Introduction to Data Science

Assignment 3

Péter Pölös

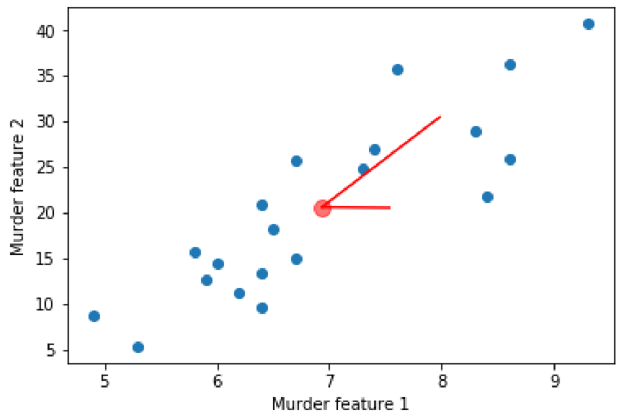
Exercise 1

a) I did my own implementation of PCA, and then I compared my results with the built-in one. It is interesting that they produce the same eigenvalues, but their eigenvectors were pointing into opposite direction (their elements had the same values, only their signs were opposite, although the eigenvectors can be scaled by any number and still be the same eigenvectors)

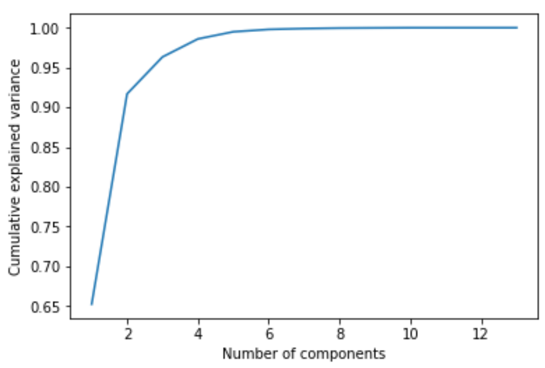
I believe this difference is only due to the fact that Python normalizes the eigenvectors, as it is pointed out at this website: https://www.researchgate.net/post/Why_eigenvectors_seem_incorrect_in_python

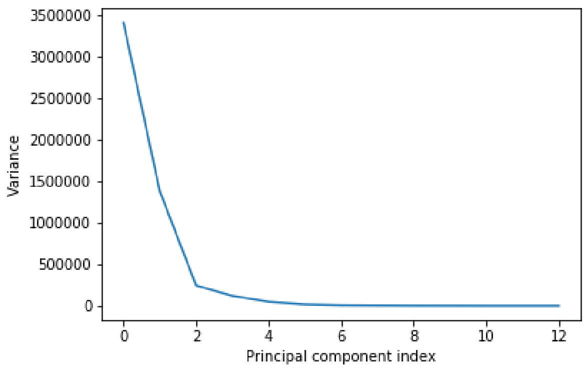
Where they say: " that by scaling the vectors, even the **sign** can change. That's why positive and negative elements might get flipped ". This was the best explanation I could find, I believe this is what happens in my case too, cause the values of the eigenvectors match, only the signs of them are opposite.

b) First I was a bit surprised by this image, that the vectors are not orthogonal, then I realized that the scale of the two axis is different. I kept this plot since when I was applying equal scales to both axis to show that the vectors are indeed orthogonal than most of the datapoints were left out of the plot.



c) As it can be seen it needs 2 dimensions to capture 90% of the cumulative variance and it needs 3 dimensions to capture 95%.



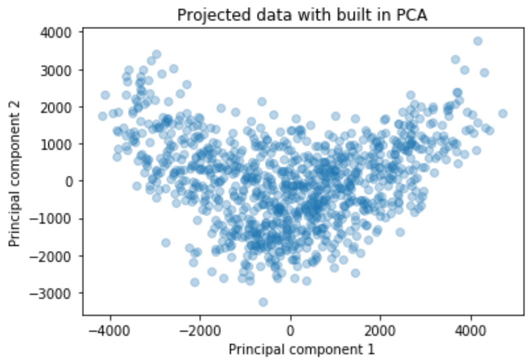


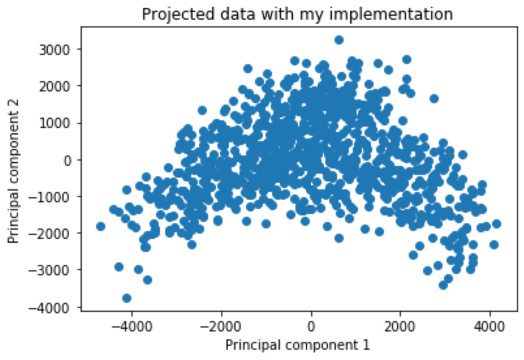
```
array([0.65232418, 0.91676936, 0.96325171, 0.98591075, 0.99491908,
       0.99788133, 0.99894743, 0.99955754, 0.99979513, 0.99998844,
       0.99999496, 1.        , 1.        ])
```

Exercise 2

As we can see my implementation results in the inverse of the image we got with the built in PCA, which is not surprising as in the previous exercise I have pointed out, that the values of my eigenvectors and the ones coming from the built in PCA have the same values, except they have opposite signs, but they are still the same vector, they just simple point to the opposite direction.

I can just repeat myself, I think both solutions are correct since a scaled eigenvector is still an eigenvector, their crucial property is to be orthogonal to one another, and that does not change.





Exercise 3

I have attached my code here just to make the explanation easier:

Step 1

- first step of my function is to choose the first n data points as the initial cluster centers. (choosing them randomly could have led to multiple solutions cause we can not know for sure if our function has converged to a global or local optimum)

Step 2

- then I computed the distance of each data point from each cluster center and assigned the label of the closest center to each, using this pairwise_distances_argmin function
- calculate the new cluster centers by taking the mean of all data points assigned to that centroid's cluster.

Step 3

- then I was repeating **Step 2**, by a while a loop until the centers were converging, meaning that the new centers were the same as the ones in the previous step

```
from sklearn.metrics import pairwise_distances_argmin

# MY implementation
def findclusters(X, n_clusters):
    #initial centers with the first two data points
    centers = X[0:n_clusters]

    while True:
        # calculate distance of all data points from all centers
        labels = pairwise_distances_argmin(X, centers)

        # centroid recalculation: taking the mean of all data points again
        new_centers = np.array([X[labels == i].mean(axis=0)
                                for i in range(n_clusters)])
        # do it until convergence
        if np.all(centers == new_centers):
            break
        centers = new_centers

    return centers, labels

findclusters(x_train, 2)
```

And the two cluster centers I have ended up are:

```
array([[5.70726496e+00, 4.93012821e+01, 7.92408120e+02, 3.85595940e+03,
        3.38821368e+03, 1.35652778e+03, 2.91737179e+02, 1.29989316e+02,
        6.86111111e+01, 3.81880342e+01, 1.87692308e+01, 4.13461538e+00,
        4.42307692e-01],
       [2.19924812e+00, 1.40018797e+01, 1.73727444e+02, 1.40094549e+03,
        3.18759962e+03, 2.62043985e+03, 1.00147368e+03, 6.31413534e+02,
        4.95295113e+02, 2.95238722e+02, 1.45689850e+02, 2.91466165e+01,
        2.82330827e+00]])
```