Movie Rating Prediction using RBM

http://www.cs.toronto.edu/~rsalakhu/papers/rbmcf.pdf

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.228.3905&rep=rep1&type=pdf

Dhruv Khattar - 201402087
Pinkesh Badjatiya - 201402002
Siddhartha Gairola - 201402068

Project Goal

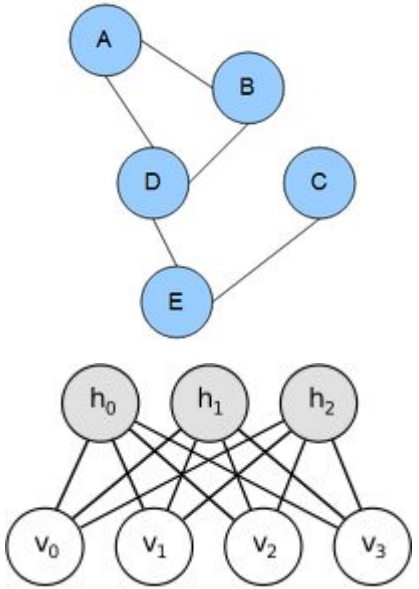
- The existing approaches to collaborative filtering are unable to handle huge data sets. So we use a class of 2 layer undirected graphical model called **Restricted Boltzmann Machines** to model tabular data - which in our case is Movie Ratings given by users. We can use the data sets available on **Netflix** or IMDB.
- From the above we aim to predict the ratings for the movies which have not yet been given a rating by the users.

What is RBM, EBM and BM and Bayesian nets?

$$p(x) = \frac{e^{-E(x)}}{Z}.$$

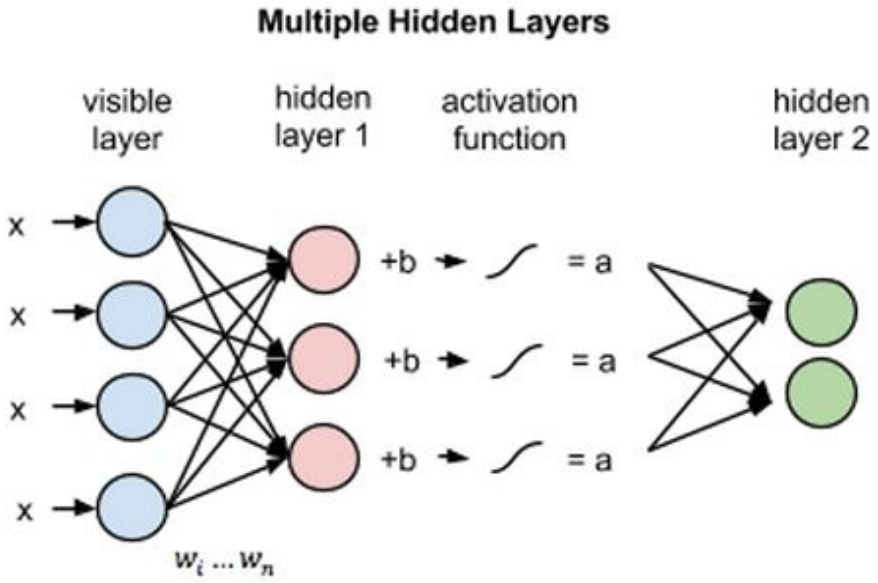
$$P(x) = \sum_h P(x,h) = \sum_h \frac{e^{-E(x,h)}}{Z}.$$

- Energy Based Models
- EBM with Hidden units
- Boltzmann machine or **a undirected graphical model, is a set of random variables having a Markov property**
- **RBM:** restrict BMs to those without visible-visible and hidden-hidden connections.



RBM vs Neural Networks

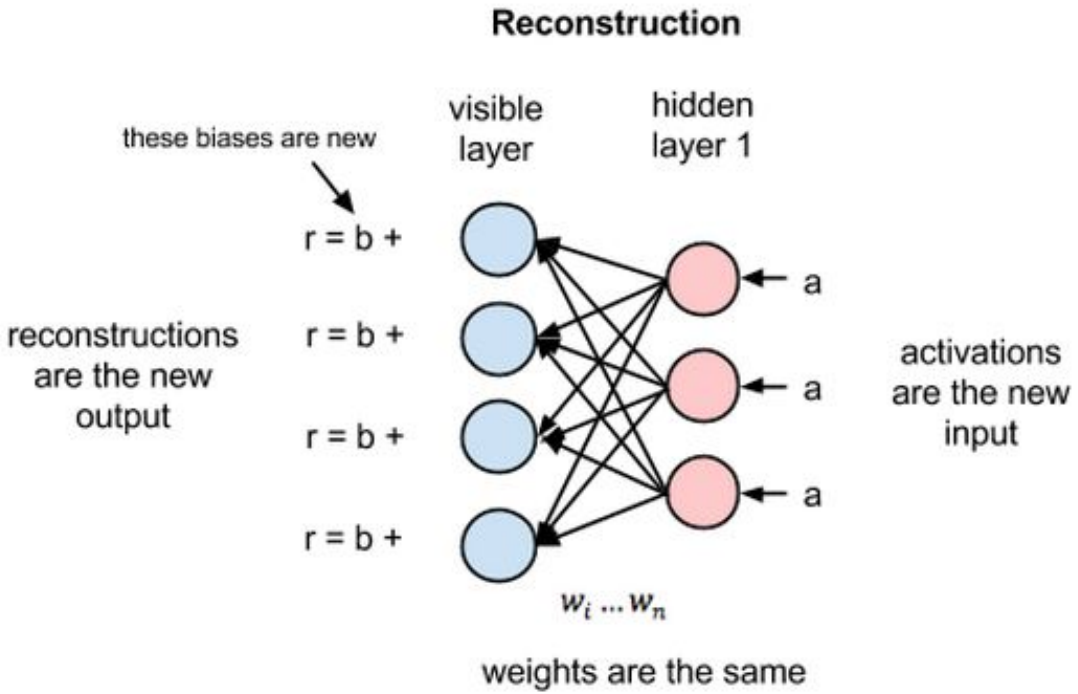
- RBMs are shallow, two-layer neural nets that constitute the building blocks of deep-belief networks.
- The 1st layer of the RBM is called the visible, or input, layer, and the second is the **hidden layer**.

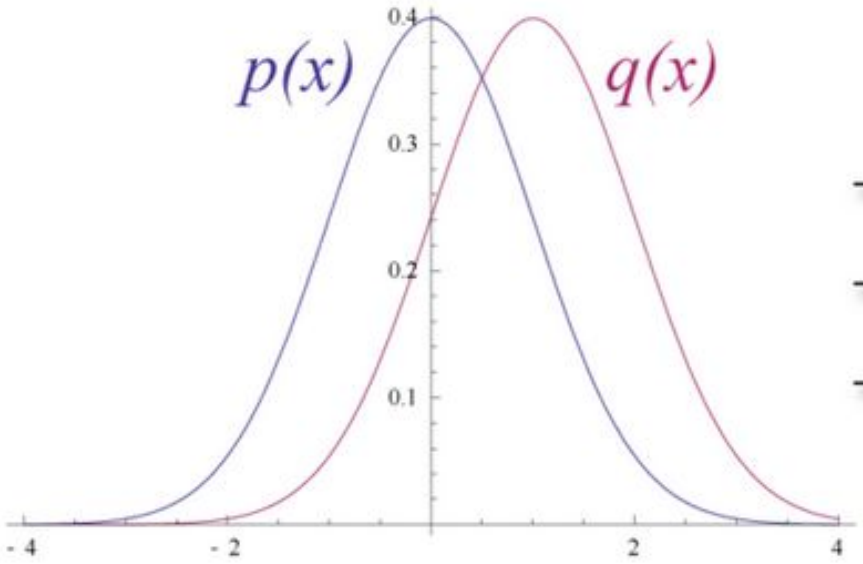


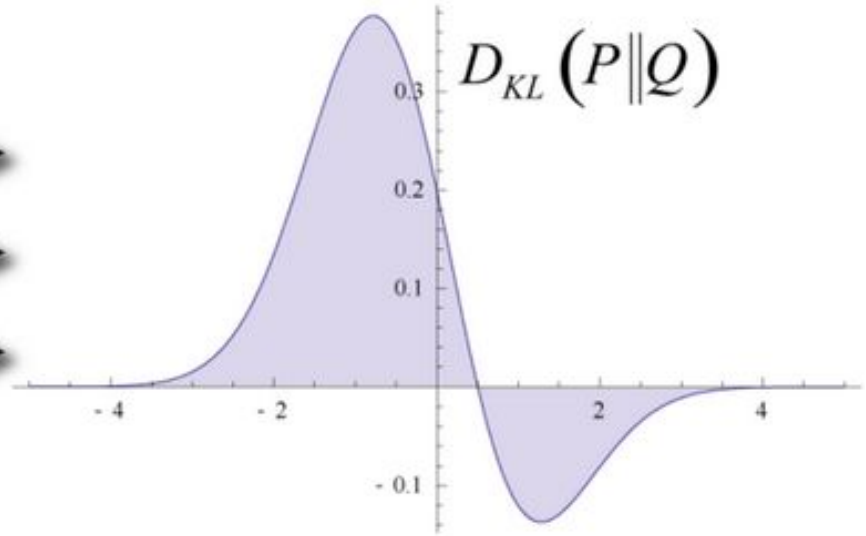
input

Reconstruction in RBM’s

- How do they perform better without involving a deeper network?
- Reconstruction estimates a continuous value based on many inputs and makes guesses about which discrete label to apply to a given input.




p(x)
q(x)
0.4
0.3
0.2
0.1
- 4
- 2
2
4


$D_{KL}\left(P\|Q\right)$
0.3
0.2
0.1
- 0.1
- 4
- 2
2
4

Steps and methods we will be using.

- Restricted Boltzmann Machines
- RBM’s with Binary Hidden Units

- Python
- numpy

Dataset Available to us

- Netflix dataset from 1998 to 2005
- 3 types of data
  - **Training data:** 100M ratings from 480k users randomly chosen on 18k movies.
  - **Validation data:** Containing 1.4M ratings.
  - **Test data(probe data):** Has 2.8M user/movie pairs without any rating.

https://web.archive.org/web/20090925184737/http://archive.ics.uci.edu/ml/datasets/Netflix+Prize

Alternative Methods

- ALS
- Matrix Factorization
- Gradient Descent
- Auto Encoders

Traditional approach

A low dimensional feature vector is assigned to each user and a low dimensional feature vector to each movie so that the rating each user assigns to each movie is modeled by the scalar-product of the 2 feature vectors. -

This means that the N × M matrix of ratings that N users assign to M movies is modeled by the matrix X which is the product of an N × C matrix U whose rows are the user feature vectors and a C × M matrix V whose columns are the movie feature vectors. The rank of X is C the number of features assigned to each user or movie.

Restricted Boltzmann Machines

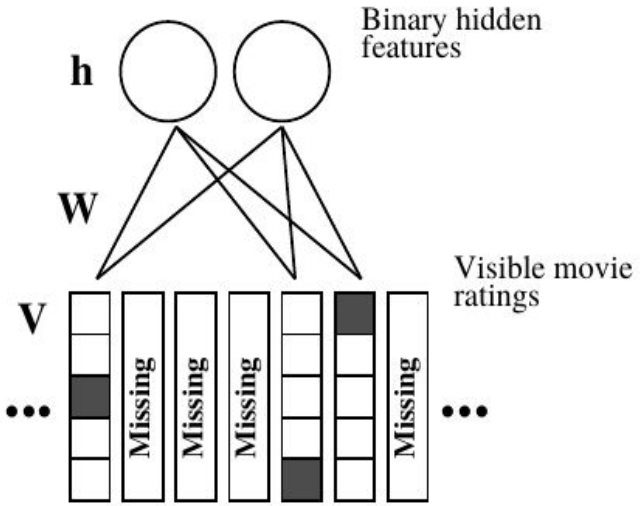
Why we use RBM? - Low-rank approximations based on minimizing the sum-squared distance can be found using SVD(singular value Decomposition). But in collaborative filtering domain, most of the data sets are sparse and this creates a complex non-convex problem. Thus we use RBM.

If we have M movies, N users and integral ratings from 1-K.

If all N users rated the same set of M movies then we just use a single training case for an RBM which has M softmax visible units symmetrically connected to a set of binary hidden units.

But in case of lot of ratings missing, we model a different RBM for each user.

Every RBM has the same number of hidden units, but an RBM only has visible softmax units for the movies rated by that user, so an RBM has few connections if that user rated few movies. Each RBM only has a single training case, but all of the corresponding weights and biases are tied together, so if two users have rated the same movie, their two RBM's must use the same weights between the softmax visible unit for that movie and the hidden units.



Tasks done up till 2nd phase -

- Implemented a simple rbm with binary hidden units.
- Modified the rbm to suit our model.
- Data Handling - Extracted the data from the netflix dataset and have made a dictionary for every user and the movies he has rated with their respective ratings.

Tasks completed for final submission

- Implemented a simple rbm with binary hidden units.
- Modified the rbm to suit our model.
- Data Handling - Extracted the data from the netflix dataset and have made a dictionary for every user and the movies he has rated with their respective ratings.
- Performing the training and using the weights learned for predicting.
- Predictor for predicting the movie rating given a movie id and a user id after training is complete.

Our Model

- The visible layer consists of x units where x is the number of movies a particular user has rated.
- Each unit in the visible layer is a vector of length 5 (since ratings are from 1 to 5), and the i_{th} index is one corresponding to the rating the user has given, the rest are zeros.
- The hidden layer consist of 100 units which are binary.
- The activation function we have used is the sigmoid function both for forward propagation and backward propagation.

Training our RBM

- 2 cycles - Forward propagation and Backward propagation.
- Weights initially have been assigned randomly.
- Forwards propagation - find the positive associations after finding the hidden units.
- Backward propagation - from the hidden units found in forward propagation we find the visible units and then again do forward propagation to find the negative associations.
- The difference between the positive and the negative associations gives us Δw the value with which we change the current weights for a user in the RBM.

Learning

Contrastive Divergence

To avoid computing <·>model, we follow an approximation to the gradient of a different objective function called CD.

Expectation <.>T represents a distribution of samples from running the Gibbs sampler.

$$\Delta W_{ij}^k = \epsilon(<v_i^k h_j>_{data} - <v_i^k h_j>_T)$$

Forward Propagation -

$$p(h_j = 1|\mathbf{V}) = \sigma(b_j + \sum_{i=1}^{m}\sum_{k=1}^{K} v_i^k W_{ij}^k)$$

Backward Propagation -

$$p(v_i^k = 1|\mathbf{h}) = \frac{\exp\left(b_i^k + \sum_{j=1}^{F} h_j W_{ij}^k\right)}{\sum_{l=1}^{K}\exp\left(b_i^l + \sum_{j=1}^{F} h_j W_{ij}^l\right)}$$

Sigmoid function -

$$\sigma(x) = 1/(1+e^{-x})$$

Making Predictions

$$p(v_q^k = 1|\mathbf{V}) \propto \sum_{h_1,\ldots,h_p} \exp(-E(v_q^k, \mathbf{V}, \mathbf{h}))$$

$$\propto \Gamma_q^k \prod_{j=1}^{F} \sum_{h_j \in \{0,1\}} \exp\left(\sum_{il} v_i^l h_j W_{ij}^l + v_q^k h_j W_{qj}^k + h_j b_j\right)$$

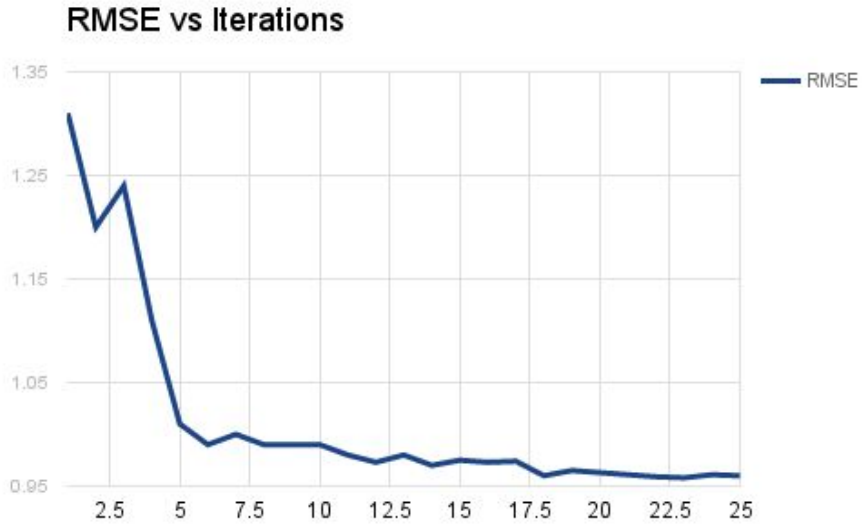
$$= \Gamma_q^k \prod_{j=1}^{F} \left(1 + \exp\left(\sum_{il} v_i^l W_{ij}^l + v_q^k W_{qj}^k + b_j\right)\right)$$

$$\Gamma_q^k = \exp\left(v_q^k b_q^k\right).$$

We compute the probabilities for each rating from 1-5.

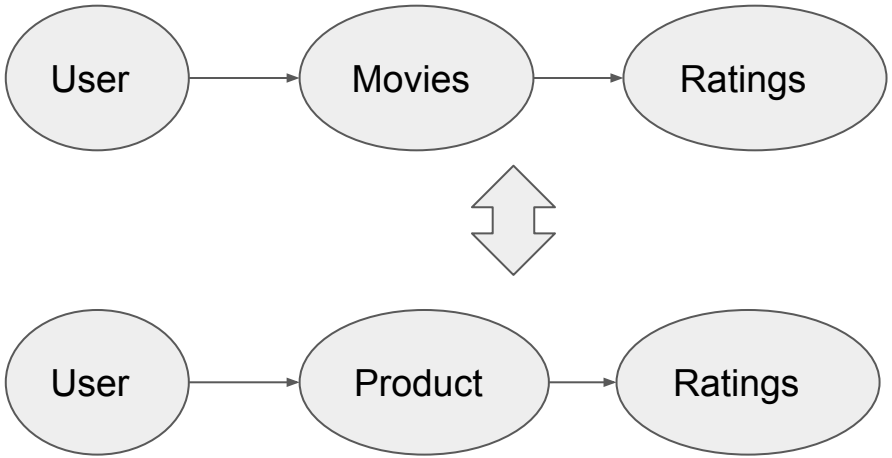
The maximum probability from the calculated values gives us the rating which a particular user would give that movie.

Results

- RMSE as obtained by our model is **~0.96** with only 1000 users and 25 iterations while the paper reports **~0.91** with about 2M users.
- 

Extending our current model

- Extend the rating prediction by formulating the problem as a recommendation problem.



GOAL: Predicting user ratings for the movies which he has not given rating yet.

GOAL: Predicting new ratings for products using user similarity.

Extending our current model

- One baseline method for rating prediction is based on users' preferences and products' characteristics.
- Another baseline method based on trust networks is the nearest-neighbor algorithm.
- The first model doesn't consider the trust network, while the latter one doesn't consider the bias from products and users.

Extending our model

We believe that both factors are useful for rating prediction, thus the rating prediction algorithm to estimate strengths of multi-faceted trust are shown below:

$$\hat{R}(u,i) = \alpha * (\mu + b(u) + c(i)) + (1-\alpha) * \frac{\sum_{v \epsilon N(u,i)} A(v,u) * R(v,i)}{\sum_{v \epsilon N(u,i)} A(v,u)}$$

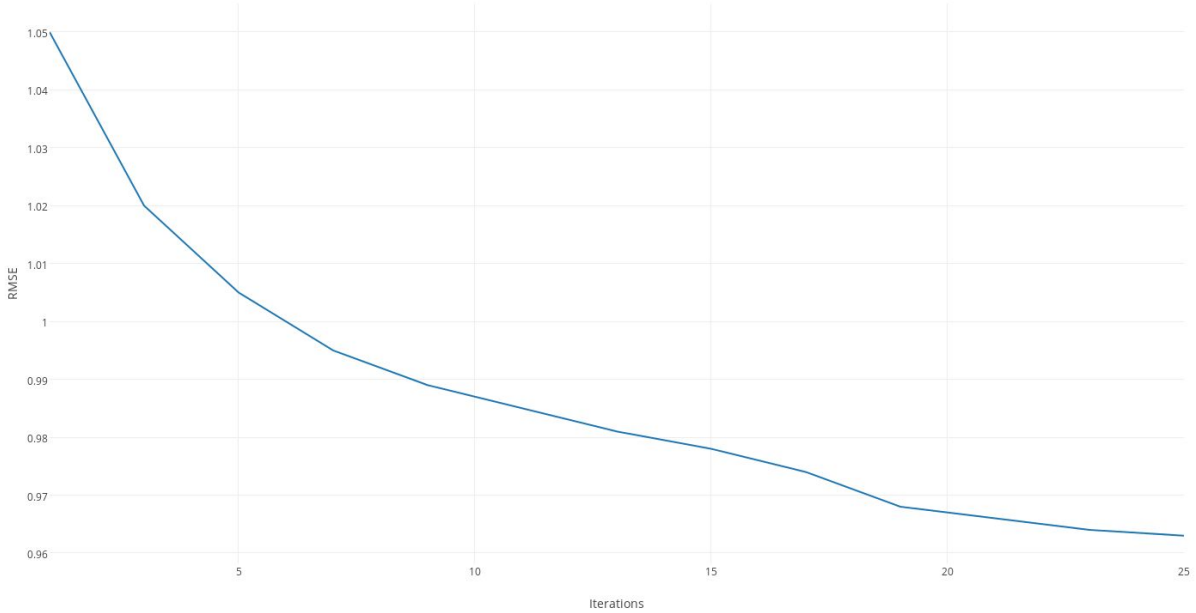
To estimate the parameters {A, b, c}, we solve the following optimization problem:

$$min_{A,b,c} \sum_{(u,i) \epsilon O} E^2(u,i) + \lambda * (\sum_u b^2(u) + \sum_i c^2(i))$$

$$E(u,i) = R(u,i) - \hat{R}(u,i)$$

Results after new model

RMSE obtained with 1000 users is ~0.96



THANK YOU :)