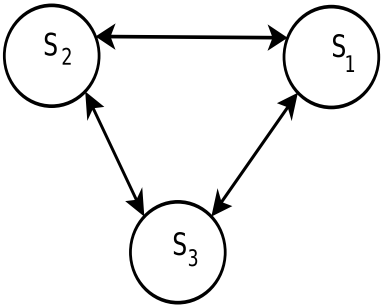
Restricted Boltzmann Machine (RBM)

- Recurrent Neural Network with stochastic units
- A particular case of undirected graphical model.
- Belongs to what are often called energy-based models.
- A generative model (representing a probability distribution).
- Suitable for unsupervised learning.
- Can be used as a discriminative model too.

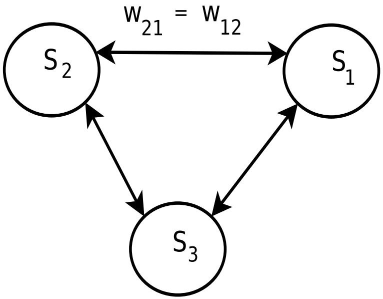
Hopfield Network

- A fully connected recurrent neural network
- Binary units/neurons



Hopfield Network

- Output (state) of i^{th} neuron: $s_i \in \{0,\ 1\}$
- w_{ij} – weight between i and j
  We assume: $w_{ij} = w_{ji}, \quad w_{ii} = 0$



Hopfield Network – Dynamics

- Consider n neurons with outputs (states) $s_1(k), \cdots, s_n(k)$ at time k.
- At each k, only one randomly chosen neuron changes output.

$$s_i(k+1) = 1 \;\text{ iff }\; \sum_j w_{ij} s_j(k) + b_i > 0$$

- State of network: $\mathrm{s}(k) = (s_1(k) \;\cdots\; s_n(k))$
- $\mathrm{s}(k)$ evolves over $\{0,\ 1\}^n$.

- Define energy, $E : \{0,\ 1\}^n \rightarrow \Re$

$$E(\mathbf{s}) = -\frac{1}{2}\sum_{i,j} w_{ij} s_i s_j - \sum_i b_i s_i$$

- The dynamics of Hopfield network ensures that

$$E(\mathbf{s}(k+1)) - E(\mathbf{s}(k)) \leq 0,\ \forall k$$

- Dynamics converges to a minimum of energy.
- From current point we move to a 'neighbour' point in $\{0,\ 1\}^n$, seeking to minimize energy.

Hopfield Network seeks minimum of energy

$$
\begin{aligned}
E(\mathbf{s}) &= -\frac{1}{2}\sum_{q,l} w_{ql} s_q s_l - \sum_q b_q s_q \\
&= -\frac{1}{2}\sum_{q,l\neq i} w_{ql} s_q s_l - s_i \sum_{q\neq i} w_{iq} s_q - \sum_{q\neq i} b_q s_q - b_i s_i
\end{aligned}
$$

- Suppose: $s_j(k+1) = s_j(k),\ \forall j \neq i$.
- Now we can write

$$
E(\mathbf{s}(k+1) - E(\mathbf{s}(k)) = (s_i(k) - s_i(k+1)) \left(\sum_{q\neq i} w_{iq} s_q(k) + b_i \right)
$$

- The dynamics: $s_i(k+1) = 1,$ iff $\sum_{q\neq i} w_{iq} s_q(k) + b_i \geq 0$.
- Hence $E(\mathbf{s}(k+1) - E(\mathbf{s}(k)) \leq 0$.

- Hopfield network is one of the very influential early recurrent network models.
- Its dynamics seeks minima in an energy landscape
- Can be used as an optimizer.
- Can be used as associative memory (or for pattern completion) – a ‘generative’ model

Boltzmann Machine

- Suppose we make the dynamics stochastic:
  at each instant, only one randomly selected neuron changes state; but the change is stochastic

$$\text{Prob}[s_i(k+1) = 1] = \frac{1}{1 + \exp(-\{\sum_j w_{ij} s_j(k) + b_i\})}$$

- Now, $\mathbf{s}(k) = (s_1(k) \ \cdots \ s_n(k))$ would be a Markov chain over the state space $\{0, \ 1\}^n$.
- The transition structure is such that two states communicate iff they differ in only one coordinate.

- As before define energy by

$$E(\mathbf{s}) = -\frac{1}{2}\sum_{i,j} w_{ij} s_i s_j - \sum_i b_i s_i$$

- If s, s' differ only in i^{th} component, then

$$E(\mathbf{s}) - E(\mathbf{s}') = \left(\sum_{j \neq i} w_{ij} s_j + b_i\right)(s_i' - s_i)$$

- Let $\mathbf{s}^1, \mathbf{s}^0 \in \{0,\ 1\}^n$ with $s_j^1 = s_j^0, \forall j \neq i$ and $s_i^1 = 1, s_i^0 = 0$.
- Going from $\mathbf{s}^0$ to $\mathbf{s}^1$ involves choosing i^{th} neuron and setting it to 1. Hence this transition probability is

$$\begin{aligned}
P(\mathbf{s}^0, \mathbf{s}^1) &= \frac{1}{n} \frac{1}{(1 + \exp(-\{\sum_j w_{ij} s_j^0(k) + b_i\}))} \\
&= \frac{1}{n} \frac{1}{(1 + \exp((s_i^0 - s_i^1) \left(\{\sum_j w_{ij} s_j^0(k) + b_i\}\right)))} \\
&= \frac{1}{n} \frac{1}{(1 + \exp(E(\mathbf{s}^1) - E(\mathbf{s}^0)))}
\end{aligned}$$

- Going from $\mathbf{s}^1$ to $\mathbf{s}^0$ involves choosing i^{th} neuron and setting it to 0.
- Noting that $1 - \frac{1}{1+e^{-x}} = \frac{1}{1+e^{x}}$

$$\begin{aligned}
P(\mathbf{s}^1, \mathbf{s}^0) &= \frac{1}{n}\frac{1}{(1+\exp(\{\sum_j w_{ij} s_j^0(k) + b_i\}))} \\
&= \frac{1}{n}\frac{1}{(1+\exp((s_i^1 - s_i^0)\left(\{\sum_j w_{ij} s_j^0(k) + b_i\}\right)))} \\
&= \frac{1}{n}\frac{1}{(1+\exp(E(\mathbf{s}^0) - E(\mathbf{s}^1)))}
\end{aligned}$$

- Transitions are only to states differing in exactly one coordinate
- The transition structure is

$$P(\mathbf{s}, \mathbf{s}') = \frac{1}{n} \; \frac{1}{(1 + \exp(E(\mathbf{s}') - E(\mathbf{s})))}$$

- The chain prefers going to lower energy states.
- This Markov chain is finite, irreducible, aperiodic and hence ergodic.
- The stationary distribution is given by

$$p(\mathbf{s}) = \frac{1}{Z} \exp(-E(\mathbf{s}))$$

The stationary distribution of this chain

- if p satisfies

$$p(\mathbf{s}')P(\mathbf{s}',\mathbf{s}) = p(\mathbf{s})P(\mathbf{s},\mathbf{s}'), \quad \text{for all states} \quad \mathbf{s},\mathbf{s}'$$

  then it is the stationary distribution

- We can easily see that

$$\frac{\exp(-E(\mathbf{s}'))}{1+\exp(E(\mathbf{s})-E(\mathbf{s}'))} = \frac{\exp(-E(\mathbf{s}))}{1+\exp(E(\mathbf{s}')-E(\mathbf{s}))}, \forall \mathbf{s},\mathbf{s}'$$

- Hence, the stationary distribution is

$$p(\mathbf{s}) = \frac{1}{Z}\exp(-E(\mathbf{s}))$$

  where $Z = \sum_{\mathbf{s}} \exp(-E(\mathbf{s}))$ is the normalizing constant.

Energy based Model

- The boltzman machine is a generative model.
- It represents a distribution over $\{0,\ 1\}^n$ given by

$$p(\mathbf{s}) = \frac{1}{Z}\exp(-E(\mathbf{s}))$$

where Z is the normalizing constant and

$$E(\mathbf{s}) = -\frac{1}{2}\sum_{ij} w_{ij}s_i s_j - \sum_i b_i s_i$$

- Each configuaration (point in $\{0,\ 1\}^n$) has an energy which determines its probability.

- Boltzmann machine can also be used as a discriminative model.
- If we hold some neurons at some values, then at steady state what we see at the other neurons is the conditional distribution.
- This is how we can use it as classifier.

- We have a generative model: $p(\mathbf{s})$.
- We learn this probability distribution from given data.
- We can use it as a discriminative model by designating some components of $\mathbf{s}$ as ‘input’ and some as ‘output’
- However, we learn the probability distribution through Unsupervised learning – ‘treat all variables uniformly’

Need for hidden units

- We learn a generative model: $p(\mathbf{s})$:

$$p(\mathbf{s}) = \frac{1}{Z} \exp(-E(\mathbf{s}))$$

where Z is the normalizing constant and

$$E(\mathbf{s}) = -\frac{1}{2} \sum_{ij} w_{ij} s_i s_j - \sum_i b_i s_i$$

- The probability depends on energy which is quadratic
- We can essentially model only upto second-order statistics.
- Hence we need hidden nodes to learn 'proper representation'

Need for hidden units

- Let $n = 3$.
- Let distribution p_1: uniform over $\{0,\ 1\}^3$.
- Let distribution p_2: uniform only over the four tuples: $(0,0,0),(0,1,1),(1,0,1),(1,1,0)$. (The other four tuples have zero probability)
- But p_1 and p_2 have identical statistics upto second order. ($P[s_i s_j = 1]$, $P[s_i = 1]$ are same under both)
- We can not distinguish between p_1 and p_2 with our current formulation
- Note that p_2 models (or represents) the XOR function

Boltzmann Machine

- We would have $(m+n)$ units: $v_1, \cdots, v_m, h_1, \cdots, h_n$.
- The v_i are called visible and the h_i are called hidden.
- The formulation is still the same. The network is fully connected. (Simply some of the s_i are visible while others are hidden).
- The Energy is defined the same way and the machine represents a distribution

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} \exp(-E(\mathbf{v}, \mathbf{h}))$$

- But we are only interested in the distribution on visible units which is given by

$$p(\mathbf{v}) = \sum_{\mathbf{h}} \frac{1}{Z}\ \exp(-E(\mathbf{v}, \mathbf{h}))$$

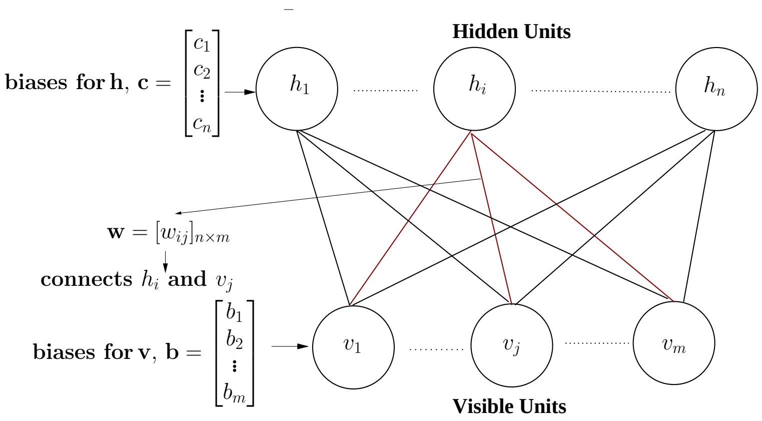
- Training data would be samples of visible units only
- The hidden units are there to help us realize the needed distribution on the visible units.
- This is the original Boltzmann machine proposed.

- The weights in Boltzmann machine are learnt (or estimated) through maximum likelihood estimation.
- But training is slow and computationally intensive
- A conditional independence assumption makes the model more tractable.
- That is the Restricted Boltzmann Machine

Restricted Boltmann Machine

- As earlier we have m visible and n hidden units: $v_1, \cdots, v_m, h_1, \cdots, h_n$.
- But connections are only between visible and hidden units; no connections from visible to visible or hidden to hidden.
- Notation:
  w_{ij} is weight of connection between v_j and h_i
  b_i is bias for v_i
  c_i is bias for h_i

RBM



Hidden Units
biases for h, $\mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$
h_1
h_i
h_n
$\mathbf{w} = [w_{ij}]_{n\times m}$
connects h_i and v_j
biases for v, $\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$
v_1
v_j
v_m
Visible Units

RBM parameter, $\theta = \{\mathbf{w}, \mathbf{b}, \mathbf{c}\}$

- Now the energy is given by

$$\begin{aligned} E(\mathbf{v},\mathbf{h}) &= -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}h_i v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i \\ &= -\mathbf{h}^T W\mathbf{v} - \mathbf{b}^T\mathbf{v} - \mathbf{c}^T\mathbf{h} \end{aligned}$$

- The probability distribution, over $\{0,\ 1\}^{(m+n)}$, represented by the RBM is

$$p(\mathbf{v},\mathbf{h}) = \frac{1}{Z}\exp(-E(\mathbf{v},\mathbf{h}))$$

Conditional Independence

- Consistent with the interconnections, we get the following conditional independence:

$$p(\mathbf{h}|\mathbf{v}) = \prod_i p(h_i|\mathbf{v}); \quad p(\mathbf{v}|\mathbf{h}) = \prod_j p(v_j|\mathbf{h})$$

- The conditional probabilities are given by

$$\begin{aligned}\text{Prob}[h_i = 1|\mathbf{v}] &= \text{sig}\left(\sum_{j=1}^{m} w_{ij} v_j + c_i\right) \\ \text{Prob}[v_i = 1|\mathbf{h}] &= \text{sig}\left(\sum_{j=1}^{n} w_{ji} h_j + b_i\right)\end{aligned}$$

where $\text{sig}(x) = 1/(1 + \exp(-x))$.

- We have

$$\begin{aligned}
\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} &= \sum_{\mathbf{h}} e^{\mathbf{b}^T\mathbf{v}} e^{\sum_{i=1}^{n} h_i(c_i+\sum_{j=1}^{m} w_{ij}v_j)} \\
&= e^{\mathbf{b}^T\mathbf{v}} \sum_{\mathbf{h}} \prod_{i=1}^{n} e^{h_i(c_i+\sum_{j=1}^{m} w_{ij}v_j)} \\
&= e^{\mathbf{b}^T\mathbf{v}} \sum_{h_1} e^{h_1(c_i+\sum_{j=1}^{m} w_{ij}v_j)} \dots \sum_{h_n} e^{h_n(c_i+\sum_{j=1}^{m} w_{ij}v_j)} \\
&= e^{\mathbf{b}^T\mathbf{v}} \prod_{i=1}^{n} (1 + e^{c_i+\sum_{j=1}^{m} w_{ij}v_j})
\end{aligned}$$

- This gives us an expression for the marginal distribution on the visible units

$$\begin{aligned} p(\mathbf{v}) &= \frac{1}{Z} \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \\ &= \frac{1}{Z}\, e^{\mathbf{b}^T\mathbf{v}} \prod_{i=1}^{n} (1 + e^{c_i + \sum_{j=1}^{m} w_{ij} v_j}) \end{aligned}$$

- This also gives us the conditional distribution

$$\begin{aligned} p(\mathbf{h}|\mathbf{v}) &= \frac{e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}} \\ &= \frac{e^{\mathbf{b}^T\mathbf{v}} e^{\sum_{i=1}^{n} h_i(c_i+\sum_{j=1}^{m} w_{ij}v_j)}}{e^{\mathbf{b}^T\mathbf{v}} \prod_{i=1}^{n}(1+e^{c_i+\sum_{j=1}^{m} w_{ij}v_j})} \\ &= \prod_{i=1}^{n} \frac{e^{h_i(c_i+\sum_{j=1}^{m} w_{ij}v_j)}}{(1+e^{c_i+\sum_{j=1}^{m} w_{ij}v_j})} \end{aligned}$$

- Shows that h_i are conditinally independent given $\mathbf{v}$

- Now we get

$$\begin{aligned} p(h_i|\mathbf{v}) &= \sum_{h_j, j \neq i} \prod_{j=1}^{n} \frac{e^{h_j(c_j + \sum_{q=1}^{m} w_{jq} v_q)}}{(1 + e^{c_j + \sum_{q=1}^{m} w_{jq} v_q})} \\ &= \frac{e^{h_i(c_i + \sum_{q=1}^{m} w_{iq} v_q)}}{(1 + e^{c_i + \sum_{q=1}^{m} w_{iq} v_q})} \end{aligned}$$

- This gives us

$$\text{Prob}[h_i = 1|\mathbf{v}] = \text{sig}\left(c_i + \sum_{j=1}^{m} w_{ij} v_j\right)$$

- A similar expression holds for Prob$[v_i = 1|\mathbf{h}]$

Summary of RBM model

- The RBM is a generative model with m visible and n hidden nodes:

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} \exp(-E(\mathbf{v}, \mathbf{h}))$$

where the energy is given by

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{h}^T W \mathbf{v} - \mathbf{b}^T \mathbf{v} - \mathbf{c}^T \mathbf{h}$$

and the normalizing constant, Z, is given by

$$Z = \sum_{\mathbf{v}, \mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}))$$

Summary $\cdots$

- The following conditional independences hold:

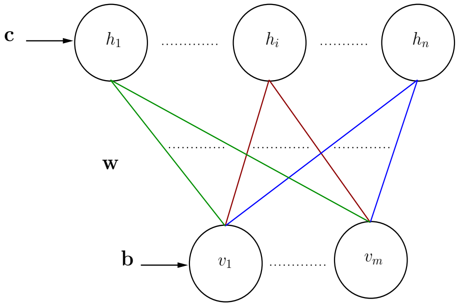
$$p(\mathbf{h}|\mathbf{v}) = \prod_i p(h_i|\mathbf{v}); \quad p(\mathbf{v}|\mathbf{h}) = \prod_j p(v_j|\mathbf{h})$$

- The conditional probabilities are

$$\begin{aligned} \text{Prob}[h_i = 1|\mathbf{v}] &= \text{sig}\left(\sum_{j=1}^{m} w_{ij}v_j + c_i\right) \\ \text{Prob}[v_i = 1|\mathbf{h}] &= \text{sig}\left(\sum_{j=1}^{n} w_{ji}h_j + b_i\right) \end{aligned}$$

RBM as graphical model

- RBM can be specified as an undirected graphical model.



- Any two hidden nodes are conditionally independent given all visible nodes and vice versa

Recall: Undirected graphical models

- Suppose a generative model for $\mathbf{X} = (X_1, \cdots, X_n)$ is an undirected graphical model. Let $\mathcal{C}$ be the set of maximal cliques.
- Then the joint distribution should be of the form

  $$p(\mathbf{x}) = \frac{1}{Z}\, e^{\sum_c \psi_c(\mathbf{x})}$$

  where the ψ_c are called clique potentials.
- $\psi_c(\mathbf{x})$ depends only on those components of $\mathbf{x}$ that are in c.
  (Hammersley-Clifford Theorem)

Case of RBM

- Thus every MRF is an energy-based model with energy function given by

$$E(\mathbf{x}) = -\sum_{c \in \mathcal{C}} \psi_c(\mathbf{x})$$

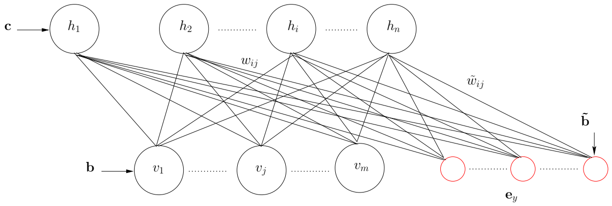
- In an RBM, the cliques are $\{v_i, h_j\}$. Since these are binary, a general energy function would have terms involving v_i, h_j and $v_i h_j$
- The graphical model comes from conditional independence we imposed
- For the given graphical model, our choice of energy function is very general.

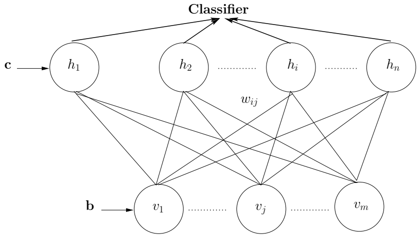
Representational Power of RBMs

- As said earlier, we are only interested in modelling the distribution over visible units.
- The hidden units are there only so that we can represent the desired distribution.
- Question: What distributions over $\{0,\ 1\}^m$ are representable by an RBM with m visible nodes (given sufficient number of hidden nodes)
- Given $n = 2^m + 1$ hidden nodes, RBM can represent any distrubution over $\{0,\ 1\}^m$.
- If the target distribution is supported on fewer terms, then we can do with lesser number of hidden nodes.

RBM as a discriminative model

- We can use RBM as a classifier also.
- We can take $\mathbf{v} \in \{0,\ 1\}^m$ as 'feature vector' and take $\mathbf{y} \in \{0,\ 1\}^K$ as class label. (Note $\mathbf{y}$ takes only K distinct values in a K-class problem)
- For RBM, $(\mathbf{v}, \mathbf{y})$ together make all the visible units and we can have n hidden units.


$\mathbf{c}$
h_1
h_2
h_i
h_n
w_{ij}
$\tilde{w}_{ij}$
$\tilde{\mathbf{b}}$
$\mathbf{b}$
v_1
v_j
v_m
$\mathbf{e}_y$


Classifier
$\mathbf{c}$
h_1
h_2
h_i
h_n
w_{ij}
$\mathbf{b}$
v_1
v_j
v_m

- Now the training data would be labelled samples.
- We would learn the distribution of $(\mathbf{v}, \mathbf{y})$

- Let
  w_{ij} be weight between h_i and v_j
  $\tilde{w}_{ij}$ be weight between y_j and h_i.
  b_i be bias for v_i, $\tilde{b}_i$ for y_i and c_i for h_i
- Now we can write the probability model of RBM as

$$p(\mathbf{v}, \mathbf{y}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v},\mathbf{y},\mathbf{h})}$$

  where the energy is given by

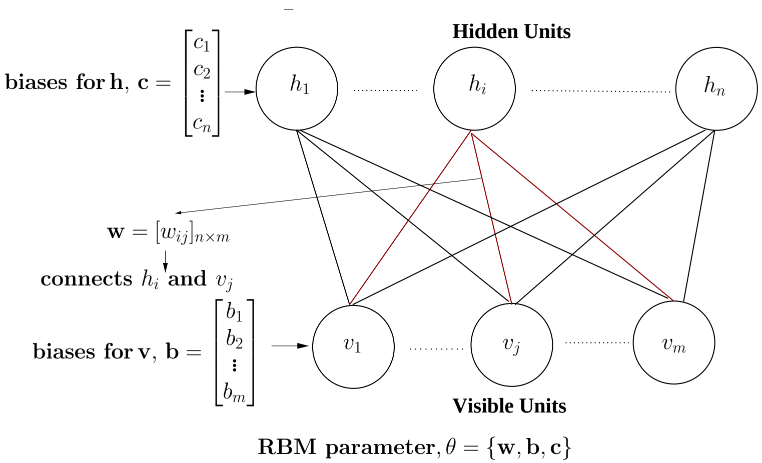
$$E(\mathbf{v}, \mathbf{y}, \mathbf{h}) = -\mathbf{h}^T W \mathbf{v} - \mathbf{b}^T \mathbf{v} - \mathbf{c}^T \mathbf{h} - \mathbf{h}^T \tilde{W} \mathbf{y} - \tilde{b}^T \mathbf{y}$$

- As mentioned earlier $\mathbf{y}$ takes only the K unit vectors as values.
- After learning all the weights we need to predict $\mathbf{y}$ given $\mathbf{v}$.
- We calculate

$$p(\mathbf{y}|\mathbf{v}) = \frac{\sum_{\mathbf{h}} p(\mathbf{v}, \mathbf{y}, \mathbf{h})}{\sum_{\mathbf{h},\mathbf{y}} p(\mathbf{v}, \mathbf{y}, \mathbf{h})}$$

  for all the unit vectors and take the one with highest probability.

Learning RBM


Hidden Units
biases for $\mathbf{h}$, $\mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$
h_1
h_i
h_n
$\mathbf{w} = [w_{ij}]_{n \times m}$
connects h_i and v_j
biases for $\mathbf{v}$, $\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$
v_1
v_j
v_m
Visible Units
RBM parameter, $\theta = \{\mathbf{w}, \mathbf{b}, \mathbf{c}\}$

Need to learn parameters from training data.

- Given some value for parameters, θ, RBM represents a probability density $p_\theta(\mathbf{v})$.
- Given *iid* training data from some distribution, we want to learn a parametric model of the distribution.
- Hence we can use ML estimation of θ.

- The training data: $\mathcal{D} = \{\mathbf{v}^1, \mathbf{v}^2, \cdots \mathbf{v}^M\}$
- The data likelihood is

$$\mathcal{L}(\theta|\mathcal{D}) = \prod_{k=1}^{M} p_\theta(\mathbf{v}^k)$$

- The log-likelihood is given by

$$\ell(\theta) = \ln(\mathcal{L}(\theta)) = \sum_{k=1}^{M} \ln(p_\theta(\mathbf{v}^k))$$

- We can minimize the negative log-likelihood (NLL) to get the ML estimate.
- (Recall that) Minimizing NLL is same as minimizing KL divergence between p_{data} and p_θ

- For simplicity of notation, assume we have only one training sample, $\mathbf{v}^1$.
- The log-likelihood is

$$\ell(\theta) = \ln(p_\theta(\mathbf{v}^1)) = \ln\left(\frac{1}{Z}\sum_{\mathbf{h}} e^{-E(\mathbf{v}^1,\mathbf{h})}\right)$$

- We want to do gradient descent on negative log-likelihood.

$$\frac{\partial \ell(\theta)}{\partial \theta} = \frac{\partial}{\partial \theta}\left(\ln\left(\sum_{\mathbf{h}} e^{-E(\mathbf{v}^1,\mathbf{h})}\right) - \ln(Z)\right)$$

- we have

$$\frac{\partial \ell(\theta)}{\partial \theta} = \frac{\partial}{\partial \theta}\left(\ln\left(\sum_{\mathbf{h}} e^{-E(\mathbf{v}^1,\mathbf{h})}\right) - \ln(Z)\right)$$

- The parameter vector, θ, consists of all weights, w_{ij} and the biases b_i and c_i.
- The above is a vector equation with one derivative for each component of parameter vector.
- Let us write $\frac{\partial}{\partial w}$ for any one component. (w may be w_{ij}, b_i or c_i).

- Noting that $Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}$, we have

$$\begin{aligned}
\frac{\partial \ell(\theta)}{\partial w} &= \frac{\partial}{\partial w} \ln \left(\sum_{\mathbf{h}} e^{-E(\mathbf{v}^1,\mathbf{h})} \right) - \frac{\partial}{\partial w} \ln(Z) \\
&= \frac{\sum_{\mathbf{h}} \frac{\partial e^{-E(\mathbf{v}^1,\mathbf{h})}}{\partial w}}{\sum_{\mathbf{h}} e^{-E(\mathbf{v}^1,\mathbf{h})}} - \frac{\sum_{\mathbf{v},\mathbf{h}} \frac{\partial e^{-E(\mathbf{v},\mathbf{h})}}{\partial w}}{Z} \\
&= f(\theta) - g(\theta), \text{ (say)}
\end{aligned}$$

- The first term can be simplified as follows

$$
\begin{aligned}
f(\theta) &= \sum_{\mathbf{h}} \frac{\frac{\partial e^{-E(\mathbf{v}^1,\mathbf{h})}}{\partial w}}{\sum_{\mathbf{h}} e^{-E(\mathbf{v}^1,\mathbf{h})}} \\
&= \sum_{\mathbf{h}} \left(\frac{e^{-E(\mathbf{v}^1,\mathbf{h})}}{\sum_{\mathbf{h}} e^{-E(\mathbf{v}^1,\mathbf{h})}} \; \frac{\partial(-E(\mathbf{v}^1,\mathbf{h}))}{\partial w} \right) \\
&= \sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v}^1) \; \frac{\partial(-E(\mathbf{v}^1,\mathbf{h}))}{\partial w} \\
&= \mathcal{E}_{\mathbf{h}|\mathbf{v}^1} \left[\frac{\partial(-E(\mathbf{v}^1,\mathbf{h}))}{\partial w} \right]
\end{aligned}
$$

where $\mathcal{E}_{\mathbf{h}|\mathbf{v}^1}$ denotes expectation with respect to the conditional distribution of $\mathbf{h}$ given $\mathbf{v}$ at $\mathbf{v} = \mathbf{v}^1$ (and at the current values of parameters).

- Now let us look at the second term in gradient of log-likelihood

$$
\begin{aligned}
g(\theta) &= \frac{\sum_{\mathbf{v},\mathbf{h}} \frac{\partial e^{-E(\mathbf{v},\mathbf{h})}}{\partial w}}{Z} \\
&= \sum_{\mathbf{v},\mathbf{h}} \frac{e^{-E(\mathbf{v},\mathbf{h})}}{Z} \; \frac{\partial(-E(\mathbf{v},\mathbf{h}))}{\partial w} \\
&= \sum_{\mathbf{v},\mathbf{h}} p(\mathbf{v},\mathbf{h}) \frac{\partial(-E(\mathbf{v},\mathbf{h}))}{\partial w} \\
&= \mathcal{E}_{\mathbf{v},\mathbf{h}} \left[\frac{\partial(-E(\mathbf{v},\mathbf{h}))}{\partial w} \right]
\end{aligned}
$$

where $\mathcal{E}_{\mathbf{v},\mathbf{h}}$ denotes expectation with respect to the joint distribution of $(\mathbf{v},\mathbf{h})$ (at the current parameter values, θ).

- In the above we have taken only one example and set $\ell(\theta) = \ln(p(\mathbf{v}^1))$.
- But we have M samples and hence have to take sum over them for log-likelihood.
- We need to maximize

$$\ell(\theta) = \frac{1}{M}\sum_{k=1}^{M} \ln(p(\mathbf{v}^k))$$

- We have taken average log-likelihood above which does not affect the optimization but results in interesting interpretation for the final algorithm.

- Now, the earlier function $f(\theta)$ becomes

$$\begin{aligned} f(\theta) &= \frac{1}{M} \sum_{k=1}^{M} \mathcal{E}_{\mathbf{h}|\mathbf{v}^k} \left[\frac{\partial(-E(\mathbf{v}^k, \mathbf{h}))}{\partial w} \right] \\ &= \sum_{k=1}^{M} p_{\text{data}}(\mathbf{v}^k) \mathcal{E}_{\mathbf{h}|\mathbf{v}^k} \left[\frac{\partial(-E(\mathbf{v}^k, \mathbf{h}))}{\partial w} \right] \\ &= \mathcal{E}_{\text{data}} \left[\frac{\partial(-E(\mathbf{v}, \mathbf{h}))}{\partial w} \right] \end{aligned}$$

where $\mathcal{E}_{\text{data}}$ denotes expectation w.r.t. ‘distribution given by data’.

- The second term $g(\theta)$ now becomes

$$\begin{aligned} g(\theta) &= \frac{1}{M} \sum_{k=1}^{M} \mathcal{E}_{\mathbf{v},\mathbf{h}} \left[\frac{\partial(-E(\mathbf{v},\mathbf{h}))}{\partial w} \right] \\ &= \mathcal{E}_{\mathbf{v},\mathbf{h}} \left[\frac{\partial(-E(\mathbf{v},\mathbf{h}))}{\partial w} \right] \\ &= \mathcal{E}_{\theta} \left[\frac{\partial(-E(\mathbf{v},\mathbf{h}))}{\partial w} \right] \end{aligned}$$

where $\mathcal{E}_\theta$ represents expectation with respect to the distribution 'represented by the parameter vector θ'.

- Thus, finally we get the gradient of log-likelihood is

$$\frac{\partial \ell(\theta)}{\partial w} = \mathcal{E}_{\text{data}}\left[\frac{\partial(-E(\mathbf{v},\mathbf{h}))}{\partial w}\right] - \mathcal{E}_{\theta}\left[\frac{\partial(-E(\mathbf{v},\mathbf{h}))}{\partial w}\right]$$

- Or, the gradient of negative log-likelihood is

$$\frac{\partial(-\ell(\theta))}{\partial w} = \mathcal{E}_{\text{data}}\left[\frac{\partial(E(\mathbf{v},\mathbf{h}))}{\partial w}\right] - \mathcal{E}_{\theta}\left[\frac{\partial(E(\mathbf{v},\mathbf{h}))}{\partial w}\right]$$

- Note that the above is true for general Boltzmann machine as well because we have not used the conditional independences anywhere.

- By definition of energy, we have

$$\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial w_{ij}} = -h_i v_j; \quad \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial b_i} = -v_i; \quad \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial c_i} = -h_i$$

- Hence we get, for gradient descent on negative log-likelihood

$$\Delta w_{ij} = -\frac{\partial(-\ell(\theta))}{\partial w_{ij}} = \left(\mathcal{E}_{\mathsf{data}}\left[h_i v_j\right] \;-\; \mathcal{E}_{\theta}\left[h_i v_j\right]\right)$$

- Similar updating is done for b_i and c_i.
- This is the learning algorithm for the original Boltzmann machine too.

- Let us look at how we can calculate the first term

$$\begin{aligned}\mathcal{E}_{\text{data}}[h_i v_j] &= \sum_{k=1}^{M} (1/M)\, v_j^k \;\text{Prob}[h_i = 1|\mathbf{v}^k] \\ &= \sum_{k=1}^{M} (1/M)\, v_j^k \;\text{sig}\left(\sum_{s=1}^{m} w_{si} v_s^k + c_i\right)\end{aligned}$$

- Note that we have used the RBM conditional distribution here.
- The (1/M) factor is not important. For each training sample, w_{ij} is updated by $v_j^k \text{Prob}[h_i = 1|\mathbf{v}^k]$

- Now let us look at the second term in the update equation.

$$\mathcal{E}_\theta\left[h_i v_j\right] = \sum_{\mathbf{v},\mathbf{h}} p(\mathbf{v},\mathbf{h}) h_i v_j$$

- Computationally hard to – summation is over 2^{m+n} terms.
- We can try the same decomposition of the expectation as in the first term

$$\begin{aligned}\mathcal{E}_\theta\left[h_i v_j\right] &= \mathcal{E}_{p(\mathbf{v})}\mathcal{E}_{p(\mathbf{h}|\mathbf{v})}\left[h_i v_j\right] \\ &= \sum_{\mathbf{v}} p(\mathbf{v})\, v_j \mathrm{Prob}[h_i = 1|\mathbf{v}]\end{aligned}$$

- Still hard to compute – summation is over 2^m terms.
- Hence we approximate the expectation by sample mean.

- The gradient of the log-likelihood contains two terms – both are expectations.
- One of them is easy to compute for the RBM because of the conditional independence.
- But the second term is intractable and hence expectation is approximated by sample mean.
- For the general Boltzmann machine, both terms are intractable and both terms need approximation through sample mean.

- Let us consider an algorithm where we use a single sample.
- Let $\mathbf{v}'$ a training sample. Let $\mathbf{v}$ be a sample from the distribution determined by current θ. Then

$$\Delta w_{ij} = v'_j \text{Prob}[h_i = 1|\mathbf{v}'] \; - \; v_j \text{Prob}[h_i = 1|\mathbf{v}]$$

- All we need is a method to sample from the distribution $p(\mathbf{v})$ (at the current parameter values).

CD(k) Algorithm

- The distribution we are interested in is the stationary distribution of the Markov chain.
- So, we essentially run the chain for k steps and take the state as the sample.
- Gradient descent on NLL with this approximate gradient is called the Contrastive Divergence algorithm.
- To obtain the sample, we run the chain through what is called Gibbs sampling.
- We first look at Hastings-Metropolis algorithm and explain Gibbs sampling through that.

Hastings-Metropolis Sampling

- Let $\mathbf{X}$ be a random vector with mass function

  $$p(\mathbf{x}) = b(\mathbf{x})/Z$$

  We want samples from this without knowing Z.
- Let $q(\mathbf{x}, \mathbf{x}')$ (called the proposal distribution) be the transition probabilities of some ergodic chain on the same state space.
- Define a Markov chain with transition probabilities

  $$P(\mathbf{x}, \mathbf{x}') = q(\mathbf{x}, \mathbf{x}')\alpha(\mathbf{x}, \mathbf{x}'),\ \mathbf{x} \neq \mathbf{x}'$$

  At $\mathbf{x}$, generate $\mathbf{x}'$ using q and accept it with probability $\alpha(\mathbf{x}, \mathbf{x}')$ (stay at $\mathbf{x}$ with remaining prob)

- Given a q, we choose α as

$$\alpha(\mathbf{x}, \mathbf{x}') = \min\left(\frac{p(\mathbf{x}')q(\mathbf{x}', \mathbf{x})}{p(\mathbf{x})q(\mathbf{x}, \mathbf{x}')},\ 1\right) = \min\left(\frac{b(\mathbf{x}')q(\mathbf{x}', \mathbf{x})}{b(\mathbf{x})q(\mathbf{x}, \mathbf{x}')},\ 1\right)$$

- Note that one of $\alpha(\mathbf{x}, \mathbf{x}')$, $\alpha(\mathbf{x}', \mathbf{x})$ has to be 1.
- Then one can show that the Markov chain (corresponding to P) has $p(\cdot)$ as the stationary distribution.

- Suppose

$$\alpha(\mathbf{x},\mathbf{x}') = \frac{p(\mathbf{x}')q(\mathbf{x}',\mathbf{x})}{p(\mathbf{x})q(\mathbf{x},\mathbf{x}')} < 1 \text{ (and hence } \alpha(\mathbf{x}',\mathbf{x}) = 1)$$

- Then we have

$$\begin{aligned} p(\mathbf{x})P(\mathbf{x},\mathbf{x}') &= p(\mathbf{x})q(\mathbf{x},\mathbf{x}')\alpha(\mathbf{x},\mathbf{x}') \\ &= p(\mathbf{x}')q(\mathbf{x}',\mathbf{x}) \\ &= p(\mathbf{x}')q(\mathbf{x}',\mathbf{x})\alpha(\mathbf{x}',\mathbf{x}) \\ &= p(\mathbf{x}')P(\mathbf{x}',\mathbf{x}) \end{aligned}$$

  thus showing $p(\mathbf{x})$ is the stationary distribution.

Gibbs Sampling

- Gibbs sampling corresponds to a special case of q.
- Let $\mathbf{x} = (x_1, \cdots x_m)$. Let $\mathbf{x}_{-i}$ denote a vector containing all components of $\mathbf{x}$ except the i^{th} one.
- In Gibbs sampling, we generate successive states as

$$(x_1, x_2, \cdots, x_m) \rightarrow (x_1', x_2, \cdots, x_m) \rightarrow (x_1', x_2', \cdots, x_m) \cdots$$

  where x_i' is generated from the distribution $p(x_i|\mathbf{x}_{-i})$
- This corresponds to a q with $q(\mathbf{x}, \mathbf{x}') = 0$ if $\mathbf{x}, \mathbf{x}'$ differ in more that one component and is given by the above conditional distribution otherwise.

- If we use this q in Hastings-Metropolis, then α would always be 1.

$$\begin{aligned}
\alpha(\mathbf{x},\mathbf{x}') &= \frac{p(\mathbf{x}')q(\mathbf{x}',\mathbf{x})}{p(\mathbf{x})q(\mathbf{x},\mathbf{x}')} \quad (\mathbf{x},\mathbf{x}' \text{ differ in } i^{th}) \\
&= \frac{p(x_i'|\mathbf{x}'_{-i})p(\mathbf{x}'_{-i})p(x_i|\mathbf{x}'_{-i})}{p(x_i|\mathbf{x}_{-i})p(\mathbf{x}_{-i})p(x_i'|\mathbf{x}_{-i})} \\
&= \frac{p(x_i'|\mathbf{x}_{-i})p(\mathbf{x}_{-i})p(x_i|\mathbf{x}_{-i})}{p(x_i|\mathbf{x}_{-i})p(\mathbf{x}_{-i})p(x_i'|\mathbf{x}_{-i})} = 1
\end{aligned}$$

because $\mathbf{x}_{-i} = \mathbf{x}'_{-i}$.

Gibbs Chain for RBM

- Due to the conditional independences in the RBM graph, Gibbs sampling is efficient here.
- For any h_i the conditional distribution given all the other nodes is same as that given only the visible nodes.
- Hence, given current $\mathbf{v}$ we can choose the next values for all h_i together using the corresponding conditional distributions.
- Similarly, given current $\mathbf{h}$ we can choose the next values for all v_i together.
- This is how the Gibbs chain for the RBM is run.

CD(k) Algorithm

- We can now summarize the CD(k) algorithm.
- Let $\mathbf{v}^{(0)}$ denote a training sample.
- We initialize the chain with this and run it for k steps. The state of visible units be $\mathbf{v}^{(k)}$
- We update w_{ij} as

$$\Delta w_{ij} = v_j^{(0)} \text{Prob}[h_i = 1 | \mathbf{v}^{(0)}] \ - \ v_j^{(k)} \text{Prob}[h_i = 1 | \mathbf{v}^{(k)}]$$

  (Similar updates are done for b_i and c_i)
- We need to do this with each training sample.

- In CD(k) we are approximating the expectation with a single sample.
- Only as k tends to infinity, the sample would be from the needed distribution. For finite k it is a biased estimate.
- CD(k) is quite effective in practice.

A Variation – PCD

- A variation is Persistent CD or PCD.
- Here, instead of initializing the Gibbs chain with the training sample each time, we let the Gibbs chain run independently.
- After each parameter update, the Gibbs chain is started in the last state of the previous Gibbs chain.
- The idea is that this results in a better ‘mixing’
- PCD results in better learning in terms of smoother and better increase in log-likelihood.
- There is also a variation that maintains multiple Gibbs chains called Parallel Tempering.

- There are many generalizations possible on RBMs
- RBMs are good for unsupervised learning of representations.
- The representation on the hidden layer can be a good feature representation.
- Hence one variation is the so called convolutional RBMs.
- Here, we can have local connectivity (and also weight sharing with multiple feature planes)
- Found useful for feature detection in, e.g., speech

- So far we considered both v_i and h_j to be binary.
- Training data is for visible units.
- So, this RBM can learn distribution for binary data only
- We may want to learn distributions over $\Re^n$.
- We want RBMs where output of visible unit is a real number.
- Such an extension is possible.

Gaussian-Binary RBM

- We take hidden units to be still binary.
- But we take visible units to be real-valued.
- We make the same conditional independence assumptions. (Same graphical model)
- We retain the same conditional distribution of h_i given $\mathbf{v}$.
- We can get a model where the conditional distribution on visible units given $\mathbf{h}$ is Gaussian.

Gaussian-Binary RBM

- We take $\mathbf{v} \in \Re^m$ and $\mathbf{h} \in \{0,\ 1\}^n$.
- We can prescribe the conditional distributions as below:

$$p(\mathbf{v}|\mathbf{h}) \sim \mathcal{N}(\mathbf{v}; \mathbf{b} + W\mathbf{h}, \sigma^2 I)$$

$$\text{Prob}[h_i = 1|\mathbf{v}] = \text{sig}\left(c_i + \frac{1}{\sigma^2}\sum_{j=1}^{m} w_{ij}v_j\right)$$

- We can define energy as below to realize the above distributions

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

- The only extra parameter here is σ^2 which is the variance of the Gaussian.