JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Introduction to Neural Networks

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

Dr. 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

$$\alpha = \left[00110100\right]$$

$$\beta = \left[10011010\right]$$

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

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:

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

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

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.

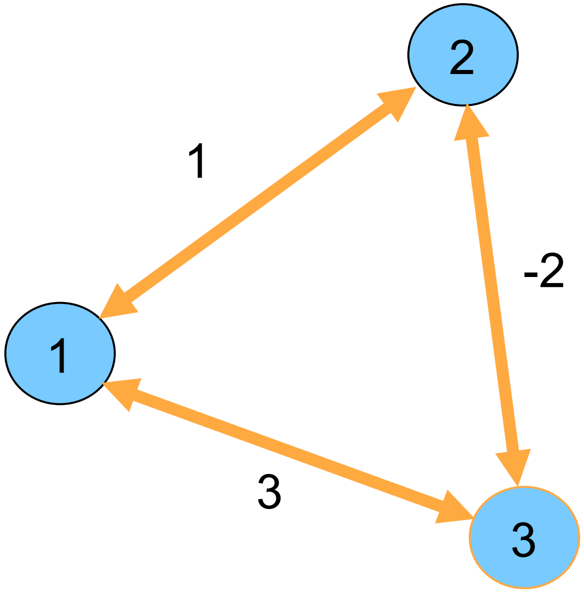
Training

Sometimes you'll see this written

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

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

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			
Weights	1	-2	3		
Config	x1	x2	x3	Energy	Boltzmann Numerator
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:

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

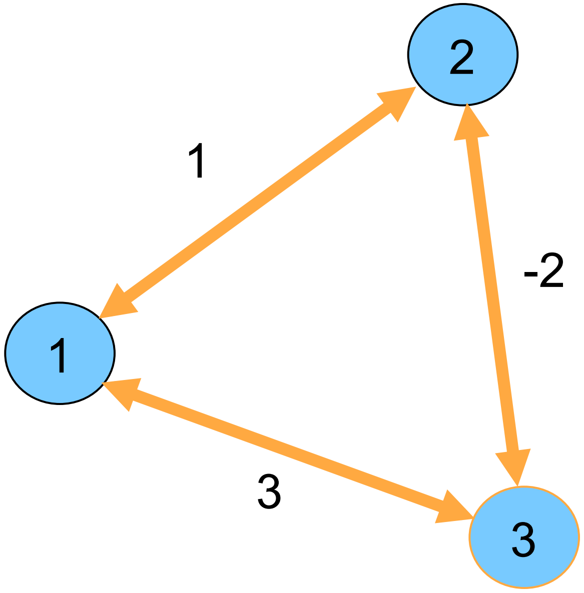
Method of Steepest Descent:

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

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

An Example

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

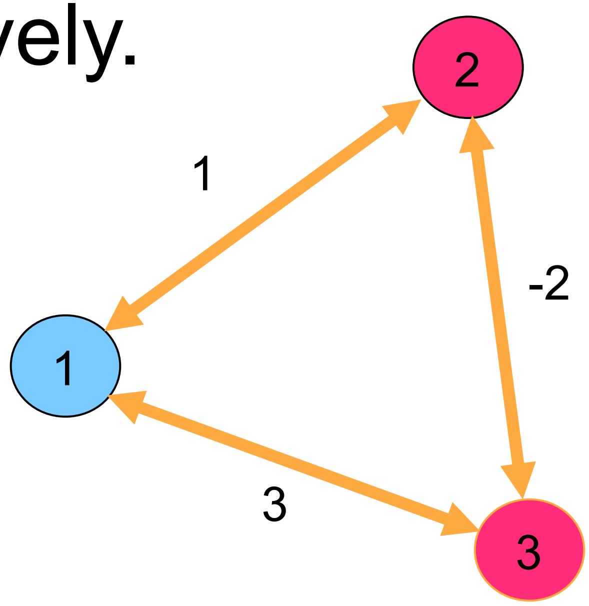


Step 1:

- 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:

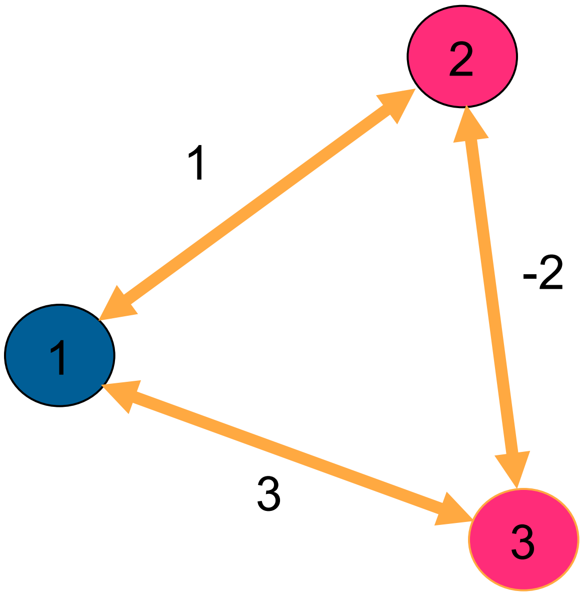
- Select at random, a node to update.
- We'll do this by using a random number generator:
- Since there are 3 nodes, we do the following:
- Obtain a uniform, 0-1 pseudo-random number

Select a Node

- Get a pseudo-random 0-1 number:
  0.4468921
- If number $0 \leq r < 0.33333$, select Node 1
- If number $0.33333 \leq r < 0.66666$, select Node 2.
- If number $0.66666 \leq r < 1$, 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.

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

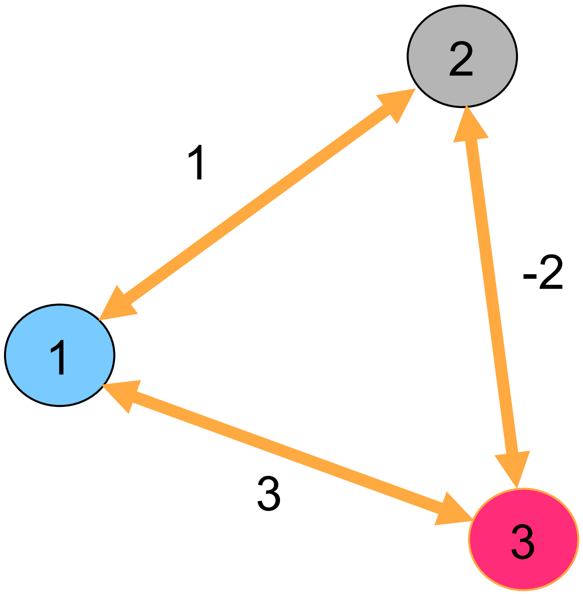
Using this function,

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

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.