10-707: Advanced Deep Learning **Spring 2020**

Lecture 13: March 2

Lecturer: Andrej Risteski *Scribes: Sail Kim, Beomjo Park, Jeffrey Lee*

13.1 Learning Completely Observable Energy Models

Our goal is to sample from some distribution $p_\theta(x)$ up to a constant of proportionality, where

$$p_\theta(x) = \frac{1}{Z_\theta} \exp\left(-E_\theta(x)\right), \qquad Z_\theta = \int_{\mathrm{Supp}(\mathcal{X})} \exp\left(-E_\theta(x)\right) \mathrm{d}x$$

Our approach to learning p_θ is the usual maximum likelihood approach, where given some samples $x_1, \ldots, x_n$, we want to to solve

$$\max_{\theta\in\Theta} \sum_{i=1}^{n} \log p_\theta(x_i) = \max_{\theta\in\Theta} - \sum_{i=1}^{n} \log(E_\theta(x_i)) - \log(Z_\theta) \tag{13.1}$$

We can attempt to optimize via *gradient descent*. It is typically easy to take the gradient of the first term, i.e. $\nabla_\theta E(\theta)$, such as when $E(\theta)$ is a neural net, Ising model, etc. However, the second term $\nabla_\theta \log Z_\theta$ may be harder. We have

$$\begin{aligned}
\nabla_\theta \log Z_\theta &= \frac{1}{Z_\theta} \nabla_\theta Z_\theta \\
&= \frac{1}{Z_\theta} \nabla_\theta \left(\int_x \exp(-E_\theta(x)) \right) \\
&= \frac{1}{Z_\theta} \int_x \exp(-E_\theta(x)) \nabla_\theta \left(-E_\theta(x)\right) \\
&= \mathbb{E}_{p_\theta}[-\nabla_\theta E_\theta(x)].
\end{aligned}$$

Overall, the gradient can be written as

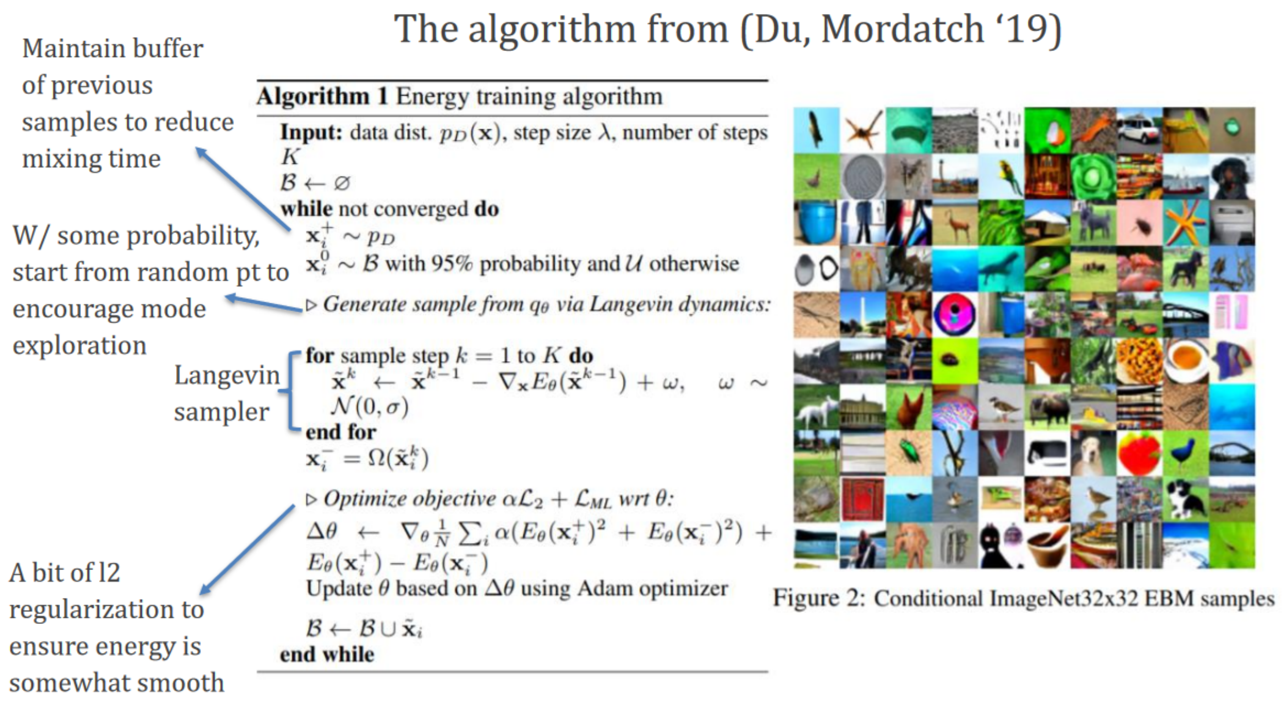
$$\begin{aligned}
\nabla_\theta \left(\frac{1}{n} \sum_{i=1}^{n} \log p_\theta(x_i) \right) &= \frac{1}{n} \left(\sum_i -\nabla_\theta E_\theta(x_i) \right) - \mathbb{E}_{p_\theta}[-\nabla_\theta E_\theta(x)] \\
&\approx \mathbb{E}_{p_{data}}[-\nabla_\theta E_\theta(x_i)] - \mathbb{E}_{p_\theta}[-\nabla_\theta E_\theta(x)].
\end{aligned}$$

as n goes to infinity. This makes intuitive sense because gradient descent is simply trying to make the expectation of the energy function with respect to the data distribution and the current distribution match.

An obvious way to produce estimates for $\mathbb{E}_{p_\theta}[-\nabla_\theta E_\theta(x)]$ would be to sample approximately from p_θ using Langevin dynamics. This was only discovered recently since Langevin dynamics weren't in view of the community until recently and high-dimensional data like image distributions will typically be very multimodal, which may lead to the tendency of getting stuck at a mode.

In the following two sections 13.1.1 and 13.1.2, we go through two recent developments.

13.1.1 Applying Gradient Descent to Maximum Likelihood



The algorithm from (Du, Mordatch '19)

```
Algorithm 1 Energy training algorithm
Input: data dist. $p_D(\mathbf{x})$, step size $\lambda$, number of steps $K$
$\mathcal{B} \leftarrow \varnothing$
while not converged do
    $\mathbf{x}_i^+ \sim p_D$
    $\mathbf{x}_i^0 \sim \mathcal{B}$ with 95% probability and $\mathcal{U}$ otherwise
    ▷ Generate sample from $q_\theta$ via Langevin dynamics:
    for sample step $k = 1$ to $K$ do
        $\tilde{\mathbf{x}}^k \leftarrow \tilde{\mathbf{x}}^{k-1} - \nabla_{\mathbf{x}} E_\theta(\tilde{\mathbf{x}}^{k-1}) + \omega, \quad \omega \sim \mathcal{N}(0, \sigma)$
    end for
    $\mathbf{x}_i^- = \Omega(\tilde{\mathbf{x}}_i^k)$
    ▷ Optimize objective $\alpha\mathcal{L}_2 + \mathcal{L}_{ML}$ wrt $\theta$:
    $\Delta\theta \leftarrow \nabla_\theta \frac{1}{N}\sum_i \alpha(E_\theta(\mathbf{x}_i^+)^2 + E_\theta(\mathbf{x}_i^-)^2) + E_\theta(\mathbf{x}_i^+) - E_\theta(\mathbf{x}_i^-)$
    Update $\theta$ based on $\Delta\theta$ using Adam optimizer
    $\mathcal{B} \leftarrow \mathcal{B} \cup \tilde{\mathbf{x}}_i$
end while
```

Maintain buffer of previous samples to reduce mixing time

W/ some probability, start from random pt to encourage mode exploration

Langevin sampler

A bit of l2 regularization to ensure energy is somewhat smooth

Figure 2: Conditional ImageNet32x32 EBM samples

Figure 13.1: Algorithm from (Du, Mordatch '19) for training energy based model using gradient descent, with a few tweaks.

The above algorithm is almost vanilla gradient descent on maximum likelihood, which was described earlier, and performs respectably though not as well as GANs. A few points to note are:

- We keep a buffer $\mathcal{B}$ of images from previous Langevin runs. We warm start the current run with that sample with high probability. This is presumably this sample has already taken some number of steps in a previous walk and is closer to the stationary distribution. However, we still choose a uniformly random starting point with low probability to encourage mode exploration in case a previous run got "stuck".
- A vanilla Langevin sampler is used.
- l2 regularization is added to smooth E_θ in a Lipschitz sense and facilitate training.

13.1.2 Applying Gradient Descent to Surrogate Likelihood

The algorithm described in [Song and Ermon (2019)] instead considers a *surrogate likelihood*, and the objective they minimize is

$$\mathbb{E}_{p_{data}} ||\nabla_x \log p_{data}(x) - (-E_\theta(x))||^2$$

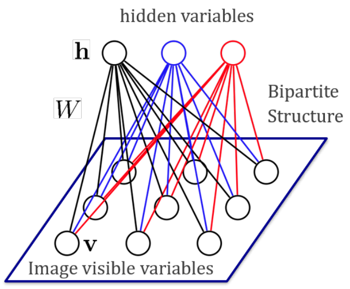
which can be rewritten in terms of E_θ (using integration by parts) for taking gradients. A few things to note are:

- Simulated tempering was used to alleviate multimodality.
- To deal with sparser areas of the data distribution where estimates of $\nabla_x \log p_{data}(x)$ are worse, smoothing by convolving with Gaussians is used.

The results are even better than those of the previous paper.

13.2 Learning Restricted Boltzmann Machines (RBMs)

Restricted Boltzmann Machines is an undirected latent-variable model with visible variables $\boldsymbol{x} \in \{0,1\}^D$ are connected to hidden variables $\boldsymbol{h} \in \{0,1\}^F$ as in the following figure:



The energy of the joint configuration:

$$\begin{aligned}\mathbb{E}(\boldsymbol{x},\boldsymbol{h}) &= -\boldsymbol{h}^\top W\boldsymbol{x} - c^\top \boldsymbol{x} - b^\top \boldsymbol{h} \\ &= -\sum_j \sum_k W_{j,k} h_j x_k - \sum_k c_k x_k - \sum_j b_j h_j\end{aligned}$$

Probability of the joint configuration:

$$p(\boldsymbol{x},\boldsymbol{h}) = \exp(-\mathbb{E}(\boldsymbol{x},\boldsymbol{h}))/Z$$

The posterior over the hidden variables is easy to sample from (Conditional independence)

$$p(\boldsymbol{h}|\boldsymbol{x}) = \prod_j p(h_j|\boldsymbol{x}), p(h_j = 1|\boldsymbol{x}) = \frac{1}{1+\exp(-(b_j + W_j\boldsymbol{x}))}$$

$$p(\boldsymbol{x}|\boldsymbol{h}) = \prod_k p(x_k|\boldsymbol{h}), p(x_k = 1|\boldsymbol{h}) = \frac{1}{1+\exp(-(c_k + \boldsymbol{h}^\top W_k))}$$

Recall that from Equation (13.1), we need the marginal distribution $p_\theta(\boldsymbol{x})$. Since we have latent variables, we need to express the likelihood when we marginalize out the latents:

$$\begin{aligned}
p_\theta(\boldsymbol{x}) &= \sum_{\boldsymbol{h}\in\{0,1\}^H} \exp(\boldsymbol{h}^\top W\boldsymbol{x} + c^\top \boldsymbol{x} + b^\top \boldsymbol{h})/Z \\
&= \exp(c^\top \boldsymbol{x}) \sum_{h_1\in\{0,1\}} \cdots \sum_{h_H\in\{0,1\}} \exp\left(\sum_j h_j W_j \boldsymbol{x} + b_j h_j\right)/Z \\
&= \exp(c^\top \boldsymbol{x}) \left(\sum_{h_1\in\{0,1\}} \exp(h_1 W_1 \boldsymbol{x} + b_1 h_1)\right)\dots\left(\sum_{h_H\in\{0,1\}} \exp(h_H W_H \boldsymbol{x} + b_H h_H)\right)/Z \\
&= \exp(c^\top \boldsymbol{x})(1+\exp(b_1 + W_1\boldsymbol{x}))\dots(1+\exp(b_H + W_H\boldsymbol{x}))/Z \\
&= \exp(c^\top \boldsymbol{x})\exp(\log(1+\exp(b_1 + W_1\boldsymbol{x})))\dots\exp(\log(1+\exp(b_H + W_H\boldsymbol{x})))/Z \\
&= \exp\left(\underbrace{c^\top \boldsymbol{x} + \sum_{j=1}^H \log(1+\exp(b_j + W_j\boldsymbol{x}))}_{=F(\boldsymbol{x})}\right)/Z \\
&= \exp(F(\boldsymbol{x}))/Z
\end{aligned}$$

Note from Equation (13.1):

$$\nabla_\theta\left(\frac{1}{n}\sum_{i=1}^n \log p_\theta(x_i)\right) = \frac{1}{n}\left(\sum_i -\nabla_\theta F_\theta(x_i)\right) - \mathbb{E}_{p_\theta}\left[-\nabla_\theta F_\theta(\boldsymbol{x})\right] \tag{13.2}$$

With above reduction, we have the following analytic form of the first part:

$$\begin{aligned}
\nabla_{\boldsymbol{W}} F_\theta(\boldsymbol{x}) &\overset{(*)}{=} \boldsymbol{h}(\boldsymbol{x})\boldsymbol{x}^\top, \qquad \left(\nabla_{\boldsymbol{W}_{ij}} F_\theta(\boldsymbol{x}) = \mathbb{P}(\boldsymbol{h}_j = 1|\boldsymbol{x}_i)\right) \\
\nabla_{\boldsymbol{b}} F_\theta(\boldsymbol{x}) &= \boldsymbol{h}(\boldsymbol{x}), \\
\nabla_c F_\theta(\boldsymbol{x}) &= \boldsymbol{x}
\end{aligned}$$

where $\boldsymbol{h}(\boldsymbol{x}) := \text{sign}(\text{b} + \text{Wx})$.

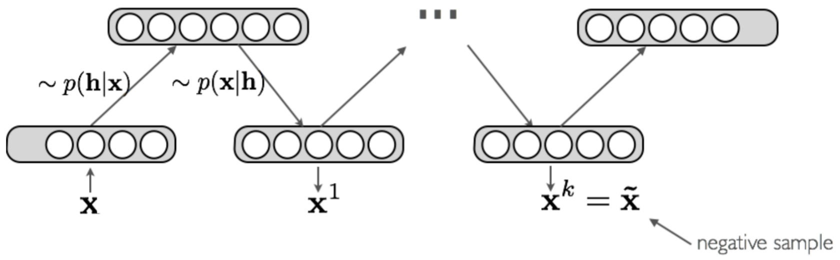
Derivation of $(*)$:

$$\nabla_{\boldsymbol{W}_{ij}} F_\theta(\boldsymbol{x}) = \nabla_{\boldsymbol{W}_{ij}}\left(\boldsymbol{c}^\top\boldsymbol{x} + \sum_{j=1}^H \log\left(1+\exp(b_j + \boldsymbol{W}_{j\cdot}\boldsymbol{x})\right)\right) = \frac{\exp(b_j + \boldsymbol{W}_{j\cdot}\boldsymbol{x})}{1+\exp(b_j + \boldsymbol{W}_{j\cdot}\boldsymbol{x})} = \mathbb{P}(\boldsymbol{h}_j = 1|\boldsymbol{x})\boldsymbol{x}_i$$

The hard part is the expectation in Equation (13.2). We resort to *Gibbs sampler*:

Within each update iteration,
Repeat k times:
1. Sample $\boldsymbol{h} \sim \mathbb{P}(\boldsymbol{h}|\boldsymbol{x})$
2. Sample $\boldsymbol{x} \sim \mathbb{P}(\boldsymbol{x}|\boldsymbol{h})$

to obtain $\tilde{\boldsymbol{x}}$



$\sim p(\mathbf{h}|\mathbf{x})$
$\sim p(\mathbf{x}|\mathbf{h})$
$\mathbf{x}$
$\mathbf{x}^1$
$\mathbf{x}^k = \tilde{\mathbf{x}}$
negative sample

In general, the bigger k is, the less biased the estimate of the gradient will be, however, in practice, k=1 works well for learning good features and for pretraining.

To sum up into the whole machinary (*Contrastive Divergence*[Hinton (2002)]):

Algorithm 1 Contrastive Divergence(CD)-k

```
Require: Step size $\alpha$.
   For each traning example $\boldsymbol{x}$,
1: repeat
2:     Generate a negative sample $\tilde{\boldsymbol{x}}$ from $k$ Gibbs iterations from $\boldsymbol{x}$
3:     Update
4:         $\boldsymbol{W} \leftarrow \boldsymbol{W} + \alpha \left(\boldsymbol{h}(\boldsymbol{x})\boldsymbol{x}^\top - \boldsymbol{h}(\tilde{\boldsymbol{x}})\tilde{\boldsymbol{x}}^\top\right)$
5:         $\boldsymbol{b} \leftarrow \boldsymbol{b} + \alpha \left(\boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{h}(\tilde{\boldsymbol{x}})\right)$
6:         $\boldsymbol{c} \leftarrow \boldsymbol{c} + \alpha \left(\boldsymbol{x} - \tilde{\boldsymbol{x}}\right)$
7: until Converge
```

Algorithm 1 can be improved to *Persistent CD* by the warm start, i.e., instead of initializing the chain to $\boldsymbol{x}$, initialize the chain to the negative sample of the last iteration $\tilde{\boldsymbol{x}}$. See [Tieleman (2008), ICML] for detail.

We rely on approximate *tricks* since it is not easy to debug training RBMs (e.g. using gradient checks)

1. Plot the average stochastic reconstruction and see if it tends to decrease

2. For inputs that correspond to image, we visualize the connection coming into each hidden unit as if it was an image

3. Gives an idea of the type of visual feature each hidden unit detects

4. We can also try to approximate the partition function Z and see whether the (approximated) NLL decreases. See [Salakhutdinov & Murray (2008), ICML] for detail.

13.3 Learning Deep Belief Networks (DBNs)

In this section, we stack the layers on top of the RBM. If we think that the the RBM hidden node captures the *low-level features*, such as edges, stacked hidden layers will capture the *high-level features* by the combination of edges. Figure 13.2 illustrates the typical 3-Layer DBN structure we'll use throughout this section to understand the mechanism.

Notice that this is a generative model that mixes undirected and directed connections between variables, in that (1) top 2 layers' distribution $p(\boldsymbol{h}^{(2)}, \boldsymbol{h}^{(3)})$ is an RBM and (2) the other layers form a belief network with the conditional distributions:

$$\mathbb{P}(h_j^{(1)} = 1 | \boldsymbol{h}^{(2)}) = \text{sign}\left(\mathrm{b}^{(1)} + {\mathrm{W}^{(2)}}^\top \mathrm{h}^{(2)}\right)$$

$$\mathbb{P}(x_i = 1 | \boldsymbol{h}^{(1)}) = \text{sign}\left(\mathrm{b}^{(0)} + {\mathrm{W}^{(1)}}^\top \mathrm{h}^{(1)}\right)$$

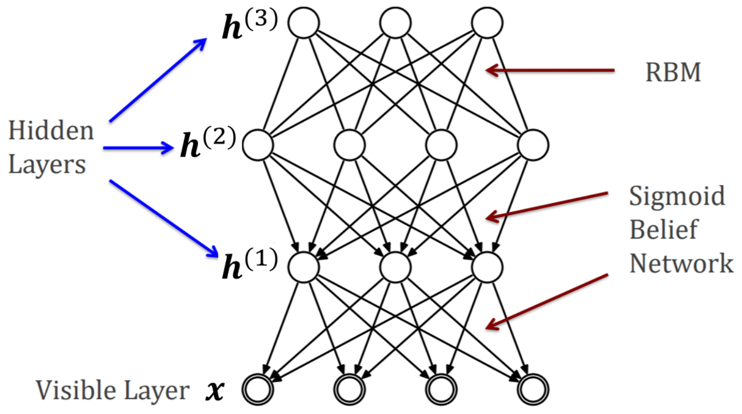


Figure 13.2: 3-Layer Deep Belief Network

Then, the joint distribution of a DBN is given as

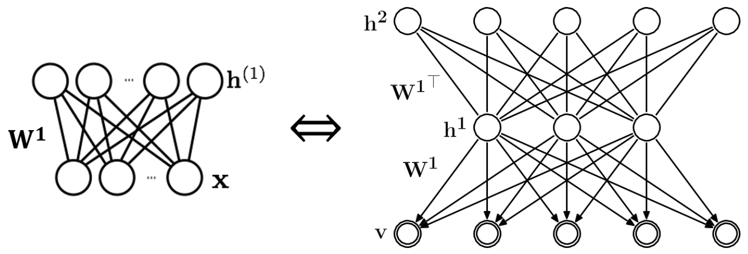
$$\begin{aligned}
\mathbb{P}(\boldsymbol{x}, \boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, \boldsymbol{h}^{(3)}) &= \mathbb{P}(\boldsymbol{h}^{(2)}, \boldsymbol{h}^{(3)})\mathbb{P}(\boldsymbol{h}^{(1)}|\boldsymbol{h}^{(2)})\mathbb{P}(\boldsymbol{x}|\boldsymbol{h}^{(1)}) \\
\mathbb{P}(\boldsymbol{h}^{(2)}, \boldsymbol{h}^{(3)}) &= \frac{1}{Z}\exp\left(\boldsymbol{h}^{(2)\top}\boldsymbol{W}^{(3)}\boldsymbol{h}^{(3)} + \boldsymbol{b}^{(2)\top}\boldsymbol{h}^{(2)} + \boldsymbol{b}^{(3)\top}\boldsymbol{h}^{(3)}\right) \\
\mathbb{P}(\boldsymbol{h}^{(1)}|\boldsymbol{h}^{(2)}) &= \prod_j \mathbb{P}(h_j^{(1)}|\boldsymbol{h}^{(2)}) \\
\mathbb{P}(\boldsymbol{x}|\boldsymbol{h}^{(1)}) &= \prod_i \mathbb{P}(x_i|\boldsymbol{h}^{(1)})
\end{aligned}$$

The reasoning behind the parameterization is based on the *Gibbs variational principle*. Let's just consider an RBM $\{\boldsymbol{x}, \boldsymbol{h}^{(1)}\}$. Then, the evidence lowerbound(ELBO) is

$$\log \mathbb{P}(\boldsymbol{x}) \geq \sum_{\boldsymbol{h}^{(1)}} q(\boldsymbol{h}^{(1)}|\boldsymbol{x}) \log \mathbb{P}(\boldsymbol{x}, \boldsymbol{h}^{(1)}) - \sum_{\boldsymbol{h}^{(1)}} q(\boldsymbol{h}^{(1)}|\boldsymbol{x}) \log q(\boldsymbol{h}^{(1)}|\boldsymbol{x}) \tag{13.3}$$

since $\mathbb{P}(\boldsymbol{x}) = \sum_{\boldsymbol{h}^{(1)}} \mathbb{P}(\boldsymbol{x}, \boldsymbol{h}^{(1)})$ and $\mathbb{P}(\boldsymbol{h}^{(1)}) \propto \mathbb{P}(\boldsymbol{x}, \boldsymbol{h}^{(1)})$. Note that the equality in Equation (13.3) is attained if $q(\boldsymbol{h}^{(1)}|\boldsymbol{x}) = \mathbb{P}(\boldsymbol{h}^{(1)}|\boldsymbol{x})$ which implies the variational distribution $q(\boldsymbol{h}^{(1)}|\boldsymbol{x})$ exactly matches the conditional distribution $p(\boldsymbol{h}^{(1)}|\boldsymbol{x})$ in a KL distance. We claim that ELBO improves as we add layers.

First off, a two-layer DBN with appropriately tied weights is equivalent to an RBM:



We provide the sketch intuition for brevity:

- Gibbs sampling converges to model distribution in first case.
- Gibbs sampling on top two layers, plus one last sample of $\boldsymbol{x}$ given $\boldsymbol{h}^{(1)}$ converges to model distribution in second.

• The steps in these two random walks are *exactly* the same.

Since adding 2nd layer means untying the parameters, Equation (13.3) is now

$$\log \mathbb{P}(\boldsymbol{x}) \geq \sum_{\boldsymbol{h}^{(1)}} q(\boldsymbol{h}^{(1)}|\boldsymbol{x}) \left(\log p(\boldsymbol{h}^{(1)}) + \log p(\boldsymbol{h}^{(1)})\right) - \sum_{\boldsymbol{h}^{(1)}} q(\boldsymbol{h}^{(1)}|\boldsymbol{x}) \log q(\boldsymbol{h}^{(1)}|\boldsymbol{x})$$

• When adding a second layer, we model using a separate set of parameters of the RBM involving $\boldsymbol{h}^{(1)}$ and $\boldsymbol{h}^{(2)}$. Thus, $p(\boldsymbol{h}^{(1)})$ is now the marginalization of the second hidden layer

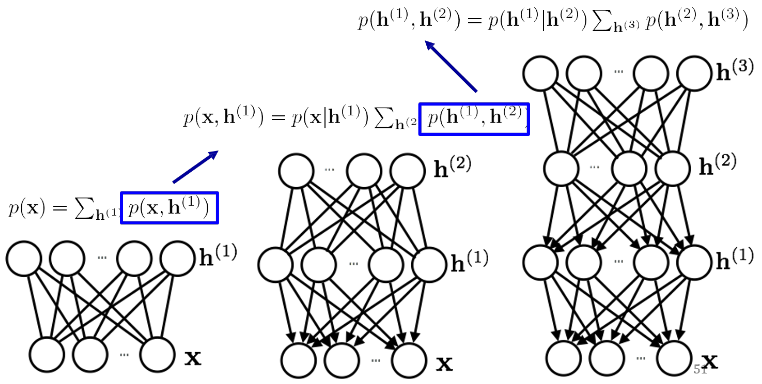
$$p(\boldsymbol{h}^{(1)}) = \sum_{\boldsymbol{h}^{(2)}} p(\boldsymbol{h}^{(1)}), \boldsymbol{h}^{(2)})$$

we can train the parameters of the new second layer by maximizing the bound. This is equivalent to minimizing the following, since the other terms are constant:

$$-\sum_{\boldsymbol{h}^{(1)}} q(\boldsymbol{h}^{(1)}|\boldsymbol{x}) \log p(\boldsymbol{h}^{(1)})$$

This is same as training an RBM on data generated from $q(\boldsymbol{h}^{(1)}|\boldsymbol{x})$. For $q(\boldsymbol{h}^{(1)}|\boldsymbol{x})$, we use the posterior of the first layer RBM. This is equivalent to a feed-forward (sigmoidal) layer, followed by sampling. Weights of the second layer RBM is initialized as the transpose of the first layer weights. Thus, the bound become initially tight! (As we showed, a 2-layer DBN with tied weights is equivalent to a 1-layer RBM). What need not keep being tight is as $p(\boldsymbol{h}^{(1)}$ changes, so does $p(\boldsymbol{h}^{(1)}|\boldsymbol{x})$, and so does the KL to $q(\boldsymbol{h}^{(1)}|\boldsymbol{x})$

Hence,RBM stacking procedure is improving prior on last layer by adding another hidden layer as below image


$p(\mathbf{h}^{(1)}, \mathbf{h}^{(2)}) = p(\mathbf{h}^{(1)}|\mathbf{h}^{(2)}) \sum_{\mathbf{h}^{(3)}} p(\mathbf{h}^{(2)}, \mathbf{h}^{(3)})$
$p(\mathbf{x}, \mathbf{h}^{(1)}) = p(\mathbf{x}|\mathbf{h}^{(1)}) \sum_{\mathbf{h}^{(2)}} p(\mathbf{h}^{(1)}, \mathbf{h}^{(2)})$
$p(\mathbf{x}) = \sum_{\mathbf{h}^{(1)}} p(\mathbf{x}, \mathbf{h}^{(1)})$
$\mathbf{h}^{(3)}$
$\mathbf{h}^{(2)}$
$\mathbf{h}^{(1)}$
$\mathbf{x}$

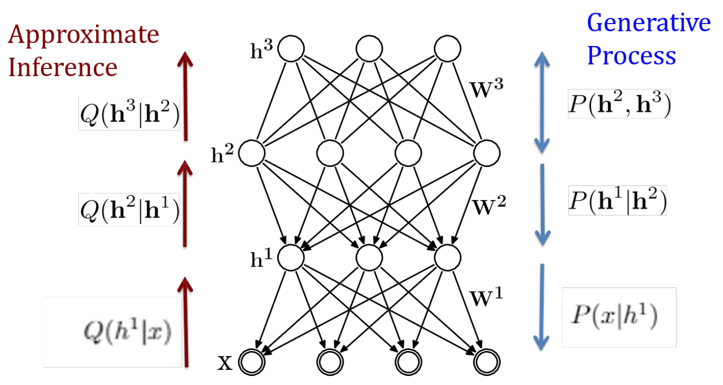
We can summarize the layer-wise learning and inference (See Figure 13.3) as follows:

• First step is learning an RBM W^1 with an input layer $\boldsymbol{x}$ and a hidden layer $\boldsymbol{h}$.

• Second step is treating inferred values $Q(\boldsymbol{h}^1|\boldsymbol{x}) = P(\boldsymbol{h}^1|\boldsymbol{x})$ as the data for training 2nd-layer RBM. After that, learn and freeze 2nd layer RBM.

• Then, proceed to the next layer. This process of adding layers can be recursively done.

Intuitively, when adding new layers, we obtain the greedy layer-wise pre-training procedure for neural networks. This procedure corresponds to maximizing a ELBO in the DBN.


Approximate Inference
Generative Process
$Q(\mathbf{h}^3|\mathbf{h}^2)$
$Q(\mathbf{h}^2|\mathbf{h}^1)$
$Q(h^1|x)$
$P(\mathbf{h}^2,\mathbf{h}^3)$
$P(\mathbf{h}^1|\mathbf{h}^2)$
$P(x|h^1)$
h³
h²
h¹
x
W³
W²
W¹

$$Q(\mathbf{h}^t|\mathbf{h}^{t-1}) = \prod_j \sigma\left(\sum_i W^t h_i^{t-1}\right) \qquad P(\mathbf{h}^{t-1}|\mathbf{h}^t) = \prod_j \sigma\left(\sum_i W^t h_i^t\right)$$

Figure 13.3: Layer-wise Learning and Inference structure of DBN

Even though in theory, approximation of $q(\boldsymbol{h}^{(1)}|\boldsymbol{x})$ is far from the true posterior, we could still extracting better features. Fine-tuning is done by the Up-Down algorithm which is a fast learning algorithm for deep belief nets. [Hinton, Teh, Osindero (2006)] proposed the following idea.

- To sample from the DBN model: $P(\boldsymbol{x}, \boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, \boldsymbol{h}^{(3)}) = P(\boldsymbol{x}|\boldsymbol{h}^{(1)})P(\boldsymbol{h}^{(1)}|\boldsymbol{h}^{(2)})P(\boldsymbol{h}^{(2)}|\boldsymbol{h}^{(3)})$
- Sample $\boldsymbol{h}^{(2)}$ using alternating Gibbs sampling from RBM. Sample lower layers using sigmoid belief network.

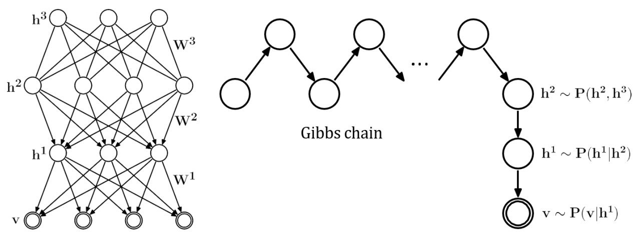


Figure 13.4: [Hinton, Teh, Osindero (2006)]