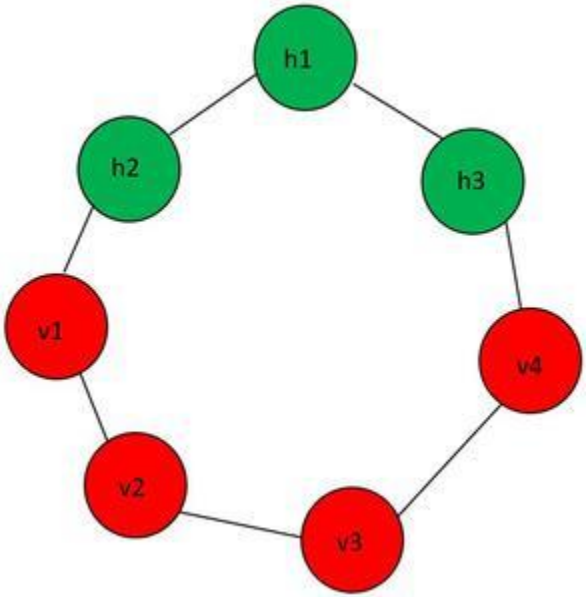
BOLTZMANN MACHINE

Dr. Amit Sinhal
Professor
Department of Computer Science & Engineering

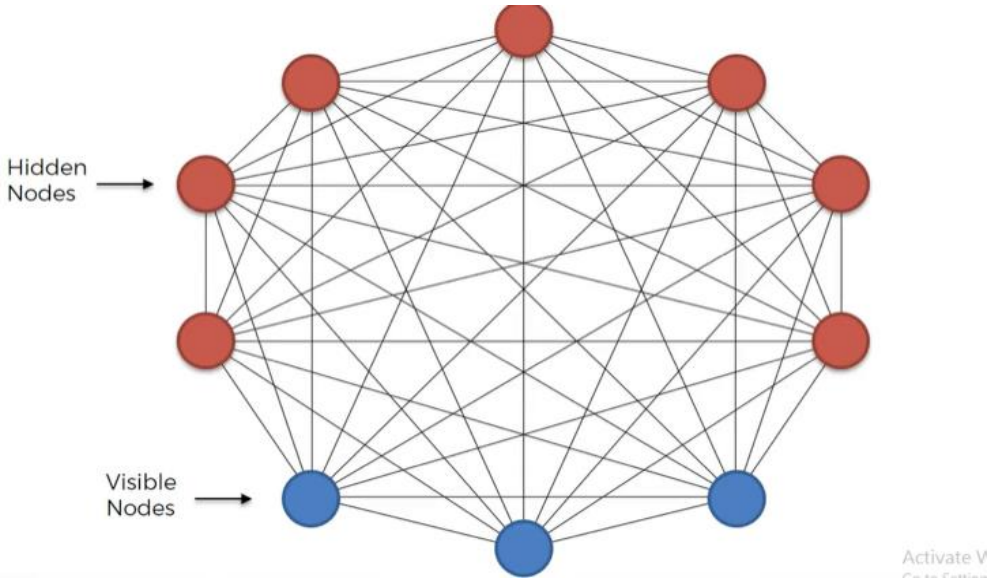
What is the Boltzmann Machine?

- A Boltzmann machine is a type of ANN that is inspired by the behavior of physical systems in **thermodynamics**. It was introduced by the Austrian physicist, **Ludwig Boltzmann**, in the 19th century.
- The visible layer is denoted as **v** and the hidden layer is denoted as the **h**. In Boltzmann machine, there is **no output layer**. Boltzmann machines are random and generative neural networks capable of learning internal representations and are able to represent and (given enough time) solve tough combinatory problems.



What is the Boltzmann Machine?

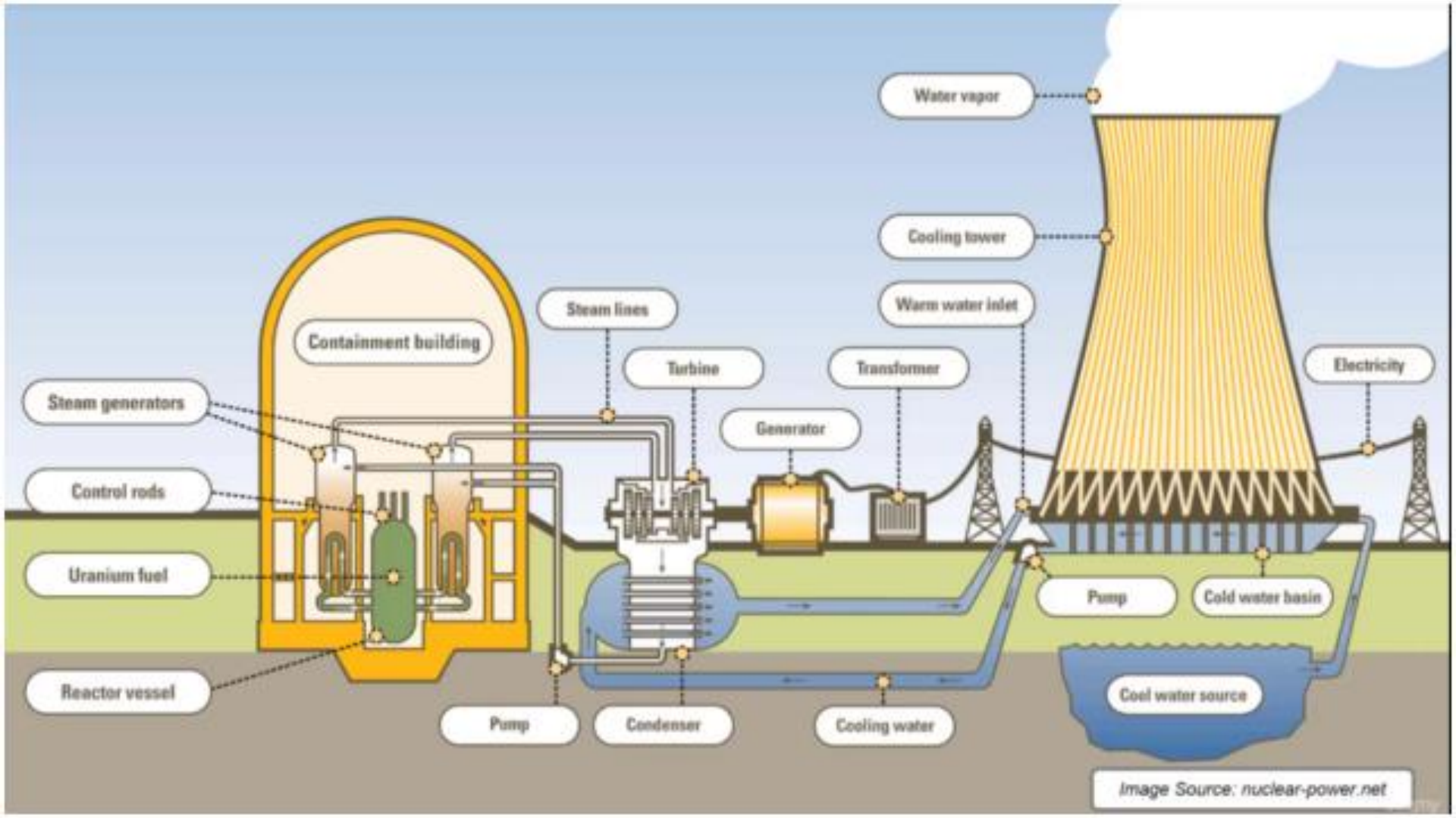
- The network uses a **stochastic (random)** process to activate the neurons, and the probability of a given state is calculated using the **Boltzmann distribution** (also known as **Gibbs Distribution**), which is a statistical distribution used in thermodynamics to describe the behavior of systems at thermal equilibrium.
- Boltzmann machines are particularly useful for unsupervised learning, where the goal is to discover the underlying structure of the data without any explicit labels or guidance. However, training Boltzmann machines can be computationally expensive.



What is the Boltzmann Machine?

- Boltzmann Machine is a generative unsupervised model, which involves learning a probability distribution from an original dataset and using it to make inferences about **never before seen data**.
- Boltzmann Machine has an input layer (also referred to as the **visible layer**) and one or several hidden layers (also referred to as the **hidden layer**).
- Boltzmann Machine uses neural networks with neurons that are connected not only to other neurons in other layers but also to neurons within the same layer.
- **Everything is connected to everything**. Connections are bidirectional, visible neurons connected to each other and hidden neurons also connected to each other
- Boltzmann Machine **doesn't expect input data**, it generates data. Neurons generate information regardless they are hidden or visible.
- For Boltzmann Machine all neurons are the same, it doesn't discriminate between hidden and visible neurons.

Nuclear Power Plant Example



How Boltzmann Machine Works?

- The Boltzmann machine uses a stochastic process to activate the neurons. The probability of a neuron being "**on**" or "**off**" is determined by the weighted sum of the inputs to that neuron, which is then passed through a sigmoid activation function. The sigmoid function ensures that the output of each neuron is between 0 and 1, which can be interpreted as the probability that the neuron is "on."
- During training, the Boltzmann machine learns the weights of the connections between neurons by minimizing an **energy function**. The energy function is a measure of how well the Boltzmann machine is able to reconstruct the input data. The goal of training is to minimize the difference between the actual input data and the reconstructed data produced by the Boltzmann machine.
- The training process involves a series of iterations, where the network is presented with input data, and the weights are adjusted to improve the reconstruction of the data. The Boltzmann machine uses a technique called **contrastive divergence** to update the weights.

Steps in Training a Boltzmann Machine

1. **Initialization**: The weights of the connections between the neurons are initialized randomly.
2. **Gibbs Sampling**: The Boltzmann machine uses Gibbs sampling to activate the neurons stochastically. Gibbs sampling involves repeatedly **updating the state of the neurons** in the network by **sampling from the probability distribution of the network**, given the current state.
3. **Compute the Energy**: After each Gibbs step, the **energy** of the current state of the network is computed. The energy function is a measure of how well the Boltzmann machine is **able to reconstruct the input data**.
4. **Update the weights**: The weights of the connections between the neurons are updated using a technique called **contrastive divergence**. Contrastive divergence involves computing the difference between the probabilities of the network being in a particular state before and after updating a single neuron. This difference is then used to adjust the weights of the connections between the neurons.
5. **Repeat steps 2-4**: Steps 2-4 are repeated for a fixed number of iterations, or until the **energy** of the network **converges** to a minimum.

Gibbs Sampling

Gibbs sampling is used to **update the state of the neurons** stochastically. The algorithm involves updating the state of a single neuron at a time, while holding the states of the other neurons fixed.

The algorithm proceeds as follows:

1. Initialize the state of the network by setting the states of the visible neurons to the input data, and the states of the hidden neurons to either 0 or 1.
2. Select a hidden neuron at random, and compute the probability that it is "on" given the states of the other neurons in the network.
3. Sample a new state for the selected neuron based on its probability of being "on."
4. Repeat steps 2-3 for each hidden neuron in the network.
5. Update the state of the visible neurons in the same way as the hidden neurons, by computing the probability that each visible neuron is "on" given the states of the other neurons in the network, and sampling a new state for each visible neuron.
6. Repeat steps 2-5 for a fixed number of iterations, or until the states of the neurons in the network converge to a stable distribution.

By repeating this process many times, the Boltzmann machine is **able to learn the probability distribution of the input data**. The probabilities of the network being in different states can be used to estimate the energy of the network, which is used to update the weights of the connections between the neurons during training.

Contrastive Divergence

- Contrastive divergence is a technique which is a variant of the stochastic gradient descent algorithm used to **train the weights** of a Boltzmann machine, based on a form of **approximate maximum likelihood estimation**. The basic idea is to adjust the weights of the connections between the neurons in the network to **minimize** the difference between the **distributions** of the input data and the data generated by the network.
- By iteratively **adjusting the weights** of the connections between the neurons to minimize this difference, the Boltzmann machine can learn the distribution of the input data.
- The intuition behind contrastive divergence is that the **gradient** of the log-likelihood of the data can be approximated by the difference between the **correlations** of the visible and hidden neurons under the training and reconstructed distributions.
- Contrastive divergence is a computationally efficient method for training Boltzmann machines, and it has been shown to work well in practice for a variety of tasks, such as image and text processing.

Steps in Contrastive Divergence

The contrastive divergence algorithm proceeds as follows:

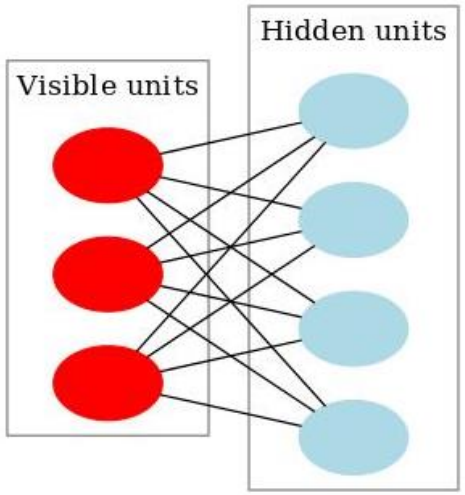
1. Initialize the network by setting the weights of the connections between the neurons randomly.
2. Sample a training example from the input data, and use Gibbs sampling to compute the probabilities of the states of the visible and hidden neurons in the network. The probabilities of the hidden neurons are used to generate a "**reconstructed**" sample of the input data.
3. Repeat step 2 for a fixed number of Gibbs sampling steps, typically only a few steps are used.
4. Compute the difference between the expected values of the products of the activations of the visible and hidden neurons, under the distributions defined by the training example and the reconstructed sample. This is known as the **contrastive divergence gradient**.
5. Update the weights of the connections between the neurons by following the gradient of the log-likelihood of the data.
6. Repeat steps 2-5 for a fixed number of training examples or until the weights **converge** to a minimum.

Types of Boltzmann Machine

Restricted Boltzmann Machine (RBM): An RBM is a type of Boltzmann machine with a bipartite graph structure, where the visible neurons are only connected to the hidden neurons, and vice versa. This makes the training of RBMs more efficient, as the hidden neurons can be updated independently of each other.

Deep Belief Networks (DBNs): These are a type of artificial neural network that consists of multiple layers of Restricted Boltzmann Machines (RBMs)

Deep Boltzmann Machine (DBM): A DBM is a type of Boltzmann machine that consists of multiple layers of hidden neurons, with connections between adjacent layers but not within the same layer. DBMs can learn more complex representations of the input data than RBMs, but they are also more difficult to train and require more computational resources.

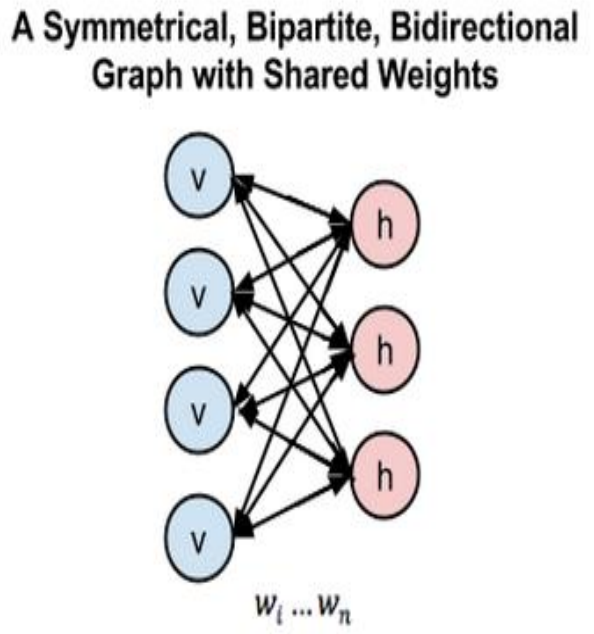


Restricted Boltzmann Machine (RBM)

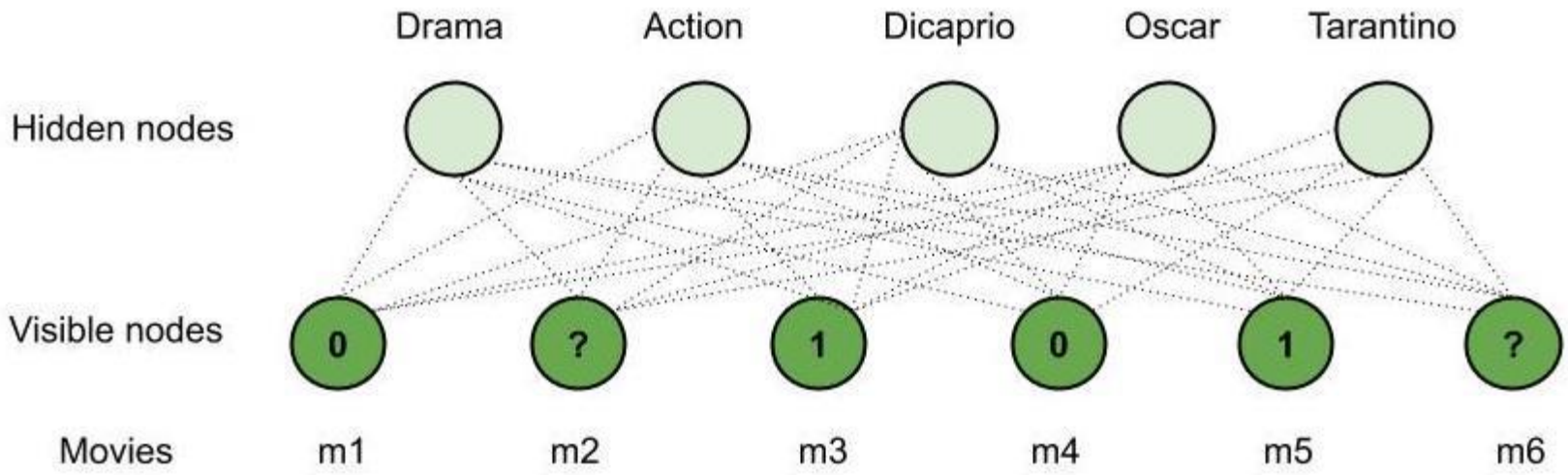
- An RBM is a type of Boltzmann machine with a **bipartite** graph structure, where the visible neurons are only connected to the hidden neurons, and vice versa.
- The RBM is called "**restricted**" because the connections between the neurons in the same layer are not allowed. In other words, each neuron in the visible layer is only connected to neurons in the hidden layer, and vice versa. This allows the RBM to learn a compressed representation of the input data by reducing the dimensionality of the input.
- Movies like Avengers, Avatar, and Interstellar have strong associations with the latest fantasy and science fiction factor. Based on the user rating RBM will discover latent factors that can explain the activation of movie choices. In short, RBM describes variability among correlated variables of input dataset in terms of a potentially lower number of unobserved variables.

Key Characteristics of RBM

- It is a probabilistic, unsupervised, generative deep machine learning algorithm.
- RBM is the neural network that belongs to the energy-based model
- RBM's objective is to find the joint probability distribution that maximizes the log-likelihood function.
- In their learning process, RBMs attempt to link low energy states with high probability ones and vice versa.
- There are no connections between the layers.
- They use recurrent and symmetric structure.



Working of RBM – Illustrative Example



Consider, Mary watches four movies out of the six available movies and rates four of them.

Say, she watched m1, m3, m4 and m5 and likes m3, m5 (rated 1) and dislikes the other two, that is m1, m4 (rated 0) whereas the other two movies – m2, m6 are unrated.

Working of RBM – Illustrative Example

- Now, using our RBM, we will recommend one of these movies for her to watch next. Say –
- m3, m5 are of ‘Drama’ genre.
- m1, m4 are of ‘Action’ genre.
- ‘Dicaprio’ played a role in m5.
- m3, m5 have won ‘Oscar.’
- ‘Tarantino’ directed m4.
- m2 is of the ‘Action’ genre.
- m6 is of both the genres ‘Action’ and ‘Drama’, ‘Dicaprio’ acted in it and it has won an ‘Oscar’.

How RBM Works ?

We have the following observations –

- Mary likes m3, m5 and they are of genre ‘Drama,’ she probably likes ‘Drama’ movies.
- Mary dislikes m1, m4 and they are of action genre, she probably dislikes ‘Action’ movies.
- Mary likes m3, m5 and they have won an ‘Oscar’, she probably likes an ‘Oscar’ movie.
- Since ‘Dicaprio’ acted in m5 and Mary likes it, she will probably like a movie in which ‘Dicaprio’ acted.
- Mary does not like m4 which is directed by Tarantino, she probably dislikes any movie directed by ‘Tarantino’.

Therefore, based on the observations and the details of m2, m6; our RBM recommends m6 to Mary (‘Drama’, ‘Dicaprio’ and ‘Oscar’ matches both Mary’s interests and m6). This is how an RBM works and hence is used in recommender systems.

How RBM Works ?

In RBM there are two phases :

1st Phase: In this phase, we take the input layer and using the concept of weights and biased we are going to activate the hidden layer. This process is said to be **Feed Forward Pass**. In Feed Forward Pass we are identifying the positive association and negative association.

- **Positive Association:** When the association between the visible unit and the hidden unit is positive.
- **Negative Association:** When the association between the visible unit and the hidden unit is negative.

2nd Phase: As we don't have any output layer. Instead of calculating the output layer, we are reconstructing the input layer through the activated hidden state. This process is said to be **Feed Backward Pass**. We are just backtracking the input layer through the activated hidden neurons. After performing this we have reconstructed Input through the activated hidden state. So, we can calculate the error and adjust weight in this way:

Error = Reconstructed Input Layer-Actual Input layer

Adjust Weight = Input*error*learning rate (0.1)

Types of RBM

There are mainly two types of Restricted Boltzmann Machine (RBM) based on the types of variables they use:

- **Binary RBM**: In a binary RBM, the input and hidden units are binary variables. Binary RBMs are often used in modeling binary data such as images or text.
- **Gaussian RBM**: In a Gaussian RBM, the input and hidden units are continuous variables that follow a Gaussian distribution. Gaussian RBMs are often used in modeling continuous data such as audio signals or sensor data.

Variations of RBM

- **Deep Belief Network (DBN)**: A DBN is a type of generative model that consists of multiple layers of RBMs. DBNs are often used in modeling high-dimensional data such as images or videos.
- **Convolutional RBM (CRBM)**: A CRBM is a type of RBM that is designed specifically for processing images or other grid-like structures. In a CRBM, the connections between the input and hidden units are local and shared, which makes it possible to capture spatial relationships between the input units.
- **Temporal RBM (TRBM)**: A TRBM is a type of RBM that is designed for processing temporal data such as time series or video frames. In a TRBM, the hidden units are connected across time steps, which allows the network to model temporal dependencies in the data.

Advantages and Disadvantages of RBM

Advantages :

- Expressive enough to encode any distribution and computationally efficient.
- Faster than traditional Boltzmann Machine due to the restrictions in terms of connections between nodes.
- Activations of the hidden layer can be used as input to other models as useful features to improve performance.

Disadvantages :

- Training is more difficult as it is difficult to calculate the Energy gradient function.
- CD-k algorithm used in RBMs is not as familiar as the back propagation algorithm.
- Weight Adjustment.

Applications of RBM

- **Collaborative filtering**: RBMs are widely used in collaborative filtering for recommender systems. They learn to predict user preferences based on their past behavior and recommend items that are likely to be of interest to the user.
- **Image and video processing**: RBMs can be used for image and video processing tasks such as object recognition, image de-noising, and image reconstruction. They can also be used for tasks such as video segmentation and tracking.
- **Natural Language Processing**: RBMs can be used for natural language processing tasks such as language modeling, text classification, and sentiment analysis. They can also be used for tasks such as speech recognition and speech synthesis.
- **Bioinformatics**: RBMs have found applications in bioinformatics for tasks such as protein structure prediction, gene expression analysis, and drug discovery.
- **Financial Modeling**: RBMs can be used for financial modeling tasks such as predicting stock prices, risk analysis, and portfolio optimization.
- **Anomaly detection**: RBMs can be used for anomaly detection tasks such as fraud detection in financial transactions, network intrusion detection, and medical diagnosis.
- It is used in Filtering, Feature Learning, Classification, Risk Detection, Business and Economic analysis.

Deep Belief Networks (DBNs)

Consider stacking numerous RBMs so that the outputs of the first RBM serve as the input for the second RBM, and so forth. Deep Belief Networks are the name given to these networks. Each layer's connections are undirected (as each layer is an RBM). Those between the strata are simultaneously directed (except for the top two layers – whose connections are undirected). The DBNs can be trained in two different ways:

- **Greedy Layer-wise Training Algorithm**: RBMs are trained using a greedy layer-by-layer training algorithm. The orientation between the DBN layers is established as soon as the individual RBMs have been trained (i.e., the parameters, weights, and biases, have been defined).
- **Wake-sleep Algorithm**: The DBN is trained from the bottom up using a wake-sleep algorithm (connections going up indicate wake), and then from the bottom up using connections indicating sleep.
- In order to ensure that the layer connections only work downwards, we stack the RBMs, train them, and then do so (except for the top two layers).