Movie Recommendation System with Apache Spark

Alexandros Ferles ferles@kth.se
George Zervakis zervakis@kth.se

Abstract

Recommendation systems is a very interesting research area of Machine Learning. The ability to predict users needs based on past activity and similar users behaviour, and offer them suggestions that reflect on their interests, is a critical factor in providing a reliable service. Since the number of possible options in such systems is already huge and constantly growing, there is a necessity to combine Big Data and Data Intensive applications with recommendation systems. One interesting use case scenario that is already met in high profile services like IMDB and Netflix, is a movie recommendation system. Through this work, we study the state-of-the art Alternating Least squares method in the Netflix prize Kaggle competition. More importantly, we propose a content-based movie recommendation system, that is able to provide meaningful suggestions regarding future movies to watch, based on past users ratings.

1 Introduction

Recommendation systems are an interesting case study in the field of Machine Learning. The study and design of such systems, can be met in several user services, spanning from the well known song, book, movie recommendation systems to product-buying services in general. The typical form of a recommendation algorithm is to uncover similarities between a range of different users, rating a product so that they can predict efficiently future suggestions to be made at each individual user. Under the Machine Learning study scope, it usually takes some form of clustering users, and deriving similar products from users that come from the same cluster. In movie recommendation systems, typically that means accumulating the reviews each user has provided at movies, and deriving some clusters of interest. Since the number of available movies and reviews is significantly higher than the interests of each user, sparsity in the available data is a common issue. Hence, the need to tackle the problem of sparsity is a very important factor in designing such a system. Through this work, we make use of the Alternating Least Squares algorithm that has been successfully applied in the MovieLens dataset, studying its performance in the Netflix dataset. However, the most important contribution of this paper is a content-based recommendation system, that through Natural Language Processing uncovers similarities between movies and users, based on the movies plot summaries found in the Movie Corpus dataset.

2 Data

The datasets used for the study and design of our recommendation systems are the following:

- The **MovieLens dataset**[1] was created by GroupLens Research and it contains ratings from a great number of movies, varying from movie ratings published in 1998 till today. There exist multiple versions of this dataset, each one more suitable depending on the task at hand. For the purposes of this project, we utilized the *small* and *full* versions of MovieLens. The former consists of 100K ratings and 3.6K tag applications applied to 9K movies by 600

[1] `https://grouplens.org/datasets/movielens/`

users and it was used during the training phase. The latter is composed of 27M ratings and 1.1M tag applications applied to 58K movies by 280K users and it was used for evaluation. Since the Alternating Least Squares method has been succesfully applied to MovieLens several times, we provide this dataset only for study purposes.

- The **Netflix Prize dataset**[2] was originally used in the Netflix Prize open competition for the best algorithm to predict user ratings for films. The collection consists of several files, including the training dataset which contains 17.770 movie ratings on a 1-5 star scale along with the corresponding dates, from 480.189 users, the movies file which holds info like movie id, year of release e.t.c. In terms of test sets, the qualifying set is provided for local test purposes. Full evaluation of the contestant's work was based on the probe data, where the true values were unknown.
- The **CMU Book Summary Dataset**[3][1] comprises 42.306 movie plot summaries taken from Wikipedia plus aligned metadata extracted from Freebase, including:
  - Movie box office revenue, genre, release date, runtime and language
  - Character names and aligned information about the actors who portray them, including gender and estimated age at the time of the movie's release.

3 Methods

3.1 Alternating Least Squares (ALS)

Formulating a problem such as a recommendation system based on user's reviews can be seen as having a matrix that stores the ratings for each user. Under a real-life setting, the number of ratings for each user is far more high than in the actual problem formulation, thus the matrix under study is expected to be $sparse$. *Matrix Factorization* refers to the studying and implementations of techniques that handles sparse data, by predicting the values of the blank points in such matrices. *Alternating Least Squares* (ALS) is a Matrix Factorization technique that has been successfully applied in Movie Recommendation systems and can be found under various solution proposed on the MovieLens dataset. Spark's *Machine Learning library* comes up with an implementation of ALS, which we study and make use of in the Netflix dataset.

We define a matrix A of $N \times M$ dimension, where a_{ij} is the $(i,j)^{th}$ element of the matrix, that corresponds to the rating of item j from user i, N is the number of users and M is the number of reviewed products. Let x_i, y_j be two k dimensional vectors (where K is a small number) that represent a summary of each user and reviewed products respectively. We can then make a prediction for a_{ij} simply by calculating $x_i^\top y_j$. If we extend this for all N users and M items, we have a $K \times M$ user matrix X and a $K \times M$ item matrix Y so that we can approximate ratings matrix as $A \approx X^\top Y$. This turns out to be an optimization problem where we seek for optimal X, Y by minimizing the least square error of the known ratings:

$$min \sum_{known a_{ij}} (a_{ij} - x_i^\top y_j)^2 + \lambda(\sum_i \|x_i\|^2 + \|y_j\|^2)$$

3.2 Document-to-Vector Content Based Recommendation System

Another basis in which we can build a recommendation system, is a *Content-based recommendation system*. Instead of representing the whole content a matrix representation of users and ratings, we seek to formulate the content of each product in an interpretable manner, and find similarities based on this formulation.

In this method, we chose to use the Corpus Movie dataset, and represent the plot summaries available for some of the movies that come up with this dataset, through *Natural Language Processing operations*.

As a result, we chose the Doc2Vec[4] model, which builds up on the Word2Vec model[5] in order to represent texts in an array form. This way, we were able to transform the given summaries to matrix

[2] `https://www.kaggle.com/netflix-inc/netflix-prize-data/home.`

[3] `http://www.cs.cmu.edu/~ark/personas/`

representations, which we used in order to find similarities with reviewed movies that also come up with their plot summaries available.

Our metric of similarity is the cosine similarity, which is widely used in search engines and information retrieval settings:

$$similarity = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A^2}\sqrt{\sum_{i=1}^{n} B^2}}$$

In this setting, in order to produce a recommendation based on a user's review, we generate matrix representation of all the reviewed movies, and produce a mean representation of these representations. Since the importance of each movie is not equal among the others, we create a weighted average schema so that movies with higher ratings have greater contribution in the production of the mean vector:

$$mean\_vector = \sum_{movie=1}^{N} w_{movie.review} * vector(movie.summary)$$

An important note to be made, is that the weights should sum up to 1, so that the generated mean vector shares values in the same range in comparison with its similar matrix representations. Furthermore, movies with ratings equal to 1 (lowest rating possible) are considered as outliers and excluded from our computations. For this particular setting, we made sure that all ratings in the discrete space $[1.5, 5]$ with a step of 0.5 were made available, so that we can create cluster of them, and average then before computing their weighted contribution to the mean vector, otherwise we would need too deal with sparse review spaces and thus need to design a complex procedure on how to assign weights that represent the importance of each review efficiently and stil satisfy the condition to sum up to 1.

The advantage of this method is that we do not need to provide suggestions based only on movies that belong in the intersection of the user's reviewed movies and the ones that are available to the dataset. Even if a user provides us with reviews to movies that we come up to for the first time, we are still able to generate a mean vector representation based on their plots and then compare them with our available representations in terms of cosine similarity, and deliver meaningful suggestions.

4 Evaluation

4.1 Root Mean Square Error (RMSE)

For our work regarding the alternate least squares method, we evaluate our work in terms of *Root Mean Square Error* (RMSE) in the test set that was generated with a random split of the original dataset:

The test set RMSE performance was **0.8419420411595426** which is in the same range with other reported RMSE error performances.

4.2 Human based evaluation

For the content based recommendation system, we asked two individuals to provide us with their personal reviews regarding some movies. We then generated some top suggestions for them and asked for their feedback. In general terms, the results can be allocated in two categories: i) Suggested movies that the users had already seen and ii) Suggested movies they had not. In terms of the first category, their feedback was that they liked these movies, and in terms of the second category their feedback was that from the plot summary they were interested to watch these movies.

5 Discussion and Conclusion

5.1 Future work

Through this work we presented a user-based recommendation system that takes advantage of the Alternate Least Squares method along with a content-based recommendation system. While in general we are satisfied with their performance, we discuss the following point to improve and expand our methods:

- Study and try a handful of other Collaborative Filtering techniques such as [3] that uses a combination of Genetic Algorithms and Self-Organizing Maps, or use methods from the work of [2] who won the Netflix prize data contest.
- Find a way to handle sparse reviews in the content based recommendation system.
- Implement a REST API service for our recommendation system, which was already listed in our optional part work in our project proposal and were not available to implement due to time constraints.

6 Technical details

The source code of our work is provided in two separate Jupyter Notebook, one for each method. Moreover, it is implemented in PySpark. In case you wish to execute the notebooks instead of using them for read only purposes, you need to have Spark installed in your local computer. Furthermore, you need to manipulate some environment variables in order to have a fully operating Pyspark Jupyter notebook.

The contents of the .bashrc or .bashprofile file in your local machine should look like this:

export PYSPARK_PYTHON=python3.7

export PYSPARK_DRIVER_PYTHON=jupyter
export PYSPARK_DRIVER_PYTHON_OPTS='notebook'

where you should replace the python version with the one you are using.

Last but not least, the datasets are not submitted. Should you find any trouble with downloading them, we are available to provide them at your disposal.

References

[1] D. Bamman, B. O'Connor, and N. A. Smith. Learning latent personas of film characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 352–361, 2013.

[2] Y. Koren. The bellkor solution to the netflix grand prize. *Netflix prize documentation*, 81:1–10, 2009.

[3] N. P. Kumar and Z. Fan. Hybrid user-item based collaborative filtering. *Procedia Computer Science*, 60:1453–1461, 2015.

[4] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1188–II–1196. JMLR.org, 2014.

[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.