Assignment 2a

David Boerema (s3683869)
Marios Souroulla (s4765125)
Marius Captari (s4865928)
Max Valk (s3246922)

Group 3

September 21, 2021

2.1.1

$$A = \begin{bmatrix} 3 & 4 \\ 5 & 8 \end{bmatrix} \quad B = \begin{bmatrix} 4 & 2 \\ 3 & 1 \end{bmatrix}$$

Eigen values and vectors can be calculated from:

$$\begin{aligned} Av &= \lambda v \\ Av - \lambda v &= 0 \\ (A - \lambda I) \cdot v &= 0 \\ Det(A - \lambda I) &= 0 \end{aligned}$$

Where λ is the Eigen value and v is the Eigen vector.

Matrix A first:

$$Det(A) = \begin{bmatrix} 3-\lambda & 4 \\ 5 & 8-\lambda \end{bmatrix}$$
$$(3-\lambda) \cdot (8-\lambda) - (4 \cdot 5) = 0$$
$$\lambda^2 - 11\lambda + 4 = 0$$
$$\lambda_1 = \frac{-\sqrt{105} + 11}{2}$$
$$\lambda_2 = \frac{\sqrt{105} + 11}{2}$$
$$A - \lambda_1 I = \begin{bmatrix} \frac{\sqrt{105}-5}{2} & 4 \\ 5 & \frac{\sqrt{105}+5}{2} \end{bmatrix}$$
$$A - \lambda_2 I = \begin{bmatrix} \frac{-\sqrt{105}-5}{2} & 4 \\ 5 & \frac{-\sqrt{105}+5}{2} \end{bmatrix}$$

Having both Eigen values λ_1 and λ_2 we can solve each homogeneous system to find the respective Eigen vectors:

Calculating the Eigen vector for λ_1 first:

$$\left[\begin{array}{cc|c} \frac{\sqrt{105}-5}{2} & 4 & 0 \\ 5 & \frac{\sqrt{105}+5}{2} & 0 \end{array}\right] \begin{array}{c} r_1 / \frac{\sqrt{105}-5}{2} \\ r_2 - 5r_1 \end{array} \left[\begin{array}{cc|c} 1 & \frac{\sqrt{105}+5}{10} & 0 \\ 0 & 0 & 0 \end{array}\right]$$

$$x_1 = \frac{-\sqrt{105}-5}{10} \cdot x_2$$
$$x_2 = x_2$$
Eigen vector $\vec{v_1} = \begin{bmatrix} \frac{-\sqrt{105}-5}{10} \\ 1 \end{bmatrix}$

Calculating the Eigen vector for λ_2 now:

$$\left[\begin{array}{cc|c} \frac{-\sqrt{105}-5}{2} & 4 & 0 \\ 5 & \frac{-\sqrt{105}+5}{2} & 0 \end{array}\right] \quad \begin{matrix} r_1/\frac{-\sqrt{105}-5}{2} \\ r_2 - 5r_1 \end{matrix} \quad \left[\begin{array}{cc|c} 1 & \frac{-\sqrt{105}+5}{10} & 0 \\ 0 & 0 & 0 \end{array}\right]$$

$$x_1 = \frac{\sqrt{105}-5}{10} \cdot x_2$$
$$x_2 = x_2$$
Eigen vector $\vec{v_2} = \begin{bmatrix} \frac{\sqrt{105}-5}{10} \\ 1 \end{bmatrix}$

Normalizing both vectors:

$$\text{Normalized Eigen vector } \vec{v_1} = \frac{\begin{bmatrix} \frac{-\sqrt{105}-5}{10} \\ 1 \end{bmatrix}}{\sqrt{(\frac{-\sqrt{105}-5}{10})^2 + 1^2}} = \begin{bmatrix} -0.836 \\ 0.548 \end{bmatrix}$$

$$\text{Normalized Eigen vector } \vec{v_2} = \frac{\begin{bmatrix} \frac{\sqrt{105}-5}{10} \\ 1 \end{bmatrix}}{\sqrt{(\frac{\sqrt{105}-5}{10})^2 + 1^2}} = \begin{bmatrix} 0.465 \\ 0.885 \end{bmatrix}$$

Matrix B now:

$$Det(B) = \begin{bmatrix} 4-\lambda & 2 \\ 3 & 1-\lambda \end{bmatrix}$$
$$(4-\lambda) \cdot (1-\lambda) - (2 \cdot 3) = 0$$
$$\lambda^2 - 5\lambda - 2 = 0$$
$$\lambda_1 = \frac{-\sqrt{33}+5}{2}$$
$$\lambda_2 = \frac{\sqrt{33}+5}{2}$$
$$B - \lambda_1 I = \begin{bmatrix} \frac{\sqrt{33}-+3}{2} & 2 \\ 3 & \frac{\sqrt{33}-3}{2} \end{bmatrix}$$
$$B - \lambda_2 I = \begin{bmatrix} \frac{-\sqrt{33}+3}{2} & 2 \\ 3 & \frac{-\sqrt{33}-3}{2} \end{bmatrix}$$

Having both Eigen values λ_1 and λ_2 we can solve each homogeneous system to find the respective Eigen vectors:

Calculating the Eigen vector for λ_1 first:

$$\left[\begin{array}{cc|c} \frac{\sqrt{33}+3}{2} & 2 & 0 \\ 3 & \frac{\sqrt{33}-3}{2} & 0 \end{array}\right] \begin{array}{c} r_1/\frac{\sqrt{33}+3}{2} \\ r_2-3r_1 \end{array} \left[\begin{array}{cc|c} 1 & \frac{\sqrt{33}-3}{6} & 0 \\ 0 & 0 & 0 \end{array}\right]$$

$x_1 = \dfrac{-\sqrt{33}+3}{6} \cdot x_2$

$x_2 = x_2$

Eigen vector $\vec{v_1} = \begin{bmatrix} \frac{-\sqrt{33}+3}{6} \\ 1 \end{bmatrix}$

Calculating the Eigen vector for λ_2 now:

$$\left[\begin{array}{cc|c} \frac{-\sqrt{33}+3}{2} & 2 & 0 \\ 3 & \frac{-\sqrt{33}-3}{2} & 0 \end{array}\right] \begin{array}{c} r_1/\frac{-\sqrt{33}+3}{2} \\ r_2-3r_1 \end{array} \left[\begin{array}{cc|c} 1 & \frac{-\sqrt{33}-3}{6} & 0 \\ 0 & 0 & 0 \end{array}\right]$$

$x_1 = \dfrac{\sqrt{33}+3}{6} \cdot x_2$

$x_2 = x_2$

Eigen vector $\vec{v_2} = \begin{bmatrix} \frac{\sqrt{33}+3}{6} \\ 1 \end{bmatrix}$

Normalizing both vectors:

$$\text{Normalized Eigen vector } \vec{v_1} = \frac{\begin{bmatrix} \frac{-\sqrt{33}+3}{6} \\ 1 \end{bmatrix}}{\sqrt{(\frac{-\sqrt{33}+3}{6})^2 + 1^2}} = \begin{bmatrix} -0.416 \\ 0.909 \end{bmatrix}$$

$$\text{Normalized Eigen vector } \vec{v_2} = \frac{\begin{bmatrix} \frac{\sqrt{33}+3}{6} \\ 1 \end{bmatrix}}{\sqrt{(\frac{\sqrt{33}+3}{6})^2 + 1^2}} = \begin{bmatrix} 0.824 \\ 0.566 \end{bmatrix}$$

2.1.2

Firstly, lets calculate the mean and variance of each student group:

Groups	N	Mean	Variance
High	9	7.67	1.12
Medium	6	7.08	0.54
Low	5	6.60	0.67

Calculate the degrees of freedom:

$df(Between) = k - 1$(where k = number of groups)
$df(Between) = 3 - 1 = \mathbf{2}$
$df(Within) = n - k$(where n = number of occurrences)
$df(Within) = 9 + 6 + 5 - 3 = \mathbf{17}$
$df(Total) = df(Between) + df(Within)$
$df(Total) = 17 + 2 = \mathbf{19}$

Calculate the sum of squares between and within groups:

$$SS(Between) = \sum_{i=1}^{k} m_i(\bar{x} - \bar{x_i})^2 \text{ with } m_i = \text{number of samples in the group}$$
$$SS(Between) = 9 \cdot (7.67 - 7.22)^2 + 6 \cdot (7.08 - 7.22)^2 + 5 \cdot (6.60 - 7.22)^2 = \mathbf{3.83}$$
$$SS(Within) = \sum_{i=1}^{k} v_i(m_i - 1) \text{ with } v_i = \text{variance of the group}$$
$$SS(Within) = (1.12 \cdot 8) + (0.54 \cdot 5) + (0.67 \cdot 4) = \mathbf{14.41}$$
$$SS(Total) = SS(Between) + SS(Within)$$
$$SS(Total) = 3.83 + 14.41 = \mathbf{18.24}$$

With the sum of squares calculated we can now calculate the mean square:

$$MS(Between) = \frac{SS(Between)}{k - 1}$$
$$MS(Between) = 3.83 \,/\, (3 - 1) = \mathbf{1.91}$$
$$MS(Within) = \frac{SS(Within)}{n - k}$$
$$MS(Within) = 14.40 \,/\, (20 - 3) = \mathbf{0.84}$$

And lastly we calculate the value of F-stat:

$$\text{F-stat} = \frac{MS(Between)}{MS(Within)}$$
$$\text{F-stat} = 1.91/0.84 = \mathbf{2.26}$$

With all the values calculated we can fill the ANOVA table as such:

Source	SS	DF	MS	F
Between	3.83	2	1.91	2.26
Within	14.41	17	0.84	NA
Total	18.24	19	NA	NA

2.2

Please find all code under their respective folders in the repository.

2.2.1

The goal of this task was to implement a function that computes the F-statistic based on a given set of dependent and independent variables.

More specifically, the first step is to identify how many unique classes are included in the Dependent Variables (DV) vector, if there is only one class then an error is raised. The next step is to count how many occurrences each class has and what are the corresponding observations. After that, we compute the mean and the variance for each class and the weighted mean between all the classes. Then, we compute the MS(Between) and MS(Within). The last step is to compute the F-statistic using MS(Between) and MS(Within) and output it.

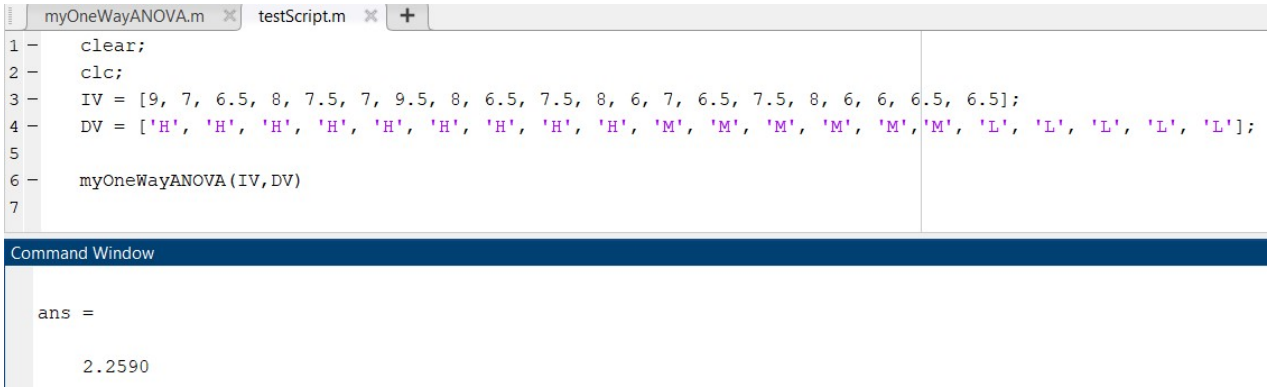

Figure 1: Example of ANOVA

Figure 1 shows the output of the function myOneWayANOVA for the given data.

2.2.2

Please see the handed in code for the implementation. Of note is the multiplication with the reciprocal of $M-1$. In the lecture slides, it was indicated that this should be contained within A (element-wise), but in the implementation this was done outside of A. This was based on other texts describing the process, as well as the observation that the built-in matlab functions generated different output for example inputs. After the announcement stating that each element in A is to be divided by $\sqrt{M-1}$ we realized that this will yield identical results to our implementation. If the reciprocal is multiplied with the matrix A and its transpose ($\frac{1}{M-1}A^TA$), using our definition of A, the resulting matrix is identical to the matrix resulting from using the definition of A as in the announcement and multiplying A^TA. Later in the function, A is multiplied with the eigenvectors of A^TA to obtain the eigenvectors of AA^T. Our different definition of A does not change the result, as our A is simply a scaled version of the A one would use if the division was included in each element of the matrix. As we rescale the eigenvectors to be of length one, the result is identical for both methods.

2.2.3

To complete the `myEigenFaces.m` script, a couple of lines are added:

- `T = testingData - repmat(meanTrainingFace,1,size(testingData,2));` which centers the testing data based on the mean training face.
- `trainingVectors = pc(:, 1:K)' * X;` where $X \in \{A, T\}$, which projects the training/testing data on the principal components.

Additionally, the `myOneWayANOVA.m` script was modified to support numerical labels instead of strings and the `mypca.m` was modified to account for A already being centered and having the rows and columns switched. Please see the source code for the complete scripts.

After running the `myEigenFaces.m` and `myFeatureSelectionwithANOVA.m` scripts with empty parameters, we get the results from figures 2 and 3. In these figures, we can see a couple of things:

- The best accuracy with PCA is achieved using 10 components (85%). Adding more components degrades the accuracy, which stabilizes just above 80%.
- ANOVA is able to reach a very high accuracy rate (close to 0.9), but needs at least 300 features. Another local optimum is reached around 100 features with a little bit under 0.85 accuracy rate.
- PCA and ANOVA show a similar sort of pattern, with two clear local optima and a stabilised but lower accuracy after adding many features/components.

From this it becomes clear that PCA is able to reach a higher accuracy more efficiently than ANOVA for this data set. However, ANOVA is able to reach a higher accuracy rate than PCA in total, which can also be important in accuracy sensitive use cases. Another upside that ANOVA has is that it's a more transparent approach than PCA, making it easier to understand and apply. However, PCA is faster than ANOVA and might thus be better to use in cases where less detail but amount and speed of dimensionality reduction is more important than accuracy and ease of use.

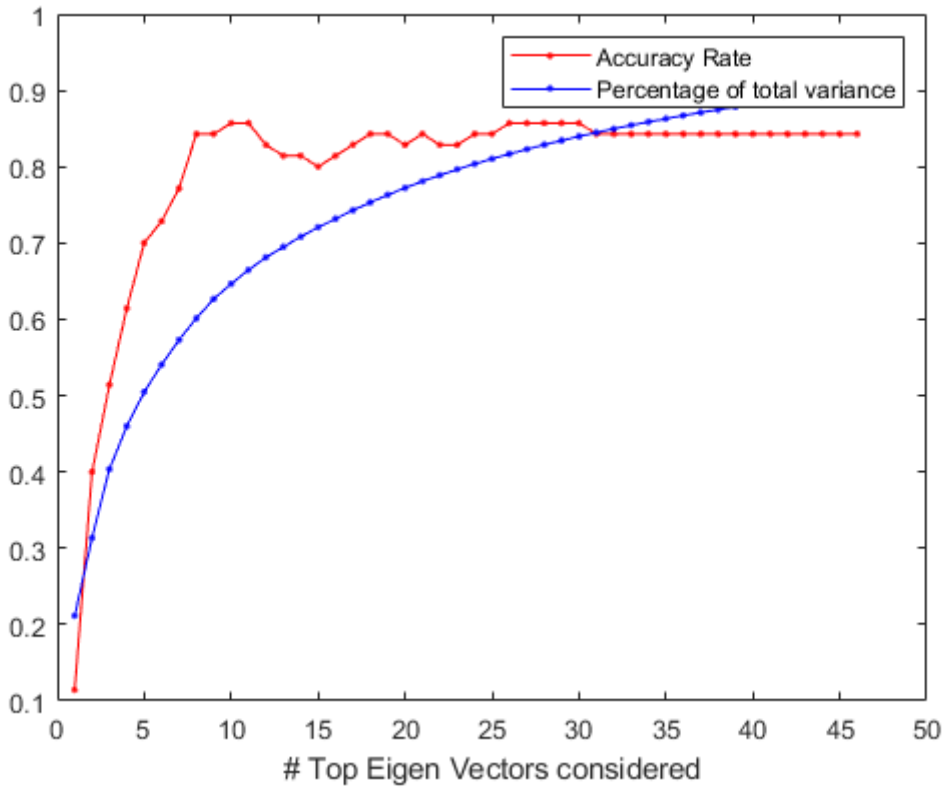

Figure 2: Plot of the face accuracy and percentage total variance versus the amount of principal components considered, as generated by `myEigenFaces.m`

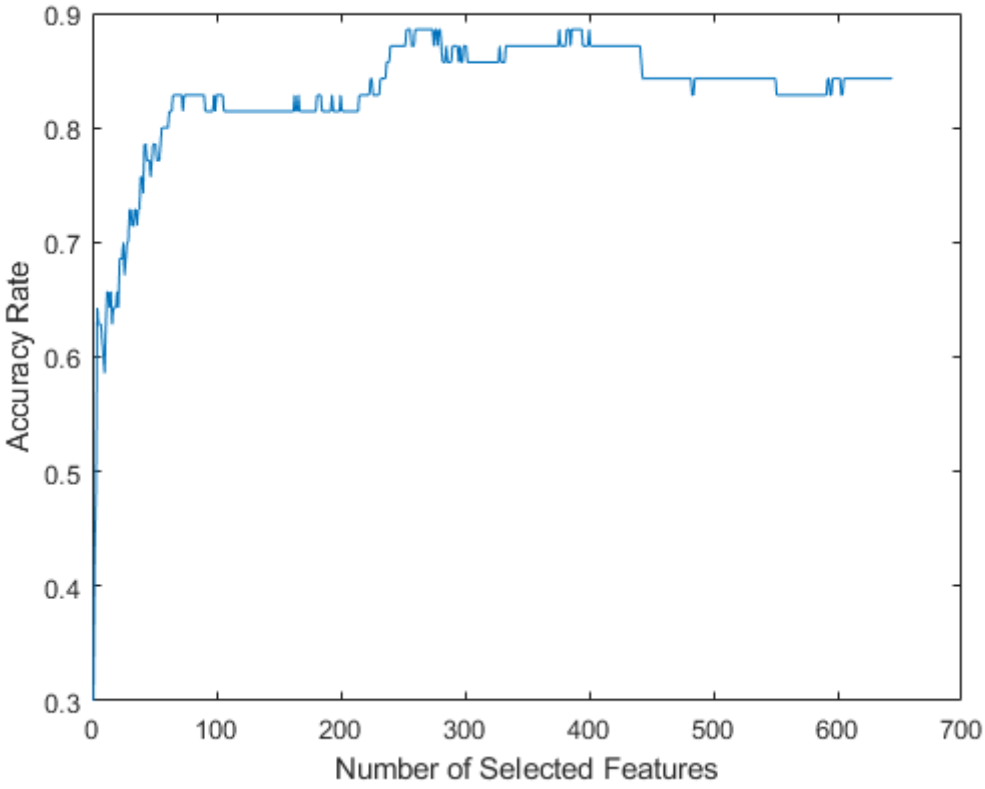

Figure 3: Plot of the accuracy versus the amount of selected features, as generated by `myFeatureSelectionwithANOVA.m`