Notes on Boltzmann Machines

Patrick Kenny
Centre de recherche informatique de Montréal
Patrick.Kenny@crim.ca

I. Introduction

Boltzmann machines are probability distributions on high dimensional binary vectors which are analogous to Gaussian Markov Random Fields in that they are fully determined by first and second order moments. A key difference however is that augmenting Boltzmann machines with hidden variables enlarges the class of distributions that can be modeled, so that in principle it is possible to model distributions of arbitrary complexity [1]. (On the other hand, marginalizing over hidden variables in a Gaussian distribution merely gives another Gaussian.) Training Boltzmann machines still seems to be more of an art than a science, but a variational Bayes expectation maximization algorithm has been developed which deals with this problem in a reasonably efficient way for a class of sparsely connected Boltzmann machines that includes the deep Boltzmann machines studied in [2]. I will explain how this approach provides a principled framework for constructing complex Boltzmann machines incrementally and how it can be extended straightforwardly to handle higher order Boltzmann machines (e.g. Boltzmann machines defined by third order moments).

The binary/Gaussian distinction is not an exclusive dichotomy: hybrid models containing both types of variable can be constructed. The term Gaussian-Bernoulli Boltzmann machine is usually used to refer to a model in which the vector of visible variables $\boldsymbol{v}$ is continuous and the vector of hidden variables $\boldsymbol{h}$ is binary. For each binary configuration $\boldsymbol{h}$, the conditional distribution of $\boldsymbol{v}$ given $\boldsymbol{h}$ is Gaussian so writing

$$P(\boldsymbol{v}) \quad = \quad \sum_{\boldsymbol{h}} P(\boldsymbol{h}) P(\boldsymbol{v}|\boldsymbol{h})$$

shows that this model can be interpreted as a Gaussian mixture with a prodigious number of components.[1] This type of model is frequently use to binarize continuous data (for example it would be possible in to associate with each data vector $\boldsymbol{v}$, the maximum *a posteriori* estimate of $\boldsymbol{h}$) so that binary Boltzmann machines can be used in subsequent processing. In speech processing applications, a model like this can be used as a Universal Background Model and it can accommodate data vectors of much higher dimension than traditional Gaussian mixture models. In particular, cepstral coefficients extracted from 10 – 15 frames can be concatenated and treated as a single data vector.

Boltzmann machines may be used to implement either generative or discriminative approaches to pattern recognition problems. Broadly speaking, generative approaches work best in speaker recognition and discriminative approaches work best in speech recogntion. State of the art speaker recognition systems use several types of generative model as feature extractors (the Universal Background Model, Joint Factor Analysis and the i-vector extractor) and as classifiers (Probabilistic Linear Discriminant Analysis and the cosine distance metric) [4], [5], [6]. These generative models rely heavily on Gaussian assumptions, some of which are quite questionable. Using Boltzmann machines to develop alternative generative models for speaker recognition promises to be an interesting line of research. I will sketch very briefly how such a program might be carried out.

In the machine learning literature, Boltzmann machines are principally used in unsupervised training of another type of generative model known as a sigmoid belief network or deep belief network (DBN). The hidden variables in a DBN can be thought of as providing causal explanations of data so that calculating the posteriors of the hidden variables in the DBN can be viewed as a sort of high level feature extraction. The standard (mean-field) approximation used for posterior calculations in a DBN is implemented by a feed forward neural net (where the top level softmax layer usually used to make recognition decisions is missing). Suppose then that a DBN can be trained in an unsupervised way on a given data population in such a way that the high level feature extractors learned in the course of training are useful for recognizing patterns in the data. Under such circumstances, it would be reasonable to use the posterior calculations in the DBN as an initialization for backpropagation training of a feed forward neural network designed to discriminate between these patterns.

This strategy has been successfully deployed in speech recognition, notably at Microsoft. Interestingly, Microsoft has also recently reported excellent speech recognition results obtained with unaided backpropagation in deep neural nets [7] (that is, without using a DBN-based initialization) so the question of whether deep belief networks will play an important role in speech recognition in the future remains to be settled.

For background reading, the principal reference is [1]. See also [8], [3] (particularly the Chapter on Boltzmann machines) and [9] (Chapter 11). The January 2012 issue of the IEEE Transactions on Audio, Speech and Language Processing contained a special section on Deep Learning.

[1] To get an idea of how large the number 2^{1000} is, see Chapter 29 of [3].

II. Preliminaries

This section summarizes the probabilistic arguments needed to understand the literature on Boltzmann machines and deep belief networks. See [8].

Conventions We use $P(\cdot)$ to indicate probabilities calculated with a given model, irrespective of whether these probabilities can be evaluated exactly or not. As a rule, we use the symbol $Q(\cdot)$ to refer to posterior probabilities of hidden variables given data. These posteriors may be approximate or exact, depending on the context.

We use $\boldsymbol{h}$ to denote hidden vector-valued variables, $\boldsymbol{v}$ to denote visible variables and $\boldsymbol{x}$ to denote generic variables (in situations where there is no need to distinguish between hidden and visible variables).

The Kullback-Leibler divergence between two distributions $P_1(\boldsymbol{x})$ and $P_2(\boldsymbol{x})$ is denoted by $D\left(P_1(\boldsymbol{x})||P_2(\boldsymbol{x})\right)$ and given by

$$D\left(P_1(\boldsymbol{x})||P_2(\boldsymbol{x})\right) \quad = \quad \sum_{\boldsymbol{x}} P_1(\boldsymbol{x}) \ln \frac{P_1(\boldsymbol{x})}{P_2(\boldsymbol{x})}.$$

It has the property that $D\left(P_1(\boldsymbol{x})||P_2(\boldsymbol{x})\right) \geq 0$ with equality iff P_1 and P_2 are identical. The divergence is an asymmetric measure of the difference between the two distributions.

The entropy of $P(\boldsymbol{x})$ is denoted by $H(P(\boldsymbol{x}))$ and given by

$$H(P(\boldsymbol{x})) \quad = \quad -\sum_{\boldsymbol{x}} P(\boldsymbol{x}) \ln P(\boldsymbol{x}).$$

It has the property that $H(P(\boldsymbol{x})) \geq 0$ with equality iff $P(\boldsymbol{x})$ is concentrated on a single value of $\boldsymbol{x}$.

A. The evidence inequality

Suppose we have a model with both hidden and visible variables. The marginal probability of the visible variables, given by

$$P(\boldsymbol{v}) \quad = \quad \sum_{\boldsymbol{h}} P(\boldsymbol{v}, \boldsymbol{h})$$

is known as the *evidence*. For interesting models, this summation is intractable. The problem of evaluating or approximating $P(\boldsymbol{v})$ turns out to be equivalent to the problem of evaluating or approximating the posterior $Q(\boldsymbol{h}|\boldsymbol{v})$.

Define

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

where the expectation is taken with respect to $Q(\boldsymbol{h}|\boldsymbol{v})$ which may be the exact posterior or an approximate posterior.

This can be written in several different forms, all of them useful:

$$\begin{aligned} & E\left[\ln P(\boldsymbol{v}, \boldsymbol{h})\right] + H(Q(\boldsymbol{h}|\boldsymbol{v})) \\ E\left[\ln P(\boldsymbol{v}|\boldsymbol{h})\right] - & \ D\left(Q(\boldsymbol{h}|\boldsymbol{v})||P(\boldsymbol{h})\right) \\ \ln P(\boldsymbol{v}) - & \ D\left(Q(\boldsymbol{h}|\boldsymbol{v})||P(\boldsymbol{h}|\boldsymbol{v})\right). \end{aligned}$$

The first follows from the definition of $\mathcal{L}$ and the second from the first (write $P(\boldsymbol{v}, \boldsymbol{h}) = P(\boldsymbol{v}|\boldsymbol{h})P(\boldsymbol{h})$). In the third expression, we use the notation $P(\boldsymbol{h}|\boldsymbol{v})$ to indicate the *exact* posterior of $\boldsymbol{h}$ given $\boldsymbol{v}$ so that

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

The non-negativity property of the Kullback-Leibler divergence implies that $\mathcal{L} \leq \ln P(\boldsymbol{v})$ with equality holding iff the posterior is exact. This is the *evidence inequality*.

The lower bound is sometimes written as

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

where "data" refers to the posterior distribution $Q(\boldsymbol{h}|\boldsymbol{v})$. (Similarly, $\langle \cdot \rangle_{\text{model}}$ is used to refer to an expectation taken with respect to the prior distribution $P(\boldsymbol{v}, \boldsymbol{h})$.)

The quantity $E\left[\ln P(\boldsymbol{v}, \boldsymbol{h})\right]$ is known as the *EM auxiliary function*.

B. The EM algorithm

If we are given a training set consisting of a collection of visible vectors $\{\boldsymbol{v}^n\}$, the lower bound is given by

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

Evaluating these expectations is the E-step of the EM algorithm.

In order to fit the model to the data we want to make $\mathcal{L}$ as large as possible. $\mathcal{L}$ can be increased in two ways:

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

$$\sum_n \sum_{\boldsymbol{h}} Q(\boldsymbol{h}|\boldsymbol{v}^n) \ln P(\boldsymbol{h}, \boldsymbol{v}^n)$$

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

$$\sum_n D\left(Q(\boldsymbol{h}|\boldsymbol{v}^n) \| P(\boldsymbol{h}|\boldsymbol{v}^n)\right).$$

Writing

$$\ln P(\boldsymbol{v}, \boldsymbol{h}) = \ln P(\boldsymbol{h}) + \ln P(\boldsymbol{v}|\boldsymbol{h}),$$

we have 2 special cases of 1), namely adjusting $P(\boldsymbol{v}|\boldsymbol{h})$ so as to maximize $\sum_n \sum_{\boldsymbol{h}} Q(\boldsymbol{h}|\boldsymbol{v}^n) \ln P(\boldsymbol{v}^n|\boldsymbol{h})$ and adjusting $P(\boldsymbol{h})$ so as to maximize $\sum_n \sum_{\boldsymbol{h}} Q(\boldsymbol{h}|\boldsymbol{v}^n) \ln P(\boldsymbol{h})$. The latter is equivalent to minimizing the divergence

$$\sum_n D\left(Q(\boldsymbol{h}|\boldsymbol{v}^n) \| P(\boldsymbol{h})\right)$$

and is sometimes referred to as *minimum divergence* estimation of $P(\boldsymbol{h})$.

In the case where $Q(\boldsymbol{h}|\boldsymbol{v}) = P(\boldsymbol{h}|\boldsymbol{v})$ the lower bound is tight ($\ln P(\boldsymbol{v}) = \mathcal{L}$) 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

C. Variational Bayes

In situations where the posterior is intractable we can seek an approximation of the form

$$Q(\boldsymbol{h}|\boldsymbol{v}) \quad = \quad \prod_i Q(h_i|\boldsymbol{v})$$

where the factors on the right hand side are chose so as to maximize $\mathcal{L}$. Fix i and suppose we want to estimate $Q(h_i|\boldsymbol{v})$, assuming that $Q(h_j\boldsymbol{v})$ fixed for $j \neq i$. Writing $\boldsymbol{h} = (h_i, \boldsymbol{h}_{\setminus i})$,

$$\begin{aligned}
\mathcal{L} &= E_{Q(\boldsymbol{h}|\boldsymbol{v})} \left[\ln \frac{P(\boldsymbol{v}, \boldsymbol{h})}{Q(\boldsymbol{h}|\boldsymbol{v})} \right] \\
&= E_{Q(h_i)} \left[E_{Q(\boldsymbol{h}_{\setminus i})} \left[\ln \frac{P(h_i, \boldsymbol{h}_{\setminus i}, \boldsymbol{v})}{Q(h_i) Q(\boldsymbol{h}_{\setminus i})} \right] \right] \\
&= E_{Q(h_i)} \left[E_{Q(\boldsymbol{h}_{\setminus i})} \left[\ln \frac{P(h_i, \boldsymbol{h}_{\setminus i}, \boldsymbol{v})}{Q(h_i)} \right] \right] + \text{ constant} \\
&= E_{Q(h_i)} \left[\ln \frac{\tilde{Q}(h_i)}{Q(h_i)} \right] + \text{ constant} \\
&= -D\left(Q(h_i) \| \tilde{Q}(h_i) \right)
\end{aligned}$$

where $\tilde{Q}(h_i)$ is the distribution defined by

$$\ln \tilde{Q}(h_i) \quad = \quad E_{Q(\boldsymbol{h}_{\setminus i})} \left[\ln P(h_i, \boldsymbol{h}_{\setminus i}, \boldsymbol{v}) \right] + \text{ constant},$$

the constant being determined by the condition that $\sum_{h_i} \tilde{Q}(h_i) = 1$. (In future, we will use the symbol $\equiv$ to indicate equality up to an irrelevant additive constant.) Thus to maximize $\mathcal{L}$, we minimize the divergence and take $Q(h_i) = \tilde{Q}(h_i)$. It is necessary to cycle through the factors until convergence. (Convergence is guaranteed.)

D. Gibbs sampling

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

$$\boldsymbol{x}^1 \to \boldsymbol{x}^2 \to \ldots \to \boldsymbol{x}^n \ldots$$

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 $\boldsymbol{x}^1 \to \boldsymbol{x}^2, \boldsymbol{x}^2 \to \boldsymbol{x}^3, \ldots$.) For n sufficiently large, $\boldsymbol{x}^n$ will be distributed according to $P(\boldsymbol{x})$. (The model distribution is the equilibrium distribution of the Markov chain.)

The principle extends straightforwardly to posterior distributions so that samples can be drawn from a posterior $Q(\boldsymbol{h}|\boldsymbol{v})$ provided that for each i it is easy to sample from $Q(h_i|\boldsymbol{h}_{\setminus i}, \boldsymbol{v})$. Generally speaking Gibbs sampling is much more computationally expensive than variational Bayes (it is an exact method but variational Bayes is not).

III. Boltzmann Machines

A Boltzmann machine is a probability distribution on binary vectors $\boldsymbol{x}$ of the form

$$P(\boldsymbol{x}) \quad = \quad \frac{1}{Z} e^{-E(\boldsymbol{x})}$$

where the "energy function" $E(\boldsymbol{x})$ has the form

$$E(\boldsymbol{x}) \quad = \quad -\sum_{i<j} x_i w_{ij} x_j,$$

the sum extending over all pairs (i, j) such that $i < j$. The normalizing constant Z is referred to as the partition function.

Alternatively

$$E(\boldsymbol{x}) \quad = \quad -\frac{1}{2} \sum_{i,j} x_i w_{ij} x_j,$$

the sum extending over all pairs (i, j). Units are sometimes referred to as neurons and the weights w_{ij} as synaptic weights. We denote the weight matrix by $\boldsymbol{W}$. Some of the units may be distinguished as visible and others as hidden. When it is necessary to make this distinction we will use the symbol v_i for the binary variable associated with a visible unit i and the symbol h_j for the binary variable associated with a hidden unit j. The matrix $\boldsymbol{W}$ is assumed to be symmetric with zero diagonal. (The diagonal entries are forced to be zero to prevent neurons from interacting with themselves.) First order terms such as $\sum_j w_j x_j$ could also be included but there is no gain in generality as this is equivalent to forcing some of the components of $\boldsymbol{x}$ to be 1. We will exclude these "bias" terms in order to keep the derivations simple.

Energy based models differ from probability distributions defined by directed acyclic graphs (Bayesian networks). In particular, it is not usually straightforward to evaluate the partition function and we will not address this question for Boltzmann machines as it is not needed for the applications we have in mind (see [2], [8]). The energy function can be thought of as specifying soft constraints on the data (regions of low energy – sometimes referred to as ravines in the energy surface – have high probability) and additional constraints can be imposed by adding terms to the energy function. But the energy function does not provide a simple recipe for sampling from the probability distribution. (Some type of Markov Chain Monte Carlo (MCMC) method such as Gibbs sampling or the Metropolis algorithm is needed.)

A. Gibbs sampling

The Gibbs sampling formulas shows that a Boltzmann machine can be regarded as a stochastic neural network.

$$\begin{aligned} Q(x_i = 1|\boldsymbol{x}_{\setminus i}) \quad &= \quad \frac{P(x_i = 1, \boldsymbol{x}_{\setminus i})}{P(x_i = 0, \boldsymbol{x}_{\setminus i}) + P(x_i = 1, \boldsymbol{x}_{\setminus i})} \\ &= \quad \frac{\exp\left(\sum_{j \neq i} w_{ij} x_j\right)}{1 + \exp\left(\sum_{j \neq i} w_{ij} x_j\right)} \\ &= \quad \sigma\left(\sum_{j \neq i} w_{ij} x_j\right) \end{aligned}$$

where σ is the sigmoid (that is, S-shaped) non-linearity defined by

$$\sigma(u) = (1 + e^{-u})^{-1}.$$

It is easily verified that

$$\begin{aligned}\sigma(-u)+\sigma(u) &= 1\\ \sigma'(u) &= \sigma(u)(1-\sigma(u)).\end{aligned}$$

The argument of σ (rather than the value it returns), namely $\sum_{j\neq i} w_{ij}x_j$, is referred to in [8] as the *activation* of the unit i and σ is called the activation function.

B. The mean field approximation

In order to convert the Boltzmann machine into a deterministic neural network we can apply variational Bayes to the prior distribution $P(\boldsymbol{x})$, that is, we calculate a variational approximation

$$\begin{aligned}P(\boldsymbol{x}) &\approx \prod_i Q(x_i)\\ &= \prod_i \mu_i^{x_i}\end{aligned}$$

where $\mu_i = Q(x_i = 1)$. (In the context of Boltzmann machines, variational Bayes is usually referred to as the mean field approximation.)

The variational update equations are

$$\begin{aligned}\ln Q(x_i) &\equiv E_{Q(\boldsymbol{x}_{\setminus i})}\left[\ln P(x_i, \boldsymbol{x}_{\setminus i})\right]\\ &= E_{Q(\boldsymbol{x}_{\setminus i})}\left[\sum_{j\neq i} x_i w_{ij} x_j\right]\\ &= \begin{cases}\sum_{j\neq i} w_{ij}\mu_j & \text{if} \quad x_i = 1\\ 0 & \text{if} \quad x_i = 0\end{cases}\end{aligned}$$

so

$$\begin{aligned}Q(x_i = 1) &= \frac{\exp\left(\sum_{j\neq i} w_{ij}\mu_j\right)}{1+\exp\left(\sum_{j\neq i} w_{ij}\mu_j\right)}\\ &= \sigma\left(\sum_{j\neq i} w_{ij}\mu_j\right)\end{aligned}$$

Since $\mu_i = Q(x_i = 1)$, the fixed point equations are

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

These are the same as the equations for Gibbs sampling except that the x's are replaced by their mean values, the μ's. This explains the term "mean field" and shows how the Boltzmann distribution can be approximated by a deterministic neural net.

Note that the mean field approximation is *not* a good approximation to $P(\boldsymbol{x})$ since it treats the units as statistically independent and the approximate distribution is unimodal. (A unimodal approximation is generally only reasonable for approximating the posterior of hidden variables given an observation. The hidden variables are usually thought of as explaining the observation; to say that the posterior distribution on the hidden variables is unimodal just means that there is only one way of explaining the obseration.) In practice, variational Bayes should only be applied to posterior distributions, not prior distributions.

C. The Markov property

The most interesting situation is when the matrix $\boldsymbol{W}$ is sparse. Let $\boldsymbol{x}_{\setminus\{i,j\}}$ denote the set of variables other than x_i and x_j. If $\boldsymbol{x}_{\setminus\{i,j\}}$ is given and $w_{ij} = 0$ can write the energy function as

$$E(\boldsymbol{x}) = ax_i + bx_j + c$$

where a, b and c depend on $\boldsymbol{x}_{\setminus\{i,j\}}$ but not on x_i or x_j. Thus x_i and x_j are conditionally independent if $\boldsymbol{x}_{\setminus\{i,j\}}$ is given, provided that $w_{ij} = 0$.

A Boltzmann machine is represented by a graphical model with *undirected* edges joining all pairs of vertices i, j for which w_{ij} is *not* zero. The conditional independence property can be generalized as follows. 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 vertices in A and C. Then $\boldsymbol{x}_A$ and $\boldsymbol{x}_C$

are conditionally independent if $\boldsymbol{x}_B$ is given where $\boldsymbol{x}_A = \{x_i : i \in A\}$ etc. This is called the Markov property: the "future" (C) is independent of the "past" (A) if the "present" (B) is given.

So the graph provides information about how conditional distributions such as $P(\cdot|\boldsymbol{x}_B)$ factorize. (This is different from the case of directed graphical models where the topology of the graph tells you how the *prior* distribution factorizes.) With practice, it is possible to tell by inspecting an undirected graphichal model whether Gibbs sampling and variational Bayes are practically feasible.

Restricted Boltzmann machines In this case there is a hidden layer and a visible layer with no hidden-to-hidden or visible-to-visible connections. The vectors $\boldsymbol{h}, \boldsymbol{v}$ are of dimension $J \times 1$ and $I \times 1$ and $\boldsymbol{W}$ is of dimension $I \times J$. The energy function is

$$E(\boldsymbol{v}, \boldsymbol{h}) \quad = \quad -\boldsymbol{v}^T \boldsymbol{W} \boldsymbol{h}.$$

By the Markov property, $Q(\boldsymbol{h}|\boldsymbol{v})$ and $P(\boldsymbol{v}|\boldsymbol{h})$ both factorize

$$\begin{aligned} Q(\boldsymbol{h}|\boldsymbol{v}) &= \prod_j Q(h_j|\boldsymbol{v}) \\ P(\boldsymbol{v}|\boldsymbol{h}) &= \prod_i P(v_i|\boldsymbol{h}) \end{aligned}$$

and it turns out that each of the factors can easily be evaluated exactly. Thus there is no need for variational Bayes and Gibbs sampling can be implemented efficiently by alternating between the hidden and visible levels. This is known as *block Gibbs sampling*.

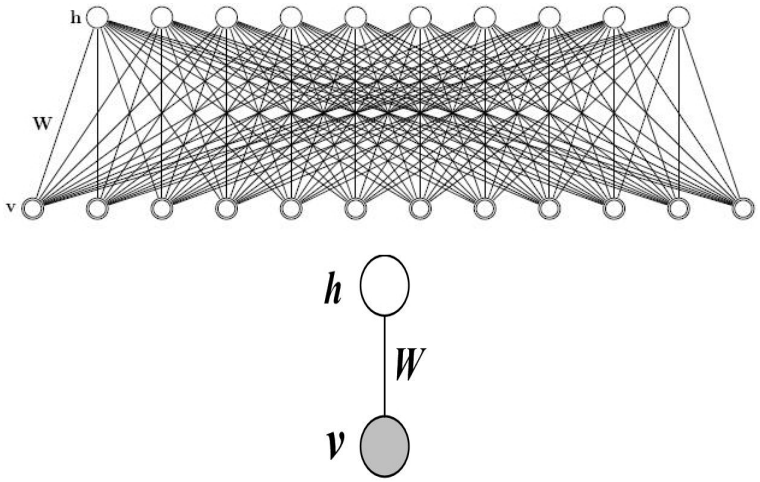

Fig. 1. Restricted Boltzmann machine

The marginal distribution $P(\boldsymbol{v})$ does not factorize (if it did the model would be trivial) and, by symmetry, the same is true of the prior $P(\boldsymbol{h})$. (Thus the hidden variables are *not* independent in the prior, contrary to what experience with directed graphical models might suggest.) We will see later how to evaluate $P(\boldsymbol{v})$ and $P(\boldsymbol{h})$ up to the partition function Z.

Gaussian Markov random fields The continuous analog of a Boltzmann distribution is a multivariate Gaussian distribution with a sparse precision matrix (i.e. inverse covariance matrix) $\boldsymbol{W}$, otherwise known as a Gaussian Markov random field. (In order for the distribution to be normalizable the precision matrix has to be positive definite so the diagonal entries of $\boldsymbol{W}$ have to be strictly positive in this case. On the other hand in the binary case we assumed that the diagonal entries are zero.) Like Boltzmann machines, Gaussian Markov random fields are not straightforward to implement in very high dimensions. In order to evaluate the partition function, the determinant of the precision matrix has to be evaluated and in order to sample from the distribution (without resorting to Gibbs sampling) the precision matrix would need to be diagonalized. As for maximum likelihood estimation, if sparsity constraints are not imposed, the correct procedure would be to estimate the covariance matrix by equating the model covariance matrix and the training data covariance matrix

$$\langle x_i x_j \rangle_{\text{data}} \quad = \quad \langle x_i x_j \rangle_{\text{model}}$$

and then invert the covariance matrix to get the precision matrix (we have assumed that the mean vector is zero for simplicity). However, imposing sparsity constraints on the precision matrix is not straightforward (although imposing sparsity constraints on the covariance matrix presents no difficulty).

D. Role of hidden units

Both Boltzmann machines and Gaussian Markov random fields can model only the second order statistics of data. Perhaps the most interesting difference between the two is that including hidden variables in Boltzmann machines extends the class of distributions that can be modeled but this is not the case for Gaussian Markov random fields. If $P(\boldsymbol{v}, \boldsymbol{h})$ is a Gaussian Markov random field then the marginal distribution of $P(\boldsymbol{v})$ is just a Gaussian. On the other hand if $P(\boldsymbol{v}, \boldsymbol{h})$ is a Boltzmann distribution with energy function $E(\boldsymbol{v}, \boldsymbol{h})$ we can represent the marginal distribution $P(\boldsymbol{v})$ as an energy based model with energy function $F(\boldsymbol{v})$, that is

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

by defining the *free energy* $F(\boldsymbol{v})$ as

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

It is clear the free energy cannot be represented as a quadratic form in $\boldsymbol{v}$ so introducing hidden variables extends the class of distributions that can be modeled by Boltzmann machines. It appears that by adding sufficiently many hidden variables *any* distribution on discrete binary vectors can be accommodated [1]. For a concrete example of a distribution that can be modeled by a Boltzmann machine with hidden variables but not by a Boltzmann machine, see [3].

E. Contrastive divergence training

Boltzmann machines and related models are traditionally trained by stochastic gradient ascent rather than in batch mode as is usual in speech processing.

First let us calculate the gradient of the log likelihood with respect to the model parameters, given a training token $\boldsymbol{x}^1$.

$$\begin{aligned}
\left.\frac{\partial \ln P(\boldsymbol{x})}{\partial w_{ij}}\right|_{\boldsymbol{x}=\boldsymbol{x}^1} &= -\left.\frac{\partial F(\boldsymbol{x})}{\partial w_{ij}}\right|_{\boldsymbol{x}=\boldsymbol{x}^1} - \frac{\partial \ln Z}{\partial w_{ij}} \\
&= x_i^1 x_j^1 - \frac{\partial}{\partial w_{ij}} \ln \sum_{\boldsymbol{x}} e^{-F(\boldsymbol{x})} \\
&= x_i^1 x_j^1 - \frac{1}{\sum_{\boldsymbol{x}} e^{-F(\boldsymbol{x})}} \sum_{\boldsymbol{x}} e^{-F(\boldsymbol{x})} \frac{\partial F(\boldsymbol{x})}{\partial w_{ij}} \\
&= x_i^1 x_j^1 - \frac{1}{Z} \sum_{\boldsymbol{x}} e^{-F(\boldsymbol{x})} x_i x_j \\
&= x_i^1 x_j^1 - \sum_{\boldsymbol{x}} P(\boldsymbol{x}) x_i x_j \\
&= x_i^1 x_j^1 - \langle x_i x_j \rangle_{\text{model}}
\end{aligned}$$

If there are N training tokens $x^1, \ldots, x^N$, the gradient of the log likelihood function is

$$\frac{\partial \ln P(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N)}{\partial w_{ij}} \quad = \quad x_i^1 x_j^1 + \ldots + x_i^N x_j^N - N \langle x_i x_j \rangle_{\text{model}}$$

so that

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

which is zero at a critical point of the likelihood function, just as in the case of a Gaussian Markov random field. We will refer to the correlations $\langle x_i x_j \rangle_{\text{data}}$ and $\langle x_i x_j \rangle_{\text{model}}$ as the *data statistics* and the *model statistics*.

In practice, training is implemented sequentially and the model is updated after each training token or mini-batch is presented. Specifically, if a training token $\boldsymbol{x}^1$ is presented, the model is updated according to

$$w_{ij} \leftarrow w_{ij} + \alpha \left.\frac{\partial \ln P(\boldsymbol{x})}{\partial w_{ij}}\right|_{\boldsymbol{x}=\boldsymbol{x}^1}$$

where α is a learning rate. The model statistics $\langle x_i x_j \rangle_{\text{model}}$ are typically estimated by *contrastive divergence* (CD) which in the general formulation given by Bengio and Dellaleau [10] can be applied using any MCMC algorithm to simulate the model (not just Gibbs sampling). Starting at the given training token $\boldsymbol{x}^1$, run the Markov chain for n steps: $\boldsymbol{x}^1 \rightarrow \ldots \rightarrow \boldsymbol{x}^{n+1}$. If n is sufficiently large then $\boldsymbol{x}^{n+1}$ will be approximately distributed according to the model distribution $P(\boldsymbol{x})$ so

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

and we can approximate

$$\left.\frac{\partial \ln P(\boldsymbol{x})}{\partial w_{ij}}\right|_{\boldsymbol{x}=\boldsymbol{x}^1} \approx x_i^1 x_j^1 - x_i^{n+1} x_j^{n+1}.$$

This approximation is known as CD-n. Surprisingly CD-1 works well in practice. CD-1 acts so as to depress the energy surface at the real datum $\boldsymbol{x}^1$ and increase the energy at a nearby fictitious datum $\boldsymbol{x}^2$. It can be argued that the net effect is to fit the energy surface closely to the real data points.

Persistent contrastive divergence A set of samples (or "particles") $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N$ drawn from the model distribution is maintained and updated whenever the model is updated. (N Markov chains are run in parallel and, on every update, several steps of Gibbs sampling are performed in each chain.) A small learning rate ensures that the samples are always drawn from the model distribution even though the model is constantly being updated. The model statistics are derived by averaging over the particles:

$$\langle x_i x_j \rangle_{\text{model}} \quad = \quad \frac{1}{N} \sum_n x_i^n x_j^n$$

Persistent CD generally works better than CD-1.

Note CD-1 is inherently sequential but persistent CD could be modified to run in batch mode (or with large mini-batches) and so parallelized. Parallelization would appear to be very advantageous so there must be some good reasons why this is not done in practice. Sequential training is inherently noisy and so less likely to get stuck in a local optimum. When persistent CD is run sequentially, there is an interesting interaction between the particle dynamics and the changing energy surface. A particle which is stuck in a ravine of the energy surface will eventually be dislodged as the model parameters change. This causes the Markov chains to mix well [2].

Training with hidden units To train Boltzmann machines with hidden units, we use the EM algorithm. Given a data vector $\boldsymbol{v}$, we seek to maximize

$$\langle \ln P(\boldsymbol{x}) \rangle_{\text{data}}$$

with respect to the model parameters where $\boldsymbol{x} = (\boldsymbol{v}, \boldsymbol{h})$ and $\langle \cdot \rangle_{\text{data}}$ refers to the expectation calculated with the posterior of $\boldsymbol{h}$ given $\boldsymbol{v}$. We can ignore the entropy term in the lower bound since it is independent of the model parameters. Differentiating with respect to w_{ij},

$$\frac{\partial}{\partial w_{ij}} \langle \ln P(\boldsymbol{x}) \rangle_{\text{data}} \quad = \quad \langle x_i x_j \rangle_{\text{data}} - \langle x_i x_j \rangle_{\text{model}}$$

The fact that not all of the units are visible makes no difference to the way the model statistics are handled and the only difference is in the treatment of the data statistics. In the case of a restricted Boltzmann machine, the posterior $Q(\boldsymbol{h}|\boldsymbol{v})$ can be evaluated exactly and calculating the data statistics is straightforward. In the general case you could use either Gibbs sampling or variational Bayes to calculate the posterior expectation needed to evaluate the data statistics. Variational Bayes is used in practice as Gibbs sampling is too slow. This is Salakhutdinov's algorithm for training Boltzmann machines in a nutshell [2].

Note Variational Bayes could be computationally expensive as it may be necessary to cycle through all of the variables many times to achieve convergence. A standard trick which alleviates this problem is to initialize the variational Bayes updates for a given training token using the variational approximation calculated the last time the token was visited in the course of training.

F. Posterior and free energy calculations for restricted Boltzmann machines

We return to the restricted Boltzmann machine depicted in Fig. 1. We assume that there are I visible units, J hidden units and no hidden-to-hidden or visible-to-visible connections. Thus $\boldsymbol{h}$ is of dimension $J \times 1$, $\boldsymbol{v}$ is of dimension $I \times 1$, and we can write the energy function in the form

$$E(\boldsymbol{v}, \boldsymbol{h}) \quad = \quad -\boldsymbol{v}^T \boldsymbol{W} \boldsymbol{h}$$

where $\boldsymbol{W}$ is dimension $I \times J$. We denote the ith row of $\boldsymbol{W}$ by $\boldsymbol{W}_{i\cdot}$ and the jth column by $\boldsymbol{W}_{\cdot j}$.

Recall that the free energy is defined by

$$F(\boldsymbol{v}) \quad = \quad -\ln \sum_{\boldsymbol{h}} \exp\left(-E(\boldsymbol{v}, \boldsymbol{h})\right).$$

To evaluate this,

$$\begin{aligned}
\sum_{\boldsymbol{h}} \exp\left(-E(\boldsymbol{v},\boldsymbol{h})\right) &= \sum_{\boldsymbol{h}} \exp\left(\boldsymbol{v}^T\boldsymbol{W}\boldsymbol{h}\right) \\
&= \sum_{h_1,\ldots,h_J} \exp\left(\sum_{j=1}^{J} \boldsymbol{v}^T\boldsymbol{W}_{\cdot j}h_j\right) \\
&= \sum_{h_1,\ldots,h_J} \prod_{j=1}^{J} \exp\left(\boldsymbol{v}^T\boldsymbol{W}_{\cdot j}h_j\right) \\
&= \prod_{j=1}^{J} \sum_{h_j\in\{0,1\}} \exp\left(\boldsymbol{v}^T\boldsymbol{W}_{\cdot j}h_j\right) \\
&= \prod_{j=1}^{J} \left(1+\exp\left(\boldsymbol{v}^T\boldsymbol{W}_{\cdot j}\right)\right).
\end{aligned}$$

As for the posteriors,

$$\begin{aligned}
Q(\boldsymbol{h}|\boldsymbol{v}) &= \frac{\prod_{j=1}^{J}\exp\left(\boldsymbol{v}^T\boldsymbol{W}_{\cdot j}h_j\right)}{\prod_{j=1}^{J}\left(1+\exp\left(\boldsymbol{v}^T\boldsymbol{W}_{\cdot j}\right)\right)} \\
&= \prod_{j=1}^{J} \frac{\exp\left(\boldsymbol{v}^T\boldsymbol{W}_{\cdot j}h_j\right)}{\left(1+\exp\left(\boldsymbol{v}^T\boldsymbol{W}_{\cdot j}\right)\right)} \\
&= \prod_{j=1}^{J} Q(h_j|\boldsymbol{v})
\end{aligned}$$

where

$$Q(h_j=1|\boldsymbol{v}) = \sigma(\boldsymbol{v}^T\boldsymbol{W}_{\cdot j}).$$

Similarly

$$P(v_i=1|\boldsymbol{h}) = \sigma(\boldsymbol{W}_{i\cdot}\boldsymbol{h}).$$

G. Sparsely connected Boltzmann machines

We use the term *layer* to refer to a set of units none of which are connected another, so that the corresponding submatrix of the weight matrix is zero. We say that a Boltzmann machine is *sparsely connected* (or simply sparse) if the units can be partitioned into a small number of layers. We will denote the variables corresponding to the layers by $\boldsymbol{h}^0,\ldots,\boldsymbol{h}^L$ where $\boldsymbol{h}^0$ corresponds to the visible layer.

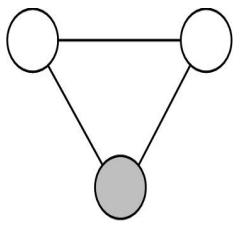

Fig. 2. Sparse Boltzmann machine with one visible and two hidden layers

The weight matrix looks like

$$\begin{pmatrix}
\mathbf{0} & & & \\
& \mathbf{0} & & \\
& & \ddots & \\
& & & \mathbf{0}
\end{pmatrix}$$

There is one $\mathbf{0}$ matrix for each layer and no restrictions on the off-diagonal blocks. In the case of a restricted Boltzmann machine, there are just 2 blocks. In the case of a deep Boltzmann machine [2], the layers are stacked so that each layer

communicates with only with its neighbours above and below. In this situation the leading super-diagonal and sub-diagonal blocks are non zero and everything else is $\mathbf{0}$:

$$\begin{pmatrix} \mathbf{0} & * & & & \\ * & \mathbf{0} & & & \\ & & \ddots & & \\ & & & \mathbf{0} & * \\ & & & * & \mathbf{0} \end{pmatrix}.$$

If $\boldsymbol{W}^{kl}$ denotes the matrix of weights on the branches joining units in layer l to those in layer k, the energy function is given by

$$E(\boldsymbol{h}) \quad = \quad -\sum_{k<l} \boldsymbol{h}^{kT} \boldsymbol{W}^{kl} \boldsymbol{h}^{l}$$

where k, l range from 0 to L.

For any pair of layers k, l, the *conditional* joint distribution $P(\boldsymbol{h}^k, \boldsymbol{h}^l | \boldsymbol{h}^{\setminus k,l})$ is a restricted Boltzmann machine (whose weight matrix is a modified version of $\boldsymbol{W}^{kl}$); the *unconditional* joint distribution is not. In fact,

$$P(\boldsymbol{h}^k, \boldsymbol{h}^l) \quad = \quad \sum_{\boldsymbol{h}^{\setminus k,l}} P(\boldsymbol{h}^k, \boldsymbol{h}^l, \boldsymbol{h}^{\setminus k,l})$$

which shows that the unconditional joint distribution can be regarded as a Boltzmann machine with hidden variables and we know that the class of Boltzmann machines with hidden variables is larger than the class of Boltzmann machines. In particular, it follows that

$$\left\langle h_i^k h_j^l \right\rangle_{\text{model}} \neq \left\langle h_i^k h_j^l \right\rangle_{\text{RBM}}$$

in general, where RBM refers to the restricted Boltzmann machine defined by the matrix $\boldsymbol{W}^{kl}$. In training a sparse BM, one might be tempted to estimate the weight matrices $\boldsymbol{W}^{kl}$ individually using RBM training but that would not be a correct procedure.

However sparse Boltzmann machines are easier to train than general Boltzmann machines because both Gibbs sampling and variational Bayes can be implemented very efficiently by cycling among layers (rather than cycling among individual units), in exactly the same way as Gibbs sampling can be carried out in RBMs by alternating between the hidden and visible layer. Note that, for each layer l, the posterior $Q(\boldsymbol{h}^l | \boldsymbol{h}^{\setminus l})$ is factorial since

$$\begin{aligned} \ln Q(\boldsymbol{h}^l | \boldsymbol{h}^{\setminus l}) \quad &= \quad \ln P(\boldsymbol{h}) - \ln P(\boldsymbol{h}^{\setminus l}) \\ &\equiv \quad \sum_{\boldsymbol{h}^{\setminus l}} E(\boldsymbol{h}^l, \boldsymbol{h}^{\setminus l}) \\ &= \quad \boldsymbol{a}^T \boldsymbol{h}^l \end{aligned}$$

where $\boldsymbol{a}$ depends on $\boldsymbol{h}^{\setminus l}$ but not on $\boldsymbol{h}^l$. Thus we can write

$$Q(\boldsymbol{h}^l | \boldsymbol{h}^{\setminus l}) \quad = \quad \prod_i Q(h_i^l | \boldsymbol{h}^{\setminus l})$$

and implement Gibbs sampling at layer l by sampling the various units independently of each other.

Likewise for variational Bayes, assuming that the variational posteriors factorizes over layers, that is

$$Q(\boldsymbol{h} | \boldsymbol{h}^0) \quad = \quad \prod_{l=1}^{L} Q(\boldsymbol{h}^l | \boldsymbol{h}^0),$$

is enough to ensure a full factorization. This is because the variational update formula for a layer l is

$$\begin{aligned} Q(\boldsymbol{h}^l | \boldsymbol{h}^0) \quad &\equiv \quad E_{\boldsymbol{h}^{\setminus l}} \left[\ln P(\boldsymbol{h}) \right] \\ &\equiv \quad E_{\boldsymbol{h}^{\setminus l}} \left[-E(\boldsymbol{h}) \right] \\ &= \quad \sum_{k<l} \boldsymbol{h}^{kT} \boldsymbol{W}^{kl} \boldsymbol{h}^l \\ &= \quad \boldsymbol{a}^T \boldsymbol{h}^l \end{aligned}$$

where $\boldsymbol{a}$ depends on $\boldsymbol{h}^{\setminus l}$ but not on $\boldsymbol{h}^l$. Thus we can implement variational Bayes at layer l by updating the variational posteriors for the various units independently.

The variational lower bound is also easily evaluated if the partition function is known. Set

$$\mu_i^l = Q(h_i^l = 1|\boldsymbol{h}^0)$$

so that the expectation of $\boldsymbol{h}^l$ calculated with the variational posterior is

$$\left\langle \boldsymbol{h}^l \right\rangle = \boldsymbol{\mu}^l$$

Note that if $k \neq l$ then $\boldsymbol{h}^k$ and $\boldsymbol{h}^l$ are independent in the variational posterior distribution so

$$\begin{aligned} \left\langle \boldsymbol{h}^l \boldsymbol{h}^{kT} \right\rangle &= \left\langle \boldsymbol{h}^l \right\rangle \left\langle \boldsymbol{h}^k \right\rangle^T \\ &= \boldsymbol{\mu}^l \boldsymbol{\mu}^{kT}. \end{aligned}$$

Thus

$$\begin{aligned} \left\langle \boldsymbol{h}^{kT} \boldsymbol{W}^{kl} \boldsymbol{h}^l \right\rangle &= \left\langle \operatorname{tr}\left(\boldsymbol{W}^{kl} \boldsymbol{h}^l \boldsymbol{h}^{kT} \right) \right\rangle \\ &= \operatorname{tr}\left(\boldsymbol{W}^{kl} \left\langle \boldsymbol{h}^l \boldsymbol{h}^{kT} \right\rangle \right) \\ &= \operatorname{tr}\left(\boldsymbol{W}^{kl} \boldsymbol{\mu}^l \boldsymbol{\mu}^{kT} \right) \\ &= \boldsymbol{\mu}^{kT} \boldsymbol{W}^{kl} \boldsymbol{\mu}^l \end{aligned}$$

and

$$\begin{aligned} \mathcal{L} &= \langle \ln P(\boldsymbol{h}) \rangle + H \\ &= -\langle E(\boldsymbol{h}) \rangle - \ln Z + H \\ &= \sum_{k<l} \boldsymbol{\mu}^{kT} \boldsymbol{W}^{kl} \boldsymbol{\mu}^l - \ln Z - \sum_l \sum_i \left(\mu_i^l \ln \mu_i^l + (1 - \mu_i^l) \ln(1 - \mu_i^l) \right) \end{aligned}$$

If the partition function is not known, this calculation gives a lower bound on the free energy (which is enough for many purposes).

H. Gaussian-Bernoulli RBMs

The visible vectors are assumed to be real valued, the hidden vectors binary valued and the energy function is given by

$$E(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{2}(\boldsymbol{v} - \boldsymbol{b})^T (\boldsymbol{v} - \boldsymbol{b}) - \boldsymbol{c}^T \boldsymbol{h} - \boldsymbol{v}^T \boldsymbol{W} \boldsymbol{h} \quad (\boldsymbol{v} \in \mathbb{R}^I, \boldsymbol{h} \in \{0,1\}^J)$$

If $\boldsymbol{h}$ is fixed this is quadratic in $\boldsymbol{v}$ so

$$\ln P(\boldsymbol{v}|\boldsymbol{h}) \equiv -\frac{1}{2} \boldsymbol{v}^T \boldsymbol{v} + \boldsymbol{v}^T (\boldsymbol{b} + \boldsymbol{W} \boldsymbol{h}).$$

Comparing this with the quadratic form which defines a Gaussian with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, namely

$$-\frac{1}{2}(\boldsymbol{v} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{v} - \boldsymbol{\mu}) = -\frac{1}{2} \boldsymbol{v}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{v} + \boldsymbol{v}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} - \frac{1}{2} \boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}$$

shows that $P(\boldsymbol{v}|\boldsymbol{h})$ is Gaussian with mean $\boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}$ and identity covariance matrix. As for $P(\boldsymbol{h}|\boldsymbol{v})$,

$$\begin{aligned} \ln P(\boldsymbol{h}|\boldsymbol{v}) &\equiv (\boldsymbol{c}^T + \boldsymbol{v}^T \boldsymbol{W}) \boldsymbol{h} \\ &= \sum_j a_j h_j \end{aligned}$$

where $a_j = c_j + \boldsymbol{v}^T \boldsymbol{W}_{\cdot j}$ so that

$$\begin{aligned} Q(\boldsymbol{h}|\boldsymbol{v}) &= \prod_j Q(h_j|\boldsymbol{v}) \\ Q(h_j = 1|\boldsymbol{v}) &= \frac{e^{a_j}}{1 + e^{a_j}} \\ &= \sigma(a_j) \\ &= \sigma(c_j + \boldsymbol{v}^T \boldsymbol{W}_{\cdot j}). \end{aligned}$$

The marginal $P(\boldsymbol{h})$, that is,

$$\int P(\boldsymbol{v}, \boldsymbol{h}) \mathrm{d}\boldsymbol{v}$$

cannot be evaluated explicitly but if we write

$$P(\boldsymbol{v}) \quad = \quad \sum_{\boldsymbol{h}} P(\boldsymbol{h}) P(\boldsymbol{v}|\boldsymbol{h})$$

we can interpret $P(\boldsymbol{v})$ as a Gaussian mixture with a prodigious number of components, where the mixture weights are evaluated in a complicated way and each component has the identity matrix as covariance matrix. Of course, the assumption of common identity covariance matrix is unreasonable unless the data has been appropriately preprocessed (e.g. by whitening it so that the mean of the data is zero and the global covariance matrix is the identity matrix.) Modeling a diagonal covariance matrix common to all mixture components but different from the identity matrix is not too difficult [11].

However the general case, where there is a full covariance matrix or a full precision matrix for each mixture component $\boldsymbol{h}$ is more difficult. The approach in [12] assumes that the precision matrix associated with a mixture component $\boldsymbol{h}$ depends in $\boldsymbol{h}$ in a linear way (analogous to the mean vector). All of the precision matrices are modeled using a set of rank one precision matrices. For a given mixture component $\boldsymbol{h}$, the corresponding precision matrix is a weighted linear combination of the rank one precision matrices where the weights are linear in $\boldsymbol{h}$. (Care is needed to ensure positive definiteness.) Some readers will recognize a parallel with a model developed by IBM called SPAM (subspace precision and mean).

Universal background model for speech In speech processing, and especially in speaker recognition, the term Universal Background Model is used to refer to a large Gaussian mixture model (with hundreds or thousands of mixture components), typically trained on hundreds of hours of data and which is intended to embrace all possible sources of variability in the speech signal. In using Boltzmann machines for speech recognition or speaker recognition, the first step is to train a Gaussian-Bernoulli RBM on acoustic feature vectors (e,g, cepstral coefficients) which as we have just explained can be thought of as Gaussian mixture model with an astronomical number of mixture components [13], [14], [15], [12], [16], [7].

The role of this UBM is to represent an acoustic feature vector $\boldsymbol{v}$ as a "mixture component index" $\boldsymbol{h}$. In its simplest incarnation the idea here is that a data vector $\boldsymbol{v}$ could be mapped to the binary vector $\boldsymbol{h}$ which maximizes $Q(\boldsymbol{h}|\boldsymbol{v})$. In practice, a mean field approximation is generally used, so that a vector of Bernoulli probabilities (that is, the probabilities $Q(h_j = 1|\boldsymbol{v})$) rather than a binary vector is produced instead [2]. This representation is suitable for subsequent processing involving variational Bayes computations with binary Boltzmann machines.

by calculating the posterior $Q(\boldsymbol{h}|\boldsymbol{v})$. (A point estimate of $\boldsymbol{h}$ could be used but the mean field approximation seems to be used in practice.) In this way, acoustic observations are represented as binary vectors for all subsequent processing. A key point is that this representation is capable of working with feature vectors of relatively high dimension (several hundred) so that information can be extracted from long windows (e.g. whole phonemes).

In [13] the acoustic feature vectors are extracted as follows. Every 10 ms, a vector of cepstral coefficients is produced in the usual way and first and second derivatives are calculated. Vectors extracted over an 11 frame window are concatenated to form an observation vector. These vectors are normalized so that the variance of each component is 1. (It is implicit in the way the energy function for the Gaussian-Bernoulli RBM is formulated that each mixture component has the identity matrix as its covariance matrix.) Interestingly, the vectors are *not* prewhitened. Because the derivatives depend linearly on the cepstral coefficients, the effect of prewhitening would be to remove the first and second derivatives, something to be avoided as these derivatives compensate for the diagonal covariance assumption in the mixture. (The effect of the diagonal assumption is to treat successive frames as being statistically independent. It is well known in speech processing that if no mechanism is available to model these dependencies then including the derivatives is helpful.)

Thus the design of the front end in [13] and many subsequent papers is rather artificial, as it is largely intended to compensate for the inadequacies of the Gaussian-Bernoulli RBM. These inadequacies are addressed in the construction of the mean covariance RBM [12]. In working with this type of RBM, the derivatives are not included in the front end and the data is prewhitened (to reduce its dimensionality).

I. Research topics in speaker recognition

Assume provisionally that we have a representation of whole utterances (of arbitrary duration) by binary vectors (of fixed length), analagous to the i-vector representation [5]. Stated in its most general form, the speaker verification problem can be formulated as one of determining whether two collections of utterances were uttered by the same speaker or by different speakers. If the collections are denoted by E (for enrollment) and T (for test), the likelihood ratio for the verification trial is

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

where each term is the joint probability of a collection of utterances which are *not* independent. For example, $P(E,T)$ is the joint probabilty of all of the utterances in the enrollment and test sets, calculated under the hypothesis that they were uttered by a single speaker. Thus the problem confronting us is to use the Boltzmann machinery to construct a joint distribution on an arbitrary set of utterances uttered by a single speaker. The construction ought to be such that the joint distribution is invariant under permutations of the utterances. We will sketch two ways of doing this here and a third later on.

Siamese twins Suppose that we build a sparse Boltzmann machine (of whatever topology) to represent the marginal distribution of individual utterances. We can construct a model for pairs of utterances by taking two copies of the original Boltzmann machine and gluing them together by adding branches joining units in one copy to units in the other. (This is a natural construction for energy based models since adding extra terms to the energy function can be thought of as imposing additional soft constraints on the data. In this case, the constraints encode the dependencies between utterances by the same speaker.) Requiring that the additional weight matrix be symmetric ensures that the distribution on pairs of utterances is symmetric. The construction extends straightforwardly to handling N-tuples of utterances (replicate the glue for every pair of utterances). The twin model is a sparse Boltzmann machine if the singleton model is (but it is not a deep Boltzmann machine in the sense of [2]). This model can be thought of as learning an analogue of the cosine distance metric [5].

So the numerator and the denominator in the verification likelihood ratio can be approximated by the variational free energy calculation for sparse Boltzmann machines. Note that the partition functions are not needed (although it would be desireable to have them to see how well calibrated the likelihood ratios are). In practice verification decisions are made by comparing the likelihood ratio with a decision threshold whose value could be determined in theory from the parameters of a NIST-like detection cost function but which is usually determined empirically in practice. As long as the determination is done empirically, the partition functions can be absorbed into the decision threshold and their values need not be known.

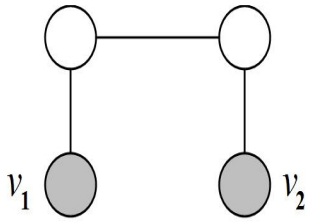

Fig. 3. Example of the Siamese twin construction

Speaker factors and channel factors An alternative approach to constructing a joint distribution on pairs of utterances by a speaker would be to take any distribution on pairs of visible vectors $(\boldsymbol{v}_1, \boldsymbol{v}_2)$ and modify it so as to obtain a symmetric distribution. For example we could start with a RBM and tie the weight matrices as in Fig. 4. The units in the hidden layer would account for both utterances in the pair so that they would play a role analagous to speaker factors in Joint Factor Analysis or Probabilistic Linear Discriminant Analysis [4], [6]. We will refer to units like this as *tied units*. This model can obviously be extended to handle multiple utterances in a permutation-invariant way but it would not be very powerful and something analagous to channel factors would seem to be needed, as in Fig. 5.

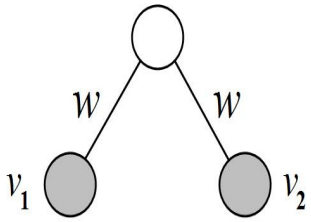

Fig. 4. Tied hidden variables

The model in Fig 5 is an RBM (although it may not look like one). As such the free energies needed to evaluate the verification likelihood ratio (up to the partition functions) can be evaluated exactly. The computational burden of evaluating the free energy of a set of N utterances is proportional to N rather than to N^2 as in the case of the twin model. Obviously it would be possible to design more general sparse Boltzmann machines containing both tied and untied hidden layers.

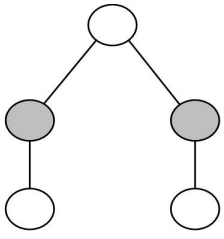

Fig. 5. An RBM with tied hidden variables (on top) and untied hidden variables (on the bottom)

i-vectors So far we have assumed that a representation of utterances by binary vectors of fixed length is somehow available. For the purposes of algorithm developement, we are currently using a Gaussian-Bernoulli RBM to binarize i-vectors [5]. In the long run we will want to explore building new generative models whose hidden variables are binary valued and can play the role of i-vectors. It would be natural to use the binary representation of acoustic observations (derived from the Universal Background Model) as the input to such a model. We would also require hidden variables tied across all of the acoustic observations in an utterance but differing from one utterance to another (analgous to the way speaker factors are common to all recordings of a speaker but differ from one speaker to another). These hidden variables taken together would constitute the binary representation of a given utterance.

J. Incremental training of sparse Boltzmann machines

It is apparent that it will be necessary to build sparse Boltzmann machines with complex toplogies for speaker recognition. In this section we outline a way of training a sparse Boltzmann machine by adding layers incrementally. The variational lower bound is guaranteed to increase each time a layer is added. This procedure could be used to train deep Boltzmann machines (although Salakhutdinov does not use it). Unlike Hinton's incremental construction of deep Belief nets which we will come to later, no constraints need to be imposed on the numbers of units in the layers in order to ensure increases in the variational lower bound.

Suppose we have already constructed a sparse BM with layers $0, \ldots, l$ where layer 0 is visible and we want to add a layer $l+1$ specified by a weight matrices $\boldsymbol{W}^{k,l+1}$ $(k = 1, \ldots, l)$. We can consider layer $l+1$ as being already present with $\boldsymbol{W}^{k,l+1} = \boldsymbol{0}$ so it is natural to initialize $\boldsymbol{W}^{k,l+1}$ as $\boldsymbol{0}$ and assign random values to μ_i^{l+1}. Recall that the conditional joint distribution $P(\boldsymbol{h}^k, \boldsymbol{h}^{l+1} | \boldsymbol{h}^{\setminus k,l+1})$ is an RBM but the unconditional joint distribution $P(\boldsymbol{h}^k, \boldsymbol{h}^{l+1})$ is not. So, even though the pair consisting of layer k and layer $l+1$ could be construed as an RBM with weight matrix $\boldsymbol{W}^{k,l+1}$, this weight matrix cannot be estimated by the standard RBM training algorithm.

For each training token $\boldsymbol{h}^0$, we assume that the variational posterior

$$Q(\boldsymbol{h}^1, \ldots, \boldsymbol{h}^l | \boldsymbol{h}^0) \quad = \quad Q(\boldsymbol{h}^1 | \boldsymbol{h}^0) \ldots Q(\boldsymbol{h}^l | \boldsymbol{h}^0)$$

has been calculated. By sampling from the posteriors $Q(\boldsymbol{h}^l | \boldsymbol{h}^0)$ for each $\boldsymbol{h}^0$ we obtain a set of "visible" vectors which serve as the input to layer $l+1$. (This collection of visible vectors is referred to as the "aggregated" training set. This term formalizes the idea of using hidden variables in one model as features for training the next model in the hierarchy.)

Fix a branch joining a unit i at layer k to a unit j at layer $l+1$. For each training vector $\boldsymbol{h}^0$, each sample $\boldsymbol{h}^k \sim Q(\boldsymbol{h}^k | \boldsymbol{h}^0)$ contributes

$$h_i^k \left\langle h_j^{l+1} \right\rangle_{\text{data}}$$

to the data dependent statistics. Here $\boldsymbol{h}^{l+1} \sim Q(\boldsymbol{h}^{l+1} | \boldsymbol{h}^k)$ which is the posterior calculated using the matrix $\boldsymbol{W}^{k,l+1}$, just as in a restricted Boltzmann machine. Averaging over all samples drawn from $Q(\boldsymbol{h}^k | \boldsymbol{h}^0)$ gives

$$\sum_{\boldsymbol{h}^k} Q(\boldsymbol{h}^k | \boldsymbol{h}^0) h_i^k \left\langle h_j^{l+1} \right\rangle_{\text{data}} \quad = \quad \left\langle h_i^k \right\rangle_{\text{data}} \left\langle h_j^{l+1} \right\rangle_{\text{data}} .$$

The tricky part is the model dependent statistics. Since

$$\left\langle h_i^k h_j^{l+1} \right\rangle_{\text{model}} \neq \left\langle h_i^k h_j^{l+1} \right\rangle_{\text{RBM}} ,$$

it is not enough to run a Gibbs sampler which alternates between layers k and $l+1$ to calculate the model dependent statistics. The correct Gibbs sampling procedure involves updating the *all* of the vectors $\boldsymbol{h}^1, \ldots, \boldsymbol{h}^{l+1}$ at each step. (This is necessary even if only one weight matrix is added.)

This way of estimating the weight matrices $\boldsymbol{W}^{k,l+1}$ is guaranteed to increase the variational lower bound since it is just a special cases of the general training algorithm (namely the case where the variational posteriors at layers $1, \ldots, l$ are held fixed and only the posterior at layer $l+1$ and the matrices $\boldsymbol{W}^{k,l+1}$ are updated.) This observation applies in the case where all of the matrices $\{\boldsymbol{W}^{k,l+1} : k \neq l+1\}$ are updated simultaneously. (It is not necessary to update these matrices one by one which would increase the computational burden entail in calculating the model statistics.)

After layer $l+1$ has been added the lower bound can be further increased by updating the posteriors at all layers. Of course it can be increased still further by applying the general training algorithm to update all of the weight matrices and posteriors. The guarantee on the variational lower bound holds irrespective of how many units are added at layer $l+1$ and it also holds if these units are tied.

K. Third order RBMs

In a third order Boltzmann machine, units are partitioned into 3 sets I, J and K. For $i \in I$, denote the corresponding binary variable by x_i, for $j \in J$ by y_j and for $k \in K$ by z_k. The energy function is

$$E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \quad = \quad -\sum_{ijk} w_{ijk} x_i y_j z_k$$

(first and second order terms are usually included as well). In some applications $\boldsymbol{x}$ and $\boldsymbol{y}$ are hidden and $\boldsymbol{z}$ is visible; in others $\boldsymbol{x}$ is visible and $\boldsymbol{y}$ and $\boldsymbol{z}$ are hidden. To fix ideas, we will consider the latter case only.

As with sparse Boltzmann machines, Gibbs sampling can be implemented very efficiently. If, for example, $\boldsymbol{z}$ is given then the conditional joint distribution $Q(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x})$ is a restricted Boltzmann distribution. If in addition $\boldsymbol{y}$ is given, then $Q(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y})$ is factorial (since this is true of any restricted Boltzmann distribution). Likewise $Q(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{z})$ and $Q(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{z})$ are factorial. So Gibbs sampling can be implemented by cyclically updating $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$.

Variational Bayes can also be implemented very efficiently. Assuming a factorization of the form

$$Q(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x}) \quad = \quad Q(\boldsymbol{y}|\boldsymbol{x}) Q(\boldsymbol{z}|\boldsymbol{x})$$

implies a complete factorization

$$Q(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x}) \quad = \quad \prod_j Q(y_j|\boldsymbol{x}) \prod_k Q(z_k|\boldsymbol{x}).$$

For example, the variational update for $Q(\boldsymbol{y}|\boldsymbol{x})$ is

$$\begin{aligned} \ln Q(\boldsymbol{y}|\boldsymbol{x}) \quad &\equiv \quad E_{Q(\boldsymbol{z}|\boldsymbol{x})} \left[-E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \right] \\ &= \quad \sum_{i,j,k} w_{ijk} x_i y_j \left\langle z_k \right\rangle \\ &= \quad \sum_j a_j y_j \end{aligned}$$

where $\langle z_k \rangle$ indicates the expectation taken with respect to $Q(\boldsymbol{z}|\boldsymbol{x})$ and

$$a_j \quad = \quad \sum_{i,k} w_{ijk} x_i \left\langle z_k \right\rangle .$$

This shows that the y_j's are independent in the variational posterior.

The gradient of the log likelihood function can be calculated straightforwardly. For example,

$$\begin{aligned} \frac{\partial \left\langle \ln P(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \right\rangle}{\partial w_{ijk}} \quad &= \quad \left\langle x_i y_j z_k \right\rangle_{\text{data}} - \left\langle x_i y_j z_k \right\rangle_{\text{model}} \\ &= \quad x_i \left\langle y_j \right\rangle_{\text{data}} \left\langle z_k \right\rangle_{\text{data}} - \left\langle x_i y_j z_k \right\rangle_{\text{model}} \end{aligned}$$

where we have used the fact that y_j and z_k are independent in the variational posterior. Since Gibbs sampling is very efficient there is no difficulty implementing persistent contrastive divergence. Thus in principle, training a third order Boltzmann machine is straightforward. Of course this statement ignores the fact that the number of parameters w_{ijk} may be unmanageably large.

In [17], this third order tensor is assumed to be expressed as a sum of rank 1 tensors:

$$w_{ijk} \quad = \quad \sum_{f=1}^{F} w_{if}^{\boldsymbol{x}} w_{jf}^{\boldsymbol{y}} w_{kf}^{\boldsymbol{z}}.$$

Only minor modifications to the training algorithm are required. Ignoring the entropy term, the EM auxiliary function is

$$\left\langle \ln P(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \right\rangle_{\text{data}} \quad = \quad \sum_{f=1}^{F} w_{if}^{\boldsymbol{x}} w_{jf}^{\boldsymbol{y}} w_{kf}^{\boldsymbol{z}} x_i \left\langle y_j \right\rangle_{\text{data}} \left\langle z_k \right\rangle_{\text{data}} - \ln Z$$

and the derivative with respect to, say, $w_{if}^{\boldsymbol{x}}$ is given by

$$\begin{aligned} \frac{\partial}{\partial w_{if}^{\boldsymbol{x}}} \left\langle \ln P(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \right\rangle_{\text{data}} \quad &= \quad \sum_{jk} w_{jf}^{\boldsymbol{y}} w_{kf}^{\boldsymbol{z}} x_i \left\langle y_j \right\rangle_{\text{data}} \left\langle z_k \right\rangle_{\text{data}} - \frac{1}{Z} \sum_{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}} \exp\left(-E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\right) \sum_{jk} w_{jf}^{\boldsymbol{y}} w_{kf}^{\boldsymbol{z}} x_i y_j z_k \\ &= \quad \sum_{jk} w_{jf}^{\boldsymbol{y}} w_{kf}^{\boldsymbol{z}} x_i \left\langle y_j \right\rangle_{\text{data}} \left\langle z_k \right\rangle_{\text{data}} - \sum_{jk} w_{jf}^{\boldsymbol{y}} w_{kf}^{\boldsymbol{z}} \left\langle x_i y_j z_k \right\rangle_{\text{model}} \\ &= \quad \sum_{jk} w_{jf}^{\boldsymbol{y}} w_{kf}^{\boldsymbol{z}} \left(x_i \left\langle y_j \right\rangle_{\text{data}} \left\langle z_k \right\rangle_{\text{data}} - \left\langle x_i y_j z_k \right\rangle_{\text{model}} \right) . \end{aligned}$$

Research directions in speaker recognition The possibilities for third order Boltzmann machines seem to be most interesting in the case where there are two sets of hidden varibles and one set of visible variables, rather than one set of hidden variables and two sets of visible variables. Yet most of the literature on third order RBMs deals with the latter configuration. In [17], references are cited to support the contention that two sets of hidden variables may be problematic to work with because it may be hard for the model to know which effects should be attributable to which set. However the evidence from Joint Factor Analysis suggests that it may be possible to make such an attribution reliably if one set is tied (like speaker factors) and the other untied (like channel factors).

Thus it may be possible to use a third order Boltzmann machine to implement a version of Probabilistic Linear Discriminant Analysis. Likewise it may also be possible to use this type of model with a set of hidden variables tied across all of the acoustic observations in an utterance and so characterize whole utterances by binary valued vectors analogous to i-vectors.

IV. Sigmoid Belief Networks

A belief network is a directed graphical model which implements probability calculations similar to Boltzmann machines. The graph typically has the same topology as a deep Boltzmann machine and only the leaf nodes are visible, with the crucial difference that the arrows in the graph are directed rather than undirected. (It is usual to refer to units as nodes in this context.) In the case of a deep toplogy, sigmoid belief networks are referred to as deep belief networks (DBNs — not to be confused with DBMs). The directed arrows inidicate causal relationships and the binary variables at the higher levels of the network can be thought of as high level features.

For each node i in the graph, let $Pa(i)$ denote the set of parents of i. The probability distribution can be expressed in the form

$$P(\boldsymbol{x}) \quad = \quad \prod_i P(x_i|\boldsymbol{x}_{Pa(i)}).$$

Implicit in this formula is the assumption that the joint distribution on root nodes (i.e. nodes without parents) is factorial. However, this condition on the root nodes is *not* satisfied in Hinton's construction of DBNs (described later).

For each node i, the factor $P(x_i|\boldsymbol{x}_{Pa(i)})$ is evaluated as

$$P(x_i = 1|\boldsymbol{x}_{Pa(i)}) \quad = \quad \sigma(u_i)$$

where the activation or "local field" u_i is defined by

$$u_i \quad = \quad \sum_{p \in Pa(i)} w_{ip} x_p.$$

Here, p ranges over the parents of i and w_{ip} is the weight associated with the directed arc $p \to i$. Note that in a deep belief network, the local field depends on the layer above the given node i, rather than on the layer above and the layer below as in a deep Boltzmann machine.

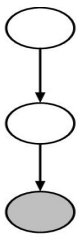

Fig. 6. Deep sigmoid belief network

For a sigmoid belief network all of whose nodes are visible, all calculations are easy. (Of course the directed acyclic graph cannot have a deep architecture in this case.) It is easy to draw samples from the prior distribution, there is no difficulty with the partition function and training is straightforward. However, if there are hidden variables, posterior calculations are intractable and training seems to be challenging. We survey these models briefly because they provide insight into how deep feed forward neural nets work.

The difficulty with posterior calculations arises because of junctions of the form $\circ \to \circ \leftarrow \circ$. In the continuous case, we could have for example $Z = X + Y + \epsilon$. Suppose $P(X, Y) = P(X)P(Y)$. It will not be the case that $Q(X, Y|Z) = Q(X|Z)Q(Y|Z)$. (Knowing both Z and X tells you something about Y.) In the binary case, could have

- $X = 1$ iff there is an earthquake
- $Y = 1$ iff a truck hits the house
- $Z = 1$ iff the house moves

If the house moves and the radio says that an earthquake has taken place, then the probability that a truck has hit the house decreases. The possibility that the truck hits the house has been "explained away" by the earthquake.

A. Training

We assume to begin with that there are no hidden nodes. and derive the gradient of the log likelihood function. Note that

$$\frac{\partial}{\partial w_{ip}} \ln P(\boldsymbol{x}) \quad = \quad \frac{\partial}{\partial w_{ip}} \ln P(x_i|\boldsymbol{x}_{Pa(i)}).$$

Claim

$$\frac{\partial}{\partial w_{ip}} \ln P(x_i|\boldsymbol{x}_{Pa(i)}) \quad = \quad x_p x_i - x_p \sigma(u_i)$$

For, in the case $x_i = 1$,

$$\begin{aligned} \frac{\partial}{\partial w_{ip}} \ln P(x_i = 1|\boldsymbol{x}_{Pa(i)}) &= \frac{\partial \ln \sigma(u_i)}{\partial w_{ip}} \\ &= (1 - \sigma(u_i))x_p \\ &= x_i x_p - \sigma(u_i)x_p \text{ if } x_i = 1 \end{aligned}$$

and in the case $x_i = 0$,

$$\begin{aligned} \frac{\partial}{\partial w_{ip}} \ln P(x_i = 0|\boldsymbol{x}_{Pa(i)}) &= -\frac{1}{P(x_i = 0|\boldsymbol{x}_{Pa(i)})} \frac{\partial \sigma(u_i)}{\partial w_{ip}} \\ &= -\frac{\sigma'(u_i)}{1 - \sigma(u_i)} w_{ip} \\ &= -\sigma(u_i)x_p \\ &= x_i x_p - \sigma(u_i)x_p \text{ if } x_i = 0. \end{aligned}$$

Note that there is an analogy here with CD-1. Since $\sigma(u_i)$ can be interpreted as $P(x_i = 1|\boldsymbol{x}_{Pa(i)})$, we have the following stochastic approximation

$$\frac{\partial}{\partial w_{ip}} \left\langle \ln P(x_i|\boldsymbol{x}_{Pa(i)}) \right\rangle_{\text{data}} \quad \approx \quad x_p x_i - x_p \tilde{x}_i$$

where $\tilde{\boldsymbol{x}}$ is a stochastic reconstruction of $\boldsymbol{x}$ such that $\tilde{x}_i$ takes the value 1 with probability $\sigma(u_i)$.

B. Problem with Gibbs sampling

The stochastic gradient approximation can easily be extended in principle to the case there are hidden variables using Gibbs sampling. (Given a training token $\boldsymbol{v}$, draw a sample $\boldsymbol{x}$ from the posterior $Q(\boldsymbol{x}|\boldsymbol{v})$ and use it to build a 1-step reconstruction $\tilde{\boldsymbol{x}}$.) But there is a problem in practice, namely that Gibbs sampling for sigmoid belief networks is computationally expensive, even if the directed acyclic graph has a deep architecture.

Consider variables at 3 levels $\boldsymbol{h}^{l+1}, \boldsymbol{h}^l$ and $\boldsymbol{h}^{l-1}$. The distribution $P(\boldsymbol{h}^l|\boldsymbol{h}^{l+1})$ is factorial but the distribution $Q(\boldsymbol{h}^l|\boldsymbol{h}^{l+1}, \boldsymbol{h}^{l-1})$ is not because of explaining away. This means that the nodes at level l interfere with each other during Gibbs sampling. On the other hand in the case of a deep Boltzmann machine the distribution $Q(\boldsymbol{h}^l|\boldsymbol{h}^{l+1}, \boldsymbol{h}^{l-1})$ is factorial and there is no interference.

To calculate $Q(x_i|\boldsymbol{x}_{\setminus i})$ in a general directed acyclic graph, note that the factors in the prior $P(\boldsymbol{x})$ which involve x_i are $P(x_i|\boldsymbol{x}_{Pa(i)})$ and $P(x_j|x_{Pa(j)})$ where $i \in Pa(j)$ i.e. $j \in C(i)$. Thus

$$Q(x_i|\boldsymbol{x}_{\setminus i}) \propto P(x_i|\boldsymbol{x}_{Pa(i)}) \prod_{j \in C(i)} P(x_j|\boldsymbol{x}_{Pa(j)}).$$

Other than i itself, the variables appearing in on the right hand side are of three types: the parents of i, the children of i and the parents of the children of i. This collection of nodes is known as the *Markov blanket* of i. Denoting the Markov blanket by $B(i)$, $Q(x_i|\boldsymbol{x}_{\setminus i}) = Q(x_i|\boldsymbol{x}_{B(i)})$ [8].

In the case of a deep sigmoid belief net, the Markov blanket consists of the layer containing i, as well as the layer above and the layer below.

C. Variational Bayes

Likewise variational Bayes runs into trouble for deep belief networks.

- As in the case of Gibbs sampling, a full factorization is needed. (The fixed point equations for each node involve the variational parameters for all of the nodes in its Markov blanket, not just the nodes in the layer above and the layer below.)
- Even with a full factorization, the variational lower bound cannot be evaluated exactly and a cruder lower bound has to be used.
- Since the EM auxiliary function is essentially the same as the variational lower bound, the EM algorithm cannot be implemented exactly either.

See Section 11.12 of Haykin [9] for more information.

D. The wake sleep algorithm

Hinton has proposed a way of training deep sigmoid belief networks which attempts to circumvent the intractability of the posterior calculation. Recall that two ways of writing the EM auxiliary function are

$$\begin{aligned} &\langle \ln P(\boldsymbol{v}, \boldsymbol{h}) \rangle + H(Q(\boldsymbol{h}|\boldsymbol{v})) \\ \ln P(\boldsymbol{v}) - &\; D\left(Q(\boldsymbol{h}|\boldsymbol{v}) \| P(\boldsymbol{h}|\boldsymbol{v})\right). \end{aligned}$$

To improve the fit of the model to a given data vector $\boldsymbol{v}$, we can

1) Hold the posterior $Q(\boldsymbol{h}|\boldsymbol{v})$ fixed and increase $\langle \ln P(\boldsymbol{x}) \rangle$ (where $\boldsymbol{x}$ is a short-hand for $(\boldsymbol{h}, \boldsymbol{v})$)
2) Hold the model $P(\boldsymbol{v})$ fixed and decrease the divergence between the true posterior $P(\boldsymbol{h}|\boldsymbol{v})$ and the approximate posterior $Q(\boldsymbol{h}|\boldsymbol{v})$

and alternate between the two.

For 1), we need to evaluate the gradient with respect to the weight matrix of $\langle \ln P(\boldsymbol{x}) \rangle$. If we can sample from $\boldsymbol{x} \sim Q(\boldsymbol{x}|\boldsymbol{v})$, we can use the stochastic approximation

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

where $\tilde{\boldsymbol{x}}$ is a stochastic reconstruction of $\boldsymbol{x}$. (This is the wake phase: the model is paying attention to what is going on in the real world, namely the data vector $\boldsymbol{v}$.)

The problem of sampling from the posterior is addressed by 2). Recall that sampling from the *prior* distribution of a deep belief net is easy using the decomposition

$$P(\boldsymbol{x}) \quad = \quad \prod_i P(x_i | \boldsymbol{x}_{Pa(i)})$$

and that training a deep belief net is easy if all of the needs are visible. For the sleep phase, assume that the posterior $Q(\boldsymbol{x}|\boldsymbol{v})$ can be *approximated* as

$$Q(\boldsymbol{x}|\boldsymbol{v}) \quad = \quad \prod_i Q(x_i | \boldsymbol{x}_{C(i)}),$$

that is, by a sigmoid belief net with the arrows reversed. Thus the visible vectors form the input layer and, for each branch $p \to i$ in the original graph with weight w_{ip} there is a branch $i \to p$ in the dual graph with a different weight w'_{pi} associated with it. The "visible" data used to train the dual deep belief network is obtained by sampling from the model, $P(\boldsymbol{x})$, rather than from the real world. (This is the sleep phase: the model is dreaming.) The weights w_{ip} are referred to as the *generative* weights and the weights w'_{pi} as the *recognition* weights; it is only the recognition weights that are used to evaluate the approximate posterior.

Suppose that we are given a training set $\{\boldsymbol{v}^n\}$, then the quantity that is being optimized in the wake phase is just

$$\sum_n \sum_{\boldsymbol{h}} Q(\boldsymbol{h}|\boldsymbol{v}^n) \ln P(\boldsymbol{h}, \boldsymbol{v}^n)$$

which is exactly what is required by the EM algorithm. In the sleep phase, the model $P(\boldsymbol{v}, \boldsymbol{h})$ is fixed and a *synthetic* training set $\{\boldsymbol{v}^n, \boldsymbol{h}^n\}$ is generated by sampling from the model. 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)

$$\begin{aligned} \sum_n \ln Q(\boldsymbol{h}^n|\boldsymbol{v}^n) \quad &= \quad \sum_n \sum_{\boldsymbol{h}} P(\boldsymbol{h}|\boldsymbol{v}^n) \ln Q(\boldsymbol{h}|\boldsymbol{v}^n) \\ &\equiv \quad -\sum_n D\left(P(\boldsymbol{h}|\boldsymbol{v}^n) \| Q(\boldsymbol{h}|\boldsymbol{v}^n)\right) \end{aligned}$$

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 $D\left(P(\boldsymbol{h}|\boldsymbol{v}) \| Q(\boldsymbol{h}|\boldsymbol{v})\right)$, rather than $D\left(Q(\boldsymbol{h}|\boldsymbol{v}) \| P(\boldsymbol{h}|\boldsymbol{v})\right)$ as required by the EM algorithm.

Unfortunately this elegant idea does not seem to work very well in practice [18]. Although they cannot be implemented on very large scales, Gibbs sampling and variational Bayes perform better than wake-sleep training (you get what you pay for).

Mean field approximation

The adavantage of the wake-sleep algorithm is computational efficiency, particularly if the mean field approximation is used to calculate approximate posteriors. Instead of propagating arrays of random binary vectors from the data layer up to the higher layers, the mean field approximation propagates arrays of Bernoulli probablities. Sampling from the approximate posterior is then just a matter of sampling independently from these Bernoulli distributions.

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

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

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

$$\mu_j^{l+1} = \sigma\left(\sum_i w'_{ji}\mu_i^l\right).$$

Note that the mean field computation operates in the same way as a feed forward neural net, although the top layer of the neural net (the one that makes the recognition decisions) is missing.

To the extent that it is possible to train DBNs and carry out inference (that is, to do posterior calculations) with them, DBNs will be useful for constructing deep neural nets. Regarded as a generative model and trained in an unsupervised way, the variables at the higher levels of a DBN provide causal explanations of the data; inferring the values of these variables for given a data vector (using the recognition weights) can be regarded as extracting high level features from the data. Ideally, such high level features should prove more useful than the raw data vectors for recognizing patterns.

V. Feed forward neural nets

A feed forward neural net consists of

- A set of layers which implement the mean field posterior calculation for a DBN
- A softmax layer for making recognition decisions

The *activation* u_i for a node i is $\boldsymbol{w}_i^T\boldsymbol{x}$ where $\boldsymbol{x}$ 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, σ is replaced by the softmax function. The probability assigned to the ith 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.

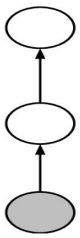

Fig. 7. Feed forward neural network

A. Backpropagation

Recall for any node i the activation u_i is given by $u_i = \sum_p w_{ip}x_p$ where p ranges over the parents of i and the value taken by the node $x_i = \sigma(u_i)$. (When we talk about parents in this context we are assuming that the arrows are directed as in Fig. 7.) Suppose that u_i and x_i have been calculated for all nodes i in a forward pass. We explain how to calculate the derivatives

$$\frac{\partial\mathcal{E}}{\partial w_{ip}}.$$

It is enough to calculate $\partial\mathcal{E}/\partial u_i$ since

$$\begin{aligned}\frac{\partial\mathcal{E}}{\partial w_{ip}} &= \frac{\partial\mathcal{E}}{\partial u_i}\frac{\partial u_i}{\partial w_{ip}} \\ &= \frac{\partial\mathcal{E}}{\partial u_i}x_p.\end{aligned}$$

To begin with, let i be node other than a label node, so that $x_i = \sigma(u_i)$.

$$\begin{aligned}\frac{\partial\mathcal{E}}{\partial u_i} &= \frac{\partial\mathcal{E}}{\partial x_i}\frac{\partial x_i}{\partial u_i} \\ &= \frac{\partial\mathcal{E}}{\partial x_i}\sigma'(u_i)\end{aligned}$$

and, since $u_j = \sum_i w_{ji} x_i$,

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial x_i} &= \sum_j \frac{\partial \mathcal{E}}{\partial u_j}\frac{\partial u_j}{\partial x_i} \\ &= \sum_j w_{ji}\frac{\partial \mathcal{E}}{\partial u_j}\end{aligned}$$

so

$$\frac{\partial \mathcal{E}}{\partial u_i} = \sigma'(u_i)\sum_j w_{ji}\frac{\partial \mathcal{E}}{\partial u_j}.$$

Since w_{ji} is the weight associated with the branch $i \to j$, $w_{ji} = 0$ and j contributes nothing to the sum on the right hand side unless j is in the layer above i. Thus the calculation of the derivatives propagates in the backwards direction (from the labels back to the data).

All that remains is to explain how to initialize the recursion. If i is a node in the softmax layer,

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial u_i} &= -\sum_k t_k \frac{\partial}{\partial u_i}\left(u_k - \ln\sum_j e^{u_j}\right) \\ &= -\sum_k t_k \left(\delta_{ik} - \frac{1}{\sum_j e^{u_j}}\sum_j e^{u_j}\delta_{ij}\right) \\ &= -\sum_k t_k \left(\delta_{ik} - p_i\right) \\ &= -(t_i - p_i).\end{aligned}$$

B. Speech recognition with a neural net

Acoustic observation vectors represented as binary vectors or arrays of Bernoulli probabilities using a Gaussian-Bernoulli RBM Universal Background Model can be used as input to a train a feed forward neural network for speech recognition.

Although in previous applications of neural nets to speech recognition the aim was to classify acoustic observation vectors as phonemes (about 40 in all), Microsoft (most recently [7], and the references cited there) uses a different inventory consisting of a few thousand classes as in conventional speech recognizers. The idea is that the acoustic manifestation of a phoneme depends on the phonemes which precede and follow it through co-articulation effects. A naive approach to dealing with the co-articulation problem would be to model each phoneme-in-context with its own hidden Markov model (HMM). But, even if only the immediate left and right context of a phoneme is taken into account, the number of HMM output distributions to be modeled would be prohibitively large. The standard way of dealing with this is to use a decision tree to cluster the number of distributions down to a manageable number (typically a few thousand) of "triphone states" or "senones". Thus the role of the neural network then is to discriminate between a few thousand triphone states. Every acoustic observation in the training data is labelled as belonging to exactly one of these states by a forced Viterbi alignment. So training a feedforward neural network to discriminate between the states is straightforward, to the extent that backpropagation can be relied on to work properly.

In order to use it in a Viterbi search in a speech recognizer, the posterior distribution $P(\boldsymbol{c}|\boldsymbol{v})$ produced by the neural should be converted to generative form. This is trivial. By Bayes rule,

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

The term $P(\boldsymbol{c})$ can be estimated by counting state frequencies in the training data and the term $P(\boldsymbol{v})$ can be ignored (since it is common to all classes). The distributions $P(\boldsymbol{v}|\boldsymbol{c})$ serve as the HMM output distributions in an otherwise conventional, single pass speech recognition architecture.

Compared with a conventional, single pass discriminatively trained approach (MMI-HMM) the neural net approach achieves 30% error rate reductions on hard benchmarks (the Switchboard and Fisher conversational telephone speech corpora). Of course state of the art conventional systems are not single pass systems. (State of the art systems are adaptive, using provisional transcriptions of the data to be recognized to adjust the recognizer's parameters. Multiple interations of adaptation may be performed.) The performance of the single pass neural net system is about the same as that of a complex adaptive state-of-the-art system.

At this writing, no work seems to have been done on the problem of how a neural net speech recognition system might be made adaptive. In a standard recognition system, most adaptation methods use a provisional transcription of the data (that is, a labeling of acoustic observations by triphone states) to adjust the generative output distributions $P(\boldsymbol{v}|\boldsymbol{c})$ so as to produce a

better match with the data. It is hard to guage how difficult adapting a discriminative neural net to a provisional transcription might be.

It came as a surprise (not least to the Microsoft authors) that very good results could be achieved with unaided back-propagation, apparently starting from random initializations, Conventional wisdom has it that deep neural networks cannot be trained in this way in practice. (The authors suggest that discriminating between several thousand triphone states rather than forty phonemes may explain their success. Even a single hidden layer produces very good results in this situation [14]. They also point to the benefits of using a Gaussian-Bernoulli RBM which models feature vectors defined by concatenating cepstral features extracted from blocks of 11 frames.) In fact, the authors' starting point for training neural nets was the deep belief net construction to be described in the next section; only after the fact did they do due dilligence. They found that many of the excellent results they had originally obtained with the deep belief net construction could also be obtained without it. The jury is still out on this.

VI. Hinton's deep belief networks

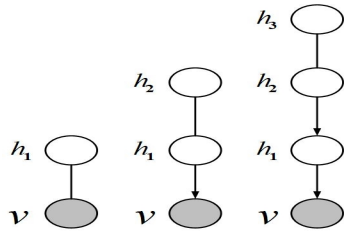

Fig. 8. Unfolding an RBM into a DBN

To motivate Hinton's construction of deep Belief networks, observe that the models in Fig. 8 are all equivalent. For the second graph, the pair $(\boldsymbol{h}^1, \boldsymbol{h}^2)$ is a mirror image of the pair $(\boldsymbol{v}, \boldsymbol{h}^1)$ in the first graph. For the third graph, the pair $(\boldsymbol{h}^2, \boldsymbol{h}^3)$ is a mirror image of the pair $(\boldsymbol{h}^1, \boldsymbol{h}^2)$ in the second graph.

Comparing the first and second graphs, the models are the same because $P(\boldsymbol{v}|\boldsymbol{h}^1)$ is the same in both cases and, by the mirror image condition, $P(\boldsymbol{h}^1)$ is the same in both cases:

$$\begin{aligned} P(\boldsymbol{h}^1) &= \sum_{\boldsymbol{v}} P(\boldsymbol{v})P(\boldsymbol{h}^1|\boldsymbol{v}) \\ &= \sum_{\boldsymbol{h}^2} P(\boldsymbol{h}^2)P(\boldsymbol{h}^1|\boldsymbol{h}^2) \end{aligned}$$

so $P(\boldsymbol{v})$ is the same in both cases. Comparing the second and third graphs, the argument we have just given shows that $P(\boldsymbol{h}^1)$ is the same in both cases; since $P(\boldsymbol{v}|\boldsymbol{h}^1)$ is the same in both cases, $P(\boldsymbol{v})$ is the same in both cases. (This argument can be extended indefinitely, so that a restricted Boltzmann machine can be interpreted as an infinitely deep sigmoid belief network.)

For these arguments to be valid, the weight matrix for the original RBM has to be copied to all of the levels, but we obtain a more flexible class of models by allowing different weight matrices at different levels. It has become standard usage to refer to models of this type as deep belief networks but this departs from traditional usage since the prior on the hidden variables at the top level (e.g. $\boldsymbol{h}^3$) is not factorial.

A. Hinton's construction

Given training data $\{\boldsymbol{v}^n\}$, Hinton and colleagues [19] show how addding successive levels to a DBN in the way just described and calculating an approximate posterior on the hidden variables guarantees the variational lower bound increases everytime a level is added. In the case of a model with l levels, the joint distribution of the hidden and visible variables and the approximate posterior of the hidden variables are given by

$$\begin{aligned} P(\boldsymbol{h}^1, \ldots, \boldsymbol{h}^l, \boldsymbol{v}) &= P(\boldsymbol{v}|\boldsymbol{h}^1)P(\boldsymbol{h}^1|\boldsymbol{h}^2)\ldots P(\boldsymbol{h}^{l-2}|\boldsymbol{h}^{l-1})P(\boldsymbol{h}^{l-1}, \boldsymbol{h}^l) \\ Q(\boldsymbol{h}^1, \ldots, \boldsymbol{h}^l|\boldsymbol{v}) &= Q(\boldsymbol{h}^1|\boldsymbol{v})Q(\boldsymbol{h}^2|\boldsymbol{h}^1)\ldots Q(\boldsymbol{h}^l|\boldsymbol{h}^{l-1}). \end{aligned} \quad (1)$$

The probabilities $P(\boldsymbol{v}|\boldsymbol{h}^1), P(\boldsymbol{h}^1|\boldsymbol{h}^2), \ldots, P(\boldsymbol{h}^{l-2}|\boldsymbol{h}^{l-1})$ and the joint distribution $P(\boldsymbol{h}^{l-1}, \boldsymbol{h}^l)$ can be calculated exactly using the RBM weight matrices at the various levels. The terms $Q(\boldsymbol{h}^1|\boldsymbol{v}), Q(\boldsymbol{h}^2|\boldsymbol{h}^1), \ldots, Q(\boldsymbol{h}^l|\boldsymbol{h}^{l-1})$ could also be calculated exactly using the RBM weight matrices but in practice a mean field approximation is used (as in the wake-sleep algorithm).

The weight matrix associated with level $l+1$ is estimated using the RBM training algorithm applied to an aggregated training set.[2]. That is, for each training token $\boldsymbol{v}^n$ we sample from the posterior $Q(\boldsymbol{h}^l|\boldsymbol{v}^n)$ and the collection of samples obtained in

[2]We encountered this type of synthetic data set in Section III-J

this way constitutes the training set for RBM training of the matrix at level $l+1$. Thus the hidden variables at one level serve as features for training the next.

Consider the case $l = 1$. Suppose that after training a first level RBM with weight matrix $\boldsymbol{W}^1$ we wish to add a second level. Construct a second RBM with weight matrix $\boldsymbol{W}^2$ whose "visible" level is the hidden level of the first RBM and let $P(\boldsymbol{h}^2|\boldsymbol{W}^2)$ be the marginal distribution of the hidden level of the second RBM. Use the aggregated training set at level 1 and ordinary RBM training to estimate $\boldsymbol{W}^2$. The criterion which is optimized is

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

where $\boldsymbol{h}^1$ ranges over the aggregated training set. This is equivalent to optimizing the EM auxiliary function

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

which reduces to optimizing

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

since $P(\boldsymbol{v}^n|\boldsymbol{h}^1, \boldsymbol{W}^1, \boldsymbol{W}^2)$ is independent of $\boldsymbol{W}^2$. Here $P(\boldsymbol{h}^1|\boldsymbol{W}^2)$ is the marginal distribution calculated by regarding $\boldsymbol{h}^1$ as the visible variables for the second RBM:

$$P(\boldsymbol{h}^1|\boldsymbol{W}^2) \quad = \quad \sum_{\boldsymbol{h}^2} P(\boldsymbol{h}^1, \boldsymbol{h}^2|\boldsymbol{W}^2)$$

(Since the only part of the complete likelihood function $P(\boldsymbol{v}, \boldsymbol{h}^1)$ that is being re-estimated is the prior on the hidden variables $\boldsymbol{h}^1$ so this is an instance of minimum divergence training.) We can initialize the RBM training of $\boldsymbol{W}^2$ by copying $\boldsymbol{W}^1$, so RBM training ensures that

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

Of course this way of guaranteeing that the EM auxiliary function increases constrains the number of nodes added at level 2 to be the same as the number of visible nodes at level 1. After adding the second level, the joint distribution of $(\boldsymbol{v}, \boldsymbol{h}^1, \boldsymbol{h}^2)$ is

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

where the first factor is calculated with the first RBM and the second with the second RBM. (Note the asymetry: one is a posterior distribution, the other a joint distribution.) The posterior of $(\boldsymbol{h}^1, \boldsymbol{h}^2)$ given $\boldsymbol{v}$, that is

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

cannot be calculated exactly so we use the approximation given by (1)

$$Q(\boldsymbol{h}^1, \boldsymbol{h}^2|\boldsymbol{v}) \quad = \quad Q(\boldsymbol{h}^1|\boldsymbol{v}) Q(\boldsymbol{h}^2|\boldsymbol{h}^2)$$

instead, where the first term is calculated with $\boldsymbol{W}^1$ and the second with $\boldsymbol{W}^2$. We must show that the variational lower bound at level 2 calculated with this posterior, $\mathcal{L}_2$, is greater than the variational lower bound calculated at level 1, $\mathcal{L}_1$.

By definition

$$\mathcal{L}_2 \quad = \quad \sum_n E_{Q(\boldsymbol{h}^1, \boldsymbol{h}^2|\boldsymbol{v}^n)} \left[\ln P(\boldsymbol{v}^n, \boldsymbol{h}^1, \boldsymbol{h}^2) \right] + \sum_n H(Q(\boldsymbol{h}^1, \boldsymbol{h}^2|\boldsymbol{v}^n)),$$

which we rewrite as follows

$$\begin{aligned}
\mathcal{L}_2 &= \sum_n E_{Q(\boldsymbol{h}^1, \boldsymbol{h}^2|\boldsymbol{v}^n)} \left[\ln P(\boldsymbol{v}^n, \boldsymbol{h}^1, \boldsymbol{h}^2) \right] + \sum_n H(Q(\boldsymbol{h}^1, \boldsymbol{h}^2|\boldsymbol{v}^n)) \\
&= \sum_n E_{Q(\boldsymbol{h}^1, \boldsymbol{h}^2|\boldsymbol{v}^n)} \left[\ln Q(\boldsymbol{h}^2|\boldsymbol{h}^1) + \ln P(\boldsymbol{v}^n, \boldsymbol{h}^1) \right] + \sum_n H(Q(\boldsymbol{h}^1, \boldsymbol{h}^2|\boldsymbol{v}^n)) \\
&= \sum_n \sum_{\boldsymbol{h}^1, \boldsymbol{h}^2} Q(\boldsymbol{h}^1, \boldsymbol{h}^2|\boldsymbol{v}^n) \ln Q(\boldsymbol{h}^2|\boldsymbol{h}^1) + \sum_n \sum_{\boldsymbol{h}^1} Q(\boldsymbol{h}^1|\boldsymbol{v}^n) \ln P(\boldsymbol{v}^n|\boldsymbol{h}^1) P(\boldsymbol{h}^1|\boldsymbol{W}^2) \\
&\quad - \sum_n \sum_{\boldsymbol{h}^1, \boldsymbol{h}^2} Q(\boldsymbol{h}^1, \boldsymbol{h}^2|\boldsymbol{v}^n) \left(\ln Q(\boldsymbol{h}^2|\boldsymbol{h}^1) + \ln Q(\boldsymbol{h}^1|\boldsymbol{v}^n) \right) \\
&= \sum_n \sum_{\boldsymbol{h}^1} Q(\boldsymbol{h}^1|\boldsymbol{v}^n) \ln P(\boldsymbol{v}^n|\boldsymbol{h}^1) P(\boldsymbol{h}^1|\boldsymbol{W}^2) + \sum_n H(Q(\boldsymbol{h}^1|\boldsymbol{v}^n)).
\end{aligned}$$

On the other hand,

$$\begin{aligned}\mathcal{L}_1 &= \sum_n \sum_{\boldsymbol{h}^1} Q(\boldsymbol{h}^1|\boldsymbol{v}^n) \ln P(\boldsymbol{v}^n, \boldsymbol{h}^1|\boldsymbol{W}^1) + \sum_n H(Q(\boldsymbol{h}^1|\boldsymbol{v}^n)) \\ &= \sum_n \sum_{\boldsymbol{h}^1} Q(\boldsymbol{h}^1|\boldsymbol{v}^n) \ln P(\boldsymbol{v}^n|\boldsymbol{h}^1) P(\boldsymbol{h}^1|\boldsymbol{W}^1) + \sum_n H(Q(\boldsymbol{h}^1|\boldsymbol{v}^n)).\end{aligned}$$

and the inequality $\mathcal{L}_2 \geq \mathcal{L}_1$ follows from the fact that $\boldsymbol{W}^2$ was chosen so that

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

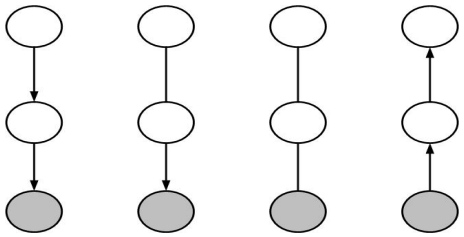

Fig. 9. Standard deep belief net, Hinton deep belief net, deep Boltzmann machine and feed forward neural net

Note The posterior approximation (1) is not as good as it might be because the way that a term such as $Q(\boldsymbol{h}^k|\boldsymbol{h}^{k-1})$ is evaluated depends on the construction up to level k but not on the construction of subsequent levels. On the other hand the variational Bayes posterior calculation used in training deep Boltzmann machines takes account of all levels above and below in evaluating the posterior at a given level k.

The argument that the variational lower bound increases as successive layers are added to a DBN requires that the weight matrices have the same dimensions at all levels; there is no such restriction for DBMs. Also the DBN level building algorithm is inherently greedy, but the DBM training algorithm is not (although it may have to be initialized greedily in practice, a process commonly referred to as pretraining).

Generative pattern recognition In [19], the authors show how, after building a multilayer DBN in the way just described a slight modification of the wake-sleep algorithm can be used to improve it. As in the case of traditional deep belief networks, each weight matrix (expect for the weight matrix which defines the top layer RBM) is a replaced by a pair of matrices, one of which is used to calculate posteriors and the other to sample from the generative model.

By adding a softmax layer with 10 label nodes, the DBN can serve as a generative model of label-image pairs in the MNIST handwritten digit data set. The model can be used to recognize digits using the minimum Bayes risk principle (that is, by calculating posteriors of the form $P(\mathrm{label}|\mathrm{digit})$). At the time of publication, results on the MNIST test set obtained in this way were state of the art.

Discriminative pattern recognition Suppose we have built a DBN with L levels, so that the calculation of $Q(\boldsymbol{h}^L|\boldsymbol{v})$ using the mean field approximation (1) mimics the activity of a feed forward neural net (without the softmax layer). The generative model is trained in an unsupervised way (without class labels) but we assume that training behaves in such a way that the highest level hidden variables are the most useful features for discriminating between classes. Thus we can assume a decomposition of the form

$$Q(\boldsymbol{c}|\boldsymbol{v}) = \sum_{\boldsymbol{h}^L} Q(\boldsymbol{c}|\boldsymbol{h}^L) Q(\boldsymbol{h}^L|\boldsymbol{v})$$

where $\boldsymbol{c}$ denotes a class label vector and $Q(\boldsymbol{c}|\boldsymbol{h}^L)$ is calculated with a softmax layer. Most of the parameters in this neural network (namely those which specify $Q(\boldsymbol{h}^L|\boldsymbol{v})$) are derived from the generative model of the data. The backpropagation algorithm is very successful if initialized from the generative model in this way. This seems to have come as something of a surprise to the machine learning community as it had long been argued that one should not pay much attention to the term $P(\boldsymbol{v})$ in the decomposition

$$P(\boldsymbol{v}, \boldsymbol{c}) = Q(\boldsymbol{c}|\boldsymbol{v}) P(\boldsymbol{v})$$

since $P(\boldsymbol{v})$ (that is, the generative model) plays no direct role in recognition decisions.

The success of generative modeling as an initialization for backpropagatation is attributed to the fact that, if left to its own devices, backpropagation generally gets stuck in a local minimum of the error function or overfits the training data. (Hinton frequently argues that generative modeling is less susceptible to overfitting than discriminative modeling.) See Section 4 of [1] for a discussion of these questions.

B. DBNs vs DBMs

Salakhutdinov suggests that Hinton's construction can be used with a minor modification to initialize deep Boltzmann machines [2].

Consider the case $l = 2$. The joint distribution of the hidden and visible variables in a 2 layer DBN is

$$P(\boldsymbol{v}, \boldsymbol{h}^1, \boldsymbol{h}^2) \quad = \quad P(\boldsymbol{v}|\boldsymbol{h}^1, \boldsymbol{W}^1)P(\boldsymbol{h}^1, \boldsymbol{h}^2|\boldsymbol{W}^2)$$

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

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

and normalize the probabilities so that they sum to 1, we would get a Boltzmann distribution but this expression counts the prior on $\boldsymbol{h}^1$ twice. (The first and second level RBMs each contribute a prior on $\boldsymbol{h}^1$.) To avoid this doubling effect Salakhudtinov suggests halving the weight matrices $\boldsymbol{W}^1$ and $\boldsymbol{W}^2$ before taking the product and renormalizing. The higher levels are handled in the same way. This heuristic is merely used an initialization for subsequent DBM training.

It appears that, when used as feature extractors for discriminative pattern recognition, DBMs outperform DBNs [2] but more research seems to be needed here. Training DBMs is much slower than training DBNs becasue the variational Bayes posterior calculation for DBMs is more computationally demanding than the mean field approximation used to evaluate posteriors for DBNs. The variational Bayes posterior calculation requires cycling over all of the levels in the DBM rather than performing a single feed forward pass. This results in a better posterior calculation but, contrary to the case of DBNs, it does not suggest an immediately obvious way of using the DBM weight matrix to initialize a feed forward neural network for discriminative pattern recognition. Salakhutdinov proposes such an initialization in the case of a DBM with two hidden layers but it is not clear to me how this would extend to the case of three hidden layers.

VII. Some research problems for Boltzmann machines

Although the DBM training algorithm appears to be better than the DBN training algorithm because sparse Boltzmann machines support efficient variational Bayes inference, only Salakhutdinov appears to have explored using DBMs in building discriminative closed-set pattern recognizers in a similar manner to DBNs and, as discussed above, Salakhutdinov's solution appears to be incomplete.

Although interest in DBNs was revived by the way they were used to build a a generative pattern recognizer in Hinton, Osindero and Teh [19], very little work seems to have been done on using DBMs for generative pattern recognition. Of course a straightforward approach to building generative pattern recognizers using BMs would be to construct a separate generative model for each class to be recognized but this seems to be less interesting than the approach of designing a single generative model to recognize all classes as in [19]. Ideally, a single generative model constructed in this way should lend itself to being turned into a (presumably better) discriminative pattern recognizer. Moreover, when applied to speech recognition, it should be capable of accommodating utterance level hidden variables which could serve to characterize the speaker and channel effects in a given recording. Estimating these hidden variables for a given utterance would enable speech recognition to be carried out in a speaker adaptive way.

An easier way of designing a speaker adaptive neural net speech recognizer might be to use traditional feature space transformation methods such as fMLLR. This would decouple the problem of estimating utterance level hidden variables from that of training the neural net. Accommodating utterance level hidden variables in a triphone state recognizer trained with backpropagation does not seem to be easy but the general problem of how to integrate utterance level hidden variables into a discriminative speech recognizer looks interesting.

A. Generative pattern recognition

We have seen how variables of different types (binary and real-valued) can be jointly modeled in the Gauss-Bernoulli RBM. Note that it is not the case that the marginal distribution of the real-valued variables is Gaussian; rather the conditional of each real-valued valued variable given all of the other variables is Gaussian. This observation indicates how Boltzmann machines can be extended to accommodate other types of distribution.

The "exponential family harmonium" (Welling *et al.*) is a generalization where the nodes in the graph have associated with them variables which may be discrete (binary or categorical) or continous (scalar or vector valued) and the conditional distribution of one variable given the others is in the exponential family. Different types of exponential distribution can be accomodated in a single model. All of the VBEM and contrastive divergence training algorithms can be extended to the harmonium (I think). Because categorical variables are supported generative models for N-way pattern recognition (as distinct from 2-way pattern recognition) can be built (although this doesn't seeem to have been explored widely).

Because this type pattern recognizer is a generative model, it should not be hard to introduce utterance level hidden variables for speech recognition applications. All that would be required would be to designate some of the hidden variables as being tied across all of the acoustic observations in a given utterance.

B. Discriminative pattern recognition

To fix ideas suppose we have built a generative BM pattern recognizer with 2 visible layers (one for data, the other for labels) and 4 hidden layers and that 3 iterations of variational Bayes are needed to perform pattern recognition. The 3 variational Bayes iterations can be unfolded into 15 (i.e. 3×4) mean field updates; that is, they can be implemented by a 15 layer feed forward neural net with tied weight matrices. Standard backpropagation could be used to untie and re-estimate the weight matrices, converting the generative pattern recognizer into a discriminative pattern recognizer. (Perhaps the label layer ought to be handled differently, for instance as in [2].)

If the generative pattern recognizer is designed so as to contain utterance level hidden variables the same would be true of the discriminative pattern recognizer.

REFERENCES

[1] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[2] R. Salakhutdinov and G. Hinton, "An efficient learning procedure for deep boltzmann machines," in *CSAIL Technical report, MIT-CSAIL-TR-2010-37*, Aug. 2010.

[3] D. MacKay, *Information theory, inference and learning algorithms*. New York, NY: Cambridge University Press, 2003.

[4] P. Kenny, P. Ouellet, N. Dehak, V. Gupta, and P. Dumouchel, "A study of inter-speaker variability in speaker verification," *IEEE Trans. Audio, Speech and Lang. Process.*, vol. 16, no. 5, pp. 980–988, July 2008. [Online]. Available: http://www.crim.ca/perso/patrick.kenny

[5] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 19, no. 4, pp. 788–798, May 2011.

[6] P. Kenny, "Bayesian speaker verification with heavy tailed priors," in *Proc. Odyssey 2010: The speaker and Language Recognition Workshop*, Brno, Czech Rebublic, June 2010.

[7] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Proc. Eurospeech 2011*, 2011.

[8] C. Bishop, *Pattern Recognition and Machine Learning*. New York, NY: Springer Science+Business Media, LLC, 2006.

[9] S. Haykin, *Neural networks: a comprehensive introduction*. Upper Saddle River, New Jersey: Prentice-Hall, Inc., 1999.

[10] Y. Bengio and O. Delalleau, "Justifying and generalizing contrastive divergence," *Neural Computation*, vol. 21, no. 6, pp. 1601–1621, 2009.

[11] K. Cho, A. Ilin, and T. Raiko, "Improved learning of Gaussian-Bernoulli restricted Boltzmann machines," in *Master's Thesis, Aalto University*, 2011.

[12] G. Dahl and G. Hinton, "Phone recognition with the mean-covariance restricted boltzmann machine," in *Advances in Neural Information Processing 23*, 2010.

[13] A. Mohammed, G. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Trans. on Audio, Speech, and Language Processing*, 2010.

[14] G. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing (Special Issue on Deep Learning for Speech and Language Processing)*, Jan. 2012.

[15] A. Mohammed, T. Sainath, G. Dahl, B. Ramabhadran, G. Hinton, and M. Picheny, "Deep belief networks using discriminative features for phone recognition," in *Proc. ICASSP 2011*, 2011.

[16] G. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large vocabulary speech recognition," in *Advances in Neural Information Processing 23* .

[17] R. Memisevic and G. Hinton, "Learning to represent spatial transformations with factored higher-order Boltzmann machines," *Neural Computation*, vol. 22, pp. 1473–1492.

[18] B. Frey, G. Hinton, and P. Dayan, "Does the wake-sleep algorithm produce good density estimators?" in *Advances in Neural Information Processing 8*, 1996.

[19] G. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, pp. 1527–1554, 2006.