CSE 258-Web Mining & Recommender Systems

Assignment 1
Google Local Visit and Rating Prediction

Girish Bathala, (**Kaggle username : No One**) November 29, 2017

1 Objective

The objectives of this assignment are :

- **Visit Prediction:** Predict for (user,business) pairs from the test set whether the user visited the business, i.e, whether it was one of the business they reviewed
- **Rating Prediction:** Predict people's star ratings of businesses' as accurately as possible, for (user,business) pairs in the test data.

2 Task A : Visit Prediction

The goal of this task is classify each (user,business) pair as "1" or "0". "1" ("visited") signifies that the user visited the business. "0" ("non-visited") signifies that the user did not visit the business. The following pipeline was followed to implement the task

2.1 Data-set preprocessing and preparation

The Google Local dataset has 200,000 reviews of businesses' rated by users. The dataset contains many attributes but for the purpose of this task, only the following were used:

- **businessID:** The ID of the business. This is a hashed product identifier from Google.
- **userID:** The ID of the reviewer. This is a hashed user identifier from Google.
- **rating:** The star rating of the user's review.

The dataset was divided into train, validation in the following way. For the train-validation pipeline, the first 100,000 user-business-rating triplets were used as the training set. The rating is considered as class "1" or "visited". The second 100,000 user-business pairs were used for cross-validation. Furthermore, an additional 100,000 data points were added to the validation set by selecting user,business pairs, such that the user had never visited the business. These additional pairs were added to make the validation set balanced with the "0" or "Non-visited" class. The entire 200,000 data points were used for building the final model once all the parameters were estimated after validation

2.2 Training algorithms used

The approach used was to build an ensemble of 4 different collaborative filtering(**CF**) algorithms to model the relation between the users and businesses :

1. User-User Similarity CF model : Pearson
2. Item-Item Similarity CF model : Pearson
3. User-User Similarity CF model : Jaccard
4. Item-Item Similarity CF model : Jaccard

The following steps were followed to implement the above algorithms:

- The first step is to form the utility matrix R with n_u X n_i dimensions, where n_u is the number of unique users in the train set and n_i is the number of unique bussineses in the test set.
- Definitions : I_u = set of items purchased by user u, U_i= set of users who purchased item i, Each entry R_{ui} for user u and business i is either "1" or "0", signifying whether the user visited the business or not.
- Pearson Similarity between any two vectors is calculated by the formula in figure 1. It is shown for two users (u,v) in this case. It can be done in a symmetric way to find similarity between items.

items rated by both users average rating by user *v*

$$\text{Sim}(u,v) = \frac{\sum_{i \in I_u \cap I_v} (R_{u,i} - \bar{R_u})(R_{v,i} - \bar{R_v})}{\sqrt{\sum_{i \in I_u \cap I_v} (R_{u,i} - \bar{R_u})^2 \sum_{i \in I_u \cap I_v} (R_{v,i} - \bar{R_v})^2}}$$

Figure 1: Pearson Similarity

- Jaccard Similarity between any two sets is given by

$$J(U_i, U_j) = \frac{|U_i \cap U_j|}{|U_i \cup U_j|}$$

  where U_i is the set of users who rated item i and U_j is the set of users who rated item j. This gives the similarity between two items sim(i,j). A symmteric approach can be followed to compute the Jaccard Similarity between two items

- For user-user similarity a n_u X n_u dimension matrix U was constructed, where n_u was the number of users in the train set. Each element $U_{uv} = sim(u, v)$ calculated above. Two combinations are possible here by using either Jaccard or Pearson similarity
- For item-item similarity a n_i X n_i dimension matrix I was constructed, where n_i was the number of items in the train set. Each element $I_{ij} = sim(i, j)$ calculated above. Two combinations are possible here by using either Jaccard or Pearson similarity, making it a total of 4 models

- Matrix U and I are sorted along rows in descending order. By doing so, the most similar items (users) for a given item (user) are present on the left side of the sorted matrix. A neighbourhood of 'K' most similar items/users is selected in each of the 4 cases above

2.3 Prediction and Validation

Using the sorted matrices in each of the 4 cases above, predictions can be made on the validation (test) set :

- User-User Similarity Case : For each (user-u,business-i) pair in the validation (test) set, if i is present in any of the businesses rated by the users $v \in N_u^k$, i.e, the top K similar users for u, the output class is predicted as "1", otherwise 0.
- Item-Item Similarity Case : For each (user-u,business-i) pair in the validation (test) set, if u rated any item $j \in N_i^k$, i.e, the top K similar items for i, the output class is predicted as "1", otherwise 0.
- Each of the 4 models are then validated to obtain the 4 parameters K_1, K_2, K_3 and K_4 for the 4 models described above.
- The final output is considered as an ensemble of the 4 models. A voting procedure is followed and the class with the maximum votes is predicted as the output. For example, if more than two models predict "1", the output is 1
- In case any user is not present or if it is a tie between the 4 models, the item is looked up in the most popular business array (Hw3, Q1). If it is present, the output is predicted as "1", otherwise "0". If item is not present in the training set, the output is predicted as "0".

3 Task B : Rating Prediction

The goal of this task is to predict a user's star rating of a particular business. The following pipeline was followed to implement the task :

3.1 Data-set preprocessing and preparation

The Google Local dataset has 200,000 reviews of businesses' rated by users. The dataset contains many attributes but for the purpose of this task, only the following were used:

- **businessID:** The ID of the business. This is a hashed product identifier from Google.
- **userID:** The ID of the reviewer. This is a hashed user identifier from Google.
- **rating:** The star rating of the user's review.

The dataset was divided into train, validation in the following way. For the train-validation pipeline, the first 100,000 user-business-rating triplets were used as the training set. The second 100,000 user-business pairs were used for k-fold cross-validation. The entire 200,000 data points were used for building the final model once all the parameters were estimated after validation

3.2 Training algorithms used

The approach used was to build an ensemble of 6 different recommender system algorithms to model the relation between the users and businesses. For each algorithm the 100,000 X 3 train-set was fed directly into SurPRISE's (Simple Python Recommendation System Engine) algorithm libraries.

3.2.1 BaselineOnly

Algorithm predicting the baseline rating estimate $\hat{r}_{ui}$ for given user u and item i.

$$\hat{r}_{ui} = b_{ui} = \mu + b_u + b_i$$

If user u is unknown, then the bias b_u is assumed to be zero. The same applies for item i with b_i.

This model tries to minimize the following regularized square error, where λ is the regularization parameter.

$$\sum_{r_{ui} \in R_{train}} \left(r_{ui} - (\mu + b_u + b_i)\right)^2 + \lambda \left(b_u^2 + b_i^2\right)$$

Two separate models are built using the "baselineOnly" idea from surprise :

- Using **Stochastic Gradient Descent** (**SGD**): The parameters for SGD are :
  - 'reg' or $lambda$: The regularization parameter of the cost function that is optimized
  - 'learning rate': The learning rate of SGD
  - 'n_epochs': The number of iteration of the SGD procedure
- Using **Alternating Least Squares** (**ALS**): The update equation for μ is the simple average of all the all ratings centered after subtracting b_u + b_i. The update equations for b_u and b_i are in fig : , where K is the $R_{trainset}$, $\lambda_2 = reg_i, and \lambda_3 = reg_u$, Parameters for the ALS algorithm are :

$$b_i = \frac{\sum_{u:(u,i)\in\mathcal{K}}(r_{ui} - \mu)}{\lambda_2 + |\{u|(u,i) \in \mathcal{K}\}|}.$$

$$b_u = \frac{\sum_{i:(u,i)\in\mathcal{K}}(r_{ui} - \mu - b_i)}{\lambda_3 + |\{i|(u,i) \in \mathcal{K}\}|}.$$

Figure 2: ALS: bi and bu update equations

  - 'reg_i': The regularization parameter for items.
  - 'reg_u': The regularization parameter for users.
  - 'n_{epochs}': The number of iteration of the ALS procedure.

3.2.2 KNNBaseline

A basic collaborative filtering algorithm taking into account a baseline rating. It also accounts for the similarity sim(u,v) between user pairs (u,v). Let $N_i^k(u)$ denote the K-user-neighbourhood of user u and people who have rated item i. The predicted rating r_{ui} is given by :

$$\hat{r}_{ui} = b_{ui} + \frac{\sum\limits_{v \in N_i^k(u)} \text{sim}(u,v) \cdot (r_{vi} - b_{vi})}{\sum\limits_{v \in N_i^k(u)} \text{sim}(u,v)}$$

The similarity between users is computed by using the pearson similarity between user u and v $\in N_i^k(u)$ for parameter K and item i. These sim(u,v) values are i,j positions in similarity Matrix R for the ratings/ utility matrix C, as in figure 3 The parameters for KNN are

$$R_{ij} = \frac{C_{ij}}{\sqrt{C_{ii} * C_{jj}}}$$

Figure 3: Pearson Similarity

- K – The (max) number of neighbors to take into account for aggregation
- $sim_{options}$ – Fixed for this task, pearson similarity
- $bsl_{options}$ – Similar to baseline updates and parameters in previous section "BaselineOnly"

3.2.3 Latent Factor Model : SVD

The utility matrix U is approximated in a lower dimension form as two matrices Q and P. Each representing the low dimensional representation of a user and business. The low dim representation is K = 1 for this task.

The prediction $\hat{r}_i$ is set as:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

If user u is unknown, then the bias b_u and the factors p_u are assumed to be zero. The same applies for item ii with b_i and q_i.

To estimate all the unknown, we need to minimize the following regularized squared error:

$$\sum_{r_{ui} \in R_{train}} (r_{ui} - \hat{r}_{ui})^2 + \lambda \left(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2 \right)$$

The minimization is performed by a very straightforward stochastic gradient descent:

$$\begin{aligned} b_u &\leftarrow b_u + \gamma(e_{ui} - \lambda b_u) \\ b_i &\leftarrow b_i + \gamma(e_{ui} - \lambda b_i) \\ p_u &\leftarrow p_u + \gamma(e_{ui} \cdot q_i - \lambda p_u) \\ q_i &\leftarrow q_i + \gamma(e_{ui} \cdot p_u - \lambda q_i) \end{aligned} \tag{1}$$

where $e_{ui} = r_{ui}$-$\hat{r}_{ui}$ These steps are performed over all the ratings of the trainset and repeated n_{epochs} times. Baselines are initialized to 0. User and item factors are randomly initialized according to a normal distribution

The parameters for SVD are as follows :

- $n_{factors}$ – The number of factors.
- n_{epochs} – The number of iteration of the SGD procedure
- lr_{all} – The learning rate for all parameters.
- reg_{all} – The regularization term for all parameters

3.2.4 Non-Negative Matrix Factorization : NMF

A collaborative filtering algorithm based on Non-negative Matrix Factorization. This algorithm is very similar to SVD. The prediction $\hat{r}_u i$ is set as:

$$\hat{r}_{ui} = q_i^T p_u,$$

where user and item factors are kept positive. The optimization procedure is a (regularized) stochastic gradient descent with a specific choice of step size that ensures non-negativity of factors, provided that their initial values are also positive.

At each step of the SGD procedure, the factors f or user u and item i are updated as follows:

$$\begin{aligned} p_{uf} &\leftarrow p_{uf} \cdot \frac{\sum_{i \in I_u} q_{if} \cdot r_{ui}}{\sum_{i \in I_u} q_{if} \cdot \hat{r_{ui}} + \lambda_u |I_u| p_{uf}} \\ q_{if} &\leftarrow q_{if} \cdot \frac{\sum_{u \in U_i} p_{uf} \cdot r_{ui}}{\sum_{u \in U_i} p_{uf} \cdot \hat{r_{ui}} + \lambda_i |U_i| q_{if}} \end{aligned} \tag{2}$$

where λ_u and λ_i are regularization parameters The Parameters for the model are :

- $n_{factors}$ – The number of factors
- n_{epochs} – The number of iteration of the SGD procedure

3.2.5 Slope One

: A simple yet accurate collaborative filtering algorithm.The prediction $\hat{r}_{ui}$ is set as:

$$\hat{r}_{ui} = \mu_u + \frac{1}{|R_i(u)|} \sum_{j \in R_i(u)} \text{dev}(i, j),$$

where $R_i(u)$ is the set of relevant items, i.e. the set of items j rated by u that also have at least one common user with i. $Dev(i, j)$ is defined as the average difference between the ratings of i and those of j:

$$\text{dev}(i, j) = \frac{1}{|U_{ij}|} \sum_{u \in U_{ij}} r_{ui} - r_{uj}$$

3.3 Prediction and Validation

The 6 models presented above are validated independently on the 100,000 validation set to find their respective best parameters. The Baseline ALS algorithm, Latent Factor Model, and KNNBaseline were found to be the best performing out of the three. The predictions were made based on the following update equation

$$\hat{r}_{ui} = \frac{\sum_{k=1}^{k=6} w_k * \hat{r}_{uik}}{\sum_{k=1}^{k=6} w_k}$$

where, w_k are the weights associated with each model prediction and $\hat{r}_{uik}$ is the predicted rating of each model. This ensemble model was validated to get the best performing weights w_k. The final model was trained using the best parameters and the entire training set (200,000). The weighted average of these models outputs on the test set were submitted on Kaggle