CS.479/679, Fall 2015, Representation Learning — Lecture 12 - 11/05/2015

Restricted Boltzmann Machines (RBM), Deep Belief Networks(DBN)

Lecturer: Raman Arora — *Scribe: Tianyi Chen*

In this lecture, we present several generative models including Boltzmann Machines, Restricted Boltzmann Machine, Deep Belief Network, and Deep Boltzmann Machine.

1 Restricted Boltzmann Machines(RBM)

1.1 Boltzmann Machines

Boltzmann machines were originally introduced as a general "connectionist" approach to learning arbitrary probability distributions over binary vectors. Boltzmann Machines form the basis of a large number of popular variants. (Connectionism: a set of approaches in the fields of artificial intelligence, cognitive psychology, cognitive science, neuroscience, and philosophy of mind, that models mental or behavioral phenomena as the emergent processes of interconnected networks of simple units)

We define our Boltzmann machine over a d-dimensional binary random vector $x \in \{0,1\}^d$. The Boltzmann machine is an energy-based model, meaning we define the joint probability distribution over the model variable using an energy function, the density of joint distribution is:

$$P(x) = \frac{exp(-E(x))}{Z} \tag{1}$$

where $E(x)$ is the energy function, $exp(-E(x))$ is potential, Z is the partition function(normalization term), which ensure $\Sigma_x P(x) = 1$. The energy function of Boltzmann machine is given by:

$$E(x) = -x^T U x - d^T x \tag{2}$$

where U is the weight matrix of model parameters, and d are the offsets for each x.

The Boltzmann machine becomes more powerful when there exists missing variables. Since in this case, the non-observed variables, or latent variables, can act similarly as hidden units in MLP(Multilayer perceptron), and model higher-order interactions among visible units. Comparing with MLP, a Boltzmann machine with hidden units can perform as a universal approximator of probability mass functions rather than only modeling linear relationships between variables[3].

Under the idea of visible variables, and hidden variables, we can decompose the units into two subsets:

$$X_v = \{\text{visible units } x_v\} \ , \ X_h = \{\text{hidden or latent variables } x_h\}$$

Then Eq.(2) can be rewritten as

$$E(x_v, x_h) = -x_v^T R x_v - x_v^T W x_h - x_h^T S x_h - b^T x_v - c^T x_h. \tag{3}$$

To learn any parameter θ of Boltzmann machine, we usually adopt EM algorithm. Given a dataset of d visiable samples $X_v = \{x_v^{(1)}, x_v^{(2)}, ..., x_v^{(d)}\}$, we want to maximize the likelihood. Assume data is i.i.d, consider to maximize the log-likelihood:

$$l(\theta) = logP(X_v) = \Sigma_{t=1}^{d} log(P(x_v^{(t)})) \tag{4}$$

Boltzmann machine doesn't explicitly parametrize a distribution over the visible units as $P(x_v^{(t)})$. $P(x_v^{(t)})$ is recovered via energy function Eq(3) to joint density function over x_v and x_h, and marginalize over x_h.

$$P(x_v^{(t)}) = \Sigma_{x_h} P(x_v^{(t)}, x_h^{(t)}) = \Sigma_{x_h} \frac{1}{Z} exp\{-E(x_v^{(t)}, x_h^{(t)})\} \tag{5}$$

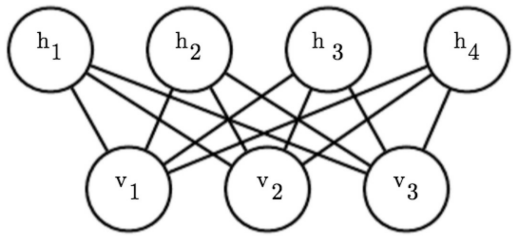

Figure 1: Restricted Boltzmann Machine(RBM)

Notice Eq.(4), Eq.(5) gives us objective function to be maximized. However, since Z is a function of model parameters, maximizing the likelihood function is not amenable to analytically solution. Hence, we prefer to use gradient ascent method to update the parameters of Boltzmann machines. The gradient of Boltzmann machine likelihood is

$$\begin{aligned}\frac{\partial l(\theta)}{\partial \theta} &= \frac{\partial}{\partial \theta}\left(\Sigma_{t=1}^{d}[\frac{1}{Z}log\Sigma_{x_h}exp\{-E(x_v^{(t)},x_h^{(t)})\}]\right)\\ &= \Sigma_{t=1}^{d}\frac{\partial}{\partial \theta}[log\Sigma_{x_h}exp\{-E(x_v^{(t)},x_h^{(t)})\}] - \frac{\partial Z}{\partial \theta}\\ &= \Sigma_{t=1}^{d}\left[\Sigma_{x_h}\frac{exp\{-E(x_v^{(t)},x_h^{(t)})\}}{\Sigma_{x_h}exp\{-E(x_v^{(t)},x_h^{(t)})\}}\frac{\partial}{\partial \theta}E(x_v^{(t)},x_h^{(t)})\right] - \frac{\partial Z}{\partial \theta}\end{aligned} \tag{6}$$

1.2 Restricted Boltzmann Machines

Restricted Boltzmann machines, are some of the most common building blocks of deep probabilistic models. They are undirected probabilistic graphical models containing a layer of observable variables and a single layer of latent variables.

Comparing with general Boltzmann Machines built on undirected graph, the Restricted Boltzmann Machines is built on undirected bipartite graph. $G = (V \cup H, E)$, where visible nodes and hidden nodes form two bipartitions V, H of the whole undirected bipartite graph respectively, like Figure 1. Like general Boltzmann machine, we still use energy-based model for joint probability distribution, with density function

$$P(V = v, H = h) = \frac{1}{Z}exp\{-E(v,h)\} \tag{7}$$

$E(v,h)$ is the energy function that parametrizes the relationship between the visible and hidden variables. Comparing with Eq.(3), since RBM is built on bipartite graph, we have

$$E(v,h) = -b^T v - c^T h - v^T W h \tag{8}$$

Z is the normalization term:

$$Z = \Sigma_v \Sigma_h exp\{-E(v,h)\} \tag{9}$$

1.2.1 Preliminary: Gibbs Sampler on RBM to generate $p(h|v), p(v|h)$

In practice, Z is often intractable, causing joint distribution intractable consequently. But by the special structure of bipartite graph, visible and hidden units are conditionally independent given one-another. Using

this property, we can notice $P(h|v), P(v|h)$ are factorial, relatively simple to compute, or sample from.

$$\begin{aligned}
p(h|v) &= \frac{p(v,h)}{p(v)} = \frac{p(v,h)}{\Sigma_h p(v,h)} \\
&= \frac{1/Z exp(-E(v,h))}{\Sigma_h 1/Z exp(-E(v,h))} \\
&= \frac{exp(b^T v + c^T h + v^T W h)}{\Sigma_h exp(b^T v + c^T h + v^T W h)} \\
&= \frac{\Pi_j^n exp(v^T W_{:,j} h_j + c_j h_j) exp(b^T v)}{\Sigma_h \Pi_j^n exp(v^T W_{:,j} h_j + c_j h_j) exp(b^T v)} \\
&= \frac{\Pi_j^n exp(v^T W_{:,j} h_j + c_j h_j)}{\Sigma_h \Pi_j^n exp(v^T W_{:,j} h_j + c_j h_j)} \\
&= \frac{\Pi_j^n exp(v^T W_{:,j} h_j + c_j h_j)}{\Pi_j^n \Sigma_h exp(v^T W_{:,j} h_j + c_j h_j)} \\
&= \frac{\Pi_j^n p(v, h_j)}{\Pi_j^n p(v)} \\
&= \Pi_j^n p(h_j|v)
\end{aligned} \tag{10}$$

Similarly, we have

$$\begin{aligned}
p(v|h) &= \frac{p(v,h)}{p(h)} = \frac{p(v,h)}{\Sigma_v p(v,h)} \\
&= \frac{1/Z exp(-E(v,h))}{\Sigma_v 1/Z exp(-E(v,h))} \\
&= \frac{exp(b^T v + c^T h + v^T W h)}{\Sigma_v exp(b^T v + c^T h + v^T W h)} \\
&= \frac{\Pi_i^d exp(v_i W_{i,:} h + b_i v_i) exp(c^T h)}{\Sigma_v \Pi_i^d exp(v_i W_{i,:} h + b_i v_i) exp(c^T h)} \\
&= \frac{\Pi_i^d exp(v_i W_{i,:} h + b_i v_i)}{\Sigma_v \Pi_i^d exp(v_i W_{i,:} h + b_i v_i)} \\
&= \frac{\Pi_i^d exp(v_i W_{i,:} h + b_i v_i)}{\Pi_i^d \Sigma_v exp(v_i W_{i,:} h + b_i v_i)} \\
&= \frac{\Pi_i^d p(v_i, h)}{\Pi_i^d p(h)} \\
&= \Pi_i^d p(v_i|h)
\end{aligned} \tag{11}$$

Hence, we can try to use Gibbs Sampling to generate joint distribution of h,v.

To complete Gibbs Sampling, deriving conditional distribution is necessary.Since we are conditioning given visible observations units v, assume DH is the range of value of h_j, then $p(h|v)$ follows immediately by Eq.(10)

$$\begin{aligned}
p(h_j = h|v) &= \frac{\tilde{p}(h_j = h|v)}{\Sigma_{h' \in DH} \tilde{p}(h_j = h'|v)} \\
&= \frac{exp\{c_j h + v^T W_{:,j} h\}}{\Sigma_{h' \in DH} exp\{c_j h' + v^T W_{:,j} h'\}}
\end{aligned} \tag{12}$$

For binary case, $j \in \{1, 2, ..., n\}$, we have the following beautiful conditional density:

$$\begin{aligned} p(h_j = 1|v) &= \frac{\tilde{p}(h_j = 1|v)}{\tilde{p}(h_j = 0|v) + \tilde{p}(h_j = 1|v)} \\ &= \frac{exp\{c_j + v^T W_{:,j}\}}{exp(0) + exp\{c_j + v^T W_{:,j}\}} \\ &= \frac{exp\{c_j + v^T W_{:,j}\}}{1 + exp\{c_j + v^T W_{:,j}\}} \\ &= sigm(c_j + v^T W_{:,j}) \end{aligned} \tag{13}$$

$$\begin{aligned} p(h_j = 0|v) &= 1 - p(h_j = 1|v) \\ &= 1 - sigmoid(c_j + v^T W_{:,j}) \\ &= sigmoid(-c_j - v^T W_{:,j}) \end{aligned} \tag{14}$$

Similarily,

$$p(v_i = 1|h) = sigm(b_i + W_{i,:}h) \tag{15}$$

$$\begin{aligned} p(v_i = 0|h) &= 1 - sigm(b_i + W_{i,:}h) \\ &= sigm(-b_i - W_{i,:}h) \end{aligned} \tag{16}$$

We can also write the conditional density as

$$p(h|v) = \Pi_{j=1}^{n} \sigma((2h - 1) \circ (c + W^T v))_j \tag{17}$$

$$p(v|h) = \Pi_{i=1}^{d} \sigma((2v - 1) \circ (b + Wh))_i \tag{18}$$

where $A \circ B$ means multiplying A and B elementwise, and choose activation function σ as sigmoid function.

Visible units and hidden units are conditional independent given hidden units and visible units respectively. This factorial property is a very useful property of the RBM, and allows us to efficiently draw samples from the joint distribution via block Gibbs sampling strategy. Block Gibbs sampling simply refers to the situation where in each step of Gibbs sampling, multiple variables(or a 'block' variables) are sampled jointly. For RBM, visible units can be sampled simultaneously given hidden units. Similarly, hidden units are sampled simultaneously given visible units.

In summary, there are two steps for Gibbs sampling:

1. Sample $h^{(l)} \sim p(h|v^{(l)})$
2. Sample $v^{(l+1)} \sim p(v|h^{(l)})$

where $h^{(l)}$ refers to the set of all hidden units at $l - th$ step on such Markov chain. By such step, $h_j^{(l+1)}$ is randomly chosen to be 1 (versus 0) with probability $sigm(c_j + v^{(l)T} W_{:,j})$. Similarly, $v_i^{(l+1)}$ is randomly chosen to be 1 (versus 0) with probability $sigm(b_i + W_{i,:}h^{(l)})$. This is can be illustrated graphically in Figure 2.

As $l \to \infty$, samples $(v^{(l)}, h^{(l)})$ should converge to accurate samples from the distribution of $p(v, h)$. It is well-known to say such sampling would be expensive.

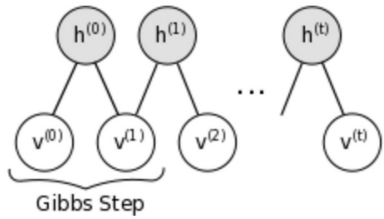

Figure 2: Graphical illustration of block Gibbs sampling on RBM

1.2.2 train RBM after generating $p(h|v), p(v|h)$

As a probabilistic model, a sensible inductive principle for estimating the model parameters with maximum likelihood. We'd like to minimize the negative log-likelihood(NLL) to reach maximum likelihood. Assume we have joint density $p(h, v)$, we can marginalize over h to obtain the negative likelihood of visible observations.

$$L(W, b, c|v) = -p(v) = -\Sigma_h p(v, h) = -\Sigma_h \frac{exp\{-E(v, h)\}}{Z} \tag{19}$$

Here, we introduce a new notation Free energy, inspired from physics, as follows

$$F(v) = -log\Sigma_h exp\{-E(v, h)\} \tag{20}$$

Then, for general variables, Eq.(7) translates directly to the following free energy formula:

$$F(v) = -b^T v - \Sigma_j log\Sigma_{h_j} exp\{(c_j + v^T W_{:,j})h_j\} \tag{21}$$

And the free energy of an RBM with binary units further simplifies to:

$$F(v) = -b^T v - \Sigma_j^n log\left(1 + exp(c_j + v^T W_{:,j})\right) \tag{22}$$

We also have

$$p(v) = \frac{exp\{-F(v)\}}{Z} \tag{23}$$

where $Z = \Sigma_v exp\{-F(v)\}$
Then the gradient of (NLL)negative log-likelihood has an interesting form, let $\theta = \{b, c, W\}$

$$\frac{\partial l(W, b, c|v)}{\partial \theta} = \frac{\partial - logp(v)}{\partial \theta} = \frac{\partial F(v)}{\partial \theta} - \Sigma_{v'} p(v') \frac{\partial F(v')}{\partial \theta} \tag{24}$$

Notice that the above gradient contains two terms, which are referred to as the positive and negative phase. The terms positive and negative do not refer to the sign of each term in the equation, but rather reflect their effect on the probability density defined by the model.

Notice the gradient of NLL under free energy form is used in theano for practical implementation.

More generally,given i.i.d visible dataset $\{v^{(1)}, v^{(2)}, ..., v^{(n)}\}$, where $v^{(t)} \in R^d$, the average negative log-

likelihood can be written as

$$\begin{aligned} l(W,b,c) &= -\frac{1}{n}\Sigma_{t=1}^{n} log p(\mathrm{v}^{(t)}) \\ &= -\frac{1}{n}\Sigma_{i=1}^{n} log \Sigma_h p(\mathrm{v}^{(t))}, h) \\ &= -\frac{1}{n}\left(\Sigma_{t=1}^{n} log \Sigma_h exp\{-E(\mathrm{v}^{(t)}, h)\}\right) + \frac{1}{n} n log Z \\ &= -\frac{1}{n}\left(\Sigma_{t=1}^{n} log \Sigma_h exp\{-E(\mathrm{v}^{(t)}, h)\}\right) + \frac{1}{n} n log \Sigma_{v,h} exp\{E(\mathrm{v}^{(t)}, h)\} \\ &= -\frac{1}{n}\left(\Sigma_{t=1}^{n} log \Sigma_h exp\{-E(\mathrm{v}^{(t)}, h)\}\right) + log \Sigma_{v,h} exp\{E(\mathrm{v}^{(t)}, h)\} \end{aligned} \tag{25}$$

Consequently, the gradient of average negative log-likelihood with ($\theta = \{b, c, W\}$) is

$$\begin{aligned} \frac{\partial}{\partial\theta} l(\theta) &= -\frac{1}{n}\frac{\partial}{\partial\theta}\left(\Sigma_{t=1}^{n} log \Sigma_h exp\{-E(\mathrm{v}^{(t)}, h)\}\right) + \frac{\partial}{\partial\theta} log \Sigma_{v,h} exp\{-E(\mathrm{v}, h)\} \\ &= -\frac{1}{n}\Sigma_{t=1}^{n} \frac{exp\{-E(\mathrm{v}^{(t)}, h)\}\frac{\partial}{\partial\theta}(-E(\mathrm{v}^{(t)}, h))}{\Sigma_h exp\{-E(\mathrm{v}^{(t)}, h)\}} + \frac{\Sigma_{v,h} exp\{-E(v,h)\}\frac{\partial}{\partial\theta}(-E(v,h))}{\Sigma_{v,h} exp\{-E(v,h)\}} \\ &= \frac{1}{n}\Sigma_{t=1}^{n} E_{p(h|v^{(t)})}[\frac{\partial}{\partial\theta}(E(\mathrm{v}^{(t)}, h))] - E_{p(v,h)}[\frac{\partial}{\partial\theta}(E(v,h))] \end{aligned} \tag{26}$$

As Eq.(26) shown ,the gradient of average negative log-likelihood can be regarded as the difference between two expectations of the gradient of energy function.

We call the first term as positive phase, the second as negative phase. The real meaning is we want to increase the likelihood of our visible samples, and reduce the noise.

Next, we can derive the gradient term $\frac{\partial}{\partial\theta}(E(v,h))$ respect to W

$$\begin{aligned} \frac{\partial}{\partial W}(E(v,h)) &= \frac{\partial}{\partial W}(-b^T v - c^T h - v^T W h) \\ &= -hv^T \end{aligned} \tag{27}$$

Similarly, the gradient respect to b and c are

$$\frac{\partial}{\partial b}(E(v,h)) = -v \tag{28}$$

$$\frac{\partial}{\partial c}(E(v,h)) = -h \tag{29}$$

Putting it all together, we can derive the full gradients respect to RBM parameters, given n training examples.

$$\frac{\partial}{\partial W} l(W,b,c) = -\frac{1}{n}\Sigma_{t=1}^{n} h(v^{(t)}) v^{(t)T} + E_{p(v,h)}[hv^T] \tag{30}$$

$$\frac{\partial}{\partial b} l(W,b,c) = -\frac{1}{n}\Sigma_{t=1}^{n} v^{(t)} + E_{p(v,h)}[v] \tag{31}$$

$$\frac{\partial}{\partial c} l(W,b,c) = -\frac{1}{n}\Sigma_{t=1}^{n} h(v^{(t)}) + E_{p(v,h)}[h] \tag{32}$$

where we define $h(v)$ as

$$h(v) = E_{p(h|v)}[h] = sigmoid(c + vW) \tag{33}$$

Now, we have derived these gradient expressions. However, we can not calculate them directly, since these expectations require $E_{p(v,h)}[]$ term with joint distribution $p(v,h)$, which is hard to calculate.

Even though it is impractical to calculate the gradients exactly, there exist several methods to calculate them approximately. Remember we can use Gibbs sampling to generate $p(v|h), p(h|v)$. With reasonable prior, like flat prior (uniform prior in the implementation of theano). We can imagine using, T MCMC samples from generated approximate joint density $p(v,h)$ to form a Monte Carlo estimate of the expectations over the joint distribution:

$$E_{p(v,h)}[f(v,h)] \approx \frac{1}{T}\Sigma_{i=1}^{T} f(\tilde{v}^{(i)}, \tilde{h}^{(i)}) \tag{34}$$

MCMC chains typically require a burn-in period, and only average the samples during burn-in period in terms of convergence.

In real practice,

1. We can even replace the expectation term by a point estimate at $\tilde{v}$.

$$E_{p(h|v^{(t)})}[\frac{\partial}{\partial \theta}(E(\mathrm{v}^{(t)}, h))] \approx \frac{\partial E(v^{(t)}, \tilde{h}^{(t)})}{\partial \theta}, E_{p(v,h)}[\frac{\partial}{\partial \theta}(E(v,h))] \approx \frac{\partial E(\tilde{v}, \tilde{h})}{\partial \theta} \tag{35}$$

2. At training iteration t, we will start sampling chain at some $V_{prior}^{(t)}$, obtain point $\tilde{v}$ by Gibbs Sampling.

Hence, we have the pseudocode of training RBM.

1. Select a batch of training examples $\mathrm{v}^{(t)} = \{v_1^{(t)}, v_2^{(t)}, ..., v_{n'}^{(t)}\}$ with batch size n'.
   (a) generate sample $\tilde{v}_i$ for each $v_i^{(t)}$ for each using Gibbs Sampling.
   (b) update parameters

$$W^{(t+1)} \leftarrow W^{(t)} + \alpha \left(\frac{1}{n'} \Sigma_{i=1}^{n'} h(v_i^{(t)}) v_i^{(t)T} - E_{p(v,h)}[hv^T] \right) \tag{36}$$

$$b^{(t+1)} \leftarrow b^{(t)} + \alpha \left(\frac{1}{n'} \Sigma_{i=1}^{n'} v_i^{(t)} - E_{p(v,h)}[v] \right) \tag{37}$$

$$c^{(t+1)} \leftarrow c^{(t)} + \alpha \left(\frac{1}{n'} \Sigma_{i=1}^{n'} h(v_i^{(t)}) - E_{p(v,h)}[h] \right) \tag{38}$$

2. Go back to 1 if stop criterion is not satisfied.

where α is the learning rate.

However, Gibbs Sampling has a potential weakness: slow convergence. Sampling may be expensive. There exist many methods to solve this issue. Three of them are: contrasive divergence, persistent CD and stochastic maximum likelihood.

1.2.3 (CD)Contrastive Divergence Training of the RBM

Contrastive Divergence uses two tricks to speed up the sampling process:

1. since we eventually want $p(v) \approx p_{train}(v)$ (the true, underlying distribution of the data), we initialize the Markov chain with a training example at each training iteration(i.e., from a distribution that is expected to be close to p, so that the chain will be already close to having converged to its final distribution p).
2. CD does not wait for the chain to converge. Samples are obtained after only K-steps of Gibbs sampling.

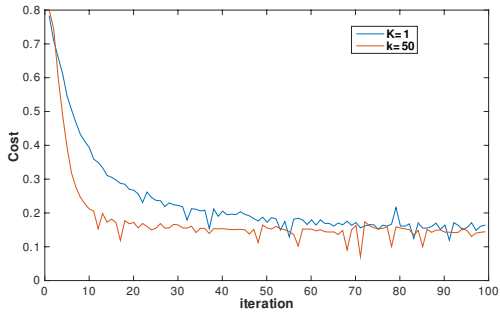

Figure 3: Contrastive Divergence Training of the RBM under different K, cost versus iteration

In one word, at training iteration t, CD will start sampling chain at some real training sample $v^{(t)}$ to obtain $\tilde{v}$ by sample K-times.

The pseudocode of training RBM under CD is the following:

1. Select a batch of training examples $\mathbf{v}^{(t)} = \{v_1^{(t)}, v_2^{(t)}, ..., v_{n'}^{(t)}\}$ with batch size n'.
   (a) generate sample $\tilde{v}_i$ for each $v_i^{(t)}$ for each using $K - times$ Gibbs Sampling.
   (b) update parameters

$$W^{(t+1)} \leftarrow W^{(t)} + \alpha \left(\frac{1}{n'} \Sigma_{i=1}^{n'} h(v_i^{(t)}) v_i^{(t)T} - \frac{1}{n'} \Sigma_{i=1}^{n'} h(\tilde{v}_i) \tilde{v}_i^T \right) \tag{39}$$

$$b^{(t+1)} \leftarrow b^{(t)} + \alpha \left(\frac{1}{n'} \Sigma_{i=1}^{n'} v_i^{(t)} - \frac{1}{n'} \Sigma_{i=1}^{n'} \tilde{v}_i \right) \tag{40}$$

$$c^{(t+1)} \leftarrow c^{(t)} + \alpha \left(\frac{1}{n'} \Sigma_{i=1}^{n'} h(v_i^{(t)}) - \frac{1}{n'} \Sigma_{i=1}^{n'} h(\tilde{v}_i) \right) \tag{41}$$

2. Go back to 1 if stop criterion is not satisfied.

In pratice, K=1 has been shown to work surprisingly well. Using a new java Deep Learning Library(name it as DLcty): we can observe loss value descents fast and sufficiently at K=1 with SGD, shown in Figure 3,

1.2.4 Persistent CD

Persistent CD uses another approximation. Instead of restarting a chain for each observed training sample, Persistent CD will run only one Markov chain. For each time parameter update, Persistent CD will obtain $\tilde{v}$ for any training samples by continuing sample K times on this Markov chain.

The pseudocode of training RBM under persistent CD is the following:

1. Select a batch of training examples $\mathbf{v}^{(t)} = \{v_1^{(t)}, v_2^{(t)}, ..., v_{n'}^{(t)}\}$ with batch size n'.
   (a) if $t = 0$, initialize a Markov chain MC

(b) generate a shared sample $\tilde{v}$ for any $v_i^{(t)}$, $i \in \{1, 2, ..., n'\}$, using $K - times$ Gibbs Sampling on MC.

(c) update parameters

$$W^{(t+1)} \leftarrow W^{(t)} + \alpha \left(\frac{1}{n'} \Sigma_{i=1}^{n'} h(v_i^{(t)}) v_i^{(t)T} - h(\tilde{v}) \tilde{v}^T \right) \tag{42}$$

$$b^{(t+1)} \leftarrow b^{(t)} + \alpha \left(\frac{1}{n'} \Sigma_{i=1}^{n'} v_i^{(t)} - \tilde{v} \right) \tag{43}$$

$$c^{(t+1)} \leftarrow c^{(t)} + \alpha \left(\frac{1}{n'} \Sigma_{i=1}^{n'} h(v_i^{(t)}) - h(\tilde{v}) \right) \tag{44}$$

2. Go back to 1 if stop criterion is not satisfied.

1.2.5 Stochastic Maximum Likelihood for the RBM

While contrastive divergence has been the most popular method of training RBMs,the stochastic maximum likelihood (SML) algorithm is known to be a competitive alternative
Comparing with CD, instead of initializing the k-step MCMC chain with training examples, in SML, we initialize the MCMC chain at iteration s with the last state of the MCMC chain from the last training iteration (s-1).

References

[1] DeepLearning 0.1 documentation

[2] Deep Learning,unpublished,Book in preparation for MIT Press,2015

[3] Deep Belief Networks are compact universal approximators, 2009