Table of Contents

Data

	businessID	categories	categoryID	rating	reviewHash	reviewText	reviewTime	unixReviewTime	userID
142684	B245273025	[Pizza Restaurant, European Restaurant, Italia...	NaN	5.0	R120932416	great little place on the corner in maplewood ...	Jan 3, 2012	1325649978	U424076415
184001	B613896389	[Vietnamese Restaurant, Asian Restaurant, Sout...	2.0	4.0	R261542656	Excellent Vietnamese food	Apr 26, 2012	1335484800	U124775493
911	B315570575	[Polish Restaurant, Eastern European Restauran...	3.0	4.0	R909813365	You don't come here for anything other than cl...	Aug 27, 2012	1346123715	U179531035
43129	B446355818	[American Restaurant]	0.0	4.0	R597131645	Good atmosphere with outdoor seating. The menu...	Jun 16, 2013	1371434045	U180811523
9528	B451079132	[Park]	NaN	4.0	R582684656	This is the park with the fewest weirdos, so i...	Jul 24, 2013	1374691155	U955518733

The data was loaded into a pandas data frame. Out of the 9 columns of the data frame, the models used in the assignment only considered userID, businessID, categories and rating. Before using the data for generating the models, the data frame was shuffled to simulate independent random sampling

Task 1 – Visit Prediction

Task 1 Involved predicting whether a user has visited a business or not. This problem can be considered as a binary classification problem, or a recommender system problem. Multiple methods, making use of both approaches were tried out. The predictions were uploaded under the username **JoJoy**.

Approach 1 – Logistic Regression

A logistic regression model was tried first, making use of the following features,

1. Average rating given by a user to previous visited businesses
2. Average rating given to a business by users of visited that business
3. Number of categories that are similar between the business and that of businesses visited a user
4. Number of users who visited a particular business

Several dictionaries were created, like userVisitHistory which stores the list of businesses visited by a user and businessVisitHistory which stores the list of users who visited the business and so on, to easily access data corresponding to a particular user or a business. These dictionaries were then used to create the above features. If a user or business had no prior data, the features were set to 0.

The model takes inner product of the features with the parameters of the logistic regression model and then checks if that value is greater than zero or not and predicts 1 and 0 for respective cases. The logistic regression model used the log likelihood method to find the best model parameters. Since a logistic regression model require both positive and negative training data, negative samples, which corresponds to user business pairs which were not present in the original dataset. The negative samples were generated randomly and was made sure to not include any business user pair that were present in the data.

$$y_i = \begin{cases} 1 & \text{if } X_i \cdot \theta > 0 \\ 0 & \text{otherwise} \end{cases}$$

The basic logistic regression model, X_i represent the feature matrix, Θ represent the unknowns (Courtesy: CSE 258 Lecture slides.

The training samples containing positive and negative instances were used to train the model. The value of unknown parameter theta were found to be -5.126, 3.618, 0.359, 1.516 and -0.003 , where the first value is the bias (constant value) .After training the model on the training set, validation set was used to find the regularization constant. The best value of regularization constant (lambda) was found to be 0.1. This model when uploaded to kaggle had accuracy of around 0.64

Insights

The major shortcoming of this model was that its simplistic and since negative samples had to be generated, there would be no guarantee of proper accuracy.

Algorithm:

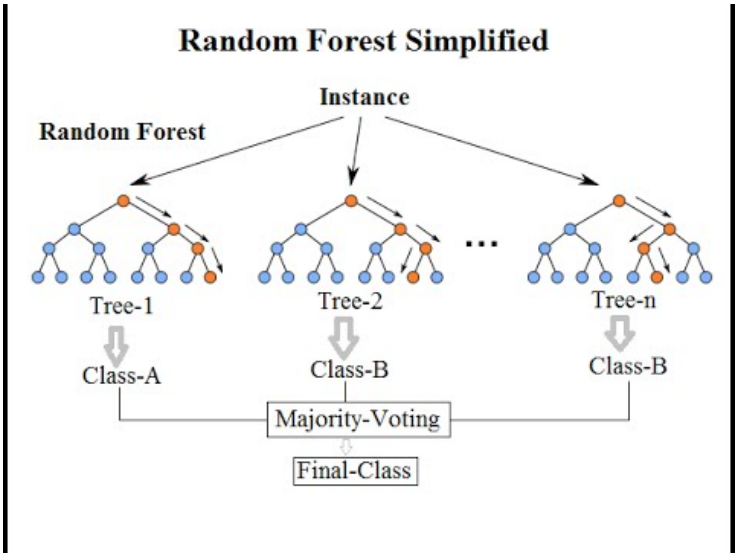
1. Create dictionaries to store informations like previous business visited by a user, users who visited a business using businessID and userID as keys.
2. Split data into training and validation sets and extract features from both sets, with label 1, corresponding to visit. Add randomly generated negative samples to training and validation sets, which has label 0 corresponding to no visit
3. Extract features from both training and validation sets
4. Use training features and training label to find the unknown parameters Θ and validation set to find hyper parameter lamda and choose the lamda with largest validation accuracy
5. When using the businessID and userID from test file, if the user and business were present in the dictionary key set, use their information from dictionary to find features. If user or business ID were not present in dictionaries, set all features to zero.
6. If inner product of Θ with features is greater than 1, predict 1. Else predict 0.

Approach 2 – Random Forest classifier

A binary random forest model was trained using the following features

1. Number of businesses visited by a user
2. Number of categories that are similar between the business and that of businesses visited a user
3. Jaccard similarity between the business and the last business visited by the user
4. Jaccard similarity between the categories of businesses visited by the user and that of the user of last visited the business
5. Mean rating given to the business

The random forest works by training each tree of the forest with some randomly chosen data, to find the best features and thresholds that leads to best classification using gini impurity index. Once certain number of trees are trained, the model uses all of the trees to predict whether a user would visit a business or not, and the outcome predicted by majority number of trees are used as the final prediction. This is an example of an ensemble method, which combines multiple simple prediction models to form an unbiased prediction with small variance.



The basic random forest classifier. This model works by training each tree separately and then using the majority voting to find predictions (courtesy: https://www.youtube.com/watch?v=ajTc5y3OqSQ)

The model also used randomly generated negative samples. This model was trained on 75 % of data and the rest was used to find hyper parameters, number of trees, which was around 50 and maximum depth which was around 2. The feature importance calculated by the random forest classifiers were [0.00985363 0.76330149 0.0494063 0.17642229 0.00101628]. The random forest classifier function was used from the sklearn module. The model had test accuracy of around 0.69.

Insights

Compared to the previous approach, this model also suffers from the fact that randomly generated negative samples were used to train the model, which led to lower than optimal accuracy.

Algorithm:

1. Create dictionaries to store information like previous business visited by a user, users who visited a business using businessID and userID as keys.
2. Split data into training and validation sets and extract features from both sets, with label 1, corresponding to visit. Add randomly generated negative samples to training and validation sets, which has label 0 corresponding to no visit
3. Extract features from both training and validation sets.
4. Use the training set and training labels to fit the random forest classifier and validation set to find optimum number of trees and maximum depth of each tree
5. When using the businessID and userID from test file, if the user and business were present in the dictionary key set, use their information from dictionary to find features. If user or business ID were not present in dictionaries, set all features to zero.
6. Use the fitted model to make prediction.

Approach 3 – Collaborative filtering (using business – business similarity)

A similarity matrix was calculated which represented the Jaccard similarity between two businesses, by computing the number of common users who visited the businesses and dividing it by the total number of unique users of visited either of the two businesses. This similarity matrix

was used to find the similarity between the business and the businesses visited by the user in the past. The maximum similarity, 1 would represent the case when the same group of users visited both businesses, and the minimum value of 0 represents the case where the users visited one business and none of them visited the other. If a similarity was established, the model would predict visit, else not visit using 1 and 0 respectively.

Insights

This model was not using any randomly generated negative samples, and was based on the actual visit data. So, out of the three approaches, this one was least corrupted by randomly generated data. Hence, compared to the previous approaches, this approach resulted in the best performance. The model had the highest test accuracy, around 0.88.

Algorithm:

1. Create dictionaries to store information like previous business visited by a user, users who visited a business using businessID and userID as keys.
2. Use entire dataset to find similarity matrix between each business-business pair, using Jaccard similarity metric.
3. Get userID and businessID from testing data and find similarity of that particular business with all business the user visited previously. IF there is similarity, predict 1, else 0

Task 2 – Rating prediction

Task 2 involved predicting the rating a user would give to a business. Multiple approaches were used to generate this model

Approach 1 – Model with user and business bias (α, β_u, β_i)

This model assumed that the rating can be modelled of the form $\alpha + \beta_u + \beta_I$. Here the scalar, α represents a sort of bias that is common to all ratings given by a user to a particular business. β_u, β_I represent the bias of each user to a particular business and each business to a particular user respectively and hence are one dimensional vectors.

To train the model, a rating matrix was generated where each row corresponds to a particular user and each column to a particular business. The matrix dimension was around 18000 x 20000 and was mostly sparse. Since we had lots of unknown variables, alternating least squares method was used to minimized regularized error. This method works by fixing every other parameter except one, and then finding the optimum value for that one parameter. Since closed form equations were used, this model converged in few iterations.

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

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

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

Close form equations for the alternating least squares method (courtesy: CSE 258 Lecture slides)

The parameters β_u + β_I which corresponds to the bias of a particular user and a business respectively, were stored in dictionaries, with userID and businessID as keys. The bias values for all users and business were initialized with zero. The ALS method was running multiple iterations to obtain the optimum parameter values.

Hyper parameter lambda, the regularization constant that minimized validation error was found to be around 5. The model predicted the rating by adding α with the bias values corresponding to the userID and businessID. If the user or business was new, these values were set to zero. The model had an accuracy of around 0.76.

Insights

The major advantage of this model is the fast convergence and the major disadvantage with this model is its simplistic approach to rating prediction. This model only uses the rating and no other feature to make the prediction.

Algorithm:

1. Create dictionaries to store informations like previous business visited by a user, users who visited a business using businessID and userID as keys.
2. Initialize all bias values to zero, and create a dictionary for β_u and β_I with userID and business ID as keys.
3. Using each training sample, first find optimum value of alpha using the first equation of ALS method, keeping other unknowns fixed
4. Once alpha was found, use the second and third equations of ALS method to find the β_u and β_I keeping other unknowns fixed
5. Repeat above two steps multiple iterations to get proper convergence and optimum values of all unknowns
6. Get userID and businessID from test file, if that userID and businessID was present in the dictionary keys, use the paramters corresponding to the ID and predict rating. If userID or businessID was not present, then set the bias value corresponding to that ID to zero and find final rating. If userID and businessID were both not present on dictionaries, rating was set to alpha.

Approach 2 – Latent factor model

Rating was modelled as $\alpha + \beta_u + \beta_I + \gamma_u \cdot \gamma_i$. This model had an extra terms γ_u and γ_i , which represents the low dimensional representation of each user and business. The low dimensional representations along with the bias terms were stored in dictionaries with userID and businessID as keys. The bias values were initialized to zero, and the gamma values were initialized to randomly generated numpy arrays, to implement gradient descent method to find optimum parameter values. Along with gradient descent, ALS method was also used.

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

The above term was minimized using gradient descent to find parameter values (courtesy: CSE 258 Lecture slides)

Learning rate of 0.001 was used. Hyper parameter k (dimensionality of γ_u and γ_i) and lambda were found to be around 5 and 4 respectively. The model had a test RMSE of around 0.75.

Insights

The major advantage of this model is the fact the we can determine a low dimensional representation of a user and a business and it is used to find the rating. The major disadvantage of the model is it takes lots of iterations to converge.

Algorithm:

1. Create dictionaries to store informations like previous business visited by a user, users who visited a business using businessID and userID as keys.
2. Initialize all bias values to zero, and create a dictionary for β_u , β_I with userID and business ID as keys. Initialize γ_u and γ_I with randomly generated numpy arrays, also store them in dictionaries.
3. Using each training sample, first find optimum value of alpha using the first equation of ALS method shown in the previous approach, keeping other unknowns fixed. An extra term will be present, γ_u . γ_I
4. Once alpha was found, use the second and third equations of ALS method to find the β_u and β_I keeping other unknowns fixed
5. The values of γ_u and γ_I was updated using gradient descent and was run for multiple iterations to get optimal value.
6. Repeat above three steps multiple iterations to get proper convergence and optimum values of all unknowns
7. Get userID and businessID from test file, if that userID and businessID was present in the dictionary keys, use the paramters corresponding to the ID and predict rating. If userID or businessID was not present, then set the bias value and gamma values corresponding to that ID to zero and find final rating. If userID and businessID were both not present on dictionaries, rating was set to alpa.

Results

Kaggle user name: **JoJoy**

Task 1 – Visit Prediction

1. Approach 1 (logistic regression)
   Regularization constant, lamda = 0.1
   Unknown parameters, Θ = -5.126, 3.618, 0.359, 1.516, -0.003
   Test accuracy = 0.64
2. Approach 2 (Random Forest)
   Number of trees in the ensemble = 50
   Maximum tree depth = 2
   Feature importance = 0.00985363, 0.76330149, 0.0494063, 0.17642229, 0.00101628
   Test accuracy = 0.69
3. Approach 3 (Business – Business Collaborative filtering)
   Similarity metric used – Jaccard similarity
   Test accuracy = 0.88 (Used for final Kaggle submission)

Final rank = 51
Best test accuracy = 0.882

Task 2 – Rating Prediction

1. Approach 1 (Model with user and business bias)
   Regularization constant, lamda = 5
   Test accuracy = 0.76

2. Approach 2 (Latent Factor model)
   Number of dimensions of gamma, k = 5
   Regularization constant, lamda = 5
   Test accuracy = 0.75

Final rank = 207
Best test accuracy = 0.75756