Homework 5 - Math 4803

Dennis Goldenberg

April 2023

1 Theoretical Questions

1. Section 10.10 Exercise 2

a) - Show that if we add a constant c to each of the z_ℓ (in the softmax function) then the probability is unchanged.

b) Show that if we add constants c_j, $j = 0, 1, ..., p$ (in the softmax function) to each of the corresponding coefficients for each of the features, then the predictions at any new point x are unchanged.

Solution

a)

Proof. From equation 10.13, we get that, the output from the soft-max function is, with Y being the variable representing classifications and X being the input (generalizing to when there are K possible classes):

$$f_m(X) = \mathbb{P}(Y = m|X) = \frac{e^{Z_m}}{\sum_{\ell=0}^{K} e^{Z_\ell}}$$

Say we add a constant c to each of the linear model outputs and call them Z'_ℓ. Then, we have that $Z'_\ell = c + Z_\ell$. If we run the function again, calling the function on the new linear outputs $f'_m(X)$:

$$\begin{aligned}
f'_m(X) &= \frac{e^{Z'_m}}{\sum_{\ell=0}^{K} e^{Z'_\ell}} \\
&= \frac{e^{c+Z_m}}{\sum_{\ell=0}^{K} e^{c+Z_\ell}} \\
&= \frac{e^{c}e^{Z_m}}{\sum_{\ell=1}^{K} e^{c}e^{Z_\ell}} \\
&= \frac{e^{c}e^{Z_m}}{e^{c}\sum_{\ell=1}^{K} e^{Z_\ell}} \\
&= \frac{e^{Z_m}}{\sum_{\ell=1}^{K} e^{Z_\ell}} = \mathbb{P}(Y = m|X) = f_m(X)
\end{aligned}$$

Thus, the addition of the constant does not change the probabilities. $\square$

b)

Proof. Equation 4.13 in the textbook is as follows:

$$\mathbb{P}(Y = k|X = x) = \frac{e^{\beta_{k0}+\beta_{k1}x_1+...+\beta_{kp}x_p}}{\sum_{\ell=1}^{K} e^{\beta_{\ell 0}+\beta_{\ell 1}x_1+...+\beta_{\ell p}x_p}}$$

Let this be equal to a function $f_k(x)$. As in (a), Y corresponds to the prediction probabilities for each potential class given an input value. Now, we have that $X = x$, where $x = (x_1, x_2, ..., x_p)$. Let us hold that fixed, and consider new coefficients β'_{ki} that are just the old coefficients with a constant c_i added to them, where this constant is the same among all classes but there is a different features; ergo, for any arbitrary class k:

$$\beta'_{ki} = \beta_{ki} + c_i \forall i \in \{1, 2, ..., p\}$$

Note that $\beta'_{k0} = \beta_{k0}$. If we recalculate the probability, calling the function with thew new coefficients $f'_k(X)$:

$$\begin{aligned}
f'_k(x) &= \frac{e^{\beta'_{k0}+\beta'_{k1}x_1+...+\beta'_{kp}x_p}}{\sum_{\ell=1}^{K} e^{\beta'_{\ell 0}+\beta'_{\ell 1}x_1+...+\beta'_{\ell p}x_p}} \\
&= \frac{e^{\beta_{k0}+(\beta_{k1}+c_1)x_1+...+(\beta_{kp}+c_p)x_p}}{\sum_{\ell=1}^{K} e^{\beta_{\ell 0}+(\beta_{\ell 1}+c_1)x_1+...+(\beta_{\ell p}+c_p)x_p}} \\
&= \frac{\left(e^{c_1x_1+c_2x_2+...+c_px_p}\right) e^{\beta_{k0}+\beta_{k1}x_1+...+\beta_{kp}x_p}}{\sum_{\ell=1}^{k} \left(e^{c_1x_1+c_2x_2+...+c_px_p}\right) e^{\beta_{\ell 0}+\beta_{\ell 1}x_1+...+\beta_{\ell p}x_p}} \\
&= \frac{\left(e^{c_1x_1+c_2x_2+...+c_px_p}\right) e^{\beta_{k0}+\beta_{k1}x_1+...+\beta_{kp}x_p}}{\left(e^{c_1x_1+c_2x_2+...+c_px_p}\right) \sum_{\ell=1}^{k} e^{\beta_{\ell 0}+\beta_{\ell 1}x_1+...+\beta_{\ell p}x_p}} \\
&= \frac{e^{\beta_{k0}+\beta_{k1}x_1+...+\beta_{kp}x_p}}{\sum_{\ell=1}^{K} e^{\beta_{\ell 0}+\beta_{\ell 1}x_1+...+\beta_{\ell p}x_p}} = \mathbb{P}(Y = k|X = x) = f_k(x)
\end{aligned}$$

Thus, even with the added constants, the prediction probabilities are exactly the same, which means that the class with the largest probability for a given point x will also be the same; thus the predictions will be the same. □

2. Section 12.6 Exercise 1

a) - Let $C_1, C_2, ..., C_K$ be the K mutually disjoint clusters from a dataset. Let $x_i = (x_{i1}, ..., x_{ip})$ represent a point int the dataset K-means was implemented on. Prove (12.18) from the textbook:

$$\frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 = 2 \sum_{i=1}^{C_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj})^2$$

b) - On the basis of this identity, argue that the K-means clustering algorithm decreases the objective at each iteration:

$$\text{minimize}_{C_1, C_2, ..., C_k} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}$$

Solution

a)

Proof. We can do some algebraic manipulation to solve this problem. We first define $\bar{x}_{kj}$ as the average of the value of the j'th feature in the k'th cluster. Note that $\bar{x}_{kj} = \frac{1}{|C_k|} \sum_{i=1}^{C_k} x_{ij}$ and use that to solve:

$$\begin{aligned}
&\frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \\
&= \frac{1}{|C_k|} \sum_{i \in C_k} \sum_{i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \\
&= \frac{1}{|C_k|} \sum_{i=1}^{C_k} \sum_{i'=1}^{C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \\
&= \frac{1}{|C_k|} \sum_{i=1}^{C_k} \sum_{i'=1}^{C_k} \sum_{j=1}^{p} \left((x_{ij} - \bar{x}_{kj}) - (x_{i'j} - \bar{x}_{kj})\right)^2 \\
&= \frac{1}{|C_k|} \sum_{i=1}^{C_k} \sum_{i'=1}^{C_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj})^2 - \frac{1}{|C_k|} \sum_{i=1}^{C_k} \sum_{i'=1}^{C_k} \sum_{j=1}^{p} 2(x_{ij} - \bar{x}_{kj})(x_{i'j} - \bar{x}_{kj}) \\
&+ \frac{1}{|C_k|} \sum_{i=1}^{C_k} \sum_{i'=1}^{C_k} \sum_{j=1}^{p} (x_{i'j} - \bar{x}_{kj})^2
\end{aligned}$$

Note that the 1st term is calculating the euclidean distance from 2 different points and iterating over i, whereas the 3rd term is calculating over two different points and iterating over i'. However, i and i' are both indices of elements in the same cluster; therefore, these two summations are summing the same value of the same dataset. Therefore, they are equal. We use this fact to continue computations:

$$
\begin{aligned}
&\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-x_{i'j})^2\\
&=\frac{2}{|C_k|}\sum_{i=1}^{C_k}\sum_{i'=1}^{C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2-\frac{2}{|C_k|}\sum_{i=1}^{C_k}\sum_{i'=1}^{C_k}\sum_{j=1}^{p}2(x_{ij}-\bar{x}_{kj})(x_{i'j}-\bar{x}_{kj})\\
&=\frac{2}{|C_k|}\sum_{i=1}^{C_k}\sum_{j=1}^{p}\sum_{i'=1}^{C_k}(x_{ij}-\bar{x}_{kj})^2-\frac{2}{|C_k|}\sum_{i=1}^{C_k}\sum_{i'=1}^{C_k}\sum_{j=1}^{p}2(x_{ij}-\bar{x}_{kj})(x_{i'j}-\bar{x}_{kj})\\
&=\frac{2}{|C_k|}\sum_{i=1}^{C_k}\sum_{j=1}^{p}|C_k|(x_{ij}-\bar{x}_{kj})^2-\frac{2}{|C_k|}\sum_{i=1}^{C_k}\sum_{j=1}^{p}2(x_{ij}-\bar{x}_{kj})\sum_{i'=1}^{C_k}(x_{i'j}-\bar{x}_{kj})\\
&=\frac{2}{|C_k|}\sum_{i=1}^{C_k}\sum_{j=1}^{p}|C_k|(x_{ij}-\bar{x}_{kj})^2-\frac{2}{|C_k|}\sum_{i=1}^{C_k}\sum_{j=1}^{p}2(x_{ij}-\bar{x}_{kj})(|C_k|\bar{x}_{kj}-|C_k|\bar{x}_{kj})\\
&=\frac{2}{|C_k|}\sum_{i=1}^{C_k}\sum_{j=1}^{p}|C_k|(x_{ij}-\bar{x}_{kj})^2-0\\
&=\frac{2}{|C_k|}*|C_k|*\sum_{i=1}^{C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2\\
&=2\sum_{i=1}^{C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2
\end{aligned}
$$

Having shown the two values to be equal, we have proven the identity. □

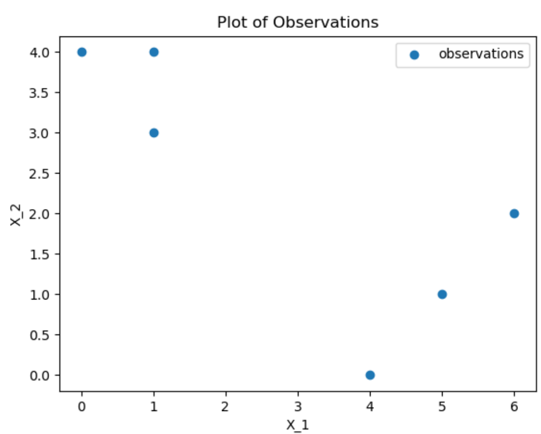
b)

Proof. In the first part of step 2 of the algorithm, after the cluster centroids are computed, the observations are reassigned the cluster with the closest centroid by euclidean distance. Therefore, for any given cluster C_k, the sum of euclidean distances is being decreased as the observations in C_k closer to other clusters are systematically moved to closer clusters, decreasing the sum of the euclidean distance in C_k by more than what the cluster obtaining that other cluster increases by, and the points closer to C_k than the centroid of their current cluster are moved into C_k, decreasing the sum of euclidean distance in that other cluster by at least the amount than the sum of euclidean distance in C_k increases; therefore, among all clusters, there is never an increase in total euclidean distance when reassignments happen; via our identity proven in part a, the sum of euclidean distances for a cluster is equal to the within-cluster variation for that cluster, and therefore, at any reassignment, the sum of all within-cluster variations does not increase after the step. With each reassignment of clusters, the sum of within-cluster variations is monotone non-increasing. □

3. Section 12.6 Exercise 3

Obs.	X_1	X_2
1	1	4
2	1	3
3	0	4
4	5	1
5	6	2
6	4	0

a) - Plot the observations
b) - Randomly assign a cluster label to each observation. ($K = 2$)
c) - Compute the centroid for each cluster.
d) - Assign each observation to the centroid to which it is closest, in terms of Euclidean distance. Report the cluster labels for each observation.
e) - Repeat (c) and (d) until the answers obtained stop changing. **f)** - In your plot from (a), color the observations according to the cluster labels obtained.
<u>Solution</u>
a) - The following plot was generated in `hw5_coding.py`:



b) - The following randomized initial clustering was generated in `hw5_coding.py` (with the labels "red" and "blue"):

```
['blue' 'blue' 'red' 'blue' 'red' 'blue']
```

Label the observations O_i for $i \in \{1, 2, 3, 4, 5, 6\}$. Therefore, our initial clusters are $C^0_{blue} = \{O_1, O_2, O_4, O_6\}$ and $C^0_{red} = \{O_3, O_5\}$.
c) - The original centroids were computed in `hw5_coding.py`, and are shown below:

```
Original Centroids: {'red': array([3., 3.]), 'blue': array([2.75, 2.  ])}
```

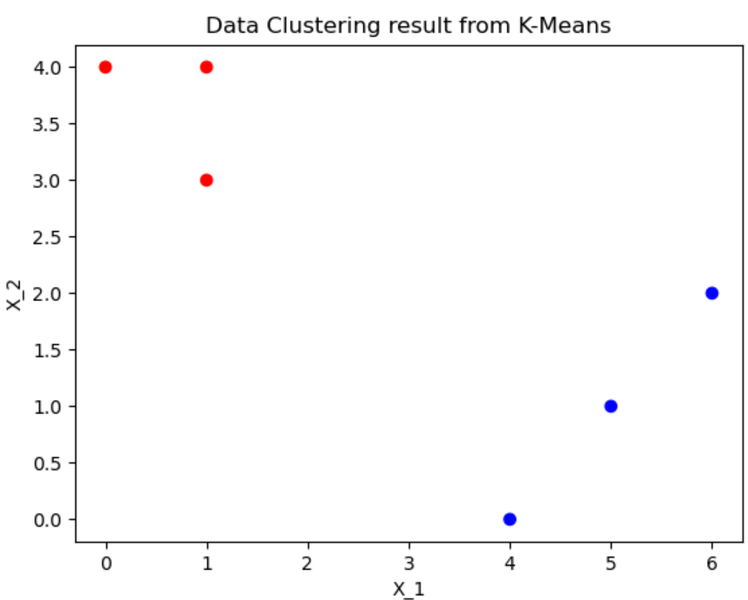
So, $\bar{x}^0_{\text{red}} = (3, 3)$ and $\bar{x}^0_{\text{blue}} = (2.75, 2)$.
d) - After the first iteration of K-means, the reassignment algorithm resulted in the following (implemented in `hw5_coding.py`):

```
Clusters after 1 iteration:  ['red', 'red', 'red', 'blue', 'red', 'blue']
```

Therefore, $C^1_{\text{red}} = \{O_1, O_2, O_3, O_5\}$ and $C^1_{\text{blue}} = \{O_4, O_6\}$.
e) - This was completed in `hw5_coding.py`. There was a total of 2 iterations of reassignment.
f) - The following clusters were the result after K-means was completed:

```
Final Clusters:  ['red', 'red', 'red', 'blue', 'blue', 'blue']
```

Therefore $C_{\text{red}} = \{O_1, O_2, O_3\}$ and $C_{\text{blue}} = \{O_4, O_5, O_6\}$. The following graph was generated after the observations were colored accordingly:



The clustering as a result from the algorithm seems intuitively correct upon examination of the graph.

4. Consider an n-complete graph, with vertex $\{v_1, v_2, ..., v_n\}$. Every vertex is connected with the other $n-1$ vertices, and the connected edge has weight 1.
1. What is the graph Laplacian?
2. What are the eigenvalues and the eigenvectors of the graph Laplacian?
Solution
1. - This graph is complete, which means that every vertex v_i is connected to every other vertex. Thus, $\forall v_i \in V(G), \deg_G(v_i) = n-1$, where $V(G)$ is the set of vertices. Thus the degree graph of this matrix is:

$$G = \begin{bmatrix} n-1 & 0 & 0 & 0 & \dots & 0 \\ 0 & n-1 & 0 & 0 & \dots & 0 \\ 0 & 0 & \ddots & 0 & \dots & 0 \\ \vdots & \vdots & 0 & \ddots & \dots & 0 \\ 0 & 0 & \dots & 0 & n-1 & 0 \\ 0 & 0 & \dots & 0 & 0 & n-1 \end{bmatrix}$$

Since each vertex is connected to every other vertex, for the Adjacency matrix W, all elements not on the diagonal are equal to 1, and all elements on the diagonal are 0 due to there being no edge from the vertex to itself; in other words:

$$W = \begin{bmatrix} 0 & 1 & 1 & 1 & \dots & 1 \\ 1 & 0 & 1 & 1 & \dots & 1 \\ 1 & 1 & \ddots & 1 & \dots & 1 \\ \vdots & \vdots & 1 & \ddots & \dots & 1 \\ 1 & 1 & \dots & 1 & 0 & 1 \\ 1 & 1 & \dots & 1 & 1 & 0 \end{bmatrix}$$

In class, we defined the graph Laplacian L as the adjacency matrix subtracted from the degree matrix; in other words:

$$L = G - W = \begin{bmatrix} n-1 & -1 & -1 & -1 & \dots & -1 \\ -1 & n-1 & -1 & -1 & \dots & -1 \\ -1 & -1 & \ddots & -1 & \dots & -1 \\ \vdots & \vdots & -1 & \ddots & \dots & -1 \\ -1 & -1 & \dots & -1 & n-1 & -1 \\ -1 & -1 & \dots & -1 & -1 & n-1 \end{bmatrix}$$

2. - To find the eigenvectors and eigenvalues, we must solve the following equation:

$$Lv = \lambda v$$

Here λ is the eigenvalue, and v the eigenvector. Solving this problem simplifies to:

$$\det(A - \lambda I_{n \times n}) = \overrightarrow{0}$$

From lecture, we know that the number of eigenvectors when $\lambda = 0$ is the number of connected components, and one of them has to be $1_{n\times 1}$ (a column of ones). Note that a complete graph has one connected component, so this is the only zero eigenvector.
Note also, if $\lambda = n$:

$$L - \lambda I_{n\times n} = -1_{n\times n}$$

This is a graph with all -1 values; it only has 1 linearly independent column as a result, so the other $n-1$ columns are linearly dependent; therefore, an eigenvector can be created from the operation of subtracting of these columns. Thus, from this matrix, you can create the following $n-1$ orthogonal eigenvectors:

$$\vec{v} \in \left\{ \begin{pmatrix} 1 \\ -1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ -1 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \cdots \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ -1 \end{pmatrix} \begin{pmatrix} -1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \right\}$$

In total, a this matrix L can only have (L) eigenvectors, so we are now done. In summary:

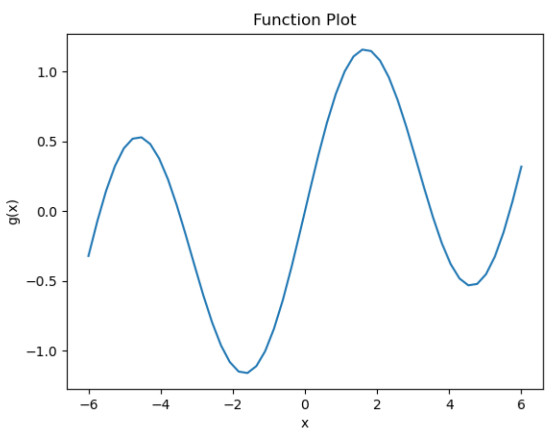
$$\lambda_1 = 0 \text{ and } v = \{1_{n\times 1}\}$$
and
$$\lambda_2 = n \text{ and } v = \left\{ \begin{pmatrix} 1 \\ -1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ -1 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \cdots \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ -1 \end{pmatrix} \begin{pmatrix} -1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \right\}$$

2 Programming Questions

5. Section 10.10, Exercise 6

Solution

a) - Let $g(x) = \sin(x) + \frac{x}{10}$. We plot $g(x)$ where $x \in [-6, 6]$ below:



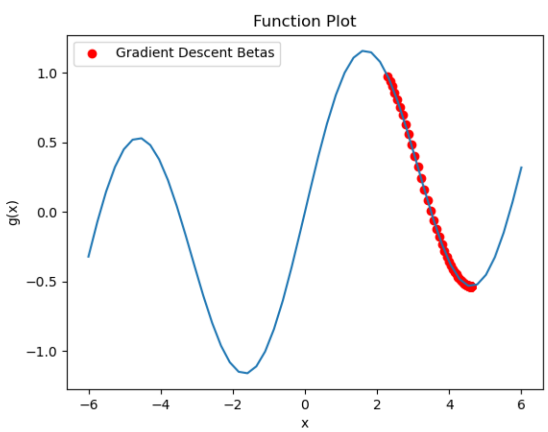
b) - To get the derivative of this function, we get:

$$\frac{d(g(x))}{dx} = \cos(x) + \frac{1}{10}$$

c) - Gradient descent was implemented in `hw5_coding.py`. The local minima at reached was the following:

```
Minima reached:  4.612221533862191
```

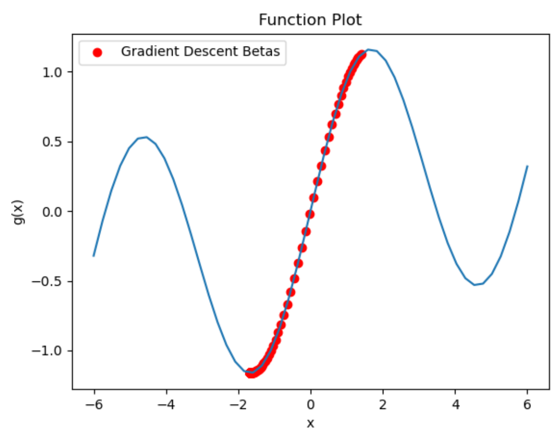
Thus, the minima of the function is roughly $x = 4.61222$. The plot of the betas is shown below.



d) - Gradient descent was implemented once again in `hw5_coding.py`. The local minima reached was the following:

```
Minima reached:  -1.6709637147384306
```

So the local minima reached is different, at the value roughly $x = -1.671$. The following is the resultant graph:



6. Section 10.10, Exercise 7

Solution

The following Neural Network was implemented in `hw5_coding.py`:

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Input_Layer (InputLayer)    [(None, 3)]               0

 Dense_Layer (Dense)         (None, 10)                40

 Dropout_Layer (Dropout)     (None, 10)                0

 dense (Dense)               (None, 2)                 22

=================================================================
Total params: 62
Trainable params: 62
Non-trainable params: 0
```

Cross-Validation was done to ensure no overfitting was being done on a specific subsection of the dataset. The following were the hyper-parameters used:

Parameter	**Value**
Number of Epochs	10
Loss Function	Cross-Entropy
Optimizer	Stochastic Gradient Descent (Adam)
Number of Folds in CV	5

When comparing the Neural Network to the Logistic Regression Model, the following Cross-Validation Error was generated:

```
Neural Net Cross-Validation Error (k = 5):  0.0334
Logistic Regression Cross-Validation Error (k = 5):  0.0317
```

Note that Logistic Regression slightly outperformed the neural network. This indicates that predicting a default from the default dataset may be a somewhat linear classification task, as logistic regression is a linear classifier.

7. Section 12.6, Exercise 10

8. Consider a dataset near two circles, both centered at the origin and having radius 1 and 2. We will generate the data in the following steps:

- First generate 200 samples on each circle. Let us paramterize the circles as $x_1(t) = r\cos(t)$, $x_2(t) = r\sin(t)$ where $r = 1, 2$ respectively. For each circle, 200 uniform samples of $t \in [0, 2\pi)$ give rise to 200 points on the circle.

- Add Gaussian noise to each sample above. The noise vector is $[n_1, n_2]$ where $n_1, n_2 \sim Normal(0, \sigma^2)$, where $\sigma = 0.05$.

a) Apply K means with $K = 2$, and display the clustering results.
b) Apply spectral clustering and cluster this data set into 2 clusters. Construct the ϵ-neighborhood graph, and display the clustering results with three choices of ϵ. Try a large ϵ, a proper ϵ, and a small ϵ. What is a good range of ϵ such that we can cluster the two circles? c) Apply spectral clustering and cluster this dataset into 2 clusters. Construct the k-nearest neighbor graph, and display the clustering results with three choices of k. Try a large k, a proper k, and a small k. What is a good range of k such that we can cluster the two circles?
d) Repeat the experiments in (b) and (c) where the dataset has large noise: $\sigma = 0.2$.
<u>Solution</u>