# Boltzmann Machine

Tai, Wei Hsuan

December 14, 2025

# Outline

1. What is a Boltzmann Machine(BM)?
2. Why does lower energy mean higher probability?
3. How does BM learn?
4. Why BM is considered more brain-like than Backpropagation(BP)?
5. What have we learned from BMs?
6. My Thoughts

# What is a Boltzmann Machine(BM)?

# Basic concept

Boltzmann Machine (BM) is an undirected stochastic neural network that can learn a probability distribution over the whole network states, including visible and hidden units. It is named after the Boltzmann distribution in statistical mechanics, which describes the distribution of energy states in a system at thermal equilibrium.

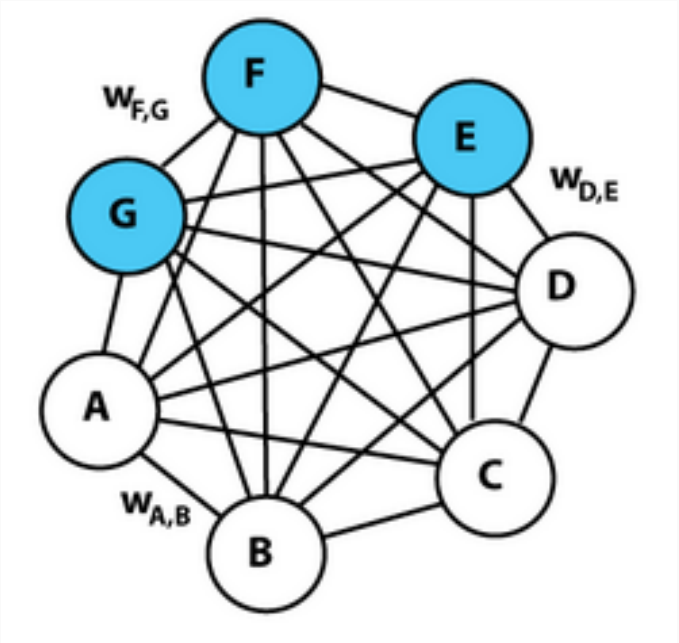


**Figure 1:** BM utilizes a network structure to model complex relationships between variables.(Source: wikipedia)

# Theorem Foundations

A BM is a binary-valued network, set $s_i$ as the state of unit $i$, which can be either 0 or 1. For the i-th unit of the network, its total input is given by:

$$z_i = b_i + \sum_j s_j w_{ij}$$

where $b_i$ is the bias of unit $i$, $w_{ij}$ is the weight between unit $i$ and unit $j$, and $s_j$ is the state of unit $j$. Then it would update its state according to the following probability:

$$P(s_i = 1) = \frac{1}{1 + e^{-z_i}}$$

Note that the main idea of this update is not sigmoid activation, it's just looking similar.

# Theorem fundations(cont.)

If all the units are updated many times, the network will reach thermal equilibrium, and the probability of the network being in a certain state $s$ is given by the Boltzmann distribution:

$$P(s) = \frac{e^{-E(s)}}{\sum_{s'} e^{-E(s')}}$$

where $E(s)$ is the energy of state $s$, defined as:

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

# Why does lower energy mean higher probability?

# Statistical Physics

The content above are striving to maximize the probability by minimizing the energy, but why do lower energy states have higher probability? This can be explained by statistical physics.

Statistical physics describes the behavior of systems with a large number of particles, the interactions between particles lead to different energy states. The BM model borrows this concept to model the interactions between neurons in a neural network. $s_i$ can be thought of as the state of a particle, and the weights $w_{ij}$ represent the interactions between particles.

# Boltzmann Distribution

In statistical physics, the Boltzmann distribution describes the probability of a system being in a certain energy state at thermal equilibrium. The probability of a system being in a state with energy $E$ is given by:

$$P(E) = \frac{e^{-E/kT}}{Z}$$

where $k$ is the Boltzmann constant, $T$ is the temperature, and $Z$ is the partition function, which normalizes the probabilities. In the context of BM, we can set $kT = 1$ for simplicity, leading to the earlier mentioned formula:

$$P(s) = \frac{e^{-E(s)}}{Z}, Z = \sum_{s'} e^{-E(s')}$$

This shows that states with lower energy have higher probabilities, as the exponential function decreases rapidly with increasing energy.

# How does BM learn?

# Target for BMs

The overall target of training a BM is to adjust the weights and biases to minimize the difference between the model's distribution and the target distribution. It can be represented as:

$$\max_{w} \mathbb{E}_{v \sim P_{data}} [\log P_{model}(v)]$$

where $P_{data}$ is the target distribution, and $P_{model}$ is the distribution learned by the BM.

# How an original BM learns?

To achieve the target above, we can use gradient ascent to update the weights and biases. The gradient of the log-likelihood with respect to a weight $w_{ij}$ is given by:

$$\langle \frac{\partial \log P(v)}{\partial w_{ij}} \rangle_{data} = \langle s_i s_j \rangle_{data} - \langle s_i s_j \rangle_{model}$$

where $\langle s_i s_j \rangle_{data}$ is the expected value of the product of states $s_i$ and $s_j$ under the data distribution, and $\langle s_i s_j \rangle_{model}$ is the expected value under the model's distribution. The weights are then updated as follows:

$$w_{ij} \leftarrow w_{ij} + \eta (\langle s_i s_j \rangle_{data} - \langle s_i s_j \rangle_{model})$$

where $\eta$ is the learning rate. Note that $s_i$ and $s_j$ are the states of units $i$ and $j$, respectively. Thus, the weight update is based on the difference between the correlations of the units in the data and the model.

# Restricted Boltzmann Machine(RBM)

However, computing $\langle s_i s_j \rangle_{model}$ requires sampling from the model's equilibrium distribution, which is computationally expensive. To address this problem, a simplified version of BM called Restricted Boltzmann Machine (RBM) is introduced.

RBM restricts the connections between units, allowing only connections between visible and hidden units, but not within the same layer. This restriction simplifies the learning process and makes it more efficient.

# Two phases of learning

Here are the two phases of RBM learning:

- Positive phase: In this phase, the visible units are clamped to the data, and the hidden units are updated based on the visible units. The correlations $\langle s_i s_j \rangle_{data}$ are computed during this phase.
- Negative phase: In this phase, the visible units are updated based on the hidden units learned in the positive phase, and the hidden units are updated based on the visible units. The correlations $\langle s_i s_j \rangle_{model}$ are computed during this phase.

In short:
Positive phase: $data \rightarrow v_{data} \rightarrow h_{data}$
Negative phase: $h_{data} \rightarrow v_{model} \rightarrow h_{model}$

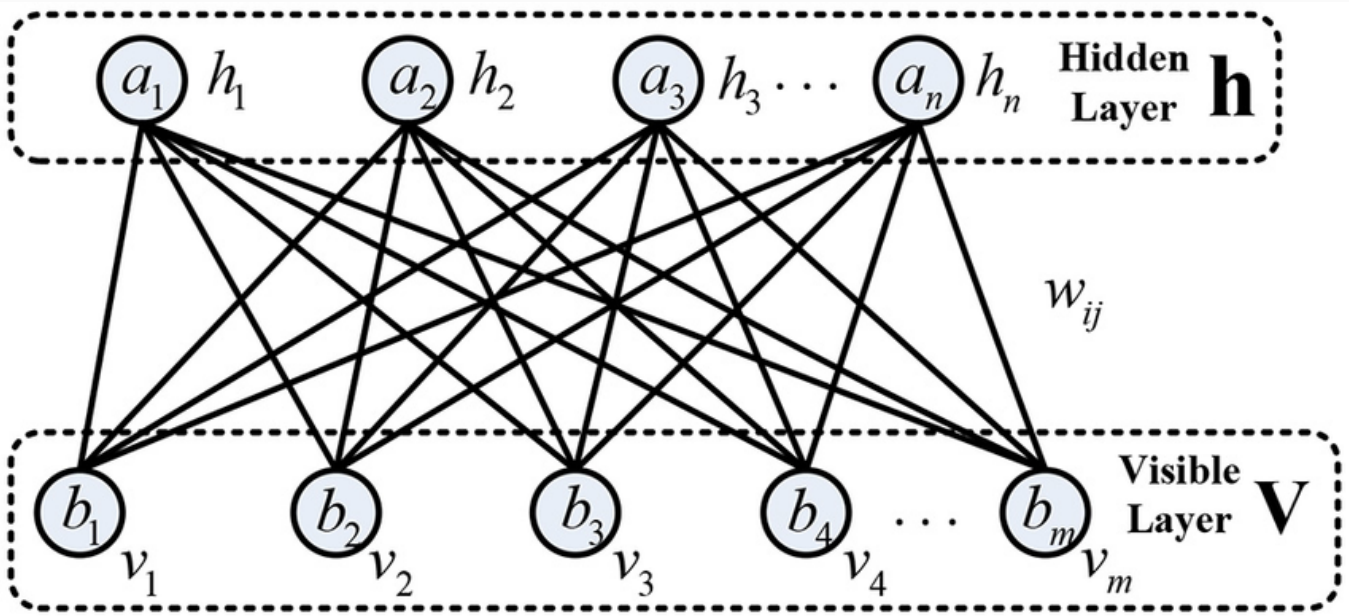


**Figure 2:** RBM learning process consists of two phases: positive phase and negative phase.(doi: 10.3389/fnins.2018.00680)

# Contrastive Divergence(CD)

The learning rule of Boltzmann Machines consists of a positive phase and a negative phase. While the positive phase can be computed exactly, the negative phase requires sampling from the model's equilibrium distribution, which is computationally prohibitive.

Thus, Hinton proposed a method called Contrastive Divergence (CD) to approximate the negative phase. CD performs a small number of Gibbs sampling steps (often just one step, known as CD-1) starting from the data distribution to obtain samples for the negative phase. This approximation significantly speeds up the learning process while still providing good results.

# CD (cont.)

Given a data vector $v^{(0)}$, the positive phase samples the hidden units according to:

$$p(h_j = 1 \mid v^{(0)}) = \sigma\left(b_j + \sum_i v_i^{(0)} w_{ij}\right)$$

Using the sampled hidden units $h^{(0)}$, a reconstruction of the visible units is generated in the negative phase by:

$$p(v_i^{(1)} = 1 \mid h^{(0)}) = \sigma\left(a_i + \sum_j h_j^{(0)} w_{ij}\right)$$

Finally we can sample the hidden units again from the reconstructed visible units:

$$p(h_j^{(1)} = 1 \mid v^{(1)}) = \sigma\left(b_j + \sum_i v_i^{(1)} w_{ij}\right)$$

# Parameter update

With the samples $v^{(0)}, h^{(0)}, v^{(1)}, h^{(1)}$, the weight update rule becomes:

$$w_{ij} \leftarrow w_{ij} + \eta(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{recon})$$

where $\langle v_i h_j \rangle_{data}$ is computed using $v^{(0)}$ and $h^{(0)}$, and $\langle v_i h_j \rangle_{recon}$ is computed using $v^{(1)}$ and $h^{(1)}$.

# How does a BM work?

We put the label of the data as a part of visible units while training the BM. During prediction, we clamp the known visible units (features) and let the hidden units and unknown visible units (labels) be updated through Gibbs sampling. After several iterations, the states of the unknown visible units can be used as the predicted labels.

While executing generation, we can also clamp all the visible units except for some units (e.g. label) to generate new samples.

# Another Usage: RBM for Representation Learning

The hidden units of an RBM learn latent representations that capture meaningful structure in the input data, such as strokes and local patterns. These representations can be reused as features for downstream tasks.

I've done an experiment that deploying a single-layer RBM as an unsupervised feature extractor for the MNIST dataset. After pretraining the RBM, the learned hidden representations are fed into a linear classifier trained with backpropagation. This setup allows us to evaluate the quality of the representations learned by the RBM. Following is the result

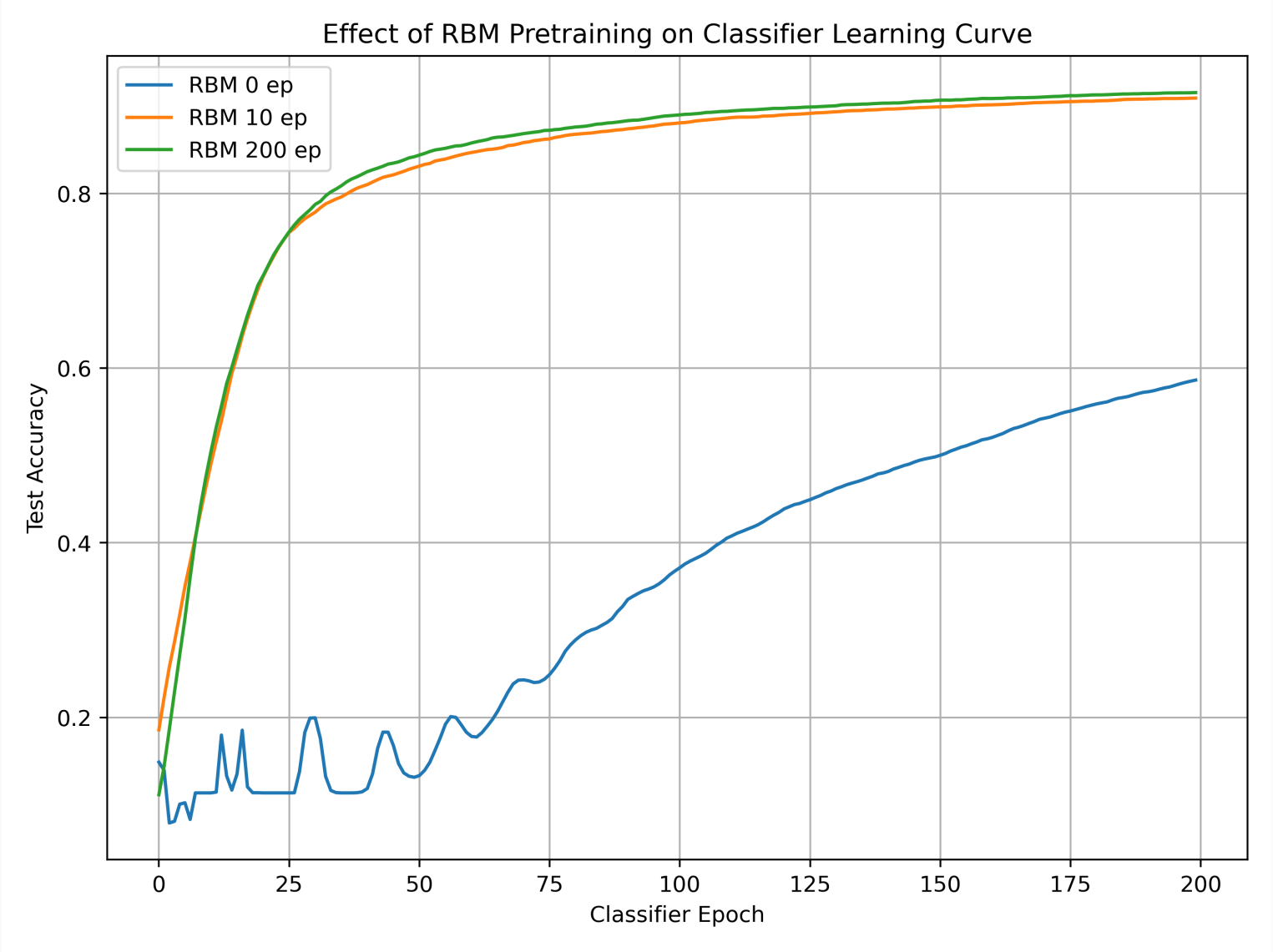


**Figure 3:** RBM as feature extractor on MNIST dataset.

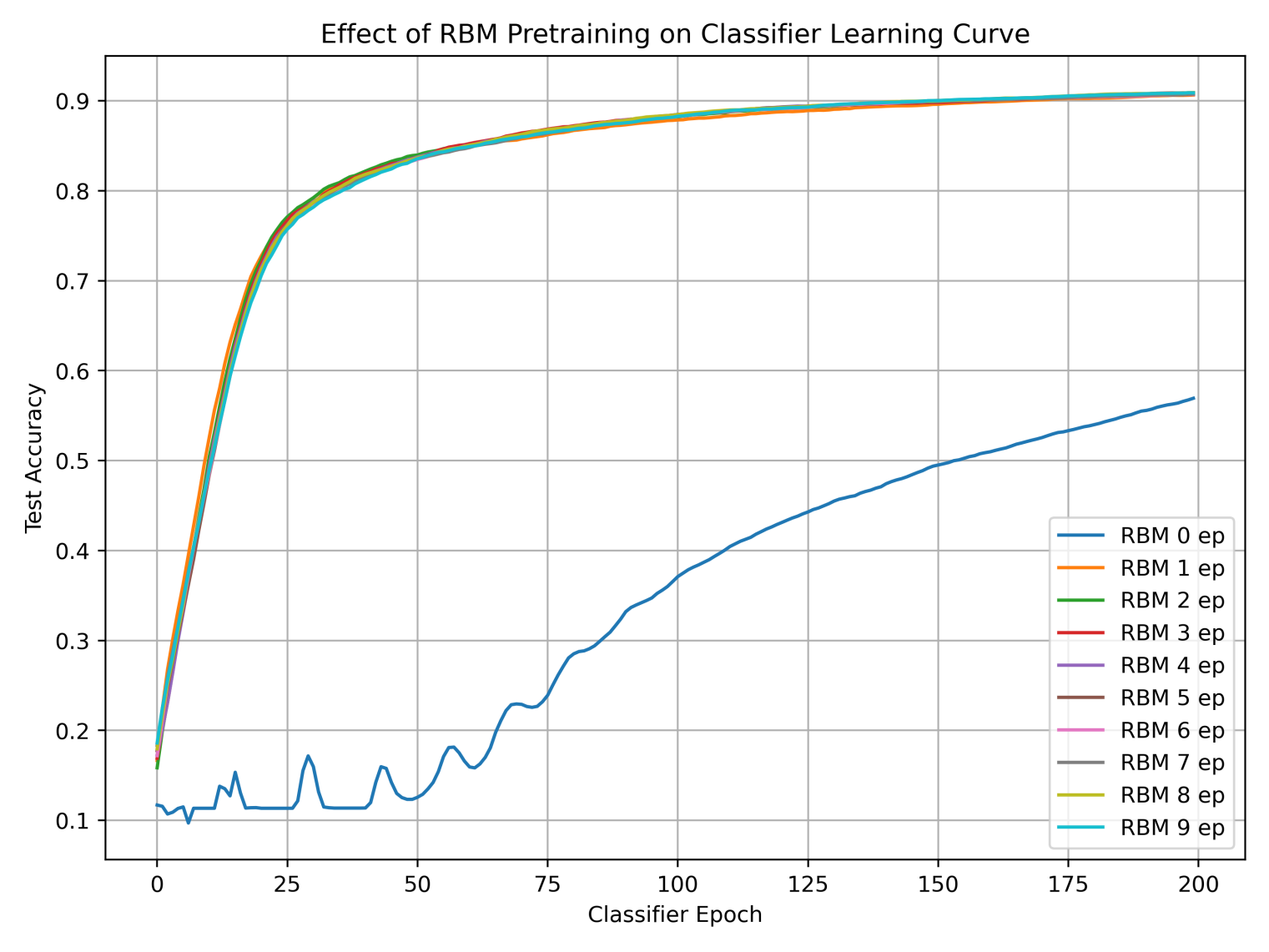


**Figure 4:** RBM as feature extractor on MNIST dataset.

# Why BM is considered more brain-like than Backpropagation(BP)?

# What is Backpropagation (BP)?

Backpropagation is a learning algorithm used to train artificial neural networks by minimizing a global loss function. It computes the gradient of the loss with respect to each weight by applying the chain rule of calculus and propagating error signals backward through the network.

To conclude it, weight updates in BP depend on:

- A globally defined error signal
- Precise gradient information
- Symmetric forward and backward weight paths

However, there is little biological evidence that the brain implements such global error backpropagation mechanisms.

# Hebbian Learning

Hebbian learning is a biologically motivated learning rule stating that the connection between two neurons is strengthened when they are co-activated. It depends only on local information, namely the activities of the pre- and post-synaptic neurons.

Unlike backpropagation, Hebbian learning does not require a global error signal or backward propagation of gradients, making it more biologically plausible.

Importantly, the learning rule of Boltzmann Machines can be interpreted as a form of **contrastive Hebbian learning**, where synaptic updates are driven by the difference between neural correlations observed in the positive (data) and negative (model) phases.

# The difference between brain and BM

- There's no evidence that the brain is minimizing a global energy function like BMs do.
- The stochastic methods.
- BMs require symmetric weights, which is not biologically realistic.

# What have we learned from BMs?

# Why were Boltzmann Machines popular in the early days?

- Probabilistic foundation for neural networks
- Principled unsupervised learning
- Avoidance of the vanishing gradient problem
- Biological and physical plausibility
- Foundation for deep learning pretraining

# Why are BMs rarely used in practice today?

- Computation complexity
- Difficulty in training deep architectures
- Emergence of more efficient models

# BM in nowadays

- EBM, DBN
- Transfer learning
- Diffusion model
- GAN
- VAE

# My Thoughts

# References


- Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for Boltzmann Machines. Cognitive Science, 9(1), 147-169. https://doi.org/10.1207/s15516709cog0901_7
- Hinton, G.E. (2012). A Practical Guide to Training Restricted Boltzmann Machines. In: Montavon, G., Orr, G.B., Müller, KR. (eds) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol 7700. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-35289-8_32
- http://www.scholarpedia.org/article/Boltzmann_machine
- Hu, H., Gao, L., & Ma, Q. (2016). Deep Restricted Boltzmann Networks. ArXiv, abs/1611.07917.
- Rachael L. Sumner, Meg J. Spriggs, Suresh D. Muthukumaraswamy, Ian J. Kirk, The role of Hebbian learning in human perception: a methodological and theoretical review of the human Visual Long-Term Potentiation paradigm, Neuroscience & Biobehavioral Reviews,Volume 115,2020,Pages 220-237,ISSN 0149-7634,https://doi.org/10.1016/j.neubiorev.2020.03.013.