Recommendation System Using Restricted Boltzmann Machine For Collaborative Filtering

A Project report submitted in partial fulfillment of

the requirements for the degree of

Bachelor of Engineering

In

COMPUTER ENGINEERING

Submitted by

Fenil Patel (Roll No. 7401)
Jainam Savla (Roll No. 7412)
Paritosh Shirodkar (Roll No7416)

Under the guidance of

Prof. Brijmohan Daga
(Associate Professor, Computer Engineering Department)

Department of Computer Engineering
Fr. Conceicao Rodrigues College of Engineering
Bandra, Mumbai-400050
Year: 2017-2018

Internal Approval Sheet

CERTIFICATE

This is to certify that the project entitled **"Recommendation System using Restricted Boltzmann Machine for Collaborative Filtering"** is a bona fide work of **Fenil Patel(7401), Jainam Savla (7412), Paritosh Shirodkar(7416)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **"Bachelor of Engineering"** in **"Computer Engineering"**.

Prof.Brijmohan Daga

Supervisor/Guide

Dr. Sunil Surve

Head of Department

Dr. Srija Unnikrishnan

Principal

Approval Sheet

Project Report Approval

This project report entitled '**Recommendation System Using Restricted Boltzmann Machine For Collaborative Filtering**' by **Fenil Patel, Jainam Savla, Paritosh Shirodkar** is approved for the degree of **Bachelor of Engineering** in **Computer Engineering**.

Examiners:

1.----------

2.----------

Date:

Place:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Fenil Patel(Roll No. 7401)_____

Jainam Savla(Roll No.7412)_____

Paritosh Shirodkar(Roll No. 7416)_____

Date:

Abstract

In the last decade, with the boom in the internet industry, there has been a steady rise in E-commerce, for buying new products online. To achieve this, “Recommender systems” are used which apply statistical and knowledge discovery techniques to the problem of making product recommendations and thereby provide real time customer interaction.

In this project, we implement a recommender system while using Restrictive Boltzmann Machine For Collaborative Filtering on datasets.

Restricted Boltzmann machines (RBMs) are probabilistic graphical models that can be interpreted as stochastic neural networks. The increase in computational power and the development of faster learning algorithms have made them applicable to relevant machine learning problems. They attracted much attention recently after being proposed as building blocks of multi-layer learning systems called deep belief networks.

Most accurate recommender systems are black-box models, hiding the reasoning behind their recommendations. Yet explanations have been shown to increase the user’s trust in the system in addition to providing other benefits such as suitability, meaning the ability to verify the validity of recommendations. This gap between accuracy and transparency or explain ability has generated an interest in automated explanation generation methods. Restricted Boltzmann Machines (RBM) are accurate models for CF that also lack interpretability. We focus on RBM based collaborative filtering recommendations, and further assume the absence of any additional data source, such as item content or user attributes. We thus propose a new Explainable RBM technique that computes the top-n recommendation list from items that are explainable. Experimental results show that our method is effective in generating accurate and explainable recommendations. Most of the existing approaches to collaborative filtering cannot handle very large data sets. In this paper we show how a class of two-layer undirected graphical models, called Restricted Boltzmann Machines (RBM’s), can be used to model tabular data, such as user’s ratings of movie

Acknowledgements

We have great pleasure in presenting the report on **"Recommendation System using Restricted Boltzmann Machine for Collaborative Filtering "**. We take this opportunity to express my sincere thanks towards the guide **Prof. Brijmohan Daga**, C.R.C.E, Bandra (W), Mumbai, for providing the technical guidelines, and the suggestions regarding the line of this work. We enjoyed discussing the work progress with him during our visits to the department.

We thank Dr. Sunil Surve, Head of Computer Dept., Principal and the management of C.R.C.E., Mumbai for encouragement and providing the necessary infrastructure for pursuing the project.

We also thank all non-teaching staff for their valuable support, to complete our project.

Fenil Patel(Roll No. 7401)

Jainam Savla(Roll No.7412)

ParitoshShirodkar(Roll No.7416)

Date:

Contents

List of Figures

Introduction

Making choices is an integral part of everyday life, especially today when users are overwhelmed with information, from the Internet and television, to shelves in local stores and bookshops. We cope with this information overload by relying on daily recommendations from family, friends, authoritative users, or users who are simply willing to offer such recommendations. This is especially important when we lack information to make a rational decision, for example, choosing a hotel for vacations in a new city, selecting a new movie to watch, or choosing which new restaurant to visit.

Recommender systems facilitate decision-making processes through informed assistance and enhanced user experience. To aid in the decision-making process, recommender systems use the available data on the items themselves, such as item taxonomies, descriptions and other, and/or data on user experience with items of interest, for example, user choices, rankings, scores, tags, etc. Personalized recommender systems subsequently use this input data, and convert it to an output in the form of ordered lists or scores of items in which a user might be interested. These lists or scores are the final result the user will be presented with, and their goal is to assist the user in the decision-making process.

Recommender systems represent a broad and very active field of research and were, from their origins in the 1990s, somewhat detached from the data mining field. This was mostly due to the specific form of data that recommender systems used. There are three basic types of recommender systems: collaborative filtering, content based, and hybrid systems. Collaborative filtering recommender systems use social information, preferences, and experiences of other users in order to find users with similar taste. The assumption of these systems is that users of similar tastes might enjoy and consume similar items. Content filtering recommender systems use the available structured and unstructured information on users or items to recommend further items of interest. They assume that the user might be interested in items similar to the ones in which an interest has already been displayed. Both of these systems have their advantages, which are additionally reinforced, or disadvantages, which are diminished with the usage of hybrid systems-systems which combine both the collaborative and the content-based recommendation approach.

Recommender systems are ubiquitous, and an average Internet user has almost certainly had experiences with them, intentionally or not. For example, the well-known Internet commerce, Amazon.com employs a recommender system that recommends products its users might be interested in, based on the shopping habits of other users. Social networking sites like Facebook or LinkedIn use recommender systems for recommending new friends to users based on their social network structure. The music website Last.fm uses a recommender system to recommend new music to a user, based on the listening habits of users with similar music taste. The Internet Movie Database (IMDb) recommends similar movies, based on the content and style of the movies user previously browsed. Streaming provider Netflix tries to predict new movies a user might be interested in based on his watching habits and movie ratings, compared to other users. These, and numerous other examples like Stumble Upon, Google AdSense, YouTube, etc., which differ in services provided, like audio, video, general item, social network, other Internet content, books, etc., demonstrate the importance of these systems.

As illustrated by the previous application examples a unifying challenge for most recommender systems is to recommend items that will be useful for, appreciated or bought by a specific user or group of users. Depending on the specific applications, recommendations are inferred from information extracted from item descriptions, user ratings, shopping histories or user item views. Some approaches even examine social structures, by making use of the relationships among users to infer additional information, to further improve the systems performance, i.e. the match of recommendations with the users needs. Among these application areas a great deal of research has been conducted on the subject of movie recommendation, from the time on when Netflix announced of the Netflix Price challenge in 2007. Around 20,000 research groups joined the competition which awarded the winning team with a 1,000,000$ price in 2009. The task was to identify movies that a particular user will most likely enjoy based on feedback provided by the user beforehand. The amount of feedback is mostly very small compared to the total number of available movies to recommend, e.g. the Netflix Price dataset contained 18,000 different movie titles.

Collaborative Filtering(CF) approaches provide recommendations to users based on their collective recorded interests on items, typically relying on the similarity between users or items, giving rise to neighborhood-based CF approaches, which can be user-based or item-based. Neighborhood-based CF methods are white-box approaches that can be explained based on the ratings of similar users or items.

Most accurate recommender systems are model based methods that are black-boxes. Among model-based approaches are Restricted Boltzmann Machines (RBM) (Hinton, 2010) that can assign a low dimensional set of features to items in a latent space. The newly obtained set of features capture the user's interests and different items groups. RBM approaches have recently proved to be powerful for designing deep learning techniques to learn and predict patterns in large datasets because they can provide very accurate results (Hinton & Salakhutdinov, 2006).

Boltzmann machines (BMs) have been introduced as bidirectionally connected networks of stochastic processing units, which can be interpreted as neural network models. A BM can be used to learn important aspects of an unknown probability distribution based on samples from this distribution. In general, this learning process is di□cult and time-consuming. However, the learning problem can be simplified by imposing restrictions on the network topology, which leads us to restricted Boltzmann machines.

A (restricted) BM is a parameterized generative model representing a probability distribution. Given some observations, the training data, learning a BM means adjusting the BM parameters such that the probability distribution represented by the BM fits the training data as well as possible. Boltzmann machines consist of two types of units, so called visible and hidden neurons, which can be thought of as being arranged in two layers. The visible units constitute the first layer and correspond to the components of an observation (e.g., one visible unit for each pixel of a digital input image). The hidden units model dependencies between the components of observations (e.g., dependencies between pixels in images). They can be viewed as non-linear feature detectors.

In this work a recommendation framework based on a statistical model called restricted Boltzmann machine, is presented, which is able to cope well with the common problems of high dimensional and sparse data as well as in the so called cold start scenarios, where a new user or item joins the systems (Bogers & Van Den Bosch 2011). The proposed framework arguments rating based user feedback with labels assigned to items to tackle the problem of a great deal of missing values and thus improve the accuracy of recommendations for users that have only provided little feedback. The developed approach is applied to a movie recommendation task, where based on user provided ratings of movies, unseen movies that the user will also enjoy should be recommended.

This work is structured as follows. First, will give an overview over the different approaches of recommender systems in Section 2 and provide a detailed survey over methods of one family of approaches termed collaborative filtering. In Section 2 - Background the basic concepts, e.g. restricted Boltzmann machine, are introduced on which the later applied approaches base, to provide the reader with the knowledge required to follow the subject of this thesis. In Section 3 – Problem formulation the general challenges of recommender systems and the particular deficiencies of current approaches are outlined which lead to the problem this thesis approaches. Having the problem defined, Section 4 – Overview of the project that describes the flow of our project and intuition of restricted Boltzmann machine. The code detail of project in Section 5. In Section 6 – Implementation of our project is explained. The final section of this work is formed by a discussion of this work and the obtained results w.r.t. recommendation systems and the general body of knowledge within the field of data science.

Literature Review

2.1 Existing Systems

Previous Recommender System Structure:

The first step is to collect the preferences of the users. Our Collaborative Filtering (CF) implementation stores the data in two 2D matrices. So for each user in a row we have columns for each item that he or she has rated. The matrix is quite sparse, since a lot of users only buy one item. After getting the 2D dataset matrix, we implement two kinds of recommendation system models: item-based and user-based correlative filtering. For both models, we need to compute the similarity and prediction score. In the following part, we will explicitly show how we build our item-based and user-based recommendation system, and compare different models in the following subsection.

Item Similarity Computation:

Computing the similarity between items is the fundamental step of our recommendation system, since we want to recommend similar items to customers based on what they have bought before. The basic idea of similarity computation between two items i and j is to firstly isolate the users who have rated both of these items and then to apply a similarity computation technique to determine the similarity between i and j . Figure 2 shows the isolation of the co-rated items and similarity computation. We present three ways to compute similarity. In our recommendation system, we use the second method. The reason is stated below.

Cosine-based Similarity:

In this case, two items are thought of as two vectors in m dimensional user-space. The similarity between them is measured by computing cosine angle between two vectors. Similarity between items A and B, denoted by sim(i, j) is given by

$$sim(i,j) = \frac{\overrightarrow{i.j}}{\overrightarrow{|\imath|} * \overrightarrow{|\jmath|}}$$

Correlation-based Similarity:

In this case, the similarity between two items is measured by computing correlation corr i,j . Denoting the set of users who both rate i and j as U , the correlation similarity is given by

$$sim(i,j) = \frac{\sum_{u \epsilon U}(R_{u,i} - \acute{R}_i)(R_{u,j} - \grave{R}_j)}{\sqrt{\sum_{u \epsilon U}(R_{u,i} - \acute{R}_i)}\sqrt{\sum_{u \epsilon U}(R_{u,j} - \grave{R}_j)}}$$

Adjusted Cosine Similarity:

Computing similarity by using basic cosine measure in item based case has one obvious drawback, and it is the difference n rating scale between different users. We can subtract this kind of bias, so the similarity using this scheme is given by

$$sim(i,j) = \frac{\sum_{u \epsilon U}(R_{u,i} - \acute{R}_u)(R_{u,j} - \grave{R}_u)}{\sqrt{\sum_{u \epsilon U}(R_{u,i} - \acute{R}_u)}\sqrt{\sum_{u \epsilon U}(R_{u,j} - \grave{R}_u)}}$$

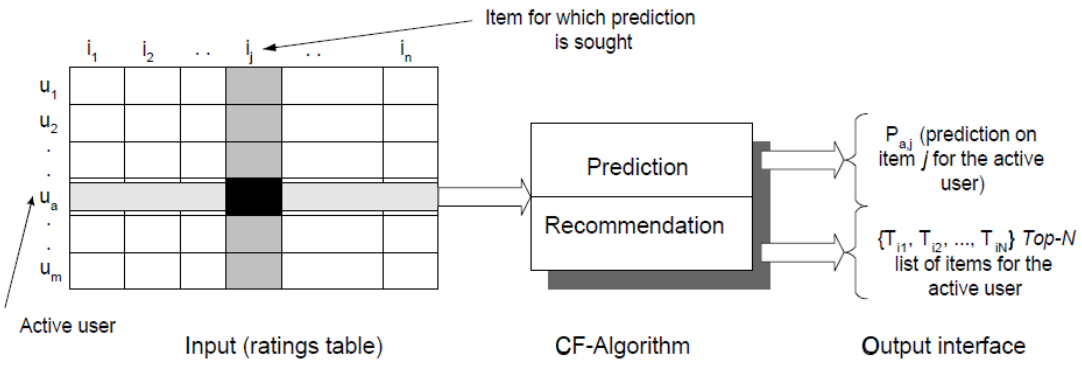

Figure 2.1.1 :The collaborative filtering process

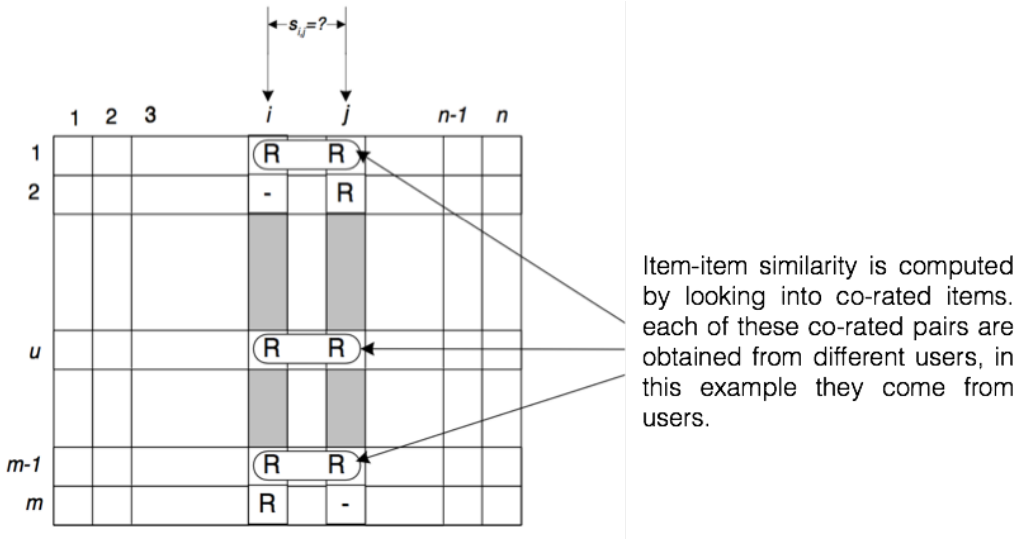

Figure 2.1.2:Isolation of the co-rated items and similarity computation

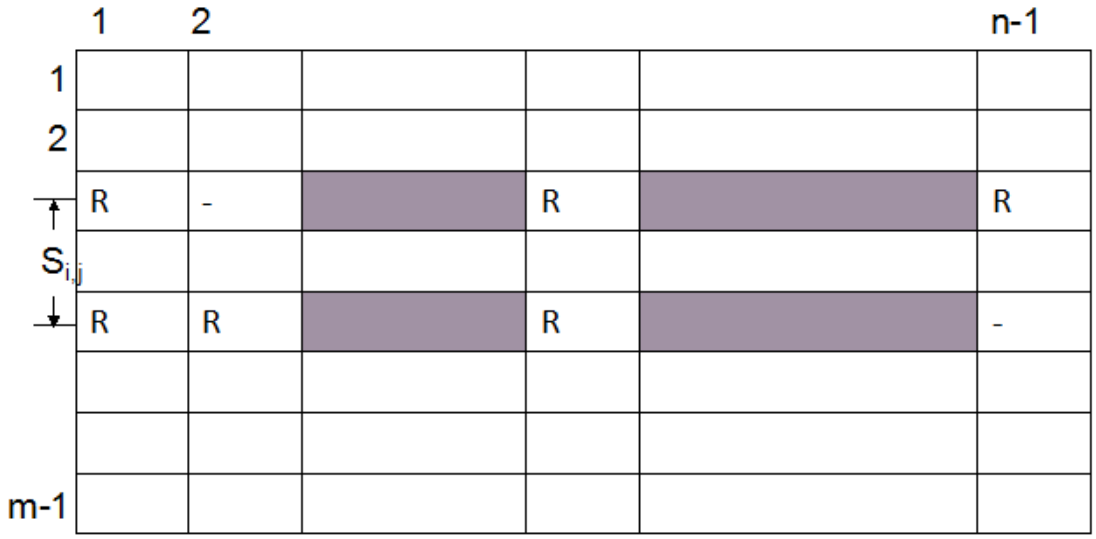

Figure2.1.3:Isolation of the co-rated users and similarity computation.

2.2 Prediction Computation:

After getting similarity between two different items, we then compute the prediction on an item i for a user u by computing the sum of the ratings given by the user on the items similar to i. Each rating is weighted by the corresponding similarity s i,j . And final weighted sum is divided by the sum of similarity to get normalized prediction value.

2.3Different Model Comparison:

The user-based CF has some limitations. One is its difficulty in measuring the similarities between users, and the other is the scalability issue. As the number of customers and products increases, the computation time of algorithms grows exponentially. The item-based CF was proposed to overcome the scalability problem as it calculates item similarities in an offline basis. It assumes that a user will be more likely to purchase items that are similar or related to the items that he or she has already purchased. Another one is the ratings, which are some discrete values, cannot provide us much information about relationship between users and items. The content-based filtering (CBF) method applies content analysis to target items.

2.4 Recommender System-Overview

Burke (2007), distinguishes recommendation techniques by (1) the kind of information the technique bases its recommendations on, (2) the required information to initiate a single recommendation process, i.e. the input needed to obtain an individual recommendation, and (3) the method deployed during the process of inferring a recommendation. Burke identifies 5 common recommendation techniques: Knowledge-based, Utility-based, Demographic, Content-Based and Collaborative. (Burke 2007)

Knowledge-based recommendation systems map a user's needs to an item's characteristics and decide based on e.g. cosine similarity if an item should be recommended. The user's needs are inferred from a user profile which can be a history record, or a specific query made by the user.

Utility-based recommendation systems make recommendations by maximizing a user specific utility function, which is evaluated on all available attributes. A critical part of these techniques is how the utility function is obtained. One way to do this is to have a highly selective/extensive query provided by the user which can be transformed into a utility function or to learn the function from given data, e.g. infer from a user profile.

The remaining three techniques have in common that their algorithms base on explicit ratings given by users on items and extract their recommendation based on a ranking of unassessed items. The technique assign a predicted rating for each unrated user/item pair to build s users specific item ranking.

Demographic recommendation systems make use of a user profile consisting of demographic information to compute similarities among users. A sum is formed over ratings of the k most similar users, factorized with the corresponding similarity value. The sum is evaluated for a user/item combination over similar users that rated this item and states a prediction for this rating.

Content-based recommender systems build a user profile out of items a user has rated considering their characteristic. The result is a description of a user by his appreciation for certain attributes. The predicted rating is inferred from the similarity of an unrated item to the user profile.

Collaborative recommendation systems also infer predicted ratings by building a representation that captures the dependencies among users and items based on the given user item ratings. Similarities are formed in this often-lower dimensional representation and based on these predicted ratings are extracted.

2.5 Collaborative Filtering

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	1	3	-	-	-
User 2	5	1	3	2	2
User 3	2	-	1	5	5
User 4	5	-	4	3	-

Figure 2.2: Example ratings provided by user on items

One commonly used and good/top performing method to make recommendations is called Collaborative Filtering (CF) (Bogers & Van Den Bosch 2011). In CF there are 3 main groups of approaches: memory-based, model-based and combinations of the former two.

Within memory-based approaches there are again three subcategories: user-based (U-CF), item-based (I-CF) and a combination of the both often referred as hybrid collaborative filtering (Su, 2009). Assuming we have data describing ratings of users on items, U-CF would create a neighborhood around each user consisting of other users that have similar taste, determined by ratings on common rated items using a distance metric (often cosine similarity). Ratings from this neighborhood, on an item the considered user has not rated, are summed while incorporating the similarity of neighbors obtained by the chosen distance metric. The resulting value estimates how the user would have rated the item. Considering the ratings described in Table 1 we want to know how user 4 would rate item 5. To do this we first identify similar users: in this simple example user 2 provided similar ratings on items 1 and 3; and second, we infer from the rating user 2 has given item 5 the predicted rating of user 4 on item 5, which will in this case be around 2. Computing this value for all missing item ratings enables us to choose items with highest estimated rating as recommendations. The drawback of U-CF models is that they are strongly dependent of a sufficient amount of provided ratings per user to produce useful recommendations, often referred as user-cold-start problem.

Similar to U-CF, but this time from the item perspective, in I-CF we define the neighborhood over items instead of users. The similarity of items is also measured by a distance metric, but this time evaluated on ratings the items received. Picking up the previous example: which rating would user 4 assess item 5. We first identify similar items: item 4 has received identical rating of users 2 and 3 as item 5; and second infer from the rating user 4 has given item 4 the predicted rating of item 5, which would be 3.

In general, in this view items are recommended based on the result of the weighted sum of ratings an items neighbors received. These estimated ratings can be again used to rank not rated items of users and present the highest rated as recommendation to a user. The benefit of I-CF is, that it can cope with the user-cold-start better than U-CF and offers performance improvements, for application where the number of items is smaller than the number of users, since the similarity metric needs to be evaluated only on the smaller number of items. (Su & Khoshgoftaar 2009)

The mentioned advantages of I-CF and the reason it's often implemented is, it's capability to cope with the user-cold-start problem. The reason for that is, that it, like shown in the example, considers only ratings given by other users and not the provided ratings of that user. The price for that is that I-CF suffers when the number of ratings on an item is very low, e.g. a new item has joined the data base and as not been rated jet, which is called item-cold-start problem.

Combined CF models try to merge both of the previous methods to cancel out each other's deficits and is called hybrid or global approach. A naive method would be switch between the two models depending on the number of given ratings (user and item wise) for the current considered user/item combination. Beside the mentioned cold-start problems memory-based models also struggle with the omnipresent sparsity of the data in this kind of applications.

Model-based CF approaches try to cope with the cold-start problems by building a model which can represent latent dependencies among items and users. They have access to global information, generated during the training phase, to compensate missing data in the cold-start case and are able to handle the sparse data problem in general better. Several model-based approaches have been successfully applied to the recommendation problem with better scores than the memory-based models. However, the advantage in precision come at the cost of complex model building compared to the straight forward comparisons done by most memory-based models. Scalability of the models is also an important factor which, when working with data sets that exceed an item count of 1 million.

2.6 Restricted Boltzmann Machine

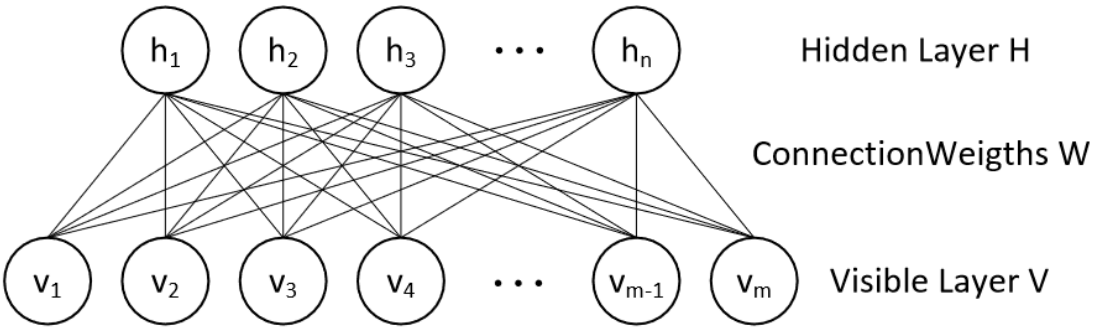

Figure 2.3: Graph representation of a Restricted Boltzmann machine illustrating the two bipartite sets of vertices, i.e. hidden layer H and visible layer V, and their weighted connections W.

Invented by Geoff Hinton, a Restricted Boltzmann machine is an algorithm useful for dimensionality reduction, classification, regression, collaborative filtering, feature learning and topic modeling.

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

Each circle in the graph above represents a neuron-like unit called a *node*, and nodes are simply where calculations take place. The nodes are connected to each other across layers, but no two nodes of the same layer are linked.
That is, there is no intra-layer communication – this is the *restriction* in a restricted Boltzmann machine. Each node is a locus of computation that processes input, and begins by making stochastic decisions about whether to transmit that input or not. (*Stochastic* means "randomly determined", and in this case, the coefficients that modify inputs are randomly initialized.)
A Restricted Boltzmann machine is a Markov Random Field (MRF) and can be described as an undirected bipartite graph $G = (V,E)$, where V are the vertices of the graph consisting of m visible units $\boldsymbol{V} = (V1,\ldots,Vm)$ and n hidden units $\boldsymbol{H} = (H1,\ldots,Hn)$, these sets of units are often referred as layers, and edges E defining connections among the vertices. A MRF describes a set of random variables $\boldsymbol{X} = (X1,\ldots,Xm{+}n)$ that fulfill the property, which is called the local Markov property, that a random variable Xu is only dependent on its neighbors or from the other point of view: it is independent of all other variables given its neighbors. Referring back to our graph G two units u and w taken from V are neighbors if there is a connecting edge $\{u, w\} \in E$, in general the neighborhood of unit u is defined by $Nu = \{w \in V: \{u, w\} \in E\}$.

Additionally the joined probability distribution of $\boldsymbol{X}$, i.e. the probability that variables in $\boldsymbol{X}$

have the values $\boldsymbol{x} = (x1,\ldots,xm+n)$, of a MRF is given by the Gibbs distribution

$$P(\boldsymbol{X} = \boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} e^{-E(\boldsymbol{x})}, \tag{1}$$

where the function $E(\boldsymbol{x})$ is referred as the energy function, inspired from physical systems, stating the energy of the system for a configuration $\boldsymbol{x}$.

The function Z marginalizes over all the variables and is called partition function. It is defined as

$$Z(\boldsymbol{x}) = \sum_{\boldsymbol{x}} e^{-E(\boldsymbol{x})}. \tag{2}$$

With the RBMs bipartite characteristics and the local Markov property a unit's state is only dependent on states of the other layer's units, i.e. the state of hidden unit $\square j \in \boldsymbol{H}$ depends solely on states of the visible units $\boldsymbol{v} = (v1,\ldots,vm) \in \boldsymbol{V}$ and the state of visible unit $vi \in \boldsymbol{V}$ is dependent on states of the hidden units $\boldsymbol{h} = (\square 1,\ldots,\square n) \in \boldsymbol{H}$.

Thus the probability of a layer being in a certain state is only dependent on the other layers state, i.e. the probability that the units in H take values $\boldsymbol{h}$ is dependent on the values $v1,\ldots,vm$ of visible units in V and vice versa.

Therefore the joined conditional probability of the hidden layer factorizes and can be expressed as

$$p(\boldsymbol{h}|\boldsymbol{v}) = \prod_{j=1}^{m} p(h_j|\boldsymbol{v}) \tag{3}$$

and for the same reasons the joined conditional distribution of the states of the visible units is given by

$$p(\boldsymbol{v}|\boldsymbol{h}) = \prod_{i=1}^{n} p(v_i|\boldsymbol{h}). \tag{4}$$

The conditional distribution of a single unit being in a specific state is for a binary unit given by

$$p(h_j|\boldsymbol{v}) = sigmoid\left(\sum_{i=1}^{n} v_i w_{ij} + c_j\right) \tag{5}$$

and

$$p(v_i|\boldsymbol{h}) = sigmoid\left(\sum_{j=1}^{m} h_j w_{ij} + b_i\right), \tag{6}$$

where wij is the value of the weight matrix W associating a weight wij to each edge, i.e. a factor wij is assigned to the edge connecting visible unit vi and hidden unit $\square j$. For each unit a bias term is introduced, bi for visible unit $vi \in \boldsymbol{V}$ and cj for hidden unit $\square j \in \boldsymbol{H}$, to provide flexibility to the model. The deployed sigmoid function is defined as

$$sigmoid(x) = \frac{1}{1 + e^{-x}}. \tag{7}$$

With the joined and conditional distributions a RBM's marginal distribution for a visible layer's state is given as

$$P(\boldsymbol{v}) = \frac{1}{Z(v)} \sum_{j=1}^{n} e^{-E(\boldsymbol{v}, h_j)} \tag{8}$$

Likewise thermal energy of physical systems is determined by evaluating the current state of the system, the energy E of a RBM is defined with respect to the state of its units. For a given state of visible units $\boldsymbol{v}$ and hidden units $\boldsymbol{h}$, E is defined by

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij} v_i h_j - \sum_{i=1}^{m} b_i v_i - \sum_{j=1}^{n} c_j h_j. \tag{9}$$

A RBM can be considered as a thermal system, where the state of a unit is interpreted as it's temperature. A unit's temperature is, analogically to physical systems, dependent on the temperature of surrounding unit's, i.e. a low temperature unit connected to mostly high temperature ones will , with probability determined by Eq. (1), also change its state to a higher temperature. Like other thermal systems tend to reach a state of equilibrium, the energy of an RBM can be manipulated via the weight matrix W and biases b,c.

2.7 Training of Restricted Boltzmann Machine

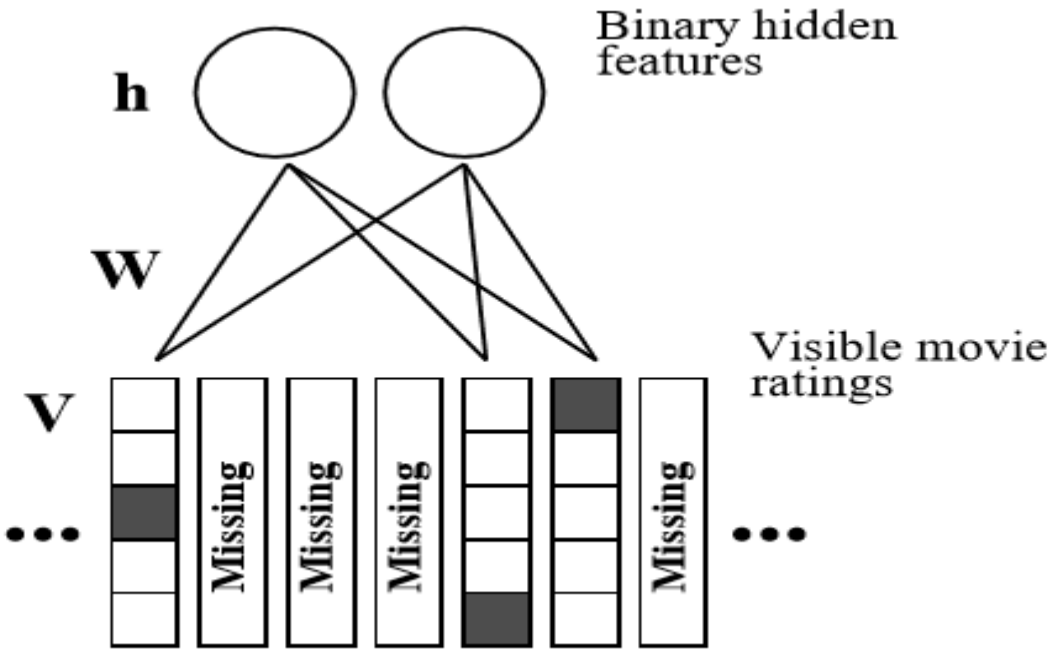

Figure 2: A restricted Boltzmann machine with binary hidden units and SoftMax visible units.

In order to learn a representation of some data $\mathcal{D}$ with any model, the likelihood L is introduced, which measures the similarity of a unknown distribution q, which $\mathcal{D}$ is drawn from, and the distribution p of the model. The likelihood of a RBM with parameters θ, e.g. wij, $bi$and cj, given data $\mathcal{D} = (\boldsymbol{x1},\ldots,\boldsymbol{xn})$ is defined as

$$\mathcal{L}(\theta|\mathcal{D}) = \prod_{i=1}^{m} p(\boldsymbol{x_i}|\theta). \tag{10}$$

Maximizing the likelihood of a RBM is performed numerical since it is not possible to find a analytical solution for the underlying Gibbs distribution (Fischer & Igel 2014). Usually the loglikelihood, i.e. log L, is chosen to optimize due to the fact, that it simplifies the computation of the derivatives used in the latter optimization process, by transforming the product in (10) into a sum of logs in (11). The log-likelihood becomes

$$\log \mathcal{L}(\theta|\mathcal{D}) = \sum_{i=1}^{m} \log p(\boldsymbol{x_i}|\theta) \tag{11}$$

and the log-likelihood of a RBM for a data sample $\boldsymbol{v} = \boldsymbol{xi}$ is given by

$$\begin{aligned}\log \mathcal{L}(\theta|\boldsymbol{v}) = \log p(\boldsymbol{v}|\theta) &= \log \frac{1}{Z(\boldsymbol{v})} \sum_{j=1}^{n} e^{-E(\boldsymbol{v},\boldsymbol{h}_j)} \\ &= \log \sum_{j=1}^{n} e^{-E(\boldsymbol{v},\boldsymbol{h}_j)} - \log \sum_{i=1}^{m} \sum_{j=1}^{n} e^{-E(\boldsymbol{e}_i,\boldsymbol{h}_j)}\end{aligned} \tag{12}$$

Gradient ascent is a numerical method to find an optimal set of parameters numerically by maximizing the averaged log-likelihood of the model over data $\mathcal{D}$. This is achieved by updating the parameters of the model θ according to the gradient of the log-likelihood. The typical update rule used to train RBMs is defined by

$$\theta^{(t+1)} = \theta^{(t)} + \underbrace{\eta \frac{\partial}{\partial \theta^{(t)}} \left(\log \mathcal{L}\left(\theta^{(t)}|\mathcal{D}\right)\right) - \lambda \theta^{(t)} + \alpha \Delta \theta^{(t-1)}}_{\Delta \theta^{(t)}} \tag{13}$$

and consists of the summation of four essential parts: 1. The initial state of the set of parameters $\theta(t)$. 2. The gradient of the log-likelihood, w.r.t. parameters $\theta(t)$, weighted by the learning rate η. 3. The initial set of parameters $\theta(t)$ multiplied by the weight cost parameter λ which penalizes large parameter values. 4. The change of the parameters $\theta(t-1)$, defined by 2, 3 and 4, of the previous step weighted by the momentum α, this accelerates the training process by increasing the parameter update w.r.t. the previous update $\Delta\theta(t-1)$.

Considering the log □ given in (12) the gradient for sample $\boldsymbol{v}$ becomes

$$\frac{\partial \log \mathcal{L}(\theta|\boldsymbol{v})}{\partial \theta} = -\sum_{j=1}^{n} p\left(h_j|\boldsymbol{v}\right) \frac{\partial E\left(\boldsymbol{v}, h_j\right)}{\partial \theta} + \sum_{i=1}^{m} \sum_{j=1}^{n} p\left(v_i|h_j\right) \frac{\partial E\left(v_i, h_j\right)}{\partial \theta}. \tag{14}$$

From (14) we can infer the derivatives w.r.t. the different model parameters. Considering a weight parameter wij the derivative of the log-likelihood is given as

$$\begin{aligned}\frac{\partial \log \mathcal{L}(\theta|\boldsymbol{v})}{\partial w_{ij}} &= -\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) \frac{\partial E\left(\boldsymbol{v}, h_j\right)}{\partial w_{ij}} + \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} p(\boldsymbol{v}|\boldsymbol{h}) \frac{\partial E\left(v_i, h_j\right)}{\partial w_{ij}} \\ &= \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v})\, v_i h_j - \sum_{\boldsymbol{v}} p(\boldsymbol{v}) \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v})\, v_i h_j.\end{aligned} \tag{15}$$

Similar the derivative w.r.t. the bias bi becomes

$$\frac{\partial \log \mathcal{L}(\theta|\boldsymbol{v})}{\partial b_i} = -\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) \frac{\partial E(\boldsymbol{v}, h_j)}{\partial b_i} + \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} p(\boldsymbol{v}|\boldsymbol{h}) \frac{\partial E(v_i, h_j)}{\partial b_i} \\ = v_i - \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} p(\boldsymbol{v}|\boldsymbol{h}) v_i \tag{16}$$

and w.r.t. the bias parameter cj

$$\frac{\partial \log \mathcal{L}(\theta|\boldsymbol{v})}{\partial c_j} = -\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) \frac{\partial E(\boldsymbol{v}, h_j)}{\partial c_j} + \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} p(\boldsymbol{v}|\boldsymbol{h}) \frac{\partial E(v_i, h_j)}{\partial c_j} \\ = \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) h_j - \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} p(\boldsymbol{v}|\boldsymbol{h}) h_j \tag{17}$$

Unfortunately the second term in (14) and thus the second terms in (15), (16) and (17) are computationally intractable since the sum spans over all of the models variables. Therefore, the derivative of the log L is approximated using contrastive divergence (CD).

2.8 Contrastive Divergence

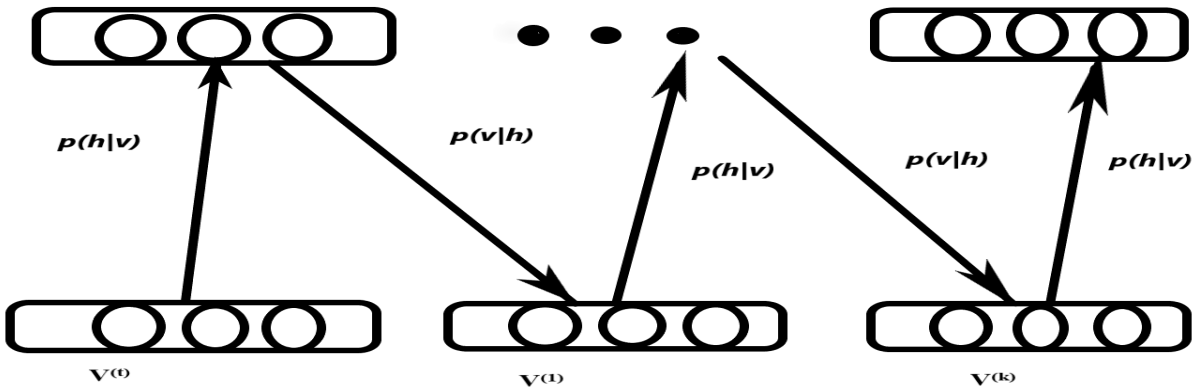

Figure 3: Contrastive Divergence

Contrastive Divergence (CD) was introduced to RBM training by Hinton (2002) and is a commonly deployed method to overcome the computational complexity of the computation of the log derivative by using Gibbs sampling to generate an approximation. Gibbs sampling is an iterative process, where in each step the joined probability distributions of the visible and hidden layers are used to sample a state $\boldsymbol{h}+$ of the hidden layer H and from this sample a sample $\boldsymbol{v}-$ of the visible layer V is computed. This process is repeated in a loop or so called Markov chain, starting with an initial seed $\boldsymbol{v}+$ picked randomly from the training data $\mathcal{D}$.

Fortunately, a single Gibbs step is sufficient to produce an approximation of the gradient which is good enough to let the gradient ascent method learn a good representation.
The single steps of CD can be summarized as

1. Sample $\boldsymbol{h}+$ using $p(\boldsymbol{h}|\boldsymbol{v}+)$, where $\boldsymbol{v}+$ is a training sample taken from $\mathcal{D}$ and
2. Sample $\boldsymbol{v}-$ using $p(\boldsymbol{v}|\boldsymbol{h}-)$.

With these samples we can approximate the required gradient-based parameter update for a weight wij:

$$\frac{\partial \log \mathcal{L}\left(\theta^{(t)}|\mathcal{D}\right)}{\partial w_{ij}} \sim D^{+} - D^{-}, \tag{18}$$

where $D+ = \boldsymbol{v}+ \cdot p(i = 1|\boldsymbol{v}+)$ and $D- = \boldsymbol{v}- \cdot p(i = 1|\boldsymbol{v}-)$.

Analogously the update of the bias parameters is approximated with

$$\frac{\partial\, log\, \mathcal{L}\left(\theta^{(t)}|\mathcal{D}\right)}{\partial b_i} \sim \boldsymbol{v}^{+} - \boldsymbol{v}^{-} \tag{19}$$

and

$$\frac{\partial \log \mathcal{L}\left(\theta^{(t)}|\mathcal{D}\right)}{\partial c_j} \sim p(h_i = 1|\boldsymbol{v}^+) \; - p(h_i = 1|\boldsymbol{v}^-) \tag{20}$$

With these approximation, the update rule given in (13) becomes

$$\begin{aligned}
w_{ij}{}^{(t+1)} &= w_{ij}{}^{(t)} + \underbrace{\eta(D^+ - D^-) - \lambda w_{ij}{}^{(t)} + \alpha \Delta w_{ij}{}^{(t-1)}}_{\Delta w_{ij}{}^{(t)}} \\
b_i{}^{(t+1)} &= b_i{}^{(t)} + \underbrace{\eta(v^+ - v^-) - \lambda b_i{}^{(t)} + \alpha \Delta b_i{}^{(t-1)}}_{\Delta b_i{}^{(t)}} \\
c_j{}^{(t+1)} &= c_j{}^{(t)} + \underbrace{\eta\left(p(h_i = 1|\boldsymbol{v}^+) \; - p(h_i = 1|\boldsymbol{v}^-)\right) - \lambda c_j{}^{(t)} + \alpha \Delta c_j{}^{(t-1)}}_{\Delta c_j{}^{(t)}}
\end{aligned} \tag{21}$$

Problem Statement

3.1 Challenges

Recently developed recommender systems depend greatly on advances made in other data science disciplines like data-mining or information retrieval and borrows approaches from the fields of machine learning and artificial intelligence (Bogers & Van Den Bosch 2011). The domain of recommendation systems is of particular research interest due to some of its demanding challenges: high-dimensionality and sparsity of the data as well as the related cold start problem, which is especially challenging for these systems, since the users satisfaction is greatly dependent on the first interactions with the system and at the same time it is a very important factor for a recommender systems' success and acceptance.

The application field of a recommender systems can span over all sorts of items from music over movies to documents and more without the need to make adaptions to the system. As described in Section 2.1, it is the available data that limits the system's ability to generalization over multiple application areas and not the type of content that is recommended. Thus, the problem can be approached considering a specific application scenario while the method itself will maintain its generalization capabilities. The representative application chosen is the recommendation of movies, as it is one of the major disciplines where data is freely available and easy to access.

The typical scenario for movie recommenders is where users have provided ratings on movies expressing their favor. To illustrate the problems of high-dimensional data and sparsity in the context of movie recommendations, consider a small dataset1, consisting of 1000 users and 1700 movies, the dimensionality of the problem space is defined by the number users and the number of movies, which results in 1.7 million possible unique user/movie combinations. Comparing this number with the number of ratings provided by the users, roughly 100,000, depicts the sparsity of the data, since just ~6% of the possible ratings are present. This makes it challenging for a system to identify movies out of the 94% remaining possible matchings.

3.2 Solution to Above Challenges

To overcome high-dimensionality and sparsity of the data, we have used *Restricted Boltzmann machine(RBM)* which is useful for dimensionality reduction, classification, regression, collaborative filtering, feature learning and topic modeling.
RBMs are probabilistic graphical models that can be interpreted as stochastic neural networks.

The problem this work tries to solve is to develop a framework that is able to give good recommendations by overcoming the sparsity problem while being able to cope with the arising high dimensional data.

Project Description

4.1 Overview of project

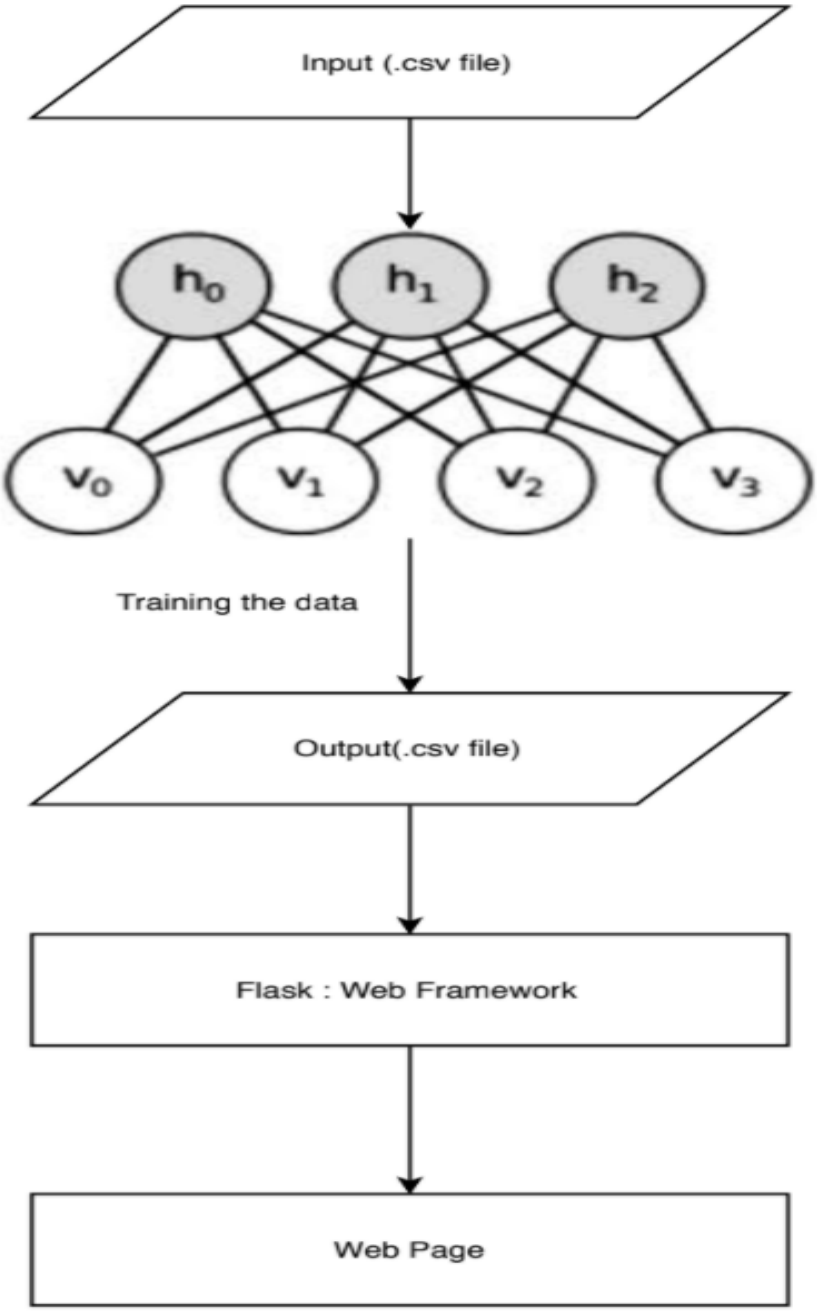

Figure 4.1: Basic Flowchart of our System

- The dataset is split into training and test set.
- The data is trained using Restricted Boltzmann Machine model .
- As model is explained below , after the training the output is generated and given as input to flask which is a web framework.
- Final step is the UI page.

4.2 Model

1. The visible layer consists of x units where x is the number of movies a particular user has rated.
2. Each unit in the visible layer is a vector of length 5 (since ratings are from 1 to 5), and the ith index is one corresponding to the rating the user has given, the rest are zeros. The hidden layer consists of 100 units which are binary.
3. The activation function we have used is the sigmoid function both for forward propagation and backward propagation.
4. After that we have used contrastive divergence algorithm to train our model to generate an output file.
5. We tested our approach on the Movie Lens 1 ratings data which consists of 100, 000 ratings, on a scale of 1 to 5, for 1700 movies and 1000 users. The data is split into training and test sets such that 10% of the latest ratings from each user are selected for the test set and the remaining are used in the training set. Ratings are normalized between 0 and 1, to be used as RBM input. We compare our results with RBM, user-based top-n and non-personalized most popular items. Each experiment is run 10 times and the average results are reported.
6. To assess the rating prediction accuracy, we used the Root Mean Squared Error (RMSE) metric:

   RMSE = $\sqrt{\sum_{(u,i)\in testset}(r_{ui} - \acute{r_{ui}})^2/|testset|}$.

   Note that RMSE can only be calculated for the prediction-based methods and not the top-n recommendation technique.

4.3 Example from our project

Training Dataset used:

Movie Name	Awards	Year	Category	Duration	Genre	Given
Sense	Bafta and Academy Award	1995	Hollywood	2 hr 20 mins	Action	Liked
Golden Eye	Bafta Award	1995	Hollywood	2hr 10 mins	Action	Liked

Test set used:

Movie Name	Awards	Year	Category	Duration	Genre	Given	Expected
Casino	Academy Award	1995	Hollywood	2 hr 30 mins	Action	Liked	Liked

Training the model:
Once the training dataset is presented to the model, the model identifies common parameters among the movies based on which the prediction will be made. This is one of the major advantages using Restricted Boltzmann Machines that feature/parameter identification occurs automatically.

Once the model is trained it will try and look for the same features that it identified during the training in the movies which are presented in the test set.
In this case the features will be Awards, Year, Category, Duration and Genre of the movie.

Eg. When the movie Casino is presented to the model it checks the values for all these features.

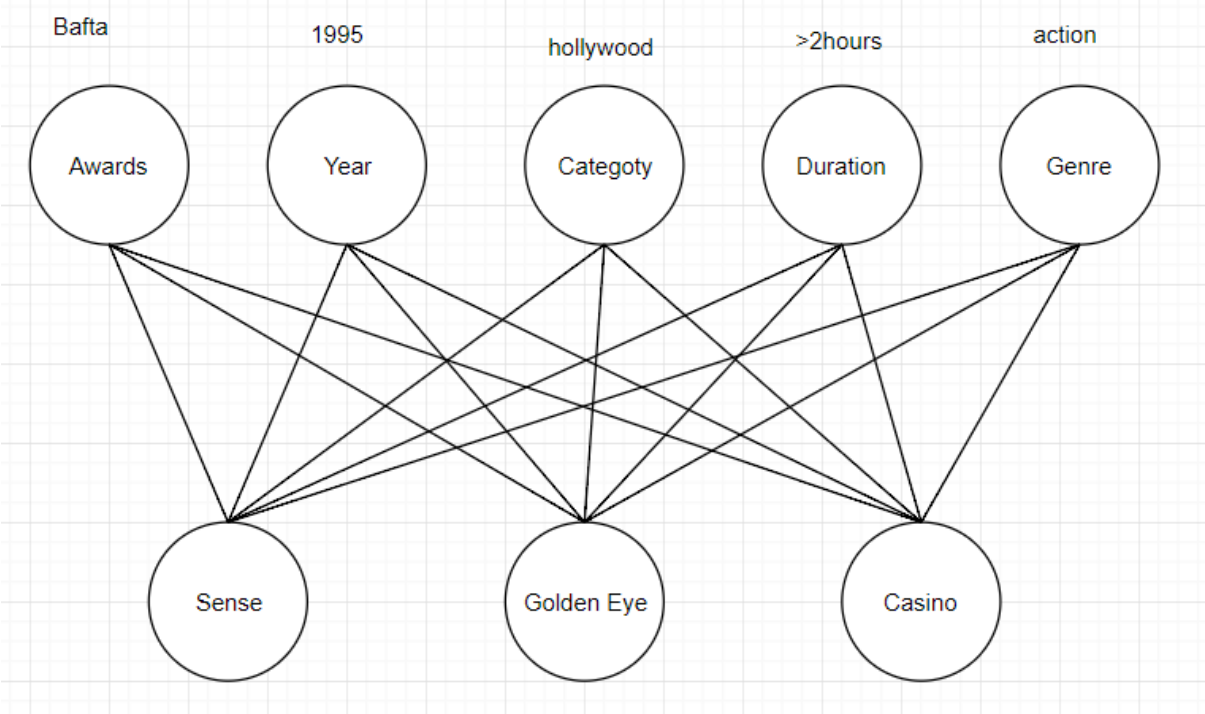

Figure 4.3: Example of working of Restricted Boltzmann Machine

Now since the movie has the following features:

Common Features - Year = 1995, Category = Hollywood, Duration >2 hours and Genre = Action

Uncommon Feature - Awards = Bafta Awards

Now since the RBM Model finds four out of five features to be present in the movie it predicts that the user will like the movie (which is the same as the expected output).

Code Details

5.1 Algorithm of Contrastive Divergence:

Algorithm 1. k-step contrastive divergence

Input: RBM $(V_1, \ldots, V_m, H_1, \ldots, H_n)$, training batch S
Output: gradient approximation Δw_{ij}, Δb_j and Δc_i for $i = 1, \ldots, n$, $j = 1, \ldots, m$

init $\Delta w_{ij} = \Delta b_j = \Delta c_i = 0$ for $i = 1, \ldots, n$, $j = 1, \ldots, m$
forall the $v \in S$ **do**
  $v^{(0)} \leftarrow v$
  for $t = 0, \ldots, k-1$ **do**
    for $i = 1, \ldots, n$ **do** sample $h_i^{(t)} \sim p(h_i \mid v^{(t)})$
    for $j = 1, \ldots, m$ **do** sample $v_j^{(t+1)} \sim p(v_j \mid h^{(t)})$
  for $i = 1, \ldots, n$, $j = 1, \ldots, m$ **do**
    $\Delta w_{ij} \leftarrow \Delta w_{ij} + p(H_i = 1 \mid v^{(0)}) \cdot v_j^{(0)} - p(H_i = 1 \mid v^{(k)}) \cdot v_j^{(k)}$
    $\Delta b_j \leftarrow \Delta b_j + v_j^{(0)} - v_j^{(k)}$
    $\Delta c_i \leftarrow \Delta c_i + p(H_i = 1 \mid v^{(0)}) - p(H_i = 1 \mid v^{(k)})$

5.2 Code of Contrastive Divergence:

```
class RBM():
    def __init__(self, nv, nh):
        self.W = torch.randn(nh, nv)
        self.a = torch.randn(1, nh)
        self.b = torch.randn(1, nv)
    def sample_h(self, x):
        wx = torch.mm(x, self.W.t())
        activation = wx + self.a.expand_as(wx)
        p_h_given_v = torch.sigmoid(activation)
        return p_h_given_v, torch.bernoulli(p_h_given_v)
    def sample_v(self, y):
        wy = torch.mm(y, self.W)
        activation = wy + self.b.expand_as(wy)
        p_v_given_h = torch.sigmoid(activation)
        return p_v_given_h, torch.bernoulli(p_v_given_h)
    def train(self, v0, vk, ph0, phk):
        self.W += torch.mm(v0.t(), ph0) - torch.mm(vk.t(), phk)
        self.b += torch.sum((v0 - vk), 0)
        self.a += torch.sum((ph0 - phk), 0)
    def predict(self, x): # x: visible nodes
        _, h = self.sample_h( x)
        _, v = self.sample_v( h)
        return v
```

5.3 Flask Code:

```
import csv
import random
from flask import Flask, render_template, request, redirect, url_for

number_of_users = 100
number_of_movies = 100

app = Flask(__name__)

def random_colour_generation():
        colour = ['aqua', 'green', 'yellow', 'red']
        secure_random = random.SystemRandom()
        return secure_random.choice(colour)

def generate_recommendation():
   with open("output_new.csv") as csvfile:
      reader = csv.reader(csvfile)

      # a list to store all the movie names
      column_header = next(reader)

      # a matrix to store the binary data for users and the corresponding movies
      binary_data = [row for row in reader]

      # recommendation list
      rec_list = []

      # temporary list that will be appended to to rec_list after it is updated for each user
      temp_list = []

      # for loop for users i.e. rows
      for i in range(0, number_of_users):
         temp_list.append(i + 1)

         # for loop for movies i.e. columns
         for j in range(1, number_of_movies):
            if binary_data[i][j] == '1':
               temp_list.append(column_header[j])
         rec_list.append(temp_list)
         temp_list = []

      print(rec_list)

   return rec_list


@app.route('/')
```

```
@app.route('/index/')
@app.route('/home/')
def index():
        return render_template('index_form.html')



@app.route('/check_user_id/', methods=['POST'])
def check_user_id():

    # function to render the page only with user specific recommendations
    user_id = int(request.form['user_id'])
    number_of_movies = int(request.form['no_of_movies'])
    final_rec_list = generate_recommendation()

    # user specific recommendation list is present in serial order of the user_id
    if user_id >= 1:
        user_rec_list = final_rec_list[user_id-1]

    return render_template('user_recommendation_page.html', user_rec_list =
user_rec_list, colour = random_colour_generation(), user_id = user_id, number_of_movies =
number_of_movies)



@app.route('/recommendations_page/')
def recommendations_page():
  # function to render the page with all the recommendations
  final_rec_list = generate_recommendation()

    # change the for-loop range which generates the small box for movies depending
upon the number of movies
  return render_template('recommendation_page.html', final_rec_list = final_rec_list, colour
= random_colour_generation())



if __name__ == '__main__':
  app.run(debug = True)
```

Implementation Details

6.1 Hardware Requirement

Processor: Intel Pentium Family or above
Processor speed: 667 MHz or more
RAM: 4GB or more
Hard Disk: 40 GB or more
Language: Python

6.2 Software and libraries requirement

As the system has been developed in purely on python and its dependent libraries, so its library mentions below

Tools required: Anaconda IDE for executing Python program.

Python 3.6
Python is a widely used high level programming language for general-purpose programming created by Guido van Rossum and first released in 1991.

An interpreted language, Python has a design philosophy which emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly braces or keywords), and a syntax which allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java.

Pytorch
PyTorch is an open source machine learning library for Python, based on Torch, used for applications such as natural language processing.

PyTorch is a python package that provides two high-level features:

- Tensor computation (like NumPy) with strong GPU acceleration.

You can reuse your favorited python packages such as NumPy, SciPy and Python to extend PyTorch when needed.

Pandas
pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

NumPy
NumPy for adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays and to help with calculations during pre-processing.

Flask
Flask is considered more Pythonic than the Django web framework because in common situations the equivalent Flask web application is more explicit. Flask is also easy to get started with as a beginner because there is little boilerplate code for getting a simple app up and running.

6.3 Datasets

The rating information and the required label information is taken from two datasets. Both of them are provided by the Group Lens research lab and contain user ratings of movies (Movie Lens 10M dataset) and tags assigned to movies (tag genome dataset) respectively. The datasets can be combined easily since Group Lens uses consistent movie ids throughout their datasets. The tag information is provided using a specific model termed tag genome. The tag genome model represents an item over a set of available tags by assigning relevance's, on a continuous scale from 0 to 1, for each tag for the considered item. The tag genomes inferred from free-form text descriptors assigned by users to movies.
Ratings of users on movies are collected from the Movie Lens web site, a movie recommendation service where users are asked to rate movies to get recommendations. Ratings are provided on a 0 to 5 scale, where 5 corresponds to highly appreciated.

movies - DataFrame

Index	MovieID	Title	Genres
0	1	Toy Story (1995)	Animation|Children's|Comedy
1	2	Jumanji (1995)	Adventure|Children's|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama
4	5	Father of the Bride Part II (1995)	Comedy
5	6	Heat (1995)	Action|Crime|Thriller
6	7	Sabrina (1995)	Comedy|Romance
7	8	Tom and Huck (1995)	Adventure|Children's
8	9	Sudden Death (1995)	Action
9	10	GoldenEye (1995)	Action|Adventure|Thriller
10	11	American President, The (1995)	Comedy|Drama|Romance
11	12	Dracula: Dead and Loving It (1995)	Comedy|Horror
12	13	Balto (1995)	Animation|Children's
13	14	Nixon (1995)	Drama
14	15	Cutthroat Island (1995)	Action|Adventure|Romance
15	16	Casino (1995)	Drama|Thriller
16	17	Sense and Sensibility (1995)	Drama|Romance
17	18	Four Rooms (1995)	Thriller
18	19	Ace Ventura: When Nature Calls (1995)	Comedy
19	20	Money Train (1995)	Action
20	21	Get Shorty (1995)	Action|Comedy|Drama

Figure 6.3.1: Dataset of movies and its genres

ratings - DataFrame

Index	UserID	MovieID	Rating	Timestamp
0	1	1193	5	978300760
1	1	661	3	978302109
2	1	914	3	978301968
3	1	3408	4	978300275
4	1	2355	5	978824291
5	1	1197	3	978302268
6	1	1287	5	978302039
7	1	2804	5	978300719
8	1	594	4	978302268
9	1	919	4	978301368
10	1	595	5	978824268
11	1	938	4	978301752
12	1	2398	4	978302281
13	1	2918	4	978302124
14	1	1035	5	978301753
15	1	2791	4	978302188
16	1	2687	3	978824268
17	1	2018	4	978301777
18	1	3105	5	978301713
19	1	2797	4	978302039
20	1	2321	3	978302205

Figure 6.3.2: Ratings by user from 1-5

users - DataFrame

Index	UserID	Gender	Age	Occupation	Zip Code
0	1	F	1	10	48067
1	2	M	56	16	70072
2	3	M	25	15	55117
3	4	M	45	7	02460
4	5	M	25	20	55455
5	6	F	50	9	55117
6	7	M	35	1	06810
7	8	M	25	12	11413
8	9	M	25	17	61614
9	10	F	35	1	95370
10	11	F	25	1	04093
11	12	M	25	12	32793
12	13	M	45	1	93304
13	14	M	35	0	60126
14	15	M	25	7	22903
15	16	F	35	0	20670
16	17	M	50	1	95350
17	18	F	18	3	95825
18	19	M	1	10	48073
19	20	M	25	14	55113
20	21	M	18	16	99353

Figure 6.3.3: User Dataset

6.4 Output Generated

output - DataFrame

Index	User-Id	Toy Story (1995)	Jumanji (1995)	npier Old Men (1	iting to Exhale (19	of the Bride Part II	Heat (1995)	Sabrina (1995)	m and Huck (199	idden Death (199
0	1	1	1	1	0	1	1	1	1	1
1	2	0	0	0	0	0	1	0	1	0
2	3	0	0	0	0	0	1	0	0	0
3	4	0	0	0	0	0	1	0	1	0
4	5	0	0	0	0	0	1	0	1	0
5	6	0	0	0	0	0	1	0	1	0
6	7	0	0	0	0	0	1	0	0	0
7	8	0	0	0	0	0	1	0	1	0
8	9	1	0	1	1	0	1	1	1	1
9	10	1	0	1	1	0	1	1	1	1
10	11	1	0	1	1	0	1	1	1	1
11	12	1	1	1	0	1	1	1	1	1
12	13	0	0	0	0	0	1	0	1	0
13	14	1	0	1	1	0	1	1	1	1
14	15	1	0	1	1	0	1	1	1	1
15	16	1	0	1	1	0	1	1	1	1
16	17	0	0	1	0	0	1	0	1	0
17	18	1	0	1	1	0	1	1	1	1
18	19	0	0	1	0	0	1	1	1	0
19	20	0	0	1	0	0	1	1	1	0
20	21	1	0	1	1	0	1	1	1	1

Figure 6.4.1: output csv file(if the user has liked the movie)

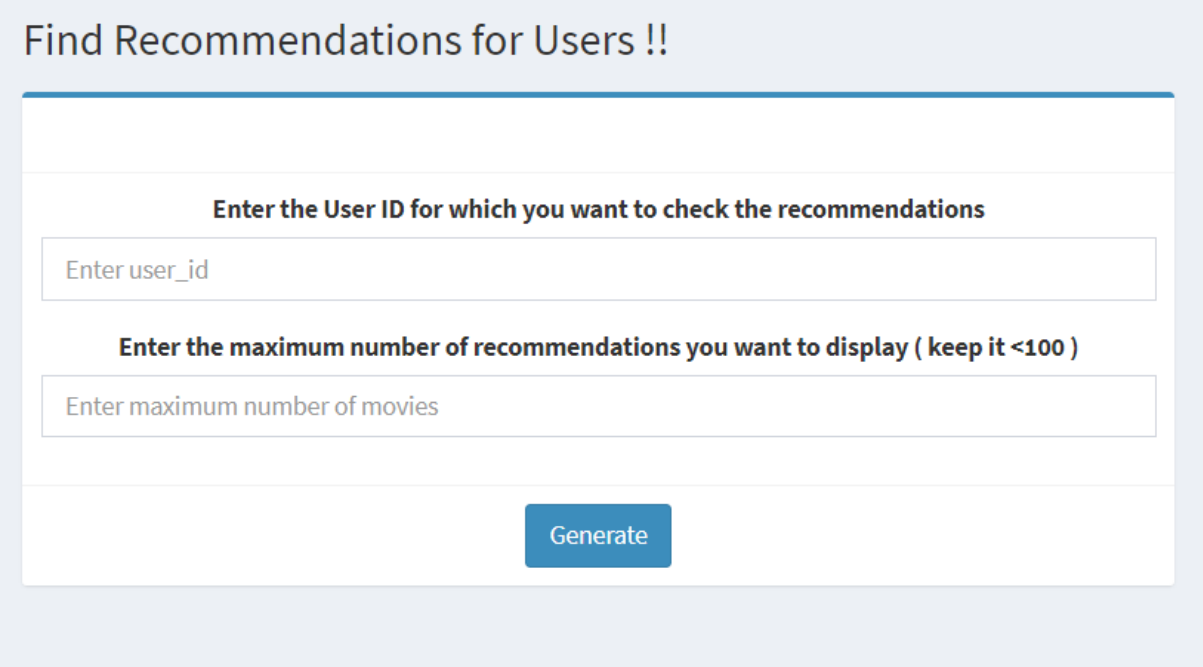

Figure 6.4.2: Graphical User Interface for selecting recommendation for particular user id

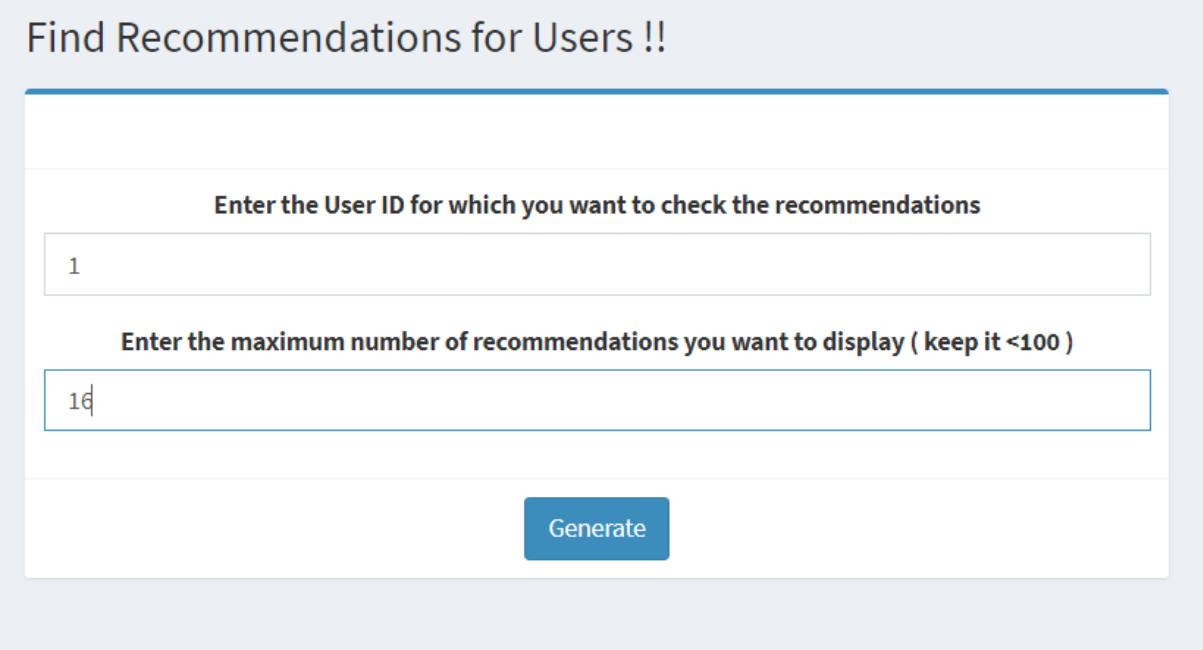

Figure 6.4.3: Graphical User Interface for selecting maximum number of recommendations

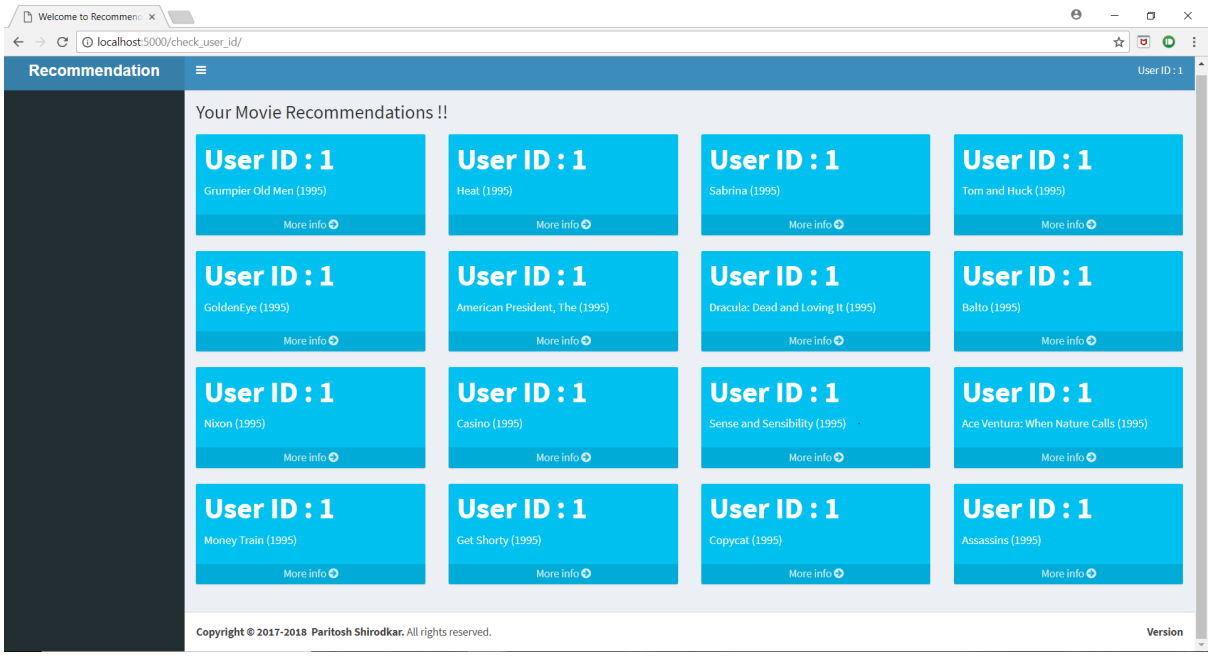

Figure 6.4.4: Movie Recommendation for User ID 1

Conclusion

7.1 Conclusion

We introduced a class of two-layer undirected graphical models (RBM's), suitable for modeling tabular or count data, and presented efficient learning and inference procedures for this class of models. We also demonstrated that RBM's can be successfully applied to a large dataset containing over 100 million user/movie ratings. When the predictions of multiple RBM models and multiple SVD models are linearly combined, we achieve an error rate that is well over 6% better than the score of Netflix's own system.

A recommender system based on additional label information still holds the potential to improve given recommendations which becomes more and more important, when the number of options increases more and more, to provide assistance in make the best choice.
An increasing number of online companies are utilizing recommendation systems to increase user interaction and enrich shopping potential. Use cases of recommendation systems have been expanding rapidly across many aspects of eCommerce and online media over the last 4-5 years, and we expect this trend to continue.
Recommendation systems (often called "recommendation engines") have the potential to change the way websites communicate with users and to allow companies to maximize their ROI based on the information they can gather on each customer's preferences and purchases.
We presented an explainable RBM approach for CF recommendations that achieves both accuracy and interpretability by learning an RBM network that tries to estimate accurate user ratings while also taking into account the explain ability of an item to a user. Both rating prediction and explain ability are integrated within one learning goal allowing to learn a network model that prioritizes the recommendation of items that are explainable.
More than 80 per cent of the TV shows people watch on Netflix are discovered through the platform's recommendation system. That means the majority of what you decide to watch on Netflix is the result of decisions made by a mysterious, black box of an algorithm.

7.2 Benefits

The success of applications that recommend is growing. Recommendation systems are no longer a novelty. They're being built for almost every domain where we can give recommendations, and their advantages are clear.

- **Based on Real Activity**

  The biggest benefit of recommendation systems is that they record, and then base their recommendations on actual user behavior. Their recommendations are not based on guesswork, but on an objective reality. This is the holy grail of design: watching people in their natural environment and making design decisions based directly on the results. Recommendation systems are not perfect, but because they predict the future based on the past, they are remarkably good.

- **Great for Discovery**

  Sometimes it can feel like we're the victims of horribly inefficient advertising. This is apparent when we go to the movies and none of the previews are interesting, or when none of the music on the radio is aligned to our tastes. Recommendations systems help alleviate this problem because they allow us to discover things that are similar to what we already like. And, as in the Beatles/Radiohead example, they can make some pretty surprising recommendations that we probably wouldn't have found out about otherwise.

- **Personalization**

  We often take recommendations from friends and family because we trust their opinion. Part of that trust is built on our relationship: they know us better than anyone else so they know what we like and what we don't like. They're good at recommending things for that very reason. This is what recommendation systems try to model: the intimate knowledge of what we like and don't like.

- **Always Up-To-Date**

  Recommendation systems are dynamically updated, and therefore are always up-to-date. Some new, interesting news? It will be on Digg within minutes. An interesting new movie on Netflix? It quickly gets recommended as long as people rate it highly. The ability for a recommendation system to bubble up activity in real time is a huge advantage because the system is always on.

- **Reduced Organizational Maintenance**

  Building recommendation systems is quite different from how we've built information-rich web sites in the past. For many designers the primary task of building an information-rich web site is creating navigation systems built on top of an underlying taxonomy. The taxonomy is built out of the designer's knowledge of users and the domain, generated from observations made during field research, insights from persona creation, or knowledge gained from other design techniques. Most of the organizational maintenance of a site is keeping the navigation system and taxonomy in line with the users' changing needs.
  With recommendation systems, much of this organizational maintenance goes away. The users organize their own content, in a sense, as the system monitors their constant activity to decide what navigation options go where. What related links could go on a page dedicated to the Beatles? Bob Dylan and Radiohead, for starters. This doesn't mean that navigation-related decisions go away, however. It still takes a designer to decide what type of information should be displayed on what screen.

- **Improving cart value**

  A company with an inventory of thousands and thousands of items would be hard pressed to hard-code product suggestions for all of it's products, and it's obvious that such static suggestions would quickly be out-of-date or irrelevant for many customers. By using various means of "filtering", eCommerce giants can find opportune times to suggest (on their site, via email, or though other means) new products that you're likely to buy.

7.3 Future Enhancements

Learning Deep Generative Models

Recently, (Hinton et al., 2006) derived a way to per- form fast, greedy learning of deep belief networks one layer at a time, with the top two layers forming an undirected bipartite graph which acts as an associative memory.
The learning procedure consists of training a stack of RBM's each having only one layer of latent (hidden) feature detectors. The learned feature activations of one RBM are used as the "data" for training the next RBM in the stack.
An important aspect of this layer-wise training procedure is that, provided the number of features per layer does not decrease, each extra layer increases a lower bound on the log probability of data. So, layer-by-layer training can be recursively applied several times[3] to learn a deep, hierarchical model in which each layer of features captures strong high-order correlations be- tween the activities of features in the layer below.
Learning multi-layer models has been successfully applied in the domain of dimensionality reduction with the resulting models significantly outperforming Latent Semantic Analysis, a well-known document retrieval method based on SVD. It has also been used for modeling temporal data and learning nonlinear embeddings. We are currently exploring this kind of learning for the Netflix data. For classification of the MNIST digits, deep networks reduce the error significantly and our hope is that they will be similarly helpful for the Netflix data.

References

[1] Sarwar, Badrul, et al.” Item-based collaborative filtering recommendation algorithms.” Proceedings of the 10th international conference on World Wide Web. ACM, 2001.

[2] R. Salakhutdinov, A. Mnih, and G. Hinton, “Restricted Boltzmann machines for collaborative filtering,” in Proc. Int. Conf. Mach. Learning, Corvallis, OR, USA, June 2007, pp. 791–798.

[3] (Apr.04,2017) Netflix prize movie rating contest. [Online]. Available: http://www.netflixprize.com/index.html.

[4] B. Abdollahi and O. Nasraoui, “Explainable restricted Boltzmann machines for collaborative filtering,” in Proc. ICML Workshop Human Interpretability in Mach. Learning, New York, NY, USA, June 2016, pp. 31–35

[5] Y. Liu, Q. Tong, Z. Du, and L. Hu, “Content-boosted restricted Boltzmann machine for recommendation,” in Lecture Notes in Computer Science, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, C. Pandu Rangan, B. Steffen, D. Terzopoulos

[6] Nikolas Alexander Huhnstock," Evaluation of Label incorporated recommender Systems based on restricted Boltzmann Machines"

[7] LeCun, Y., Bengio, Y. & Hinton, G.E., 2015. Deep learning. Nature, 521(7553), pp.436–444.

[8] Bogers, T. & Van Den Bosch, A., 2011. Recommender Systems Handbook F. Ricci et al., eds., Boston, MA: Springer US.

[9] Burke, R., 2007. Hybrid web recommender systems. In The adaptive web. Heidelberg, Berlin: Springer, pp. 377–408

[10] Su, X. & Khoshgoftaar, T.M., 2009. A Survey of Collaborative Filtering Techniques. Advances in Artificial Intelligence, 4.

[11] Hinton, G.E., 2002. Training products of experts by minimizing contrastive divergence. Neural computation, 14(8), pp.1771–1800

[12] Ranzato, M.A. & Hinton, G.E., 2010. Factored 3-Way Restricted Boltzmann Machines For Modeling Natural Images. Artificial Intelligence, 9, pp.621–628.

[13] Fischer, A. & Igel, C., 2014. Training restricted Boltzmann machines: An introduction. Pattern Recognition, 47(1), pp.25–39.