Pattern Recognition Homework (4)

Feature Selection, PCA, Fisher LDA

Instructed by Dr. Zohreh Azimifar

Abbas Mehrbanian

abbas.mrbn@gmail.com, Stu ID: 40130935

Reza Tahmasebi

saeed77t@gmail.com, Stu ID: 40160957

Table of Contents

1 Feature selection methods

1.1 Concepts

1.1.1 Chi-square

One common feature selection method that is used with text data is the Chi-Square feature selection. The χ2 test is used in statistics to test the independence of two events. More specifically in feature selection we use it to test whether the occurrence of a specific term and the occurrence of a specific class are independent. Given the data of two variables, we can get observed count O and expected count E. Chi-Square measures how expected count E and observed count O deviates each other.

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

Where O_i is the observed value and E_i is the expected value.

Since the chi-square test measures dependence between stochastic variables, so using this function "weeds out" the features that are the most likely to be independent of class and therefore irrelevant for classification.

1.1.2 RFE

RFE is a wrapper-type feature selection algorithm. This means that a different machine learning algorithm is given and used in the core of the method, is wrapped by RFE, and used to help select features. This is in contrast to filter-based feature selections that score each feature and select those features with the largest (or smallest) score.

Technically, RFE is a wrapper-style feature selection algorithm that also uses filter-based feature selection internally. RFE works by searching for a subset of features by starting with all features in the training dataset and successfully removing features until the desired number remains.

This is achieved by fitting the given machine learning algorithm used in the core of the model, ranking features by importance, discarding the least important features, and re-fitting the model. This process is repeated until a specified number of features remains

In short, given an external estimator that assigns weights to features (e.g., the coefficients of a linear model), the goal of recursive feature elimination (RFE) is to select features by recursively considering smaller and smaller sets of features. First, the estimator is trained on the initial set of features and the importance of each feature is obtained either through any specific attribute or callable. Then, the least important features are pruned from current set of features. That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached.

1.1.3 Univariate methods

The statistical measures used in filter-based feature selection are generally calculated one input variable at a time with the target variable. As such, they are referred to as univariate statistical measures. This may mean that any interaction between input variables is not considered in the filtering process. Statistical tests can be used to select those features that have the strongest relationship with the output variable. Univariate feature selection works by selecting the best features based on univariate statistical tests. It can be seen as a preprocessing step to an estimator.

The *scikit-learn* library provides the SelectKBest class that can be used with a suite of different statistical tests to select a specific number of features.

Many different statistical tests can be used with this selection method.

- For regression: r_regression, f_regression, mutual_info_regressi
- For classification: chi2, f_classif, mutual_info_classi

The methods based on F-test estimate the degree of linear dependency between two random variables. On the other hand, mutual information methods can capture any kind of statistical dependency, but being nonparametric, they require more samples for accurate estimation

1.2 Implementation

We used sklearn library in this part of project to do feature selection. For chi-square and f_classif we used SelectKBest class that we discussed in 1.1.3. For our univariate method we used f_classif which computes the ANOVA F-value for the provided sample.

We implemented this part of project as a class named FeatureSelection. We initialize each 3 types of feature selectors with given k. In outputs of our implementation, we reported the difference between estimation accuracy with original dataset and selected features using selection methods, rather than accuracy values, for better understanding about changes. To do such task we implemented *selected_features_report* function to compute the accuracy of logistic regression estimator accuracy with selected features as its dataset with given selection method, *original_report* function to compute the accuracy of logistic regression estimator for original train data set. And to compare these two metrics we implemented *compare_metrics* function which compares the difference between two given inputs.

And finally, we used *report_results* to give a complete report for all feature selection methods on given k and dataset.

1.3 Results

For a more detailed review of the results, we ran our program for multiple times and report 3 random of these runs in the tables shown in next pages. “-“ symbol in the tables means there were no differences.

1.3.1 New & original data accuracy difference

The first thing we noticed was that for each run the difference in accuracies were different. We concluded that for these particular datasets all the features are important since in some runs the selected feature dataset resulted in a better accuracy in compare to original train dataset, and in some runs it didn’t. as it can be seen from secondary tables next to accuracy results, some of selected features were different in each run of program. We can conclude that the final result is highly sensitive to which features are selected.

1.3.2 When to use feature selection?

When presented data with very high dimensionality, models usually choke because:

- Training time increases exponentially with number of features.
- Models have increasing risk of overfitting with increasing number of features.
- There might be dimensions which are considered noises in our model.

Feature Selection methods helps with these problems by reducing the dimensions without much loss of the total information. It also helps to make sense of the features and its importance.

1.3.3 Feature selection methods comparison

By comparing the results, we concluded that RFE method worked better overall on all datasets. The reasons for our conclusion are:

- Applying feature selection using RFE resulted in better accuracies difference between selected features and original for given testing dataset, in compare to other methods.
- If the test dataset accuracy difference wasn't better, in most cases it didn't change for or the difference was lower than other methods, which simply means a better dimensionality reduction we less accuracies difference.

Table 1: Chi-square accuracy comparison results using k = 5

Dataset		**anneal**		**diabetes**		**hepatitis**		**kr-vs-kp**		**vote**	
		train	**test**	**train**	**test**	**train**	**test**	**train**	**test**	**train**	**test**
Run	**#1**	↓ 7.52%	↓ 6.94%	↓ 7.46%	↓ 0.88%	↓ 7.62%	–	↓ 5.63%	↓ 8.5%	↓ 3.53%	↑ 2.41%
	#2	↓ 4.0%	↓ 2.99%	↓ 9.31%	↓ 2.56%	↓ 8.49%	↑ 10.53%	↓ 7.1%	↓ 5.3%	↓ 2.05%	↓ 1.2%
	#3	↓ 9.2%	↑ 1.45%	↓ 7.21%	↓ 6.84%	↓ 8.74%	↑ 13.04%	↓ 6.39%	↓ 7.4%	↓ 2.05%	↓ 1.22%

Table 2: Selected features using Chi-square with k = 5

Dataset		**anneal**	**diabetes**	**hepatitis**	**kr-vs-kp**	**vote**
Run	**#1**	'x31' 'x33' 'x35' 'x58' 'x59'	'x16' 'x18' 'x23' 'x25' 'x27'	'x33' 'x35' 'x37' 'x47' 'x49'	'x18' 'x19' 'x41' 'x65' 'x66'	'x6' 'x9' 'x10' 'x13' 'x34'
	#2	'x31' 'x33' 'x35' 'x47' 'x79'	'x16' 'x18' 'x23' 'x25' 'x27'	'x33' 'x35' 'x37' 'x49' 'x51'	'x14' 'x18' 'x41' 'x65' 'x66'	'x6' 'x9' 'x10' 'x13' 'x34'
	#3	'x29' 'x31' 'x33' 'x35' 'x58'	'x18' 'x20' 'x23' 'x25' 'x27'	'x33' 'x35' 'x37' 'x47' 'x49'	'x14' 'x18' 'x41' 'x65' 'x66'	'x6' 'x9' 'x10' 'x13' 'x34'

Table 3: Chi-square accuracy comparison results using k = 10

Dataset		anneal		diabetes		hepatitis		kr-vs-kp		vote	
		train	test	train	test	train	test	train	test	train	test
Run	#1	↓ 5.57%	↓ 0.69%	↓ 6.64%	↓ 6.67%	↓ 8.82%	–	↓ 2.6%	↓ 2.26%	↓ 3.23%	↓ 1.19%
	#2	↓ 3.17%	↓ 2.7%	↓ 3.86%	↓ 2.54%	↓ 4.04%	↓ 4.17%	↓ 3.11%	↓ 1.77%	↓ 2.05%	↓ 1.2%
	#3	↓ 4.49%	↑ 2.29%	↓ 6.48%	↓ 3.25%	↓ 1.94%	–	↓ 3.34%	↓ 2.27%	↓ 3.52%	↑ 1.2%

Table 4: Selected features using Chi-square with k = 10

Dataset		anneal	diabetes	hepatitis	kr-vs-kp	vote
Run	#1	'x29' 'x31' 'x33' 'x35' 'x43' 'x47' 'x56' 'x58' 'x59' 'x79'	'x16' 'x18' 'x20' 'x21' 'x22' 'x23' 'x25' 'x27' 'x72' 'x74'	'x23' 'x32' 'x33' 'x35' 'x37' 'x45' 'x47' 'x49' 'x51' 'x67'	'x14' 'x18' 'x19' 'x30' 'x31' 'x41' 'x42' 'x63' 'x65' 'x66'	'x6' 'x7' 'x9' 'x10' 'x12' 'x13' 'x21' 'x33' 'x34' 'x39'
	#2	'x11' 'x31' 'x33' 'x35' 'x43' 'x47' 'x56' 'x58' 'x59' 'x79'	'x16' 'x18' 'x20' 'x21' 'x22' 'x23' 'x25' 'x27' 'x72' 'x74'	'x20' 'x23' 'x32' 'x33' 'x35' 'x37' 'x45' 'x47' 'x49' 'x51'	'x14' 'x18' 'x19' 'x30' 'x31' 'x41' 'x42' 'x63' 'x65' 'x66'	'x6' 'x7' 'x9' 'x10' 'x12' 'x13' 'x21' 'x33' 'x34' 'x39'
	#3	'x29' 'x31' 'x33' 'x35' 'x43' 'x47' 'x56' 'x58' 'x59' 'x79'	'x16' 'x18' 'x20' 'x21' 'x22' 'x23' 'x25' 'x27' 'x102' 'x104'	'x31' 'x33' 'x35' 'x37' 'x45' 'x47' 'x49' 'x51' 'x66' 'x67'	'x14' 'x18' 'x19' 'x30' 'x31' 'x41' 'x42' 'x63' 'x65' 'x66'	'x6' 'x7' 'x9' 'x10' 'x12' 'x13' 'x21' 'x24' 'x33' 'x34'

Table 5: RFE accuracy comparison results using k = 5

Dataset		anneal		diabetes		hepatitis		kr-vs-kp		vote	
		train	test	train	test	train	test	train	test	train	test
Run	#1	↓ 7.52%	↓ 6.94%	↓ 3.43%	↑ 3.51%	↓ 6.67%	–	↓ 2.43%	↓ 3.6%	↓ 2.06%	↑ 1.2%
	#2	↓ 7.3%	↓ 2.99%	↓ 5.15%	↓ 1.71%	↓ 8.49%	↑ 10.53%	↓ 3.07%	↓ 2.6%	↓ 2.05%	↓ 1.2%
	#3	↓ 9.2%	↑ 1.45%	↓ 7.21%	↓ 6.84%	↓ 8.74%	↑ 13.04%	↓ 6.39%	↓ 7.4%	↓ 2.05%	↓ 1.22%

Table 6: Selected features using RFE with k = 5

Dataset		anneal	diabetes	hepatitis	kr-vs-kp	vote
Run	#1	'x17' 'x33' 'x34' 'x58' 'x64'	'x16' 'x27' 'x73' 'x75' 'x104'	'x1' 'x32' 'x34' 'x35' 'x47'	'x18' 'x41' 'x42' 'x63' 'x65'	'x7' 'x9' 'x10' 'x31' 'x33'
	#2	'x17' 'x33' 'x35' 'x58' 'x64'	'x17' 'x24' 'x26' 'x73' 'x104'	'x20' 'x32' 'x34' 'x35' 'x47'	'x18' 'x41' 'x42' 'x64' 'x65'	'x6' 'x9' 'x10' 'x31' 'x39'
	#3	'x29' 'x31' 'x33' 'x35' 'x58'	'x18' 'x20' 'x23' 'x25' 'x27'	'x33' 'x35' 'x37' 'x47' 'x49'	'x14' 'x18' 'x41' 'x65' 'x66'	'x6' 'x9' 'x10' 'x13' 'x34'

Table 7: RFE accuracy comparison results using k = 10

Dataset		anneal		diabetes		hepatitis		kr-vs-kp		vote	
		train	**test**	**train**	**test**	**train**	**test**	**train**	**test**	**train**	**test**
Run	**#1**	↓ 5.39%	↓ 0.69%	↓ 3.02%	↓ 6.67%	↓ 4.9%	↑ 4.0%	↓ 2.76%	↓ 2.74%	↓ 0.59%	–
	#2	↓ 5.47%	↓ 6.08%	↓ 1.42%	–	↓ 1.01%	–	↓ 3.35%	↓ 1.94%	↓ 0.29%	↑ 1.2%
	#3	↓ 6.39%	↑ 1.53%	↓ 7.69%	↓ 5.69%	↓ 6.8%	↑ 4.35%	↓ 3.62%	↓ 2.59%	↓ 3.52%	–

Table 8: Selected features using RFE with k = 10

Dataset		anneal	diabetes	hepatitis	kr-vs-kp	vote
Run	**#1**	'x13' 'x24' 'x32' 'x33' 'x34' 'x35' 'x47' 'x59' 'x65' 'x78'	'x16' 'x21' 'x26' 'x27' 'x56' 'x61' 'x72' 'x73' 'x83' 'x105'	'x2' 'x3' 'x22' 'x25' 'x32' 'x33' 'x35' 'x36' 'x48' 'x49'	'x18' 'x19' 'x30' 'x41' 'x42' 'x58' 'x64' 'x65' 'x66' 'x70'	'x6' 'x7' 'x9' 'x10' 'x19' 'x25' 'x28' 'x30' 'x31' 'x33'
	#2	'x16' 'x24' 'x32' 'x33' 'x34' 'x35' 'x46' 'x54' 'x59' 'x64'	'x14' 'x17' 'x23' 'x27' 'x56' 'x63' 'x73' 'x74' 'x95' 'x105'	'x5' 'x15' 'x20' 'x21' 'x30' 'x33' 'x34' 'x35' 'x46' 'x47'	'x18' 'x19' 'x30' 'x41' 'x42' 'x58' 'x64' 'x65' 'x66' 'x69'	'x6' 'x7' 'x9' 'x10' 'x24' 'x27' 'x30' 'x31' 'x33' 'x43'
	#3	'x30' 'x31' 'x32' 'x33' 'x34' 'x35' 'x46' 'x47' 'x58' 'x59'	'x18' 'x19' 'x20' 'x21' 'x22' 'x23' 'x24' 'x25' 'x26' 'x27'	'x34' 'x35' 'x36' 'x37' 'x44' 'x45' 'x46' 'x47' 'x48' 'x49'	'x14' 'x15' 'x18' 'x19' 'x41' 'x42' 'x63' 'x64' 'x65' 'x66'	'x6' 'x7' 'x9' 'x10' 'x12' 'x13' 'x22' 'x24' 'x33' 'x34'

Table 9: Univariate accuracy comparison results using k = 5

Dataset		anneal		diabetes		hepatitis		kr-vs-kp		vote	
		train	**test**	**train**	**test**	**train**	**test**	**train**	**test**	**train**	**test**
Run	**#1**	↓ 7.69%	↓ 7.64%	↓ 7.46%	↓ 0.88%	↓ 7.62%	–	↓ 5.63%	↓ 8.5%	↓ 2.65%	↑ 2.41%
	#2	↓ 7.3%	↓ 2.99%	↓ 9.31%	↓ 2.56%	↓ 9.43%	↑ 10.53%	↓ 7.1%	↓ 5.3%	↓ 2.05%	↓ 1.2%
	#3	↓ 9.38%	↑ 0.72%	↓ 7.21%	↓ 6.84%	↓ 8.74%	↑ 4.35%	↓ 6.39%	↓ 7.4%	↓ 2.05%	↓ 1.22%

Table 10: Selected features using Univariate with k = 5

Dataset		**anneal**	**diabetes**	**hepatitis**	**kr-vs-kp**	**vote**
Run	**#1**	'x30' 'x32' 'x33' 'x34' 'x35'	'x22' 'x23' 'x24' 'x25' 'x27'	'x34' 'x35' 'x37' 'x48' 'x49'	'x18' 'x19' 'x41' 'x42' 'x65'	'x6' 'x7' 'x9' 'x10' 'x13'
	#2	'x32' 'x33' 'x34' 'x35' 'x59'	'x21' 'x22' 'x23' 'x24' 'x25'	'x32' 'x33' 'x34' 'x35' 'x48'	'x18' 'x19' 'x41' 'x42' 'x66'	'x6' 'x7' 'x9' 'x10' 'x13'
	#3	'x30' 'x32' 'x33' 'x34' 'x35'	'x23' 'x24' 'x25' 'x26' 'x27'	'x34' 'x35' 'x47' 'x48' 'x49'	'x18' 'x19' 'x41' 'x42' 'x66'	'x6' 'x7' 'x9' 'x10' 'x13'

Table 11: Univariate accuracy comparison results using k = 10

Dataset		anneal		diabetes		hepatitis		kr-vs-kp		vote	
		train	test	train	test	train	test	train	test	train	test
Run	#1	↓ 7.3%	↓ 2.07%	↓ 7.24%	↓ 7.5%	↓ 7.84%	–	↓ 2.71%	↓ 2.58%	↓ 3.23%	↓ 1.19%
	#2	↓ 5.47%	↓ 6.08%	↓ 7.72%	–	↓ 4.04%	↑ 4.17%	↓ 3.31%	↓ 1.77%	↓ 3.23%	↓ 1.2%
	#3	↓ 6.39%	↑ 1.53%	↓ 7.69%	↓ 5.69%	↓ 6.8%	↑ 4.35%	↓ 3.62%	↓ 2.59%	↓ 3.52%	–

Table 12: Selected features using Univariate with k = 10

Dataset		anneal	diabetes	hepatitis	kr-vs-kp	vote
Run	#1	'x30' 'x31' 'x32' 'x33' 'x34' 'x35' 'x46' 'x47' 'x58' 'x59'	'x18' 'x19' 'x20' 'x21' 'x22' 'x23' 'x24' 'x25' 'x26' 'x27'	'x32' 'x33' 'x34' 'x35' 'x36' 'x37' 'x46' 'x47' 'x48' 'x49'	'x14' 'x15' 'x18' 'x19' 'x30' 'x41' 'x42' 'x64' 'x65' 'x66'	'x6' 'x7' 'x9' 'x10' 'x12' 'x13' 'x21' 'x22' 'x33' 'x34'
	#2	'x30' 'x31' 'x32' 'x33' 'x34' 'x35' 'x46' 'x47' 'x58' 'x59'	'x18' 'x19' 'x20' 'x21' 'x22' 'x23' 'x24' 'x25' 'x26' 'x27'	'x32' 'x33' 'x34' 'x35' 'x36' 'x37' 'x46' 'x47' 'x48' 'x49'	'x14' 'x15' 'x18' 'x19' 'x30' 'x41' 'x42' 'x64' 'x65' 'x66'	'x6' 'x7' 'x9' 'x10' 'x12' 'x13' 'x22' 'x24' 'x33' 'x34'
	#3	'x30' 'x31' 'x32' 'x33' 'x34' 'x35' 'x46' 'x47' 'x58' 'x59']	'x18' 'x19' 'x20' 'x21' 'x22' 'x23' 'x24' 'x25' 'x26' 'x27'	'x34' 'x35' 'x36' 'x37' 'x44' 'x45' 'x46' 'x47' 'x48' 'x49'	'x14' 'x15' 'x18' 'x19' 'x41' 'x42' 'x63' 'x64' 'x65' 'x66'	'x6' 'x7' 'x9' 'x10' 'x12' 'x13' 'x22' 'x24' 'x33' 'x34'

2 PCA

2.1 Implementation

We implemented this part of project a class named PCA. The main steps of implementing PCA in done inside of *fit* function.

This function applies zero mean in first step by calculating average of samples and subtracting sample from the average. In the next step we'll a 2d array in order to computer covariance matrix. So, we simply reshaped our centered samples resulted from previous step, from (400, 64, 64) to (400, 4096).

Now that we have the covariance matrix, we can simply calculate our eigen vectors and values by using SVD. Singular value decomposition (SVD) is a tool from linear algebra that computes the principal components of a matrix. To calculate eigen vectors and eigen values we used *svd* function from *numpy.linalg*.

We also calculated variance explained and cumulative variance explained in the last step of this function.

2.1.1 Transformation

The transformation process in done in a function named *transform*. Which accepts number of components to reduce the samples to that given number of components using eigen vector we just computed.

2.1.2 Reconstruction

The reconstruction process in done in a function named *reconstruct*. Which accepts number of components that the samples were reduced too and computes the reconstructed samples with help of eigen vector we computed before.

2.2 Results

2.2.1 Dataset visualization

To see our samples, we plotted first 10 samples of dataset.

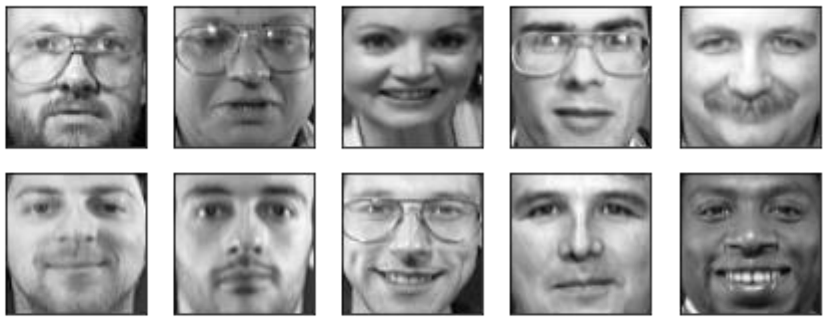

Figure 1: 10 random samples from Olitive dataset

2.2.2 Reduced dataset visualization

We reduced our dataset using the first two & three principal components and visualized it using 2D and 3D plots.

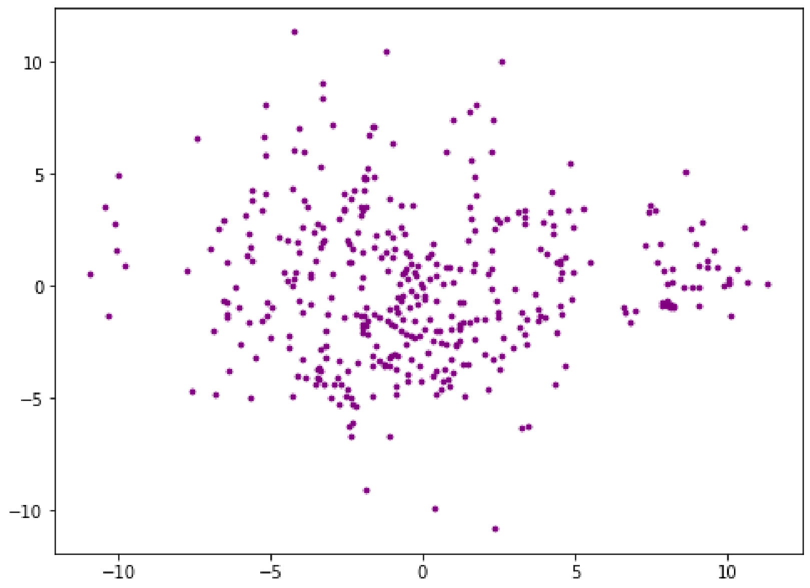

Figure 2: first two principal components visualized in a 2D plot

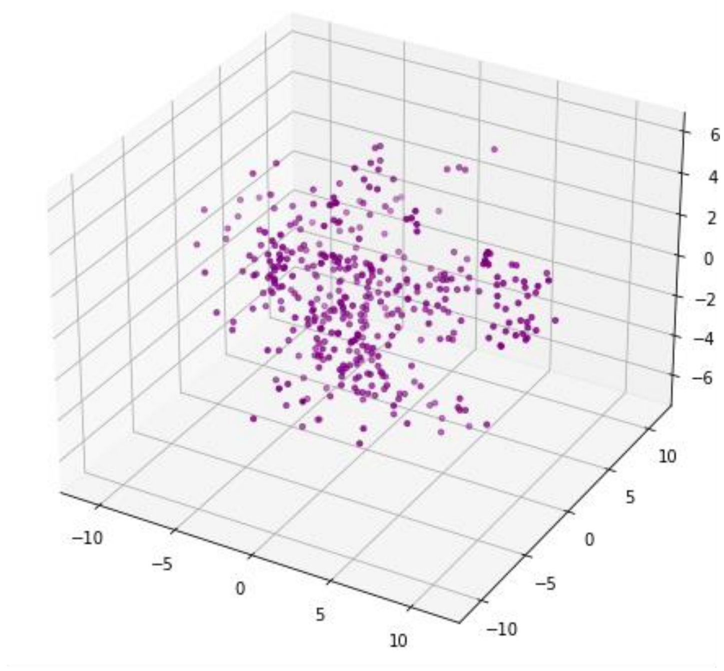

Figure 3: First 3 principal components visualization in a 3D plot

2.2.3 Sample reconstructed images

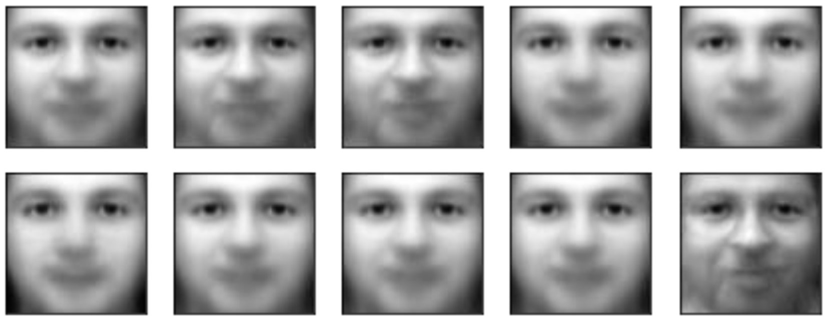

Figure 4: Reconstructed PCA images with k=1

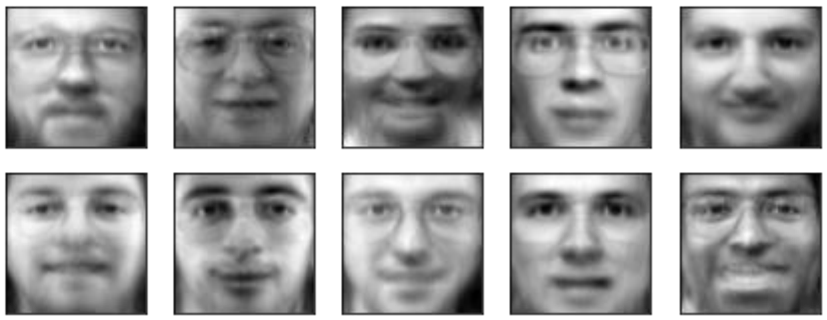

Figure 5: Reconstructed PCA images with k=20

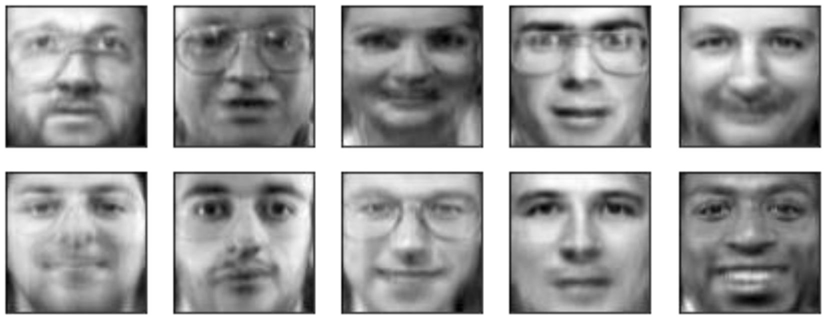

Figure 6: Reconstructed PCA images with k=50

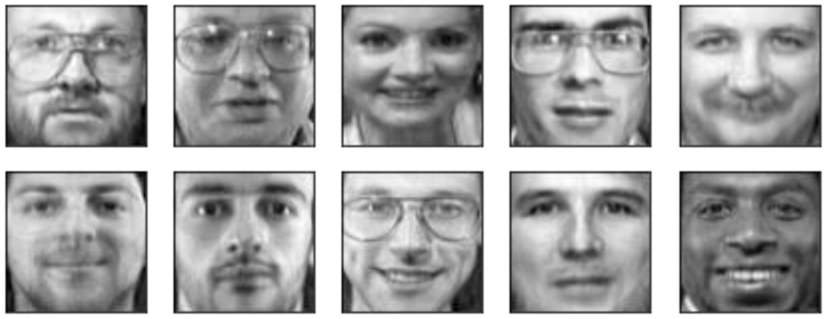

Figure 7: Reconstructed PCA images with k=150

2.2.4 MSE Plots

The FFF shows the MSE for first 1000 components with step size of 10. We calculated MSE with step size of 10 to reduce the computations.

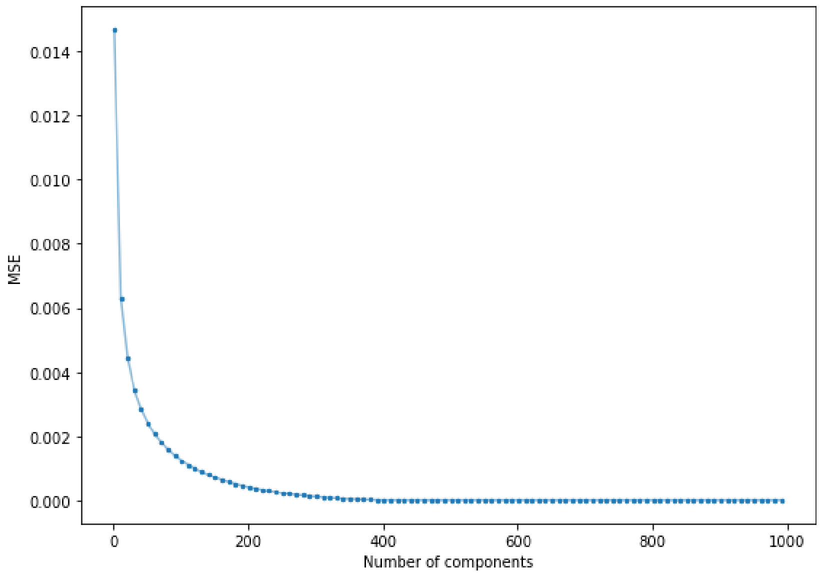

Figure 8: MSE between the original and reconstructed images in terms of the number of eigenvectors

Using step size doesn't affect in final plot which should be descending.

2.2.5 First 20 principal components visualization

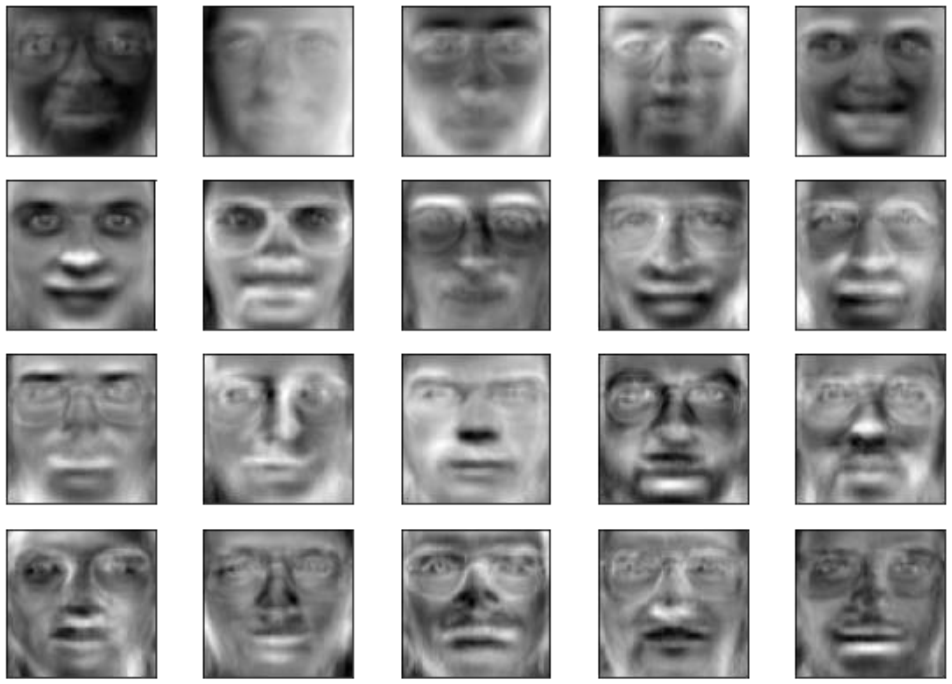

Figure 9: Visualization of first 20 principal components

2.2.6 Cumulative variance retained by the first 300 components

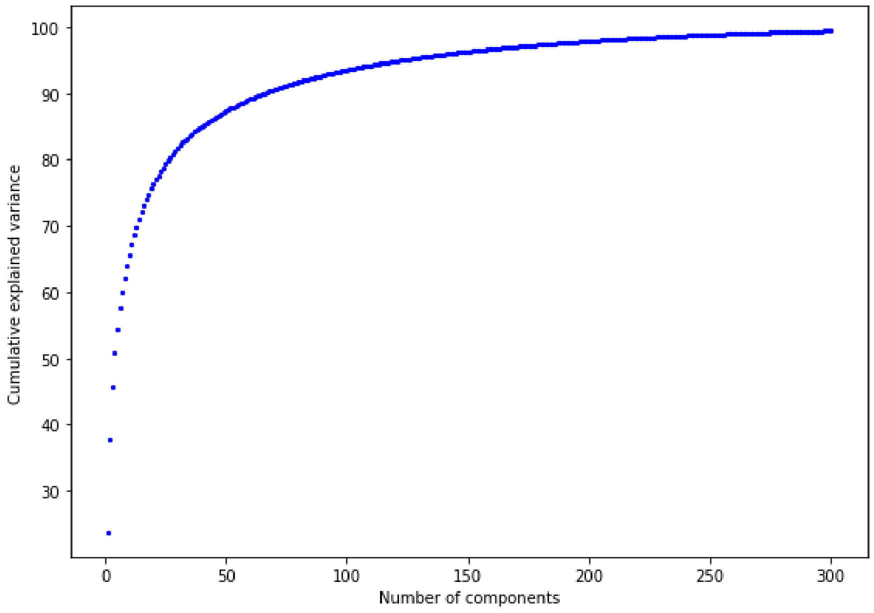

Figure 10: plot of cumulative variance retained by the first 300 components

2.2.7 Instructed results

The answer to instructed results for this project are:

- How much variance is retained by the first component? 23.81%
- How much variance is retained by the first five component? 54.38%
- How many components are needed to keep 75% of the variance? 19 PCs
- How many components are needed to keep 95% of the variance? 123 PCs

2.3 Conclusions

2.3.1 Best number of PCs?

To decide how many eigenvalues/eigenvectors to keep, we should consider our reason for doing PCA in the first place. There are also numbers of possible ways to select principal components. The common way of selecting the Principal Components to be used is to set a threshold of explained variance, such as 80%, and then select the number of components that generate a cumulative sum of explained variance as close as possible of that threshold. There are two main problems with this method:

- It requires a subjective choice of threshold. The 80% or 90% thresholds do not have, in most cases, a fair motive to be chosen, they are arbitrary.
- Some part of that variance may be pure noise and not signal. You have no way of knowing, beforehand, if this threshold you chose removes only noise, and if so, how much or if you are actually removing signal.

Plots like what was provided in Figure 10, can help in such manner.

There are also others ways life permutation test. Still to choose which approach is the best we'll have to consider the reason for doing PCA first.

3 Fisher LDA

LDA is a widely used dimensionality reduction technique built on Fisher's linear discriminant. These concepts are fundamentals of machine learning theory. In this article, I'll go through an example of a classifier using Fisher's linear discriminant and derive the optimal solution for Fisher's criterion. Finally, I compare LDA as a dimensionality reduction technique to PCA.

Fisher's linear discriminant can be used as a supervised learning classifier. Given labeled data, the classifier can find a set of weights to draw a decision boundary, classifying the data. Fisher's linear discriminant attempts to find the vector that maximizes the separation between classes of the projected data. Maximizing "separation" can be ambiguous. The criteria that Fisher's linear discriminant follows to do this is to maximize the distance of the projected means and to minimize the projected within-class variance.

3.1 Implementation

To implement fisher LDA, we used the *fetch_olivetti_faces* function to read our data set.

We split our images and classes, labeled them into x and y, then passed them to the *CalculateScaters* function to calculate the scatters

3.1.1 compute_scatters function

fisher LDA uses two scatters to extract features: 1-within scatter matrix and 2-between scatter matrix. This method calculates both scatters with the formulas below:

$$S_W = \sum_c S_c$$

$$S_c = \sum_c (x_i - \overline{x_c}).\,(x_i - \overline{x_c})^T$$

$$S_B = \sum_c n_c(\overline{x_c} - \bar{x}).\,(\overline{x_c} - \bar{x})^T$$

3.1.2 *GeneralSoloution* function

This method uses the formula below to solve the Eigen vectors and Eigen values of data.

$$S_W^{-1} S_B$$

Because our within-scatter matrix is not ranked and can't invert it, we used a unique shrinkage method to inverse the matrix. In this method, we add a small amount to the main diagonal of the scatter matrix. And then it's invisible.

After that *np.linalg.eig* function has been used to compute the eigenvectors and eigenvalues of our data. We can reshape our eigen vectors and plot them. Some examples are shown in the figure below:

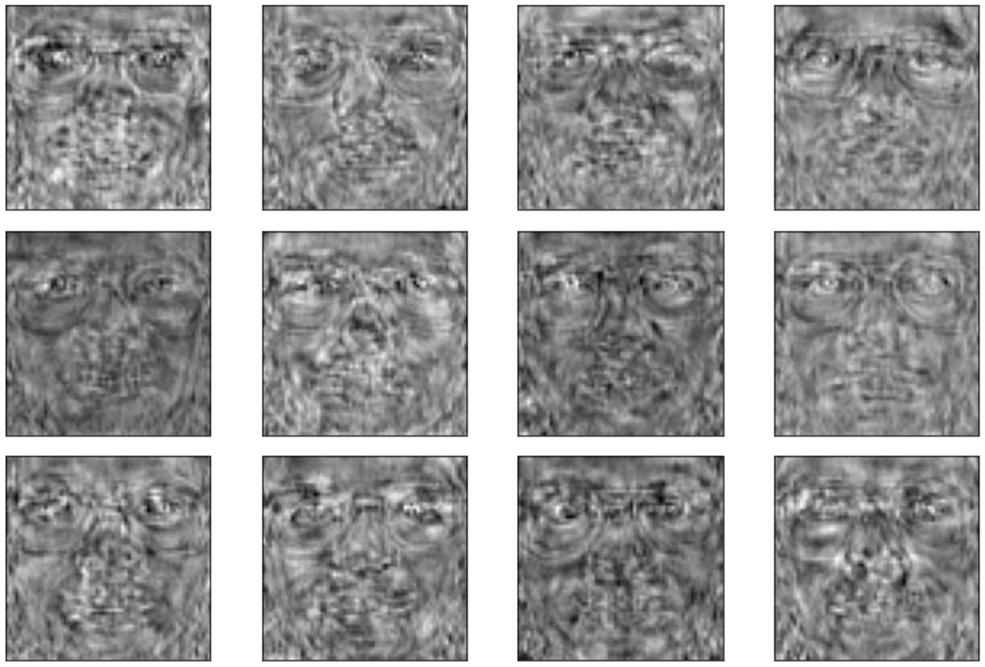

Figure 11: examples of reshaped eigen vectors

3.1.3 *Sorteigens* function

This function gets our eigenvectors and eigenvalues. Then eigenvectors and eigenvalues were sorted in decreasing order

3.1.4 *Transform* function

This method gets x, eigenvectors, and K as arguments and then uses the dot product of the eigenvectors and data to create transformed samples.

And this function returns transformed samples. (feature-reduced models)

3.1.5 *Reconstruct* function

This method tries to reconstruct our original data from transformed values.

This function reconstructs original data with the dot product of eigenvectors and transformed data.

3.2 Results

We plotted our reconstructed samples as instructed in instructions. This plot can be seen in figure below:

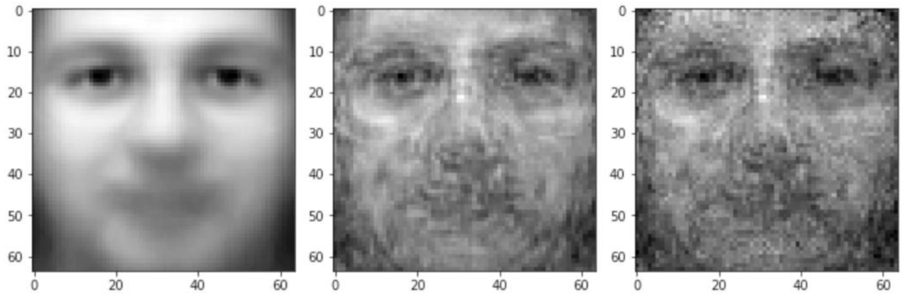

Figure 12: Reconstructed samples by using (a) K =1 (b) K = 40 (c) K = 60 basis vectors

It can be seen that after k = 40, the pictures won't get any better, but they also become noisier. In other words when the selected components are more than number of our classes the reconstruction won’t get any better and might get even noisier as well.

As shown in the figure below, if we increase the K after getting to the number of classes, in this case, 40, the image won't be any better, but it becomes noisier.

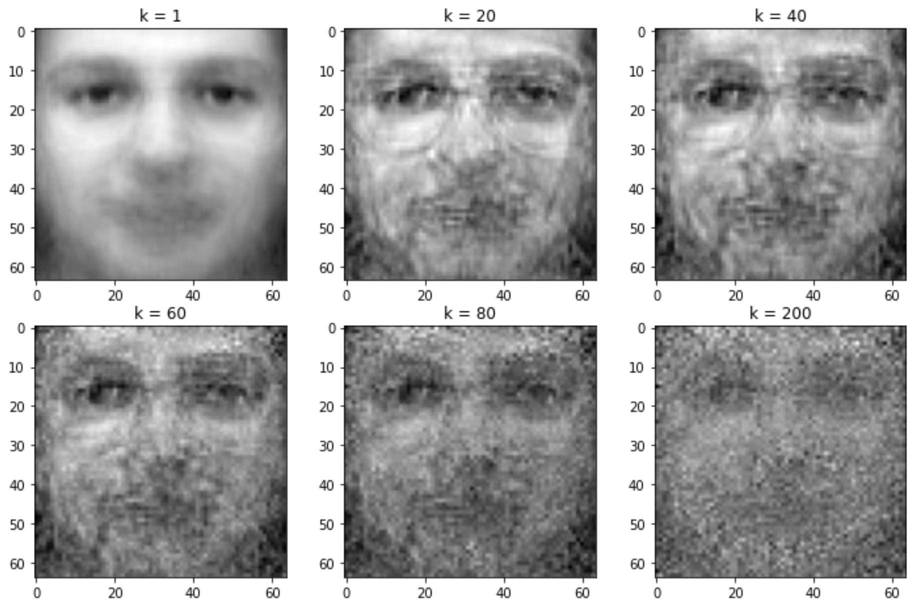

Figure 13: reconstructed samples to k = 200

3.2.1 MSE plot

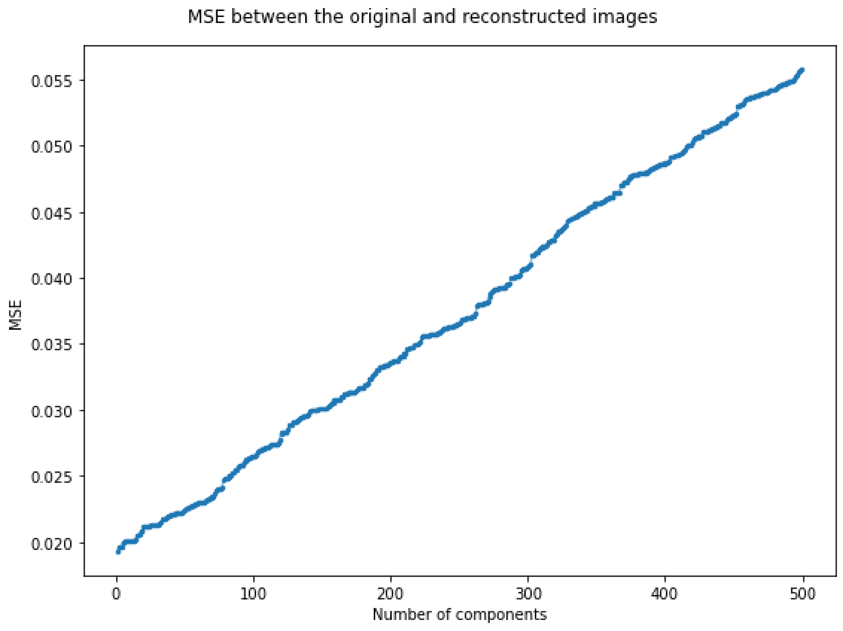

Figure 14: MSE/Number of components plot for LDA Fisher

3.3 Questions answer

3.3.1 The problem with applying Fisher LDA to the dataset

After using Fisher LDA, the original data can hardly be reconstructed. Also, the within-scatter matrix is not positive definite; therefore, it cannot be inverted.

3.3.2 What would happen if we had many outliers in the dataset?

Because we are using the mean of our data to calculate the within-scatter and between-scatter matrixes, our error gets high if we have many outliers in our dataset. It affects our projection (feature reduction), and also it involves our reconstruction.