LAHORE UNIVERSITY OF MANAGEMENT SCIENCES
Syed Babar Ali School of Science and Engineering

EE212 Mathematical Foundations for Machine Learning and Data Science
Summer Semester 2020

Laboratory 2 – EVD and SVD

Issued: Wednesday 22 July, 2020

Total Marks: 50

Contribution to Final Assessment: 1.5%

Submission: Wednesday 22 July, 2020.

Goal

The goal of this laboratory is to implement EVD and SVD algorithm on Python and introduce you to the use of SVD in image recovery using rank reduction.

Instructions

If you have any concerns, you can ask us in the live zoom session, or in the chat. Each of you has been allotted a TA/RA, so when you are done with the lab, let them know and they will mark it. It is your responsibility to ensure you get your work checked.

Name your files Task1.py, Task2.py and so on. Compress them in a **single** file and name it as LabXX_YourRollNumber. Submit this file on LMS before the deadline. No late submissions will be accepted.

Before starting, import the following libraries from python:

```
import numpy as np
import scipy
from matplotlib.image import imread
import pandas as pd
import matplotlib.pyplot as plt
import math
```

Task 1: EVD Overview (20 Marks)

In this task we will be quickly implementing Eigen Value Decomposition in Python, as you have studied in class. For the following matrix:

$$\boldsymbol{A} = \begin{pmatrix} 5 & 1 & 2 \\ 3 & 3 & 4 \\ 1 & 9 & 8 \end{pmatrix}$$

1. Find the eigen vector matrix $\boldsymbol{P}$ and diagonal matrix $\boldsymbol{D}$ such that $\boldsymbol{A} = \boldsymbol{PDP^{-1}}$.
2. Reconstruct $\boldsymbol{A}$ from this decomposition and verify you get the original matrix.

In an EVD of a matrix $\boldsymbol{A}$, the column vectors in $\boldsymbol{P}$ and the row vectors in $\boldsymbol{P^{-1}}$ corresponding to a zero eigen value, are the basis for null space of $\boldsymbol{A^*}$ and $\boldsymbol{A}$ respectively.

3. Construct a matrix $\boldsymbol{A}$ with the following eigen vectors and their respective eigen values:

$$\boldsymbol{p_1} = \begin{pmatrix} 1 \\ 2 \\ 7 \end{pmatrix}, \quad \boldsymbol{p_2} = \begin{pmatrix} 3 \\ 4 \\ 11 \end{pmatrix}, \quad \boldsymbol{p_3} = \begin{pmatrix} 1 \\ 6 \\ 9 \end{pmatrix} \quad \text{and} \quad \lambda_1 = 1, \quad \lambda_2 = 3, \quad \lambda_3 = 0$$

4. Find the basis for null space of $\boldsymbol{A^*}$ and $\boldsymbol{A}$ without using any built-in command for null space.

Task 2: SVD for low rank approximation (30 marks)

SVD is similar to EVD, but is more generalized is a sense that while EVD is defined for square matrices, SVD can be applied to rectangular matrices as well. The decomposition is represented as:

$$\boldsymbol{A} = \boldsymbol{U\Sigma V^T}$$

where $\boldsymbol{U}$ contains the left singular vectors, $\boldsymbol{V}$ contains the right singular vectors and $\boldsymbol{\Sigma}$ is a diagonal matrix containing singular values. Before we move on to low rank approximation, let us first review SVD using Python. For the following matrix:

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

1. Find $\boldsymbol{U}$, $\boldsymbol{\Sigma}$ and $\boldsymbol{V}$ such that $\boldsymbol{A} = \boldsymbol{U\Sigma V^T}$.
2. Reconstruct $\boldsymbol{A}$ from its decomposition and verify you get the same matrix.

We will now demonstrate how powerful SVD can be when we are talking about reducing the rank of a matrix. We will load an image into a matrix and perform SVD on that matrix. Instead of using the entire $\boldsymbol{U}$, $\boldsymbol{\Sigma}$ and $\boldsymbol{V}$ matrices to reconstruct the image, we will take the first r columns of $\boldsymbol{U}$, first r x r block of $\boldsymbol{\Sigma}$ and first r rows of $\boldsymbol{V^T}$ to reconstruct the image.

What we are doing is taking some portion of the matrices and trying to approximately reconstruct the image. r can be any integer from 1 to m, in the latter case we would perfectly reconstruct the image but with 0 compression. However, you will see that even for a very small value of r, compared to the dimension of the original matrix, you get a very good approximation of the image. Hence you do not need to store the entire image, which can take up a lot of space. You simply store these reduced matrices which save space and still recreate the image with good accuracy.

3. Loading the image

   - Load the 'image.jpg' into a matrix using the imread() command.
   - Convert this into a 2D matrix by averaging over the third dimension (take mean over the third dimension).
   - Convert the image into gray-scale and view it on your screen with the following command ($\boldsymbol{A}$ is the matrix you loaded the image in):

```
img=plt.imshow(A)
img.set_cmap('gray')
plt.show()
```

4. Decompose the image matrix using SVD and plot a graph of the singular values against the index. What is your interpretation of this graph?

5. Another quantity that you can define is the average cumulative sum of singular values. This is defined as:

$$C = \frac{\sum_{i=1}^{r} \sigma_i}{\sum_{j=1}^{n} \sigma_j}$$

   Plot the graph of C against r for $r = 1, 2, ..., m$ where m is the total number of columns of U and σ_i is the i-th singular value. This clearly shows that most of the information content is stored in the first few singular values, and the latter ones have negligible contribution.

6. We can exploit the fact that most of the information resides on the first few singular values and use it to reconstruct the image using reduced dimension matrices. Take the first r columns of $\boldsymbol{U}$, this is our $\boldsymbol{U_r}$ matrix. Take first r rows of $\boldsymbol{V^T}$, this is our $\boldsymbol{V_r^T}$ matrix. Now, take the first r singular values and construct a diagonal matrix, this is $\boldsymbol{\Sigma_r}$. Reconstruct the image using the equation:

$$\hat{\boldsymbol{A}} = \boldsymbol{U_r \Sigma_r V_r^T}$$

   . Do this for:

   - $r = 5$
   - $r = 10$
   - $r = 50$

You can now see that by just using the first 50 rows and columns, we can reconstruct the image with great accuracy. Now, we also just need to store matrices $\boldsymbol{U_r}$,$\boldsymbol{V_r}$ and $\boldsymbol{\Sigma_r}$ which are much smaller than the original ones. This demonstrates the powerf of singular value decomposition.