Matteo D’Ospina s252973

Homework 1 - PCA

Purpose of project:

1. **Principal Component Analysis:**
   Principal component analysis (PCA) is a technique used for identification of a smaller number of uncorrelated variables known as principal components from a larger set of data. In this case the dataset has 1087 images with dimension 227 x 227 x 3. Images belonging to four different classes: Dog, Guitar, Person and House.

2. **Classification with Naïve Bayes Classifier:**
   A naive Bayes classifier is an algorithm that uses Bayes' theorem to classify objects. In this project is used firstly with real images, and then with on PC.
   The classifier assume a Gaussian class-conditional distribution. Considering the following formula.

$$\hat{y} = \underset{y \in \{1,2,3,4\}}{\operatorname{argmax}}\, p(y|x_1, x_2, \ldots, x_d) = \underset{y \in \{1,2,3,4\}}{\operatorname{argmax}}\, p(y) \prod_{i=1}^{d} p(x_i|y) = \underset{y \in \{1,2,3,4\}}{\operatorname{argmax}}\, \frac{1}{4} \prod_{i=1}^{d} \frac{1}{\sqrt{2\pi\sigma_y^2}} e^{-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}}$$

Principal Component Analysis:

1)Data Preparation:

All images ware saved in single numpy array. So it was created a matrix named X that has the images on the rows and the features on the columns.

Usually the range of values of features varies widely, then is really important to standardize each feature (zero-mean and unit-variance) in order to give more emphasis to those features having higher variances than to those with very low variances. It is done in this part of code:

```
X_std = (X - np.mean(X, axis=0))/(np.std(X, axis=0))
```

2) PCA:

Using the function pca.fit() on the standardized matrix X_std, it return X_ t that is a data structure with PC components on the column. immediately after was extracted the first 60, 6, 2 and the last 6 principal components. Remembering that the first components are those that allow less information loss, once the data are projected.

```
pca = PCA()
X_t = pca.fit(X_std)
pca60 = X_t.components_[0:60, :]
pca6 = pca60[0:6, :]
pca2 = pca6[0:2, :]
last6 = X_t.components_[-6:, :]
```

After that it was necessary to project the standardized data to the principal components just found.

```
trasformed2 = my_trasform(X_std, pca2)
trasformed6 = my_trasform(X_std, pca6)
trasformed60 = my_trasform(X_std, pca60)
trasformedlast6 = my_trasform(X_std, last6)
```

```
def my_trasform(data, components):
    return np.dot(data, components.T)
```

The function *my_transform* performs a dot product between X_std (1087 x 154587) and the transposed matrix of the principal components (154587 x N where N is the number of PC). The result matrix has a dimension of 1087 x N. Since N is less than 154587, it's possible to recognize one of the advantages of using PCs, that is, reducing the number of variables.

```
trasformed60 = my_inverse_trasform(trasformed60, pca60)
trasformed6 = my_inverse_trasform(trasformed6, pca6)
trasformed2 = my_inverse_trasform(trasformed2, pca2)
trasformedlast6 = my_inverse_trasform(trasformedlast6, last6)
```

The inverse_trasform function inverts the process of transforming bringing back the dataset to its original dimensions but with a loss of accuracy caused by PCA transformation.

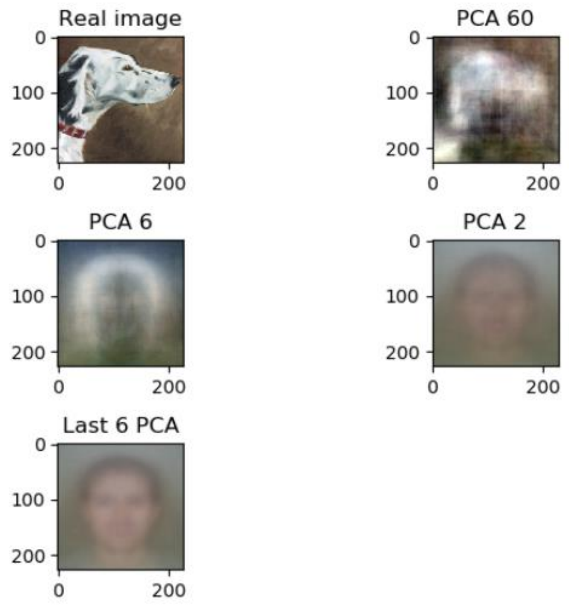
```
def my_inverse_trasform(data_reduced, components):
    return np.dot(data_reduced, components)
```

The function *my_inverse_transform* performs a dot product between transformed data and principal components.

After "destandardize" the inverse transformed data in the following piece of code:

```
trasformed60 = trasformed60 * np.std(X, axis=0) + np.mean(X, axis=0)
trasformed6 = trasformed6 * np.std(X, axis=0) + np.mean(X, axis=0)
trasformed2 = trasformed2 * np.std(X, axis=0) + np.mean(X, axis=0)
trasformedlast6 = trasformedlast6 * np.std(X, axis=0) + np.mean(X,
axis=0)
```

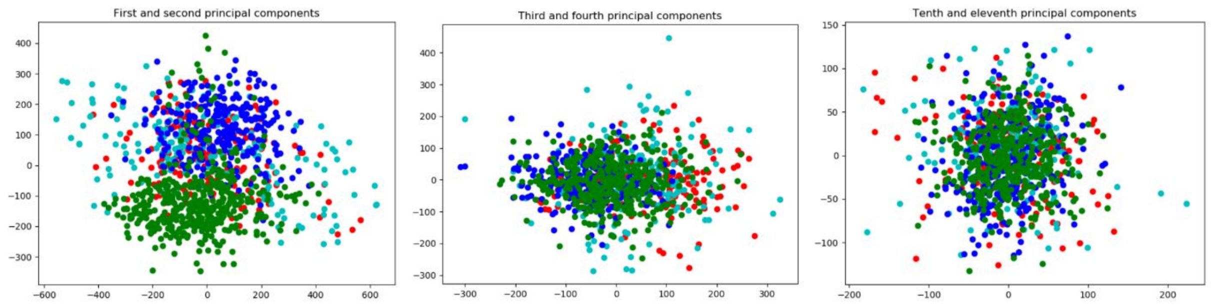
I plot the re-projection of the first image on the first 60PC / 6 PC / 2 PC and the last 6 PC and that's what I got:



In general it’s possible to see that in all images there is a loss of information. This loss in more noticeable in the re-projection on the last 6 PC where we can recognize an human face instead of a dog, because the last six are the poorest PCs. Even in the re-projection on the first 2 and 6 PC the loss is still high so we can’t recognize the original image, probably because 2 or 6 components are too few compared to the starting number. Just with 60 PC we can guess that we are considering the image of a dog.

3) Data Visualization:

Plotting the standardized data on two different PC this three graphs.



The principal component is the direction where there is most variance, the line where the data are most spread out when projected onto it. For this reason, in the first image, where we project the data on the first two PC, allow us to distinguish clearly two different area with the point of two different classes, instead, in the second and third images all the points that belongs to images of different classes are overlapped. In general in the first image compared to the other two, the visualization of samples is more clear than other two, although it is not very clear. But it’s important to remember that original dataset has 154587 variables, and this is a visualization on only 2 variables, so it’s very good.

4) Classification with Naïve Bayes Classifier:

Data Preparation:

In order to implement a Naive Bayes Classifier I split my initial dataset into two different set: train and test, with test size = 0.2.

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.2)
```

Then I trained my classifier with the training set.

```
clf = GaussianNB()
clf.fit(X_train, Y_train)
```

At the end I evaluate my model with the test set.

```
print(clf. predict(X_test))
print('\nAccuracy = ' + str(int(clf.score(X_test, Y_test)*100)))
```

I repeat splitting, training and testing also on data projected onto the first two PC and the third and fourth PC.
The obtained accuracy of the model in 74% with original data. When I project the data on the PC the accuracy decreases because of a loss of information. For the first two PC I got 63% of accuracy and in the last case I got 43%.