Deep belief networks

Deep Learning Lecture 12

Samuel Cheng

School of ECE
University of Oklahoma

Spring, 2017
(Slides credit to Larochelle)

Table of Contents

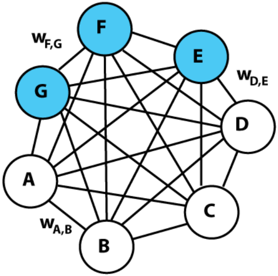
Review

- We looked into different variations of RNN in the last several weeks (LSTMs, memory networks, neural Turing machines)
- We will look into unsupervised learning for the next couple lectures
- We will first discuss restricted Boltzmann machines and deep belief networks today

Unsupervised learning

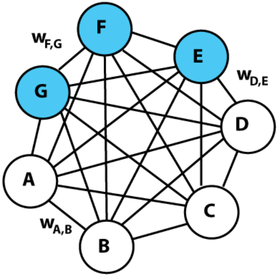
- We mostly looked into supervised learning problem throughout the course, where essentially the expected outputs (labels) are always given for the training data
- For unsupervised learning, we are only given with data signals but appropriate "labels" of the signals are not known
  - Clustering is one major subproblem but not the only one
  - For example, another problem can be data modeling. How to create generative model for the given data

Boltzmann machines



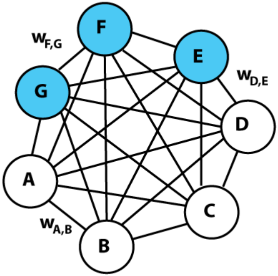
- Boltzmann machines were invented by Geoffrey Hinton and Terry Sejnowski in 1985
- It is a binary generative model
- Probability of a "configuration" is government by the Boltzmann distribution $\frac{\exp(-E(\mathbf{x}))}{Z}$, where Z is a normalization factor and called the partition function (a name originated from statistical physics)
- The energy function $E(\mathbf{x})$ has a very simple form $E(\mathbf{x}) = -\mathbf{x}^T W\mathbf{x} - \mathbf{c}^T\mathbf{x}$
- Typically some variables are hidden whereas others are visible

Boltzmann machines



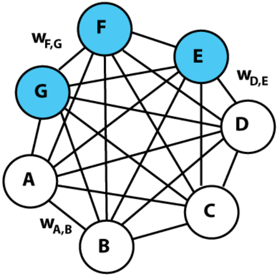
- Boltzmann machines were invented by Geoffrey Hinton and Terry Sejnowski in 1985
- It is a binary generative model
- Probability of a "configuration" is government by the Boltzmann distribution $\frac{\exp(-E(\mathbf{x}))}{Z}$, where Z is a normalization factor and called the partition function (a name originated from statistical physics)
- The energy function $E(\mathbf{x})$ has a very simple form $E(\mathbf{x}) = -\mathbf{x}^T W\mathbf{x} - \mathbf{c}^T\mathbf{x}$
- Typically some variables are hidden whereas others are visible

Boltzmann machines



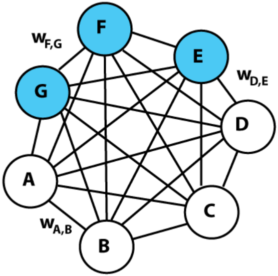
- Boltzmann machines were invented by Geoffrey Hinton and Terry Sejnowski in 1985
- It is a binary generative model
- Probability of a "configuration" is government by the Boltzmann distribution $\frac{\exp(-E(\mathbf{x}))}{Z}$, where Z is a normalization factor and called the partition function (a name originated from statistical physics)
- The energy function $E(\mathbf{x})$ has a very simple form $E(\mathbf{x}) = -\mathbf{x}^T W \mathbf{x} - \mathbf{c}^T \mathbf{x}$
- Typically some variables are hidden whereas others are visible

Boltzmann machines



- Boltzmann machines were invented by Geoffrey Hinton and Terry Sejnowski in 1985
- It is a binary generative model
- Probability of a "configuration" is government by the Boltzmann distribution $\frac{\exp(-E(\mathbf{x}))}{Z}$, where Z is a normalization factor and called the partition function (a name originated from statistical physics)
- The energy function $E(\mathbf{x})$ has a very simple form $E(\mathbf{x}) = -\mathbf{x}^T W\mathbf{x} - \mathbf{c}^T\mathbf{x}$
- Typically some variables are hidden whereas others are visible

Boltzmann machines



- Boltzmann machines were invented by Geoffrey Hinton and Terry Sejnowski in 1985
- It is a binary generative model
- Probability of a "configuration" is government by the Boltzmann distribution $\frac{\exp(-E(\mathbf{x}))}{Z}$, where Z is a normalization factor and called the partition function (a name originated from statistical physics)
- The energy function $E(\mathbf{x})$ has a very simple form $E(\mathbf{x}) = -\mathbf{x}^T W \mathbf{x} - \mathbf{c}^T \mathbf{x}$
- Typically some variables are hidden whereas others are visible

Restricted Boltzmann machines

- Boltzmann machine is a very powerful model. But with unconstrained connectivity, there are not known *efficient* methods to learn data and conduct inference for practical problems
- Consequently, restricted Boltzmann machine (RBM) (originally called Harmonium) was introduced by Paul Smolensky in 1986. It restricted the hidden units and the visible units from connecting to themselves
- The model rose to prominence after fast learning algorithm was invented by Hinton and his collaborators in mid-2000s

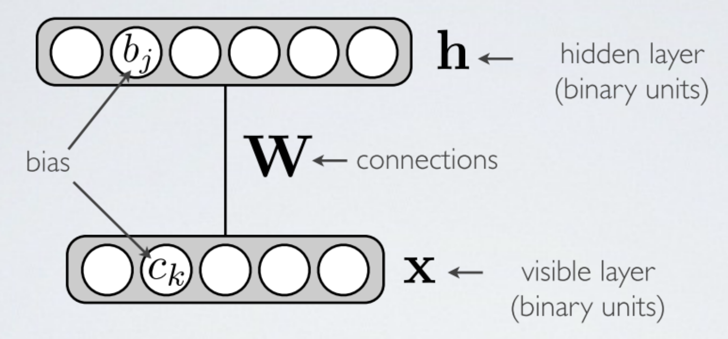
Restricted Boltzmann machines

- Boltzmann machine is a very powerful model. But with unconstrained connectivity, there are not known *efficient* methods to learn data and conduct inference for practical problems
- Consequently, restricted Boltzmann machine (RBM) (originally called Harmonium) was introduced by Paul Smolensky in 1986. It restricted the hidden units and the visible units from connecting to themselves
- The model rose to prominence after fast learning algorithm was invented by Hinton and his collaborators in mid-2000s

Restricted Boltzmann machines

- Boltzmann machine is a very powerful model. But with unconstrained connectivity, there are not known *efficient* methods to learn data and conduct inference for practical problems
- Consequently, restricted Boltzmann machine (RBM) (originally called Harmonium) was introduced by Paul Smolensky in 1986. It restricted the hidden units and the visible units from connecting to themselves
- The model rose to prominence after fast learning algorithm was invented by Hinton and his collaborators in mid-2000s

Restricted Boltzmann machines

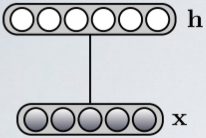


Energy function: $E(\mathbf{x},\mathbf{h}) = -\mathbf{h}^T W\mathbf{x} - \mathbf{c}^T\mathbf{x} - \mathbf{b}^T\mathbf{h}$

Distribution:

$$p(\mathbf{x},\mathbf{h}) = \frac{\exp(-E(\mathbf{x},\mathbf{h}))}{Z} = \frac{\exp(\mathbf{h}^T W\mathbf{x})\exp(\mathbf{c}^T\mathbf{x})\exp(\mathbf{b}^T\mathbf{h})}{Z}$$

Conditional probabilities

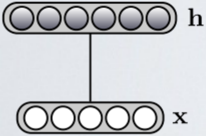


$$p(\mathbf{h}|\mathbf{x}) = \prod_j p(h_j|\mathbf{x})$$

$$p(h_j = 1|\mathbf{x}) = \frac{1}{1 + \exp(-(b_j + \mathbf{W}_{j\cdot}\mathbf{x}))}$$

$$= \mathrm{sigm}(b_j + \mathbf{W}_{j\cdot}\mathbf{x})$$

j th row of $\mathbf{W}$



$$p(\mathbf{x}|\mathbf{h}) = \prod_k p(x_k|\mathbf{h})$$

$$p(x_k = 1|\mathbf{h}) = \frac{1}{1 + \exp(-(c_k + \mathbf{h}^\top \mathbf{W}_{\cdot k}))}$$

$$= \mathrm{sigm}(c_k + \mathbf{h}^\top \mathbf{W}_{\cdot k})$$

k th column of $\mathbf{W}$

Derivation of conditional probabilities

$$p(\mathbf{h}|\mathbf{x}) = \frac{p(\mathbf{x},\mathbf{h})}{\sum_{\mathbf{h}'} p(\mathbf{x},\mathbf{h}')} = \frac{\exp(\mathbf{h}^T W\mathbf{x} + \mathbf{c}^T\mathbf{x} + \mathbf{b}^T\mathbf{h})/Z}{\sum_{\mathbf{h}'\in\{0,1\}^M} \exp(\mathbf{h}'^T W\mathbf{x} + \mathbf{c}^T\mathbf{x} + \mathbf{b}^T\mathbf{h}')/Z}$$

$$= \frac{\exp\left(\sum_i h_i W_i\mathbf{x} + b_i h_i\right)}{\sum_{h_1'\in\{0,1\}} \cdots \sum_{h_M'\in\{0,1\}} \exp(\sum_i h_i' W_i\mathbf{x} + b_i h_i')} \qquad \left(W = \begin{pmatrix} W_1 \\ \cdots \\ W_M \end{pmatrix}\right)$$

$$= \frac{\prod_i \exp\left(h_i W_i\mathbf{x} + b_i h_i\right)}{\sum_{h_1'\in\{0,1\}} \cdots \sum_{h_M'\in\{0,1\}} \prod_i \exp(h_i' W_i\mathbf{x} + b_i h_i')}$$

$$= \frac{\prod_i \exp\left(h_i W_i\mathbf{x} + b_i h_i\right)}{\left(\sum_{h_1'\in\{0,1\}} \exp(h_1' W_1\mathbf{x} + b_1 h_1')\right) \cdots \left(\sum_{h_M'\in\{0,1\}} \exp(h_M' W_M\mathbf{x} + b_M h_M')\right)}$$

$$= \prod_i \frac{\exp\left(h_i W_i\mathbf{x} + \mathbf{c}^T\mathbf{x} + b_i h_i\right)/Z}{\left(\sum_{h_i'\in\{0,1\}} \exp(h_i' W_i\mathbf{x} + \mathbf{c}^T\mathbf{x} + b_i h_i')\right)/Z} = \prod_i p(h_i|\mathbf{x})$$

N.B. Can also be obtained immediately since $h_1, h_2, \cdots, h_M$ are conditionally independent given $\mathbf{x}$

Derivation of conditional probabilities

$$p(\mathbf{h}|\mathbf{x}) = \frac{p(\mathbf{x},\mathbf{h})}{\sum_{\mathbf{h}'} p(\mathbf{x},\mathbf{h}')} = \frac{\exp(\mathbf{h}^T W\mathbf{x} + \mathbf{c}^T\mathbf{x} + \mathbf{b}^T\mathbf{h})/Z}{\sum_{\mathbf{h}'\in\{0,1\}^M} \exp(\mathbf{h}'^T W\mathbf{x} + \mathbf{c}^T\mathbf{x} + \mathbf{b}^T\mathbf{h}')/Z}$$

$$= \frac{\exp\left(\sum_i h_i W_i \mathbf{x} + b_i h_i\right)}{\sum_{h_1'\in\{0,1\}} \cdots \sum_{h_M'\in\{0,1\}} \exp(\sum_i h_i' W_i\mathbf{x} + b_i h_i')} \qquad \left(W = \begin{pmatrix} W_1 \\ \cdots \\ W_M \end{pmatrix}\right)$$

$$= \frac{\prod_i \exp\left(h_i W_i \mathbf{x} + b_i h_i\right)}{\sum_{h_1'\in\{0,1\}} \cdots \sum_{h_M'\in\{0,1\}} \prod_i \exp(h_i' W_i\mathbf{x} + b_i h_i')}$$

$$= \frac{\prod_i \exp\left(h_i W_i \mathbf{x} + b_i h_i\right)}{\left(\sum_{h_1'\in\{0,1\}} \exp(h_1' W_1\mathbf{x} + b_1 h_1')\right) \cdots \left(\sum_{h_M'\in\{0,1\}} \exp(h_M' W_M\mathbf{x} + b_M h_M')\right)}$$

$$= \prod_i \frac{\exp\left(h_i W_i \mathbf{x} + \mathbf{c}^T\mathbf{x} + b_i h_i\right)/Z}{\left(\sum_{h_i'\in\{0,1\}} \exp(h_i' W_i\mathbf{x} + \mathbf{c}^T\mathbf{x} + b_i h_i')\right)/Z} = \prod_i p(h_i|\mathbf{x})$$

N.B. Can also be obtained immediately since $h_1, h_2, \cdots, h_M$ are conditionally independent given $\mathbf{x}$

Derivation of conditional probabilities

$$p(\mathbf{h}|\mathbf{x}) = \frac{p(\mathbf{x},\mathbf{h})}{\sum_{\mathbf{h}'} p(\mathbf{x},\mathbf{h}')} = \frac{\exp(\mathbf{h}^T W\mathbf{x} + \mathbf{c}^T\mathbf{x} + \mathbf{b}^T\mathbf{h})/Z}{\sum_{\mathbf{h}'\in\{0,1\}^M} \exp(\mathbf{h}'^T W\mathbf{x} + \mathbf{c}^T\mathbf{x} + \mathbf{b}^T\mathbf{h}')/Z}$$

$$= \frac{\exp\left(\sum_i h_i W_i\mathbf{x} + b_i h_i\right)}{\sum_{h_1'\in\{0,1\}} \cdots \sum_{h_M'\in\{0,1\}} \exp(\sum_i h_i' W_i\mathbf{x} + b_i h_i')} \qquad \left(W = \begin{pmatrix} W_1 \\ \cdots \\ W_M \end{pmatrix} \right)$$

$$= \frac{\prod_i \exp\left(h_i W_i\mathbf{x} + b_i h_i\right)}{\sum_{h_1'\in\{0,1\}} \cdots \sum_{h_M'\in\{0,1\}} \prod_i \exp(h_i' W_i\mathbf{x} + b_i h_i')}$$

$$= \frac{\prod_i \exp\left(h_i W_i\mathbf{x} + b_i h_i\right)}{\left(\sum_{h_1'\in\{0,1\}} \exp(h_1' W_1\mathbf{x} + b_1 h_1')\right) \cdots \left(\sum_{h_M'\in\{0,1\}} \exp(h_M' W_M\mathbf{x} + b_M h_M')\right)}$$

$$= \prod_i \frac{\exp\left(h_i W_i\mathbf{x} + \mathbf{c}^T\mathbf{x} + b_i h_i\right)/Z}{\left(\sum_{h_i'\in\{0,1\}} \exp(h_i' W_i\mathbf{x} + \mathbf{c}^T\mathbf{x} + b_i h_i')\right)/Z} = \prod_i p(h_i|\mathbf{x})$$

N.B. Can also be obtained immediately since $h_1, h_2, \cdots, h_M$ are conditionally independent given $\mathbf{x}$

Derivation of conditional probabilities

$$p(\mathbf{h}|\mathbf{x}) = \frac{p(\mathbf{x},\mathbf{h})}{\sum_{\mathbf{h}'} p(\mathbf{x},\mathbf{h}')} = \frac{\exp(\mathbf{h}^T W\mathbf{x} + \mathbf{c}^T\mathbf{x} + \mathbf{b}^T\mathbf{h})/Z}{\sum_{\mathbf{h}'\in\{0,1\}^M} \exp(\mathbf{h}'^T W\mathbf{x} + \mathbf{c}^T\mathbf{x} + \mathbf{b}^T\mathbf{h}')/Z}$$

$$= \frac{\exp\left(\sum_i h_i W_i \mathbf{x} + b_i h_i\right)}{\sum_{h'_1\in\{0,1\}} \cdots \sum_{h'_M\in\{0,1\}} \exp(\sum_i h'_i W_i\mathbf{x} + b_i h'_i)} \qquad \left(W = \begin{pmatrix} W_1 \\ \cdots \\ W_M \end{pmatrix}\right)$$

$$= \frac{\prod_i \exp\left(h_i W_i \mathbf{x} + b_i h_i\right)}{\sum_{h'_1\in\{0,1\}} \cdots \sum_{h'_M\in\{0,1\}} \prod_i \exp(h'_i W_i\mathbf{x} + b_i h'_i)}$$

$$= \frac{\prod_i \exp\left(h_i W_i \mathbf{x} + b_i h_i\right)}{\left(\sum_{h'_1\in\{0,1\}} \exp(h'_1 W_1\mathbf{x} + b_1 h'_1)\right) \cdots \left(\sum_{h'_M\in\{0,1\}} \exp(h'_M W_M\mathbf{x} + b_M h'_M)\right)}$$

$$= \prod_i \frac{\exp\left(h_i W_i \mathbf{x} + \mathbf{c}^T\mathbf{x} + b_i h_i\right)/Z}{\left(\sum_{h'_i\in\{0,1\}} \exp(h'_i W_i\mathbf{x} + \mathbf{c}^T\mathbf{x} + b_i h'_i)\right)/Z} = \prod_i p(h_i|\mathbf{x})$$

N.B. Can also be obtained immediately since $h_1, h_2, \cdots, h_M$ are conditionally independent given $\mathbf{x}$

Derivation of conditional probabilities

$$p(\mathbf{h}|\mathbf{x}) = \frac{p(\mathbf{x},\mathbf{h})}{\sum_{\mathbf{h}'} p(\mathbf{x},\mathbf{h}')} = \frac{\exp(\mathbf{h}^T W\mathbf{x} + \mathbf{c}^T\mathbf{x} + \mathbf{b}^T\mathbf{h})/Z}{\sum_{\mathbf{h}'\in\{0,1\}^M} \exp(\mathbf{h}'^T W\mathbf{x} + \mathbf{c}^T\mathbf{x} + \mathbf{b}^T\mathbf{h}')/Z}$$

$$= \frac{\exp\left(\sum_i h_i W_i\mathbf{x} + b_i h_i\right)}{\sum_{h_1'\in\{0,1\}} \cdots \sum_{h_M'\in\{0,1\}} \exp(\sum_i h_i' W_i\mathbf{x} + b_i h_i')} \qquad \left(W = \begin{pmatrix} W_1 \\ \cdots \\ W_M \end{pmatrix} \right)$$

$$= \frac{\prod_i \exp\left(h_i W_i\mathbf{x} + b_i h_i\right)}{\sum_{h_1'\in\{0,1\}} \cdots \sum_{h_M'\in\{0,1\}} \prod_i \exp(h_i' W_i\mathbf{x} + b_i h_i')}$$

$$= \frac{\prod_i \exp\left(h_i W_i\mathbf{x} + b_i h_i\right)}{\left(\sum_{h_1'\in\{0,1\}} \exp(h_1' W_1\mathbf{x} + b_1 h_1')\right) \cdots \left(\sum_{h_M'\in\{0,1\}} \exp(h_M' W_M\mathbf{x} + b_M h_M')\right)}$$

$$= \prod_i \frac{\exp\left(h_i W_i\mathbf{x} + \mathbf{c}^T\mathbf{x} + b_i h_i\right)/Z}{\left(\sum_{h_i'\in\{0,1\}} \exp(h_i' W_i\mathbf{x} + \mathbf{c}^T\mathbf{x} + b_i h_i')\right)/Z} = \prod_i p(h_i|\mathbf{x})$$

N.B. Can also be obtained immediately since $h_1, h_2, \cdots, h_M$ are conditionally independent given $\mathbf{x}$

Derivation of conditional probabilities

$$p(\mathbf{h}|\mathbf{x}) = \frac{p(\mathbf{x},\mathbf{h})}{\sum_{\mathbf{h}'} p(\mathbf{x},\mathbf{h}')} = \frac{\exp(\mathbf{h}^T W\mathbf{x} + \mathbf{c}^T\mathbf{x} + \mathbf{b}^T\mathbf{h})/Z}{\sum_{\mathbf{h}'\in\{0,1\}^M} \exp(\mathbf{h}'^T W\mathbf{x} + \mathbf{c}^T\mathbf{x} + \mathbf{b}^T\mathbf{h}')/Z}$$

$$= \frac{\exp\left(\sum_i h_i W_i \mathbf{x} + b_i h_i\right)}{\sum_{h_1'\in\{0,1\}} \cdots \sum_{h_M'\in\{0,1\}} \exp(\sum_i h_i' W_i \mathbf{x} + b_i h_i')} \qquad \left(W = \begin{pmatrix} W_1 \\ \cdots \\ W_M \end{pmatrix}\right)$$

$$= \frac{\prod_i \exp\left(h_i W_i \mathbf{x} + b_i h_i\right)}{\sum_{h_1'\in\{0,1\}} \cdots \sum_{h_M'\in\{0,1\}} \prod_i \exp(h_i' W_i \mathbf{x} + b_i h_i')}$$

$$= \frac{\prod_i \exp\left(h_i W_i \mathbf{x} + b_i h_i\right)}{\left(\sum_{h_1'\in\{0,1\}} \exp(h_1' W_1 \mathbf{x} + b_1 h_1')\right) \cdots \left(\sum_{h_M'\in\{0,1\}} \exp(h_M' W_M \mathbf{x} + b_M h_M')\right)}$$

$$= \prod_i \frac{\exp\left(h_i W_i \mathbf{x} + \mathbf{c}^T\mathbf{x} + b_i h_i\right)/Z}{\left(\sum_{h_i'\in\{0,1\}} \exp(h_i' W_i \mathbf{x} + \mathbf{c}^T\mathbf{x} + b_i h_i')\right)/Z} = \prod_i p(h_i|\mathbf{x})$$

N.B. Can also be obtained immediately since $h_1, h_2, \cdots, h_M$ are conditionally independent given $\mathbf{x}$

Derivation of conditional probabilities

$$p(h_i = 1|\mathbf{x}) = \frac{\exp\left(W_i\mathbf{x} + b_i\right)}{\left(\sum_{h'_i \in \{0,1\}} \exp(h'_i W_i \mathbf{x} + b_i h'_i)\right)}$$

$$= \frac{\exp\left(W_i\mathbf{x} + b_i\right)}{(1 + \exp(\ W_i\mathbf{x} + b_i))}$$

$$= \mathrm{sigm}(b_i + W_i\mathbf{x})$$

Derivation of conditional probabilities

$$p(h_i = 1|\mathbf{x}) = \frac{\exp\left(W_i\mathbf{x} + b_i\right)}{\left(\sum_{h_i' \in \{0,1\}} \exp(h_i' W_i \mathbf{x} + b_i h_i')\right)}$$

$$= \frac{\exp\left(W_i\mathbf{x} + b_i\right)}{(1 + \exp(\ W_i\mathbf{x} + b_i))}$$

$$= \mathrm{sigm}(b_i + W_i\mathbf{x})$$

Derivation of conditional probabilities

$$p(h_i = 1|\mathbf{x}) = \frac{\exp\left(W_i\mathbf{x} + b_i\right)}{\left(\sum_{h_i' \in \{0,1\}} \exp(h_i' W_i \mathbf{x} + b_i h_i')\right)}$$

$$= \frac{\exp\left(W_i\mathbf{x} + b_i\right)}{(1 + \exp(\ W_i\mathbf{x} + b_i))}$$

$$= \text{sigm}(b_i + W_i\mathbf{x})$$

Data generation

Equipped with the conditional probabilities $p(\mathbf{x}|\mathbf{h})$ and $p(\mathbf{h}|\mathbf{x})$, we can generate simulated data given some hidden variables $\mathbf{h}'$ using Gibbs sampling

- Sample $\mathbf{x}'$ from $p(\mathbf{x}|\mathbf{h}')$
- Sample $\mathbf{h}''$ from $p(\mathbf{h}|\mathbf{x}')$
- Sample $\mathbf{x}''$ from $p(\mathbf{x}|\mathbf{h}'')$
- $\cdots$

Marginal probability $p(\mathbf{x})$

$$
\begin{aligned}
p(\mathbf{x}) &= \sum_{\mathbf{h}\in\{0,1\}^M} \exp(\mathbf{h}^T W\mathbf{x} + \mathbf{c}^T\mathbf{x} + \mathbf{b}^T\mathbf{h})/Z \\
&= \frac{\exp(\mathbf{c}^T\mathbf{x})}{Z} \sum_{h_1\in\{0,1\}} \cdots \sum_{h_M\in\{0,1\}} \exp\left(\sum_i h_i W_i\mathbf{x} + b_i h_i\right) \\
&= \frac{\exp(\mathbf{c}^T\mathbf{x})}{Z} \left(\sum_{h_1\in\{0,1\}} e^{(h_1 W_1\mathbf{x} + b_1 h_1)}\right) \cdots \left(\sum_{h_M\in\{0,1\}} e^{(h_M W_M\mathbf{x} + b_M h_M)}\right) \\
&= \frac{\exp(\mathbf{c}^T\mathbf{x})}{Z} \left(1 + e^{(W_1\mathbf{x} + b_1)}\right) \cdots \left(1 + e^{(W_M\mathbf{x} + b_M)}\right) \\
&= \frac{\exp(\mathbf{c}^T\mathbf{x})}{Z} \exp\left(\log(1 + e^{(W_1\mathbf{x} + b_1)}) + \cdots + \log(1 + e^{(W_M\mathbf{x} + b_M)})\right) \\
&= \exp\left(\mathbf{c}^T\mathbf{x} + \sum_i \log(1 + e^{(W_i\mathbf{x} + b_i)})\right)/Z
\end{aligned}
$$

Marginal probability $p(\mathbf{x})$

$$p(\mathbf{x}) = \sum_{\mathbf{h}\in\{0,1\}^M} \exp(\mathbf{h}^T W\mathbf{x} + \mathbf{c}^T\mathbf{x} + \mathbf{b}^T\mathbf{h})/Z$$

$$= \frac{exp(\mathbf{c}^T\mathbf{x})}{Z} \sum_{h_1\in\{0,1\}} \cdots \sum_{h_M\in\{0,1\}} \exp\left(\sum_i h_i W_i\mathbf{x} + b_i h_i\right)$$

$$= \frac{\exp(\mathbf{c}^T\mathbf{x})}{Z}\left(\sum_{h_1\in\{0,1\}} e^{(h_1 W_1\mathbf{x}+b_1 h_1)}\right)\cdots\left(\sum_{h_M\in\{0,1\}} e^{(h_M W_M\mathbf{x}+b_M h_M)}\right)$$

$$= \frac{\exp(\mathbf{c}^T\mathbf{x})}{Z}\left(1+e^{(W_1\mathbf{x}+b_1)}\right)\cdots\left(1+e^{(W_M\mathbf{x}+b_M)}\right)$$

$$= \frac{\exp(\mathbf{c}^T\mathbf{x})}{Z}\exp\left(\log(1+e^{(W_1\mathbf{x}+b_1)}) + \cdots + \log(1+e^{(W_M\mathbf{x}+b_M)})\right)$$

$$= \exp\left(\mathbf{c}^T\mathbf{x} + \sum_i \log(1+e^{(W_i\mathbf{x}+b_i)})\right)/Z$$

Marginal probability $p(\mathbf{x})$

$$p(\mathbf{x}) = \sum_{\mathbf{h}\in\{0,1\}^M} \exp(\mathbf{h}^T W\mathbf{x} + \mathbf{c}^T\mathbf{x} + \mathbf{b}^T\mathbf{h})/Z$$

$$= \frac{exp(\mathbf{c}^T\mathbf{x})}{Z} \sum_{h_1\in\{0,1\}} \cdots \sum_{h_M\in\{0,1\}} \exp\left(\sum_i h_i W_i\mathbf{x} + b_i h_i\right)$$

$$= \frac{\exp(\mathbf{c}^T\mathbf{x})}{Z} \left(\sum_{h_1\in\{0,1\}} e^{(h_1 W_1\mathbf{x}+b_1 h_1)}\right) \cdots \left(\sum_{h_M\in\{0,1\}} e^{(h_M W_M\mathbf{x}+b_M h_M)}\right)$$

$$= \frac{\exp(\mathbf{c}^T\mathbf{x})}{Z} \left(1 + e^{(W_1\mathbf{x}+b_1)}\right) \cdots \left(1 + e^{(W_M\mathbf{x}+b_M)}\right)$$

$$= \frac{\exp(\mathbf{c}^T\mathbf{x})}{Z} \exp\left(\log(1 + e^{(W_1\mathbf{x}+b_1)}) + \cdots + \log(1 + e^{(W_M\mathbf{x}+b_M)})\right)$$

$$= \exp\left(\mathbf{c}^T\mathbf{x} + \sum_i \log(1 + e^{(W_i\mathbf{x}+b_i)})\right)/Z$$

Marginal probability $p(\mathbf{x})$

$$p(\mathbf{x}) = \sum_{\mathbf{h}\in\{0,1\}^M} \exp(\mathbf{h}^T W\mathbf{x} + \mathbf{c}^T\mathbf{x} + \mathbf{b}^T\mathbf{h})/Z$$

$$= \frac{exp(\mathbf{c}^T\mathbf{x})}{Z} \sum_{h_1\in\{0,1\}} \cdots \sum_{h_M\in\{0,1\}} \exp\left(\sum_i h_i W_i\mathbf{x} + b_i h_i\right)$$

$$= \frac{\exp(\mathbf{c}^T\mathbf{x})}{Z} \left(\sum_{h_1\in\{0,1\}} e^{(h_1 W_1\mathbf{x}+b_1 h_1)}\right) \cdots \left(\sum_{h_M\in\{0,1\}} e^{(h_M W_M\mathbf{x}+b_M h_M)}\right)$$

$$= \frac{\exp(\mathbf{c}^T\mathbf{x})}{Z} \left(1 + e^{(W_1\mathbf{x}+b_1)}\right) \cdots \left(1 + e^{(W_M\mathbf{x}+b_M)}\right)$$

$$= \frac{\exp(\mathbf{c}^T\mathbf{x})}{Z} \exp\left(\log(1 + e^{(W_1\mathbf{x}+b_1)}) + \cdots + \log(1 + e^{(W_M\mathbf{x}+b_M)})\right)$$

$$= \exp\left(\mathbf{c}^T\mathbf{x} + \sum_i \log(1 + e^{(W_i\mathbf{x}+b_i)})\right)/Z$$

Marginal probability $p(\mathbf{x})$

$$
\begin{aligned}
p(\mathbf{x}) &= \sum_{\mathbf{h}\in\{0,1\}^M} \exp(\mathbf{h}^T W\mathbf{x} + \mathbf{c}^T\mathbf{x} + \mathbf{b}^T\mathbf{h})/Z \\
&= \frac{exp(\mathbf{c}^T\mathbf{x})}{Z} \sum_{h_1\in\{0,1\}} \cdots \sum_{h_M\in\{0,1\}} \exp\left(\sum_i h_i W_i \mathbf{x} + b_i h_i\right) \\
&= \frac{\exp(\mathbf{c}^T\mathbf{x})}{Z} \left(\sum_{h_1\in\{0,1\}} e^{(h_1 W_1 \mathbf{x} + b_1 h_1)}\right) \cdots \left(\sum_{h_M\in\{0,1\}} e^{(h_M W_M \mathbf{x} + b_M h_M)}\right) \\
&= \frac{\exp(\mathbf{c}^T\mathbf{x})}{Z} \left(1 + e^{(W_1\mathbf{x}+b_1)}\right) \cdots \left(1 + e^{(W_M\mathbf{x}+b_M)}\right) \\
&= \frac{\exp(\mathbf{c}^T\mathbf{x})}{Z} \exp\left(\log(1 + e^{(W_1\mathbf{x}+b_1)}) + \cdots + \log(1 + e^{(W_M\mathbf{x}+b_M)})\right) \\
&= \exp\left(\mathbf{c}^T\mathbf{x} + \sum_i \log(1 + e^{(W_i\mathbf{x}+b_i)})\right) / Z
\end{aligned}
$$

Marginal probability $p(\mathbf{x})$

$$
\begin{aligned}
p(\mathbf{x}) &= \sum_{\mathbf{h}\in\{0,1\}^M} \exp(\mathbf{h}^T W\mathbf{x} + \mathbf{c}^T\mathbf{x} + \mathbf{b}^T\mathbf{h})/Z \\
&= \frac{exp(\mathbf{c}^T\mathbf{x})}{Z} \sum_{h_1\in\{0,1\}} \cdots \sum_{h_M\in\{0,1\}} \exp\left(\sum_i h_i W_i\mathbf{x} + b_i h_i\right) \\
&= \frac{\exp(\mathbf{c}^T\mathbf{x})}{Z} \left(\sum_{h_1\in\{0,1\}} e^{(h_1 W_1\mathbf{x} + b_1 h_1)}\right) \cdots \left(\sum_{h_M\in\{0,1\}} e^{(h_M W_M\mathbf{x} + b_M h_M)}\right) \\
&= \frac{\exp(\mathbf{c}^T\mathbf{x})}{Z} \left(1 + e^{(W_1\mathbf{x}+b_1)}\right) \cdots \left(1 + e^{(W_M\mathbf{x}+b_M)}\right) \\
&= \frac{\exp(\mathbf{c}^T\mathbf{x})}{Z} \exp\left(\log(1 + e^{(W_1\mathbf{x}+b_1)}) + \cdots + \log(1 + e^{(W_M\mathbf{x}+b_M)})\right) \\
&= \exp\left(\mathbf{c}^T\mathbf{x} + \sum_i \log(1 + e^{(W_i\mathbf{x}+b_i)})\right)/Z
\end{aligned}
$$

$$p(\mathbf{x}) = \exp\left(\mathbf{c}^T\mathbf{x} + \sum_i \log(1 + e^{(W_i\mathbf{x}+b_i)})\right)/Z$$

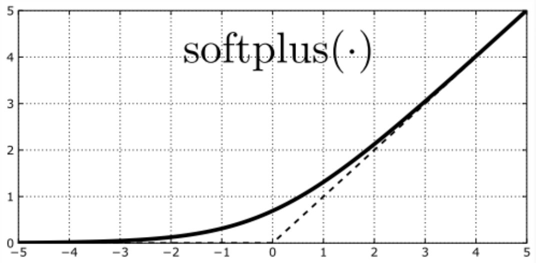
$$= \exp\left(\mathbf{c}^T\mathbf{x} + \sum_i \text{softplus}(W_i\mathbf{x} + b_i)\right)/Z \triangleq \exp(-F(\mathbf{x}))/Z,$$

where $F(\mathbf{x})$ is known to be free energy, a term borrowed from statistical physics. Note that $\frac{\partial \text{softplus(t)}}{\partial t} = \text{sigmod}(t)$

$$p(\mathbf{x}) = \exp\left(\mathbf{c}^T\mathbf{x} + \sum_i \log(1 + e^{(W_i\mathbf{x}+b_i)})\right)/Z$$

$$= \exp\left(\mathbf{c}^T\mathbf{x} + \sum_i \text{softplus}(W_i\mathbf{x} + b_i)\right)/Z \triangleq \exp(-F(\mathbf{x}))/Z,$$

where $F(\mathbf{x})$ is known to be free energy, a term borrowed from statistical physics. Note that $\frac{\partial \text{softplus(t)}}{\partial t} = \text{sigmod}(t)$



Training RBM

Use the cross entropy loss,

$$l(\theta) = \frac{1}{T}\sum_t -\log p(\mathbf{x}^{(t)}) = \frac{1}{T}\sum_t F(\mathbf{x}^{(t)}) - \log Z,$$

where $Z = \sum_{\mathbf{x}} \exp(-F(\mathbf{x}))$. And

$$\frac{\partial -\log p(\mathbf{x}^{(t)})}{\partial\theta} = \frac{\partial F(\mathbf{x}^{(t)})}{\partial\theta} - \sum_{\mathbf{x}} \frac{\exp(-F(\mathbf{x}))}{Z}\frac{\partial F(\mathbf{x})}{\partial\theta}$$

$$= \underbrace{\frac{\partial F(\mathbf{x}^{(t)})}{\partial\theta}}_{\text{positive phase}} - \underbrace{E\left[\frac{\partial F(\mathbf{x})}{\partial\theta}\right]}_{\text{negative phase}}$$

N.B. The naming of the terms is not related to the sign in the equation. It refers to the fact that adjusting the +ve phase terms to increase the probability of the training data and the -ve terms to decrease the probability of the rest of $\mathbf{x}$

Training RBM

Use the cross entropy loss,

$$l(\theta) = \frac{1}{T}\sum_t -\log p(\mathbf{x}^{(t)}) = \frac{1}{T}\sum_t F(\mathbf{x}^{(t)}) - \log Z,$$

where $Z = \sum_{\mathbf{x}} \exp(-F(\mathbf{x}))$. And

$$\frac{\partial - \log p(\mathbf{x}^{(t)})}{\partial \theta} = \frac{\partial F(\mathbf{x}^{(t)})}{\partial \theta} - \sum_{\mathbf{x}} \frac{\exp(-F(\mathbf{x}))}{Z} \frac{\partial F(\mathbf{x})}{\partial \theta}$$

$$= \underbrace{\frac{\partial F(\mathbf{x}^{(t)})}{\partial \theta}}_{\text{positive phase}} - \underbrace{E\left[\frac{\partial F(\mathbf{x})}{\partial \theta}\right]}_{\text{negative phase}}$$

N.B. The naming of the terms is not related to the sign in the equation. It refers to the fact that adjusting the +ve phase terms to increase the probability of the training data and the -ve terms to decrease the probability of the rest of $\mathbf{x}$

Training RBM

Use the cross entropy loss,

$$l(\theta) = \frac{1}{T}\sum_t -\log p(\mathbf{x}^{(t)}) = \frac{1}{T}\sum_t F(\mathbf{x}^{(t)}) - \log Z,$$

where $Z = \sum_{\mathbf{x}} \exp(-F(\mathbf{x}))$. And

$$\frac{\partial -\log p(\mathbf{x}^{(t)})}{\partial\theta} = \frac{\partial F(\mathbf{x}^{(t)})}{\partial\theta} - \sum_{\mathbf{x}} \frac{\exp(-F(\mathbf{x}))}{Z}\frac{\partial F(\mathbf{x})}{\partial\theta}$$

$$= \underbrace{\frac{\partial F(\mathbf{x}^{(t)})}{\partial\theta}}_{\text{positive phase}} - \underbrace{E\left[\frac{\partial F(\mathbf{x})}{\partial\theta}\right]}_{\text{negative phase}}$$

N.B. The naming of the terms is not related to the sign in the equation. It refers to the fact that adjusting the +ve phase terms to increase the probability of the training data and the -ve terms to decrease the probability of the rest of $\mathbf{x}$

Training RBM

Use the cross entropy loss,

$$l(\theta) = \frac{1}{T}\sum_t -\log p(\mathbf{x}^{(t)}) = \frac{1}{T}\sum_t F(\mathbf{x}^{(t)}) - \log Z,$$

where $Z = \sum_{\mathbf{x}} \exp(-F(\mathbf{x}))$. And

$$\frac{\partial -\log p(\mathbf{x}^{(t)})}{\partial\theta} = \frac{\partial F(\mathbf{x}^{(t)})}{\partial\theta} - \sum_{\mathbf{x}} \frac{\exp(-F(\mathbf{x}))}{Z}\frac{\partial F(\mathbf{x})}{\partial\theta}$$

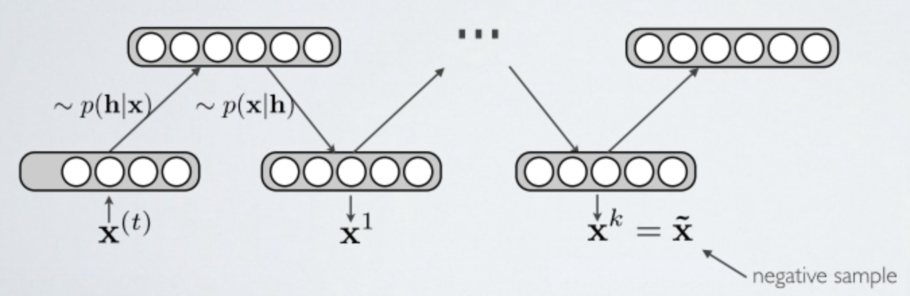
$$= \underbrace{\frac{\partial F(\mathbf{x}^{(t)})}{\partial\theta}}_{\text{positive phase}} - \underbrace{E\left[\frac{\partial F(\mathbf{x})}{\partial\theta}\right]}_{\text{negative phase}}$$

N.B. The naming of the terms is not related to the sign in the equation. It refers to the fact that adjusting the +ve phase terms to increase the probability of the training data and the -ve terms to decrease the probability of the rest of **x**

Contrastive divergence (CD-k)

The negative phase term is very hard to compute exactly as we need to sum over all $\mathbf{x}$. The natural way out is to approximate using sampling $\Rightarrow$ contrastive divergence (CD-k) training

Key idea:
1. Start sampling chain at $\mathbf{x}^{(t)}$
2. Obtain the point $\tilde{\mathbf{x}}$ with k Gibbs sampling steps
3. Replace the expectation by a point estimate at $\tilde{\mathbf{x}}$



N.B. CD-1 works surprisingly well in practice

Parameters update

So we have $\frac{\partial l(\theta)}{\partial \theta} = \frac{\partial F(\mathbf{x}^{(t)})}{\partial \theta} - \frac{\partial F(\tilde{\mathbf{x}})}{\partial \theta}$. Recall that

$$F(\mathbf{x}) = -\mathbf{c}^T\mathbf{x} - \sum_i \text{softplus}(W_i\mathbf{x} + b_i)$$

$$\frac{\partial F(\mathbf{x})}{\partial c_i} = -x_i$$

$$\frac{\partial F(\mathbf{x})}{\partial b_i} = -\text{sigmoid}(W_i\mathbf{x} + b_i)$$

$$\frac{\partial F(\mathbf{x})}{\partial W_{ij}} = -\text{sigmoid}(W_i\mathbf{x} + b_i)x_j$$

This gives us

$$\mathbf{c} \Leftarrow \mathbf{c} + \alpha(\mathbf{x}^{(t)} - \tilde{\mathbf{x}})$$

$$\mathbf{b} \Leftarrow \mathbf{b} + \alpha(\text{sigmoid}(W\mathbf{x}^{(t)} + \mathbf{b}) - \text{sigmoid}(W\tilde{\mathbf{x}} + \mathbf{b}))$$

$$W \Leftarrow W + \alpha(\text{sigmoid}(W\mathbf{x}^{(t)} + \mathbf{b})\mathbf{x}^{(t)T} - \text{sigmoid}(W\tilde{\mathbf{x}} + \mathbf{b})\tilde{\mathbf{x}}^T)$$

Parameters update

So we have $\frac{\partial l(\theta)}{\partial\theta}=\frac{\partial F(\mathbf{x}^{(t)})}{\partial\theta}-\frac{\partial F(\tilde{\mathbf{x}})}{\partial\theta}$. Recall that

$$F(\mathbf{x})=-\mathbf{c}^T\mathbf{x}-\sum_i \text{softplus}(W_i\mathbf{x}+b_i)$$

$$\frac{\partial F(\mathbf{x})}{\partial c_i}=-x_i$$

$$\frac{\partial F(\mathbf{x})}{\partial b_i}=-\text{sigmoid}(W_i\mathbf{x}+b_i)$$

$$\frac{\partial F(\mathbf{x})}{\partial W_{ij}}=-\text{sigmoid}(W_i\mathbf{x}+b_i)x_j$$

This gives us

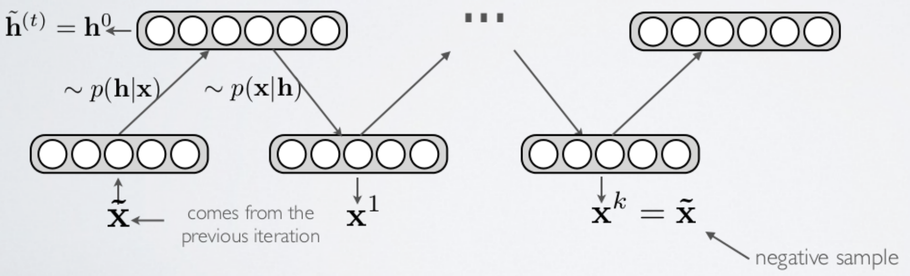
$$\mathbf{c}\Leftarrow\mathbf{c}+\alpha(\mathbf{x}^{(t)}-\tilde{\mathbf{x}})$$

$$\mathbf{b}\Leftarrow\mathbf{b}+\alpha(\text{sigmoid}(W\mathbf{x}^{(t)}+\mathbf{b})-\text{sigmoid}(W\tilde{\mathbf{x}}+\mathbf{b}))$$

$$W\Leftarrow W+\alpha(\text{sigmoid}(W\mathbf{x}^{(t)}+\mathbf{b}){\mathbf{x}^{(t)}}^T-\text{sigmoid}(W\tilde{\mathbf{x}}+\mathbf{b})\tilde{\mathbf{x}}^T)$$

Persistent CD
Tieleman, ICML 2008

- Idea: Instead of initializing the chain to $\mathbf{x}^{(t)}$, initialize the chain to the negative sample of the last iteration
- This has a similar effect of CD-k with a large k and yet can have much lower complexity



Gaussian-Bernoulli RBM

Extension to continuous variables

- RBM is a binary model and thus is not suitable for continuous data
- One simple extension to allow the visible variables **x** to be continuous while keeping the hidden variables **h** to be binary
- In particular, we can simply add a quadratic term $\frac{1}{2}\mathbf{x}^T\mathbf{x}$ to the energy function, i.e.,

$$E(x,h) = -h^T W x - c^T x - b^T h + \frac{1}{2}x^T x$$

  to get Gaussian distributed $p(x|h)$
- For efficient training, the input data are typically preprocessed with zero-mean and unit variance
- A smaller learning rate is needed compared to a regular RBM

Gaussian-Bernoulli RBM

Extension to continuous variables

- RBM is a binary model and thus is not suitable for continuous data
- One simple extension to allow the visible variables **x** to be continuous while keeping the hidden variables **h** to be binary
- In particular, we can simply add a quadratic term $\frac{1}{2}\mathbf{x}^T\mathbf{x}$ to the energy function, i.e.,

$$E(x,h) = -h^T Wx - c^T x - b^T h + \frac{1}{2}x^T x$$

  to get Gaussian distributed $p(x|h)$
- For efficient training, the input data are typically preprocessed with zero-mean and unit variance
- A smaller learning rate is needed compared to a regular RBM

Gaussian-Bernoulli RBM

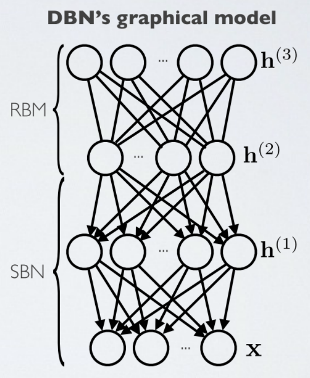
Extension to continuous variables

- RBM is a binary model and thus is not suitable for continuous data
- One simple extension to allow the visible variables **x** to be continuous while keeping the hidden variables **h** to be binary
- In particular, we can simply add a quadratic term $\frac{1}{2}\mathbf{x}^T\mathbf{x}$ to the energy function, i.e.,

$$E(x,h) = -h^T Wx - c^T x - b^T h + \frac{1}{2}x^T x$$

  to get Gaussian distributed $p(x|h)$
- For efficient training, the input data are typically preprocessed with zero-mean and unit variance
- A smaller learning rate is needed compared to a regular RBM

Deep belief networks (DBN)

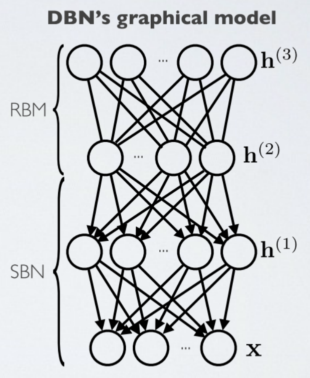


- DBN is a generative model that mixes undirected and directed connections
- Top 2 layers' distribution $p(\mathbf{h}^{(2)}, \mathbf{h}^{(3)})$ is an RBN
- Other layers form a Bayesian network:
  - The conditional distributions of layers given the one above it are

$$p(h_i^{(1)} = 1|\mathbf{h}^{(2)}) = \text{sigm}(b_i^{(1)} + W^{(2)}{}_i\mathbf{h}^{(2)})$$
$$p(h_i^{(1)} = 1|\mathbf{h}^{(1)}) = \text{sigm}(b_i^{(0)} + W^{(1)}{}_i\mathbf{h}^{(1)})$$

  - This is referred to as a sigmoid belief network (SBN)
- Note that DBN is not a feed-forward network

Deep belief networks (DBN)

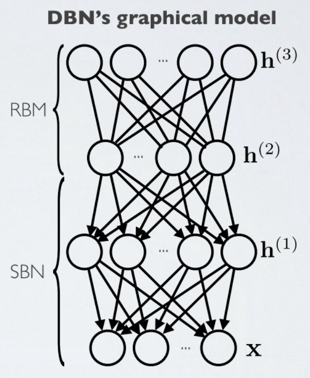


- DBN is a generative model that mixes undirected and directed connections
- Top 2 layers' distribution $p(\mathbf{h}^{(2)}, \mathbf{h}^{(3)})$ is an RBN
- Other layers form a Bayesian network:
    - The conditional distributions of layers given the one above it are

$$p(h_i^{(1)} = 1|\mathbf{h}^{(2)}) = \text{sigm}(b_i^{(1)} + W^{(2)}{}_i\mathbf{h}^{(2)})$$
$$p(h_i^{(1)} = 1|\mathbf{h}^{(1)}) = \text{sigm}(b_i^{(0)} + W^{(1)}{}_i\mathbf{h}^{(1)})$$

    - This is referred to as a sigmoid belief network (SBN)
- Note that DBN is not a feed-forward network

Deep belief networks (DBN)



- DBN is a generative model that mixes undirected and directed connections
- Top 2 layers' distribution $p(\mathbf{h}^{(2)}, \mathbf{h}^{(3)})$ is an RBN
- Other layers form a Bayesian network:
  - The conditional distributions of layers given the one above it are

$$p(h_i^{(1)} = 1|\mathbf{h}^{(2)}) = \mathrm{sigm}(b_i^{(1)} + W^{(2)}{}_i\mathbf{h}^{(2)})$$
$$p(h_i^{(1)} = 1|\mathbf{h}^{(1)}) = \mathrm{sigm}(b_i^{(0)} + W^{(1)}{}_i\mathbf{h}^{(1)})$$

  - This is referred to as a sigmoid belief network (SBN)
- Note that DBN is not a feed-forward network

History of DBNs

According to HInton's coursera's course

- Professor Hinton was working on algorithms to train Sigmoid belief network but gave up after many different ideas
- He moved on to work with RBMs and invented the CD-k algorithm for training RBMs
- Since CD-k is very effective, it is very tempting to think if one can train a Sigmoid belief network one layer at a time by treating each layer as a RBM
  - The procedure is working great. But it actually trains a different model, the DBN instead of SBN (with some complicated math behind), pointed out by Yee-Whye Teh
- DBN is actually the first successful deep neural network model and revived the entire neural network field
- Try not to get confused of DBN with deep Boltzmann machines (DBMs), where each layer is composed of an RBM

History of DBNs

According to HInton's coursera's course

- Professor Hinton was working on algorithms to train Sigmoid belief network but gave up after many different ideas
- He moved on to work with RBMs and invented the CD-k algorithm for training RBMs
- Since CD-k is very effective, it is very tempting to think if one can train a Sigmoid belief network one layer at a time by treating each layer as a RBM
  - The procedure is working great. But it actually trains a different model, the DBN instead of SBN (with some complicated math behind), pointed out by Yee-Whye Teh
- DBN is actually the first successful deep neural network model and revived the entire neural network field
- Try not to get confused of DBN with deep Boltzmann machines (DBMs), where each layer is composed of an RBM

History of DBNs
According to HInton's coursera's course

- Professor Hinton was working on algorithms to train Sigmoid belief network but gave up after many different ideas
- He moved on to work with RBMs and invented the CD-k algorithm for training RBMs
- Since CD-k is very effective, it is very tempting to think if one can train a Sigmoid belief network one layer at a time by treating each layer as a RBM
    - The procedure is working great. But it actually trains a different model, the DBN instead of SBN (with some complicated math behind), pointed out by Yee-Whye Teh
- DBN is actually the first successful deep neural network model and revived the entire neural network field
- Try not to get confused of DBN with deep Boltzmann machines (DBMs), where each layer is composed of an RBM

History of DBNs

According to HInton's coursera's course

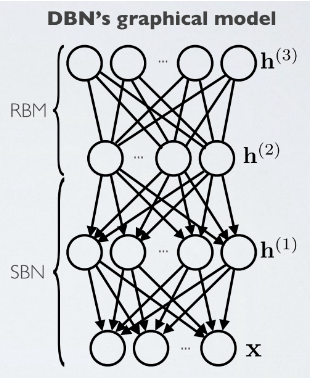
- Professor Hinton was working on algorithms to train Sigmoid belief network but gave up after many different ideas
- He moved on to work with RBMs and invented the CD-k algorithm for training RBMs
- Since CD-k is very effective, it is very tempting to think if one can train a Sigmoid belief network one layer at a time by treating each layer as a RBM
  - The procedure is working great. But it actually trains a different model, the DBN instead of SBN (with some complicated math behind), pointed out by Yee-Whye Teh
- DBN is actually the first successful deep neural network model and revived the entire neural network field
- Try not to get confused of DBN with deep Boltzmann machines (DBMs), where each layer is composed of an RBM

History of DBNs

According to HInton's coursera's course

- Professor Hinton was working on algorithms to train Sigmoid belief network but gave up after many different ideas
- He moved on to work with RBMs and invented the CD-k algorithm for training RBMs
- Since CD-k is very effective, it is very tempting to think if one can train a Sigmoid belief network one layer at a time by treating each layer as a RBM
  - The procedure is working great. But it actually trains a different model, the DBN instead of SBN (with some complicated math behind), pointed out by Yee-Whye Teh
- DBN is actually the first successful deep neural network model and revived the entire neural network field
- Try not to get confused of DBN with deep Boltzmann machines (DBMs), where each layer is composed of an RBM

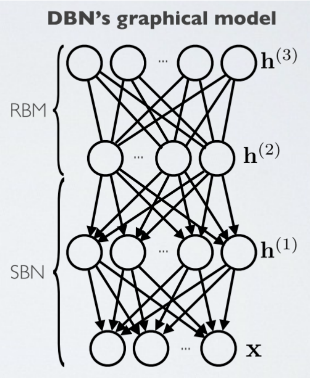
Pretraining of DBNs



As mentioned in the previous slide

- Treat the bottom two layers as an RBM and train it with the input data $\mathbf{x}$
- Treat the next two layers as an RBM and train it with the $\mathbf{h}^{(1)}$ obtained in the last step
- Keep continuing while keeping the trained weights

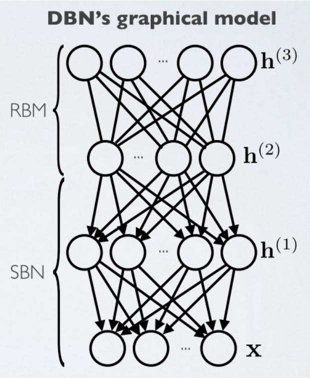
Pretraining of DBNs



As mentioned in the previous slide

- Treat the bottom two layers as an RBM and train it with the input data **x**
- Treat the next two layers as an RBM and train it with the $\mathbf{h}^{(1)}$ obtained in the last step
- Keep continuing while keeping the trained weights

Pretraining of DBNs



As mentioned in the previous slide

- Treat the bottom two layers as an RBM and train it with the input data $\mathbf{x}$
- Treat the next two layers as an RBM and train it with the $\mathbf{h}^{(1)}$ obtained in the last step
- Keep continuing while keeping the trained weights

Fine-tuning of DBN

Up-down algorithm (aka contrastive wake-sleep algorithm)

After learning many layers of features, we can fine-tune the features to improve generation

1. Do a stochastic bottom-up pass
    - Construct hidden variables with reconstruction weight R (initialized as the transpose of W)
    - Use the approximated hidden variables to fine tune W
2. Do a few iterations of sampling in the top level RBM
    - Adjust top-level RBM weights using CD-k
3. Do a stochastic top-down pass
    - Generate simulation data and use that to fine-tune the reconstruction weights R

Fine-tuning of DBN

Up-down algorithm (aka contrastive wake-sleep algorithm)

After learning many layers of features, we can fine-tune the features to improve generation

1. Do a stochastic bottom-up pass
   - Construct hidden variables with reconstruction weight R (initialized as the transpose of W)
   - Use the approximated hidden variables to fine tune W
2. Do a few iterations of sampling in the top level RBM
   - Adjust top-level RBM weights using CD-k
3. Do a stochastic top-down pass
   - Generate simulation data and use that to fine-tune the reconstruction weights R

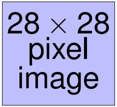
Fine-tuning of DBN

Up-down algorithm (aka contrastive wake-sleep algorithm)

After learning many layers of features, we can fine-tune the features to improve generation

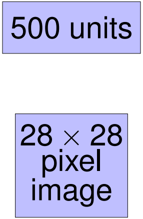
1. Do a stochastic bottom-up pass
    - Construct hidden variables with reconstruction weight R (initialized as the transpose of W)
    - Use the approximated hidden variables to fine tune W
2. Do a few iterations of sampling in the top level RBM
    - Adjust top-level RBM weights using CD-k
3. Do a stochastic top-down pass
    - Generate simulation data and use that to fine-tune the reconstruction weights R

MNIST example



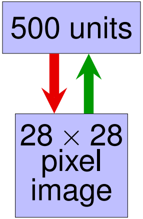
- Test on MNIST dataset
- Train 500 hidden units with the image block as input
- Train another 500 hidden units with the trained 500 hidden units as input
- Prepare another 2000 hidden units
- Train the 2000 hidden units with the previously trained 500 hidden units and target labels as input
- Error rate is about 1%

MNIST example



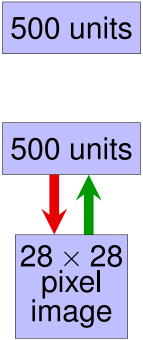
- Test on MNIST dataset
- Train 500 hidden units with the image block as input
- Train another 500 hidden units with the trained 500 hidden units as input
- Prepare another 2000 hidden units
- Train the 2000 hidden units with the previously trained 500 hidden units and target labels as input
- Error rate is about 1%

MNIST example



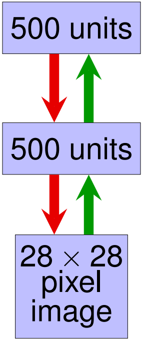
- Test on MNIST dataset
- Train 500 hidden units with the image block as input
- Train another 500 hidden units with the trained 500 hidden units as input
- Prepare another 2000 hidden units
- Train the 2000 hidden units with the previously trained 500 hidden units and target labels as input
- Error rate is about 1%

MNIST example



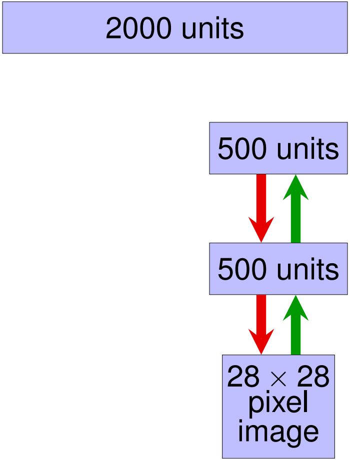
- Test on MNIST dataset
- Train 500 hidden units with the image block as input
- Train another 500 hidden units with the trained 500 hidden units as input
- Prepare another 2000 hidden units
- Train the 2000 hidden units with the previously trained 500 hidden units and target labels as input
- Error rate is about 1%

MNIST example



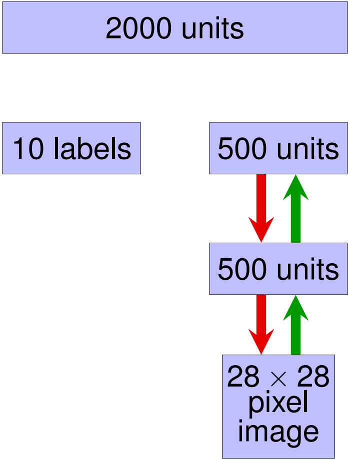
- Test on MNIST dataset
- Train 500 hidden units with the image block as input
- Train another 500 hidden units with the trained 500 hidden units as input
- Prepare another 2000 hidden units
- Train the 2000 hidden units with the previously trained 500 hidden units and target labels as input
- Error rate is about 1%

MNIST example



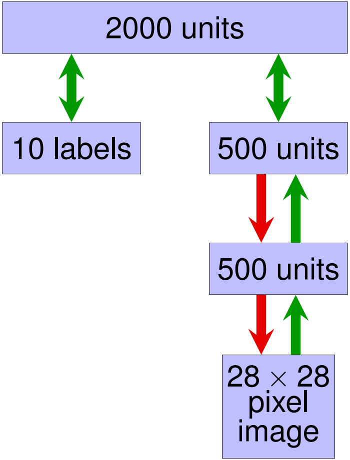
- Test on MNIST dataset
- Train 500 hidden units with the image block as input
- Train another 500 hidden units with the trained 500 hidden units as input
- Prepare another 2000 hidden units
- Train the 2000 hidden units with the previously trained 500 hidden units and target labels as input
- Error rate is about 1%

MNIST example



- Test on MNIST dataset
- Train 500 hidden units with the image block as input
- Train another 500 hidden units with the trained 500 hidden units as input
- Prepare another 2000 hidden units
- Train the 2000 hidden units with the previously trained 500 hidden units and target labels as input
- Error rate is about 1%

MNIST example



- Test on MNIST dataset
- Train 500 hidden units with the image block as input
- Train another 500 hidden units with the trained 500 hidden units as input
- Prepare another 2000 hidden units
- Train the 2000 hidden units with the previously trained 500 hidden units and target labels as input
- Error rate is about 1%

Demo

http://www.cs.toronto.edu/~hinton/adi/index.htm

Conclusions

- Restricted Boltzmann machines (RBMs) and deep belief networks (DBNs) are both generative models
- RBMs can be trained efficiently with contrastive divergence (CD-k) algorithm
- DBNs can be trained by first pre-trained each pair of layers as an RBM and then fine-tune with up-down algorithm
- DBNs are the earliest deep neural network model and essential the starting point of “deep learning” research