Lab 1 + Project 1

1 Requirement

- Choose one of the following pattern recognition tasks to do. You are free to do more analyses other than the steps outlined below.
- Submit a brief report. Outline what you have done and summarize your findings of the experiment; it could be
  - Scientific discovery of the data from the pattern recognition method,
  - An improvement for the method,
  - Some tricky implementation details you found essential or non-intuitive.
  - ...
- Submission in group (max size = 4) is allowed.
- Deadline: 23:59, 15 Oct 2025.

1.1 Matrix Recovery for single-cell RNA sequence imputation

Background:

Single-cell RNA sequencing (scRNA-seq) is a revolutionary technique in bioinformatics, which can provide a better understanding of the function of an individual cell. It also allows to discover subtypes within seemingly similar cells; this is particularly advantageous for characterizing cancer heterogeneity.

However, scRNA-seq typically suffers from a high number of dropout events (some transcripts are not detected). Recovering the missing data based on the observations would be very valuable for many downstream analyses.

It is usual to pre-process the raw gene expression count data to form a "cell-gene" matrix Y, where the entry Y_{ij} represents the level of expression of gene j in cell i. The matrix Y has a lot of missing entries, which remain to be recovered. To accomplish the matrix recovery task, we assume that the "cell-gene" matrix Y is of low-rank. This is biologically plausible, since genes usually do not work in isolation. Researchers have also suggested that a small number of interdependent biophysical functions trigger the functioning of transcription factors, which in turns influence the expression levels of genes.

Task:

1. Download the incomplete "cell-gene" matrix `Preimplantation_cell_gene.mat` from the shared folder in our QQ group. The cells were taken from mouse preimplantation embryos. In total, the data has 317 cells and 18298 genes.

The matrix is created in Matlab. For Python users, you can load the matrix by using `scipy.io.loadmat()`.

Remark: the zero entries in the "cell-gene" matrix (denoted as Y) correspond to unobserved values. Therefore, $Y_{ij} = 0$ should not be interpreted as gene j having zero expression in cell i; rather, it indicates that the value of Y_{ij} is missing.

2. Randomly choose a fraction of non-zero matrix elements to form a test set

$$D_{\text{test}} = \{Y_{ij}^{\text{test}} \mid \text{some randomly chosen } (ij)\}$$

which will not be used in the matrix recovery algorithm. The remaining matrix elements correspond to the observation, denoted as Y^{obs}.

Thus, Y^{obs} should also have zero values in the test set entries, as if they were unobserved.

3. Assuming that the "cell–gene" matrix is low-rank, the objective is to apply a matrix recovery algorithm to recover the missing entries in Y^{obs}.

In this task, you are asked to formulate the problem as a nuclear norm regularization problem and implement the corresponding proximal gradient algorithm from scratch.

Try to experiment with different hyperparameters, particularly the nuclear norm regularization strength. What's the rank of your completed matrix $\hat{Y}$ for each setting?

4. Evaluate the quality of matrix recovery by comparing the predictions $\hat{Y}_{ij}$ to the test data D_{test} on the corresponding entries.

5. The resulting "cell-gene" matrix has quite a high dimension. Use some dimensionality reduction method to visualize the data and describe your observations. You can view each cell as a data point, and its gene expressions as its high dimensional feature vector.

1.2 Robust PCA for moving object detection

Background:

We learnt in the lecture (Part 1) that robust PCA is useful to decompose the observation matrix Y as $Y = L + S$, where L is a low-rank component and S is a sparse component. This can be applied to video surveillance.

Given a sequence of surveillance video frames, we often need to identify activities that stand out from the background. If we stack the video frames as columns of a matrix Y, then the low-rank component L represents the stationary background and the sparse component S captures the moving objects in the foreground.

Task:

1. Download the Wallflower MovedObject data set from the website:

`https://www.microsoft.com/en-us/download/details.aspx?id=54651`

or from the shared folder in our QQ group.

2. Select 20 images, such that they represent a scene with a person moving in and out. See the following figures for an example.

3. In your computer program, read each of the 20 images as a 2D gray-scale array (size of 120×160). Flatten each image array to be a 1D vector (size $120 \times 160 = 19200$). Stack all 20 vectors to form the observation matrix Y (size 19200×20).

4. The objective is solve the robust PCA problem for detecting the moving objects, by decomposing the matrix Y as $Y = L + S$, where L is a low-rank component (background) and S is a sparse component (moving objects).

   In this task, you are asked to implement the Alternating Direction Methods of Multipliers (ADMM) algorithm from scratch for solving the robust PCA problem.

   Run experiments on different regularization parameters and observe the effect of the strength of regularization. You can also try to increase the number of images.

5. To visually evaluate the performance, select a column of Y, L, S using the same column index (i.e., take `Y[:, i], L[:, i], S[:, i]`), reshape them back to 2D arrays of size 120×160. Plot the 2D arrays as images to see whether the moving object has been subtracted from the background.

1.3 Other projects about sparsity

If you are not interested in the datasets mentioned above, you can design your own project, as long as it is related to sparsity. Alternatively, you can explore other sparsity-related models, such as sparse coding or sparse neural networks.