CSE 701: Personalizing Web/Social Media

Yelp Business Rating Prediction

Harshad Deshpande
harshadr@buffalo.edu
5013 3685

Index

Introduction

- The target of this project is to build a model that will provide rating prediction for users for new businesses or existing business that they have not yet rated, on the basis of the ratings that they have already posted.
- Based on the predicted ratings, businesses can be recommended to the users.
- The goal of this project is to build a model that can deal with the problem of very small number of ratings provided by users, also known as the cold start problem.
- The Model should be efficient in coming up with predictions that can overcome the cold start problem.
- The nature of the data and the mapping for each user to businesses that he/she has rated, along with the rating and the matrix thus formed will be explained in the subsequent section.
- The overview of the data provided by Yelp can be expressed as a set of files that consist information about
    - All the users that have provided the ratings,
    - private users who have rated businesses but choose to keep their information private,
    - All the businesses that have been rated
    - Average rating for each public user
    - Average stars/rating for each business
    - Check-in information for each user
    - Brief textual review by each user for each business

Dataset Information

Dataset Overview:

- The dataset dump provided by Yelp consists of reviews, businesses, users and check-ins for Phoenix, Arizona.
- There are two sets, namely, the training set and the testing set.
- The training set consists of
    - 43,873 users
    - 229,907 reviews
    - 11,537 businesses.
- The testing set consists of
    - 5,105 users
    - 1,205 businesses
    - 22,956 reviews to predict

- Each "*business_id"* corresponds to a Business object that can be described by the following JSON:

{
type: *business*,
business_id: *encrypted_business_id*,
name: *business_name*,
city: *business_city*,
.
.
.
stars: *business_rating (stars)*
}

Each "*review*" JSON object is as follows:

{
type: *review*,
business_id: *encrypted_business_id*,
user_id: *encrypted_user_id*,
stars: *star_rating*,
text: *text_review*,
.
.
}

Each user's profile is represented thus:

{
type: *user*,
user_id: *encrypted_user_id*,
name: *user_name*,
review_count: *review_count*,
average_stars: *average_stars*,
}

Analyzing Data

- The size of the main data (reviews) file is 260MB and it contains 229907 lines.
- Each line represents a JSON object or dictionary, for instance:

{"votes": {"funny": 0, "useful": 5, "cool": 2},

"user_id": "rLtl8ZkDX5vH5nAx9C3q5Q",

"review_id": "fWKvX83p0-ka4JS3dc6E5A",

"stars": 5, "date": "2011-01-26",

"text": "My wife took me here on my birthday for breakfast and it was excellent. The weather was perfect which made sitting outside overlooking their grounds an absolute pleasure. Our waitress was excellent and our food arrived quickly on the semi-busy Saturday morning. It looked like the place fills up pretty quickly so the earlier you get here the better.\n\nDo yourself a favor and get their Bloody Mary. It was phenomenal and simply the best I've ever had. I'm pretty sure they only use ingredients from their garden and blend them fresh when you order it. It was amazing.\n\nWhile EVERYTHING on the menu looks excellent, I had the white truffle scrambled eggs vegetable skillet and it was tasty and delicious. It came with 2 pieces of their griddled bread with was amazing and it absolutely made the meal complete. It was the best \"toast\" I've ever had.\n\nAnyway, I can't wait to go back!",

"type": "review",

"business_id": "9yKzy9PApeiPPOUJEtnvkg"} //End of File

- The other required files are the files that contain information about all the users and all the businesses.
- Each row in both the files can be shown as follows:

business_file:

{

"business_id": "rncjoVoEFUJGCUoC1JgnUA",

"full_address": "8466 W Peoria Ave\nSte 6\nPeoria, AZ 85345",

"open": true,

"categories": ["Accountants", "Professional Services", "Tax Services", "Financial Services"],

"city": "Peoria",

"review_count": 3,

"name": "Peoria Income Tax Service",

"neighborhoods": [],

"longitude": -112.241596, "state": "AZ", "stars": 5.0, "latitude": 33.581867000000003, "type": "business"

} //End of File

user_file:

{"votes": {"funny": 0, "useful": 7, "cool": 0},

"user_id": "CR2y7yEm4X035ZMzrTtN9Q",

"name": "Jim",

"average_stars": 5.0,

"review_count": 6,

"type": "user"

} //End of File

Specifications, Scope and Approaches

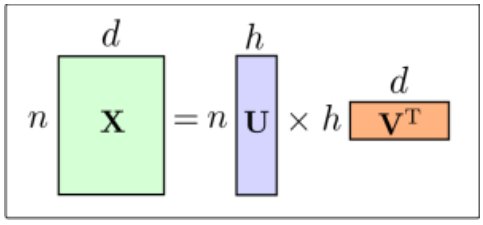
- After parsing through the entire file, putting each user_id and business_id in a hashset, it is found that there are appx 45000 unique user_ids and 11,537 business_ids.
- However, this is contradictory to the information of users provided in the users file, (which gives information about 43,873).
- Hence, we extract all the public users from the users file and store them in a hashset. After which we iterate through all the rows of the reviews file and check if the user_id that has rated for the respective business_id belongs to the hashset (that is, if the user is a public user and has some information associated with him.), if yes then we consider that user for future rating prediction.
- After building a ratings matrix from this data, the dimensions of that matrix are 43873 by 11537.
- This matrix contains 506162801 entries. After traversing the full matrix, it is observed that very few entries are non zero and that its is a highly sparse matrix.
- For such a starting matrix (also referred as cold start problem), Content Based or Collaborative Filtering are not the best options as the matrix is highly sparse consisting of mostly zeroes. Thus, Matrix Factorization stands as a good approach for this problem.
- Unfortunately, the memory of the machine in use is not sufficient to load and support operations on a matrix of this size with float64 type entries.
- Hence, pruning the dataset for a certain number of users and businesses for using the aforesaid model is required.

Pruning based on Maximizing Ratings Values

- For pruning the dataset for a certain users and businesses, instead of randomly selecting the users and businesses, which can result in a complete 0s matrix, we determine for each user the sum of all the ratings that he/she has submitted.
- Then we select all the users that have the sum of ratings more than 255.
- This gives us a matrix of 406 users by 11537 businesses.
- We apply the same pruning technique to this new formed matrix to further find the best businesses.
- Thus the new matrix we gain by applying the pruning is of the shape 406 users by 274 businesses.

Algorithm: Matrix Factorization

- The goal of Matrix Factorization is to reconstruct the given matrix as a dot product of two distinct matrices that have dimension of the order ‘n’ by ‘h’ and ‘h’ by ‘d’, where ‘n’ and ‘d’ are the dimensions of the original ratings matrix and the ‘h’ is less than both ‘n’ and ‘d’.



- Which can be stated as : $X \approx U \times V^T = \hat{X}$
- Here, ‘h’ represents the number of latent features (or can be considered as hidden properties) that each user has. The goal of matrix factorization is to find the hidden features for each user so that the cold start problem can be overcome.
- After we have calculated both the matrices U and V, we can predict the score for a business ‘d’ by a user ‘u’ by the following equation:

$$\sum_{k=1}^{h} u_{ik} v_{kj}$$

- The construction of these two matrices starts with creating two matrices of size ‘n x h’ and ‘h x d’ and filling them with random values.
- Then for a decided number of iterations, we calculate the error between the element in the resultant matrix of the dot product of U and V and the original matrix in each iteration.
- We try to minimize this error by using gradient descent (partial differentiation with two variables) which gives us two parameters, as (-2 * $error_{i,j}$ * $u_{k,j}$) and (-2 * $error_{i,j}$ * $v_{k,j}$) where error at each step is calculated as follows:

$$error_{ij}^2 = (x_{ij} - \hat{x}_{ij})$$

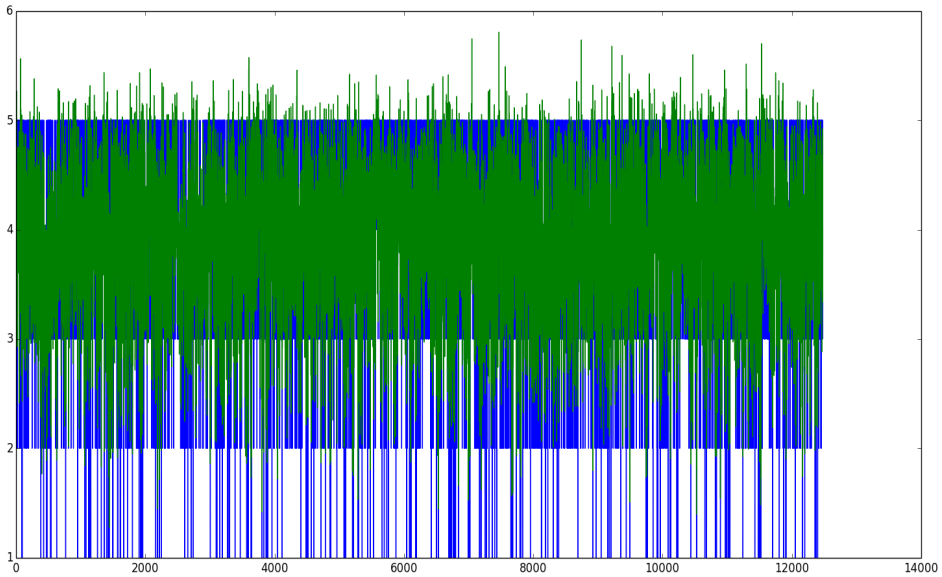
- We multiply these factors with a small constant weight and add it to the respective U and V matrix values to come close to the value in the original ratings matrix.
- We also add regularization factor beta as follows to avoid overfitting:

$$\frac{\beta}{2}\sum_{k=1}^{K}(\|U^2\| + \|V^2\|)$$

(beta's favored value could be taken a range of 0.03 to 0.08.)

Results Obtained & Evaluation

- For our ratings matrix, we obtain two matrices U and V which have the least RMSE of 0.45 and give fairly good results.
- The scatter plot of nonzero values in the original matrix and the new values at the same indexes calculated by the model can be shown as follows: (alpha = 0.0005, beta = 0.05, iterations = 80, latent Features = 80)



Blue = original values, Green = reconstructed values

- It can be observed from the scatter plot, that, for most of the values, the reconstructed values come close to the original value, and no errored value is beyond 6. With better parameter tuning the results should improve.
- The evaluation metric being used is RMSE (Root Mean Squared Error).
- We evaluate the RMSE for all the values that were nonzero in the original matrix and calculate the difference between them and the new values formed in the new matrix.
- This difference can be used to calculate RMSE which can be given by the following equation:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \hat{x}_i)^2}{n}}$$

Where:

n = the total number of non zero ratings
$\hat{x}_i$ = the predicted rating
x_i = the actual rating

The optimal RMSE (considering the run time of the algorithm) was 0.518 for 70 latent features, 80 iterations, beta = 0.05 and alpha = 0.0005.

Some iterations have yieled the following observations:

Alpha	Beta	LatentFeatures	GradientDescentSteps	RMSE
0.0002	0.02	70	100	0.703315927802
0.0002	0.02	80	100	0.681162738532
0.0003	0.03	90	100	0.65212242015
0.0005	0.05	70	80	0.522318603646

Conclusion & Future Work:

- Thus, in this project we have tried to reconstruct the original sparse matrix using two matrices of lesser dimensions using matrix factorization.
- Due to memory constraints, we have used a pruned dataset of the original dataset and also used serialization and deserialization of objects (matrices) formed during the algorithm.
- The predicted results are serialized and stored and can be used to see how a user will rate any business.
- With a framework like Hadoop or any other parallel processing techniques, we will try to use the whole original data and factorize it to actually find