Boltzmann Machines

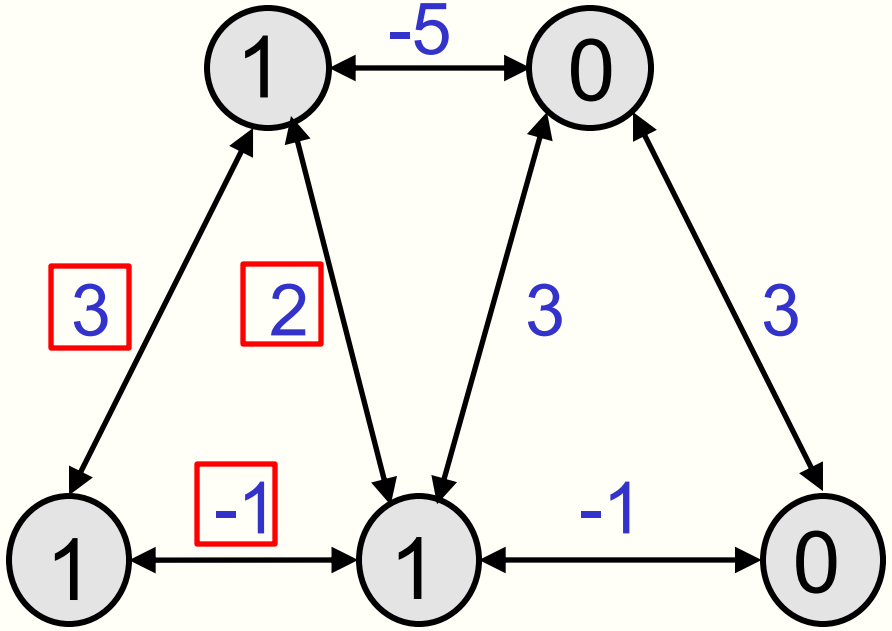
Geoffrey Hinton

University of Toronto
&
Vector Institute

Hopfield Nets

(the version with states of 1 = active and 0 = inactive)

- The neurons are binary and the weighted connections are **symmetri**c.
- The global state of the whole network is called a “configuration”.
- Each configuration has a **goodness** which is simply the sum of all the weights between pairs of active neurons.
  - The energy of a configuration is minus the goodness.

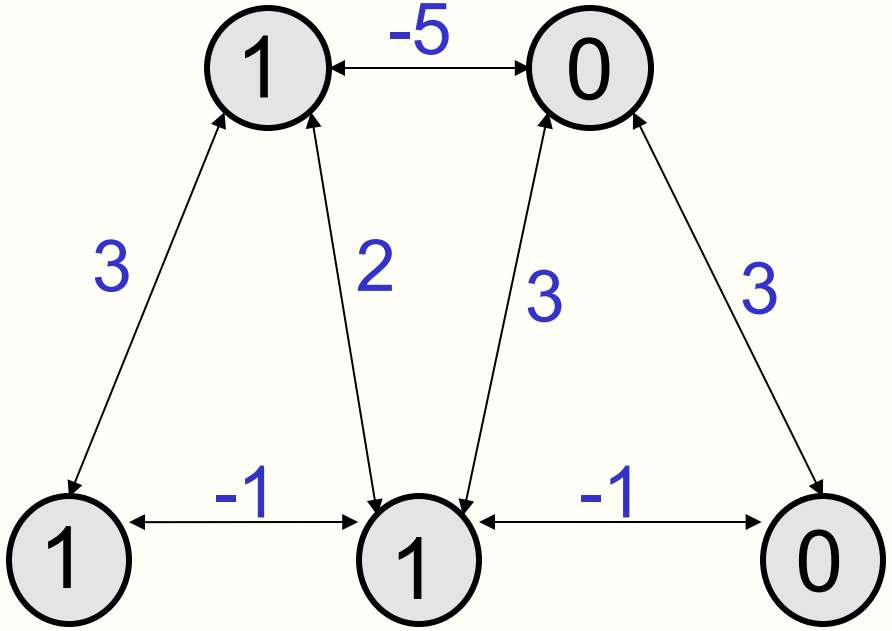


- E = goodness = 4

Settling to an energy minimum

- The whole point of a Hopfield net is that each neuron can easily compute what it should do to minimize the global energy (energy is badness).
  - If the total weighted input coming from other active neurons is positive, turn on.
  - If the total weighted input is negative turn off.
- If each neuron keeps using this rule, the network will settle to a configuration that is an energy minimum.

A configuration that is an energy minimum

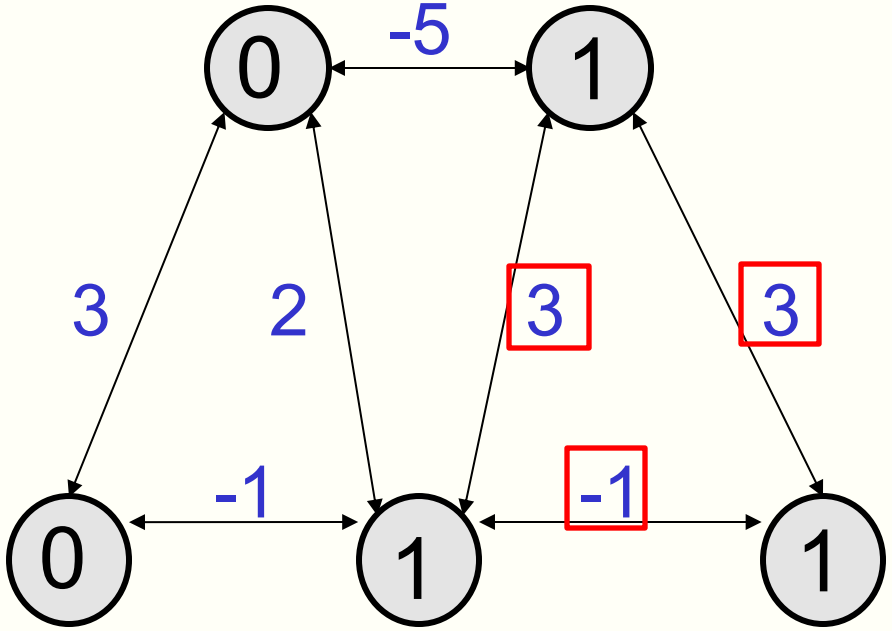


- E = goodness = 4

Settling to an energy minimum

- A Hopfield net can have many different energy minima.
- Which minimum it ends up in depends on where it starts.
  - It can also be affected by the order in which the neurons are updated.

A better energy minimum



- E = goodness = 5

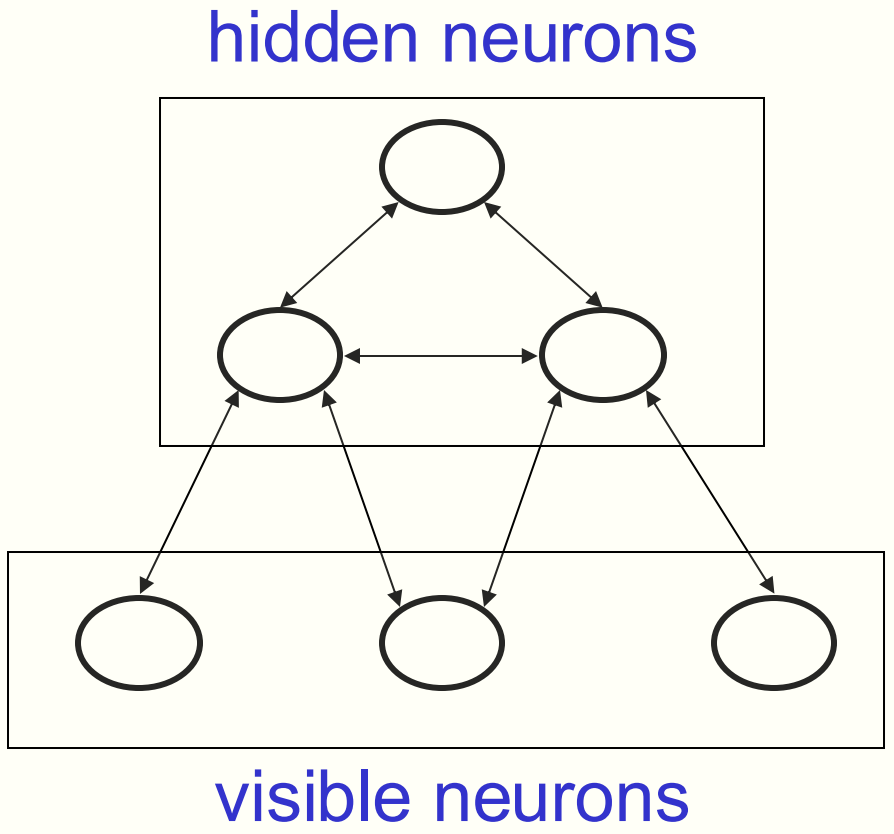
A neat way to make use of this type of neural network

- Hopfield (1982) proposed that memories could be energy minima of a neural net.
  - The binary decision rule can then be used to "clean up" incomplete or corrupted memories.
  - Start with the corrupted memory and settle to an energy minimum.
- Using energy minima to represent memories gives a content-addressable memory:
  - An item can be accessed by just knowing part of its content.
  - Settling to an energy minimum then fills in the missing bits.

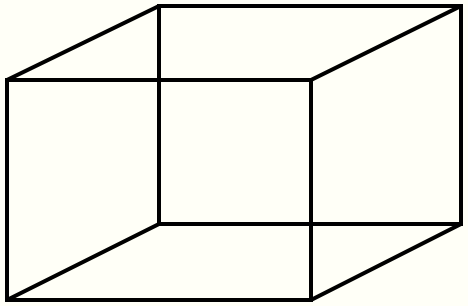
A different computational role for Hopfield nets

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

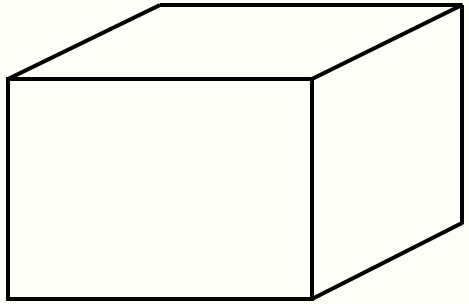
- The energy represents the badness of the interpretation.



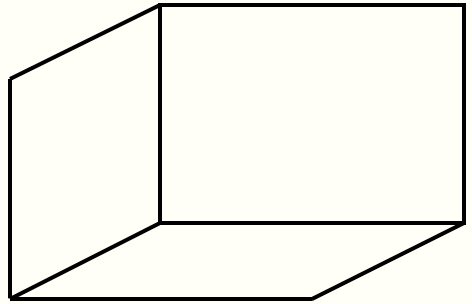
Interpreting a line drawing as a 3-D object



Ambiguous line drawing



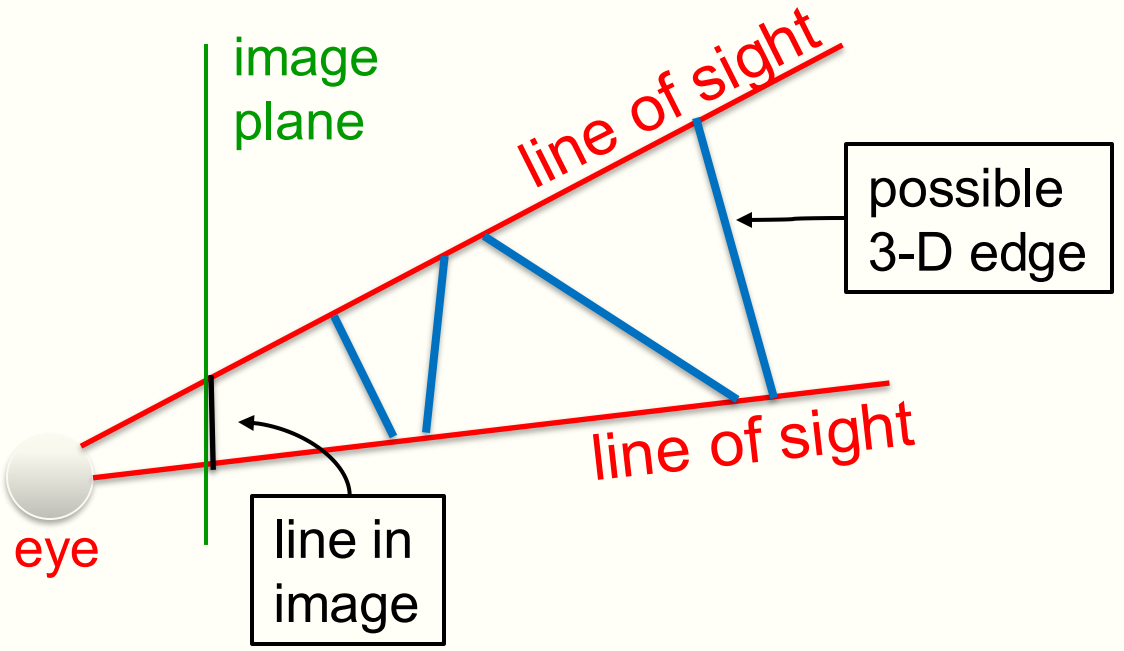
Interpretation 1



Interpretation 2

What can we infer about 3-D edges from 2-D lines in an image?

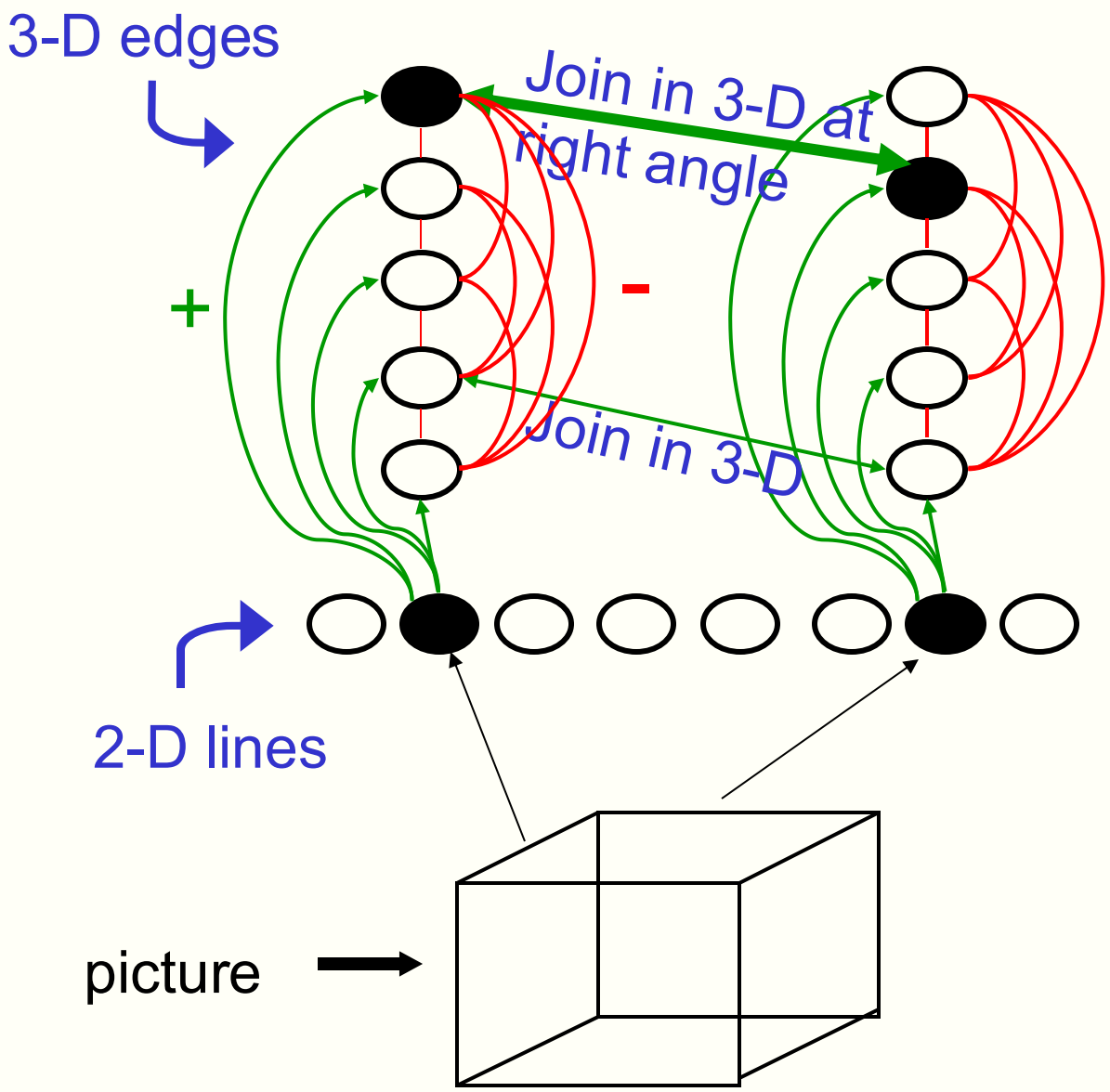
- A 2-D line in an image could have been caused by many different 3-D edges in the world.
- The information that has been lost in the image is the 3-D depth of each end of the 2-D line.
  - The end of the edge could be anywhere on the line of sight.
  - So there is a whole family of 3-D edges that all correspond to the same 2-D line.



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

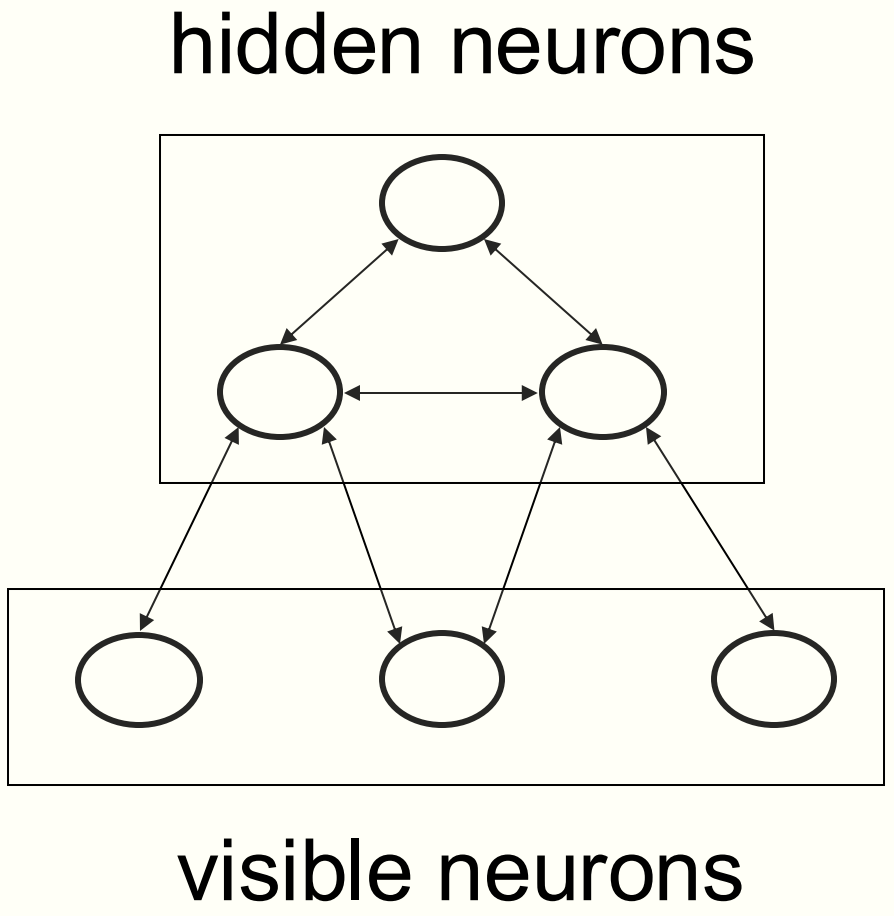
A thought experiment: Interpreting a line drawing

- Use one "2-D line" neuron for each possible line in the picture.
  - Any particular picture only activates a few of the line neurons.
- Use one "3-D edge" neuron for each possible 3-D edge in the scene.
  - Each 2-D line neuron tries to activate all the 3-D edge neurons that could project to that line.
  - These 3-D edges compete by using connections with negative weights
- Make 3-D edges support each other if they join in 3-D.
- Make them **strongly support** each other if they join at right angles in 3-D.



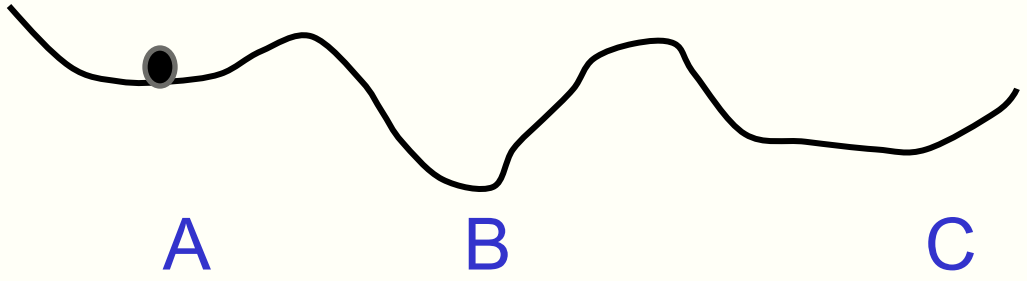
Two difficult computational issues

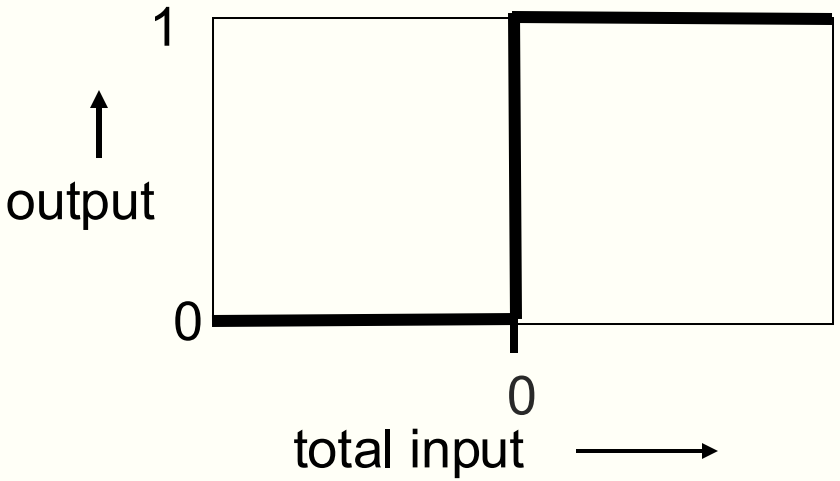
- Using the states of the hidden neurons to represent an interpretation of the input raises two difficult issues:
  - Search: How do we avoid getting trapped in poor local minima of the energy function?
  - Learning: How do we learn the weights on the connections to the hidden neurons and between the hidden neurons?



Noisy networks find better energy minima

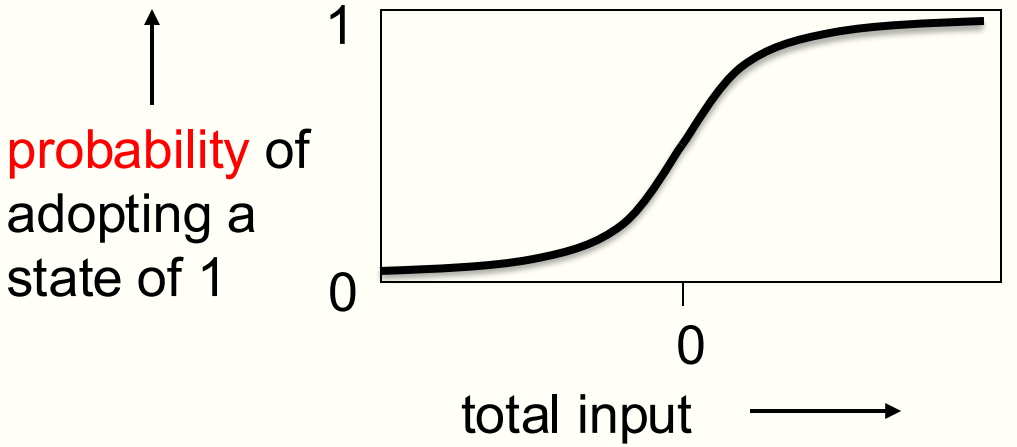
- A Hopfield net always makes decisions that reduce the energy.
  - This makes it impossible to escape from local energy minima.





Stochastic binary neurons use random noise to escape from poor minima.

  - Noise allows the state of the neuron to occasionally go uphill in energy.

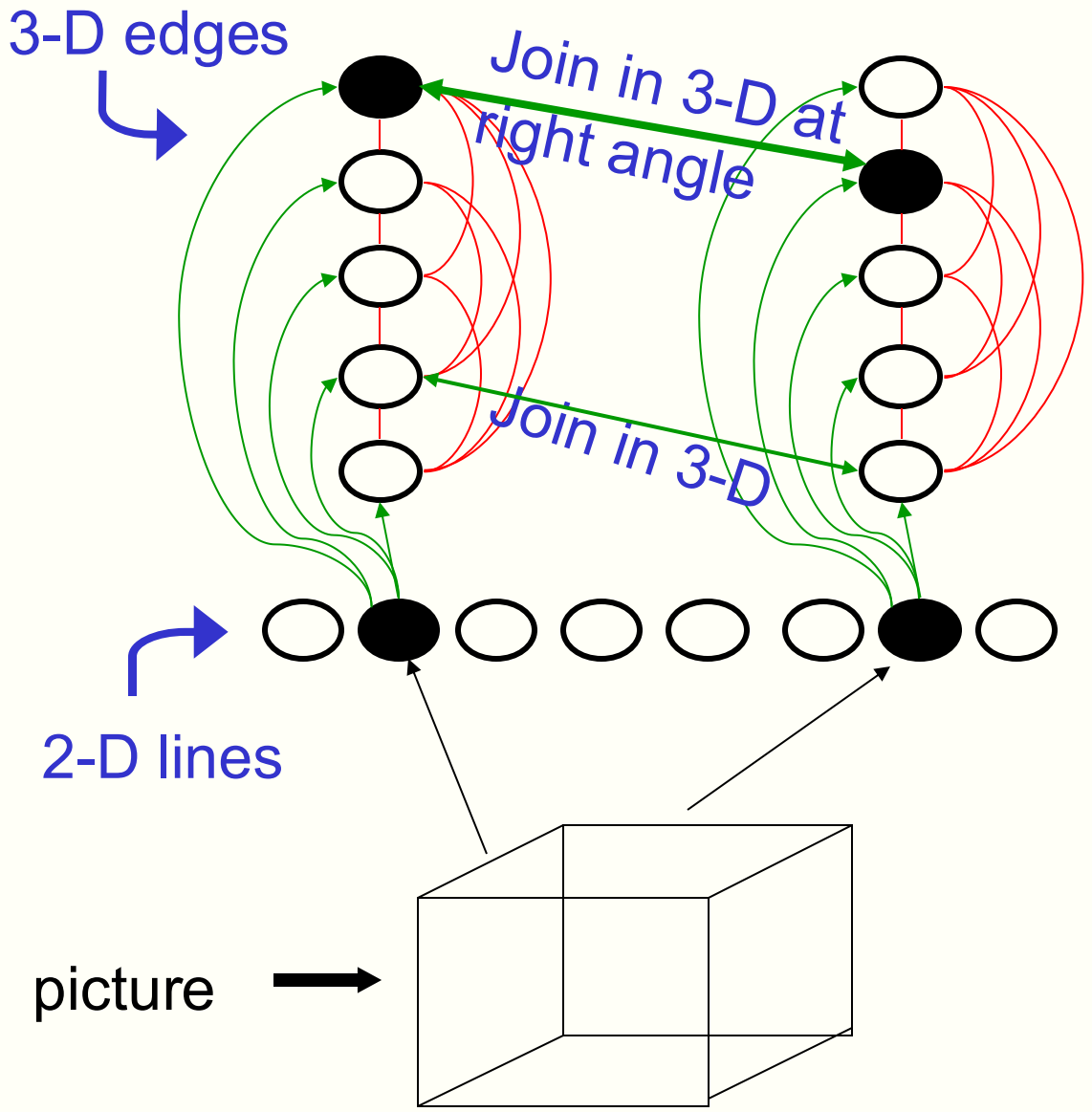


Interpreting a binary image with stochastic binary neurons

- Clamp the binary image on the visible neurons and start from a random binary state for each of the hidden neurons.
- Pick a hidden neuron, compute the total input it is getting given the binary states of the other neurons, and update its state.
  - If the total input is big and positive it nearly always turns on.
  - If the total input is big and negative it nearly always turns off.
  - For smaller total inputs it behaves more randomly.
- Keep picking hidden neurons and updating their states until the whole network approaches thermal equilibrium (explained later)
- The states of the hidden units are then an interpretation of the binary image.
  - Low energy interpretations will be more probable than high energy ones.

Interpreting a line drawing

If we can learn the right weights on the connections, the low energy interpretations will be sensible interpretations.



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 Boltzmann distribution.
  - The probability of finding the system in a configuration is determined solely by the energy of that configuration.
- There is a nice intuitive way to think about thermal equilibrium:
  - Imagine a huge ensemble of systems that all have exactly the same energy function.
  - Each system makes its own random decisions when it updates the states of its neurons
- The probability of a configuration is just the fraction of the systems that have that configuration.
  - After a while these fractions do not change.

Generating a binary image with stochastic binary neurons
(letting the network dream)

- Start by picking a random binary state for each of the hidden and visible neurons.
- Then pick a hidden or visible neuron, compute the total input it is getting given the binary states of the other neurons, and update its state.
  - If the total input is big and positive it nearly always turns on.
  - If the total input is big and negative it nearly always turns off.
  - For smaller total inputs it behaves more randomly.
- Keep picking hidden or visible neurons and updating their states until the whole network approaches thermal equilibrium.
- The states of the visible neurons are then a binary image generated by the network.

The aim of learning in a Boltzmann Machine

- The aim of the learning is to make the images that the network generates when it is dreaming resemble the images it sees when it is perceiving the world.
  - If we can achieve this, the states of the hidden neurons will be a good way to represent the underlying causes of an image.

- Learning the weights in the network is equivalent to figuring out how the hidden neurons should be used to model the structure in the images it perceives.
  - This seems like a very hard problem.

An outrageously optimistic approach

- Start with a neural net that has a lot of hidden neurons with random weights.
- Show it a very large number of images.
- Hope that the network can learn for itself how to use the hidden neurons to model the underlying causes of images.
  - *e. g.* 3-D edges are the underlying causes of 2-D lines in the image.

An amazingly simple learning procedure
(Hinton and Sejnowski, 1983)

- **The wake phase** (when the network is perceiving images)
  - Settle to thermal equilibrium by repeatedly updating the hidden neurons with an image clamped on the visible neurons.
  - For every pair of connected neurons, if they are both on, add a small amount to the weight between them.
- **The sleep phase** (when the network is dreaming)
  - Settle to thermal equilibrium by repeatedly updating both the hidden and the visible neurons.
  - For every pair of connected neurons, if they are both on, subtract a small amount from the weight between them.

What this simple learning procedure achieves

- On average, the learning changes the weights so as to increase the probability that the images generated by the net when it is dreaming will resemble the images it perceives when it is awake.
  - For statisticians and other people who fit models to data: In expectation, learning follows the gradient of the log likelihood.

- To put it another way: The weights change so that the images that the network finds plausible (i.e. low energy) resemble the images it sees when it is awake.
  - The learning lowers the energy of interpretations of real data and raises the energy of the network's fantasies.

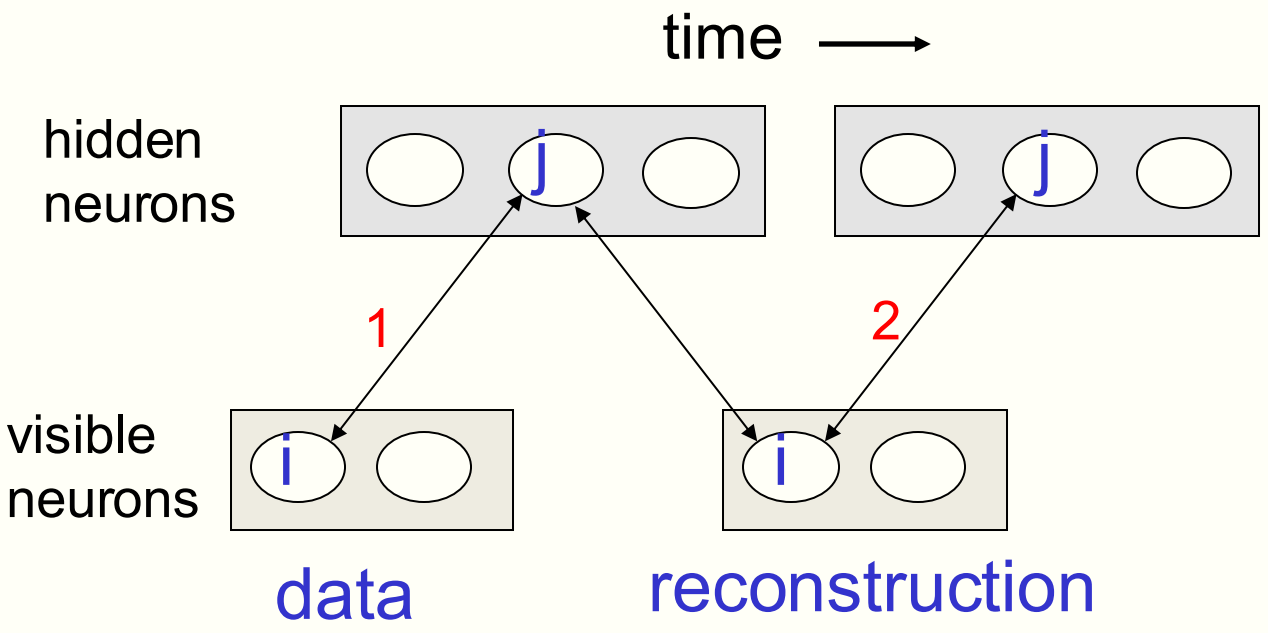
What the process of settling to thermal equilibrium achieves

- Everything that a weight between two neurons needs to know about the other weights in order to learn correctly is conveyed by the difference between two quantities that can be measured locally:
  - How often the two neurons are on together when the network is observing data.
  - How often the two neurons are on together when the network is dreaming.
- The backpropagation algorithm requires a backward pass to convey information about remote weights.
  - This backward pass conveys a different type of signal than the forward pass.
  - This makes it biologically implausible.
- Boltzmann Machines replace the forward and backward passes with a wake and a sleep phase in which the neurons behave in exactly the same way. But they are too slow.

Restricted Boltzmann Machines (17 years later!)

- If there are no connections between hidden neurons, the wake phase becomes much simpler:
  - Clamp an input on the visible neurons and update all of the hidden neurons in parallel.
  - The network reaches thermal equilibrium after a single parallel update of the hidden neurons!
- The sleep phase still requires repeated alternations between updating all of the hidden neurons in parallel and updating all of the visible neurons in parallel.
- But there is a shortcut called "contrastive divergence" which works well in practice.

Contrastive divergence: A very surprising short-cut



Start with a data vector on the visible neurons.

Then update all the hidden neurons in parallel.

Then update all the visible neurons in parallel to get a "reconstruction".

Then update the hidden neurons again.

Measure how often neurons i and j are on together when:

1. The hidden units are in equilibrium with the data.
2. The hidden units are in equilibrium with the reconstruction.

Change the weight between i and j in proportion to the difference between these two measurements.

A practical application of Restricted Boltzmann Machines

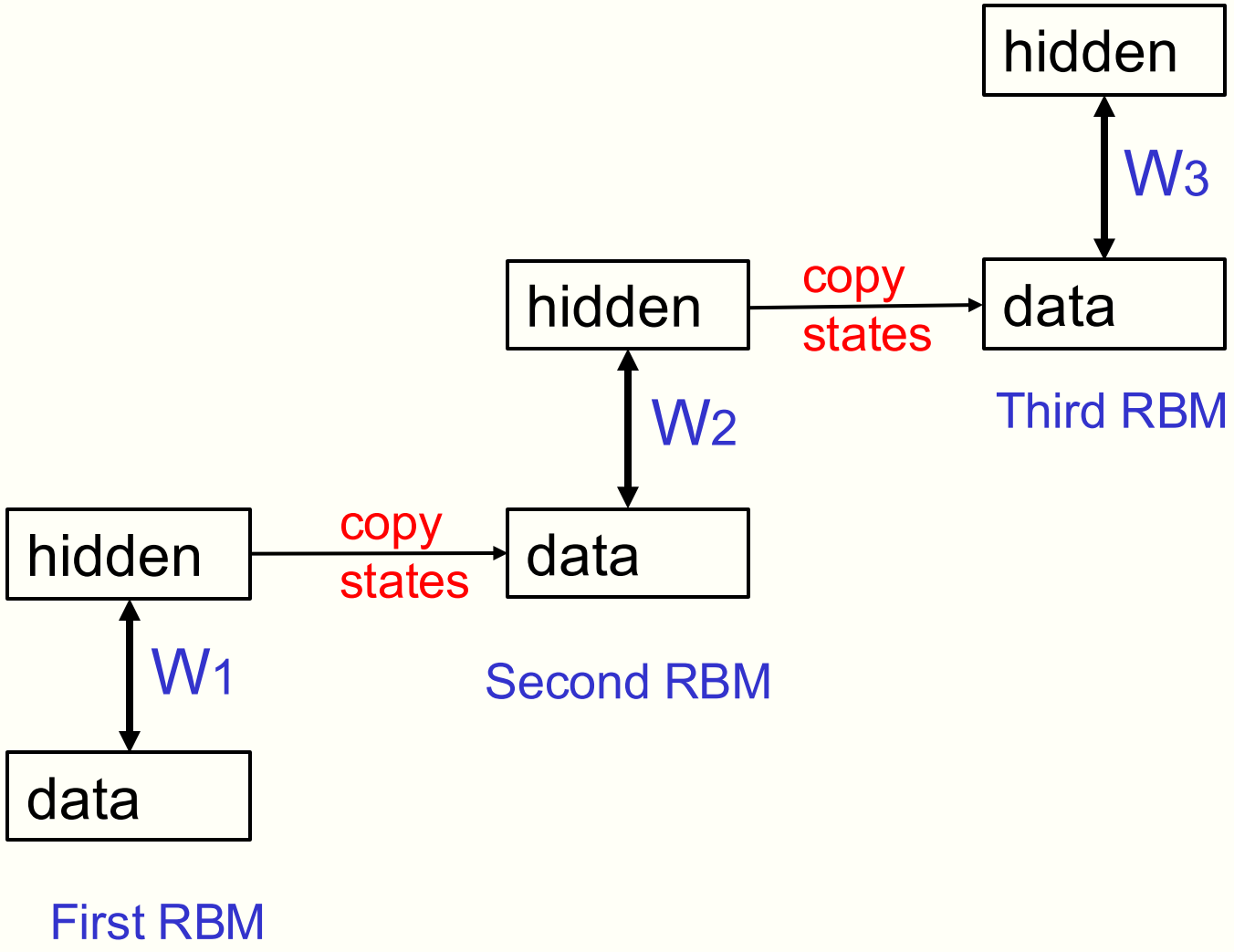
- Netflix combined RBMs with other methods to improve their predictions of which movies users would like.

- But with the restriction to a single layer of hidden neurons it *seems* impossible to learn a hierarchy of feature detectors
  - Hierarchies of feature detectors are required for recognizing words in speech or objects in images.
  - To learn them we require many layers of hidden neurons.

Stacking Restricted Boltzmann Machines

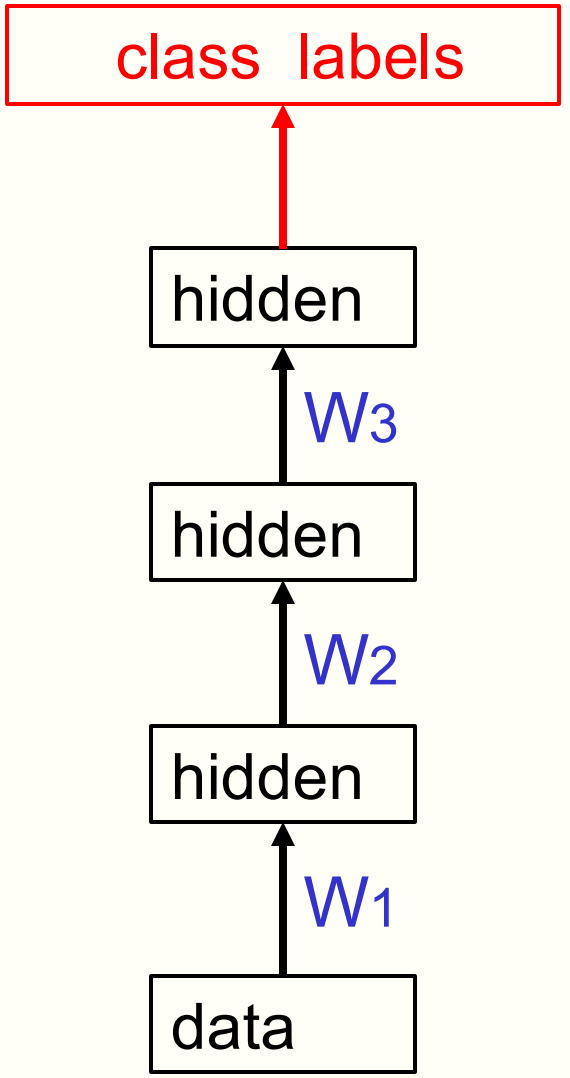
Hinton and Salakhutdinov (2006)

- First use an RBM to learn one layer of hidden features that capture structure in the data.
- Then treat the states of the hidden neurons as data for training a second RBM.
- Keep doing this to learn as many layers of features as desired.



Using stacked RBMs to initialize backpropagation

- After learning a stack of restricted Boltzmann machines, use the learned weights of the RBMs to initialize a feed forward net.
- Add a final layer of output neurons and fine-tune the whole net using backpropagation.
- This way of initializing the weights of a deep net makes backpropagation learn much faster.
- It also needs far fewer labeled images because useful features can be learned without using labels. The labels just fine-tune the features.



The legacy of Boltzmann Machines

- Between 2006 and 2011, pretraining with a stack of RBMs made deep learning work better and helped unleash the deep learning revolution.
- Researchers subsequently found other ways of initializing backpropagation networks so they no longer use stacks of RBMs
- Stacked RBMs were like an enzyme.
  - They helped researchers to make the transition to deep learning, but once this transition was achieved they were no longer needed.
- The idea of using “unlearning” during sleep to avoid the biologically unrealistic backward pass of backpropagation is still an active research area.

THE END