# Distributed Computation

Prof. Dr. H. H. Takada

Quantitative Research – Itaú Asset Management
Institute of Mathematics and Statistics – University of São Paulo

## Hopfield Networks

- Hopfield networks are models of biological memory in which a pattern is represented by the activity of a set of $V$ interconnected neurons.
- At time $t$ neuron $i$ fires $v_i(t) = +1$ or is quiescent $v_i(t) = -1$ (not firing) depending on the states of the neurons at the preceding time $t-1$.
- Explicitly, neuron $i$ fires depending on the potential

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

  where $w_{ij}$ characterizes the efficacy with which neuron $j$ transmits a binary signal to neuron $i$. The threshold $\theta_i$ relates to the neuron's predisposition to firing.

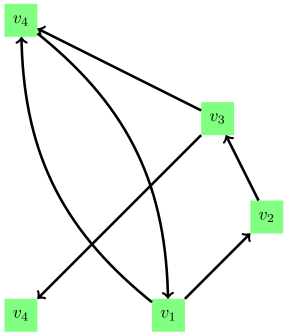


Figure: A depiction of a Hopfield network (for 5 neurons). The connectivity of the neurons is described by a weight matrix with elements $w_{ij}$. The graph represents a snapshot of the state of all neurons at time $t$ which simultaneously update as function of the network at the previous time $t-1$.

## Simultaneous updating

Writing the state of the network at time $t$ as $\mathbf{v}(t) \equiv (v_1(t), \ldots, v_V(t))^{\mathsf{T}}$, the probability that neuron $i$ fires at time $t+1$ is modelled as

$$p(v_i(t+1) = 1|\mathbf{v}(t)) = \sigma_\beta\left(a_i(t)\right)$$

where $\sigma_\beta(x) = 1/(1+e^{-\beta x})$ and $\beta$ controls the level of stochastic behaviour of the neuron.

---

### Deterministic limit

In the limit $\beta \to \infty$, the neuron updates deterministically

$$v_i(t+1) = \mathsf{sign}\left(a_i(t)\right)$$

## Bayesian network representation

In a synchronous Hopfield network all neurons update independently and simultaneously, and we can represent the temporal evolution of the neurons as a dynamic Bayes network,

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

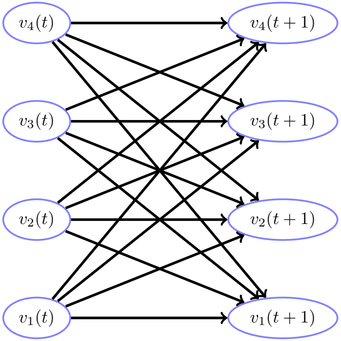


Figure: A Dynamic Bayesian Network representation of a Hopfield Network. The network operates by simultaneously generating a new set of neuron states according to a Markov transition matrix, modelling the transition probability $\mathbf{v}(t) \to \mathbf{v}(t+1)$ and furthermore imposes the constraint that the neurons are conditionally independent given the previous state of the network.

## Learning a sequence

- Given a sequence of network states, $\mathcal{V} = \{\mathbf{v}(1), \ldots, \mathbf{v}(T)\}$, we would like the network to store this sequence such that it can be later recalled under some cue.
- In particular, we will store the sequence in the weights and thresholds, and use the dynamics of the network to recall the patterns.
- That is, if the network is initialized in the correct starting state of the training sequence $\mathbf{v}(t = 1)$, the remainder of the training sequence for $t > 1$ should be reproduced under the deterministic dynamics, without error.
- This is a primitive analogue of process believed to be similar to memory recall and generalisation in natural organisms.

## The ‘classical’ correlation Hebb rule

$$w_{ij} = \frac{1}{V} \sum_{t=1}^{T-1} v_i(t+1) v_j(t)$$

The Hebb rule can be motivated mathematically by considering

$$\begin{aligned}\sum_j w_{ij} v_j(t) &= \frac{1}{V} \sum_{\tau=1}^{T-1} v_i(\tau+1) \sum_j v_j(\tau) v_j(t) \\ &= v_i(t+1) + \frac{1}{V} \sum_{\tau \neq t}^{T-1} v_i(\tau+1) \sum_j v_j(\tau) v_j(t)\end{aligned}$$

If the patterns are uncorrelated then the ‘interference’ term

$$\Omega \equiv \frac{1}{V} \sum_{\tau \neq t}^{T-1} v_i(\tau+1) \sum_j v_j(\tau) v_j(t)$$

will be relatively small and sign $\left(\sum_j w_{ij} v_j(t)\right) \approx v_i(t+1)$.

## The interference term

For randomly drawn patterns, the mean of $\Omega$ is zero, since the patterns are randomly $\pm 1$. The variance is therefore given by

$$\left\langle \Omega^2 \right\rangle = \frac{1}{V^2} \sum_{\tau,\tau' \neq t}^{T-1} \sum_{j,k} \left\langle v_i(\tau+1) v_i(\tau'+1) v_j(\tau) v_j(t) v_k(\tau') v_k(t) \right\rangle$$

For $j \neq k$, all the terms are independent and contribute zero on average. Therefore

$$\left\langle \Omega^2 \right\rangle = \frac{1}{V^2} \sum_{\tau,\tau'=1}^{T-1} \sum_{j} \left\langle v_i(\tau+1) v_i(\tau'+1) v_j(\tau) v_j(\tau') v_j^2(t) \right\rangle$$

When $\tau \neq \tau'$ all the terms are independent zero mean and contribute zero. Hence

$$\left\langle \Omega^2 \right\rangle = \frac{1}{V^2} \sum_{\tau \neq t} \sum_{j} \left\langle v_i^2(\tau+1) v_j^2(\tau) v_j^2(t) \right\rangle = \frac{T-1}{V}$$

Provided $V \gg T$ the average size of the interference will therefore be small. A careful analysis beyond our scope shows that the Hebb rule is capable of storing a random (uncorrelated) temporal sequence of length $0.269V$ time steps. However, the Hebb rule performs poorly for the case of correlated patterns since interference from the other patterns becomes significant.

## Pseudo inverse rule

The PI rule finds a matrix $[\mathbf{W}]_{ij} = w_{ij}$ that solves the linear equations

$$\sum_j w_{ij} v_j(t) = v_i(t+1), \qquad t = 1, \ldots, T-1$$

Under this condition $\text{sign}\left(\sum_j w_{ij} v_j(t)\right) = \text{sign}\left(v_i(t+1)\right) = v_i(t+1)$ so that patterns will be correctly recalled. In matrix notation we require

$$\mathbf{W}\mathbf{V} = \hat{\mathbf{V}}$$

where

$$[\mathbf{V}]_{it} = v_i(t), \quad t = 1, \ldots, T-1, \qquad \left[\hat{\mathbf{V}}\right]_{it} = v_i(t+1), \quad t = 2, \ldots, T$$

For $T < V$ the problem is under-determined so that multiple solutions exist. One solution is given by the pseudo inverse:

$$\mathbf{W} = \hat{\mathbf{V}} \left(\mathbf{V}^{\mathsf{T}}\mathbf{V}\right)^{-1} \mathbf{V}^{\mathsf{T}}$$

The Pseudo Inverse (PI) rule can store any sequence of $V$ linearly independent patterns. Whilst attractive compared to the standard Hebb rule in terms of its ability to store longer correlated sequences, this rule suffers from very small basins of attraction for temporally correlated patterns.

## The maximum likelihood Hebb rule

Given that we initialize the network in a state $\mathbf{v}(t=1)$, we wish to adjust the network parameters such that the probability

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

is maximal. The sequence log (conditional) likelihood is

$$L(\mathbf{w}, \theta) \equiv \sum_{t=1}^{T-1} \log p(\mathbf{v}(t+1) | \mathbf{v}(t)) = \sum_{t=1}^{T-1} \sum_{i=1}^{V} \log \sigma_\beta \left( v_i(t+1) a_i(t) \right)$$

Our task is then to find weights $\mathbf{w}$ and thresholds $\theta$ that maximise $L(\mathbf{w}, \theta)$. This corresponds to a straightforward computational problem since the log likelihood is a convex function.

## Gradient training

To increase the likelihood of the sequence, we can use a simple method such as gradient ascent

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

where

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

where

$$\gamma_i(t) \equiv 1 - \sigma_\beta \left( v_i(t+1) a_i(t) \right), \qquad a_i(t) = \theta_i + \sum_j w_{ij} v_j(t)$$

The learning rate $\eta$ is set small enough to ensure convergence. The Hebb rule is given when $\gamma_i(t) \equiv 1$. As learning progresses, the factors $\gamma_i(t)$ 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).

# Recalling Patterns

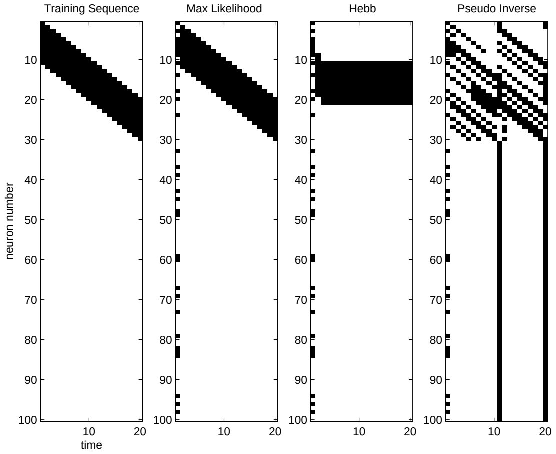


Figure: Leftmost panel: The temporally highly-correlated training sequence we desire to store. The other panels show the temporal evolution of the network after initialization in the correct starting state but corrupted with 30% noise. During recall, deterministic updates $\beta = \infty$ were used.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

## Storage capacity of the ML Hebb rule

- The ML Hebb rule is capable of storing a sequence of $V$ linearly independent patterns.
- To see this, we first 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.
- For perfect recall of the patterns, we therefore need only that the vectors constituting the pattern sequence be linearly separable. This will be the case if the patterns are linearly independent, regardless of the outputs $v_i(t+1), t = 1, \ldots, T-1$.

## Relation to the perceptron rule

In the limit that the activation is large, $|a_i| \gg 1$

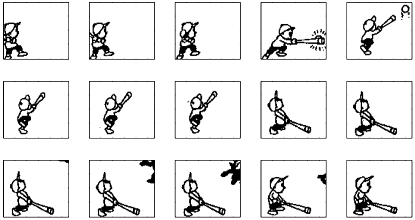
$$\gamma_i(t) \approx \begin{cases} 1 & v_i(t+1)a_i < 0 \\ 0 & v_i(t+1)a_i \geq 0 \end{cases}$$

Provided the activation and desired next output are the same sign, no update is made for neuron $i$. In this limit, (**??**) is called the perceptron rule. For an activation $a$ that is close to the decision boundary, a small change can lead to a different sign of the neural firing. To guard against this it is common to include a stability criterion

$$\gamma_i(t) = \begin{cases} 1 & v_i(t+1)a_i < M \\ 0 & v_i(t+1)a_i \geq M \end{cases}$$

where $M$ is an empirically chosen positive threshold.

## Recalling sequences under perpetual noise


(a)

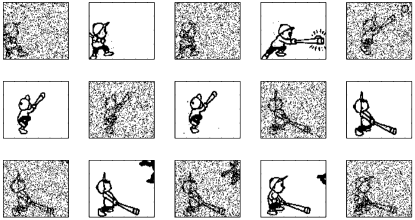

(b)

Figure: **(a)**: Original $T = 15$ binary video sequence on a set of $81 \times 111 = 8991$ neurons. **(b)**: The reconstructions beginning from a 20% noise perturbed initial state. Every odd time reconstruction is also randomly perturbed. Despite the high level of noise the basis of attraction of the pattern sequence is very broad and the patterns immediately fall back close to the pattern sequence even after a single timestep.

## Multiple sequences

We now address the learning a set of sequences $\{\mathcal{V}^n, n = 1, \ldots, N\}$. If we assume that the sequences are independent, the log likelihood of a set of sequences is the sum of the individual sequences. The gradient is given by

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

where

$$\gamma_i^n(t) \equiv 1 - \sigma_\beta \left( v_i^n(t+1) a_i^n(t) \right), \qquad a_i^n(t) = \theta_i + \sum_j w_{ij} v_j^n(t)$$

The log likelihood remains convex since it is the sum of convex functions, so that the standard gradient based learning algorithms can be used successfully here as well.

## Boolean networks

- The Hopfield network is one particular parameterisation of the table $p(v_i(t+1) = 1|\mathbf{v}(t))$. In the fully unconstrained case each neuron $i$ has an associated $2^V$ parental states.
- An interesting restriction is to consider that each neuron has only $K$ parents, so that each table contains $2^K$ entries. Learning the table parameters by Maximum Likelihood is straightforward since the log likelihood is a convex function of the table entries.
- Hence, for any given sequence (or set of sequences) one may readily find parameters that maximise the sequence reconstruction probability. The Maximum Likelihood method also produces large basins of attraction for the associated stochastic dynamical system. Such models are of potential interest in Artificial Life and Random Boolean networks in which emergent macroscopic behaviour appears from local update rules. Such systems are also used to study the robustness of chemical and gene regulatory networks.

## Sequence disambiguation

- A limitation of time-independent first order networks defined on visible variables alone (such as the Hopfield network) is that the observation transition $p(\mathbf{v}_{t+1}|\mathbf{v}_t = \mathrm{v})$ is the same every time the joint state v is encountered. This means that if the sequence contains a subsequence such as $\mathrm{a}, \mathrm{b}, \mathrm{a}, \mathrm{c}$ this cannot be recalled with high probability since a transitions to different states at different times.
- One could attempt to resolve this using a higher order Markov model to account for a longer temporal context, or by using a time-dependent model.
- Alternatively, latent variables can be used for sequence disambiguation. In the Hopfield model the recall capacity can be increased using latent variables by make a sequencing in the joint latent-visible space that is linearly independent, even if the visible variable sequence alone is not.