

Using a Hopfield Network: A Nuts and Bolts Approach

November 4, 2013

Gershon Wolfe, Ph.D.

Hopfield Model as Applied to Classification

- Hopfield network
- Training the network
- Updating nodes
- Sequencing of node updates
- When to stop updating
- Examples
- Cellular automation

Hopfield Network

- A neural network in a highly interconnected system of simple two-state elements
- Inspired by statistical mechanics

Networks with symmetric connections \approx Equilibrium properties of magnetic systems

Hopfield showed that this equivalence can be used to design neural circuits for associative memory

Example: Spin glasses, magnetic systems with randomly distributed ferromagnetic and antiferromagnetic interactions.

The spin glass phase (low temp phase) has long range interactions that create a rich topology with many local minima very close to the Energy of the ground state. **Neural systems share many properties with long range spin glasses.**

Hopfield Network

- Hopfield network (1980s) is a recurrent neural network (RNN) where connections between units form a directed cycle
 - An RNN where all connections are symmetric
 - This guarantees its dynamics will converge
- If trained using Hebbian learning, the Hopfield can perform as a robust content-addressable memory, and resistant to connection alterations

Hebb Rule

- Introduced by Donald Hebb in 1949 in order to explain associative memory.
- Hebb states that: Cell A need to take part in firing cell B, and can occur only if cell A firs just before, but not at the same time as, cell B
- From the point of view of artificial neural networks, the Hebb rule can be describes as determining how to alter the weights between modal neurons
 - Weights increase if two neurons act simultaneously
 - Weights decrease if they act seperatly

Hebb Rule

- Nodes that are both positive or both negative tend to have strong positive weights, nodes opposite in sign tend to have strong negative weights
- Hebb learning: $w_{ij} = x_i x_j$

w_{ij} = weight of connection between neuron i and neuron

x = inputs for the particular neuron

This is pattern learning where weights update after every training example.

Hebb Rule

- In a Hopfield network, connections w_{ij} are set to zero if $i=j$. With binary neurons, with activations of 1 or 0, the connection is:
 - 1 if connected neurons have the same activation for a pattern

Binary Neurons

- An artificial neuron is a mathematical function modeled after biological neurons
- An artificial neuron receives one or more inputs (dendrites) and sums them to produce an output (axon)
- The sums of each node are weighted
- The sums are passed through a linear function known as a transfer function

Binary Neurons

- For a neuron with $m+1$ inputs with signals $x_0 - x_m$ and weight $w_0 - w_m$ the output of the k th neuron is:

$$y_k = \varphi \sum_{j=0}^m w_{kj} x_j$$

The output, y_k , of the transfer function, φ , is binary depending on if the input meets a specific threshold, θ , where:

$$y_k = 1 \text{ if } \theta \geq 0$$

$$y_k = 0 \text{ if } \theta < 0$$

Hopfield Model

- Energy based model because property is derived from a global energy function
- The units of a Hopfield network are binary:
 - Example: 1 and -1
 - Example: 1 and 0
- Every pair of units in the net have a connection that is described by a weight
 - w_{ij}

Hopfield Network

- John Hopfield realized that if the connections are symmetric, there is a global energy function
 - Each binary configuration, or assignment of binary values to each neuron of the network, of the network has an energy
- With the right energy function, the binary threshold decision rule causes the network to settle to a minimum of the energy function
- Keep applying this rule, the network will end up in an energy minimum

The Energy Function

- The global energy is the sum of many contributions, and each contribution depends on one connection weight, and the binary state of two neurons. Low energy is good:

$$E = - \sum_i s_i b_i - \sum_{i < j} s_i s_j w_{ij}$$

w_{ij} \equiv Symmetric connection strength between two neurons

s \equiv Activities of the two connected neurons, these are binary variables (1 or 0)

b \equiv Bias term that only involves state of individual units

The Energy Function

- This simple energy function makes it possible for each unit to compute locally how its state effects the global energy
- Define an energy gap as the difference between when I is off and on

$$E_{\text{gap}} = \Delta_{\pm_i} = E(s_i = 0) - E(s_i = 1) = b_i + \sum_j s_j w_{ij}$$

This difference is what is being computed by the binary threshold rule.

The Energy Function

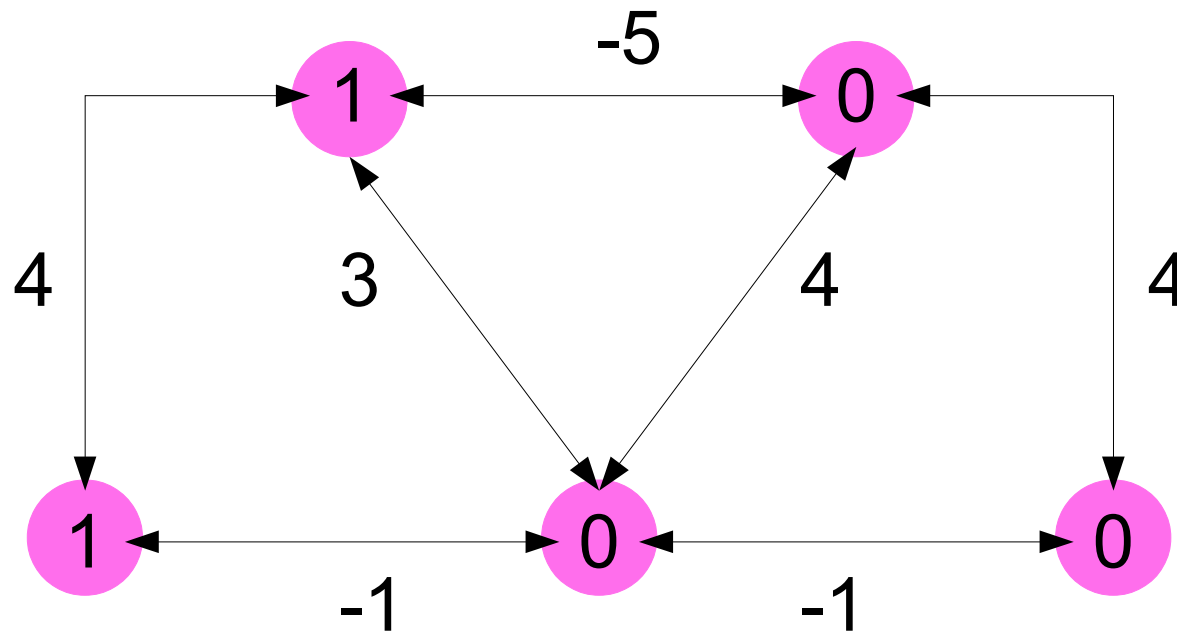
- Differentiate the energy equation wrt the state if the i^{th} unit (a binary variable)

$$\frac{dE_{\text{gap}}}{ds_i} = \Delta_{\pm_i} = b_i + \sum_j s_j w_{ij}$$

One way to find an energy minimum is to start from a random state and update units one at a time in random order. For each unit, save the lowest global energy of the two states. Independent of the state it was previously in. → The binary threshold decision rule.

Finding Energy Minimum

- Start with a random global state, use binary threshold decision rule

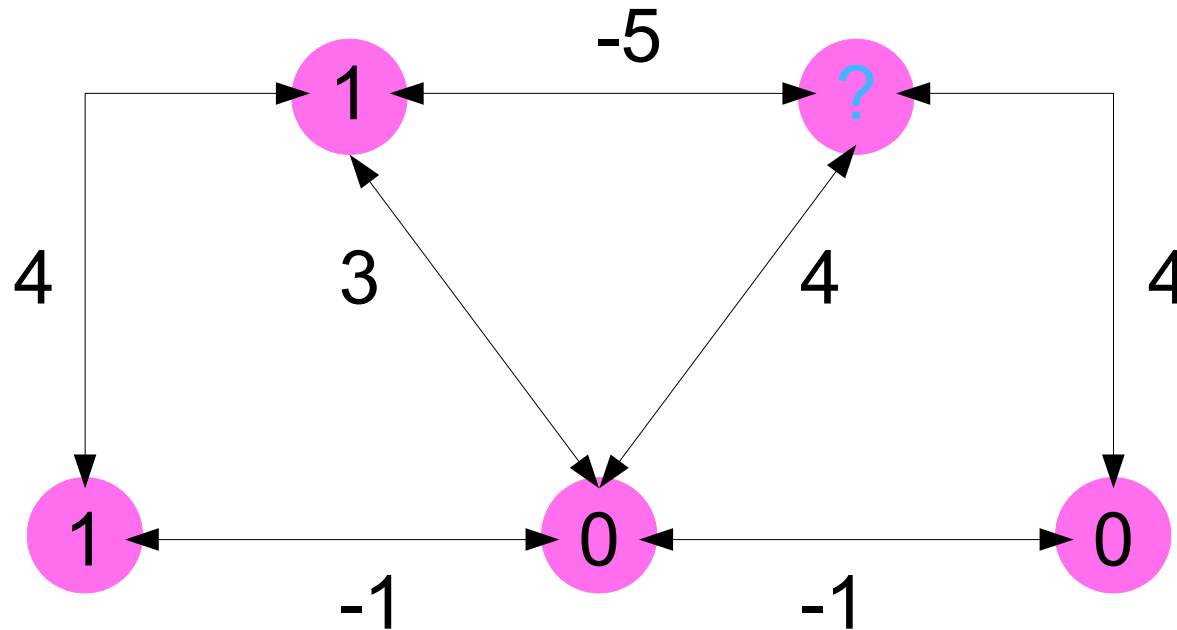


$$-E_{\text{random}} = 4$$

Look at all pairs that are on

Probing Random Units

- Start probing units at random, and ask what state should this unit be in given current state of all the other units

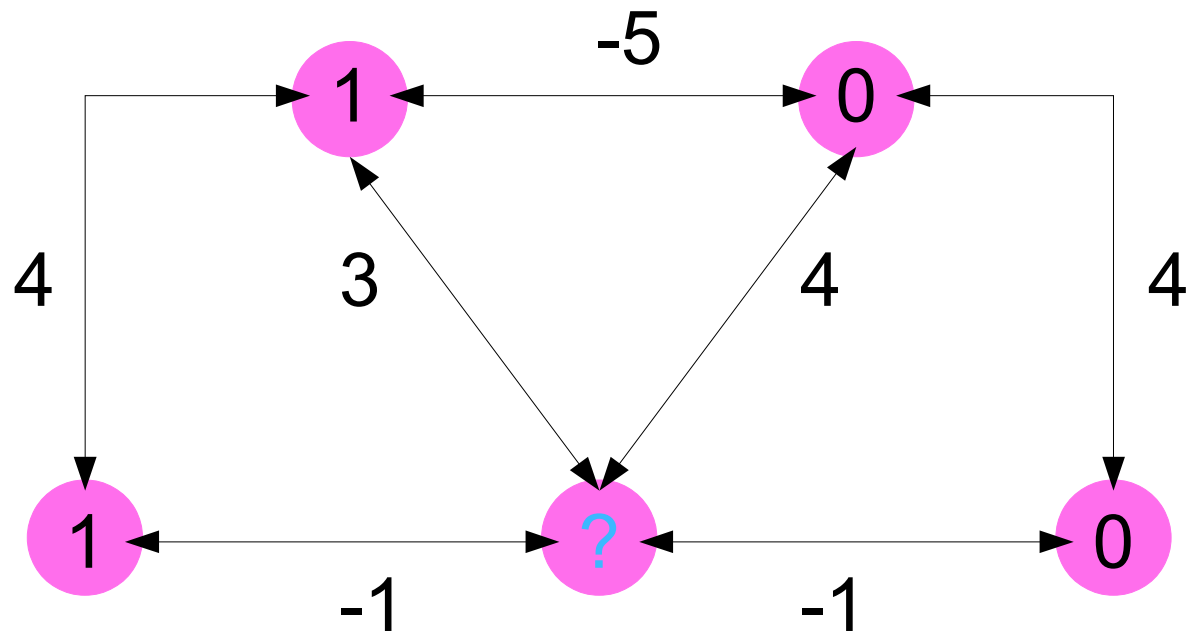


$$E_{?} = -5*(1) + 4*(0) + 4*(0) = -5$$

Since $-5 < 0 \rightarrow$ node is turned off $? \rightarrow 0$

Probing Random Units

- Choose another node



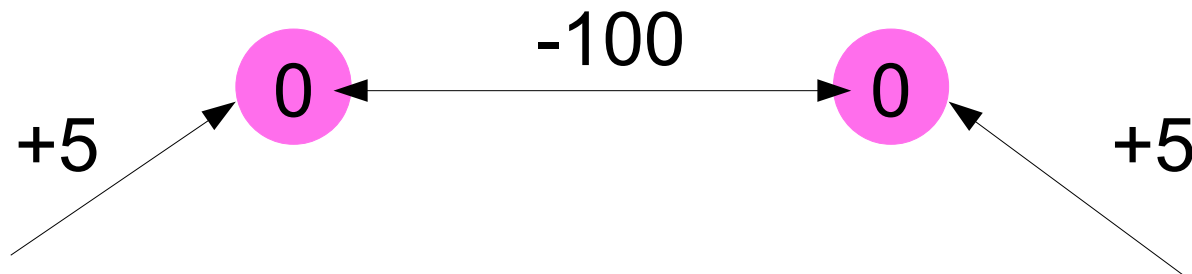
$$E_{?} = -1*(1) + 3*(1) + 4*(0) = 2$$

Since $-2 \geq 0 \rightarrow$ node is turned on $? \rightarrow 1$

When on, the global energy changes $\rightarrow E_{\text{global}} = 6$

Decisions Need to be Sequential

- If units make simultaneous decisions the energy could go up
- With parallel updates you can get oscillations with a period of 2



Bias of +5 and a weight between them of -100

Hopfield Model

- Hopfield (1982) proposed that memories could be energy minima of a neural net with symmetric weights
 - The binary threshold decision rule can take partial memory and clean them up into full memories
- The idea of memories as energy minima was proposed by I.A. Richards in 1924
 - Memories are like a large crystal than can sit on different faces
- Using energy minima to represent memories gives a content-addressable memory
 - An item can be accessed by just knowing part of its content
 - Like reconstructing a dinosaur from a few of its bones

Storing Memories in a Hopfield Net

- If we use activities of 1 and -1, you can store a binary state vector by incrementing the weight between any two units by the product of their activities

$$\Delta W_{ij} = S_i S_j$$

- Not error driven, you are not comparing what you would of predicted with what the right answer is and then making small adjustments
- With states of 0 and 1 the rule is slightly more complicated

$$\Delta W_{ij} = 4(S_i - 1/2)(S_j - 1/2)$$

Hopfield Model

- A simple pattern that is meant to be recognized by a Hopfield net





Trained-on pattern



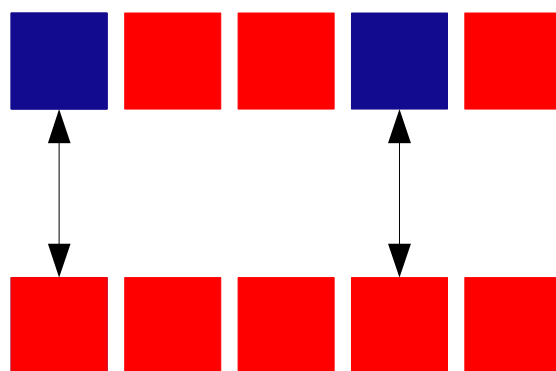
Input pattern

Hopfield Model

- Each pixel, or colored square, is one node in the network
- You train, or assign weights, to multiple patterns
- Given the input pattern,  , the Hopfield network iterates through the weights and will reproduce this pattern 
- The array of pixels could represent anything
 - 2D spectra, mosaic patterns, quasi-crystals, topology, sound waves, etc.

Hopfield Model

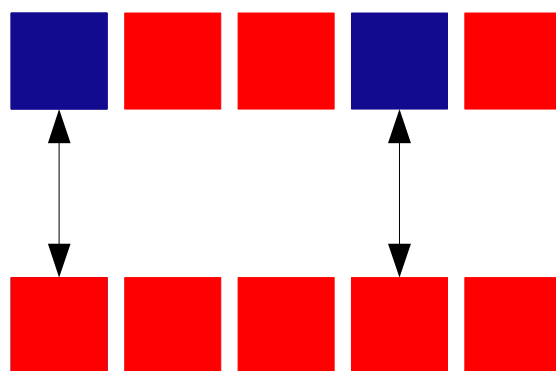
- Hamming distance: The number of positions between two strings of equal length in which the corresponding positions are different



Measuring the minimum number of substitutions required to change one string into the other. In this example the Hamming distance is 2. The Levenshtein distance is a bit more sophisticated, and will handle strings of different lengths, and is related to pairwise string alignment..

Hopfield Model

- Hamming distance: The number of positions between two strings of equal length in which the corresponding positions are different



Measuring the minimum number of substitutions required to change one string into the other. In this example the Hamming distance is 2. The Levenshtein distance is a bit more sophisticated, and will handle strings of different lengths, and is related to pairwise string alignment..

Levenstein Distance

- Used in optical character recognition
- Pattern recognition: Trained from labelled or unlabeled data
 - Supervised learning: Analyze training data and produce a function that can be used to map new examples. Both inputs and outputs are provided.
 - Unsupervised learning: Learn to represent particular input patterns in a way that reflects the statistical structure of the overall collection. It must learn these by itself.
-

Hopfield Model

- Computational time: N pixels, N^2 weights,
 - Can become computationally time demanding

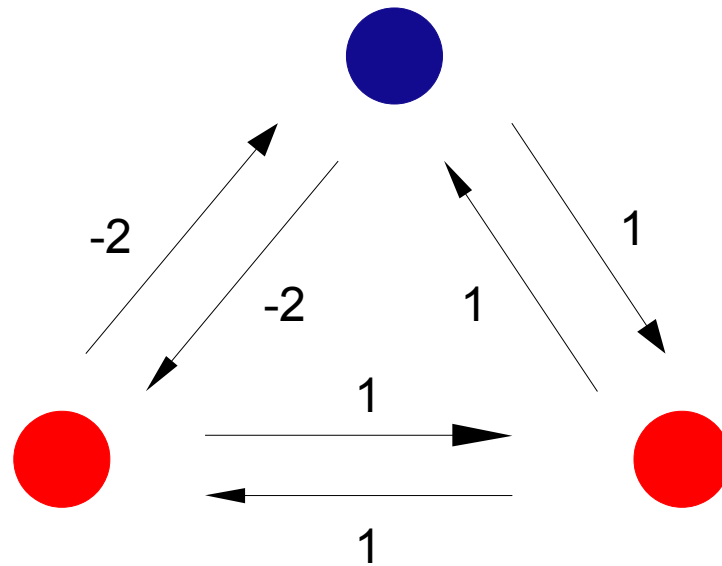
Unsupervised Learning

- If there is no teacher, how do unsupervised networks learn?
 - Association of different bits of information
 - Very useful for character recognition

-

Hopfield Model

- All nodes in a Hopfield network are both inputs and outputs, and fully interconnected.
- Links from each node to itself have a weight of zero.



Hopfield Network

- From previously slide, all nodes are input to each other, and they are also outputs.
- How it works
 - Input a distorted pattern onto the nodes of the network
 - Iterate many times
 - Eventually, the distorted pattern will be transformed onto one of the trained patterns

Hopfield Model

- What you need to know
 - How to train the network
 - How to update a node in the network
 - Sequencing of node update
 - How to tell when you are at one of the trained patterns

Training

- Calculate values of weights, W_{ij}

Suppose you want to store a set of states, V^s
 Where $s = 1, \dots, n$. The weights are calculated
 As follows:


$$W_{ij} = \sum_{s=1}^n (2\Diamond_E^\Delta | 1)(2\Diamond_Z^\Delta | 1)$$

$$\wedge_{EZ} = 0 \quad \wedge_{EE} = Z$$

$\leq \wedge_{EK} \oplus \wedge_{KJ} \wedge_{JK} \wedge_{EK}$:

$$\wedge_{EZ} = (2\Diamond_E | 1)(2\Diamond_Z | 1)$$

Training

- Example: A five node network and you want to train it on the following pattern: 
- Generate a matrix of 5x5 weights

0	W_{12}	W_{13}	W_{14}	W_{15}
W_{21}	0	W_{23}	W_{24}	W_{25}
W_{31}	W_{32}	0	W_{34}	W_{35}
W_{41}	W_{42}	W_{43}	0	W_{45}
W_{51}	W_{52}	W_{53}	W_{54}	0

Training

- The weights are symmetric

$$/_{\mathbb{Z}} = (2\hat{\diamond}_{\mathbb{E}} | 1)(2\hat{\diamond}_{\mathbb{Z}} | 1) = (2\hat{\diamond}_{\mathbb{Z}} | 1)(2\hat{\diamond}_{\mathbb{E}} | 1) = /_{\mathbb{Z}}$$

Only have to calculate the upper diagonal, then copy each of the weights into its inverse weight

If $V = (0 \ 1 \ 1 \ 0 \ 1)$, then $V_1 = 0$, $V_2 = 1$, $V_3 = 1$, $V_4 = 0$, $V_5 = 1$

Training

- Calculate weights for upper triangle

$$\mathbf{W}_{12} = (2\mathbf{V}_1 - 1)(2\mathbf{V}_2 - 1) = (0 - 1)(2 - 1) = (-1)(1) = -1$$

$$\mathbf{W}_{13} = (2\mathbf{V}_1 - 1)(2\mathbf{V}_3 - 1) = (0 - 1)(2 - 1) = (-1)(1) = -1$$

$$\mathbf{W}_{14} = (2\mathbf{V}_1 - 1)(2\mathbf{V}_4 - 1) = (0 - 1)(0 - 1) = (-1)(-1) = 1$$

$$\mathbf{W}_{15} = (2\mathbf{V}_1 - 1)(2\mathbf{V}_5 - 1) = (0 - 1)(2 - 1) = (-1)(1) = -1$$

$$\mathbf{W}_{23} = (2\mathbf{V}_2 - 1)(2\mathbf{V}_3 - 1) = (2 - 1)(2 - 1) = (1)(1) = 1$$

$$\mathbf{W}_{24} = (2\mathbf{V}_2 - 1)(2\mathbf{V}_4 - 1) = (2 - 1)(0 - 1) = (1)(-1) = -1$$

$$\mathbf{W}_{25} = (2\mathbf{V}_2 - 1)(2\mathbf{V}_5 - 1) = (2 - 1)(2 - 1) = (1)(1) = 1$$

$$\mathbf{W}_{34} = (2\mathbf{V}_3 - 1)(2\mathbf{V}_4 - 1) = (2 - 1)(0 - 1) = (1)(-1) = -1$$

$$\mathbf{W}_{35} = (2\mathbf{V}_3 - 1)(2\mathbf{V}_5 - 1) = (2 - 1)(2 - 1) = (1)(1) = 1$$

$$\mathbf{W}_{45} = (2\mathbf{V}_4 - 1)(2\mathbf{V}_5 - 1) = (0 - 1)(2 - 1) = (-1)(1) = -1$$

Training

- Weight matrix

0	-1	-1	1	-1
-1	0	1	-1	1
-1	1	0	-1	1
1	-1	-1	0	-1
-1	1	1	-1	0

Training

- Remember that original formula was set up to have n patterns
 - $V^1 = (0 \ 1 \ 1 \ 0 \ 1)$
 - $V^2 = (1 \ 0 \ 1 \ 0 \ 1)$

Recall that the weights for n states are:

$$W_{ij} = \sum_{s=1}^n (2\langle \Delta_E | 1 \rangle)(2\langle \Delta_Z | 1 \rangle)$$

Training

- Calculate weights for the two sets

$$\begin{aligned}W_{12} &= \sum (2\Diamond^{\Delta}_E | 1)(2\Diamond^{\Delta}_Z | 1) \\&= (2\Diamond^1_1 | 1)(2\Diamond^1_2 | 1) + (2\Diamond^2_1 | 1)(2\Diamond^2_2 | 1) \\&= (2/0 | 1)(2/1 | 1) + (2/1 | 1)(2/0 | 1) \\&= (0 | 1)(2 | 1) + (2 | 1)(0 | 1) \\&= (21)(1) + (1)(21) \\&= 21 + 21 \\&= 22\end{aligned}$$

Training

- The complete weight matrix for the two patterns

0	-2	0	0	0
-2	0	0	0	0
0	0	0	-2	2
0	0	-2	0	-2
0	0	2	-2	0

Updating Nodes

- The weight matrix is meant to recognize the patterns:




Differ by 2 bits



Patterns this close together might be difficult to tell apart

Updating Nodes

- Use an input state of (1 1 1 1 1) or 
- For example, if node 3 is being updated:
 - The values of all the other nodes are used as input values, and the weights from those nodes to node 3 as the weights
- Steps:
 - Do a weighted sum of inputs from other nodes
 - Then if value is \geq to 0, output a 1
 - Otherwise output a 0

Updating Nodes

- In formula form:

$$V_i^{\text{in}} = \sum_{z \in Z} \phi_z$$

$$\phi_E \rightarrow 1 \quad \forall \phi_E \geq 0$$

$$J \otimes \phi_E \rightarrow 0$$

Updating Nodes

- Using the weight matrix from the previous slide

0	-2	0	0	0
-2	0	0	0	0
0	0	0	-2	2
0	0	-2	0	-2
0	0	2	-2	0

$$\begin{aligned}
 V_3^{\text{in}} &= \sum_{Z \neq 3} W_{3Z} \diamond_Z \\
 &= 1 \diamond_1 + 2 \diamond_2 + 4 \diamond_4 + 5 \diamond_5 \\
 &= 0/1 + 0/2 + 2/1 + 2/1 \\
 &= 0 \quad \text{OKUJ } 0 \geq =, \text{ITJK } \diamond_3 = 1
 \end{aligned}$$

Sequencing of Node Updates

- How do you update a neuron if the values of its inputs are changing? There are two ways
 - Synchronous updating: All nodes are updated at the same time. Just multiply the weight matrix by the vector of the current state. However, this is not realistic in a neural sense, because neurons don't update at the same rate. They have delays, firing times, etc... → More realistic is to update the nodes in random order. In fact, this is the method described by John Hopfield in 1982.

Sequencing of Node Updates

- Keep updating the neurons in a random manner until the system is in a stable state.
- An example of a sequence for node updates could look like this: 3 2 1 5 4 2 3 1 5 4, etc...

When to Stop Updating Network

- Simply put, if you go through all the nodes and none of them change, then you can stop.

Finish the Example

- Given the weight matrix for a 5 node net with $(0\ 1\ 1\ 0\ 1)$ and $(1\ 0\ 1\ 0\ 1)$ as attractors, start at state $(1\ 1\ 1\ 1\ 1)$ and see where it goes.
- Update in sequence: 3 1 5 2 4 3 1 5 2 4, etc...
We know from above that node 3 did not change.
- The order of the sequence of updates has a huge effect on classifying patterns that are very similar.

Finishing the Example

- Update node 1:
 - $V_1 \text{in} = (0 \ -2 \ 0 \ 0 \ 0) * (1 \ 1 \ 1 \ 1 \ 1) = -2$
 - Since $-2 < 0$, $V_1 = 0$ (changes)
- Update node 5:
 - $V_5 \text{in} = (0 \ 0 \ 2 \ -2 \ 0) * (0 \ 1 \ 1 \ 1 \ 1) = 0$
 - Since $0 \geq 0$, $V_5 = 1$ (does not change)
- Update node 2:
 - $V_2 \text{in} = (-2 \ 0 \ 0 \ 0 \ 0) * (0 \ 1 \ 1 \ 1 \ 1) = 0$
 - Since $0 \geq 0$, $V_2 = 1$ (does not change)

Finishing the Example

- Update node 4:
 - $V_1 \text{in} = (0 \ 0 \ -2 \ 0 \ -2) * (0 \ 1 \ 1 \ 1 \ 1) = -4$
 - Since $-4 < 0$, $V_4 = 0$ (changes)
- Update node 3:
 - $V_3 \text{in} = (0 \ 0 \ 0 \ -2 \ 2) * (0 \ 1 \ 1 \ 0 \ 1) = 2$
 - Since $2 \geq 0$, $V_3 = 0$ (does not change)
- Update node 1:
 - $V_1 \text{in} = (0 \ -2 \ 0 \ 0 \ 0) * (0 \ 1 \ 1 \ 0 \ 1) = -2$
 - Since $-2 < 0$, $V_1 = 0$ (does not change)

Finishing the Example

- Update node 5:
 - $V_5 \text{in} = (0 \ 0 \ -2 \ 2 \ 0) * (0 \ 1 \ 1 \ 0 \ 1) = 2$
 - Since $2 \geq 0$, $V_5 = 1$ (does not change)
- Update node 2:
 - $V_2 \text{in} = (-2 \ 0 \ 0 \ 0 \ 0) * (0 \ 1 \ 1 \ 0 \ 1) = 0$
 - Since $0 \geq 0$, $V_2 = 1$ (does not change)
- Update node 4:
 - $V_4 \text{in} = (0 \ 0 \ -2 \ 0 \ -2) * (0 \ 1 \ 1 \ 0 \ 1) = -4$
 - Since $-4 < 0$, $V_4 = 0$ (does not change)

Finishing the Example

- Each node has been updated without any change, so the iterations can stop and the state has been classified.
- The network ends up at the attractor (0 1 1 0 1)
- However, trying the exact same problem with a different order for the updating sequence will classify the state into the other attractor, (0 1 0 1 0)

Cellular Automation

- Can be used to propagate information onto a two dimensional surface
- Any signal can be transformed into an array of information using a rule based system
 - Using Rule 30 or Rule 110 to generate a fractal image
 - These images can be trained on using a Hopfield network and via unsupervised learning, can distinguish between states of a system.
 - Systems can include: Drug discovery, up and down regulation, early detection of disease, micro-array data, etc...

Summary

- Shown how the Hopfield model can be used to classify data
- Gave examples
- Integration of cellular automation to prepare data sets for classification