SYDE 572: Assignment 1

Kritik Kaushal, Fall 2023, [k3kausha@uwaterloo.ca](mailto:k3kausha@uwaterloo.ca)

# Exercise 1

1. Using the *numpy.random.multivariate\_normal* function, the 5 samples generated for each of the 2 classes presented in the question are shown below:

A computer screen shot of numbers

Description automatically generated

The sample mean and covariance matrices of the two classes from the 5 samples were calculated by hand as shown below:

A math equations on a piece of paper

Description automatically generated

A math equations on a piece of paper

Description automatically generated

1. Using the calculated sample mean and covariance matrices, the eigenvalues and eigenvectors for each class were calculated as follows:

A white paper with purple writing on it

Description automatically generated

A close-up of a math problem

Description automatically generated

A yellow paper with writing on it

Description automatically generated

1. Using the above mean and covariance matrices, the equiprobability contours for each class were plotted in Python.

Class 1 Equiprobability contours:

A graph of a random data

Description automatically generated with medium confidence

Class 2 Equiprobability contours:

A diagram of a random data

Description automatically generated with medium confidence

1. It can be observed from the equiprobability contours above that the highest values of the PDF function are found near the sample mean.

For example, the calculated sample mean for class 1 is [4.09, 3.18]. Looking at the equiprobability contours for class 1, we see that the contours are centered near the sample mean of [4.09, 3.18].

Also, the effects of the negative covariance in the class 2 calculated covariance matrix can be seen in its equiprobability; if you imagine a line, it has a negative slope.

Along a similar line, the covariance in class 1 between the two random variables is positive, which is why the equiprobability contour has a shape that tends to go towards the top right (can think of it as having “positive slope”)

1. Using a sample size of 100, the sample mean, covariance, eigenvalues, and eigenvectors were determined using Python. The statistics for each class are shown below, respectively.

A computer screen with numbers and symbols

Description automatically generated

A computer screen with numbers and symbols

Description automatically generated

The equiprobability contours for each class with 100 samples are shown below:

Class 1 Equiprobability contours:

A graph of a random data

Description automatically generated

Class 2 Equiprobability contours:

A chart of a random data

Description automatically generated with medium confidence

There is a difference between the contours generated for 100 samples and 5 samples. The contours for the 100 samples case are more representative of the original class definition.

The contours are more centered around the defined class means, and the contours are more representative of the covariance matrices in the original class definition.

This makes sense. A larger sample size will give results more in line with the original class definitions, which is what we observe in this case.

# Exercise 2

1. Considering the same 5 sample datasets produced in Exercise 1, the MED decision boundary was found by hand as the following:

A math equations on a piece of paper

Description automatically generated

Then, this boundary along with the 5 sample data points was plotted. The result is shown below.

A graph with a line and a line

Description automatically generated

As shown, the MED decision boundary is a great classifier for this case because the data is linearly separable. There are no outliers, and the classifier clearly separates both sets of datapoints.

1. Using the Sympy library for symbolic variables, the decision boundary for the MED classifier was found to be:

A black background with white text

Description automatically generated

In a more readable format:

The 100 datapoints for both classes along with the MED classifier decision boundary were plotted as shown below.

A graph with blue and orange dots

Description automatically generated

1. 50 new samples were generated for each class. Additionally, white noise was added to this new set of 50 samples. The data, along with both decision boundaries (n = 5 and n = 100) are shown below.

A graph with numbers and colored dots

Description automatically generated

Based off the above, the classification accuracy of each classifier is calculated below.

For the n = 5 decision boundary,

For the n = 100 decision boundary,

1. Overall, the classifier trained on 100 samples is better when there was no noise added to the data. However, when noise was added to the data, the data was no longer linearly separable, so the two classifiers performed more or less the same.

The n = 5 decision boundary seemed to produce better results, but the sample size was low, so this is expected. In general, classification accuracy increases with increasing sample size, and this can be observed in this case. Without noise, the n = 100 decision boundary is more accurate. With noise, the data is no longer linearly separable so both classifiers have similar performance.

# Exercise 3

1. Using the Euclidean distance metric, the kNN classifier decision boundaries for varying values of k from 1 to 5 are shown in the plots below. The original training data is superimposed on the plots of the boundaries.

A graph with blue and orange dots

Description automatically generated

A diagram of a map with orange and blue dots

Description automatically generated

A map with orange and blue dots

Description automatically generated

A diagram of a map with blue and orange dots

Description automatically generated

A map with orange and blue dots

Description automatically generated

1. Using the noisy data generated from Exercise 2, the classification accuracy of each ‘k’ kNN classifier was determined. The data and plot are below. The Excel file I used to generate and plot the accuracy is attached in the submission.

A white sheet with black numbers and a black line

Description automatically generated

1. From the graph above, the ‘k’ value of 1 seems to be producing the best results. This is because the noisy data tends to follow the training data, thus the closest neighbour’s value produces a more accurate result. The noise applied to the training data doesn’t seem to be much, hence why the noisy data still generally follows the training data. As the number of ‘k’ neighbours increases, the overall accuracy of the classifier decreases but tends to hover around 50%.
2. The kNN classifier produces similar results to the MED classifier from the previous exercise. The kNN classifier may be marginally better, but it could be made much more accurate with a different distance metric than Euclidean distance. The performance of the MED and kNN classifier could be so similar due to the structure of the data; the training data was not very separated/separated to begin with, so boundaries may be much more ambiguous.

All code and supplementary files are included in the submission. I commented out all plots except the kNN plot. To see the other plots, please uncomment them and run the file again. The kNN classifier code takes a little while to run. I have set it to k=3 in the submitted code, with the noisy data plotted.