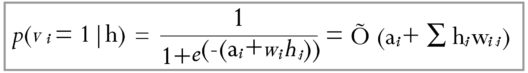
**Training of Restricted Boltzmann Machine**

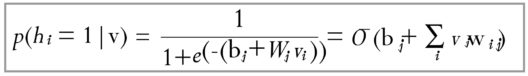
The training of the Restricted Boltzmann Machine differs from the training of regular **neural networks** via stochastic gradient descent.

The Two main Training steps are:

* **Gibbs Sampling**

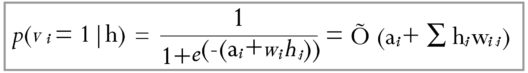
The first part of the training is called *Gibbs Sampling*. Given an input vector **v** we use **p(h|v)**for prediction of the hidden values **h.**Knowing the hidden values we use **p(v|h)** :

for prediction of new input values **v**. This process is repeated *k* times. After *k* iterations, we obtain another input vector **v\_k**which was recreated from original input values **v\_0**.

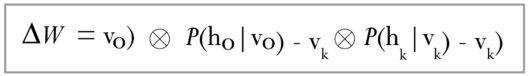
****

* **Contrastive Divergence step**

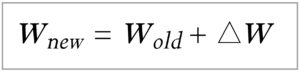
The update of the weight matrix happens during the *Contrastive Divergence*step. Vectors **v\_0** and **v\_k**are used to calculate the activation probabilities for hidden values **h\_0**and **h\_k :**



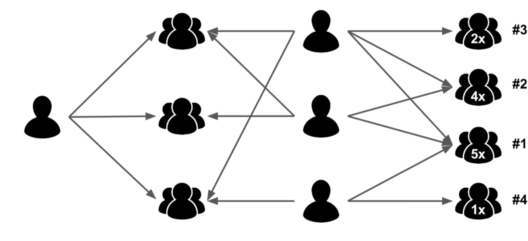
The difference between the outer products of those probabilities with input vectors **v\_0**and **v\_k**results in the updated matrix :



Using the update matrix the new weights can be calculated with gradient **ascent,**given by:

Now that you have an idea of what are Restricted Boltzmann Machines and the layers of RBM, let’s move on with our Restricted Boltzmann Machine Tutorial and understand their working with the help of an example.

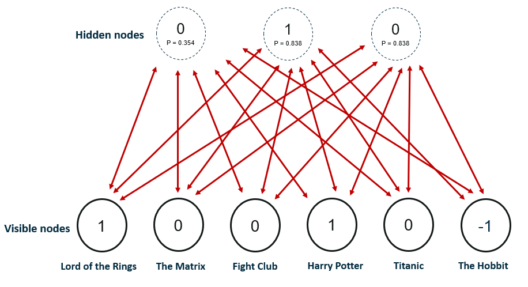
## ****Restricted Boltzmann Machine : Collaborative Filtering****



RBMs have found applications in dimensionality reduction, classification, collaborative filtering and many more. They can be trained in either supervised or unsupervised ways, depending on the task.

### **Recognizing Latent factors in the Data**

Let us assume that some people were asked to rate a set of movies in the scale of 1-5 and each movie could be explained in terms of a set of latent factors such as drama, fantasy, action and many more. Restricted Boltzmann Machines are used to analyze and find out these underlying factors.

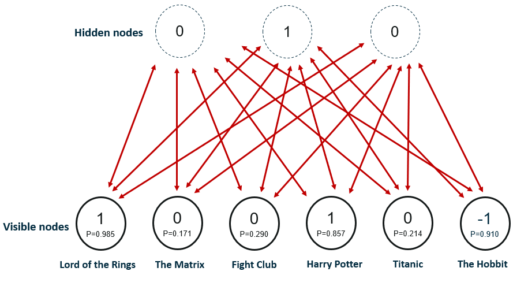


The analysis of hidden factors is performed in a binary way, i.e, the user only tells if they liked (rating 1) a specific movie or not (rating 0) and it represents the inputs for the input/visible layer. Given the inputs, the RMB then tries to discover latent factors in the data that can explain the movie choices and each hidden neuron represents one of the latent factors.

Let us consider the following example where a user likes *Lord of the Rings and* Harry Potter but does not like The Matrix, Fight Club and Titanic. The Hobbit has not been seen yet so it gets a -1 rating. Given these inputs, the Boltzmann Machine may identify three hidden factors Drama, Fantasy and Science Fiction which correspond to the movie genres.

### **Using Latent Factors for Prediction**

After the training phase, the goal is to predict a binary rating for the movies that had not been seen yet. Given the training data of a specific user, the network is able to identify the latent factors based on the user’s preference and sample from Bernoulli distribution can be used to find out which of the visible neurons now become active.



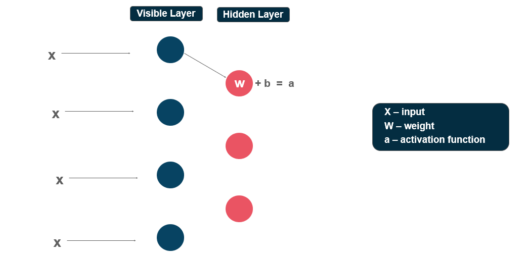
The image shows the new ratings after using the hidden neuron values for the inference. The network identified Fantasy as the preferred movie genre and rated The Hobbit as a movie the user would like.

The process from **training** to the **prediction** phase goes as follows:

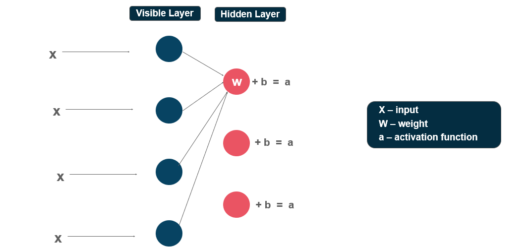
* Train the network on the data of all users
* During inference-time, take the training data of a specific user
* Use this data to obtain the activations of hidden neurons
* Use the hidden neuron values to get the activations of input neurons
* The new values of input neurons show the rating the user would give yet unseen movies

## ****Working of Restricted Boltzmann Machine****

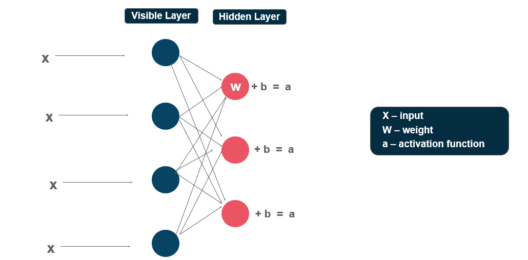
Each visible node takes a low-level feature from an item in the dataset to be learned. At node 1 of the hidden layer, **x is multiplied** by a **weight**and added to a **bias**. The result of those two operations is fed into an **activation function**, which produces the node’s output, or the strength of the signal passing through it, given input x.



Next, let’s look at how several inputs would combine at one hidden node. Each **x is multiplied** by a separate weight, the products are summed, added to a bias, and again the result is passed through an activation function to produce the node’s output.



At each hidden node, each input**x is multiplied** by its respective weight w. That is, a single input x would have three weights here, making 12 weights altogether (4 input nodes x 3 hidden nodes). The weights between the two layers will always form a matrix where the rows are equal to the input nodes, and the columns are equal to the output nodes.



Each hidden node receives the four inputs multiplied by their respective weights. **The sum** of those products is again **added to a bias** (which forces at least some activations to happen), and the result is passed through the activation algorithm producing one output for each hidden node.

Now that you have an idea about how Restricted Boltzmann Machine works, let’s continue our Restricted Boltzmann Machine Tutorial and have a look at the steps involved in the training of RBM.