The Hopfield Network

R. Rojas

Presented by
Madhumanti Ray

Outline

- Synchronous & Asynchronous networks
- Bidirectional Associative Memory
- Hopfield network
- Hebbian learning and Hopfield network
- Perceptron learning and Hopfield network
- Application of Hopfield network to combinatorial problems
- Limitations of Hopfield network in NP-complete problems
- Implementation of Hopfield network
- Closing comments

Synchronous networks

requires the computing elements to evaluate the inputs and compute the outputs simultaneously.

excitation of output units calculated at each node by evaluating sign function.

Asynchronous networks

selects an unit randomly to compute excitation.

changes state to 1 or -1 depending on < or > threshold value

independent of other units

Recurrent Associative networks

output of network fed back to input units using additional feedback connections

Bidirectional Associative Memory (BAM)

Synchronous network

Modification of recurrent associative network

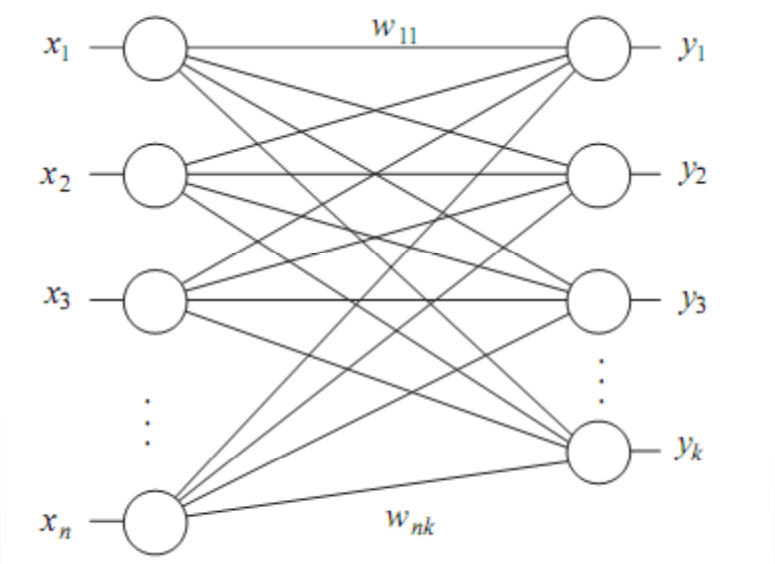
two layers - input and output

input layers send their computations to output layer using *bidirectional edges*. Final output of output layer sent back using

using same edges.

Activation function : sign function

Encoding: Bipolar



map n- dimensional row vector x0 to k-dimensional row vector y0

Weight matrix **W** (n x k matrix)

$$y_i = sgn(x_i \, \boldsymbol{W})$$

Feedback

$$x_{i+1}{}^{T} = \mathrm{sgn}(\mathbf{W} y_i{}^{T})$$

After some iterations a *fixedpoint* (x,y) is found for which there is no change in both:

$$y = \mathrm{sgn}(x\mathbf{W}) \quad \text{and} \quad x^{T} = \mathrm{sgn}(\mathbf{W}y^{T})$$

For a given vector pair (x,y) which is a fixed point, used Hebbian learning to compute matrix $\mathbf{W} = x^T y$

$$y = \text{sgn}(xW) = \text{sgn}(xx^T y) = \text{sgn}(||x||^2 y) = y \qquad \text{and,}$$

$$x^T = \text{sgn}(Wy^T) = \text{sgn}(x^T y\, y^T) = \text{sgn}\,(x^T ||y||^2) = x^T$$

Therefore, for *m* vectors: $\mathbf{W} = x_1^T y_1 + x_2^T y_2 + \ldots + x_m^T y_m$

BAM can be used to represent *autoassociative networks*, where each unit has a self feedback loop:

$$X = XW \qquad \text{and} \qquad X^T = W X^T$$

Where, X is a matrix with *m* rows representing n-dimensional vectors

Energy function of BAM

For the first vector *x0* presented to the network with weight matrix W:

$$y0 = \text{sgn}(x0W)$$

During feedback, excitation of input layer: $\mathbf{e^T = Wy_0}$

(x0,y0) is a stable state of the network if:

$$\mathbf{sgn(e) = x_0}$$

Vectors *e* closer to x_0 will give a smaller value for : $\mathbf{-x_0 e^T}$

which is used in the energy function E of the network:

$$E = -x_0 e^T = - x_0 W y_0^T$$

Therefore,

$$\boldsymbol{E(x_i,y_i) = -1/2 x_i\, W y_i^T}$$

where, $\mathbf{y_i}$ output of right layer of units in i-th iteration and x_i output of left layer

For units with a threshold and the step function as the activation function,

we update the input vectors with an additional component

so now vector *x* will be (x1,x2,….xn,1)

vector *y* is similarly updated

weight matrix W is updated to W' with additional row and column

row *n+1* has the negative thresholds of the units in the right layer

column *k+1* has those of units in left layer

The energy function will, therefore, be modified for extended network to:

$$E(x_i,y_i) = -\tfrac{1}{2}\, x_i\, W y_i^T + \tfrac{1}{2}\, \theta_r y_i^T + \tfrac{1}{2}\, \theta_l^T x_i$$

where θ_l represents row vector of thresholds of k units in left layer and θ_r gives thresholds of n units in right layer

Using energy function of BAM, prove network converges to stable state after some iterations using synchronous or asynchronous updates

For $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n)$

Weight matrix $(n \times k)$ $W = \{w_{ij}\}$

$$E(\mathbf{x}, \mathbf{y}) = -\frac{1}{2}(x_1, x_2, \ldots, x_n) \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1k} \\ w_{21} & w_{22} & \cdots & w_{2k} \\ \vdots & & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nk} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{pmatrix}.$$

Multiplying W_i by y^T, get excitation for unit i in left layer. Denote by $g_1, g_2, \ldots, g_n$

$$E(\mathbf{x}, \mathbf{y}) = -\frac{1}{2}(x_1, x_2, \ldots, x_n) \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{pmatrix}.$$

Similary $W_i \cdot x \rightarrow$ excitation of unit *i* in right layer. Denote by $e_1, e_2, \ldots, e_k$

$$E(\mathbf{x}, \mathbf{y}) = -\frac{1}{2}(e_1, e_2, \ldots, e_k) \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{pmatrix}.$$

Equivalent form:

$$E(\mathbf{x},\mathbf{y}) = -\frac{1}{2}\sum_{i=1}^{k} e_i y_i \quad \text{and} \quad E(\mathbf{x},\mathbf{y}) = -\frac{1}{2}\sum_{i=1}^{n} g_i x_i.$$

State of unit changes when excitation g_i or e_i have different sign than x_i.
updated to x_i to x_i'

$$E(\mathbf{x},\mathbf{y}) - E(\mathbf{x}',\mathbf{y}) = -\frac{1}{2} g_i (x_i - x_i').$$

x_i & x_i' have different signs than g_i.

$$E(\mathbf{x},\mathbf{y}) - E(\mathbf{x}',\mathbf{y}) > 0$$

Updating network state ⟶ reduces total energy

Reach state (**a,b**) s.t energy cannot be reduced further
local minimum of energy function
state (a,b) is *attractor* of the system

Hopfield Network

Proposed by American physicist John Hopfield in 1982

asynchronous recurrent neural network

special case of BAM although precedes it

Architecture:

consists of n totally coupled units. No self feed back .

symmetric because weights, $w_{ij} = w_{ji}$ i.e. single bidirectional connection

Features:

$n \times n$ symmetric weight matrix W with zero diagonal

necessary for the network to converge to stable states (applies to all asynchronous networks)

threshold θ set to a value $\neq 0$, if $e < \theta$ then state is -1 otherwise $e > 0$ state is 1

Hopfield network can be considered a asynchronous recurrent network of perceptrons

the activation function of units ≈ units of perceptrons

Therefore, the energy function of state x of a Hopfield network is given by:

$$E(x) = -\frac{1}{2} x W x^T + \theta x^T$$

the θ_r and θ_l is replaced by θ since, $\theta = \theta_r = \theta_l$

The energy function can be further expanded into the following form:

$$E(x) = -\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{n} w_{ij} x_i x_j + \sum_{i=1}^{n} \theta_i x_i$$

A Hopfield Network always finds a local minimum of the energy function.

The Flip-Flop example

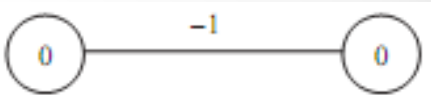


Fig. 13.4. A flip-flop

Network two units with threshold zero
only two stable states(1,-1) and (-1,1)

One unit forces the other to change states to stabilize the network
Assume complementary logic values

Energy function with weights, $w_{12} = w_{21} = x_1\, x_2 = 1$

$$E(x_1\, x_2) = x_1\, x_2$$

Flip flop in examples ahead…

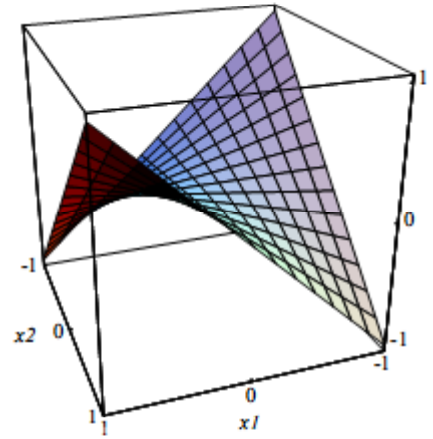


Fig. 13.5. Energy function of a flip-flop

Ising Models and Hopfield Network

Ising model of Magnetism:

The model consists of discrete variables called *spins* that can be in one of two states – ½ (up), - ½ (down). The spins are arranged in a lattice or graph, and each spin interacts only with its nearest neighbors.

Total magnetic field sensed by an atom i : $h_i = \sum_{j=1}^{n} w_{ij} x_j + h^*$

h^* - external field

w_{ij} – magnetic coupling between atoms i and j

Hence, potential energy E of a state of in Ising model derived from above:

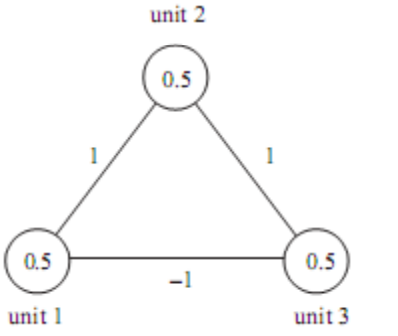
$$E = -\frac{1}{2} \sum_{i,j}^{n} w_{ij} x_i x_j + \sum_{i}^{n} - h^* x_i$$

**this function (at temperature zero) is *isomorphic* to the energy function of a Hopfield network

Hopfield Network Dynamics

Analyze with following example:

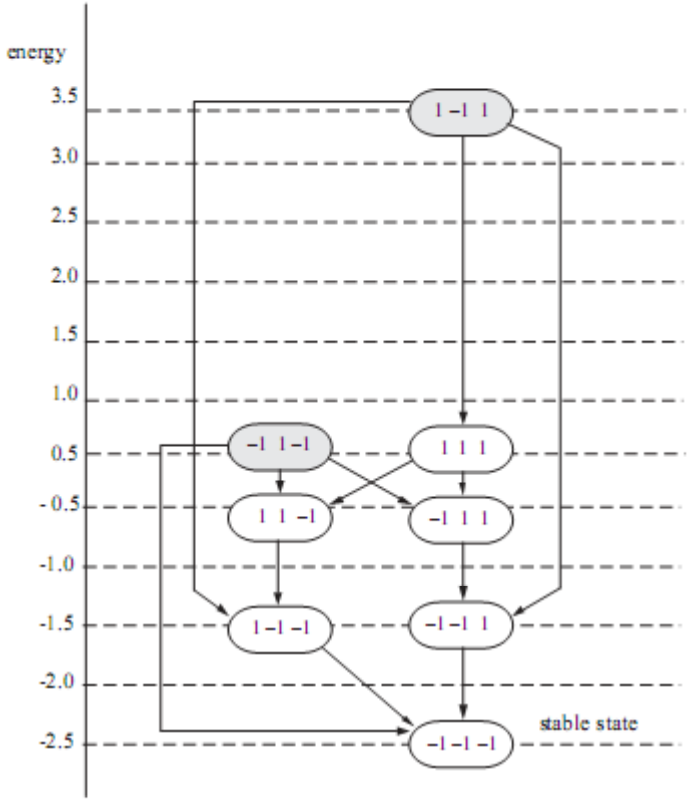
Given a network with three units, each has arbitrary weights and thresholds.



Observations:

- each transition has same probability = 1/3
- State (1,-1,1) & (-1,1,-1) very unstable, Pr of leaving is 1
- Single stable state (-1,-1,-1)

Hopfield network eventually finds a local minimum & state where network cannot change anymore.



Convergence of Hopfield Network

Proposition 20: *A Hopfield network with n units and asynchronous dynamics, which starts from any given network state, eventually reaches a stable state at a local minimum of the energy function.*

Proof: We know energy function for state x : $E(x) = -\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{n} w_{ij}\, xi\, xj + \sum_{i=1}^{n} \theta_i x_i$

If state of an unit changes, a new global state $x' = (x_1, \ldots, x_k', \ldots x_n)$

The difference of E(x) and E(x′) is then given as:

$$E(x) - E(x') = (- \sum_{j=1}^{n} w_{kj}\, x_k\, x_j + \theta_k\, x_k) - (- w_{kj}\, x'_k\, x_j + \theta_k x'_k)$$

since $w_{kk} = 0$,

$$E(x) - E(x') = - (x_k - x'_k) \sum_{j=1}^{n} w_{kj}\, x_j + \theta_k (x_k - x'_k)$$

$$= - (x_k - x'_k) \sum_{j=1}^{n} w_{kj}\, xj - \theta_k \qquad = - (x_k - x'_k)\, e_k$$

For state to change, e_k has to have different sign than x_k and x'_k , so above equation is +ve.

therefore, $E(x) - E(x') > 0$ => total energy of network reduces every time an units state changes.

The other proof:

- Classify units into 2 sets, one containing units with state 1 & other, units with state -1
- Units are fully connected among themselves
- Randomly select an unit and compute its "attraction" by units in its sets and units in second set. "attraction" = sum of weights of all edges between an unit and other units in the sets.
- If attraction from other set is higher, unit changes state and side. Else, stays in the same set.
- Repeat several times. Sum of weights of edges reduces over time.
- A global stable state is reached where attraction cannot be further reduced, since network is finite.
- *Minimal Cut Problem*

Hebbian Learning

Hopfield network can be used as associative memory – "imprint" m different stable states.

Variation of BAM – which uses Hebbian learning – could use in Hopfield network.

Implemented by loading *m* n-dimensional stable states on the network,

weights update according to ,

$$w_{ij} \quad w_{ij} + x_i^k \, x_j^k \qquad i,j = 1,\ldots n \;\&\; i \neq j$$

Weight matrix W with zero diagonal & symmetric , $W_1 = x_1^T \, x_1 - I_1$

where I is a *n*x*n* identity matrix.

Subtracting it guarantees that diagonal of W becomes zero, since for any bipolar vector $x_k^i \, x_k^i = 1$.

Hebbian Learning

Then Hopfield energy function becomes:

$$E(x) = -\tfrac{1}{2}\, x W_1 x_1^T = -\tfrac{1}{2}\,(x\, x_1^T x_1 x^T - x\, x^T)$$

where $x\, x^T = n$ for bipolar vectors, $E(x) = -\tfrac{1}{2}\,||\, x\, x_1^T\,||^2 + n/2$

Solving gives,

$$E(x) = -n^2/2 + n/2$$

which implies the function has local minimum at $x = x_1$, is a stable state.

For *m* different vectors, weight matrix W would be:

$$W = x_1^T x_1 + x_2^T x_2 + \ldots + x_m^T x_m - mI$$

If the network is initialized with vector x_1, excitation vector **e** of units becomes

$$e = x_1 W = x_1 x_1^T x_1 + x_1 x_2^T x_2 + \ldots + x_{m1} x_m^T x_m +- m\, x_1 I$$

$$= (n-m)\, x_1 + \sum_{j=2}^{m} \alpha_{1j}\, x_j$$

α represents scalar product of first vector with other m-1 vectors.

2nd term in *perturbation term* – which should small.

Observations for Hebbian learning on Hopfield network:

for m < n and small perturbation term, sgn(e) = sgn (x_i)

best results achieved when vectors x_1, x_2,…x_m are orthogonal or almost orthogonal

useful for computing weight matrix but sometimes cannot find a
a W for which *m* given vectors are not stable states
if the vectors lie near each other, perturbation term can grow very large and not give a solution

Perceptron learning is used…

Perceptron learning & Hopfield network

Transform a Hopfield network learning problem into Perceptron learning problem.

HN n units (perceptrons), threshold $\neq 0$, step function,

state = 1 if $e - \theta > 0$

state = -1 if $e - \theta < 0$

… perceptron learning could be used for finding W and thresholds

We have a Hopfield net, $W = \{w_{ij}\}$ and threshold θ

A vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ to be imprinted on a HN will be a stable state ← does not change network global state:

(e – θ) same sign as current state

First set of n inequalities:

for unit 1: $\text{sgn}(x_1)\ (0 + x_2w_{12} + x_3w_{13} + \ldots + x_nw_{1n} - \theta_1) < 0$

for unit 2: $\text{sgn}(x_2)\ (x_1w_{21} + 0 + x_3w_{23} + \ldots + x_nw_{2n} - \theta_2) < 0$

…

for unit n : $\text{sgn}(x_n)\ (x_1w_{n1} + x_2w_{n2} + x_3w_{n3} + \ldots + x_{n-1}w_{n(n-1)} + 0 - \theta_2) < 0$ **must hold**

Factor $\text{sgn}(x_i)$ used to obtain same inequality operator – '<'

Perceptron learning in Hopfield network

In previous set of inequalities, $n(n-1)/2$ entries of W & n thresholds

Define a $n+(n-1)/2$ vector **v**

components: non-diagonal entries of W ($i < j$), n thresholds with '-' sign

$$\mathbf{v} = (\underbrace{w_{12}, w_{13}, \ldots, w_{1n}}_{n-1}, \underbrace{w_{23}, w_{24}, \ldots, w_{2n}}_{n-2}, \ldots, \underbrace{w_{(n-1)n}}_{1}, \underbrace{-\theta_1, \ldots, -\theta_n}_{n})$$

Then the vector **x** is transformed into $n+(n-1)/2$ *dim* auxiliary vector **z :**

$$z_1 = (\underbrace{x_2, x_3, \ldots, x_n}_{n-1}, 0, 0, \ldots, \underbrace{1}_{n}, 0, \ldots, 0)$$

$$z_2 = (x_1, \underbrace{0, \ldots, 0}_{n-1}, \underbrace{x_3, \ldots, x_n}_{n-2}, \underbrace{0,0,\ldots,0,1}_{n}, \ldots, 0)$$

…

$$z_n = (0,0, \underbrace{x_1}_{n-1}, \underbrace{0, 0, \ldots, x_2}_{n-2}, \underbrace{0,0,\ldots,0,0,\ldots,1}_{n})$$

Perceptron learning in Hopfield network

So the first set of inequalities become:

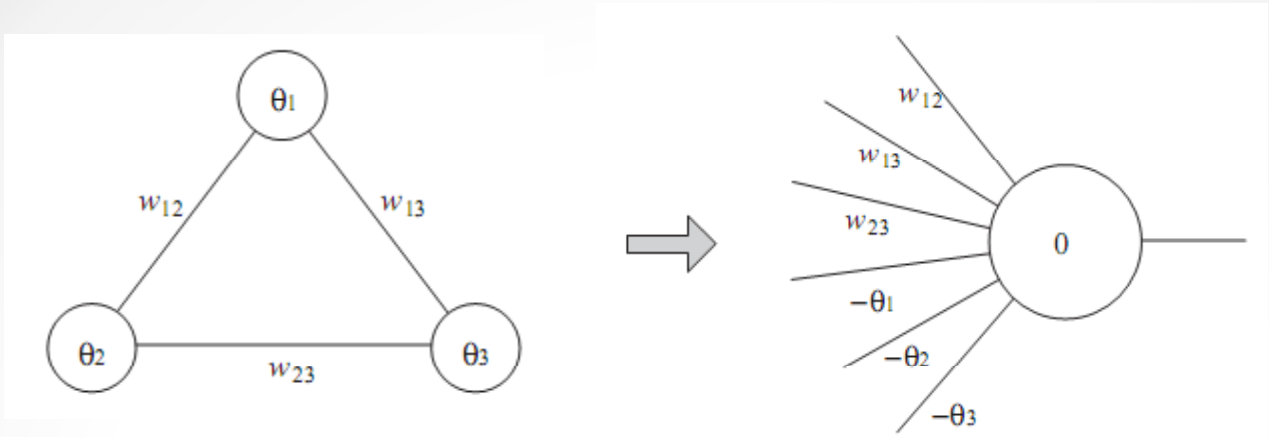
unit 1: $sgn(x_1)\, z_1 \cdot v > 0$
unit 2: $sgn(x_2)\, z_2 \cdot v > 0$
…
unit n: $sgn(x_n)\, z_n \cdot v > 0$

Solution is found by computing linear separation of vectors $z_1, z_2, \ldots, z_n$.
sgn(x) holds for those belonging to +ve half-space
sgn(x) = -1 for those in –ve half-space

Vector **v** of weights is used for the linear separation. W is deduced from this.
For *m* given vectors:

nm unique auxiliary vectors – to be linearly separated
solution in **v** ⟵ if z's linearly separable

Learning problem of a *n* unit HN transformed into *n+(n-1)/2* dimension learning problem for a perceptron



Each iteration updates weights of edges to a single unit & its threshold

If sign of unit's excitation ≠ sign of desired state,

weights of individual unit (perceptron) corrected

Perceptron learning can be used locally

Every learning algorithm for perceptrons can be used as learning method for HN

They take polynomial time..eg linear separation of *nm* vectors in polynomial time

Therefore, learning problem for HN can be solved in polynomial time

Problems of Combinatorics using Hopfield network

Multiflop problem:

Binary n dim vector where all zeros except for single 1.

When an unit is set to 1, inhibits other units through edges w = -2

Threshold = -1

Set all units to 0 $\rightarrow$ e = 0 > threshold

First unit selected will flip to state to 1 – a stable state

Energy function: $E(x_1,\ldots,x_n) = \sum_{i=1}^{n} (x_i - 1)^2$

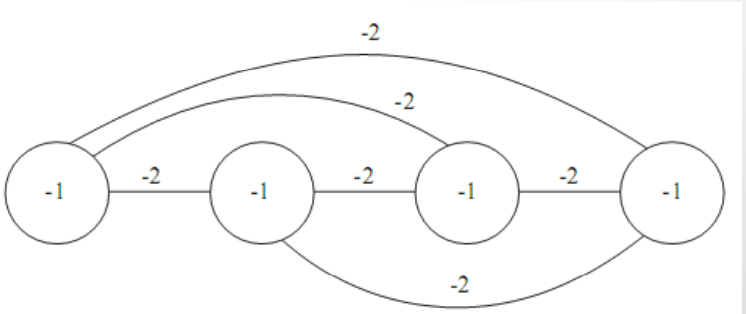
also written as, $E(x_1,\ldots,x_n) = \sum x_i^2 + \sum x_i x_j - 2 \sum x_i + 1$

for binary states, $x_i = x_i^2$

Finally get,

$$E(x_1,\ldots,x_n) = -\tfrac{1}{2} \sum (-2)\, x_i x_j + \sum (-1)\, x_i + 1$$

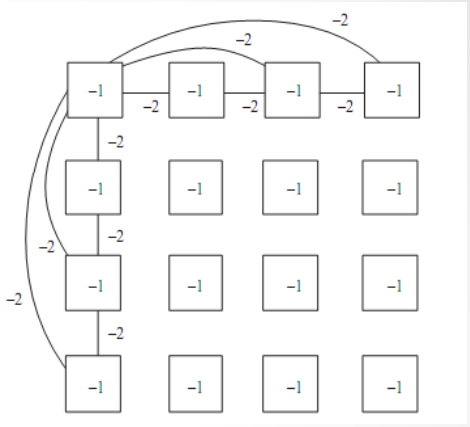
this energy function isomorphic to Hopfield network one.



The Eight Rooks Problem:

Position *n* rooks on a chess board such that no one figure can take another. Each rook will be positioned in a different row and column.

In each row only one square set to 1. Generalization of multiflop probl
The network set up and initialized same as multiflop problem.



Overlapping of multiflop problems. To find weights of the network:

x_{ij} – state of unit corresponding to square *ij* on board
# of 1s given by $\sum x_{ij}$,
If only single 1 allowed in each column/row, minimize following functions:

$\mathbf{E_c () = \sum(\sum x_{ij} - 1)^2}$ minimum when only 1 rook in every column

$\mathbf{E_r() = \sum(\sum x_{ij} - 1)^2}$

Each is sum of *n* independent functions (each row/column)
Threshold of row and column = -1 + -1 = -2 ⟶ upto 2 units set to 1. Not what we want.

Threshold set to -1 ⟶ only one unit in each row or column set to 1, rest set to 0

Eight Queens Problem:

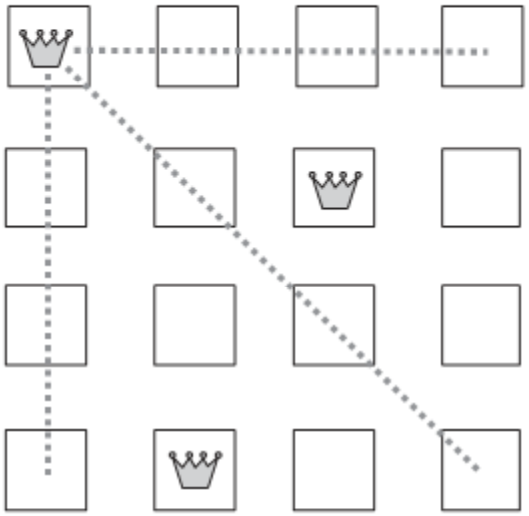
Each diagonal of a *4x4* matrix can be occupied at most once by a queen.

Three multiflop problems overlap – row, column & diagonal.

$w_{ij} = -2$ where $i \neq j$, belong to same row and column

However, this connection pattern does not always give correct solution.

- Diagonal can or cannot be occupied
- energy function too complex, compromise weights we choose for row, column & diagonal



Traveling Salesman & Hopfield Network

Find paths through n cities S1, S2, …Sn such that every city is visited at least once and the length of a round trip is minimal.

Distance d_{ij} is distance between cities S_i and S_j.

Round trip represented by nxn matrix

- n rows – associated with a city
- n columns – associated with n necessary visits.
- $n+1$ column = 1st, roundtrip

Matrix fulfills the same condition as in the Rooks problem.

	1	2	3	4
S_1	1	0	0	0
S_2	0	1	0	0
S_3	0	0	1	0
S_4	0	0	0	1

Length function that needs to be minimized:

$$L = \tfrac{1}{2} \sum_{i,j,k}^{n} d_{ij}\, x_{ik}\, x_{j,k+1}$$ where x_{ik} , $x_{j,k+1} = 1$

Add constraints for legal path, use rooks problem function for new energy function:

$$L = \tfrac{1}{2} \sum_{i,j,k}^{n} d_{ij}\, xi_k\, xj_{,k+1} + \gamma/2 \;\; (\sum_{j=1}^{n}(\sum_{i=1}^{n} (x_{ij} - 1)^2) + \sum_{i=1}^{n}(\sum_{j=1}^{n} (x_{ij} - 1)^2)$$

γ regulates weight given to minimization of length or generation of legal path

Traveling Salesman & Hopfield network

Weights of the network set to $-\gamma$ and threshold to $-\gamma/2$.

weights are modified by adding length between states

Therefore, weight of edge between unit *ik* and *j,k+1*:

$$w_{ik,jk+1} = -d_{ij} + t_{ik,jk+1}$$

$t_{ik,jk+1} = -\gamma$ for units of same row, column. Otherwise is 0.

Issues:

paths generated are not always legal. Can force to generate only legal paths :

set γ to a very large value – ignores distances completely

Very large γ means large # of cities – many local minimums –> no global minimum.

Need massively parallel systems to solve TSP

no. of units increase quadratically with no. of cities (*n)*

no. of weights increase proportionally to n^4

Limits of Hopfield Network

NP class – are a set of non-deterministic problems whose possible solutions can be verified in polynomial time but cannot be found since time taken increases rapidly as size of problem grows.

Eg. – Traveling Salesman

NP-complete – is a subset of NP.

A problem *p* in NP is said to be in NP-C iff all other problems in NP can be transformed into *p* in polynomial time. Also known as "**NP-hard"**.

P class – problems with algorithms that reach a solution in polynomial time belong to this class.

Here, the application of Hopfield Network is extended to NP-hard problems.

Observations:

size of network explodes during attempts to transform all local minima of HN into an optimal solution.

Complement of NP class…

Limits of Hopfield Network

In class P, for a member problem its complement belongs to P as well.

e.g.: "For problem instance X, is X true for I?"

complement $\longrightarrow$ " For problem instance X, is X false for I?"

A polynomial time algorithm will terminate on each.

Not true for problems in NP. For *Traveling Salesman Decision Problem* computation of tour's length can be verified in polynomial time.

However, complementary problem " Is there no tour with a total length smaller than R?"

for a "yes" – no polynomial time algorithm to verify this assertion

To prove the assertion – new data structure *Co-NP ≠ NP*

Computer scientists hope this will be proven eventually.

Few lemmas to determine conditions under which NP = Co-NP …

Lemma 1: *If there is an NP-C problem X whose complement X^c belongs to NP, then NP = co-NP*

Problem Y in NP – reduced to X (NP-C) in polynomial time, implies Y^c reduced to X^c.

Since a solution of X^c can be verified in polynomial time, same stands true for Y^c .

Therefore, NP = co-NP

Lemma 2: *Let L be an NP –C decision problem and H a Hopfield network with a number of weights bounded by a polynomial on the size of the problem. If H can solve L (100% success rate) then NP = co-NP.*

Given L assigned a size, there is a polynomial bound on no. of weights for H to be derived from energy function.

100% success rate $\longrightarrow$ all local minima help decide truth or falsity of L. H used to verify whether solution found is indeed local minimum.

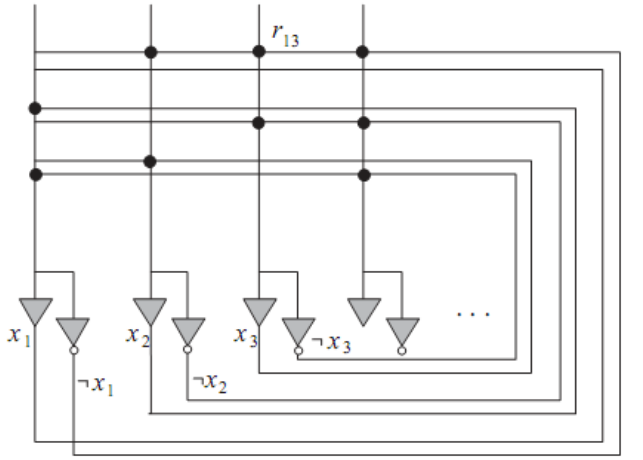
Polynomial size of net makes the decision possible in polynomial time. In TSDP, "yes" by comparing tour with decision threshold. For complement, optimal tour compared with same.

Both belong to NP. So from Lemma 1, NP = co-NP.

H does not exist otherwise.

Implementation of Hopfield Network

Electrical Implementation:



$x_1, x_2, \ldots x_n$ equivalent to states of HN

Resistance, $r_{13} = 1/ w_{13}$

Inhibiting connections, inverted outputs

Network of *n* amplifiers, current I_i (excitation of unit i)

$$I_i = \sum_{j=1}^{n} x_j / r_{ij} = \sum_{j=1}^{n} x_j \, w_{ij}$$

r_{ij} is –ve if input of an unit is the inverted output.

Total excitation transformed into 0 or 1

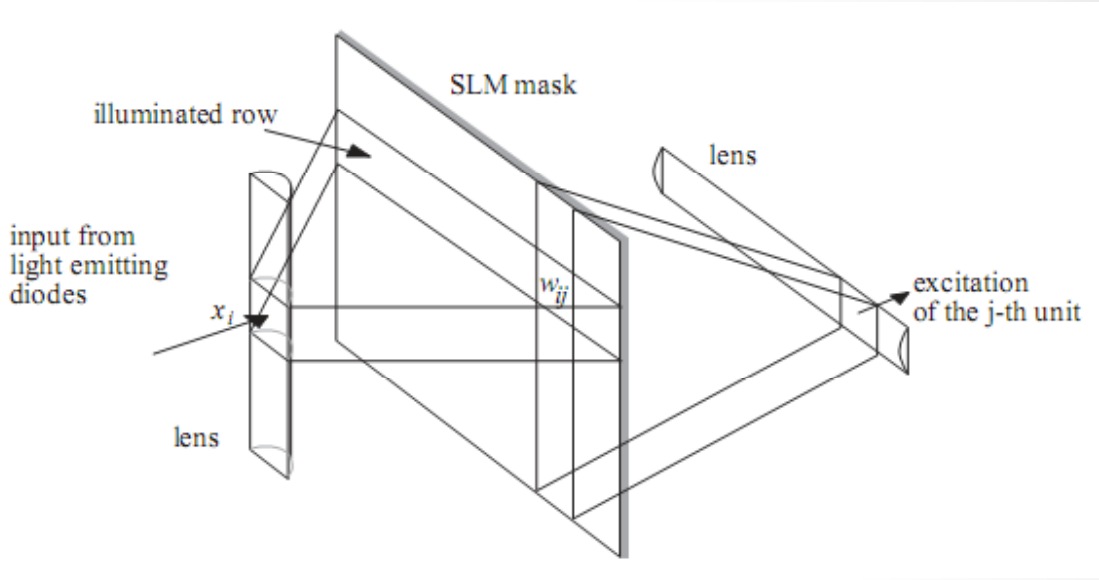
Optical Implementation:

Used to perform matrix multiplication faster.

States x_i projected using LED.

Luminosity proportional to state.

Incoming light at lens – light from a column of the mask – collected at single position.



$$s_j = \sum_{i=1}^{n} w_{ij} x_i$$

total excitation processed by analog/digital circuit. The unit state used for new iteration.

Weights are normalized. No direct connections. Easy to implement large networks.

Comments

Hopfield networks is a very effective tool for analyzing convergence of neural networks.

It has a simple implementation.

Not applicable to NP problems

Need special hardware to compete with methods used in sequential computers.