CODER

SHUBHAM PACHORI

TANMAY KHARE

SWATI CHELAWAT

DEEPENDRA BUNDELA

INTRODUCTION

In today's web applications space, each user wants a personalized shopping treatment where the website is able to learn about his/her taste and recommend stuff accordingly. For example, if you go to Amazon website and purchase a book, it will show you other related books that you would likely to be interested in. Similarly, Facebook recommender system is able to recommend you with new friends under its "people whom you may know" section. Basically, there are two types of recommender system algorithms that are widely used in these industries:

1. Content Based: It examines the properties of the items and recommends similar items to a user based on his past buy/search history
2. Collaborative Based: It recommends items based on similarity between users and/or items.

Similarly, in this project, we have used Collaborative filtering approach for following scenarios:-

a. Recommending movies to users
b. Predicting what rating will a user give to a movie

DATA PREPARATION:

In recommendation systems, we need the data to be present in a matrix. In our data we have following data:

1. Y is a matrix where each row represents a Movie and each column represents a User. Each cell of our matrix contains the ratings given by User to a Movie.
2. R is a binary matrix which contains 1 for a movie that is rated and 0 otherwise.

3. X and Weights are our initial precomputed values, where X represents the features of the movies such as action, comedy etc. and Weights represents user's taste. These assumed values are created to use them in our collaborative filtering algorithm later.

PROBLEM STATEMENT:

PREDICT MOVIE RATINGS AND RECOMMEND MOVIES

We have considered a case where a company sells online movies to its users. For each movie, you also allow them to rate movies using 1-5 ratings where 1 means that a user does not liked the movie and 5 shows that he/she liked the movie very much.

Now let's try to understand this problem by taking a very simple example shown below:

Movies	**Dave**	**Tom**	**John**	**Harry**
Titanic	5	5	1	2
Notebook	4	5	1	?
Mirror	1	2	5	5
Conjuring	?	1	5	4
Fast and Furious	4	?	?	0

In first glance, above matrix helps us to understand following things:

- Dave and Tom are more drawn towards watching romantic/action genre
- John and Harry likes to watch horror movies
- Some of the movies are not seen by these users and they are denoted by a? in the matrix

Now, let's understand that if we want to recommend movies to above users, how we should do that.

Let's introduce some notation:

- N_u – Num_Users
- N_m – Num_Movies
- Ri, j) - 1 if user j has rated movie i
- Y(i,j) - rating given by user j to move i (defined only if R(i,j) = 1)
- So for this example $N_u = 4$, $N_m = 5$

Broadly speaking, to recommend movies, we need to do the following two things:

- Given R(i,j) and Y(i,j) - go through and try and predict missing values (?s)
- Come up with a learning algorithm that can fill in these missing values

CONTENT BASED RECOMMENDATION - OVERVIEW

Using our example above, how do we predict the missing ratings for users?

For each movie we have a feature which measure degree to which each film is a

- Romance (x1)
- Action (x2)

Movies	**Dave**	**Tom**	**John**	**Harry**	**X_1 (Romance)**	**X_2 (Action)**
Titanic	5	5	1	2	0.9	0
Notebook	4	5	1	?	1	0.01
Mirror	1	2	5	5	0.1	0.99
Conjuring	?	1	5	4	0.1	1
Fast and Furious	4	?	?	0	0	0.9

Two more notations are added:

- X^1 : a feature (i) vector containing values 0.9 and 0 for the first movie Titanic. Similarly X^2, X^3 etc. can be formed.
- Weight: a user (j) vector containing some initial random values.
- Mj: Number of movies rated by the user (j)

We can predict the ratings for our movies by taking an inner product of transpose of Weights and Features.

Ratings = $(\text{Weight}^j)^T X^i$

Here we are applying a linear regression method for each user so that we can determine a future rating based on their interest in romance and action genre.

We can use the below linear regression equation to learn Weights.

$$\min_{\theta^{(1)},\ldots,\theta^{(n_u)}} \frac{1}{2}\sum_{j=1}^{n_u}\sum_{i:r(i,j)=1}\left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}(\theta_k^{(j)})^2$$

This approach is called content-based approach because we assume we have features regarding the content which will help us identify things that make them appealing to a user

However, often such features are not available and hence we use Collaborative filtering approach.

COLLABORATIVE FILTERING – OVERVIEW

Collaborative filtering systems have many forms, but many common systems can be reduced to two steps:

- Look for users who share the same rating patterns with the active user (the user whom the prediction is for).
- Use the ratings from those like-minded users found in step 1 to calculate a prediction for the active user

The collaborative filtering algorithm does feature learning i.e. it can learn for features itself. In reality, this can be a tedious job since we would want more features than just two.

So - let's change the problem and pretend we have a data set where we don't know any of the features associated with the films.

Movies	**Dave**	**Tom**	**John**	**Harry**	**X_1 (Romance)**	**X_2 (Action)**
Titanic	5	5	1	2	?	?
Notebook	4	5	1	?	?	?
Mirror	1	2	5	5	?	?
Conjuring	?	1	5	4	?	?
Fast and Furious	4	?	?	0	?	?

Now let's make a different assumption

We've polled each user and found out how much each user likes Romantic films and Action films which has generated the following parameter set

Dave and Tom like romance but hate action. John and Harry like action but hate romance

If we can get these parameters from the users we can infer the missing values from our table

Let's look at "Titanic". Dave and Tom loved it but John and Harry hated it.

Based on the factor Dave and Tom liked "Titanic" and John and Harry hated it we may be able to correctly conclude that "Titanic" is a romantic film

This is a bit of a simplification in terms of the math, but what we're really asking is what feature vector should X^1 be so that $(Weight1)^T X^1$ is about 5, $(Weight2)^T X^1$ is about 5, $(Weight3)^T X^1$ is about 0, $(Weight4)^T X^1$ is about 0

From this we can guess that X^1 may be [1 1 0]

Using that same approach we should then be able to determine the remaining feature vectors for the other films. From the collaborative filtering point of view we have to minimize our user vector and weight vector to converge at a global minimum.

Minimizing $x^{(1)}, \ldots, x^{(n_m)}$ and $\theta^{(1)}, \ldots, \theta^{(n_u)}$ simultaneously:

$$J(x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)}) = \frac{1}{2} \sum_{(i,j):r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

$$\min_{\substack{x^{(1)}, \ldots, x^{(n_m)} \\ \theta^{(1)}, \ldots, \theta^{(n_u)}}} J(x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)})$$

So, for a particular movie we are summing over squared difference between actual rating of that user and predicted rating of that user. To calculate this minimized value for every user we just summed over every user vector for every given movie in our dataset.

Recommending new movies to a user

Collaborative filtering algorithm helps us to learn the features of the movies. Now lets say we have learned the following features: Romance, Action, Comedy, etc. The features learned may not always be human interpretable but they allow a good way to calculate movie similarity. If we have two movies we want to minimize $||X_i - X_j||$ i.e. the distance between those two movies.

MEAN NORMALIZATION:

Let's consider a case when a user has not rated any movies and we still want to recommend some movies. This is a cold start problem and can be reduced to some extent by mean normalization.

Movies	Dave	Tom	John	Harry	New User
Titanic	5	5	1	2	?
Notebook	4	5	1	?	?
Mirror	1	2	5	5	?
Conjuring	?	1	5	4	?

```
normalize_ratings = function(Y)
{
  mean_per_movie = as.numeric(mean_per_movie)
  Y_temp1 = replace(Y,Y==0,NA)
  temp = Y_temp1-mean_per_movie
  temp[is.na(temp)]=0
  Ynorm = as.matrix(temp)
  return(Ynorm)
}
```

Our above function in R returns the normalized value (Ynorm) by subtracting the actual ratings data with the mean value of all users for a movie.

```
params_final = optim(params,calcCost,calcGradient,method = 'L-BFGS-B',lower = 0,upper = 1,
      params,Ynorm,num_users, num_movies,num_features,lambda)
```

```
my_prediction = P[,944]+ mean_per_movie
```

Next, we have used the normalized value in our cost minimization function. And added the mean value to the final ratings prediction.

CHALLENGES AND FUTURE WORK:

COLD START PROBLEM

This is a typical problem faced by recommendation systems where the model fails to recommend movies to a new user who has not rated any movies or a newly introduced item. One solution to this problem can be to use social networks information in order to fill the gap existing in cold-start problem and find similarities between users. We can utilize User database, personal information about the user, information from social media and based on that we recommended the movies.

MATRIX SPARSITY

Normally, there are always less users who rate the movies and hence we do not get enough ratings data to make more intelligent predictions. Measures should be taken to collect as much ratings data to get rid of this problem.

HYBRID MODELS

The above problems can also possibly be solved by using a hybrid approach. This type of algorithm used mixture of content based and collaborative algorithms to utilize the important properties of both the approaches into one model and make better recommendations.