华南理工大学

South China University of Technology

The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
WuKun,SunXingxing

Supervisor:
Qingyao Wu

Student ID：
201721045480(wukun),
201721045350(sunxingxing)

Grade:
Postgraduate

December 28, 2017

Recommender System Based on Matrix Decomposition

Abstract Recommender systems have received great attention in recent years owing to their ability in mitigating the information overload problem. In the light of this, recommender systems have been widely applied in multiple domains such as music, image, restaurant, and mobile app. Existing studies on recommender systems mainly focused on recommending individual and independent items to users. Good personalizedrecommendations can add another dimension to the userexperience, e-commerce leaders like Amazon.com and Netflix have made recommender systems a salient part of their websites.

Keywords: Recommender System, Matrix decompostion,

I. Introduction

Broadly speaking, recommender systems are based on one of two strategies. The content filtering approach creates a profile for each user or product to characterize its nature. For example, a movie profile could include attributes regarding its themes, the participating actors,.The profiles allow programsto associate users with matching products. Of course,content-based strategies require gathering external information that might not be available or easy to collect.

II. Methods and Theory

The two primary areas of collaborative filtering are the neighborhood methods and latent factor models. Neighbor-hood methods are centered on computing the relationships between items or, alternatively, between users. The item oriented approach evaluates a user's preference for an item based on ratings of "neighboring" items by the same user. A product's neighbors are other products that tend to get similar ratings when rated by the same user. For example, consider the movie 2012. Its neighbors might include war movies, servival movies, among others. To predict a particular user's rating for Following the wind , we would look for the movie's nearest neighbors that this user actually rated.

Latent factor models are an alternative approach that tries to explain the ratings by characterizing both items and users on, say 20 to 100 factors inferred from the ratings patterns. In a sense, such factors comprise a computerized alternative to the aforementioned human-created song. For movies, the discovered factors might measure obvious dimensions such as comedy versus drama, amount of action or some others.

Here we use alternating least square formulation to sovle recommender systems problems.The detailed content is followed.

$$Q_{ui} = \begin{cases} r & \text{if user u rate item i} \\ 0 & \text{if user u did not rate item i} \end{cases}$$

where r is what rating values can be. If we have mm users and nn items, then we want to learn a matrix of factors which represent movies. That is, the factor vector for each movie and that would be how we represent the movie in the feature space. Note that, we do not have any knowledge of the category of the movie at this point. We also want to learn a factor vector for each user in a similar way how we represent the movie.

Note that, we do not have any knowledge of the category of the movie at this point. We also want to learn a factor vector for each user in a similar way how we represent the movie. Factor matrix for movies $Y \in Rfxn$ and factor matrix for users $X \in Rmxf$ each user is a row vector. However, we have two unknown variables. Therefore, we will adopt an alternating least squares approach with regularization. By doing so, we first estimate YY using XX and estimate XX by using YY. After enough number of iterations, we are aiming to reach a convergence point where either the matrices XX and YY are no longer changing or the change is quite small. However, there is a small problem in the data. We have neither user full data nor full items data, this is also why we are trying to build the recommendation engine in the first place. Therefore, we may want to penalize the movies that do not have ratings in the update rule. By doing so, we will depend on only the movies that have ratings from the users and do not make any assumption around the movies that are not rated in the recommendation. Let's call this weight matrix W_{ui} such:

$$w_{ui} = \begin{cases} 0 & \text{if } q_{ui} = 0 \\ 1 & \text{else} \end{cases}$$

Then, cost functions that we are trying to minimize is in the following:

$$J(x_u) = (q_u - x_u Y) W_u (q_u - x_u Y)^T + \lambda x_u x_u^T$$

$$J(y_i) = (q_i - X y_i) W_i (q_i - X y_i)^T + \lambda y_i y_i^T$$

III. Experiment

Our dataset consists 10,000 comments from 943 users out of 1682 movies. At least, each user comment 20 videos. Users and movies are numbered consecutively from number 1 respectively. The data is sorted randomly.

Our procuduresare are follows.

- Read the data set and divide it. Populate the original scoring matrix against the raw data, and fill 0 for null values.
- Initialize the user factor matrix and the item (movie) factor matrix , where is the number of potential features.
- Determine the loss function and the hyperparameter learning rate and the penalty factor .
- Use alternate least squares optimization method to decompose the sparse user score matrix, get the user factor matrix and item (movie) factor matrix:
  a. With fixd item factor matrix, find the loss partial derivative of each row (column) of the user factor matrices, ask the partial derivative to be zero and update the user factor matrices.
  b. With fixd user factor matrix, find the loss partial derivative of each row (column) of the item factor matrices, ask the partial derivative to be zero and update the item
  c. Calculate the on the validation set, comparing with the of the previous iteration to determine if it has converged.
- Repeat step 4. several times, get a satisfactory user factor matrix and an item factor matrix , Draw a curve with varying iterations.
- The final score prediction matrix is obtained by multiplying the user factor matrix and the transpose of the item factor matrix .

Fig. 1 The steps of ALS

Resutl:

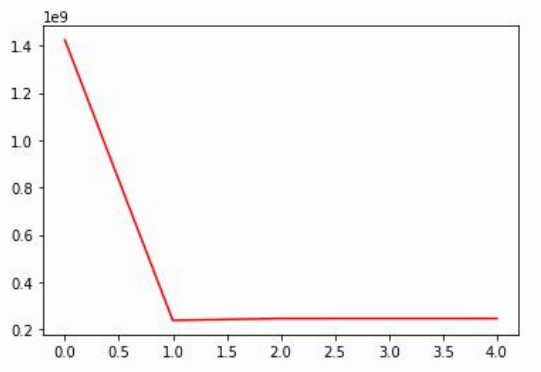

Fig. 2 loss value and iteration

From the figure 2,we find that it is so quicky find the best parameters .By fixing a single variable,we optimize another univariate.After one iterantion,we get the matrix U, V corresponding to the optimal solution.

IV. CONCLUSION

In the more general sense, collaborative filtering is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources, etc.

Our team have written a class that trains a matrix factorization model using ALS. In an attempt to finish the recommender system, the code is relegated to this GitHub gist.The class also has the functionality to calculate a learning curve where we plot our optimization metric against the number of iterations of our training algorithm.