Relaxation and Hopfield Networks

Neural Networks

Bibliography

Hopfield, J. J., "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences* **79**:2554-2558, 1982.

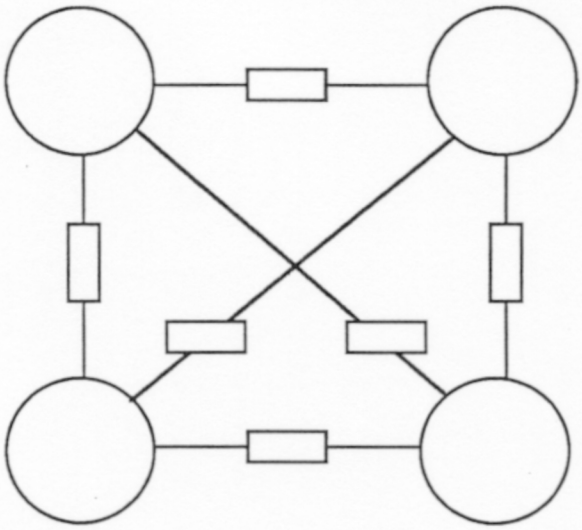
Hopfield, J. J., "Neurons with graded response have collective computational properties like those of two-state neurons." *Proceedings of the National Academy of Sciences* 81: 3088-3092, 1984.

Abu-Mostafa, and J. St. Jacques, Information Capacity of the Hopfield Model, *IEEE Trans. on Information Theory*, Vol. IT-31, No. 4, 1985.

Hopfield Networks

Relaxation

Totally Connected



Bidirectional Links (Symmetric)

Auto-Associator

1 0 0 1 1 0 0 1 0 0
0 0 1 1 1 0 1 0 1 1

Energy Landscapes formed by weight settings

No learning - Programmed weights through an energy function

Early Hopfield

Each unit is a threshold unit (0,1)

Real valued weights

$$V_j = \sum_{i \neq j}^{n} (T_{ij} \cdot V_i) - I_j$$

More recent models use sigmoid rather than Threshold

Similar in overall functionality

Sigmoid gives improved performance

System Energy equation

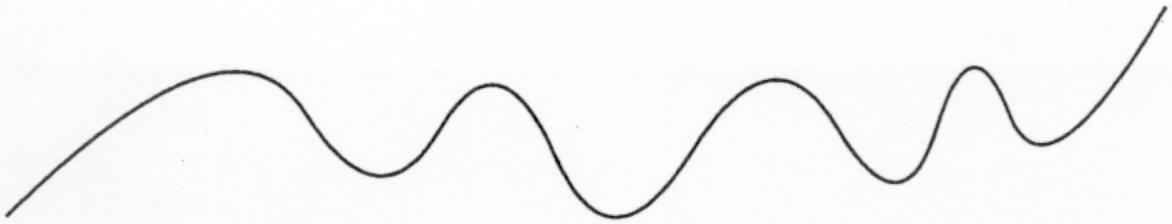
$$E = -\frac{1}{2}\sum_{ij}^{n}(T_{ij} \cdot V_i V_j) - \sum_{j=0}^{n}(I_j \cdot V_j)$$

T: weights
V: outputs
I: Bias

Correct Correlation gives Lower System energy

Thus, minima must have proper correlations fitting weights

Programming the Hopfield Network

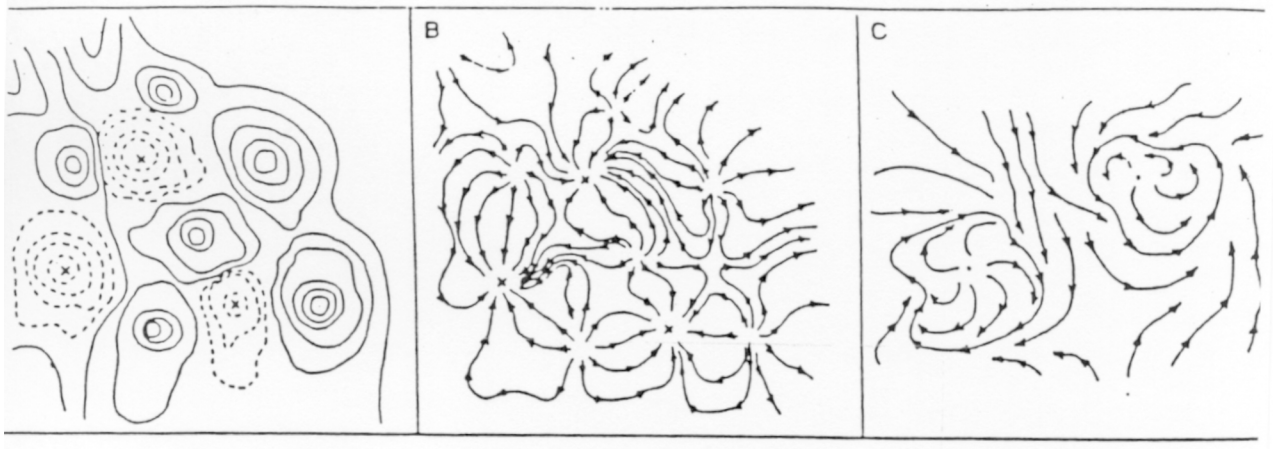


Derive Proper Energy Function

Stable local minima represent good states (memory)

Set connectivity and weights to match the energy function.

Relaxation and Energy Contours



When does a node update

$$Vj = \sum_{i \neq j}^{n} (T_{ij} \cdot V_i) - I_j$$

Continuous - Real System

Random Update - Discrete Simulation

If not random then oscillations can occur

Processing:

Start system in initial state

random
partial
total

Will relax to nearest stable minima

Hopfield as a CAM

(Content Addressable Memory)

Start with totally connected network with number of node equal number of bits in the training set patterns

Set the weights according to:

$$T_{ij} = \sum_{s=1}^{n} (2V_i^s - 1)(2V_j^s - 1)$$

i.e. increment weight between two nodes when they have the same value, else decrement the weight

Could be viewed as a distributed learning mechanism in this case

Number of storable patterns ≈ .15N

No Guarantees, Saturation

Limited by Lower order constraints

Has no hidden nodes, higher order units

All nodes visible

i.e. Program as CAM

0 0 0
0 1 1
1 0 1
1 1 0

However, relaxing auto-association allows a garbled input to return a clean output

Assume two patterns trained
A -> X
B -> Y

Now enter the example with .6A and .4B
Result in a Backprop model?
Result in the Hopfield autoassociator: X

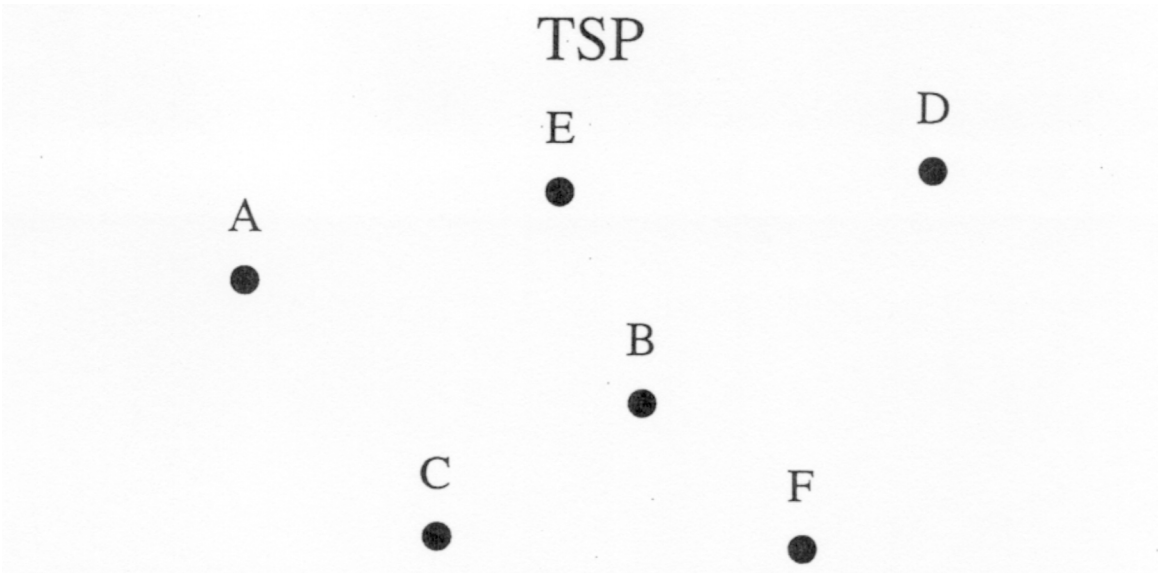
Hopfield as a Computation Engine

Optimization

Travelling Salesman Problem (TSP)
NP-Complete

"Good" vs. Optimal Solutions

Very Fast Processing



Shortest Cycle with no repeat cities

	1	2	3	4	5	6
A	0	0	0	0	1	0
B	0	0	1	0	0	0
C	1	0	0	0	0	0
D	0	0	0	1	0	0
E	0	1	0	0	0	0
F	0	0	0	0	0	1

N cities requires N^2 nodes

2 ** (N^2) possible states

N! Legal paths

N!/2N distinct legal paths

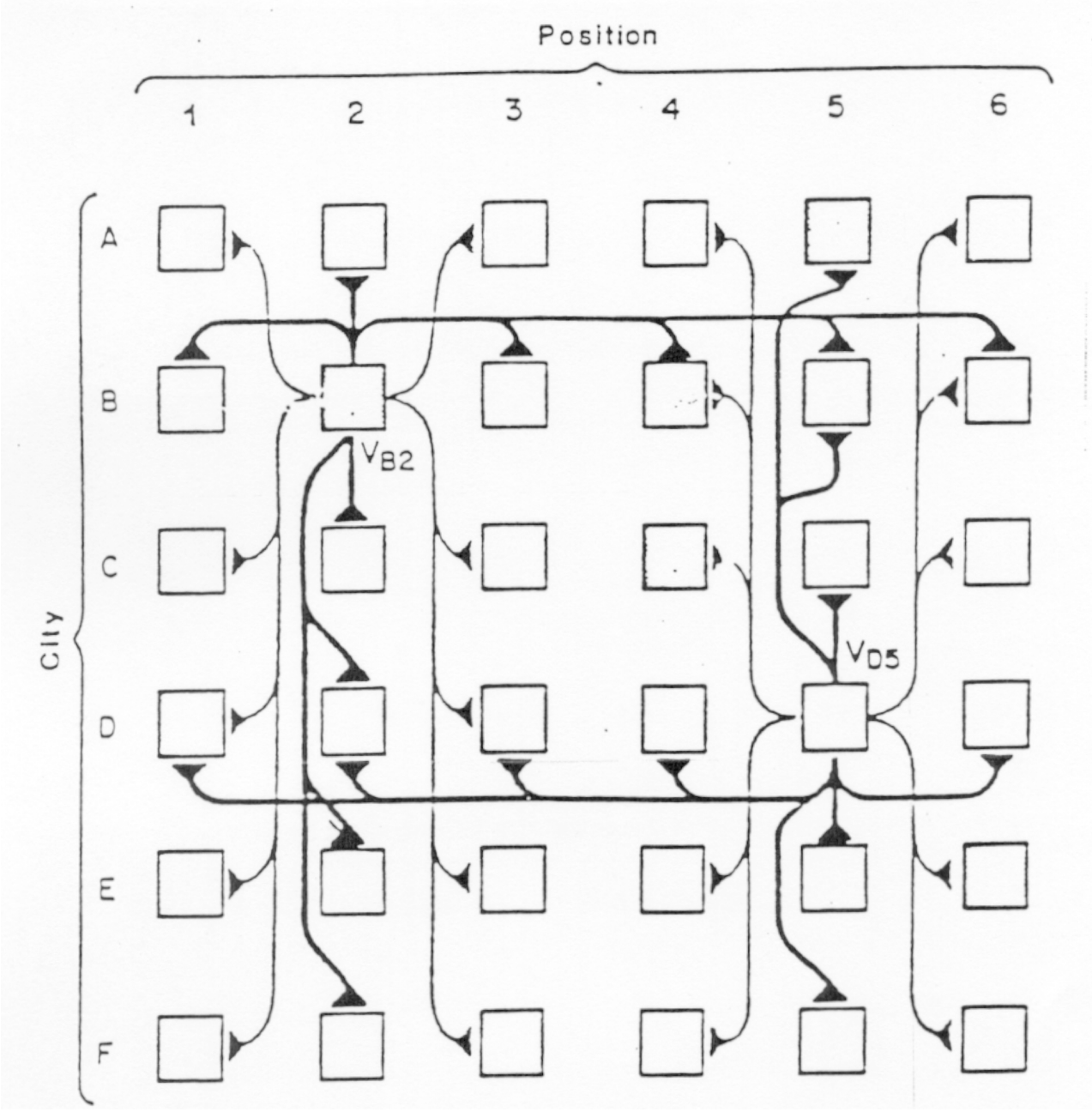
Derive Energy equation for TSP

1. **Legal State**
2. **Good State**

Set weights accordingly

How would we do it

Network Weights



Actual Function used and Parameters

$$E = A/2 \sum_X \sum_i \sum_{j \neq i} V_{Xi} V_{Xj} + B/2 \sum_i \sum_X \sum_{X \neq Y} V_{Xi} V_{Yi}$$

$$+ C/2 \left(\sum_X \sum_i V_{Xi} - n \right)^2$$

$$1/2 D \sum_X \sum_{Y \neq X} \sum_i d_{XY} V_{Xi} (V_{Y,i+1} + V_{Y,i-1}).$$

Weight Settings

$T_{Xi,Yj} = -A\delta_{XY}(1-\delta_{ij})$ "inhibitory connections within each row"

$-B\delta_{ij}(1-\delta_{XY})$ "inhibitory connections within each column"

$-C$ "global inhibition"

$-D d_{XY}(\delta_{j,i+1} + \delta_{j,i-1})$ "data term"

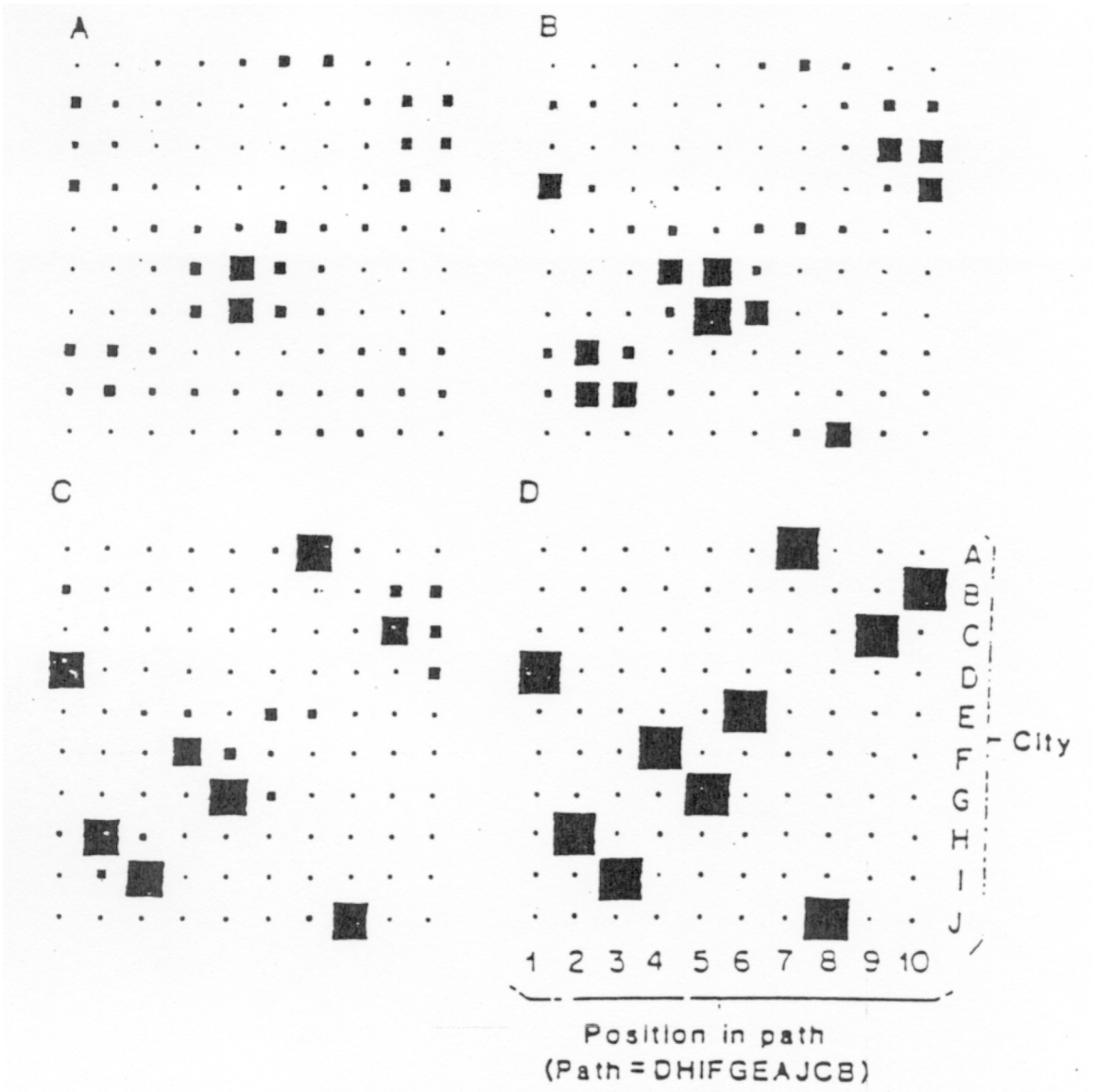
$[\delta_{ij} = 1$ if $i = j$ and is 0 otherwise].

The external input currents are:

$I_{Xi} = +Cn$ "excitation bias".

Parameters

A=B=D=500 C=200 n=15 μ_0 = .02



For N=30

$4.4 * 10^{30}$ Distinct Legal Paths
Typically finds one of 10^7 best,
Thus pruning 10^{23}

How do you handle occasional bad minima?

Summary

Much Current Work

Saturation and No Convergence

For Optimization, Saturation is moot

Many important Optimization problems

Non learning, but reasonably intuitive programming - extensions to learning

Highly Parallel

Expensive Interconnect

Lots of Physical Implementation work, optics