

10 January, 2017

# Neural Networks

---

Course 12: Boltzmann Machines

# Overview

---

- ▶ Hopfield Nets (brief)
- ▶ Boltzmann Machine
- ▶ Learning on Boltzmann Machines
- ▶ Restricted Boltzmann Machines

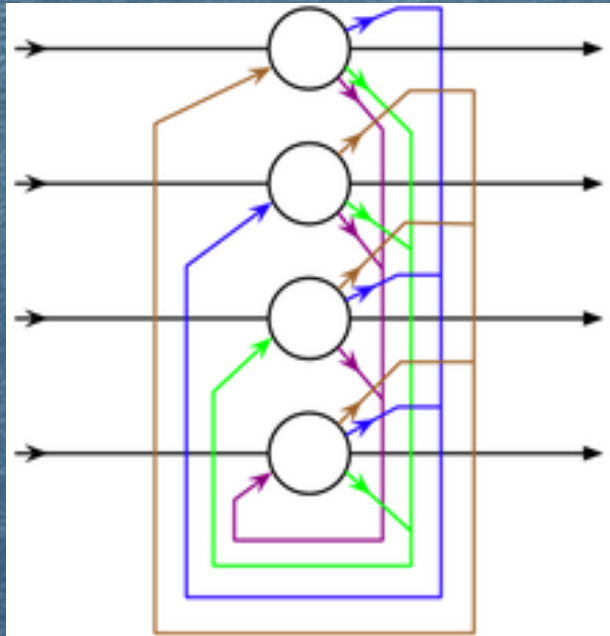


# Hopfield Net (brief)

---

# Hopfield Net (brief)

---

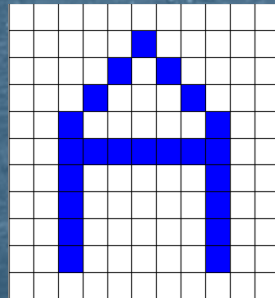
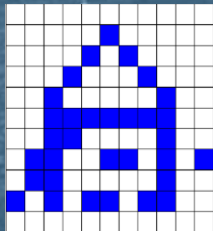
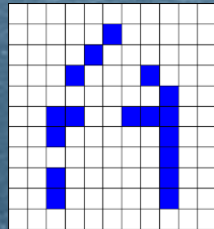
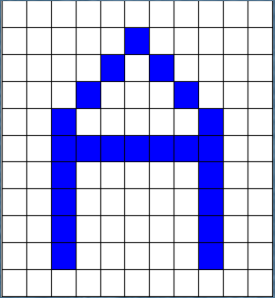


- ▶ It is composed of only one layer of units, binary threshold neurons, where each unit is symmetrically connected with all the other units (except itself).
- ▶ So each unit acts as input for other units, but also as output
- ▶ Each unit has an output value, known as its binary state, that is usually -1 or 1 (can also be 0 or 1)



# Hopfield Net (brief)

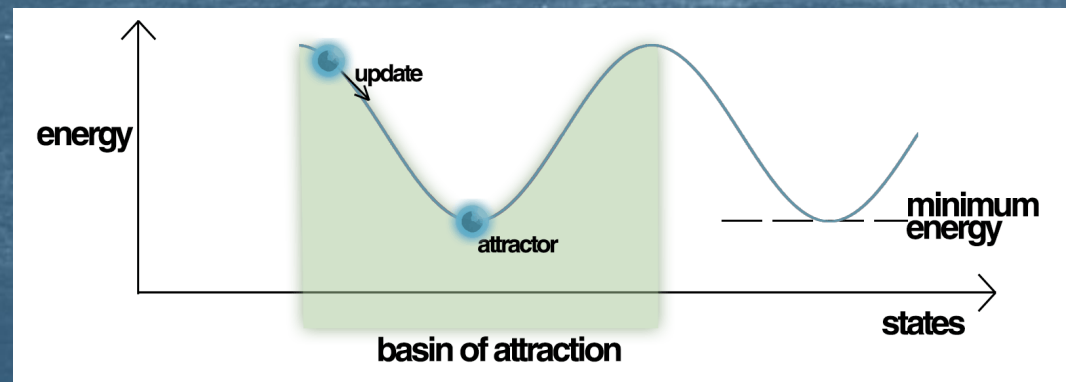
---



- ▶ The Hopfield net is considered autoassociative, meaning that it learns to map a noisy input to a learned pattern
- ▶ As opposed to indexed memory, associative memory only needs some parts to retrieve a stored item.
- ▶ For example, if the letter A was memorized, than you only need a subset of the bits to retrieve it

# Hopfield Net (brief)

- ▶ A Hopfield net is a type of recurrent artificial neural network that assigns an Energy value to each of its configuration.
- ▶ Starting from an arbitrary configuration the network will converge to a low energy minimum by asynchronously changing one unit at a time in respect to the rest of the units
- ▶ The network is trained such that configurations that correspond to training data (memories) have low energies





# Hopfield net (brief)

---

- ▶ Global Energy:

- ▶ The global energy is the sum of many contributions, computed locally

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

where

$s_i$  = activation of unit  $i$

$w_{ij}$  = the symmetric weight between unit  $i$  and unit  $j$

- ▶ So, each unit affects the global energy. The way it affects it, can be computed locally:

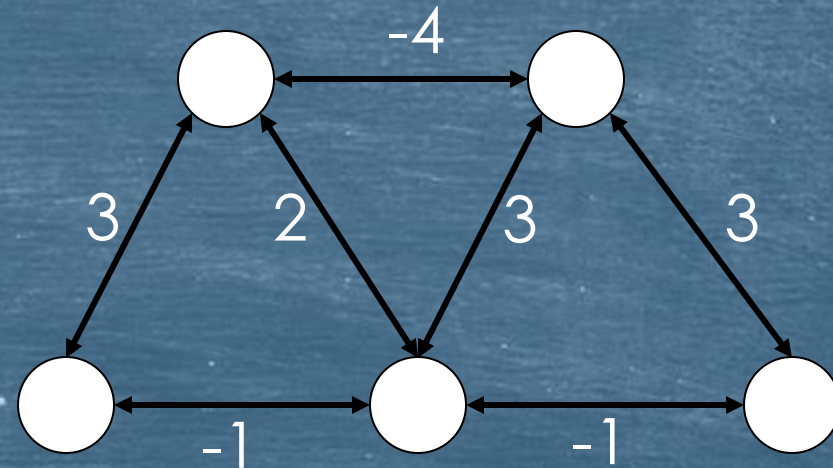
(Energy gap)  $\Delta E_i = E(s_i = 0) - E(s_i = 1) = b_i + \sum_j s_j w_{ij}$



# Hopfield net (brief)

Finding the Energy minimum:

- Start from a random state
- Update units in random order, one at a time, to the state that gives the lowest global energy
- Stop when there is no unit that changes its output

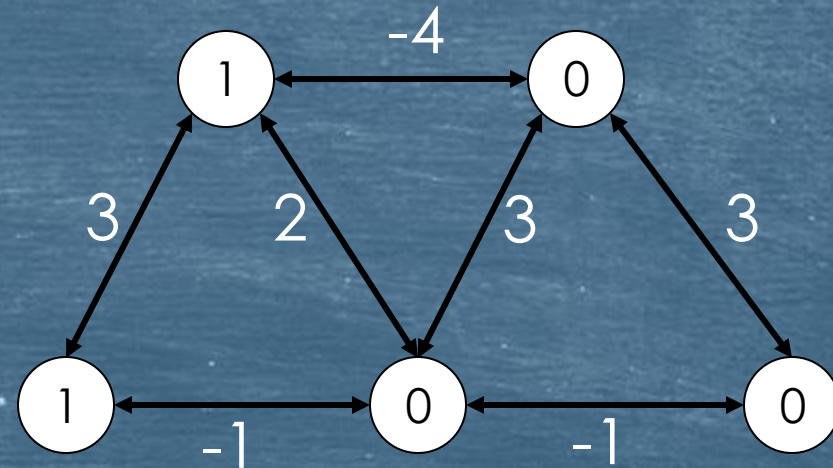




# Hopfield net (brief)

Finding the Energy minimum:

- Start from a random state
- Update units in random order, one at a time, to the state that gives the lowest global energy
- Stop when there is no unit that changes its output



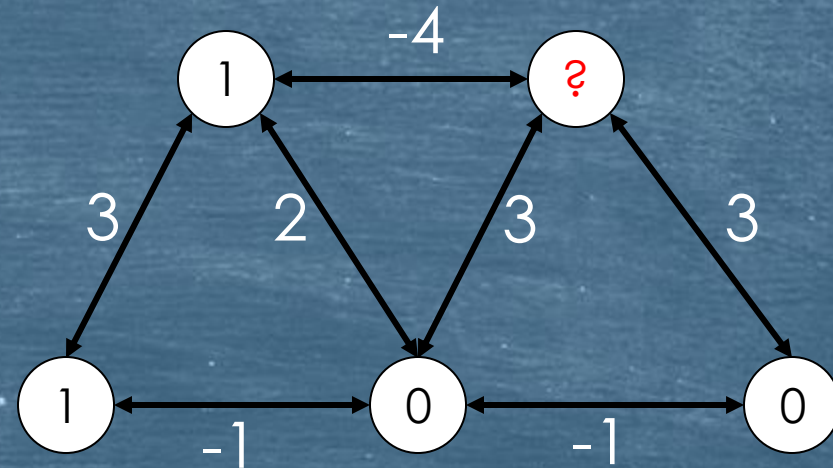
$$-E = \text{goodness} = 3$$



# Hopfield net (brief)

Finding the Energy minimum:

- Start from a random state
- Update units in random order, one at a time, to the state that gives the lowest global energy
- Stop when there is no unit that changes its output



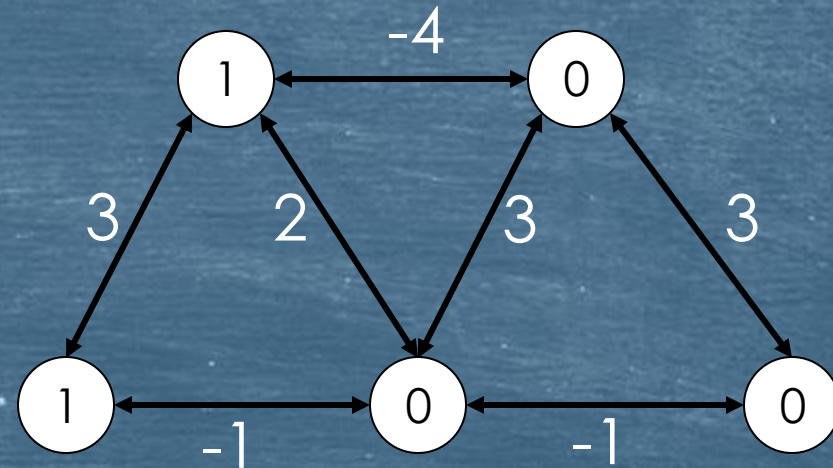
$$-E = \text{goodness} = 3$$



# Hopfield net (brief)

Finding the Energy minimum:

- Start from a random state
- Update units in random order, one at a time, to the state that gives the lowest global energy
- Stop when there is no unit that changes its output



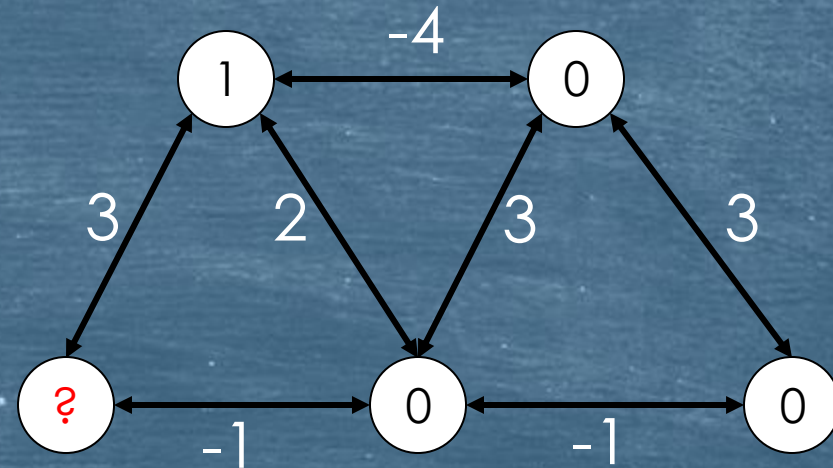
$$-E = \text{goodness} = 3$$



# Hopfield net (brief)

Finding the Energy minimum:

- Start from a random state
- Update units in random order, one at a time, to the state that gives the lowest global energy
- Stop when there is no unit that changes its output



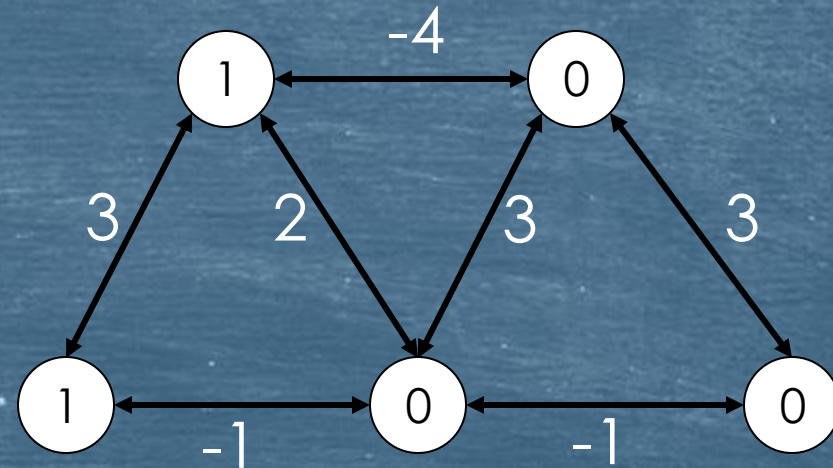
$$-E = \text{goodness} = 3$$



# Hopfield net (brief)

Finding the Energy minimum:

- Start from a random state
- Update units in random order, one at a time, to the state that gives the lowest global energy
- Stop when there is no unit that changes its output



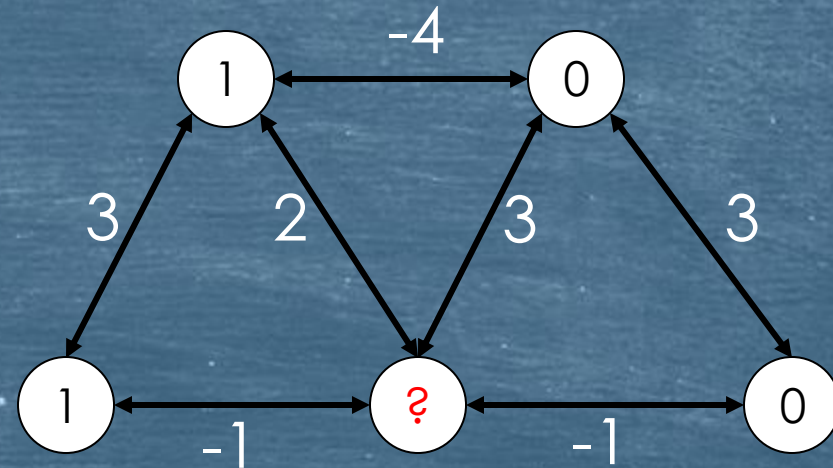
$$-E = \text{goodness} = 3$$



# Hopfield net (brief)

Finding the Energy minimum:

- Start from a random state
- Update units in random order, one at a time, to the state that gives the lowest global energy
- Stop when there is no unit that changes its output



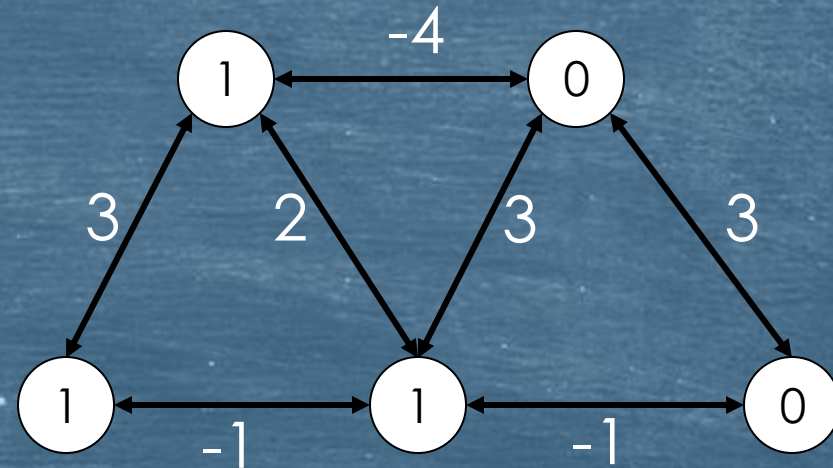
$$-E = \text{goodness} = 3$$



# Hopfield net (brief)

Finding the Energy minimum:

- Start from a random state
- Update units in random order, one at a time, to the state that gives the lowest global energy
- Stop when there is no unit that changes its output
- This is a local minimum. Any unit that we probe, will not change its activation



$$-E = \text{goodness} = 4$$



# Hopfield net (brief)

---

- Considering the equation of Energy (no biases)

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

- The network has the lowest energy (given a pattern  $s_i^p$ ) when:
  - Units with same activation are linked with a positive weight
  - Units with different activation are linked with a negative weight
- Given these observations, if units have activations of  $(-1,1)$  the weights will be adjusted in the following way:

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

- If neurons have activation of  $(0,1)$ , then equation changes to

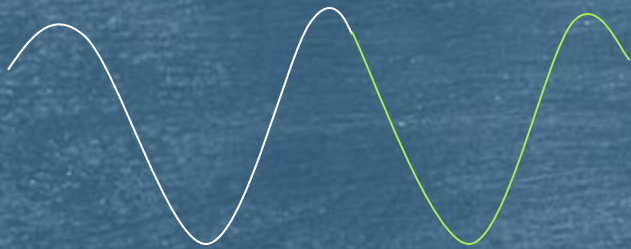
$$\Delta w_{ij} = \frac{1}{4} (s_i - \frac{1}{2})(s_j - \frac{1}{2})$$



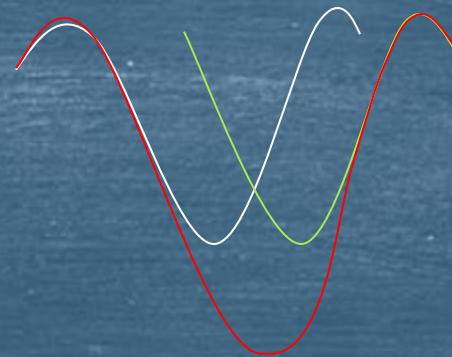
# Hopfield net (brief)

---

- The main problem of Hopfield nets is spurious minima.
- Spurious minima represents the situation where the network settles to a local minimum that does not represents a learned pattern
- This happens when to patterns, represented by two nearby minima configurations, merge together to create a new minima that has an even lower energy



Trained patterns



Trained patterns with spurious minima



# Hopfield net (brief)

---

- Solution for spurious minima:
- An interesting solution that was proposed by Hopfield et. al, was to use unlearning:
  - Start from an initial random state
  - Let the network reach a minimum (which is probable to be a spurious pattern)
  - Do the opposite of learning.
    - $\Delta w_{ij} = -s_i s_j$

The authors showed that this worked, but had no analysis. For example, how much unlearning should we do?

Other authors that have tested this idea found that the network ends up working erratically (Cristos 1994)



# Boltzmann Machine

---

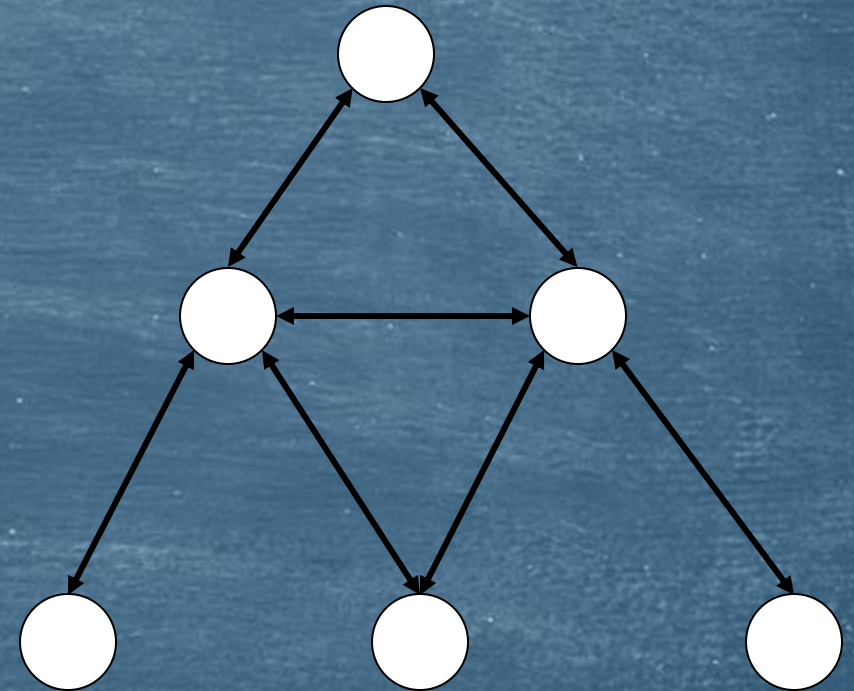


# Boltzmann Machine

---

Boltzmann Machines are similar to Hopfield nets:

- Network of binary units, connected together by symmetric connections
- There's the same energy function
- The unit update rule is similar (depends on the other units)

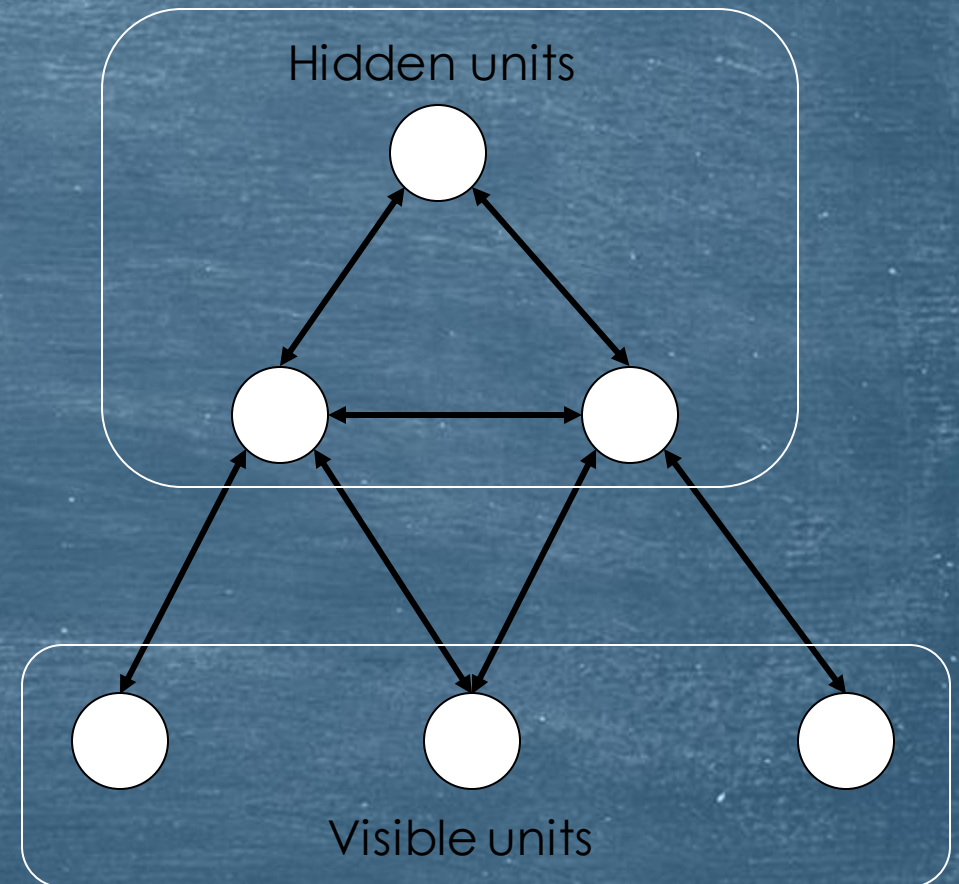




# Boltzmann Machine

Boltzmann Machines are different from Hopfield Nets:

- Instead of storing memories, the purpose of the Boltzmann Machine is to find a representation of the input data
- For this, there is another layer (of hidden units) that aren't set by the data, but are influenced by it (through weights)
- The badness of the interpretation is represented by the energy (next slide)
- Using the representation, the network should be able to reconstruct the data
- We're interested in achieving global minima as opposed to local minima in Hopfield nets





# Boltzmann Machine

- The energy of a configuration  $(v, h)$  changes to

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

Bias terms  
for visible  
units

Bias terms  
for hidden  
units

Interaction  
between  
visible units

Interaction  
between  
hidden units

Interaction  
between  
visible and  
hidden units



# Boltzmann Machine: thermodynamic concepts

---

- The idea of the network is based on Boltzmann distribution (Gibbs distribution, 1868), from thermodynamics.
- It gives the probability that a system will be in a certain state as a function of that state's energy (and temperature)

$$p_i = \frac{e^{-\frac{E_i}{T}}}{\sum_{j=1}^N e^{-\frac{E_j}{T}}}$$

← Energy is proportional to  $e^{-\frac{E_i}{T}}$

← The sum over the energies of all the states. Used to make it a probability.

This is called partition function



# Boltzmann Machine: thermodynamic concepts

---

- Analogy with a thermodynamic system
- Think of a glass of water:



It has a volume, a pressure, a temperature, etc.

These are all macro states of the system and are measurable because the system is in (thermal) equilibrium (they don't change)

However, these are given by the particles which collide with each other, bounce in the glass, release heat, etc.



# Boltzmann Machine: thermodynamic concepts

---

- Each particle has some kind of energy (kinetic, rotational, etc.). The system's energy is made of the sum of all energies of the particles plus the energy of their interaction.
- The way these atoms are distributed at a certain time in the system determines a microstate
- So, even though the system is in equilibrium (at a macro state ), the microstates change very often and so does the energy of the system at that time.
- What Boltzmann discovered is that the probability of the system to be in a certain state is proportional to the exponential of the state's energy

$$p_i \propto e^{-\frac{E_i}{T}}$$



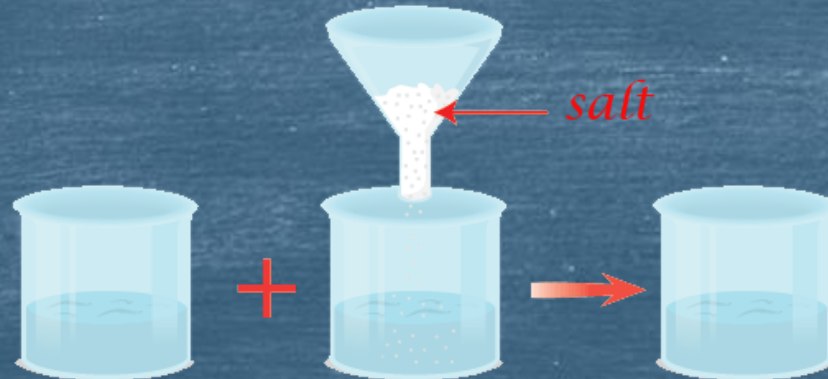
# Boltzmann Machine: thermodynamic concepts

---

- The Boltzmann distribution is true at thermal equilibrium

Imagine that salt is dropped in to the glass:

- The temperature, volume, pressure starts to change
- As the salt dissolves in the water the macro states of the system will change
- Eventually, the system will arrive at thermal equilibrium and the above measurements will remain constant. Only then the Boltzmann distribution will be valid.





# Boltzmann Machine

---

- Think of the neural network as being the glass of water (system).
  - Each unit is a particle.
  - The units have energy (0 or 1) and the way they interact (weights) make up the energy of the system (the network).
  - Since we want the interpretation for the data to be the most probable, given the Boltzmann distribution, it means that the configuration of the units must correspond to the lowest energy.
  - This must be computed at thermal equilibrium.
- Several modification must be made to the Hopfield net so that the Boltzmann distribution holds true.



# Boltzmann Machine

---

- The units must be made stochastic:
  - On Hopfield nets, the units were turned on/off based on the energy gap:

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

- On the Boltzmann Machine, the unit will turn according to a probability based on the energy gap:

$$p_i = \text{on} = \frac{1}{1 + e_i^{-\frac{\Delta E}{T}}}$$

how the formula was obtained: [https://en.wikipedia.org/wiki/Boltzmann\\_machine](https://en.wikipedia.org/wiki/Boltzmann_machine)

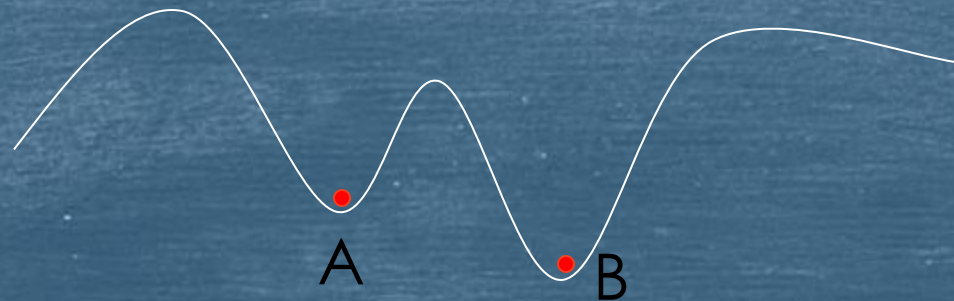
- Observe, that if  $T=0$ , we have a Hopfield Net. If  $T$  is very big (infinite), then  $p_i = \frac{1}{2}$ . We will use  $T=1$ .



# Boltzmann Machine

---

- So a Boltzmann Machine is a stochastic Hopfield network with hidden units.
- Using stochastic units can also be viewed as a way to escape from local minima:
  - Since Hopfield net always drives the energy down (or remains constant) it is impossible to escape from local minima.
  - The temperature ( $T$ ) can be viewed as a way of flattening the Energy landscape





# Boltzmann Machine

---

- Thermal Equilibrium:
- The Boltzmann Distribution can only be used if the system is on thermal equilibrium.
- Thermal equilibrium doesn't mean that the units will not change the value anymore. It actually means that the probability distribution of the configurations has settled down.
- Since the system can be viewed as a Markov Chain, it has a stationary distribution (most do). That means that the probability of going from one state (configuration) to another doesn't change anymore.



# Boltzmann Machine: example stationary distribution

---

- Insurance companies classify each driver to high risk or low risk if they got a speeding ticket during the year.
- The probability of a high risk driver turning low risk on the next year is 0.4 (so remaining high risk is 0.6)
- The probability of a low risk driver turning high risk on the next year is 0.15 (so remaining low risk is 0.85)
- If in the initial population 10% are high risk drivers? How would the distribution look for the next year? (what about 10, 25 or 50 years ?)



# Boltzmann Machine: example stationary distribution

---

Initial state:  $\begin{matrix} & H & L \\ & [0.1, 0.9] \end{matrix}$

Transition Matrix:  $\begin{matrix} & H & L \\ H & [0.6 & 0.4] \\ L & [0.15 & 0.85] \end{matrix}$

After the first year, the distribution of high/low risk drivers is:

$$[0.1, 0.9] \begin{bmatrix} 0.6 & 0.4 \\ 0.15 & 0.85 \end{bmatrix} = [0.195, 0.805]$$

After the second year, the distribution of high/low risk driver is:

$$[0.195, 0.805] \begin{bmatrix} 0.6 & 0.4 \\ 0.15 & 0.85 \end{bmatrix} = [0.23775, 0.76225]$$



# Boltzmann Machine: example stationary distribution

---

- 5-th year :  $[0.1, 0.9] \begin{bmatrix} 0.6 & 0.4 \\ 0.15 & 0.85 \end{bmatrix}^5 = [0.26953, 0.73046]$
- 10-th year:  $[0.1, 0.9] \begin{bmatrix} 0.6 & 0.4 \\ 0.15 & 0.85 \end{bmatrix}^{10} = [0.27267, 0.72733]$
- 20-th year:  $[0.1, 0.9] \begin{bmatrix} 0.6 & 0.4 \\ 0.15 & 0.85 \end{bmatrix}^{20} = [0.27272, 0.72727]$
- 25-th year:  $[0.1, 0.9] \begin{bmatrix} 0.6 & 0.4 \\ 0.15 & 0.85 \end{bmatrix}^{25} = [0.27272, 0.72727]$
- 50-th year:  $[0.1, 0.9] \begin{bmatrix} 0.6 & 0.4 \\ 0.15 & 0.85 \end{bmatrix}^{50} = [0.27272, 0.72727]$

The Markov chain has reached its stationary distribution. It does not change anymore



# Boltzmann Machine

---

- Thermal Equilibrium:
  - Think that you have many Boltzmann machines with the same weights.
  - They could all start in the same configuration or for each configuration there will be the same number of systems (uniform distribution)
  - On every unit, of every system, we pick units at random, compute the energy gap and turn the unit on or off probabilistically based on the energy gap
  - After an amount of time (iterations), the fraction of systems in each configuration will remain constant. Each system may change its configuration, but the fraction of systems in each configuration will not change.

This is because there are many more systems than configurations.



# Boltzmann Machine

---

- ▶ Given the way the Energy function is defined and given the Boltzmann distribution, the probability of a joint configuration  $(v, h)$  is proportional to it's energy:

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

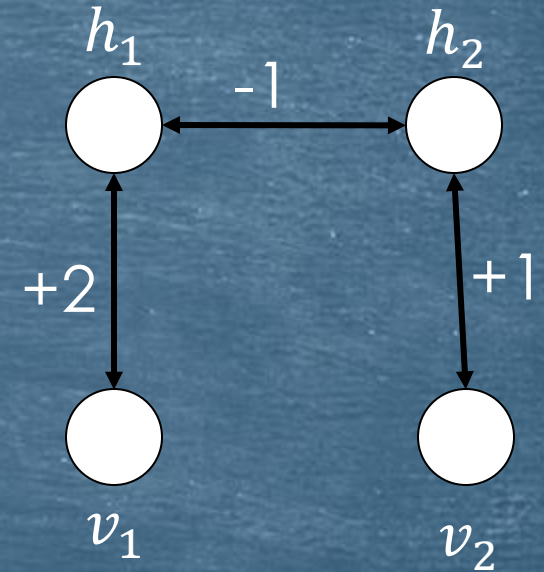
- ▶ The probability of a visible configuration is a marginal distribution over all the possible hidden vectors

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



# Boltzmann Machine: how weights define distribution

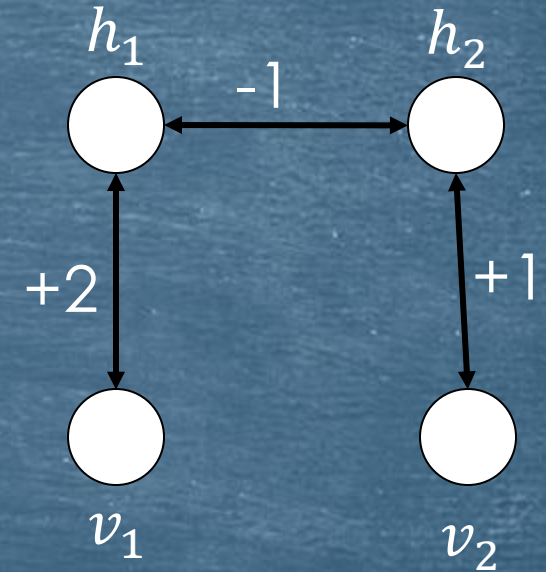
| Visible |   | Hidden |   |
|---------|---|--------|---|
| 1       | 1 | 1      | 1 |
| 1       | 1 | 1      | 0 |
| 1       | 1 | 0      | 1 |
| 1       | 1 | 0      | 0 |
| 1       | 0 | 1      | 1 |
| 1       | 0 | 1      | 0 |
| 1       | 0 | 0      | 1 |
| 1       | 0 | 0      | 0 |
| 0       | 1 | 1      | 1 |
| 0       | 1 | 1      | 0 |
| 0       | 1 | 0      | 1 |
| 0       | 1 | 0      | 0 |
| 0       | 0 | 1      | 1 |
| 0       | 0 | 1      | 0 |
| 0       | 0 | 0      | 1 |
| 0       | 0 | 0      | 0 |





# Boltzmann Machine: how weights define distribution

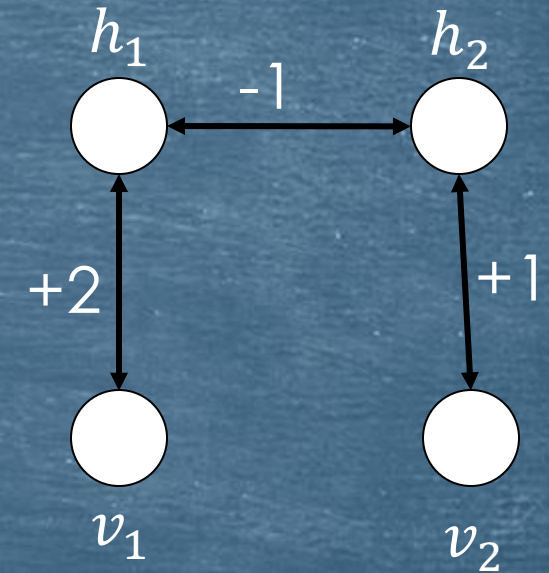
| Visible |   | Hidden |   | $-E$ |
|---------|---|--------|---|------|
| 1       | 1 | 1      | 1 | 2    |
| 1       | 1 | 1      | 0 | 2    |
| 1       | 1 | 0      | 1 | 1    |
| 1       | 1 | 0      | 0 | 0    |
| 1       | 0 | 1      | 1 | 1    |
| 1       | 0 | 1      | 0 | 2    |
| 1       | 0 | 0      | 1 | 0    |
| 1       | 0 | 0      | 0 | 0    |
| 0       | 1 | 1      | 1 | 0    |
| 0       | 1 | 1      | 0 | 0    |
| 0       | 1 | 0      | 1 | 1    |
| 0       | 1 | 0      | 0 | 0    |
| 0       | 0 | 1      | 1 | -1   |
| 0       | 0 | 1      | 0 | 0    |
| 0       | 0 | 0      | 1 | 0    |
| 0       | 0 | 0      | 0 | 0    |





# Boltzmann Machine: how weights define distribution

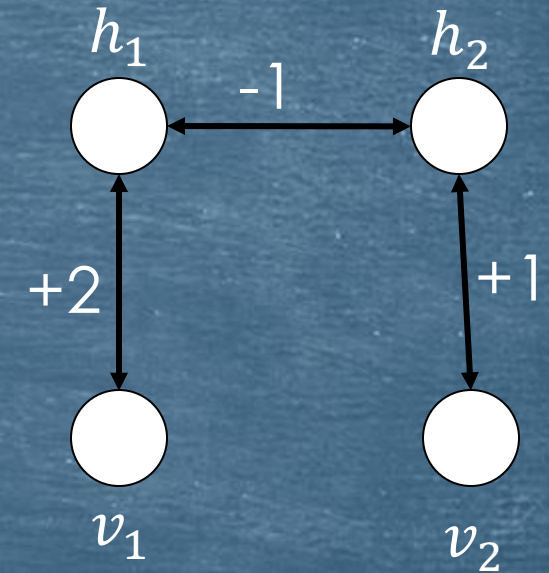
| Visible |   | Hidden |   | $-E$ | $e^{-E}$ |
|---------|---|--------|---|------|----------|
| 1       | 1 | 1      | 1 | 2    | 7.39     |
| 1       | 1 | 1      | 0 | 2    | 7.39     |
| 1       | 1 | 0      | 1 | 1    | 2.72     |
| 1       | 1 | 0      | 0 | 0    | 1        |
| 1       | 0 | 1      | 1 | 1    | 2.72     |
| 1       | 0 | 1      | 0 | 2    | 7.39     |
| 1       | 0 | 0      | 1 | 0    | 1        |
| 1       | 0 | 0      | 0 | 0    | 1        |
| 0       | 1 | 1      | 1 | 0    | 1        |
| 0       | 1 | 1      | 0 | 0    | 1        |
| 0       | 1 | 0      | 1 | 1    | 2.72     |
| 0       | 1 | 0      | 0 | 0    | 1        |
| 0       | 0 | 1      | 1 | -1   | 0.37     |
| 0       | 0 | 1      | 0 | 0    | 1        |
| 0       | 0 | 0      | 1 | 0    | 1        |
| 0       | 0 | 0      | 0 | 0    | 1        |
|         |   |        |   |      | 39.7     |





# Boltzmann Machine: how weights define distribution

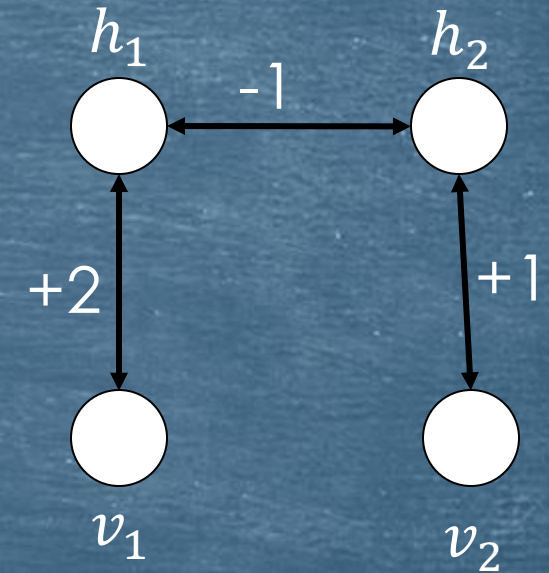
| Visible |   | Hidden |   | $-E$ | $e^{-E}$ | $p(v, h)$ |
|---------|---|--------|---|------|----------|-----------|
| 1       | 1 | 1      | 1 | 2    | 7.39     | 0.186     |
| 1       | 1 | 1      | 0 | 2    | 7.39     | 0.186     |
| 1       | 1 | 0      | 1 | 1    | 2.72     | 0.069     |
| 1       | 1 | 0      | 0 | 0    | 1        | 0.025     |
| 1       | 0 | 1      | 1 | 1    | 2.72     | 0.069     |
| 1       | 0 | 1      | 0 | 2    | 7.39     | 0.186     |
| 1       | 0 | 0      | 1 | 0    | 1        | 0.025     |
| 1       | 0 | 0      | 0 | 0    | 1        | 0.025     |
| 0       | 1 | 1      | 1 | 0    | 1        | 0.025     |
| 0       | 1 | 1      | 0 | 0    | 1        | 0.025     |
| 0       | 1 | 0      | 1 | 1    | 2.72     | 0.069     |
| 0       | 1 | 0      | 0 | 0    | 1        | 0.025     |
| 0       | 0 | 1      | 1 | -1   | 0.37     | 0.009     |
| 0       | 0 | 1      | 0 | 0    | 1        | 0.25      |
| 0       | 0 | 0      | 1 | 0    | 1        | 0.025     |
| 0       | 0 | 0      | 0 | 0    | 1        | 0.025     |
|         |   |        |   | 39.7 |          |           |





# Boltzmann Machine: how weights define distribution

| Visible |   | Hidden |   | $-E$ | $e^{-E}$ | $p(v, h)$ | $p(v)$ |
|---------|---|--------|---|------|----------|-----------|--------|
| 1       | 1 | 1      | 1 | 2    | 7.39     | 0.186     | 0.466  |
| 1       | 1 | 1      | 0 | 2    | 7.39     | 0.186     |        |
| 1       | 1 | 0      | 1 | 1    | 2.72     | 0.069     |        |
| 1       | 1 | 0      | 0 | 0    | 1        | 0.025     |        |
| 1       | 0 | 1      | 1 | 1    | 2.72     | 0.069     | 0.305  |
| 1       | 0 | 1      | 0 | 2    | 7.39     | 0.186     |        |
| 1       | 0 | 0      | 1 | 0    | 1        | 0.025     |        |
| 1       | 0 | 0      | 0 | 0    | 1        | 0.025     |        |
| 0       | 1 | 1      | 1 | 0    | 1        | 0.025     | 0.144  |
| 0       | 1 | 1      | 0 | 0    | 1        | 0.025     |        |
| 0       | 1 | 0      | 1 | 1    | 2.72     | 0.069     |        |
| 0       | 1 | 0      | 0 | 0    | 1        | 0.025     |        |
| 0       | 0 | 1      | 1 | -1   | 0.37     | 0.009     | 0.084  |
| 0       | 0 | 1      | 0 | 0    | 1        | 0.25      |        |
| 0       | 0 | 0      | 1 | 0    | 1        | 0.025     |        |
| 0       | 0 | 0      | 0 | 0    | 1        | 0.025     |        |
|         |   |        |   |      | 39.7     |           |        |





# Boltzmann Machine

---

On the previous example, we were able to compute the probability of a configuration  $(v, h)$  or the probability of a visible vector because the network was small. Based on that probability we can generate a sample.

Here, we were able to compute all the possible energies in the network in order to compute the partition function  $(\sum_{u,g} e^{-E(u,g)})$ .

To generate a sample from a bigger network we follow the following steps:

1. start from a random position
2. keep updating the units in a stochastic way based on the energy gap
3. After some iterations, the network will arrive at the stationary distribution (thermal equilibrium)
4. On that point, we'll have a sample of that distribution. The sample we'll be generated with the probability proportional to  $e^{-E(v,h)}$



# Training a Boltzmann Machine

---



# Training a Boltzmann Machine


---

- ▶ The goal of training is to maximize the probability that the Boltzmann Machine assigns to the training vectors.
- ▶ This is equivalent to:
  - Maximize the product of probabilities assigned to each binary training vector
  - Maximize the sum of the log probabilities assigned to each binary training vector
- ▶ It can also be viewed as:
  - Maximize the probability of obtaining the  $N$  training vectors if we let the Boltzmann Machine settle to its stationary distribution and sample  $N$  visible vectors



# Training a Boltzmann Machine

- The learning is very simple:

$$\frac{\partial \log p(v)}{\partial w_{ij}} = \langle s_i s_j \rangle_v - \langle s_i s_j \rangle_{\text{model}}$$


Expected value of product of states when the visible units are fixed with a training data (at thermal equilibrium):

How often are  $s_i$  and  $s_j$  activated together when the visible units are fixed on a training data (at thermal equilibrium)

Expected value of product of states at thermal equilibrium.

How often are  $s_i$  and  $s_j$  activated together at thermal equilibrium. (nothing is fixed here)



# Training a Boltzmann Machine

- How was the learning rule obtained?

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

$$\frac{\partial \ln(p(v))}{\partial w_{ij}} = \frac{\partial \ln(\sum_h e^{-E(v,h)})}{\partial w_{ij}} - \frac{\partial \ln(\sum_{u,g} e^{-E(u,g)})}{\partial w_{ij}}$$

The first term refers to the energy of the system for a certain visible vector (when a visible vector is fixed) while the second term refers to the energy of the entire model (nothing is fixed)

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

$$\frac{\partial \ln(\sum_h e^{-E(v,h)})}{\partial w_{ij}} = 0 + 0 + \frac{\partial w_{ij} \partial \ln(\sum_{i < j} v_i v_j w_{ij})}{\partial w_{ij}} + \frac{\partial w_{ij} \partial \ln(\sum_{k < l} h_k h_l w_{kl})}{\partial w_{ij}} + \frac{\partial w_{ij} \partial \ln(\sum_{i,k} v_i h_k w_{ik})}{\partial w_{ij}}$$

The result of the above will be  $s_i s_j$  where  $s_i$  and  $s_j$  are the units ( $v$  or  $h$ ) linked by the weight  $w_{ij}$   
The other term of the equation is obtained in the same way



# Training a Boltzmann Machine

---

- ▶ The learning is very simple:

$$\Delta w_{ij} = \langle s_i s_j \rangle_{data} - \langle s_i s_j \rangle_{model}$$

The change on the weight  $w_{ij}$  will be proportional to  $\Delta w_{ij}$

$\Delta w_{ij}$  represents the expected product of the activities averaged over all the training set when visible units are set on the visible units, minus the expected product of the activities when the visible units are not set.

- ▶ The first term is similar to the learning method in Hopfield Net  
The second term is similar to the unlearning method in Hopfield Net (to eliminate spurious minima)



# Training a Boltzmann Machine

---

- ▶ The learning is very simple:

$$\Delta w_{ij} = \langle s_i s_j \rangle_{data} - \langle s_i s_j \rangle_{model}$$

Since we add this term to weights, that means that using this update rule

- the energy of the system will be decreased when the visible units are used by growing the weights that correspond to units that are more on when the training data is fixed than when anything else is used.
- So, it will increase the probability of the configuration that contains the training data and some hidden data



# Training a Boltzmann Machine

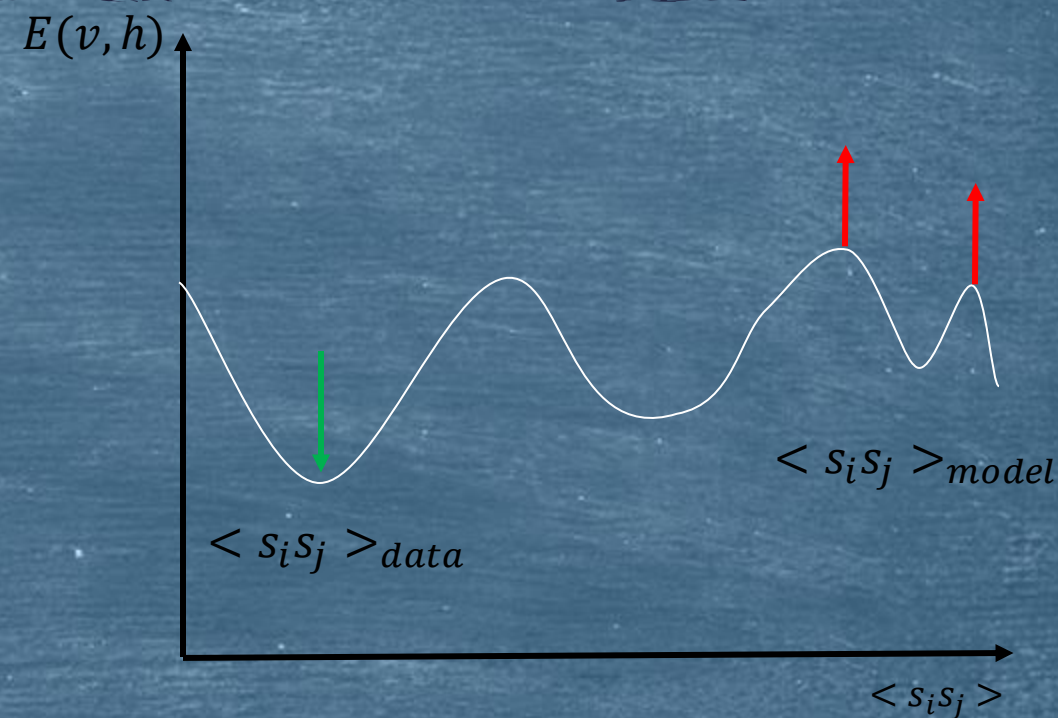
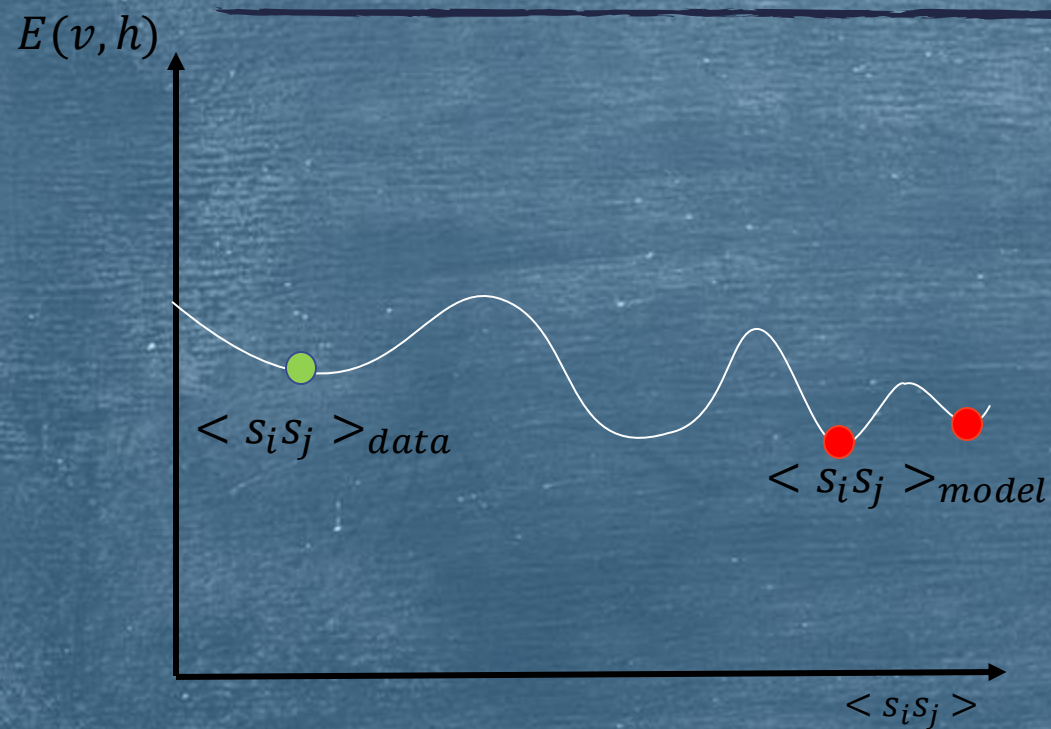
---

Since we add this term to weights, that means that using this update rule

- the energy of the system will be increased when anything else is used but the training data by lowering the weights on the most probable configurations that do not use the training data
- So it will decrease the probability of the most probable configurations that do not use the training data. This will decrease the partition function.
- So, the probability of the configuration using the training data is increased, and the partition function is decreased



# Training a Boltzmann Machine



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

← This gets larger

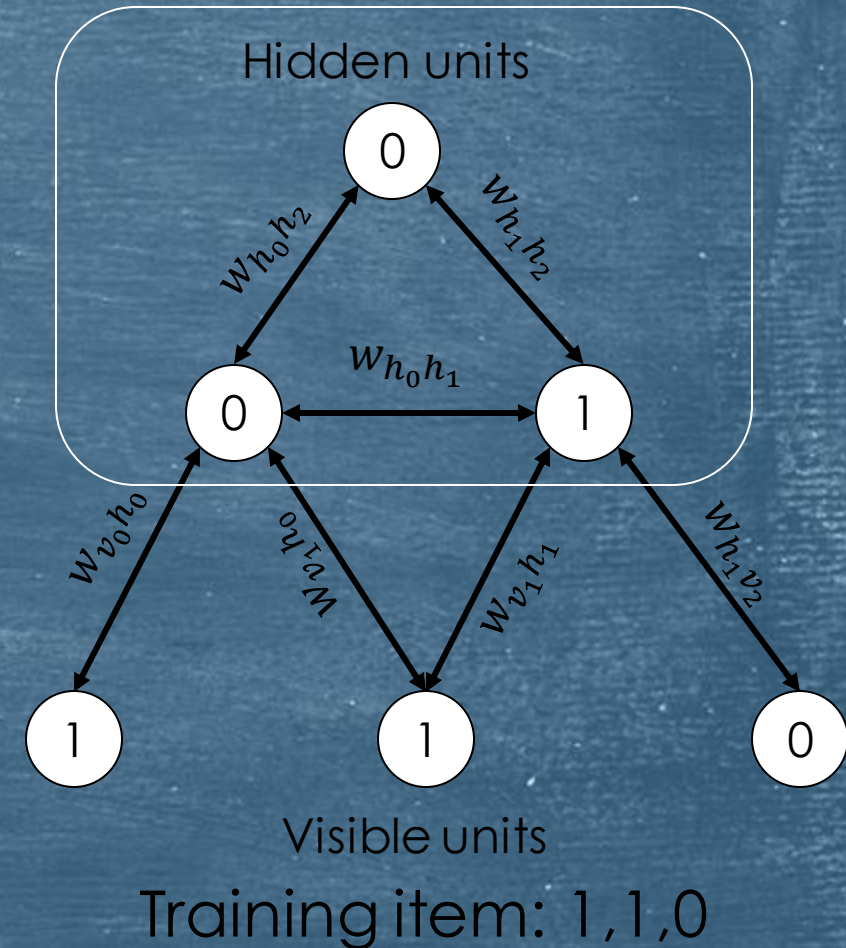
← This gets smaller



# Training a Boltzmann Machine

► Learning procedure (positive step):

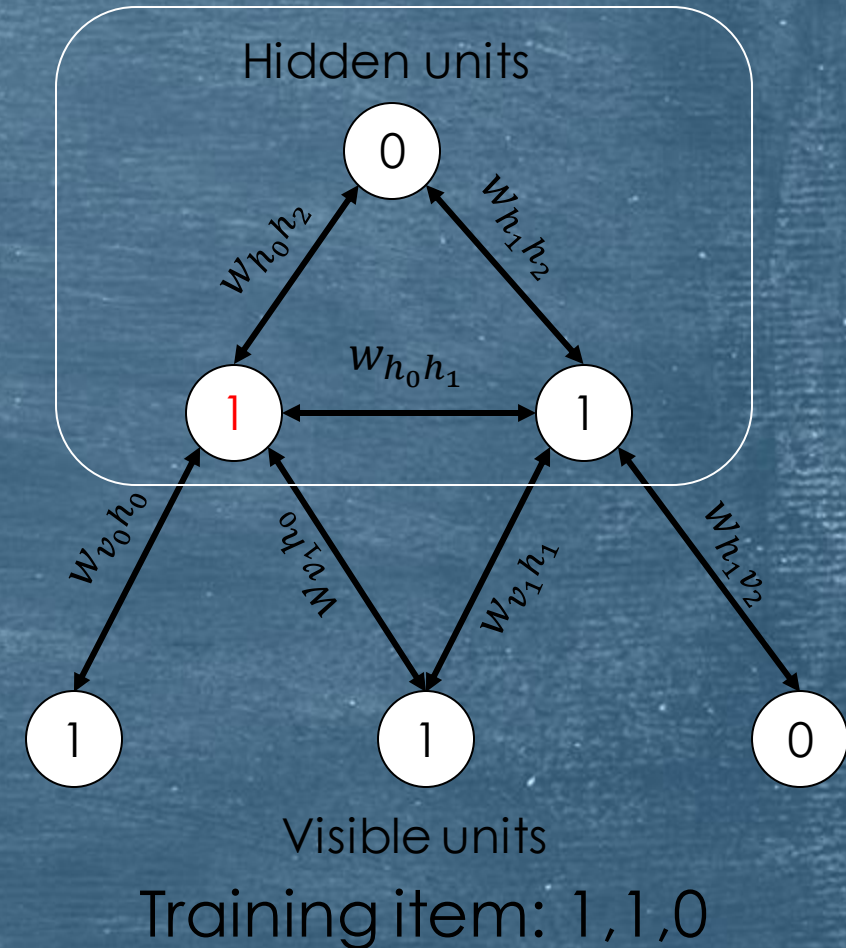
1. Initialize the visible units with a training vector and the hidden units with random activations





# Training a Boltzmann Machine

- Learning procedure (positive step):
  1. Initialize the visible units with a training vector and the hidden units with random activations
  2. With the visible units fixed, pick hidden units at random and activate it based on the energy gap

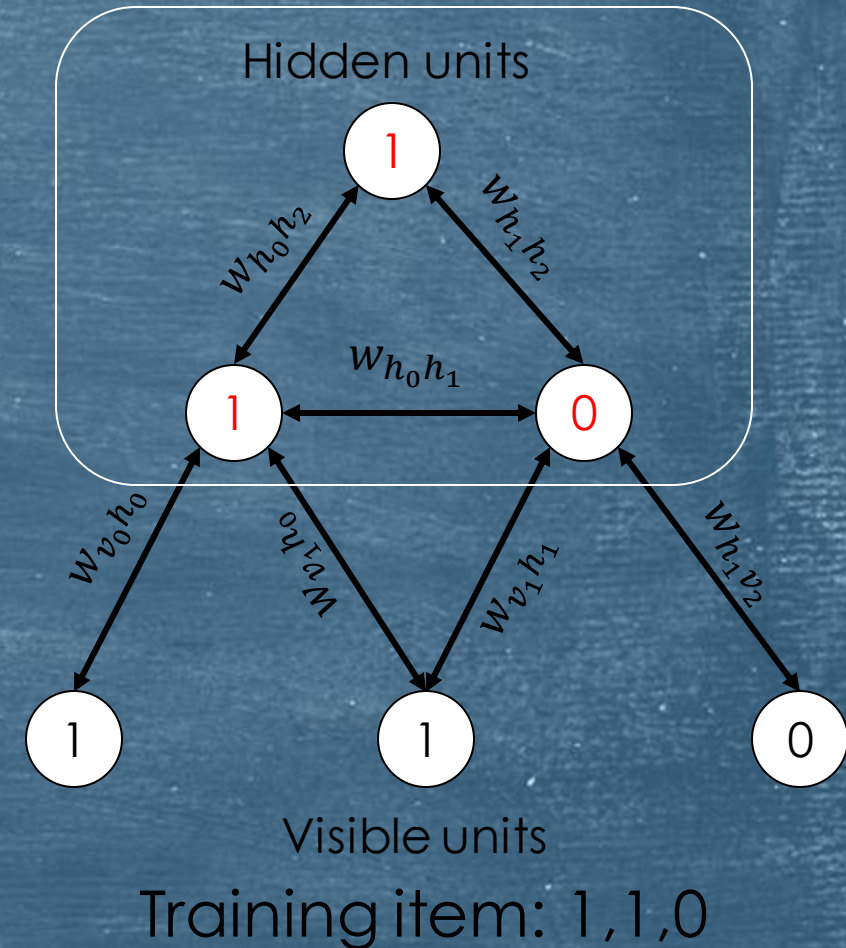




# Training a Boltzmann Machine

► Learning procedure (positive step):

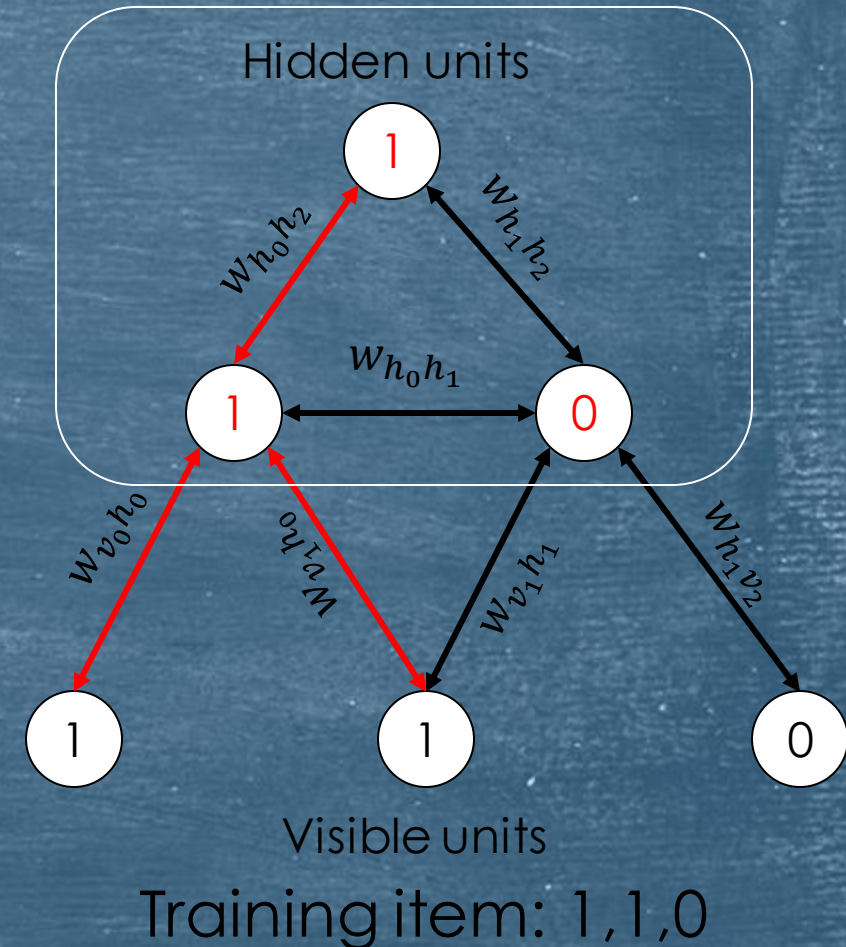
1. Initialize the visible units with a training vector and the hidden units with random activations
2. With the visible units fixed, pick hidden units at random and activate it based on the energy gap
3. After some iterations, the network will arrive at thermal equilibrium.





# Training a Boltzmann Machine

- Learning procedure (positive step):
  1. Initialize the visible units with a training vector and the hidden units with random activations
  2. With the visible units fixed, pick hidden units at random and activate it stochastically (energy gap)
  3. After some iterations, the network will arrive at thermal equilibrium.
  4. Take each two units and see if they active together.

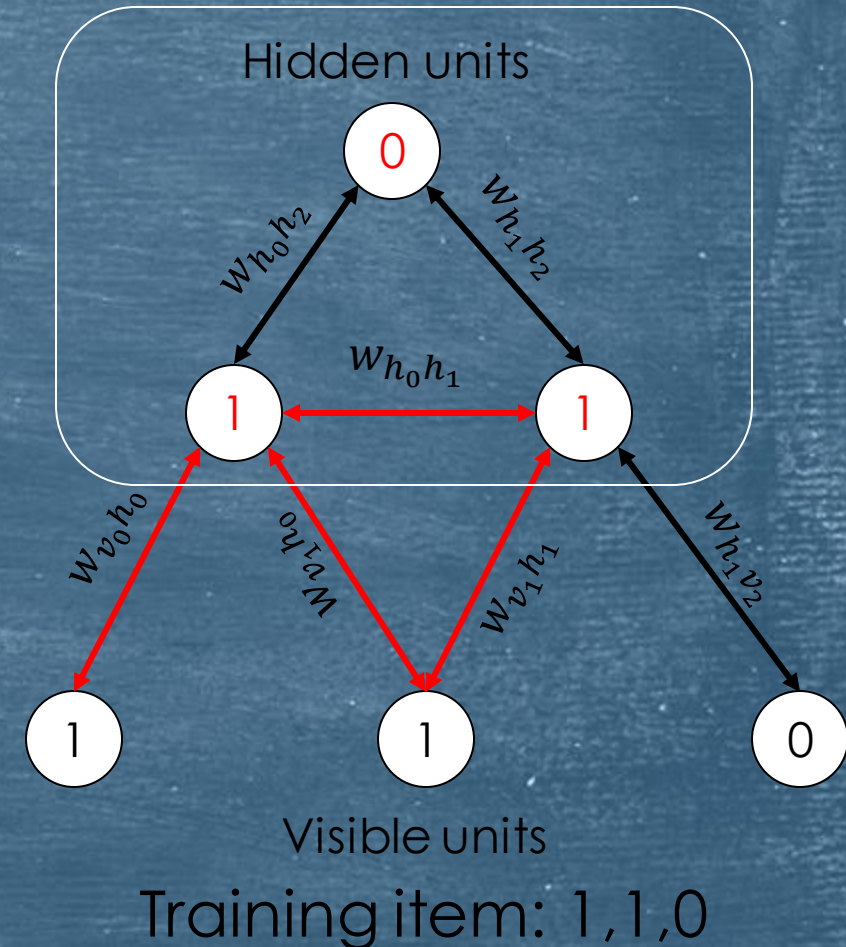




# Training a Boltzmann Machine

► Learning procedure (positive step):

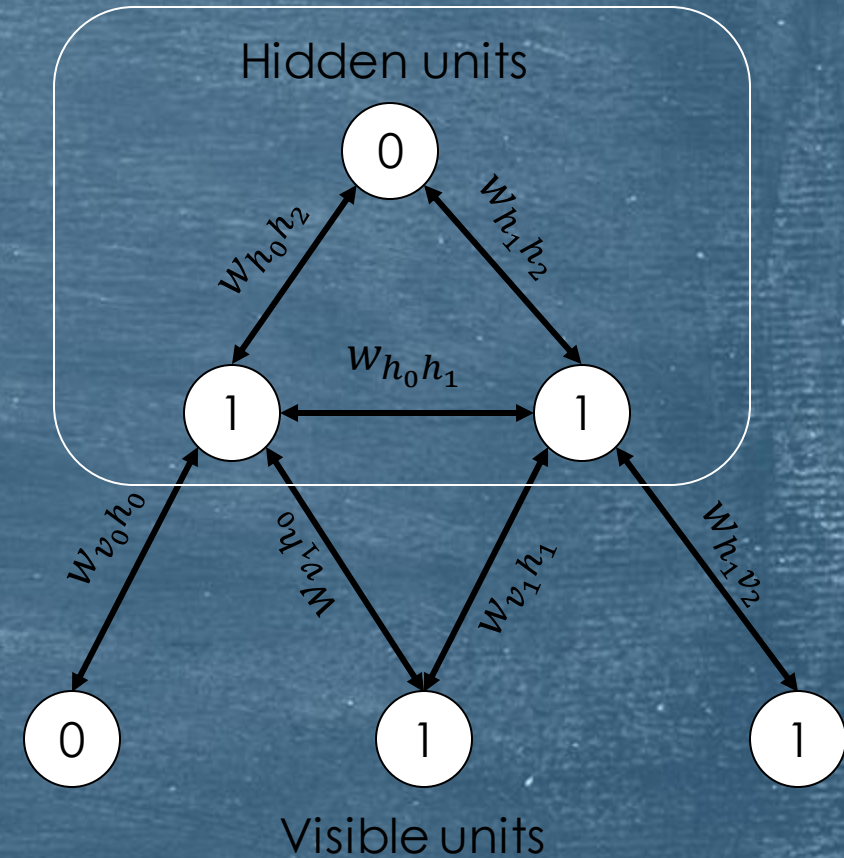
1. Initialize the visible units with a training vector and the hidden units with random activations
2. With the visible units fixed, pick hidden units at random and activate it stochastically (energy gap)
3. After some iterations, the network will arrive at thermal equilibrium.
4. Take each two units and see if they active together.
5. Activate hidden units again (generate another sample) and see, for each two units, if they activate together
6. We collect many samples and for each weight  $w_{ij}$ , compute the average  $\langle s_i s_j \rangle_{\text{training\_item}}$





# Training a Boltzmann Machine

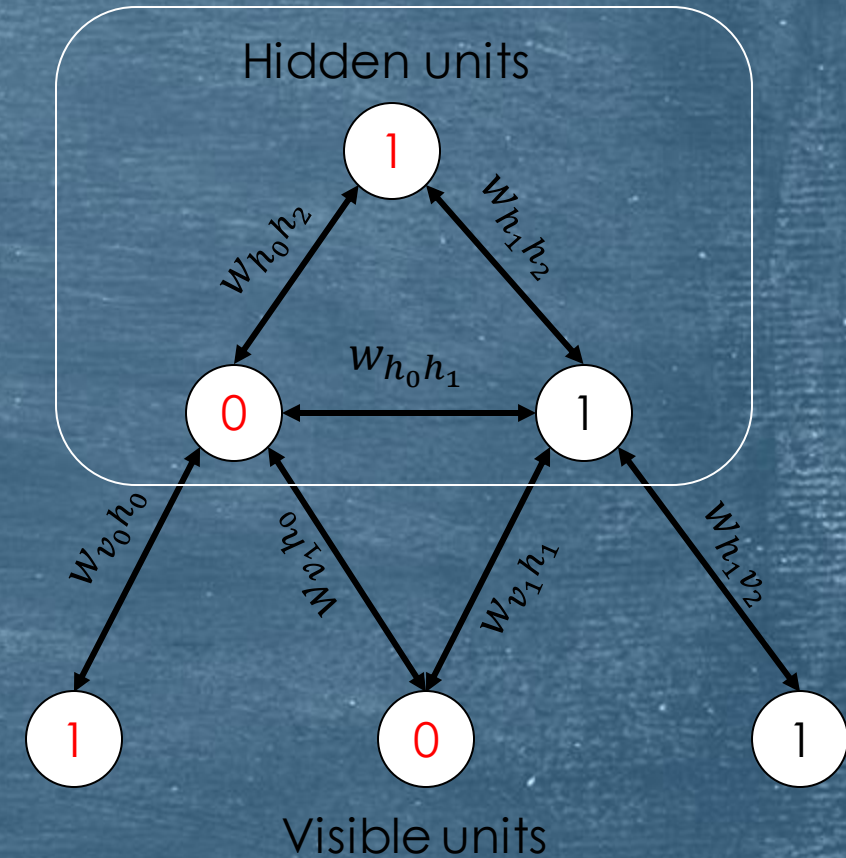
- Learning procedure (negative step):
  1. Initialize the visible units and the hidden units with random activations





# Training a Boltzmann Machine

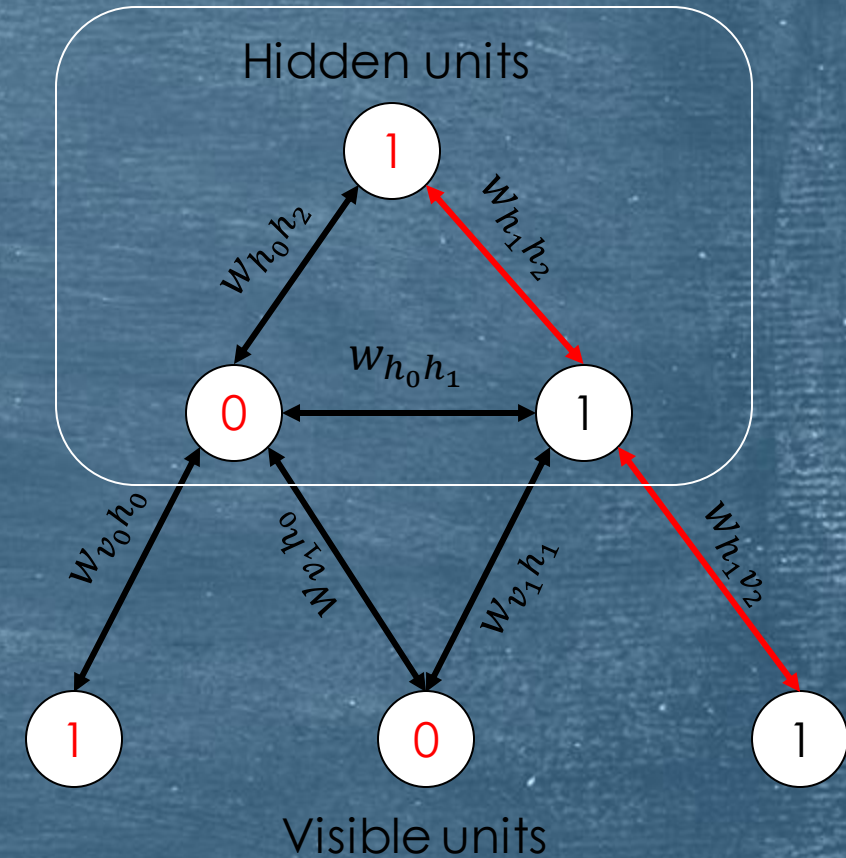
- Learning procedure (negative step):
  1. Initialize the visible units and the hidden units with random activations
  2. Until thermal equilibrium is reached, pick a unit at random and activate-it, stochastically, based on its energy gap.





# Training a Boltzmann Machine

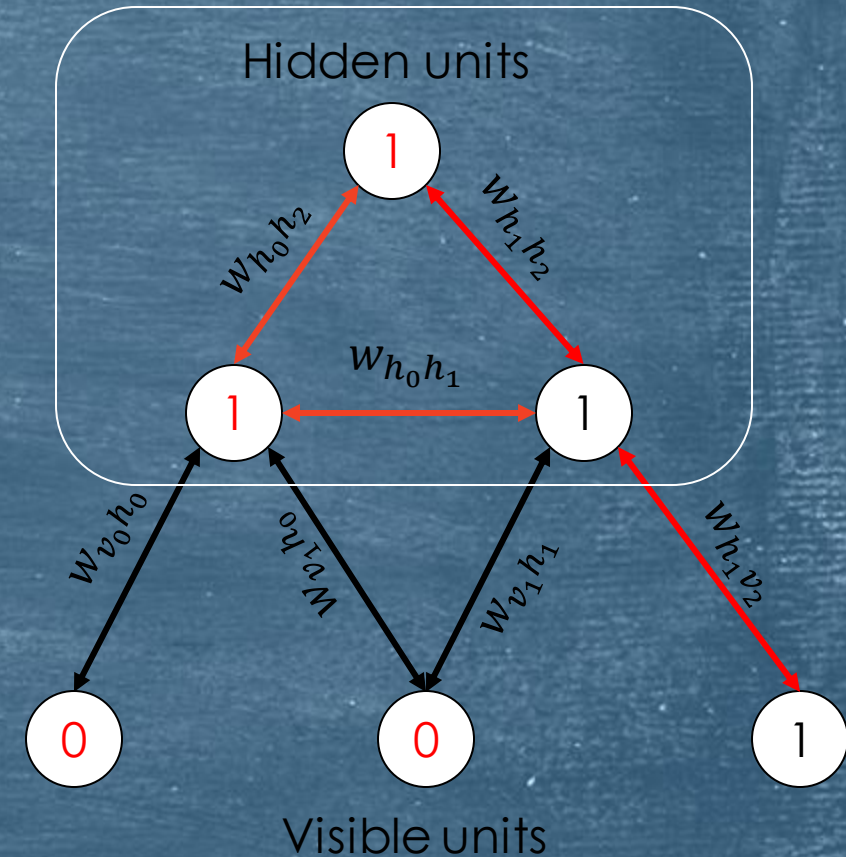
- Learning procedure (negative step):
  1. Initialize the visible units and the hidden units with random activations
  2. Until thermal equilibrium is reached, pick a unit at random and activate-it, stochastically, based on its energy gap.
  3. For each connection  $w_{ij}$  see if the units  $i$  and  $j$  are being activated together.





# Training a Boltzmann Machine

- Learning procedure (negative step):
  1. Initialize the visible units and the hidden units with random activations
  2. Until thermal equilibrium is reached, pick a unit at random and activate-it, stochastically, based on its energy gap.
  3. For each connection  $w_{ij}$  see if the units  $i$  and  $j$  are being activated together.
  4. Generate another sample and see, for each connection  $w_{ij}$ , if the units  $i$  and  $j$  are being activated together.
  5. Average the results and compute for each connection  $w_{ij}$ ,  $\langle s_i s_j \rangle_{model}$





# Training a Boltzmann Machine

---

► Learning procedure:

Compute the difference

$$\Delta w_{ij_{training}} = \langle s_i s_j \rangle_{training} - \langle s_i s_j \rangle_{model}$$

Pick another training item and compute the same amount.

$$\Delta w_{ij} = \langle \Delta w_{ij_{data}} \rangle \text{ (or just average all the } s_i s_j \text{ from all the data)}$$

Average all the training updates and multiply by a learning rule

$$w_{ij} = w_{ij} + \eta \Delta w_{ij}$$



# Restricted Boltzmann Machine

---



# Restricted Boltzmann Machine

---

Even though the Boltzmann machine theoretically works, in practice it is not that used.

This is because, as the number of units grow, reaching the thermal equilibrium takes more and more iterations

In this sense, a restricted version of the Boltzmann machine was introduced. This was highly used in practice and marked the beginning of deep architecture in supervised learning



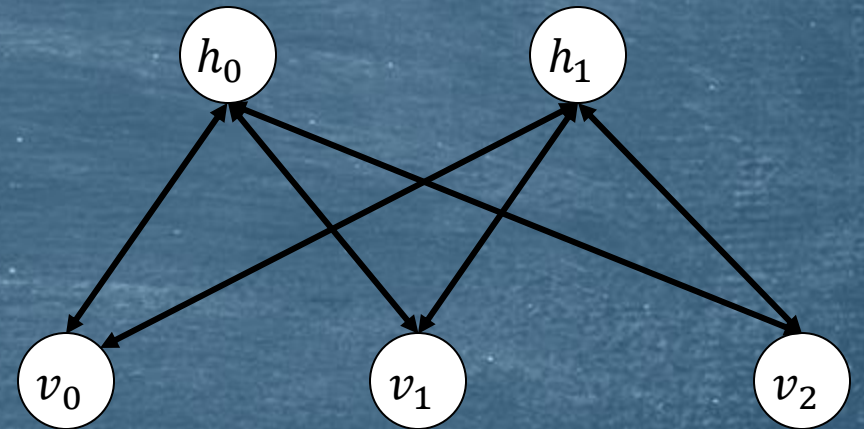
# Restricted Boltzmann Machine

---

A restricted Boltzmann Machine is a Boltzmann machine (so it has an Energy functions and units are stochastic), but:

- There is only 1 layer of hidden and 1 layer of visible units
- There are no connection between hidden units
- There are no connection between visible units
- Each hidden units is connected with every visible unit

(so we have a bipartite non-oriented graph)





# Restricted Boltzmann Machine

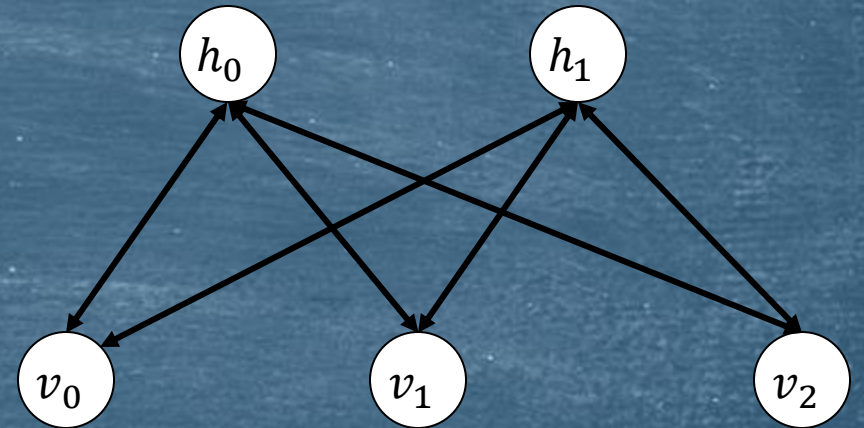
---

Learning is performed as in a Boltzmann Machine:

$$\Delta w_{ij} = \langle s_i s_j \rangle_{data} - \langle s_i s_j \rangle_{model}$$

However, on the positive phase the thermal equilibrium is achieved in one step.

This is because each hidden unit is only influenced by the visible units (which don't change)





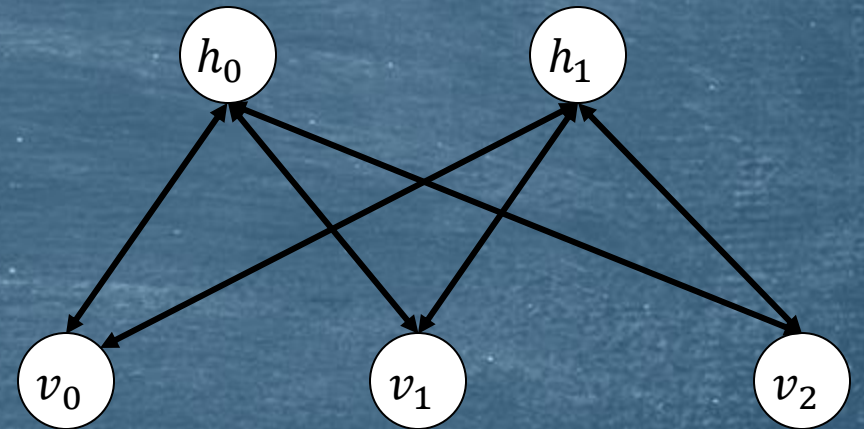
# Restricted Boltzmann Machine

This means that computing the  $\langle s_i s_j \rangle_{data}$  is fast, since we know the probability of  $h_j$  and  $v_i$  is fixed

$$p(h_j = 1) = \frac{1}{1 + e^{-(b_j + \sum_{i \in vis} v_i w_{ij})}}$$

However, it still takes a long time reaching the thermal equilibrium on the negative phase (when the visible units are modified)

So computing  $\langle s_i s_j \rangle_{model}$  still takes time.

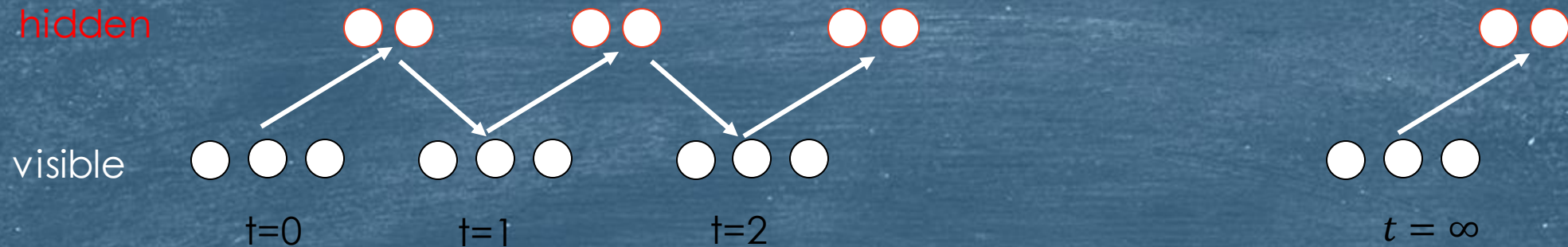




# Restricted Boltzmann Machine

Reaching the thermal equilibrium on the negative phase is done in the following manner:

1. Start with a random configuration (or a training data) on the visible units
2. Stochastically update hidden units based on the visible units
3. Stochastically update visible units based on the hidden units
4. Repeat 2 and 3 until thermal equilibrium is reached





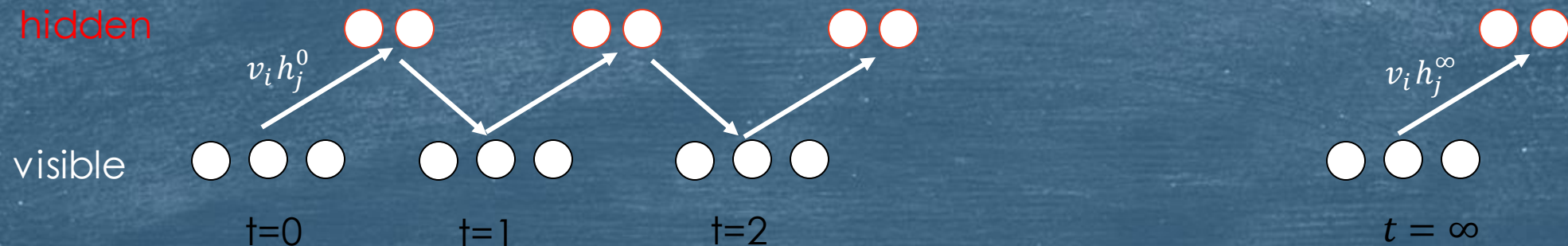
# Restricted Boltzmann Machine

If the visible units are initialized with a training item, then the update rule becomes:

$$\Delta w_{ij} = \eta (< v_i h_j >^0 - < v_i h_j >^\infty)$$

Where

- $< v_i h_j >^0$  represents the estimate of product activations, after the hidden units have been update for the first time
- $< v_i h_j >^\infty$  represents the estimate of product of activation at thermal equilibrium





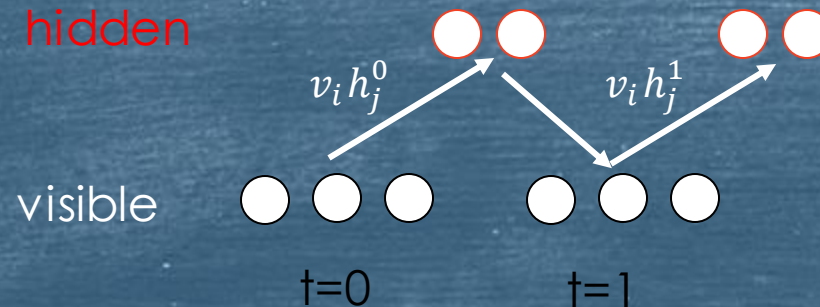
# Restricted Boltzmann Machine

Contrastive divergence:

However, Hinton discovered a shortcut of that, called contrastive divergence.

Instead of waiting to thermal equilibrium, use the statistics after just 1 step

$$\text{CD1: } \Delta w_{ij} = \eta (\langle v_i h_j \rangle^0 - \langle v_i h_j \rangle^1)$$





# Restricted Boltzmann Machine

---

Why does CD works?

When the hidden units are updated, at  $t_0$ , the system gets away from the data and heads to its stationary distribution given by its current weights (which are random).

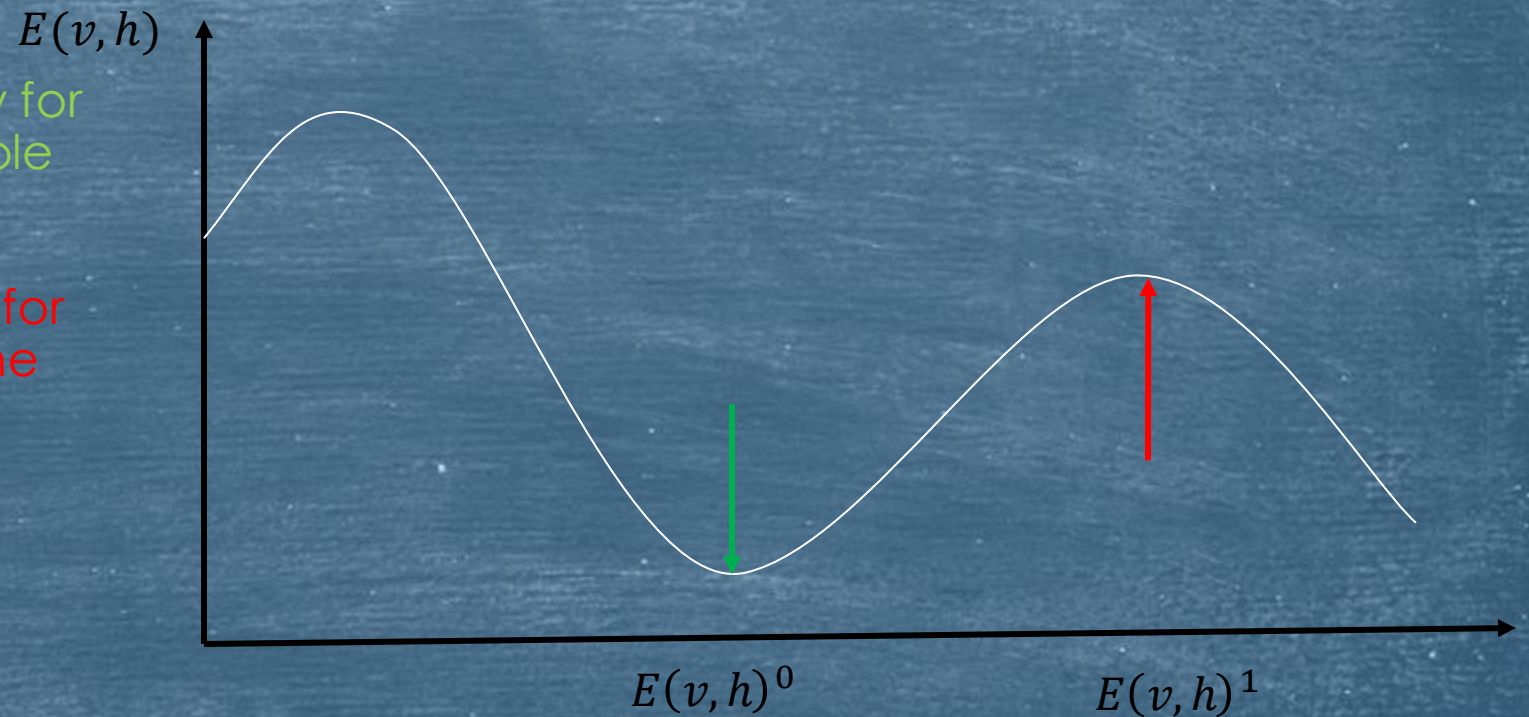
Since we know the weights are going to drive the system to a wrong distribution, we can increase the probability of the data and decrease the probability of wondering away from the data after just 1 step.

This is performing updates as we notice that the system begins to display signs of improper fitting of the data (provided that the signs are visible in the time frame)



# Restricted Boltzmann Machine

1. Lowers the energy for the  $(v, h)$  responsible for the data
2. Raises the energy for the sample that the model wants to produce given its current weights.





# Restricted Boltzmann Machine

---

What is the disadvantage of CD1?

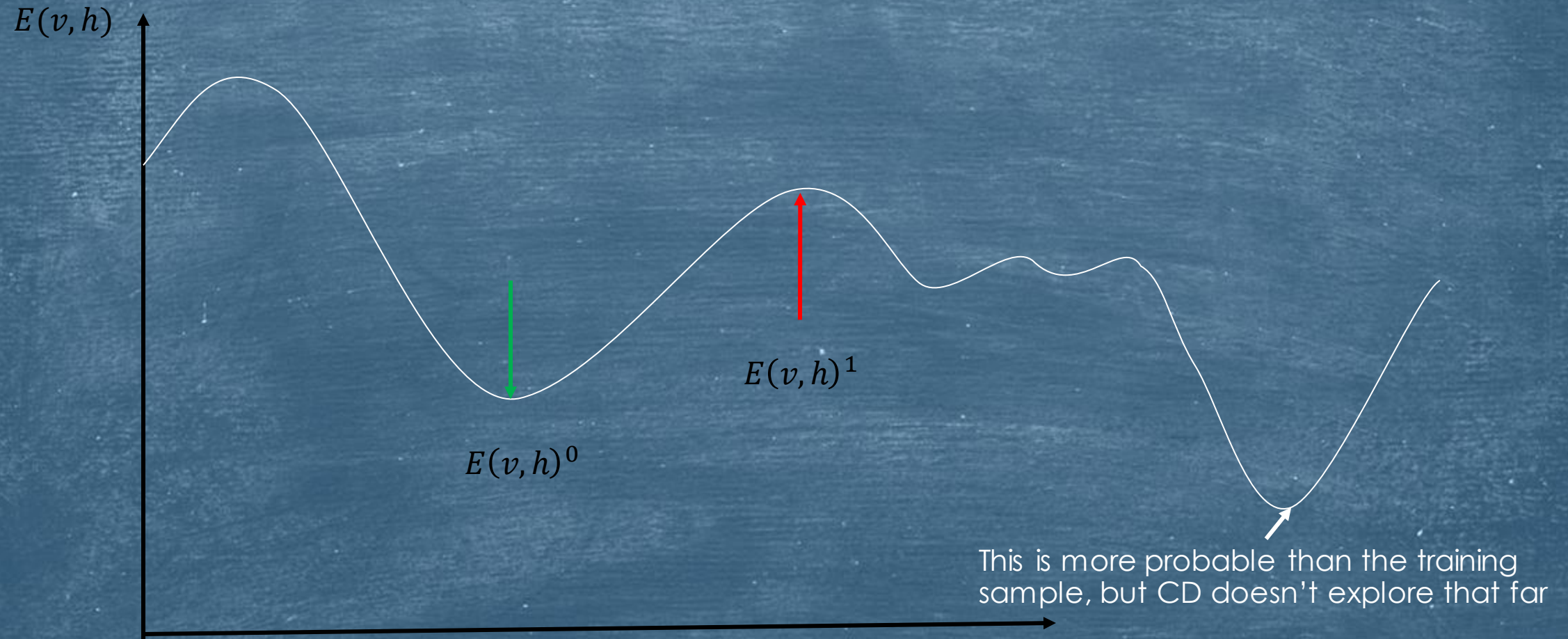
The algorithm doesn't explore the energy space.  
It decreases the energy of the training sample and increases the energy of an item which is very close (1-step) from the training item.

However, at thermal equilibrium, the Energy could be lower than the one obtained by increasing the Energy of the training item.

This means that CD-1 doesn't minimize the partition function as strong as it could. Having a big value for the partition function decreases the probability of the training item



# Restricted Boltzmann Machine



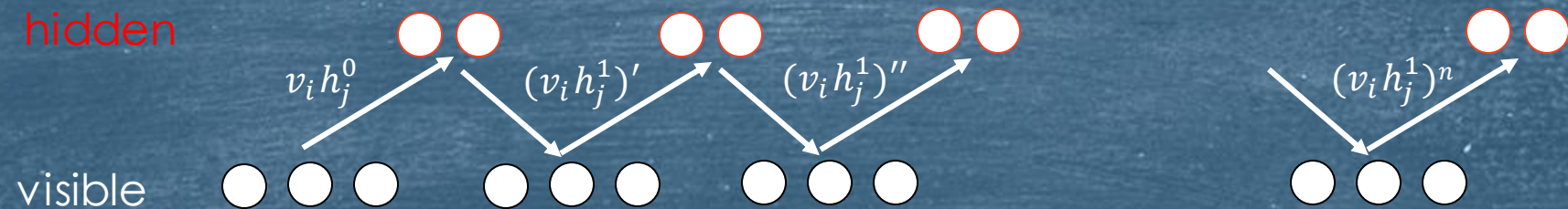


# Restricted Boltzmann Machine

A simple solution was invented by Tieleman(2008) called persistent contrastive divergence (PCD).

The algorithm starts as contrastive divergence.

However, on the next steps, when  $\langle v_i h_j \rangle^1$  needs to be computed, it doesn't start the chain from an original sample, but continues the previously aborted chain.





# Questions & Discussion

---



# References

---

- ▶ [https://en.wikipedia.org/wiki/Boltzmann\\_machine](https://en.wikipedia.org/wiki/Boltzmann_machine)
- ▶ [https://en.wikipedia.org/wiki/Boltzmann\\_distribution](https://en.wikipedia.org/wiki/Boltzmann_distribution)
- ▶ <http://rocknrollnerd.github.io/ml/2015/07/18/general-boltzmann-machines.html>
- ▶ <http://web.archive.org/web/20110718022336/http://learning.cs.toronto.edu/~hinton/absps/cogscibm.pdf>
- ▶ <https://www.cs.toronto.edu/~hinton/csc321/readings/boltz321.pdf>
- ▶ Quick explanation of stationary distribution of markov chain:  
<https://www.youtube.com/watch?v=fmY1iZTZIE&t=643s>
- ▶ <https://www.youtube.com/watch?v=IYaOMor9qvE>