Homework 5

Deadline: 2018.06.19 (Tuesday) 23:59

Problem: Recommender Systems [hw5.ipynb]

In this homework, you are asked to implement four different recommendation methods without using any recommendation packages. Specifically, you need to implement User-based Collaborative Filtering, Item-based Collaborative Filtering, Hybrid Method, and Latent Factor Model (Matrix Factorization) without and with regularization, global mean, user bias, and item bias. Then your tasks are to compare the performance of these typical recommendation methods with different settings.

We provide you a user-movie rating data "**ratings.data**" (a full-text file), which contains 943 users, 1682 movies, and their 100000 rating records. Each user has rated at least 20 movies. Users and items are numbered consecutively from 1. The data is randomly ordered. This is a tab separated list of

```
user id | item id | rating | timestamp.
```

The time stamps are unix seconds since 1/1/1970 UTC. You may want to convert a unix timestamp to a readable date by referring to https://bit.ly/2kwHOi3 . It is optional to use timestamps in developing a more accurate recommendation. We let recommendation with time information as a bonus.

The performance of a recommender system is evaluated by Rooted Mean Square Error (RMSE), you can import `mean_squared_error` in `sklearn.metrics` to compute RMSE. In User-based and Item-based CF, you will need to vary K, which is the number of most similar items and users, to see how K affects the performance. In addition, you need to set the percentage of training and testing as training:testing=75%:25% using `cross_validation` of `sklearn`. For example,

```
train_data, test_data = cv.train_test_split(df, test_size=0.25)
```

(1) User-based Collaborative Filtering (U-CF)

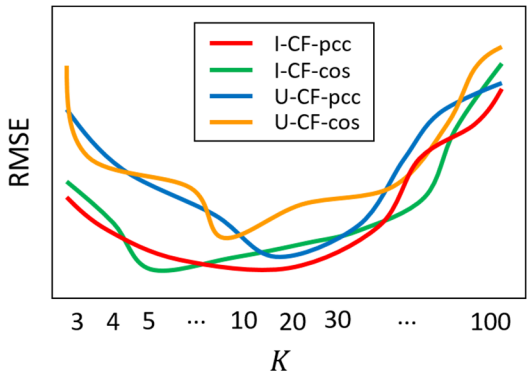
In user-based CF, your task is to use Cosine (cos) and Pearson Correlation Coefficient (pcc) as two different similarity metrics to compute the similarity between users and find the top-K similar users. So there are two methods, **U-CF-cos** and **U-CF-pcc**. After obtaining the similar users, you can generate the predicted rating by: $\hat{r}_{ui} = \frac{\sum_{v \in N} s_{uv} \cdot r_{vi}}{\sum_{v \in N} s_{uv}}$, where N is the set of K users most similar to user u who have rated item i, s_{uv} is the similarity between users u and v, r_{vi} is the rating of item i rated by user v. In the detailed settings, please consider missing ratings either as zeros. You will need to vary K to examine how K affect the performance.

(2) Item-based Collaborative Filtering (I-CF)

In item-based CF, your task is again to use Cosine (cos) and Pearson Correlation Coefficient (pcc) as two different similarity metrics to compute the similarity between users and find the top-K similar items. So there are two methods, **I-CF-cos** and **I-CF-pcc**. After obtaining the similar items, you can generate the predicted rating by: $\hat{r}_{ui} = \frac{\sum_{j \in N(i;u)} s_{ij} \cdot r_{uj}}{\sum_{j \in N(i;u)} s_{ij}}$, where $N(i;u)$ is the set of K items rated by user u most similar to item i, s_{ij} is the similarity between items i and j, r_{ui} is the rating of item i rated by user u. In the detailed settings, please consider missing ratings either as zeros. You will need to vary K to examine how K affect the performance.

[Report the Performance for (1) and (2)]

Your task is also to present the performance in terms of RMSE, in addition to the implementation of different collaborative filtering methods. By varying the value $K = 3,4,5,6,7,8,9,10,20,30,40,50,60,70,80,90,100$ as the x-axis, and putting the RMSE values in the y-axis, we can evaluate and compare how different CF methods perform in the ***testing*** user-movie pairs. You should show the results by plotting a figure like the following one. Note that this figure is only an example for illustrating how to organize the results. The performance (RMSE) curves are NOT necessarily to be the same as shown this figure.



(3) Latent Factor Model (Matrix Factorization)

Your last task is to implement the latent factor model, i.e., Matrix Factorization (MF), to predict user-movie ratings. Assume there are m users and n items. Given the rating matrix R_{train} ($m \times n$) constructed from the training data, along with missing and testing ratings in R_{train}, your goal is to find the user latent matrix P ($m \times k$) and the item latent matrix Q ($n \times k$), where k is the number of latent features/factors, such that R_{train} can be approximated: $R_{train} \approx P^T Q$. In other words, you will be able to generate the predicted rating matrix $\hat{R} = P^T Q$, i.e., $\hat{r}_{ui} = \boldsymbol{p}_u^T \boldsymbol{q}_i$, in which the predicted ratings of testing user-item pairs can be obtained. You may know that MF aims at solving the following equation:

$$\min_{P,Q} \frac{1}{2} \sum_{(u,i)\in R} (r_{ui} - \boldsymbol{p}_u^T \boldsymbol{q}_i)^2 + \frac{\lambda}{2}(\|\boldsymbol{p}_u\|^2 + \|\boldsymbol{q}_i\|^2)$$

where λ is the regularization parameter, $\boldsymbol{p}_u$ is the vector of latent features/factors for user u in user latent matrix P, and $\boldsymbol{q}_i$ is the vector of latent features/factors for item i in item latent matrix Q. $(u,i) \in R_{train}$ means all training user-item rating pairs. $\|\boldsymbol{p}_u\|$ is the L^2-norm of vector $\boldsymbol{p}_u$, e.g. given $\boldsymbol{p}_u = (1,2,3)$, $\|\boldsymbol{p}_u\| = \sqrt{1^2 + 2^2 + 3^2}$, so $\|\boldsymbol{p}_u\|^2 = 1^2 + 2^2 + 3^2$. Since neither user bias nor item bias are used here, we call this MF method as **MF-nobias**. In order to solve this equation, you are asked to **implement the Stochastic Gradient Descent (SGD) by yourself**. That says, you cannot simply rely on any package (e.g. `scipy`) to find the optimal user- and item- latent matrices P and Q. All you can use to implement SGD are `numpy` and `pandas`. Note that you may need to derive the gradients ∇p_{uk} and ∇q_{ik} by yourself.

On the other hand, you also need to implement MF with global mean μ, user bias b_u, and item bias b_i. That says, the predicted rating would be $\hat{r}_{ui} = \mu + b_u + b_i + \boldsymbol{p}_u^T \boldsymbol{q}_i$. In this case, MF aims at solving the following equation:

$$\min_{P,Q} \frac{1}{2} \sum_{(u,i)\in R} \left(r_{ui} - (\mu + b_u + b_i + \boldsymbol{p}_u^T \boldsymbol{q}_i)\right)^2 + \frac{\lambda}{2}(\|\boldsymbol{p}_u\|^2 + \|\boldsymbol{q}_i\|^2 + b_i^2 + b_u^2 + \mu^2)$$

Since both user bias and item bias are considered here, we call this MF method as **MF-bias**. To solve this equation, you are asked to implement the Stochastic Gradient Descent (SGD) by yourself. That says, you cannot simply rely on any package (e.g. `scipy`) to find the optimal user- and item- latent matrices P and Q, user bias b_u, item bias b_i, and global mean μ. All you can use to implement SGD are `numpy` and `pandas`. Note that you need to derive the gradients ∇p_{uk}, ∇q_{ik}, ∇b_u, ∇b_i, and $\nabla \mu$ by yourself before executing SGD.

Below we provide the default hyperparameter settings for Stochastic Gradient Descent (SGD):

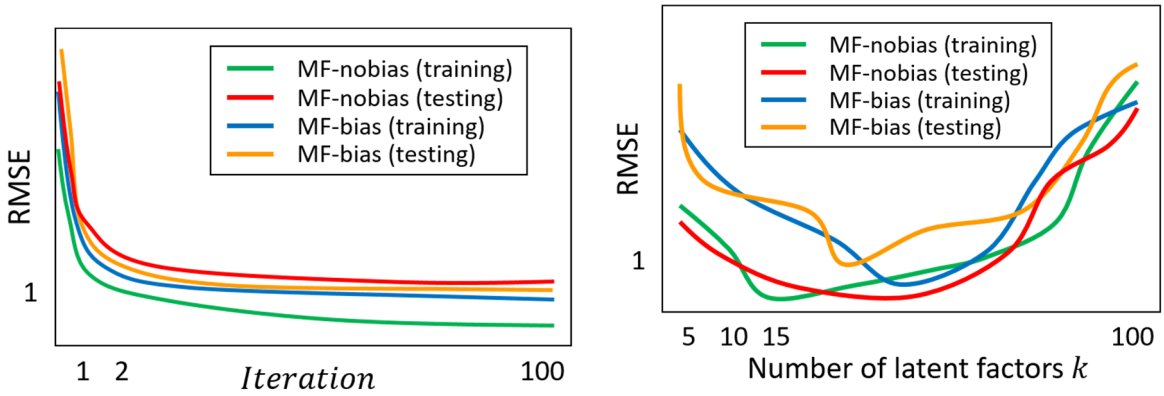
- Regularization parameter $\lambda = 0.1$
- Learning rate $\eta = 0.01$ (η is for SGD)
- Number of latent features/factors $k = 20$
- Number of iterations for SGD $= 100$

Please randomly initialize all the elements of P and Q to be between 0 and 3. In addition, please initialize μ to be the average value over all **training** user-movie non-zero ratings, and also initialize:

$$b_u = (\text{aveage of user } u\text{'s training nonzero item ratings}) - \mu$$
$$b_i = (\text{aveage of item } i\text{'s training nonzero user ratings}) - \mu.$$

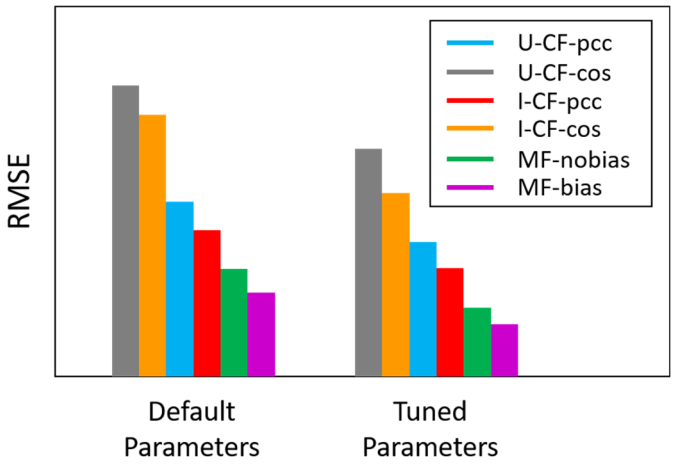
[Report the Performance for (3)]

Your task is also to present the performance in terms of RMSE using training and testing user-movie pairs, in addition to the implementation of different MF methods. First, by varying the iteration number from 1, 2, …, 100 as the x-axis, and putting the RMSE values in the y-axis, we can evaluate and compare how different CF methods perform in the ***training*** and ***testing*** user-movie pairs and how SGD coverages as the iteration number increases. Second, after the end of 100 SGD iterations, by varying the number of latent factors/features $k = 5, 10, 15, \ldots, 100$ as the x-axis, and putting the RMSE values in the y-axis, we can examine how the number of latent factors/features affect the performance in the ***training*** and ***testing*** user-movie pairs. In short, you are asked to show the results by plotting figures like the following two. Note that these figures are only examples for illustrating how to organize the results. The performance (RMSE) curves are NOT necessarily to be the same as shown these figures.



[Report the Performance for (1), (2) and (3)]

Your last task is to compare the performance of various U-CF, I-CF, and MF methods, in terms of RMSE using testing user-movie pairs. By putting all of the eight methods (U-CF-pcc, U-CF-cos, I-CF-pcc, I-CF-cos, MF-nobias, and MF-bias) in the x-axis, and putting the RMSE values in the y-axis, we can evaluate and compare how different CF and MF methods perform in the ***testing*** user-movie pairs. For the two MF methods, please make sure the SGD converges and report the RMSE values when they have converged. You should generate the results by two different settings of hyperparameters. The first set is default parameters (for CF: $K = 6$; for MF, please use the above default hyperparameter list). The second set is tuned parameters. You can set all of the hyperparameters by your eyes (i.e., your observation) and use better ones to predicting the ratings of ***testing*** user-movie pairs. Note that for the four U-CF and I-CF methods, you may want to **separately** tune and select the best K (according to the observations from previous evaluation) to generate the RMSE values. You are asked to show the results by plotting a barchart figure like the following one. Note that this figure is only an example for illustrating how to organize the results. The performance (RMSE) curves are NOT necessarily to be the same as shown this figure.



Note that in your codes for these problems,
you need to write some comments to describe the meaning of each part.

How to Submit Your Homework?

Submission in NCKU Moodle. Before submitting your homework, please zip the files (**hw5.ipynb**) in a zip file, and name the file as "學號_hw5.zip". For example, if your 學號 of your team are H12345678, then your file name is:

"H12345678_hw**5**.zip" or "H12345678_hw**5**.rar"

When you zip your files, please follow the instructions provided by TA's slides to submit your file using NCKU Moodle platform http://moodle.ncku.edu.tw .

Have Questions about This Homework?

Please feel free to visit TAs, and ask/discuss any questions in their office hours. We will be more than happy to help you.