ARTICLE Communicated by Stuart Geman

A Monte Carlo EM Approach for Partially Observable Diffusion Processes: Theory and Applications to Neural Networks

Javier R. Movellan
movellan@inc.ucsd.edu
Machine Perception Laboratory, Institute for Neural Computation, University of California, San Diego, La Jolla, CA 92093, U.S.A.

Paul Mineiro
paul@mineiro.com
Department of Cognitive Science, University of California, San Diego, La Jolla, CA 92093, U.S.A.

R. J. Williams
williams@math.ucsd.edu
Department of Mathematics and Institute for Neural Computation, University of California, San Diego, La Jolla, CA 92093, U.S.A.

We present a Monte Carlo approach for training partially observable diffusion processes. We apply the approach to diffusion networks, a stochastic version of continuous recurrent neural networks. The approach is aimed at learning probability distributions of continuous paths, not just expected values. Interestingly, the relevant activation statistics used by the learning rule presented here are inner products in the Hilbert space of square integrable functions. These inner products can be computed using Hebbian operations and do not require backpropagation of error signals. Moreover, standard kernel methods could potentially be applied to compute such inner products. We propose that the main reason that recurrent neural networks have not worked well in engineering applications (e.g., speech recognition) is that they implicitly rely on a very simplistic likelihood model. The diffusion network approach proposed here is much richer and may open new avenues for applications of recurrent neural networks. We present some analysis and simulations to support this view. Very encouraging results were obtained on a visual speech recognition task in which neural networks outperformed hidden Markov models.

1 Introduction

Since Hopfield's seminal work (Hopfield, 1984), continuous deterministic neural networks and discrete stochastic neural networks have been thor-

oughly studied by the neural network community (Pearlmutter, 1995; Ackley, Hinton, & Sejnowski, 1985). However, the continuous stochastic case has been conspicuously ignored. This is surprising considering the success of continuous stochastic models in other fields (Oksendal, 1998).

In this article, we focus on the continuous stochastic case and present a Monte Carlo expectation-maximization (EM) approach for training continuous-time, continuous-state, stochastic recurrent neural network models. The goal is to learn probability distributions of continuous paths, not just equilibrium points. This is important for problems involving sequences, such as speech recognition and object tracking. The approach proposed here potentially opens new avenues for applications of recurrent neural networks showing results comparable to, if not better than, those obtained with hidden Markov models. In addition, the maximum likelihood learning rules used to train these networks are based on inner products that are computable using local Hebbian statistics. This is an aspect of potential value for neurally plausible learning models and for potential generalizations of kernel methods (Aizerman, Braverman, & Rozoner, 1964; Burges, 1998).

Continuous-time, continuous-state recurrent neural networks (hereafter referred to simply as recurrent neural networks) are dynamical systems consisting of n point neurons coupled by synaptic connections. The strength of these connections is represented by an $n \times n$ real-valued matrix w, and the network dynamics are governed by the following differential equations,

$$\frac{dx_j(t)}{dt} = \mu_j(x(t), \lambda), \text{ for } t \in [0, T], j = 1, \ldots, n, \tag{1.1}$$

where

$$\mu_j(x(t), \lambda) = \kappa_j \left(-\rho_j x_j(t) + \xi_j + \sum_{i=1}^{n} \varphi(x_i(t)) w_{ij} \right), \tag{1.2}$$

x_j is the soma potential of neuron j, $1/\rho_j > 0$ is the transmembrane resistance,[1] $1/\kappa_j > 0$ is the input capacitance, ξ_j is a bias current, w_{ij} is the conductance (synaptic weight) from unit i to unit j, and φ is a nonlinear activation function, typically a scaled version of the logistic function

$$\varphi(v) = \theta_1 + \theta_2 \frac{1}{1 + e^{-\theta_3 v}}, \text{ for } v \in \mathbb{R}, \tag{1.3}$$

[1] We allow $1/\rho_j$ or $1/\kappa_j$ to be $+\infty$, corresponding to situations in which $\rho_j = 0$ or $\kappa_j = 0$, respectively.

where $\theta = (\theta_1, \theta_2, \theta_3) \in \mathbb{R}^3$ are fixed scale and gain parameters. Here, $\lambda \in \mathbb{R}^p$ represents the w, ξ, κ, and ρ terms, whose values are typically varied in accordance with some learning rules.

1.1 Hidden Units. A variety of algorithms have been developed to train these networks, and are commonly known as *recurrent neural network algorithms* (Pearlmutter, 1995). An important achievement of these algorithms is that they can train networks with hidden units. Hidden units allow these networks to develop time-delayed representations and feature conjunctions. For this reason, recurrent neural networks were expected to become a standard tool for problems involving continuous sequences (e.g., speech recognition) in the same way that backpropagation networks became a standard tool for problems involving static patterns.

Recurrent network learning algorithms have proved useful in some fields. In particular, they have been useful for understanding the role of neurons in the brain. For example, when these networks are trained on simple sequences used in controlled experiments with animal subjects, the hidden units act as "memory" neurons similar to those found in neural recordings (Zipser, Kehoe, Littlewort, & Fuster, 1993). While these results have been useful to help understand the brain, recurrent neural networks have yielded disappointing results when applied to engineering problems such as speech recognition or object tracking. In such domains, probabilistic approaches, such as hidden Markov models and Kalman filters, have proved to be superior.

1.2 Diffusion Networks. We believe that the main reason that recurrent neural networks have provided disappointing results in some engineering applications (e.g., speech recognition) is due to the fact that they implicitly rely on a very simplistic likelihood model that does not capture the kind of variability found in natural signals. We will elaborate on this point in section 8.1. To overcome this deficiency, we propose adding noise to the standard recurrent neural networks dynamics, as would be done in the stochastic filtering and systems identification literature (Lewis, 1986; Ljung, 1999). Mathematically, this results in a diffusion process, and thus we call these models *diffusion neural networks*, or *diffusion networks* for short (Movellan & McClelland, 1993). While recurrent neural networks are defined by ordinary differential equations (ODEs), diffusion networks are described by stochastic differential equations (SDEs). Stochastic differential equations provide a rich language for expressing probabilistic temporal dynamics and have proved useful in formulating continuous-time inference problems, as, for example, in the continuous Kalman-Bucy filter (Kalman & Bucy, 1961; Oksendal, 1998).

Diffusion networks can be interpreted as a low-power version of recurrent networks, in which the thermal noise component is nonnegligible. The temporal evolution of a diffusion network with vector parameter λ defines

an n-dimensional stochastic process X^λ that satisfies the following SDE:[2]

$$dX^\lambda(t) = \mu(X^\lambda(t), \lambda)dt + \sigma dB(t), t \in [0, T], \tag{1.4}$$

$$X^\lambda(0) \sim \nu. \tag{1.5}$$

Here $\mu = (\mu_1, \ldots, \mu_n)'$ is called the *drift* vector. We use the same drift as recurrent neural networks, but the learning algorithm presented here is general and can be applied to other drift functions; B is a standard n-dimensional *Brownian motion* (see section 2), which provides the random driving noise for the dynamics; $\sigma > 0$ is a fixed positive constant called the *dispersion*, which governs the amplitude of the noise; $T > 0$ is the length of the time interval over which the model is used; and ν is the probability distribution of the initial state $X^\lambda(0)$. We regard $T > 0$ and ν as fixed henceforth.

1.3 Relationship to Other Models. Figure 1 shows the relationship between diffusion networks and other approaches in the neural network and stochastic filtering literature. Diffusion networks belong to the category of partially observable Markov models (Campillo & Le Gland, 1989), a category that includes standard hidden Markov models and Kalman filters as special cases. Standard hidden Markov models are defined in discrete time, and their internal states are discrete. Diffusion networks can be viewed as hidden Markov models with continuous-valued hidden states and continuous-time dynamics. The continuous-time nature of the networks is convenient for data with dropouts or variable sample rates, since continuous-time models define all of the finite dimensional distributions. The continuous-state representation (see Figure 2) is well suited for problems involving inference about continuous unobservable quantities, such as object tracking tasks, and modeling of cognitive processes (McClelland, 1993; Movellan & McClelland, 2001).

If the logistic activation function φ is replaced by a linear function, the weights between the observable units and from the observable to the hidden units are set to zero, the ρ parameters are set to zero, and the probability distribution of the initial states is constrained to be gaussian, diffusion networks have the same dynamics underlying the continuous-time Kalman-Bucy filter (Kalman & Bucy, 1961). If, on the other hand, the weight matrix w is symmetric, at stochastic equilibrium, diffusion networks behave like continuous-time, continuous-state Boltzmann machines (Ackley et al.,

[2] We use X^λ to make explicit the dependence of the solution process on the parameter λ. The following assumptions are sufficient for existence and uniqueness in distribution of solutions to equations 1.4 and 1.5; ν has bounded support, $\mu(\cdot, \lambda)$ is continuous and satisfies a linear growth condition $|\mu(u, \lambda)| \leq K_\lambda(1 + |u|)$, for some $K_\lambda > 0$ and all $u \in \mathbb{R}^n$, where $|\cdot|$ denotes the Euclidean norm (see e.g., proposition 5.3.6 in Karatzas & Shreve, 1991). These assumptions are satisfied by the recurrent neural network drift function.

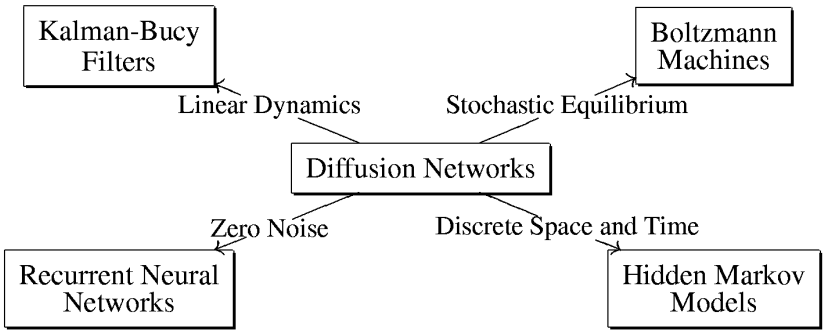

Figure 1: Relationship between diffusion networks and other approaches in the neural network and stochastic filtering literature.

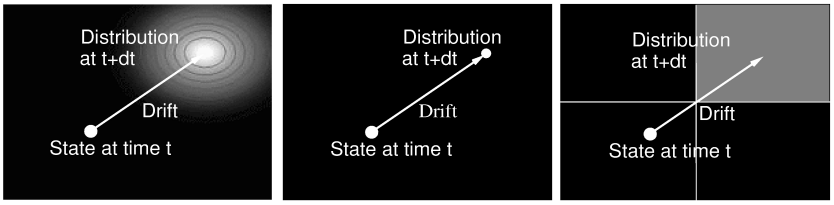

Figure 2: In stochastic differential equations (left), the states are continuous and the dynamics are probabilistic. Given a state at time t, there is a distribution of possible states at time $t+dt$. In ordinary differential equations (center), the states are continuous, and the dynamics are deterministic. In hidden Markov models (right) the hidden states are discrete and probabilistic. This is represented in the figure by partitioning a continuous state-space into four discrete regions and assigning equal transition probability to states within the same region.

1985). Finally, if the dispersion constant σ is set to zero, the network becomes a standard deterministic recurrent neural network (Pearlmutter, 1995).

In the past, algorithms have been proposed to train expected values and equilibrium points of diffusion networks (Movellan, 1994, 1998). Here, we present a powerful and general algorithm to learn distributions of trajectories. The algorithm can be used to train diffusion networks with hidden units, thus maintaining the versatility of the recurrent neural network approach. As we will illustrate, the algorithm is sufficiently fast to train networks with thousands of parameters using current computers and fares well when compared to other probabilistic approaches, such as hidden Markov models.

2 Mathematical Preliminaries

2.1 Brownian Motion and Stochastic Differential Equations. Brownian motion is a stochastic process originally designed to model the behavior of pollen grains subject to random collisions with molecules of water. A stochastic process $B = \{B(t), t \in [0, \infty)\}$ is a standard one-dimensional Brownian motion under a probability measure P if (1) it starts at 0 with P-probability one; (2) for any $t \in [0, \infty)$, $\Delta t > 0$, the increment $B(t + \Delta t) - B(t)$ is a gaussian random variable under P with zero mean and variance Δt; and (3) for all $l \in \mathbb{N}$, and $0 \leq t_0 < t_1 \cdots < t_l < \infty$, the increments $B(t_k) - B(t_{k-1})$, $k = 1, \cdots, l$ are independent random variables under P. An n-dimensional Brownian motion consists of n independent one-dimensional Brownian motion processes. One can always choose a realization of Brownian motion that has continuous paths.[3] However, these paths are nowhere differentiable with probability one (Karatzas & Shreve, 1991). Brownian motion can be realized as the limit in distribution as $\Delta t \to 0$ of the processes obtained by the following iterative scheme:

$$B(0) = 0, \tag{2.1}$$

$$B(t_{k+1}) = B(t_k) + \sqrt{\Delta t} Z(t_k), \tag{2.2}$$

$$B(s) = B(t_k), \text{ for } s \in [t_k, t_{k+1}), \tag{2.3}$$

where $t_k = k\Delta t$, $k = 0, 1, \cdots$, and the $Z(t_k)$ terms are independent gaussian random variables with zero mean and unit variance.

It is common in the engineering literature to represent stochastic differential equations as ordinary differential equations with an additive white noise component,

$$\frac{dX(t)}{dt} = \mu(X(t)) + \sigma W(t), \tag{2.4}$$

where W represents white noise, a stochastic process that is required to have the following properties: (1) W is a zero-mean stationary gaussian process; (2) for all t, t', the covariance between $W(t')$ and $W(t)$ is a Dirac delta function of $t - t'$.

While white noise does not exist as a proper stochastic process, equation 2.4 can be given mathematical meaning by thinking of it as defining a process of the form

$$X(t) = X(0) + \int_0^t \mu(X(s))\, ds + \sigma \int_0^t W(s)\, ds, \tag{2.5}$$

[3] A Brownian motion for which $P(\text{for all } t \geq 0, \lim_{s \to t} X_s = X_t) = 1$.

where $\int_0^t W(s)\,ds$ is now a stochastic process required to have continuous paths and zero-mean, independent stationary increments. It turns out that Brownian motion is the only process with these properties. Thus, in stochastic calculus, equation 2.4 is seen just as a symbolic pointer to the following integral equation,

$$X(t) = X(0) + \int_0^t \mu(X(s))\,ds + \sigma B(t), \tag{2.6}$$

where B is Brownian motion. Moreover, if it existed, the white noise $W(t)$ would be the temporal derivative of Brownian motion, $dB(t)/dt$. Since Brownian paths are nowhere differentiable with probability one, in the mathematical literature, the following symbolic form is preferred to equation 2.4, and it is the one adopted in this article:

$$dX(t) = \mu(X(t))dt + \sigma\,dB(t). \tag{2.7}$$

Intuitive understanding of the solution to this equation can be gained by thinking of it as the limit as $\Delta t \to 0$ of the processes defined by the following iterative scheme,

$$X(t_{k+1}) = X(t_k) + \mu(X(t_k))\Delta t + \sigma\sqrt{\Delta t}Z(t_k), \tag{2.8}$$

$$X(s) = X(t_k) \text{ for } s \in [t_k, t_{k+1}), \tag{2.9}$$

where $t_k = k\Delta t, k = 0, 1, \ldots$. Under mild assumptions on μ and the initial distribution of X, the processes obtained by this scheme converge in distribution to the solution of equation 2.7 (Kloeden & Platen, 1992). Under the assumptions in this article, the solution of equation 2.7 is unique only in distribution, and thus it is called a distributional (also known as a weak or statistical) solution.[4]

2.2 Probability Measures and Densities. We think of a random experiment as a single run of a diffusion network in the time interval $[0, T]$. The outcome of an experiment is a continuous n-dimensional path x: $[0, T] \to \mathbb{R}^n$ describing the state of the n units of a diffusion network throughout time. We let Ω denote the set of continuous functions defined on $[0, T]$ taking values in $\mathbb{R}^n$. This contains all possible outcomes. We are generally interested in measuring probabilities of sets of outcomes. We call these sets events, and we let $\mathcal{F}$ represent the set of events.[5]

[4] For an explanation of the notion of weak solutions of stochastic differential equations, see section 5.3 of Karatzas and Shreve (1991).

[5] In this article, the set of events $\mathcal{F}$ is the smallest sigma algebra containing the open sets of Ω. A set A of Ω is open if for every $x \in A$ there exists a $\delta > 0$ such that the set $B(x, \delta) = \{\omega \in \Omega\colon \max_{t\in[0,T]} |x(t) - \omega(t)| < \delta\}$ is a subset of A.

A probability measure Q is a function Q: $\mathcal{F} \to [0, 1]$ that assigns probabilities to events in accordance with the standard probability axioms (Billingsley, 1995). A random variable Y is a function Y: $\Omega \to \mathbb{R}$ such that for each open set A in $\mathbb{R}$, the inverse image of A under Y is an event.

We represent expected values using integral notation. For example, the expected value of the random variable Y with respect to the probability measure Q is represented as follows:

$$E^Q(Y) = \int_\Omega Y(x)\, dQ(x). \tag{2.10}$$

Probability densities of continuous paths are defined as follows. Let P and Q be probability measures on $(\Omega, \mathcal{F})$. If it exists, the density of P with respect to Q is a nonnegative random variable L that satisfies the following relationship,

$$E^P(Y) = \int_\Omega Y(x)\, dP(x) = \int_\Omega Y(x)L(x)\, dQ(x) = E^Q(YL), \tag{2.11}$$

for any random variable Y with finite expected value under P. The function L is called the Radon-Nikodym derivative or Radon-Nikodym density of P with respect to Q and is commonly represented as dP/dQ. Conditions for the existence of these derivatives can be found in any measure theory textbook (Billingsley, 1995). Intuitively, $(dP/dQ)(x)$ represents how many times the path x is likely to occur under P relative to the number of times it is likely to occur under Q.

In this article, we concentrate on the probability distributions on the space Ω of continuous paths associated with diffusion networks, and thus we need to consider distributional solutions of SDEs. We let the n-dimensional random process X^λ represent a solution of equations 1.4 and 1.5. We regard the first d components of X^λ as observable and denote them by O^λ. The last $n-d$ components of X^λ are denoted by H^λ and are regarded as unobservable or hidden.

We define the observable and hidden outcome spaces Ω_o and Ω_h with associated event spaces $\mathcal{F}_o$ and $\mathcal{F}_h$ by replacing n by d and $n-d$, respectively, in the definitions of Ω and $\mathcal{F}$ provided previously. The observable and hidden components O^λ and H^λ of a solution $X^\lambda = (O^\lambda, H^\lambda)$ of equations 1.4 and 1.5 take values in Ω_o and Ω_h, respectively. Note that $\Omega = \Omega_o \times \Omega_h$ and $\mathcal{F}$ is generated by sets of the form $A_o \times A_h$ where $A_o \in \mathcal{F}_o$ and $A_h \in \mathcal{F}_h$. For each path $x \in \Omega$, we write $x = (x_o, x_h)$, where $x_o \in \Omega_o$ and $x_h \in \Omega_h$.

The process X^λ induces a unique probability distribution Q^λ on the measurable space $(\Omega, \mathcal{F})$. Intuitively, $Q^\lambda(A)$ represents the probability that a diffusion network with parameter λ produces paths in the set A. Since our goal is to learn a probability distribution of observable paths, we need the

probability measure Q_o^λ associated with the observable components alone. If there are no hidden units, $d = n$ and $Q_o^\lambda = Q^\lambda$. If there are hidden units, $d < n$, and we must marginalize Q^λ over the unobservable components:

$$Q_o^\lambda(A_o) = Q^\lambda(A_o \times \Omega_h) \quad \text{for all } A_o \in \mathcal{F}_o. \tag{2.12}$$

We will also need to work with the marginal probability measure of the hidden components:

$$Q_h^\lambda(A_h) = Q^\lambda(\Omega_o \times A_h) \quad \text{for all } A_h \in \mathcal{F}_h. \tag{2.13}$$

Intuitively, $Q_o^\lambda(A_o)$ is the probability that a diffusion network with parameter λ generates observable paths in the set A_o, and $Q_h^\lambda(A_h)$ is the probability that the hidden paths are in the set A_h.

Finally, we set $\mathcal{Q} = \{Q^\lambda\colon \lambda \in \mathbb{R}^p\}$ and $\mathcal{Q}_o = \{Q_o^\lambda\colon \lambda \in \mathbb{R}^p\}$, referring to the entire family of probability measures parameterized by λ and defined on $(\Omega, \mathcal{F})$ and $(\Omega_o, \mathcal{F}_o)$, respectively.

3 Density of Observable Paths

Our goal is to select a value of λ on the basis of training data, such that Q_o^λ best approximates a desired distribution. To describe a maximum likelihood or a Bayesian estimation approach, we need to define probability densities L_o^λ of continuous observable paths. In discrete time systems, like hidden Markov models, the Lebesgue measure[6] is used as the standard reference with respect to which probability densities are defined. Unfortunately, for continuous-time systems, the Lebesgue measure is no longer valid. Instead, our reference measure R will be the probability measure induced by a diffusion network with dispersion σ, initial distribution ν but with no drift,[7] so that $R(A)$ represents the probability that such a network generates paths lying in the set A.

An important theorem in stochastic calculus, known as Girsanov's theorem,[8] tells us how to compute the relative density of processes with the same diffusion term and different drifts (see Oksendal, 1998). Using Girsanov's

[6] The Lebesgue measure of an interval (a, b) is $b - a$, the length of that interval.

[7] More formally, for each $A \in \mathcal{F}$, $R(A) = \int_{\mathbb{R}^n} R^u(A)\, d\nu(u)$ where for each $u \in \mathbb{R}^n$ the measure R^u on $(\Omega, \mathcal{F})$ is such that under it, the process $B = \{B(t, x) = (x(t) - x(0))/\sigma\colon t \in [0, T], x \in \Omega\}$ is a standard n-dimensional Brownian motion and $R^u(x(0) = u) = 1$.

[8] The conditions on μ mentioned in the introduction are sufficient for Girsanov's theorem to hold.

theorem, it can be shown that

$$L^\lambda(x) = \frac{dQ^\lambda}{dR}(x) = \exp\left\{\frac{1}{\sigma^2}\int_0^T \mu(x(t),\lambda)\cdot dx(t) - \frac{1}{2\sigma^2}\int_0^T |\mu(x(t),\lambda)|^2\,dt\right\} \tag{3.1}$$

$$= \exp\left\{\sum_{j=1}^n \left(\frac{1}{\sigma^2}\int_0^T \mu_j(x(t),\lambda)\,dx_j(t) - \frac{1}{2\sigma^2}\int_0^T (\mu_j(x(t),\lambda))^2\,dt\right)\right\}, \tag{3.2}$$

for $x \in \Omega$. The integral $\int_0^T \mu_j(x(t),\lambda)\,dx_j(t)$ is an Itô stochastic integral (see Oksendal, 1998). Intuitively, we can think of it as the (mean square) limit, as $\Delta t \to 0$ of the sum: $\sum_{k=0}^{l-1} \mu_j(x(t_k),\lambda)(x_j(t_{k+1}) - x_j(t_k))$, where $0 = t_0 < t_1 \cdots < t_l = T$ are the sampling times and $t_{k+1} = t_k + \Delta t$, and $\Delta t > 0$ is the sampling period. The term L^λ is a Radon-Nikodym derivative. Intuitively, it represents the likelihood of a diffusion network with parameter λ generating the path x relative to the likelihood for the reference diffusion network with no drift. For a fixed path $x \in \Omega$, the term $L^\lambda(x)$ can be treated as a likelihood function[9] of λ.

If there are no hidden units, $d = n$, $Q^\lambda = Q_o^\lambda$, and thus we can take $R_o = R$ and $L_o^\lambda = L^\lambda$. If there are hidden units, more work is required. For the construction here, we impose the condition that the initial probability measure ν is a product measure $\nu = \nu_o \times \nu_h$, where ν_o, ν_h are probability measures on $\mathbb{R}^d$ and $\mathbb{R}^{n-d}$, respectively.[10] It then follows from the independence of the components of Brownian motion that

$$R(A_o \times A_h) = R_o(A_o)R_h(A_h), \tag{3.3}$$

for all $A_o \in \mathcal{F}_o$, $A_h \in \mathcal{F}_h$, where

$$R_o(A_o) = R(A_o \times \Omega_h), \tag{3.4}$$

$$R_h(A_h) = R(\Omega_o \times A_h). \tag{3.5}$$

[9] Unfortunately, since R depends on σ, L^λ cannot be treated as a likelihood function of σ. For this reason, estimation of σ needs to be treated differently from estimation of λ for continuous-time problems. We leave the issue of continuous-time estimation of σ for future work.

[10] Here, $\mathbb{R}^d$ and $\mathbb{R}^{n-d}$ are endowed with their Borel sigma algebras, which are generated by the open sets in these spaces.

To find an appropriate density for Q_o^λ, note that for $A_o \in \mathcal{F}_o$,

$$Q_o^\lambda(A_o) = Q^\lambda(A_o \times \Omega_h) \tag{3.6}$$

$$= \int_{A_o} \int_{\Omega_h} L^\lambda(x_o, x_h)\, dR_h(x_h)\, dR_o(x_o), \tag{3.7}$$

and therefore the density of Q_o^λ with respect to R_o is

$$L_o^\lambda(x_o) = \frac{dQ_o^\lambda}{dR_o}(x_o) = \int_{\Omega_h} L^\lambda(x_o, x_h)\, dR_h(x_h), \text{ for } x_o \in \Omega_o. \tag{3.8}$$

Similarly, the density of Q_h^λ with respect to R_h is as follows:

$$L_h^\lambda(x_h) = \frac{dQ_h^\lambda}{dR_h}(x_h) = \int_{\Omega_o} L^\lambda(x_o, x_h)\, dR_o(x_o), \text{ for } x_h \in \Omega_h. \tag{3.9}$$

4 Log-Likelihood Gradients

Let P_o represent a probability measure on $(\Omega_o, \mathcal{F}_o)$ that we wish to approximate (e.g., a distribution of paths determined by the environment). Our goal is to find a probability measure from the family $\mathcal{Q}_o$ that best matches P_o. The hope is that this will provide an approximate model of P_o, the environment, that could be used for tasks such as sequence recognition, sequence generation, or stochastic filtering. We approach this modeling problem by defining a Kullback-Leibler distance (White, 1996) between P_o and Q_o^λ:

$$E^{P_o}\left(\log \frac{\Lambda_o}{L_o^\lambda}\right), \tag{4.1}$$

where E^{P_o} is an expected value with respect to P_o and $\Lambda_o = dP_o/dR_o$ is the density of the desired probability measure.[11] We seek values of λ that minimize equation 4.1. In practice, we estimate such values by obtaining $\tilde{n}$ fair sample paths[12] $\{x_o^i\}_{i=1}^{\tilde{n}}$ from P_o and seeking values of λ that maximize the following function:

$$\Phi(\lambda) = \frac{1}{\tilde{n}} \left(\sum_{i=1}^{\tilde{n}} \log L_o^\lambda(x_o^i) \right) - \Psi(\lambda). \tag{4.2}$$

[11] We are assuming that the expectation in equation 4.1 is well defined and finite and in particular Λ_o exists, which is also an implicit assumption in finite dimensional density estimation approaches.

[12] Fair samples are samples obtained in an independent and identically distributed manner.

This function is a combination of the empirical log-likelihood and a regularizer Ψ: $\mathbb{R}^p \to \mathbb{R}$ that encodes prior knowledge about desirable values of λ (Bishop, 1995). To simplify the notation, hereafter we present results for $\tilde{n} = 1$ and $\Psi(\lambda) = 0$ for all $\lambda \in \mathbb{R}^p$. Generalizing the analysis to $\tilde{n} > 1$ and interesting regularizers is easy but obscures the presentation.

The gradient of the log density of the observable paths with respect to λ is of interest to apply optimization techniques such as gradient ascent and the EM algorithm. If there are no hidden units, equation 3.1 gives the density of the observable measure, and differentiation yields[13]

$$\frac{\partial}{\partial \lambda_i} \log L^\lambda(x) = \frac{1}{\sigma^2} \sum_{j=1}^{n} \left(\int_0^T \frac{\partial \mu_j(x(t), \lambda)}{\partial \lambda_i} \, dx_j(t) - \int_0^T \frac{\partial \mu_j(x(t), \lambda)}{\partial \lambda_i} \mu_j(x(t), \lambda) \, dt \right) \tag{4.3}$$

$$= \frac{1}{\sigma^2} \sum_{j=1}^{n} \int_0^T \frac{\partial \mu_j(x(t), \lambda)}{\partial \lambda_i} \, dI_j^\lambda(x, t), \text{ for } x \in \Omega, \tag{4.4}$$

where $\sigma^{-1} I^\lambda$ is a (joint) innovation process (Poor, 1994):

$$I^\lambda(x, t) = x(t) - x(0) - \int_0^t \mu(x(s), \lambda) \, ds. \tag{4.5}$$

Such a process is a standard n-dimensional Brownian motion under Q^λ.

5 Stochastic Teacher Forcing

If there are no hidden units, the likelihood and log-likelihood gradient of paths can be obtained directly via equations 3.1 and 4.4. If there are hidden units, we need to take expected values over hidden paths:

$$L_o^\lambda(x_o) = \int_{\Omega_h} L^\lambda(x_o, x_h) \, dR_h(x_h) = E^{R_h}(L^\lambda(x_o, \cdot)). \tag{5.1}$$

[13] Conditions that are sufficient to justify the differentiation leading to equation 4.4 are that the first and second partial derivatives of $\mu(u, \lambda)$ with respect to λ exist and together with μ are continuous in (u, λ) and satisfy a linear growth condition of the form $|\mu(u, \lambda)| \leq K_\lambda(1 + |u|)$, where K_λ can be chosen independent of λ whenever λ is restricted to a compact set in $\mathbb{R}^p$ (Levanony, Shwartz, & Zeitouni, 1990; Protter, 1990). These conditions are satisfied by the neural network drift (see equation 1.2).

In this article, we propose estimates of this likelihood obtained by adapting a technique known as Monte Carlo importance sampling (Fishman, 1996). Instead of averaging with respect to R_h, we average with respect to another distribution S_h and multiply the variables being averaged by a correction factor known as the importance function. This correction factor guarantees that the new Monte Carlo estimates will remain unbiased. However, by using an appropriate sampling distribution S_h, the estimates may be more efficient than if we just sample from R_h (they may require fewer samples to obtain a desired level of precision). Following equation 2.11, we have that

$$L_o^\lambda(x_o) = \int_{\Omega_h} L^\lambda(x_o, x_h) \frac{dR_h}{dS_h}(x_h)\, dS_h(x_h) = E^{S_h}\left(L^\lambda(x_o, \cdot) \frac{dR_h}{dS_h}(\cdot) \right), \quad (5.2)$$

where S_h is a fixed distribution on $(\Omega_h, \mathcal{F}_h)$ for which the density dR_h/dS_h exists. This density thus acts as the desired importance function. We can obtain unbiased Monte Carlo estimates of the expected value in equation 5.2 by averaging over a set of hidden paths $\mathcal{H} = \{h^1, \ldots, h^m\}$ sampled from S_h:

$$\hat{L}_o^\lambda(x_o) = \sum_{l=1}^{m} p^\lambda(x_o, h^l), \quad (5.3)$$

where

$$p^\lambda(x_o, h) = \begin{cases} \frac{1}{m} L^\lambda(x_o, h) \frac{dR_h}{dS_h}(h), & \text{for } h \in \mathcal{H}, \\ 0 & \text{else.} \end{cases} \quad (5.4)$$

We use the gradient of the log of this density estimate for training the network

$$\nabla_\lambda \log \hat{L}_o^\lambda(x_o) = \sum_{l=1}^{m} p_{h|o}^\lambda(h^l \mid x_o) \nabla_\lambda \log L^\lambda(x_o, h^l), \quad (5.5)$$

where

$$p_{h|o}^\lambda(h \mid x_o) = \frac{p^\lambda(x_o, h)}{p_o^\lambda(x_o)}, \text{ for } h \in \Omega_h, \quad (5.6)$$

and

$$p_o^\lambda(x_o) = \hat{L}_o^\lambda(x_o) = \sum_{l=1}^{m} p^\lambda(x_o, h^l). \quad (5.7)$$

Note $\sum_{h\in\Omega_h} p^{\lambda}_{h|o}(h \mid x_o) = 1$, and thus we can think of $p^{\lambda}_{h|o}(\cdot \mid x_o)$ as a probability mass function on Ω_h. Moreover, $\nabla_\lambda \log \hat{L}^{\lambda}_{o}(x_o)$ is the average of the joint log-likelihood gradient $\nabla_\lambda \log L^{\lambda}(x_o, \cdot)$ with respect to that probability mass function.

While the results presented here are general and work for any sampling distribution for which dR_h/dS_h exists, it is important to find a distribution S_h from which we know how to sample, for which we know dR_h/dS_h, and for which the Monte Carlo estimates are relatively efficient (i.e., do not require a large number of samples to achieve a desired reliability level). An obvious choice is to sample from R_h itself, in which case $dR_h/dS_h = 1$. In fact, this is the approach we used in previous versions of this article (Mineiro, Movellan, & Williams, 1998) and which was also recently proposed in Solo (2000). The problem with sampling from R_h is that as learning progresses, the Monte Carlo estimates become less and less reliable, and thus there is a need for better sampling distributions that change as learning progresses. We have obtained good results by sampling in a manner reminiscent of the teacher forcing method from deterministic neural networks (Hertz, Krogh, & Palmer, 1991). The idea is to obtain a sample of hidden paths from a network whose observable units have been forced to exhibit the desired observable path x_o. Consider a diffusion network with parameter vector $\lambda_s \in \mathbb{R}^p$, not necessarily equal to λ. The role of this network will be to provide sample hidden paths. For each time $t \in [0, T]$, we fix the output units of such a network to $x_o(t)$, and let the hidden units run according to their natural dynamics,

$$\begin{aligned} dH(t) &= \mu_h(x_o(t), H(t), \lambda_s)dt + \sigma dB_h(t), \\ H(0) &\sim \nu_h, \end{aligned} \tag{5.8}$$

where μ_h is the hidden component of the drift. We repeat this procedure m times to obtain the sample of hidden paths $\mathcal{H} = \{h^l\}_{l=1}^{m}$. One advantage of this sampling scheme is that we can use Girsanov's theorem to obtain the desired importance function,

$$\begin{aligned} \frac{dR_h}{dS_h}(x_h) = \exp\Bigg\{ &-\frac{1}{\sigma^2}\int_0^T \mu_h(x_o(t), x_h(t), \lambda_s)\cdot dx_h(t) \\ &+\frac{1}{2\sigma^2}\int_0^T |\mu_h(x_o(t), x_h(t), \lambda_s)|^2\, dt\Bigg\}, \end{aligned} \tag{5.9}$$

where S_h, the sampling distribution, is now the distribution of hidden paths induced by the network in equation 5.8 with fixed parameter λ_s. If we let

$\lambda_s = \lambda$ then the p^λ function defined in equation 5.4 simplifies as follows:

$$\begin{aligned} p^\lambda(x_o, h) &= \frac{1}{m} L^\lambda(x_o, h) \frac{dR_h}{dS_h}(h) \\ &= \frac{1}{m} \exp\left\{ \frac{1}{\sigma^2} \int_0^T \mu_o(x_o(t), h(t), \lambda) \cdot dx_o(t) \right. \\ &\qquad \left. - \frac{1}{2\sigma^2} \int_0^T |\mu_o(x_o(t), h(t), \lambda)|^2 \, dt \right\}, \\ &\quad \text{for } h \in \mathcal{H}. \end{aligned} \tag{5.10}$$

6 Monte Carlo EM learning

Given a fixed sampling distribution S_h and a fixed set of hidden paths $\mathcal{H} = \{h^1, \ldots, h^m\}$ sampled from S_h, our objective is to find values of λ that maximize the estimate of the likelihood $\hat{L}_o^\lambda(x_o)$. We can search for such values using standard iterative procedures, like gradient ascent or conjugate gradient. Another iterative approach of interest is the EM algorithm (Dempster, Laird, & Rubin, 1977). On each iteration of the EM algorithm, we start with a fixed parameter vector $\bar{\lambda}$ and search for values of λ that optimize the following expression:[14]

$$M(\bar{\lambda}, \lambda, x_o) = \sum_{l=1}^{m} p_{h|o}^{\bar{\lambda}}(h^l \mid x_o) \log p^\lambda(x_o, h^l). \tag{6.1}$$

Note that since $\sum_{l=1}^m p_{h|o}^{\bar{\lambda}}(h^l \mid x_o) = 1$ and $p_o^\lambda(x_o) = p^\lambda(x_o, h^l)/p_{h|o}^\lambda(h^l \mid x_o)$, then

$$\log \hat{L}_o^\lambda(x_o) = \log p_o^\lambda(x_o) = \sum_{l=1}^{m} p_{h|o}^{\bar{\lambda}}(h^l \mid x_o) \log \frac{p^\lambda(x_o, h^l)}{p_{h|o}^\lambda(h^l \mid x_o)} \tag{6.2}$$

$$= M(\bar{\lambda}, \lambda, x_o) - \sum_{l=1}^{m} p_{h|o}^{\bar{\lambda}}(h^l \mid x_o) \log p_{h|o}^\lambda(h^l \mid x_o), \tag{6.3}$$

and

$$\begin{aligned} \log \hat{L}_o^\lambda(x_o) - \log \hat{L}_o^{\bar{\lambda}}(x_o) &= M(\bar{\lambda}, \lambda, x_o) - M(\bar{\lambda}, \bar{\lambda}, x_o) \\ &\quad + \mathrm{KL}(p_{h|o}^{\bar{\lambda}}(\cdot \mid x_o), p_{h|o}^\lambda(\cdot \mid x_o)), \end{aligned} \tag{6.4}$$

[14] If a regularizer is used, redefine $p^\lambda(x_o, h)$ in equation 5.4 as $L^\lambda(x_o, h) dR_h/dS_h(h) \exp(-\Psi(\lambda))$.

where KL stands for the Kullback-Leibler distance between the functions $p^{\bar{\lambda}}_{h|o}(\cdot \mid x_o)$, and $p^{\lambda}_{h|o}(\cdot \mid x_o)$. Since KL is nonnegative, it follows that if $M(\bar{\lambda}, \lambda, x_o) > M(\bar{\lambda}, \bar{\lambda}, x_o)$, then $\hat{L}^{\lambda}_o(x_o) > \hat{L}^{\bar{\lambda}}_o(x_o)$. Since this adaptation of the EM algorithm maximizes an estimate of the log likelihood instead of the log likelihood itself, we refer to it as stochastic EM.

On each iteration of the stochastic EM procedure, we find a value of λ that maximizes $M(\bar{\lambda}, \lambda, x_o)$, and we let $\bar{\lambda}$ take that value. The procedure guarantees that the estimate of the likelihood of the observed path $\hat{L}^{\lambda}_o(x_o)$ will increase or stay the same, at which point we have converged. Our approach guarantees convergence only when the sampling distribution S_h and the sample of hidden paths $\mathcal{H}$ are fixed. In practice, we have found that even with relatively small sample sizes, it is beneficial to change S_h and $\mathcal{H}$ as learning progresses. For instance, in the approach proposed at the end of section 7, we change the sampling distribution and the sample paths after each iteration of the EM algorithm. While we have not carefully analyzed the properties of this approach, we believe the reason that it works well is that as learning progresses, it uses sampling distributions S_h that are better suited for the values of λ that EM moves into.

7 The Neural Network Case

Up to now we have presented the learning algorithm in a general manner applicable to generic SDE models. In this section, we show how the algorithm applies to diffusion neural networks. Let w_{sr} be the component of λ representing the synaptic weight (conductance) from the sending unit s to the receiving unit r. For the neural network drift, we have

$$\frac{\partial \mu_j(x(t), \lambda)}{\partial w_{sr}} = \kappa_j \frac{\partial}{\partial w_{sr}} \sum_{i=1}^{n} \varphi(x_i(t)) w_{ij} = \delta_{jr} \kappa_r \varphi(x_s(t)), \text{ for } x \in \Omega, \tag{7.1}$$

where δ is the Kronecker delta function ($\delta_{jr} = 1$ if $j = r$, 0 else) and x_i stands for the ith component of x (i.e., the activation of unit i at time t in path x).

7.1 Networks Without Hidden Units. Combining equations 4.4 and 7.1, we get

$$\frac{\partial \log L^{\lambda}(x)}{\partial w_{sr}} = \frac{1}{\sigma^2} \sum_{j=1}^{n} \int_0^T \frac{\partial \mu_j(x(t), \lambda)}{\partial w_{sr}} \, dI^{\lambda}_j(x, t) \tag{7.2}$$

$$= \frac{1}{\sigma^2} \sum_{j=1}^{n} \delta_{jr} \kappa_r \int_0^T \varphi(x_s(t)) \, dI^{\lambda}_j(x, t) \tag{7.3}$$

$$= \frac{1}{\sigma^2} \kappa_r \int_0^T \varphi(x_s(t)) \, dI^{\lambda}_r(x, t), \tag{7.4}$$

where $\sigma^{-1} I_r^\lambda$ is the innovation process of the receiving unit, that is,

$$dI_r^\lambda(x,t) = dx_r(t) - \mu_r(x(t),\lambda)dt \tag{7.5}$$
$$= dx_r(t) + \kappa_r \rho_r x_r(t)dt - \kappa_r \xi_r dt - \kappa_r \sum_{i=1}^{n} \varphi(x_i(t)) w_{ir} dt.$$

Combining equations 7.4 and 7.5, we get

$$\frac{\partial \log L^\lambda(x)}{\partial w_{sr}} = \frac{1}{\sigma^2}\kappa_r^2 \left(b_{sr}(x) - \sum_{i=1}^{n} a_{si}(x) w_{ir} \right), \tag{7.6}$$

where

$$b_{sr}(x) = \frac{1}{\kappa_r}\int_0^T \varphi(x_s(t))\, dx_r(t) + \rho_r \int_0^T \varphi(x_s(t)) x_r(t)\, dt$$
$$- \xi_r \int_0^T \varphi(x_s(t))\, dt, \tag{7.7}$$

$$a_{si}(x) = \int_0^T \varphi(x_s(t))\varphi(x_i(t))\, dt. \tag{7.8}$$

Let $\nabla_w \log L^\lambda(x)$ be an $n \times n$ matrix with cell i, j containing the derivative of $\log L^\lambda(x)$ with respect to w_{ij} for $i, j = 1, \ldots, n$. It follows that

$$\nabla_w \log L^\lambda(x) = \frac{1}{\sigma^2}(b(x) - a(x)w)\mathcal{K}^2, \tag{7.9}$$

where $\mathcal{K}$ is a diagonal matrix with diagonal elements $\kappa_1, \ldots, \kappa_n$. Note that all the terms involved in the computation of the gradient of the log likelihood are Hebbian; they involve time integrals of pairwise activation products. Also note that each cell of the matrix a is an inner product in the Hilbert space of squared integrable functions. This may open avenues for generalizing standard kernel methods (Aizerman et al., 1964; Burges, 1998) for learning distributions of sequences.

The matrix $a(x)$ is positive semidefinite. To see why, let $v \in \mathbb{R}^n$, $y(t) = \sum_{j=1}^n v_j \varphi(x_j(t))$ for all $t \in [0, T]$, and note that

$$\int_0^T y(t)^2\, dt = v'av \geq 0. \tag{7.10}$$

Thus, if $a(x)$ is invertible, there is a unique maximum for the log-likelihood function. The maximum likelihood estimate of w follows:

$$\hat{w} = (a(x))^{-1} b(x). \tag{7.11}$$

A similar procedure can be followed to find the maximum likelihood estimates of the other parameters:

$$\hat{\xi}_r = \frac{(x_r(T) - x_r(0))\kappa_r^{-1} + \int_0^T \left(\rho_r x_r(t) \ - \sum_{j=1}^n w_{jr}\varphi(x_j(t))\right) dt}{T}, \quad (7.12)$$

$$\hat{\rho}_r = \frac{\int_0^T x_r(t)\xi_r \, dt + \sum_{j=1}^n \varphi(x_j(t))w_{jr}x_r(t)dt - \kappa_r^{-1}\int_0^T x_r(t)\, dx_r(t)}{\int_0^T x_r^2(t)\, dt}, \quad (7.13)$$

$$\hat{\kappa}_r = \frac{\int_0^T \bar{\mu}_r(x(t), \lambda)\, dx_r(t)}{\int_0^T \bar{\mu}_r(x(t), \lambda)^2\, dt}, \text{ for } r = 1, \ldots, n, \quad (7.14)$$

where $\bar{\mu}_r(x(t), \lambda) = \mu_r(x(t), \lambda)/\kappa_r$, which is not a function of κ_r. Equations 7.11 through 7.14 maximize a parameter or set of parameters assuming the other parameters are fixed.

7.2 Networks with Hidden Units. If there are hidden units, we use the stochastic EM approach presented in section 6. Given an observable path, $x_o \in \Omega_o$, we obtain a fair sample of hidden paths $\mathcal{H}_m = \{h^1, \ldots, h^m\}$ from a sampling distribution S_h. The gradient of the log–likelihood estimate with respect to w is as follows:

$$\nabla_w \log \hat{L}_o^\lambda(x_o) = \frac{1}{\sigma^2}(\tilde{b}^\lambda(x_o) - \tilde{a}^\lambda(x_o)w)\mathcal{K}^2, \quad (7.15)$$

where

$$\tilde{b}^\lambda(x_o) = \sum_{l=1}^m p_{h|o}^\lambda(h^l \mid x_o)b(x_o, h^l), \quad (7.16)$$

$$\tilde{a}^\lambda(x_o) = \sum_{l=1}^m p_{h|o}^\lambda(h^l \mid x_o)a(x_o, h^l), \quad (7.17)$$

and a, b are given by equations 7.7 and 7.8. Note that in this case, the $\tilde{a}$ and $\tilde{b}$ coefficients depend on w in a nonlinear manner in general, even if the activation function φ is linear. We can find values of w for which the gradient vanishes by using standard iterative procedures like gradient ascent or conjugate gradient. The situation simplifies when the EM algorithm is used. Let $\bar{\lambda}$ and λ represent the parameter vectors of two n-dimensional diffusion networks that have the same values for the ξ, κ and ρ terms. Let $\bar{w}$ and w represent the connectivity matrices of these two networks. Following the argument presented in section 6, we seek values of w that maximize $M(\bar{\lambda}, \lambda, x_o)$. To do so, we first find the gradient with respect to w:

$$\nabla_w M(\bar{\lambda}, \lambda, x_o) = \frac{1}{\sigma^2}(\tilde{b}^{\bar{\lambda}}(x_o) - \tilde{a}^{\bar{\lambda}}(x_o)w)\mathcal{K}^2. \quad (7.18)$$

The matrix $\tilde{a}^{\bar{\lambda}}(x_o)$ is an average of positive semidefinite matrices, and thus it is also positive semidefinite. Thus, if $\tilde{a}^{\bar{\lambda}}$ is invertible, we can directly maximize $M(\bar{\lambda}, \lambda, x_o)$ with respect to w by setting the gradient to zero and solving for w. The solution,

$$\hat{w} = (\tilde{a}^{\bar{\lambda}}(x_o))^{-1}\tilde{b}^{\bar{\lambda}}(x_o), \tag{7.19}$$

becomes the new $\bar{w}$, and equation 7.19 is iterated until convergence is achieved. Note that this procedure is iterative and guarantees convergence only to a local maximum of the estimate of the likelihood function. A similar procedure can be followed to train the ξ, κ, and ρ parameters.

Summary of the Stochastic EM Learning Algorithm for the Neural Network Case

1. Choose an initial network with parameter $\bar{\lambda} \in \mathbb{R}^p$ including the values of $\bar{w}$, $\bar{\xi}$, $\bar{\kappa}$, and $\bar{\rho}$. The initial connectivity matrix $\bar{w}$ is commonly the zero matrix. Hereafter, we concentrate on how to train the connectivity matrix. Training the other parameters in $\bar{\lambda}$ is straightforward.
2. With the observation units forced to exhibit a desired sequence x_o, run the hidden part of the network m times, starting at time 0 and ending at time T, to obtain m different hidden paths: $h^1, \ldots, h^m$.
3. Compute the weight, here represented as $\pi(\cdot)$, of each hidden path:

$$\pi(l) = \exp\left\{ \frac{1}{\sigma^2} \sum_{j=1}^{d} \int_0^T \mu_j(x^l(t), \bar{\lambda})\, dx_j^l(t) - \frac{1}{2\sigma^2} \int_0^T (\mu_j(x^l(t), \bar{\lambda}))^2\, dt \right\}, \tag{7.20}$$

   for $l = 1, \ldots, m$, where $x^l = (x_o, h^l)$ is a joint sequence consisting of the desired sequence x_o for the observation units and the sampled sequence h^l for the hidden units, and the index $j = 1, \ldots, d$ goes over the output units. In practice, the integrals in equation 7.20 are approximated using discrete time approximations, as described in section 9.
4. Compute the a and b matrices of each hidden sequence:

$$a_{ij}(l) = \int_0^T \varphi(x_i^l(t))\varphi(x_j^l(t))\, dt, \text{ for } i, j = 1, \ldots, n, \tag{7.21}$$

$$b_{ij}(l) = \frac{1}{\kappa_j} \int_0^T \varphi(x_i^l(t))\, dx_j^l(t) + \rho_j \int_0^T \varphi(x_i^l(t)) x_j^l(t)\, dt - \xi_j^l \int_0^T \varphi(x_i^l(t))\, dt, \text{ for } i, j = 1, \ldots, n. \tag{7.22}$$

5. Compute the averaged matrices $\tilde{a}$ and $\tilde{b}$:

$$\tilde{a} = \frac{\sum_{l=1}^{m} \pi(l) a(l)}{\sum_{l=1}^{m} \pi(l)}, \tag{7.23}$$

$$\tilde{b} = \frac{\sum_{l=1}^{m} \pi(l) b(l)}{\sum_{l=1}^{m} \pi(l)}. \tag{7.24}$$

6. Update the connectivity matrix:

$$\bar{w} = \tilde{a}^{-1} \tilde{b}. \tag{7.25}$$

7. Go back to step 2 using the new value of $\bar{w}$.

8 Comparison to Previous Work

We use diffusion networks as a way to parameterize distributions of continuous-time varying signals. We proposed methods for finding local maxima of a likelihood estimate. This process is known as learning in the neural network literature, system identification in the engineering literature, and parameter estimation in the statistical literature. The main motivation for the diffusion network approach is to combine the versatility of recurrent neural networks (Pearlmutter, 1995) with the well-known advantages of stochastic models(Oksendal, 1998). Thus, our work is closely related to the literature on continuous-time recurrent neural networks and the literature on stochastic filtering.

8.1 Continuous-Time Recurrent Neural Networks. We use a recurrent neural network drift function and allow full interconnectivity between hidden and observable units, as is standard in neural network applications. In recurrent neural networks, the dynamics are deterministic ($\sigma = 0$), while in diffusion networks they are not ($\sigma > 0$). Learning algorithms for recurrent neural networks typically find values of the parameter vector λ that minimize a mean squared error of the following form (Pearlmutter, 1995),

$$\Phi(\lambda) = \int_0^T (o(t, \lambda) - r(t))^2 \, dt, \tag{8.1}$$

where r is a desired path we want the network to learn and $o(\cdot, \lambda)$ is the unique observable path produced by a recurrent neural network with parameter vector λ. Since the optimal properties of maximum-likelihood methods are well understood mathematically, it is of interest to find under what conditions the solutions found by minimizing equation 8.1 are maximum-likelihood estimates. This will give us a sense of the likelihood model implicitly used by standard learning algorithms for deterministic neural networks.

For a given deterministic neural network with parameter vector λ, we define a stochastic process O^λ by adding white noise to the unique observable path $o(\cdot, \lambda)$ produced by that network:

$$O^\lambda(t) = o(t, \lambda) + \sigma W(t). \tag{8.2}$$

To obtain a mathematical interpretation of equation 8.2, we introduce

$$Z^\lambda(t) = \int_0^t O^\lambda(s)\, ds, \tag{8.3}$$

which is thus governed by the following SDE,

$$dZ^\lambda(t) = o(t, \lambda)dt + \sigma dB(t), \tag{8.4}$$

where B is standard Brownian motion. Note that if Z^λ is known, O^λ is known and vice versa, so no information is lost by working with Z^λ instead of O^λ. To find the likelihood of a path o given a network with parameter vector λ, we first compute the integral trajectory z:

$$z(t) = \int_0^t o(s)\, ds. \tag{8.5}$$

Using Girsanov's theorem, the likelihood of z given that is generated as a realization of the SDE model in equation 8.4 can be shown to be as follows:

$$L^\lambda(z) = \frac{1}{\sigma^2} \int_0^T o(t, \lambda)\, dz(t) - \frac{1}{2\sigma^2} \int_0^T o(t, \lambda)^2\, dt. \tag{8.6}$$

Considering that $dz(t) = o(t)dt$, it is easy to see that maximizing $L^\lambda(z)$ with respect to λ is equivalent to minimizing $\Phi(\lambda)$ of equation 8.1.

Thus, standard continuous-time neural network learning algorithms can be seen as performing maximum likelihood estimation with respect to the stochastic process defined in equation 8.2. In other words, the generative model underlying those algorithms consists of adding white noise to the unique observable trajectory generated by a network with deterministic dynamics. Note that this is an extremely poor generative model. In particular, this model cannot handle bifurcations and time warping, two common sources of variability in natural signals.

Instead of adding white noise to a deterministic path, in diffusion networks, we add white noise to the activation increments; that is, the activations are governed by an SDE of the form

$$dX^\lambda(t) = \mu(X(t), \lambda)dt + \sigma dB(t). \tag{8.7}$$

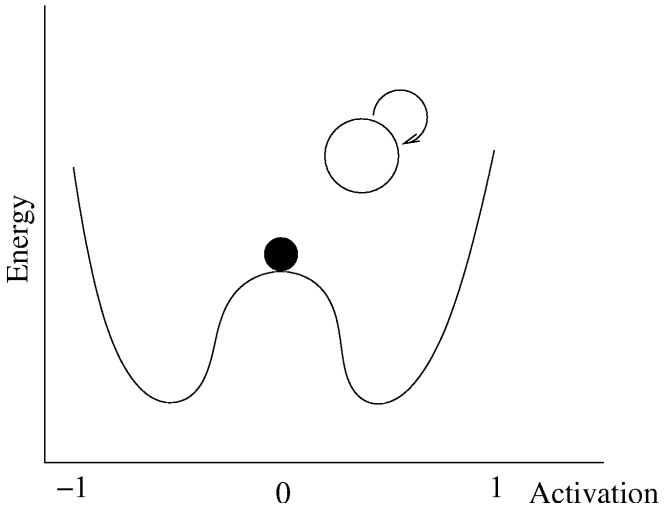

Figure 3: An illustration of the different effects of adding noise to the output of a deterministic network versus adding it to the network dynamics. See the text for an explanation.

This results in a much richer and more realistic likelihood model. Due to the probabilistic nature of the state transitions, the approach is resistant to time warping, for the same reason hidden Markov models are. In addition, the approach can also handle bifurcations. Figure 3 illustrates this point with a simple example. The figure shows a single neuron model in which the drift is proportional to an energy function with two wells. If we let the network start at the origin and there is no internal noise, the activation of the neuron will be constant, since the drift at the origin is zero. Adding white noise to this constant activation will result in a unimodal gaussian distribution centered at the origin. However, if we add noise to the activation dynamics, the neuron will bifurcate, sometimes moving toward the left well, sometimes toward the right well. This will result in a bimodal distribution of activations.

Diffusion networks were originally formulated in Movellan and McClelland (1993). Movellan (1994) presented an algorithm to train diffusion networks to approximate expected values of sequence distributions. Movellan (1998) presented algorithms for training equilibrium probability densities. Movellan and McClelland (2001) showed the relationship between diffusion networks and classical psychophysical models of information integration. Mineiro et al. (1998) presented an algorithm for sequence density estimation with diffusion networks. In that article, we used a gradient-descent approach instead of the EM approach presented here, and we sampled from

the reference distribution R_h instead of the distribution of hidden states with clamped observations. These two modifications greatly increased learning speed and made possible the use of diffusion networks on realistic problems involving thousands of parameters.

8.2 Stochastic Filtering. Our work is also closely related to the literature on stochastic filtering and systems identification. As mentioned in section 1.3, if the logistic activation function φ is replaced by a linear function, the weights between the observable units and from the observable to the hidden units are set to zero, the ρ terms are set to zero, and the probability distribution of the initial states is constrained to be gaussian, diffusion networks have the same dynamics as the continuous-time Kalman-Bucy filter (Kalman & Bucy, 1961). While the usefulness of the Kalman-Bucy filter is widely recognized, its limitations (due to the linear and gaussian assumptions) have become clear. For this reason, many extensions of the Kalman filter have been proposed (Lewis, 1986; Fahrmeir, 1992; Meinhold & Singpurwalla, 1975; Sage & Melsa, 1971; Kitagawa, 1996). This article contributes to that literature by proposing a new approach for training partially observable SDE models. An important difference between our work and stochastic filtering approaches is that stochastic filtering is generally restricted to models of the following form:

$$dH(t) = \mu_h(H(t))dt + \sigma dB_h(t), \tag{8.8}$$

$$dO(t) = \mu_o(H(t))dt + \sigma dB_o(t). \tag{8.9}$$

In our case, this restriction would require setting to zero the coupling between observable units and the feedback coupling from observable units to hidden units. While this restriction simplifies the mathematics of the problem, having a general algorithm that does not require it is important for the following reasons: (1) such restriction is not commonly used in the neural network literature; (2) when modeling biological neural networks, such restriction is not realistic; (3) in many physical processes, the observations are coupled by inertia and dampening processes (e.g., bone and muscles with large dampening properties are involved in the production of observed motor sequences); and (4) in practice, the existence of connections between observable units results in a much richer set of distribution models.

Iterative solutions for estimation of drift parameters of partially observable SDE models are well known in the statistics and stochastic filtering literatures. However, most current approaches are very expensive computationally, do not allow training fully coupled systems, and have been used only for problems with very few parameters. Campillo and Le Gland (1989) present an approach that involves numerical solution of stochastic partial differential equations. The approach has been applied to models with only a handful of parameters. Shumway and Stoffer (1982) present an exact EM approach for discrete-time linear systems; however, the approach does not

generalize to nonlinear systems. Ljung (1999) presents a general approach for discrete-time systems with nonlinear drifts. However, the approach requires the inversion of very large matrices (order of length of the training sequence times number of states). Ghahramani and Roweis (1999) present an ingenious method to learn discrete-time dynamical models with arbitrary drifts. The approach approximates nonlinear drift functions using gaussian radial basis functions. While promising, the approach has been shown to work on only a single parameter problem.

Our work is closely related to Kitagawa (1996), the work by Blake and colleagues (North & Blake, 1998; Blake, North, & Isard, 1999), and Solo (2000). Kitagawa (1996) presents a discrete-time Monte Carlo method for general-purpose nonlinear filtering and smoothing. While Kitagawa did not experiment with maximum likelihood estimation of drift parameters, he mentioned the possibility of doing so. Blake et al. (1999) independently presented a Monte Carlo EM algorithm at about the same time we published our earlier work on diffusion networks (Mineiro et al., 1998). The main differences between their approach and ours are as follows: (1) they work in discrete time while we work in continuous time; (2) they require that there be no coupling between observation units and no feedback coupling from observations to hidden units, whereas we do not have such requirements; and (3) they sample from an approximation to the distribution of hidden units given observable paths, as proposed by Kitagawa (1996). Instead, we sample from the distribution of hidden units with clamped observations and use corrections to obtain unbiased estimates of the likelihood gradients. Solo (2000) presents a simulation method for generating approximate likelihood functions for partially observable SDE models and for finding approximate maximum likelihood estimators. His approach can be seen as a discrete-time version of the method we independently proposed in Mineiro et al. (1998). The approach we propose here generalizes (Mineiro et al., 1998; Solo, 2000). We incorporate the use of importance sampling in continuous time and do not need to restrict the observation units to be uncoupled from each other or the hidden units to be uncoupled from the observation units.

9 Simulations

Sample source code for the simulations presented here can be found at J. R. M.'s Web site (mplab.ucsd.edu). In practice, lacking hardware implementations of diffusion networks, we model them on digital computers using discrete-time approximations. While more sophisticated techniques are available for simulating SDEs in digital computers (Kloeden & Platen, 1992; Karandikar, 1995), we opted to start with simple first-order Euler approximations.

Consider the term $\hat{L}_o^\lambda(x_o)$ in equation 5.3. It requires, among other things, computing integrals of the form

$$\log L^\lambda(x) = \frac{1}{\sigma^2}\int_0^T \mu(x(t),\lambda)\cdot dx(t) - \frac{1}{2\sigma^2}\int_0^T |\mu(x(t),\lambda)|^2\,dt, \quad \text{for } x \in \Omega. \tag{9.1}$$

In practice, we approximate such integrals using the following sum:

$$\frac{1}{\sigma^2}\sum_{k=0}^{s-1}\mu(x(t_k),\lambda)\cdot(x(t_{k+1})-x(t_k)) - \frac{1}{2\sigma^2}\sum_{k=0}^{s-1}|\mu(x(t_k),\lambda)|^2\Delta t, \tag{9.2}$$

where $0 = t_0 < t_1 \cdots < t_s = T$ are the sampling times and $t_{k+1} = t_k + \Delta t$, and $\Delta t > 0$ is the sampling period.

The Monte Carlo approach proposed here requires generating sample hidden paths $\{h^l\}_{l=1}^m$ from a network with the observable units clamped to the path x_o. We obtain these sample paths by simulating a diffusion network in discrete time as follows:

$$H(t_{k+1}) = H(t_k) + \mu_h(x_o(t_k), H(t_k), \lambda)\Delta t_k + \sigma Z_k\sqrt{\Delta t_k}, \tag{9.3}$$
$$H(0) \sim \nu_h,$$

where $Z_1, \ldots, Z_s$ are independent $(n-d)$-dimensional gaussian random vectors with zero mean and covariance equal to the identity matrix.

9.1 Example 1: Learning to Bifurcate. This simulation illustrates the capacity of diffusion networks with nonlinear activation functions to learn bifurcations. This is a property important for practical applications, which allows these networks to entertain multiple distinct hypotheses about the state of the world. Standard deterministic neural networks and linear stochastic networks like Kalman filters cannot learn bifurcations. A diffusion network with an observable unit and a hidden unit was trained on the distribution of paths shown in Figure 4. The top of the figure shows there were four equally probable desired paths, organized as a double bifurcation. The network was trained for 60 iterations of the EM algorithm, with 10 sample hidden paths per iteration. The bottom row shows the distribution learned by a standard recurrent neural network. As expected, the network learned the average path—a constant zero output. The trajectories displayed on the figure were produced by estimating the variance of the desired outputs from the learned trajectory and adding gaussian noise of that estimated variance to the output learned by the network.

The second row of Figure 4 shows 48 sample paths produced by the diffusion network after training. While not perfect, the network learned to

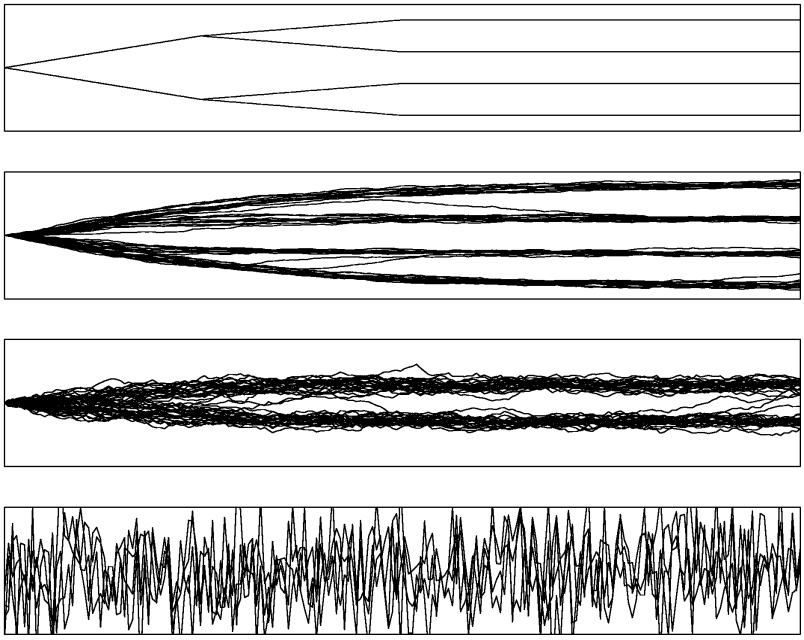

Figure 4: (Top row) The double bifurcation defines a desired path distribution with four equally probable paths. (Second row) Observable unit sequences obtained after training a diffusion network. (Third row) Hidden unit sequences of the trained diffusion network. (Bottom row) Sequences obtained after training a standard recurrent neural network and adding an optimal amount of noise to the output of that network.

bifurcate twice and produced a reasonable approximation to the desired distribution. We were surprised that this problem was learnable with a single hidden unit, until we saw the ingenious solution learned by the network (see Table 1). Basically the network learned to toss two binary random variables sequentially and compute their sum. The observable unit learned a relatively small time constant and, if disconnected from the hidden unit, it bifurcated once, with each branch taking approximately equal probability. The hidden unit had a faster time constant and learned to bifurcate unaffected by the observable unit. Thus, the hidden unit basically behaved as a Bernoulli random bias for the observable unit. The result was a double bifurcation of the observable unit, as desired. Figure 5 shows four example paths for the observable and hidden units.

9.2 Example 2: Learning a Beating Heart. In this section we show the capacity of diffusion networks to learn natural oscillations. The task was

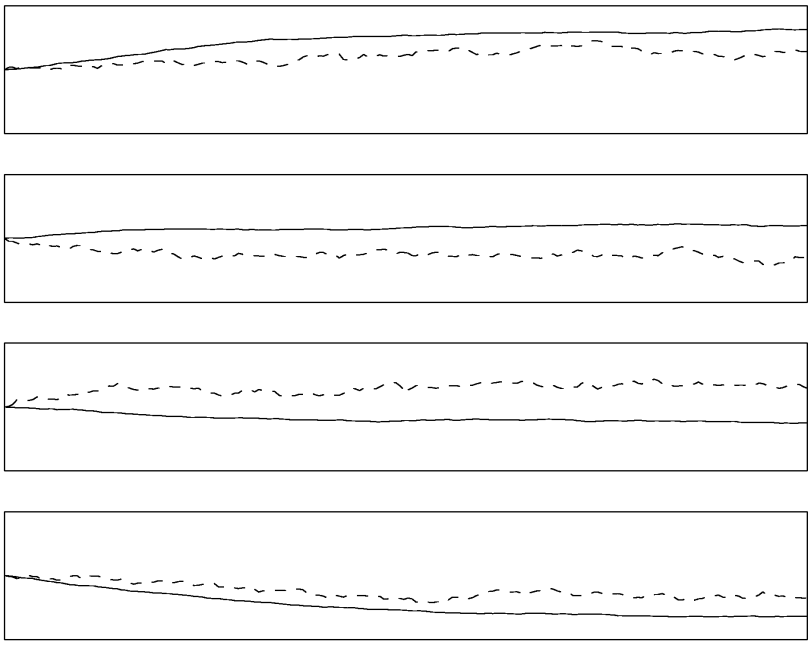

Figure 5: Four sample paths learned in the bifurcation problem. Solid lines represent observable unit paths, and dashed lines represent hidden unit paths.

Table 1: Parameters Learned for the Double Bifurcation Problem.

$w_{oo} = 4.339$	$w_{oh} = -0.008$	$\xi_o = -0.005$	$\kappa_o = 0.429$
$w_{ho} = 2.303$	$w_{hh} = 0.5912$	$\xi_h = 0.000$	$\kappa_h = 1.175$

Notes: All the weights and biases were initialized to zero and the κ terms to 1. The fixed parameters were set to the following values: $\theta_1 = -1, \theta_2 = 2, \theta_3 = 7, \sigma = 0.2$.

learning the expansion and contraction sequence of a beating heart. Once a distribution of normal sequences is learned, the network can be used for tracking sequences of moving heart images (Jacob, Noble, and Blake (1998) or to detect the presence of irregularities.

Jacob et al. (1998) tracked the contour of a beating heart from a sequence of ultrasound images and applied principal component analysis (PCA) to the resulting sequence of contours. Figure 6 shows the first principal component coefficient as a function of time for a typical sequence of contours. North and Blake (1998) compared two discrete-time models of the heart expansion sequence: (1) a linear model with time delays and no hidden units and (2) a linear model with time delays and hidden units. The first model was defined

by the following stochastic difference equation,

$$O(t+1) = O(t) + \lambda_1 + \lambda_2 O(t) + \lambda_3 O(t-1) + \sigma_1 Z_1(t), \tag{9.4}$$

where O represents the observed heart expansion, $Z_1(1), Z_1(2), \ldots$ is a sequence of independent and identically distributed gaussian random variables with zero mean and unit variance, and $\lambda_1, \lambda_2, \lambda_3 \in \mathbb{R}$, $\sigma_1 > 0$ are adaptive parameters. The second model had the following form,

$$H(t+1) = H(t) + \lambda_1 + \lambda_2 H(t) + \lambda_3 H(t-1) + \sigma_1 Z_1(t), \tag{9.5}$$

$$O(t) = H(t) + \sigma_2 Z_2(t), \tag{9.6}$$

where $Z_2(1), Z_2(2), \ldots$ is a sequence of zero mean, unit variance, independent gaussian random variables, and $\sigma_2 > 0$ is also adaptive. The two models were trained on the heart expansion sequence displayed in Figure 6. After training, the two models were run with $\sigma = 0$ to see the kind of process they had learned. The two linear models learned to oscillate at the correct frequency, but their motion was too damped (see Figure 6). It should be noted that both models are perfectly capable of learning undampened oscillations; they just could not learn this particular kind of oscillation.

Unfortunately, the original heart expansion data are not available. Instead, we recovered the data by digitizing a figure from the original article and automatically sampling it at 872 points. We tried the two linear systems on these data and obtained results identical to those reported in Jacob et al. (1998), so we feel the digitization process worked well. We then trained a diffusion network with one observation unit and one hidden unit. We did not include time delays and instead allowed the network to develop its own delayed representations by fully coupling the observation and hidden units. The network was trained using 60 iterations of the stochastic EM algorithm, with 10 hidden samples per iteration. Table 2 shows the parameters learned by the network. Figure 6 shows a typical sample path produced by the trained network. Note that the network did not exhibit the dampening problem found with the linear models. We also tried a diffusion network with one observation unit and one hidden unit, but with linear activation functions instead of logistic. The result was again a very dampened process, like the one depicted in Figure 6. It appears that the saturating nonlinearity of diffusion networks was beneficial for learning this task.

9.3 Example 3: Learning to Read Lips. In this section, we report on the use of diffusion networks with thousands of parameters for a sequence classification task involving a body of realistic data. We compare a diffusion network approach with the best hidden Markov model approaches published in the literature for this task. The question at hand is whether diffusion networks may be a viable alternative to hidden Markov models for sequence recognition problems. The main difference between diffusion networks and

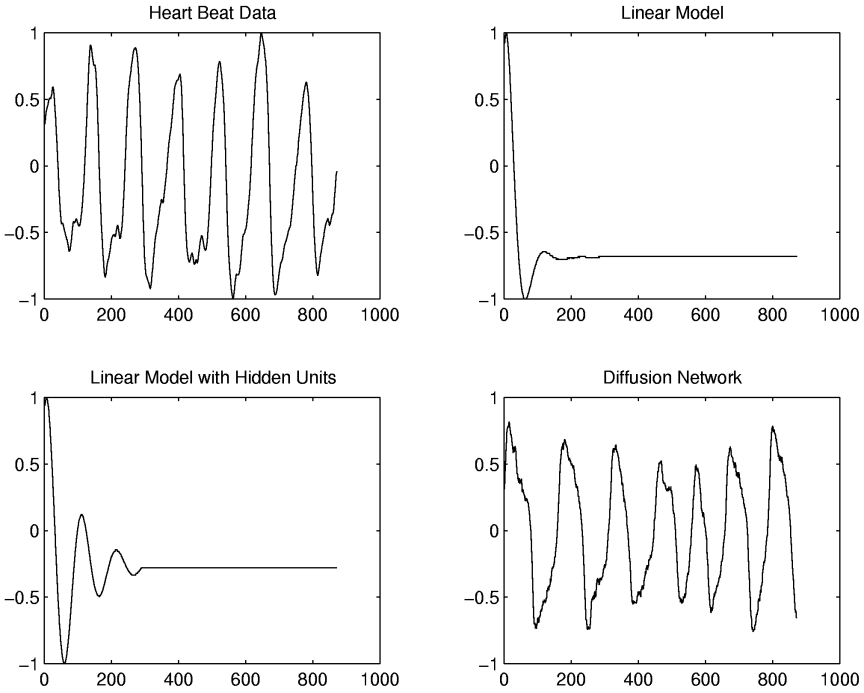

Figure 6: (Top left) The evolution of the first PCA coefficient for a beating heart. (Top right) The path learned by a second-order linear model with no hidden units. (Bottom left) The path learned by a second-order linear model with hidden units. (Bottom right) Typical path learned by a diffusion network.

Table 2: Parameters Learned for the Heart Beat Problem.

$w_{oo} = 1.280$	$w_{oh} = 0.673$	$\xi_o = -0.143$	$\kappa_o = 0.709$
$w_{ho} = -1.218$	$w_{hh} = 0.088$	$\xi_h = 0.001$	$\kappa_h = 1.087$

Notes: All the weights and biases were initialized to zero and the κ terms to 1. The fixed parameters were set to the following values: $\theta_1 = -1, \theta_2 = 2, \theta_3 = 7, \sigma = 0.2$.

hidden Markov models is the nature of the hidden states: diffusion networks use continuous-state representations, while hidden Markov models use discrete-state representations.[15] It is possible that continuous-state representations may be beneficial for modeling some natural sequences.

[15] While HMMs can use continuous observations, the hidden states are always discrete.

Our approach to sequence recognition using diffusion networks is similar to the approach used in the hidden Markov model literature. First, several diffusion networks are independently trained with samples of sequences from each of the categories at hand. For example, if we want to discriminate between c categories of image sequences, we would first train c different diffusion networks. The first network would be trained with examples of category 1, the second network with examples of category 2, and the last network with examples of category c. This training process results in c values of the parameter λ, each of which has been optimized to represent a different category. We represent these values as $\lambda_1^*, \ldots, \lambda_c^*$. Once the networks are trained, we can classify a new observed sequence x_o as follows: we compute $\log \hat{L}_o^{\lambda_i^*}(x_o)$ for $i = 1, \ldots, c$. These log likelihoods are combined with the log-prior probability of each category, and the most probable category of the sequence is chosen.

9.3.1 Training Database. We used Tulips1 (Movellan, 1995), a database consisting of 96 movies of nine male and three female undergraduate students from the Cognitive Science Department at the University of California, San Diego. For each student, two sample utterances were taken for each of the digits "one" through "four" (see Figure 7). The database is challenging due to variability in illumination, gender, ethnicity of the subjects, and position and orientation of the lips. The database is available at J. R. M.'s Web site.

9.3.2 Visual Processing. We used a 2×2 factorial experimental design to explore the performance of two different image processing techniques (contours and contours plus intensity) in combination with two different recognition engines (hidden Markov models and diffusion networks). The image processing was performed by Luettin, Thacker, and Beet (1996a, 1996b). They employ point density models, where each lip contour is represented by a set of points; in this case, both the inner and outer lip contour are represented, corresponding to Luettin's double contour model (see Figure 7). The dimensionality of the representation of the contours was reduced using principal component analysis. For the work presented here, 10 principal components were used to approximate the contour, along with a scale parameter that measured the pixel distance between the mouth corners; associated with each of these 11 components was a corresponding delta component. The value of the delta component for the frame sampled at time t_k equals the value of the original component at that time minus the value of the original component at the previous sampling time t_{k-1} (these delta components are defined to be zero at t_0). In this manner, 22 components were used to represent lip contour information for each still frame. These 22 components were represented using diffusion networks with 22 observation units, one per component.

Figure 7: Example images from the Tulips1 database.

We also tested the performance of a representation that used intensity information in addition to contour shape information. We obtained Luettin et al. (1996b) representations in which gray-level values are sampled along directions perpendicular to the lip contour. These local gray-level values are then concatenated to form a single global intensity vector that is compressed using the first 20 principal components. There were 20 associated delta components, for a total of 40 components of intensity information per still frame. These 40 components were concatenated with the 22 contour components, for a total of 62 components per still frame. These 62 components were represented using diffusion networks with 62 observation units, one per component. Thus, the total number of weight parameters was $(62 + (n - d))^2$, where $n - d$ is the number of hidden units. We tested networks with up to 100 hidden units—26,244 parameters.

9.3.3 Training. We independently trained four diffusion networks to approximate the distributions of lip-contour trajectories of each of the four words to be recognized; the first network was trained with examples of the word *one* and the last network with examples of the word *four*. Each network had the same number of nodes, and the drift of each network was given by equation 1.2 with $\kappa = 1, \rho = 0$, and a hyperbolic tangent activation function—$\theta_1 = -1, \theta_2 = 2, \theta_3 = 1$ in equation 1.3. The connectivity matrix w

and the bias parameters ξ were adaptive. The initial state of the hidden units was set to $(1, \ldots, 1)'$ with probability 1, and σ was set to 1 for all networks.

The diffusion network dynamics were simulated using the forward Euler technique described in previous sections. In our simulations, we set $\Delta t = 1/30$ seconds, the time between video frame samples. Each diffusion network was trained with examples of one of the four digits using the following cost function (see equation 4.2),

$$\Phi(\lambda) = \sum_i \log \hat{L}_o^{\lambda}(x_o^i) - \Psi(\lambda), \tag{9.7}$$

where each x_o^i is a movie from the Tulips1 database—a sample from the desired empirical distribution P_o, and $\Psi(\lambda) = \frac{1}{2}\alpha|\lambda|^2$ for $\alpha > 0$. Several values of α were tried, and performance is reported with the optimal values. Here, $\Psi(\lambda)$ acts as a gaussian prior on the network parameters. Training was done using 20 hidden sample paths per observed path. These paths were sampled using the "teacher forcing" approach described in equations 5.8 through 5.10.

9.3.4 Results. The bank of diffusion networks was evaluated in terms of generalization to new speakers. Since the database is small, generalization performance was estimated using a jackknife procedure (Efron, 1982). The four models (one for each digit) were trained with labeled data from 11 subjects, leaving a subject out for generalization testing. Percentage correct generalization was then tested using the decision rule

$$D(x_o) = \arg \max_{i \in \{1,2,3,4\}} \log \hat{L}_o^{\lambda_i^*}(x_o), \tag{9.8}$$

where $\log \hat{L}_o^{\lambda_i^*}$ is the estimate of the log likelihood of the test sequence evaluated at the optimal parameters λ_i^* found by training on examples of digit i. This rule corresponds to assuming equal priors for each of the four categories under consideration. The entire procedure was repeated 12 times, each time leaving a different subject out for testing, for a total of 96 generalization trials (4 digits $\times$ 12 subjects $\times$ 2 observations per subject). This procedure mimics that used by Luettin et al. (1996a, 1996b; Luettin, 1997) to test hidden Markov model architectures.

We tested performance using a variety of architectures, some including no hidden units and some with up to 100 hidden units. Best generalization performance was obtained using four hidden units. These generalization results are shown in Table 3. The hidden Markov model results are those reported in Luettin et al. (1996a, 1996b) and Luettin (1997). The only difference between their approach and ours is the recognition engine, which is a bank of hidden Markov models in their case and a bank of diffusion networks in our case. The image representations mentioned above were optimized

Table 3: Generalization Performance on the Tulips1 Database.

Approach	Correct Generalization
Best HMM, shape information only	82.3%
Best diffusion network, shape information only	85.4
Untrained human subjects	89.9
Best HMM, shape and intensity information	90.6
Best diffusion network, shape and intensity information	91.7
Trained human subjects	95.5

Notes: Shown in order are the performance of the best-performing HMM from Luettin et al. (1996a, 1996b) and Luettin (1997) which uses only shape information; the best diffusion network obtained using only shape information; the performance of untrained human subjects (Movellan, 1995); the HMM from Luettin's thesis (Luettin, 1997) which uses both shape and intensity information; the best diffusion network obtained using both shape and intensity information; and the performance of trained human lip readers (Movellan, 1995).

by Luettin et al. to work with hidden Markov models. They also tried a variety of hidden Markov model architectures and reported the best results obtained with them.

In all cases, the best diffusion networks outperformed the best hidden Markov models reported in the literature using exactly the same visual preprocessing. The results show that diffusion networks may outperform hidden Markov model approaches in sequence recognition tasks. While these results are very promising, caution should be exercised since the database is relatively small. More work is needed with larger databases.

10 Conclusion

We think the main reason that recurrent neural networks have not worked well in practical applications is that they rely on a very simplistic likelihood model: white noise added to a deterministic sequence. This model cannot cope well with known issues in natural sequences like bifurcations and dynamic time warping. We proposed adding noise to the activation dynamics of recurrent neural networks. The resulting models, which we called diffusion networks, can handle bifurcations and dynamic time warping.

We presented a general framework for learning path probability densities using continuous stochastic models and then applied the approach to train diffusion neural networks. The approach allows training networks with hidden units, nonlinear activation functions, and arbitrary connectivity. Interestingly, the gradient of the likelihood function can be computed using Hebbian coactivation statistics and does not require backpropagation of error signals.

Our work was inspired by the rich literature on continuous stochastic filtering and recurrent neural networks. The idea was to combine the versatil-

ity of recurrent neural networks and the well-known advantages of stochastic modeling approaches. The continuous-time nature of the networks is convenient for data with dropouts or variable sample rates, since the models we use define all the finite dimensional distributions. The continuous-state representation is well suited to problems involving continuous unobservable quantities, as in visual tracking tasks. In particular, the diffusion approach may be advantageous for problems in which continuity and sparseness constraints are useful. Diffusion networks naturally enforce a sparse state transition matrix (there is an infinite number of states, and given any state, there is a very small volume of states to move into with nonnegligible probability). It is well known that enforcing sparseness in state transition matrices is beneficial for many sequence recognition problems (Brand, 1998). Diffusion networks naturally enforce continuity constraints in the observable paths, and thus they may not have the well-known problems encountered when hidden Markov models are used as generative models of sequences (Rabiner & Juang, 1993). We presented simulation results on realistic sequence modeling and sequence recognition tasks with very encouraging results.

While the results were highly encouraging, the databases used were relatively small, and thus the current results should be considered only as exploratory. It should also be noticed that there are situations in which hidden Markov models will be preferable to diffusions. For example, it is relatively easy to compose hierarchies of simple trainable hidden Markov models, capturing acoustic, syntactic, and semantic constraints. At this point, we have not explored how to compose trainable hierarchies of diffusion networks. Another disadvantage of diffusion networks relative to conventional hidden Markov models is training speed, which is significantly slower for diffusion networks than for hidden Markov models. However, once a network was trained, the computation of the density functions needed in recognition was fast and could be done in real time.

Significant work remains to be done. In this article, we assume that the data are a continuous stochastic process. However, in many applications, the data take the form of discrete-time samples from a continuous-time process. In this article, we approached this issue by using simple discrete-time Euler approximations to stochastic integrals. In the future, we plan to investigate alternative approximations that allow for path-wise convergence, such as that in Karandikar (1995). The theoretical properties of the stochastic EM procedure when combined with discrete-time approximations of stochastic integrals remain to be investigated. In this article, we did not consider dispersions that are a function of the state, and we have not optimized the dispersion and the initial distribution of hidden states. Optimization of the initial distribution of hidden states is easy but unnecessarily obscures the presentation of the main issues in the article. Optimization of the dispersion is easy in discrete-time systems but presents mathematical challenges in continuous-time systems; thus, we have deferred the issue for future work. Theoretical issues concerning the approximating power of diffusion

networks need to be explored. In deterministic neural networks, it is known that with certain choices of activation function and sufficiently many hidden units, neural networks can approximate a large set of functions with arbitrary accuracy (Hornik, Stinchombe, & White, 1989). An analogous result for diffusion networks stating the class of distributions that can be approximated arbitrarily closely would be useful. It is also important to compare the properties of the algorithm presented here with alternative learning algorithms for partially observable stochastic processes that could also be applied to diffusion networks (Campillo & Le Gland, 1989; Blake et al., 1999).

Another aspect of theoretical interest is the potential connection between diffusion networks and recent kernel-based approaches to learning. In particular, notice that the learning rule developed in this article depends on a matrix of inner products in the Hilbert space of squared integrable functions, as defined in equation 7.8. This may allow application of standard kernel methods (Aizerman et al., 1964; Burges, 1998) to the problem of learning distributions of sequences.

We are currently exploring applications of diffusion networks to real-time stochastic filtering problems (face tracking) and sequence-generation problems (face animation). Our work shows that diffusion networks may be a feasible alternative to hidden Markov models for problems in which state continuity and sparseness in the state transition distributions are advantageous. The results obtained for the visual speech recognition task are encouraging and reinforce the possibility that diffusion networks may become a versatile general-purpose tool for a wide variety of continuous-signal processing tasks.

Acknowledgments

We thank Anthony Gamst, Ian Fasel, Tim Kalman Marks, and David Groppe for helpful discussions and Juergen Luettin for generously providing lip contour data and technical assistance. The research of R. J. W. was supported in part by NSF grant DMS 9703891. J. R. M. was supported in part by the NSF under grant 0086107.

References

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

Aizerman, M. A., Braverman, E. M., & Rozoner, L. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control, 15*, 821–837.

Billingsley, P. (1995). *Probability and measure*. New York: Wiley.

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

Blake, A., North, B., & Isard, M. (1999). Learning multi-class dynamics. In M. S. Kearns, S. Solla, & D. Cohn (Eds.), *Advances in neural information processing systems*, *11* (pp. 389–395). Cambridge, MA: MIT Press.

Brand, M. (1998). Pattern discovery via entropy minimization. In D. Heckerman & J. Whittaker (Eds.), *Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics*. San Mateo, CA: Morgan Kaufmann.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, *2*(2), 121–167.

Campillo, F., & Le Gland, F. (1989). MLE for partially observed diffusions—direct maximization vs. the EM algorithm. *Stochastic Processes and Their Applications*, *33*(2), 245–274.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Stat. Soc.*, *39*, 1–38.

Efron, A. (1982). *The jacknife, the bootstrap and other resampling plans*. Philadelphia: SIAM.

Fahrmeir, L. (1992). Posterior mode estimation by extended Kalman filter for multivariate dynamics generalized linear models. *Journal of the American Statistical Association*, *87*, 501–509.

Fishman, G. S. (1996). *Monte Carlo sampling: Concepts, algorithms, and applications*. New York: Springer-Verlag.

Ghahramani, Z., & Roweis, S. T. (1999). Learning nonlinear dynamical systems using an EM algorithm. In M. S. Kearns, S. Solla, & D. Cohn (Eds.), *Advances in neural information processing systems*, *11*. Cambridge, MA: MIT Press.

Hertz, J., Krogh, A., & Palmer, R. (1991). *Introduction to the theory of neural computation*. Reading, MA: Addison-Wesley.

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.

Hornik, K., Stinchombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*, 259–366.

Jacob, G., Noble, A., & Blake, A. (1998). Robust contour tracking of electrochardiographic sequences. In *Proc 6th Int. Conf. on Computer Vision* (pp. 408–413).

Kalman, R. E., & Bucy, R. S. (1961). New results in linear filtering and prediction theory. *Transactions ASME J. of Basic Eng.*, *83*, 95–108. Bombay: IEEE Computer Society Press.

Karandikar, R. L. (1995). On pathwise stochastic integration. *Stochastic Processes and Their Applications, 57*, 11–18.

Karatzas, I., & Shreve, S. E. (1991). *Brownian motion and stochastic calculus*. New York: Springer-Verlag.

Kitagawa, G. (1996). Monte Carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, *5*(1), 1–25.

Kloeden, P. E., & Platen, E. (1992). *Numerical solutions to stochastic differential equations*. Berlin: Springer-Verlag.

Levanony, D., Shwartz, A., & Zeitouni, O. (1990). Continuous-time recursive estimation. In E. Arikan (Ed.), *Communication, control, and signal processing*. Amsterdam: Elsevier.

Lewis, F. L. (1986). *Optimal estimation—with an introduction to stochastic control theory*. New York: Wiley.

Ljung, L. (1999). *System identification: Theory for the user*. Upper Saddle River, NJ: Prentice-Hall.

Luettin, J. (1997). *Visual speech and speaker recognition*. Unpublished doctoral dissertation, University of Sheffield.

Luettin, J., Thacker, N., & Beet, S. (1996a). Statistical lip modelling for visual speech recognition. In *Proceedings of the VIII European Signal Processing Conference*. Trieste, Italy.

Luettin, J., Thacker, N. A., & Beet, S. W. (1996b). Speechreading using shape and intensity information. In *Proceedings of the Internal Conference on Spoken Language Processing*. Philadelphia: IEEE.

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, artifical intelligence, and cognitive neuroscience* (pp. 655–688). Cambridge, MA: MIT Press.

Meinhold, R. J., & Singpurwalla, N. D. (1975). Robustification of Kalman filter models. *IEEE Transactions on Automatic Control*, *84*, 479–486.

Mineiro, P., Movellan, J. R., & Williams, R. J. (1998). Learning path distributions using nonequilibrium diffusion networks. In M. Kearns (Ed.), *Advances in neural information processing systems, 10* (pp. 597–599). Cambridge, MA: MIT Press.

Movellan, J. (1994). A local algorithm to learn trajectories with stochastic neural networks. In J. Cowan, G. Tesauro, & J. Alspector (Eds.), *Advances in neural information processing systems, 6* (pp. 83–87). San Mateo, CA: Morgan Kaufmann.

Movellan, J. (1995). Visual speech recognition with stochastic neural networks. In G. Tesauro, D. Touretzky, & T. Leen (Eds.), *Advances in neural information processing systems, 7*. Cambridge, MA: MIT Press.

Movellan, J. R. (1998). A learning theorem for networks at detailed stochastic equilibrium. *Neural Computation*, *10*(5), 1157–1178.

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

Movellan, J. R., & McClelland, J. L. (2001). The Morton-Massaro law of information integration: Implications for models of perception. *Psychological Review, 1*, 113–148.

North, B., & Blake, A. (1998). Learning dynamical models by expectation maximization. In *Proc 6th Int. Conf. on Computer Vision* (pp. 911–916). Bombay: IEEE Computer Society Press.

Oksendal, B. (1998). *Stochastic differential equations: An introduction with applications* (5th ed.). Berlin: Springer-Verlag.

Pearlmutter, B. A. (1995). Gradient calculations for dynamic recurrent neural networks: A survey. *IEEE Transactions on Neural Networks, 6*(5), 1212–1228.

Poor, H. V. (1994). *An introduction to signal detection and estimation*. Berlin: Springer-Verlag.

Protter, P. (1990). *Stochastic integration and differential equations*. Berlin: Springer-Verlag.

Rabiner, L. R., & Juang, B.-H. (1993). *Fundamentals of speech recognition*. Englewood Cliffs, NJ: Prentice Hall.

Sage, A. P., & Melsa, J. L. (1971). *Estimation theory with application to communication and control*. New York: McGraw-Hill.

Shumway, R., & Stoffer, D. (1982). An approach to time series smoothing and forecasting using the EM algorithm. *J. Time Series Analysis*, *3*, 253–265.

Solo, V. (2000). Unobserved Monte-Carlo method for identification of partially observed nonlinear state space systems, Part II: Counting process observations. In *Proc. IEEE Conference on Decision and Control*. Sidney, Australia.

White, H. (1996). *Estimation, inference and specification analysis*. Cambridge: Cambridge University Press.

Zipser, D., Kehoe, B., Littlewort, G., & Fuster, J. (1993). A spiking network model of short-term active memory. *Journal of Neuroscience*, *13*(8), 3406–3420.

Received July 2, 1999; accepted November 15, 2001.