Evidence Inequality

Hidden variables $h$, visible variables $v$. Evidence

$$P(v) = \sum_h P(v, h)$$

$Q(h|v)$ any approximate or exact posterior

$$\mathcal{L} = E \left[ \ln \frac{P(v, h)}{Q(h|v)} \right]$$

$$E \left[ \ln P(v, h) \right] + H(Q(h|v))$$

$$E \left[ \ln P(v|h) \right] - D(Q(h|v)\|P(h))$$

$$\langle P(v, h) \rangle_{\text{data}} + H$$

$$\ln P(v) - D(Q(h|v)\|P(h|v))$$
In the third expression, we use the notation $P(h|v)$ to indicate the *exact* posterior of $h$ given $v$ so that

$$
\mathcal{L} = E \left[ \ln \frac{P(v, h)}{Q(h|v)} \right] = E \left[ \ln \frac{P(h|v)P(v)}{Q(h|v)} \right] = \ln P(v) + E \left[ \ln \frac{P(h|v)}{Q(h|v)} \right] = \ln P(v) - D(Q(h|v)\|P(h|v))
$$

$D(Q(h|v)\|P(h|v)) \geq 0$ always so $\mathcal{L} \leq \ln P(v)$. 

$2 / 58$
Variational Bayes

\[ Q(h|v) \approx \prod_i Q(h_i|v) \]

\[ \ln Q(h_i) = E_{Q(h_i \setminus i)} [\ln P(h, v)] + \text{constant} \]

Guaranteed to increase \( \mathcal{L}. \)
Training set consists visible vectors \( \{ \mathbf{v}^n \} \)

\[
\mathcal{L} = \sum_n E_{Q(h|\mathbf{v}^n)} \left[ \ln \frac{P(h, \mathbf{v}^n)}{Q(h|\mathbf{v}^n)} \right].
\]

\( \mathcal{L} \) can be increased in two ways:

1. Fix the posteriors \( Q(h|\mathbf{v}^n) \) and adjust the parameters of the model \( P(\mathbf{v}, h) \) so as to increase the EM auxiliary function

\[
\sum_n \sum_h Q(h|\mathbf{v}^n) \ln P(h, \mathbf{v}^n)
\]

2. Fix the model \( P(\mathbf{v}, h) \) and adjust the posteriors \( Q(h|\mathbf{v}^n) \) so as to decrease the divergence

\[
\sum_n D(Q(h|\mathbf{v}^n) \parallel P(h|\mathbf{v}^n)).
\]
In the case where $Q(h|v) = P(h|v)$ the lower bound is tight (\(\ln P(v) = \mathcal{L}\)) so alternating between 1) and 2) is guaranteed to increase the likelihood of the training set. This is the classical EM algorithm.

In the general case where an approximate variational posterior is used, the value of the lower bound is guaranteed to increase but this does not guarantee that the likelihood increases. This is the variational Bayes EM algorithm.
Gibbs sampling

It may be the case that it is hard to sample from the prior $P(x)$ but is easy to sample efficiently from the posterior distribution $Q(x_i|x_{\setminus i})$. For example suppose $x$ has just 2 components. We generate a Markov chain

$$x^1 \rightarrow x^2 \rightarrow \ldots \rightarrow x^n$$

by taking a random value $x_2^0$ and sampling

$$x_1^1 \sim Q(x_1^1|x_2^0), x_2^1 \sim Q(x_2^1|x_1^1), x_1^2 \sim Q(x_1^2|x_2^1), x_2^2 \sim Q(x_2^2|x_1^2), \ldots$$

(Two samples are needed for each full step $x^1 \rightarrow x^2, x^2 \rightarrow x^3, \ldots$) For $n$ sufficiently large, $x^n$ will be distributed according to $P(x)$. 
Boltzmann Machines (BMs)

\[ P(\mathbf{x}) = \frac{1}{Z} e^{-E(\mathbf{x})} \]

where the “energy function” \( E(\mathbf{x}) \) has the form

\[ E(\mathbf{x}) = - \sum_{i<j} x_i w_{ij} x_j, \]

the sum extending over all pairs \((i, j)\) such that \(i < j\). Or,

\[ E(\mathbf{x}) = -\frac{1}{2} \sum_{i \neq j} x_i w_{ij} x_j, \]

the sum extending over all pairs \((i, j)\). Diagonal of \( \mathbf{W} \) is \( \mathbf{0} \) The normalizing constant \( Z \) is referred to as the partition function.
Stochastic neural net

\[ Q(x_i = 1 | x_{\setminus i}) = \frac{P(x_i = 1, x_{\setminus i})}{P(x_i = 0, x_{\setminus i}) + P(x_i = 1, x_{\setminus i})} \]

\[ = \frac{\exp \left( \sum_{j \neq i} w_{ij}x_j \right)}{1 + \exp \left( \sum_{j \neq i} w_{ij}x_j \right)} \]

\[ = \sigma \left( \sum_{j \neq i} w_{ij}x_j \right) \]

where \( \sigma \) is the sigmoid non-linearity defined by

\[ \sigma(u) = \frac{1}{1 + e^{-u}}. \]
Variational Bayes in BMs – mean field approximation

Deterministic neural net

\[ P(x) \approx \prod_i Q(x_i) \]

\[ = \prod_i \mu_i^{x_i} \]

where \( \mu_i = Q(x_i = 1) \).

\[ \ln Q(x_i) \equiv E_{Q(x_{\setminus i})} \left[ \ln P(x_i, x_{\setminus i}) \right] \]

\[ = E_{Q(x_{\setminus i})} \left[ \sum_{j \neq i} x_i w_{ij} x_j \right] \]

\[ = \begin{cases} 
\sum_{j \neq i} w_{ij} \mu_j & \text{if } x_i = 1 \\
0 & \text{if } x_i = 0 
\end{cases} \]
\[
Q(x_i = 1) = \frac{\exp \left( \sum_{j \neq i} w_{ij} \mu_j \right)}{1 + \exp \left( \sum_{j \neq i} w_{ij} \mu_j \right)}
\]

\[
= \sigma \left( \sum_{j \neq i} w_{ij} \mu_j \right)
\]

\[
\mu_i = \sigma \left( \sum_{j \neq i} w_{ij} \mu_j \right).
\]

Do variational Bayes on posteriors, not priors!
If $P(v, h)$ is a Boltzmann distribution with energy function $E(v, h)$ we can represent the marginal distribution $P(v)$ as an energy based model with energy function $F(v)$, that is

$$P(v) = \frac{1}{\mathcal{Z}} e^{-F(v)},$$

by defining the free energy $F(v)$ as

$$F(v) = -\ln \sum_{h} e^{-E(v, h)}.$$

Not a Boltzmann machine. (Contrary to Gaussian case.)
Likelihood gradient for stochastic gradient ascent (no hidden units)

\[ \frac{\partial \ln P(x)}{\partial w_{ij}} \bigg|_{x=x^1} = x^1_i x^1_j - \langle x_i x_j \rangle_{\text{model}} \]

where

\[ \langle x_i x_j \rangle_{\text{model}} = \sum_x P(x) x_i x_j \]

Similarly, if there are \( N \) training tokens \( x^1, \ldots, x^N \),

\[ \frac{1}{N} \frac{\partial \ln P(x^1, \ldots, x^N)}{\partial w_{ij}} = \langle x_i x_j \rangle_{\text{data}} - \langle x_i x_j \rangle_{\text{model}} \]

where

\[ \langle x_i x_j \rangle_{\text{data}} = \frac{1}{N} \left( x^1_i x^1_j + \ldots + x^N_i x^N_j \right) . \]
Given a training token $x^1$, need to evaluate $x_i^1 x_j^1 - \langle x_i x_j \rangle_{\text{model}}$. Markov Chain Monte Carlo (e.g. Gibbs sampling) is the standard way of estimating $\langle x_i x_j \rangle_{\text{model}}$. Starting at the training token $x^1$, run the Markov chain for $n$ steps

$$x^1 \rightarrow \ldots \rightarrow x^{n+1}$$

If $n$ is sufficiently large

$$E \left[ x_i^{n+1} x_j^{n+1} \right] \approx \langle x_i x_j \rangle_{\text{model}}$$

Surprisingly $n = 1$ works well in practice.
Run $N$ Markov chains \{${\mathbf{x}}^1, \ldots, {\mathbf{x}}^N$\} in parallel and update each of them whenever the model is updated. If the learning rate is sufficiently slow, the Markov chains continue to sample from the model distribution even though the model is continually being updated.

$$\langle x_i x_j \rangle_{\text{model}} \approx \frac{1}{N} \left( x_i^1 x_j^1 + \ldots + x_i^N x_j^N \right).$$
Use variational Bayes EM. Given a data vector $\mathbf{v}$, the variational lower bound is

$$\langle \ln P(\mathbf{x}) \rangle_{\text{data}} + H$$

where $\mathbf{x} = (\mathbf{v}, \mathbf{h})$ and $\langle \cdot \rangle_{\text{data}}$ refers to the expectation calculated with the variational posterior of $Q(\mathbf{h}|\mathbf{v})$ and $H$ is the entropy of this posterior (which can be ignored).
\[
\frac{\partial}{\partial w_{ij}} \langle \ln P(x) \rangle_{\text{data}} = \langle x_i x_j \rangle_{\text{data}} - \langle x_i x_j \rangle_{\text{model}}.
\]

If, for example, \(i\) is visible and \(j\) is hidden

\[
\langle v_i h_j \rangle = v_i \mu_j
\]

where \(\mu_j = Q(h_j = 1|v)\). If \(i\) and \(j\) are both hidden

\[
\langle h_i h_j \rangle = \langle h_i \rangle \langle h_j \rangle = \mu_i \mu_j.
\]

The term \(\langle x_i x_j \rangle_{\text{model}}\) handled by persistent contrastive divergence as before.
The variational lower bound

Given a data vector $v$, if the variational posterior is given by

$$\mu_j = Q(x_j = 1|v)$$

then the variational lower bound is given by

$$\langle \ln P(x) \rangle + H$$

$$= -\langle E(x) \rangle - \ln Z + H$$

$$= \sum_{i<j} \mu_i w_{ij} \mu_j - \ln Z$$

$$- \sum_i (\mu_i \ln \mu_i + (1 - \mu_i) \ln(1 - \mu_i)).$$

Even in the simplest cases the partition function $Z$ cannot be evaluated in closed form but it turns out that this expression is still useful.

If the contribution of $Z$ is ignored, this computation gives an approximation to the free energy. (This can be evaluated exactly in some cases.)
The Markov property

If $x \setminus \{i, j\}$ is given and $w_{ij} = 0$

$$E(x) = ax_i + bx_j + c$$

where $a$, $b$ and $c$ depend on $x \setminus \{i, j\}$ but not on $x_i$ or $x_j$. Hence $x_i$ and $x_j$ are conditionally independent.

Suppose we are given a partition of the units into three subsets $A$, $B$ and $C$ with the property that there is no edge between units in $A$ and $C$. Then $x_A$ and $x_C$ are conditionally independent if $x_B$ is given where $x_A = \{x_i : i \in A\}$ etc.
Figure: Restricted Boltzmann machines (RBMs)
No hidden-to-hidden or visible-to-visible connections.

\[ E(\mathbf{v}, \mathbf{h}) = -\mathbf{v}^T \mathbf{W} \mathbf{h}. \]

\( Q(\mathbf{h}|\mathbf{v}) \) and \( P(\mathbf{v}|\mathbf{h}) \) both factorize

\[ Q(\mathbf{h}|\mathbf{v}) = \prod_{j} Q(h_j|\mathbf{v}) \]
\[ P(\mathbf{v}|\mathbf{h}) = \prod_{i} P(v_i|h). \]

Free energies can be calculated exactly Gibbs sampling very efficient. No need for variational Bayes. \( P(\mathbf{h}) \) is not factorial.
The free energy works out to be

\[ F(\mathbf{v}) = - \ln \prod_{j=1}^{J} \left( 1 + \exp \left( \mathbf{v}^T \mathbf{W}_j \right) \right) \]

If \( \mathbf{W}_i \) is the \( i \)th row of \( \mathbf{W} \) and \( \mathbf{W}_j \) the \( j \)th column,

\[ Q(h_j = 1 | \mathbf{v}) = \sigma(\mathbf{v}^T \mathbf{W}_j) \]

\[ P(v_i = 1 | \mathbf{h}) = \sigma(\mathbf{W}_i \cdot \mathbf{h}) \]
To calculate free energies, by the distributive law,

\[
\sum_h \exp(-E(v, h)) = \sum_h \exp(v^T W h)
\]

\[
= \sum_{h_1, \ldots, h_J} \exp\left(\sum_{j=1}^J v^T W_j h_j\right)
\]

\[
= \sum_{h_1, \ldots, h_J} \prod_{j=1}^J \exp\left(v^T W_j h_j\right)
\]

\[
= \prod_{j=1}^J \sum_{h_j \in \{0, 1\}} \exp\left(v^T W_j h_j\right)
\]

\[
= \prod_{j=1}^J \left(1 + \exp\left(v^T W_j\right)\right)
\]
\[
Q(h|v) = \frac{P(v, h)}{P(v)} \times \frac{e^{-E(v,h)}}{\sum_{h'} e^{-E(v,h')}}
= \frac{\prod_{j=1}^{J} \exp(v^T W_j h_j)}{\prod_{j=1}^{J} (1 + \exp(v^T W_j))}
= \prod_{j=1}^{J} \frac{\exp(v^T W_j h_j)}{(1 + \exp(v^T W_j))}
= \prod_{j=1}^{J} Q(h_j|v)
\]

\[
Q(h_j = 1|v) = \frac{\exp(v^T W_j)}{(1 + \exp(v^T W_j))}
= \sigma(v^T W_j)
\]
Gaussian-Bernoulli RBMs

\[ E(v, h) = \frac{1}{2}(v - b)^T(v - b) - c^T h - v^T Wh \quad (v \in \mathbb{R}^I) \]

\( P(v|h) \) is Gaussian with mean \( b + Wh \) and identity covariance matrix.

\[ Q(h|v) = \prod_j Q(h_j|v) \]

where

\[ Q(h_j = 1|v) = \sigma(c_j + v^T W_j). \]

The marginal \( P(h) \), that is,

\[ \int P(v, h) d\mathbf{v} \]

cannot be evaluated explicitly.
Even so, we can write $P(\mathbf{v})$ in the form of a Gaussian mixture:

$$P(\mathbf{v}) = \sum_h P(h)P(\mathbf{v}|h)$$

- Used to represent data vector $\mathbf{v}$ by array of Bernoulli probabilities $Q(h_j = | \mathbf{v})$ for subsequent variational Bayes processing with binary BMs.
- Data has to be preprocessed e.g. by pre-whitening (not needed for the “mean covariance RBM”).
- Speech recognition: $\mathbf{v}$ is an 11 frame window + cepstral derivatives.
- Speaker recognition (time being): $\mathbf{v}$ is an i-vector.
\[ E(x, y, z) = - \sum_{ijk} w_{ijk} x_i y_j z_k \]

Suppose that \( x \) are the visible variables. If \( x \) is fixed, \( Q(y, z|x) \) is a Boltzmann machine hence the conditionals \( Q(y|z, x) \) and \( Q(z|y, x) \) are factorial so Gibbs sampling is easy. A variational factorization of the form

\[ Q(y, z|x) = Q(y|x)Q(z|x) \]

implies a complete factorization

\[ Q(y, z|x) = \prod_j Q(y_j|x) \prod_k Q(z_k|x) \]
\[ \frac{\partial \langle \ln P(x, y, z) \rangle}{\partial w_{ijk}} = \langle x_i y_j z_k \rangle_{\text{data}} - \langle x_i y_j z_k \rangle_{\text{model}} \]

\[ = x_i \langle y_j \rangle_{\text{data}} \langle z_k \rangle_{\text{data}} - \langle x_i y_j z_k \rangle_{\text{model}} \]

if the visible variables are \( x \). To keep the number of free parameters manageable, decompose \( w_{ijk} \) as a sum of rank one terms

\[ w_{ijk} = \sum_{f=1}^{F} w_{if}^x w_{jf}^y w_{kf}^z \]
Sparse and deep BMs (DBMs)

No connections within layers.

**Figure:** Sparse Boltzmann machine with one visible and two hidden layers

Blocks of 0’s on the diagonal of the weight matrix.

\[
\begin{pmatrix}
0 & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 \\
\vdots & \ddots & \ddots & \vdots \\
0 & 0 & \cdots & 0
\end{pmatrix}
\]
By the Markov property, the distribution of the units in each layer conditioned on all of the others is factorial. Thus Gibbs sampling factorizes. For variational Bayes, assuming a partial factorization is enough to guarantee a full factorization.

Caveat: The joint distribution of a pair of layers is *not* an RBM and the corresponding weights cannot be estimated properly by using a Gibbs sampler that alternates between the two layers to implement persistent contrastive divergence. It is necessary to cycle over all the layers.
Suppose you have built a sparse or deep BM and with $l$ layers you would like to add another layer.

Produce “training data” by sampling from the variational posteriors at level $l$ calculated with a real training token $v$. Using a slightly modified version of RBM training, it is possible to estimate a weight matrix for the new layer in such a way that the variational lower bound is always guaranteed to increase.

Different from the Hinton/Salakhutdinov construction.
Assume that we have a representation of whole utterances (of arbitrary duration) by binary vectors (of fixed length), analogous to the i-vector representation. The speaker verification problem can be formulated as one of determining whether two collections of utterances $E$ and $T$ were uttered by the same speaker or by different speakers. The likelihood ratio is

$$\frac{P(E, T)}{P(E)P(T)}$$

Each term is the joint probability of a collection of utterances which are *not* independent. The joint distributions should be permutation invariant. It is enough to be able to calculate variational free energies.
Figure: Example of the Siamese twin construction
Figure: Tied hidden variables
Figure: An RBM with tied hidden variables (on top) and untied hidden variables (on the bottom)
For each node \( i \) in the graph, let \( Pa(i) \) denote the set of parents of \( i \), \( C(i) \) the set of children.

\[
P(x) = \prod_i P(x_i | x_{Pa(i)})
\]

where

\[
P(x_i = 1 | x_{Pa(i)}) = \sigma(u_i)
\]
\[
u_i = \sum_{p \in Pa(i)} w_{ip} x_p.
\]

\( w_{ip} \) is the weight associated with the \textit{directed} branch, \( p \rightarrow i \). In a deep network, the activation of a node depends only on the layer above the node. If all nodes are visible, no problems with partition functions or sampling from the prior \( P(x) \).
Unlike a deep Boltzmann machine (DBM), Gibbs sampling and variational Bayes are infeasible in practice. "Explaining away". The distribution $P(h^l|h^{l+1})$ is factorial but the distribution $Q(h^l|h^{l+1}, h^{l-1})$ is not.
Training without hidden nodes

\[
\frac{\partial}{\partial w_{ip}} \ln P(x) = \frac{\partial}{\partial w_{ip}} \ln P(x_i | x_{Pa(i)})
\]

\[
\frac{\partial}{\partial w_{ip}} \ln P(x_i | x_{Pa(i)}) = x_p x_i - x_p \sigma(u_i)
\]

\[
\sigma(u_i) = P(x_i = 1 | x_{Pa(i)})
\]

so \[
\frac{\partial}{\partial w_{ip}} \ln P(x_i | x_{Pa(i)}) \approx x_p x_i - x_p \tilde{x}_i
\]

where \(\tilde{x}\) is a stochastic reconstruction of \(x\) (\(\tilde{x}_i = 1\) with probability \(\sigma(u_i)\)). Compare CD-1.
Training with hidden nodes

Given a training token $v$, sample $x \sim Q(x|v)$ and create a stochastic 1-step reconstruction $\tilde{x}$

$$\frac{\partial}{\partial w_{ip}} \langle \ln P(x_i|x_{Pa(i)}) \rangle_{data} \approx x_p x_i - x_p \tilde{x}_i$$

Unfortunately, neither Gibbs sampling nor variational Bayes works in practice.
To calculate $Q(x_i|x_{\setminus i})$, note that the factors in the prior $P(x)$ which involve $x_i$ are $P(x_i|x_{Pa(i)})$ and $P(x_j|x_{Pa(j)})$ for $i \in Pa(j)$ i.e. $j \in C(i)$. So

$$Q(x_i|x_{\setminus i}) \propto P(x_i|x_{Pa(i)}) \prod_{j \in C(i)} P(x_j|x_{Pa(j)})$$

*Markov Blanket of $i$: parents of $i$, children of $i$, parents of the children of $i$.*

In a DBN: the Markov blanket is the layer containing $i$, the layer above and the layer below.

Calculating $Q(x_i|x_{\setminus i})$ requires taking account of the neighbours of $i$ on the same layer. No factorization as in a deep Boltzmann machine.

Variational Bayes doesn’t factorize either.
The wake-sleep algorithm

Two ways of writing the EM auxiliary function are

\[
\langle \ln P(v, h) \rangle + H(Q(h|v)) \\
\ln P(v) - D(Q(h|v)\|P(h|v)).
\]

To improve the fit of the model to a given data vector \( v \), we can

1. Hold the posterior \( Q(h|v) \) fixed and increase \( \langle \ln P(x) \rangle \) (where \( x \) is a short-hand for \( (h, v) \))

2. Hold the model \( P(v) \) fixed and decrease the divergence between the true posterior \( P(h|v) \) and the approximate posterior \( Q(h|v) \)

and alternate between the two.
For 1), if it is possible to sample from \( x \sim Q(x|v) \), we use the stochastic approximation

\[
\frac{\partial \langle \ln P(x) \rangle}{\partial w_{ip}} \approx x_p x_i - x_p \tilde{x}_i
\]

where \( \tilde{x} \) is a stochastic reconstruction of \( x \).

For 2), approximate the posterior by a deep belief net with the arrows reversed and a different set of weights (“recognition weights”).

\[
Q(x|v) = \prod_i Q(x_i|x_{C(i)}),
\]

Recall that it is easy to train a deep belief net if all the variables are visible. For this, create a synthetic training set \( \{v^n, h^n\} \) by sampling from the model distribution \( P(v, h) \). (The model is dreaming.)
The quantity that is being optimized is the log likelihood of this synthetic training set calculated with the dual deep belief net (thus the optimization is with respect to $Q$)

$$
\sum_n \ln Q(h^n|v^n) = \sum_n \sum_h P(h|v^n) \ln Q(h|v^n)
\equiv - \sum_n D(P(h|v^n)\|Q(h|v^n))
$$

There are two problems with this: a synthetic training set rather than the real training set is being used and the optimand involves divergences of the form $D(P(h|v)\|Q(h|v))$, rather than $D(Q(h|v)\|P(h|v))$ as required by the EM algorithm.
Mean field approximation for $Q$

We use the superscript $l$ to indicate variables associated with nodes at level $l$. Given a data vector $\mathbf{v}$, set

$$Q(h_i^l = 1| \mathbf{v}) = \mu_i^l$$

In the mean field approximation, $\mu_j^{l+1}$ is calculated from the Bernoulli probabilities at level $l$ using the recognition weights on the branches joining level $l$ to level $l+1$

$$\mu_j^{l+1} = \sigma \left( \sum_i w_{ji} \mu_i^l \right).$$
A set of layers which implement the mean field posterior calculation for a DBN

A softmax layer for making recognition decisions

The interpretation of “parents” and “children” is dictated by the direction of the arrows
Softmax nodes

The activation $u_i$ of a node $i$ is $\mathbf{w}_i^T \mathbf{x}$ where $\mathbf{x}$ is the configuration of the layer underneath it. (This is the argument for the sigmoid activation function, not the value returned by the activation function.)

For “label” nodes, $\sigma$ is replaced by the softmax function. The probability assigned to the $i$th class label is

$$p_i = \frac{e^{u_i}}{\sum_j e^{u_j}}$$

The error signal associated with a training token is usually taken to be

$$\mathcal{E} = -\sum_i t_i \ln p_i$$

where the target value $t_i$ is 1 if $i$ is the correct class label for the training token and 0 otherwise.
Backpropagation

Object is to calculate $\frac{\partial E}{\partial w_{ip}}$. It is enough to calculate $\frac{\partial E}{\partial u_i}$ since

$$\frac{\partial E}{\partial w_{ip}} = \frac{\partial E}{\partial u_i} \frac{\partial u_i}{\partial w_{ip}}$$

$$= \frac{\partial E}{\partial u_i} u_{ip}$$

because $u_i = \sum_p w_{ip} x_p$ where $p$ ranges over the parents of $i$.

If $i$ is not a label node, $x_i = \sigma(u_i)$ so

$$\frac{\partial E}{\partial u_i} = \frac{\partial E}{\partial x_i} \frac{\partial x_i}{\partial u_i}$$

$$= \frac{\partial E}{\partial x_i} \sigma'(u_i).$$
Since \( u_j = \sum_i w_{ji} x_i \),

\[
\frac{\partial \varepsilon}{\partial x_i} = \sum_j \frac{\partial \varepsilon}{\partial u_j} \frac{\partial u_j}{\partial x_i} = \sum_j w_{ji} \frac{\partial \varepsilon}{\partial u_j}
\]

Combining these,

\[
\frac{\partial \varepsilon}{\partial u_i} = \sigma'(u_i) \sum_j w_{ji} \frac{\partial \varepsilon}{\partial u_j}.
\]

\( w_{ji} \) is the weight associated with the branch \( i \rightarrow j \). Hence \( w_{ji} = 0 \) unless \( j \) is in the layer above \( i \). Thus the derivatives propagate backwards (from the labels to the data).
To initialize the recursion: if \( i \) is a label node,

\[
p_i = \frac{e^{u_i}}{\sum_j e^{u_j}}
\]

so

\[
\frac{\partial \mathcal{E}}{\partial u_i} = - \sum_k t_k \frac{\partial}{\partial u_i} \left( u_k - \ln \sum_j e^{u_j} \right)
\]

\[
= - \sum_k t_k \left( \delta_{ik} - \frac{1}{\sum_j e^{u_j}} \sum_j e^{u_j} \delta_{ij} \right)
\]

\[
= - \sum_k t_k (\delta_{ik} - p_i)
\]

\[
= -(t_i - p_i)
\]
Speech recognition

- Represent acoustic observation vectors $\mathbf{v}$ as arrays of Bernoulli probabilities $Q(h_j|\mathbf{v})$ using a Gaussian-Bernoulli RBM so they can serve as input to a neural network.
- Using a forced Viterbi alignment, label each vector as belonging to a triphone state (rather than a phoneme).
- Train the neural network to discriminate between triphone states.
- Convert the discriminative probabilities to generative probabilities

$$P(\mathbf{v}|c) = \frac{P(c|\mathbf{v})P(\mathbf{v})}{P(c)}.$$ 

where $c$ is a class label. (No need to calculate $P(\mathbf{v})$)
Use the generative probabilities $P(v|c)$ as HMM output distributions in a conventional single pass, non-adaptive recognizer (Viterbi decoder)

30% error rate reductions compared with single pass MMI-HMM

Adapting discriminative distributions may not be easy

Backpropagation appears to work with random initializations, even for deep networks

Even a single hidden layer works well

Proposed explanations: triphone state inventory, Gaussian-Bernoulli RBM
Hinton’s construction of DBNs

These models all equivalent, provided that the weight matrices are tied. For the second graph, the pair \((h^1, h^2)\) is a mirror image of the pair \((v, h^1)\) in the first graph. For the third graph, the pair \((h^2, h^3)\) is a mirror image of the pair \((h^1, h^2)\) in the second graph.

So untie the weight matrices.
Like a standard belief net, a Hinton deep belief net can be converted into a feed forward neural net by using a mean field approximate posterior calculation.
Given training data \( \{ \mathbf{v}^n \} \), Hinton shows how to

1. Add successive layers, so that the top-level RBM is trained in the usual way (with a synthetic training set)

2. Do a bottom-up approximate posterior calculation for all of the hidden variables

The variational lower bound calculated with the approximate posterior is guaranteed to increase whenever a layer is added, provided the dimensions of the weight matrices are consistent with the mirror image condition.
Suppose you have a single layer with weight matrix $W^1$ and you want to add another layer with weight matrix $W^2$.

For the synthetic training set, sample $h^1 \sim Q(h^1|v^n)$

Optimize

$$
\sum_n \sum_{h^1} \ln P(v^n, h^1|W^1, W^2) = \sum_n \sum_{h^1} \ln P(h^1|W^2) + \text{constant}
$$

since $P(h^1|v^n, W^1)$ remains unchanged. This is just the criterion optimized by RBM training of $W^2$ (where the $h^1$’s are treated as visible). If the mirror image condition is satisfied, $W^2$ can be initialized with $W^1$ and

$$
\sum_n \sum_{h^1} \ln P(v^n, h^1|W^1, W^2) \geq \sum_n \sum_{h^1} \ln P(v^n, h^1|W^1, W^1)
$$

Since $h^1 \sim Q(h^1|v^n)$ the right hand side can be interpreted as the lower bound for the single level RBM.
It is necessary to define an approximate joint posterior $Q(h^1, h^2|v^n)$ in such a way that the right hand side above can be interpreted as the lower bound for the 2 level model. Some algebra shows that this definition works:

$$Q(h^1, h^2|v^n) = Q(h^1|v)Q(h^2|h^1)$$

where the first term is evaluated with $W^1$ and the second with $W^2$. (Both calculations are RBM posteriors. Use mean field in practice.)

As for standard DBNs the exact posterior is intractable

$$\frac{P(v|h^1, W^1)P(h^1, h^2|W^2)}{P(v)}$$

since $P(v)$ cannot be evaluated.
Some objections

- Inherently greedy

- Bottom up posterior calculation fails to take account of higher levels

(Contrary to deep Boltzmann machines.) Can be remedied by a modified wake-sleep algorithm with two sets of weight matrices, one for sampling from the generative model, the other for mean-field posterior calculations. Softmax layer can be included. Enabled Hinton, Osindero and Teh to obtain state of the art recognition results on MNIST digits with a purely generative approach.
Inferring the hidden variables (i.e. calculating posteriors) in a generatively trained DBN is interpreted as feature extraction with the topmost hidden variables $h^L$ being the highest level features. With this interpretation, the highest level features should be the most useful for pattern recognition, so assume

\[ Q(c|v) = \sum_{h^L} Q(c|h^L)Q(h^L|v) \]

where $Q(c|h^L)$ is implemented by a softmax layer and $Q(h^L|v)$ by a mean field calculation in a DBN. Formally, this is just a feed forward neural net, so use backpropagation for fine tuning.

Whether this is really helpful in speech recognition remains to be seen.
The DBN construction can be used to initialize DBMs. In a 2 layer DBN,

\[ P(v, h^1, h^2) = P(v|h^1, W^1)P(h^1, h^2|W^2). \]

which is not a Boltzmann distribution. If we were to take the product

\[ P(v, h^1|W^1)P(h^1, h^2|W^2) \]

and normalize the probabilities so that they sum to 1, we would get a Boltzmann distribution but this expression counts the prior on \( h^1 \) twice. So halve the weight matrices \( W^1 \) and \( W^2 \).

DBMs appear to be better feature extractors than DBNs and the training algorithm and posterior calculations are less crude. But calculating posteriors by cycling over all the levels in a DBM (as required by variational Bayes), rather than a single feed forward pass (as in DBNs) does not translate into an obvious way of initializing deep neural networks.