Modeling Path Distributions Using Partially Observable Diffusion Networks: A Monte-Carlo Approach.

Javier R. Movellan
Department of Cognitive Science &
Institute for Neural Computation
University of California San Diego
Paul Mineiro
Department of Cognitive Science
University of California San Diego
R. J. Williams
Department of Mathematics &
Institute for Neural Computation
University of California San Diego

Technical Report 99.01

Machine Perception Laboratory
University of California, San Diego
La Jolla, CA 92093-0515

June 30, 1999

Modeling Path Distributions Using Partially Observable Diffusion Networks: A Monte-Carlo Approach.

Javier R. Movellan
Department of Cognitive Science &
Institute for Neural Computation
University of California San Diego
Paul Mineiro
Department of Cognitive Science
University of California San Diego
R. J. Williams
Department of Mathematics &
Institute for Neural Computation
University of California San Diego

Abstract

Hidden Markov models have been more successful than recurrent neural networks for problems involving temporal sequences, e.g., speech recognition. One possible reason for this is that recurrent neural networks are being used in ways that do not handle temporal uncertainty well. In this paper we present a framework for learning, recognition and stochastic filtering of temporal sequences based on a probabilistic version of continuous recurrent neural networks. We call these networks diffusion (neural) networks for they are based on stochastic diffusion processes defined by adding Brownian motion to the standard recurrent neural network dynamics. The goal is to combine the versatility of recurrent neural networks with the power of probabilistic techniques. We focus on the problem of learning to approximate a desired probability distribution of sequences. Once a distribution of sequences has been learned, well known techniques can be applied for the generation, recognition and stochastic filtering of new sequences. We present an adaptive importance sampling scheme for estimation of log–likelihood gradients. This allows the use of iterative optimization techniques, like gradient descent and the EM algorithm, to train diffusion networks. We present results for an automatic visual speech recognition task in which diffusion networks provide excellent performance when compared to hidden Markov models.

1 Introduction

Many natural signals, like pixel gray-levels, line orientations, object position, velocity and shape parameters, are modeled well as continuous functions of time. Since the solutions to many decision theoretic problems of interest in machine perception, e.g., signal detection, classification, stochastic filtering, and estimation, are naturally formulated using probability distributions, it is desirable to have a flexible framework for approximating distributions of continuous signals. Such a framework could prove as useful for problems involving continuous–state signals as conventional hidden Markov models have proven for problems involving discrete state sequences[1]. Of particular importance is the ability to represent multimodal path distributions. Systems that cannot easily represent such path distributions (e.g., deterministic neural networks, Kalman-Bucy filters) typically learn to average paths. In practice, the inability to learn multimodal distributions leads to poor performance on realistic machine perception tasks, such as visual tracking in cluttered environments (Isard & Blake, 1998).

The approach we explore in this paper involves the use of diffusion (neural) networks, an extension of continuous–time, continuous–state, recurrent neural networks in which the dynamics are probabilistic (Movellan & McClelland, 1993; Movellan, 1994; Mineiro, Movellan, & Williams, 1998; Movellan, 1998). While recurrent neural networks are defined via ordinary differential equations (ODE), diffusion networks are described by stochastic differential equations (SDEs). Stochastic differential equations provide a rich language for expressing probabilistic temporal dynamics and have proven useful in formulating continuous–time inference problems, as for example in the continuous Kalman-Bucy filter (Oksendal, 1991).

We can think of diffusion networks as generators of continuous random paths whose parameters can be tuned to match a desired distribution. In this paper we propose Monte-Carlo sampling techniques to obtain estimates of the gradient of the log–likelihood with respect to the network parameters. This allows the use of iterative optimization techniques, like gradient descent and the EM algorithm to train networks to approximate desired path probability densities. Once a network has been trained, standard techniques can be used for tasks such as sequence recognition, stochastic filtering, prediction, etc. We illustrate the approach on a visual speech recognition task in which diffusion networks performed very well when compared with hidden Markov models.

The paper proceeds as follows. First, we define diffusion networks, state the probability density induced by diffusion networks, and derive the gradient of this density with respect to the network parameters. Second we introduce a change of measure technique for efficient Monte–Carlo estimation of the likelihood and log–likelihood gradient of observed paths. Following this we present an application to visual speech recognition in which diffusion networks outperformed standard hidden Markov models. Discussion of the implications of this work and future directions for research follow. The paper relies on aspects of measure theory and probability theory of continuous–time processes that may be unfamiliar to some readers. For a concise review of these concepts the reader is referred to Chapter 2 of Kloeden and Platen (1992).

[1] While standard hidden Markov models can handle continuous observations, they work with discrete internal states and do not enforce temporal continuity on the observations.

1 Diffusion networks

Diffusion networks are continuous–time, continuous–state recurrent neural networks with a probabilistic component. A network consists of a set of n units coupled via synaptic connections. The strength of these connections is represented by a real valued vector λ which is tuned to optimize some aspect of the network's behavior. For each value of λ the network produces a unique probability distribution of continuous paths and thus we can treat diffusion networks as a convenient way of representing a family of probability distributions of paths.

The temporal evolution of the activation of the n units in a network is represented as a stochastic process X^λ that satisfies the following Itô stochastic differential equation (SDE):

$$dX^\lambda(t) = \mu(t, X^\lambda(t), \lambda)dt + \sigma\, dB(t),\ t \in [0, T], \tag{1}$$

$$X^\lambda(0) \sim \nu. \tag{2}$$

Here $\mu : [0, T] \times \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$ is a function called the *drift* which is the deterministic aspect of the dynamics analogous to that in recurrent neural networks; B is a standard n-dimensional *Brownian motion* (cf. Protter, 1990) 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; ν is a fixed probability measure on $\mathbb{R}^n$ with finite moments of all orders that describes the initial distribution of X^λ; and $\lambda \in \mathbb{R}^p$ is a vector of synaptic weights. Mild assumptions on μ are sufficient for the existence and uniqueness in law of solutions to (1)–(2): for each $\lambda \in \mathbb{R}^p$, $\mu(\cdot, \cdot, \lambda)$ is continuous and satisfies a linear growth condition

$$|\mu(t, u, \lambda)| \leq K_\lambda(1 + |u|), \tag{3}$$

for some $K_\lambda > 0$ and all $t \in [0, T]$, $u \in \mathbb{R}^n$, where $|\cdot|$ denotes the Euclidean norm (see e.g., Prop. 5.3.6 in Karatzas & Shreve, 1991). These assumptions are met by the drift commonly used in the recurrent neural network literature, which is of the form:

$$\mu_i(t, u, \lambda) = \frac{1}{\kappa_i}\left(\xi_i(t) - \frac{1}{\rho_i}u_i + \sum_{j=1}^{n} w_{ij}\,\varphi(u_j)\right),\ \text{for } u \in \mathbb{R}^n,\ j = 1, \cdots, n, \tag{4}$$

$$\varphi(v) = \theta_1 + \theta_2\,\frac{1}{1 + e^{-v}},\ \text{for } v \in \mathbb{R}, \tag{5}$$

where μ_i is the i^{th} component of μ, i.e., the drift for the i^{th} unit in the network. Here each unit is interpreted as a point neuron model, X_i^λ is the soma potential (presynaptic activation) of unit i, the function φ, which is translated and scaled by the parameters $\theta = (\theta_1, \theta_2) \in \mathbb{R}^2$, represents the input-output characteristic amplification (see p. 3089, Hopfield, 1984). The variable $\varphi(X_j^\lambda)$ is the post-synaptic activation of unit j, $\rho_j > 0$ is the transmembrane resistance, $\kappa_j > 0$ is the input capacitance, ξ_j is a bias current, of the form $\xi_i(t) = \beta_i h_i(t)$ where $h_i : [0, T] \to \mathbb{R}$ is a given continuous function of time representing an external current injected into the neuron and $\beta_i \in \mathbb{R}$ is a parameter controlling the gain of the current. The term $w_{ij} \in \mathbb{R}$ is the conductance between the output $\varphi(X_j^\lambda)$ and the body of cell i. In this implementation of diffusion networks the adaptive vector λ may contain

the $1/\kappa_i, 1/\rho_i, \beta_i, \theta$ and w_{ij} terms as its components[2]. If one sets $\sigma = 0$ in (1), then the SDE becomes an ordinary differential equation (ODE) representing a resistance-capacitance charging equation (see p. 3089, Hopfield, 1984). Thus, we can think of diffusion networks as models of low-power non-linear electronic circuits which have a non-negligible thermal noise component.

2 Learning continuous path probability distributions

In this paper we focus on how to train diffusion networks to learn probability distributions of continuous signals. The signals we wish to learn are modeled as the first d components of the n-dimensional ($n \geq d$) process X^λ. We regard the first d components 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*. Hidden components are included to allow inference about unobservable quantities and for modeling non-Markovian temporal dependencies in the observable components.

Given a sample of continuous paths (e.g., a collection of speech waves) our goal is to select a value of the parameter λ that maximizes the likelihood of those "training" paths. To do so, we need to consider the family (indexed by λ) of probability distributions associated with the observable components O^λ of X^λ, for $\lambda \in \mathbb{R}^p$. To precisely describe these distributions, which live on a space of continuous paths, we introduce the following notation.

Let Ω denote the space of continuous functions $x : [0, T] \to \mathbb{R}^n$, endowed with the topology of uniform convergence on $[0, T]$. Any solution $X^\lambda = \{X^\lambda(t) : t \in [0, T]\}$ of (1)–(2) takes values in Ω. Let $\mathcal{F}$ denote the sigma algebra of subsets of Ω generated by the open sets in Ω. Similarly, define Ω_h and Ω_o with associated sigma algebras $\mathcal{F}_h$ and $\mathcal{F}_o$, by replacing n by $n-d$ and d, respectively, in the above definitions of Ω and $\mathcal{F}$. The hidden and observable components H^λ and O^λ of any solution $X^\lambda = (O^\lambda, X^\lambda)$ of (1)–(2) 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 $x_h \in \Omega_h$.

We regard $T > 0$ and ν as fixed henceforth. In this paper we concentrate on the probability distributions of continuous paths induced by diffusion networks and thus we just need to consider weak (also known as distributional or statistical)[3] solutions to (1)–(2). Any (weak) solution X^λ induces a unique probability distribution Q^λ on the measurable space $(\Omega, \mathcal{F})$. For each A in $\mathcal{F}$, the expression $Q^\lambda(A)$ represents the probability that a network with parameter vector λ generates paths in 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 $Q_o^\lambda = Q^\lambda$. On the other hand, if there are hidden units, $d < n$ and we must marginalize Q^λ over the unobservable components:

$$Q_o^\lambda(A_o) = Q^\lambda(A_o \times \Omega_h) \quad \text{for all } A_o \in \mathcal{F}_o. \tag{6}$$

[2]We allow $1/\kappa_i$ and $1/\rho_i$ to take the value 0.

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

For later use, we define the probability law of the hidden components:

$$Q_h^\lambda(A_h) = Q^\lambda(\Omega_o \times A_h) \quad \text{for all } A_h \in \mathcal{F}_h. \tag{7}$$

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

2.1 Density of observable state 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 apply the familiar techniques of maximum likelihood and Bayesian estimation we need a reference measure R_o with respect to which a probability density $L_o^\lambda = dQ_o^\lambda/dR_o$ exists for each element of $\mathcal{Q}_o$ (p. 317ff of Poor, 1994). In discrete time systems, like HMMs, the Lebesgue measure[4] is used as the standard reference. For continuous–time systems a different reference measure is needed. In this subsection we introduce such a reference measure and the corresponding densities.

Let $\{R^u : u \in \mathbb{R}^n\}$ be the indexed set of probability measures on $(\Omega, \mathcal{F})$ such that for each u the process $W = \{W(t,\omega) = \omega(t)/\sigma : t \in [0,T], \omega \in \Omega\}$ is a standard n-dimensional Brownian motion under R^u with respect to its natural filtration and $R^u(W(0) = u) = 1$. Let R be the probability measure defined as follows

$$R(A) = \int_{\mathbb{R}^n} R^{u/\sigma}(A)\, \nu(du), \text{ for all } A \in \mathcal{F}. \tag{8}$$

The measure R can serve as a reference measure with respect to which probability densities can be defined for each element of $\mathcal{Q}$. Using Girsanov's theorem[5] (p. 303 Karatzas & Shreve, 1991) we have that for all A in $\mathcal{F}$

$$Q^\lambda(A) = \int_A L^\lambda(x)\, dR(x), \tag{9}$$

where[6]

$$L^\lambda(x) = \frac{dQ^\lambda}{dR}(x) = \exp\left\{\frac{1}{\sigma^2}\int_0^T \mu(t, x(t), \lambda) \cdot dx(t) - \frac{1}{2\sigma^2}\int_0^T |\mu(t, x(t), \lambda)|^2 dt\right\}, \tag{10}$$

for $x \in \Omega$. The first integral in (10) is an Itô stochastic integral, which can be chosen to be $\mathcal{F}$–measurable, the second is a standard Riemann integral. The term L^λ is a Radon-Nikodym derivative that represents the probability density of Q^λ with respect to R. For a fixed path $x \in \Omega$ the term $L^\lambda(x)$ can be treated as a likelihood function 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

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

[5] The conditions on μ mentioned above are sufficient conditions for Girsanov's theorem to hold.

[6] Throughout the paper we interchangeably use the notation $\int_A X(x)P(dx)$ and $\int_A X(x)\, dP(x)$ to signify the Lebesgue integral over the set A of a random variable X with respect to a measure P.

condition that the initial probability measure ν is a product measure $\nu = \nu_o \otimes \nu_h$, where ν_o, ν_h are probability measures on $\mathbb{R}^d$ and $\mathbb{R}^{n-d}$, respectively. It then follows from the independence of the components of Brownian motion that

$$R(A_h \times A_o) = R_h(A_h) R_o(A_o), \tag{11}$$

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

$$R_h(A_h) = R(\Omega_o \times A_h)\ , \tag{12}$$
$$R_o(A_o) = R(A_o \times \Omega_h). \tag{13}$$

To find an appropriate density for Q_o^λ, note that for $A_o \in \mathcal{F}_o$, by (9)

$$Q_o^\lambda(A_o) = Q^\lambda(A_o \times \Omega_h) \tag{14}$$
$$= \int_{A_o} \int_{\Omega_h} L^\lambda(x_o, x_h) dR_h(x_h)\ dR_o(x_o), \tag{15}$$

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

$$L_o^\lambda(x_o) = \frac{dQ_o^\lambda}{dR_o}(x_o) = \int_{\Omega_h} L^\lambda(x_o, x_h)\ dR_h(x_h), \text{ for } x_o \in \Omega_o. \tag{16}$$

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

$$L_h^\lambda(x_h) = \frac{dQ_h^\lambda}{dR_h}(x_h) = \int_{\Omega_o} L^\lambda(x_o, x_h)\ dR_o(x_o), \text{ for } x_h \in \Omega_h. \tag{17}$$

2.2 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. This modeling problem can be approached by defining a Kullback-Leibler distance (White, 1996) between P_o and Q_o^λ:

$$E^{P_o}\left(\log \frac{\Lambda_o}{L_o^\lambda}\right), \tag{18}$$

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[7]. We seek values of λ that make (18) small. In practice we

[7] We are assuming that the expectation in (18) is well defined and finite and in particular Λ_o exists, which is also an implicit assumption in finite dimensional density estimation approaches.

estimate such values by obtaining $\tilde{n}$ paths $\{x_o^i\}_{i=1}^{\tilde{n}}$ sampled in an i.i.d. manner[8] from P_o and seek values of λ that maximize the following function:

$$\Phi(\lambda) = \frac{1}{\tilde{n}} \left(\sum_{i=1}^{\tilde{n}} \log L_o^\lambda(x_o^i) \right) - \Psi(\lambda). \tag{19}$$

This function is a combination of the empirical log–likelihood and a regularizer $\Psi : \mathbb{R}^p \to \mathbb{R}$ which encodes prior knowledge about desirable values of λ (p. 338 of 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 main idea.

The random variable $\nabla_\lambda \log L_o^\lambda$, the gradient of the log density of the observable paths with respect to λ, is of interest to apply optimization techniques such as gradient descent and the EM algorithm. If there are no hidden units, equation (10) gives the density of the observable measure, and differentiation yields[9]

$$\nabla_\lambda \log L^\lambda(x) = \frac{1}{\sigma^2} \int_0^T J(t, x(t), \lambda) \cdot dI^\lambda(t, x), \text{ for } x \in \Omega, \tag{20}$$

where J is an $n \times n$ matrix, the transpose of the Jacobian matrix of the drift with respect to λ:

$$J_{ij}(t, x(t), \lambda) = \frac{\partial \mu_j(t, x(t), \lambda)}{\partial \lambda_i}, \tag{21}$$

and $\sigma^{-1} I^\lambda$ is the (joint) *innovation* process (p. 319ffm of Poor, 1994):

$$I^\lambda(t, x) = x(t) - x(0) - \int_0^t \mu(s, x(s), \lambda)\, ds. \tag{22}$$

Such a process is a Brownian motion under Q^λ. If there are hidden units, the density of an observable path is given by equation (16), and differentiation yields

$$\nabla_\lambda \log L_o^\lambda(x_o) = \int_{\Omega_h} L_{h|o}^\lambda(x_h \mid x_o) \nabla_\lambda \log L^\lambda(x_o, x_h)\, dR_h(x_h), \tag{23}$$

where

$$L_{h|o}^\lambda(x_h \mid x_o) = \frac{L^\lambda(x_o, x_h)}{L_o^\lambda(x_o)}. \tag{24}$$

[8] Independent and identically distributed.

[9] Conditions that are sufficient to justify the differentiation leading to equations (20) and (23) are that the first and second partial derivatives of $\mu(t, x, \lambda)$ with respect to λ exist and together with μ are continuous in (t, x, λ) and satisfy a linear growth condition of a similar form to (3) 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 (4).

2.3 Importance sampling

If there are no hidden units, the likelihood and log–likelihood gradient of paths can be obtained directly via (10) and (20). If there are hidden units we need to take expected values over hidden paths with respect to the reference measure R_h. One approach to computing these expected values requires solving stochastic partial differential equations (Campillo & Le Gland, 1989). In this paper we propose a different approach based on the use of Monte–Carlo sampling techniques. The idea is to estimate expected values using averages over samples. In an effort to improve the efficiency of these estimates we use an importance sampling scheme in which instead of sampling from R_h we sample from another sampling distribution S_h and multiply the variables being averaged by a correction factor known as the importance function (Fishman, 1996). Note

$$L_o(x_o) = \int_{\Omega_h} L^\lambda(x_o, x_h)\, \frac{dR_h}{dS_h}(x_h)\, dS_h(x_h), \tag{25}$$

$$\nabla_\lambda \log L_o^\lambda(x_o) = \int_{\Omega_h} L_{h|o}^\lambda(x_h \mid x_o)\, \frac{dR_h}{dS_h}(x_h)\, \nabla_\lambda \log L^\lambda(x_o, x_h)\, dS_h(x_h), \tag{26}$$

where S_h is a distribution on $(\Omega_h, \mathcal{F}_h)$ for which the density dR_h/dS_h exists. We can obtain unbiased estimates of these expected values by averaging over a set of hidden paths $\mathcal{H} = \{h^l\}_{l=1}^m$ sampled in an i.i.d. manner from S_h:

$$\hat{L}_o^\lambda(x_o) = \sum_{l=1}^m p^\lambda(x_o, h^l), \tag{27}$$

$$\nabla_\lambda \log \hat{L}_o^\lambda(x_o) = \frac{\nabla_\lambda \hat{L}_o^\lambda(x_o)}{\hat{L}_o^\lambda(x_o)} = \sum_{l=1}^m p_{h|o}(h^l \mid x_o) \nabla_\lambda \log L^\lambda(x_o, h^l), \tag{28}$$

where the functions $p^\lambda : \Omega_o \times \Omega_h \to \mathbb{R}$ and $p_{h|o}^\lambda : \Omega_h \to \mathbb{R}$ are defined as follows[10]:

$$p^\lambda(x_o, h) = \begin{cases} \frac{1}{m}\, L^\lambda(x_o, h) \frac{dR_h}{dS_h}(h), & \text{for } h \in \mathcal{H}, \\ 0 & \text{else}, \end{cases} \tag{29}$$

$$p_{h|o}^\lambda(h \mid x_o) = \frac{p^\lambda(x_o, h)}{p_o^\lambda(x_o)}, \text{ for } h \in \Omega_h, \tag{30}$$

and

$$p_o^\lambda(x_o) = \hat{L}_o^\lambda(x_o) = \sum_{l=1}^m p^\lambda(x_o, h^l). \tag{31}$$

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

[10] For simplicity in our notation we suppressed the dependency of the p^λ functions on the sampling distribution S_h and the sample $\mathcal{H}$.

with respect to that probability mass function. The probability density dR_h/dS_h is known in the Monte–Carlo sampling literature as an *importance function* (p. 257, Fishman, 1996). Required properties of sampling distributions (S_h) are: 1) we must know how to obtain i.i.d. samples from S_h; and 2) we must know how to compute dR_h/dS_h. Desirable properties of S_h are: 1) it helps avoid numerical overflow and underflow problems when computing the p^λ functions using digital computers; 2) it speeds up the computation of p^λ; and 3) it reduces the variance of the Monte–Carlo estimates (27) and (28).

We obtained good results by using a sampling distribution S_h^{λ,x_o} which changed with the parameter λ and the observed path x_o. In particular we let S_h^{λ,x_o} be the probability measure induced by a diffusion network with parameter λ and which has been forced to exhibit the path x_o over the observable units. One interesting property of this approach is that the sampling distribution S_h^{λ,x_o} is adaptive, i.e., it changes as learning progresses, since it depends on λ. This approach is reminiscent of the technique of teacher forcing from deterministic neural networks (Hertz, Krogh, & Palmer, 1991). To find the appropriate importance function consider the following SDE parameterized by λ and $x_o \in \Omega_o$,

$$\begin{aligned} dH(t) &= \mu_h(t, x_o(t), H(t), \lambda)dt + \sigma\ dB_h(t), \\ H(0) &\sim \nu_h, \end{aligned} \tag{32}$$

where μ_h is the hidden component of the drift. Equation (32) represents the hidden node dynamics of a diffusion network with the observable nodes forced to follow the path x_o. Using Girsanov's theorem the law S_h^{λ,x_o} of H satisfies

$$\begin{aligned} \frac{dR_h}{dS_h^{\lambda,x_o}}(x_h) = \exp\Bigg\{ -\Bigg(& \frac{1}{\sigma^2}\int_0^T \mu_h(t, x_o(t), x_h(t), \lambda)\cdot dx_h(t) \\ & -\frac{1}{2\sigma^2}\int_0^T |\mu_h(t, x_o(t), x_h(t), \lambda)|^2 dt\Bigg)\Bigg\}, \end{aligned} \tag{33}$$

which is the importance function. For this sampling distribution the p^λ function defined in (29) is as follows:

$$\begin{aligned} p^\lambda(x_o, h) = & \frac{1}{m}\, L^\lambda(x_o, h)\frac{dR_h}{dS_h^{\lambda,x_o}}(h) = \frac{1}{m}\exp\Bigg\{ \frac{1}{\sigma^2}\int_0^T \mu_o(t, x_o(t), h(t), \lambda)\cdot dx_o(t) \\ & -\frac{1}{2\sigma^2}\int_0^T |\mu_o(t, x_o(t), h(t), \lambda)|^2 dt\Bigg\}, \text{ for } h \in \mathcal{H}. \end{aligned} \tag{34}$$

We can get i.i.d. sample paths $\{h^l\}_{l=1}^m$ from S_h^{λ,x_o} by numerically simulating (32) with the observable units forced to exhibit the path x_o.

2.4 Monte–Carlo learning

Let $\mathcal{H} = \{h^l\}_{l=1}^m$ be a fixed set of hidden paths i.i.d. sampled from a fixed distribution S_h. The goal of learning is to find values of λ which maximize the Monte–Carlo estimate $\log \hat{L}^\lambda(x_o)$. We can search for such values using standard iterative procedures, like gradient ascent or conjugate gradient. Another iterative approach of interest is the EM (expectation

maximization) 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[11]

$$M(\bar{\lambda}, \lambda, x_o) = \sum_{l=1}^{m} p^{\bar{\lambda}}_{h|o}(h^l \mid x_o) \log p^{\lambda}(x_o, h^l). \tag{35}$$

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

$$\log \hat{L}^{\lambda}_o(x_o) = \log p^{\lambda}_o(x_o) = \sum_{l=1}^{m} p^{\bar{\lambda}}_{h|o}(h^l \mid x_o) \log \frac{p^{\lambda}(x_o, h^l)}{p^{\lambda}_{h|o}(h^l \mid x_o)} \tag{36}$$

$$= M(\bar{\lambda}, \lambda, x_o) - \sum_{l=1}^{m} p^{\bar{\lambda}}_{h|o}(h^l \mid x_o) \log p^{\lambda}_{h|o}(h^l \mid x_o), \tag{37}$$

and

$$\log \hat{L}^{\lambda}_o(x_o) - \log \hat{L}^{\bar{\lambda}}_o(x_o) = M(\bar{\lambda}, \lambda, x_o) - M(\bar{\lambda}, \bar{\lambda}, x_o) + \mathrm{KL}(p^{\bar{\lambda}}_{h|o}(\cdot \mid x_o),\ p^{\lambda}_{h|o}(\cdot \mid x_o)), \tag{38}$$

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 non-negative, it follows that if $M(\bar{\lambda}, \lambda, x_o) > M(\bar{\lambda}, \bar{\lambda}, x_o)$ then $\log \hat{L}^{\lambda}_o(x_o) > \log \hat{L}^{\bar{\lambda}}_o(x_o)$. Once we find a value of λ that maximizes $M(\bar{\lambda}, \lambda, x_o)$ we let $\bar{\lambda}$ take that value and repeat the cycle searching for a value of λ that maximizes the new M function. These cycles are repeated until convergence is achieved. At that point we may change to a new sampling distribution, better suited for the values of $\bar{\lambda}$ we converged to, get a new sample of hidden paths, and re-apply the EM procedure.

2.5 The neural network case

Let $w^{\lambda} \in \mathbb{R}^n \otimes \mathbb{R}^n$ be the matrix of connections in a diffusion neural network controlled by the parameter λ. Let the components of w^{λ} take the first $n \times n$ components of λ, i.e., $\lambda_{(r-1)n+s} = w^{\lambda}_{rs}$ for $r, s = 1, \cdots, n$. Here w^{λ}_{rs} represents the weight (conductance) of the connection from (sending) unit s into (receiving) unit r in a network with parameter λ. To simplify the presentation, in this section we fix r and s to two arbitrary values in $\{1, \cdots, n\}$ and let $\lambda_i = w^{\lambda}_{rs}$, i.e., $i = (r-1)n + s$. For the neural drift (4), (21) becomes

$$J_{ij}(t,x(t), \lambda) = \frac{\partial \mu_j(t, x(t), \lambda)}{\partial \lambda_i} = \frac{\partial \mu_j(t, x(t), \lambda)}{\partial w^{\lambda}_{rs}} \tag{39}$$

$$= \frac{\partial}{\partial w^{\lambda}_{rs}} \frac{1}{\kappa_j} \sum_{m=1}^{n} w^{\lambda}_{jm}\, \varphi(x_m(t)) = \delta_{jr} \frac{1}{\kappa_r}\, \varphi(x_s(t)), \text{ for } x \in \Omega, \tag{40}$$

[11] If a regularizer is used, redefine $p^{\lambda}(x_o, h^l)$ in (29) as $L^{\lambda}(x_o, h^l) dR_h / dS_h(h^l) \exp(-\Psi(\lambda))$.

where δ is the Kronecker delta function and x_m, stands for the m^{th} component of x, i.e., the activation of unit m at time t in path x. Combining (20) and (40) we get

$$\frac{\partial \log L^\lambda(x)}{\partial \lambda_i} = \frac{\partial \log L^\lambda(x)}{\partial w^\lambda_{rs}} = \frac{1}{\sigma^2}\sum_{j=1}^{n}\int_0^T J_{ij}(t,x(t),\lambda)\, dI^\lambda_j(t,x) \tag{41}$$

$$= \frac{1}{\sigma^2}\sum_{j=1}^{n}\int_0^T \delta_{jr}\,\frac{1}{\kappa_r}\,\varphi(x_s(t))\, dI^\lambda_j(t,x) \tag{42}$$

$$= \frac{1}{\sigma^2\ \kappa_r}\int_0^T \varphi(x_s(t))\, dI^\lambda_r(t,x), \tag{43}$$

where $\sigma^{-1}I^\lambda_r$ is the innovation process of the receiving unit, i.e.,

$$dI^\lambda_r(t,x) = dx_r(t) - \mu_r(t,x(t),\lambda)\ dt = dx_r(t) + \frac{1}{\kappa_r\ \rho_r}\, x_r(t)\, dt$$
$$- \frac{\xi_r(t)}{\kappa_r}\, dt - \frac{1}{\kappa_r}\sum_{m=1}^{n} w^\lambda_{rm}\,\varphi(x_m(t))\, dt. \tag{44}$$

Combining (43) and (44) we get

$$\frac{\partial \log L^\lambda(x)}{\partial w^\lambda_{rs}} = b_{rs}(x) - \frac{1}{\kappa_r^2}\sum_{m=1}^{n} w^\lambda_{rm}\, a_{sm}(x), \tag{45}$$

where

$$b_{rs}(x) = \frac{1}{\sigma^2\ \kappa_r}\left(\int_0^T \varphi(x_s(t))dx_r(t) + \frac{1}{\kappa_r\ \rho_r}\ \int_0^T \varphi(x_s(t))\, x_r(t)\, dt\right.$$
$$\left. - \frac{1}{\kappa_r}\int_0^T \varphi(x_s(t))\,\xi_r(t)\, dt\right), \tag{46}$$

$$a_{sm}(x) = \frac{1}{\sigma^2}\int_0^T \varphi(x_s(t))\,\varphi(x_m(t))\, dt. \tag{47}$$

Consider now the vector $w^\lambda_{r\cdot}$ of synaptic weights converging into unit r, i.e., $w^\lambda_{r\cdot} = (w^\lambda_{r,1}, \cdots, w^\lambda_{r,n})'$. The gradient of $\log L^\lambda(x)$ with respect to these weights can be expressed as follows

$$\nabla_{w^\lambda_{r\cdot}} \log L^\lambda(x) = \begin{pmatrix} b_{r1}(x) \\ b_{r2}(x) \\ \vdots \\ b_{rn}(x) \end{pmatrix} - \frac{1}{\kappa_r^2}\begin{pmatrix} a_{11}(x) & \cdots & a_{1n}(x) \\ a_{21}(x) & \cdots & a_{2n}(x) \\ \vdots & \ddots & \vdots \\ a_{n1}(x) & \cdots & a_{nn}(x) \end{pmatrix}\begin{pmatrix} w^\lambda_{r1} \\ w^\lambda_{r2} \\ \vdots \\ w^\lambda_{rn} \end{pmatrix}. \tag{48}$$

It is easy to see that $-a/\kappa_r^2$ is the Hessian matrix of $\log L^\lambda$ with respect to $w^\lambda_{r\cdot}$. Now let $v \in \mathbb{R}^n$ and let $y(t) = \sum_{j=1}^n v_j \varphi(x_j(t))$ for all $t \in [0,T]$. Note that

$$\int_0^T y(t)^2\, dt = \sigma^2\ v'a\ v \geq 0, \tag{49}$$

showing that a is positive-semi definite. The integral will be larger than zero if the postsynaptic activations $(\varphi(x_1), \cdots, \varphi(x_n))$ are not a linear combination of each other. In such case values of $w^{\lambda}_{r\cdot}$ that make the gradient vanish maximize the log-likelihood function. Finding such values involves solving a system of n equations with n variables. Note these equations are linear in $w^{\lambda}_{r\cdot}$ even though the activation function φ may be non-linear. Since the a matrix does not depend on r we only need to invert a single $n \times n$ matrix to find the optimal values of $n \times n$ weight parameters.

Figure 1 shows results of a computer simulation in which a 2 unit network was trained to oscillate. We tried an oscillation pattern because of its relevance for the application we explore in a later section, which involves recognizing sequences of lip movements. The training sequence was a pair of out–of–phase sinusoids. The 2×2 matrix of synaptic weights was found by setting the gradient in (48) to zero and solving for $w_{r\cdot}$ analytically. The figure shows the "training" path and a couple of sample paths, one obtained with the σ parameter set to 0, and one with the parameter set to 0.5.

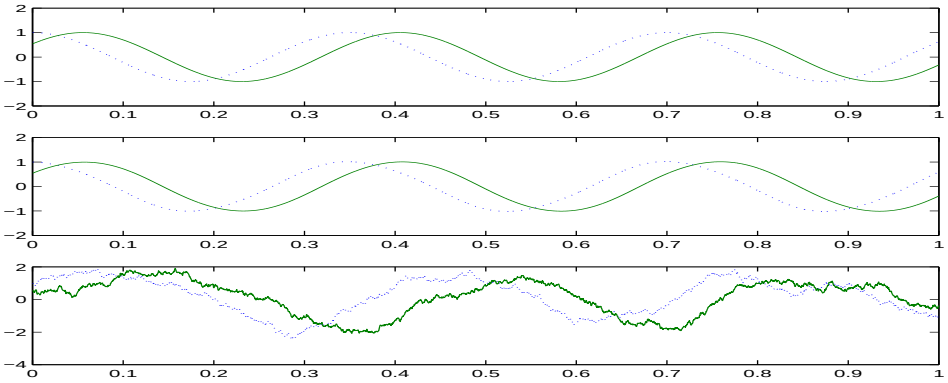

Figure 1: Training a 2 unit network to maximize the likelihood of a sinusoidal path. The top graph shows the training path. It consists of two sinusoids out of phase each representing the activation of the two units in the network. The center graph shows a sample path obtained after training the network and setting $\sigma = 0$, i.e., no noise. The bottom graph shows a sample path obtained with $\sigma = 0.5$.

If there are hidden units, we can use the sampling approach presented in the previous section. Given an observable path, $x_o \in \Omega_o$ we obtain m hidden paths $\{h^l\}_{l=1}^m$ i.i.d. sampled from S_h. The gradient of the log–likelihood estimate is as follows:

$$\nabla_{w^{\lambda}_{r\cdot}} \hat{L}^{\lambda}_o(x_o) = \begin{pmatrix} \tilde{b}^{\lambda}_{r1}(x_o) \\ \tilde{b}^{\lambda}_{r2}(x_o) \\ \vdots \\ \tilde{b}^{\lambda}_{rn}(x_o) \end{pmatrix} - \frac{1}{\kappa_r^2} \begin{pmatrix} \tilde{a}^{\lambda}_{11}(x_o) & \cdots & \tilde{a}^{\lambda}_{1n}(x_o) \\ \tilde{a}^{\lambda}_{21}(x_o) & \cdots & \tilde{a}^{\lambda}_{2n}(x_o) \\ \vdots & \ddots & \vdots \\ \tilde{a}^{\lambda}_{n1}(x_o) & \cdots & \tilde{a}^{\lambda}_{nn}(x_o) \end{pmatrix} \begin{pmatrix} w^{\lambda}_{r1} \\ w^{\lambda}_{r2} \\ \vdots \\ w^{\lambda}_{rn} \end{pmatrix}, \tag{50}$$

where

$$\tilde{b}_{ij}^{\lambda}(x_o) = \sum_{l=1}^{m} p_{h|o}^{\lambda}(h^l \mid x_o)\, b_{ij}(x_o, h^l), \tag{51}$$

$$\tilde{a}_{ij}^{\lambda}(x_o) = \sum_{l=1}^{m} p_{h|o}^{\lambda}(h^l \mid x_o)\, a_{ij}(x_o, h^l). \tag{52}$$

Note in this case the $\tilde{a}$ and $\tilde{b}$ coefficients depend on λ in a non-linear manner, even if the activation function φ is linear. We can find values of λ 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. The gradient of $\hat{M}(\bar{\lambda}, \lambda, x_o)$ with respect to $w_{r\cdot}^{\lambda}$ follows:

$$\nabla_{w_{r\cdot}^{\lambda}} \hat{M}(\bar{\lambda}, \lambda, x_o) = \begin{pmatrix} \tilde{b}_{r1}^{\bar{\lambda}}(x_o) \\ \tilde{b}_{r2}^{\bar{\lambda}}(x_o) \\ \vdots \\ \tilde{b}_{rn}^{\bar{\lambda}}(x_o) \end{pmatrix} - \frac{1}{\kappa_r^2} \begin{pmatrix} \tilde{a}_{11}^{\bar{\lambda}}(x_o) & \cdots & \tilde{a}_{1n}^{\bar{\lambda}}(x_o) \\ \tilde{a}_{21}^{\bar{\lambda}}(x_o) & \cdots & \tilde{a}_{2n}^{\bar{\lambda}}(x_o) \\ \vdots & \ddots & \vdots \\ \tilde{a}_{n1}^{\bar{\lambda}}(x_o) & \cdots & \tilde{a}_{nn}^{\bar{\lambda}}(x_o) \end{pmatrix} \begin{pmatrix} w_{r1}^{\lambda} \\ w_{r2}^{\lambda} \\ \vdots \\ w_{rn}^{\lambda} \end{pmatrix}. \tag{53}$$

We can directly maximize $\hat{M}(\bar{\lambda}, \lambda, x_o)$ with respect to λ by setting the gradient to zero and solving for $w_{r\cdot}^{\lambda}$. Since the $\tilde{a}$ and $\tilde{b}$ coefficients are now a function of $\bar{\lambda}$, not λ, the equation is linear in $w_{r\cdot}^{\lambda}$ and can be solved analytically. The solution becomes the new $\bar{\lambda}$ and (53) can be solved again until convergence is achieved.

2.6 Time discretization

Working with an underlying continuous–time model is advantageous for the following reasons: 1) the solutions obtained can be used for any time interval without altering the model parameters; 2) the models enforce continuity constraints that may be present in natural signals; 3) they can easily handle irregular sampling rates and missing samples and; 4) they serve as a theoretical guide for hardware implementations in analog VLSI. In practice, lacking hardware implementations of diffusion networks, we model them on digital computers using discrete time approximations: stochastic integrals are replaced by sums, and stochastic differential equations are approximated using an Euler technique.

For an example of discrete time approximations to SDEs note that the Monte–Carlo estimate $\hat{L}_o^{\lambda}(x_o)$ in (27) requires, among other things, computing integrals of the form

$$\log L^{\lambda}(x) = \frac{1}{\sigma^2} \int_0^T \mu(t, x(t), \lambda) \cdot dx(t) - \frac{1}{2\sigma^2} \int_0^T |\mu(t, x(t), \lambda)|^2 dt, \text{ for } x \in \Omega. \tag{54}$$

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

$$\frac{1}{\sigma^2} \sum_{k=0}^{s-1} \mu(t_k, x(t_k), \lambda) \cdot (x(t_{k+1}) - x(t_k)) - \frac{1}{2\sigma^2} \sum_{k=0}^{s-1} |\mu(t_k, x(t_k), \lambda)|^2 \Delta t \tag{55}$$

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

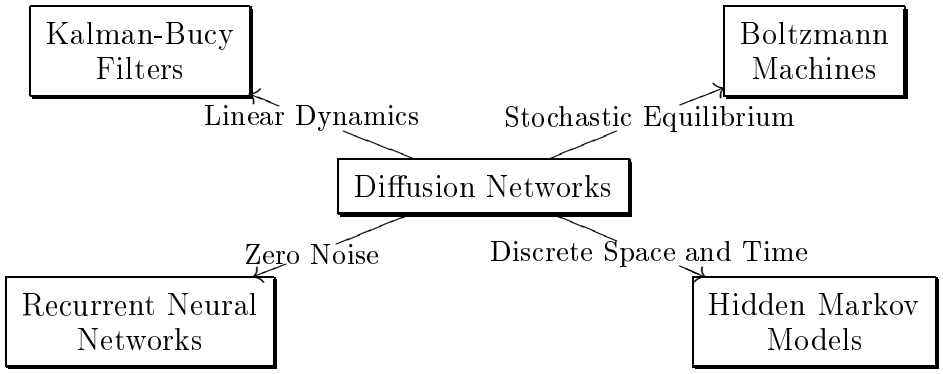

Figure 2: Relationship between diffusion filters and other approaches in the neural network and stochastic filtering literature.

For an example of discrete time approximation to stochastic differential equations note that the Monte–Carlo approach described in the previous section requires generating sample paths $\{h^l\}_{l=1}^m$ from $S_{h|o}^{\lambda,x_o}$. We obtain these paths by simulating the diffusion network dynamics using the following Euler approximation:

$$\begin{aligned} H(t_{k+1}) &= H(t_k) + \mu_h(t_k, x_o(t_k), H(t_k), \lambda)\Delta t_k + \sigma Z_k \sqrt{\Delta t_k}, \\ H(0) &\sim \nu, \end{aligned} \tag{56}$$

where $Z_1 \cdots Z_s$ are i.i.d. $(n-d)$-dimensional Gaussian random vectors with zero mean and covariance equal to the identity matrix.

3 Comparison with previous work

The motivation for this paper was to combine the versatility of recurrent neural networks with the well known advantages of stochastic modeling approaches. Our work builds upon the rich literature on continuous recurrent neural networks (Pearlmutter, 1989), stochastic neural networks (Geman & Geman, 1984; Ackley, Hinton, & Sejnowski, 1985; Smolensky, 1986; Apolloni & de Falco, 1991), and continuous stochastic filtering (Campillo & Le Gland, 1989).

Figure 2 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 SDE models (Campillo & Le Gland, 1989) and 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 the model defines all the finite dimensional distributions. The continuous–state representation is well suited to problems involving inference about continuous unobservable quantities, such as object tracking tasks. Since these networks enforce continuity constraints in the observable paths they may not have the well known problems encountered when HMMs are used to generate sequences (Rabiner & Juang, 1993).

If the logistic activation function φ in (5) is replaced by a linear function, the weights between the observable units and from the observable to the hidden units 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. If the previous constraints are preserved but the initial distribution is allowed to be non-Gaussian then the resulting dynamics are the same as those underlying the stochastic model used in the condensation algorithm (Isard & Blake, 1998). If the weight matrix w is symmetric, at stochastic equilibrium diffusion networks behave like continuous time, continuous state Boltzmann machines (Movellan, 1998). Finally, if the dispersion constant σ is set to zero the network becomes a standard deterministic recurrent neural network.

Campillo and Le Gland (1989) present an alternative approach to maximum likelihood estimation for partially observable SDE models. Their approach, which is restricted to models in which the drift of the unobservable units is not a function of the observable states, involves solving stochastic partial differential equations. Contrary to Campillo and Le Gland (1989) we allow for the unobservable drift to be a function of the observable states, i.e., we allow feedback connections from the observable units into the hidden units. Moreover our approach is based on the use of Monte–Carlo sampling techniques. In our choice of a Monte–Carlo approach we were inspired by the success of the condensation algorithm (Isard & Blake, 1998), which is a sequential sampling technique used in the computer vision literature to track the contours of objects. The sequential sampling technique used in the condensation approach can be easily ported to diffusion networks if the goal at hand is stochastic filtering (e.g., object tracking). The learning algorithm presented here generalizes the one presented on page 242 of Blake and Isard (1998). Their algorithm is restricted to the case in which there are no hidden state units, i.e., $d = n$ and the activation function is linear, e.g., $\varphi(x) = x$. The use of hidden units in the system dynamics proved beneficial for the application we tried, which is described in the next section.

4 Applications

As mentioned earlier we use diffusion networks as a way to parameterize distributions of time varying signals. The analysis presented in previous sections provides methods to find values of λ for which $\nabla_\lambda \Phi(\lambda)$ vanishes. This process is known as learning in the neural network literature, system identification in the engineering literature and parameter estimation in the statistical literature. Trained networks can then be used for a variety of tasks including stochastic filtering, prediction, smoothing, decoding, sequence generation, and sequence recognition. In this paper we focus on sequence recognition applications since such applications are currently dominated by standard HMM approaches.

Figure 3 illustrates our approach to sequence recognition. 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

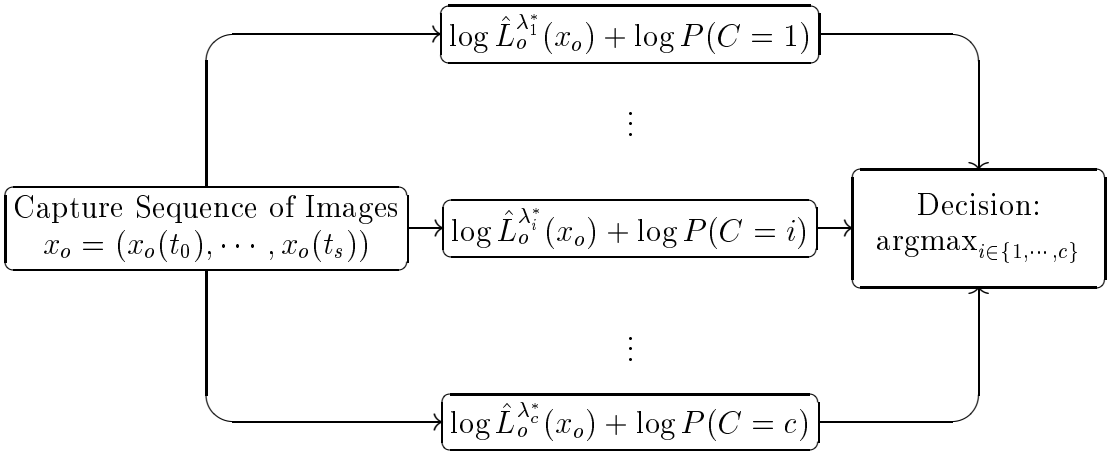

Figure 3: Schematic representation of the use of a bank of diffusion networks to classify a sequence x_o into one of c categories.

$\lambda_1^*, \cdots, \lambda_c^*$. Once the networks are trained we can classify a new observed sequence x_o as follows: we compute $\log L_o^{\lambda_i^*}(x_o)$ for $i = 1, \cdots, c$. These log–likelihoods are combined with the log-prior probability of each category and the most probable category of the sequence is chosen. In the next section we present an application of such an approach to a speaker independent visual speech recognition task.

4.1 Example: Visual speech recognition

In this section we illustrate the use of diffusion networks for a sequence classification task of reasonable difficulty with a body of realistic data. We compare a diffusion network approach with classic hidden Markov model approaches.

Training database: We used Tulips1 (Movellan, 1995), a database consisting of 96 movies of 9 male and 3 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" e.g., Figure 4. The database is challenging due to variability in illumination, gender, ethnicity of the subjects and position/orientation of the lips. The database is available at http://cogsci.ucsd.edu.

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 2 different recognition engines (HMMs and diffusion networks). The image processing was performed by Luettin and colleagues (Luettin, Thacker, & 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 4). The dimensionality of the representation of the contours is reduced using principal component analysis. For the work presented here 10

principal components were used to approximate the contour, along with a scale parameter which 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}, with k varying across all the sampling times (these delta components are defined to be zero at t_0). In this manner a total of 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.

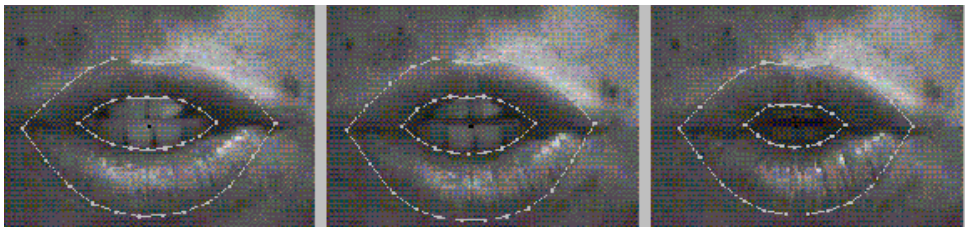

Figure 4: Excerpt from an image sequence of a subject saying the digit three, from the Tulips 1 database. Contours where tracked by Luettin and colleagues.

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 which 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.

Training: We independently trained 4 diffusion networks, to approximate the distributions of lip-contour trajectories of each of the four words to be recognized, i.e., 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 (4) with $\kappa_i = 1$, $1/\rho_i = 0$ for all units, and $\xi(t) = \beta$ constant with respect to time. Thus, $\lambda = (w_{1,1}, w_{1,2}, \cdots w_{n,n}, \beta_1, \cdots, \beta_n)'$. 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 a 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 4 digits using the cost function

$$\Phi(\lambda) = \sum_i \log \hat{L}_o^\lambda(x_o^i) - \Psi(\lambda), \tag{57}$$

where each x_o^i is a movie from the Tulips1 database, i.e., 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

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%

Table 1: Generalization performance on the Tulips1 database. Shown in order are the performance of the best performing HMM from (Luettin et al., 1996), 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 lipreaders (Movellan, 1995).

is reported with the optimal values. Here $\Psi(\lambda)$ acts as a Gaussian prior on the network parameters. The log–likelihood gradients were estimated using (28) with $m = 20$, i.e., we generated 20 hidden sample paths per observed path. These paths were sampled using the "teacher forcing" approach described in equations (32)–(34). The function (57) was optimized using a conjugate gradient method (Press, Teukolsky, Vetterling, & Flannery, 1992).

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 labelled data from 11 subjects, leaving a subject out for generalization testing. Percent correct generalization was then tested using the decision rule

$$D(x_o) = \underset{i \in \{1,2,3,4\}}{\operatorname{argmax}} \ \log \hat{L}_o^{\lambda_i^*}(x_o), \tag{58}$$

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. In terms of Figure 3 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 HMM architectures.

We tested performance using a variety of architectures, some including no hidden units, some with up to 100 hidden units. Best generalization performance was obtained using 4 hidden units. These generalization results are shown in Table 1. The HMM results are those reported in Luettin et al. (1996a, 1996b; Luettin, 1997). The only difference between their approach and ours is the recognition engine, which is a bank of HMMs in their case

and a bank of diffusion networks in our case. The image representations mentioned above were optimized by Luettin et al. to work with HMMs. They also tried a variety of HMM architectures and reported the best results obtained with them.

In all cases the best diffusion networks outperformed the best HMMs reported in the literature using exactly the same visual preprocessing. The results show that diffusion networks may outperform HMM 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.

5 Discussion

We presented a framework for density estimation of continuous time varying signals using diffusion networks, a stochastic version of continuous recurrent neural networks. The approach allows training diffusion networks with non-linear activation functions and hidden states. Once a network has been trained, standard approaches from statistical decision theory can be used to do sequence classification, stochastic filtering (e.g. contour tracking), and sequence generation within the same underlying framework. We presented an example of how the approach could be used on a realistic sequence recognition task.

Significant work remains to be done, and we have identified several areas for further research. First, the various restrictions we have imposed on the dynamics in equations (1)–(2) need to relaxed. In particular, we have not considered dispersions which are a function of the state, and we have not optimized the dispersion and the initial distribution of hidden states. Optimization of the dispersion is especially intriguing since it is (with appropriate scaling of the drift) equivalent to optimization of the time scale of the model.

Second, faster training techniques would be desirable. At this point the main disadvantage of diffusion networks relative to conventional hidden Markov models is training speed. The diffusion networks used here were approximately 10 times slower to train than HMMs. Fortunately the Monte–Carlo approximations employed herein, which represent the bulk of the computational burden, lend themselves to parallel and hardware implementations. Moreover, once a network is trained, the computation of the density functions L_o needed in recognition tasks can be done very quickly.

Theoretical issues concerning the approximating power of diffusion networks need to be explored. For example, in the application presented in the previous section we set the parameter $1/\rho_i$ to zero, which represents infinite transmembrane resistance. This value is non-standard in the neural network literature but, to our surprise, gave us substantially better results. We do not have a good theory to explain this observation. 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. An analogous result for diffusion networks, stating a class of distributions over the observable nodes which can be approximated arbitrarily closely given sufficient hidden units and constraints on the drift, dispersion and initial distribution, might provide insight as to what are "good" choices for diffusion network dynamics.

Finally we need to explore applications of diffusion networks to stochastic filtering problems (e.g., contour tracking) and sequence generation tasks, not just sequence recognition

problems. In current applications objects are commonly tracked using SDE models and then sequences of these tracked objects (e.g., sequences of lip contours) are recognized using HMMs (Blake & Isard, 1998). In principle it should be possible to do stochastic filtering (e.g., contour tracking) and sequence recognition with the same diffusion network, potentially saving computing time by dispensing with the HMMs. Our work shows that diffusion networks may be a feasible alternative to HMMs for problems in which state continuity is advantageous. The results obtained for the visual speech recognition task are encouraging and reinforce our belief that diffusion networks may become a versatile neural network tool for a very wide variety of continuous signal processing tasks.

Acknowledgements

The authors thank Anthony Gamst for helpful discussion, and Juergin Luettin for generously providing lip contour data and technical assistance. Research of R. J. Williams is supported in part by NSF Grant DMS 9703891.

6 References

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

Apolloni, B., & de Falco, D. (1991). Learning by asymmetric parallel Boltzmann machines. *Neural Computation*, *3*(3), 402–408.

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

Blake, A., & Isard, M. (1998). *Active contours*. London: Springer.

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, Pennsylvania: SIAM.

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

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.

Hertz, J., Krogh, A., & Palmer, R. (1991). *Introduction to the theory of neural computation*. 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.

Isard, M., & Blake, A. (1998). Condensation: conditional density propagation for visual tracking. *Int. J. Computer Vision*, *29*(1), 5–28.

Karatzas, I., & Shreve, S. (1991). *Brownian motion and stochastic calculus*. Springer.

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

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

Luettin, J. (1997). *Visual speech and speaker recognition*. PhD thesis, University of Sheffield.

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

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

Mineiro, P., Movellan, J., & Williams, R. J. (1998). Learning path distributions using nonequilibrium diffusion net works. In M. Kearns (Ed.), *Advances in neural information processing systems*. Cambridge, Massachusetts: 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*, Vol. 6 (pp. 83–87). 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*, Vol. 7. MIT Press.

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

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

Oksendal, B. (1991). *Stochastic differential equations*. Springer-Verlag.

Pearlmutter, B. A. (1989). Learning state space trajectories in recurrent neural networks. *Neural Computation*, *1*(2), 263–269.

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

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1992). *Numerical recipes in C*. Cambridge University Press, second edition.

Protter, P. (1990). *Stochastic integration and differential equations*. Springer.

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

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 (Chap. 6, pp. 194–281). Cambridge, MA: MIT Press.

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