SYDE 572: Intro to Pattern Recognition
Assigment 3

Cooper Ang (20768006)

November 3, 2022

1 Exercise 1

1.1 Question 1

(25 pts) Use histogram based estimation with region sizes of {1,10,100} to estimate the probability distribution of your dataset. Note that you will need to estimate the distribution of each class separately. Plot your probability distributions for all the region sizes. Which region size seems to be the best for estimating the probabilities? Explain.

Applying histogram based estimation groups labeled samples into discrete regions to estimate the approximate probabilities. I had to calculate two histograms for the two corresponding classes yielding plots for $\hat{p}(x|0)$ and $\hat{p}(x|1)$. Region sizes (bin width in the title) of 1, 10, and 100 were used.

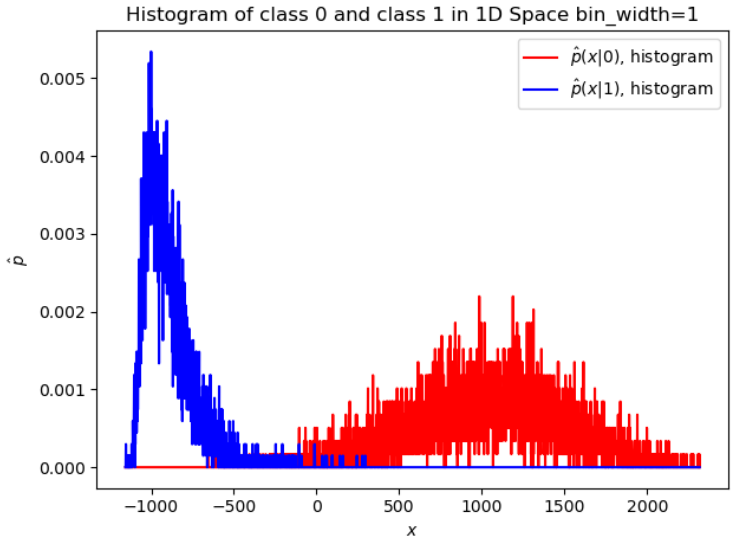

Figure 1: Histogram Based Estimation of MNIST 0 and 1 classes (region size 1)

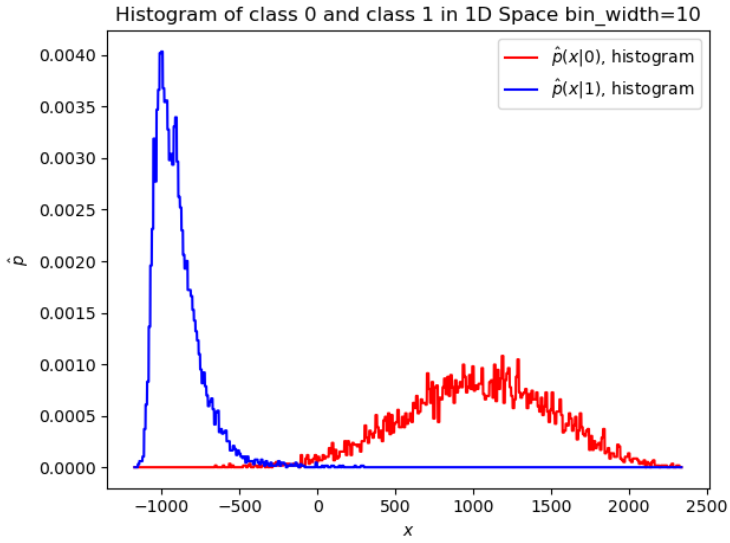

Figure 2: Histogram Based Estimation of MNIST 0 and 1 classes (region size 10)

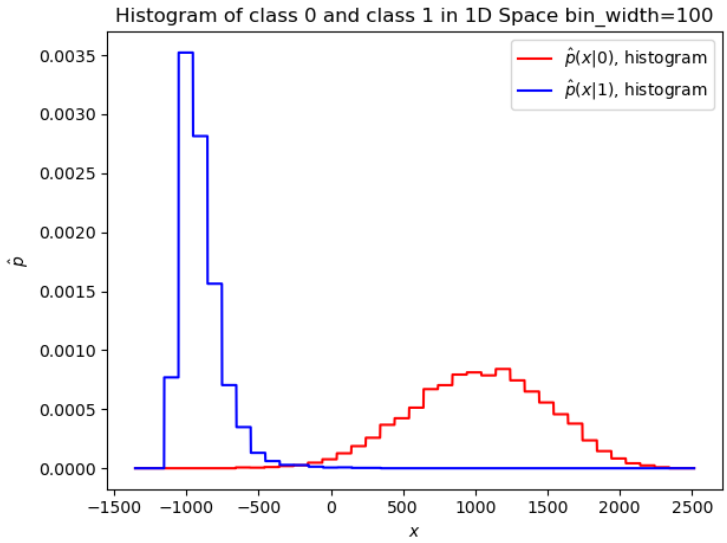

Figure 3: Histogram Based Estimation of MNIST 0 and 1 classes (region size 100)

By looking at the histogram plots of the different region sizes, you can see that there is a trade off between resolution along the x axis and along the $\hat{p}(x)$ axis. In the case of region size of 1, you can clearly see that there is high noise in the histogram, which may cause incorrect classifications for outliers. There will also be out-sized impact on histogram bins that contain no data points, which will result in zero probabilities. Assuming that the actual distribution of the class is smooth, this would be not a great classifier. For the region size of 10, you can see that there is a reduction in the resolution in the x axis, but the PDF smooths out in comparison to the region size of 1. With a region size of 100, you can see an even bigger drop in the resolution of x axis, but in return there is a better resolution in the PDF. For the region size of 100, $\hat{p}(x|1)$ the discrete steps in the x axis result in large changes in probabilities at the borders of the steps, which is unwanted. Due to the nature of the trade off of resolutions, I believe that the region size of 10 is the best estimate of the probability as it strikes the best balance between x axis and PDF resolutions with lower noise than region size 1 and less brunt discrete steps in the x axis. Although I think a region size between 10 and 100 would even be more optimal.

1.2 Question 2

(5 pts) Use the estimated distributions with an ML classifier to predict labels of your test data. Report the test error for using the distributions estimated with all the region sizes. Which region size seems to be the best in terms of the test error? Explain.

Error for Histogram Based Estimation (region size 1): 0.0113475
Error for Histogram Based Estimation (region size 10): 0.0033096
Error for Histogram Based Estimation (region size 100): 0.0052009

In terms of test error the histogram based estimation with the best performance is the region size of 10. As explained in the Q1, I believe that the region size of 10 best captured the true probability distribution of the classes. The region size of 1 has too much noise and thus some outliers skew the estimation to misclassify more of the test set thus leading to a higher error. The region size of 100 is close in performance to the region size of 10, but some the larger discrete steps in the x axis probably lead to some more unwanted misclassifications. It is important to point out that within my code, I assumed that for equal class probabilities (or both probabilities equaling zero), I classify the result as class 0. Changing this to classify the tiebreakers as class 1 results in a tie between region sizes of 10 and 100 with errors of 0.0052009, and region size of 1 with error of 0.003309. Thus, you can see that region sizes of 10 and 100 are close in overall test set performance.

1.3 Question 3

(20 pts) Repeat the above two steps using the kernel-based density estimation. You can use a Gaussian kernel with $\sigma = 20$

Given N samples, the KDE estimation is given by Equation 1. Where ϕ is the kernel function of choice.

$$\hat{p}(x) = \frac{1}{N} \sum_i \frac{1}{h} \phi \left(\frac{x - x_i}{h} \right) \tag{1}$$

In this case I used a Gaussian kernel so:

$$\phi(x) = \frac{1}{\sqrt{2\pi}} \exp \left(-\frac{1}{2} x^2 \right) \tag{2}$$

Using the the provided σ value of 20 as the scale factor, I fit the KDE classifier, which creates a Gaussian kernel for each sample which by applying superposition for each kernel and and normalizing to a valid PDF forms the non-parametric estimation used to make predictions. The KDE (parzen window) estimations of the PDF are seen in Figure 4.

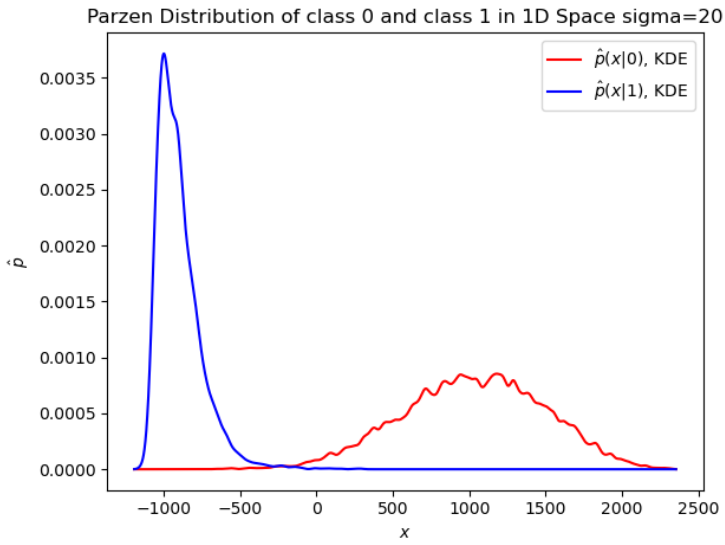

Figure 4: Kernel-based Density Estimation of MNIST 0 and 1 classes ($\sigma = 20$)

Error for KDE estimation classifier ($\sigma = 20$): 0.003309692

1.4 Question 4

(5 pts) Which of the two approaches (histogram or kernel) for probability estimations best represent your data and produce the lowest error? Explain.

The region size of 10 for histogram estimation and kernel estimation with ($sigma = 20$) have the same lowest error of 0.003309692. In terms of best representing data, the histogram based approach is purely discrete, leading to a non-smooth curve of a PDF. In other words there is still either noise or large discrete steps in the PDF, which may not match up with the true class distribution. For the KDE based estimation you can see that creates a much smoother curve. Since KDE is smoother than the histogram based approach and overall the assumption based on the parametric estimation from the previous labs was that the true class distribution is Gaussian, the KDE method seems to best represent the data in this case.

1.5 Question 5

(5 pts) Compare your results in this exercise with the results for parametric estimations of your data in exercise 2 of assignment 2. Do you think non-parametric approaches are better than the parametric approaches for density estimation? Explain.

As a reminder from the last lab (1D MNIST data), the parametric estimations yielded:

Error for MLE Exponential Classifier: 0.0264775
Error for MLE Uniform Classifier: 0.02316784
Error for MLE Gaussian Classifier: 0.003782

The best non-parametric approach (KDE or Histogram region size = 10) results in 0.003309 error. Purely by a numbers point of view, it seems like the non-parametric approaches are better. Analyzing the parametric approaches it is assumed that the distribution is known, and thus if the type of distribution is incorrect, you see much worse performance. You can clearly see this by how the error of the MLE classifier using the exponential and uniform distributions performed poorly compared to the Gaussian MLE classifier. Comparing that to the non-parametric approach to density estimation it seems like that the performance of the classifier is not reliant on knowing what type of distribution the true class is, and thus can be potentially better if you did not have an idea of which class distribution

the true class is. As a generalization I think the non-parametric approaches are better if you don't know the class distributions and the parametric approach are better if you know the true class distribution. Knowing nothing about the data, I'd suggest attempting the non-parametric approaches first, as they have less assumptions and as a result are more flexible to work with different distributions of data.

2 Exercise 2

2.1 Question 1

(10 pts) Implement the k-means clustering algorithm with euclidean distance as the distance metric.

To implement the k-means clustering algorithm I first randomly initialized the centroids. Then I applied the euclidean distance to do the cluster assignment step to cluster the samples to the nearest cluster. Afterwards I moved the centroid to the mean of the samples clustered in the previous step. This process was iteratively repeated until the condition where the centroids did not move was satisfied.

2.2 Question 2

(10 pts) Apply your k-means implementation to the MNIST dataset. Use k={5,10,20,40}. Also, you will not be using class labels during the clustering process as it is an unsupervised learning problem.

To ensure quick runtime only the first 100 samples of each class were used (1000 samples total). Please see code for the implementation and result of this implementation.

2.3 Question 3

(7 pts) As we do not use class labels for clustering we cannot use test error to evaluate the k-means algorithm. Instead, we can test our clustering implementation using cluster consistency: all data points in a cluster should belong to the same class. For example, if your cluster number 1 has all the data points belonging to class "5" then this cluster has perfect consistency. You can find the cluster consistency Q_i for each cluster i as follows:

$$Q_i = \frac{m_i}{N_i} \tag{3}$$

where, N_i is the total number of data points in cluster i, and m_i represents the total number of data points belonging to the most common class in cluster i. You can find m_i by first counting the total number of data points belonging to each of the 10 classes in cluster i, and then taking the max of this list. To find the overall clustering consistency:

$$Q = \frac{1}{k}\sum_{i=1}^{k} Q_i \tag{4}$$

Report the cluster consistency of your k-means clustering on the MNIST dataset for all four values of k. The cluster consistency will vary quite a lot as the k-means algorithm relies on the random initialization of the starting centroids. These results come from a decreased subset of the data (100 from each class).

K-means clustering consistency (K=5): 0.3728134
K-means clustering consistency (K=10): 0.6051866
K-means clustering consistency (K=20): 0.7346128
K-means clustering consistency (K=40): 0.7452547

2.4 Question 4

(8 pts) Which k value produces the best results? Explain. Can the results from cluster consistency be misleading? Explain. [HINT Intuitively, what k value should produce the best results on the MNIST

dataset?]

The results show that the best results come from the highest k value of 40. Intuitively the best results should come from K=10, because this corresponds to the number of classes in the dataset. The higher clustering consistency coming from k=40 can be explained because a higher k-value allows more clusters to form, and under the assumption that samples in the same classes are close in euclidean distance, will result in better smaller clusters (higher clustering consistency. In the extreme example where I had K=1000 for 1000 samples with each cluster having one unique sample inside, would result in a perfect clustering consistency, although this would not be useful as a clustering tool.