SYDE: Introduction to Pattern Recognition

Assignment 2

Due: 11:59 PM (EST), Oct 17, 2023, submit on LEARN.

Include your name and student number!

Submit your write-up in pdf and all source code in a zip file (with proper documentation). Write a script for each programming exercise so that the TAs can easily run and verify your results. Make sure your code runs!
[Text in square brackets are hints that can be ignored.]

For all the exercises, you are only allowed to use the basic Python and numpy libraries, unless specified otherwise. You can use any plotting library, such as Matplotlib.

Exercise 1: MMD Classifier (35 pts)

Let's consider the following two classes:

- Class 1: $N_1 = 5$, $\mu_1 = [1, 3]^T$, $\Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 15 \end{bmatrix}$
- Class 2: $N_2 = 5$, $\mu_1 = [20, 31]^T$, $\Sigma_1 = \begin{bmatrix} 3 & 4 \\ 4 & 11 \end{bmatrix}$

1. (1 pt) Use the *numpy.multivariate_normal* function to generate 5 samples each for the above two classes. For each of the 10 samples above, use *numpy.multivariate_normal* to generate a sample with $\mu = [2, 2]^T$, and $\Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}$ and add it to the class samples. For example, let's say [4, 4] is a sample generated for class 1. Then you generate a sample using the noise distribution, let's say it is [2, 3]. Add this noisy data to the original sample, resulting in [6, 7] as the sample for your class 1. Find the sample mean and covariance matrices for the two classes by hand.

2. (10 pts) Use the sample means and covariance matrices to find the eigenvalues and eigenvectors of the covariance matrices by hand. Use the eigenvalues and eigenvectors to get the inverse of the covariance matrices.

3. (2 pts) Using the above calculations, find the decision boundary for the MMD classifier by hand. Plot the boundary and the data.

4. (10 pts) Follow the same procedure as in step 1, to generate 100 samples each for the two classes. Find the decision boundary for the MMD classifier in Python. Plot the boundary and the data.

5. (5 pts) Generate 50 new samples per class for the two classes using step 1. Now, add white noise to each of the 100 samples using *numpy.multivariate_normal* with $\mu = [0, 0]^T$, and $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Using the MMD classifiers learned in step 2, predict the labels for the 100 noisy samples generated for the two classes (50 samples per class). Report the classification accuracy of the classifiers using:

$$error = \frac{Number\ of\ correct\ predictions}{total\ number\ of\ data\ points\ in\ the\ test\ set} \tag{1}$$

6. (2 pts) Which of the two classifiers is better, the one learned from 5 samples or the one with 100 samples? Why?

7. (5 pts) Now, compare the MMD classifier in step 2 with the MED classifier that you developed in the previous assignment. You will need to run the MED on the data generated in this assignment for a fair comparison. Is MMD better than MED? Explain.

Exercise 2: ML and MAP Classifiers (20 pts)

In this exercise, you will use the same data as in exercise 1 for the ML and MAP classifiers.

1. (5 pts) Using the sample mean and covariance calculated for the 100 sample data of the two classes (step 4 of exercise 1), you can estimate the probability distributions of the two classes. Based on this, develop an ML classifier and test it on the data generated in step 5 of exercise 1. Report the classification accuracy.
2. (5 pts) Now, let's assume that the prior probabilities for the two classes are $p(C_1) = 0.58$ and $p(C_2) = 0.42$. Using these prior probabilities, and the means and covariances of the two classes, develop an MAP classifier and test it on the data generated in step 5 of exercise 1. Report the classification accuracy.
3. (5 pts) Based on the results, do you think that assuming the probability distributions of the two classes as Gaussian was correct? What if you remove the extra step of adding noise in your training data (step 1 exercise 1)? Do you think that would make these classifiers better? Explain.
4. (5 pts) Compare the ML, MAP, MMD, and MED classifiers based on the classification accuracy. Which classifier is the best? Could the inferior classifiers be better for different data? Explain.

Exercise 3: Parametric Estimation (45 pts)

In this exercise, you will be estimating the parameters of different distributions using the maximum likelihood principle. You can assume 1-d data, which means x is not a vector but just a real number.

1. (10 pts) Let's assume that the probability distribution of your dataset is an exponential distribution (Wiki). In this case, you only need to estimate a single parameter λ. Use MLE to derive an expression for finding λ.
2. (10 pts) Now, let's assume the probability distribution of your dataset is uniform (Zhang Ch. 19.8). In this case, you will estimate two parameters a and b. Use MLE to derive an expression for finding the two parameters.
3. (15 pts) We derived the expressions for the mean and standard deviation of the Gaussian distribution in class. Now, let's assume the probability distribution of your dataset is not just a single Gaussian distribution but a product of a Gaussian distribution and an exponential distribution:

$$p(X|\theta) = \frac{1}{2}[\mathcal{N}(X|\theta_1, \theta_2) \times f(x; \lambda)] \tag{2}$$

   In this case, you will be estimating three parameters (θ_1, θ_2, and λ), where θ_1 and θ_2 are the mean and variance of the Gaussian distribution. Using MLE, derive an expression for finding the three parameters of your probability distribution defined by eq. (2).
4. (10 pts) Generate 100 samples for two classes using the following statistics:
   - Class 1: $N_1 = 5$, $\mu_1 = 0.5$, $\sigma_1 = 1$
   - Class 2: $N_2 = 5$, $\mu_1 = 5$, $\sigma_1 = 3$

   Using the formulae derived for the above three distributions, develop three ML classifiers using the generated data. Now, generate 50 samples for each class, and add white noise to each of the samples by using *numpy.random.normal* with 0 mean and 1 standard deviation. Test the three ML classifiers on the noisy test data. Which of the three ML classifiers is the best? Why?