Name: Haofan Wu

School ID: A53237402

Kaggle Account: User name: haw013   Display name: Haofan

CSE 258 Assignment1 Report

1. Introduction of Datasets

Training data is in train.json.gz, in which it has 200,000 reviews to be used for training. The fields that I used are: businessID, userID and rating, which represent the ID of business, the ID of the reviewer and the star rating of the user's review separately.

For testing, pairs_Visit and pairs_Rating is two files that I need to predict.

2. Task1 Visit prediction

2.1. Introduction:

Predict given a (user,business) pair from 'pairs Visit.txt' whether the user visited the business.

2.2. Naive Model:

Using the single feature in the data like popularity and rating to predict the visiting.

For popularity: Predict true if the business is 'popular', which means that it appears more than half times in the total purchases.

To realize it, I calculate the total number of purchases first, then count each business's occurrences and compare these with half of the total purchases. If it is greater than the half of total purchases, I assign it to 1 in the prediction file.

The result for this simple model is not good in kaggle, the accuracy is only 0.6225.

2.3. KNN Algorithm:

Since I need to find whether a user visit a business, then I think using KNN to predict the process is reasonable. Since I can using the index of the users and the index of the business to get the distance between each user. To be clear, for two users, I denote distance is the number of business that they both visited. Then I can get a dictionary, that each user has a list of distance of other users. I choose the k-th greatest distance to be a user's neighbors.

In the process of predicting visit: for each pair (user, business), I need to find if it is in the user's or its neighbor's business lists. If it is, I predict that this user has visited this business. If it is not in these lists, I predict false.

For those data that doesn't appear on the training data, I don't forget my former simple model, for those 'business' in the training data and 'user' doesn't in the data, I use the popularity model in part 2.1 to predict them.

2.4. Change k and get better results

The result is pretty good comparing with the former method. I chose k = 6 to test KNN and got the accuracy 0.7660 in kaggle.

Then I began to increase the value of k in order to find the best point in visit prediction. I found that the accuracy will increase with the k increase, and choosing k = 00 help me get 0.8705 in kaggle. And I keep increase k to 500 and get 0.8732.

The final k I choose is 5000, which got the accuracy of 0.8758.

2.5. Conclusion

To solve this problem, I tried some different algorithms such as the popularity algorithm and the KNN algorithm. And I find that KNN can do much better in predicting whether a user has visited a business. Finally, I combined these two methods and found that it will improve the accuracy (For example, only using 1300-KNN got accuracy of 0.8753 but combine it with popularity helped me get 0.8754).

But KNN algorithm will not always increase the accuracy dramatically with the increase of k. When k reached a big value (such as above 1000), the increase of k will only have little affect on the accuracy. Although I can't test the maximum accuracy with KNN because of the huge cost of time, I can analyze from my dataset from k = 6 to k = 5000 and come to a conclusion that the max accuracy in this problem by only use KNN (or combine one simple method) will not greater than 0.88. So there will be a better algorithm to analyze this problem and I will try some different algorithm after class.

3. Task2 Rating prediction

3.1. Introduction:

Predict people's star ratings as accurately as possible for a (user,item) pairs.

3.2. latent factor model

3.2.1 Simple model

To test the latent factor model, I firstly use the simplest model:

$$f(u,i) = \alpha$$

By calculating the global average rating of a (user, item) pair. It is obvious that it will perform poorly.

3.2.2 More complex model

Since we can consider each user and item to predict the rating, so I got:

$$f(u,i) = \alpha + \beta_u + \beta_i$$

In coding procedure, we need to calculate beta_user and beta_item recursively until convergence:

$$\alpha = \frac{\sum_{u,i \in \text{train}} (R_{u,i} - (\beta_u + \beta_i))}{N_{\text{train}}}$$

$$\beta_u = \frac{\sum_{i \in I_u} R_{u,i} - (\alpha + \beta_i)}{\lambda + |I_u|}$$

$$\beta_i = \frac{\sum_{u \in U_i} R_{u,i} - (\alpha + \beta_u)}{\lambda + |U_i|}$$

Then I change different lambda to fit the best beta and alpha, and the best lambda is 4.2, which will got RMSE of 0.7835.

3.2.3 Add gamma:

The final improvement of latent factor models is adding gamma_user and gamma_item :

$$f(u,i) = \alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i$$

Then the aim is to minimize this equation and do the similar process in 3.2.2:

$$\arg\min_{\alpha,\beta,\gamma} \sum_{u,i}(\alpha+\beta_u+\beta_i+\gamma_u\cdot\gamma_i-R_{u,i})^2+\lambda\left[\sum_u \beta_u^2+\sum_i \beta_i^2+\sum_i \|\gamma_i\|_2^2+\sum_u \|\gamma_u\|_2^2\right]$$

The coding is similar, and we can use gradient descent to update the alpha, beta and gamma.

After changing the lambda and the iteration times i, I got some better output in Kaggle and found that lambda = 4.8 , i = 300 is the best and can got the RMSE of 0.7569.

3.2.4 Tricky way:

I happened find a different latent model in web and that is to chose different lambda in alpha + beta_user + beta_item. And I call two lambda lam_user and lam_item. Then I change different lam_user and lam_item in 3.2.2 and crazily it got better RMSE than 3.2.3, and the best RMSE is 0.7567 when lam_user =3.4 and lam_item = 4.9.

3.3. Conclusion

To solve this problem, I tried different latent factor models and tried many different parameters, which make me understand all kinds of latent factor models well. The final answers I submitted are the output of ‘adding gamma’ and the output of ‘two lambda’.