



JOHNS HOPKINS

WHITING SCHOOL
of ENGINEERING



Introduction to Neural Networks

Johns Hopkins University
Engineering for Professionals Program
605-447/625-438

Dr. Mark Fleischer

Copyright 2014 by Mark Fleischer

Module 10.1: The Boltzmann Machine

What We've Covered So Far that is Relevant to Boltzmann Machines

Simulated Annealing

- The thermodynamic basis of SA
- The stationary and transition probabilities associated with SA
- Examples of combinatorial optimization problems

Hopfield Recurrent Neural Networks

- How to define the weight matrix.
- Examined their capability to recall/remember exemplar patterns.
- Examined their memory capacity.

In This Module We Will Cover

The Boltzmann Machine

- A stochastic version of the Hopfield Network
- Consensus/Energy function
- Using the sigmoid function as the activation function
- Briefly discussed the training formulae
- Look at calculating the respective stationary probabilities of various configurations (sets of states)
- Look at how to modify the weights so as to obtain desired stationary probabilities
- Examples

Recall the Hopfield Network

- Memory capacity about 11% of the length of exemplars. E.g., an exemplar vector with 100 elements could store approximately 11 exemplars with very high accuracy.
- Accuracy based on statistical independence of the exemplars, and a 3σ standard deviation.
 - This means a near certainty ($\geq 99\%$) for accurately determining the most likely exemplar that an input vector represents.
 - Remember, the input vector is an exemplar perturbed by some noise.
 - Variance of the 'noise' associated with inputs is approximately $(n - 1)(P - 1)$

A Tradeoff!

- Can we sacrifice some certainty associated with memory recall/completion for **greater memory capacity**?
- Can't be based on issue of 'noise' --- once an input is provided, it is known/certain.
- Don't want it to be based on statistical independence --- relationship among exemplars is not the issue insofar as 'certainty' is concerned.

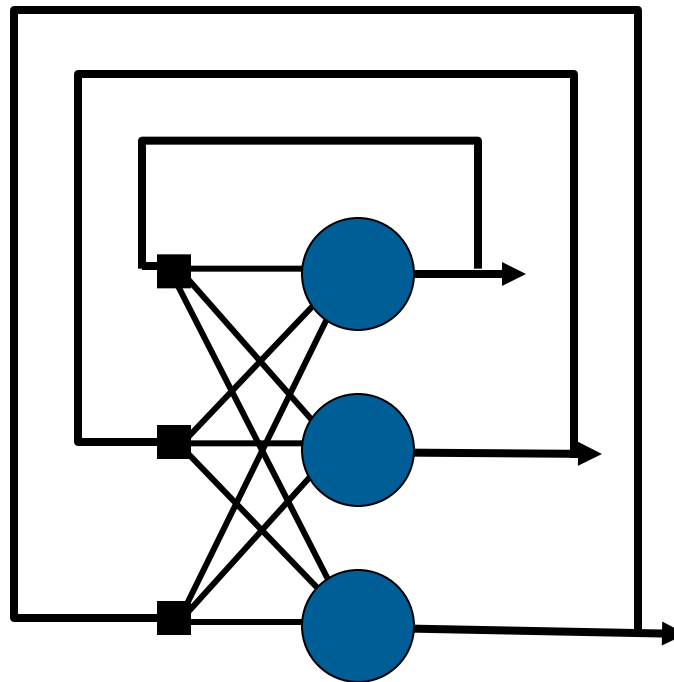
The 'Uncertainty'

- Want to let the network 'run' and hopefully arrive at the correct exemplar.
- Could allow some probability the network will 'arrive' at the wrong exemplar.
- Allow the network to make some mistakes.
- How?

Let the nodes take on random states!

Recurrent Network Topology Reprise

Let's view the network a bit differently.



Exemplar:

$(1, -1, -1)$

What is the weight from
Node 1 to Node 2?

Boltzmann/Hopfield Comparisons

- Very similar in architecture.
- Boltzmann uses stochastic methods for updating node states.
- Hopfield uses bipolar state values.
- Boltzmann typically uses binary state values.

Activity Functions and Energy

- Asynchronous update of neuron activations.
- Cell activity S_i is computed (here without a bias term):

$$S_i = \sum_j w_{ij} x_j$$

The Hecht-Nielsen Function

Define the energy function E

$$E = - \sum_{i < j} w_{ij} x_i x_j + \sum_i \theta_i x_i$$

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



Energy → Consensus

Minimize Energy

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

Maximize Consensus

$$C = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij} x_i x_j - \sum_i \theta_i x_i$$

Energy and Consensus values are
additive inverses of one another!

An Optimization Problem ala Simulated Annealing

- Minimize energy or Maximize consensus.
- Change the state of a node to modify a candidate energy/consensus function value.
 - Means changing it from $0 \rightarrow 1$ or $1 \rightarrow 0$.
- Accept new configuration probabilistically as in SA.

Calculating ΔE

Recall:

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

WLG:

$$E = -\frac{1}{2} \sum_{\substack{i=1 \\ i \neq k}}^n \sum_{\substack{j=1 \\ j \neq k}}^n w_{ij} x_i x_j + \sum_{\substack{i=1 \\ i \neq k}}^n \theta_i x_i - \frac{1}{2} \sum_{j=1}^n w_{kj} x_k x_j - \frac{1}{2} \sum_{i=1}^n w_{ik} x_i x_k + \theta_k x_k$$



Calculating ΔE

$$E_{\text{cand}} = -\frac{1}{2} \sum_{\substack{i=1 \\ i \neq k}}^n \sum_{\substack{j=1 \\ j \neq k}}^n w_{ij} x_i x_j + \sum_{\substack{i=1 \\ i \neq k}}^n \theta_i x_i - \frac{1}{2} \sum_{j=1}^n w_{kj} x'_k x_j - \frac{1}{2} \sum_{i=1}^n w_{ik} x_i x'_k + \theta_k x'_k$$

$$E_{\text{cur}} = -\frac{1}{2} \sum_{\substack{i=1 \\ i \neq k}}^n \sum_{\substack{j=1 \\ j \neq k}}^n w_{ij} x_i x_j + \sum_{\substack{i=1 \\ i \neq k}}^n \theta_i x_i - \frac{1}{2} \sum_{j=1}^n w_{kj} x_k x_j - \frac{1}{2} \sum_{i=1}^n w_{ik} x_i x_k + \theta_k x_k$$

$$E_{\text{cand}} = -\frac{1}{2} \sum_{\substack{i=1 \\ i \neq k}}^n \sum_{\substack{j=1 \\ j \neq k}}^n w_{ij} x_i x_j + \sum_{\substack{i=1 \\ i \neq k}}^n \theta_i x_i - x'_k \sum_{i=1}^n w_{ik} x_i + \theta_k x'_k$$

$$E_{\text{cur}} = -\frac{1}{2} \sum_{\substack{i=1 \\ i \neq k}}^n \sum_{\substack{j=1 \\ j \neq k}}^n w_{ij} x_i x_j + \sum_{\substack{i=1 \\ i \neq k}}^n \theta_i x_i - x_k \sum_{i=1}^n w_{ik} x_i + \theta_k x_k$$

Calculating ΔE

$$E_{\text{cand}} = -\frac{1}{2} \sum_{\substack{i=1 \\ i \neq k}}^n \sum_{\substack{j=1 \\ j \neq k}}^n w_{ij} x_i x_j + \sum_{\substack{i=1 \\ i \neq k}}^n \theta_i x_i - x'_k \sum_{i=1}^n w_{ik} x_i + \theta_k x'_k$$

$$E_{\text{cur}} = -\frac{1}{2} \sum_{\substack{i=1 \\ i \neq k}}^n \sum_{\substack{j=1 \\ j \neq k}}^n w_{ij} x_i x_j + \sum_{\substack{i=1 \\ i \neq k}}^n \theta_i x_i - x_k \sum_{i=1}^n w_{ik} x_i + \theta_k x_k$$

$$\Delta E = E_{\text{cand}} - E_{\text{cur}}$$



Calculating ΔE

$$E_{\text{cand}} - E_{\text{cur}} = -x'_k \sum_{i=1}^n w_{ik} x_i + \theta_k x'_k - -x_k \sum_{i=1}^n w_{ik} x_i - \theta_k x_k$$

$$\Delta E = -x'_k \sum_{i=1}^n w_{ik} x_i + \theta_k x'_k + x_k \sum_{i=1}^n w_{ik} x_i - \theta_k x_k$$

$$= x'_k \left[- \sum_{i=1}^n w_{ik} x_i + \theta_k \right] + x_k \left[\sum_{i=1}^n w_{ik} x_i - \theta_k \right]$$

Change in state
of Node k

$$= x'_k \left[- \sum_{i=1}^n w_{ik} x_i + \theta_k \right] - x_k \left[- \sum_{i=1}^n w_{ik} x_i + \theta_k \right]$$

$$= (x'_k - x_k) \left[- \sum_{i=1}^n w_{ik} x_i + \theta_k \right]$$

Calculating ΔC

Similarly for the consensus value. Thus,

$$\begin{aligned} C_{\text{cand}} - C_{\text{cur}} &= x'_k \sum_{i=1}^n w_{ik} x_i - \theta_k x'_k - x_k \sum_{i=1}^n w_{ik} x_i + \theta_k x_k \\ \Delta C &= x'_k \sum_{i=1}^n w_{ik} x_i + \theta_k x'_k - x_k \sum_{i=1}^n w_{ik} x_i + \theta_k x_k \\ &= x'_k \left[\sum_{i=1}^n w_{ik} x_i + \theta_k \right] - x_k \left[\sum_{i=1}^n w_{ik} x_i + \theta_k \right] \\ &= x'_k \left[\sum_{i=1}^n w_{ik} x_i + \theta_k \right] - x_k \left[\sum_{i=1}^n w_{ik} x_i + \theta_k \right] \\ &= (x'_k - x_k) \left[\sum_{i=1}^n w_{ik} x_i + \theta_k \right] \end{aligned}$$

Relating the Change of State to Probability

$$\pi_i(t) = \frac{e^{-E_i/t}}{\sum_i e^{-E_i/t}}$$
$$\frac{\pi_i(t)}{\pi_{i'}(t)} = \frac{\frac{e^{-E_i/t}}{\sum_i e^{-E_i/t}}}{\frac{e^{-E_{i'}/t}}{\sum_i e^{-E_i/t}}}$$
$$= \frac{e^{-E_i/t}}{e^{-E_{i'}/t}} = e^{(E_{i'} - E_i)/t} = e^{\Delta E/t}$$

Now, taking the logarithm of both sides we get ...

Relating the Change of State to Probability

$$\ln \left(\frac{\pi_i(t)}{\pi_{i'}(t)} \right) = \ln e^{\Delta E/t} = \frac{\Delta E}{t}$$

$$\ln \pi_i(t) - \ln \pi_{i'}(t) = \frac{\Delta E}{t}$$

$$\ln(\Pr\{x_k = 1\}) - \ln(\Pr\{x_k = 0\}) = \frac{\Delta E}{t}$$

$$\ln(\Pr\{x_k = 1\}) - \ln(1 - \Pr\{x_k = 1\}) = \frac{\Delta E}{t}$$

Relating the Change of State to Probability

$$\ln(\Pr\{x_k = 1\}) - \ln(1 - \Pr\{x_k = 1\}) = \frac{\Delta E}{t}$$

$$\ln\left(\frac{\Pr\{x_k = 1\}}{1 - \Pr\{x_k = 1\}}\right) = \frac{\Delta E}{t}$$

$$\ln\left(\frac{1 - \Pr\{x_k = 1\}}{\Pr\{x_k = 1\}}\right) = \frac{-\Delta E}{t}$$

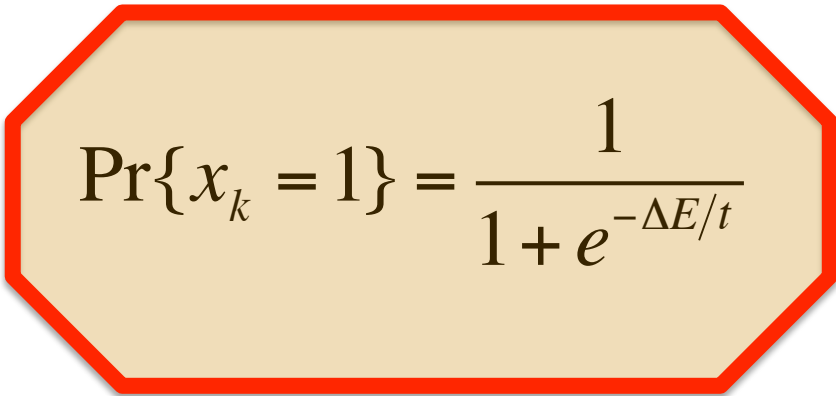
Relating the Change of State to Probability

$$\ln\left(\frac{1 - \Pr\{x_k = 1\}}{\Pr\{x_k = 1\}}\right) = \frac{-\Delta E}{t}$$

$$\frac{1 - \Pr\{x_k = 1\}}{\Pr\{x_k = 1\}} = e^{-\Delta E/t}$$

$$\frac{1}{\Pr\{x_k = 1\}} - 1 = e^{-\Delta E/t}$$

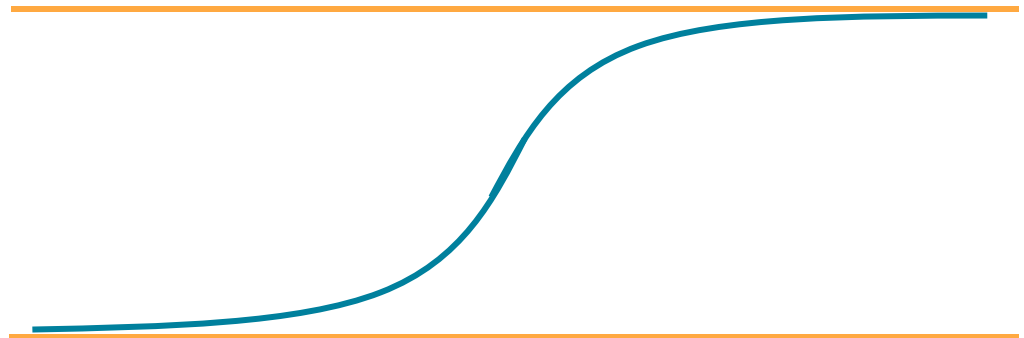
$$\frac{1}{\Pr\{x_k = 1\}} = 1 + e^{-\Delta E/t}$$


$$\Pr\{x_k = 1\} = \frac{1}{1 + e^{-\Delta E/t}}$$

Dynamics

Recall the Sigmoid activation function

$$\frac{1}{1 + e^{-S_i/T}}$$



We're dealing with stochastic neurons.
What does this curve remind you of?

Dynamics

- Yes, a probability distribution function (monotonically increasing to 1).
- Set cell activation (state) according to:

$$x_i = \begin{cases} 1 & \text{w/prob } p_i \\ 0 & \text{w/prob } 1 - p_i \end{cases} \quad p_i = \frac{1}{1 + e^{-S_i/T}}$$

At high temperatures, what is the probability of $x_i = 1$?

Summary

- Each node is update asynchronously and probabilistically.
- The temperature is lowered to minimize the energy value of the network or maximize the consensus value of the network as the case may be.
- Since the node states are probabilistic, **all information is encoded in the weights!**



JOHNS HOPKINS

WHITING SCHOOL
of ENGINEERING



Introduction to Neural Networks

Johns Hopkins University
Engineering for Professionals Program
605-447/625-438

Dr. Mark Fleischer

Copyright 2014 by Mark Fleischer

Module 10.2: Training The Boltzmann Machine

What We've Covered So Far

We derived a formula for $\Delta E/\Delta C$ and showed that it is **equivalent to the activity function**.

$$\Delta E_i = - \sum_j w_{ij} x_j$$

We derived the probability value of a node's state to be equal to the sigmoid function.

$$\Pr\{X_i = 1\} = \frac{1}{1 + e^{-\Delta E_i/t}} \longrightarrow \Pr\{X_i = 1\} = \frac{1}{1 + e^{-S_i/t}}$$

Modalities for Applications

- BM can learn distributions ala supervised learning.
 - by updating weights
- BM can solve constrained optimization problems
 - some of the weights are established by the nature of the problem.
 - using simulated annealing approaches.

Behavior

1. Set T to an initially high temperature value.
2. Randomly select a free-running node i .
3. Compute activation function S_i .
4. Set node i to 1 or 0 according to acceptance function.
5. Reduce T .
6. Goto step 2 if not at equilibrium.

This looks like an annealing process. And it is!

So what are we minimizing?

Dynamics

Goal is to minimize E or maximize C .

Modify the network and in effect run a simulated annealing algorithm on it.

This approach is appropriate when the weights have been determined by the nature of the problem. *E.g.*, COPs.

Other problems require you to train the weights so that the probability distribution for the configurations match a given distribution.

Training Boltzmann Machines

- The Boltzmann distribution is the probability distribution for different configurations.
- The BM ‘learns’ the distributions (not specific targets)!

Training

- If we want to ‘train’ the network, what does this mean?
- We want the relative frequency of configurations to match those of a training set.
- The relative frequency of a configuration is affected by the weight connections.

Training

- Thus, we must find a way to determine how to update weights based on differences between the actual frequency of configurations and ‘desired’ frequency of configurations.

Probability of Configurations

For any two system configurations

$$\alpha = [00110100]$$

$$\beta = [10011010]$$

$$\frac{p_{\alpha}}{p_{\beta}} = e^{-(E_{\alpha} - E_{\beta})/T}$$

Training

- Slow, and arduous.
- Intimately connected with entropy.

Define

p_{α} = The probability of state α based on training examples.

p'_{α} = The probability of state α at equilibrium.

$$G = \sum_{\alpha} p_{\alpha} \ln \left(\frac{p_{\alpha}}{p'_{\alpha}} \right)$$

Ackley et al. 1986

Value of G

If there is a mismatch in probabilities, some values of

$$\left(\frac{p_{\alpha}}{p'_{\alpha}} \right)$$

in the terms in the summation of G will
of necessity be greater than AND less than 1.
Why?

Training

Use function G for gradient descent formulation:

$$\Delta w_{ij} = -\eta \frac{\partial G}{\partial w_{ij}} \text{ where}$$

$$\frac{\partial G}{\partial w_{ij}} = -\frac{1}{T} (p_{ij} - p'_{ij})$$

Where p_{ij} is the probability that x_i and $x_j = 1$ when system is clamped (according to input distribution or training set). Similarly for free-running.

Estimating these values requires a great deal of computation.

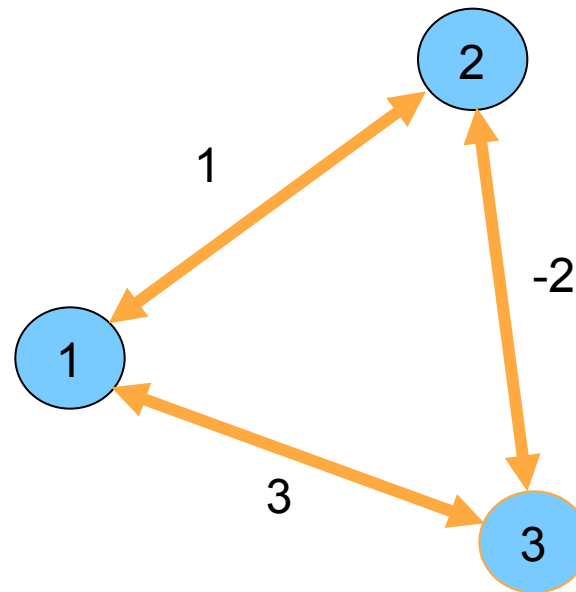
Training

Sometimes you'll see this written

$$\Delta w_{ij} = -\eta \frac{\partial G}{\partial w_{ij}} \text{ where}$$

$$\frac{\partial G}{\partial w_{ij}} = -\frac{1}{T} (p_{ij} - p'_{ij}) = -\frac{1}{T} (\langle s_i s_j \rangle - \langle s'_i s'_j \rangle)$$

A Simple Example



What are the energy values associated with each ‘configuration’ ?



A Simple Example

- We have 8 possible configurations,
- Each has an energy value,
- Each has a probability based on the Boltzmann Distribution at a given temperature
- For a small problem, we can cheat in order to highlight our general approach

A Simple Example

- For each configuration, we calculate:
 - The energy/consensus function.
 - The Boltzmann Distribution
 - Calculate the numerators, sum them up to determine the normalizing Partition Function value, then divide into the numerators to determine the steady-state probability
 - In the real-world, we would run the algorithm at a fixed temperature to estimate the steady-state probabilities

A Simple Example

| | | Temperature = 10 | | | Energy | Boltzmann Numerator |
|---------|--------|------------------|----|----|---|---|
| Weights | Config | x1 | x2 | x3 | | |
| | 1 | 0 | 0 | 0 | 0 | 1 |
| | 2 | 0 | 0 | 1 | 0 | 1 |
| | 3 | 0 | 1 | 0 | 0 | 1 |
| | 4 | 0 | 1 | 1 | 2 |  |
| | 5 | 1 | 0 | 0 | 0 | |
| | 6 | 1 | 0 | 1 |  | |
| | 7 | 1 | 1 | 0 | | |
| | 8 | 1 | 1 | 1 | | |

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

A Simple Example

- Now we can examine the actual, steady-state probabilities, versus the “training set” or the “desired probabilities”.
- Let’s look at a spread sheet where we calculate the values for G and the gradient of G . First, recall

Learning Equations

Akin to an error function:

$$G = \sum_{\alpha} p_{\alpha} \ln \left(\frac{p_{\alpha}}{p'_{\alpha}} \right)$$

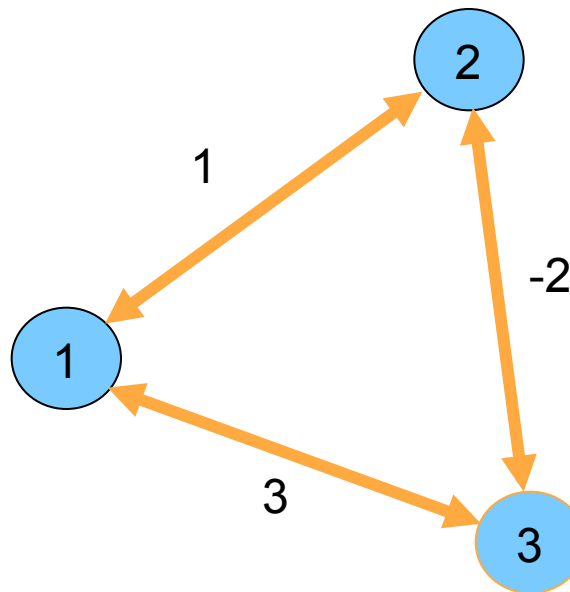
Method of Steepest Descent:

$$\Delta w_{ij} = -\eta \frac{\partial G}{\partial w_{ij}} \text{ where}$$

$$\frac{\partial G}{\partial w_{ij}} = -\frac{1}{T} (p_{ij} - p'_{ij})$$

An Example

Let's say we have the following network with the indicated weights and using Temperature = 1:

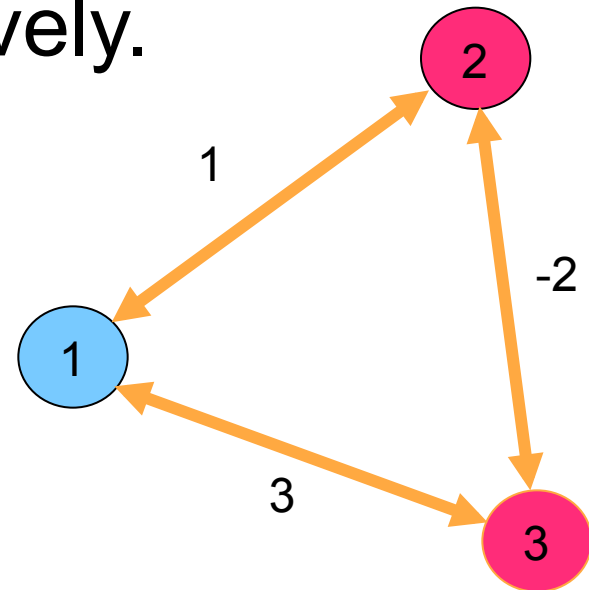


Step 1:

- Assign random initial states:
- Use a 0-1 pseudo-random number generator.
- If number $0 \leq r < 0.5$, assign that node a 0 state.
- If number $0.5 \leq r < 1$, assign that node a 1 state.

Step 1, cont.

- We have 3 random numbers:
- 0.233691, .622338, 901122
- Therefore we assign values of 0, 1 and 1 to nodes 1, 2 and 3 respectively.



Step 2:

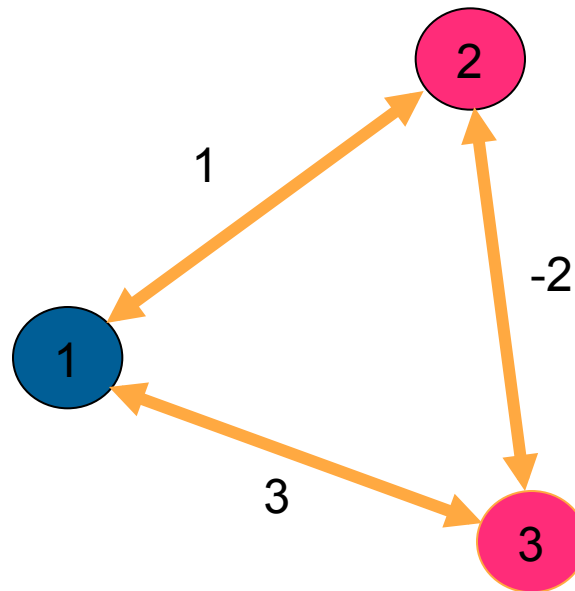
- Select at random, a node to update.
- We'll do this by using a random number generator:
- Since there are 3 nodes, we do the following:
- Obtain a uniform, 0-1 pseudo-random number

Select a Node

- Get a pseudo-random 0-1 number:
0.4468921
- If number $0 \leq r < 0.333333$, select Node 1
- If number $0.333333 \leq r < 0.666666$, select Node 2.
- If number $0.666666 \leq r < 1$, select Node 3

Step 2: Calculate the Activity

- We selected node 2.
- $S_2 = 0 \cdot 1 + -2 \cdot 1 = -2$



Step 3: Assign new state

Given the Activity value, use the sigmoid function to determine a probability of assigning a 0 or 1.

$$x_i = \begin{cases} 1 & \text{w/prob } p_i = \frac{1}{1 + e^{-S_i/T}} \\ 0 & \text{w/prob } 1 - p_i \end{cases}$$

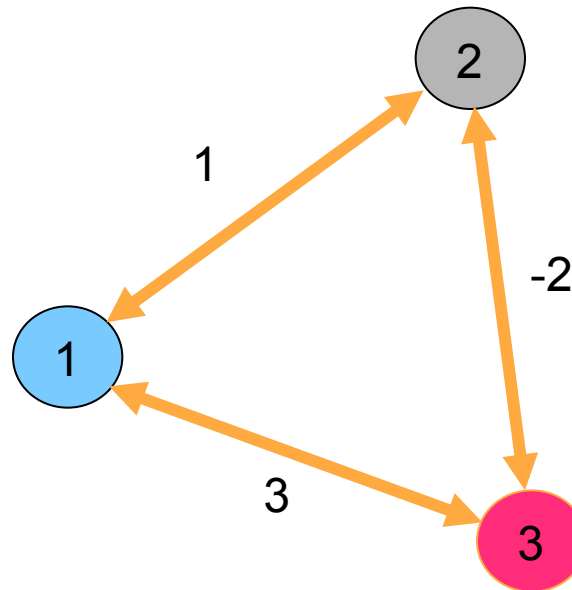
Using this function,

$$\Pr\{x_2 = 1\} = 1/(1 + \exp\{2\}) = 0.11920.$$

Step 3, cont.

- Using a 0-1 pseudo-random number generator,
- assigning a 1 with prob. = 0.11920
- Means if $r \leq 0.11920$, assign 1, otherwise, assign 0.
- $r = 0.314721$, therefore assign a 0.

Our New Network Configuration and we start the process over.



Embellishments

- We could start with a high initial temperature and random initial configuration
- Do many iterations at that temperature.
- Then, lower the temperature and again, do many iterations,
- Then, lower the temperature and again, do many iterations,...
- Eventually, stop.

Or, we could leave the temperature fixed

- In this case, we can compute that relative frequency of each configuration.
- Requires a great deal of computation... many iterations at a given temperature.

Summary

- Showed how the activity function affects the probabilistic state of a node.
- Showed how to ‘anneal’ a network
 - Randomly assigning initial states
 - Stochastically updating states
- Showed how to induce these steady-state probabilities to match a given distribution.