**Advance Algorithms, Project 2,**

***Image compression using SVD and dimensionality reduction using PCA***

**Part I**: The singular value decomposition of an *m×n* matrix *M* is a factorization of the form

M=UΣV\*

where *U* is a *m*×*m* unitary matrix, Σ is an *m×n* rectangular diagonal matrix with nonnegative numbers on the diagonal, and *V\** is the transpose of a unitary matrix *V*. (from Wiki) The objective of this part of the project is to understand how SVD (one of the most important matrix operations) works and the application of SVD to image compression. Note: you are not required to implement SVD; you may use any available implementation you like (please tell me the source).

To demonstrate that you understand how SVD works, you must give detail presentation orally and on paper (your report) how you’ve used SVD to process your image and the rate of compression you have achieved using SVD by varying the rank of your approximation as well as by relaxing the norm (2-norm) of your error matrix. Please use tables and/or charts to present your results. The rate of compression should be measured by (size of original image – size of approximated image)/size of the original image.

To make our grading easier, you’ll work on pgm images, starting with ascii pgm files. You need to write a demo program so that you could convert an ascii image file to a binary image file and back forth. Your program needs to use common-line arguments to separate the tasks. When you run your program with

o   Yourprogram.exe 1 image.pgm

Your program will save the necessary information in your image in ascii pgm format using binary coding. Use 2 bytes to save the wide of the image and 2 bytes to save the height of the image; use one byte for the grey scale levels; and use one byte for the grey level of each pixel. For example, if your image.pgm contains:

P2

#my image

4 5

255

234 23 12 3 45 22 11 22 11 22

1 2 3 4 5 6 7 8 9 10

Your program should save the file using 2+2+1+20=25 bytes. (Note: you will not save the comment line) Name the saved file **image\_b.pgm**

o   Yourprogram.exe 2 image\_b.pgm

Your program will reverse what you’ve done for the part 1, converting the binary file to an ascii file. Save the output as **image2.pgm**. (Your program will not be able to recover the comment line since it’s not saved.)

o   Yourprogram.exe 3 header.txt SVD.txt k

Your program will save the necessary information in your approximated image (you have to figure out what have to be saved) in a binary format. The input include the header of the original image, three matrices U, Σ, V and an integer k representing the rank of the approximation. You need to make sure the saved image can be recovered and can be viewed using any pgm viewer. Save the output as **image\_b.pgm.SVD**. (Note: your goal is to save storage.)

o   The header.txt contains 3 integers (width, height, grey scale levels). For example: 4 5 255

o   SVD.txt contains 3 matrices (i.e. 4x4+4x5+5x5 = 61 floats)

(Note: your goal is to save storage with as less error as possible. Consider what you can do to store a float with 2 bytes without loss too much precision.)

o   k is an integer. For example it could be 2.

o   Yourprogram.exe 4 image\_b.pgm.SVD

Your program will reverse what you’ve done for the part 3. In addition, you need to save your image to a file that can be viewed using any pgm viewer. Save the output image as **image\_k.pgm**. (Open it to see if the image is what you would expect.)

Use your demo program to obtain **the rate of compression for your own test cases**.

**Part II**: Principal component analysis (PCA) is a statistical procedure that uses orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components (from Wiki). PCA is a commonly use technique for dimensionality reduction for visualization of high-dimensional datasets. The objective of this part is to understand how PCA works, the connection between SVD and PCA, and the application of PCA to dimensionality reduction. You are required to pick a dataset that’s big and interesting to demonstrate your understanding of PCA. Make sure you give us the background of your data and tell me what you can conclude from the PCA results. (Note: you are not required to implement PCA; you may use any available implementation you like (please tell me the source).)

**Requirement:**

1.      Source code for your demo program (one running program)

2.      Group project presentation in class (8 min), Q&A (2 min)

3.      **Individual project report** (3/4 pages, no longer than 5 pages)

**Note:**

The project is to be done in a group of **two/three** students. But each one of you needs to submit a report and a statement stating your responsibilities in your group.

***Specific requirements:***

1.      Programs that do not meet the input and output format will be returned without a grade.

2.      You must include a README file to give the reader a high-level explanation of your implementation and how to run your program. Also included should be sample test-case results.

3.      Your ***report*** should include the following sections:

o   Describing SVD & PCA.

o   Describing **the code you implemented** to **illustrate** how SVD & PCA work (Note. For SVD and PCA, you can use any implementation, such as the ones in Matlab).

o   **Theoretical analyses and experimental run results** (with storage).

(This part is worth a lot. Please include your own test cases. Make sure I could tell you understand SVD & PCA.)

o   Discuss any insights on the problem and algorithm (efficiency, modularity, generalizability, stability, etc.).

4.      Your presentation should be prepared using Microsoft PowerPoint.

***Submission instructions:***

o   Due at the beginning of the due day.

o   A printed copy of your **individual project report.**

o   When you are ready to submit, obtain a printed copy of [Academic Integrity Pledge cover sheet](http://www.cs.uakron.edu/~duan/class/635/projects/Pledge.pdf), one copy per group and signed by every member of the group.

o   Submit an electronic copy of the program using our drop box at Springboard. You are required to zip your solution inside an archive. Follow these steps:

1. Create a folder named zippy\_1 (but use your group's name).
2. Place just the source files inside the folder.
3. Right-click on the folder and choose Send To... Compressed Folder (or use some other Zip utility to archive the entire folder). The goal is to create a zip archive named zippy\_1.zip (your group name) that contains the folder which contains the source files for your project.
4. Drop this single zipped file in your dropbox at Springboard. Each group needs to submit only one copy.

Please pay attention to the naming conventions for the submission files. These must be followed exactly in order to receive credit. Invalid submissions will need to be resubmitted with a penalty assessed.

Be sure to electronically submit your working solution before the due date! Do not submit non-working programs. The electronic submission time will be used to assess late penalties (if applicable).

***Presentation schedule:***