Predicting Yelp Rating Based on Social Network

Tiecheng Su, Yanfu Zhang
ECE Department, University of Rochester

I. Introduction

The goal of our project is to predict the users' reviews based on existing data. We interpret the yelp dataset as a sample from the whole table of user-business pairs, in which case the prediction task is converted into finding the best estimation of the whole table. The first section of this report describes a common collaborative filtering recommendation method, as well as our revise; the second section shows the detailed implementation and an analysis of the results; the last section briefly depicts our future plan.

II. A Collaborative Filtering Algorithm

i. A common Collaborative Filtering Algorithm

The rating prediction can be viewed as a function of certain user and business. In this algorithm, each user is allocated a vector, x, and each business, θ. Roughly speaking, the business vector denotes the categories it belongs to, while the user vector denotes one's preferences to different categories. The rating function is a measure of how closely a business catering to a user. A natural choice is:

$$\hat{y}^{(i,j)} = (\theta^{(j)})^T x^i \tag{1}$$

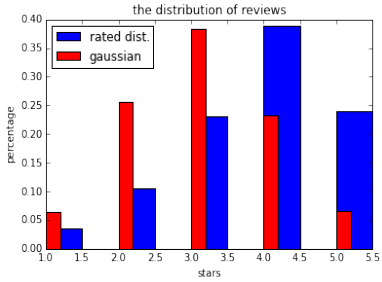
here $\hat{y}^{(i,j)}$ is the estimated rating of user i to business j; x_i is the parameter of the users, θ_j is the parameter of businesses and $r(i,j)$ represents that whether user i has rated business j or not. The loss function of this model is:

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

the last 2 are regularization terms. Using stochastic gradient descent method an optimized solution can be found.

ii. A revise on the original algorithm

In practice, we can only get a small sample of the whole loss function, namely those reviews actually occur in the yelp dataset. In another word, people tend to visit those restaurant with higher rates. If this sample was uniformly chosen from the entire table, the solution would be a good estimation in the meaning of minimize the total variance. However, it is not always the case.



From above figure it is clear that high ratings are more common. One explanation is that people tends to rate high. Another, which our main work is based on, suggests that people only visit and rate a business when they expect some good experiences. That is to say, the sample we observe is biased by one's rating. We assume that as a whole the rating obey Gaussian distribution. We use $q(i,j)$ to denote these absences of low ratings. This $q(i,j)$ should satisfy below requirements:
1. it should relate to both x_i and θ_j, or $r(i,j)$.
2. it should be monotonous increasing.
3. it should agree with the samples in the sense of max likelihood.
The rating data only ranges from 1 to 5, which gives us a great freedom in choosing such $q(i,j)$. Our choice is exponential function, mainly because it's extremely easy to handle in MLE process. To eliminate the sampling bias in the dataset, we gives every data point a weight in the loss function. It has the same result with re-sample the dataset, from expectation's perspective. So the new loss function is defined as:

$$q(i,j) = exp(-a\hat{y}^{(i,j)}) \tag{3}$$

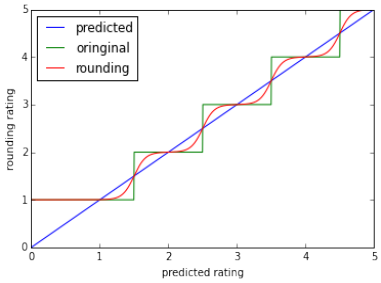
$$J(x^{(1)},\dots,x^{(n_u)},\theta^{(1)},\dots\theta^{(n_b)}) = \frac{1}{2}\sum_{(i,j):r(i,j)=1)}(\hat{y}^{(i,j)} - y^{(i,j)})^2 exp(-a\hat{y}^{(i,j)}) + \frac{\lambda}{2}\sum_{i=1}^{n_u}\sum_{k=1}^{n}(x_k^{(i)})^2 + \frac{\lambda}{2}\sum_{j=1}^{n_b}\sum_{k=1}^{n}(\theta_k^{(j)})^2 \tag{4}$$

SGD can still be used in the computing of x and θ in the revised algorithm. The Hessian for the new loss function should be the same with the total loss, which is convex, if q is accurate; if not, the Hessian will introduce a coefficient. In the worst case, assuming the $\hat{a}$ is our estimated number, then the Hessian is $(2-(2+2(a-\hat{a}))\sum_{(i,j):r(i,j)=1)}(\hat{y}^{(i,j)} - y^{(i,j)}) + (a-\hat{a})\sum_{(i,j):r(i,j)=1)}((\hat{y}^{(i,j)} - y^{(i,j)})^2)\theta\theta^T$, which is positive if we choose $q(i,j)$ and initial states carefully.

iii. Rounding

One of the reason that we introduce another rounding method is that we are not sure about the rating when our prediction right in the middle of two stars. Here we introduce other rounding method:

$$\hat{y}^{(i,j)} = 1 + \sum_{k=1}^{4}[1 + exp((-\theta^j)^T x^i - C_k)]^{-1} \tag{5}$$



From the graph above we can see that the density near each rating become larger. Another reason that we use this rounding method is that the interval between two stars is not likely to be the same. So we plug this rounding back into the stochastic gradient descent. The rounding parameters are also variables, updated by the SGD.

iv. Influence of friends

In order to improve our accuracy, we will consider the influence of social network. The main idea here is to represent each user as a weighted sum of oneself and friends, with weights related to the similarity of them. Suppose $w^{(i,k)}$ is the weight of friend k for user i. Then our prediction become:

$$\hat{y}_{(i,j)} = (\sum_{k:e(i,k)=1} w^{(i,k)} x^{(k)})\theta^{(j)} \tag{6}$$

where $w^{(i,k)} = x^{(i)}x^{(k)}$ indicate the similarity between friends and users. In another word, we add friend's influence to a user with weight. By introducing the influence of friends, we improve the performance of our algorithm by about 2

III. Implementation and Results

In order to test the result of our algorithm, we first need to trim our dataset. We select all the restaurants in the city of Pittsburgh. And then we only keep the restaurants and users with reviews greater than 10. That ends up with 651 users and 803 businesses with reviews 16425. We split the trimmed dataset by ratio of 1:9 into test and training dataset. For the selected dataset, the restaurants can be divided into 276 categories.

Our algorithm for revised CF:

- Initialize $x^{(1)}, \dots x^{(n_u)}, \theta^{(1)}, \dots \theta^{(n_b)}$ to small values. In this case, we initialize $x^{(i)}$ with 1 and $\theta^{(j)}$ with $\frac{1}{num\,of\,categories(276)}$
- Loops:
  Minimize $J(x^{(1)}, \dots x^{(n_u)}, \theta^{(1)}, \dots \theta^{(n_b)})$ using gradient descent. For every $j = 1, \dots n_b, i = 1, \dots, n_u$:

$$x_k^{(i)} : x_k^{(i)} - \eta(\frac{\sum_{j:r(i,j)=1} (1 - \frac{1}{8}(\hat{y}^{(i,j)} - y^{(i,j)}))(\hat{y}^{(i,j)} - y^{(i,j)})exp(-a\hat{y}^{(i,j)})\theta_k^{(j)}}{\sum_j r(i,j)} + \lambda x_k^{(i)}) \tag{7}$$

$$\theta_k^{(j)} : \theta_k^{(j)} - \eta\left(\frac{\sum_{i:r(i,j)=1)} (1 - \frac{1}{8}(\hat{y}^{(i,j)} - y^{(i,j)}))(\hat{y}^{(i,j)} - y^{(i,j)})exp(-a\hat{y}^{(i,j)})x_k^{(i)}}{\sum_i r(i,j)} + \lambda\theta_k^{(j)}\right) \tag{8}$$

For new rounding, the basic idea is the same, just with substituting above formula with new gradient of x_i,θ_j, and c_k.

The parameter for $q(i,j)$we choose is $1/4$. For a user with parameters x and a business with(learned) freatures θ, predict a star rating of $\theta^T x$

	collaborative	revised collaborative
$\lvert\hat{y} - y\rvert \leq 0.5$	0.359	0.337
$\lvert\hat{y} - y\rvert \leq 0.5$ and rating < 3	0.137	0.217
$\lvert\hat{y} - y\rvert \leq 0.5$ and rating ≥ 3	0.389	0.357
$\lvert\hat{y} - y\rvert \leq 1$ and rating < 3	0.332	0.447
MAE	0.855	0.936
MAE($\hat{y} < 3$)	1.422	1.213
MAE($\hat{y} \geq 3$)	0.762	0.891

Where MAE(Mean absolute error) is the average absolute deviation of predictions to the ground truth data.

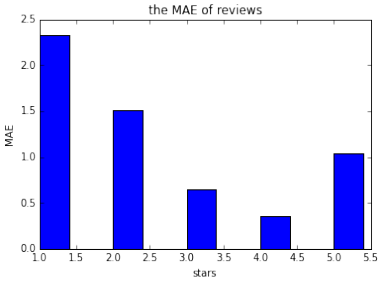
$$MAE = \frac{\sum |\hat{y} - y|}{m} \tag{9}$$

where m is the number of testing instances. The smaller the MAE, The better the inference.

From the result, we notice that the performance of revised collaborative algorithm is a lot better than the basic one if rating<3. That is because the gaussian curve has more lower rate part than the actual rating curve. And the quantity of lower rate restaurant is relatively small. The performance for the overall rating of two algorithm are pretty much the same. But according to our analysis above, the revised algorithm should perform better for the overall restaurants' rating.

	collaborative	Rounding
$\lvert\hat{y} - y\rvert \leq 0.5$	0.359	0.386
$\lvert\hat{y} - y\rvert \leq 0.5$ and rating < 3	0.137	0.058
$\lvert\hat{y} - y\rvert \leq 0.5$ and rating ≥ 3	0.389	0.440
$\lvert\hat{y} - y\rvert \leq 1$ and rating < 3	0.332	0.181
MAE	0.855	0.788
MAE($\hat{y} < 3$)	1.422	1.577
MAE($\hat{y} \geq 3$)	0.762	0.659

As we can see from the table above, with rounding our overall performance is better than the original one, especially for rating greater than 3. However, for rating smaller than 3, the performance is not that good.



From the MAE distribution base on different rating, we find out that our algorithm works good for rating equals 4 or 3. As for rating equals 5, we can revised our algorithm with the average rating. Cause if a restaurant got extremely high stars, it is safe to say that every user will give 5 star rating. However, when the rating equals 1 or 2, our algorithm works bad. We further explore the reason. First, we check the average rating of these restaurants, we find out that these restaurants are actually not that bad, most of them can get an average rating of 3. And then we explore the rating of users' friends, but the low rating is not that relevant to friends' rating. Let's suppose that these people are mean to the restaurants and they always give the low rating. But that's not true either. So we say these low rating are just small probability event, it has nothing to do with users' personal interest. Anywhere, since we are building a recommender system, we care more about the high rating part, the low rating is not that important to us.

IV. Future plan

Since our algorithm was not improved too much by friendship information, we may consider our method method to utilize the social influence. Another thing is that we may consider our models other than collaborative filer to solve the low rating part problem.

References

[1] Andrew Ng's Lecture on Recommender System

[2] He, J., & Chu, W. W. (2010). A social network-based recommender system (SNRS). Springer US.