SYDE 572 – Assignment 3

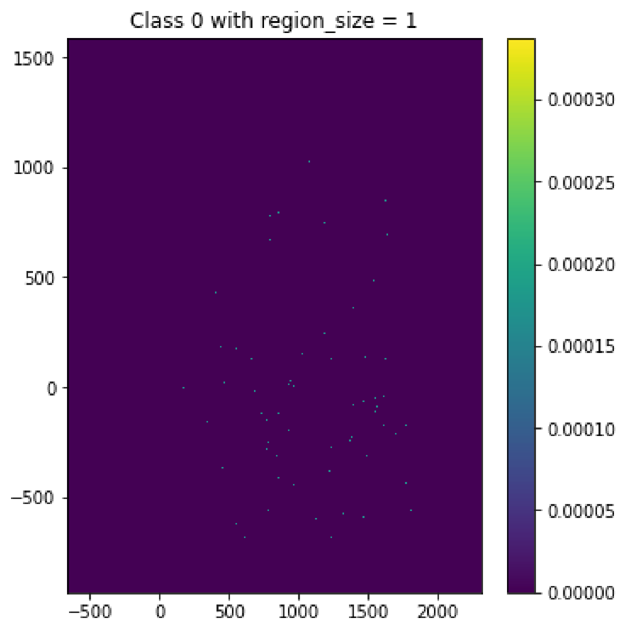
Calvin Tran – 20826392

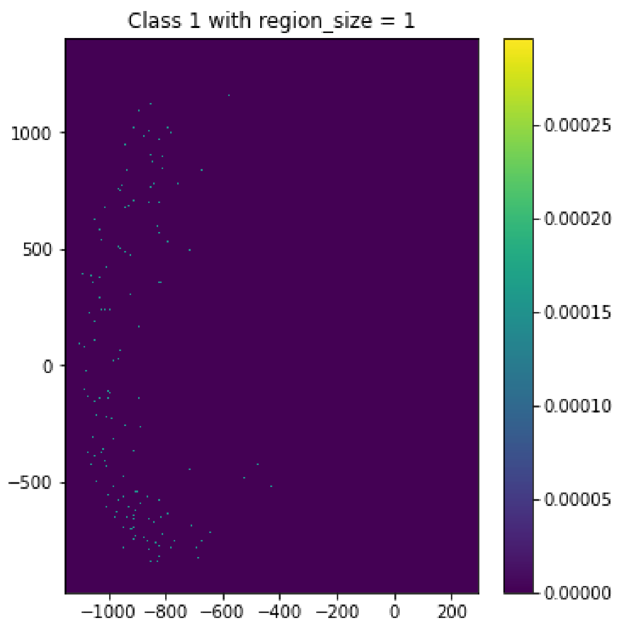
All code for each exercise can be found in Assignment_3.ipynb.

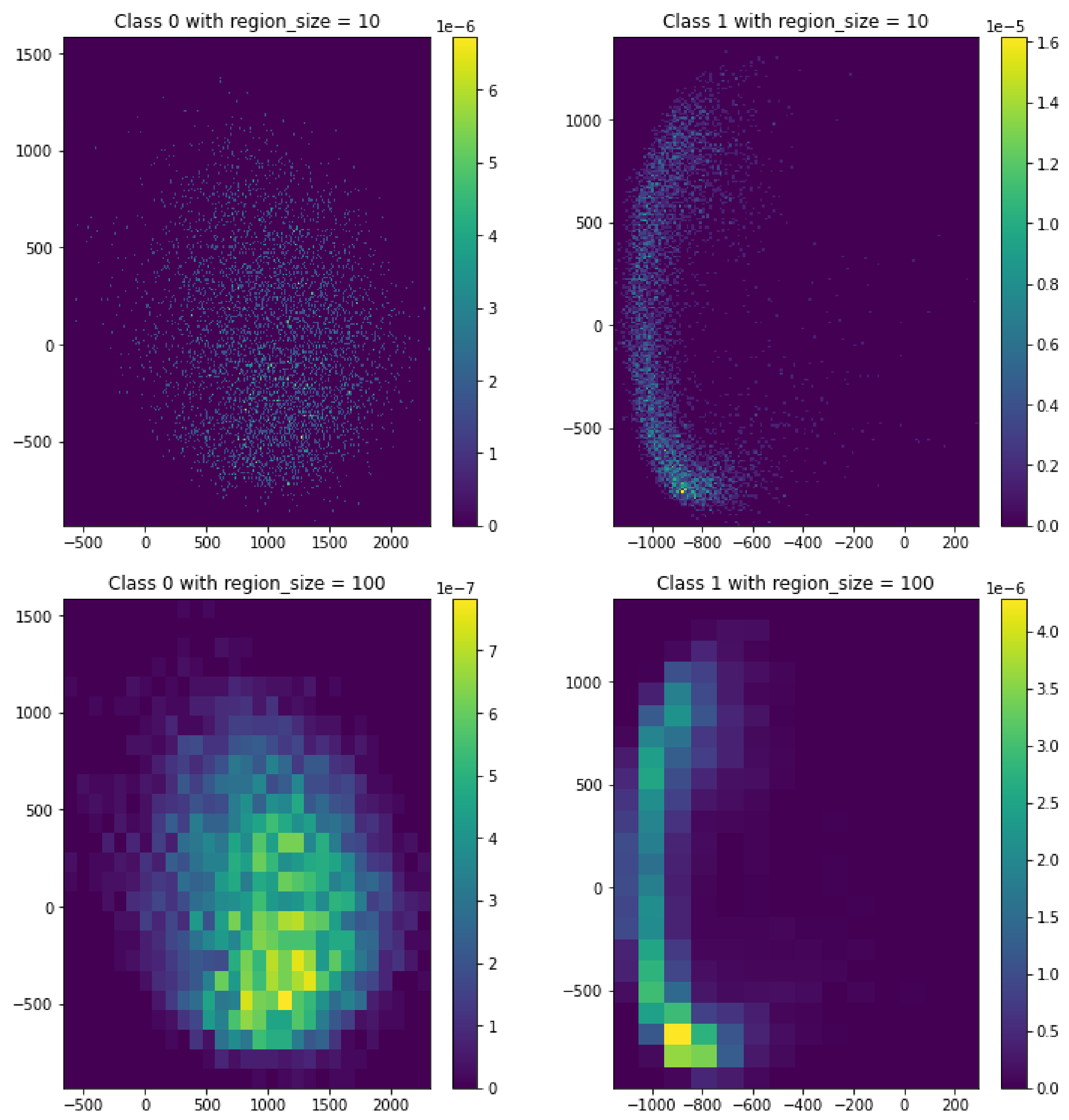
Exercise 1

1.

The plots for each class and region size can be seen below.







A region size of 100 seems to be the best at capturing patterns in the data. Due to its high granularity, both region sizes of 1 and 10 have histograms that are too precise, making them extremely sensitive to noise and outliers. By having a larger region size and thus a larger sample size per bucket, the histogram becomes less sensitive to noise and outliers. Thus, the region size of 100 is the best at highlighting patterns in the data.

2.

The following test error percentages were reported:

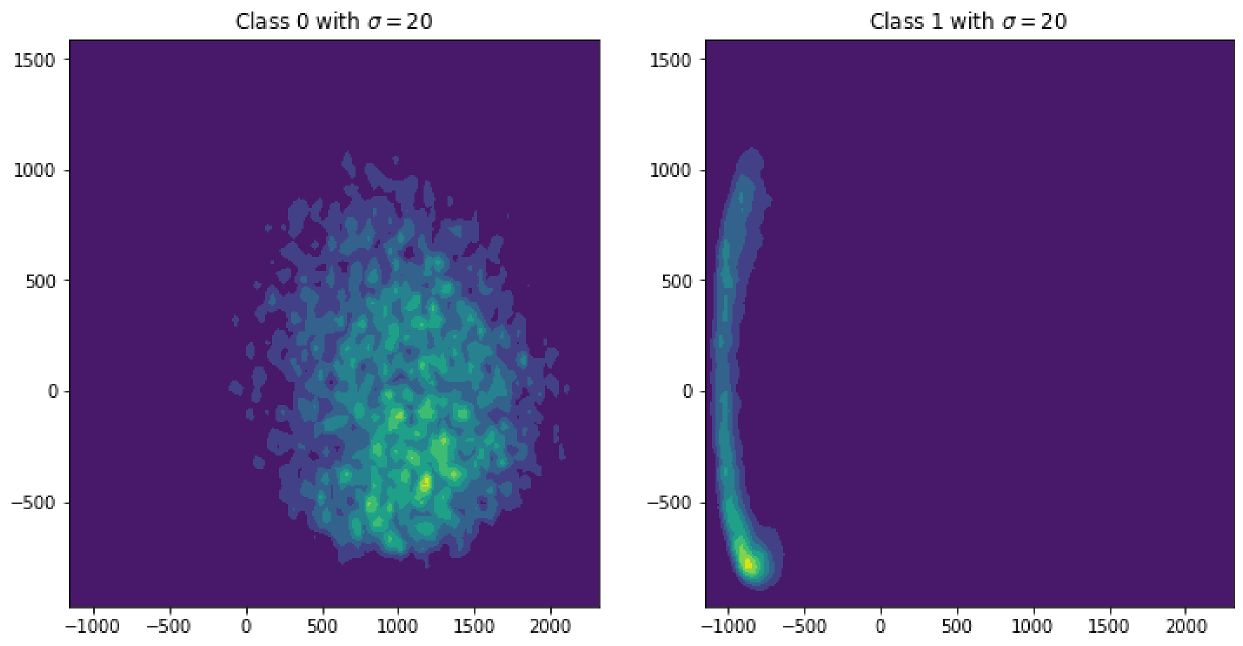
```
The percent error for a region size of 1 is 53.24%.
The percent error for a region size of 10 is 25.25%.

The percent error for a region size of 100 is 0.38%.
```

The percent error for a region size of 100 is significantly lower than the other two region sizes, making it the best option.

3.

The graph below shows the graph using a kernel-based density estimation.



The percent error for this classifier was 0.28%, which was lower than all of the other histogram-based estimations.

4.

The kernel-based estimation seems to best represent the data due to it having the lowest error.

While the histogram-based estimation with a region size of 100 also had a very low error rate, it also assumes that $P(x)$ is constant throughout the region. With a larger region size, this becomes less true. Meanwhile, the kernel density estimation is able to create smoother visualizations, with points affecting an area around it, rather than just a single point. Although it may be more computationally intensive, the kernel-based estimation with a bandwidth of 20 performs better than the histogram-based estimation with a region size of 100.

5.

Parametric methods rely on the assumption that data adheres to a particular distribution, such as Gaussian or exponential. This assumption only works well if the data actually adheres to the specified distribution. On the other hand, non-parametric methods, such as the ones above, refrain from making the same distribution constraints, allowing them to accommodate data exhibiting various patterns.

When the data's distribution is well-known and matches the assumptions, parametric methods can be more appropriate. However, when the data's distribution is uncertain or doesn't conform to parametric assumptions, non-parametric methods offer an attractive alternative. In this case, since we were given no information on the data's distribution, the non-parametric approach for density estimation may be the better option. Nonetheless, it may be beneficial to perform experiments using both methods and comparing the results.

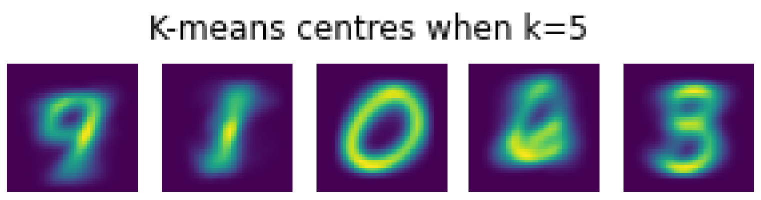
Exercise 2

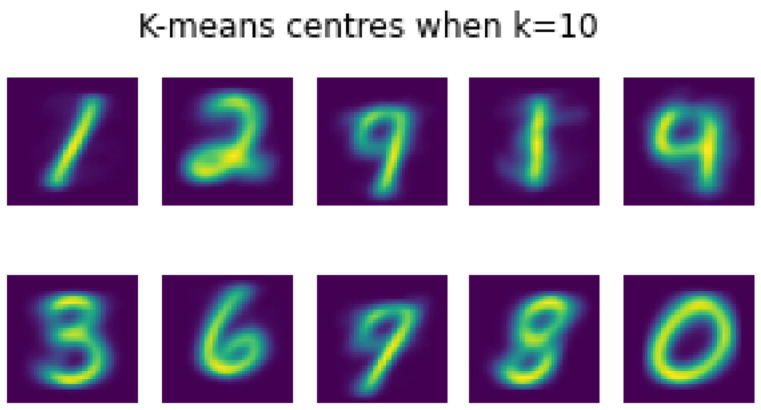
1.

Refer to the code in Assignment_3.ipynb.

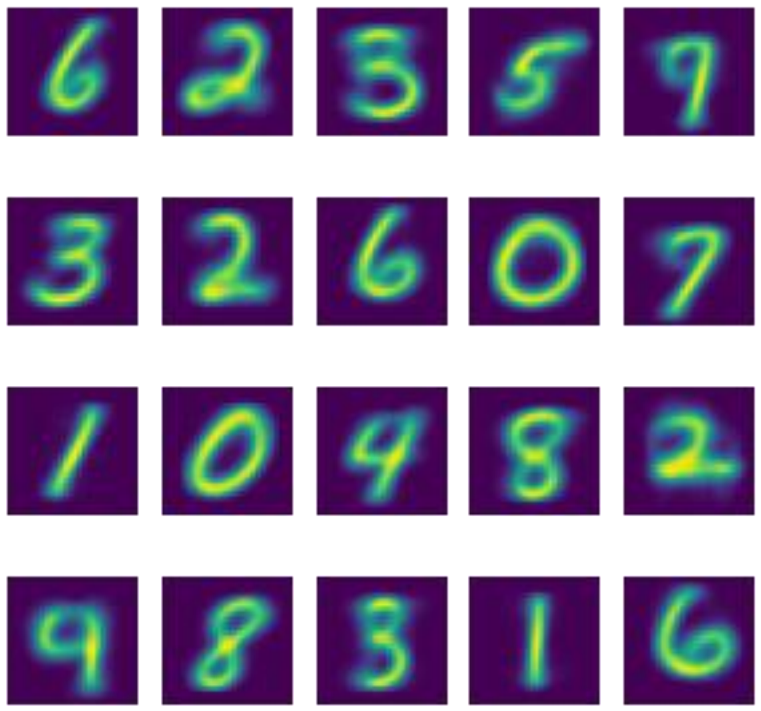
2.

The k-means implementation was applied to the MNIST dataset and the following clusters were returned for each k- value.

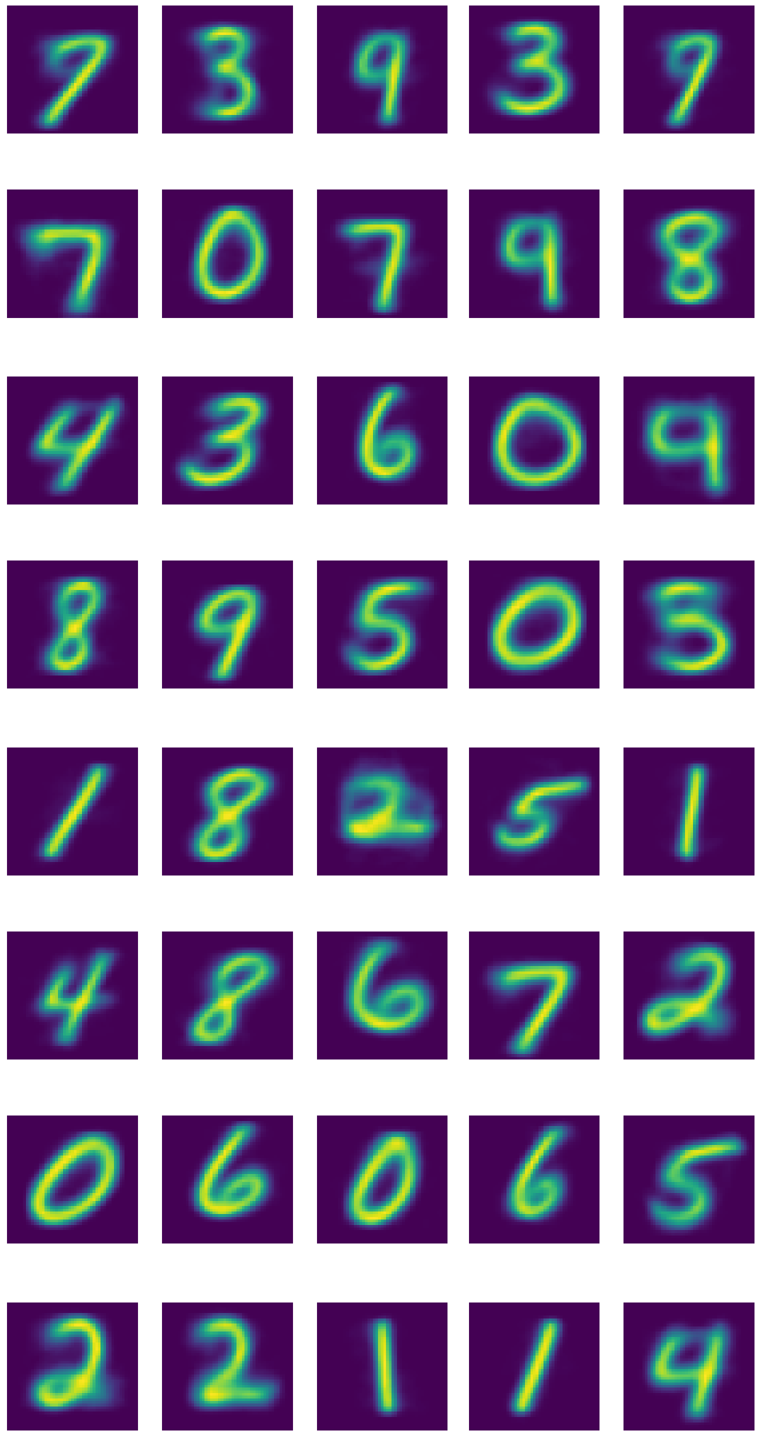




K-means centres when k=20



K-means centres when k=40



3.

The output from the code found in Assignment_3.ipynb can be found below, reporting the cluster consistencies as percentages:

The cluster consistency with a k-value of 5 is 58.16%

The cluster consistency with a k-value of 10 is 69.67%

The cluster consistency with a k-value of 20 is 75.98%

The cluster consistency with a k-value of 40 is 83.38%

4.

Looking at the results from the previous question, it can be clearly seen that the cluster consistency increases as the k-value increases. However, this should not be how the 'best results' are chosen. With more k-means, there is a higher likelihood that numbers in the same class will be grouped together as there will be a mean with a smaller Euclidean distance that is closer to the point being classified. This then increases the cluster consistency. This, however, does not accurately reflect the number of classes that are actually present. In the dataset, it is known that there are 10 digits being represented. A useful version of the classifier would need a k-value of 10, representing the 10 possible classes, with each class representing one of the 10 digits.

Having a k-value above 10 does have its benefits. With more k-means, some of the less noticeable details of numbers in the same class become separated into different classes. For example, looking at the output above for k=40, we see that there is a class with 7 without a horizontal strike through the centre, and another class with 7 with a horizontal strike through the centre. This may be useful in some scenarios, but for the most part, it is likely that a k-value of 10 is the most ideal. Furthermore, increasing k too high may lead to overfitting, which causes the model to fit to data points that may be noise, instead of generalizing well to other inputs.

Exercise 3

1.

The EM algorithm is comprised of two main steps: the expectation step and the maximization step.

We first perform the expectation step, which is the same for both the Gaussian mixture model and the diagonal Gaussian mixture model.

The responsibilities are first calculated using Bayes' Theorem:

$$r_k^i = \frac{\pi_k N(x_i|\hat{\mu}_k, I)}{\sum_{j=1}^{K} \hat{\pi}_j N\left(x_i\middle|\hat{\mu}_j, I\right)}$$

$$= \frac{\pi_k |S_k|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x_i - \mu_k)^T S_k^{-1}(x_i - \mu_k)\right)}{\sum_{j=1}^{K} \pi_j \, |S_k|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\left(x_i - \mu_j\right)^T S_j^{-1}\left(x_i - \mu_j\right)\right)}$$

We normalize this using:

$$r_k^i = \frac{r_{ik}}{r_i}$$

For the maximization step, we update the mixing coefficients π_k for each component.

$$\pi_k = \frac{1}{N}\sum_{i=1}^{N} r_k^i$$

We also update the means μ_k for each component.

$$\mu_k = \frac{\sum_{i=1}^{N} r_k^i x^i}{\sum_{i=1}^{N} r_k^i}$$

Finally, we update the diagonal covariances Σ_k for each component.

$$\begin{aligned} S_j &= \frac{\sum_{i=1}^{N} r_k^i \left(x_{ij} - \mu_{kj}\right)^2}{\sum_{i=1}^{N} r_k^i} \\ &= \frac{\sum_{i=1}^{N} r_k^i \left(x_{ij}\right)^2}{\sum_{i=1}^{N} r_k^i} - {\mu_{kj}}^2 \end{aligned}$$

The diagonal Gaussian mixture model differs from the Gaussian mixture model in the calculation of the covariance. In the diagonal algorithm, the variances of each feature is calculated separately. This is where the term $\left(x_{ij} - \mu_{kj}\right)^2$ comes from. With the GMM, this squared difference between a data point and the jth feature mean is not computed.

The final implementation can be found in the Python code.

2.