Reducing dimensionality using Principal Component Analysis and Factorial Discriminant Analysis

Reconnaissance des Formes

Iacob Sergiu

19th March 2020

Contents

Chapter 1

Introduction

1.1 Context

This report is the result of a university assignment. It aims to prove that the student understood the motivation, goals and means of studying pattern recognition. Therefore, this is a way of summarizing a series of observations and experiments done on images containing shapes.

1.2 Motivation

Visualising data is a critical and essential point of analysing the data. But we can't visualise, for example, 6D data, so it would be nice if we could *reduce the dimensionality* of our problem. Luckily, there are multiple solutions such as Principal Component Analysis that, according to Wikipedia 2020, help us "convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables". We can already see advantages to this: having less variables will make visualisation possible and will also make models' training faster.

1.3 Goals

Our goal will be to successfully apply methods such as PCA on a training data set in order to reduce the space of representation for some data. On our example, we'll work in a 2 dimensional space and we'll try to find an axis that separates our points the best. We'll also be interested on seeing how a classification algorithm will do if we apply this pre-processing on it.

CHAPTER 2

EXPERIMENTS

2.1 VISUALISING THE DATA

Let us first look at our data. We're working with 2D data, so we can consider them as points. There are 300 points in the training data set, as well as 300 points in the data set. All of these points are *labeled*, so we are dealing with a *supervised classification* problem. We have 3 classes and each class has 100 points in each dataset.

In the figure below, we can see how the points are arranged. Each color stands for a class, and the *mean point* for each class was drawn using a filled circle.

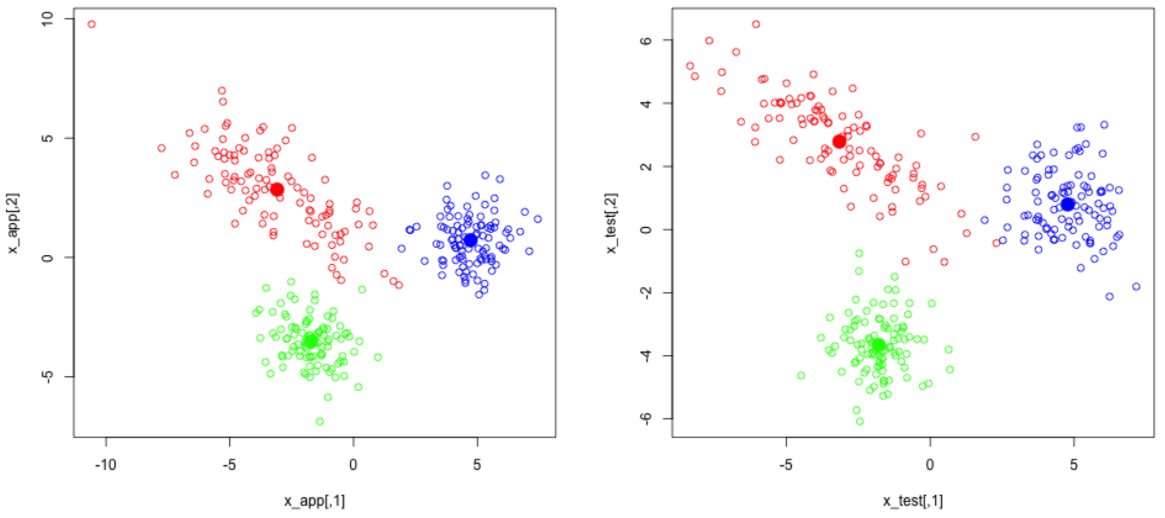


FIGURE 2.1
Train data on the left, test data on the right

Visualising the figures, we see that the points are similarily dispersed. The red points (class 1) go from the upper left to the middle, the green points (class 2) tend to be in the middle-bottom and the blue ones (class 3) tend to be in the middle-right. It also seems that the points in the test dataset are more scattered, while in the train dataset they tend to have a stronger "cohesion". As a result, it seems easier to draw decision boundaries for the train dataset, but overall they aren't very hard to separate.

2.2 PRINCIPAL COMPONENT ANALYSIS

2.2.1 FINDING THE BEST COMPONENT

We'll use PCA to find the best axis to project our data from. That should be the one that allows the data to best be separated, or equally said, *discriminated*. To find that, we'll calculate the *covariance matrix*, afterwards we'll choose the *eigen vector* corresponding to the highest *eigen value* for the covariance matrix.

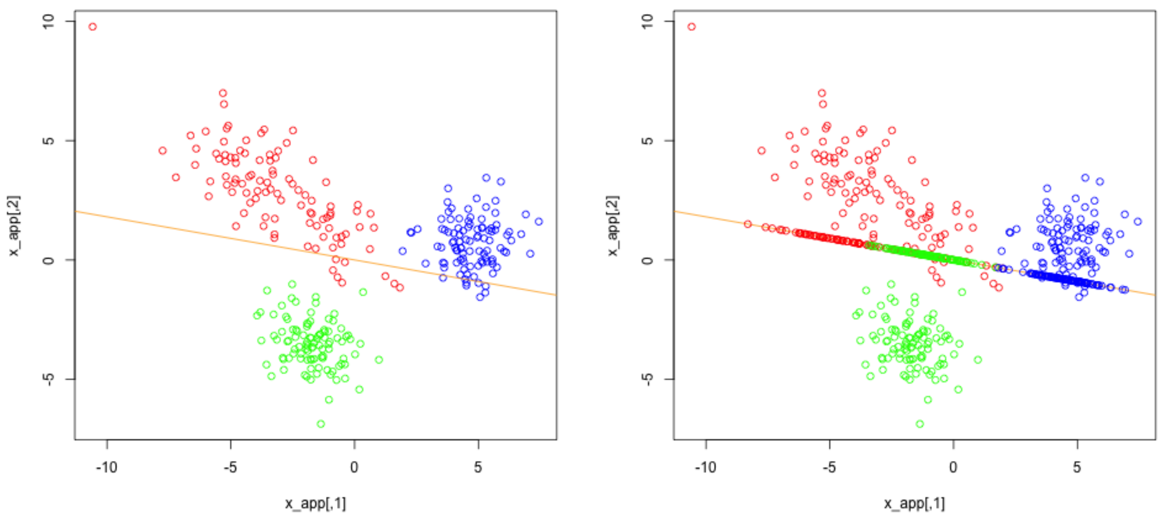
```r
# calculate covariance
covariance <- cov(x_app)
# get the eigen vectors and values
Vp <- eigen(covariance)
pente <- Vp$vectors[2, 1] / Vp$vectors[1, 1]
# visualise the best axis to project the data on
plot (x_app, col = trainColors)
abline(a = 0, b = pente, col = "orange")
```

LISTING 2.1
Calculating the most discriminating axis

Having this, we can now *project* our training data on the axis:

```r
ScalarProduct_app_ACP <- x_app %*% (Vp$vectors[, 1]) / sqrt(sum(Vp$vectors[, 1] * Vp
    $vectors[, 1]))
appProjections <- array(dim = dim(x_app))
appProjections[, 1] <- ScalarProduct_app_ACP * Vp$vectors[1, 1]
appProjections[, 2] <- ScalarProduct_app_ACP * Vp$vectors[2, 1]
```

LISTING 2.2
Calculating the most discriminating axis



(A) Best axis according to PCA

(B) Train data projected on the axis

If we were to separate the training data based on their projections, we wouldn't have much luck. Separating the blue class from the others doesn't pose a big problem, but the red and green classes overlap each other in a big proportion.

2.2.2 CLASSIFYING DATA USING PCA AND LDA

We can now use Linear Discriminant Analysis to predict our data, in combination with PCA. We'll just train the LDA on the scalar products calculated above, as those resulted from PCA reducing our problem's dimensionality. But before moving on, let's see how the test dataset looks like when it's projected on the axis we calculated:

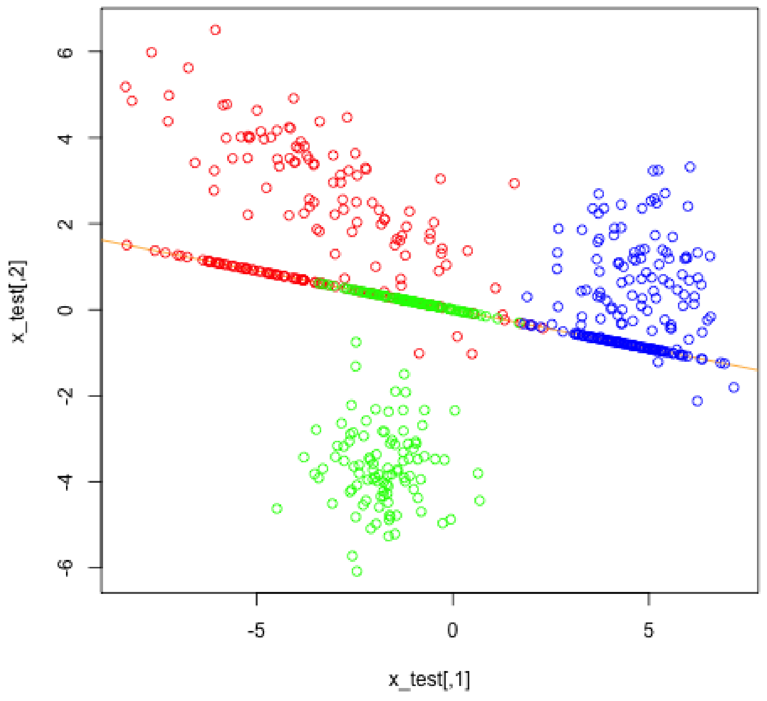


FIGURE 2.3
Test data projected on the PCA axis

We can see that the same thing happens for the test data. The red class overlaps the green class in a big proportion, so we can expect the test instances from the red class to not be properly predicted. This is the script that calculates the predictions, based on the results from PCA:

```r
ScalarProduct_test_ACP <- x_test %*% (Vp$vectors[, 1]) / sqrt(sum(Vp$vectors[, 1] *
    Vp$vectors[, 1]))
x_app_ACP.lda <- lda(ScalarProduct_app_ACP, classe_app)
assigne_test <- predict(x_app_ACP.lda,  newdata = ScalarProduct_test_ACP)
```

LISTING 2.3
Training LDA with PCA

Below we can see the results of the prediction. In the figure below 2.4 and the following figures that contain predictions, for each point, the color represents its true class, and the shape represents our prediction. Therefore, correct predictions are: red squares, green circles and blue triangles. Anything else represents an error, a missclassification.

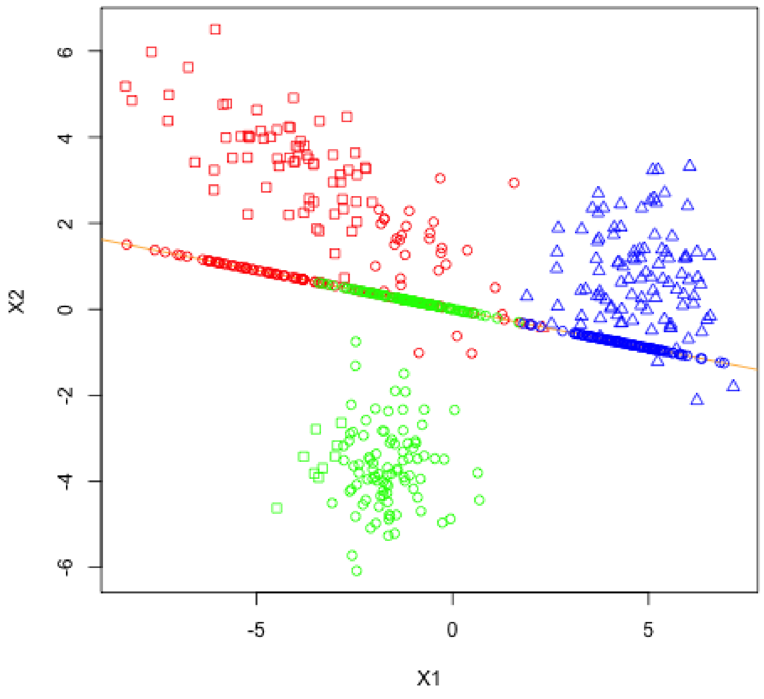


FIGURE 2.4
Test predictions, LDA with PCA

As we concluded from the figure above 2.3, we got a bad accuracy for the red points: only 69% of them were classified correctly, because of the overlapping. For the green class, the result is satisfying. The blue class even has 100% accuracy, since the blue projections can easily be separated from the rest.

Method name	Red accuracy	Green accuracy	Blue accuracy	Total accuracy
LDA&PCA	69%	91%	100%	$\approx$ 87%

2.3 FACTORIAL DISCRIMINANT ANALYSIS

2.3.1 FINDING THE MOST DISCRIMINATING AXIS

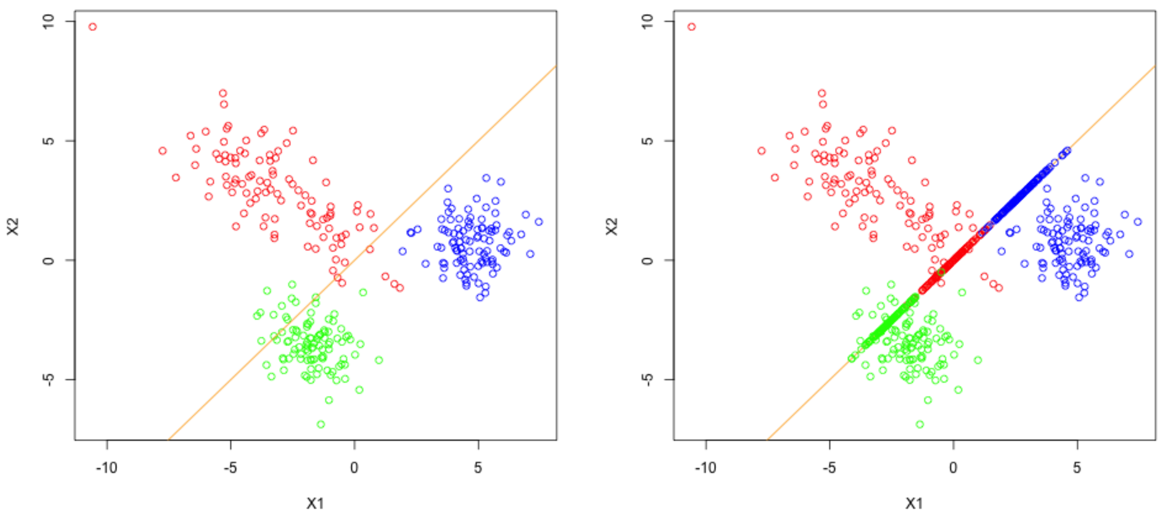
As Macaire 2020 states, in order to find the most discriminating axis using Factorial Analysis, we first need need some pre-calculations. Having the means shown in the first figures 2.1, we can calculate *intra-class* co-variance and *inter-class* co-variance. We then apply the same method of finding the "best" eigen vector for the intra-class co-variance matrix.

```r
trainMeans = array(dim = c(noClasses, dim(x_app)[2]))
for (i in 1:noClasses){
    trainMeans[i, 1] = mean(x_app[classe_app == i, 1])
    trainMeans[i, 2] = mean(x_app[classe_app == i, 2])
}
mean = colMeans(trainMeans)
# calculate covariances
S1 <- cov(x_app[classe_app==1,])
S2 <- cov(x_app[classe_app==2,])
S3 <- cov(x_app[classe_app==3,])
# intra-class, inter-class covariances
Sw = S1 + S2 + S3
Sb <- 0
for (i in 1:3)
    Sb <- Sb + (trainMeans[i,] - mean) %*% t(trainMeans[i,] - mean)

invSw <- solve(Sw)
invSw_by_Sb <- invSw %*% Sb
Vp <- eigen(invSw_by_Sb)
```

LISTING 2.4
Finding best discriminant axis using FDA

Here's the axis found by FDA, along with the projections from the train dataset on that axis:



(A) Best axis according to FDA

(B) Train data projected on the axis

We can definitely see a big improvement. The overlapping problem is much, much lower. So, intuitively, we should expect better results if we were to combine FDA with LDA.

2.3.2 LDA WITH FDA, COMPARISON WITH PCA

To classify our test data using AFD, we use the exact same code, but with the scalar products that resulted from FDA. Here are the results:

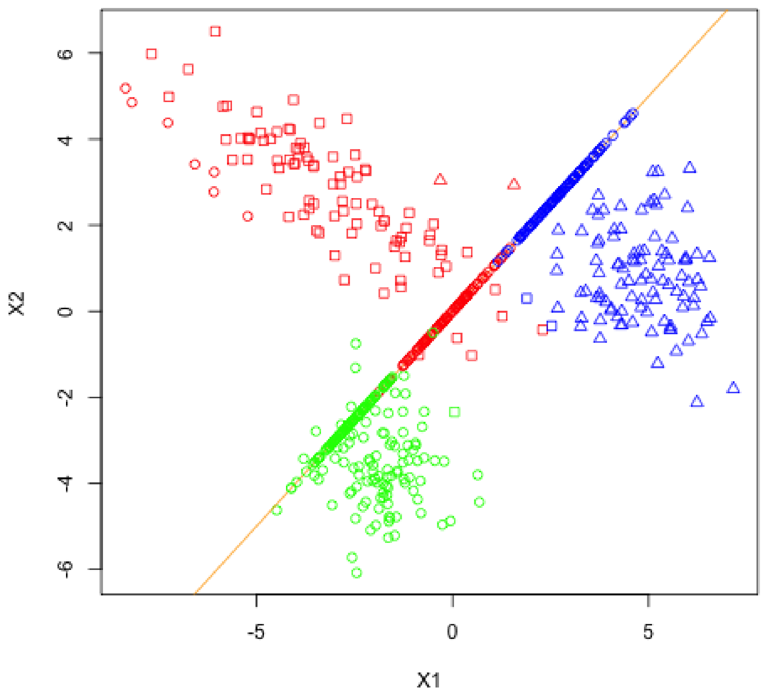


FIGURE 2.6
Test predictions, LDA with FDA

Method name	Red accuracy	Green accuracy	Blue accuracy	Total accuracy
LDA&PCA	69%	91%	100%	$\approx$ 87%
LDA&FDA	91%	99%	98%	96%

Comparing FDA with PCA, we have an improvement of 9% on the total accuracy, which is a lot better. This is mostly due to the fact that the red class is now classified much better, because we have a lot less overlapping. The green instances have also seen a 7% increase in accuracy, while we lost 2% for the blue ones. But, overall, it's much better.

2.4 TRAINING LDA DIRECTLY ON THE TRAINING DATASET

For a final comparison, we can try to train the LDA directly on the training dataset. The code is the same, we just train the LDA algorithm directly with our points: `x_app.lda <- lda(x_app, classe_app)`.

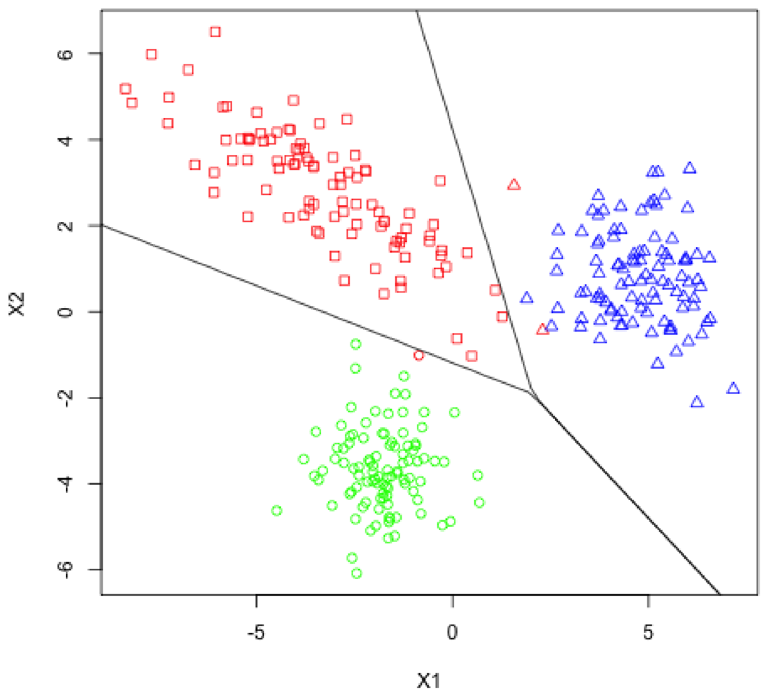


FIGURE 2.7
Test predictions, LDA trained directly on the training dataset

The figure above shows the decision boundaries for the LDA algorithm. Here are the results:

Method name	Red accuracy	Green accuracy	Blue accuracy	Total accuracy
LDA&PCA	69%	91%	100%	$\approx$ 87%
LDA&FDA	91%	99%	98%	96%
LDA&original data	97%	100%	100%	99%

There's a visible increase in accuracy, and that's normal. Overall, LDA trained on the original data will give a total accuracy of 99%, which is really good. It's normal – if the algorithm has more data to work with, it will provide better results. But this is a trade-off that the decision-maker must settle. If we want our algorithm to be faster, or to avoid the *curse of dimensionality*, we should reduce the dimensionality of the problem (using, for example, PCA or AFD, like we did here), so the algorithm will train faster. But, at the same time, it will most likely give less accurate results – however this is a question of overfit vs. underfit.

Chapter 3

Conclusion

By using methods such as Principal Component Analysis and Factorial Discriminant Analysis, we were successfully in reducing our number of features. We have seen that applying this pre-processing negatively impacts our accuracy (for our case), but it also simplifies the visualisation of data. The most important thing is that we managed to reach our most important goal: find the most discriminant axis for our points, using each method, then projecting our points on that axis.

Bibliography

Wikipedia (2020). *Principal Component Analysis*. Wikimedia Foundation. URL: https://en.wikipedia.org/wiki/Principal_component_analysis#Dimensionality_reduction (visited on 27th Feb. 2020).

Macaire, Ludovic (2020). *Réduction de la dimension de l'espace de représentation*. Université de Lille. URL: http://master-ivi.univ-lille1.fr/fichiers/Cours/M1IVI_RDF_Cours7a_2015.pdf (visited on 19th Mar. 2020).