EECE 5698 Spring 2019 Term Project Report

Sai Sneha Yachamaneni, Kuan-Chih Lee
Instructor: Prof. Stratis Ioannidis

Introduction:

For diners, the decision of selecting a restaurant in the location they are is often very confusing process. In a recent survey conducted, it's been shown that 94% of diners will choose restaurants based on online reviews. Yelp is very good at providing this kind of recommendations.

Problem Statement:

Yelp is one of the mostly used service by the citizens of the USA for providing business reviews with their own experience. Yelp helps businesses like restaurants, hotels etc to establish their business page on their website. Our project is to build a Restaurant Recommendation Model using dataset provided by Yelp_Data_Challenge.

Data Description:

As part of Yelp data challenge, Yelp has provided data which consists of various JSON files. Primary file is review.json which consists of 6 million user reviews information for 200,000 restaurants.

Photo.json	Contains information about photos and consists of photo_id, business_id of the photo and to label of the photo
Checkin.json	Checkins on a business indicating number of checkins at a particular time of a day and day of the week
Tip.json	Tips written by a user on a business. Tips are shorter than reviews and tend to convey quick suggestions
Review.json	Contains full review text data including the user_id that wrote the review and the business_id the review is written for
User.json	User data including the user's friend mapping and all the metadata associated with the use
Business.json	Contains business data including location data, attributes, and categories

Data Preparation:

We need to transform the data to feed into recommendation models. Input files we will be using are review and business which are in json format.

Step 1: Read the json files into a SQL data frame using spark.read.json

```
$ review_df = spark.read.json("yelp_academic_dataset_review.json")
$ business_df = spark.read.json("yelp_academic_dataset_business.json")
```

Step 2: Glimpse of list of columns in both the data frames

```
$ review_df.columns

['business_id', 'cool', 'date', 'funny', 'review_id', 'stars', 'text', 'useful', 'user_id']

$ business_df.columns

['address', 'attributes', 'business_id', 'categories', 'city', 'hours', 'is_open', 'latitude', 'longitude', 'name', 'neighborhood',
```

'postal_code', 'review_count', 'stars', 'state']

Step 3: Drop the column "stars" from business_df to avoid duplicate columns after join

$ business_df = business_df.drop('stars')

Step 4: Perform inner join based on "business_id"

$ joinedrdd = review_df.join(business_df,['business_id'])

Step 5: Dropping irrelevant columns

$ joinedrdd_drop = joinedrdd.drop('cool', 'date', 'funny', 'review_id','text', 'useful','address', 'attributes', 'categories', 'hours', 'is_open', 'latitude', 'longitude', 'neighborhood', 'postal_code', 'review_count')

Step 6: Converting the SQL Data Frame to RDD

$ rdd = joinedrdd_drop.rdd.map(tuple)

Step 7: Encoding the Unicode formatted columns

$ rdd_encode = rdd.map(lambda (bid, star, uid, city, name, state): (bid.encode("UTF-8"),star,uid.encode("UTF-8"),city.encode("UTF-8"),name.encode("UTF-8"),state.encode("UTF-8")))

Step 8: Saving the RDD into a file

$ rdd_encode.coalesce(4,True).saveAsTextFile("ProcessedData")

Step 9: Loading saved data into program (model):

(In pyscript)
sc.textFile("FILE_DIR", use_unicode=False).map(lambda (bid, star, uid, city, name, state): (uid, bid, star))

Proposed Methods:

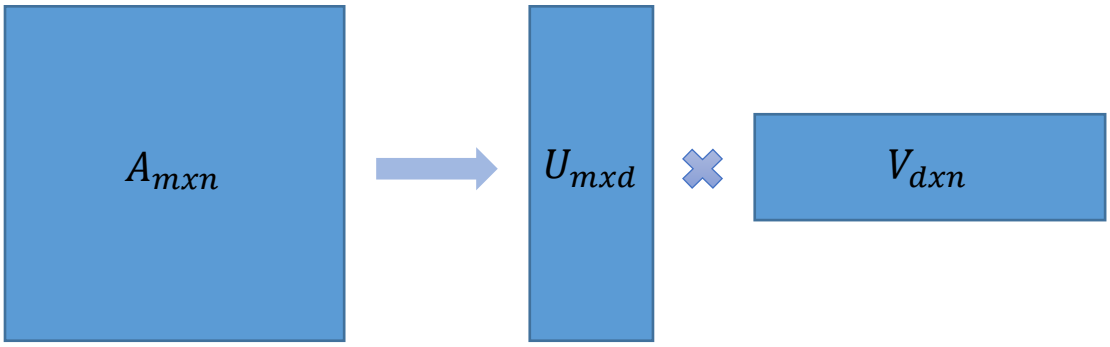
Why Latent Factor Model:

The original user-item rating score matrix is sparse and missing lots of rating scores

Example:

User/item	1	2	3	4	5
1	5	?	4	5	?
2	?	5	?	?	?
3	?	3	1	4	4
4	4	3	?	2	?
5	2	3	3	?	?
…					

For the first Netflix Prize competition, the number of users is 480,000 and number of movies is 17,700. The rating matrix should be 8.49 x 10^9 entries, but there are only 100 million non-zeros entries, about 1.17% of entries is rated. Therefore, we can use Latent Factor Model (SVD) to map high dimensional user and item profile into low latent dimension.



Method 1: RSVD with Bias

Compared with User-user and Item-item recommendation systems, Latent Factor Model (SVD) provides profound insights to figure out what kinds of item will interest users at most. Here, we further improve traditional SVD model with considering global and local biases of users' behavior. Global effect performs the baseline rating score of population, and local effect takes individual behavior into consideration and further adjust the rating score in person. With regularization, here we use L-2 norm, we can achieve more precise prediction than before without overfitting.

$$\mathrm{SE}(\mathrm{U},\mathrm{V}) = \sum_{i,j}(u_i^T V_j + \mu + b_i + b_j - r_{ij})^2 + \lambda_1\left(\|u_i\|^2 + \left\|v_j\right\|^2\right) + \lambda_2\left(\|b_i\|^2 + \left\|b_j\right\|^2\right)$$

$\mathrm{U} \in R^{kxd}$ is user vector, k is total user numbers of set{users} and d is latent dimension
$\mathrm{V} \in R^{wxd}$ is item vector, w is total item numbers of set{items} and d is latent dimension
μ is global mean of rating scores
b_i is local mean of user *i* rating scores
b_j is local mean of item *j* rating scores
r_{ij} is the rating score of user *i* to item *j*

Method 2 : SVD++

In this project, we implement simplified version of Asymmetric-SVD (ASVD), which is called SVD++. SVD++ inherent the nature of Bias-RSVD model, and it further concerns that the implicit behavior of each user. Implicit behavior, also called Neighborhood models, can be regarded as items that given user has seen or the user didn't see them before. The derivation can be found in Yehuda Koren's paper

$$\mathrm{SE}(\mathrm{P},\mathrm{Q},\mathrm{Y}) = \sum_{u,i}(q_i^T(p_u + |N(u)|^{-1/2}\sum_{j\in N(u)} y_j) + \mu + b_u + b_i - r_{ui})^2$$
$$+ \lambda_1\left(\|p_u\|^2 + \|q_i\|^2 + \sum_{j\in N(u)}\left\|y_j\right\|^2\right) + \lambda_2(\|b_u\|^2 + \|b_i\|^2)$$

$\mathrm{P} \in R^{kxd}$ is user vector, *k* is total user numbers of set{users} and *d* is latent dimension
$\mathrm{Q} \in R^{wxd}$ is item vector, *w* is total item numbers of set{items} and *d* is latent dimension
$\mathrm{Y} \in R^{wxd}$ is implicit user-item vector, *w* is total item numbers of set{items} and *d* is latent
$|N(u)|$ is total number of items that user *u* has seem
b_u is local mean of user *u* rating scores
b_i is local mean of item *i* rating scores
r_{ui} is the rating score of user *u* to item *i*

Implementation and Results:

To execute code in parallel, we need to connect to multiple machines at same time. Below are the steps for it

Step 1: Go to the folder contains /spark folder and Slurm batch file

Step 2: launch master

$ sbatch –reservation=eece5698 –mem 100G spark-mater-slurm

Step3: adding workers to master through master IP

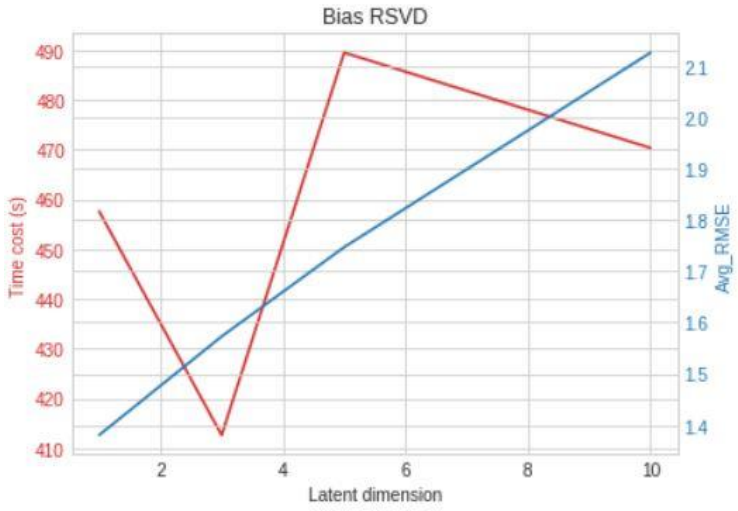
$ sbatch -N SLAVE_NUM --mem 100Gb --exclusive spark-workers.slurm spark://MASTER_IP:7077

Step4: Back to program folder and run code by sbatch or spark-submit

Case 1: Running Bias RSVD with 6 machines

For each fold: Location specific to state Nevada

1. About 1,200,000 train samples
2. Training set contains 410,000 users and 7,400 items
3. Each fold, about 591,555 train samples and 295,779 test samples
4. Training set contains 265,239 users and 6,780 items



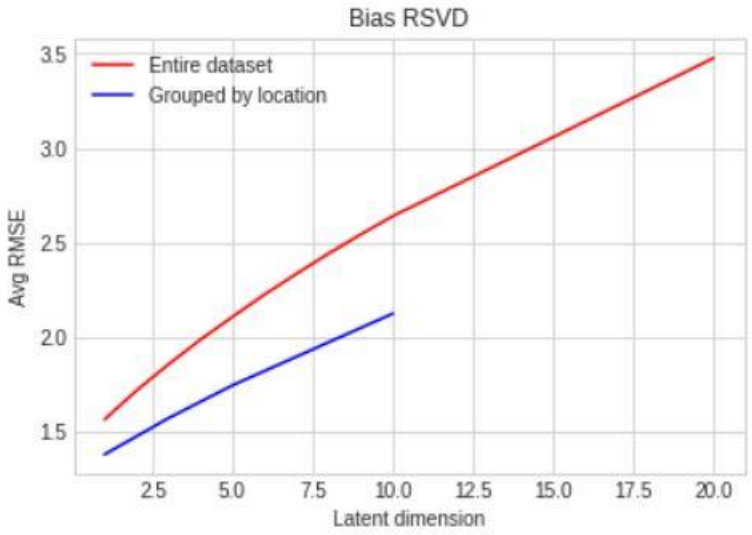
- Best latent dimension is 1 with regularization 0.1
- Compared time complexity with different latent dimension, overall, latent dimension is no a key factor for time cost.

For each fold:

Compared with Entire dataset (no consideration about location)

About 3,600,000 training samples and 1,200,000 test samples

1. Training set contains 1,144,371 users and 151,380 items
2. Data grouped by location and category is more precise



For training model: Location specific to state Nevada

With latent dimension=1 and regularization=0.1

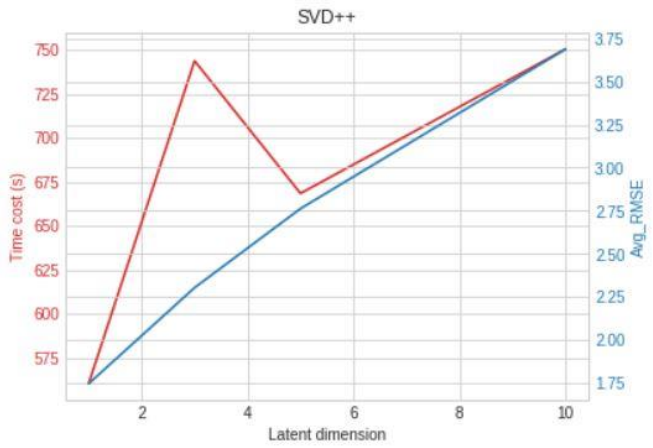
1. About 1,200,000 train samples
2. Training set contains 410,000 users and 7,400 items
3. Training RMSE loss after 20 iteration is 1.1637

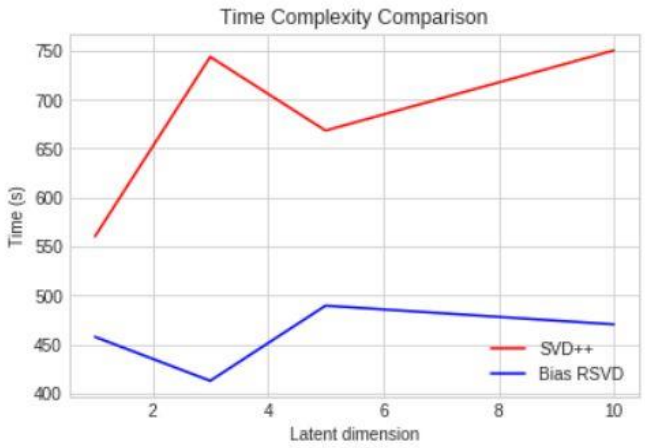
```
Running single training over training set with 1183051 train samples. Test RMSE computes RMSE on training set
Training set contains 409253 users and 7431 items
Iteration: 1    Time: 8.773871  TranRMSE: 1.697072      TestRMSE: 1.697072
Iteration: 2    Time: 23.442971 TranRMSE: 1.531071      TestRMSE: 1.531071
Iteration: 3    Time: 38.224732 TranRMSE: 1.453967      TestRMSE: 1.453967
Iteration: 4    Time: 52.540088 TranRMSE: 1.404012      TestRMSE: 1.404012
Iteration: 5    Time: 70.437762 TranRMSE: 1.366693      TestRMSE: 1.366693
Iteration: 6    Time: 86.734069 TranRMSE: 1.336905      TestRMSE: 1.336905
Iteration: 7    Time: 103.842804        TranRMSE: 1.312265      TestRMSE: 1.312265
Iteration: 8    Time: 121.146942        TranRMSE: 1.291384      TestRMSE: 1.291384
Iteration: 9    Time: 138.904573        TranRMSE: 1.273369      TestRMSE: 1.273369
Iteration: 10   Time: 155.498472        TranRMSE: 1.257604      TestRMSE: 1.257604
Iteration: 11   Time: 171.440297        TranRMSE: 1.243650      TestRMSE: 1.243650
Iteration: 12   Time: 186.916603        TranRMSE: 1.231181      TestRMSE: 1.231181
Iteration: 13   Time: 203.774055        TranRMSE: 1.219948      TestRMSE: 1.219948
Iteration: 14   Time: 220.361484        TranRMSE: 1.209759      TestRMSE: 1.209759
Iteration: 15   Time: 236.847709        TranRMSE: 1.200461      TestRMSE: 1.200461
Iteration: 16   Time: 254.006028        TranRMSE: 1.191932      TestRMSE: 1.191932
Iteration: 17   Time: 268.961496        TranRMSE: 1.184071      TestRMSE: 1.184071
Iteration: 18   Time: 284.696386        TranRMSE: 1.176796      TestRMSE: 1.176796
Iteration: 19   Time: 300.941314        TranRMSE: 1.170038      TestRMSE: 1.170038
Iteration: 20   Time: 317.075093        TranRMSE: 1.163739      TestRMSE: 1.163739
Latent 1, regularization 0.100000, average error is: 1.163739, total time cost 317.075093
```

Prediction:
Recommend items to user (key in user id)
User Id: msQe1u7Z_XuqjGoqhB0J5g (Jonathan)

Rating Score	Restaurant Name
6.470	Farm Basket
6.463	Golden Bar & Restaurant Equipment
6.286	Lisa Marie's Catering Services

Case 2: Running SVD++ with 6 machines



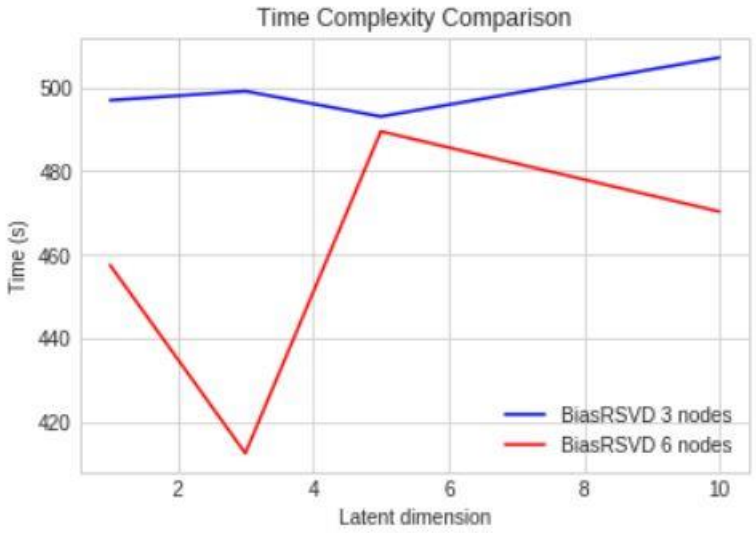


We run SVD++ with K-folds which have same sample size in Case I. Top left plot is grid search with 3-folds and 15 iteration for each fold. Top right plot is SVD++ model with best parameter from grid search. Here, we would like to compare time complexity between these two models. Also, as we expect, SVD++ takes more time and space as training.

Prediction:
Recommend items to user (key in user id)
User Id: msQe1u7Z_XuqjGoqhB0J5g (Jonathan)

Rating Score	Restaurant Name
7.05	Farm Basket
7.03	Golden Bar & Restaurant Equipment
6.87	Lisa Marie's Catering Services

Case 3: Running Bias RSVD with 3 machines



Here, we would like to compare the time cost between 3 nodes and 6 nodes used as we implement parallelism. The model is Bias-RSVD and the training data is same as Case I. As we can see that 3 nodes computation is a more time consuming than 6 nodes do. Although, the difference is quite small as we measure in each iteration, the accumulation time loss is huge.

Conclusion and Discussion:

In this project, we implement latent factor based models to implement recommendation system with Yelp dataset. With over 6 million data, we started from extracting the features that interests us, such as geography information. Then, we implemented two advanced algorithms, Bias-RSVD and SVD++, and we compare not only RMSE but also time cost for each model. Also, we used unprocessed raw data and another data grouped by geography information to train our models. We proved that people living in close area will have similar taste rather than people who live far from them.

About parallelism, overall, considering more resources from Discovery Cluster, we can improve parallel computation a lot. The consequence is shown not only on time cost for computation but also as we request more space to store temporary files for models, which happens as we call join() or other key-value functions. Take SVD++ as an example. It's more space consuming than traditional SVD and Bias RSVD because it has to create more matrix to store neighborhood model, Y and N(u) matrix. Also, because user and item dimensions are over 1 million, as we try to use 6 nodes to train a higher latent dimension model, d=50, the system will return an error "No space left on device." Therefore, space issue is another viewpoint we can think about as we want to leverage the power of distributed system.

References:

1. https://www.yelp.com/dataset/documentation/main
2. https://www.cs.rochester.edu/twiki/pub/Main/HarpSeminar/Factorization_Meets_the_Neighborhood-_a_Multifaceted_Collaborative_Filtering_Model.pdf