SI231 - Matrix Computations, Fall 2020-21

Homework Set #5

Prof. Yue Qiu and Prof. Ziping Zhao

Name: Chengrui Zhang **Major:** Master in CS

Student No.: 20181201923318323456 **E-mail:** zhangchr@shanghaitech.edu.cn

Acknowledgements:

1) Deadline: **2020-12-07 23:59:00**
2) Submit your homework at **Gradescope**. Homework #5 contains two parts, the theoretical part the and the programming part.
3) About the the theoretical part:
   (a) Submit your homework in **Homework 5** in gradescope. Make sure that you have correctly select pages for each problem. If not, you probably will get 0 point.
   (b) Your homework should be uploaded in the **PDF** format, and the naming format of the file is not specified.
   (c) No handwritten homework is accepted. You need to use LaTeX in principle.
   (d) Use the given template and give your solution in English. Solution in Chinese is not allowed.
4) About the programming part:
   (a) Submit your codes in **Homework 5 Programming part** in gradescope.
   (b) When handing in your homework in gradescope, package all your codes into your_student_id+hw4_code.zip and upload. In the package, you also need to include a file named README.txt/md to clearly identify the function of each file. (".zip" format package rather than ".rar, .7zip" should be uploaded, solution of the results should be named according to requirements.)
   (c) Make sure that your codes can run and are consistent with your solutions.
5) **Late Policy details can be found in the bulletin board of Blackboard.**

Study Guide

This homework concerns the following topics:

- **Regularization**, see Lecture 8 of Zhao, Lecture 20 of Qiu.
- **SVD computation**, see Lecture 7 of Zhao, Lecture 18 of Qiu.
- **Iterative methods**, see Lecture 2 of Zhao, Lecture 21 of Qiu.
- **Application of SVD, PCA**, see Lecture 7 of Zhao, Lecture 19 of Qiu.

I. Least square with regularization

Problem 1. In this problem, we will learn how to solve LS when there is noise. for some $\lambda > 0$, where the term $\lambda ||\mathbf{x}||_k^k$ is added to improve the system conditioning, thereby attempting to reduce noise sensitivity. Usually, we set $k = 1$ which is called $\ell 1$ regularization, or set $k = 2$ which is called $\ell 2$ regularization.

$\ell 1$ regularized LS:

$$\mathbf{x_{LS}} = \min_{\mathbf{X} \in \mathbb{R}^n} f(\mathbf{x}), \quad f(\mathbf{x}) = ||\mathbf{Ax} - \mathbf{y}||_2^2 + \lambda ||\mathbf{x}||_1 \tag{1}$$

$\ell 2$ regularized LS:

$$\mathbf{x_{LS}} = \min_{\mathbf{X} \in \mathbb{R}^n} f(\mathbf{x}), \quad f(\mathbf{x}) = ||\mathbf{Ax} - \mathbf{y}||_2^2 + \lambda ||\mathbf{x}||_2^2 \tag{2}$$

Write programs to solve the regularized least square problem, any programming language is suitable. where $\mathbf{A} \in \mathbb{R}^{m \times n}$ is a matrix representing the predefined data set with m data samples of n dimensions (m=1000, n=210), and $\mathbf{y} \in \mathbb{R}^m$ represents the labels. The data samples are provided in the "data.txt" file, and the labels are provided in the "label.txt" file, you are supposed to load the data before solving the problem. In this problem, we set $\lambda = 0.1$ for two algorithms.

1) (**8 points**) Solve $\ell 1$ regularized LS using Majorization-Minimization method, we set $c = 1e{+}5$ in this problem. The Majorization-Minimization method for solving problem updates $\mathbf{x}$ as
$$\mathbf{x^{(k+1)}} = \mathbf{soft}(\frac{1}{c}\mathbf{A^T}(\mathbf{y} - \mathbf{Ax^{(k)}}) + \mathbf{x^{(k)}}, \lambda/c),$$
where **soft** is called the soft-thresholding operator and is defined as follows: if $\mathbf{z} = \mathbf{soft}(\mathbf{x}, \sigma)$, then $z_i = sign(x_i)\max\{|x_i| - \sigma, 0\}$.
2) (**8 points**) Solve $\ell 2$ regularized LS using gradient descent method. The gradient descent method for solving problem updates $\mathbf{x}$ as
$$\mathbf{x} = \mathbf{x} - \gamma \cdot \nabla_{\mathbf{x}} f(\mathbf{x}),$$
where γ is the step size of the gradient decent methods. We suggest that you can set $\gamma = 1e - 5$.
3) (**4 points**) Compare two methods above.
   (a) compare L0 norm (the number of non-zero elements) of $\mathbf{x_{LS}}$ computed by above two algorithms;
   (b) Compare the loss $||\mathbf{Ax} - \mathbf{y}||_2^2$ for results $\mathbf{x} = \mathbf{x_{LS}}$ of above two algorithms.

Remarks:

- The solution of the two methods should be printed in files named "sol1.txt" and "sol2.txt" and submitted in gradescope. The format should be same as the input file (210 rows plain text, each rows is a dimension of the final solution).
- Make sure that your codes are executable and are consistent with your solutions.

Solution. The code is shown in **HW5_1.ipynb**.

- 1), 2): The answer is in *sol1.txt*, *sol2.txt*, respectively.
- 3): The L0 norm of these two methods are all 210, and the loss $||Ax - y||_2^2$ for these two results are 2945.2600724980653 and 2945.5282348047917 respectively, which are very similar.

II. Computations of SVD

Problem 2. In this problem you will see singular value stability and discover computation of SVD.

1) (**4 points**) Consider a 4×4 matrix

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \\ \frac{1}{6000} & 0 & 0 & 0 \end{bmatrix},$$

find its singular values and eigenvalues. In this sub-problem you can use eigenvalue decomposition ('eig') and singular value decomposition function ('svd') in Matlab or Python.

2) (**8 points**) Compute the SVD decomposition of a 3×2 matrix by Algorithm 1

$$\mathbf{B} = \begin{bmatrix} 3 & 2 \\ 1 & 4 \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}.$$

Are the results right? If not, can you get the correct answer through the results of Algorithm 1? The In this sub-problem you can use the eigenvalue decomposition ('eig') and QR decomposition function ('qr') in Matlab or Python.

3) (**8 points**) Consider an $m \times m$ upper triangular matrix with 0.1 on the main diagonal and 1 everywhere above the diagonal. Fine the smallest singular value through the singular value decomposition function ('svd') and algorithm 1. You are required to plot two curves on a log scale for $m = 1, \ldots, 30$ and show which one is the desired result.

Algorithm 1: SVD Decomposition by $\mathbf{A}^T\mathbf{A}$

```
Input  : Thin matrix A
1 Form $\mathbf{A}^T\mathbf{A}$.
2 Compute the eigenvalue decomposition $\mathbf{A}^T\mathbf{A}$=$\mathbf{V}^T\mathbf{\Lambda}\mathbf{V}$.
3 Let $\mathbf{\Sigma}$ be an m×n diagonal matrix with diagonal entries being the nonnegative square root of diagonal
    entries of $\mathbf{\Lambda}$.
4 Solve the system $\mathbf{U\Sigma} = \mathbf{AV}$for orthogonal matrix $\mathbf{U}$ (e.g., via QR factorization).
Output: $\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}$.
```

Remarks

1) Singular values cannot be negative.
2) Please show your result and insert figures in your PDF.

Solution.

The code is shown in **'HW5_2.m'**.

- 1): The singular value of matrix A is $[3, 2, 1, \frac{1}{6000}]$, and the eigenvalue of A is $[0.1778, 0.1778i, -0.1778, -0.1778i]$

- 2): After doing the Algorithm. 1, we can get:

$$B^TB = \begin{bmatrix} 10.25 & 10.25 \\ 10.25 & 10.25 \end{bmatrix} \; V = \begin{bmatrix} -0.8481 & 0.5299 \\ 0.5299 & 0.8481 \end{bmatrix} \; \Lambda = \begin{bmatrix} 3.8455 & 0 \\ 0 & 26.6545 \end{bmatrix}$$

  Then, we can also get

$$\Sigma = \begin{bmatrix} 1.9610 & 0 \\ 0 & 5.1628 \\ 0 & 0 \end{bmatrix}$$

  And $QR = BV$:

$$Q = \begin{bmatrix} -0.7570 & -0.6364 & -0.1482 \\ 0.6484 & -0.7597 & -0.0494 \\ -0.0811 & -0.1335 & 0.9877 \end{bmatrix} \; R = \begin{bmatrix} 1.9610 & 0 \\ 0 & -5.1628 \\ 0 & 0 \end{bmatrix}$$

  If we use this algorithm, we can not directly get the result since the Σ is not invertible. Therefore, we let $U * \Sigma = QR = Q * G * \Sigma$, then we can get $U = Q * G = \begin{bmatrix} 3 & 2 \\ 1 & 4 \\ 0.5 & 0.5 \end{bmatrix}$ is the correct answer. Here,

$$G = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \; U = \begin{bmatrix} -0.7570 & 0.6364 & -0.1482 \\ 0.6484 & 0.7597 & -0.0494 \\ -0.0811 & 0.1335 & 0.9877 \end{bmatrix}$$

- 3): The figure is shown in Fig. 1, and we can find that the SVD method is better than Algorithm 1, since when singular value is small, the numerical error will affect the square root.

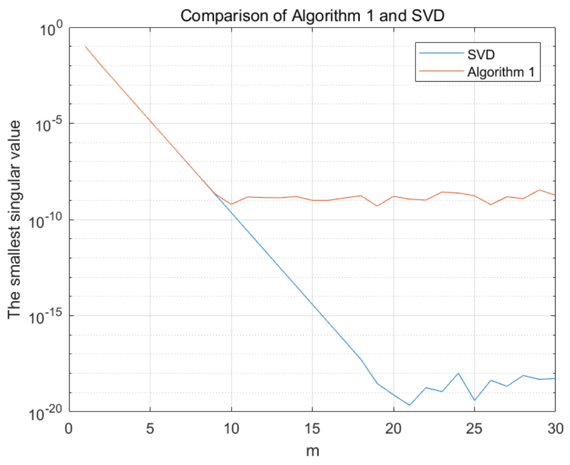

Figure 1: SVD VS Algorithm1

III. Iterative methods.

Problem 3. In this problem, you will learn how to implement there iterative methods, Jacobi, Gauss-Seidel and SOR Methods to solve the linear equation $\mathbf{b} = \mathbf{A}\mathbf{x}$, and compare the convergence of these methods. "A1.txt", "A2.txt","b1.txt", "b2.txt","x1.txt", "x2.txt" is the data of the $\mathbf{b}, \mathbf{x}, \mathbf{A}$ with different size (10 and 1000 respectively).

Algorithm 2: Iterations method

```
  Input: A starting point $\mathbf{x}^0$, maximum iterations $N$, error bound eps
  Output: Solution $\mathbf{x}$ of $\mathbf{A}\mathbf{x} = \mathbf{b}$
1 for $k = 0, 1, 2, \ldots, N$ do
2     update $\mathbf{x}^k$ to obtain $\mathbf{x}^{k+1}$ ;          // Use one of the three iteration methods
3     if $\|\mathbf{x}^{k+1} - \mathbf{x}^k\|_2 < \text{eps}$ then
4         break;
5     end
6 end
```

Suppose $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{A} = [a_{ij}] \in \mathbb{R}^{n\times n}$ and $\mathbf{b} = (b_i) \in \mathbb{R}^n$.

Each iteration method uses different updates:

For Jacobi Iteration,

$$x_i^{k+1} = (b_i - \sum_{i\neq j} a_{ij}x_j^k)/a_{ii} \quad \text{for} \quad i = 1, \ldots, n,$$

For Gauss-Seidel Iteration,

$$x_i^{k+1} = (b_i - \sum_{j>i} a_{ij}x_j^{k+1} - \sum_{j<i} a_{ij}x_j^k)/a_{ii}, \quad \text{for} \quad i = 1, \ldots, n,$$

For SOR Iteration with relaxation factor ω,

$$x_i^{k+1} = (1-\omega)x_i^k + \omega\left(b_i - \sum_{j>i} a_{ij}x_j^{k+1} - \sum_{j<i} a_{ij}x_j^k\right)/a_{ii}, \quad \text{for} \quad i = 1, \ldots, n.$$

Remarks:

1) (**8 points**) Implement the Jacobi Iteration and Gauss-Seidel Iteration methods.
2) (**8 points**) Implement the SOR Iteration method, tune the relaxation factor ω to achieve high convergence rate and plot the curve of number of iterations v.s. the value of ω.
3) (**4 points**) Plot the error $\|\mathbf{x}_{\text{true}} - \mathbf{x}^k\|_2$ for $k = 1, 2, \ldots, N$, analyze the convergence rate and the final error of three methods.

Note: The value of the “eps” in Terminal condition should not be too large!

Solution. The code is shown in **’HW5_3.m’**.

- 1): Shown in the code.

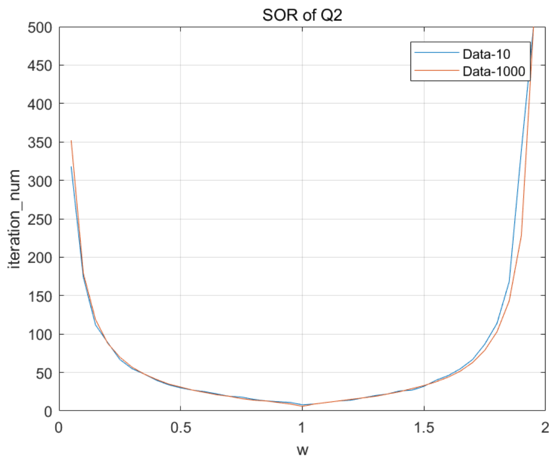

Figure 2: SOR of Q2

- 2): Shown in in Fig.2
- 3): The error Jacobi, Gauss and SOR are shown in Fig. 3, Fig.4 and (Fig.5 with w = 1 and Fig.6 with w = 0.5) respectively. We can find that the convergence rates of Jacobi and Gauss are nearly the same, while the convergence rate of SOR is highly dependent on the value of w. The final error of these methods are $5.0e-9, 2.4e-9$ and $1.6e-8$ for data-10 and $4.2e-10, 1.5e-10$ and $1.3e-8$ for data-1000 (w = 0.5 for SOR). We can find that the Gauss method is the most accurate method while the SOR has less accuracy. The accuracy of Jacobi is slightly less than the Gauss.

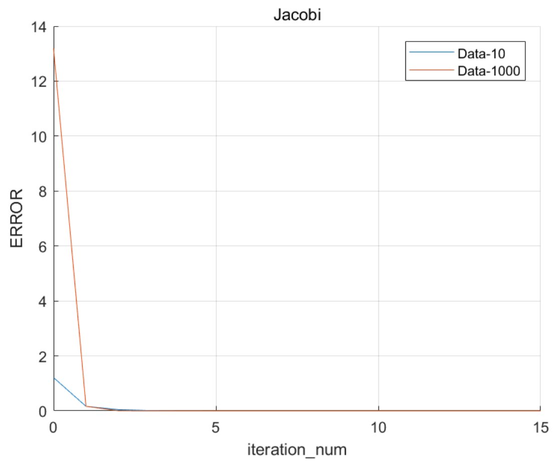

Figure 3: Jacobi

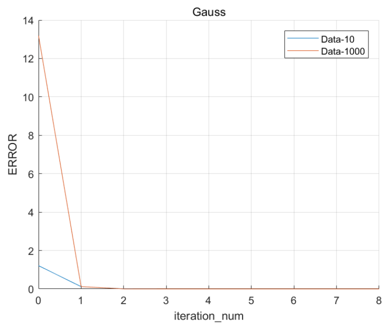

Figure 4: Gauss

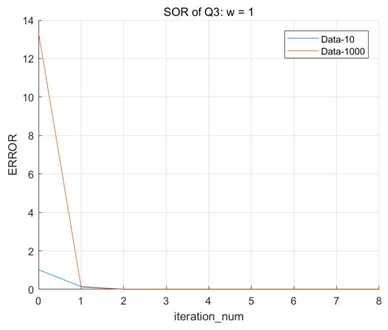

Figure 5: SOR of Q3: w = 1

IV. Application of SVD

Problem 4. In this problem you will see one of the application of SVD, what is doing face recognition on Yale B dataset. Here we will only use 4 individuals under 15 relatively brighter different illumination conditions.

Face recongition is an area of computer vision in which low-dimensional linear models such as principal component analysis (PCA) and its variations have been popular tools for capturing the variability of face images. And it has been shown that PCA can be solved by SVD optimally. In the following questions you will see how Algorithm 3 can be used.

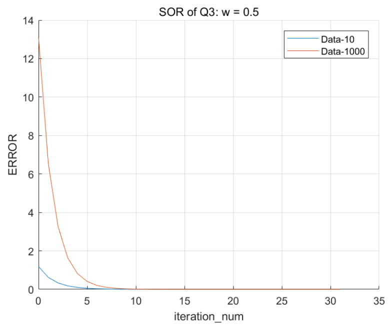

Figure 6: SOR of Q3: w = 0.5

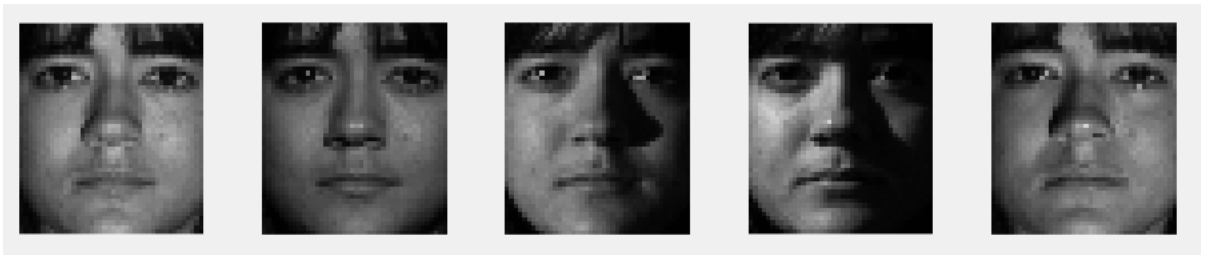

Figure 7: Face images of one individual under 5 different illumination conditions in the extended YaleB dataset. All images are frontal faces.

```
Algorithm 3: [$\boldsymbol{\mu}, \mathbf{U}, \mathbf{Y}$] = PCA_via_SVD($\mathbf{X}, d$)
1  Parameters:
2  $\mathbf{X}$:   $D \times N$ data matrix.
3  $d$:   Number of principal components.
4  Returned values:
5  $\boldsymbol{\mu}$:   Mean of the data.
6  $\mathbf{U}$:   Orthonormal basis for the subspace.
7  $\mathbf{Y}$:   Low-dimensional representation( or principal components).
8  Description:
9  Compute the SVD of the data matrix $\mathbf{X} - \boldsymbol{\mu}\mathbf{1}^T = \mathbf{U\Sigma V}^T$.
10 Sort the singular values in descending order, choose the first $d$ singular values and the corresponding $d$ left
     singular vectors $\mathbf{U}_d$.
11 Find the $d$ principal components $\mathbf{Y} = \mathbf{U}_d^T(\mathbf{X} - \boldsymbol{\mu}\mathbf{1}^T)$
```

The principal bases $\mathbf{U}$ estimated by PCA are also known as the *eigenfaces* in the computer vision literature. The first eigenface is the left singular vector corresponding to the largest singular value, the second eigenface is the left singular vector corresponding to the second largest singular value and so on.

1) (**4 points**) Apply PCA in Algorithm 3 with $d = 10$ to individual 4[1]. Plot the mean face $\boldsymbol{\mu}$, the first eigenface, the second eigenface, the third eigenface and the tenth eigenface. Describe what you have observed.
2) (**8 points**) Apply PCA in Algorithm 3 with $d = 10$ to the Training Set.
   a) Plot the mean face, the first eigenface, the second eigenface, the third eigenface and the tenth eigenface.
   b) Plot the sorted singular values.
   c) Project the Test Set onto the face subspace given by PCA, that is $\mathbf{Y}_{test} = \mathbf{U}_d^T(\mathbf{X}_{test} - \boldsymbol{\mu}\mathbf{1}^T)$. Compute the projected faces, that is Proj$(\mathbf{X}_{test}) = \boldsymbol{\mu}\mathbf{1}^T + \mathbf{U}_d\mathbf{Y}_{test}$. Then choose one projected face for each individual and plot them. And show those projected faces for $d = 2$ again.

[1]you can use the images or the mat file in data1 folder

3) (**8 points**) Classify these test faces using 1-nearest-neighbor, that is, label an image x as corresponding to individual i if its projected image y is closest to one of projected training image y_j of individual i[2]. Report the percentage of incorrectly classified face images for $d = 2, ..., 8$ and put them into the following table. Which value of d gives the best recognition performance?

Table I: Summary of errors

# eigenvectors	2	3	4	5	6	7	8
error	55%	45%	40%	25%	15%	0	0

Remarks:

1) Data are provided in two forms: images and mat data. You only need to use one of them.
2) 10 images for each individuals in the training data, and 5 images for each individuals in the test data. All images are of the same size 48×42.
3) In mat file **all_data.mat**, **data_train** and **data_test** are the matrices of training data and test data where each column is the column stack of an image[3], **Y_label_train** and **Y_label_test** are ground truth labels.
4) 'Plot the mean face or eigenface' means you should reshape the vector to the size of image and show it.
5) Recommend to use truncated(thin) SVD for fast implementation.
6) Please **insert your figures in your PDF**.

Solution.

- 1): The faces are shown in Fig. 8. We can find that the larger singular value, the clearer eigenface, if we use the 10-th singular value, the eigenface will be very hard to identify.

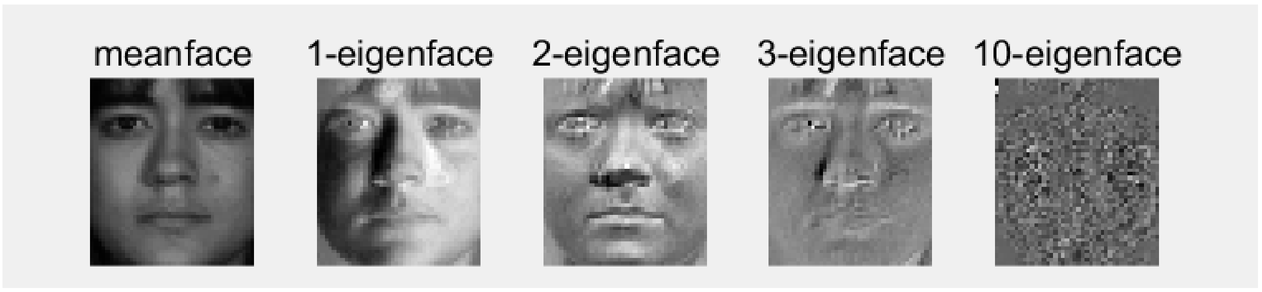

Figure 8: Each face of data1

- 2):

  a) The faces are shown in Fig. 9 b) The sorted singular values are shown in Fig: 10 c) The projected face of $d = 10$ is shown in Fig. 11 and Fig. 12 (normalization and without normalization). The normalization means that we normalize each face to $[0, 1]$ and then plot them. The projected face of $d = 2$ is shown in Fig. 13 and Fig. 14 (normalization and without normalization)
- 3) The result is shown in Table. I, and the best recognition performance is $d = 7, 8$

[2]we use ℓ_2 norm as the distance measurement: $\|y - y_j\|_2$

[3]the column stack of an image matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ is denoted by $x = [X_{11}, X_{21}, X_{31}, ..., X_{mn}]^T \in \mathbb{R}^{mn}$

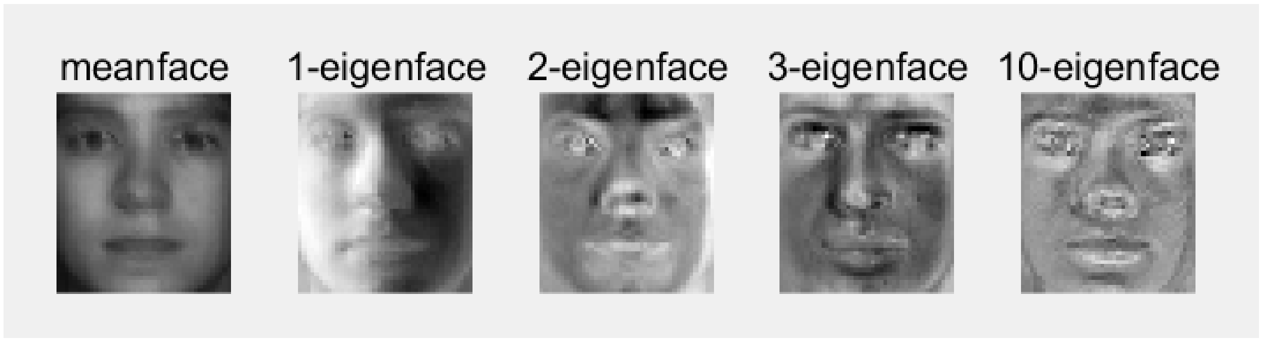

Figure 9: Each face of training set

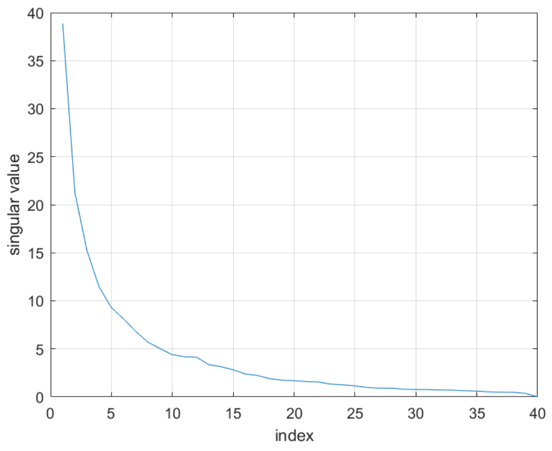

Figure 10: Sorted singular value

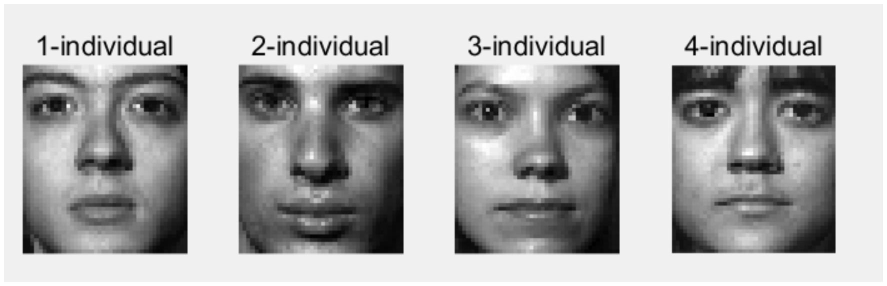

Figure 11: d = 10 with normalization

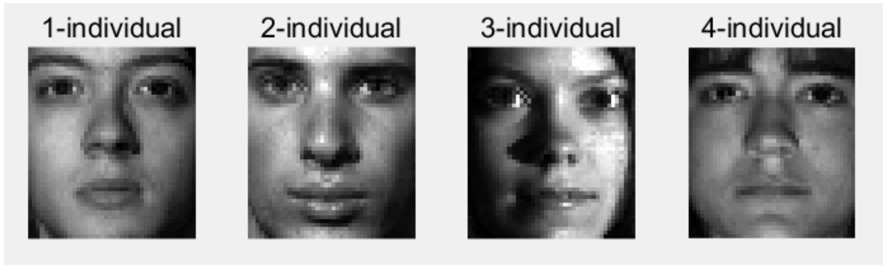

Figure 12: d = 10 without normalization

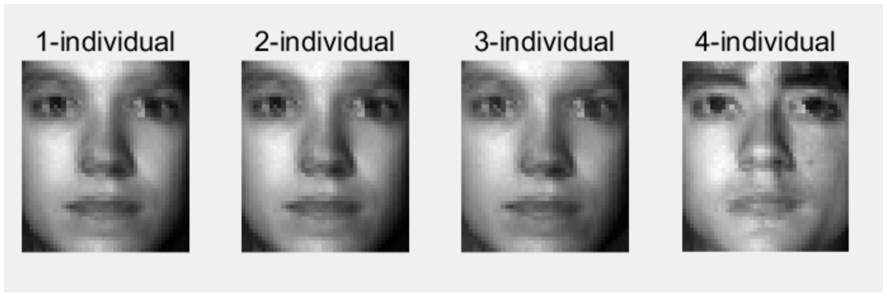

Figure 13: d = 2 with normalization

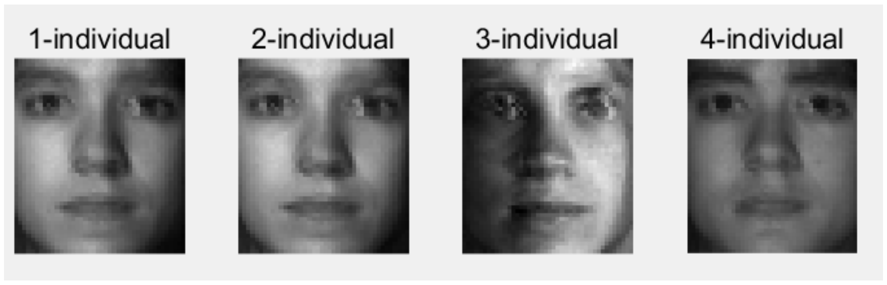

Figure 14: d = 2 without normalization

Problem 5. In this problem, we will learn how to use SVD to compress, or reduce the dimension of the data. The real data is with low rank internal, but often corrupted, or contaminated by noise, which leads to the full rank of the data matrix $\mathbf{A} \in \mathbb{R}^{D\times N}$. To reduce the storage consumption of the real data (note that D is extremely large in practical), we seek to find a low rank approximation to $\mathbf{A}$, that is, we want to solve the problem

$$\min_{\mathbf{X}\in\mathbb{R}^{D\times N}} \quad \|\mathbf{X}-\mathbf{A}\|_F^2 \tag{3}$$

$$\text{s.t.} \quad \text{rank}(\mathbf{X}) \le d \tag{4}$$

where $\|\cdot\|_F$ is the Frobenius norm, $d << D$ is unknown. By the Theorem of PCA via SVD, the above problem is equivalent to the following problem (suppose the eigenvalues of $\mathbf{A}$ are given by $\sigma_1(\mathbf{A}) \ge \sigma_2(\mathbf{A}) \ge \cdots \ge \sigma_K(\mathbf{A}) \ge 0$ and $K = \min\{D, N\}$):

$$\min_{d=1,\ldots,K} J(d) = \sum_{i=d+1}^{K} \sigma_i^2(\mathbf{A}) \tag{5}$$

However, this is not a good criterion, since the optimal solution is given by $d^* = \text{rank}(\mathbf{A})$ $(J(d^*) = 0)$.

The problem of determining the optimal dimension d is in fact a "model selection" problem, which is due to the fact that choice of d balances the complexity of the model and the storage of the data. There are many model selection criterion, we will learn two of them.

1) The first way is to bound the residual of approximation, that is, suppose $\hat{\mathbf{X}}$ is the best $\text{rank}-d$ approximation of $\mathbf{A}$, given a threshold $\tau > 0$, we want to minimize the residual with respect to Frobinius-norm, that is

$$\min_{d=1,\ldots,K} \quad \|\mathbf{A}-\hat{\mathbf{X}}\|_F^2 \le \tau \tag{6}$$

the problem can be explicitly written as

$$d^* = \min_{d=1,\ldots,K} \left\{ d \,\middle|\, \sum_{i=d+1}^{K} \sigma_i^2(\mathbf{A}) \le \tau \right\} \tag{7}$$

2) Note that the first criterion (7) depends on specific problem (singular values of data matrix are not invariant with respect to linear transformations), it is hard to chose a reasonable threshold. To generalize our criterion, we consider the normalized version of (7), this new criterion, *a.k.a* **variance explained ratio** in machine learning, is given by

$$d^* = \min_{d=1,\ldots,K} \left\{ d \,\middle|\, \frac{\sum_{i=d+1}^{K} \sigma_i^2(\mathbf{A})}{\sum_{i=1}^{K} \sigma_i^2(\mathbf{A})} \le \tau \right\} \tag{8}$$

Now, suppose the data matrix $\mathbf{A} \in \mathbb{R}^{D\times N}$ is the same as Problem 4 (data1/data1.mat). You are required to

1) (**8 points**) Plot the squared singular values of $\mathbf{A}$ along with the threshold $\tau = 150$, what is the solution to (7)?
2) (**12 points**, **6 points** for plot, **6 points** for table) Plot the figure of function

$$f(d) = \frac{\sum_{i=d+1}^{K} \sigma_i^2(\mathbf{A})}{\sum_{i=1}^{K} \sigma_i^2(\mathbf{A})}, \quad d = 1, \ldots, K \tag{9}$$

and fill the following table from the figure with respect to (8):

τ	0.1	0.05	0.02	0.005
Compression rate	0.1005	0.2011	0.3016	0.4022

Table II: The compression rate with respect different threshold

where the compression rate is defined as

$$\text{compression rate} = \frac{\#\{\text{entries in } \hat{\mathbf{X}}\}}{\#\{\text{entries in } \mathbf{A}\}} = \frac{d^*(D+N+1)}{DN} \tag{10}$$

Remarks

1) Please **insert your figures in your PDF**.
2) You can just mark the solution of problem 1) on your figure (**plot a vertical line with red color or mark the solution with marker**).
3) Please draw a continuous curve for problem 2), see *stairs* function in Matlab, *plt.step* function in Python for more details.
4) For simplicity, we omit some proof details, you may refer to *Generalized Principal Component Analysis, Section 2.1* for more details.

Solution. The code is shown in **'HW5_5.m'**

- 1): The figure is shown in Fig. 15 and Fig. 16. $d^* = 2$ when $\tau = 150$

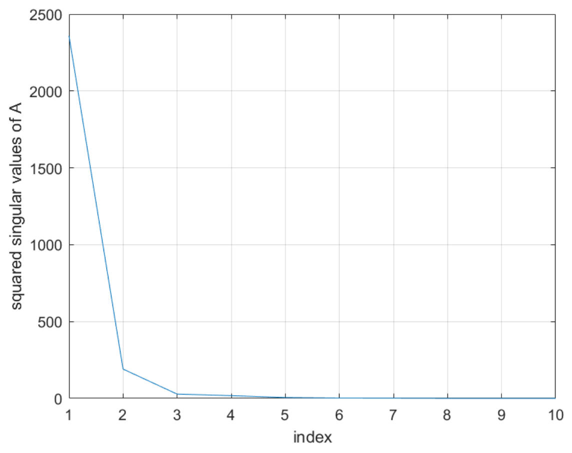

Figure 15: Squared singular values of A

- 2): The figure is shown in Fig. 17

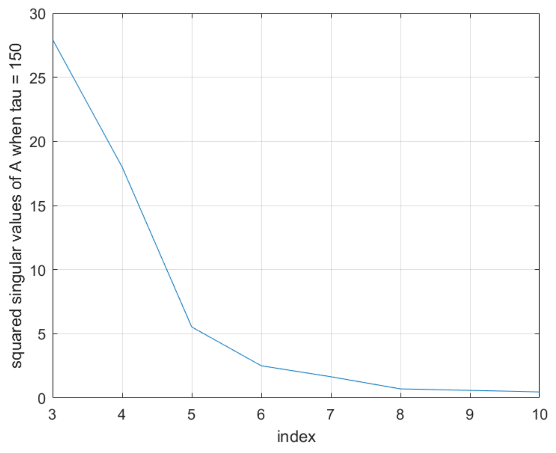

Figure 16: Squared singular values of A when tau = 150

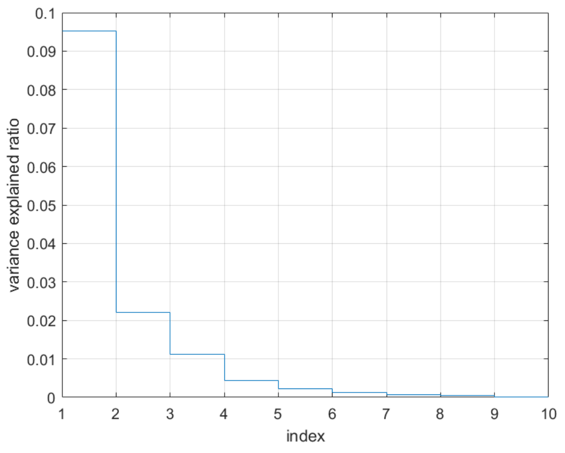

Figure 17: Function f(d)