Unsupervised Learning with Netflix

Ayushi Bansal

1 Problem Statement

Given the data set of movies and users along with their ratings, the task at hand is to predict the movie ratings of the user, movie pair given in the test data. This is basically implementation of Netflix Recommendation system that uses Machine Learning techniques to recommend the movies customized according to the history of that user and the movies that he/she has seen.

2 Overview

To solve the given unsupervised learning problem, I have used Matrix Factorization technique. In general, for recommendation system prediction problems, two types of filtering can be used- Content based and Collaborative filtering. But due to the limitation of Content based filtering on large data, Collaborative filtering is a better technique for this problem. The rating matrix (A) obtained (user-id as rows and movie-id as columns) is a sparse matrix and is converted to dense one by using sgd technique (Stochastic Gradient Descent) before applying Matrix Factorization technique. I also tried averaging the rating column wise and subtracting the average from the ratings to make the matrix dense, but that worsened my error. Another technique that is commonly used is spectral clustering, wherein data points are treated as nodes of graph and they are mapped to low-dimension space and then clusters are formed. I did not use spectral clustering mainly because of two reasons. It doesn't work efficiently on large data sets and also intial centroids chosen for k-means clustering that is applied in last step of spectral clustering may give varied results.

To achieve better accuracy of results, I have scrapped detailed IMDb data for the movies given in the train data set. Then, I have clustered similar movies based on this data set using k-means clustering technique.Then, I have merged the predicted matrix after Matrix Decomposition and the results from k-means cluster (for similar types of movies) using weighted mean to get the better results. The final prediction matrix will account into consideration the user choice, movie rating trends and also various other movie content related parameters, but user-rating being the primary feature.

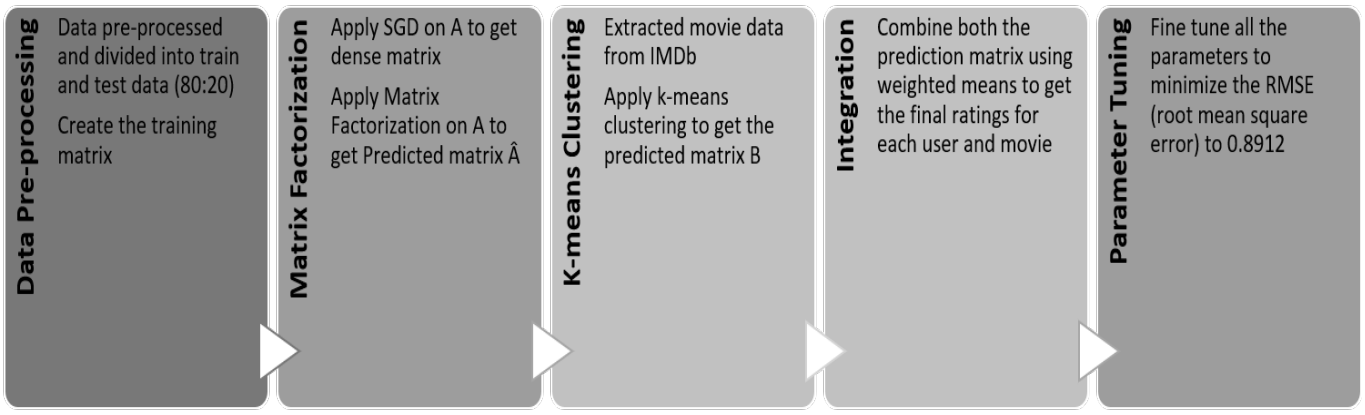
3 Model

3.1 Data Pre-processing

Data Sets used: Train data (given), Movie-titles with movie-id, Movie metadata (Scrapped from IMDb for movies in the training set based on movie-titles to better learn the movie features)
The metadata only has details of 8000 movies (subset of train data movie). It has following parameters for each movie: genre, language of movie, movie plot description, IMDb rating. The training data has 5905 unique user-ids and 16142 unique movie-ids.

3.2 Approach Used

First of all, train data is split into 80:20 ratio for testing purpose. Then, the training data (80 % of original data) is converted into user, movie matrix - A. This being sparse matrix is converted into dense using sgd technique. Then, I applied Matrix Decomposition on it to get the final predicted matrix. This, new matrix $\hat{A}$ consists of predicted ratings. And error (E) is calculated for this predicted matrix.
In a separate python code, the movie metadata is taken and k-means clustering algorithm is applied. I have created around 400 clusters of movies based on all the features. These clusters are stored in form of dictionary with key as cluster number and value as the movie-ids.
In order to combine these two results, I have tried various methods. Firstly, for each user in the $\hat{A}$ matrix, I have iterated for its movies in each cluster and the ratings of all the movies in same cluster is averaged and these ratings are in turn



replaced by the average. But the error (E1) thus calculated is worse than the error E. Secondly, I tried constructing a matrix $\hat{B}$ of same dimension as $\hat{A}$ after k-means on the movie metadata. For this, I averaged the ratings for all the movies in the same cluster for which we already have rating in the original matrix A. And this average value is put for the movies for which we don't have ratings. Then weighted mean of $\hat{A}$ and $\hat{B}$ is taken to get the final predicted matrix $\hat{C}$. The weights are changed to get the best results, that is the error (RMSE).

3.3 Error Analysis

The metric used to analyze the result, that is to calculate error for the prediction matrix generated is Root Mean Square Error. I also observed various details in the data provided. In test data, there are approximately 500 unique movie-ids that are not in the train data. Also, there are many different movie-id for same movie title when filtered using the train and movie-title data.

4 Parameters and Results

Following are the various parameters selected by me to get the most optimized result, that is minimizing the RMSE of test data (20% of train data)

K	α	β	Iterations	Error
4	0.005	0.001	50	0.8912
5	0.004	0.001	45	0.8918
3	0.006	0.001	50	0.8944
3	0.007	0.001	60	0.8995

k = number of latent dimensions
α learning rate
β regularization parameter

Parameters for k-means clustering
n_clusters=450, init='k-means++', max_iter=80, n_init=1000, init_size=1000, batch_size=500
init selects initial cluster centers for k-mean clustering in a smart way to speed up convergence. max_iter is the maximum number of iterations. n_init are the number of random initializations that are tried. init_size is the number of samples to randomly sample for speeding up the initialization.