Sofía Pérez Pérez and Paula Serna del Amo

PROJECT 1-FACE RECOGNITION

The main objective of this facial-recognition project is to build a classifier that given a photo (with the same number of pixels as the ones used to train it), is able to recognise whether this person belongs to the dataset of people used to build it, or not. However, in order to obtain this recogniser function, some steps need to be executed, such as computing PCA to facilitate computations, creating a distance-based classifier. In addition, training it to obtain the optimal values of the parameters, that proportionate the highest accuracy (taking into account that the dataset we are working with provides very high accuracies with very different values of them), and then the actual classification of a given image into the function. After this step is done, a classifier without pca is trained and used, but implying very high computational times, ending into the non-preferable one.

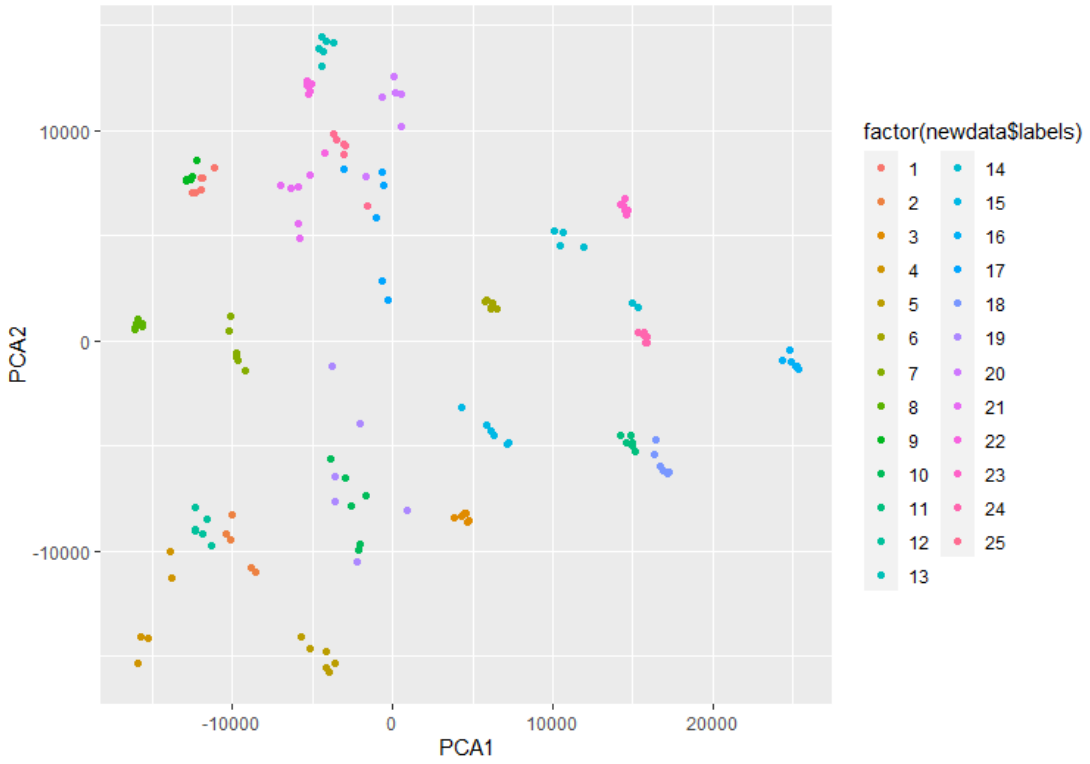
To begin with, dimensionality reduction is crucial regarding the computational time. The conversion of the photos leads to a very big matrix with 108000 columns (number of pixels), so it is more convenient to reduce the number of variables. In order to do this first part of the project, we have constructed a function (PCA), that given a working directory, it obtains the principal components of the photos contained in it. We will call this function later when training our classifier with the training dataset. With this technique we will project the photos into new variables, that are uncorrelated, orthogonal and retain most of the variance explained by the original variables. PCA () first transforms the photos of the current working directory, into a matrix in which every row is a person, and every column a pixel. Therefore, since our initial photos had 108000 pixels, that is the number of variables we obtain. In order to apply pca, scaling is essential, so we have decided to subtract the mean of each column.

Then, we realise that since this dataset has a number of columns large enough to not be able to compute the covariance matrix (needed to obtain the eigenvectors), we will need a trick. The latter consists on creating one matrix, G, that is the result of the original matrix minus the mean of each column. In addition, we will need to obtain the eigenvectors of the matrix that is G times G transposed, divided by the number of rows -1. To do this, the variance covariance matrix is built, and with the function eigen, the eigenvectors and eigenvalues are obtained. We also realised that each eigenvalue corresponds to the variance retained of its corresponding eigenvector. However, those are not the eigenvectors that we need. The ones of the original matrix are the result of the matrix multiplication between G transposed and the eigenvalues of that we obtained before. Once we have computed this matrix, it will be used to project our original data into the new principal axis. One way to realise that we are computing the matrix of eigenvectors correctly is to compute its covariance matrix, that should give a diagonal matrix, since the variables are uncorrelated, which means the covariance between them is 0, and the diagonal corresponds to the eigenvalues of each of the eigenvectors. Having this done, it will be much easier and will take less computational

time to do the needed calculations of classification. This function returns the mean of each column, the projection matrix, and the proportion of the variance explained, between others, that will be used in future steps such as the subtraction of the mean of the image we are classifying in the last step.

In addition, to be able to classify, we will need a variable that indicates the corresponding person of each photo, the vector people. With the latter and the matrix of our projected data, we will build a data frame, that is used in further steps.

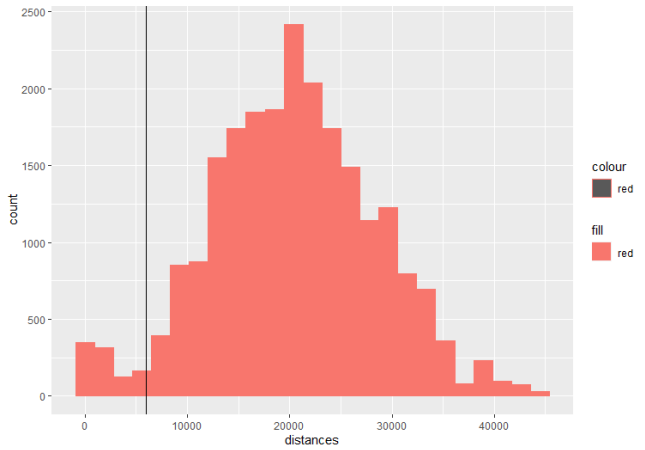
In our own task of classifying the photos, we called this function, PCA (), and we decided to use the first 40 principal components, since doing that implied retaining 95% of the original variance, and we considered it was enough. In order to obtain a more graphical understanding of what we are doing, after the data projection, we plot the first two components as shown above. We can observe that there are small groups of points, that would correspond to photos of the same person. Those are very close to each other since their distance is small compared to the photos of different people. Each group is distinguished with different colours, making the separation into groups more visual and intuitive. Nonetheless, we should take into account that this first two components, only retain 40% of the variance, therefore although it is useful to have a broad idea of what it is being done, it is not completely accurate.



After the PCA part is completed, we start creating a matrix of distances, that will be used later when choosing the correct threshold and determine whether a classification of a photo corresponds actually to an existing person in the dataset or not. If we study this

matrix, we observe that distances between photos of the same person are very small compared with photos of other people. This makes sense, since this matrix is created computing the distance of every photo with the rest, so having that two photos have small distance between them, means that they are the same person. As shown above, we can observe the part of the matrix that corresponds to the distances between photos of the person 1 (rows 1-6), that are small compared with V7, that is a photo of the second person.

	V1	V2	V3	V4	V5	V6	V7
1	0.0000	3612.458	302.2461	5868.946	4437.508	3921.197	21639.0496
2	3612.4575	0.000	3397.2302	5067.314	3684.616	4390.630	20912.6499
3	302.2461	3397.230	0.0000	5708.512	4249.472	3820.116	21567.1788
4	5868.9462	5067.314	5708.5119	0.000	1742.671	2904.297	21735.2113
5	4437.5081	3684.616	4249.4718	1742.671	0.000	2079.425	21365.1535
6	3921.1970	4390.630	3820.1160	2904.297	2079.425	0.000	20814.6950
7	21639.0496	20912.650	21567.1788	21735.211	21365.154	20814.695	0.0000



As shown in the histogram, the left part of the histogram shows smaller distances, that would correspond to distances of photos of the same people, and the right part shows bigger values, that represent distances between photos of different people. Therefore, we set our threshold in 5000 (black line), that is the distance up to two photos would correspond to the same person.

The next step is to build a classifier, that is able to recognise a person belonging to the dataset, but instead of using the original data, we will use the projected one into the new variables which will facilitate computations. One issue we had to face when implementing it, was that the standard knn function used as distance method the Euclidean, and since we wanted to compute it with different distance methods, we had to implement our own function. We called it KNN, that computes the distance between the photo being classified and the different observations of the training set with

different distance methods, and depending on the value of number of neighbours k, the k closest observations will “vote” to see which label of person corresponds to the photo.

Once the function is created, a training of the algorithm is required, to obtain the optimal value of k, and the most accurate distance method. We considered that doing a k-fold cross-validation was a good training option since at the end, every part of the dataset would have been tested and trained. However, if we had chosen repeated random subsampling, which was also considered, its main problem was that we could not assure that every observation was once in the training set and in the test set. We built a data frame, “best_acss”, that collects the parameters that proportionate the highest accuracy in each fold. As we can observe, the accuracies were really high, but in order to avoid overfitting, we decided not to choose the values in which the accuracy is 1, so in this case, row 2 would be chosen as the optimal one.

```
> best_accs#optimal parameters
   k      dist       acc
1  3 euclidean 1.0000000
2  3 euclidean 0.9333333
3  3 euclidean 1.0000000
4  3 euclidean 1.0000000
5  3 euclidean 0.9333333
6  3 euclidean 1.0000000
7  3 euclidean 1.0000000
8  3 euclidean 1.0000000
9  3 euclidean 1.0000000
10 3 euclidean 1.0000000
```

Then, the created and .RData file to store all the variables and functions required in the classifier function obtained in the PCA part, such as the mean of the dataset, that will be used to standardize the image, the matrix P, that will project the image into the principal axis, and also we will store the optimal values of the k and the distance, obtained in the training part.

After the training part, we computed the classification function, “classifier”, that receives as an input a photo. It is transformed into a vector, standardized subtracting the mean of the train, and then projected into the new variables of PCA. With KNN this image is classified as a person of the dataset, using the optimal parameters obtained in the training. However, we have to check whether the distance with a photo of the person that has been classified, is small enough to be considered the same person. It is at this point where the threshold will be considered to obtain the right result.

```
+ }
> classifier("11AT.jpg")
[1] "The person belongs to the data set, corresponds to person number 11"
```

Once this classification with pca part is concluded, we start the last part of the project, which is based in building a function that given a photo, it is able to classify it, but without having the dataset projected into the new variables, the principal axis. We will work with vectors of length 108000, which will increase the computational time a lot, resulting as the non-preferable option. To start with, we will not be able the function to predict, KNN, since it makes use of the dimensionality reduction. We will need to create another function, KNN_NORMAL, that has basically the same concept as KNN, they both

predict a label of the vector people, based on the distance of the input to the rest of the dataset.

Once we have created this function, we will need to train our classifier. We also chose to do k-fold cross validation, combining all the points of the dataset. However, we had to face a problem since the computational time in this part really increased a lot. It could take more than 2 hours to execute the loop.

Once we have our optimal values for the k and the distance, we build the function "classifier_without", in which we will load a previously created .RData file that contains the parameters obtained in the training, and also the vector people used as labels. In this function, the input image will be standardized as usual, subtracting the mean, but will not be projected into the principal axis. Apart from the classification itself, we will need to check with the threshold whether the image actually corresponds to the classification or not. We have to keep in mind that both the function KNN and KNN_NORMAL will always predict something, that is the person corresponding to the smallest distance. However, it is our task to determine whether this classification is wrong or right, that is done with the thresholds.

As we can observe, the function classifier_without works properly with the photo "8ET.jpg" of our dataset.

```
+ }
> classifier_without("8ET.jpg")
[1] "The person belongs to the data set, corresponds to person number 8"
>
```

Finally, we have come to the conclusion that using the dimensionality reduction method and the principal component analysis of our data greatly reduces the training time, and improves the performance of our algorithm as the new variables are independent from each other and explain more than the 95% of the variance of the original ones; so the function PCA correctly predicts an image. In addition, pca helps to overcome the overfitting issue by reducing the number of features since it mainly occurs when there are too many variables in the dataset.

We have noticed the difference between doing pca or not because not doing it leads to endless computations and complications when optimizing the parameters, although this way also performs good predictions.

After our experience, the principal component analysis has resulted us a more convenient and faster method, even more effective, to carry out this classification task due to the huge dimensions of the dataset.