Big Data Computing **A.Y. 2023/2024**
HW 4 **Deadline: December 22nd, 11.59pm**

Instructions. You should hand in your homeworks within the due date and time. Late deliveries will be penalized, please check homework delivery policies in the exam info.

Handing in. You should submit your work as *one zip file* containing: i) a Colab notebook corresponding to your first assignment (please include no dataset), with text explaining your implementation choices and an explanation of your results; ii) *one pdf* containing your answers and solutions to the theoretical exercise. Answers to theoretical exercises should be either typed up or hand written **very clearly** and scanned. The first option is strongly preferred.

Please deliver your homework by attaching the above zip file to an email addressed to me as follows:
To: `becchetti@diag.uniroma1.it`
Subject: `HW4 <Your last name> <Your first name> <Your Sapienza ID number>`

Typesetting in Latex. Latex is very handy for typesetting (especially math), but you need to install it. If you do not want to install Latex, you can go for Overleaf, providing an integrated, Web interface, accessible for free in its basic version (which is enough for your needs). It allows you to both type in Latex using a Web interface, and compiling your code to produce a pdf document. Overleaf's documentation also contains a tutorial on Latex essentials.

Important note. Grading of your answers to the theoretical assignments will depend on i) correctness and solidity of the arguments, ii) clarity of the presentation and iii) mathematical rigour. E.g., ill-defined quantities, missing assumptions, undefined symbols etc. are going to penalize you. Rather than writing a lot, try to write what is needed to answer and write it well.

Assignment 1.
The goal of this assignment is to use dimensionality reduction to perform clustering of movies from a Movielens dataset. In particular, we want to cluster movies based on a movie-user matrix U, such that U_{ij} is the score assigned to movie i by user j. In this case, you are requested to write a Colab notebook that at least applies the following baselines to perform this task: i) standard k-means and ii) A Truncated SVD - based embedding followed by k-means. While *you are supposed* to implement and test the two baselines above, you are free to explore further techniques or combinations thereof if you think they could reduce computation and/or improve quality of the results. These include random projections onto a few hundred dimensions (e.g., 200 - 500) followed by k-means, big data compatible versions of k-means such as mini-batch k-means. As a

test case, we will use the relatively large Movie Reviews dataset, available at `https://files.grouplens.org/datasets/rating-disposition-2023/ratings.csv`. The general question we want to answer (and your task) is:

> Does the clustering based on the above matrix U reflect a clustering of the movies into different genres?

To assess whether this is the case, we will also use information about Movies' genres to assess the semantics of the clusters that were produced (more about this in the next paragraphs).

How to proceed.

- **Important:** first of all understand the data you are going to work with. To this purpose, visit `https://grouplens.org/datasets/movielens/` and spend some time on it. Datasets and meta-data are described in the accompanying `README` files. For example, you can use `https://files.grouplens.org/datasets/movielens/ml-latest-README.html` as a reference.
- Extract the movie-user matrix from the dataset. Note that both users and movies have integer identifiers (`userId` and `movieId`).
- To perform clustering, you should decide the value of k, the number of clusters. In this case, you do not know how many topics you are supposed to find, so in part (but only in part) you should proceed by trial-and-error. Principled ways to find a reasonable tentative value for k are the following: i) if you are using k-means, you can plot inertia of the clustering you compute for increasing values of k and apply the elbow method seen in class. Inertia of a clustering is maintained in the attribute `inertia_` of the `sklearn.cluster.KMeans` object; ii) if you are using SVD, you can use the `explained_variance_` or `explained_variance_ratio_` attribute of the `sklearn.decomposition.TruncatedSVD` object to access the explained variance values of the k components (singular vectors) you kept. In this second case, you can initially set k to some predefined value (e.g., 100 to begin with, although 20 seems to already be a reasonable value for this dataset) and then plot explained variance against the number n of components for n ranging from 1 to the predefined value of k you chose. If you already notice an elbow in this interval you are done. These are simple heuristics and it should be clear that it is up to you to proceed in a sensible and feasible way.
- In order to check whether the clusters you found make some sense, you are supposed to use meta-data associated to the movies. You find the latest meta-data available in the file `movies.csv` associated to the full Movielens dataset available at `https://files.grouplens.org/datasets/movielens/ml-latest.zip`. In particular, each movie is assigned one or more IMBD genres (e.g., "Action", "Adventure", "Comedy" and so on). Please refer to the `README` page associated to the full dataset, available at `https://files.grouplens.org/datasets/movielens/`

`ml-latest-README.html` for more information. For each movie cluster, you should identify the most important genres associated to movies in that cluster. The easiest way to do this is to provide a word cloud for each cluster C, with the size of each term/genre T reflecting the frequency with which movies in cluster C are assigned genre T. To this purpose, you can use the `Wordcloud` package.

What to return. For each of the methods you identified as effective for this task (in particular, k-means and Truncated SVD + k-means) you should provide the most important genres for each cluster. For example, in the form of k word clouds if k denotes the number of clusters you chose.

Hints and suggestions. A couple of remarks:

- A (pretty small) fraction of the movies appearing in the file `ratings.csv` do not appear in the aforementioned file `movies.csv`. To address this issue, I suggest that you preprocess `ratings.csv`, removing any reviews of movies not appearing in `movies.csv`. If you proceed in a sensible way and use the right data structures, this is going to be pretty fast.
- Some IMBDB genres are "noisy" in the following sense: i) they are pervasively applied to most movies; ii) they do not identify a specific genre (e.g., "Horror"), but they rather describe a plot type. These are "Comedy", "Drama", "Thriller", "Action", "Adventure". Keeping them is fine, but at least in my experience, you get more meaningful results if you omit them. An easy way to achieve this is in the post-processing phase, by declaring them stopwords when you build word clouds associated to the different clusters. To this purpose, you can use the `stopwords` parameter of `Wordcloud`.
- Keep in mind that you will need to keep track of the association between row and column indices of the movie-user matrix U and the corresponding movies/users.

Assignment Evaluation and Grading. In general, do not focus on "how much" you are supposed to do, but try to do a good, clean job, trying to answer the general question I asked you above in a convincing way. More specifically, explain the choices you made in the Colab notebook (e.g., according to which criterion did you choose a specific value of k?), try different parameter combinations if you have multiple parameters to fix and so on. Also, use text in your Colab notebook to comment on and provide documentation for what you are doing. After the algorithmic and implementation choices you made, these are key elements I will consider when grading your work. Finally, *I expect you to be able to orally explain what you did and why on the day of discussion.*

Assignment 2.

Consider the random projection matrix

$$S = \frac{1}{\sqrt{k}}U,$$

where i) $U \in \mathbb{R}^{k \times d}$; ii) Every entry U_{ij} of U has normal distribution with 0 mean and unit variance, i.e., $U_{ij} \sim \mathcal{N}(0, 1)$; iii) The U_{ij}'s are independent. For any vector $\mathbf{v} \in \mathbb{R}^d$, let $f(\mathbf{v}) = S\mathbf{v}$. Next, consider two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$. Answer the following questions:

1. *Prove* that $\mathbb{E}\left[f(\mathbf{x})^T f(\mathbf{y})\right] = \mathbf{x}^T\mathbf{y}$.[1]
2. Assume next that vectors $\mathbf{x}$ and $\mathbf{y}$ satisfy the following: 1) $\|\mathbf{x}\|_2 = \|\mathbf{y}\|_2 = 1$; 2) the angle between $\mathbf{x}$ and $\mathbf{y}$ is θ (in radians). *Propose* an unbiased estimator of $\cos\theta$ that only uses $f(\mathbf{x})$ and $f(\mathbf{y})$ and *prove* that its expectation is equal to $\cos\theta$.[2]

Note: you should provide rigorous arguments for your answers.

[1]**Hint.** You might need the following: the expectation of a matrix whose elements are random variables is just the matrix of the expectations of its elements.

[2]**Hint.** Recall that Euclidead distance and cosine are related for vectors of unit norm.