EECE 5698
Assignment 4: Matrix Factorization

Preparation. Create a folder on discovery named after your username under the directory `/gss_gpfs_scratch`. Copy the directory

```
/gss_gpfs_scratch/EECE5698/Assignment4
```

to this folder, by typing:

```
cp -r /gss_gpfs_scratch/EECE5698/Assignment3 /gss_gpfs_scratch/$USER/
```

Make the contents of this directory private, by typing:

```
chmod -R o-rx /gss_gpfs_scratch/$USER/Assignment3
```

The directory contains the following python file:

```
MFspark.py
```

as well as two directories called `small_data` and `big_data`.

The `small_data` dataset contains a synthetic ratings dataset. The `big_data` dataset contains a subset of the MovieLens dataset.[1]

You must:

1. Provide a report, in `pdf` format, outlining the answers of the questions below. The report should be type-written in a word processor of your choice (e.g., MS Word, Latex, etc.).
2. Provide the final code file `MFspark.py`.

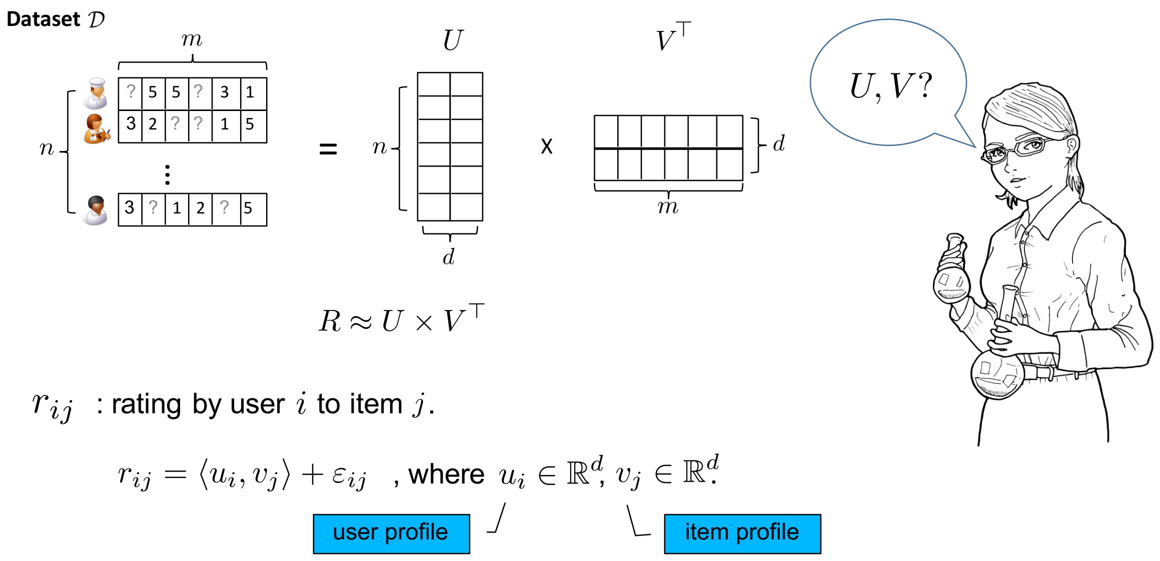
The report along with your final code should be uploaded to Blackboard. With the exeption of the bonus question, executions of the code to generate the requested output can be run on "local" mode, on a single compute node which you have reserved on the Discovery cluster. Use, e.g., a node in `ser-par-10g-3` or `ser-par-10g-4`, with 40 logical cores. To run this code, you need to increase the number of allowable threads per compute node; you can do so by typing:

```
ulimit -u 10000
```

every time that you ssh to a compute node; alternatively, you may add this line directly to your `.bashrc` file.

[1] `https://grouplens.org/datasets/movielens/`

Background: Matrix Factorization.


Dataset $\mathcal{D}$
m
n
? 5 5 ? 3 1
3 2 ? ? 1 5
3 ? 1 2 ? 5
=
U
n
d
x
$V^\top$
d
m
$U, V?$
$R \approx U \times V^\top$
r_{ij} : rating by user i to item j.
$r_{ij} = \langle u_i, v_j \rangle + \varepsilon_{ij}$, where $u_i \in \mathbb{R}^d$, $v_j \in \mathbb{R}^d$.
user profile
item profile

Matrix factorization (MF) is a technique used frequently in recommender systems [1]. In standard MF, a dataset consisting of ratings given by n users to m items (e.g., movies or songs) is provided as input. Not all users have rated all items, and the goal of an analyst is to predict the ratings of users to items not yet observed.

The underlying assumption behind matrix factorization is that ratings follow a bilinear relationship. That is, for every user $i \in \{1, \dots, n\}$ and every item $j \in \{1, \dots, m\}$, the rating $r_{ij} \in \mathbb{R}$ is given by

$$r_{ij} = u_i^\top v_j + \varepsilon_{ij}$$

where $u_i, v_j \in \mathbb{R}^d$ are "latent" d-dimensional vectors, and ε_{ij} are i.i.d. random noise variables. In other words, each user and item is characterized by a d-dimensional vector (the user and item *profiles*, respectively) such that the rating by a user to an item is, roughly, the inner product of these profiles.

Another way of stating this assumption is as follows: if $R \in \mathbb{R}^{n \times m}$ is the (completed) matrix of all ratings, then it can be written as

$$R \approx UV^\top$$

where $U = [u_i^\top]_{i \in \{1,\dots,n\}} \in \mathbb{R}^{n \times d}$ is the matrix containing all user profiles in its rows, and $V = [v_j^\top]_{j=1,\dots,m} \in \mathbb{R}^{m \times d}$ is the matrix containing all item profiles in its rows. As $\mathsf{rank}(UV^\top) = d \ll n, m$, this implies that the ratings matrix R is approximately low-rank: although its dimensions are $n \times m$, its actual rank is much smaller. The problem of finding user and item profiles is called matrix factorization as it amounts to finding the "factors" $U \in \mathbb{R}^{n \times d}, V \in \mathbb{R}^{m \times d}$, whose product recovers the full rating matrix $R \in \mathbb{R}^{n \times d}$.

References

[1] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

Question 1: Given a dataset $\mathcal{D}$ comprising tuples of the form (i, j, r_{ij}), an analyst usually constructs the user and item profiles $u_i \in \mathbb{R}^d$, $i \in \{1, \ldots, n\}$, $v_j \in \mathbb{R}^d$, $j \in \{1, \ldots, m\}$, by minimizing the following regularized square error

$$\mathsf{RSE}(U, V) = \sum_{(i,j,r_{ij}) \in \mathcal{D}} (u_i^\top v_j - r_{ij})^2 + \lambda \sum_{i=1}^{n} \|u_i\|_2^2 + \mu \sum_{j=1}^{n} \|v_j\|_2^2, \tag{1}$$

for some regularization parameters $\lambda \geq 0$, $\mu \geq 0$.

1(a) Compute the formulas of the gradients of the regularized square error w.r.t. a user profile u_i, $i \in \{1, \ldots, n\}$ and w.r.t. an item profile v_j, $j \in \{1, \ldots, m\}$, i.e.,

$$\nabla_{u_i} \mathsf{RSE}(U, V) = \left[\frac{\partial \mathsf{RSE}(U, V)}{\partial u_{ik}} \right]_{k=1}^{d} \in \mathbb{R}^d, \qquad \nabla_{v_j} \mathsf{RSE}(U, V) = \left[\frac{\partial \mathsf{RSE}(U, V)}{\partial v_{jk}} \right]_{k=1}^{d} \in \mathbb{R}^d.$$

1(b) Suppose that $d = 1$. Show that function $\ell : \mathbb{R}^2 \to \mathbb{R}$ defined as $\ell(u, v) = (u^\top v - r)^2$ is **not** a convex function. **Hint:** use the second order condition at $u = v = 0$.

1(c) Question 1(b) implies that, in general, for arbitrary $d \in \mathbb{N}$, and $\lambda, \mu \in \mathbb{R}_+$, $\mathsf{RSE}(U, V)$ is **not** a convex function of U,V. Suppose that there exist two profile matrices U^*, V^* such that

$$\nabla_{u_i} \mathsf{RSE}(U^*, V^*) = \nabla_{v_j} \mathsf{RSE}(U^*, V^*) = 0 \in \mathbb{R}^d, \quad \text{for all } i \in \{1, \ldots, n\} \text{ and } j \in \{1, \ldots, m\}.$$

Does this mean that (U^*, V^*) minimizes the regularized square error RSE?

1(d) Show that if $u_i = v_j = 0 \in \mathbb{R}^d$ for all i, j, i.e., all user and item profiles are equal to the zero vector, then the corresponding matrices U, V are such that:

$$\nabla_{u_i} \mathsf{RSE}(U, V) = \nabla_{v_j} \mathsf{RSE}(U, V) = 0 \in \mathbb{R}^d, \quad \text{for all } i \in \{1, \ldots, n\} \text{ and } j \in \{1, \ldots, m\}.$$

Question 2: Despite the lack of convexity, the regularized square error (1) is still often minimized through gradient descent. That is, starting from some initial values U^0, V^0, the user and item profiles are adapted as follows

$$u_i^{k+1} = u_i^k - \gamma^k \nabla_{u_i} \mathsf{RSE}(U^k, V^k), \quad \text{for all } i \in \{1, \dots, n\}, \tag{2a}$$

$$v_i^{k+1} = v_i^k - \gamma^k \nabla_{v_i} \mathsf{RSE}(U^k, V^k), \quad \text{for all } j \in \{1, \dots, m\}, \tag{2b}$$

where $\gamma^k > 0$ is a gain/step size (e.g., constant, decreasing, etc.)

2(a) Given the user and item profiles U^k, V^k at iteration k, let

$$\delta_{ij}^k = \langle u_i^k, v_j^k \rangle - r_{ij}, \quad \text{for all } i \in \{1, \dots, n\}, j \in \{1, \dots, m\},$$

be the difference between the predicted and actual ratings. Express

$$\mathsf{RSE}(U^k, V^k), \nabla_{u_i} \mathsf{RSE}(U^k, V^k), \text{ and } \nabla_{v_i} \mathsf{RSE}(U^k, V^k),$$

in terms of u_i^k, v_i^k and δ_{ij}^k, as opposed to r_{ij}.

2(b) Implement functions `predict`, `pred_diff`, `gradient_u`, and `gradient_v` in `MFspark.py`. Include the code that you wrote in your report.

2(c) Implement function `generateItemProfiles` in `MFspark.py`, making sure that you set the number of partitions to `N` in all operations that involve shuffles. Include the code that you wrote in your report. Why do we initialize the user and item profiles to random values? What would happen if we initialized all profiles to be zero vectors?

Question 4:

4(a) Implement function `joinAndPredictAll` in `MFspark.py`, making sure that you set the number of partitions to `N` in operations that involve shuffles. Include the code that you wrote in your report.

4(b) Implement functions `SE` and `normSqRDD` in `MFspark.py`; beyond looking at the comments, have a look at the `__main__` part of the file to get some insight on how these are used. Include the code that you wrote in your report.

4(c) Implement function `adaptU` and `adaptV` in `MFspark.py`, making sure that you set the number of partitions to `N` in operations that involve shuffles. Include the code that you wrote in your report. **Hint:** keys used in key-value pair RDDs must be hashable objects; `numpy` arrays are *not* hashable.

Question 5: The main part of `MFspark.py` is designed to train U and V through K-fold cross validation. It presumes that the data has allready been shuffled, partitioned into folds, and placed in a directory.

5(a) Indicate in your report the part of the code that constructs the training set and test set for each fold. Explain what each operation involved does and argue why this code correctly constructs the training and test sets for each of the K folds.

5(b) How is the step size γ^k set in each iteration? Provide the formula in your report in terms of the user specified parameters.

5(c) Run the code `MFspark.py` on the dataset included in `small_data` with the following parameters:

```
spark-submit --master local[40] --executor-memory 100G --driver-memory 100G  \
      MFspark.py small_data 5 --N 40 --gain 0.001 --pow 0.2 --maxiter 20 --d 1
```

(i.e., $\lambda = \mu = 0.0$). Produce a plot that shows the cross-validation Root Mean Square Error (RMSE) as a function of d, for d taking values between 1 and 10.

5(d) For the optimal value of d determined in question 5(c), run the same code with

```
--gain 0.1 --pow 0.0 --maxiter 5
```

What do you observe and why? Repeat this computation with

```
--gain 0.0001 --pow 1.0 --maxiter 20
```

and explain again the behavior you see.

5(e) For the optimal value of d determined in question 5(c), and for

```
--gain 0.001 --pow 0.2 --maxiter 20,
```

use cross validation to find the optimal λ and μ in the interval $[0, 50]$. You may use $\lambda = \mu$ in all experiments you do. Include a plot indicating the cross-validation RMSE as a function of $\lambda = \mu$ in your report.

Bonus Question: *This question is optional. Completing it will allow you to earn an extra 20% of the total points in this assignment.* Launch a spark-standalone cluster with a few workers, and explore the space of λ, μ and d through cross-validation for the `big-data` dataset. Report the cross-validation RMSE you observed for different values, as well as the optimal triplet, in your report.