UNIVERSIDAD
COMPLUTENSE
MADRID

XIV Modelling Week

Indentifying people from images of their faces

Authors: Francisco Antonio de la Asunción
Cristina Salvador López
Beatriz Amador
David Carbajo
Juan Manuel Espinosa

Master in Mathematical Engineering

Contents

List of Figures

Introduction and purpose

In this work we will make an approach to one of the most modern problems in surveillance, or security itself: the **facial recognition.** By itself it is a quite interesting problem, not necessary regarding to its applications.

Using a data set containing an amount of 5000> images from about 1000> people, our goal is to identify the person who appears on determinate photograph. To achive this result, it is quite obvious that any classic mathematic algorithm won't be proper; instead of them, we will try to use some of the called **Artificial Intelligence** algorithms.

In short words, these methods have the ability to learn from data, and usually obtain better results by letting the algorithm running considerable time. They are, on the opposite side, very complicated methods to understand them perfectly, and that is why some researchers are not confortable whit using them.

We are going to work in some algorithms such as Neural Networks or SVM. Also, to achive better results, we will try to use what is called a **pre-processing technique,** which is a way to 'clean data' before passing it to the algorithm.

Problems found in dataset

In our first approach to explore data, we noticed that a few people had more than 50 images from their faces, while most of the people had only one or two images. To solve the problems this could make, we decided to **keep only the people who had more than 20 images, and not avoiding to have more than 50 of them.** (just in case they had more) Other issue we found was the amount of differences between the images of a given person. For example, in this two images from Arnold Schwarzenegger we can see a huge difference in him, maybe because the left one photo is from one of his first films (around 1975) and the other could be from 2005.

Figure 1: Arnold Schwarzenegger

After having most of the work done, we notice that, as we will see in the next chapter, the neck of the people made us get bad results in terms of scoring. When we saw a cluster formed by people who wear tie and shirt, we started to think of this as a problem, so we decided to **slice the edges of the images to keep only the face.** We did this automatically, of course, applying a parameter in the code.

Figure 2: Slice of the images

Chapter 1

Clustering methods and pre-processing techniques

1.1 Clustering methods

Cluster analysis is a class of techniques that are used to classify objects into relative groups called clusters, which include the ones having characteristics in common. A cluster can be defined as a group of data in such a way that objects in it are more similar to each other than to those in other clusters.

There are over 100 published clustering algorithms so they can be categorized based on their cluster model:

- Connectivity-based clustering or hierarchical clustering. This algorithms are based on the idea of objects being more related to nearby objects that to objects further away, in other words it is based on their distance. Strategies for hierarchical clustering can be divided in two types: agglomerative and divise.
- Centroid-based clustering. In this kind of algorithms, we can represent a cluster by a central vector, which may not be a member of the data set. Given a number of clusters k, the goal of it is to find the k cluster centers and assign the objects to the nearest cluster center in a way that the squares distances from the cluster are minimized.
- Distribution-based clustering. Here, clusters can easily be defined as object belonging most likely to the same distributions. In this project we are not going to focus on this kind of algorithms.
- Density-based clustering. In this clustering, clusters are defined as areas of higher density that the other ones in the data set. The most popular algorithm is Density-Based Spatial Clustering of Applications with Noise, also known as DBSCAN.

As long as we want the images to be group by people, this kind of algorithms were used in this project to find 62 different clusters, so each person can be related to one of them. In the next sections it is shown how three of this algorithms works and the applications that they can have in our problem: K-means, Agglomerative Clustering and DBSCAN.

1.1.1 k-means

k-means clustering is a method that takes n observations and partitioned them into k clusters in which each observation belongs to the cluster with the nearest cluster centroid.

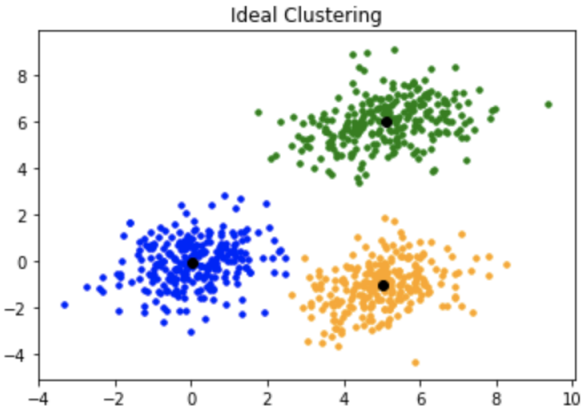


Figure 1.1: k-means clustering

This problem optimizes squared errors. Given a group of observations x, let S_i be the cluster number i and μ_i is the mean of points in S_i, the objective is to find:

$$\arg\min_S \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2 = \arg\min_S \sum_{i=1}^{k} |S_i| \text{var}(S_i) \Leftrightarrow$$

$$\Leftrightarrow \arg\min_S \sum_{i=1}^{k} \frac{1}{2|S_i|} \sum_{x,y \in S_i} \|x - y\|^2$$

Exists different variations of this algorithm, which is called *full*. For example, the *elkan* variation uses the traingle inequality or the *auto* that might change in the future for a better heuristic. We have tried this three different algorithms and the best result we have achieved is using the last one.

1.1.2 Agglomerative clustering

As we have seen before, Agglomerative clustering is part of the Hierarchical clustering or the Hierarchical Cluster Analysis (HCA), so it is based on the distance between objects: euclidean, sqared euclidean, manhattan or Mahalanobis are some examples.

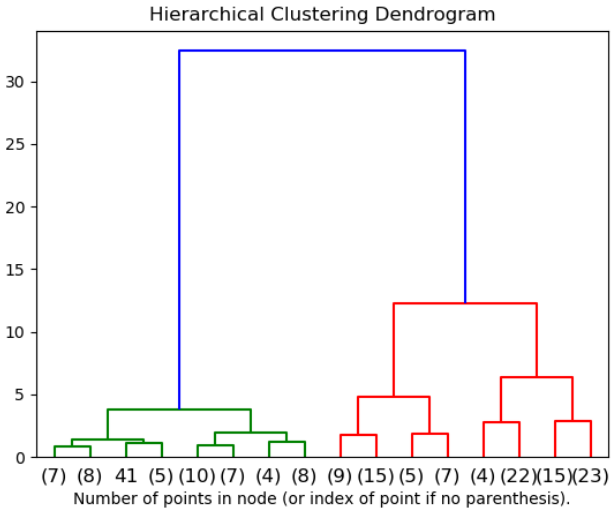


Figure 1.2: Hierarchical Clustering Dendrogram

In order to classify the objects, the algorithm can be based on different linkage criteria:

1. Ward: minimizes the variance of the clusters being merged.
2. Complete: uses the average of the distances of each observation of two sets.
3. Average: uses the maximum distance between all observations of the two sets.
4. Single: uses the minimum of the distances between all observations of the two sets.

We have defined a function that compare the results of this algorithm with different distances and linkage criteria. The best results we have achieved are using the euclidean distance and the complete criteria.

1.1.3 DBSCAN

Density-Based Spatial Clustering of Applications with Noise groups together observations with many nearby neighbors, marking as outliers observations that lie alone those whose nearest neighbors are too far away. We have used a command called *GridSearchCV*, which search for the best results changing the algorithm's parameters.

This method was not very appropriate to solve our problem because not all pictures are considered in order to create the clusters, only those that the algorithm thinks are

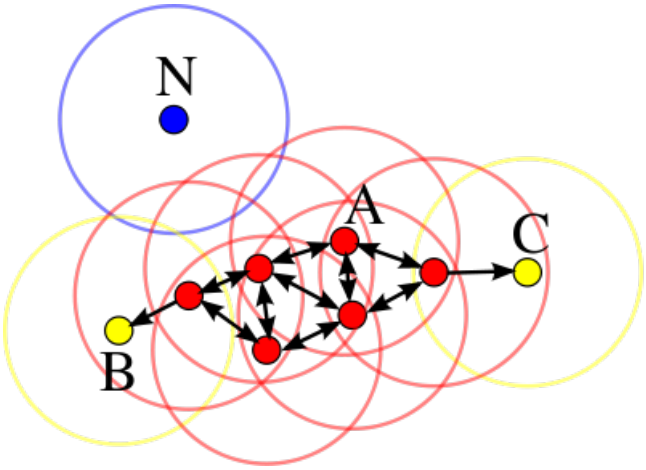


Figure 1.3: DBSCAN clustering

most relevant for the model. More concretely, the biggest cluster size we have achieved is 10, using half of our sample.

1.2 Measure of error

Due to clustering being an unsupervised learning technique, we don't have an error score so we have to find one measure to compare the results. We chose a combination of Homogeneity, which is the number of different people that appears in a cluster, and Completeness, which is the number of photos from one person in a cluster.

This measure is independent of the labels as it is just focused on the groupings and not the labels themselves and it is a value between 0 and 1.

1.3 Pre-processing techniques

1.3.1 Principal Components Analysis

The main idea of the Principal Component Analysis (PCA) is to turn a set of possibly correlated variables into a smaller set of uncorrelated variables. Our high-dimensional data-set is often described by correlated variables and only a few meaningful dimensions retains most of the information.

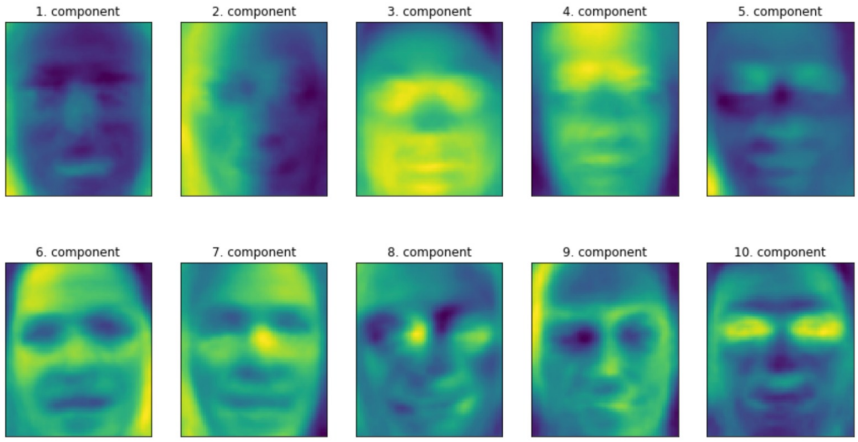
In PCA, we chose W to maximize the determinant of the total scatter matrix.

$$W_{opt} = \arg\max_{W} \left| W^T S_T W \right|,$$

where S_T is the total scatter matrix.

The eigenfaces are the eigenvectors associated to the largest eigenvalues of the covariance matrix of the training data.

In our problem, we used between 100 and 150 components, which capture at least the 90% os the total variance.



1.3.2 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a dimensionality reduction method similar to Principal Components Analysis, but the first one is a supervised method and it takes advantage of the fact that our dataset is labeled.

PCA finds linear combinations that maximize the total variance in our data, not taking into account any class information. The variation due to illumination, for example, could be larger than the variation due to facial features and PCA would retain the first one, but this is not what we would like. Nevertheless, LDA finds a combination of features that maximizes the ratio of between-classes to within-classes scatter and, therefore, it is more appropriate for our purpose.

$$W_{opt} = \arg\max_{W} \left| \frac{W^T S_B W}{S^T S_W W} \right|$$

where S_B is the between-class scatter matrix and S_W is the within-class scatter matrix.

The eigenvectors of W_{opt} associated to non-zero eigenvalues are the so-called Fisherfaces.

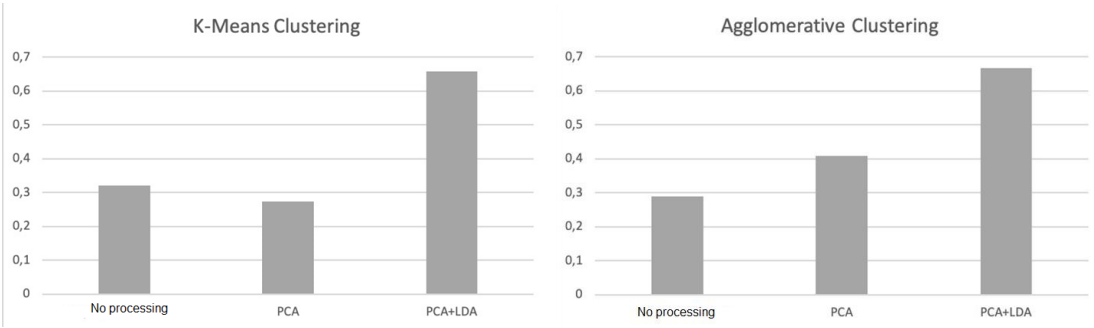
In the face recognition problem, the rank of the matrix $S_W \in \mathbb{R}^{nxn}$ is at most $N - c$, where N is the number of images in the training set, c is the number of classes and n is the number of pixels in each image. In general, N is smaller than the dimension of the input data, n, so S_W is a singular matrix. To avoid this problem, we use PCA (so that S_W is not singular anymore) before applying LDA.

1.3.3 VGG16

VGG16 is the name of convolutional neural network implemented in *keras* package for Python. It is used in many deep learning image classification problems. In our problem, we used VGG16 as a preprocessing technique before applying a classifciation method.

1.4 Clustering results

We show below the results obtained making use of k-means and agglomerative clustering after applying different preprocessing techniques. With both of them, we get the biggest measure using PCA and LDA before the clustering method. We can also see that there isn't a big difference between the results obtained making use of k-means and agglomerative clustering.



Chapter 2

Classification methods

Due to we know the label of each picture, the most suitable machine learning algorithms to our problem are classification ones. To implement these methods we need to split our data into train and test. We use a class stratified sampling to get 75% of the data to train and 25% to test.

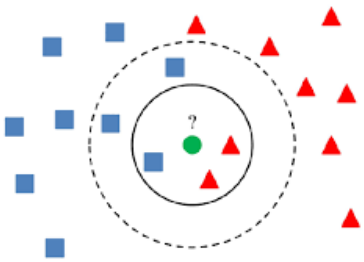
We are going to try nearest neighbors classification algorithms, classification neural network, some ensemble methods and support vector machine.

2.1 Nearest Neighbors algorithms

These algoritms assign to each observation a label based on the nearest observations of the training data.

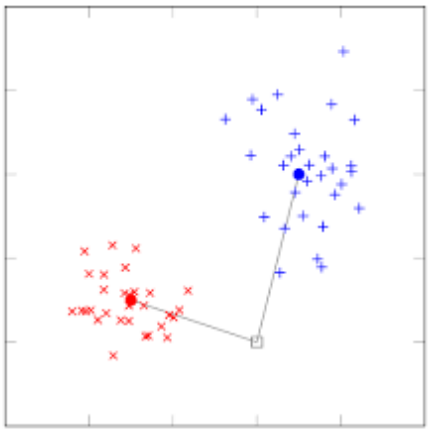
2.1.1 K-nearest neighbors

K-nearest neighbors algorithm look at the class of the k nearest training observations to assign the most comon label to the test observation.



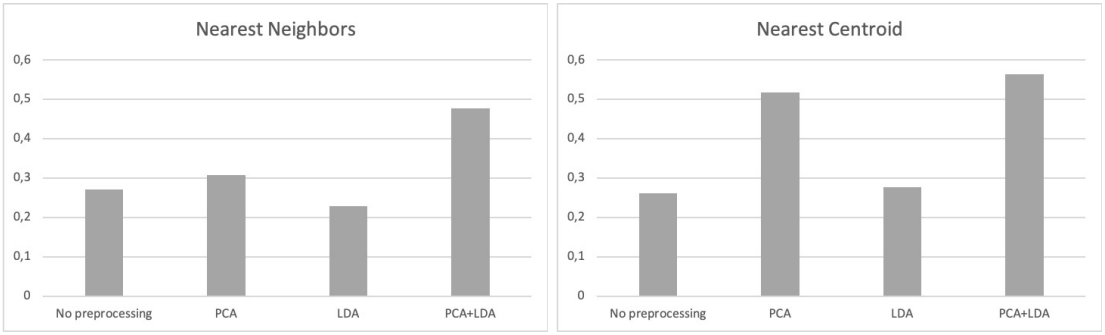
2.1.2 Nearest centroid

Nearest centroid algorithm represents each class by the centroid of its members and assign test observations to the nearest one.



2.1.3 Score comparison

As we can see at the figure below the accuracy of these models without any preprocessing technique is near to a 30%. Both algorithms got better results after using PCA or PCA+LDA as preprocessing techniques. Achieving over 50 per cent accuracy for the nearest centroid model.



2.2 Neural Networks

Neural networks consist of a set of units, called artificial neurons, connected together to transmit signals. The input information cross the neural network (where various operations are made) producing output values.

Each neuron is connected to others through links. In these links, the output value of the previous neuron is multiplied by a weight value. Similarly, at the exit of the neuron, there may be a limiting function, which modifies the result value or imposes a limit that must not be exceeded before spreading to another neuron. This function is known as an activation function.

These systems learn and train themselves, rather than being explicitly programmed, and are used in areas where detection of solutions are difficult to express with conventional

programming.

The tested activation functions are as follows:

- ReLU:
  - Computationally efficient,the network converge very quickly.
  - Non-linear: although it looks like a linear function, ReLU has a derivative function and allows for backpropagation.
  - The problem is when inputs approach zero, or are negative, the gradient of the function becomes zero, the network cannot perform backpropagation and cannot learn.
- Leaky ReLU:
  - This variation of ReLU has a small positive slope in the negative area, so it does enable backpropagation, even for negative input values.
  - Results not consistent—leaky ReLU does not provide consistent predictions for negative input values.
- Swish:
  - Is a new, self-gated activation function discovered by researchers at Google. According to their paper, it performs better than ReLU with a similar level of computational efficiency.
- Softmax:
  - Is able to handle multiple classes.
  - Typically is used only for the output layer, for neural networks that need to classify inputs into multiple categories.

Is convenient to also use the dropout function, it is used to drop random connections to prevent overfitting in first layers.

Combining this tools, we had on 35% score on test with 8 layers and Leaky Relu activation.

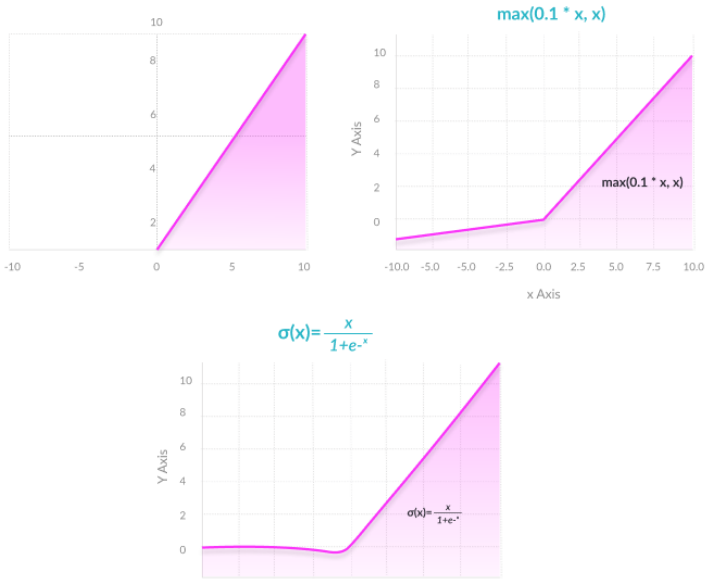


Figure 2.1: Relu, Leacky ReLU and Swish

In the next figure we can see the progression during the epochs.

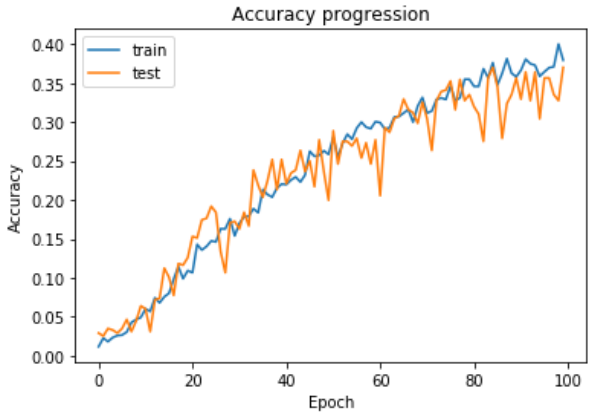


Figure 2.2: Progression of the train and test score

We can see that both, train and test, are similar so we don't have overfit. Increasing the number of epochs is over-adjusted, the train accuracy increment is near to zero, so it is not worth it.

We can also see the confusion matrix:

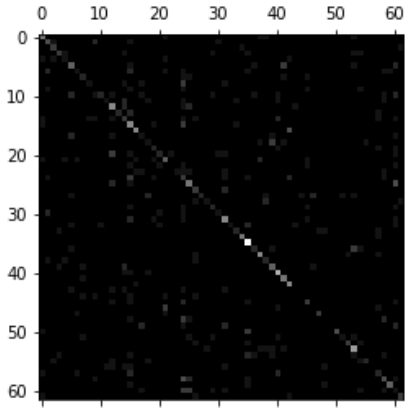


Figure 2.3: Confusion matrix

Ideally this matrix's diagonal should be white and the rest black. Searching on the matrix we found that of all the people involved in the model training these are the ones that gave most of the problems.

Figure 2.4: Gloria Macapagal Arroyo y Megawati Sukarnoputri

We find this reasonable because they look alike.

We tried also the multi-layer perceptor which is implemented in sklear from python.

A multilayer perceptron (MLP) is a class of feedforward artificial neural network (ANN). An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.

The best result was with VGG16 plus PCA plus LDA as pre-process with a 59% score and the specifications are:

- 1024 neurons
- ReLu activation
- LBFGS solver

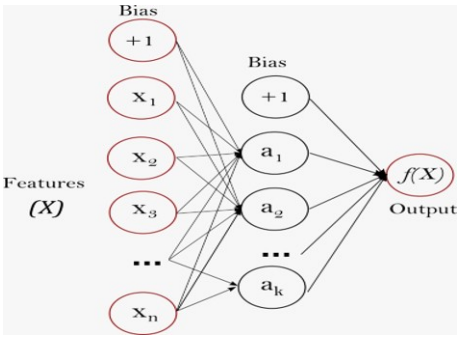


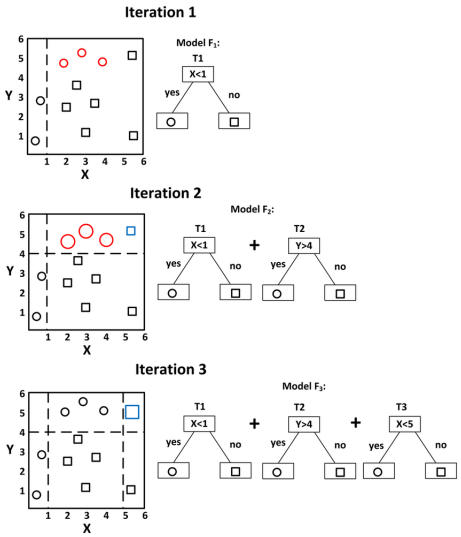
Figure 2.5: MLP

2.3 Ensemble Methods

Ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone.

2.3.1 Gradient Boosting

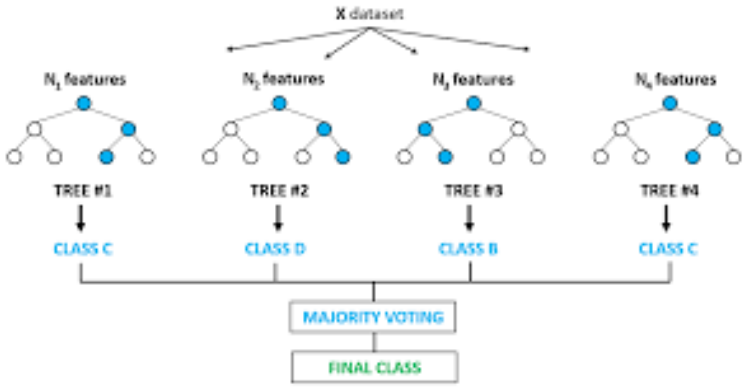
This method produces a prediction model in the form of an ensemble of others prediction models, typically decision trees.



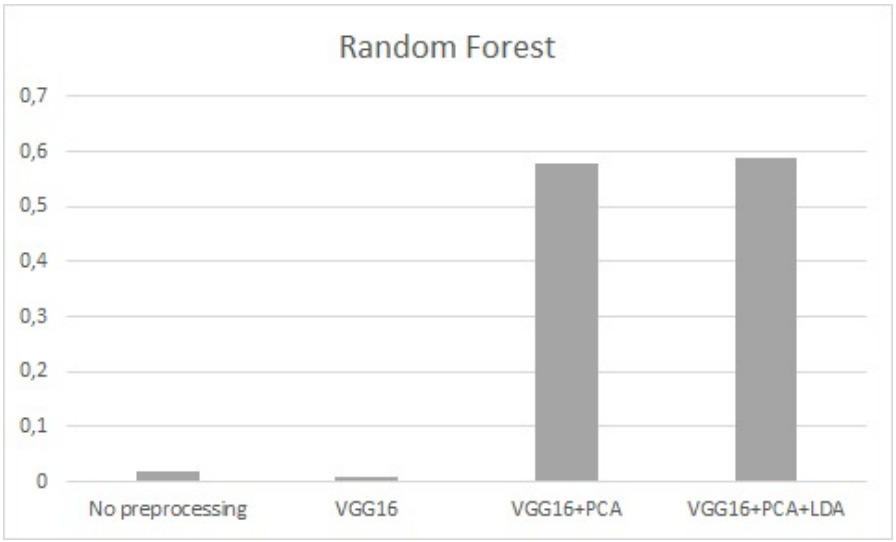
We have not achieved good results with this method but maybe by exploring more interesting things can be achieved.

2.3.2 Random Forest

This method construct a lot of decision trees and tho output is the mode of the individual trees results. With this method we obtain a 59%. This result was using 5000 trees,

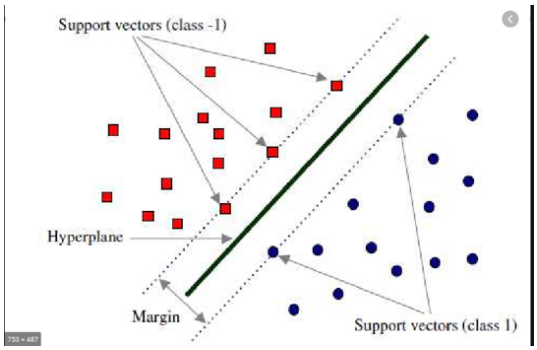


VGG16 + PCA + LDA as pre-processing and "Gini" criterion. In this method we notice that the pre-process was necessary because without it the results are really bad.

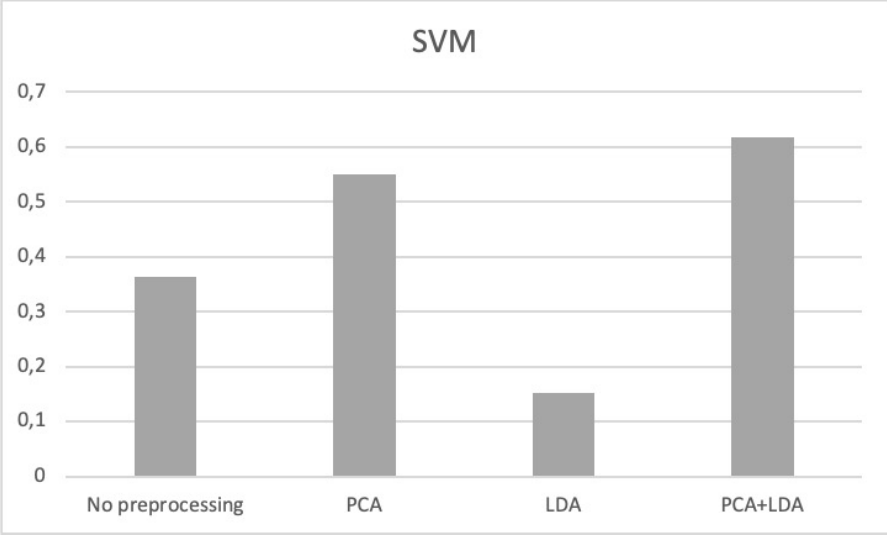


2.4 Support Vector Machine

This method looks for a hyperplane or set of hyperplanes in a very high (or even infinite) dimensional space that can separate the different classes. A good separation between the classes will allow a correct classification.



To calibrate the parameters we used the greadsearch funcion by Python. In this way and using PCA+LDA as pre-processing we obtain a 60% of score.



Chapter 3

How affects the minimum number of images per person?

We now consider changing the minimum number of photos per person. In this table we compare the results obtained considering a minimum of 20 photos per person, which implies 62 people in our dataset, and 30 images per person, which means 33 people in the dataset. For this comparison, we fixed a maximum of 50 photos, and we show in these two different columns the best score obtained with several models after applying different pre-processing techniques.

Method	Min images: 20	Min. images: 30
K-Neighbour	0.4777	0.5355
Nearest Centroid	0.5639	0.6256
Secuential Neural Network	0.3520	0.4391
Multi-Layer Perceptron	0.6007	0.6475
Random Forest	0.5930	0.6284
SVM	0.6162	0.6790

We can see that, in most of the cases, the best results are given by SVM and Mutilayer perceptron, achieving around the 60% of accuracy. As expected, by increasing the minimum number of photos to 30, the results improve in all the different cases.

Chapter 4

Conclusions

4.1 Best model

After trying different combination of options, we obtained our best model using SVM. To do this, we selected those people with a minimum of 30 photos and increased the maximum number of images to 55. Moreover, in these images we only did the initial simple crop. For the preprocessing, we made a transformation using PCA with 100 components and then we applied LDA. After that, we carried out a search for the best parameters, obtaining nearly a 70% of accuracy.

4.2 Are these results good enough?

Observing the results and despite being improvable, we consider that we achieved an acceptable accuracy taking into account the difference between photos of the same person in the dataset.

4.3 Possible improvements

As possible lines of improvement in which we would have liked to advance if we had had enough time, we propose a deeper study of different parameters in the selected models to improve them as well as looking for other types of models that we have not been able to study. We would also have liked going into detail in neural networks, since we tried some of them but it would be necessary to learn more about neural networks to obtain better results.

It would also be interesting to study data augmentation techniques in order to train neural networks and improving results, because our data is not as large as it would be necessary to obtain better results. These techniques expand the training dataset by creating modified versions of the images applying different transformations like flips, zooms, rotations or brightness variations. The Keras deep learning library provides the ImageDataGenerator class, which allows us to use data augmentation.

4.4 Greetings

As long as we were getting deeper into this project, we were learning about image treatment, as well as programming machine learning models in Python, which until now we had not seen. We have discovered the potential problems that may arise when dealing with a real problem, such as diversity in the database. And finally, we've learned about image preprocessing techniques like FisherFaces and how important they are.

Last but not least, we would like to thank Management Solutions for giving us the opportunity to prove our skills, specially the assistants for their help and for the proposal of this problem, which is, by the way, a very modern problem. Also, we thank very much to the Universidad Complutense and Valeriy for the organization of these days.