SUPERVISED CLASSIFICATION USING DISCRIMINANT ANALYSIS

RECONNAISSANCE DES FORMES

Iacob Sergiu

11th 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

In order to be able to work with shapes, we must first acquire them. Most of the time, they are not given to us in their simplest form, but found in images with all kinds of backgrounds. They must be extracted as precisely as possible, making sure we differentiate them from the background as accurately as we can.

1.3 GOALS

We'll try to apply *binary segmentation* to a series of gray images. Our goal is to find an automatic method of doing this and identify objects and backgrounds, mainly focusing on the histograms of gray values.

CHAPTER 2

BINARISATION USING BAYES' THEOREM

2.1 VISUALISING DATA

The data we are going to use is present in a file of type `Rdata` and it consists of two parts: data for *training* and data for *testing*. Each dataset has a number of 75 instances (an instance being described by 2 *features*) which are divided (not equally) amongst 3 *classes*.

Having a *true class* assigned to each instance, we can treat this problem as a *supervised classification* problem. Moreover, we also have information about true *means* and *standard deviations*, so we can use these to make comparisons between our algorithms and the truth.

Let's take a look at how the data looks:

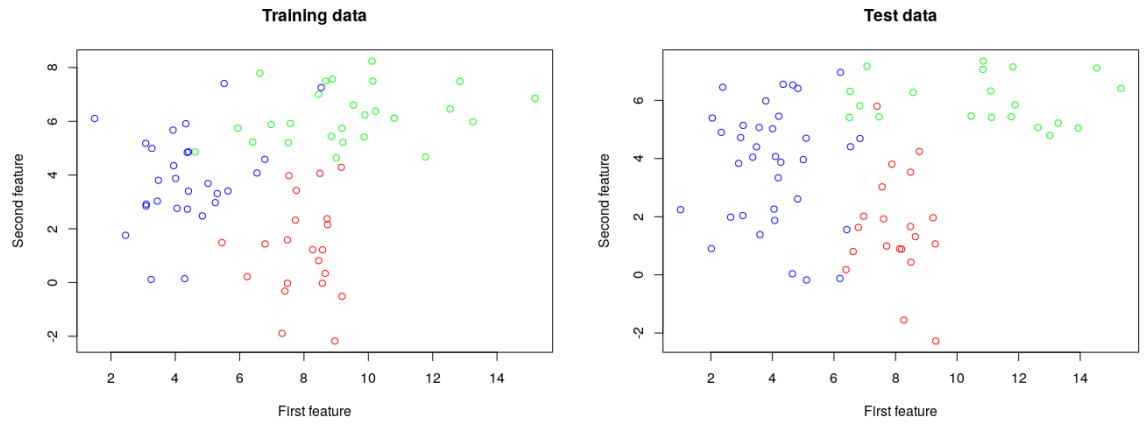

FIGURE 2.1
Plotted data. Train on left, test on right

```
data <- load(file='simul-2017.Rdata')
class_colors = c('red', 'green', 'blue')
couleur <- rep('red', n_app)
for (class_no in 1:3)
  couleur[classe_app == class_no] = class_colors[class_no]
# plot data
plot (x_app, col=couleur, main='Training data', xlab='1st feature', ylab='2nd
    feature')
for (class_no in 1:3)
  couleur[classe_test == class_no] = class_colors[class_no]
plot (x_test, col=couleur, main='Test data', xlab='1st feature', ylab='2nd feature')
```

LISTING 2.1
Loading and visualising data

We can notice more blue dots, which would mean instances are not equally distributed amongst classes. Calculating the class a priori probabilities, that is indeed true:

```
# P(w) = probability of an instance to be in the class "w"
class_probs_app <- 1:3
class_probs_test <- 1:3
for (class_no in 1:3){
    class_probs_app[class_no] = sum(class_no == classe_app) / length(classe_app)
    class_probs_test[class_no] = sum(class_no == classe_test) / length(classe_test)
}
class_probs_app
class_probs_test
```

LISTING 2.2
A priori class probabilities

The script above gives us the values $P(red) = 0.28$, $P(green) = 0.3466667$ and $P(blue) = 0.3733333$ for the training dataset and $P(red) = 0.2666667$, $P(green) = 0.2666667$ and $P(blue) = 0.4666667$, for the test dataset. The values corresepond to red, green and blue classes, in this order.

2.2 ESTIMATING MEANS AND COVARIANCE

2.2.1 CALCULATING MEANS

A first small step to analysing the data would be to look at the "centroids" for each class, that is calculating the means for each class.

```
# calculate means
means <- array(dim = c(3, 2))
for (class_no in 1:3){
    means[class_no, 1] <- mean (x_app[classe_app == class_no, 1])
    means[class_no, 2] <- mean (x_app[classe_app == class_no, 2])
}
# plot the means for train data
for (class_no in 1:3)
    couleur[classe_app == class_no] = class_colors[class_no]
plot (x_app, col=couleur, main='Means for train data', xlab='First feature', ylab='
    Second feature')
points (means, col=class_colors, p=8, cex=4)
```

LISTING 2.3
Calculating means

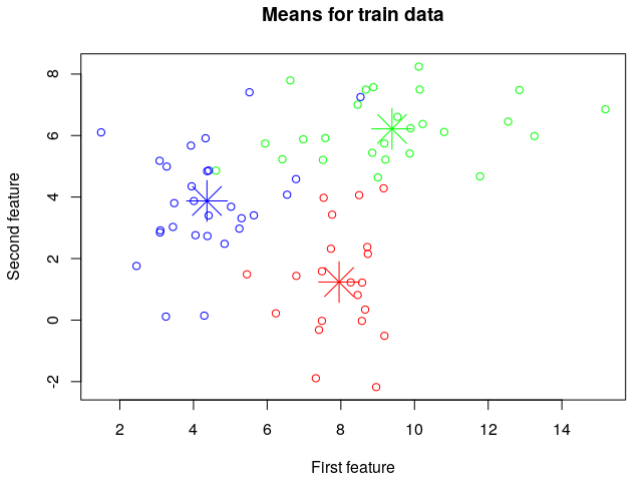

FIGURE 2.2
Visualising means with instances

True results

	Red	Green	Blue
X1	8	10	4
X2	1	6	4

My results

	Red	Green	Blue
X1	7.951986	9.389311	4.365176
X2	1.237545	6.218550	3.874318

The values[1] aren't so distant from the ground truth, so we can consider them insightful.

2.2.2 COVARIANCE MATRIX

Up next, we will calculate the covariance matrix. According to Wikipedia 2020a, covariance is a measure of how one feature changes in relation to another. Depending on the algorithm we're using, if two features are *strongly* correlated, it might be better to let one go, in order to reduce the *dimensionality* of the problem and make algorithms faster. But, for example, algorithms like Naive Bayes might actually benefit from correlated features.

```
sigma <- c()
for (class_no in 1:3){
    # number of instances = n_app = dim(x_app)[1]
    # number of features = dim(x_app)[2]
    sigma[[class_no]] <- matrix(1, dim(x_app)[2], dim(x_app)[2])
    for (i in 1:dim(x_app)[2])
    for (j in 1:dim(x_app)[2])
        sigma[[class_no]][i, j] <- cov(as.vector(x_app[classe_app == class_no, i]),
    as.vector (x_app[classe_app == class_no, j]))
}
```

LISTING 2.4
Calculating covariance

Below, we can find the script's result. For each class, we have calculated the *covariance matrix*. On the diagonal of each Σ_i matrix, we have the *variance* of each feature, and any other element $\Sigma_(j,k), j \neq k$, represents the correlation between the features j, k. Of course, these matrixes are symmetric.

$$\Sigma_1 = \begin{pmatrix} 0.95337723 & 0.09544548 \\ 0.09544548 & 3.26152120 \end{pmatrix}, \Sigma_2 = \begin{pmatrix} 5.8526867 & 0.6107413 \\ 0.6107413 & 1.0474417 \end{pmatrix}, \Sigma_3 = \begin{pmatrix} 2.0251769 & 0.7214545 \\ 0.7214545 & 3.0809579 \end{pmatrix}$$

$$\sigma_1 = \begin{pmatrix} 1 & 2 \end{pmatrix}, \sigma_2 = \begin{pmatrix} 3 & 1 \end{pmatrix}, \sigma_3 = \begin{pmatrix} 1.5 & 2 \end{pmatrix}$$

By Σ_i we denoted the covariance matrix, and by σ_i the ground truth regarding standard deviation.

2.2.3 ANALYSING DIAGONAL VALUES

Looking at the values on the diagonal and comparing them (! by not forgetting to raise the standard deviation, that is values from σ_i, to the power of 2, so we get variance), to the true values, we can see that overall they are pretty close. The only exception is the variance for the first feature, for the second class; those aren't very similar (≈ 5.8 vs. 3^2). To cite Wikipedia 2020b, variance "measures how far a set of (random) numbers are spread out from their average value". That being said, it gives us a sense of dispersity in our data.

Now let's apply the above observations to our data[2]. Before this, please note that in our plots above 2.1, the first feature X_1 was projected on the abscissa and the second feature X_2 on the ordinate. Looking at the red class and variance for the first atribute, we could that, for the first feature, the variance is not that high. That would mean that the red-class instances, projected on the x-axis, would be "gathered" around. If we look at the image, that is, indeed, true!

[1] The means were calculating only on the training dataset, not on all of the instances
[2] The covariance matrixes were calculated on the training data, so the observations we made are for that dataset

Thinking the other way around, if we look at the figure2.1 at X_1 for the green class, we can notice that the projections on the x-axis for the instances are quite dispersed. That would mean that the variance for that case is higher. And that is, once again, true (variance of ≈ 5.8 in our results), proof that our algorithm makes the cut.

2.2.4 ANALYSING CORRELATION BETWEEN FEATURES

Let us now note $\Sigma_i(j, k) = x$, a correlation between two features. Further analysing each possible value for x, we must take into account the following observations: if x is positive, then if the feature j increases, then so does the value for k. If x is negative, then if feature j increases, then the value for k decreases. If x is 0, then the features are not correlated. We say that two features j and k are strongly correlated when the x is large.

CHAPTER 3

CONCLUSION

Bayes' theorem relieves us from having to manually search for the best threshold. It provided results close to what we were looking for, with low error rates. It can be used in a multitude of scenarios as a strong classification tool, this report presenting just one simple example.

However, when segmenting the digits, for example, we did rely on already existing perfect segmentations in order to calculate probabilities like $P(w)$. Without those, Bayes' theorem could not have been applied. This is, however, a step in the right direction, especially coming from a method where we were manually selecting the threshold based on the histograms of gray values.

BIBLIOGRAPHY

Wikipedia (2020a). *Covariance*. Wikimedia Foundation. URL: https://en.wikipedia.org/wiki/Covariance (visited on 11th Mar. 2020).
– (2020b). *Variance*. Wikimedia Foundation. URL: https://en.wikipedia.org/wiki/Variance (visited on 11th Mar. 2020).