Spiking Sequence Learning using Maximum Likelihood: Hopfield Networks

David Barber* Felix Agakov

Division of Informatics, University of Edinburgh, Edinburgh EH1 2QL, UK

d.barber@anc.ed.ac.uk felixa@dai.ed.ac.uk http://anc.ed.ac.uk

July 15, 2002

Abstract

We consider the learning of correlated temporal sequences from a statistical viewpoint, using Maximum Likelihood to derive a simple local learning rule for an idealized spiking neuron model. The rule is capable of robustly storing multiple sequences of correlated patterns in a recurrent network model, unlike other prescriptions based on pattern correlations alone.

1 Introduction

Simple models of temporal sequence storage devices are typically based on recurrent networks that model the hippocampal region CA3 with training sequence inputs represented

*Corresponding author.

by projections from entorhinal cortex and dentate gyrus (Shon et al., 2002). Mechanisms by which a network can learn to store temporal sequences are usually based on local synaptic plasticity so that the efficacy w_{ij} with which neuron j mediates the firing of neuron i is a function of only the joint firing pattern of neurons i and j (Dayan and Abbott, 2001; Rolls and Treves, 1998). Many idealized models of memory interpret Hebbian synaptic plasticity (Sejnowski, 1999) in terms of correlations (or covariances) of desired memory states, possibly moderated to ensure that synaptic strengths remain within biologically plausible bounds (Dayan and Abbott, 2001). Here we examine some well known synaptic modification schemes that can encode temporal sequences in a simple stochastic recurrent network. We derive a simple learning rule using a statistically sound retrieval criterion. The rule is capable of robustly recalling multiple temporal sequences under cued retrieval. Importantly, the rule is local but *cannot* be written as a function of the correlation between desired memory states.

2 Stochastic Hopfield Networks

Hopfield networks are an idealized model of distributed computation, with particular application as a simple model of memory (van Hemmen and Kühn, 1991; Hertz et al., 1991; Coolen, 1997). The model represents a set of V interconnected neurons, each being at any time t in one of two possible states, $s_i(t) = +1$ (firing) or $s_i(t) = -1$ (quiescent). Neuron i fires depending on the potential

$$a_i(t) \equiv \theta_i + \sum_{j=1}^{V} w_{ij} s_j(t) \tag{1}$$

where w_{ij} characterizes the synaptic efficacy from neuron j (presynaptic) to neuron i (postsynaptic). The threshold θ_i relates to the neuron's predisposition to firing. Writing the state of the network at time t as $\mathbf{s}(t) \equiv (s_1(t), \ldots, s_V(t))$, the probability that neuron i fires at time $t+1$ is modelled as

$$p(s_i(t+1) = 1|\mathbf{s}(t)) = \sigma\left(a_i(t)\right) \tag{2}$$

where $\sigma(x) = 1/(1+e^{-\beta x})$ and the parameter β controls the level of stochastic behaviour of the neuron. In the limit $\beta \to \infty$, the neuron updates deterministically

$$s_i(t+1) = \operatorname{sgn}\left(a_i(t)\right). \tag{3}$$

The probability of being in the quiescent state is given by normalization, so that the neuronal update can be compactly written as

$$p(s_i(t+1)|\mathbf{s}(t)) = \sigma\left(s_i(t+1)a_i(t)\right) \tag{4}$$

which follows directly from $1-\sigma(x) = \sigma(-x)$. Under synchronous updating, the network dynamics is given by

$$p(\mathbf{s}(t+1)|\mathbf{s}(t)) = \prod_{i=1}^{V} p(s_i(t+1)|\mathbf{s}(t)), \tag{5}$$

simultaneously generating a new set of neuron states. Equation (5) defines a Markov transition matrix, modelling the probability of a transition $\mathbf{s}(t) \to \mathbf{s}(t+1)$ and furthermore imposes the constraint that the neurons are conditionally independent given the previous state of the network. Note that equations (5) and (2) differ significantly from the Boltzmann machine formulation (Hertz et al., 1991) in that there is no restriction to symmetric weights, and the form of the equilibrium distribution is unknown (Coolen, 1997). In the following section, we address in the stochastic case how to learn suitable parameters w_{ij} and θ_i to store temporal sequences.

3 Learning Sequences using Maximum Likelihood

By storage we mean that if the network is initialized in the correct starting state of the training sequence, the remainder of the training sequence will be reproduced under the network dynamics without error – a form of cued retrieval. Thus, to store a temporal sequence $\mathcal{V} = \{\mathbf{s}(1) = \mathbf{v}(1), \ldots, \mathbf{s}(T) = \mathbf{v}(T)\}$, we need to adjust the network parameters such that the conditional likelihood

$$p(\mathbf{s}(T) = \mathbf{v}(T), \mathbf{s}(T-1) = \mathbf{v}(T-1), \ldots, \mathbf{s}(2) = \mathbf{v}(2) | \mathbf{s}(1) = \mathbf{v}(1)) \tag{6}$$

is maximized. Henceforth, for brevity, we simplify the notation above to $p(\mathbf{v}(T), \mathbf{v}(T-1), \ldots, \mathbf{v}(2)|\mathbf{v}(1))$. Furthermore, we might hope that the sequence will be recalled with high probability not just when initialized in the correct state but also for states close, in Hamming distance, to the correct initial state $\mathbf{v}(1)$.

Due to the Markov nature of the dynamics, the conditional likelihood is

$$p(\mathbf{v}(T), \mathbf{v}(T-1), \ldots, \mathbf{v}(2)|\mathbf{v}(1)) = \prod_{t=1}^{T-1} p(\mathbf{v}(t+1)|\mathbf{v}(t)). \tag{7}$$

The transition probabilities are given by (5,2), so that the conditional log-likelihood becomes

$$L \equiv \log \prod_{t=1}^{T-1} p(\mathbf{v}(t+1)|\mathbf{v}(t)) = \sum_{t=1}^{T-1} \sum_{i=1}^{V} \log \sigma\left(v_i(t+1)a_i(t)\right) \tag{8}$$

where $a_i(t) = \theta_i + \sum_j w_{ij} v_j(t)$. To increase the likelihood of the sequence, we can use simple gradient ascent

$$w_{ij}^{new} = w_{ij} + \eta \frac{dL}{dw_{ij}}, \qquad \theta_i^{new} = \theta_i + \eta \frac{dL}{d\theta_i} \tag{9}$$

where

$$\frac{dL}{dw_{ij}} = \beta \sum_{t=1}^{T-1} \gamma_i(t) v_i(t+1) v_j(t), \qquad \frac{dL}{d\theta_i} = \beta \sum_{t=1}^{T-1} \gamma_i(t) v_i(t+1) \tag{10}$$

and we have defined

$$\gamma_i(t) \equiv 1 - \sigma\left(v_i(t+1)a_i(t)\right) \qquad \in (0,1). \tag{11}$$

The learning rate η is chosen empirically to be sufficiently small to ensure convergence. Throughout, we set β to 1, unless otherwise explicitly stated. The batch training procedure (10) can be readily converted to an online procedure since the updates only depend on two consecutive patterns. The above Maximum Likelihood (ML) learning rule can be seen as a modified Hebb learning rule, the basic Hebb rule being given when $\gamma_i(t) \equiv 1$. The rule can also be interpreted as a Widrow-Hoff/Delta rule (e.g. Hertz et al. (1991)) in the context of logistic regression. As learning progresses, the $\gamma_i(t)$ will typically tend to values close to either 1 or 0, and hence the learning rule can be seen as asymptotically equivalent to making an update only in the case of disagreement ($a_i(t)$ and $v_i(t+1)$ are of different signs).

To assess convergence of the learning rule, consider the Hessian of the log-likelihood[1]:

$$\frac{d^2L}{dw_{ij}dw_{kl}} \equiv H_{ij,kl} = -\sum_{t=1}^{T-1} (v_i(t+1)v_j(t))\, \gamma_i(t)(1-\gamma_i(t))v_k(t+1)v_l(t)\delta_{ik}. \tag{12}$$

This is negative definite and hence the likelihood has a single global maximum. This can be shown by calculating $\sum_{ijkl} x_{ij} H_{ij,kl} x_{kl}$ for arbitrary vectors x_{ij} and x_{kl} which gives $-\sum_{ijlt} x_{ij} v_j(t)\gamma_i(t)(1-\gamma_i(t))v_l(t)x_{il} = \sum_{it} y_i(t)\gamma_i(t)(\gamma_i(t)-1)y_j(t) \equiv \sum_t \mathbf{y}(t)^T\mathbf{D}(t)\mathbf{y}(t)$, where we defined $y_i(t) = \sum_j w_{ij}v_j(t)$. Since $\mathbf{D}(t)$ is a diagonal matrix with negative eigenvalues $\gamma_i(t)(\gamma_i(t)-1) < 0$, this shows that $\mathbf{H}$ is negative definite. Gradient ascent is therefore guaranteed to converge to the global maximum, provided that η is not too large. Since the gradient (10) is zero only for infinite weights, learning in principle never stops. However, the learning rapidly slows down after a small number of iterations, and can be safely stopped early.

3.1 Stochastic Interpretation

By straightforward manipulations, (10) can be written as

$$\frac{dL}{dw_{ij}} = \sum_{t=1}^{T-1} \frac{1}{2}\left(v_i(t+1) - \langle s_i(t+1)\rangle_{p(s_i(t+1)|a_i(t))}\right) v_j(t). \tag{13}$$

[1]We neglect the biases and set $\beta = 1$ for expositional clarity – this does not affect the conclusions.

An online, stochastic version of ML learning is then given by the weight updates

$$\Delta w_{ij}(t) = \eta \left(v_i(t+1) - \tilde{s}_i(t+1)\right) v_j(t) \tag{14}$$

where $\tilde{s}_i(t+1)$ is 1 with probability $\sigma(\theta_i + \sum_j w_{ij} v_j(t))$, and -1 otherwise. Equation (14) is interesting since it makes a clear distinction between v_i, the desired pattern state and s_i, the current network state[2]. The above form is the most biologically plausible interpretation of the learning rule. Provided that the learning rate η is small, stochastic updating will approximate the learning rule (9,10).

3.2 Capacity

The Hopfield model is capable of storing a sequence of V linearly independent patterns. To show this, we consider the case $\beta \to \infty$, such that the network performs the updates

$$s_i(t+1) = \mathrm{sgn} \sum_j w_{ij} s_j(t). \tag{15}$$

(we set θ to zero for simplicity). Here w_{ij} are the elements of the weight matrix $\mathbf{W}$ representing the transitions from time t to time $t+1$. A desired sequence $\mathbf{v}(1), \ldots, \mathbf{v}(T)$ can be recalled correctly if we can find a matrix $\mathbf{W}$ and real numbers $\tau_i(t)$ such that

$$\mathbf{W}\left[\mathbf{v}(1), \ldots, \mathbf{v}(T-1)\right] = \left[\boldsymbol{\tau}(2), \ldots, \boldsymbol{\tau}(T)\right]$$

[2] Biologically, the rule could be implemented by measuring the difference between clamping neuron i into state $v_i(t+1)$ all other neurons, $j \neq i$ into states $v_j(t)$, and subsequently releasing the clamp from neuron i. This clamping procedure requires some kind of external source, and one suggestion is that in the hippocampus this could be the role of connections from the entorhinal cortex and dentate gyrus to CA3 (Shon et al., 2002).

where the $\tau_i(t)$ are arbitrary real numbers for which $\mathrm{sgn}(\tau_i(t)) = v_i(t)$. This system of linear equations can be solved if the columns of the matrix $[\mathbf{v}(1), \ldots, \mathbf{v}(T-1)]$ are linearly independent. Thus, for perfect recall of the patterns, we only need the training sequence to form a set of linearly independent vectors. The maximum likelihood procedure then finds a matrix $\mathbf{W}$ and, implicitly, a set of values $\boldsymbol{\tau}(2), \ldots, \boldsymbol{\tau}(T)$ such that, under the dynamics (15), the correct sequence will be recalled. The reason for this is that ML learning (10) effectively stops when $\forall i . v_i(t+1)a_i(t) \gg 1$, which is precisely the condition that the transition $\mathbf{v}(t) \to \mathbf{v}(t+1)$ is stored by the network. This is the only condition under which the gradients (10) are zero, since by the assumption of linear independence, $\forall (i,t). \gamma_i(t) = 0$. Using stochastic recall (5), however, results in a small probability that the sequence would be generated incorrectly due to the stochastic nature of sampling. In practice, we train with a finite value for β since then the log-likelihood is differentiable.

3.3 Multiple Sequences

How can we train the network to learn a set of sequences $\{\mathcal{V}^s, s = 1, \ldots S\}$? If we assume that the sequences are independent, the log likelihood of a set of sequences is simply the sum of the individual sequences:

$$\frac{dL}{dw_{ij}} = \beta \sum_{s=1}^{S} \sum_{t=1}^{T-1} \gamma_i^s(t) v_i^s(t+1) v_j^s(t), \qquad \frac{dL}{d\theta_i} = \beta \sum_{s=1}^{S} \sum_{t=1}^{T-1} \gamma_i^s(t) v_i^s(t+1) \tag{16}$$

where

$$\gamma_i^s(t) \equiv 1 - \sigma\left(v_i^s(t+1) a_i^s(t)\right), \qquad a_i^s(t) = \theta_i + \sum_j w_{ij} v_j^s(t). \tag{17}$$

The log likelihood is concave since it is the sum of concave functions. This learning rule is capable of storing K sequences of length V/K. Indeed, the network is capable of storing an arbitrary set of V transitions $\mathbf{v}(t) \to \mathbf{v}(t+1)$, provided the patterns are linearly independent.

4 Relation to other rules

For deterministic dynamics (3), two well known approaches to learning a sequence $\mathcal{V} = \{\mathbf{v}(1), \ldots, \mathbf{v}(T)\}$ are the Hebb and Pseudo Inverse rules (Hertz et al., 1991):

Hebb rule: $w_{ij} = \sum_{t=1}^{T-1} v_i(t+1) v_j(t)$

Pseudo Inverse : $Q_{ij} = \sum_{t=1}^{T} v_i(t) v_j(t) \qquad w_{ij} = \sum_{t=1}^{T-1} v_i(t+1) [Q^{-1}]_{ij} v_j(t).$

By 'Hebb rule', we mean the equation above, since it is common in the literature to interpret Hebbian plasticity in terms of the pattern correlations. We recognise, however, that Hebbian plasticity is not necessarily restricted to the form above. For both the Hebb and Pseudo Inverse rules, the thresholds θ_i are usually set to zero.

The Hebb rule is capable of storing a random uncorrelated temporal sequence of length $0.269V$ time steps, in contrast to the well known capacity result of $0.138V$ for static patterns (Düring et al., 1998). However, the Hebb rule performs poorly for the case of correlated patterns. In contrast, the Pseudo Inverse (PI) rule can store a sequence of V linearly independent patterns (Hertz et al., 1991). The PI rule, whilst attractive compared to the standard Hebb in terms of its ability to store a longer sequence is unattractive for two reasons: biological implausibility and instability. Due to the inverse, the value of the connection w_{ij} is not simply a function of values computable locally at nodes i and j. By writing the inverse as an iteration, however, it is possible to recover a local learning rule (Diederich and Opper, 1986). The instability of the PI rule with respect to even very small pattern perturbations is well known, and for the case of static patterns recall is unstable if the number of (uncorrelated) patterns exceeds $V/2$ (van Hemmen and Kühn, 1991). Our experiments show that the PI rule also has small basins of attraction for temporally correlated patterns.

4.1 Relation to the Perceptron Rule

In the limit that the activation is large, $\gamma_i(t) = 1$ if $v_i(t+1)a_i(t) < 0$ and is zero otherwise. In this limit, (10) leads to the well known Perceptron rule (Hertz et al., 1991; Diederich and Opper, 1986). The Perceptron rule generalizes poorly unless we include a stability criterion such that we set $\gamma_i(t) = 1$ if $v_i(t + 1)a_i(t) < M$ and is zero otherwise, where M is some positive margin. The threshold parameter θ_i (here set to zero for simplicity), which relates to the minimal distance of each hyperplane $\mathbf{w}^i$ from the origin, should not be confused with the margin M which relates to the distance of patterns from the hyperplane $\mathbf{w}^i$. An advantage of remaining with our probabilistic version is that we do not need to find an appropriate setting of the margin. Additionally, we can assign a likelihood score to a novel temporal sequence, which can be used for classification purposes if we so wish, whilst in the deterministic limit, the likelihood score for a novel sequence is trivially either 1 or 0. Finally, the convergence of the stochastic model is guaranteed since the likelihood surface is concave.

Whilst (14) is similar to the $M = 0$ Perceptron rule, the stochastic nature of the updating has the important effect of increasing stability during recall, performing dramatically better than the $M = 0$ Perceptron rule. In our simulations, we implement our ML rule using (13), rather than (14).

5 Results

In fig(1)(left) we compared the performance of the rule (with zero thresholds) with the standard Hebb, Pseudo Inverse, and Perceptron rules for learning a single temporal sequence of length 20 in a 100-neuron network. We produced the training sequence by starting from a random initial state, $\mathbf{v}(1)$, and then choosing at random 20% percent of the neurons potentially to flip, each of the chosen neurons being flipped with probability 0.5. This produces a random training sequence with a high degree of temporal correlation. The network is initialized to a noise corrupted version of the correct initial state $\mathbf{v}(1)$ from the training sequence. This corruption is achieved by flipping with a specified probability each neuron of the correct initial state. We run the dynamics for T timesteps and measure the fraction of neurons in the final state $\mathbf{s}(T)$ that agree with the training final state $\mathbf{v}(T)$. Thus a value of 1 indicates perfect recall of the final state, and a value of 0.5 indicates a performance no better than random guessing of the final state. At each stage in the reconstruction dynamics, the state of the network was corrupted with noise by flipping each neuron

$$s_i(t+1) = \operatorname{sgn}\left(\sum_j w_{ij} s_j(t) \epsilon_j(t)\right), \quad \epsilon_j(t) \in \{-1, +1\} \tag{18}$$

with a specified flip probability $p(\epsilon_j = -1)$. In all cases, batch training was used. In the ML rule, we used the exact values of $\gamma_i(t)$ for updating.

It is immediately clear from the results that the standard Hebb rule performs poorly, particularly for small flip rates, whilst the other methods perform relatively well, being robust against small flip rates. As the flip rate increases, the PI rule becomes unstable. The non-monotonic behaviour in the performance of the Hebb rule is curious, but of minor consequence due to its inferior performance. The Perceptron rule can perform as well as the ML rule, although its performance is critically dependent on an appropriate choice of the margin M, the results for $M = 0$ Perceptron training being poor for small flip rates. An advantage of the ML rule is that it performs well without the need for fine tuning of parameters. Figure 1(b) illustrates reconstruction of a temporal sequence of length 20 by a network of 100 neurons trained using the ML, Hebb, and the PI learning rules. The sequence is chosen to be highly correlated, which makes the learning task generally more difficult. The initial state of the training sequence was corrupted by 30% noise, and subsequent states are generated according to $s_i(t+1) = \mathrm{sgn}\left(\sum_j w_{ij} s_j(t)\right)$. We see that both the Hebb and the PI rule are very sensitive to perturbations, while the ML rule and Perceptron rules perform more robustly.

6 Discussion

Maximum Likelihood is a natural criterion for temporal sequence storage in an ensemble of stochastically spiking neurons. We showed that the resulting learning rule is capable of robustly storing a temporal sequence of length equal to the number of neurons in

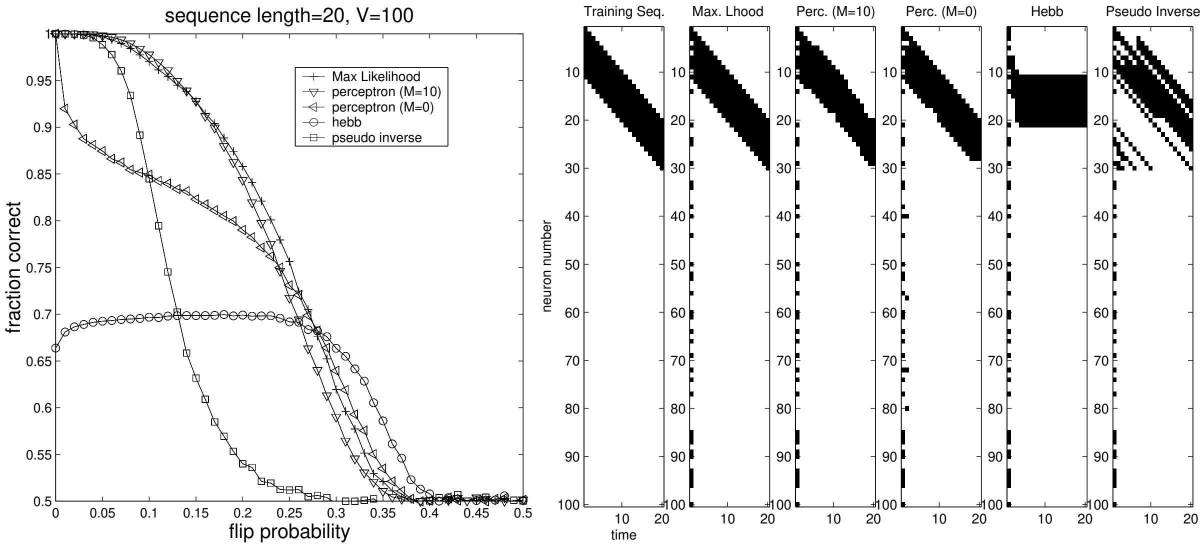

Figure 1: **Left**: Learning a temporal sequence, $T = 20, V = 100$. Plotted is the fraction of neurons $\mathbf{s}(T)$ that are in the correct state $\mathbf{v}(T)$ after the network was initialized in the correct initial state and then run for 20 timesteps. Throughout the dynamics, noise of the specified amount corrupted the network state before every update. 50 training epochs were made using $\eta = 0.05$, and the results presented are averages over 5000 simulations for a network of 100 neurons resulting in standard errors of the order of the symbol sizes. **Right** : The leftmost plot shows the training sequence. The other plots show the temporal evolution of the network after initialization in the correct starting state but corrupted with 30% noise. The network was trained using 10 batch epochs with $\eta = 0.1$, and $\beta = 1$. In all figures deterministic updates $\beta = \infty$ were used during recall.

the network. We compared this rule with other popular methods for sequence storage, including the Hebb, Pseudo Inverse and Perceptron rules, these alternatives performing less well. The Maximum Likelihood rule can be interpreted as a stochastic version of the Perceptron learning rule, where the stochastic nature of the updates greatly helps the generalization performance of the network. Whilst this paper concentrated on demonstrating the benefits of using Maximum Likelihood training for the simple Hopfield network, it is straightforward to apply the same approach to learning in complex models of spiking neuron assemblies.

Acknowledgements

We would like to thank David Sterratt, Ton Coolen, Stefano Fusi, and Amos Storkey for helpful discussions and references.

References

Coolen, A. (1997). Statistical Mechanics of Neural Networks. Lecture Notes.

Dayan, P. and Abbott, L. (2001). *Theoretical Neuroscience*. MIT Press.

Diederich, S. and Opper, M. (1986). Learning of Correlated Patterns in Spin-Glass Networks by Local Learning Rules. *Physical Review Letters*, 58(9):949–952.

Düring, A., Coolen, A., and Sherrington, D. (1998). Phase diagram and storage capacity of sequence processing neural networks. *Journal of Physics A*, 31:8607–8621.

Hertz, J., Krogh, A., and Palmer, R. (1991). *Introduction to the theory of Neural Computation.* Addison-Wesley.

Rolls, E. and Treves, A. (1998). *Neural Networks and Brain Function.* Oxford University Press.

Sejnowski, T. (1999). The Book of Hebb. *Neuron*, 24:773–776.

Shon, A., Wu, X., Sullivan, D., and Levy, W. (2002). Initial state randomness improves sequence learning an a model hippocampal. *Physical Review E*, 65.

van Hemmen, J. L. and Kühn, R. (1991). *Collective Phenomena in Neural Networks*, chapter 1. Springer.