Correlated sequence learning in a network of spiking neurons using maximum likelihood

by

David Barber, Felix Agakov

Informatics Research Report EDI-INF-RR-0149

School of Informatics **April 2002**
http://www.informatics.ed.ac.uk/

Correlated sequence learning in a network of spiking neurons using maximum likelihood

David Barber, Felix Agakov

Informatics Research Report EDI-INF-RR-0149

SCHOOL *of* INFORMATICS
Institute for Adaptive and Neural Computation
April 2002

A shorter and revised version has been submitted to Neural Computation, and also NIPS2002

Abstract :
Hopfield Networks are an idealised model of distributed computation in networks of non-linear, stochastic units. We consider the learning of correlated temporal sequences using Maximum Likelihood, deriving a simple Hebbian-like learning rule that is capable of robustly storing multiple sequences of correlated patterns. We argue that the learning rule is optimal for the case of long temporal sequences and has a natural stochastic interpretation.

Keywords : Hopfield networks, temporal sequence learning, maximum likelihood, spiking neurons, Hebb learning

Correlated sequence learning in a network of spiking neurons using maximum likelihood

David Barber† and Felix Agakov

Institute of Adaptive and Neural Computation. 5 Forrest Hill Edinburgh, EH1 2QL, U.K.

† Corresponding author : d.barber@anc.ed.ac.uk

Abstract. Hopfield Networks are an idealised model of distributed computation in networks of non-linear, stochastic units. We consider the learning of correlated temporal sequences using Maximum Likelihood, deriving a simple Hebbian-like learning rule that is capable of robustly storing multiple sequences of correlated patterns. We argue that the learning rule is optimal for the case of long temporal sequences and has a natural stochastic interpretation.

1. Stochastic Hopfield Networks

Hopfield networks are an idealised model of distributed computation, with particular application as a simple model of memory[6, 5, 2]. The model represents a set of V interconnected neurons, each being at any time t in one of two possible states, $v_i(t) = +1$ (firing) or $v_i(t) = -1$ (quiescent). Neuron i fires depending on the potential

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

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 $\mathbf{v}(t) \equiv (v_1(t), \dots, v_V(t))$, the probability that neuron i fires at time $t+1$ is modelled as

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

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

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

The probability of being in the quiescent state is given by normalization

$$p(v_i(t+1) = -1|\mathbf{v}(t)) = 1 - p(v_i(t+1) = +1|\mathbf{v}(t)) = 1 - \sigma\left(v_i(t+1)a_i(t)\right) \tag{4}$$

These two rules can be compactly written as

$$p(v_i(t+1)|\mathbf{v}(t)) = \sigma\left(v_i(t+1)a_i(t)\right) \tag{5}$$

which follows directly from $1 - \sigma(x) = \sigma(-x)$.

In a synchronous Hopfield Net, the update of the entire network is given by the following rule

$$p(\mathbf{v}(t+1)|\mathbf{v}(t)) = \prod_{i=1}^{V} p(v_i(t+1)|\mathbf{v}(t)). \tag{6}$$

The network thus operates by simultaneously generating a new set of neuron states according to (6). Equation (6) defines a Markov transition matrix, modelling the transition probability $\mathbf{v}(t) \to \mathbf{v}(t+1)$ and furthermore imposes the constraint that the neurons are conditionally independent given the previous state of the network. Note that equations (6) and (2), differ significantly from the Boltzmann machine [5] in that there is no restriction to symmetric weights, and the form of the equilibrium distribution is unknown[2].

The deterministic Hopfield net(3) does not capture the inherently stochastic nature of neurons. Furthermore, previous approaches to training the network have relied on asynchronous updating and the subsequent existence of a suitable Lyapunov function to ensure convergence to local attractor states[3, 5]. In the following section, we address how to learn suitable parameters w_{ij} and θ_i to learn given temporal sequences based on a simple local learning rule that we can readily show is capable of storing multiple temporal sequences without error.

2. Learning Sequences

2.1. A Single Sequence

For deterministic dynamics (3), two well known approaches to learning a temporal equence $\mathcal{V} = \{\mathbf{v}(1), \dots, \mathbf{v}(T)\}$ are the Hebb and Pseudo Inverse rules[5]:

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

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

In both cases, the thresholds θ_i are usually set to zero. 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. The Hebb rule is capable of storing a random uncorrelated temporal sequence of length $0.269V$ time steps[4]†. However, the Hebb rule performs poorly for the case of correlated patterns. In contrast, the Pseudo Inverse (PI) rule can store any sequence of V linearly independent patterns. 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‡. We will show in our experiments that this temporal PI rule has very small basins of attraction for temporally correlated patterns.

The deficiencies of these two well known algorithms motivated us to search for a robust, local learning rule capable of storing a sequence of V correlated patterns. First, we need to clarify what we mean by "store". Two different objectives readily come to mind:

† This is in contrast to the well known capacity result of $0.138V$ for static patterns.
‡ By writing the inverse as an iteration, however, it is possible to recover a local learning rule[3].

Unconditional Maximum Likelihood: We wish that the sequence $\mathcal{V}$ will be likely to be generated by the network, given a random starting condition. That is, adjust the parameters of the network such that the probability that the network generates the sequence, $\mathcal{V}$,

$$p(\mathbf{v}(T), \mathbf{v}(T-1), \ldots, \mathbf{v}(1)) \tag{7}$$

is high. This objective function is typically intractable to evaluate, as we will subsequently explain, and an alternative objective function would be preferable.

Conditional Maximum Likelihood: Given that we initialize the network in a state $\mathbf{v}(t = 1)$, we wish that the remaining sequence will be generated with high probability. That is, adjust the network parameters such that the probability

$$p(\mathbf{v}(T), \mathbf{v}(T-1), \ldots, \mathbf{v}(2)|\mathbf{v}(1)) \tag{8}$$

is maximized. 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 $\mathbf{v}(1)$.

These two objectives are related by†

$$p(\mathbf{v}(T), \mathbf{v}(T-1), \ldots, \mathbf{v}(2)|\mathbf{v}(1))p(\mathbf{v}(1)) = p(\mathbf{v}(T), \mathbf{v}(T-1), \ldots, \mathbf{v}(1)) \tag{9}$$

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

$$p(\mathbf{v}(T), \mathbf{v}(T-1), \ldots, \mathbf{v}(2)|\mathbf{v}(1)) = \prod_{t=1}^{T-1} p(\mathbf{v}(t+1)|\mathbf{v}(t)) \tag{10}$$

This is a product of transitions from given states to given states. Since these transition probabilities are known (6,2), the conditional likelihood can be easily evaluated. However, according to (9), to calculate the unconditional likelihood, we require knowledge of the equilibrium distribution $p(\mathbf{v}(1))$, which is non-trivial to calculate‡. We believe that the more natural use of the Hopfield network is to recall a training sequence given an initial state and will henceforth focus on this application here and how to train a network accordingly. The sequence log (conditional) likelihood is

$$L \equiv \log \prod_{t=1}^{T-1} p(\mathbf{v}(t+1)|\mathbf{v}(t)) = \sum_t \log p(\mathbf{v}(t+1)|\mathbf{v}(t)) = \sum_{t=1}^{T-1}\sum_{i=1}^{V} \log \sigma\left(v_i(t+1)a_i(t)\right) \tag{11}$$

To increase the likelihood of the sequence, we can use simple gradient ascent

$$w_{ij}^{new} = w_{ij} + \eta \frac{dL}{dw_{ij}}, \qquad \theta_i^{new} = \theta_i + \eta \frac{dL}{d\theta_i} \tag{12}$$

† Provided that the sequence is very long, and the equilibrium probability $p(\mathbf{v}(t = 1))$ is non-extreme, the two criteria become essentially equivalent.

‡ Restricting the weight matrix w to be symmetric enables the equilibrium distribution to be found, although this restriction severely limits the networks ability to store temporal sequences.

where

$$\frac{dL}{dw_{ij}} = \beta \sum_{t=1}^{T-1} \gamma_i(t) v_i(t+1) v_j(t), \qquad \frac{dL}{d\theta_i} = \beta \sum_{t=1}^{T-1} \gamma_i(t) v_i(t+1) \tag{13}$$

and we have defined

$$\gamma_i(t) \equiv 1 - \sigma\left(v_i(t+1) a_i(t)\right). \tag{14}$$

The learning rate η is chosen empirically to be sufficiently small to ensure convergence. The batch training procedure (13) can be readily converted to an online one since the updates only depend on two consecutive patterns so that an update can occur after the presentation of two consecutive patterns. The above learning rule can be seen as a modified Hebb learning rule, the basic Hebb rule being given when $\gamma_i(t) \equiv 1$. 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). The model is capable of storing a sequence of V linearly independent patterns†. To assess convergence of the learning rule, consider the Hessian of the log-likelihood‡:

$$\frac{d^2 L}{dw_{ij} dw_{kl}} = -\beta^2 \sum_{t=1}^{T-1} \left(v_i(t+1) v_j(t)\right) \gamma_i(t)(1-\gamma_i(t)) v_k(t+1) v_l(t) \delta_{ik}. \tag{15}$$

This is negative definite and hence the likelihood has a single global maximum. Gradient ascent is therefore guaranteed to converge to the optimal result, provided that the learning rate η is not too large. Since the gradient (13) is zero only for infinite weights, learning in principle never stops. However, the learning rapidly slows down after a small number of iterations, and learning can be safely stopped.

2.1.1. Relation to the Perceptron Rule In the limit that the activation is large, $\gamma_i = 1$ if $v_i(t+1)a_i < 0$ and is zero otherwise. In this limit, (16) is the well known Perceptron rule[5][3]. The Perceptron rule generalizes poorly unless we include a stability criterion such that we set $\gamma_i = 1$ if $v_i(t+1)a_i < M$ and is zero otherwise, were M is some positive threshold. An advantage of remaining with our probabilistic version is that we do not need to find an appropriate setting of the threshold. Additionally, we can assign a likelihood score to a novel temporal sequence, which can be used for classification purposes if we so wish. In the deterministic limit, the likelihood score for a novel sequence is either 1 or 0. Finally, the convergence of the stochastic model is guaranteed because the likelihood surface is convex.

† To see this, we can form an input-output training set for each neuron i, $\{(\mathbf{v}(t), v_i(t+1)), t = 1, \ldots, T-1\}$. Each neuron has an associated weight vector $\mathbf{w}^i \equiv w_{ij}, j = 1, \ldots, V$, which forms a logistic regressor or, in the limit $\beta = \infty$, a perceptron[5]. For perfect recall of the patterns, we need the training data to be linearly separable. This will be the case if the inputs are linearly independent, regardless of the outputs $v_i(t+1), t = 1, \ldots, T-1$.

‡ We neglect for expositional clarity the biases – this does not affect the conclusions.

2.1.2. Stochastic Interpretation By straightforward manipulations, (13) can be written as

$$\frac{dL}{dw_{ij}} = \sum_{t=1}^{T-1} \frac{1}{2} \left(1 - v_i(t+1) \underbrace{(2\sigma(a_i(t)) - 1)}_{\equiv \langle v_i(t+1) \rangle_{p(v_i(t+1)|a_i(t))}} \right) v_i(t+1) v_j(t) \tag{16}$$

$$= \sum_{t=1}^{T-1} \frac{1}{2} \left(v_i(t+1) - \langle v_i(t+1) \rangle_{p(v_i(t+1)|a_i(t))} \right) v_j(t) \tag{17}$$

The equivalence $\langle v_i(t+1) \rangle_{p(v_i(t+1)|a_i(t))} = 2\sigma(a_i(t)) - 1$ follows from

$$\langle v_i(t+1) \rangle_{p(v_i(t+1)|a_i(t))} = 1.p(v_i(t+1) = 1|a_i(t)) - 1.(1 - p(v_i(t+1) = 1|a_i(t))) \tag{18}$$

A stochastic, online learning rule is therefore

$$\Delta w_{ij}(t) = \eta \left(v_i(t+1) - \tilde{v}_i(t+1) \right) v_j(t) \tag{19}$$

where $\tilde{v}_i(t+1)$ is 1 with probability $\sigma(a_i(t))$, and -1 otherwise. This is the most biologically plausible interpretation of the learning rule. Provided that the learning rate η is small, this stochastic updating will approximate the learning rule (12,13). Whilst (19) 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 the Maxlimum Likelihood (ML) rule using (16), rather than (19).

2.2. Multiple Sequences

The previous section detailed how to train a Hopfield network for a single temporal sequence. 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. The gradient changes simply by introducing a loop to sum over the different sequences:

$$\frac{dL}{dw_{ij}} = \beta \sum_{s=1}^{S} \sum_{t=1}^{T-1} \gamma_i^s(t) v_i^s(t+1) v_j^s(t), \qquad \frac{dL}{d\theta_i} = \beta \sum_{s=1}^{S} \sum_{t=1}^{T-1} \gamma_i^s(t) v_i^s(t+1) \tag{20}$$

where

$$\gamma_i^s(t) \equiv 1 - \sigma\left(v_i^s(t+1) a_i^s(t) \right), \qquad a_i^s(t) = \theta_i + \sum_j w_{ij} v_j^s(t) \tag{21}$$

The log likelihood is convex since it is the sum of convex functions. This learning rule is capable of storing K patterns 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 that the patterns are linearly independent.

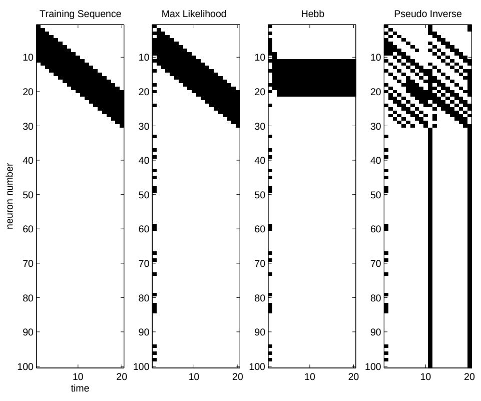

Figure 1. Left: The training sequence we desire to store. The other plots show the temporal evolution of the network after initialization in the correct starting state but corrupted with 30% noise. During recall, deterministic updates $\beta = \infty$ were used. The Maximum Likelihood rule was trained using 10 batch epochs with $\eta = 0.1$.

2.3. Illustration

As an illustration of the differences between some of the learning rule, in fig(1) we attempt to store a temporal sequence of length 20 states of 100 neurons using the three learning rules: Hebb, Maximum Likelihood and Pseudo Inverse. The sequence is chosen to be highly correlated, which makes the learning task generally more difficult. We set the thresholds to θ_i to zero throughout to facilitate comparison with other methods. The initial state of the training sequence, corrupted by 30% noise is presented to the trained networks, and we desire that the training sequence will be generated from this initial noisy state. Whilst the Hebb rule is operating in a feasible limit for uncorrelated patterns, the strong correlations in this training sequence renders very poor results. The PI rule is capable of storing a sequence of length 100, and is therefore operating well within its limits of storage. However, we see that the PI rule is not robust to perturbations from the correct initial state. The Maximum Likelihood rule performs well after a small amount of training. More extensive results are given in section (4).

Whilst the Maximum Likelihood rule promises wide basins of attraction, there is nothing explicit in the framework up to now that actively encouraged such good generalization behaviour. One of the reasons the network performs well is due to early stopping[1] (stopping updating after a limited number of weight updates). We show in the following section how early stopping can be theoretically motivated.

3. Improving stability by training with noise

An ideally trained network would be able to reproduce a sequence given a noisy initial state, or indeed, if at any stage during recall, the network state is perturbed by a small amount of noise. We will here try to build this desiderata into the learning rule explicitly. For expositional clarity, we will consider here a training sequence of just two time steps, $\mathbf{v}$ (at $t = 1$) and $\mathbf{v}'$ (at $t = 2$). The noise on the state $\mathbf{v}$ is given by $\boldsymbol{\eta}$ so that probability of making a transition from a noise corrupted version of $\mathbf{v}$ to $\mathbf{v}'$ is

$$p(\mathbf{v}'|\mathbf{v},\boldsymbol{\eta}) = \prod_{i=1}^{V} \sigma(v_i'\mathbf{w}_i^T[\mathbf{v}\circ\boldsymbol{\eta}]), \tag{22}$$

where $\mathbf{w}_i$ is the weight vector connecting the i^{th} neuron with all the other units. The vector $\mathbf{v}\circ\boldsymbol{\eta}$ has components $v_j\eta_j$, where $\eta_j = -1$ if the j^{th} bit of the training pattern is flipped to its reverse state. The noise components $\eta_1,\ldots,\eta_V$ are assumed to be independent with $q_j(\eta_j = -1) = \epsilon$.

We may hope to increase stability of a training pattern transition $\mathbf{v} \to \mathbf{v}'$ by maximizing the probability of transitions $\mathbf{v}\circ\boldsymbol{\eta} \to \mathbf{v}'$, for small noise corruptions ϵ. It is easy to see that for $L \overset{\text{def}}{=} \langle p(\mathbf{v}'|\mathbf{v}\circ\boldsymbol{\eta})\rangle_{q(\boldsymbol{\eta})}$ the gradient is expressed as (setting $\beta = 1$)

$$\begin{aligned}\frac{dL}{dw_{ij}} &= \left\langle \left(1-\sigma(v_i'\mathbf{w}_i^T[\mathbf{v}\circ\boldsymbol{\eta}])\right) v_i'v_j\eta_j\right\rangle_{q(\boldsymbol{\eta})} \\ &= v_i'v_j(1-2\epsilon) + \epsilon v_i'v_j\left\langle\sigma(v_i'\mathbf{w}_i^T[\mathbf{v}\circ\boldsymbol{\eta}]|_{\eta_j=-1})\right\rangle_{q(\boldsymbol{\eta}\backslash\eta_j)} \\ &\quad -(1-\epsilon)v_i'v_j\left\langle\sigma(v_i'\mathbf{w}_i^T[\mathbf{v}\circ\boldsymbol{\eta}]|_{\eta_j=1})\right\rangle_{q(\boldsymbol{\eta}\backslash\eta_j)} \\ &= v_i'v_j\left[1-2\epsilon+\epsilon\langle\sigma(c_{ij})\rangle_{q(\boldsymbol{\eta}\backslash\eta_j)} + (\epsilon-1)\langle\sigma(d_{ij})\rangle_{q(\boldsymbol{\eta}\backslash\eta_j)}\right],\end{aligned} \tag{23}$$

where

$$c_{ij} = v_i'\mathbf{w}_i^T[\mathbf{v}\circ\boldsymbol{\eta}]|_{\eta_j=-1} = v_i'\left[\sum_{k=1}^{V} w_{ik}v_k\eta_k - v_jw_{ij}(\eta_j+1)\right] \tag{24}$$

$$d_{ij} = v_i'\mathbf{w}_i^T[\mathbf{v}\circ\boldsymbol{\eta}]|_{\eta_j=1} = c_{ij} + 2v_i'w_{ij}v_j. \tag{25}$$

Notice that both c_{ij} and d_{ij} include summations of V random variables. Provided that the network is large and neither the weight components $\{w_{ik}\}$ nor the input patterns $\{v_k\}$ are strongly correlated, we can assume Gaussianity of the fields $c_{ij} \sim N_{ij}^{(c)}(\mu_{c_{ij}}, s_{c_{ij}}^2)$, $d_{ij} \sim N_{ij}^{(d)}(\mu_{d_{ij}}, s_{d_{ij}}^2)$ with the moments expressed as

$$\mu_{c_{ij}} = v_i'\left(\mathbf{w}_i^T\mathbf{v}(1-2\epsilon) - 2w_{ij}v_j(1-\epsilon)\right), \qquad \mu_{d_{ij}} = v_i'\left(\mathbf{w}_i^T\mathbf{v}(1-2\epsilon)+2w_{ij}v_j\epsilon\right)$$

$$s_{c_{ij}}^2 = s_{d_{ij}}^2 = 4v_i'^2\sum_{k=1,k\neq j}^{V} w_{ik}^2v_k^2\epsilon(1-\epsilon). \tag{27}$$

Substituting this approximation into expression (23) we obtain

$$\frac{dL}{dw_{ij}} \approx v_i'v_j\left[1-\langle\sigma(d_{ij})\rangle_{N_{ij}^{(d)}} + \epsilon\underbrace{\left(\langle\sigma(c_{ij})\rangle_{N_{ij}^{(c)}} + \langle\sigma(d_{ij})\rangle_{N_{ij}^{(d)}} - 2\right)}_{\leq 0}\right]. \tag{28}$$

A further simplification is given by replacing the averages $\langle\sigma(c_{ij})\rangle_{N_{ij}^{(c)}}$ by $\sigma(\mu_{c_{ij}})$ and $\langle\sigma(d_{ij})\rangle_{N_{ij}^{(d)}}$ by $\sigma(\mu_{d_{ij}})$, an approach we take in our experiments. Training with a small amount of noise (ϵ small) has a regularizing effect on the weight matrix due to the negative term in expression (28), which is the leading modification to the gradient. This provides some theoretical justification for early stopping since, if we begin training with the weights set to be very small, early stopping will bias the results to small weight matrices.

4. Results

We compared the performance of the Maximum Likelihood learning rule (with zero thresholds) with the standard Hebb, Pseudo Inverse, and Perceptron rule for learning a single temporal sequence. The network is initialized to a noise corrupted version of the correct initial state $\mathbf{v}(t = 1)$ from the training sequence. This corruption is achieved by flipping with a specified probability each bit of the correct initial state. The dynamics is then run (at $\beta = \infty$) for the same number of steps as the length of the training sequence, and the fraction of bits of the recalled final state which are the same as the training sequence final state $\mathbf{v}(T)$ is measured. 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 dynamics, the state of the network was corrupted with noise by flipping each neuron state with the specified flip probability. The results are displayed in figures fig(2). The fraction correct of the final state produced by the network with the final state in the training sequence is plotted for two different sequence lengths. Uncorrelated random temporal sequences are both less relevant and easier to train than temporally correlated sequences. For this reason we produced training sequences by starting from a random initial state, $\mathbf{v}(1)$, and then choosing at random 20% percent of the neurons to flip, each of the chosen neurons being flipped with probability 0.5. This thus produces a random training sequence with a high degree of temporal correlation.

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 Pseudo Inverse rule becomes unstable, especially for the longer temporal sequence which places more demands on the network. The non-monotonic behaviour in the performance of the Hebb and Perceptron rules is curious, but of minor consequence due to their relatively inferior performance. The perceptron rule can perform as well as the Maximum Likelihood rule, although its performance is critically dependent on an appropriate choice of the threshold M. The results for $M = 0$ Perceptron training are poor for small flip rates. An advantage of the Maximum Likelihood rule is that it performs well without the need for fine tuning of parameters. In all cases, batch training was used. In the Maximum Likelihood rule we used the exact values of $\gamma_i(t)$ for updating. Training with noise (using $\epsilon = 0.1$) has little effect on the performance here (although the weights learned are smaller), and is more relevant in the multiple sequences case (details to be published elsewhere).

5. Static Patterns

We briefly describe how the multiple sequences framework can be adapted to train stable static patterns, leaving a fuller account for elsewhere. A pattern $\mathbf{v}^\mu$ is stable if, under the dynamics,

$$p(\mathbf{v}(t+1) = \mathbf{v}^\mu | \mathbf{v}(t) = \mathbf{v}^\mu) \tag{29}$$

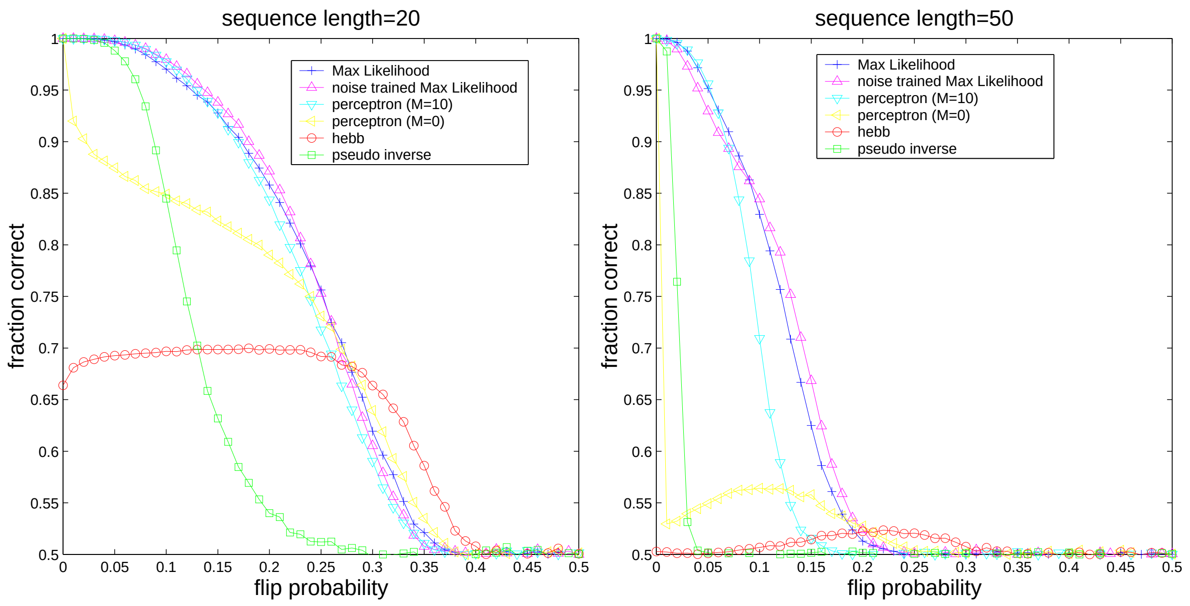

Figure 2. The fraction of neurons correct of the final state of the network produced by the trained network as a function of the flip rate for a 100 unit Hopfield network, after initialization in a noise flipped correct initial state. The correlated patterns were produced by flipping with probability 0.5, 20% of the previous state of the network. In both plots, 50 epochs of training were used. During recall, deterministic updates $\beta = \infty$ were used. For the perceptron rule, the threshold was set to 10 for both plots. Left: fraction correct for a sequence of length 20, $\eta = 0.05$. Right : fraction correct for a sequence of length 50, $\eta = 0.02$. In both plots, the results presented are averages over 5000 simulations, resulting in standard errors of the order of the symbol sizes.

is a delta function. In other words, we can attempt to maximize the stability of pattern $\mathbf{v}^\mu$ by maximizing the probability that it makes a transition to itself. The requirement that a set of patterns, $\mathbf{v}^\mu, \mu = 1, \dots P$ be stable incurs only a minor modification of the learning rules (13) to (setting $\beta = 1$)

$$\frac{dL}{dw_{ij}} = \sum_{\mu=1}^{P} \gamma_i^\mu v_i^\mu v_j^\mu \qquad\qquad \gamma_i^\mu \equiv 1 - \sigma\left(v_i^\mu \sum_{k=1}^{V} w_{ik} v_k^\mu\right) \tag{30}$$

A stochastic interpretation, similar to that given for the temporal case is straightforward:

$$\frac{dL}{dw_{ij}} = \sum_{\mu} \left(v_i^\mu - \langle v_i^\mu \rangle_{p(v_i|a_i^\mu)} \, v_j^\mu \right) \tag{31}$$

The rule (30) is similar to the rule by Diederich and Opper[3] derived for deterministic neurons. An advantage of our stochastic interpretation is that learning is readily seen to converge, and the derivation from a clear objective function is apparent, aiding the generalization to more complex models. For such short temporal patterns (only two time steps), the difference between the conditional $p(\mathbf{v}|\mathbf{v})$ and unconditional $p(\mathbf{v})$ likelihood can be very large. This is the origin of spurious attractor states since there is nothing in the unconditional likelihood objective function that prevents the network generating patterns not in the training set, given a random starting state. The Hebb rule for static patterns corresponds to setting γ_i^μ identically to 1, in which case the weight matrix learned is symmetric and a function of only second order statistics of the patterns. In contrast, the static Maximum Likelihood rule requires the whole pattern set. Additionally, the weight matrix will not be symmetric. However, in practice, the degree of asymmetry induced by the training patterns is typically negligible so that constraining the weight matrix to be symmetric has no deleterious effect on performance. Training with noise reduces dramatically the number of spurious attractor states, although it does not eliminate them completely.

6. Discussion

The classical Hebb rule can be much improved by a small modification that smooths the updates used. The resulting Maximum Likelihood rule is capable of robustly storing a temporal sequence with length the same number of neurons in the network. The current network can only recall perfectly unambiguous sequences. That is, sequences which for which a pattern maps to two or more different patterns cannot be recovered. However, this issue is straightforward to address using noisy sparse coding, and could be readily built in to the current framework. The statistical interpretation of conditional maximum likelihood training proves its worth in yielding a simple derivation of a learning rule that has guaranteed convergence, and is a useful starting point for more training more complex stochastic networks. The new rule can be interpreted as a stochastic version of the Perceptron learning rule, and the stochastic nature of the updates greatly helps the generalization performance of the network.

Acknowledgments

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

References

[1] C.M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[2] A.C.C. Coolen, *Statistical Mechanics of Neural Networks*, Lecture Notes, 1997.

[3] S. Diederich and M. Opper, *Learning of Correlated Patterns in Spin-Glass Networks by Local Learning Rules*, Physical Review Letters **58** (1986), no. 9, 949–952.

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

[5] J. Hertz, A. Krogh, and R. Palmer, *Introduction to the theory of neural computation.*, Addison-Wesley, 1991.

[6] J. Leo van Hemmen and R. Kühn, *Collective Phenomena in Neural Networks*, ch. 1, Springer, 1991.