Project 1 Report

Computational Mathematics

Daniel de las Heras

March 8, 2021

For this report we were given a data set with 721 entries which corresponded each to a different type of musical symmetry. Each of this symmetries had 10 variables.The aim was to perform dimensionality reduction to portrait this data to a two dimensional plot. To do this I used the method of Principal Component Analysis.

The first step was to arrange the data set into matrix form, with 721 rows corresponding to the different experiments or symmetries and 11 rows, which are the variables of these symmetries.

After this we computed this information from our matrix's columns(line 21):

	Range	Max	Min	Mean	Standard Deviation
V1	9	11	2	5.88349515	1.63449874
V2	9	10	1	4.7073509	1.53789445
V3	8	9	1	3.0332871	1.05283522
V4	7	8	1	2.53814147	1.02072906
V5	6	7	1	1.94868239	0.71549256
V6	5	6	1	1.73647712	0.66613199
V7	4	5	1	1.48959778	0.54756045
V8	3	4	1	1.35783634	0.50197699
V9	2	3	1	1.17475728	0.38339418
V10	1	2	1	1.09015257	0.28640021
V11	0	1	1	1	0

Using this information it was necessary to standardize the data's columns (line 37). To do this we used the formula

$$\frac{value - mean}{standarddeviation}.$$

From here we had to compute the covariance matrix (line 43). This was calculated as follows

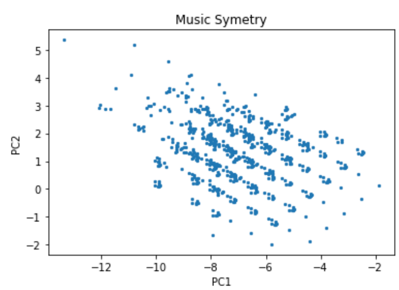
$$X^T X$$

where X is the data matrix and X^T is its transpose.
The next step was to find its eigenvalues and corresponding eigenvectors. The eigenvalues will be what we call principal components. But we were only interested in the two largest in which to project the data:

$$\sigma_1 = 65771.24017227915$$
$$\sigma_2 = 1286.9881567245973$$

With the two largest principal components we could now create a basis matrix with the two eigenvectors corresponding to the selected eigenvalues (line 76) and left multiply this with the standardize data matrix to get the projection (line 77).

By plotting (line 83-87) the projection matrix's columns we get:



As a way to check for the efficiency of this plot, I did two other plots: one projecting the data into two complete random vectors and another projecting the data into two random eigenvectors of the covariance matrix (113-130).

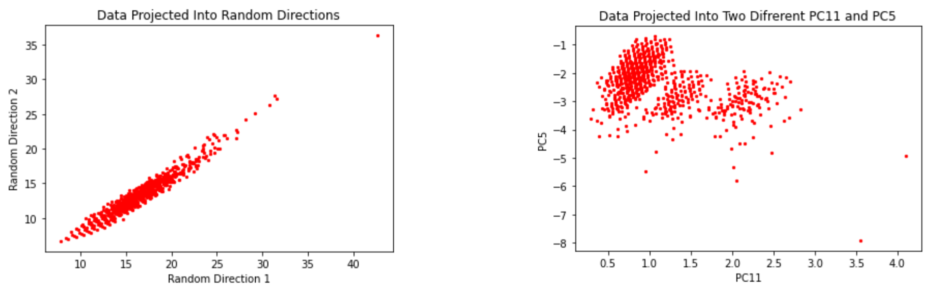


Figure: Other Projections

We can easily sea that neither of them can show a wide spread of the data. Both of them show bulks of points in areas of the plot, therefore they are not a very successful representation of the 11 variables we begun with.

Appendix

```python
import random
import numpy as np
import matplotlib.pyplot as plt


## We read in the infromation from a text file
## And process it such that it is divided into
## 721 experiemnts or rows with 11 columns of
## For each experiment. We save it in the constant X

X = []
with open("data.txt") as f:
    for line in f:
        # convert each line to a list of strings
        line_to_list = line.strip('\n').strip('][').split(', ')
        # convert list of strings to list of integerss and append to data
    list
        X.append([int(i) for i in line_to_list])
# convert data to numpy array
X = np.asarray(X)

## We compute the column mean range and standard deviation of the data matrix
## This provides some usefull information to interpret the data
## We use the funtions .mean, .std, .max, .min, .ptp to do this
mean_column = np.mean(X, axis=0)
min_column =  np.min(X, axis=0)
max_column = np.max(X, axis=0)
range_column = np.ptp(X, axis=0)
std_column = np.std(X, axis=0)


## We standarize the data by columns
## To do this we take the data matrix
## And subtract each column by its mean
## And divide each column by the standard deviation
## Of its entries
n, m = X.shape
XNormed = (X - np.mean(X, axis=0))/np.std(X, axis=0)

## We compute the Covariance Matrix by left matrix multplication
## Of the data matrix X and its transpose
## We use the fucntion .matmul which performs matrix multiplications

C = np.matmul(X.T, X)

## We use the funtion .linalg.eig(c) to obtain
```

```
## Covariance matrix eigenvalues and corresponding eigenvectors
## And store them in two arrays: eigenvalues (stores eigenvalues)
## And eigenvectors (stores eigenvectors)

eigenvalues, eigenvectors = np.linalg.eig(C)

'''
variance_explained = []
for i in eigenvalues:
     variance_explained.append((i/sum(eigenvalues))*100)
print(variance_explained)
'''

## We know that the eigenvalues are sorted such that the largest
## Apear in the leftmost columns. Therefore to access the two
## Largest ones we must get the two first columns

## Principal components
pc1 = eigenvalues[0]
pc2 = eigenvalues[1]

## Their eigenvectors
pc_axis1 = eigenvectors[0].reshape(11,1)
pc_axis2 = eigenvectors[1].reshape(11,1)

## We create a basis matrix (variable eigen_basis) with both eigenvectors
## Using the method .cocatenate
## And we use axes=1 as we want the vectors to be the columns of the matrix
## We then use .matul to multiply the data matrix with the eigen-basis
## This projects the dataset to the eigen-basis
eigen_basis = np.concatenate((pc_axis1,pc_axis2),axis=1)
data_proj = np.matmul(X, eigen_basis)

####################################################################
## Scatter plot display
####################################################################

plt.scatter(data_proj[:,0], data_proj[:,1], s=5)
plt.title("Music Symetry")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.show()

####################################################################
## Plot of data projected into two random directions
####################################################################

## We generate random directions in which we project the data
```

```
## This show a very deficient spread of the data
## So little conclusion can be made
random_dir1 = np.random.rand(11,1)
random_dir2 = np.random.rand(11,1)
new_basis = np.concatenate((random_dir1,random_dir2),axis=1)
another_data_proj = np.matmul(X, new_basis)
plt.scatter(another_data_proj[:,0], another_data_proj[:,1], s=5, c="r")
plt.title("Data Projected Into Random Directions")
plt.xlabel("Random Direction 1")
plt.ylabel("Random Direction 2")
plt.show()

####################################################################
## Plot of data projected into random principal components (not PC1 and PC2)
####################################################################

## We project the data into random eigenvectors
## Which are not the ones that correpond to the
## Two first principal components
random_number1 = random.randint(2,10)
random_number2 = random.randint(2,10)
eigen_random1 = eigenvectors[random_number1].reshape(11,1)
eigen_random2 = eigenvectors[random_number2].reshape(11,1)

## Checks two vectors are not equal by using
## The method .array_equal()
while np.array_equal(eigen_random1, eigen_random2, equal_nan=False):
    random_number2 = random.randint(2,10)
    eigen_random2 = eigenvectors[random_number2].reshape(11,1)

another_new_basis = np.concatenate((eigen_random1, eigen_random2),axis=1)
and_another_data_proj = np.matmul(X, another_new_basis)
plt.scatter(and_another_data_proj[:,0], and_another_data_proj[:,1], s=5, c="r
    ")
plt.title("Data Projected Into Two Difrerent PC{} and PC{}".format(
    random_number1+1, random_number2+1))
plt.xlabel("PC{}".format(random_number1+1))
plt.ylabel("PC{}".format(random_number2+1))
plt.show()
```