Recommendation Systems HW1 Part2 :
Amir Wolfensohn 300339785
Yishai Shapira 203016217
Or Sharir 201314796

Part 2 - Alternating Least Squares

1. **Write the objective of a regression model with global bias, user bias, item bias and L2 regularization. Is there any difference from the SGD objective?**
   ALS objective is to estimate the complete rating matrix R~$X^T Y$, Meaning to estimate the rating each of the i=1...,N items received from each of the j=1...,M users.

   The global bias μ - its main goal is to consider that both users and items have some degree of noise when examining the rating each of the pairs produces.
   We would like that noise to be distributed between a pre- determined center. The center we use is the mean of all items rating.

   Bu (User Bias)- The objective of holding a bias for each of the users is to keep track of the average item ratings of the specific user. This is important because the scale of rating differs in different users (my rating of "3" is different from your "3")

   Bi (Item Bias)- The objective of holding a bias for each of the items is to keep track of the average rating that the item gets from all of the users. By doing that we can differentiate between the different rating scale of items ratings (5/5 rating for an item with an average rating of close to 5 is not the same as a 5/5 rating for an item with an average rating of 1/5)

   L2 regularization- Adds a "squared-magnitude penalty" to the loss function, thus helps in avoiding overfitting of the model. Lambda should be carefully defined in the regression model- if lambda will be too large we can create "under-fitting" because we will put too much weight on the penalty.

   ALS vs. SGD: the objective is the same. they both solve the same problem.
   ALS uses a different way of optimizing the loss function than SGD. by fixing U (users) and V (items), alternatively, you can solve a non-convex optimization problem easily. **The objective difference** is that in ALS you can fill-in the matrix when parallelizing the optimization of the updates for users or items (but you cannot do both at the same time). Also, when dealing with large, implicit data sets SGD run-time could make the problem practically unsolvable while ALS is much faster and much more efficient in these cases.

2. **Write the update step for each parameter**
   Each update is done on each type of parameters separately

Xu (updating $X^*_{u'}$ for each user):

$$X^*_{u'} = (\sum_{D_{u'}} y_i y^T{}_i + \lambda_x I)^{-1} * \sum_{D_{u'}} (r_{u,i} - \mu - \beta_{u'} - \beta_i) y_i$$

Where $X^*_{u'}$ is the optimal value of X for the specific user u'

Yi (updating $Y^*_{i'}$ for each item):

$$Y^*_{i'} = (\sum_{D_{i'}} x_u x^T{}_u + \lambda_y I)^{-1} * \sum_{D_{i'}} (r_{u,i} - \mu - \beta_{i'} - \beta_u) x_u$$

Where $Y^*_{i'}$ is the optimal value of Y for the specific item i'

βu (updating $\beta^*_{U'}$ for each user):

$$\beta^*_{u'} = (|D_{u'}| + \lambda I)^{-1} * \sum_{D_{u'}} (r_{u,i} - \mu - \beta_i - x_{u'}{}^T y_i)$$

Where $\beta^*_{u'}$ is the optimal value of β_u for the specific item u'

βi (updating $\beta^*_{i'}$ for each item):

$$\beta^*_{i'} = (|D_{i'}| + \lambda I)^{-1} * \sum_{D_{i'}} (r_{u,i'} - \mu - \beta_u - x_u{}^T y_{i'})$$

3. Write a pseudo code for the algorithm:

Algorithm- Alternating Least Squares:

1. **Initialize** U, I, $\lambda_u, \lambda_i, \lambda_{bu}, \lambda_{bi}$
2. **repeat**
   2.1 for u' = 1,..,M **do**

   2.1.1 $X^*_{u'} = (\sum_{D_{u'}} y_i y^T{}_i + \lambda_u I)^{-1} * \sum_{D_{u'}} (r_{u,i} - \mu - \beta_{u'} - \beta_i) y_i$

   2.2. end for
   2.3 for i'=1,...,N **do**

   2.3.1 $Y^*_{i'} = (\sum_{D_{i'}} x_u x^T{}_u + \lambda_i I)^{-1} * \sum_{D_{i'}} (r_{u,i} - \mu - \beta_{i'} - \beta_u) x_u$

   2.4 end for
   2.5. for u'=1,...,M **do**

   2.5.1 $\beta^*_{u'} = (|D_{u'}| + \lambda_{bu} I)^{-1} * \sum_{D_{u'}} (r_{u',i} - \mu - \beta_i - x_{u'}{}^T y_i)$

   2.6 end for
   2.7 for i'=1,...,N **do**

   2.7.1 $\beta^*_{i'} = (|D_{i'}| + \lambda_{bi} I)^{-1} * \sum_{D_{i'}} (r_{u,i'} - \mu - \beta_u - x_u{}^T y_{i'})$

3. **until** convergence

4. **What hyper-parameters do you need to tune?**
    a. K - the dimension of the user/item latent vectors.
    b. User/item regularisation factors
5. **Explain how would you work with the validation set and how would you check for convergence?**

By the end of each epoch we will build the estimation of the R= U X I matrix and check for the error (**RMSE**) vs the training set. then, we'll do the same vs the validation set.

> By the end of each epoch we'll compare the **training** error (**RMSE**) to the 2nd last training error saved. if the delta :
>
> (**current test error** - **2nd last error**) **< 0.001**, then the specific model is considered converged.

For each converged model we will compare its last **validation error** received with the lowest validation error **ever received** on previous converged models. For the model with the lowest **val error** we will also store its hyper-parameters K, $\boldsymbol{\lambda_x, \lambda_y}$ and additional data.

We will than rum the final model (with lowest val error parameters) on the combined dataset of both training and validation sets. when the delta of **current error** - **2nd last error <0.001** ==> the final model will be considered converged.

6. **Implement an ALS solution for the model and train it using the training and validation data. Explain the main work items you had to take.**

The number of iterations for exploring needed hyper-parameters is pre-determined by us (in the code attached we are using 20 different iterations of hyper-parameters evaluations). Then, we start a hyper-parameters random search. we'll randomly pick **K**~ discrete uni{5,40}, and $\boldsymbol{\lambda_x, \lambda_y}$~ continuous uni(0.001,0.1).

we'll then run each model until its convergence:

> Following each epoch of updating all X^*_u, Y^*_i, $**\beta^*_u$, ,$** \beta^*_i$, we will build the estimation of the R= U X I matrix and check for the error (**RMSE**) vs the training set. then, we'll do the same vs the validation set.
>
> **** please note that for the script we eventually implemented the "Basic model". We tried to implement the "full model" (biases and μ, but it yielded us with a worse score, so we stick with the simple model. All lines used for the "full model" are to be found commented in our code)**
>
> We will save the **RMSE** errors received in each epoch into two separate lists (one for training errors, other for validation errors).

By the end of each epoch we'll compare the **training** error to the 2nd last training error saved. if the delta : (current training error - 2nd last error) < 0.001, we will consider the model as converged.

When a model is considered converged we will compare its last **validation error** received with the lowest validation error ever received with previous converged models. For a model with the lowest **val error** we will also store its hyper-parameters K, $\lambda_u, \lambda_i, \lambda_{bu}, \lambda_{bi}$ and additional errors calculated.

when all iterations for the models are through we will have our desired hyper-parameters.

We'll then run the final model (with desired hyper-parameters) vs our combined **Validation + trianing set** until **current error - 2nd last error <0.001**. Then the final model is considered converged. we will then build our estimation of the R estimation Matrix.

7. What is the RMSE, MAE, R^2 and MPR of your model based on the validation set?

```
Lambda user:  0.0751674251827961
Lambda item:  0.07169854966403758
RMSE:  0.7747814500438175
MSE:  0.6002862953320005
MAE:  0.616939175314185
R^2:  0.5170806652048714
```

8. Compare the ALS and SGD solutions in terms of implementation, training, and quality

a. Implementation:

i. SGD

1. easier to implement. implementation requires going over each of the data rows and updating the parameters
2. setting our current epoch learning rate and iterating over each of the rating rows, then updating.
3. parallelization is a very complex implementation as it can overwrite other data points.
4. SGD requires "early stopping" mechanism

ii. ALS

1. implementation requires going over all of the data points for each user/item
2. involves matrix multiplication and inversion calculation for each user/item and updating each of the above for U and V, and biases.

3. parallelization is an easy, straight option to be done on one of the four Users/Items/Item biases/ user biases while flattening the Users or Items parameters.
4. The model will always get to conversion, but it might take many almost useless iterations. a threshold can be used to declare convergence beforehand.

b. Training:

i. SGD

1. Updating the parameters after each rating row iteration.
2. requires lowering the learning rate by the end of each epoch
3. The Validation error will always be bigger than the test's error.
4. We should expect convergence of the model when using the appropriate scope for the learning rate and regularizations. otherwise, the model may not converge
5. Each iteration should run faster than an ALS iteration, but the number of iterations could be higher than ALS
6. May not work well on large, implicit data sets

ii. ALS

1. Updates the optimized parameters for all users/items in every epoch and then again until convergence
2. No learning rates, and no exponential decay
3. The main difference is that you can train your model on a non-convex optimization problem by fixing Xu or Yu
4. model will always get convergence
5. works better on large data sets

c. Quality:

i. Comparing both errors outputs in the exercise we got from the two methods we'll get:

Error Calc Method	ALS	SGD
RMSE	0.77	0.94
MSE	0.6	0.88
MAE	0.62	0.74
R^2	52%	28%

As it can be seen, ALS brings back better estimations

8. Submit the test result file:

Was sent according to the instructions.