Bachelor in Data Science and Engineering | Year 2

Statistical Learning Course

FACIAL RECOGNITION ALGORITHM PART 2

Principal Component Analysis

K-Nearest Neighbours

Fisher Discriminant Analysis

Support Vector Machine

Ricardo Chávez Torres 100406702

Bernardo Bouzas García 100406634

FACE RECOGNITION

As this is the final part of our face recognition algorithm, we aimed to get the best classification function possible. There are lots of different algorithms that we could use to achieve this, so we studied some of these possibilities and tried to obtain the best results.

We studied both the **K-Nearest Neighbours** and the **Support Vector Machine** classification methods combined with **Principal Component Analysis** or both PCA and **Fisher Discriminant Analysis**.

For the correct evaluation of each method, we created a data frame that contained every combination of the different hyperparameters that are needed in each method, and then we used **k-folds cross-validation** over the data, so we could get the different accuracies of every combination of hyperparameters when classifying.

In order to perform this validation, we created 6 partitions of the data set, each of them with one random image of every person in the dataset, and then chose one of those partitions to be the test set, while the 5 left were used as training set.

PERFORMING THE CROSS-VALIDATIONS

We chose to evaluate the following algorithms:

- PCA+KNN
- PCA+FDA+KNN
- PCA+SVM
- PCA+FDA+SVM

Principal Component Analysis with K-Nearest Neighbours

The complete information about this procedure can be founded in the document uploaded to the first part of the assignment.

Since that report, we added some more distances to the evaluation in the KNN algorithm, to see if our previous results could be improved.

The list of distances that we decided to test is the following:

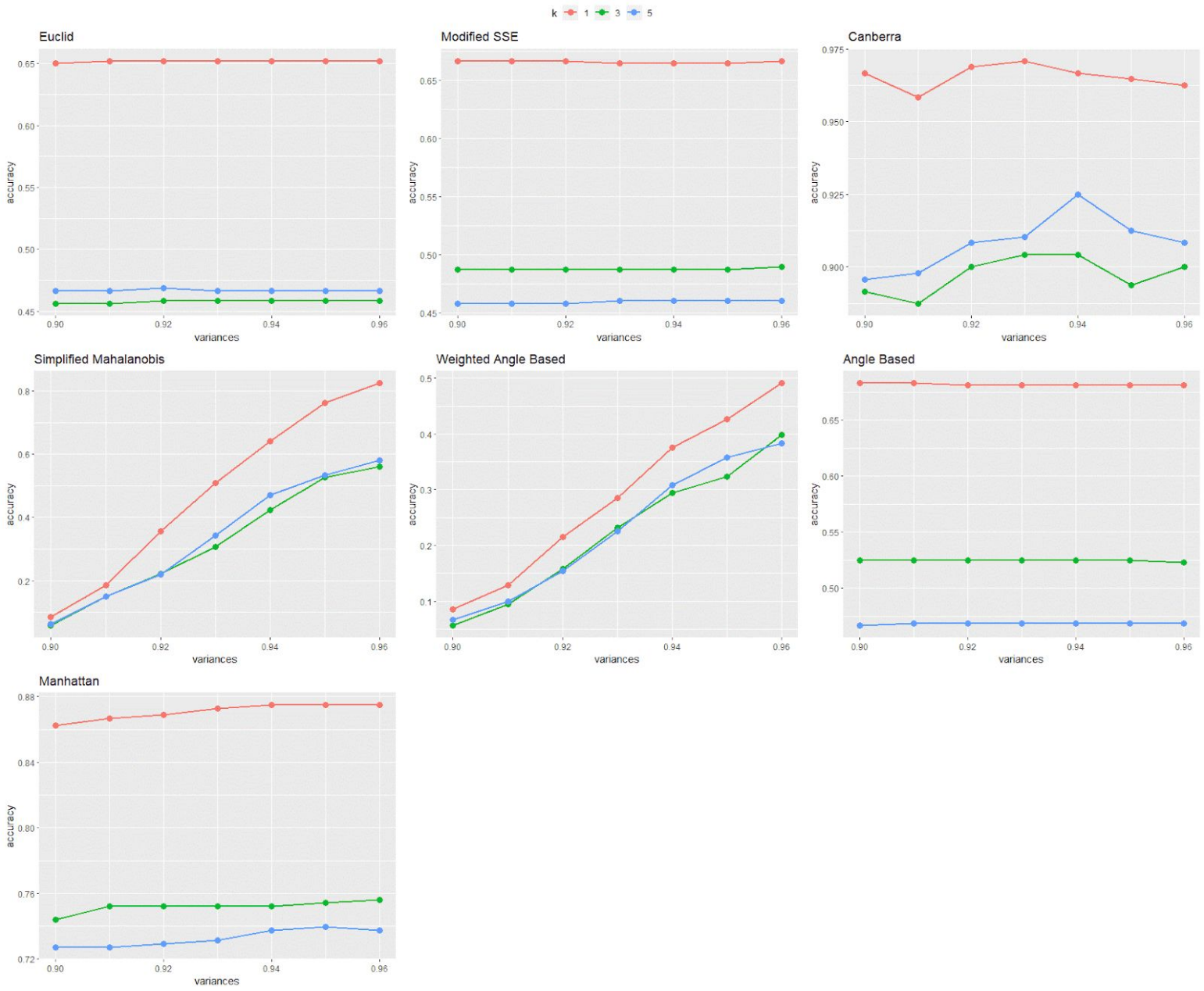
- **Euclidean**
- **Modified SSE**
- **Canberra**
- **Simplified Mahalanobis**
- **Weighted Angle Based**
- **Manhattan**
- **Angle Based**

As explained before, these distances were seen in *V. Perlibakas / Pattern Recognition Letters 25 (2004) 711–724 "***Distance measures for PCA-based face recognition***".

We studied **variances** in {0.90, 0.91, 0.92, 0.93, 0.94, 0.95, 0.96}. The reason for this constraint is because, in the previous study, we observed that a really nice variance using Canberra distance was approximately 0.95, so we decided to check the values that were similar to it.

We also studied for **k = {1,3,5}**. Using higher values for k than 6 is nonsense because in every class we only have 6 different images.

We evaluated the results plotting the accuracy for every combination of parameters:



We saw that the distances that achieved the best results are the **Canberra** with all the variances and **k = 1**, then the **Manhattan** distance followed.

This first approach gave us a good starting point to estimate which accuracy we could get from this model.

Principal Component Analysis with Fisher Discriminant Analysis and K-Nearest Neighbours

The change of this procedure is that we added an extra step after the PCA and previous to the KNN: **we projected the data** using a Fisher Discriminant Analysis.

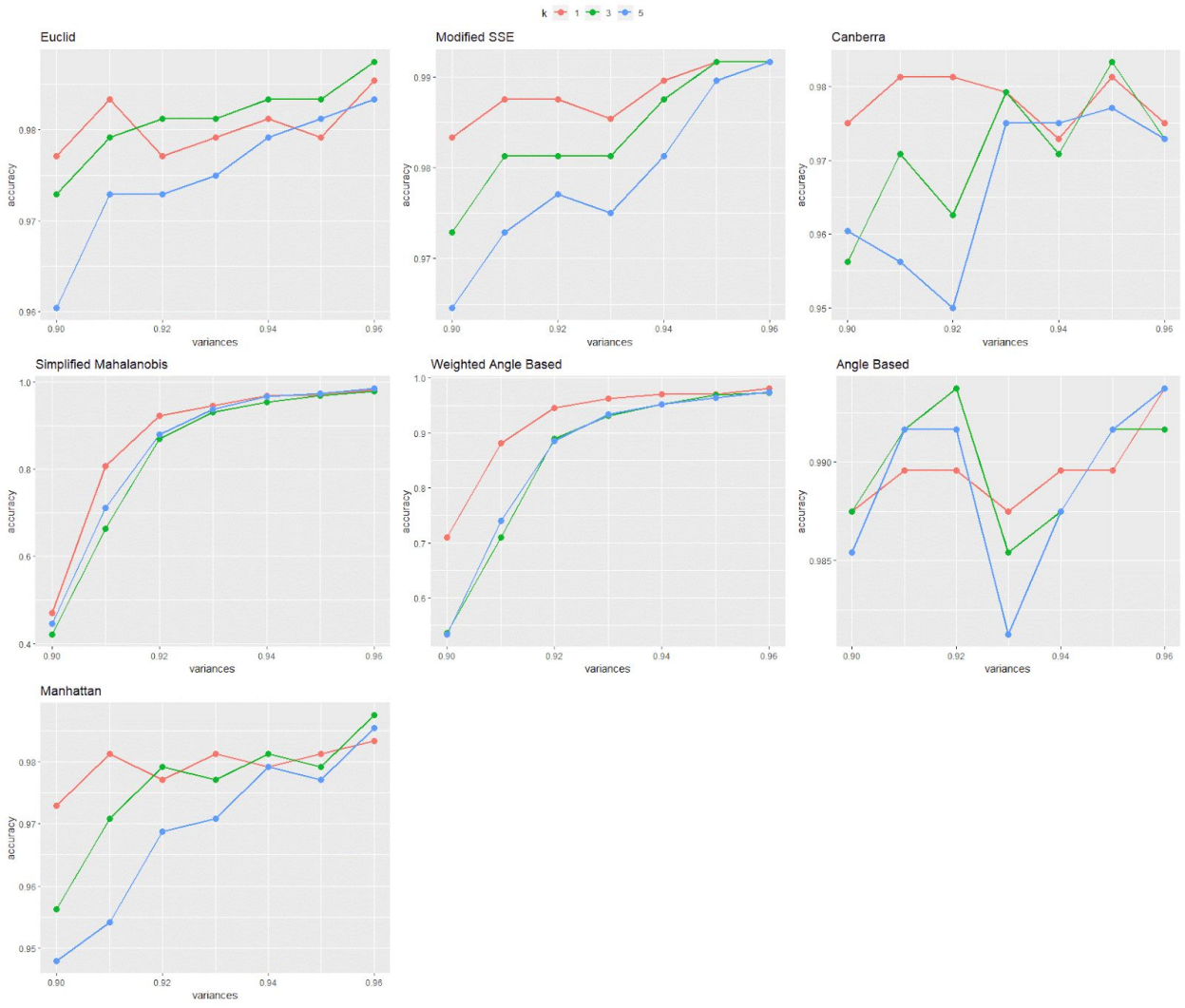
This technique allowed us to project the data set into another one in which two conditions were implemented to try to get a better result (following our thoughts):

+ **Distances** between different classes are maximized.

− **Variance** within the same class minimized.

We studied the same set of hyperparameters and the results were totally different, now every type of distance has at least one combination that gets **more than 95% accuracy** and sometimes when k is different than 1 the model gets good results.

In the following plots, we can appreciate that the distance that seems to get the best results is the **Angle based** distance.



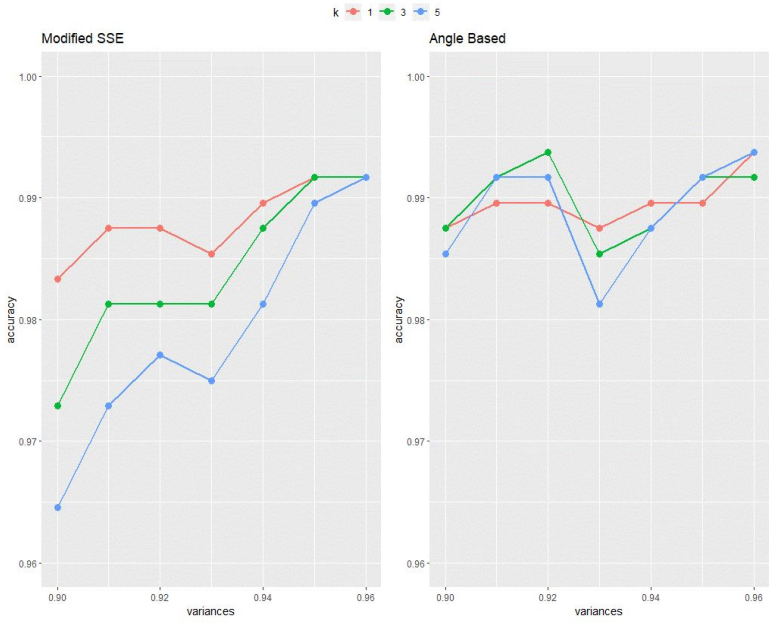
```
        angle_based            canberra             euclid
                 16                   0                  0
          manhattan         modified_SSE simplified_mahalanobis
                  0                   9                  0
weighted_angle_based
                  0
```

distances	k	variances	accuracy
angle_based	3	0.92	0.9937500
angle_based	1	0.96	0.9937500
angle_based	5	0.96	0.9937500
angle_based	3	0.91	0.9916667
angle_based	5	0.91	0.9916667
angle_based	5	0.92	0.9916667
modified_SSE	1	0.95	0.9916667
modified_SSE	3	0.95	0.9916667
angle_based	3	0.95	0.9916667
angle_based	5	0.95	0.9916667
modified_SSE	1	0.96	0.9916667
modified_SSE	3	0.96	0.9916667
angle_based	3	0.96	0.9916667
modified_SSE	5	0.96	0.9916667
angle_based	1	0.91	0.9895833
angle_based	1	0.92	0.9895833
modified_SSE	1	0.94	0.9895833
angle_based	1	0.94	0.9895833
angle_based	1	0.95	0.9895833
modified_SSE	5	0.95	0.9895833
angle_based	1	0.90	0.9875000
angle_based	3	0.90	0.9875000
modified_SSE	1	0.91	0.9875000
modified_SSE	1	0.92	0.9875000
angle_based	1	0.93	0.9875000

But in order to prove this, we also show here the combination of hyperparameters that got the top 25 best accuracies.

We see that in this top 25, the **angle based distance** was the one that appeared the most followed by the **modified SSE**

So we compare both and keep them to see which can get a **threshold that detects impostors better**. This will be explained posteriorly.



Principal Component Analysis with Support Vector Machine

Using the previous calculations established for PCA method, we tried combinations of the SVM hyperparameters, focused only on trying to improve the performance of the previously tested models.

We studied the classification type, kernel options, and gamma values combinations.

When trying to perform the prediction with polynomial and sigmoid kernels we had some troubles. The fact that the **linear kernel** gave us the best accuracy (0.86) made it clear for us: it was not worth it to study other than the linear kernel saving much time and resources, as it doesn't need a gamma value, so we omitted that extra loop.

Principal Component Analysis with Fisher Discriminant Analysis and Support Vector Machine

The result of this procedure was quite similar to the previous one: FDA improved the performance; however, the accuracy didn't exceed other results: 0.91 as maximum.

Our conclusion on using the Support Vector Machine over the KNN algorithm is that is not the most best-performance method, following two clauses:

- ❏ We weren't able to get a better result than with KNN, even though we applied the FDA.
- ❏ Using SVM makes significantly harder for the model to classify the impostors.

In relation to the second conclusion, we had an extra idea:

Two algorithms could be made, one that **detected impostors** (with one-classification) and another that **classified** (with C-classification).

The problem is that this is computationally less efficient than using a KNN where, in addition, the threshold to classify impostors is implemented in the function, so this hypothesis was discarded.

THRESHOLD

Once we chose the method that we wanted to use (**PCA+FDA+KNN**) due to the reasons explained before, we selected the hyperparameters:

- Variance: ~0.95 (180 eigenvectors)
- Distance type: angle based
  - *Note: if the threshold was easier to calculate, we would use modified SSE*
- K: 1

At this moment, was time to chose the threshold in order to **detect possible impostors**.

To do this, we calculated the distance of each observation (once we did the PCA and FDA) with every other person of the dataset, and took two highly important distances:

- The **minimum distance** to an observation of the **same class**.
- The **minimum distance** to an observation of a **different class**.

This is because the threshold will trigger and classify an impostor as an observation that belongs to some class of the dataset if the distance to the impostor (distance to an observation of a different class) is **less than the threshold**. In this situation we would have a **False Acceptance**.

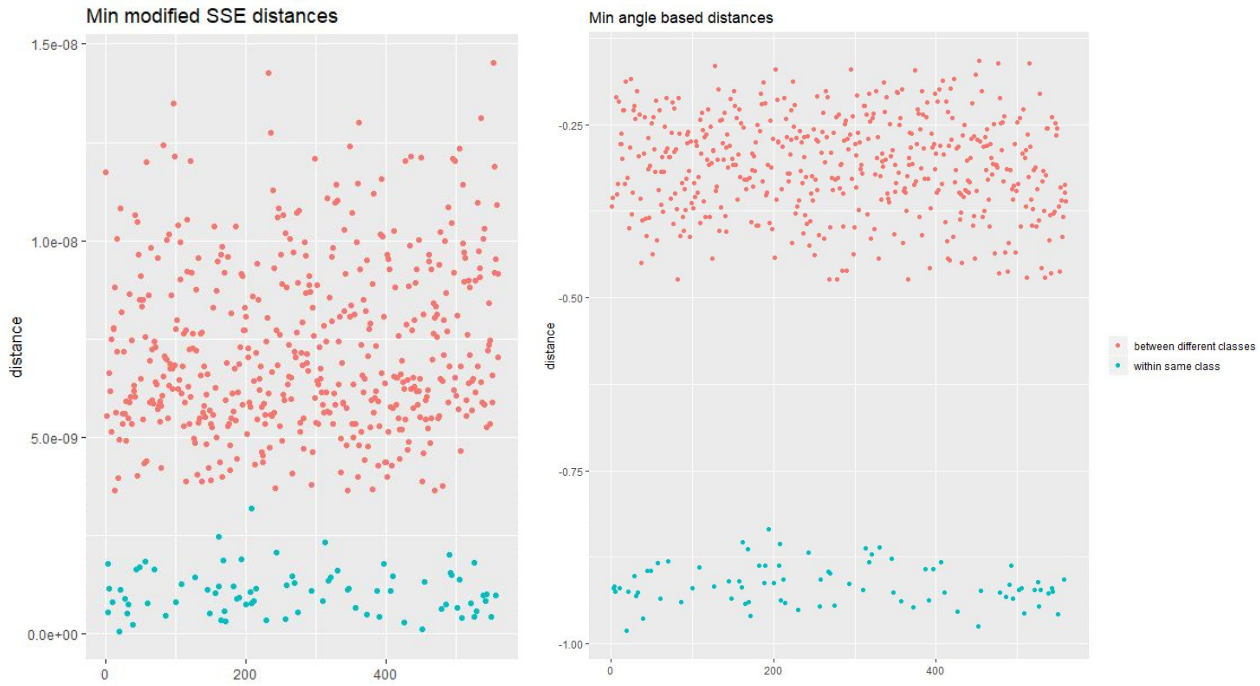
This would mean that the smaller the threshold, the less False Acceptances we will get.

However, if we try to classify an observation that belongs to some class of the training data, but the distance to the closer observation of the same class in the training data is lower than the threshold, the model will classify this as an impostor, we would have a **False Positive**.

So the threshold must be in between these two distances:

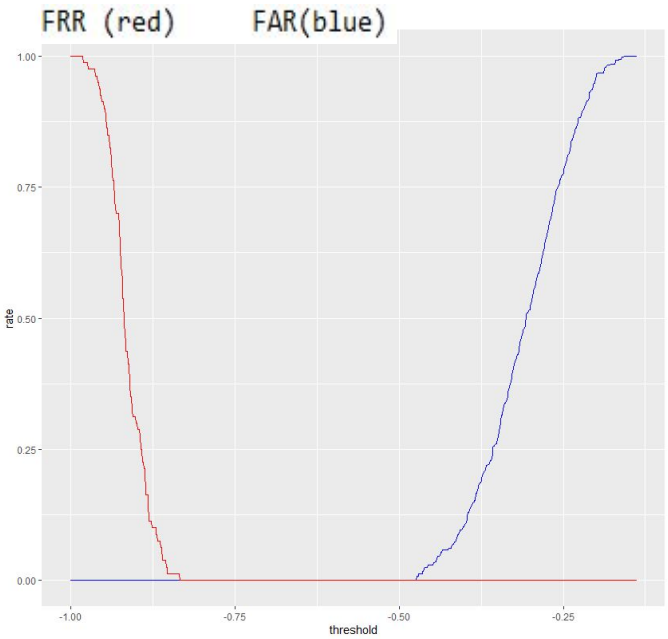
Minimum distances of observations that belong to the **same class**	THRESHOLD VALUE	Minimum distances belonging to observations of a **different class**.

In order to see this, we plotted these distances for both our **best model** classifying (**angle based**) and our model with the **second-best distance** (**modified SSE**)



We could observe that the number of values that the threshold could take was higher with the **angle based distance**.

After this, we plotted the **FRR (red)** and the **FAR (blue)** for this model depending on the threshold This allowed us to choose the threshold wisely.



Inside our model training set picking a threshold between -0.75 and -0.5 would not classify any observation FA nor FR.

One possible explanation for this to happen is that the Fisher projection was done with the training set.

For future data, picking the threshold right in the middle of that hole between -0.75 and -0.5 might classify **too many observations as impostors**, not only because of the Fisher projection but because we had **more distances between observations of different classes than distances within the same class**, so the estimation of the real FRR is probably more accurate than the real FAR .

This is why we choose as the **threshold -0.59.**

OUR MODEL

Finally, and having explained the reasons of each decision, these are the selected parameters for our model that combines Principal Component Analysis, Fisher Discriminant Analysis and K-Nearest Neighbours:

- Variance ~0.95 (180 eigenvectors)
- K = 1
- Distance = angle based
- Threshold = -0.59