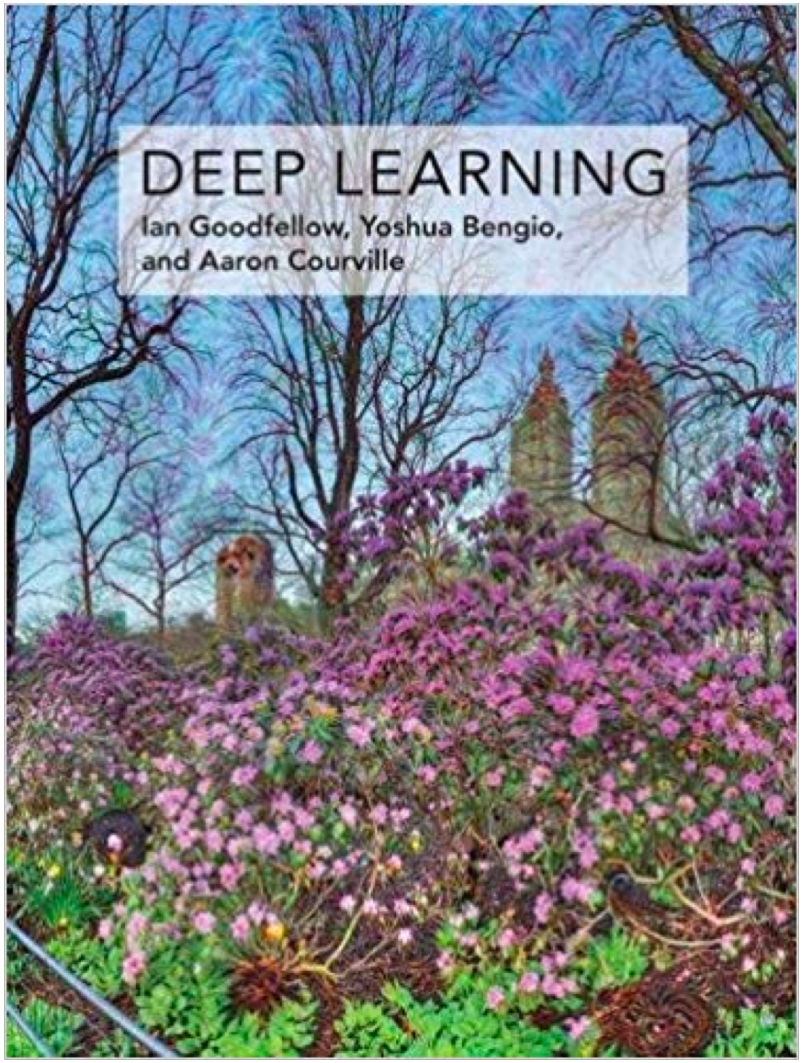
國立交通大學
National Chiao Tung University

Deep Learning
深度學習
Fall 2018

Boltzmann Machines
(Chapter 20)

Prof. Chia-Han Lee
李佳翰 副教授



Boltzmann machines

- Boltzmann machines were introduced to learning arbitrary probability distributions over binary vectors.

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

$$P(\boldsymbol{x}) = \frac{\exp\left(-E(\boldsymbol{x})\right)}{Z}, \tag{20.1}$$

where $E(\boldsymbol{x})$ is the energy function, and Z is the partition function that ensures that $\sum_{\boldsymbol{x}} P(\boldsymbol{x}) = 1$.

- The energy function of the Boltzmann machine is

$$E(\boldsymbol{x}) = -\boldsymbol{x}^\top \boldsymbol{U}\boldsymbol{x} - \boldsymbol{b}^\top \boldsymbol{x}, \tag{20.2}$$

where $\boldsymbol{U}$ is the "weight" matrix of model parameters and $\boldsymbol{b}$ is the vector of bias parameters.

Boltzmann machines

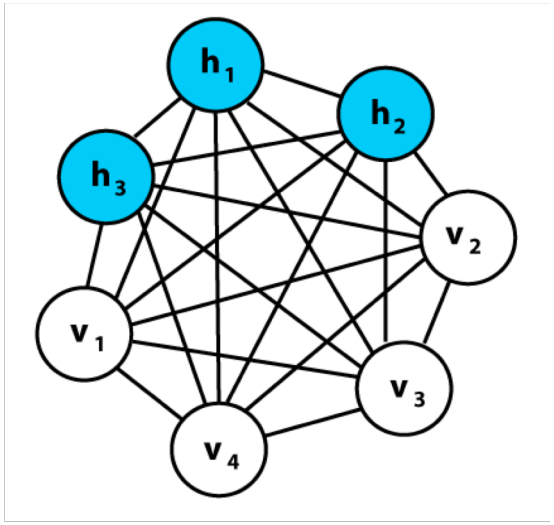
- The Boltzmann machine becomes more powerful when not all the variables are observed. In this case, the latent variables can act similarly to hidden units in a multilayer perceptron and model higher-order interactions among the visible units.

- Just as the addition of hidden units to convert logistic regression into an MLP results in the MLP being a universal approximator of functions, a Boltzmann machine with hidden units is no longer limited to modeling linear relationships between variables. Instead, the Boltzmann machine becomes a universal approximator of probability mass functions over discrete variables.

Boltzmann machines

- Formally, we decompose the units $\boldsymbol{x}$ into two subsets: the visible units $\boldsymbol{v}$ and the latent (or hidden) units $\boldsymbol{h}$. The energy function becomes

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{v}^\top \boldsymbol{R}\boldsymbol{v} - \boldsymbol{v}^\top \boldsymbol{W}\boldsymbol{h} - \boldsymbol{h}^\top \boldsymbol{S}\boldsymbol{h} - \boldsymbol{b}^\top \boldsymbol{v} - \boldsymbol{c}^\top \boldsymbol{h}. \quad (20.3)$$



- Learning algorithms for Boltzmann machines are usually based on maximum likelihood. All Boltzmann machines have an intractable partition function, so the maximum likelihood gradient must be approximated.

https://upload.wikimedia.org/wikipedia/commons/7/7a/Boltzmannexamplev1.png

Restricted Boltzmann machines

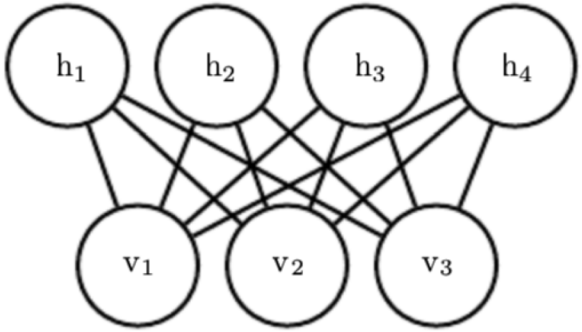
- Restricted Boltzmann machines (RBMs) are undirected probabilistic graphical models containing a layer of observable variables and a single layer of latent variables. RBMs may be stacked (one on top of the other) to form deeper models.

- The graph of the RBM is a bipartite graph, with no connections permitted between any variables in the observed layer or between any units in the latent layer.

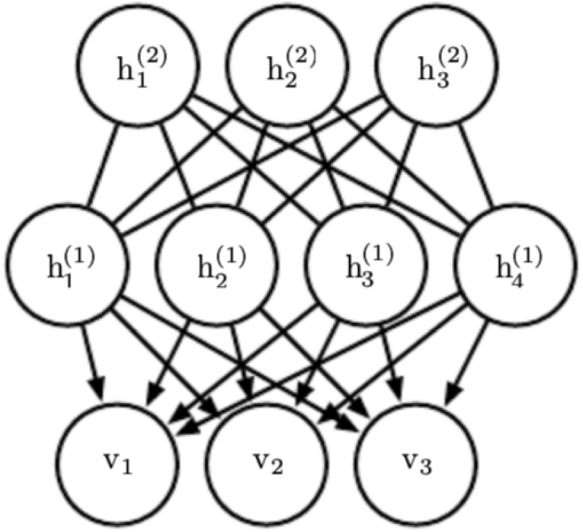
- We begin with the binary version of the restricted Boltzmann machine, but as we see later, there are extensions to other types of visible and hidden units.

Restricted Boltzmann machines

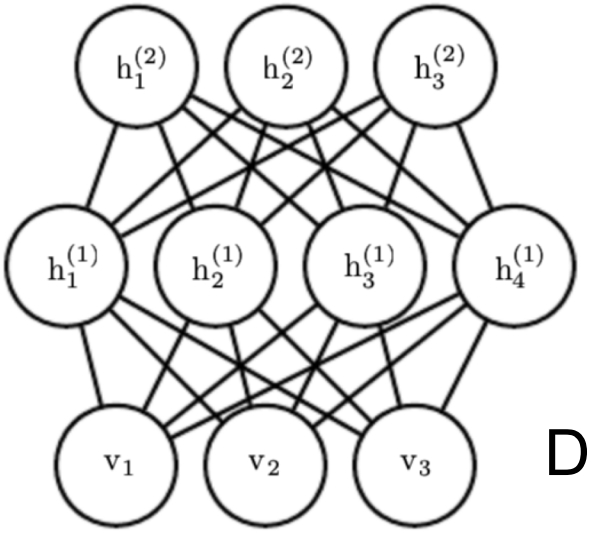
RBM



(a)



(b) Deep belief networks



(c) Deep Boltzmann machine

Restricted Boltzmann machines

- More formally, let the observed layer consist of a set of n_v binary random variables, which we refer to collectively with the vector $\mathbf{v}$. We refer to the latent, or hidden, layer of n_h binary random variables as $\boldsymbol{h}$.

- The restricted Boltzmann machine is an energy-based model with the joint probability distribution specified by its energy function:

$$P(\mathbf{v} = \boldsymbol{v}, \mathbf{h} = \boldsymbol{h}) = \frac{1}{Z} \exp\left(-E(\boldsymbol{v}, \boldsymbol{h})\right). \tag{20.4}$$

- The energy function for an RBM is given by

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{b}^\top \boldsymbol{v} - \boldsymbol{c}^\top \boldsymbol{h} - \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h}, \tag{20.5}$$

and Z is the partition function:

$$Z = \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} \exp\left\{-E(\boldsymbol{v}, \boldsymbol{h})\right\}. \tag{20.6}$$

Restricted Boltzmann machines

- From the definition of the partition function Z, the naive method of computing Z (exhaustively summing over all states) could be computationally intractable.

- It was formally proved that for RBM the partition function Z is intractable. The intractable partition function Z implies that the normalized joint probability distribution $P(\boldsymbol{v})$ is also intractable to evaluate.

- Though $P(\boldsymbol{v})$ is intractable, the bipartite graph structure of the RBM has the special property of its conditional distributions $P(\mathbf{h}|\mathbf{v})$ and $P(\mathbf{v}|\mathbf{h})$ being factorial and relatively simple to compute and sample from.

Conditional distributions

- Deriving the conditional distributions from the joint distribution is straightforward:

$$P(\boldsymbol{h} \mid \boldsymbol{v}) = \frac{P(\boldsymbol{h}, \boldsymbol{v})}{P(\boldsymbol{v})} \tag{20.7}$$

$$= \frac{1}{P(\boldsymbol{v})} \frac{1}{Z} \exp\left\{ \boldsymbol{b}^\top \boldsymbol{v} + \boldsymbol{c}^\top \boldsymbol{h} + \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h} \right\} \tag{20.8}$$

$$= \frac{1}{Z'} \exp\left\{ \boldsymbol{c}^\top \boldsymbol{h} + \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h} \right\} \tag{20.9}$$

$$= \frac{1}{Z'} \exp\left\{ \sum_{j=1}^{n_h} c_j h_j + \sum_{j=1}^{n_h} \boldsymbol{v}^\top \boldsymbol{W}_{:,j} \boldsymbol{h}_j \right\} \tag{20.10}$$

$$= \frac{1}{Z'} \prod_{j=1}^{n_h} \exp\left\{ c_j h_j + \boldsymbol{v}^\top \boldsymbol{W}_{:,j} \boldsymbol{h}_j \right\}. \tag{20.11}$$

Conditional distributions

- Since we are conditioning on the visible units $\mathbf{v}$, we can treat these as constant with respect to the distribution $P(\mathbf{h}|\mathbf{v})$.The factorial nature of the conditional $P(\mathbf{h}|\mathbf{v})$ follows from our ability to write the joint probability over the vector $\boldsymbol{h}$ as the product of (unnormalized) distributions over the individual elements, h_j.

- It is now a simple matter of normalizing the distributions over the individual binary h_j.

$$P(h_j = 1 \mid \boldsymbol{v}) = \frac{\tilde{P}(h_j = 1 \mid \boldsymbol{v})}{\tilde{P}(h_j = 0 \mid \boldsymbol{v}) + \tilde{P}(h_j = 1 \mid \boldsymbol{v})} \tag{20.12}$$

$$= \frac{\exp\left\{c_j + \boldsymbol{v}^\top \boldsymbol{W}_{:,j}\right\}}{\exp\{0\} + \exp\left\{c_j + \boldsymbol{v}^\top \boldsymbol{W}_{:,j}\right\}} \tag{20.13}$$

$$= \sigma\left(c_j + \boldsymbol{v}^\top \boldsymbol{W}_{:,j}\right). \tag{20.14}$$

Conditional distributions

- We can now express the full conditional over the hidden layer as the factorial distribution:

$$P(\boldsymbol{h} \mid \boldsymbol{v}) = \prod_{j=1}^{n_h} \sigma\left((2\boldsymbol{h} - 1) \odot (\boldsymbol{c} + \boldsymbol{W}^\top \boldsymbol{v})\right)_j . \tag{20.15}$$

- A similar derivation will show that the other condition of interest to us, $P(\boldsymbol{v}|\boldsymbol{h})$, is also a factorial distribution:

$$P(\boldsymbol{v} \mid \boldsymbol{h}) = \prod_{i=1}^{n_v} \sigma\left((2\boldsymbol{v} - 1) \odot (\boldsymbol{b} + \boldsymbol{W}\boldsymbol{h})\right)_i . \tag{20.16}$$

- Compared to other undirected models used in deep learning, the RBM is relatively straightforward to train because we can compute $P(\boldsymbol{h}|\boldsymbol{v})$ exactly in closed form.

Deep belief networks

- Deep belief networks (DBNs) were one of the first nonconvolutional models to successfully admit training of deep architectures. The introduction of deep belief networks in 2006 began the current deep learning renaissance. Prior to the introduction of deep belief networks, deep models were considered too difficult to optimize. DBNs demonstrated that deep architectures can be successful by outperforming kernelized support vector machines on the MNIST dataset.

- Today, deep belief networks have mostly fallen out of favor and are rarely used, even compared to other unsupervised or generative learning algorithms, but they are still deservedly recognized for their important role in deep learning history.

Deep belief networks

- Deep belief networks are generative models with several layers of latent variables. The latent variables are typically binary, while the visible units may be binary or real. There are no intralayer connections. Usually, every unit in each layer is connected to every unit in each neighboring layer, though it is possible to construct more sparsely connected DBNs.

- The connections between the top two layers are undirected. The connections between all other layers are directed, with the arrows pointed toward the layer that is closest to the data.

Deep belief networks

- A DBN with l hidden layers contains l weight matrices: $\boldsymbol{W}^{(1)}, \ldots, \boldsymbol{W}^{(l)}$. It also contains $l+1$ bias vectors $\boldsymbol{b}^{(0)}, \ldots, \boldsymbol{b}^{(l)}$, with $\boldsymbol{b}^{(0)}$ providing the biases for the visible layer.

- The probability distribution represented by the DBN is

$$P(\boldsymbol{h}^{(l)}, \boldsymbol{h}^{(l-1)}) \propto \exp\left(\boldsymbol{b}^{(l)\top}\boldsymbol{h}^{(l)} + \boldsymbol{b}^{(l-1)\top}\boldsymbol{h}^{(l-1)} + \boldsymbol{h}^{(l-1)\top}\boldsymbol{W}^{(l)}\boldsymbol{h}^{(l)}\right), \quad (20.17)$$

$$P(h_i^{(k)} = 1 \mid \boldsymbol{h}^{(k+1)}) = \sigma\left(b_i^{(k)} + \boldsymbol{W}_{:,i}^{(k+1)\top}\boldsymbol{h}^{(k+1)}\right) \forall i, \forall k \in 1, \ldots, l-2, \quad (20.18)$$

$$P(v_i = 1 \mid \boldsymbol{h}^{(1)}) = \sigma\left(b_i^{(0)} + \boldsymbol{W}_{:,i}^{(1)\top}\boldsymbol{h}^{(1)}\right) \forall i. \quad (20.19)$$

- In the case of real-valued visible units, substitute

$$\mathbf{v} \sim \mathcal{N}\left(\boldsymbol{v}; \boldsymbol{b}^{(0)} + \boldsymbol{W}^{(1)\top}\boldsymbol{h}^{(1)}, \boldsymbol{\beta}^{-1}\right) \quad (20.20)$$

with β diagonal for tractability.

Deep belief networks

- To generate a sample from a DBN, we first run several steps of Gibbs sampling on the top two hidden layers. This stage is essentially drawing a sample from the RBM defined by the top two hidden layers. We can then use a single pass of ancestral sampling through the rest of the model to draw a sample from the visible units.

- Inference in a deep belief network is intractable. Evaluating or maximizing the log-likelihood requires confronting not just the problem of intractable inference to marginalize out the latent variables, but also the problem of an intractable partition function within the undirected model of the top two layers. Evaluating or maximizing the standard evidence lower bound on the log-likelihood is also intractable.

Deep belief networks

- To train a deep belief network, one begins by training an RBM to maximize $\mathbb{E}_{\mathbf{v}\sim p_{\text{data}}} \log p(\boldsymbol{v})$ using contrastive divergence or stochastic maximum likelihood. The parameters of the RBM then define the parameters of the first layer of the DBN.

- A second RBM is trained to approximately maximize

$$\mathbb{E}_{\mathbf{v}\sim p_{\text{data}}} \mathbb{E}_{\mathbf{h}^{(1)}\sim p^{(1)}(\boldsymbol{h}^{(1)}|\boldsymbol{v})} \log p^{(2)}(\boldsymbol{h}^{(1)}), \tag{20.21}$$

where $p^{(1)}$ is the probability distribution represented by the first RBM, and $p^{(2)}$ is the probability distribution represented by the second RBM. In other words, the second RBM is trained to model the distribution defined by sampling the hidden units of the first RBM, when the first RBM is driven by the data.

Deep belief networks

- This procedure can be repeated indefinitely, to add as many layers to the DBN as desired, with each new RBM modeling the samples of the previous one. Each RBM defines another layer of the DBN. This procedure can be justified as increasing a variational lower bound on the log-likelihood of the data under the DBN.

- In most applications, no effort is made to jointly train the DBN after the greedy layer-wise procedure is complete. However, it is possible to perform generative fine-tuning using the wake-sleep algorithm.

Deep belief networks

- The trained DBN may be used directly as a generative model, but most of the interest in DBNs arose from their ability to improve classification models. We can take the weights from the DBN and use them to define an MLP:

$$\boldsymbol{h}^{(1)} = \sigma\left(b^{(1)} + \boldsymbol{v}^{\top}\boldsymbol{W}^{(1)}\right), \tag{20.22}$$

$$\boldsymbol{h}^{(l)} = \sigma\left(b_i^{(l)} + \boldsymbol{h}^{(l-1)\top}\boldsymbol{W}^{(l)}\right) \forall l \in 2, \ldots, m. \tag{20.23}$$

- After initializing this MLP with the weights and biases learned via generative training of the DBN, we can train the MLP to perform a classification task. This additional training of the MLP is an example of discriminative fine-tuning. This specific choice of MLP is somewhat arbitrary.

Deep Boltzmann machines

- A deep Boltzmann machine, or DBM is another kind of deep generative model. Unlike the DBN, it is an entirely undirected model. Unlike the RBM, the DBM has several layers of latent variables (RBMs have just one).

- Like the RBM, within each layer, each of the variables are mutually independent, conditioned on the variables in the neighboring layers.

- Like RBMs and DBNs, DBMs typically contain only binary units—as we assume for simplicity of our presentation of the model—but it is straightforward to include real-valued visible units.

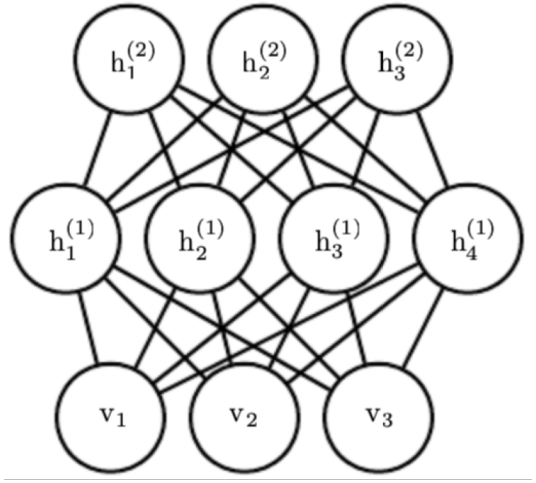
Deep Boltzmann machines

- In the case of a deep Boltzmann machine with one visible layer, $\boldsymbol{v}$, and three hidden layers, $\boldsymbol{h}^{(1)}$, $\boldsymbol{h}^{(2)}$, and $\boldsymbol{h}^{(3)}$, the joint probability is given by

$$P\left(\boldsymbol{v}, \boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, \boldsymbol{h}^{(3)}\right) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(-E(\boldsymbol{v}, \boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, \boldsymbol{h}^{(3)}; \boldsymbol{\theta})\right). \quad (20.24)$$

- To simplify our presentation, we omit the bias parameters below. The DBM energy function is then defined as follows:

$$E(\boldsymbol{v}, \boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, \boldsymbol{h}^{(3)}; \boldsymbol{\theta}) = -\boldsymbol{v}^{\top} \boldsymbol{W}^{(1)} \boldsymbol{h}^{(1)} - \boldsymbol{h}^{(1)\top} \boldsymbol{W}^{(2)} \boldsymbol{h}^{(2)} - \boldsymbol{h}^{(2)\top} \boldsymbol{W}^{(3)} \boldsymbol{h}^{(3)}. \quad (20.25)$$



Deep Boltzmann machines

- In comparison to the RBM energy function, the DBM energy function includes connections between the hidden units (latent variables) in the form of the weight matrices ($\boldsymbol{W}^{(2)}$ and $\boldsymbol{W}^{(3)}$).

- In comparison to fully connected Boltzmann machines, the DBM offers some advantages that are similar to those offered by the RBM. The DBM layers can be organized into a bipartite graph, with odd layers on one side and even layers on the other. This immediately implies that when we condition on the variables in the even (odd) layer, the variables in the odd (even) layers become conditionally independent.

Deep Boltzmann machines

- The bipartite structure of the DBM means that we can apply the same equations we have previously used for the conditional distributions of an RBM to determine the conditional distributions in a DBM.

- The units within a layer are conditionally independent from each other given the values of the neighboring layers, so the distributions over binary variables can be fully described by the Bernoulli parameters, giving the probability of each unit being active. In our example with two hidden layers, the activation probabilities are

$$P(v_i = 1 \mid \boldsymbol{h}^{(1)}) = \sigma\left(\boldsymbol{W}^{(1)}_{i,:}\boldsymbol{h}^{(1)}\right), \tag{20.26}$$

$$P(h^{(1)}_i = 1 \mid \boldsymbol{v}, \boldsymbol{h}^{(2)}) = \sigma\left(\boldsymbol{v}^\top \boldsymbol{W}^{(1)}_{:,i} + \boldsymbol{W}^{(2)}_{i,:}\boldsymbol{h}^{(2)}\right), \tag{20.27}$$

$$P(h^{(2)}_k = 1 \mid \boldsymbol{h}^{(1)}) = \sigma\left(\boldsymbol{h}^{(1)\top} \boldsymbol{W}^{(2)}_{:,k}\right). \tag{20.28}$$

Deep Boltzmann machines

- The bipartite structure makes Gibbs sampling in a deep Boltzmann machine efficient. In RBMs, it is possible to update all the units in only two iterations. RBMs allow all the visible units to be updated in one block and all the hidden units to be updated in a second block.

- Gibbs sampling can be divided into two blocks of updates, one including all even layers (including the visible layer) and the other including all odd layers. Because of the bipartite DBM connection pattern, given the even (odd) layers, the distribution over the odd (even) layers is factorial and thus can be sampled simultaneously and independently as a block.