|  |  |  |
| --- | --- | --- |
| Cairo University  Faculty of Engineering  Computer Engineering Department | | ­­­­CMPN450  Fall 2018 |
|  | **Pattern Recognition and Neural Networks.**  Lab 4 – Density Estimation |  |

**Recap:**

In the previous lab, we were introduced to the **problem of classification of multivariate data**, and we applied **Bayes classifier** to classify these points. We built our classifier based on the assumption that this data is multivariate Gaussian, and we estimated the **mean** and **covariance** of this distribution.

Finally, a test vector **x** is classified as (i.e. assigned) the class that maximizes **the posterior probability**

**Density Estimation:**

In this lab, we will drop off the assumption that the data is drawn out of a Gaussian distribution with mean and covariance . The distribution is not necessarily Gaussian, yet we will still apply Bayes classifier to classify between data points by assigning each point to the class that maximizes its posterior probability as usual.

**(Q1) What will change in Bayes classifier under the new assumption?**

*We will answer that for you! is no longer equal to the Gaussian distribution as shown in the previous equation i.e. we can’t use the Gaussian formula to substitute for .*

**(Q2) How will we figure out the unknown distribution (or likelihood function)?**

*We will answer that for you too! There are three ways for doing so:*

1. *Histogram Analysis.*
2. *Naïve Estimator.*
3. *Parzen Window Density Estimator.*

*All these techniques aim to estimate a density function for the training points, and hence you can use the density function to apply Bayes classifier and classify your test points.*

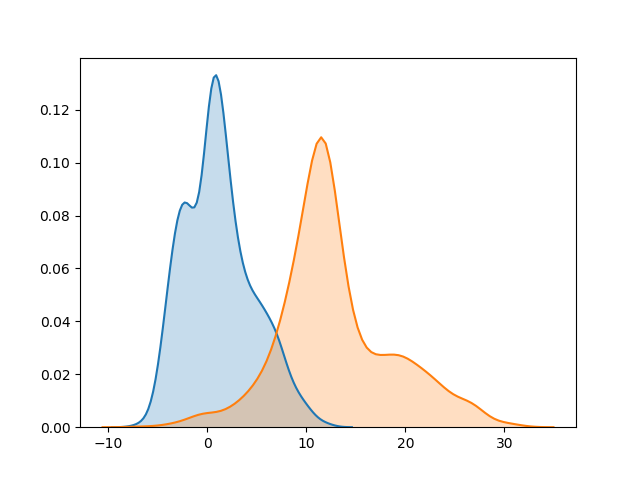
*Refer to the slides to get a brief about these techniques.*

**Parzen Window Density Estimator:**

Given files “**data.csv**”, “**test\_data.csv**”, “**test\_data\_true.csv**”. The first file contains a list of one-dimensional data points and their corresponding classes. The data points were drawn of the distribution shown in Figure 1 (*It will be your task to estimate this distribution*).

As shown in Figure 1, the points belonging to class 1 were drawn out of the distribution in blue, whereas the points belonging to class 2 were drawn out of the distribution in red.

Figure 1 Density Distribution for Class 1 and Class 2

After applying Parzen Window Density Estimator and plotting the estimated distribution for both classes, you should observe a similar distribution for each class to the corresponding distribution shown in Figure 1.

The second file contains test points without the associated classes. The true values for test points are found in “**test\_data\_true.csv**”. The format of data files is shown in the following figure.

|  |  |  |
| --- | --- | --- |
|  | Class | Value |
| Point #1 | 1 | 5.34 |
| Point #2 | 2 | 2.73 |
| Point #3 | 2 | -1.832 |

**Requirement:**

|  |  |
| --- | --- |
| 1. | Read and import the training data, test data and the true values for the test data, found in the given files respectively. |
| 2. | Choose a suitable bump function (Parzen window), and find the optimal value of h for the bump function for each class.  Hint: First, fill in the function **calculate\_h\_optimal()**. Pass the convenient parameters.  Second, implement the bump function (i.e. Parzen window). You should compute the bump function for any **x,** given a point **p** and an optimal value for **h.  You should vectorise this (i.e. your function should work if x were a single data point, or a vector of two data points, or a vector of M data points).**  **Y = F(x), where x represents the domain of the function or equivalently some values to compute the bump function at, y represents the output of the bump function and F represents the bump function.** |
| 3. | At this point, you need to iterate over the training points and apply the bump function for every training point. The density estimate will be the accumulation (i.e. the sum) of the output of these bump functions.  Hint 1: You should compute a density estimate for each class. Remember, the density estimate represents  Hint 2: To generate the x vector which represents the possible values for the one dimensional feature we have, you can find the minimum and maximum possible values for x among the training data, and generate a range of points which will represent your one-dimensional axis.  x = np.linspace(min\_x, max\_x, number\_of\_axis\_points)  Here, **min\_x** represents the minimum value for the feature x, **max\_x** represents the maximum value for the feature x and **number\_of\_axis\_points** represents the resolution of the scale (or how many points should be included between these two limits). |
| 4. | Plot the density estimates for the two classes in **ONE PLOT with DIFFERENT COLORS.** This is an ordinary plot. No helping code this time.  FAQ: What do you mean by two classes in one plot with different colors? *Answer: The two classes are overlaid together in one plot (such as Figure 1) with different colors (red and blue). We can’t be any more specific.*  **Check your plot result.**  Is it similar to Figure 1? If yes, proceed to question 5. If no, unfortunately, you have done something wrong. Debug before asking for help, as we will tell you so when ask us so! |
| 5. | It’s time to classify the test points. You know what to do already! Given the points in the test file, it’s required to classify each point whether it belongs to class 1 or 2 by applying Bayes classifier. Report the accuracy of your classifier by comparing with the true values found in the third file.  *Hint: It’s not as easy as it looks like, there’s a trick here. Be creative in your solution. Solutions may differ in this part, and this may lead to varying accuracies between students. May the odds be in your favor and you get the highest accuracy!* |