Movie Recommendation System using KNN

Shusmoy Chowdhury[1], Jayanth Madupalli[1]

Department of Computer Science
Missouri State University

Abstract

Movie recommendation system is to filter and predict only those movies that a corresponding user is most likely to want to watch. In our project we have used content-based filtering approach for movie recommendation. It is a type of recommendation system that tries to guess what a user would like based on the user's activity or prior liking. The recommendation system is created using the K-Nearest-Neighbours (KNN) algorithm, a simple, easy to implement, supervised machine learning algorithm that can be used to solve both classification and regression problems. The algorithm works on the basis of a distance method to compute distance between various data items and return 'k' nearest/similar items. In this project, We have used the KNN algorithm to compute distances between various movies using a set of extracted features from the MovieLens dataset. The end result is the ten nearest movies to the movie that the user has given as an input.
Keywords: KNN, Content-based filtering, Supervised learning

1. Introduction

This project has three components. i.e, KNN Algorithm, MovieLens dataset and the recommendation methodology.

1.1. KNN Algorithm

The K-Nearest Neighbours algorithm is a non-parametric, supervised machine learning algorithm that can solve both classification and regression problems. The algorithm uses proximity to make classifications or predictions about an individual data point as shown in figure 1. This algorithm is a "lazy learning" algorithm, i.e, it stores the training dataset and all the computation occurs when a classification or prediction is being made. It is also referred to as an instance-based or memory-based learning method because of its reliance on memory for storing the training data. As the algorithm uses proximity to classify, it needs a mechanism to determine the distance between the query data point and other data points. These distance metrics provide decision boundaries for the algorithm. There are several distance metrics that can be used with KNN, the most popular are shown in figure 2. We have used the Euclidean distance.

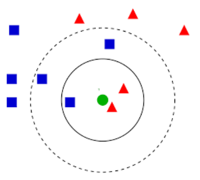


Figure 1: KNN Algorithm

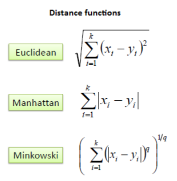

Distance functions
Euclidean
$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$
Manhattan
$\sum_{i=1}^{k}|x_i - y_i|$
Minkowski
$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

Figure 2: Different kind of Distance Metrics

1.2. MovieLens Dataset

Rating data of movies were collected over periods of time and were made available by GroupLens Research from the MovieLens website. For convenience, We have used the small dataset which contains data of 100,000 ratings, 3,600 tag applications applied to 9,000 movies by 600 users. With extra processing power, this project also works with the full dataset containing 27 million ratings, 1.1 million tag applications applied to 58,000 movies by 280,000 users.
The dataset contains the following files: links.csv, movies.csv, ratings.csv, tags.csv. This project mostly utilises data from ratings.csv for extracting the features populartiy and average rating of a specific movie, and data from movies.csv for extracting the feature genre and movie specific metadata.

2. Related Works

The concept of recommendation systems are an active platform for researchers. The system could be built to recommend a book, movies, songs, venues, sites and many more depending on the user needs. There are multiple techniques to build a recommendation system, they are: Content-based filtering, Memory-based filtering, and Model-based filtering.
Most recommendation systems rely on user input, ratings, preferences and similarities. The collected information is analysed for these to produce recommendations. There have been many proposed recommendation systems that used KNN, Neural networks and other deep learning algorithms with either content-based, memory-based or model-based filtering techniques.
R. Singh et al[1] describes an approach which offers generalized recommendations to every user, based on movie popularity and/or genre. They have illustrated the modelling of a movie recommendation system by making the use of content-based filtering in the movie recommendation system. The KNN algorithm is implemented in this model along with the principle of cosine similarity. B.-B. Cui

et al[2] designed and implemented a movie recommendation system prototype combined with the actual needs of movie recommendation through researching of KNN algorithm and collaborative filtering algorithm.

3. Methodology

3.1. Proposed Algorithm/System Design

The proposed algorithm utilises the MovieLens dataset and follows the content-based filtering technique (as shown in fig 3) with the K-Nearest-Neighbours (KNN) algorithm to provide movie recommendations based on a users' input. The algorithm follows the following steps to determine recommendations after the user input.

1. The movie that the user has given as input is our test data point. We first extract the following features from the MovieLens dataset.
    - Popularity (number of ratings)
    - Average Rating
    - Movie Genre
2. These features are extracted for all the movies in the dataset. The distance to determine the proximity is calculated using the Euclidean distance with different variations as discussed in section 3.2.
3. This distance metric is then used to determine the proximity of movies to the given movie (data point).
4. We then determine the ten closest or similar movies and then produce them as our recommendations.

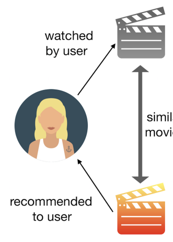


Figure 3: Content-Based Filtering

3.2. Variation of Euclidean Distance

In our project the Movielens dataset consists 19 genres of every movie and ratings by different users for each movie. We have calculated the average ratings and popularity (number users provided ratings for a movie). We have used these 19 genres, average rating and popularity of a movie as features of our KNN. Using these features we have calculated the euclidean distance between the user input and movies in the dataset to build the model.We have used four different variations of Euclidean distance based on these features

3.2.1. Normal Euclidean Distance

We have calculated the euclidean distance among the features and then used summation of all distances to build the model.

3.2.2. Genre Based Euclidean Distance

We have prioritized the genre based features. When we are calculating the distance of genre based feature, we multiply it with some weights (in this case it's 2) and calculated the distance normally for other features. In this case the results are genre-based movie recommendation.

3.2.3. Popularity Based Euclidean Distance

We have prioritized the popularity based features (average rating and popularity). When we are calculating the distance of popularity based feature, we multiply it with some weights (in this case it's 2) and calculated the distance normally for other features. In this case the results are popularity based movie recommendation.

3.2.4. Maximum Feature Euclidean Distance

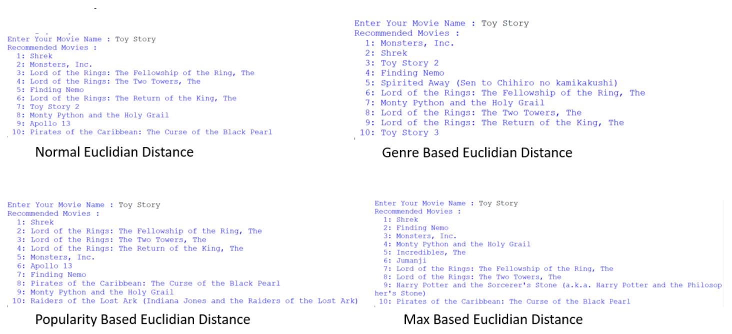
In this case we have used the maximum of genre features euclidean distance and popularity features euclidean distance to build the KNN model.

3.3. Reducing Time Complexity

In KNN machine learning algorithm the time complexity of the algorithm depends on finding the nearest neighbours of test data point. We can use the sorting algorithms to sort the data points according the data points and then take the first K elements from the sorted list as K nearest neighbours. But using the any of the sorting algorithm the best case will be $\mathcal{O}(n \log n)$ and the worst case can be $\mathcal{O}(n^2)$.
But in our project we find out that we actually do not need to sort the list. We can find out the minimum distance data point from the model and that's our first nearest neighbour. We can then exclude that data point from the model and again find out the minimum distance data point from the rest. This way we can find the second nearest neighbour. We can repeat the process for K times and find out the K nearest neighbour. The time complexity to find out the minimum data point is $\mathcal{O}(n)$ and we will run the process K times. So the overall time complexity is $\mathcal{O}(nk)$ which is less than $\mathcal{O}(n \log n)$. In this way we have reduced the time complexity of our algorithm in this project

4. Analysis of Results

In figure 4 we can see the different results using the different variations of euclidean distance. In our model we have used 10 nearest neighbours to show 10 movie recommendations. Our user has given the same input "Toy Story" for all the cases and we have generated 10 movie recommendation from the model. We can see in our results there are 5 movies (Shrek, Monster Inc Finding Nemo, Lord of the ring: The fellowship of the ring, Lord of the ring: The Two towers) that are same in all four variations. So we can say that we may get 50% or more similar results with different variations of euclidean distance. We also we can see that there are 70% (7 movies) similar results in genre based, popularity based and normal euclidean distance. So we can say that prioritizing the features do not effect much in the results.



```
Enter Your Movie Name : Toy Story
Recommended Movies :
  1: Shrek
  2: Monsters, Inc.
  3: Lord of the Rings: The Fellowship of the Ring, The
  4: Lord of the Rings: The Two Towers, The
  5: Finding Nemo
  6: Lord of the Rings: The Return of the King, The
  7: Toy Story 2
  8: Monty Python and the Holy Grail
  9: Apollo 13
 10: Pirates of the Caribbean: The Curse of the Black Pearl
```

```
Enter Your Movie Name : Toy Story
Recommended Movies :
  1: Monsters, Inc.
  2: Shrek
  3: Toy Story 2
  4: Finding Nemo
  5: Spirited Away (Sen to Chihiro no kamikakushi)
  6: Lord of the Rings: The Fellowship of the Ring, The
  7: Monty Python and the Holy Grail
  8: Lord of the Rings: The Two Towers, The
  9: Lord of the Rings: The Return of the King, The
 10: Toy Story 3
```

```
Enter Your Movie Name : Toy Story
Recommended Movies :
  1: Shrek
  2: Lord of the Rings: The Fellowship of the Ring, The
  3: Lord of the Rings: The Two Towers, The
  4: Lord of the Rings: The Return of the King, The
  5: Monsters, Inc.
  6: Apollo 13
  7: Finding Nemo
  8: Pirates of the Caribbean: The Curse of the Black Pearl
  9: Monty Python and the Holy Grail
 10: Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark)
```

```
Enter Your Movie Name : Toy Story
Recommended Movies :
  1: Shrek
  2: Finding Nemo
  3: Monsters, Inc.
  4: Monty Python and the Holy Grail
  5: Incredibles, The
  6: Jumanji
  7: Lord of the Rings: The Fellowship of the Ring, The
  8: Lord of the Rings: The Two Towers, The
  9: Harry Potter and the Sorcerer's Stone (a.k.a. Harry Potter and the Philosop
her's Stone)
 10: Pirates of the Caribbean: The Curse of the Black Pearl
```

Figure 4: Results of different variations of euclidean in KNN

5. Conclusion and Future Works

In this study we have implemented a movie recommendation system with KNN machine learning algorithm. We have used the MovieLens Dataset to train our model and provide recommendation based on the model. We have used different variation of euclidean distance in our project. We have seen that based on these different variations of euclidean distance we get different movie recommendations with a few similar movies. In our future work we can use the intersection result of these different kind of euclidean distance variation result and show them as recommended movies. We can also train our model with other distance metrics such as Manhattan distance or Minkowski distance and compare the results with euclidean distance results.

6. References

[1] R. Singh, S. Maurya, T. Tripathi, T. Narula, G. Srivastav, Movie recommendation system using cosine similarity and knn, International Journal of Engineering and Advanced Technology 9 (2020) 2249–8958. `doi:10.35940/ijeat.E9666.069520`.

[2] B.-B. Cui, Design and implementation of movie recommendation system based on knn collaborative filtering algorithm, in: ITM web of conferences, Vol. 12, EDP Sciences, 2017, p. 04008.