Using basic pattern recognition techniques to solve classification and dimensionality reduction problems.

Diego RODRIGUEZ

Universitat Pompeu Fabra, 2017

Abstract

In this report, basic pattern recognition (PR) techniques such as the Bayesian Theorem, Multivariate Gaussian Mixture Model and Principal Component Analysis are used to solve exercises from Seminar 1 of the Pattern Recognition class.

1 Fundamental Concepts

Find an example in order to explain all of these concepts:

1.1 Feature

Let's imagine we want to construct a handwriting digit recognizer. Somehow using advanced pattern recognition techniques we are able to indentify the number of straight horizontal lines in each sample containing a number. These straight horizontal lines are **features**.

1.2 Supervised Learning

An example of supervised learning is a prediction model. Imagine a dataset where you want to predict if a house is going be sold based on its price and the average income of a household of the zone where the house is, such as:

House price	Average income	Was it sold?
120000	24000	Yes
135000	30000	Yes
180000	24000	No

You could feed a learning algorithm to learn about the n features that contribute to a house being sold and try to predict if it's going get sold.

1.3 Classification

Try to predict if a picture contains a cat or a tiger.

1.4 Regression

Try to predict stock prices or estimate the remaining battery left in a phone based on several features.

1.5 Unsupervised Learning

Try to classify n amount of people into m groups. For example, you have a dataset of height and weight of your clients and you want to create the best fit for S, M, and L kind of clothes for them. Clustering would be a good approach.

1.6 Clustering

Clustering is an unsupervised learning technique used to automatically classify unlabeled data into groups, also known as clusters. After that, this new data can be used to improve the algorithm.

2 Bayesian Theorem

Walking through the jungle, you see a large feline shadow. However, you can not identify what animal it is. You know that, in this area of the jungle, 90% of felines are not dangerous cats whereas the other 10 % are pumas . In addition, you know that 80% of cats are smaller than the animal you are seeing whereas 85% of pumas has a similar size. Would you run?

Reminder: We define Bayes Theorem as follows $p(A|B) = \frac{p(B|A)p(A)}{p(B)}$, where A, B are events and $p(B) \neq 0$

Let A denote the event of feline being dangerous. Let B denote the event of watching a big feline with similar size to pumas and bigger than 80% of cats. We know that the probability of any random given feline of being a puma is 0.1 and any random feline of being a cat is 0.9. Also, we know that 85% of pumas can be a big feline like the one watched and that at most, 20% of cats are like that big feline:

$$p(A) = 0.1$$

$$p(B) = 0.1 \cdot 0.85 + 0.9 \cdot 0.2 = 0.265$$

Then, the probability of watching a big feline (B) **knowing** that the feline is a puma (A) is defined as:

$$p(B|A) = 0.85$$

Using Bayes Theorem we can calculate the probability of a feline being a puma given the event that we see a big feline: $p(A|B)$

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)} = \frac{0.85 \cdot 0.1}{0.265} = 0.32$$

Indeed, I would run.

3 Gaussian Model

Following the last example, assume that the variable size is continuous.

- Draw a figure (approximated) of the distribution functions p(size — cat) and p(size — puma). To do so, assume that the size of cats and pumas follow a normal distribution where cats size tend to be smaller than pumas. Moreover, you know that the size of different cat species use to be more variable than in the case of pumas.
- Over the same figure, and following the Bayesian Decision Theory, draw the 2 decision regions which classify cats and pumas with respect to their size. Draw also the region which corresponds to the decision error.

To do the figure I used Matlab. I implemented a script that assumes the values of $\mu_1 = 40$ and $\mu_2 = 70$, $\sigma_1 = 12$ and $\sigma_2 = 7$. Where the sub-indexes 1 and 2 denote the cat and puma population parameters respectively.

Using the script `exercise_3.m` with $\mu_1 = 40$, $\mu_2 = 70$, $\sigma_1 = 12$ and $\sigma_2 = 7$:

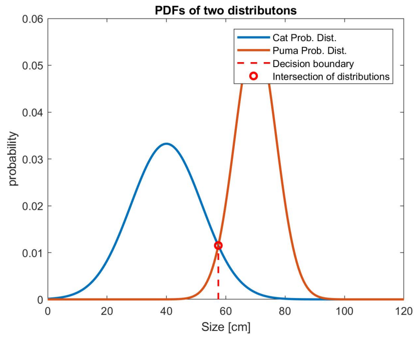


Figure 1: Probability density functions of the two feline distributions

We know that the intersection point of both distribution located **in between**, corresponds to the point of maximum uncertainty because there is a

50% chance of being one population or the other just based on the size variable. To the left of the decision boundary we predict cats, and to the right we predict puma; the more a value approaches the peaks of both distributions the higher the confidence.

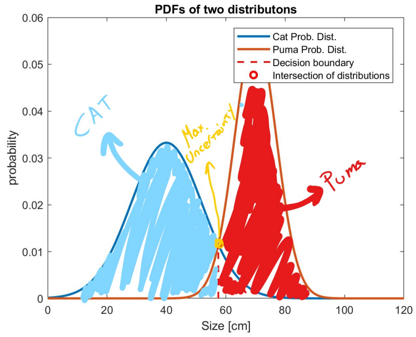


Figure 2: Prediction Regions

4 Same variance gauss model

Assume that probabilities $p(size|cat)$ and $p(size|puma)$ follow two normal distributions $N(\mu_1, \sigma)$ and $N(\mu_2, \sigma)$ respectively. Given that $p(cat) = p(puma) = 0.5$ and $\sigma_1 = \sigma_2$, find the size value x* which corresponds to the decision boundary classifying cats or pumas. Define x* as a function of μ_1 and μ_2.

Using the same variances $\sigma_1 = \sigma_2 = 12$, the size would be 55cm as shown on the figure:

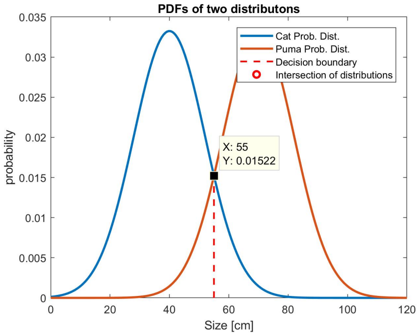


Figure 3: Using same variance.

5 Multi-Variate Gaussian Mixture Model

Imagine that you have a hen and a goose in your home. One day, you find an egg but you do not know who have laid it. You measure the weight and the height of the egg, which are 60g and 5cm respectively. Assuming that the laying frequency of the chickens is the double than the gooses one, and the following mean and covariance matrices:

$$\sigma_1 = \begin{pmatrix} 5 & \frac{1}{10} \\ \frac{1}{10} & \frac{1}{2} \end{pmatrix}$$

$$\sigma_2 = \begin{pmatrix} 8 & \frac{1}{5} \\ \frac{1}{5} & 1 \end{pmatrix}$$

$$\mu_1 = (54, 5)$$

$$\mu_2 = (65, 6)$$

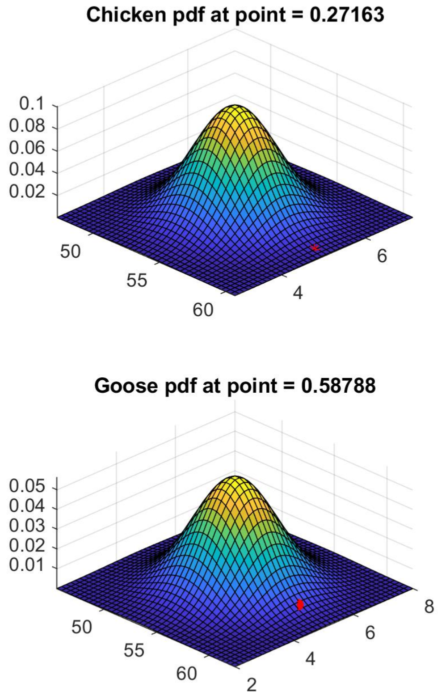
What we have to do is calculate the probability density function (pdf) of the two distributions. One pdf for the chicken eggs and another one for the goose eggs. However, this time we have two variables, weight and size, so we have to use the multivariate gaussian pdf, given by the formula:

$$pdf(x, \mu, \Sigma) = \det(2\pi\mathbf{\Sigma})^{-\frac{1}{2}}\, e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})'\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})} \tag{1}$$

So we just have to compute the pdf of the sample in both pdf distributions (chicken and goose), compare the probabilities and predict the one with a higher pdf. The probability of predicting correctly given distribution 1, as it was explained on the last exercise, is done by taking the pdf of distribution A and dividing it by the sum of the pdfs of all the distributions. That is:

$$p = \frac{pdf_1(s)}{\sum_{i=0}^{N} pdf_i(s)} \tag{2}$$

Using Matlab script `exercise_5.m` we can easily visualize the results:


Chicken pdf at point = 0.27163

Goose pdf at point = 0.58788

Figure 4: 2-Variable Gaussian Model Representation, the z-axis represents the pdf. Top view of this same representation, open file: `2d-gauss-model-top.jpg` in the `img` folder.

Then, since goose pdf is higher we predict is from a goose. To calculate the probability or confidence of the egg belonging to the goose distribution we use equation (2):

$$p_c = \frac{0.58788}{0.58788 + 0.27163} \approx 0.68 \rightarrow 68\% \text{ confidence.}$$

6 Fisher's Database

Download the data from the dataset that you will find in uci.edu[1]. Use it to classify the following unlabeled samples:

Sepal length	Petal length	Sepal width	Petal width
4.9	3.2	1.7	0.2
5	2.8	1.6	0.5
5.5	2.8	3.6	1.3
7.1	3.1	6.1	1.7

Using Matlab we just have to do the following procedure: understand the data, after that, subdivide it into different subsets (There are 3 classes), then calculate the parameters of the Gaussian distribution for each class (assuming all the three classes follow the normal distribution), so we calculate the mean μ and the covariance matrix Σ for all the variables for each distribution. After that using equation (1) we can calculate the pdf for each distribution and then we can compare each of the pdf for each distribution and predict the one with the highest pdf value. To calculate the probabilities of our prediction we do the same as in the previous exercise. It's important to recall that we are estimating the values of μ, Σ. That is why in real-world applications usually the more data we have the better our model will be. Executing the matlab script `exercise_6.m` we get the following output:

```
Reading samples...
The sample 1 is a setosa. Confidence: 100 %
The sample 2 is a setosa. Confidence: 100 %
The sample 3 is a versicolor. Confidence: 99.9972 %
The sample 4 is a virginica. Confidence: 99.9289 %
```

[1] https://archive.ics.uci.edu/ml/datasets/Iris

7 Principal Component Analysis (PCA)

Suppose that we have a set of 2D samples: [0, 0], [1, 1], [2, 3], [3, 2] and [4, 4].

- Draw the data and compute the covariance matrix.
- Apply PCA over these samples and find the basis where the data have the maximum variance. To do so, find the eigenvalues and eigenvectors of the data covariance matrix. Draw these basis in the same figure than the previous exercise.
- Project the data over the obtained basis with PCA. Discuss the relation existing between the covariance of the projected data and the eigenvalues computed in b).
- Project and re-project the data using only the basis of maximum variance. Draw the results over the original data.

All of this exercises were done at the same time. To do it I applied a very elemental PCA algorithm (not generalized) to this exercise and it plots the data, finds a new basis, plots a line along that new direction and reprojects the data:

The algorithm can be found in the `exercise_7_1.m` file and prints the following output:

```
1. Draw the data
2. Compute the covariance matrix

covarMat =

0.6250    0.5625
0.5625    0.6250

3. Apply PCA and find the basis:

eigenVector =

-0.7071    0.7071
0.7071    0.7071


eigenValue =

0.0625   0
0     1.1875

We take the vector with the highest eigenValue.
Highest Eigenvalue: 1.1875
Max. var. Eigenvector / new basis: [0.70711     0.70711]
```

```
dimReduction =

0    1.4142    3.5355    3.5355    5.6569


projected_X =

0    1.0000    2.5000    2.5000    4.0000
0    1.0000    2.5000    2.5000    4.0000
```

It's important to notice that `dimReduction` values are the projected data points in the new basis found. Also, to compute the covariance matrix we just normalize the X values in our dataset by doing $Z = \frac{X-\mu}{\text{range}}$ where range $=$ $(\max X - \min X)$, although we can apply PCA without normalizing (the first implementation worked without this step), when using more advanced techniques such as gradient descent or stochastic gradient descent, this helps the algorithms converge faster and since the values are in a more narrower spectrum, the algorithms tend to be more computationally efficient, specially if using a high level language like Python or Matlab / Octave. To obtain the eigenvector I used Matlab's eigen-decomposition algorithm, although you can use single-value decomposition as well.

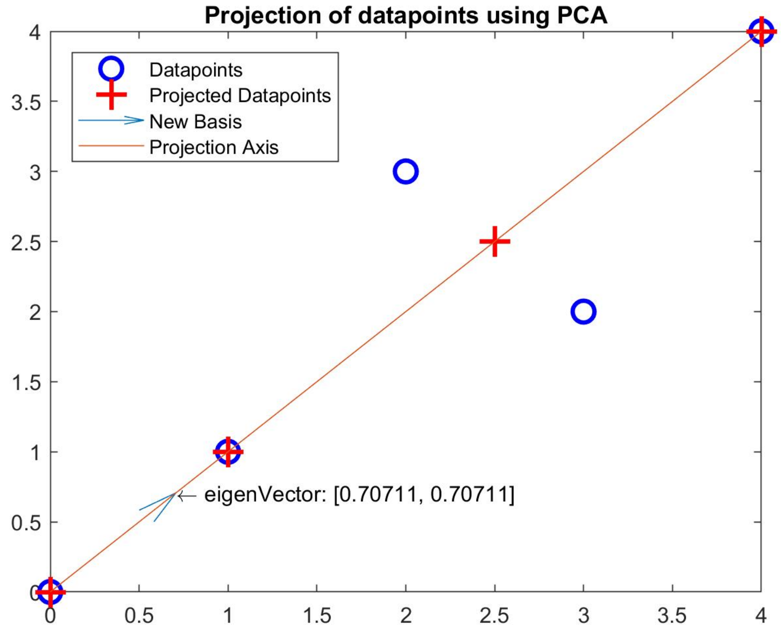


Figure 5: PCA implementation with new basis projected, re-projected datapoints and first principal component (first eigenvector).

8 PCA changing coordinates system

Repeat the previous exercise with the Cartesian samples [0, 1], [0, -1], [1, 0], [-1, 0] but using cylindrical coordinates (you first have to transform the data to cylindrical coordinates[2]).

Using the same dynamics as the previous exercise but converting from cartesian coordinates to polar coordinates I got interesting results, data-points that were far from the original datapoints without the dimensionality reduction but that had the most variable information, the angle, so by adding the mean of the radius the exact points were recovered just by using an offset, but with 1 dimension less (The code can be found at `exercise_8.m`), the output of the script plus the figure summarizes the quantitative results.

[2] I used polar coordinates since representing 2D data-points using a 3D coordinate system is redundant and another advantage of using polar coordinates is that they allow to visualize better the exercise dynamics

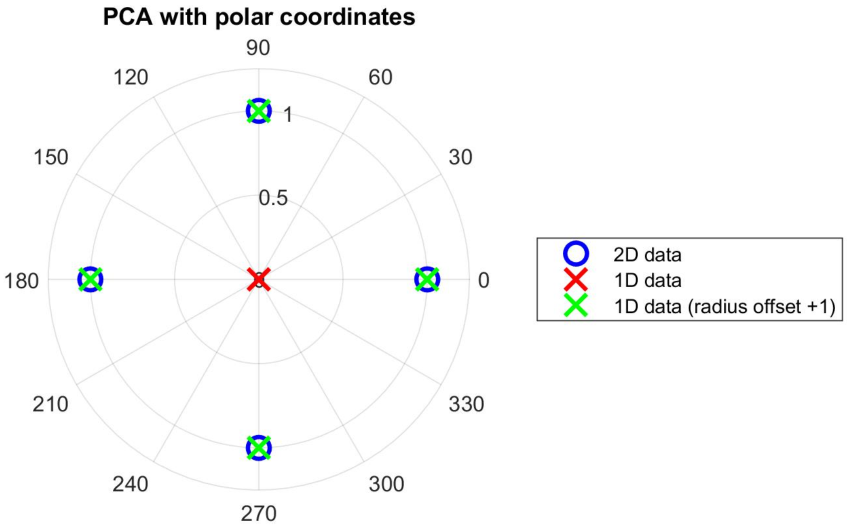


Figure 6: Using PCA with non-linear data by changing the coordinate system

Script output:

```
dataset =

0      0      1     -1
1     -1      0      0

We go from cartesian to polar...

dataset =

1.5708   -1.5708          0    3.1416
1.0000    1.0000    1.0000    1.0000


covariance_mat =

0.5556         0
0         0
```

```
eigenVector =

0     1
1     0


eigenValue =

0         0
0    0.5556

We take the vector (in our new polar space) with the highest eigenValue.
Highest Eigenvalue: 0.55556
Max. var. Eigenvector / new basis: [1  0]

dimReduction =

1.5708   -1.5708         0    3.1416


projected_data =

1.5708   -1.5708         0    3.1416
0         0         0         0
```

9 PCA + Multi Variate Gaussian Model

Apply PCA to the data from exercise 6 and classify again the unlabeled samples using the Multivariate Gaussian Mixture Model with the two more significant new attributes found with the PCA.

This exercise summarizes all the seminar. You start by extracting the dataset, then you apply PCA and keep the 2 principal components, after that you use these 2 new principal components as your new dataset and apply multivariate gaussian model to this two and using the pdf for each class you compare and predict. The accuracy shrinked (as expected since we now have less features), but it did so in less than 1% which means the two features picked were really good.

The code can be found at `exercise_9.m` and its output is:

```
Reading samples...
Applying PCA...
Using 2 out of 4 features...
97.7632% variability retained
The sample 1 is a setosa. Confidence: 100 %
The sample 2 is a setosa. Confidence: 100 %
The sample 3 is a versicolor. Confidence: 99.9823 %
The sample 4 is a virginica. Confidence: 99.9457 %
```

10 References and detailed work

To see the source code and all the very detailed explanation of each of the steps, check-out the Jupyter Notebook at `https://goo.gl/J1e7Cd`

THE END.