CSE 4345 – Homework #4

Assigned: Tuesday, February 26, 2019
Due: Tuesday, March 5, 2019 at the end of class

Note the following about the homework:

1. You must show your work to receive credit.
2. If your submission has more than one page, staple the pages. If I have to staple it, the cost is 10 points.

Assignment:

1. (solve by hand) Use the modified Gram-Schmidt orthogonalization process to find the reduced QR factorization of

$$A = \begin{bmatrix} 1 & 0 & 1 \\ -1 & 1 & 1 \\ 1 & 0 & 1 \\ -1 & 1 & 0 \end{bmatrix}$$

   Then use the QR factorization to find the least squares solution of $A\vec{x} = \vec{b}$ when

$$\vec{b} = \begin{bmatrix} 2 \\ 4 \\ 4 \\ 2 \end{bmatrix}$$

   Keep all answers exact. That is, keep something like $\sqrt{3}$ as-is instead of converting to 1.732.

2. (solve by hand) We have matrices A and B

$$A = \begin{bmatrix} 1 & 2 \\ 0 & 4 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$$

   For each, use the Power Iteration method to find the dominant eigenvalue/eigenvector with

$$\vec{x}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

   (a) Perform 4 iterations, showing $\vec{x}_i$ on each iteration.

   (b) Be clear on what the final eigenvalue and eigenvector is for each and how you arrived at that answer.

3. (solve by hand) Find the Singular Value Decomposition (SVD) of the following matrices. When the dimensions are appropriate, you can find the reduced SVD instead of the full version.

$$a) \begin{bmatrix} 3 & 5 \\ 4 & 0 \end{bmatrix}, \qquad b) \begin{bmatrix} 1 & -4 \\ -2 & 2 \\ 2 & 4 \end{bmatrix}, \quad c) \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

4. note the new submission procedure

(CS application: artificial intelligence) A popular task in artificial intelligence is to classify things; popular choices are images and documents. Many methods for performing classification have been developed; for this we will use the SVD of a matrix.

The process is as follows [Eld07]:

(a) Training samples from each of `c` classes of objects are chosen, with each object represented as a vector in $\mathbb{R}^n$ where n is the number of attributes describing each object (for example, pixels for images).

(b) For each class, create an $n \times q$ matrix where q is the number of training samples for that class.

(c) Find the SVD of each of these training matrices.

(d) Given a test sample z (which needs to be represented as a column vector), for each class calculate

$$||(I - U_k U_k^T)z||_2$$

where U_k is a matrix formed from the first k columns of the U matrix of a specific class (this assumes that U, Σ, and V are constructed such that the singular values in Σ are ordered $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$).

(e) Classify z as belonging to the class whose norm produces the smallest value.

Problem: You will create a Python program, `aiSVD.py`, that classifies digit images. On Blackboard are two files, which are a subset of the data discussed in [HTF09]:

i. `digitsTrainCombined.csv` – this is a 256×4000 matrix. Each column represents a 16×16 grayscale image of a digit. The first 400 columns are examples of zeros, the next 400 columns are examples of ones, and so forth, with the last 400 columns being examples of nines.

ii. `digitsTestCombined.csv` – this is a 256×1000 matrix. Each column represents a 16×16 grayscale image of a digit. The first 100 columns are examples of zeros, the next 100 columns are examples of ones, and so forth, with the last 100 columns being examples of nines.

For both sets of data, you will need to create the training labels. That is, in your code you will need to somehow indicate which columns represent zeros, which represent ones, and so forth. This will require some hard-coding.

If you wish to see what image d looks like, then if the entire matrix is saved as A you can use

```
plt.imshow( np.reshape(A[:,d], (16, 16)) )
plt.show()
```

assuming you have imported Numpy and matplotlib.pyplot appropriately.

Your program should

(a) Read the training and test data files from the current directory.

(b) For each digit's set of training data, produce the SVD.

(c) Then, for each digit's set of test data, determine which training data best approximates it. Do this using the first k columns of U, where $k = 1$, 5, 20, 100, and 256. Note that `np.linalg.svd()` returns U, Σ, and V^T, not V.

(d) For each value of k, report the results for each digit class and the overall results. A test image is classified correctly if the class it is assigned to matches the label for that test image.

These are my results:

```
-------------------- k = 1 --------------------
digit = 0,  right =  93, wrong =   7
digit = 1,  right = 100, wrong =   0
digit = 2,  right =  68, wrong =  32
digit = 3,  right =  87, wrong =  13
digit = 4,  right =  79, wrong =  21
digit = 5,  right =  78, wrong =  22
digit = 6,  right =  75, wrong =  25
digit = 7,  right =  87, wrong =  13
digit = 8,  right =  82, wrong =  18
digit = 9,  right =  82, wrong =  18

Totals:     right = 831, wrong = 169,  83.10%

-------------------- k = 5 --------------------
digit = 0,  right =  98, wrong =   2
digit = 1,  right = 100, wrong =   0
digit = 2,  right =  94, wrong =   6
digit = 3,  right =  92, wrong =   8
digit = 4,  right =  88, wrong =  12
digit = 5,  right =  92, wrong =   8
digit = 6,  right =  92, wrong =   8
digit = 7,  right =  95, wrong =   5
digit = 8,  right =  88, wrong =  12
digit = 9,  right =  92, wrong =   8

Totals:     right = 931, wrong =  69,  93.10%

-------------------- k = 20 --------------------
digit = 0,  right =  99, wrong =   1
digit = 1,  right = 100, wrong =   0
digit = 2,  right =  95, wrong =   5
digit = 3,  right =  98, wrong =   2
digit = 4,  right =  91, wrong =   9
digit = 5,  right =  94, wrong =   6
digit = 6,  right =  94, wrong =   6
```

```
digit = 7,  right =  99, wrong =   1
digit = 8,  right =  93, wrong =   7
digit = 9,  right =  97, wrong =   3

Totals:     right = 960, wrong =  40,  96.00%

------------------- k = 100 --------------------
digit = 0,  right =  98, wrong =   2
digit = 1,  right = 100, wrong =   0
digit = 2,  right =  98, wrong =   2
digit = 3,  right =  95, wrong =   5
digit = 4,  right =  87, wrong =  13
digit = 5,  right =  93, wrong =   7
digit = 6,  right =  85, wrong =  15
digit = 7,  right =  97, wrong =   3
digit = 8,  right =  84, wrong =  16
digit = 9,  right =  88, wrong =  12

Totals:     right = 925, wrong =  75,  92.50%

------------------- k = 256 --------------------
digit = 0,  right =  73, wrong =  27
digit = 1,  right = 100, wrong =   0
digit = 2,  right =  47, wrong =  53
digit = 3,  right =   0, wrong = 100
digit = 4,  right =  83, wrong =  17
digit = 5,  right =  96, wrong =   4
digit = 6,  right =  96, wrong =   4
digit = 7,  right =  86, wrong =  14
digit = 8,  right =  87, wrong =  13
digit = 9,  right =  91, wrong =   9

Totals:     right = 759, wrong = 241,  75.90%
```

General requirements about the Python problems:

a) As a comment in your source code, include your name.

b) The Python program should do the work. Don't perform the calculations and then hard-code the values in the code or look at the data and hard-code to this data unless instructed to do so.

c) The program should not prompt the user for values, read from files unless instructed to do so, or print things not specified to be printed in the requirements.

To submit the Python portion, do the following:

a) Create a directory using your net ID in lowercase characters plus the specific homework. For example, if your net ID is `abc1234` and the homework is `hw04`, then the directory should be named `abc1234-hw04`.

b) Place your .py files in this directory.

c) Do not submit the data files unless instructed to do so.

d) Zip the directory, not just the files within the directory. You must use the zip format and the name of the file (using the example above) will be `abc1234-hw04.zip`.

e) Upload the zip'd file to Blackboard.

References

[Eld07] Lars Eldén. *Matrix Methods in Data Mining and Pattern Recognition*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2007.

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, New York, NY, 2^{nd} edition, 2009.