Survey of Machine Learning algorithms for a Movie Recommendation System

CMPE 297-02 Machine Learning

Archana Ramalingam (SJSU ID: 010761114)

Contents

Abstract

Machine Learning has found its way into almost every domain in our everyday life. One such application is the movie recommendation system. Jinni, Netflix, IMDB, Rotten Tomatoes are some of the popular movie recommendation engines. Currently, Netflix is using advanced deep learning techniques to perform movie recommendation.

Most commonly used algorithm for recommendation systems uses *Collaborative Filtering* method, which itself is divided into *Neighbor based, Bayesian based* or *Hybrid*. *K-means* algorithm has also proved to be fruitful in clustering the similar movies. Content based recommendation method is another popular contender in this race. *Decision tree* and *Matrix factorization* are other algorithms that could solve the problem.

In this report, I will be surveying various Machine Learning algorithms that can be used to model a movie recommendation system. I will then pick the most relevant and feasible algorithms and compare their performances. Apart from the algorithm implementation in Python, I will also perform pre-processing tasks including data cleaning, feature engineering, and post-processing tasks like data visualization and validation.

I will be using datasets from a research based movie recommendation site, *MovieLens.com*. The language used for this project will be *Python*. The libraries used to implement the algorithm will be from *Scikit-Learn*.

Further, this analysis can be extended to other common recommendation systems with applications in various domains ranging from e-commerce (Amazon), Music/Audio applications (Spotify), products (Walmart) to news articles (Google News).

1 Introduction

A recommendation system (RS) uses various artificial intelligence (AI) methods to provide recommendation to users in numerous applications. The first RS, Tapestry, was introduced in 1992. Since then RS has evolved with researchers studying machine learning (ML) algorithms to solve the problem. Today, with large number of machine learning methods and their use in applications like Computer vision, self-driving cars, pattern recognition and RS, ML's potential looks promising.

RS finds application in various domains like e-commerce (Amazon), Movie streaming (Netflix), Video streaming (YouTube), Search engines (Google). Other popular sites using RS are LinkedIn, Hulu, Facebook, Twitter, Google, Pandora, Goodreads. Use of RS has become a very common element of modern web. With the help of user preference history, RS can choose an item of interest for the user. RS saves people's time and energy of letting them find their best choice by other's recommendations rather than by themselves. RS can pick and choose webpages, restaurants, articles, grocery products, movies, etc. The potential for an efficient system is unlimited. Various ML algorithms are being used in RS, with each having their own pros and cons. Hence choosing an efficient algorithm for an RS mostly depends on the application. In this paper, I will be discussing popular ML algorithms used to build a RS system, along with its efficiency and downsides.

1.1 Types of approaches in Recommendation Systems

Usually RS is divided, based on the information used to drive the system's recommendation, into three categories: collaborative, content-based, hybrid filtering.

- **Collaborative filtering (CF):** The user data is used to perform recommendation. For example, information from user profiles in an online book store, like age, location, purchase history, can be used to identify users with same preference and suggest books bought by similar users.
- **Content-based filtering:** This approach uses item data accessed by the user. When a user searches the system for an item, RS gathers information about the item and searches the database for items with similar attributes. This result is returned to the user
- **Hybrid filtering:** It combines both the above approaches. This involves using both the user and item data information to recommend items. First the RS will return a list of items based on the user information (Collaborative filtering) and then the RS will use this list of item's information to recommend items like the ones recommended by the first method (content-based filtering).

1.2 Types of data used by Recommendation Systems

RS basically works based on the data available to it. The kinds of data required by a RS to perform efficiently are:

- **User Behavior:**
    - On-site activity – clicks, page and item views, searches

    - Off-site activity – keeping track of clicks in emails, notifications and mobile applications
- **Item details:**
    - Name
    - Description
    - Price
    - Category
- **Contextual information:**
    - Location
    - Type of device used (smartphone, laptop)
    - Referral URL

All the three kinds of data are necessary to build an efficient RS. It should be noted that, apart from the user preferences, we should also consider details like location, device of usage, etc.

1.3 Types of practical applications for Recommendation Systems

- **Product recommendation:** It understands the preferences and intent of the visitor and uses it to show relevant product recommendations in real time. As it gains more information from each user, its performance improves.
- **Website personalization:** This could also be termed as market segmentation, where the visitors are categorized into segments and target them with real time offers and personalized messages.
- **Real-time notifications:** Displaying real time data like, user's activities, number of visitors at current time stamp, etc. increases the reliability and trust for the brand.

2 Algorithms used in Recommendation Systems

Each of the approach discussed in the previous chapter contains multiple types and various algorithms used under each type. In this chapter, we will be discussing the types of algorithms and their comparison.

2.1 Collaborative filtering (CF)

CF is one of the most efficient technique for RS. It is broadly classified as: Memory based, model-based and hybrid.

2.1.1 Memory based CF

Here the algorithms use entire or sample of user-item database, by combining every user into a group of people. When a new user arrives, depending on their neighbors, recommendations are predicted. *Neighborhood-based algorithm* is one of the prominent one under this approach. This technique follows steps as: Calculate similarity or weight, which defines the distance or correlation between two users; predict a recommendation for the user with weighted average of all ratings of the items or users. For a top-N recommendation, we find *k* nearest or similar users and then arrange them in descending order of similarity.

Calculating similarity, between two users or two items, is a very important task in this approach. Below we will be discussing the kinds of similarity computations commonly used along with this algorithm.

- **Correlation based similarity**

  Here we use *Pearson* correlation, which estimates the magnitude to which two variables linearly relate to each other. For user-based algorithm, the *Pearson correlation* is given by:

$$w_{u,v} = \frac{\sum_{i \in I}(r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I}(r_{u,i} - \bar{r}_u)^2}\sqrt{\sum_{i \in I}(r_{v,i} - \bar{r}_v)^2}},$$

  where, $i \in I$ sum over the items that both users, u and v have rated, r_u is the average rating of the items rated by both users.

  For item-based algorithm, the *Pearson correlation* is given by:

$$w_{i,j} = \frac{\sum_{u \in U}(r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U}(r_{u,i} - \bar{r}_i)^2}\sqrt{\sum_{u \in U}(r_{u,j} - \bar{r}_j)^2}},$$

  where, $r_{u,i}$ is the rating of user u to item i and i and j are items.

- **Vector Cosine-based similarity**

If *R* is a *m x n* user-item matrix, the similarity between two items, i and j is given by cosine of the n-dimensional vectors of i^{th} and j^{th} columns of matrix R. The vector cosine similarity between items i and j are given by:

$$w_{i,j} = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \bullet \vec{j}}{\|\vec{i}\| * \|\vec{j}\|},$$

2.1.2 Model based CF

By designing complex models and algorithms, we can let the RS to predict complex patterns on its own, by training the system with the data. These are more efficient than memory based CF. CF models could be classification algorithms for categorical user ratings and SVD methods for numerical ratings. Common algorithms in this space are discussed here. *Bayesian belief net CF* (BN) is a directed, acyclic graph (DAG), where each node $n \epsilon N$ is a random variable, each directed arc $a \epsilon A$ is a probabilistic association between variables and Θ is conditional probability estimating how much a node depends on its parents. Although this has multiple flavors, the basic one uses *Naïve Bayes* strategy as shown below.

$$\text{class} = \arg\max_{j \in \text{classSet}} p(\text{class}_j) \prod_o P(X_o = x_o \mid \text{class}_j).$$

Clustering CF has a cluster of similar data objects, dissimilar to the objects in different group. The similarity is measured between two items or users using *Minkowski* distance as shown below.

$$d(X, Y) = \sqrt[q]{\sum_{i=1}^{n} |x_i - y_i|^q},$$

Where, *n* is dimension of object, x_i, y_i are *ith* dimension of object X and Y. If *q* is 1, it is M*anhattan distance* and if q is 2 it is *Euclidian distance.* They have better scalability, as they work on groups or clusters rather than entire database at a given time.

Regression based CF algorithms are good at predictions for numerical values. Two vectors distant by Euclidean distance could be similar by Pearson or cosine similarity, in which case memory-based CF fails. Its predictions are based on a regression model.

MDP based models view RS problem as a sequential optimization problem and uses *Markov decision processes (MDP). Latent semantic CF, sparse factor analysis, CF using dimensionality reduction techniques* are few other algorithms under this method.

Hybrid CF combines the best of both the above methods to overcome their limitations. Content-based, content-boosted CF, Personality diagnosis are a few types of Hybrid CF.

2.2 Content based filtering

This approach uses: term frequency (TF) and inverse document frequency (IDF). TF is the frequency of a word in a document, IDF is inverse of document frequency. Most frequent word could be most unimportant word too, so TF-IDF is measured as below instead as a raw count. Log is used to dampen the effect of high frequency word.

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d}, & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

After TF-IDF is calculated, we use vector space models to view the ratings in vector form. Then we take the cosine of the angle between the vectors to determine the similarity. Cosine is used because, value of cosine increases with decreasing angle, hence denoting more similarity.

2.3 Comparison

Among the CF techniques, memory based CF is easy to implement, content of items are not necessary and new data addition is easy; but are dependent on human ratings, weak on sparse data and cannot predict for new users/items. Model based CF addresses the sparsity, scalability problems with better efficiency; but is expensive to build, has a trade-off between prediction efficiency and scalability and might lose useful information with dimensionality reduction. *Hybrid CF* overcomes content based filtering, CF and other recommenders' limitations; but expensive and complex implementation, with requirement to external information which might not be available.

2.4 Utility matrix

Usually, to have a better visualization of the data, we build matrix called utility matrix. Each element of the matrix is a user-item pair with a value that defines the degree of preference for that item by that user. Matrix with most elements missing is said to be sparse, with the RS's goal to fill the empty spaces of the matrix.

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

We can see a utility matrix above for a movie recommendation system, where A, B, C, D are users and HP1, HP2, HP3 are Harry Potter 1,2,3 and TW is Twilight and SW1,2,3 is Star Wars 1,2,3.

3 Implementation

We will be implementing one algorithm to understand the workings of a movie RS. The dataset used here is MovieLens 100K dataset. The language used is Python. The algorithm we are using is a nearest neighbor algorithm called KNN- K nearest neighbor.

3.1 KNN

KNN is an algorithm that predicts a new item based on the proximity to other already existing items. K defines the number of neighbors to consider in the algorithm. Smaller K means noise will have higher influence on the result, a large K will make it computationally expensive and defeats the purpose of KNN. So, K is chosen such that, it is K = n^ (1/2), where n is the number of features. We need to define the distance metric between the items in the dataset to find K closest items.

3.2 Dataset description

This dataset has 100000 ratings by 943 users on 1682 items. Each user has rated at least 20 movies. The order of columns is user id | item id | rating | timestamp. We are storing the data in a Pandas data frame and finding similarity using cosine similarity.

3.3 Result

We have implemented KNN from scratch. We have calculated the Euclidean distance between test movie and all movies in our dataset. The K nearest neighbors for this are shown below.

```
In [33]: print_neighbours(1, 10)

10 Neighbours of Toy Story (1995) (rating: 3.87831858407 ) are:

Aladdin and the King of Thieves (1996) 2.84615384615
Santa Clause, The (1994) 3.09756097561
Home Alone (1990) 3.08759124088
Aladdin (1992) 3.81278538813
Aristocats, The (1970) 3.12962962963
D3: The Mighty Ducks (1996) 2.57894736842
Love Bug, The (1969) 2.78
Wrong Trousers, The (1993) 4.46610169492
Grand Day Out, A (1992) 4.10606060606
Average Rating for all neighbours:  2.9904830749793914
```

```
In [34]: print_neighbours(123, 10)

10 Neighbours of Frighteners, The (1996) (rating: 3.2347826087 ) are:

Young Frankenstein (1974) 3.945
Tales from the Hood (1995) 2.03703703704
Howling, The (1981) 3.02631578947
Braindead (1992) 3.85714285714
Bad Taste (1987) 3.375
April Fool's Day (1986) 2.66666666667
Dracula: Dead and Loving It (1995) 2.28
Cemetery Man (Dellamorte Dellamore) (1994) 2.86956521739
Machine, The (1994) 1.5
Average Rating for all neighbours:  2.555672756771155
```

We can see that the listed movies are similar to the movies Toy story and The Frighteners.

4 Challenges and Future work

4.1 Challenges

- Cold-start is a popular problem in a RS. When new users arrive at a site, we usually have no information about them. In this case predicting a favorable recommendation is very difficult. This is called a cold-start problem. Mostly this is solved by taking a survey from the user when they create the profile. However, hybrid approaches have solved this problem to an extent.
- With increasing users and items, the information required to provide efficient recommendations also increases parallelly. This scalability problem has been overcome by a few complex models.
- Sparsity, where the utility matrix is almost empty. This happens when there is a huge number of users and items and users have rated only a few items. This can be solved by memory based CF. This problem is due to lack of information.
- Privacy plays an important role to build a user's trust of a brand. With the RS trying to capture maximum amount of information, privacy is compromised.
- Not all users display desirable characteristics. Users could be biased towards an item and rate a poor item highly and induce noise into the data. Such information is difficult to remove. Sine the RS learns from this data, the data bias will be reflected on the recommendations too.

4.2 Future work

- Despite, the complex machine learning models, mediocre efficiency and many limitations still exist. To completely overcome them, we need to move to the more advanced Deep Learning domain.
- To build the most efficient RS, we must implement neural networks, which makes predictions based on not one but multiple layers of perceptron.
- The winning Netflix challenge used the Restricted Boltzmann machine, which is used in Netflix too.
- Though RS has come a long way, influencing wide range of applications, there still exist certain limitations and issues. Despite, sparse and noisy data, future RS should be able to overcome these defects and be available to fast growing mobile applications too.

References

1. Xiaoyuan Su and Taghi M. Khoshgoftaar, "A Survey of Collaborative Filtering Techniques," Advances in Artificial Intelligence, vol. 2009, Article ID 421425, 19 pages, 2009. doi:10.1155/2009/421425
2. Daniar Asanov, Algorithms and Methods in Recommender Systems, Berlin Institute of Technology, 2011
3. Ivens Portugal, Paulo Alencar, Donald Cowan, The Use of Machine Learning Algorithms in Recommender Systems: A Systematic Review
4. P. Resnick and H. R. Varian, "Recommender systems," Communications of the ACM, vol. 40, no. 3, pp. 56–58, 1997.
5. A. Ansari, S. Essegaier, and R. Kohli, "Internet recommendation systems," Journal of Marketing Research, vol. 37, no. 3, pp. 363–375, 2000.
6. K. Miyahara and M. J. Pazzani, "Collaborative filtering with the simple Bayesian classifier," in Proceedings of the 6th Pacific Rim International Conference on Artificial Intelligence, pp. 679–689, 2000.
7. L. H. Ungar and D. P. Foster, "Clustering methods for collaborative filtering," in Proceedings of the Workshopon Recommendation Systems, AAAI Press, 1998.
8. G. Shani, D. Heckerman, and R. I. Brafman, "An MDP-based recommender system," Journal of Machine Learning Research, vol. 6, pp. 1265–1295, 2005.
9. P. Resnick and H. R. Varian, "Recommender systems," Commun. ACM, vol. 40, pp. 56–58, March 1997. [Online]. Available: http://doi.acm.org/10.1145/245108.245121
10. B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in Proceedings of the 10th international conference on World Wide Web, ser. WWW '01. New York, NY, USA: ACM, 2001, pp. 285–295.