Movie Recommendation System

Yichao Chen(yc1228)
Yang Sun(ys2603)
Yanjia Zhang(yz855)

May 12, 2017

Abstract

Matching consumers with appropriate products based on their preferences is key to E-commerce industry. Recommendation systems are widely used to identify consumers' tastes. In this project, we use the famous MovieLens dataset[1], and apply one of the major recommendation system technology: Collaborative Filtering Matrix Factorization. We approach the methodology with two implementations: Alternating Least Squares(ALS) as well as Stochastic Gradient Descent(SGD). Also, we add user bias and item bias as an improvement of SGD methodology for further improvement. Finally, we compare each model's performance, efficiencies and challenges.

1 Introduction

Nowadays, recommendation systems are particularly useful in modern E-commerce industry. Customers are likely to comment and rate after their experiences. Targeting consumers with appropriate products based on their preferences is key to enhancing user satisfaction and loyalty. Therefore, Recommendation systems become increasingly important. In the real world, huge volume of customers' reviews and ratings for movies, musics and goods are available. Based on these data, online retailers can utilize the system to analyze patterns of users' interest in products, and provide personalized recommendations that satisfy individual user's taste.

The scope of the study will focus on the three different approaches on recommendation systems technologies. We will use different implementations and parameters to build and improve models. Also, we will consider both real-world challenges and model efficiencies when we evaluate and compare our model results.

2 Methodology

Recommendation systems have three major strategies: content-based filtering, collaborative filtering and the hybrid of content-based filtering and collaborative filtering. In this project, we will discuss collaborative filtering, which is one of the most widely used technology. Collaborative filtering analyzes the relationship between users and products with a new user-item associations. In the following paragraphs, we will mainly discuss matrix factorization (MF) that has been a dominant and successfully implemented approach among collaborative filtering.

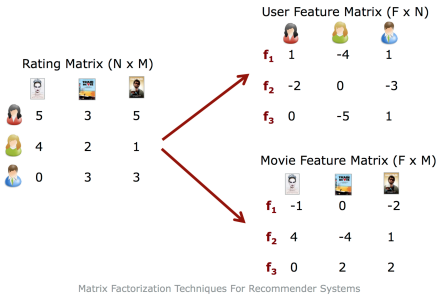
MF places different types of input data into a sparse matrix with one dimension representing a particular user and the other representing items of the user's interest. Each cell's value represents the user's rating. In general, MF maps the above matrix into a product of two lower-rank matrices: user matrix P and item matrix Q. User-item interactions are modeled as inner products that are directly related to a latent vector space of dimensionality k. As a result, MF system approximates user u's rating of item i as denoted below:

$$\hat{r_{ui}} = \sum_{j=1}^{k} p_{uj} q_{ji} \tag{1}$$

where p_{uj} indicates how user u prefers latent factor j, and q_{ji} indicates how much latent factor describes one certain movie i. To learn matrix P and Q, our approach is to minimize a regularized objective function on the set of known ratings shown below:

$$argmin \sum_{u,i} (\hat{r_{ui}} - r_{ui})^2 + \lambda(\|p_u\|^2 + \|q_i\|^2) \tag{2}$$

where the first term is the squared error between each known rating and a predicted rating, and the rest are the regularized term.



Two primary implementations to minimize the above objective function are alternating Least Squares (ALS) and Stochastic Gradient Descent (SGD). We deploy ALS model on Pyspark on clustering computing and SGD model on local machines, and we will illustrate them below.

2.1 Alternating Least Squares

Initially, we employed ALS to optimize equation (2). Because both user and item vectors are unknowns, the objective function is non-convex. However, when we keep one vector as constant, equation (2) becomes a quadratic function and can be solved using the method of least squares, and our objective function becomes a typical convex optimization problem, and answers are as follows,

$$P_u = r_{ui} Q_i (Q_i^T Q_i + \lambda I)^{-1} \tag{3}$$

$$Q_i = r_{ui} P_u (P_u^T P_u + \lambda I)^{-1} \tag{4}$$

The training process rotates and keeps one vector fixed and independent of another until the loss function converges to a certain value.The pseudocode runs as follows.

Algorithm 1 ALS Algorithm

procedure
  Initialize **P,Q**
  repeat
  for $u = 1...n$ **do**
    $P_u = (\sum_{r_{ui} \in r_u} Q_i^T Q_i + \lambda I)^{-1} \sum_{r_{ui}} r_{ui} Q_i$
  end for
  for $i = 1...m$ **do**
    $Q_i = (\sum_{r_{ui} \in r_u} P_u^T P_u + \lambda I)^{-1} \sum_{r_{ui}} r_{ui} P_i$
  end for
  until *convergence*

2.2 Stochastic Gradient Descent(SGD)

SGD approach represents a different method from ALS to optimizing the loss function. SGD loops through the training set, and takes the derivative of equation (2) with respect to each user or item vector. During each derivation, for example, set the derivative to zero respect to a user vector p_u,and recomputes a new vector p'_u in terms of p_u and other parameters. Mathematically, new user and item vectors are updated by SGD as follows,

$$p'_u = p_u + \alpha(e_{ui} q_i - \lambda p_u) \tag{5}$$

$$q'_i = q_i + \alpha(e_{ui} p_u - \lambda q_i) \tag{6}$$

Therefore, the parameters of SGD algorithm are updated proportional to the learning rate α in the opposite direction of the gradient. The algorithm could fill in the whole matrix in one epoch, but the more steps it takes, the more accurate the prediction becomes. By taking a limited number of steps, the loss function will eventually converge to a certain point. The pseudo code runs as follows.

Algorithm 2 SGD Algorithm

procedure
  Initialize **P,Q**
  repeat
  for $u = 1...n$ **do**
    for $i = 1...m$ **do**
      for $j = 1...k$ **do**
        if $r_{ui} > 0$ **then**
          $p_{uj} = p_{uj} + \alpha((r_{ui} - p_u^T q_i)q_{ji} - \lambda p_{uj})$
          $q_{ji} = q_{ji} + \alpha((r_{ui} - p_u^T q_i)p_{uj} - \lambda q_{ji})$
  end for
  until *convergence*

2.3 Stochastic Gradient Descent with Bias

While using SGD method, some biases may affect ratings' prediction significantly. For example, some users consistently overrate products they purchase, leading to higher ratings than other buyers'. Such bias of the ratings are likely to demonstrate some of the observed overrating, so it would be unwise to explain the rating value r_{ui} by using merely the product of p_u and q_i. As a result, we may add user and item biases to improve the prediction result as below,

$$\hat{r_{ui}}' = \mu + b_u + b_i + \hat{r_{ui}} \tag{7}$$

Theoretically, SGD with bias approach is similar to the classic SGD method we discussed above; b_u and b_i will be functioned as terms to improve predictive power. The modified objective function follows:

$$argmin \sum_{u,i} (\hat{r_{ui}} - r_{ui})^2 + \lambda(\|p_u\|^2 + \|q_i\|^2 + {b_u}^2 + {b_i}^2) \tag{8}$$

The pseudocode runs as follows.

Algorithm 3 SGD with Bias Term

procedure
  Initialize **P,Q**
  repeat
  for $u = 1...n$ **do**
    for $i = 1...m$ **do**
      if $r_{ui} > 0$ **then** $e_{ui} = r_{ui} - (p_u^T q_i + b_u + b_i)$
        $b_u \leftarrow b_u + \alpha(e_{ui} - \lambda b_u)$
        $b_i \leftarrow b_i + \alpha(e_{ui} - \lambda b_i)$
        for $j = 1...k$ **do**
          $p_{uj} = p_{uj} + \alpha(2e_{ui}q_{ji} - \lambda p_{uj})$
          $q_{ji} = q_{ji} + \alpha(2e_{ui}q_{ki} - \lambda q_{ki})$
  end for
  until *convergence*

Note that in practice, however, we only add one penalty coefficient λ to all regularization terms due to time constraint and limitation of computational power.More added regularization parameters will be more effective to control biases.

3 Dataset Description and Preprocessing

The dataset we use in this project is downloaded from MovieLens Latest Datasets. The original dataset has 100,004 ratings with 9,125 movies provided by 671 users between January 09, 1995 and October 16, 2016. The original dataset contains four columns: userId, movieId, rating and timestamp. The attribute userId and movieId both start at 1, and all the ratings are in a range between 1 to 5 (Figure 1).

	userId	movieId	rating	timestamp
0	1	31	2.5	1260759144
1	1	1029	3.0	1260759179
2	1	1061	3.0	1260759182
3	1	1129	2.0	1260759185
4	1	1172	4.0	1260759205

Figure 1: Original Dataset

We transform the dataset into a matrix. The row item is each user and the column item is each movie. The cell value represents the rating given by each individual user to the corresponding movie. Because all of the users have rated at least 20 movies in the dataset. The matrix has a sparsity around 6.3%. We then extract all non-zero

values out from the matrix and then randomly split 80% of these ratings into training datasets and the remaining 20% into test datasets. In ALS, we use the matrix in sparse form (Figure 2). In SGD, we transfrom the original matrix into numpy array (Figure 3).

0	859	3
0	1111	2
0	1140	1
0	1962	2
1	163	3
1	238	5
1	244	3
1	321	3
1	331	3
1	334	2

Figure 2: Input Dataset for ALS

```
array([[ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       ...,
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 4.,  0.,  0., ...,  0.,  0.,  0.],
       [ 5.,  0.,  0., ...,  0.,  0.,  0.]])
```

Figure 3: Input Dataset for SGD

4 Evaluation and Parameter Tuning

In this project, We use Root Mean Squared Error(RMSE) as evaluation method for implementations. RMSE measures the difference between the predicted value and the actual value.

$$RMSE = \sqrt{\frac{1}{n}\sum(\hat{r_{ui}} - r_{ui})^2} \tag{9}$$

4.1 ALS

As shown in Figure 4, we set $\lambda = 0$, whereas in Figure 5, we set $\lambda = 1$, with every other parameters the same, so we can prove that there is an improvement of test error when we set a reasonable regularization coefficient, and the gap in the graph is an indication of overfitting.

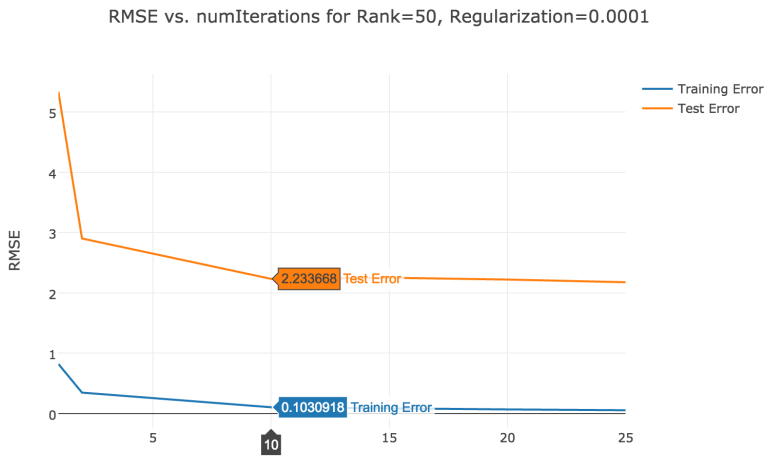


Figure 4: Indication of overfitting

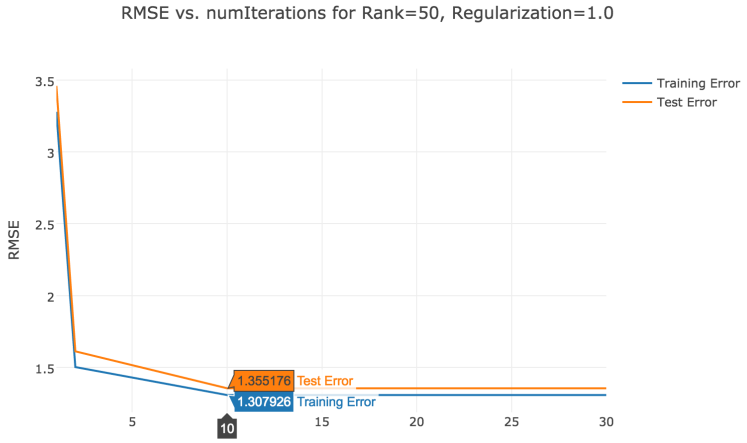


Figure 5: Increased regularization eliminates overfitting.

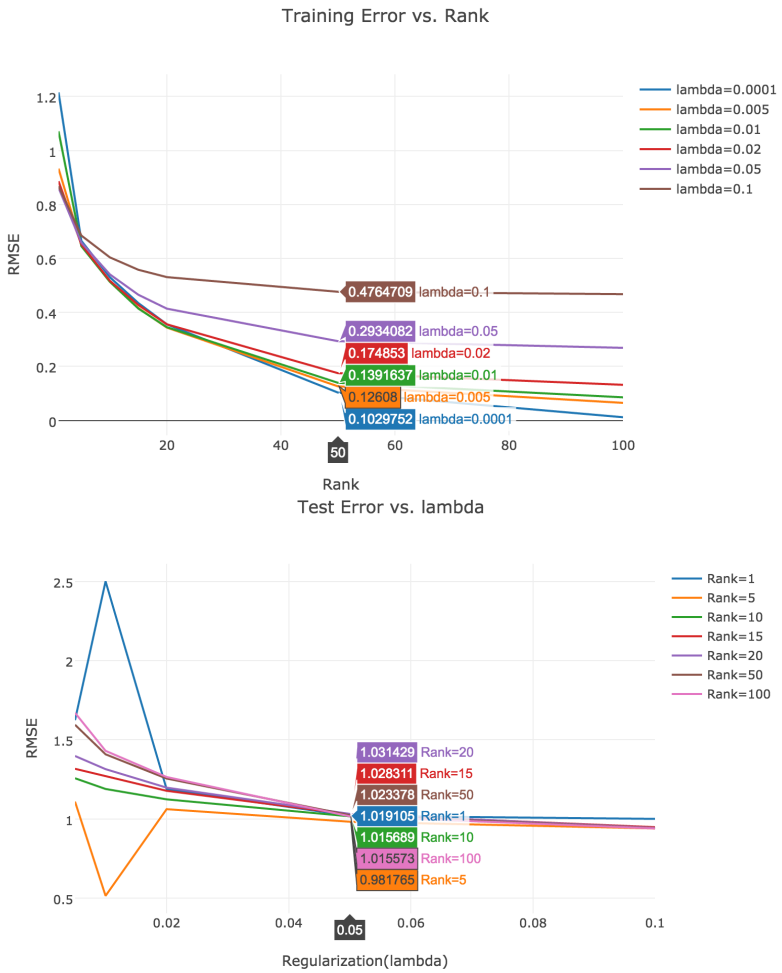


Figure 6: Training Error and Test Error for ALS

As shown in Figure 6, training error run by ALS will eventually converge to a minimal level with larger rank. With rank K = 5, λ=0.05, RMSE achieves an optimum for our dataset, 0.9817.

4.2 SGD

We run our model with SGD on training dataset, and use the equation (2) as the evaluation metric for training dataset, and RMSE as evaluation metric for test dataset. First, we try different numbers of iterations to find out the best number of steps when the loss function converges to the minimum point. Starting from step 1, the loss decreases fast as shown in Figure 7; when the algorithm approaches 250 steps, the training error decreases to 32,000 and flattens. When SGD keeps running after 250 steps, the error does not drop much. Therefore, we choose steps of 250 as our convergence point.

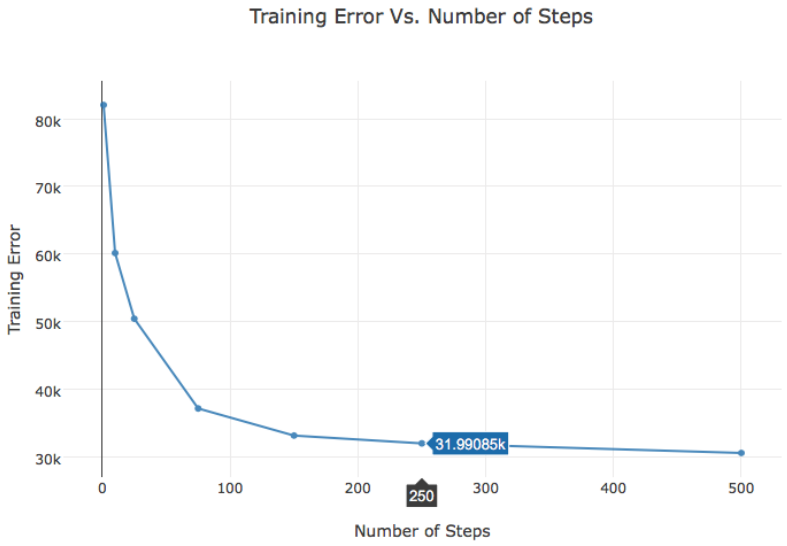


Figure 7: Training error decreases as the number of iterations increase.

Then we tune learning rate parameter: α. We keep number of steps of 250, and choose α from the following range: 0.002,0.004,0.006,0.008,0.01. The training error starts to decrease from 0.002, achieves the minimum error at 0.004, and increases again from 0.006. Therefore, we pick the α to be 0.004 in our model.

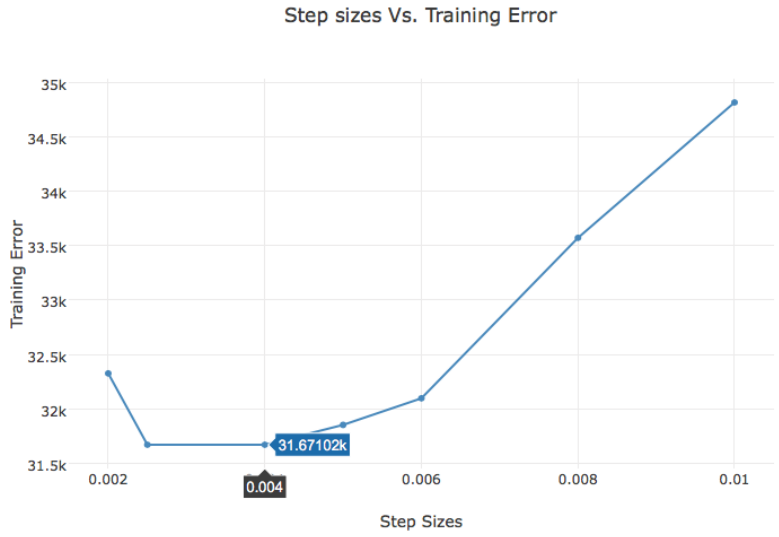


Figure 8: The plot shows training error vs. learning rate.

At next step, we tune both feature parameter K and regularization parameter λ at the same time. We choose K in the following range: 1,8,10,15,20,30,40, and λ in the following range: 1, 0.5, 0.1,0.05,0.02. Horizontal axis is number of K and vertical axis is the training error. In Figure 9, We can see on the training dataset, as K increases and λ decreases, training error decreases. As shown in Figure 10, x axis is number of K and y axis is test error; we can see when K is 15, test error drops to minimum point, and starts to increase due to overfitting. With K=15 and λ =0.5, the minimum error

we achieve is 0.95.

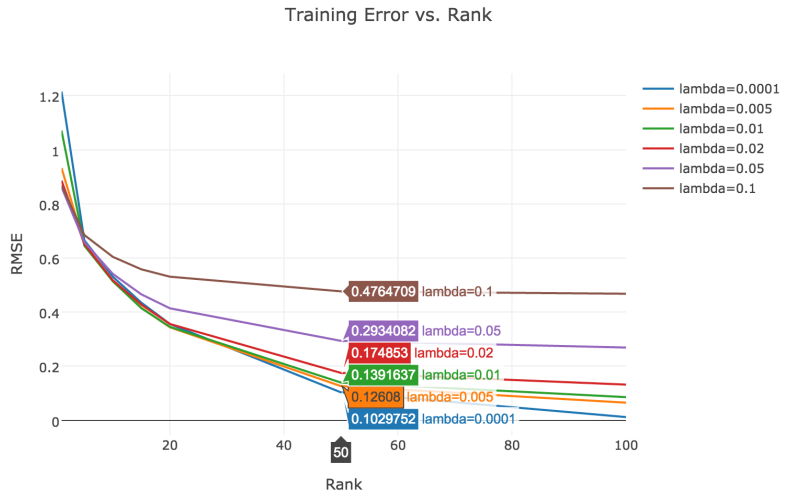


Figure 9: Training Error for SGD

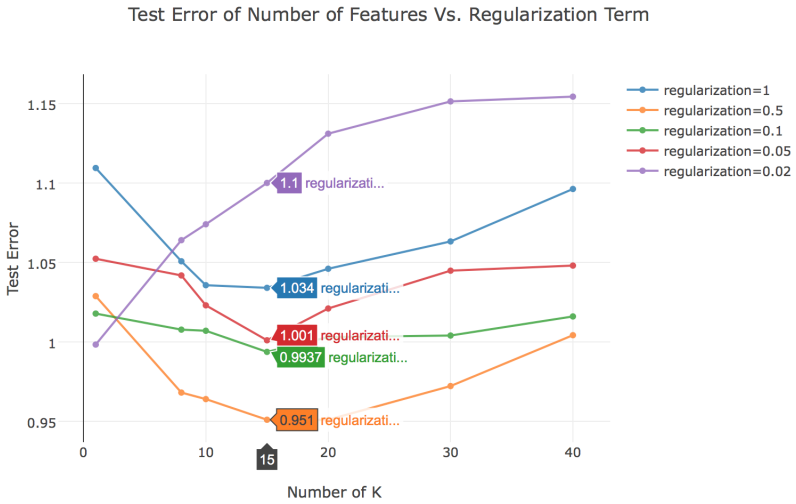


Figure 10: Test error of SGD for rank between 1 and 40.

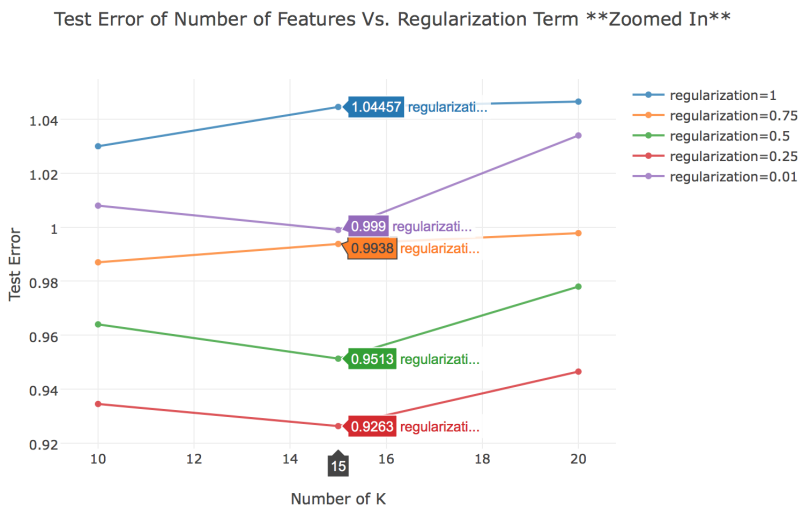


Figure 11: Test error of SGD for rank between 10 and 20.

From the above step, we realize when λ is between 0.1 and 1, and K is between 10 and 20, The test error is optimized. Therefore, we zoom in the λ range to tune our model. We choose K in the following range: 10, 15, 20, and λ in the following range: 1, 0.75, 0.5, 0.25. In Figure 11. X axis is number of K and Y axis is test error. We can see when K = 15, and $\lambda = 0.25$, we achieve the minimum test error, around 0.92.

4.3 SGD With Bias

After performing SGD, we add two bias terms in our model: user bias and item bias. Due to time constraints, we apply the best parameter from the SGD model in this SGD with bias model. In Figure 12, X axis is number of K and Y axis is test error. For each same K and same λ, the test error of SGD with bias is around 0.02 lower than the original SGD model. At K=15, the improved SGD model achieves its minimum test error: 0.90.

Figure 12: Test Error of SGD with Bias

5 Comparisons

- Compared with SGD, ALS converges faster. In our model, it takes about 10 iterations to converge for ALS, while it takes about 250 epochs to achieve a good model for SGD. A possible cause is that we need to solve two quadratic functions simultaneously in each iteration, and this parallelizable and theoretical approach guarantees that we can get the optimal point every time. In contrast, SGD randomly chooses a point in the training dataset to obtain the steepest direction towards the convergence point by calculating the gradient at each training point.
- SGD's performance is very sensitive to the choice of learning rate, so a choice of an unreasonable learning rate will yield a large training error, whereas ALS does not to choose learning rate, so choosing parameters for ALS is relatively easier.
- ALS runs much slower because of its huge computational cost. Updating each P_u will cost $O(n_i k^2 + k^3)$ where n_i is the number of movies rated by user u. Updating each q_i will cost $O(n_u k^2 + k^3)$ where n_u is the number of users who have rated movie i. After getting matrices P and Q, the computational cost for calculating the predicted rating is $O(nmk)$. This approach is tremendously expensive for large dataset. However, computational cost per iteration for SGD is only $O(nk)$, much smaller than ALS's per iteration.
- In our model SGD achieves a slightly lower test error than ALS, but this might depend on the dataset. We would better try both models on different datasets in order to generalize the result.

6 Conclusion

Model	**k**	λ	α	**N**	**RMSE**
ALS	5	0.05	n/a	10	0.98
SGD	15	0.25	0.004	250	0.92
SGD with Bias	15	0.25	0.004	250	0.90

Table 1: The performance of ALS, SGD,and SGD with Bias(k is number of latent factors; λ is regularization term; α is learning rate; N is number of iterations).

The above table illustrates the best RMSE results performed in each models: 0.98 in ALS model, 0.92 in SGD model, and 0.90 in SGD with Bias model. Therefore, in this dataset, SGD with bias model achieved the lowest error compared to the other two models. Using this model, the average difference between each predicted movie

rating and the actual movie rating is around 0.9. The entire user-movie matrix can be illustrated in Figure 13 with each rating filled:

```
array([[ 2.21838722,  1.87767994,  1.64974474, ...,  2.67569294,
         4.48138101,  3.09506292],
       [ 3.57962697,  3.09559815,  2.76539187, ...,  4.24931322,
         7.02954632,  5.11542315],
       [ 3.43383602,  2.95413693,  2.61616435, ...,  4.05326485,
         6.83537797,  4.87974085],
       ...,
       [ 3.47674657,  2.91044239,  2.52241947, ...,  4.17427163,
         6.88968817,  4.85639499],
       [ 3.67740596,  3.16273623,  2.80733477, ...,  4.39363307,
         7.26444135,  5.17595516],
       [ 3.63065019,  3.12868541,  2.77439155, ...,  4.32645402,
         7.1412422 ,  5.14696775]])
```

Figure 13

As a result, online industries are able to use predicted rating from the matrix, and decide whether they should recommend a particular movie or not to each user.

7 Challenges

7.1 Limitation of Computing Power

Computing power has become a large encounter during the project. Initially, we tried to run MF using ALS on the local machine; however, the model did not yield any result even for a relatively small dataset. Moreover, although we compared different SGD parameters, the tuning process is still time-consuming. As a result, a machine learning library on cluster computing is essential for fast learning process.

7.2 Growing Sizes of Dataset

As recommendation system becomes more and more useful in E-industry, more and more data are collected by online retailers. Thus, the growing data sizes will become a large hurdle for retailers to successfully target customers. Comparatively, SGD runs faster than ALS if the factorized dataset has much sparse data. In addition, SGD performs more accurate because it loops through every data point and takes more epochs to converge, so if a large dataset can be broken into several small ones, SGD will likely perform better than ALS.

8 Future Work

8.1 Using multiple regularization parameters

Theoretically, different penalty coefficients will have different impact on different regularization terms, thus on the objective function. As a result, with more regularization

parameters, the predictive power of the model will be improved. During the implementation of SGD and SGD with bias, however, we cannot adjust the value of penalty terms according different regularization terms because of computation limit and time constraints. We will add more penalty coefficients for future study.

8.2 Understanding latent features

Few research has studied the meaning of latent factors because they need to be interpreted by analysts' subjective opinions. Different interpretation on Extracted features, as a result, can be a powerful tool to be utilized to target customers according to their different needs[3]. As many users' information are not open for public access, we can put work on the item vectors; therefore, a more popular movie's features among audience will be grouped, and customers will be more easily targeted with the corresponding movie.

References

[1] MovieLens Latest Datasets. *https://grouplens.org/datasets/movielens/latest/.*

[2] Matrix Completion via Alternating Least Square(ALS), Haoming Li, Bangzheng He. *https://stanford.edu/ rezab/classes/cme323/S15/notes/lec14.pdf/.*

[3] Comparing ALS And SGD Over A Variable Number Of Latent Features *https://grouplens.org/datasets/movielens/latest/.*