Hopfield networks

Lluís A. Belanche
belanche@lsi.upc.edu

Soft Computing Research Group

Dept. de Llenguatges i Sistemes Informàtics (Software department)

Universitat Politècnica de Catalunya

2010-2011
Hopfield networks

Introduction

- Content-addressable autoassociative memory

- Deals with incomplete/erroneous keys: “Julia xxxxázar: Rxxxela”

- A mechanical system tends to minimum-energy states (e.g. a pole)

- Symmetric Hopfield networks have an analogous behaviour:
  
  low-energy states ⇔ attractors ⇔ memories
Hopfield networks

Introduction

A Hopfield network is a single-layer recurrent neural network, where the only layer has \( n \) neurons (as many as the input dimension)
Hopfield networks

Introduction

The data sample is a list of keys (aka patterns) $L = \{\xi_i \mid 1 \leq i \leq l\}$, with $\xi_i \in \{0, 1\}^n$ that we want to associate with themselves.

What is the meaning of autoassociativity? when a new pattern $\Psi$ is shown to the network, the output is the key in $L$ closest (in Hamming distance) to $\Psi$.

- Evidently, this can be achieved by direct programming; yet in hardware it is quite complex (and the Hopfield net does it quite distinctly)

- We therefore set up two goals:

  1. Content-addressable autoassociative memory (not easy)

  2. Tolerance to small errors, that will be corrected (rather hard)
Hopfield networks

Intuitive analysis

- The *state* of the network is the vector of activations \( e \) of the \( n \) neurons. The state space is \( \{0, 1\}^n \):

- When the network evolves, it “traverses” the hypercube
Hopfield networks

Formal analysis (I)

- For convenience, we change from \(\{0, 1\}\) to \(\{-1, +1\}\), by \(s = 2e - 1\).

- Dynamics of a neuron:

\[
s_i := \text{sgn} \left( \sum_{j=1}^{n} w_{ij}s_j - \theta_i \right), \quad \text{with} \quad \text{sgn}(z) = \begin{cases} +1 & : z \geq 0 \\ -1 & : z < 0 \end{cases}
\]

- Let us simplify the analysis by using \(\theta_i = 0\), and uncorrelated keys \(\xi_i\) (from a uniform distribution).
Hopfield networks

Formal analysis (II)

- How are neurons chosen to update their activations?
  1. Synchronous: all at once
  2. Asynchronous: there is a sequential order or a fair probability distribution

- The memories do not change; the end attractor (the recovered memory) does

- We will assume asynchronous updating

- When does the process stop?
  $\rightarrow$ When there are no more changes in the network state (we are in a stable state)
Hopfield networks

Formal analysis (III)

PROCEDURE:

1. Formula for setting the weights $w_{ij}$ to store the keys $\xi_i \in L$

2. Sanity check that the keys $\xi_i \in L$ are stable (they are autoassociated with themselves)

3. Check that small perturbations of the keys $\xi_i \in L$ are corrected (they are associated with the closest pattern in $L$)
Hopfield networks

Formal analysis (IV)

Case $l = 1$ (only one pattern $\xi$):

- Present $\xi$ to the net: $s_i := \xi_i$, $1 \leq i \leq n$

- Stability:

$$s_i := \text{sgn} \left( \sum_{j=1}^{n} w_{ij} s_j \right) = \text{sgn} \left( \sum_{j=1}^{n} w_{ij} \xi_j \right) = \ldots \text{ should be } \ldots = \xi_i$$

- One way to get this is: $w_{ij} = \alpha \xi_i \xi_j$, $\alpha > 0 \in \mathbb{R}$ (called Hebbian learning)

$$\text{sgn} \left( \sum_{j=1}^{n} w_{ij} \xi_j \right) = \text{sgn} \left( \sum_{j=1}^{n} \alpha \xi_i \xi_j \right) = \text{sgn} \left( \sum_{j=1}^{n} \alpha \xi_i \right) = \text{sgn} (\alpha n \xi_i) = \text{sgn}(\xi_i) = \xi_i$$
Hopfield networks

Formal analysis (V)

- Take $\alpha = 1/n$. This has a normalizing effect: $\|w_i\|^2 = 1/\sqrt{n}$

- Therefore $W_{n \times n} = (w_{ij}) = \frac{1}{n} \xi \times \xi$, i.e. $w_{ij} = \frac{1}{n} \xi_i \xi_j$

- If the perturbation consists in $b$ flipped bits, where $b \leq (n - 1) \div 2$, then the sign of $\sum_{j=1}^{n} w_{ij}s_j$ does not change!

  $\implies$ if the number of correct bits exceeds that of incorrect bits, the network is able to correct the error bits
Hopfield networks

Formal analysis (VI)

Case \( l \geq 1 \):

- Superposition: \( w_{ij} = \frac{1}{n} \sum_{k=1}^{l} \xi_k \xi_{kj} \) and \( W_{n\times n} = \frac{1}{n} \sum_{k=1}^{l} \xi_k \times \xi_k \)

- Defining \( E_{n\times l} = [\xi_1, \ldots, \xi_l] \), we get \( W_{n\times n} = \frac{1}{n} EE^t \)

- Note \( W_{n\times n} \) is a symmetric matrix
Hopfield networks

Formal analysis (VII)

Stability of a pattern $\xi_v \in L$:

- Initialize the net to $s_i := \xi_{vi}$, $1 \leq i \leq n$

- Stability: $s_i := sgn \left( \sum_{j=1}^{n} w_{ij} \xi_{vj} \right) = sgn(h_{vi})$

$$h_{vi} = \sum_{j=1}^{n} w_{ij} \xi_{vj} = \frac{1}{n} \sum_{j=1}^{n} \sum_{k=1}^{l} \xi_{ki} \xi_{kj} \xi_{vj} = \frac{1}{n} \sum_{j=1}^{n} \left[ \left( \sum_{k=1,k\neq v}^{l} \xi_{ki} \xi_{kj} \xi_{vj} \right) + \xi_{vi} \xi_{vj} \xi_{vj} \right]$$

$$= \xi_{vi} + \frac{1}{n} \sum_{j=1}^{n} \sum_{k=1,k\neq v}^{l} \xi_{ki} \xi_{kj} \xi_{vj} = \xi_{vi} + \text{crosstalk}(v,i)$$

- If $|\text{crosstalk}(v,i)| < 1$ then $sgn(h_{vi}) = sgn(\xi_{vi} + \text{crosstalk}(v,i)) = sgn(\xi_{vi}) = \xi_{vi}$
Hopfield networks

Formal analysis (and VIII)

- Analogously to the case $l = 1$, if the number of correct bits of a pattern $\xi_v \in L$ exceeds that of incorrect bits, the network is able to correct the bits in error

  (the pattern $\xi_v \in L$ is indeed a memory)

- When is $|\text{crosstalk}(v,i)| < 1$? If $l << n$ nothing is wrong, but as we have $l$ closer to $n$, the crosstalk term is quite strong compared to $\xi_v$: there is no stability

- The million euro question: given a network of $n$ neurons, what is the maximum value for $l$? What is the capacity of the network?
Hopfield networks

Capacity analysis

- For random uncorrelated patterns (for convenience):

\[ l_{\text{max}} = \begin{cases} \frac{n}{2\ln(n) + \ln \ln(n)} & : \text{perfect recall} \\ \approx 0.138n & : \text{small errors (} \approx 1.6\% \text{)} \end{cases} \]

- Realistic patterns will not be random, though some coding can make them more so

- An alternative setting is to have negatively correlated patterns:

\[ \xi_v \cdot \xi_u \leq 0, \ \forall v \neq u \]

This condition is equivalent to stating that the number of different bits is at least the number of equal bits
Hopfield networks

Spurious states

- When some pattern \( \xi \) is stored, the pattern \(-\xi\) is also stored. In consequence, all initial configurations of \( \xi \) where the majority of bits are incorrect will end up in \(-\xi\)!

  \( \rightarrow \) Hardly surprising: from the point of view of \(-\xi\), there are more correct bits than incorrect bits

- The network can be made more stable (less spurious states) if we set \( w_{ii} = 0 \) for all \( i \)

\[
W_{n \times n} = \frac{1}{n} \sum_{k=1}^{l} \xi_k \times \xi_k - \frac{l}{n} I_{n \times n} = \frac{1}{n} (EE^t - lI_{n \times n})
\]
Hopfield networks

Example

\[ \xi_1 = (+1 - 1 + 1), \xi_2 = (-1 + 1 - 1), \quad E_{3 \times 2} = \begin{pmatrix} +1 & -1 \\ -1 & +1 \\ +1 & -1 \end{pmatrix} \]

- \[ \begin{pmatrix} +1 \\ -1 \\ +1 \end{pmatrix} (+1 - 1 + 1) = \begin{pmatrix} +1 & -1 & +1 \\ -1 & +1 & -1 \\ +1 & -1 & +1 \end{pmatrix} \]
- \[ \begin{pmatrix} -1 \\ +1 \\ -1 \end{pmatrix} (-1 + 1 - 1) = \begin{pmatrix} +1 & -1 & +1 \\ -1 & +1 & -1 \\ +1 & -1 & +1 \end{pmatrix} \rightarrow \frac{1}{3} \begin{pmatrix} 0 & -2 & +2 \\ -2 & 0 & -2 \\ +2 & -2 & 0 \end{pmatrix} \]

\[ \begin{pmatrix} +1 + 1 + 1 \\ -1 - 1 + 1 \\ +1 - 1 - 1 \end{pmatrix} \rightarrow (+1 - 1 + 1), \quad \begin{pmatrix} +1 + 1 - 1 \\ -1 - 1 - 1 \\ -1 + 1 + 1 \end{pmatrix} \rightarrow (-1 + 1 - 1) \]

The network is able to correct 1 erroneous bit.
Hopfield networks

Practical applications (I)

Una red de 120 nodos (12x10)

CONJUNTO DE OCHO PATRONES

SECUENCIA DE SALIDA PARA LA ENTRADA "3" DISTORSIONADA

A Hopfield network that stores and recovers several digits
A Hopfield network that stores fighter aircraft shapes is able to reconstruct them. The network operates on the video images supplied by a camera on board a USAF bomber.
Hopfield networks

The energy function (I)

- If the weights are symmetric, there exists a so-called *energy function*:

\[
H(s) = - \sum_{i=1}^{n} \sum_{j>i}^{n} w_{ij} s_i s_j
\]

which is a function of the state of the system.

- The central property of \( H \) is that *it always decreases* (or remains constant) as the system evolves.

- Therefore the attractors (the stored memories) have to be the local minima of the energy function.
Hopfield networks

The energy function (II)

Proposition 1:

A change of state in one neuron $i$ yields a change $\Delta H < 0$

- Let $h_i = \sum_{j=1}^{n} w_{ij}s_j$. If neuron $i$ changes is because $s_i \neq \text{sgn}(h_i)$

- Therefore $\Delta H = -(s_i)h_i - (-s_ih_i) = 2s_ih_i < 0$

Proposition 2:

$$H(s) \geq -\sum_{i=1}^{n} \sum_{j=1}^{n} |w_{ij}|$$ (the energy function is lower-bounded)
Hopfield networks

The energy function (III)

The energy function $H$ is also useful to derive the proposed weights.

Consider the case of only one pattern $\xi$: we want $H$ to be minimum when the correlation between the current state $s$ and $\xi$ is maximum and equal to $\langle \xi, s \rangle^2 = \langle \xi, \xi \rangle^2 = \| \xi \|^2 = n$.

Therefore we choose $H(s) = -K \frac{1}{n} \langle \xi, s \rangle^2$, where $K > 0$ is a suitable constant.

For the many-pattern case, we can try to make each of the $\xi_i$ into local minima of $H$:

$$H(s) = -K \frac{1}{n} \sum_{v=1}^{n} \langle \xi_v, s \rangle^2 = -K \frac{1}{n} \sum_{v=1}^{n} \left( \sum_{i=1}^{n} \xi_{vi} s_i \right)^2 = -K \frac{1}{n} \sum_{v=1}^{n} \left( \sum_{i=1}^{n} \xi_{vi} s_i \right) \left( \sum_{j=1}^{n} \xi_{vj} s_j \right)$$

$$= -K \sum_{i=1}^{n} \sum_{j=1}^{n} \left( \frac{1}{n} \sum_{v=1}^{n} \xi_{vi} \xi_{vj} \right) s_i s_j = -K \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} s_i s_j = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j>i}^{n} w_{ij} s_i s_j$$
Hopfield networks

The energy function (and IV)

Exercises:

1. In the previous example,
   
   a) Check that $H(s) = \frac{2}{3}(s_1 s_2 - s_2 s_3 + s_1 s_3)$

   b) Show that $H$ decreases as the network evolves towards the stored patterns

2. Prove Proposition 2
Hopfield networks

Spurious states (continued)

- We have not shown that the explicitly stored memories are the only attractors of the system

- We have seen that the reversed patterns $-\xi$ are also stored (they have the same energy than $\xi$)

- There are stable mixture states, linear combinations of an odd number of patterns, e.g.:

$$\xi_{i}^{(mix)} = \text{sgn} \left( \sum_{k=1}^{\alpha} \pm \xi_{v_k,i} \right), \ 1 \leq i \leq n, \alpha \text{ odd}$$
Hopfield networks

Dynamics (continued)

Let us change the dynamics of a neuron by using an activation with memory:

\[ s_i(t + 1) = \begin{cases} 
+1 & \text{if } h_i(t + 1) > 0 \\
h_i(t) & \text{if } h_i(t + 1) = 0 \\n-1 & \text{if } h_i(t + 1) < 0 
\end{cases} \]

where \( h_i(t + 1) = \sum_{j=1}^{n} w_{ij}s_j(t) \)
Hopfield networks

Application to optimization problems

- Problem: find the optimum (or close to) of a scalar function subject to a set of restrictions

  e.g.: The traveling-salesman problem (TSP) or graph bipartitioning (GBP); often we deal with NP-complete problems.

- TSP\( (n) \) = tour \( n \) cities with the minimum total distance; TSP\( (60) \approx 6,93 \cdot 10^{79} \) possible valid tours. In general, TSP\( (n) = \frac{n!}{2^n} \).

- GBP\( (n) \) = given a graph with an even number \( n \) of nodes, partition it in two subgraphs of \( n/2 \) nodes each such that the number of crossing edges is minimum

- Idea: setup the problem as a function to be minimized, setting the weights so that the obtained function is the energy function of a Hopfield network
Hopfield networks

Application to graph bipartitioning

- **Representation:** \( n \) graph nodes = \( n \) neurons, with

  \[
  s_i = \begin{cases} 
  +1 : & \text{node } i \text{ in left part} \\
  -1 : & \text{node } i \text{ in right part}
  \end{cases}
  \]

- **Connectivity matrix:** \( c_{ij} = 1 \) indicates a connection between nodes \( i \) and \( j \).
Hopfield networks

Application to graph bipartitioning

\[ H(S) = - \sum_{i=1}^{n} \sum_{j>i}^{n} c_{ij} S_i S_j + \alpha \left( \sum_{i=1}^{n} S_i \right)^2 \]

connections between parts

\[ \text{bipartition} \]

\[ \implies H(S) = - \sum_{i=1}^{n} \sum_{j>i}^{n} (c_{ij} - \alpha) S_i S_j \equiv - \sum_{i=1}^{n} \sum_{j>i}^{n} w_{ij} S_i S_j + \sum_{i=1}^{n} \theta_i S_i \]

\[ \implies w_{ij} = c_{ij} - \alpha, \theta_i = 0 \]

(general idea: \( w_{ij} \) is the coefficient of \( S_i S_j \) in the energy function)
Hopfield networks

Application to the traveling-salesman problem

- Representation: 1 city $\rightarrow n$ neurons, coding for position; hence $n$ cities $= n^2$ neurons.

  $S_{n\times n} = (S_{ij})$ (network states); $D_{n\times n}$ (distances between cities, given)
  with $S_{ij} = +1$ if the city $i$ is visited in position $j$ and $-1$ otherwise.

- Restrictions on $S$:
  
  1. A city cannot be visited more than once $\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k>j}^{n} S_{ij}S_{ik} = 0$
     $
     \rightarrow$ every row $i$ must have only one $+1$: $R_1(S)$
  
  2. A city can only be visited in sequence (no ubiquity gift)
     $\rightarrow$ every column $j$ must have only one $+1$: $R_1(S')$

  3. Every city must be visited $\left(\sum_{i=1}^{n} \sum_{j=1}^{n} S_{ij} - n\right)^2 = 0$
     $\rightarrow$ there must be $n$ $+1$s overall: $R_2(S)$
Hopfield networks

Application to the traveling-salesman problem

4. The tour has to be the shortest possible

\[ \sum_{k=1}^{n} \sum_{i=1}^{n} \sum_{j>i}^{n} D_{ij} S_{ik} (S_{j,k-1} + S_{j,k+1}) \]

Total sum of distances in the tour: \( R_3(S, D) \)

\[
H(S) = - \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k>i}^{n} \sum_{l>j}^{n} w_{ij,kl} S_{ij} S_{kl} + \sum_{i=1}^{n} \sum_{j>i}^{n} \theta_{ij} S_{ij} \equiv \alpha R_1(S) + \beta R_1(S^t) + \gamma R_2(S) + \epsilon R_3(S, D)
\]

\[ \implies w_{ij,kl} = -\alpha \delta_{ik} (1 - \delta_{jl}) - \beta \delta_{jl} (1 - \delta_{ik}) - \gamma - \epsilon D_{ik} (\delta_{j,l+1} + \delta_{j,l-1}) \] and \( \theta_{ij} = -\gamma n \)

- This is a continuous Hopfield network: \( S_{ij} \in (-1, 1) \) since \( S_{ij} \equiv \tanh_\chi (h_{ij} - \theta_{ij}) \)

- Problem: set up good values for \( \alpha, \beta, \gamma, \epsilon > 0, \chi > 0 \)
Hopfield networks

Application to the traveling-salesman problem

Example for $n = 5$: 120 valid tours (12 different), 25 neurons

- $2^{25}$ potential attractors; actual number of them: 1,740 (among them, the valid 120). ⇒ $1,740 - 120 = 1,620$ spurious attractors (invalid tours)

- The valid tours correspond to the states of lower energy (correct design)

- But ... we have a network with 30 times more invalid tours than valid ones: a random initialization will end up in an invalid tour with probability 0.966

- Moreover, we aim for the best tour!

- A possible workaround is to use stochastic units with simulated annealing