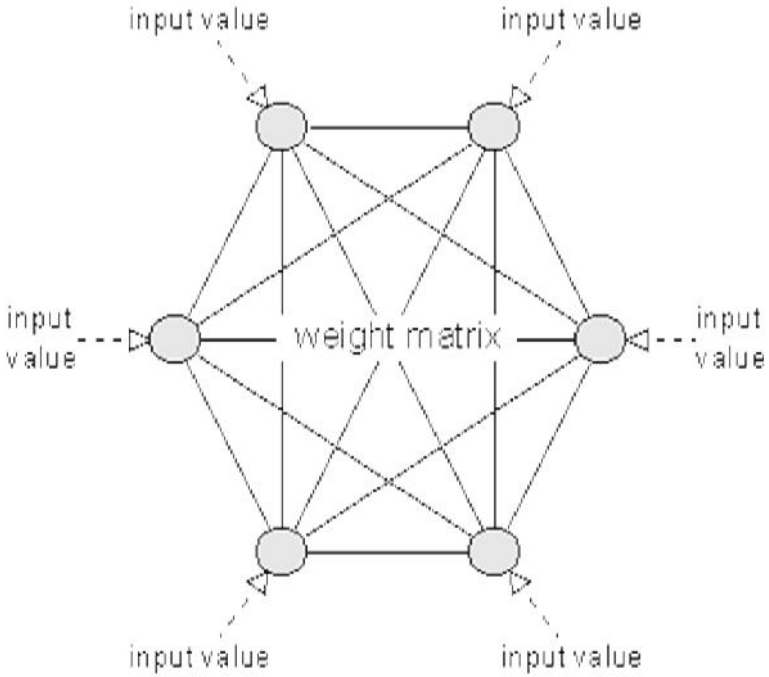
Hopfield networks and Boltzmann machines

Geoffrey Hinton et al.

Presented by Tambet Matiisen

18.11.2014

Hopfield network



Binary units

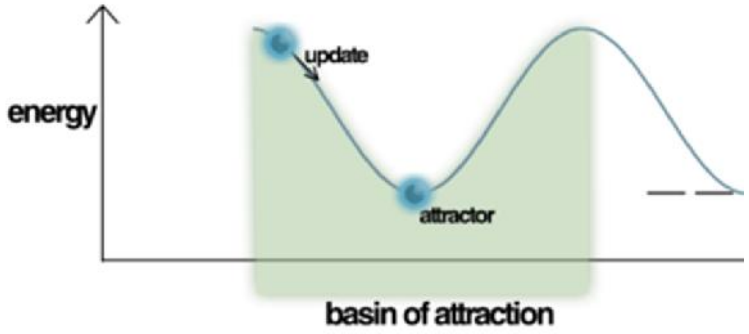
Symmetrical connections

http://www.nnwj.de/hopfield-net.html

Energy function

- The global energy:

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



- The energy gap:

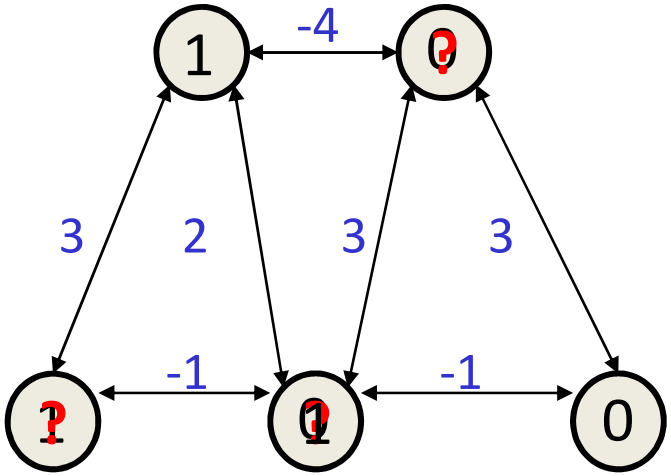
$$\Delta E_i = E(s_i = 0) - E(s_i = 1) = b_i + \sum_j s_j w_{ij}$$

- Update rule:

$$\begin{cases} s_i = 1, \text{if } b_i + \sum_j s_j w_{ij} > 0 \\ s_i = 0, \text{otherwise} \end{cases}$$

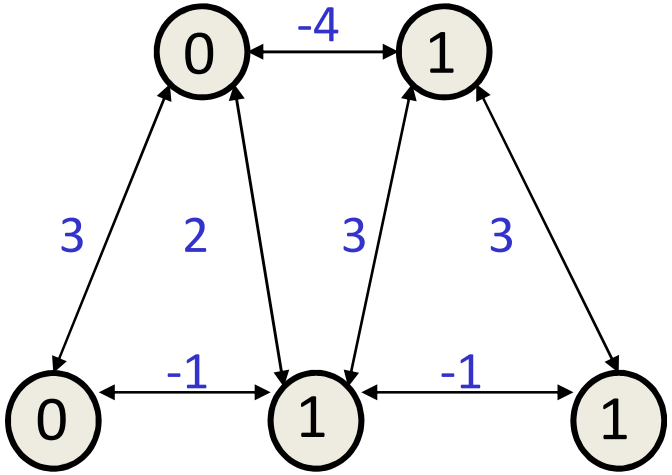
http://en.wikipedia.org/wiki/Hopfield_network

Example



- E = goodness = 4

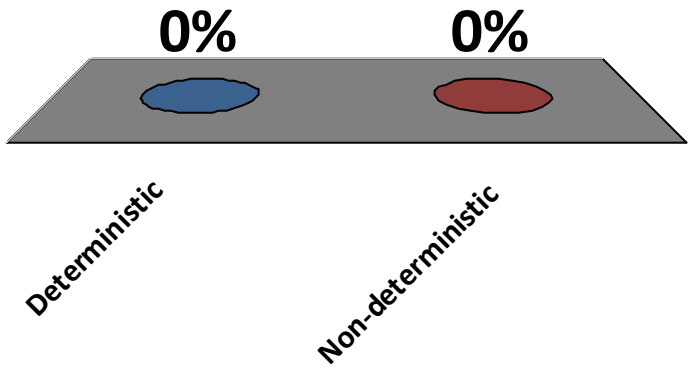
Deeper energy minimum



- E = goodness = 5

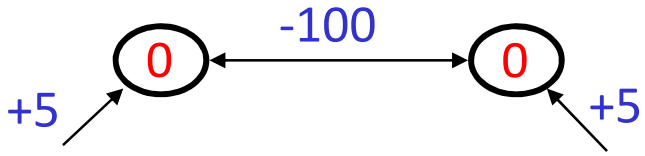
Is updating of Hopfield network deterministic or non-deterministic?

A. Deterministic
B. Non-deterministic



How to update?

- Nodes must be updated sequentially, usually in randomized order.
- With parallel updating energy could go up.

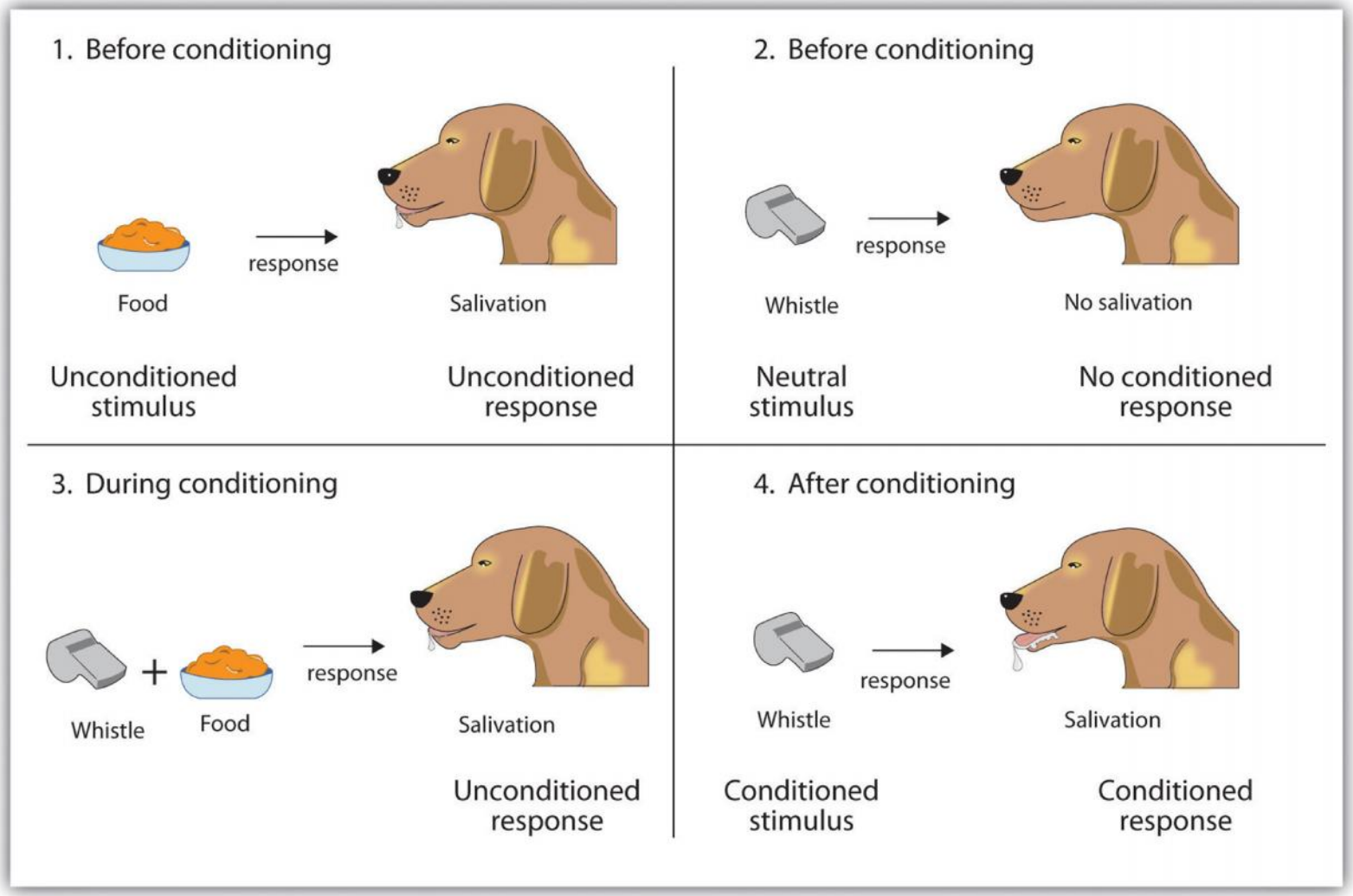


- If updates occur in parallel but with random timing, the oscillations are usually destroyed.

Content-addressable memory

- Using energy minima to represent memories gives a **content-addressable memory**.
  - An item can be accessed by just knowing part of its content.
  - Can fill out missing or corrupted pieces of information.
  - It is robust against hardware damage.

Classical conditioning



http://changecom.wordpress.com/2013/01/03/classical-conditioning/

Storing memories

- Energy landscape is determined by weights!

- If we use activities -1 and 1:

$$\Delta w_{ij} = s_i s_j$$

$$\begin{cases} \text{if } s_i = s_j \text{ then } \Delta w_{ij} = 1 \\ \text{if } s_i \neq s_j \text{ then } \Delta w_{ij} = -1 \end{cases}$$

- If we use states 0 and 1:

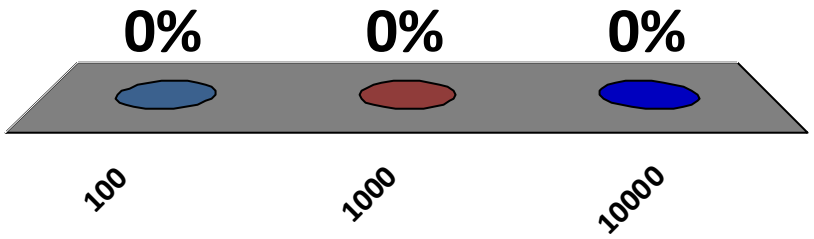
$$\Delta w_{ij} = 4\,(s_i - \tfrac{1}{2})(s_j - \tfrac{1}{2})$$

Demo

- http://www.tarkvaralabor.ee/doodler/
- (choose Algorithm: Hopfield and Initialize)

How many weights the example had?

A. 100
B. 1000
C. 10000

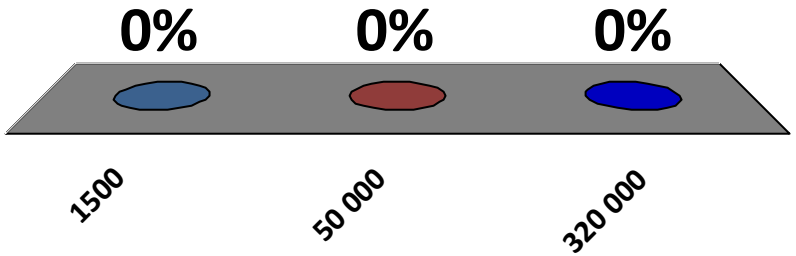


Storage capacity

- The capacity of a totally connected net with N units is only about 0.15 * N memories.
  - With N bits per one memory this is only 0.15 * N * N bits.
- The net has N^2 weights and biases.
- After storing M memories, each connection weight has an integer value in the range [–M, M].
- So the number of bits required to store the weights and biases is: $N^2 \log(2M+1)$

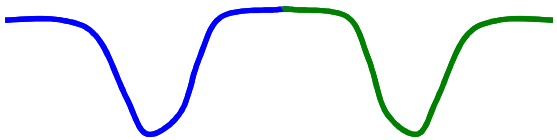
How many bits are needed to represent weights in the example?

A. 1500
B. 50 000
C. 320 000

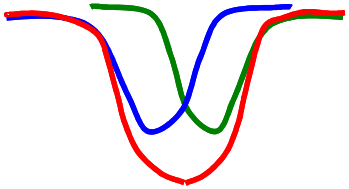


Spurious minima

- Each time we memorize a configuration, we hope to create a new energy minimum.

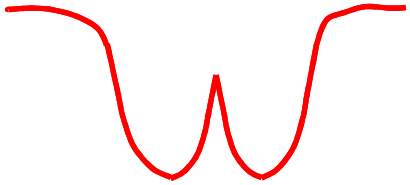


- But what if two minima merge to create a minimum at an intermediate location?



Reverse learning

- Let the net settle from a random initial state and then do unlearning.



- This will get rid of deep, spurious minima and increase memory capacity.

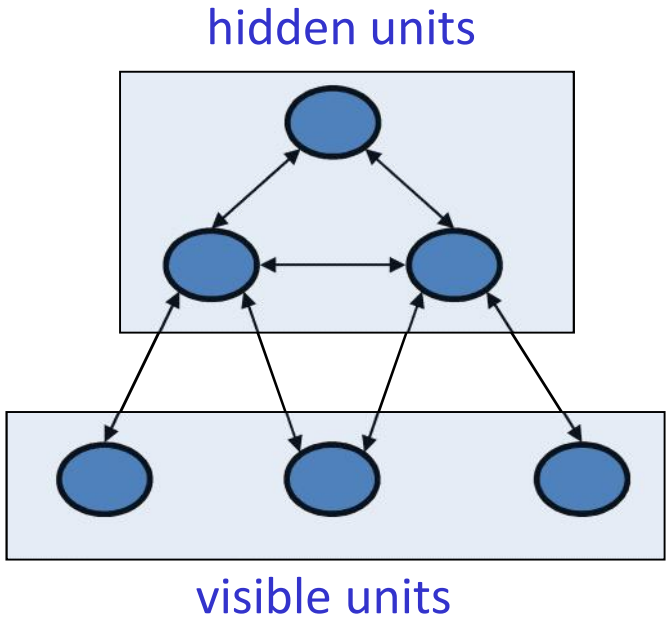
Increasing memory capacity

- Instead of trying to store vectors in one shot, cycle through the training set many times.
- Use the perceptron convergence procedure to train each unit to have the correct state given the states of all the other units in that vector.

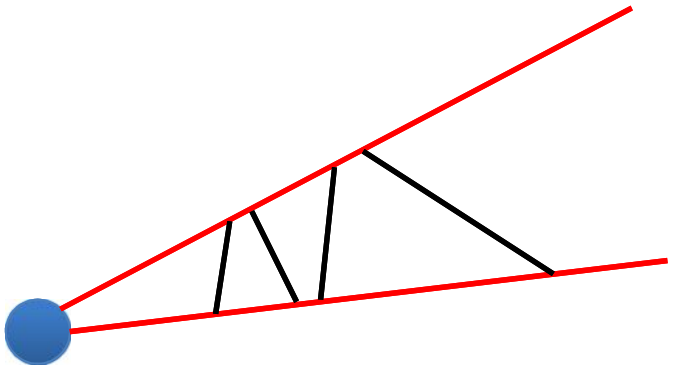
$$x_i' = f\left(\sum_{j \neq i} x_j w_{ij}\right) \qquad \Delta w_{ij} = (x_i - x_i')x_j$$

Hopfield nets with hidden units

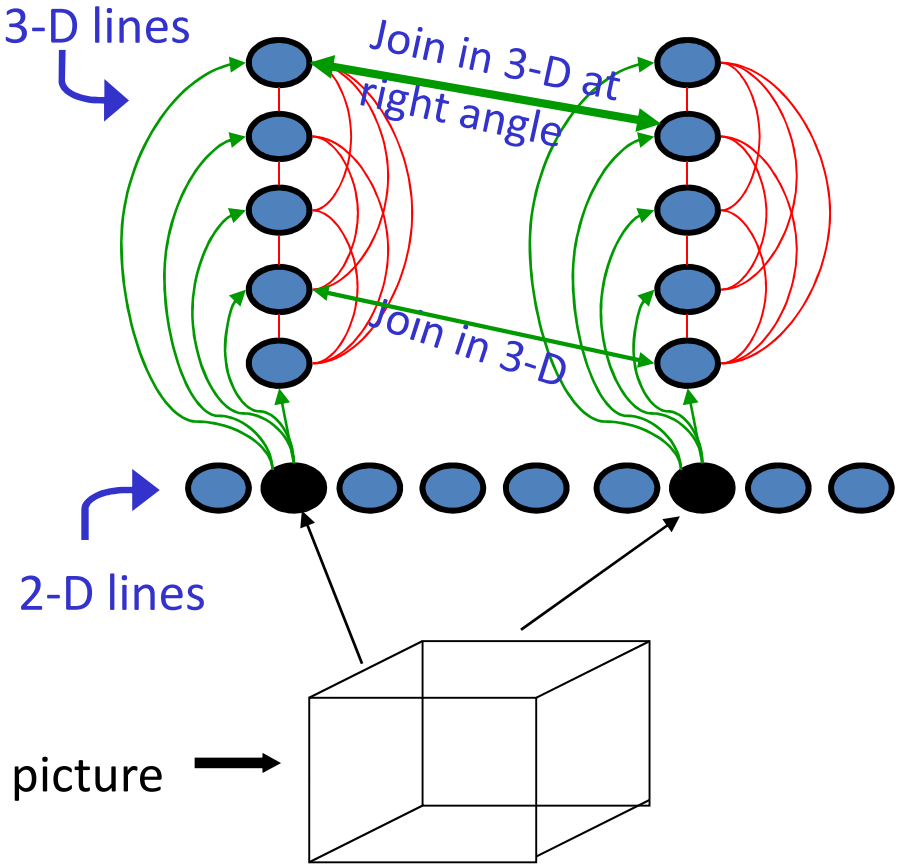
- Instead of using the net to store memories, use it to construct interpretations of sensory input.
  - The input is represented by the visible units.
  - The interpretation is represented by the states of the hidden units.
  - The badness of the interpretation is represented by the energy.



3D edges from 2D images

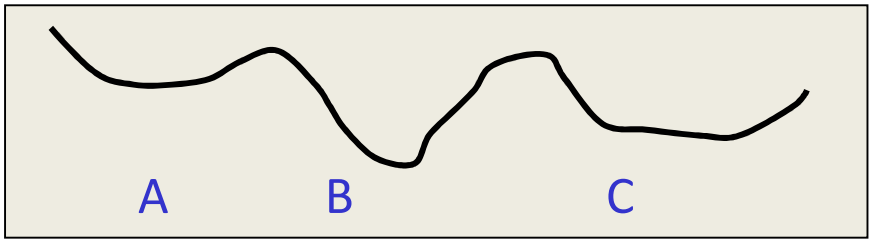


You can only see one of these 3-D edges at a time because they occlude one another.



Noisy networks

- Hopfield net tries reduce the energy at each step.
  - This makes it impossible to escape from local minima.
- We can use random noise to escape from poor minima.
  - Start with a lot of noise so its easy to cross energy barriers.
  - Slowly reduce the noise so that the system ends up in a deep minimum. This is “simulated annealing”.

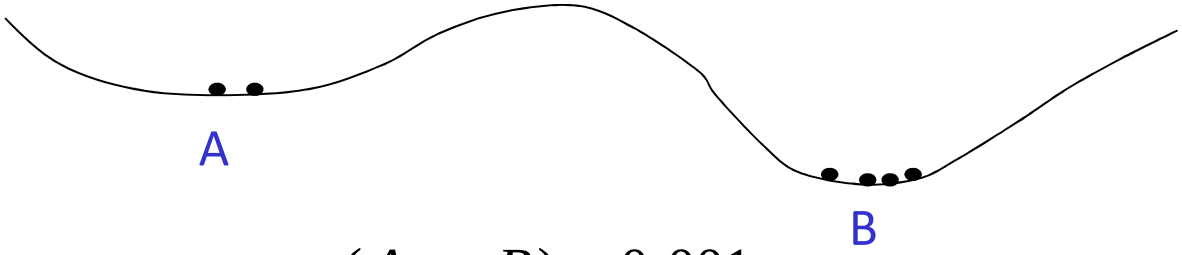


Temperature

High temperature transition probabilities

$$p(A \rightarrow B) = 0.2$$

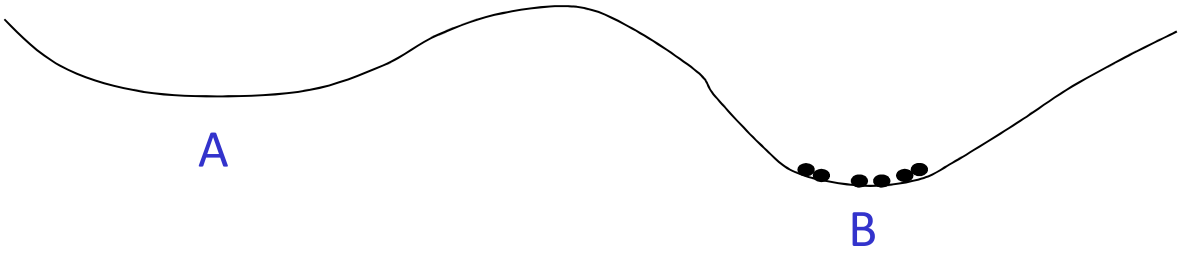
$$p(A \leftarrow B) = 0.1$$



Low temperature transition probabilities

$$p(A \rightarrow B) = 0.001$$

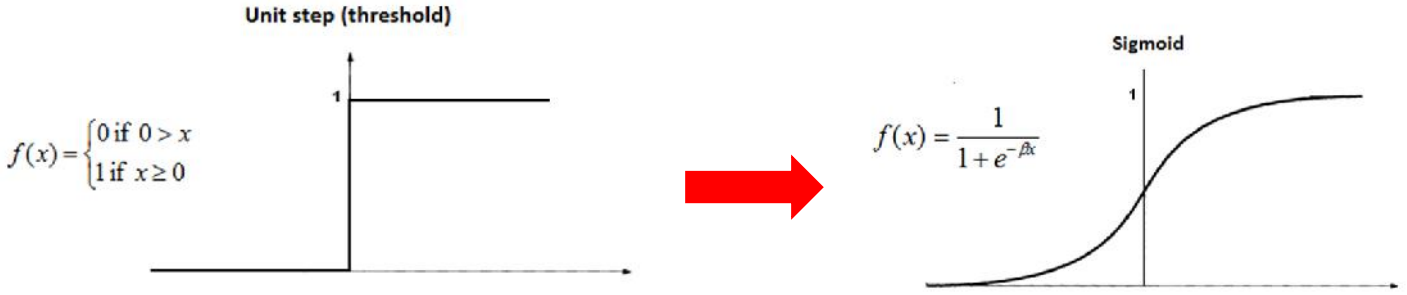
$$p(A \leftarrow B) = 0.000001$$



Stochastic binary units

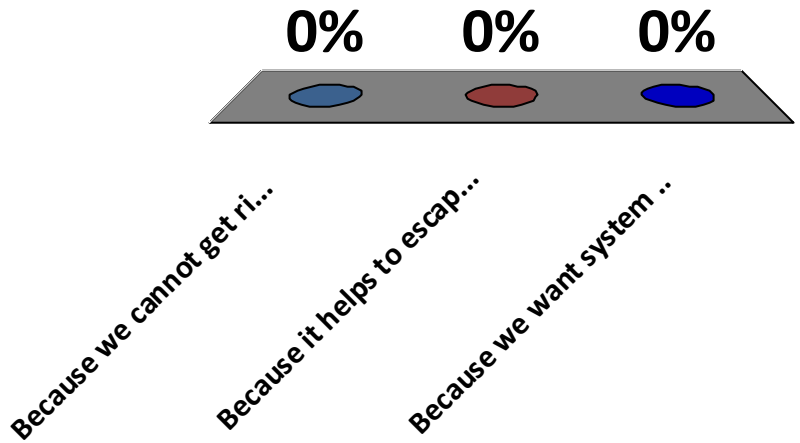
- Replace the binary threshold units by binary stochastic units that make biased random decisions.
- The "temperature" controls the amount of noise.
- Raising the noise level is equivalent to decreasing all the energy gaps between configurations.

$$p(s_i{=}1) = \frac{1}{1+e^{-\Delta E_i/T}}$$ ← temperature



Why we need stochastic binary units?

A. Because we cannot get rid of inherent noise.
B. Because it helps to escape local minima.
C. Because we want system to produce randomized results.



Thermal equilibrium

- Thermal equilibrium is a difficult concept!
  - Reaching thermal equilibrium does not mean that the system has settled down into the lowest energy configuration.
  - The thing that settles down is the **probability distribution** over configurations.
  - This settles to the stationary distribution.
  - Any given system keeps changing its configuration, but the fraction of systems in each configuration does not change.

Modeling binary data

- Given a training set of binary vectors, fit a model that will assign a probability to every possible binary vector.
- Model can be used for generating data with the same distribution as original data.
- If particular model (distribution) produced the observed data:

$$p(model_i \mid data) = \frac{p(data \mid model_i)\, p(model_i)}{\sum_j p(data \mid model_j)}$$

Boltzmann machine

- ...is defined in terms of the energies of joint configurations of the visible and hidden units.
- Probability of joint configuration:

$$p(\mathbf{v}, \mathbf{h}) \propto e^{-E(\mathbf{v}, \mathbf{h})}$$

- The probability of finding the network in that joint configuration after we have updated all of the stochastic binary units many times.

Energy of a joint configuration

binary state of unit i in **v**

bias of unit k

$$-E(\mathbf{v},\mathbf{h}) = \sum_{i \in vis} v_i b_i + \sum_{k \in hid} h_k b_k + \sum_{i<j} v_i v_j w_{ij} + \sum_{i,k} v_i h_k w_{ik} + \sum_{k<l} h_k h_l w_{kl}$$

Energy with configuration **v** on the visible units and **h** on the hidden units

indexes every non-identical pair of i and j once

weight between visible unit i and hidden unit k

From energies to probabilities

- The probability of a joint configuration over both visible and hidden units depends on the energy of that joint configuration compared with the energy of all other joint configurations.

$$p(\mathbf{v}, \mathbf{h}) = \frac{e^{-E(\mathbf{v}, \mathbf{h})}}{\sum_{\mathbf{u}, \mathbf{g}} e^{-E(\mathbf{u}, \mathbf{g})}}$$

partition function

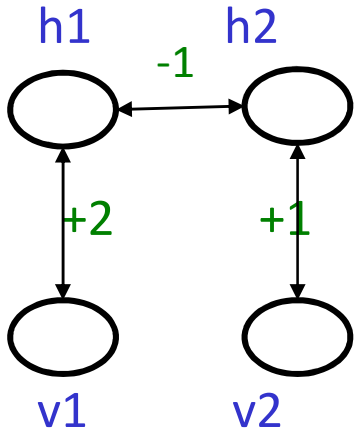
- The probability of a configuration of the visible units is the sum of the probabilities of all the joint configurations that contain it.

$$p(\mathbf{v}) = \frac{\sum_{\mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h})}}{\sum_{\mathbf{u}, \mathbf{g}} e^{-E(\mathbf{u}, \mathbf{g})}}$$

Example

$\mathbf{v}$	$\mathbf{h}$	$-E$	e^{-E}	$p(\mathbf{v}, \mathbf{h})$	$p(\mathbf{v})$
1 1	1 1	2	7.39	.186	0.466
1 1	1 0	2	7.39	.186	
1 1	0 1	1	2.72	.069	
1 1	0 0	0	1	.025	
1 0	1 1	1	2.72	.069	0.305
1 0	1 0	2	7.39	.186	
1 0	0 1	0	1	.025	
1 0	0 0	0	1	.025	
0 1	1 1	0	1	.025	0.144
0 1	1 0	0	1	.025	
0 1	0 1	1	2.72	.069	
0 1	0 0	0	1	.025	
0 0	1 1	-1	0.37	.009	0.084
0 0	1 0	0	1	.025	
0 0	0 1	0	1	.025	
0 0	0 0	0	1	.025	
			39.70		

An example of how weights define a distribution



Getting a sample from model

- We cannot compute the normalizing term (the partition function) because it has exponentially many terms.
- So we use Markov Chain Monte Carlo to get samples from the model starting from a random global configuration:
  - Keep picking units at random and allowing them to stochastically update their states based on their energy gaps.
  - Run the Markov chain until it reaches its stationary distribution
- The probability of a global configuration is then related to its energy by the Boltzmann distribution.

Getting a sample from the posterior distribution for a given data vector

- The number of possible hidden configurations is exponential so we need MCMC to sample from the posterior.
  - It is just the same as getting a sample from the model, except that we keep the visible units clamped to the given data vector.
  - Only the hidden units are allowed to change states
- Samples from the posterior are required for learning the weights. Each hidden configuration is an "explanation" of an observed visible configuration. Better explanations have lower energy.

What does Boltzmann machine really do?

A. Models probability distribution of input data.
B. Generates samples from modeled distribution.
C. Learns probability distribution of input data from samples.
D. All of above.