# Recommendation System Using Yelp Data

CS 229 Machine Learning Jia Le Xu, Yingran Xu

## 1 Introduction

Yelp Dataset Challenge provides a large number of user, business and review data which can be used for a variety of machine learning applications. Our project is aiming to create a friend and business recommendation system using yelp data. We plan to find hidden correlations among users, and then recommend new friends to users with similar interests. The second motivation is to identify what these interests are. We want to know what business the user favors more, so that we implemented the business recommendation system, which predicts the users rating on a certain business using users past experience (ratings on similar business) and other peoples rating on this business. The application of this project can extend the use of yelp to a social networking level, which allows users to find new friends and experience certain business together.

## 2 Related Work

In this project, we use K-Means to group users with similar interests. Centroid initialization in K-Means algorithm can affect the final clustering performance. There are several ways proposed for centroid initialization. Arai and Barakbah [1] used a hierarchical method to run k-means a few times, and applying hierarchical clustering algorithm to find the best centroids in the set. This method generates good results for high dimensional dataset, but it takes longer to run. Another method called K-Means++ proposed by Arthur and Vassilvitskii [2] chooses initial centroids uniformly randomly, and choose the subsequent centroid with weighted probability proportional to the squared distance from its closest existing centroid. This method improves speed and accuracy of K-Means algorithm, and we use this initialization scheme in our project. On the other hand, matrix factorization is commonly used for recommendation system and finding latent relations between users and items. It is easy to implement and does not require too much pre-processing of data[3].

# 3 Data Preprocessing

We choose to only analyze the data on one type of business (restaurant) in our project, but the application may extend to larger scopes. Yelp has grouped restaurants into many different categories. In order to have more data to show similarity among users, we choose to use the categories (not individual restaurants) to group the users in the first part of the project. Some users only have a small number of reviews (< 10) in the dataset, which is not enough for determining a users preference of certain type of restaurants. So we filter out the users with number of reviews less than 20, and left with 6770 data. Later, we also try to

use the user data with number of reviews greater than 50, 100, 150 and 200 to compare the results. Then we split the data into 70% training set and 30% test set to use for hold-out cross validation later.

## 4 Algorithms & Result Discussion

### 4.1 K-Means

The baseline for the first part of the project is to use one cluster with centroid of the total averaged rating for all users of all businesses in the training dataset. In this baseline, we used all users with review count greater than 20, and the averaged rating is found to be 3.7. K-Means algorithm basically takes each users input feature vectors (star rating and times of visit), and groups the users into clusters based on the Euclidean distance of each users feature vectors to the clusters centroids.

### 4.1.1 Methodology

To implement K-Means algorithm, first we need to determine the number of clusters to use. We run the algorithm with different values of K, and then plot their distortion measurements. The graph below shows the plot for 1000 training data. We choose value K to be the knee of the graph, which is 20 in this case.

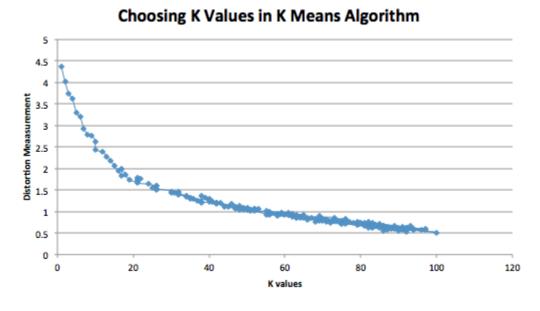


Figure 1: Finding number of clusters in K-Means

For K-Means initialization, we first initialize all centroids randomly, and then we use K-Means++ initialization, which is described earlier.

To evaluate the algorithm, we use RMSE on star rating as our error measurements. After clustering the users, we calculated the expected rating for each category in every group of users using training data. Then for evaluation, we assign each test data (a new user) to the cluster with minimum Euclidean distance, and calculate the RMSE of its true rating on each category with the expected rating calculated in each cluster.

$$Cluster_{testdata_i} = argmin_k(\sqrt{\Sigma_{j \in categories}(w_{test_j} * \phi_{test_j} - c_{kj})^2}), \forall k$$

$$RMSE = \sqrt{\frac{\Sigma_{testdata_i}\Sigma_{j \in categories}(E[rating_{cluster_i}] - rating_{testdata_i})^2}{number of ratings}}$$

For each group of training data (reviews; 20, 50, 100, etc), we train the weight vector by giving different weight values for each feature (w1 = weight for star rating; w2 = weight for times of visit) and calculate the RMSE to find the weights that gives the minimum error in the test dataset.

#### 4.1.2 Result Discussion

The results are shown in the following table:

Table 1. Results comparison using K-Means

Table 1. Testards comparison asing 11 Mounts										
Data Group	Baseline(3.7)	Reviews >	Reviews >	Reviews >	Reviews >					
		20	50	100	150					
Training data	4739	4739	1295	380	158					
Clusters	1	45	25	8	4					
Random Initialization										
$RMSE_{train}$	0.911	0.757	0.671	0.620	0.539					
$RMSE_{test}$	0.917	0.986	0.889	0.815	0.773					
K-Means++ Initialization										
$RMSE_{train}$	0.911	0.652	0.581	0.592	0.538					
$RMSE_{test}$	0.917	0.957	0.885	0.800	0.772					

Comparing to baseline algorithm, K-Means makes some improvements when we focus on the users with large number of reviews. Comparing two centroid initialization schemes,  $RMSE_{test}$  using K-Means++ is slightly smaller than using random initialization. But  $RMSE_{train}$  using K-Means++ is generally smaller than using random initialization, especially when the training dataset is larger and more random.

However, considering the highly skewed nature of the dataset (around 75% of reviews have rating of 4 or 5), a RMSE about 0.8 is not very good. So we decide to explore if users in the same area tend to have more similar interests. We used all users with review count > 20 (since if we use larger review count groups, the data will be too few), and run K-Means for each state:

Table 2. K-Means Results for Each State

State	NV	PA	AZ	IL	NC	WI
Training data	1570	228	2013	34	334	155
$RMSE_{train}$	0.781	0.676	0.696	0.523	0.590	0.626
$RMSE_{test}$	0.928	1.029	0.922	1.365	0.933	0.991

From the table above, we find that the result considering each state separately is not improved. This is mainly caused by lack of data. IL only has 34 training data, which is likely to produce a high variance result.

The high variance result is also shown in the first case where we did not split data into states. This may because the features we used are too simple. Also, due to the unsupervised nature of this problem, it is hard to accurately measure how good the result is. Here, we are only considering the star rating as our measurement, however, the users reviews may also have an impact on the clustering. For example, some users give high ratings because of the food, while others may care more about the ambience and service.

### 4.2 Matrix Factorization

We explore another collaborative filtering algorithm called matrix factorization. The basic idea of it is to decompose a matrix contain user-business ratings into two matrices. These two matrices are related via latent features, one relates users's preference to features, the other one relates business to features.

### 4.2.1 Methodology

$$R \approx PQ \Rightarrow \hat{r}_{ub} = \sum_{k=1}^{K} p_{uk} q_{kb}$$

where  $\hat{r}_{ub}$  denotes how the n-th user would rate the m-th item in the model.  $R \in \mathbb{R}^{(N \times M)}$ ,  $P \in \mathbb{R}^{(N \times K)}$ ,  $Q \in \mathbb{R}^{(K \times M)}$ , N: # of users, M: # of business, K: # of features. We then use cost function:

$$e_{ub} = \frac{1}{2}((r_{ub} - \hat{r}_{ub})^2 + \lambda(p_u^T p_u + q_b^T q_b))$$

with additional regularization term to prevent overfitting.

We then use gradient descent to find optimal  $(P^*, Q^*) = \arg\min_{(P,Q)} RMSE$ :

$$p_{uk} := p_{uk} + \alpha * (2e_{ub}q_{kb} - \lambda(p_{uk}))$$

Next we use improved regularized matrix factorization (add bias for user and businesses):

$$\hat{r}_{ub} = c_u + d_b + \sum_{k=1}^{K} p_{uk} q_{kb}$$

which trained as  $c_u := c_u + \alpha * (\hat{r}_{ub} - \beta * (c_u + d_b - global mean))$ 

#### 4.2.2 Result Discussion

We run tests on two states (AZ and NV) since they contain more users and business with higher number of reviews. For a given user-business matrix, we randomly set 30% of the rating to 0, which means not rated, and use it as our testing set. The rest 70% becomes the training set. We then experiment with different number of features and plot the RMSE of testing set. "NV-200-40" is the dataset contains user with review count larger than 200 and

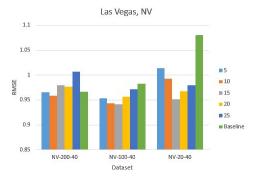


Figure 2: RMSE with Different Number of Features

businesses with review count larger than 40. Therefore, "NV-20-40" is the largest dataset (about 80,000 reviews). We can observe an increase of performance compared against the baseline. Moreover, for larger dataset, the optimal K value (number of features) also increases as expected.

## 5 Conclusion & Future Work

K-Means algorithm using user data with larger number of reviews shows a decent grouping result. However, the feature selection and evaluation methods can be improved. If we are taking this project further in the future, we would like to add more features. For example, we can find the major characteristics of each business, and the key words in user reviews, and then relate them to the users rating. We can utilize the features discovered by matrix factorization, and use it to improve K-Means or facilitate developing some supervised learning. Furthermore, we can also explore other collaborative filtering algorithms or neural networks to find correlations among the features.

# 6 References

- [1] K. Arai, A. R. Barakbah. "Hierarchical K-means: an algorithm for centroids initialization for K-means". Reports of the Faculty of Science and Engineering Saga University, vol. 36, No.1, 2007, pp. 25-31.
- [2] Arthur, D. and Vassilvitskii, S. (2007). "k-means++: the advantages of careful seeding". Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics Philadelphia, PA, USA.
- [3] Arkadiusz Paterek. "Improving regularized singular value decomposition for collaborative filtering". KDDCup.07 August 12, 2007.