Assignment 2. Working with Data in Python: NumPy

(Upload to Moodle and defend by **03.10.2025**)

If the assignment is uploaded and defended late,
then **–20% per task** will be deducted.
That means if everything is completed, the maximum grade will be **80 points**.

Task 1. (15 points) Recommender System for Online Courses

1. **Data preparation**

- Create (or load) a dataset: 100 users and 20 online courses.
- Generate a user–item rating matrix of size 100×20, where values are:
  - `0` — the user has not taken the course,
  - `1-5` — course rating.

2. **Building the user–item matrix**

- Store the data in a NumPy array.
- Find the average rating for each course and for each user.

3. **Recommendation system**

- Implement **user-based collaborative filtering (CF):**
  - compute cosine similarity between users,
  - choose k=5 most similar users for a given user,
  - recommend courses they took, but the target user did not.
- Implement **item-based CF:**
  - find similar courses based on ratings,
  - recommend 5 new courses.

4. **Quality evaluation**

- Split the data into training and test parts (e.g., 80/20).
- Implement precision and recall for k=5.

5. **Visualization**

- Plot a bar chart of the top-5 courses recommended for a specific user.
- (You may use `matplotlib.pyplot.bar`).

For cosine similarity, you can use:

```
def cosine_similarity(a, b):
    return np.dot(a, b) / (np.linalg.norm(a) *
np.linalg.norm(b))
```

For all users, it is convenient to vectorize via matrix multiplication.
Metrics can be calculated using boolean arrays: `np.isin` and `np.sum`.

Task 2. (20 points) Adjacency Matrix of a Graph

Context: we have a "friendship graph" between users (directed or undirected). The graph is stored as an adjacency matrix **A** of size N×N.

1. **Creating the matrix**

- Generate a random adjacency matrix A for N=6 users:
  - `A[i, j] = 1` if there is a connection between users i and j,
  - `0` otherwise.
- Make the graph undirected (the matrix must be symmetric).

2. **Vertex degrees**

- Find the degree of each user (row sums).
- Identify the user with the largest number of connections.

3. **Matrix of paths of length 2**

- Compute A^2 (via `np.dot`).
- Explain: what does element $A^2[i, j]$ mean in the context of the graph?

4. **Indirect friends**

- For each user, find the number of unique friends-of-friends (not direct friends).
- Hint: use `A² > 0` and boolean masks.

5. **Eigenvalues**

- Find the eigenvalues of A (`np.linalg.eigvals`).
- Identify the spectral radius (the largest absolute eigenvalue).

Task 3. (20 points) Working with Tensors in NumPy

Context: we have air temperature data in different cities, by day of the week and by hour of the day.

1. **Creating a tensor**

- Generate tensor **T** of shape (7, 24, 3):
    - 7 days of the week,
    - 24 hours per day,
    - 3 cities.
- Fill with random temperatures from –10 to +35.

2. **Indexing and slicing**

- Output temperatures in the **2nd city** for all days and hours.
- Output temperatures on the **1st day** for all cities.
- Find the temperature on the **3rd day**, at **12:00**, in the **2nd city**.

3. **Operations**

- Find the average weekly temperature for each city.
- Find the maximum temperature for each day (across all cities and hours).
- Find the hour with the lowest average temperature (averaged over all days and cities).

4. **Transformations**

- Reshape the tensor into a matrix (7, 72) — each day, all hours and cities in one vector.
- Find the correlation of temperatures between cities (based on averaged daily profiles).

Task 4. (20 points) Working with an Image Tensor

Use a ready-made image as source data, crop it to **64×64 pixels**.

1. **Loading and preparing data**

- Load an image (e.g., `image.jpg`).
- Read it as a NumPy array (`matplotlib.image.imread` or `PIL.Image`).
- Crop the top-left corner to shape (64, 64, 3).

2. **Indexing and simple transformations**

- Print pixel values at the center (32, 32) (all 3 channels).
- Extract the red channel (`I[:, :, 0]`).
- Invert the image: `I_inv = 255 - I`.

3. **Grayscale conversion**

- Convert the image to grayscale using:
  `Gray = 0.299*R + 0.587*G + 0.114*B`.
- Normalize to range [0, 1].
- Compute the average brightness of the image.

4. **Image augmentations**

- Flip the image horizontally.
- Rotate the image by 90°.
- Resize to (32, 32, 3) by averaging 2×2 blocks.

Task 5. (25 points) Matrix Factorization & Explainable Hybrid Recommender

Goal: implement matrix factorization (SVD / low-rank factorization), compare it with classic CF methods (user-based and item-based), and explain the obtained latent factors.

Steps

1. **Data preparation (3 pts)**

- Use the user–item matrix from Task 1 (100 users × 20 courses).
- Zeros in the matrix are treated as missing values.
- Split observations into train/test (e.g., 80/20 random nonzero values).

2. **Matrix factorization implementation**

- Implement `matrix_factorization(R, k, lr, reg, epochs)` with SGD.
- Maintain matrices P (users) and Q (courses).
- Return P, Q, and error history (RMSE).

3. **Recommendations**

- Compute predicted matrix `R_hat = P @ Q.T`.

- For a chosen user, find top-5 courses not yet taken.
- Compare recommendation lists from MF, user-based, and item-based for 5 random users.

4. **Quality evaluation**

- Compute RMSE on the test set.
- Implement precision@5 and recall@5 (relevant = ratings ≥4).
- Compare metrics of the three approaches (table or clear print).

5. **Factor interpretation**

- For each latent factor, find 3 courses with the highest weight in Q[:, j].
- Briefly assign a possible meaning to the factor (e.g., *"Python courses"*, *"theoretical subjects"*).
- Visualize Q as a heatmap (axes: courses × factors).

What to submit

- Code (`mf.py` or Jupyter Notebook).
- Error plot by epochs.
- Metrics table (RMSE, precision, recall).
- Heatmap of factors.
- Recommendation lists for three users.
- Short report (1–2 paragraphs): which method is better and why, influence of parameters (k, lr, reg).

Control Questions

1. What is a user–item matrix and how is it constructed?
2. What is the difference between value 0 and values 1–5 in the rating matrix?
3. How can you compute the average rating per row and per column in NumPy?
4. What is the idea of user-based collaborative filtering (CF)?
5. How is cosine similarity computed between users?
6. What does the parameter *k=5* mean in recommendations?
7. How does item-based CF differ from user-based CF?
8. How do you split data into train/test for CF?
9. How are precision@k and recall@k metrics calculated?
10. Why is visualization of recommendations (e.g., bar chart) useful?
11. What is an adjacency matrix of a graph?
12. How can you make an adjacency matrix symmetric?
13. How do you compute the degree of a vertex?
14. What does the element $A^2[i, j]$ represent?
15. How can you determine the "indirect friends" of a user?
16. What are the eigenvalues of a matrix?
17. What is the spectral radius?
18. Why is the spectral radius important for graph analysis?
19. What is a tensor in NumPy?
20. What is the dimensionality of the temperature tensor in the task?
21. How do you use slicing to select data for one city? For one day?
22. How do you compute the average temperature for a city over a week?
23. How do you determine the hour with the lowest average temperature?
24. Why is the tensor reshaped into a matrix of shape (7, 72)?
25. How can you calculate correlation between cities?
26. What is the shape of a NumPy array representing an RGB image?
27. How do you crop an image to the desired size?
28. How do you access the pixel value at position (32, 32)?
29. How do you extract only the red channel from an image?
30. How do you invert an image?
31. How is an image converted to grayscale? (Gray formula)
32. What does it mean to normalize an image to the range [0, 1]?
33. How do you perform horizontal flipping of an image in NumPy?
34. How do you downscale an image from (64, 64) to (32, 32)?
35. What is matrix factorization?
36. Which matrices are used in factorization (P and Q)?
37. How do you interpret the predicted matrix R_hat = P @ Q.T?
38. How do MF recommendations differ from user-based and item-based CF?
39. How is RMSE calculated and what does it indicate?
40. What do precision@5 and recall@5 mean in the context of recommendations?
41. How can latent factors be interpreted?
42. Why is it useful to plot a heatmap of the Q matrix?
43. Which hyperparameters (k, lr, reg) affect the quality of MF, and how?
44. Which method turned out to be better in your implementation and why?