# 

|  |  |  |
| --- | --- | --- |
|  | | |
|  | RECOMMENDER SYSTEM |  |

**-Jeevisha Anandani**

**M13449421**

**Submitted to: -**

Dr. Peng Wang

Dr. Michael Thompson

**Abstract**

This project compares two approaches to build a movie recommender system. First one implements Bayesian Network by learning the Conditional Probability Distributions from the data. Bayesian Networks belong to a class of algorithms known as probabilistic graphical models. This Bayesian network is then used to infer queries that are in turn used for collaborative filtering. It focuses on the sampling method which will help the system recommend movies for new users. Also, it uses Maximum Likelihood Estimation to estimate the conditional probability distributions. The other approach of Collaborative Filtering (Matrix Factorization) is applied to view the recommendations and the results are compared with the prior approach. Alternating Least Squares algorithm was implemented to predict user ratings using PySpark.

**Contents**

**Abstract…...………………………………………………………………………………………1**

**Introduction………………………………………………………………………………………3**

**Problem Description……………..………………………………………………………………3**

**Methodology...……………………………………………………………………………………4**

**Dataset……………………………………………………………………………………4**

**Visualizations…………………………………………………………………………….4**

**Approaches……………………………………………………………………………………….5**

**Bayesian Network………………………………………………………………………..5**

**Matrix Factorization……………………………………………………………………..6**

**Results…………………………………………………………………………………………….8**

**Conclusions……………………………………………………………………………………...10**

**Code……………………………………………………………………………………………...10**

**References……………………………………………………………………………………….10**

**Introduction**

Companies like Amazon, Netflix, and LinkedIn leverage recommender systems to help users discover new and relevant items, creating a delightful user experience while driving incremental revenue. These systems can operate using a single input, like music, or multiple inputs within and across platforms like news, books, and search queries. The need to build robust recommendation systems is extremely important given the huge demand for personalized content of modern consumers. It helps the user to select the right item by suggesting a presumable list of items and so it has become an integral part of e-commerce, movie and music rendering sites and the list goes on. They are becoming one of the most popular applications of machine learning which has gained importance in recent years.

**Types of Recommender Systems-**

**Content Based-** One popular technique of recommendation systems is content-based filtering**.**Content here refers to the content or attributes of the products you like. So, the idea in content-based filtering is to tag products using certain keywords, understand what the user likes, look up those keywords in the database and recommend different products with the same attributes.

**Collaborative Filtering-** It doesn’t need anything else except users’ historical preference on a set of items because it’s based on historical data, the core assumption here is that the users who have agreed in the past tend to also agree in the future. In terms of user preference, it is usually expressed by two categories. Explicit Rating is a rate given by a user to an item on a sliding scale, like 5 stars for Titanic. This is the most direct feedback from users to show how much they like an item. Implicit Rating suggests user’s preference indirectly, such as page views, clicks, purchase records, whether listen to a music track, and so on.

**Hybrid Model-** Hybrid recommender system is the one that combines multiple recommendation techniques together to produce the output. If one compares hybrid recommender systems with collaborative or content-based systems, the recommendation accuracy is usually higher in hybrid systems. The reason is the lack of information about the domain dependencies in collaborative filtering, and about the people’s preferences in content-based system. The combination of both leads to common knowledge increase, which contributes to better recommendations. The knowledge increase makes it especially promising to explore new ways to extend underlying collaborative filtering algorithms with content data and content-based algorithms with the user behavior data.

**Problem Description**

Recommender systems are information filtering tools that aspire to predict the rating for users and items, predominantly from big data to recommend their likes. Movie recommendation systems provide a mechanism to assist users in classifying users with similar interests. Collaborative Filtering algorithm is one of the popular successful techniques of RS, which aims to find users closely similar to the active one in order to recommend items.

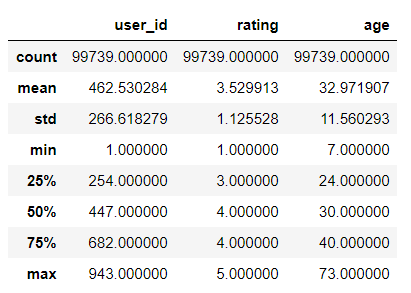
In this paper I propose two approaches to recommend movies and present their comparative analysis. First one is by implementing Bayesian Network by learning the Conditional Probability Distributions and the second is with Alternating Least Squares (ALS) algorithm. The ALS algorithm is one of the models of matrix factorization related CF which is considered as the values in the item list of user matrix.

**Methodology**

**Dataset**

The data contains 100,000 ratings by users and is taken from the MovieLens dataset and comprises of movie titles, genres and user IDs apart from ratings.

**Data Distribution**

 Figure 1

**Distribution of Movie Counts**

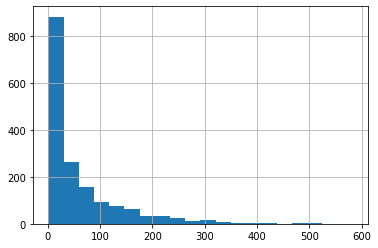
****

Figure 2

**Distribution of Average Number of Movies**

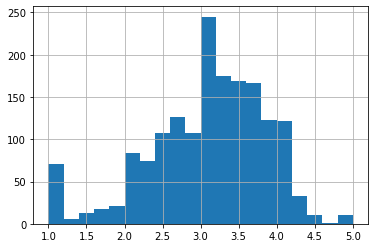
****

Figure 3

**Approaches**

**Bayesian Network**

I explored the movies dataset and performed exploratory analysis to understand the user behavior. I initially used Bayesian Networks in graphical models to recommend movies. For such models, a directed graph expresses the conditional independence assumptions between random variables, which are represented as nodes. They provide a compact representation of joint probability distributions. The advantage of BNs is that they can encode variable relationships that can be learned from data. They can be applied to infer the outcome of collaborative filtering, a technique by which we can predict outcome of a query using the information about collaboration among multiple agents. This is done by observing attributes of both the entities for which the collaboration is to be inferred. In this, for learning parameters and creating the Bayesian Network I have used the bnlearn library. This library mainly focusses on implementing Probabilistic Graphical Models in R. The learning process involves finding the Bayesian network that most accurately models data given as input – in other words, finding the Bayesian network that makes the data set most likely.

We have used Tabular conditional Probability Distributions in this project. These Conditional Probability Distributions are calculated using the data. Parameter estimation is performed by using maximum likelihood estimation. It is a method that determines values for the parameters of a model. The parameter values are found such that they maximize the likelihood that the process described by the model produced the data that were observed. In the context of this project, I am learning the Conditional probability distributions (CPD) of the Bayesian network using the most likely setting of the distribution, given the dataset. The estimates for each CPD only needs information from the parent, and this decomposition makes it possible to learn the parameters of each CPD.

Likelihood of parameter θ given data set: L(θ|D) = P(D|θ), where L(θ|D) is the likelihood of the parameter θ, given the dataset D.

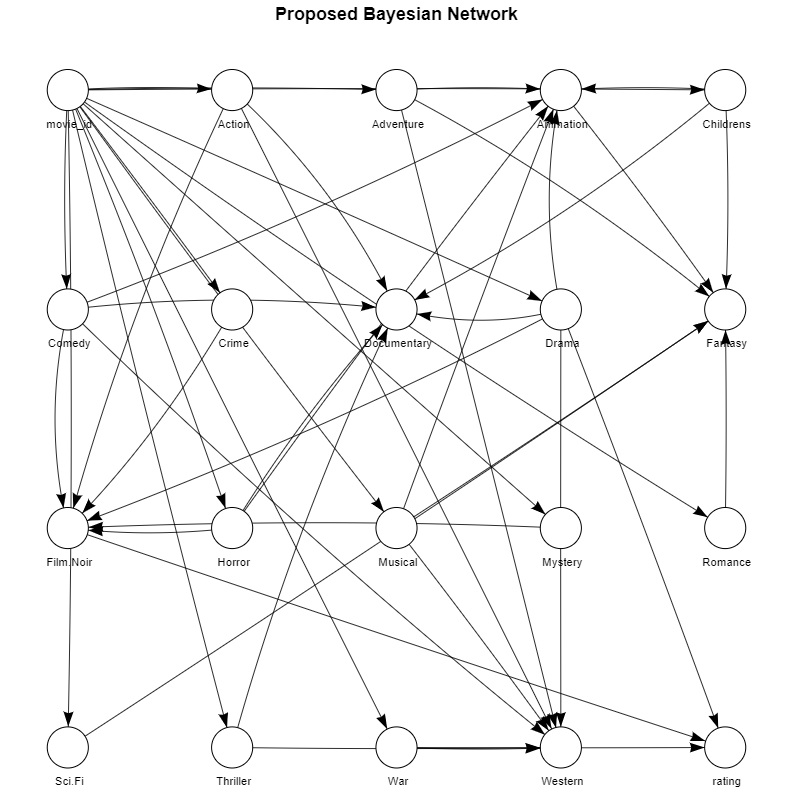


Figure 4

For the Bayesian Model, I have used RMSE has a performance indicator to assess how well has the model understood the user behavior. I split the data 80:20 and fit the model.

**Matrix Factorization**

With collaborative filtering, the idea is to approximate the rating matrix by factorizing it as the product of two matrices. That is the one that describes properties of each user (shown in green), and the other describing properties of each movie. The minimization of the error for the users/movies pairs was chosen as the basis for the selection of the two matrices. The alternating least squares algorithm (ALS) which achieves this by randomly filling the user’s matrix with values before optimizing the value of the movies was used for this purpose. The value of the user’s matrix is optimized with the movie’s matrix being kept constant. Owing to a fixed set of user factors (i.e., values in the user’s matrix), known ratings are employed to find the best values by optimizing the movie factors, written on top of the figure. The best user factor with the fixed movie factors is sleeted.

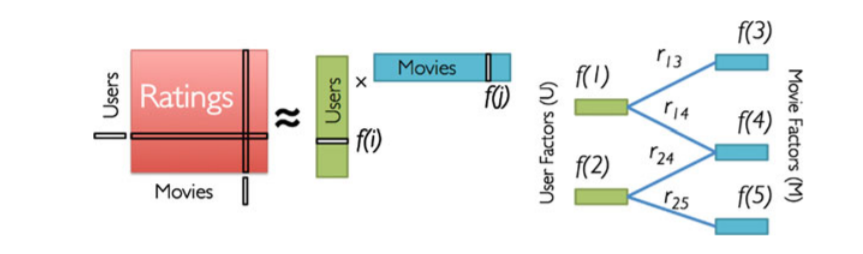


Figure 5

When using a Matrix Factorization approach to implement a recommendation algorithm you decompose your large user/item matrix into lower dimensional user factors and item factors. It will allow us to discover the latent features that define the interactions between User and Ratings. In other words, ALS uncovers the latent features. In the simplest approach you can then estimate the user rating by multiplying those factors according to the following equation:



In order to learn those factors you need to minimize the following quadratic loss function:



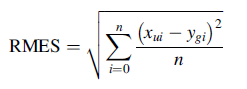
In an SGD (Stochastic Gradient descent) approach, for each example in the dataset one can compute the error  and then update the parameters by a factor in the opposite direction of the gradient. Alternating Least Squares (ALS) represents a different approach to optimizing the loss function. The key insight is that you can turn the non-convex optimization problem in Equation (2) into an easy quadratic problem if you fix either  ALS fixes each one of those alternatively. When one is fixed, the other one is computed, and vice versa.

There are two main benefits of this approach. First, this is very easy to parallelize. Second, whenever dealing with implicit datasets, which are usually not sparse, SGD is not practical (users times items can easily be in the order of billions). ALS is a much more efficient optimization technique in these cases. Spark MLlib library for Machine Learning provides a Collaborative Filtering implementation by using Alternating Least Squares. is a convenient Python library that interfaces with Spark. For large datasets, a Spark-based system has advantages because data imported into Spark RDD's/ Dataframes is partitioned and can be easily worked upon in parallel.

The implementation in MLlib has these parameters:

* **rank** is the number of latent factors in the model.
* **iterations** is the number of iterations to run.
* **implicitPrefs** specifies whether to use the explicit feedback ALS variant or one adapted for implicit feedback data.
* **alpha** is a parameter applicable to the implicit feedback variant of ALS that governs the baseline confidence in preference observations.
* **coldStartStrategy** is used when there is no data for a user which might lead to null prediction if the user on the test set has no rating in the training set. I have dropped the cold start strategy because I want to avoid such situations for our problem in hand.

In the ALS model as well, root mean squared error (RMSE) is used as a performance measure. RMSE works by measuring the difference between error rate a user gives to the system and the predicted error by the model. Below equation depicts how RMSE works on movie recommender system: -



whereby xui is the rating that user u gives to an item i in the experimental data, ygi is a predicted rating that the movie that user u gives to an item and where n is the number of ratings in the test data.

**Results**

Following are the comparison of both the models based on RMSE: -

|  |  |  |
| --- | --- | --- |
|  | Bayesian Model | Matrix Factorization |
| RMSE | 1.45 | 0.923 |

On average the mean error is the difference between the original rating and the predicted rating. Given the RMSE is lesser for the matrix factorization approach as well as the execution speed is also very less as compared to the previous one, I felt that ALS is a more suited approach for this dataset. The I tried to incorporate the user\_id in one of the nodes but the system kept crashing because of which I turned to a much better and scalable solution.

When selecting the ALS algorithm as a part of building the proposed movie recommender system, there is basic parameter through them I can determine the best rating of users for given movies. These parameters are Rank and Iterations. The parameters have been used in order to control and adjust the predicting capability of matrix factorization which is depending on ALS technique which in turn affect the evaluation of movie Recommendation System. The parameters lambda and iteration in ALS model are used with different thresholds to realize the effects of matrix factorization performance on the performance of recommendation results and thus take the most appropriate parameters by the bestModel parameter. The new RMSE with the updated parameters came out to be **0.923** which was a slight improvement over the previous one.

Also, the predictions for the ALS show that they are very close to the original ratings. To show a sample comparison between original and predicted rating, following is the sample output. Here we can see that for most of the 4/5 ratings the prediction comes out to be within the same range.

|  |  |  |  |
| --- | --- | --- | --- |
| **User** | **Movie** | **Prediction** | **Rating** |
| 833 | 320 | 4.49 | 4 |
| 148 | 169 | 4.92 | 5 |
| 392 | 178 | 4.85 | 5 |
| 623 | 50 | 4.47 | 5 |
| 65 | 318 | 4.63 | 5 |
| 593 | 318 | 4.53 | 5 |

Top 5 recommended movies based on the users viewing history from the final model came out to be: -

|  |  |  |  |
| --- | --- | --- | --- |
| **Movie** | **Movie Title** | **User** | **Prediction** |
| 1033 | Homeward Bound II | 471 | 5.43 |
| 1233 | Nenette Boni | 471 | 5.01 |
| 990 | Anna Karenina | 471 | 4.91 |
| 323 | Dante’s Peak | 471 | 4.90 |
| 1025 | Fire Down Below | 471 | 4.87 |
| 1643 | Angel Baby | 463 | 4.64 |
| 1449 | Pather Panchali | 463 | 4.51 |
| 1137 | Beautiful Thing | 463 | 4.31 |
| 1159 | Stalker | 463 | 4.30 |
| 1193 | Before the Rain | 463 | 4.29 |
| 1643 | Angel Baby | 833 | 4.99 |
| 320 | Paradise Lost | 833 | 4.49 |
| 1367 | Faust | 833 | 4.31 |
| 1368 | Mina Tannebaum | 833 | 4.28 |
| 1005 | Double vie de ver | 833 | 4.26 |
| 1643 | Angel Baby | 496 | 4.66 |
| 75 | Brother Minister | 496 | 4.55 |
| 1019 | Die Xue Shuang Xi | 496 | 4.36 |
| 115 | The Haunted World of El Superbeasto | 496 | 4.33 |
| 1137 | Beautiful Thing | 496 | 4.32 |

**Conclusion**

From the above comparisons, I can see that although we can use Bayesian Networks to predict whether a user will like the movie or not it does have scalability issues and if I use user\_id as a node the code is not able to process. On the other hand, I was able to execute an ALS recommender system in much lesser time to predict movie ratings for users that too with much closer predictions. The approach used is highly scalable and can be used with computational clusters. Moreover, my data was not sparse, but ALS can also be used in that case as it is designed for both explicit and implicit ratings.

**Code**

You can find the link to my github account containing the code:-

<https://github.com/jeevisha3008/Recommender-System>

**References**

<https://sci-hub.tw/10.1109/icmla.2018.00069>

<https://learning.oreilly.com/library/view/building-probabilistic-graphical/9781783289004/ch05s04.html>

<https://www.researchgate.net/publication/41663037_Designing_a_Context_Dependent_Movie_Recommender_A_Hierarchical_Bayesian_Approach>

<http://eeweb.poly.edu/faculty/yongliu/docs/SN_recomm_tdke.pdf>

<https://pdfs.semanticscholar.org/0ffd/d7515e342b70d4db1201f33cbbfd68a5771c.pdf>

<https://www.researchgate.net/publication/326146390_Movie_Recommender_System_Based_on_Collaborative_Filtering_Using_Apache_Spark>

<https://campus.datacamp.com/courses/big-data-fundamentals-with-pyspark/machine-learning-with-pyspark-mllib?ex=4>

<https://towardsdatascience.com/building-a-recommendation-engine-to-recommend-books-in-spark-f09334d47d67>

<https://www.linkedin.com/pulse/bayesian-network-continuous-distributions-model-wine-fernandes/>