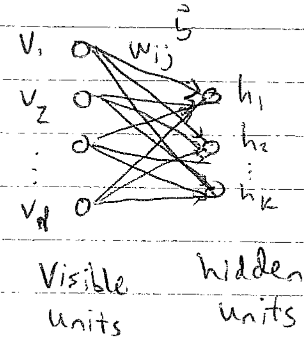
Deep Belief Networks

Recall a standard layer of an MLP:

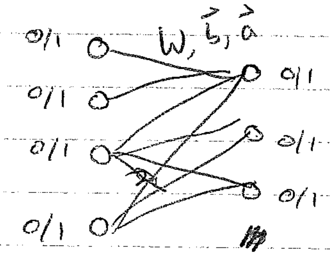


$d \times k$ weight matrix W

Sigmoid activation

$$h_i = \sigma(W_i^T v + b_i)$$

Restricted Boltzmann Machines



Standard RBM: assume visible and hidden units are <u>binary</u>.

Note $d+k$ units $\Rightarrow$ 2^{d+k} different settings of the variables.

Let $\Theta = \{W, \vec{b}, \vec{a}\}$

Define <u>energy function</u>: $E(v,h) = -\sum_{i=1}^{d} a_i v_i - \sum_{j=1}^{k} b_j h_j - \sum_{i=1}^{d}\sum_{j=1}^{k} v_i w_{ij} h_j$

$$= -\vec{a}^T\vec{v} - \vec{b}^T\vec{h} - \vec{v}^T W \vec{h}$$

Then write

$$p(v,h;\Theta) = \frac{1}{Z} \cdot e^{-E(v,h)} \propto e^{-E(v,h)}$$

$$Z = \sum_v \sum_h e^{-E(v,h)}$$ $\leftarrow$ sum has 2^{d+k} terms in it.

Once you have joint prob, can mess around with it:

$\cancel{p(v)}$ $p(v;\Theta) = \frac{1}{Z}\sum_h e^{-E(v,h)}$

$p(v|h;\theta)$ $p(h|v;\Theta)$

Not entirely trivial fact →

$$p(\vec{v}|\vec{h};\theta) = \prod_{i=1}^{d} p(v_i|h;\theta)$$

For example...

$$p\left(\begin{bmatrix}1\\1\\0\\1\end{bmatrix} \middle| \begin{bmatrix}0\\0\\1\end{bmatrix}\right) = p\left(v_1=1 \middle| \begin{bmatrix}0\\0\\1\end{bmatrix}\right) \cdot p\left(v_2=1 \middle| \begin{bmatrix}0\\0\\1\end{bmatrix}\right) \cdot p\left(v_3=0 \middle| \begin{bmatrix}0\\0\\1\end{bmatrix}\right) \cdot p\left(v_4=1 \middle| \begin{bmatrix}0\\0\\1\end{bmatrix}\right)$$

Similarly - $p(\vec{h}|\vec{v};\Theta) = \prod_{j=1}^{k} p(h_i|\vec{v};\Theta)$

And, further... $p(h_j=1|\vec{v}) = \sigma(w_j^T v + b_j)$

$$p(v_i=1|\vec{h}) = \sigma(h^T W_{:i} + a_i)$$

compare to a feed-forward network
- ~~output~~ output of sigmoid treated as an actual value, not as prob. value in FFN
- In the probabilistic case, we can "go backwards" from hidden to visible

↓

Can sample data once we know Θ and h!

See derivation page at end, or look in textbook for

How to learn parameters θ?
Maximum Likelihood!

For a single visible unit $\vec{v}$

$$p(\vec{v}) = \frac{1}{Z} \cdot \sum_h e^{-E(\vec{v},\vec{h})}$$

for a sequence of visible units $V = \vec{v}^{(1)}, \ldots, \vec{v}^{(n)}$:

$$p(V) = \prod_{p=1}^{n} \frac{1}{Z} \cdot \sum_h e^{-E(\vec{v}^{(p)},\vec{h})}$$

Treat this as a function of θ and maximize w.r.t. θ.

Take logs —

$$\log p(V) = \sum_{p=1}^{n} \left[\log\left(\sum_h e^{-E(\vec{v}^{(p)},h)}\right) - \log\left(\sum_v \sum_h e^{-E(v,h)}\right) \right]$$

Z is not a "constant" w.r.t. θ! Problematic!

~~1st term~~:

$$\frac{\partial}{\partial w_{ij}} E(v,h) = ? \quad -v_i h_j$$

1st Term of Log Likelihood:

$$\frac{\partial}{\partial w_{ij}} \sum_{i=1}^{n} \log\left(\sum_h e^{-E(\vec{v}^{(i)},h)}\right)$$

$$= \sum_{p=1}^{n} \frac{\partial}{\partial w_{ij}} \log\left(\sum_h e^{-E(\vec{v}^{(p)},\vec{h})}\right)$$

$$= \sum_{p=1}^{n} \frac{-1}{\sum_h e^{-E(\vec{v}^{(p)},h)}} \cdot \sum_h \left[e^{-E(\vec{v}^{(p)},h)} \cdot \frac{\partial}{\partial w_{ij}} E(\vec{v}^{(p)},\vec{h}) \right]$$

Circled thing is exactly $-p(\vec{h} | \vec{v}^{(p)})$:

$$p(\vec{h} | \vec{v}^{(p)}) = \frac{p(\vec{v}^{(p)}, \vec{h})}{p(\vec{v})} = \frac{\frac{1}{Z} \cdot e^{-E(\vec{v}^{(i)}, \vec{h})}}{\frac{1}{Z} \cdot \sum_h e^{-E(v^{(i)}, h)}}$$

So... 1st term is exactly

$$\sum_{p=1}^{n} \sum_{\vec{h}} p(\vec{h} | \vec{v}^{(p)}) \cdot v_i h_j$$ ← This is an expectation of $v_i h_j$ wrt. $p(\vec{h} | v^{(p)})$

Similarly, 2nd term is

$$- n \sum_{\vec{v}, \vec{h}} p(\vec{v}, \vec{h}) \cdot v_i h_j$$ ← also an expectation but wrt. joint probability

Scale by n.

Hinton writes this as

$$E_{data}(v_i h_j) - E_{model}(v_i h_j)$$

GD. would operate as

$$W_{ij} \leftarrow W_{ij} - \epsilon \left[E_{data}(v_i h_j) - E_{model}(v_i h_j) \right]$$

Question is how to compute these quantities. Short answer – you cannot.

We can get an unbiased sample of this value. How?

Given $\vec{v}^{(p)}$, sampling $p(\vec{h} | \vec{v}^{(p)})$ and then multiplying by $v_i h_j$ gives you an unbiased sample.

Sample h_j via $p(h_j = 1 | v^{(p)})$ ~~and mult.~~ and multiply by v_i. Average over all $\vec{v}^{(p)}$.

Now need sample of $E_{model}(v_i h_j)$. This is a little trickier.
Need to sample from $p(v,h)$.

Gibbs Sampling: To sample from a distribution $p(v,h)$, repeat the following:

Start with some value of ~~[illegible]~~ v (in our case, $\vec{v}^{(p)}$ is known)

Sample new h from $p(h|v)$

~~[illegible]~~ Sample new v from $p(v|h)$

repeat for a "long time"

Eventually v and h given by this algorithm will be a sample from $p(v,h)$

[Need to establish 2 things — i. that once (v,h) samples are from the true $p(v,h)$, ~~the~~ any subsequent ~~step~~ samples are also from $p(v,h)$; ii. irrespective of the initial choice of v, the distribution of (v,h) approaches the desired distribution $p(v,h)$.]

This is still slow, unfortunately. Hinton introduced a further simplification (contrastive divergence) that only takes a single step. So, the full alg. looks like this:

Average over a minibatch:

- pick a visible point $v^{(p)}$
- Sample $\mathbf{h}$ via $p(h|v^{(p)})$
- Sample $\tilde{v}$ via $p(\tilde{v}|h)$
- $\Delta w_{ij} = v_i^{(p)} \cdot h_j - \tilde{v}_i \cdot h_j$

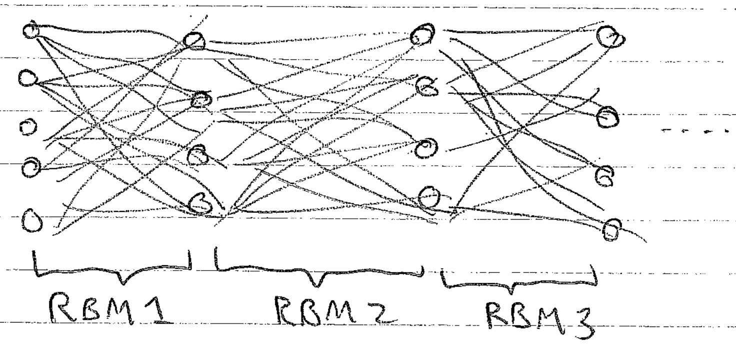
[Sometimes, instead of ~~p~~ $\tilde{v}$ use the actual probabilities $p(\tilde{v}|h)$ when computing Δw_{ij}.]

Note: this procedure is known to not converge to a local max of the likelihood.

Next Step: Deep Belief Networks

Idea: Stack several layers of RBMs on top of one another, and train layer-by-layer.

Picture:



Idea is this .. train weights of RBM 1 using visible/training data.

- Freeze weights of ~~RBM~~ RBM1.
- Treat the hidden units of RBM1 as the visible units of RBM2.
- Pass each training pt through RBM1, sampling the visible units for RBM2.
- Train RBM2 using these sampled data.
- Freeze weights of RBM2.
- etc... "Pretraining"

Next, use the weights learned by this procedure as an initialization to a feedforward sigmoidal network.

"Fine-tuning": Run SGD with supervised data

Extensions to basic RBMs

What if weights are non-binary?

Extension 1: binary $\rightarrow$ 1-of-k

For binary, we have something like

$$\sigma(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$$

Generalize to 1-of-k via softmax units

$$p_j = \frac{e^{x_j}}{\sum_{i=1}^{k} e^{x_i}}$$

Don't really have to change ~~anything~~ much at all. Energy function same. Only changes how samples are taken and probabilities computed.

$$p(h_j = 1 \mid \vec{v}; \theta) = \sigma(\vec{v}^T w_j + b_j)$$

$$\Rightarrow \quad p(h_j = c \mid \vec{v}; \theta) = \frac{e^{v^T w_j + b_j}}{\sum_{j=1}^{k} e^{v^T w_j + b_j}}$$

Real visible units / Binary hidden units

$$E(v,h) = \sum_{i \in vis} \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_{j \in hid} b_j h_j - \sum_{i,j} \frac{v_i}{\sigma_i} h_j w_{ij}$$

Usual = assume data is whitened first, so $\sigma_i = 1$

Then: we have

$$p(h_j = 1 \mid \vec{v}; \Theta) = \text{same as before}$$
$$= \sigma(\mathbf{v}^T w_j + b_j)$$

$$p(v_i = 1 \mid \vec{h}; \Theta) \sim \ ?$$

Gaussian distribution - Exercise:

What is the precise Gaussian?
(Need to compute for HW 5.)

Then G.D. works as follows:

Average over minibatch:

- Given $v^{(p)}$ sample h via $p(h \mid v^{(p)})$ (sigmoid)
- Given h sample $\tilde{v}$ via $p(\tilde{v} \mid h)$ (Gaussian)
- $\Delta w_{ij} = v_i^{(p)} h_j - \tilde{v}_i h_j$

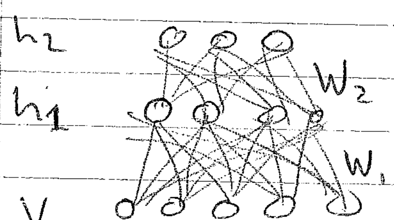
~~Convolutional Deep Belief Networks (Lee et al.)~~

Deep Boltzmann Machines

- When designing a joint probability distribution over a deep network, stacks of RBMs are not the cleanest solution.
  - For instance, they don't have a simple joint distribution

Alternative - Deep Boltzmann Machine

Suppose there are 2 hidden layers



In a DBM, define

$$E(v, h^1, h^2; \Theta) = -v^T W_1 h_1 - h_1^T W_2 h_2$$

i.e. $$p(v, \Theta) = \frac{1}{Z} \sum_{h_1, h_2} \exp(-E(v, h_1, h_2; \Theta))$$

$$p(h_j^1 = 1 \mid v, h^2) = \sigma\left(\sum_i W_{ij}^1 v_i + \sum_m W_{jm}^2 h_j^2\right)$$

$$p(h_m^2 = 1 \mid h^1) = \sigma\left(\sum_j W_{jm}^2 h_j^1\right)$$

$$p(v_i = 1 \mid h^1) = \sigma\left(\sum_j W_{ij}^1 h_j\right)$$

Can use sample-based methods for parameter learning, but can also use <u>Variational inference</u> (won't discuss)

Some Derivations

$$p(h|v;\theta) = \frac{p(v,h;\theta)}{p(v;\theta)}$$

$$= \frac{1}{p(v;\theta)} \cdot \frac{1}{Z} \cdot e^{a^T v + b^T h + v^T W h}$$

$$\propto e^{b^T h + v^T W h}$$

$W_{:j} = W_j$

$$= \exp\left(\sum_{j=1}^{k} b_j h_j + \sum_{j=1}^{k} v^T W_{:j} \cdot h_j \right)$$

$$= \prod_{j=1}^{k} \underbrace{\exp\left(b_j h_j + v^T W_{:j} \cdot h_j \right)}_{\tilde{p}(h_j=1|\vec{v};\theta)}$$

Follows that $p(h|v;\theta)$ will actually factorize over the hidden units. In particular,

$$p(h_j=1|\vec{v};\theta) = \frac{\tilde{p}(h_j=1|\vec{v};\theta)}{\tilde{p}(h_j=0|\vec{v};\theta) + \tilde{p}(h_j=1|\vec{v};\theta)}$$

$$= \frac{\exp(b_j + v^T W_{:j})}{1 + \exp(b_j + \vec{v}^T W_{:j})}$$

$$= \sigma(w_j^T \vec{v} + b_j)$$

Now pull out $e^{a^T v}$ terms on num/denom

Another way: $\frac{p(v,h;\theta)}{p(v;\theta)} = \frac{\frac{1}{Z} \cdot e^{-E(v,h)}}{\frac{1}{Z} \cdot \sum_h e^{-E(v,h)}} = \frac{e^{-E(v,h)}}{\sum_h e^{-E(v,h)}}$