Neurodynamics

Aims	• to *skim* some maths relevant to neurodynamics • to study pattern completion (content-addressable memory) via - the Hopfield model - the Brain-State-in-a-Box (BSB) model
Reference	Ch. 13: Haykin, S. 3rd ed., *Neural Networks and Learning Machines* Ch. 14: Haykin, S. 2nd ed., *Neural Networks: a Comprehensive Foundation* *Equation numbers in these notes correspond to the third edition of Haykin.*
Keywords	dynamic, state space, trajectory, Lipschitz condition, dissipative, equilibrium point, stability, Lyapunov function, energy function, positive definite, attractor, limit cycle, basin of attraction, additive model of a neuron, Hopfield network, discrete Hopfield network, content-addressable memory, spurious states, storage capacity, BSB
Plan	• study dynamical systems & state spaces, trajectories, equilibrium states, equilibrium state stability, & Lyapunov's theorems on stability • study attractors • study the additive model of a neuron • Hopfield model and manipulation of attractors • discrete Hopfield model - storage & retrieval phases, spurious states • Brain-state-in-a-box
What's Examinable	Much of the maths covered is not examinable. Definitions 1-6 and Theorems 1-2, for example, are not examinable. The associated concepts – attractors, limit cycles, separatrix, *are* examinable. Formulae from 13.39 on are examinable.

Neurodynamics

Suppose that we have a network with recurrent connections (feedback, or cycles in the activation flow graph). If we supply input values to some or all of the neurons in this network, the activations will change with time as they are propagated around the network. Such a network is dynamic: its state varies with time. Feedback in electronic circuits can lead to instability - this could happen in neural circuits too, depending on the activation function (thresholding via a sigmoid squashing function obviously prevents activations from growing without bound).

Dynamical Systems

As we are studying change of activation of a neuron $x_j(t)$ with time t, we are interested in the time derivative dx_j/dt or its discrete-time-steps analogue. This will be a function of the activations of all N neurons (N is the *order* of the system)

$$dx_j/dt = F_j(x_1,...,x_N), \quad j = 1, 2, ..., N \tag{13.1}$$

where in general the functions F_j will be non-linear. More compactly,

$$d\mathbf{x}/dt = \mathbf{F}(\mathbf{x}) \tag{13.2}$$

where $\mathbf{x} = [x_1(t),...,x_N(t)]^T$, and $\mathbf{F}$ is now a vector function.

Given an initial state $\mathbf{x}_0$, we seek a solution to equation 13.2.

State Space

While $\mathbf{x}$ is a vector of neuron activations, not a point in space, it helps us visualize neurodynamics to pretend that it is - thus we say that $\mathbf{x}$ is a point in N-dimensional state space. Thus $\mathbf{x}(t_0)$ is the point in state space corresponding to the solution to equations 13.1 / 13.2 at time t_0. Through time, $\mathbf{x}(t)$, for $t > t_0$, describes a curve in state space, called the trajectory of the system.

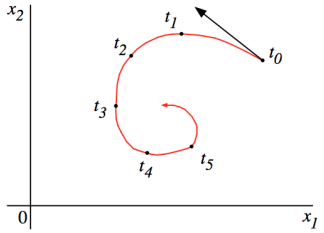


Figure 13.1: A trajectory of a two-dimensional dynamical system

Figures in these notes with numbers corresponding to those of Haykin are thereby acknowledged to have been re-drawn from Haykin

Vector Field

The instantaneous velocity of the system, $d\mathbf{x}/dt$, corresponds to the tangent vector to the trajectory. For each possible state, there will be a tangent vector. The assembly of tangents is described as a vector field, and is illustrated in Figure 13.3.

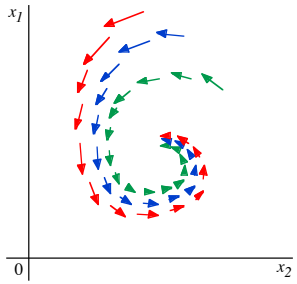


Figure 13.3: Vector field of a two-dimensional dynamical system

Lipschitz Condition

- We would like to be assured that the state-space equation (13.2) has a solution, and a unique solution.
- A solution exists if **F** is continuous.
- Uniqueness holds provided **F** satisfies a condition called the Lipschitz condition, stated in Haykin as follows: Let **x** and **u** be a pair of vectors in an open set ***M*** in state space. Then the Lipschitz condition holds if there exists a constant K such that $\|\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{u})\| \le K\|\mathbf{x} - \mathbf{u}\|$.

F is *continuous at* $\mathbf{x}_0$ if $\forall \varepsilon > 0 \exists \delta > 0 \;\; \|\mathbf{x} - \mathbf{x}_0\| < \delta \Rightarrow \|\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{x}_0)\| < \varepsilon$
F is *continuous* if it is continuous at every point $\mathbf{x}_0$ in state space.

Divergence Theorem

Consider a region of volume V and surface S in state space and assume a "flow" of points from this region. We know (eqn 13.2) that $d\mathbf{x}/dt = \mathbf{F}(\mathbf{x})$. Provided that $\mathbf{F}(\mathbf{x})$ is well-behaved, we may apply the divergence theorem from vector calculus: let **n** denote a unit vector normal to the surface and pointing outward from the enclosed volume. Then

$$\int_S (\mathbf{F}(\mathbf{x})\cdot\mathbf{n})dS = \int_V (\nabla\cdot\mathbf{F}(\mathbf{x}))dV \qquad (13.5)$$

The LHS is known as the net flux out of the region surrounded by S. If the net flux is zero, the system is called conservative. If it is negative, the system is called dissipative. It is sufficient for conservation (dissipation) that $\nabla\cdot\mathbf{F}(\mathbf{x})$ be zero (negative) within V.

Stability of equilibrium states

- Consider what happens in equation 13.2 if for some point $\mathbf{x}_0$ in state space, $\mathbf{F}(\mathbf{x}_0) = \mathbf{0}$.
- In this case, $d\mathbf{x}/dt|_{\mathbf{x}=\mathbf{x}_0} = \mathbf{0}$, and $\mathbf{x}_0$ is said to be an equilibrium point of the system, and $\mathbf{x}(t) = \mathbf{x}_0$ for all $t > 0$ is a solution of equation 13.2.
- Equilibrium points can be classified according to what happens to $\mathbf{x}(t)$ when it is close to $\mathbf{x}_0$.
- The behaviour depends on the eigenvalues of the Jacobian matrix of $\mathbf{F}(\mathbf{x})$ evaluated at $\mathbf{x} = \mathbf{x}_0$, i.e. $\partial/\partial\mathbf{x}(\mathbf{F}(\mathbf{x}))\big|_{\mathbf{x}=\mathbf{x}_0}$.
- An *eigenvector* of a square matrix A is a vector **v** such that $A\mathbf{v} = \lambda\mathbf{v}$ for some scalar constant λ. In other words, the linear transformation A maps **v** to a multiple of itself. In this case, λ is said to be an *eigenvalue* of A.
- In the two-dimensional case, if both eigenvalues are real and negative, then the equilibrium point is stable: any trajectory starting near $\mathbf{x}_0$ will converge to $\mathbf{x}_0$ in a finite amount of time.
- In other cases the solutions might asymptotically approach $\mathbf{x}_0$, or diverge from $\mathbf{x}_0$, or oscillate around $\mathbf{x}_0$. etc.

Definitions of Stability

Definition 1: The equilibrium state $\mathbf{x}'$ is said to be uniformly stable if
$\forall \varepsilon > 0 \; \exists \delta > 0 \; \|\mathbf{x}(0) - \mathbf{x}'\| < \delta \Rightarrow \; \forall t > 0 \; \|\mathbf{x}(t) - \mathbf{x}'\| < \varepsilon$.
That is, **x** stays close to $\mathbf{x}'$ if **x** starts close to $\mathbf{x}'$.

Definition 2: The equilibrium state $\mathbf{x}'$ is said to be convergent if
$\exists \delta > 0 \; \|\mathbf{x}(0) - \mathbf{x}'\| < \delta \Rightarrow \; \lim_{t\to\infty} \mathbf{x}(t) = \mathbf{x}'$.
That is, if **x** starts close enough to $\mathbf{x}'$ then **x** converges to $\mathbf{x}'$ as $t \to \infty$.

Definition 3: The equilibrium state $\mathbf{x}'$ is said to be asymptotically stable if it is both uniformly stable and convergent.

Definition 4: The equilibrium state $\mathbf{x}'$ is said to be globally asymptotically stable if it is both uniformly stable and all trajectories of the system converge to $\mathbf{x}'$ as $t \to \infty$.

In this case there will of course be only one equilibrium state.

Lyapunov's Theorems

Lyapunov found conditions under which stability and asymptotic stability can be proven, using a continuous scalar function of the state vector.

These conditions require us to know when a scalar function of $\mathbf{x}$ is positive definite.

DEFINITION 5: A function $V(\mathbf{x})$ is positive definite in a neighbourhood L of $\mathbf{x}'$ if $\forall \mathbf{x} \in$ L:
1. $V(\mathbf{x})$ has continuous partial derivatives with respect to the components of $\mathbf{x}$.
2. $V(\mathbf{x}') = 0$.
3. $V(\mathbf{x}) > 0$ if $\mathbf{x} \neq \mathbf{x}'$.

DEFINITION 6: A function $V(\mathbf{x})$ is positive semidefinite in a neighbourhood L of $\mathbf{x}'$ if $\forall \mathbf{x} \in$ L, conditions 1 and 2 of definition 5 hold and also
3'. $V(\mathbf{x}) \geq 0$ if $\mathbf{x} \neq \mathbf{x}'$.

The concepts negative definite and negative semidefinite are defined analogously.

THEOREM 1: The equilibrium state $\mathbf{x}'$ is stable if in a small neighbourhood of $\mathbf{x}'$ there exists a positive definite function $V(\mathbf{x})$ whose derivative with respect to time is negative semidefinite in that neighbourhood.

THEOREM 2: The equilibrium state $\mathbf{x}'$ is asymptotic stable if in a small neighbourhood of $\mathbf{x}'$ there exists a positive definite function $V(\mathbf{x})$ whose derivative with respect to time is negative definite in that neighbourhood.

Lyapunov Function

The function $V(\mathbf{x})$ is called a Lyapunov function for the equilibrium state $\mathbf{x}'$. Finding a Lyapunov function allows us to avoid solving the state-space equation (13.2). In many cases of interest, the "energy function" of the system serves as a Lyapunov function - otherwise it is a matter of trial and error to find one.

Attractors

An attractor is a bounded subset of state space to which non-trivial regions of initial conditions converge as time t increases. An attractor can be a single point (point attractor), a periodic trajectory (limit cycle), or a chaotic attractor (stays within a bounded region of space, but no predictable cyclic path). Each attractor is surrounded by a region called a basin of attraction - if the state starts in this region, then it will approach the attractor. The boundary separating one basin of attraction from another is called a separatrix. (Cf. Fig. 13.7 of Haykin).

Neurodynamic Models

Four useful properties of neurodynamic systems:

1. A large number of degrees of freedom (cf. human cortex with 10^{14} neuron-to-neuron connections).
2. Non-linearity. Linear systems have uninteresting properties from the point of view of neurodynamics.
3. Dissipation. This implies that the state converges to a region of lower dimension as time goes on.
4. Noise. In biological neurons, membrane noise is generated at synaptic junctions.

When N is large, neurodynamic models have properties 1-3, and so complicated attractor structures are possible. To be useful, these must be designable, or at least manipulable, say by a learning algorithm.

Additive Model of Neuron

Fig. 13.8 is a neuron model (electrical metaphor), with the w_{ji} expressed as conductances instead of resistances, I_j representing a bias/threshold, $v_j(t)$ the total net input to the nonlinearity, and R_j (C_j) the leakage resistance (capacitance).

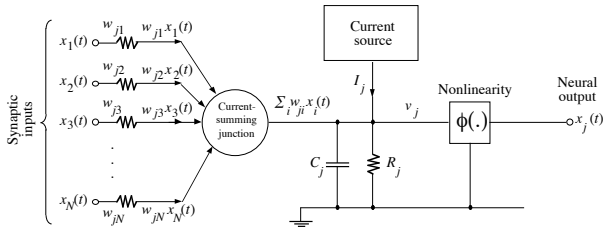


Figure 13.8 Additive model of neuron j

Equation for Additive Model of Neuron

The total current flowing away from the input of the nonlinearity is $C_j \frac{dv_j(t)}{dt} + \frac{v_j(t)}{R_j}$ while the total current flowing toward it is $\sum_{i=1}^{N} w_{ji} x_i(t) + I_j$, so by Kirchoff's law:

$$C_j \frac{dv_j(t)}{dt} + \frac{v_j(t)}{R_j} = \sum_{i=1}^{N} w_{ji} x_i(t) + I_j \tag{13.14}$$

Also obviously

$$x_k(t) = \phi(v_k(t)) \text{ for } k = 1, ..., N \tag{13.15}$$

This model is referred to a the additive model of a neuron. Eqn 13.14 can be rewritten as:

$$C_j \frac{dv_j(t)}{dt} = -\frac{v_j(t)}{R_j} + \sum_{i=1}^{N} w_{ji} x_i(t) + I_j \tag{13.16}$$

Common activation functions in this model are

$$\phi(v) = 1/(1 + \exp(-v)) \tag{13.17}$$

and related functions like $\phi(v) = \tanh(v)$ - recall/note that

$$1/(1 + \exp(-v)) = (\tanh(v/2) + 1)/2$$

The Hopfield Network

The Hopfield[1] net is an associative memory system. It consists of a set of neurons and a corresponding set of unit delays z^{-1} as illustrated in Fig. 13.9.

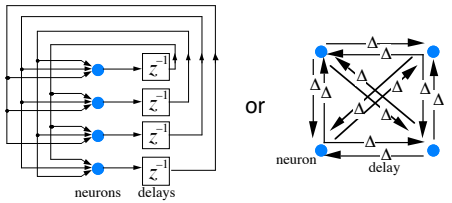


Figure 13.9 Hopfield net with 4 neurons - note no self-connections

[1] Hopfield nets were originally described in
Hopfield, J.J., (1982) Neural networks and physical systems with emergent collective computational abilities, *Proc. Nat'l Acad. Sci. USA* **79** 2554-2558
with the following quaint footnote on the first page:
The publication costs of this article were defrayed in part by a page charge payment. This article must therefore be hereby marked "*advertisement*" in accordance with 18 U.S.C. § 1734 solely to indicate this fact.

Stability of Hopfield Network

Equations 13.15 and 13.16 lead to:

$$C_j \frac{dv_j(t)}{dt} = -\frac{v_j(t)}{R_j} + \sum_{i=1}^{N} w_{ji} x_i(t) + I_j \tag{13.20}$$

We now make 3 assumptions:

1. The matrix of synaptic weights is symmetric: i.e. $w_{ij} = w_{ji}$;
2. Each neuron has a non-linear activation function (ϕ in 13.20);
3. The activation function ϕ is invertible.

We will use $x = \phi_i(v) = \tanh(a_i v/2)$.[2] a_i is called the gain of neuron i.
Note that $v = \phi_i^{-1}(x) = -(1/a_i)\log[(1-x)/(1+x)] = (1/a_i)\phi^{-1}(x)$, where ϕ is the nonlinearity of a neuron of gain 1.

[2] $\tanh(\theta) = (1 - \exp(-2\theta))/(1 + \exp(-2\theta))$

Stability of Hopfield Network 2

The energy function for the Hopfield net turns out to be:

$$E = -\tfrac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} x_i x_j + \sum_{j=1}^{N} \tfrac{1}{R} \int_0^{x_j} \phi_j^{-1}(x) dx - \sum_{j=1}^{N} I_j x_j \tag{13.28}$$

It can be shown that this meets the conditions of Lyapunov's theorem 1, and hence the model is stable, and in fact, $dE/dt < 0$ except at fixed points of the model, and hence the Hopfield model will settle into a fixed point from any initial condition.

Hopfield

John Hopfield started in Physics, but switched his research focus from physics to biology and earned an international reputation for his pioneering applications of physics-related computational techniques to the emerging field of neurobiology.

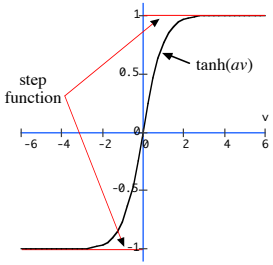
He has worked to develop a theoretical understanding of how the neural circuits of the brain perform complex calculations, investigating the way in which nerve cells work together to process sensory perceptions such as the recognition of odors or sounds. He developed the Hopfield model of neural processing.

Hopfield is at Princeton University.

Information from http://www.princeton.edu/main/news/archive/S12/37/07E74/index.xml - accessed 18 Sept 2007.

Continuous and Discrete Versions of the Hopfield Model

The continuous Hopfield model is based on the additive model of a neuron, as described above. The discrete Hopfield model is based on the McCulloch-Pitts model - that is, the nonlinearity is a step function, not the sigmoid function $\tanh(a_i v/2)$.



Using $\phi_j^{-1}(x) = (1/a_j)\phi^{-1}(x)$ and setting $I_j = 0$, equation 13.28 becomes:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}x_i x_j + \sum_{j=1}^{N}\frac{1}{a_j R_j}\int_0^{x_j}\phi^{-1}(x)dx \qquad (13.37)$$

Energy Equation for Discrete Hopfield Model

Noting that the discrete Hopfield model is the limit of the continuous model as each $a_j \to \infty$, we see that an energy function for the discrete model is

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}x_i x_j \qquad (13.39)$$

In effect, the discrete model inherits the nice properties of the continuous model.

Discrete Hopfield Model as Content-Addressable Memory

- The basic idea is that information can be stored by making it correspond to the stable states of a dynamical system. If one then starts the dynamical system in a state sufficiently close to one of the stable states, it should drift into one of these stable states and stay there.
- This can be interpreted as a kind of content-addressable memory. Everybody likes to quote Hopfield's paper here: I won't break the tradition:

  > Suppose that an item stored in memory is "H. A. Kramers & G.H. Wannier *Phys. Rev.* **60**, 252 (1941)." A general content-addressable memory would be capable of retrieving this entire memory item on the basis of sufficient partial information. The input "& Wannier, (1941)" might suffice. An ideal memory could deal with errors and retrieve this reference even from the input "Vannier[3], (1941)".

- So the aim is to map a set of memories $\boldsymbol{\xi}_\mu$ onto a set of fixed-points $\mathbf{x}_\mu$ of the dynamical system. This means that a dynamical system with known fixed points must be set up.

[3] Amusingly, the deliberate typo *Vannier* in Hopfield's paper was corrected to *Wannier* in all three editions of Haykin. Hopfield's point was that the content-addressable memory should be able to correct *Vannier* to *Wannier* and expand to the full reference.

Operational Features of the Discrete Hopfield Network

Each neuron has two states, determined by the level of the activation potential acting on it. Neuron i is on / firing if $x_i = +1$, and off / quiescent if $x_i = -1$. The state of the net is defined by the vector $\mathbf{x} = [x_1, x_2, ..., x_N]^T$.

A pair of neurons i and j are connected by synaptic weights w_{ji} - the net potential v_j acting on neuron j is given by:

$$v_j = \Sigma_{i=1..N} w_{ji} x_i - b_j \quad (b \text{ for bias})$$

Neuron j modifies its state s_j according to the rule

$$x_j = \text{sgn}(v_j) \quad \begin{cases} +1 & \text{if } v_j > 0 \\ -1 & \text{if } v_j < 0 \end{cases}$$

If $v_j = 0$, we leave x_j unchanged.

First we must set up the stable states of the net:

Hopfield Net - Storage Phase

Suppose that we wish to store a set of N-dimensional vectors denoted by $\{ \boldsymbol{\xi}_\mu | \mu = 1, ..., M \}$. We call these M vectors fundamental memories. Let $\xi_{\mu,i}$ denote the i-th element of $\boldsymbol{\xi}_\mu$. The weight w_{ji} is defined by an outer product rule:

$$w_{ji} = \frac{1}{N} \sum_{\mu=1}^{M} \xi_{\mu,j} \xi_{\mu,i} \tag{13.41}$$

(with $w_{ii} = 0$ - no self-feedback). If $\mathbf{W} = (w_{ji})$, we can write

$$\mathbf{W} = \frac{1}{N} \sum_{\mu=1}^{M} \boldsymbol{\xi}_\mu \boldsymbol{\xi}_\mu^T - \frac{M}{N} \mathbf{I} \tag{13.43}$$

Note that $\mathbf{W}$ is symmetric.

Hopfield Net - Retrieval Phase

- During this phase, an N-dimensional vector $\mathbf{x}$ called a probe, is imposed on the Hopfield net as its initial state. The components of $\mathbf{x}$ are all ± 1. Usually, it is an incomplete or noisy version of one of the fundamental memories. Processing then proceeds according to the following rule:
- Each neuron j of the network updates from time to time. The next neuron to update is chosen in a uniform random way. It updates by inspecting its current input activation and changing its state to ± 1 according to whether the activation is positive or negative (if exactly zero, the state does not change).
- Updates can be *synchronous* (think synchronized – all neurons update at once) or *asynchronous* – neurons take turns, at random, to update. The asynchronous update procedure is the one described in the previous dotpoint.
- This asynchronous serial updating procedure continues until there are no further changes, at which time the net is in a stable state $\mathbf{y}$ such that $\mathbf{y} = \text{sgn}(\mathbf{W}\mathbf{y} + \mathbf{b})$.
- Notice that if $\mathbf{b} = \mathbf{0}$, then for each stable state $\mathbf{y}$, $-\mathbf{y}$ is also a stable state, and that nothing stops a Hopfield net from having stable states other than those corresponding to fundamental memories (and their negations).

Stability of Fundamental Memory of Hopfield Net

(Case $\mathbf{b} = \mathbf{0}$, parallel update, M=1): Suppose that we start with the state = fundamental memory $\boldsymbol{\xi}=\boldsymbol{\xi}_1$. Then, in computing the update:

$$\begin{aligned} \mathbf{W}\boldsymbol{\xi} &= \left(\tfrac{1}{N}\boldsymbol{\xi}\boldsymbol{\xi}^T - \tfrac{1}{N}\mathbf{I}\right)\boldsymbol{\xi} \quad \text{since } M = 1 \\ &= \tfrac{1}{N}\left(\boldsymbol{\xi}\boldsymbol{\xi}^T\boldsymbol{\xi} - \boldsymbol{\xi}\right) \end{aligned}$$

and N times the j-th component of this will be

$$\begin{aligned} & \left(\sum_{i=1}^{N} \xi_j \xi_i \xi_i\right) - \xi_j \\ = & \left(\sum_{i=1}^{N} \xi_j\right) - \xi_j \quad \text{since } \xi_i \xi_i = 1 \text{ because } \xi_i = \pm 1 \\ = & (N-1)\xi_j \end{aligned}$$

so $\text{sgn}(\mathbf{W}\boldsymbol{\xi}) = \boldsymbol{\xi}$.

In the case of multiple fundamental memories, you get $\mathbf{W}\boldsymbol{\xi}_\mu = \boldsymbol{\xi}_\mu$ + noise, and normally the noise is sufficiently small that $\text{sgn}(\mathbf{W}\boldsymbol{\xi}_\mu) = \boldsymbol{\xi}_\mu$, so fundamental memories are stable in this case too.

Hopfield Net Example

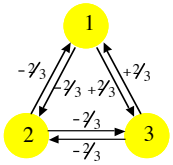
Example 14.2: Consider a Hopfield net as in Fig. 14.14(a). The weight matrix is symmetric, with zero diagonal, and satisfies:

$$3\mathbf{W} = \begin{matrix} 0 & -2 & 2 \\ -2 & 0 & -2 \\ 2 & -2 & 0 \end{matrix}$$

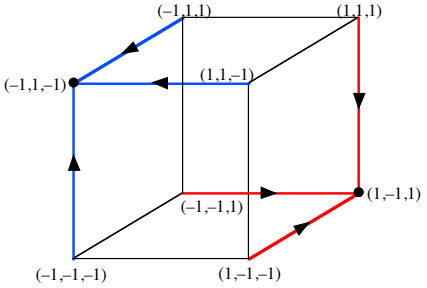
Three binary neurons means 8 states - illustrated in Fig. 13.14(b). One can calculate the next state for each possible transition, and this is also shown in Fig. 13.14(b). Two states are stable, namely $\pm [1, -1, 1]^T$. For $\mathbf{y} = [1, -1, 1]^T$

and so $$3\mathbf{W}\mathbf{y} = \begin{matrix} 0 & -2 & 2 \\ -2 & 0 & -2 \\ 2 & -2 & 0 \end{matrix} \bullet \begin{matrix} 1 \\ -1 \\ 1 \end{matrix} = \begin{matrix} 4 \\ -4 \\ 4 \end{matrix}$$

so sgn(**Wy**) = **y** - i.e. **y** is stable, and similarly for –**y**. [Obviously, then, eigenvectors of **W** corresponding to positive eigenvalues are stable.]



Hopfield net with 3 neurons



Two stable states and flow for Hopfield net example

Figure 13.14

In checking the state flow, recall that if net input is 0, no state change occurs. If the net is in state (1,1,1), say, then net input is, in fact, 0 if neuron 1 or 3 randomly and asynchronously updates. However, when neuron 2 updates, the net input is –2/3, so, after *sgn*-ing, neuron 2 changes state to –1, and the net has moved to one of its stable states.

State Flow Example Calculations

Let's start the net in state (–1, 1, 1). Recall that the weight matrix is

$$\mathbf{W} = \begin{matrix} 0 & -2/3 & 2/3 \\ -2/3 & 0 & -2/3 \\ 2/3 & -2/3 & 0 \end{matrix}$$

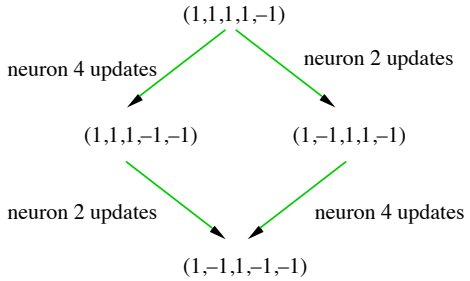
If neuron 1 updates, total net input is –1*0 + 1*(–2/3) + 1*(2/3) = 0, so no state change occurs.

With neuron 2, total net input is –1*(–2/3) + 1*0 + 1*(–2/3) = 0, so no state change occurs.

With neuron 3, total net input is –1*(2/3) + 1*(–2/3) + 1*0 = –4/3, so this neuron flips to –1. The new state is (–1, 1, –1).

State Flow Continued

In a more complex state space, more than one neuron might update. In this case, tracing the state flow might involve considering alternative paths:



Possible State Flow in Hypothetical Hopfield Net

Spurious Stable States

Hopfield nets can have spurious stable states - stable states other than the "programmed" ones, and there tend to be more as M increases. So what is the effective storage capacity of a Hopfield net?

Need for Stable States to be Dissimilar

In order for a Hopfield net to be able to reliably reconstruct stored patterns from partial patterns or from corrupted patterns, it is necessary for the patterns that are stored in the net to be reasonable different from each other – if for example

Storage Capacity of Hopfield Net

If we start a Hopfield net at one of its fundamental memories ξ_ν, then the total net input will be:

$$
\begin{aligned}
v_j &= \sum_i w_{ji}\xi_{\nu,i} \\
&= (1/N)\sum_\mu \xi_{\mu,j}\sum_i \xi_{\mu,i}\sum_i \xi_{\nu,i} \text{ using eqn 14.43} \\
&= \xi_{\nu,j} + (1/N)\sum_{\mu\neq\nu}\xi_{\mu,j}\sum_i \xi_{\mu,i}\xi_{\nu,i} \\
&= \text{signal} + \text{crosstalk}
\end{aligned}
$$

When crosstalk becomes large, the fundamental memory isn't stable. It can be shown that "storage capacity almost without errors" $M_{max} = N/\log_e N$ (see Haykin p.693ff.) This is not large, given the state space size of 2^N points, but not too bad, given that there are only N neurons in a Hopfield net.

N	$N/\log_e N$
100	21
1,000	144
10,000	1085
100,000	8685

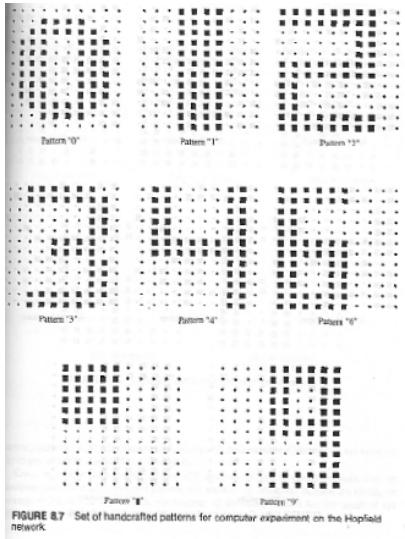


FIGURE 8.7 Set of handcrafted patterns for computer experiment on the Hopfield network.

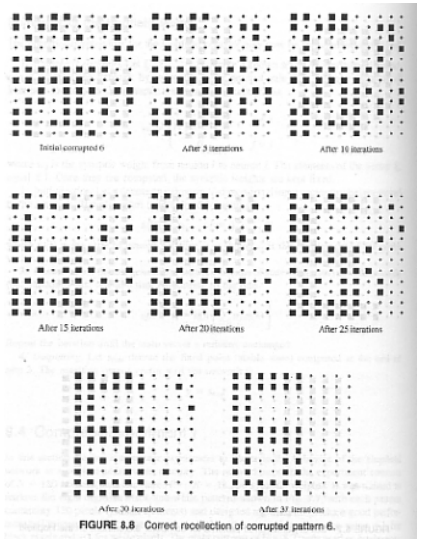


FIGURE 8.8 Correct recollection of corrupted pattern 6.

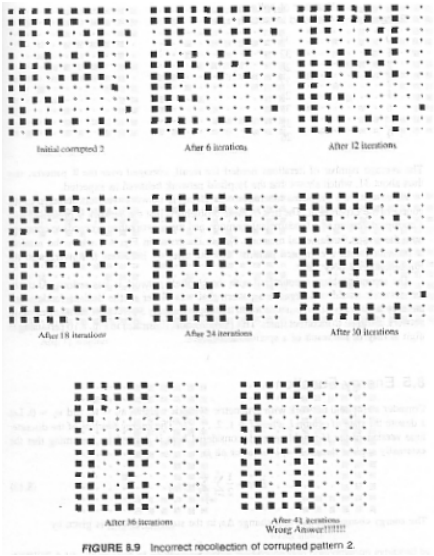


FIGURE 8.9 Incorrect recollection of corrupted pattern 2.

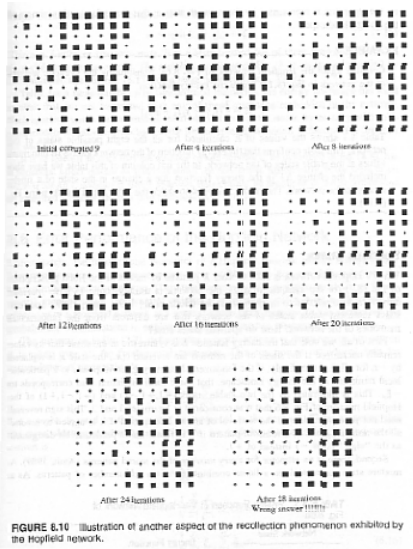


FIGURE 8.10 Illustration of another aspect of the recollection phenomenon exhibited by the Hopfield network.

Applications of Hopfield Nets

There used to be a website - `http://attrasoft.com/hopfield.html` - that described lots of Hopfield applications. While the company seems to have become less outgoing about its products, the list of application areas/links that it used to have gives you quite a good idea of the range of applicability of Hopfield nets. You could try the archived copy of this website at
http://web.archive.org/web/20060211215749/http://attrasoft.com/hopfield.html

1. What is a Hopfield neural net?
2. Hopfield neural network products
3. Hopfield neural net shareware
4. Hopfield neural net in stock market trend prediction
5. Hopfield neural net in character recognition
6. Hopfield neural net in Face recognition
7. Hopfield neural net in Internet Application
8. Hopfield neural net in Cancer Dectection
9. Hopfield neural net in Loan Application

on your favourite search engine.

Brain-State-in-Box (BSB) Model

This model is related to the Hopfield model. It was originally described by Anderson, Silverstein, Ritz, & Jones (1977) in Psychological Review **84** 413-451.

Reference in Haykin: 3rd ed. § 13.9, pages 705-711 **or** 2nd ed. § 14.10, pages 703-709. Equation numbers are taken from Haykin 3rd ed.

"Brain state" because Anderson is a cognitive psychologist, so that was what he had in mind to model. In reality, the "brain state" is a vector **x** of N activation values between –1 and 1. "Box" because the model is defined in such a way that the activation vector can never leave the N-dimensional unit hypercube.

BSB Model Definition

Let W be a symmetric weight matrix whose largest eigenvalues have positive real components. In addition, W is required to be positive semi-definite, i.e., $\mathbf{x}^T W \mathbf{x} \geq 0$ for all $\mathbf{x}$. Let $\mathbf{x}(0)$ denote the initial state of the activation vector $\mathbf{x}$.

The subsequent states of the BSB model are defined by equations 13.53 – 13.55:

$$\mathbf{y}(n) = \mathbf{x}(n) + \beta W \mathbf{x}(n) \qquad (13.53)$$

$$\mathbf{x}(n+1) = \phi(\mathbf{y}(n)) \qquad (13.54)$$

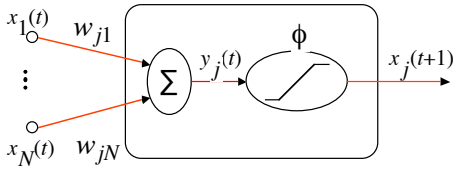
where β is a small positive constant called the feedback factor, and the squashing function ϕ is defined[4] by:

$$\phi(z) = \begin{cases} 1 & \text{if } z > 1 \\ z & \text{if } -1 \leq z \leq 1 \\ -1 & \text{if } z < -1 \end{cases} \qquad \text{(like 13.55)}$$

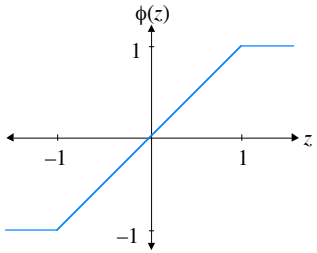
[4] for scalars – for vectors $\mathbf{z} = (z_1, z_2, \ldots, z_n)$, $\phi(\mathbf{z}) = (\phi(z_1), \phi(z_2), \ldots, \phi(z_n))$

BSB Model Diagram

Equation 14.63, which computes $\mathbf{y}(n)$, is like total net input. Equation 13.54, which computes $\mathbf{x}(n+1)$, squashes the "total net input" so it is forced back inside the box.

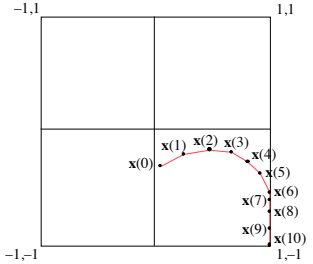


BSB Squashing Function



This function is continuous (unlike the linear threshold units used in the perceptron) but not differentiable (unlike the squashing functions used in continuous Hopfield model, and in backprop-trained feedforward networks - sometimes rather confusingly referred to as multilayer perceptrons).

A BSB Trajectory in the Unit 2-cube (= Square)



When W is has the required property, the effect of the algorithm is to drive the components of $\mathbf{x}$ towards +1 or 1. Thus continuous inputs are mapped to discrete binary outputs. The final states are of the form $(\pm 1, \pm 1, \ldots, \pm 1)$, i.e. corners of an N-dimensional hypercube.

The state vector $\mathbf{x}$ moves to the wall of the box, then "slides" along to a corner of the box. So some corners are attractors (i.e. there is a basin of attraction $\mathcal{N}(\mathbf{x})$ of $\mathbf{x}$, such that if $\mathbf{y} \in \mathcal{N}(\mathbf{x})$ then $\mathbf{y}$ will converge to $\mathbf{x}$). Note that not all corners need be attractors or even equilibrium points (an equilibrium point is one such that $\mathbf{x}(n+1) = \mathrm{x}(n)$).

The interior of the hypercube is partitioned into basins of attraction by the dynamics of the BSB. The details of the partition are determined by the matrix W and feedback constant β.

Energy Function for BSB

Here is the energy function for the BSB model

$$E = -\left(\frac{\beta}{2}\right)\sum_{ij} w_{ij} x_i x_j = -\left(\frac{\beta}{2}\right)\mathbf{x}^T W\mathbf{x}$$

It can be shown that the BSB dynamics reduces energy or at worst does not increase it.

Applications of the BSB model

One application for the BSB model is clustering. Anderson et al. (1990) *Proc of the IEEE* **78** 1646-1657 performed classification of radar signals according to the type of source using a model of this type. The matrix W has to be learned. The aim is to "show" the net a few examples of types of radar signals ("this is what an incoming Exocet missile looks like, this is what friendly submarines send out, … ").

BSB Learning Rules

Suppose that information is represented by a set of training vectors $\mathbf{x}_k$, $k = 1, 2, \ldots, K$, each of which is intended to be stable: $\mathbf{x}_k \rightarrow \mathbf{x}_k$.

- Select a training vector $\mathbf{x}_k$ at random
- modify the current weight matrix W by adding the outer product $\Delta W = \eta(\mathbf{x}_k - W\mathbf{x}_k)\mathbf{x}_k$.
- η is a learning rate parameter.
- Iterate a number of times, and over all of the training vectors
- one eventually obtains a matrix W such that for all k $W\mathbf{x}_k \approx \mathbf{x}_k$.
- [So far we have a linear associator, and the eigenvectors $\mathbf{x}_k$ are *not* corners of the unit hypercube.]

To perform radar clustering, the BSB model uses the linear associator W as the weight matrix, and uses the following slightly modified update computation:

$$\mathbf{x}(n+1) = \phi(\gamma\mathbf{x}(n) + \beta W\mathbf{x}(n) + \delta\mathbf{x}(0))$$

The differences:

- the decay constant γ ($0 < \gamma < 1$) causes the current state to decay slightly, with the aim of allowing errors to decay, eventually to zero;
- the third term $\delta\mathbf{x}(0)$ has the aim of keeping the initial state vector $\mathbf{x}(0)$ constantly present.

Summary	Neurodynamics/Hopfield Net/BSB

We have looked at:

- dynamical systems theory, which gives us a handle on the maths of neurodynamics
- continous Hopfield net → discrete Hopfield net
- spurious stable states of Hopfield net
- brain-state-in-a-box (BSB) model