Lab1 + Project 1

1 Requirement

- Complete one of the following pattern recognition tasks. 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, 9 Oct 2023.

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()`.

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.

3. Assuming the "cell-gene" matrix is of low-rank, use a matrix recovery algorithm to fill in the missing entries in Y. What's the rank of your completed matrix $\hat{Y}$?

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. (Optional) 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. Use a robust PCA algorithm to decompose the matrix $Y = L + S$, where L is a low-rank component and S is a sparse component. Run experiments on different regularization parameters and observe the effect of the strength of regularization.

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 Latent Dirichlet Allocation for topic modeling

Background:

We learnt in the lecture (Part 2b) that latent Dirichlet allocation (LDA) is a versatile tool for topic modeling. It can lead to many useful and funny applications.

For instance, LDA can serve as a tool of summarizing a scientific article, which can be subsequently recommended to researchers who are possbily interested in the subjects. The "citeulike-a" data set is a good source for exploring such a scenario:

`https://github.com/js05212/citeulike-a`

It includes titles and abstracts of a large number of articles (from users of the past CiteULike website), as well as some other side information.

Task:

1. Download the "citeulike-a" data from the above website or from the shared folder in our QQ group.

   Explore it to get some sense of the data set. The data set has been pre-processed, such that a limited number of words are chosen to form the vocabulary.

2. If the data set is too large for your computer, you can choose a subset of it (either randomly, or pick the articles collected by a few users), forming your corpus.

3. Run LDA on your corpus. You may need to decide (or experiment) on the number of topics and other hyper-parameters.

4. Interpret what you get from LDA.

5. (Optional) Topic Modeling is an unsupervised method, so its evaluation can be subjective and ambiguous. Explore some existing methods in the literature (e.g., the talk by Dr. Dietz and references therein: `https://topicmodels.info/ckling/tmt/part4.pdf`), discuss their rationals and limitations, and possibly apply to your experiments.

1.4 Your own project

If you are not interested in abovementioned data set, you can apply the methods to other data sets of your interests. You are also free to propose your own project based on the methods from the course.

For example, sparse coding has been a very popular and powerful method for solving many computer vision tasks. Although the performance may be less superior than the recent deep learning models, the sparse modeling approach has a more concrete mathematical foundation and is more interpretable. The monograph "Sparse Modeling for Image and Vision Processing" by J. Mairal et al. (available at `https://arxiv.org/abs/1411.3230`) provides a good summary on this subject. A companion software developed by the authors is available at

`https://github.com/getspams/spams-python`