CSC2535: 2013

Lecture 5
Deep Boltzmann Machines

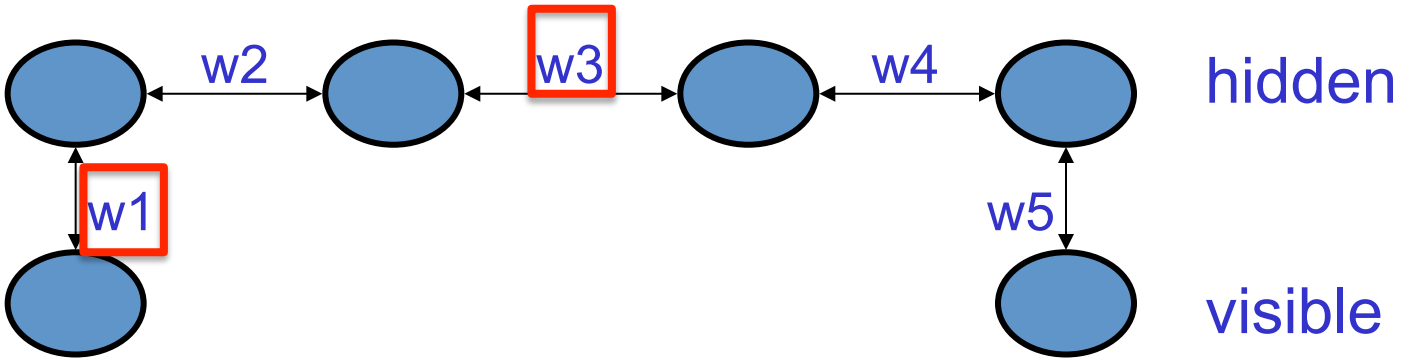
Geoffrey Hinton

The goal of learning

- We want to maximize the product of the probabilities that the Boltzmann machine assigns to the binary vectors in the training set.
  - This is equivalent to maximizing the sum of the log probabilities that the Boltzmann machine assigns to the training vectors.
- It is also equivalent to maximizing the probability that we would obtain exactly the N training cases if we did the following
  - Let the network settle to its stationary distribution N different times with no external input.
  - Sample the visible vector once each time.

Why the learning could be difficult

Consider a chain of units with visible units at the ends



If the training set consists of (1,0) and (0,1) we want the product of all the weights to be negative.

So to know how to change w1 or w5 we must know w3.

A very surprising fact

- Everything that one weight needs to know about the other weights and the data is contained in the difference of two correlations.

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

Derivative of log probability of one training vector, v under the model.

Expected value of product of states at thermal equilibrium when v is clamped on the visible units

Expected value of product of states at thermal equilibrium with no clamping

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

Why is the derivative so simple?

- The probability of a global configuration at thermal equilibrium is an exponential function of its energy.
  - So settling to equilibrium makes the log probability a linear function of the energy.

- The energy is a linear function of the weights and states, so:

$$-\frac{\partial E}{\partial w_{ij}} = s_i s_j$$

- The process of settling to thermal equilibrium propagates information about the weights.
  - We don't need backprop.

An inefficient way to collect the statistics required for learning

Hinton and Sejnowski (1983)

- Positive phase: Clamp a data vector on the visible units and set the hidden units to random binary states.
  - Update the hidden units one at a time until the network reaches thermal equilibrium at a temperature of 1.
  - Sample $< s_i s_j >$ for every connected pair of units.
  - Repeat for all data vectors in the training set and average.
- Negative phase: Set all the units to random binary states.
  - Update all the units one at a time until the network reaches thermal equilibrium at a temperature of 1.
  - Sample $< s_i s_j >$ for every connected pair of units.
  - Repeat many times (how many?) and average to get good estimates.

A better way of collecting the statistics

- If we start from a random state, it may take a long time to reach thermal equilibrium.
  - Also, its very hard to tell when we get there.
- Why not start from whatever state you ended up in last time you saw that datavector?
  - This stored state is called a “particle”.

Using particles that persist to get a “warm start” has a big advantage:

- If we were at equilibrium last time and we only changed the weights a little, we should only need a few updates to get back to equilibrium.

Neal's method for collecting the statistics (Neal 1992)

- Positive phase: Keep a set of "data-specific particles", one per training case. Each particle has a current value that is a configuration of the hidden units.
  - Sequentially update all the hidden units a few times in each particle with the relevant datavector clamped.
  - For every connected pair of units, average $s_i s_j$ over all the data-specific particles.
- Negative phase: Keep a set of "fantasy particles". Each particle has a value that is a global configuration.
  - Sequentially update all the units in each fantasy particle a few times.
  - For every connected pair of units, average $s_i s_j$ over all the fantasy particles.

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

Adapting Neal's approach to handle mini-batches

- Neal's approach does not work well with mini-batches.
  - By the time we get back to the same datavector again, the weights will have been updated many times.
  - But the data-specific particle will not have been updated so it may be far from equilibrium.
- A strong assumption about how we understand the world:
  - When a datavector is clamped, we will assume that the set of good explanations (i.e. hidden unit states) is uni-modal.
  - i.e. we restrict ourselves to learning models in which one sensory input vector does not have multiple very different explanations.

The simple mean field approximation

- If we want to get the statistics right, we need to update the units stochastically and sequentially.
- But if we are in a hurry we can use probabilities instead of binary states and update the units in parallel.
- To avoid biphasic oscillations we can use damped mean field.

$$prob(s_i = 1) = \sigma\left(b_i + \sum_j s_j w_{ij}\right)$$

$$p_i^{t+1} = \sigma\left(b_i + \sum_j p_j^t w_{ij}\right)$$

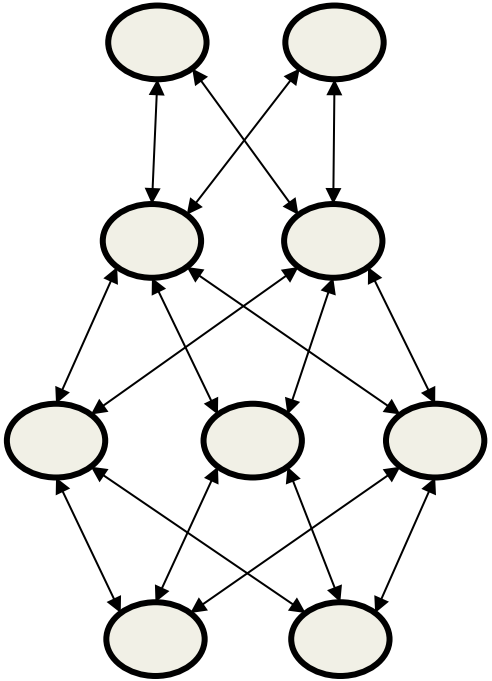
$$p_i^{t+1} = \lambda p_i^t + (1-\lambda)\,\sigma\left(b_i + \sum_j p_j^t w_{ij}\right)$$

An efficient mini-batch learning procedure for Boltzmann Machines (Salakhutdinov & Hinton 2012)

- Positive phase: Initialize all the hidden probabilities at 0.5.
  - Clamp a datavector on the visible units.
  - Update all the hidden units in parallel until convergence using mean field updates.
  - After the net has converged, record $p_i p_j$ for every connected pair of units and average this over all data in the mini-batch.
- Negative phase: Keep a set of "fantasy particles". Each particle has a value that is a global configuration.
  - Sequentially update all the units in each fantasy particle a few times.
  - For every connected pair of units, average $s_i s_j$ over all the fantasy particles.

Making the updates more parallel

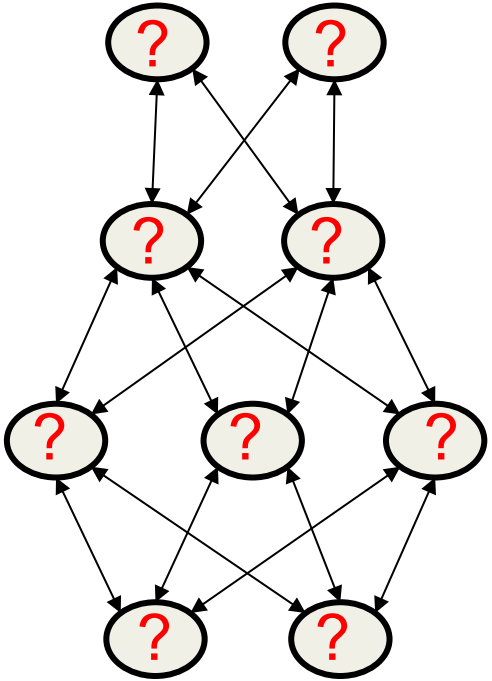
- In a general Boltzmann machine, the stochastic updates of units need to be sequential.
- There is a special architecture that allows alternating parallel updates which are much more efficient:
  - No connections within a layer.
  - No skip-layer connections.
- This is called a Deep Boltzmann Machine (DBM)
  - It's a general Boltzmann machine with a lot of missing connections.



visible

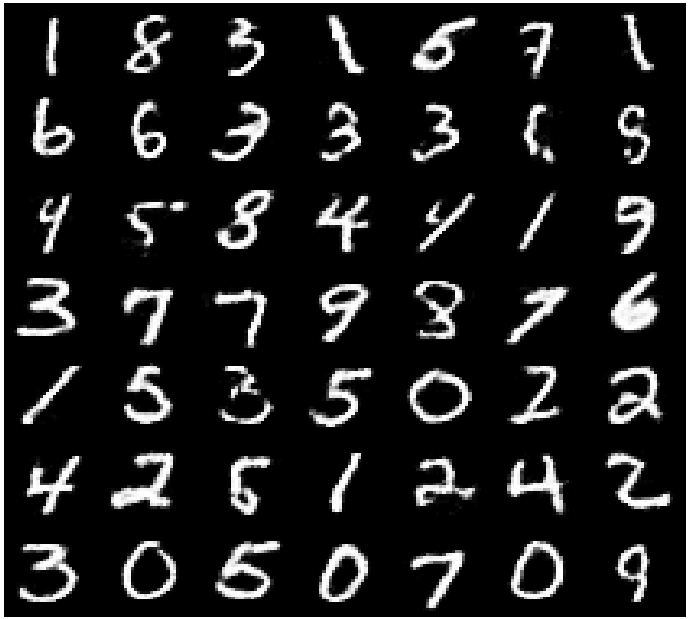
Making the updates more parallel

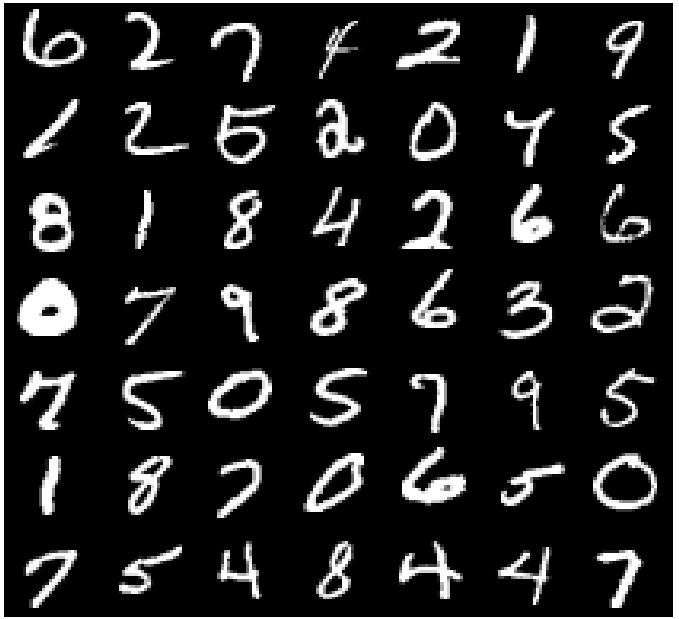
- In a general Boltzmann machine, the stochastic updates of units need to be sequential.
- There is a special architecture that allows alternating parallel updates which are much more efficient:
  - No connections within a layer.
  - No skip-layer connections.
- This is called a Deep Boltzmann Machine (DBM)
  - It's a general Boltzmann machine with a lot of missing connections.



Can a DBM learn a good model of the MNIST digits?

Do samples from the model look like real data?





A puzzle

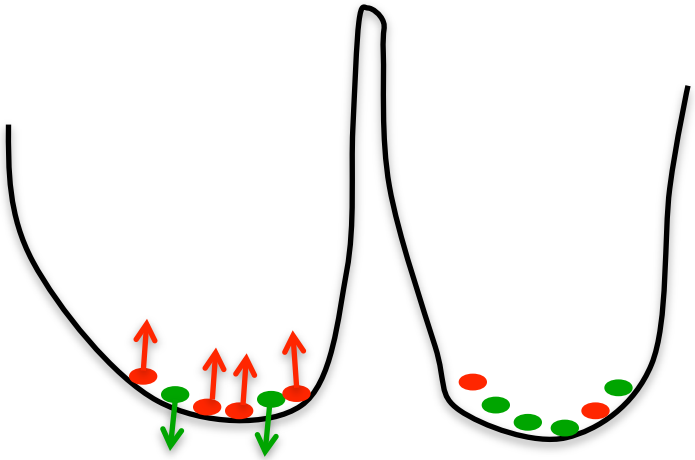
- Why can we estimate the “negative phase statistics” well with only 100 negative examples to characterize the whole space of possible configurations?
  - For all interesting problems the GLOBAL configuration space is highly multi-modal.
  - How does it manage to find and represent all the modes with only 100 particles?

The learning raises the effective mixing rate.

- The learning interacts with the Markov chain that is being used to gather the "negative statistics" (*i.e.* the data-independent statistics).
  - We cannot analyse the learning by viewing it as an outer loop and the gathering of statistics as an inner loop.
- Wherever the fantasy particles outnumber the positive data, the energy surface is raised.
  - This makes the fantasies rush around hyperactively.
  - They move around MUCH faster than the mixing rate of the Markov chain defined by the static current weights.

How fantasy particles move between the model's modes

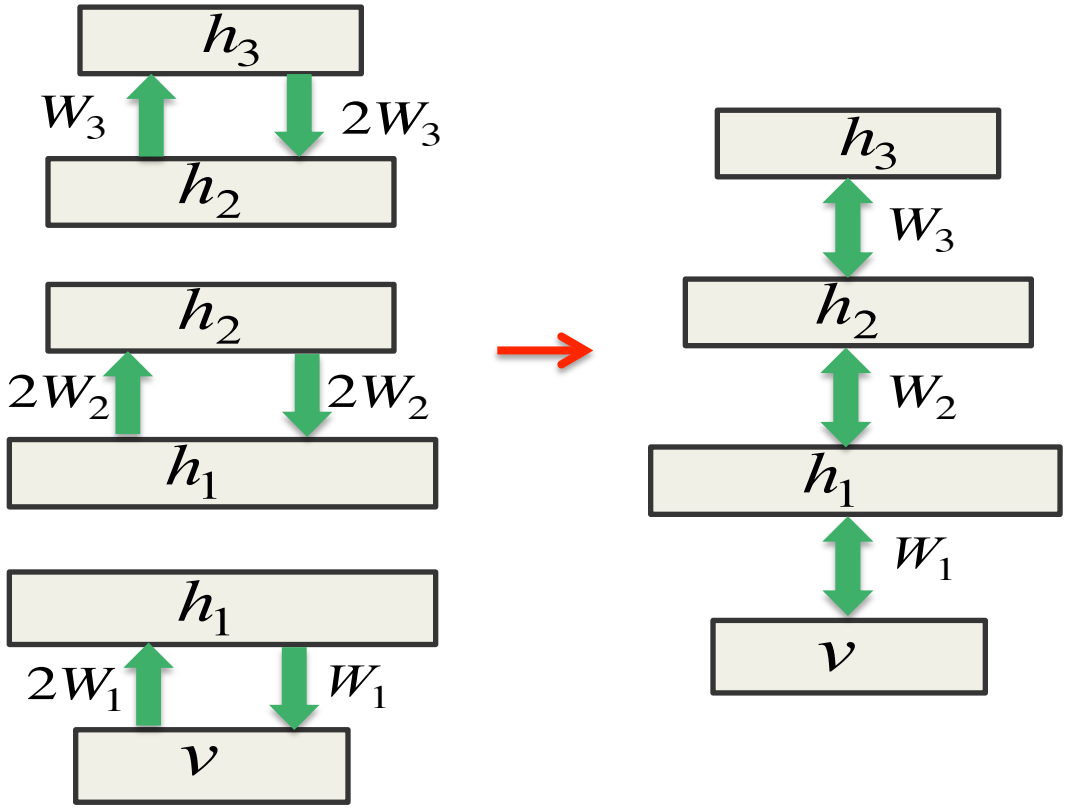
- If a mode has more fantasy particles than data, the energy surface is raised until the fantasy particles escape.
  - This can overcome energy barriers that would be too high for the Markov chain to jump in a reasonable time.
- The energy surface is being changed to help mixing in addition to defining the model.
- Once the fantasy particles have filled in a hole, they rush off somewhere else to deal with the next problem.
  - They are like investigative journalists.



This minimum will get filled in by the learning until the fantasy particles escape.

Pre-training a DBM: Combining three RBMs to make a DBM

- The top and bottom RBMs must be pre-trained with the weights in one direction twice as big as in the other direction.
  - This can be justified!
- The middle layers do geometric model averaging.



Modeling the joint density of images and captions
(Srivastava and Salakhutdinov, NIPS 2012)

- Goal: To build a joint density model of captions and standard computer vision feature vectors extracted from real photographs.
  - This needs a lot more computation than building a joint density model of labels and digit images!

1. Train a multilayer model of images.
2. Train a separate multilayer model of word-count vectors.
3. Then add a new top layer that is connected to the top layers of both individual models.
   - Use further joint training of the whole system to allow each modality to improve the earlier layers of the other modality.

Modeling the joint density of images and captions
(Srivastava and Salakhutdinov, NIPS 2012)

- Instead of using a deep belief net, use a deep Boltzmann machine that has symmetric connections between all pairs of layers.
  - Further joint training of the whole DBM allows each modality to improve the earlier layers of the other modality.
  - That's why they used a DBM.
  - They could also have used a DBN and done generative fine-tuning with contrastive wake-sleep.