CSC321 Tutorial 9: Review of Boltzmann machines and simulated annealing

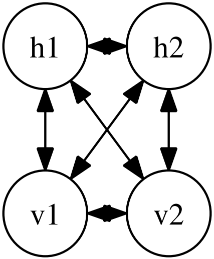
(Slides based on Lecture 16-18 and selected readings)

Yue Li
Email: yueli@cs.toronto.edu

Wed 11-12 March 19
Fri 10-11 March 21

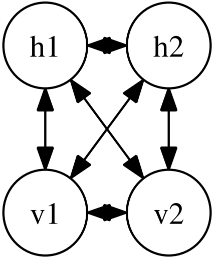
Outline

General Boltzmann Machines [1]



- Network is symmetrically connected
- Allow connection between visible and hidden units
- Each binary unit makes stochastic decision to be either on or off
- The configuration of the network dictates its "energy"
- At the equilibrium state of the network, the likelihood is defined as the exponentiated negative energy known as the Boltzmann distribution

Boltzmann Distribution



$$E(\mathbf{v}, \mathbf{h}) = \sum_i s_i b_i + \sum_{i<j} s_i s_j w_{ij} \qquad (1)$$

$$P(\mathbf{v}) = \frac{\sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}))}{\sum_{\mathbf{v},\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}))} \qquad (2)$$

where $\mathbf{v}$ and $\mathbf{h}$ are visible and hidden units, w_{ij}'s are connection weights b/w visible-visible, hidden-hidden, and visible-hidden units, $E(\mathbf{v}, \mathbf{h})$ is the energy function

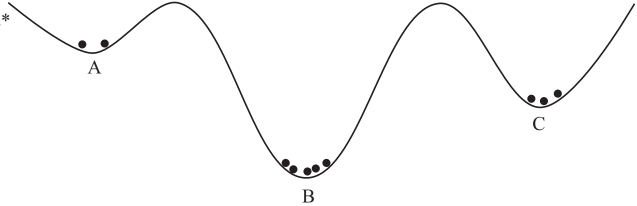
Two problems:

1. Given w_{ij}, how to achieve thermal equilibrium of $P(\mathbf{v}, \mathbf{h})$ over all possible network config. including visible & hidden units
2. Given $\mathbf{v}$, learn w_{ij} to maximize $P(\mathbf{v})$

Thermal equilibrium

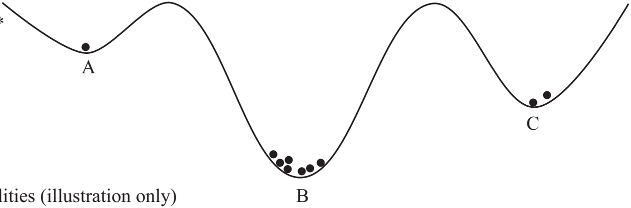
Transition probabilities*
at high temperature T:

	A	B	C
A	0.01	0.1	0.02
B	0.1	0.02	0.1
C	0.1	0.2	0.01



Transition probabilities*
at low temperature T

	A	B	C
A	1e-2	1e-3	1e-4
B	1e-9	0.3	0
C	1e-8	1e-3	0.01



*unnormalized probabilities (illustration only)

Thermal equilibrium is a difficult concept (Lec 16):

- It 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.

Simulated annealing [2]

Scale Boltzmann factor by T ("temperature"):

$$P(\mathbf{s}) = \frac{\exp(-E(\mathbf{s})/T)}{\sum_{\mathbf{s}} \exp(-E(\mathbf{s})/T)} \propto \exp(-E(\mathbf{s})/T) \tag{3}$$

where $\mathbf{s} = \{\mathbf{v}, \mathbf{h}\}$.
At state $t+1$, a proposed state $\mathbf{s}^*$ is compared with current state $\mathbf{s}^t$:

$$\frac{P(\mathbf{s}^*)}{P(\mathbf{s}^t)} = \exp\left(-\frac{E(\mathbf{s}^*) - E(\mathbf{s}^t)}{T}\right) = \exp\left(-\frac{\Delta E}{T}\right) \tag{4}$$

$$\mathbf{s}^{\mathrm{t}+1} \leftarrow \begin{cases} \mathbf{s}^*, & \text{if } \Delta E < 0 \textbf{ or } \exp(-\Delta E/T) > \texttt{rand(0,1)} \\ \mathbf{s}^t & \text{otherwise} \end{cases} \tag{5}$$

NB: T controls the stochastic of the transition:
when $\Delta E > 0$, $T \uparrow \Rightarrow \exp(-\Delta E/T) \uparrow$; $T \downarrow \Rightarrow \exp(-\Delta E/T) \downarrow$

A nice demo of simulated annealing from Wikipedia:

http://www.cs.utoronto.ca/~yueli/CSC321_UTM_2014_files/Hill_Climbing_with_Simulated_Annealing.gif

Note: simulated annealing is not used in the Restricted Boltzmann Machine algorithm discussed below. Instead, Gibbs sampling is used. Nonetheless, it is still a nice concept and has been used in many many other applications (the paper by Kirkpatrick et al. (1983) [2] has been has been cited for over 30,000 times based on Google Scholar!)

Learning weights from Boltzmann Machines is difficult

$$P(\mathbf{v}) = \prod_{n=1}^{N} \frac{\sum_{\mathbf{h}} \exp(-E(\mathbf{v},\mathbf{h}))}{\sum_{\mathbf{v},\mathbf{h}} \exp(-E(\mathbf{v},\mathbf{h}))} = \prod_{n} \frac{\sum_{\mathbf{h}} \exp(-\sum_i s_i b_i - \sum_{i<j} s_i s_j w_{ij})}{\sum_{\mathbf{v},\mathbf{h}} \exp(-\sum_i s_i b_i - \sum_{i<j} s_i s_j w_{ij})}$$

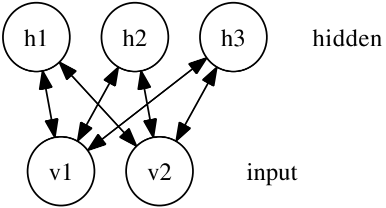
$$\begin{aligned}
\log P(\mathbf{v}) &= \sum_n \Big(\log \sum_{\mathbf{h}} \exp(-\sum_i s_i b_i - \sum_{i<j} s_i s_j w_{ij}) \\
&\quad - \log \sum_{\mathbf{v},\mathbf{h}} \exp(-\sum_i s_i b_i - \sum_{i<j} s_i s_j w_{ij}) \Big) \\
\frac{\partial \log P(\mathbf{v},\mathbf{h})}{\partial w_{ij}} &= \sum_n \Big(\sum_{s_i, s_j} s_i s_j P(\mathbf{h}|\mathbf{v}) - \sum_{s_i, s_j} s_i s_j P(\mathbf{v},\mathbf{h}) \Big) \\
&= < s_i s_j >_{data} - < s_i s_j >_{model}
\end{aligned}$$

where $< x >$ is the expected value of x. $s_i, s_j \in \{\mathbf{v},\mathbf{h}\}$. $< s_i s_j >_{model}$ is difficult or takes long time to compute.

Restricted Boltzmann Machine (RBM) [3]

- A simple **unsupervised** learning module;
- Only one layer of hidden units and one layer of visible units;
- No connection between hidden units nor between visible units (i.e. a special case of Boltzmann Machine);
- Edges are still undirected or bi-directional

e.g., an RBM with 2 visible and 3 hidden units:



Objective function of RBM - maximum likelihood:

$$E(\mathbf{v},\mathbf{h}|\theta) = \sum_{ij} w_{ij} v_i h_j + \sum_i b_i v_i + \sum_j b_j h_j$$

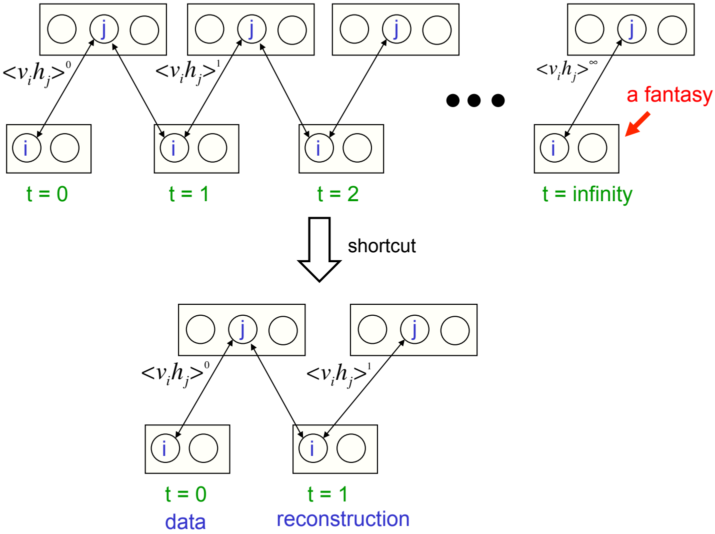
$$p(\mathbf{v}|\theta) = \prod_{n=1}^{N} \sum_{\mathbf{h}} p(\mathbf{v},\mathbf{h}|\theta) = \prod_{n=1}^{N} \frac{\sum_{\mathbf{h}} \exp(-E(\mathbf{v},\mathbf{h}|\theta))}{\sum_{\mathbf{v},\mathbf{h}} \exp(-E(\mathbf{v},\mathbf{h}|\theta))}$$

$$\log p(\mathbf{v}|\theta) = \sum_{n=1}^{N} \left(\log \sum_{\mathbf{h}} \exp(-E(\mathbf{v},\mathbf{h}|\theta)) - \log \sum_{\mathbf{v},\mathbf{h}} \exp(-E(\mathbf{v},\mathbf{h}|\theta)) \right)$$

$$\frac{\partial \log p(\mathbf{v}|\theta)}{\partial w_{ij}} = \sum_{n=1}^{N} \left[v_i \sum_{\mathbf{h}} h_j p(\mathbf{h}|\mathbf{v}) - \sum_{\mathbf{v},\mathbf{h}} v_i h_j p(\mathbf{v},\mathbf{h}) \right]$$

$$= \mathbb{E}_{\text{data}}[v_i h_j] - \mathbb{E}_{\text{model}}[\hat{v}_i \hat{h}_j] \equiv < v_i h_j >_{\text{data}} - < \hat{v}_i \hat{h}_j >_{\text{model}}$$

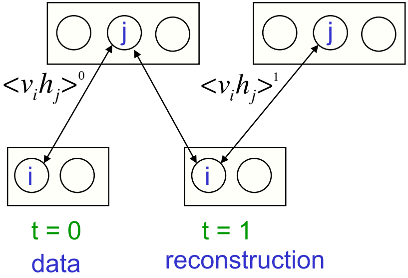
But $< \hat{v}_i \hat{h}_j >_{\text{model}}$ is still too large to estimate, we apply Markov Chain Monte Carlo (MCMC) (i.e., Gibbs sampling) to estimate it.



$$\frac{\partial \log p(\mathbf{v}^0)}{\partial w_{ij}} = < h_j^0(v_i^0 - v_i^1) > + < v_i^1(h_j^0 - h_j^1) > + < h_j^1(v_i^1 - v_i^2) > + \dots$$

$$= < v_i^0 h_j^0 > - < v_i^\infty h_j^\infty > \approx < v_i^0 h_j^0 > - < v_i^1 h_j^1 >$$

How Gibbs sampling works



1. Start with a training vector on the visible units
2. Update all the hidden units in parallel
3. Update all the visible units in parallel to get a "reconstruction"
4. Update the hidden units again

$$\Delta w_{ij} = \epsilon(< v_i^0 h_j^0 > - < v_i^1 h_j^1 >) \tag{6}$$

Approximate maximum likelihood learning

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} \approx \frac{1}{N} \sum_{n=1}^{N} \left[v_i^{(n)} h_j^{(n)} - \hat{v}_i^{(n)} \hat{h}_j^{(n)} \right] \tag{7}$$

where

- $v_i^{(n)}$ is the value of i^{th} visible (input) unit for n^{th} training case;
- $h_j^{(n)}$ is the value of j^{th} hidden unit;
- $\hat{v}_i^{(n)}$ is the **sampled** value for the i^{th} visible unit or the **negative data** generated based on $h_j^{(n)}$ and w_{ij};
- $\hat{h}_i^{(n)}$ is the **sampled** value for the j^{th} hidden unit or the **negative hidden activities** generated based on $\hat{v}_i^{(n)}$ and w_{ij};

Still how exactly the **negative data** and **negative hidden activities** are generated?

1. Positive ("wake") phase (clamp the visible units with data):
    - Use input data to generate hidden activities:

$$h_j = \frac{1}{1 + \exp(-\sum_i v_i w_{ij} - b_j)}$$

   Sample hidden state from Bernoulli distribution:

$$h_j \leftarrow \begin{cases} 1, & \text{if } h_j > \texttt{rand(0,1)} \\ 0, & \text{otherwise} \end{cases}$$

2. Negative ("sleep") phase (unclamp the visible units from data):
    - Use h_j to generate **negative data**:

$$\hat{v}_i = \frac{1}{1 + \exp(-\sum_j w_{ij} h_j - b_i)}$$

    - Use negative data $\hat{v}_i$ to generate **negative hidden activities**:

$$\hat{h}_j = \frac{1}{1 + \exp(-\sum_i \hat{v}_i w_{ij} - b_j)}$$

RBM learning algorithm (con'td) - Learning

$$\Delta w_{ij}^{(t)} = \eta \Delta w_{ij}^{(t-1)} + \epsilon_w \left(\frac{\partial \log p(\mathbf{v}|\theta)}{\partial w_{ij}} - \lambda w_{ij}^{(t-1)} \right)$$

$$\Delta b_i^{(t)} = \eta \Delta b_i^{(t-1)} + \epsilon_{vb} \frac{\partial \log p(\mathbf{v}|\theta)}{\partial b_i}$$

$$\Delta b_j^{(t)} = \eta \Delta b_j^{(t-1)} + \epsilon_{hb} \frac{\partial \log p(\mathbf{v}|\theta)}{\partial b_j}$$

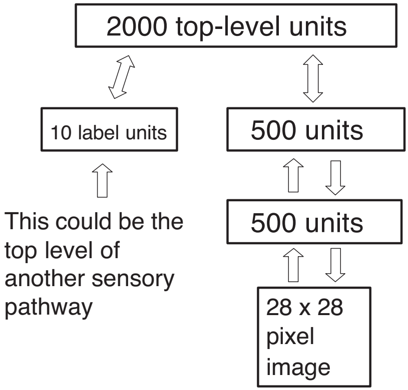
where

$$\frac{\partial \log p(\mathbf{v}|\theta)}{\partial w_{ij}} \approx \frac{1}{N} \sum_{n=1}^{N} \left[v_i^{(n)} h_j^{(n)} - \hat{v}_i^{(n)} \hat{h}_j^{(n)} \right]$$

$$\frac{\partial \log p(\mathbf{v}|\theta)}{\partial b_i} \approx \frac{1}{N} \sum_{n=1}^{N} \left[v_i^{(n)} - \hat{v}_i^{(n)} \right]$$

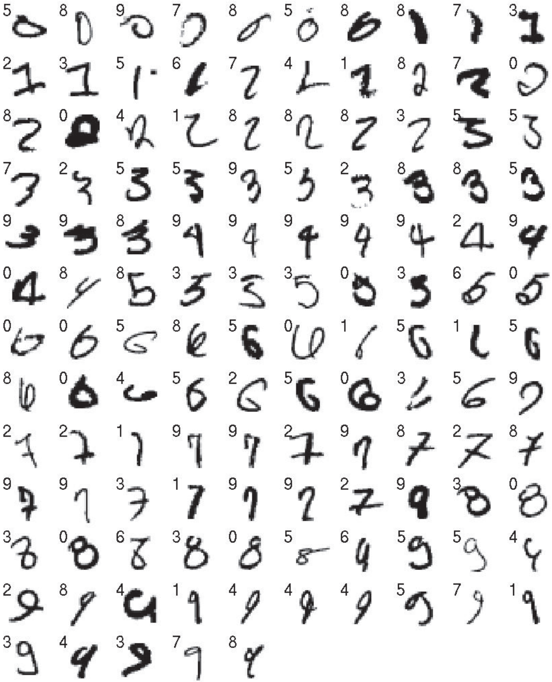
$$\frac{\partial \log p(\mathbf{v}|\theta)}{\partial b_j} \approx \frac{1}{N} \sum_{n=1}^{N} \left[h_j^{(n)} - \hat{h}_j^{(n)} \right]$$

Deep learning using stacked RBM on images [3]



- A greedy learning algorithm
- Bottom layer encode the 28×28 handwritten image
- The upper adjacent layer of 500 hidden units are used for distributed representation of the images
- The next 500-units layer and the top layer of 2000 units called "associative memory" layers, which have undirected connections between them
- The very top layer encodes the class labels with softmax

- The network trained on 60,000 training cases achieved 1.25% test error on classifying 10,000 MNIST testing cases
- On the right are the incorrectly classified images, where the predictions are on the top left corner (Figure 6, Hinton et al., 2006)



Let model generate 28×28 images for specific class label

Each row shows 10 samples from the generative model with a particular label clamped on. The top-level associative memory is run for 1000 iterations of alternating Gibbs sampling (Figure 8, Hinton et al., 2006).

Look into the mind of the network

Each row shows 10 samples from the generative model with a particular label clamped on. . . . Subsequent columns are produced by 20 iterations of alternating Gibbs sampling in the associative memory (Figure 9, Hinton et al., 2006).

Deep learning using stacked RBM on handwritten images (Hinton et al., 2006)

A real-time demo from Prof. Hinton's webpage:
`http://www.cs.toronto.edu/~hinton/digits.html`

Further References

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski.
A learning algorithm for boltzmann machines.
Cognitive science, 9(1):147–169, 1985.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi.
Optimization by simulated annealing.
Science, 220(4598):671–680, 1983.

G Hinton and S Osindero.
A fast learning algorithm for deep belief nets.
Neural computation, 2006.