1997.