Introduction to ML (CS771), Autumn 2023
Indian Institute of Technology Kanpur
Homework Assignment Number 2

Student Name: Siddhant Jakhotiya
Roll Number: 211030
Date: November 16, 2023

QUESTION 1

Achieving Step 1

Given all the cluster means $\{\mu_k\}_{k=1}^K$ For each $\mathbf{x}_n$, to find the "best" cluster mean, we simply solve the problem of

$$z_{nm} = 1 \text{ where } m = \arg\min_k \|\mathbf{x}_n - \mu_k\|^2$$

Thus by setting z_{nk} to the closest cluster for each point at a time, we are working toward minimizing the loss.

SDG-based cluster mean update

When the data point $\mathbf{x}_n$ is assigned to the cluster k, where the cluster mean is denoted by μ_k, an update equation that can be used is:

$$\bar{\mu_k} = \mu_k + \eta(\mathbf{x}_n - \mu_k)$$

Where $\eta \in [0, 1]$ is the step size.

Intuitively, we can see that this is trying to minimize the loss by the following logic: once we know that $\mathbf{x}_n$ is in the kth cluster, we are reassigning the cluster mean closer to $\mathbf{x}_n$. This reduces the value of $\|\mathbf{x}_n - \mu_k\|$. We can assess convergence when the magnitude of change of μ_k becomes small. This is possible for since when most of the points of a cluster have been processed using this stochastic process, the means are expected to be close to the mean learned from K-means.

Choosing the value of η

The magnitude of the update of μ_k closer to $\mathbf{x}_n$ depends on the value of η. Larger value of η could lead to unstable updates and overshooting whereas smaller value of η would converge slowly. I believe a fixed learning rate wouldn't be a good approach to this problem (unless carefully tuned and chosen): the reasoning is that when most points have been processed, μ_k denotes the mean of all those points that belong to that class. A new point should have a weight inversely proportional to the number of points already in that cluster.

I think a suitable learning rate for the problem would be $\frac{1}{t_k}$, where t_k is the number of times the k^{th} class mean has been updated. This would simulate μ_k to be the mean of all the points that were considered to be a part of that class, similar to K-Means, as

$$\bar{\mu_k} = \frac{t-1}{t}\mu_k + \frac{1}{t}\mathbf{x}_n$$

which is the weighted mean of the previous t-1 points (with weight t-1) and the new point (with weight 1).

Introduction to ML (CS771), Autumn 2023
Indian Institute of Technology Kanpur
Homework Assignment Number 2

Student Name: Siddhant Jakhotiya
Roll Number: 211030
Date: November 16, 2023

QUESTION 2

Define the mean of the classes as follows:

$$\mathbf{m}_+ = \frac{\sum\limits_{n:y_n=+1} \mathbf{x}_n}{\sum\limits_{n:y_n=+1} 1} \qquad \text{and} \qquad \mathbf{m}_- = \frac{\sum\limits_{n:y_n=-1} \mathbf{x}_n}{\sum\limits_{n:y_n=-1} 1}$$

Now, given a vector $\mathbf{w}$, the projection of these mean vectors on $\mathbf{w}$ is $(\mathbf{m}_+^T\mathbf{w})\hat{\mathbf{w}}$ and $(\mathbf{m}_-^T\mathbf{w})\hat{\mathbf{w}}$. Thus the square of the distance between these vectors is

$$\mathcal{L}_1(\mathbf{w}) = \left\| \left((\mathbf{m}_+ - \mathbf{m}_-)^T \mathbf{w} \right) \hat{\mathbf{w}} \right\|^2$$

where $\hat{\mathbf{w}}$ denotes the unit vector along $\mathbf{w}$.

To make the distance between the mean of the classes in the projected plane as large as possible, we can equivalently attempt to maximize the square of the distance (as mentioned in $\mathcal{L}_1$). Thus we need to maximize $\mathcal{L}_1$.

Now, to make the intra-class separation small (inputs within one class to be as close as possible), consider any training example $\mathbf{x}_n$ s.t. $y_n = 1$. Its distance from the class mean in the projected space is

$$\left\| \left((\mathbf{x}_n - \mathbf{m}_+)^T \mathbf{w} \right) \hat{\mathbf{w}} \right\|$$

Iterating over all n, we can sum it up as follows (and considering the square of distance):

$$\mathcal{L}_2(\mathbf{w}) = \sum_{n:y_n=+1} \left\| \left((\mathbf{x}_n - \mathbf{m}_+)^T \mathbf{w} \right) \hat{\mathbf{w}} \right\|^2 + \sum_{n:y_n=-1} \left\| \left((\mathbf{x}_n - \mathbf{m}_-)^T \mathbf{w} \right) \hat{\mathbf{w}} \right\|^2$$

Our aim is to minimize the within-class separation. This can be achieved if we try to minimize $\mathcal{L}_2(\mathbf{w})$ as each term would then be as close to the mean as possible (in the projected direction).

To achieve both the objectives, a possible objective function is

$$\mathcal{L}(\mathbf{w}) = \arg\max_{\mathbf{w}} \left(\mathcal{L}_1(\mathbf{w}) - \mathcal{L}_2(\mathbf{w}) \right)$$

This objective function is maximized when $\mathcal{L}_1(\mathbf{w})$ is maximized (inter-class separation is high) and when $\mathcal{L}_2(\mathbf{w})$ is minimized (intra-class separation is low).

Introduction to ML (CS771), Autumn 2023
Indian Institute of Technology Kanpur
Homework Assignment Number 2

Student Name: Siddhant Jakhotiya
Roll Number: 211030
Date: November 16, 2023

QUESTION 3

Given the eigenvector $\mathbf{v}$ for the matrix $\frac{1}{N}\mathbf{X}\mathbf{X}^T$ (call this matrix $\mathbf{M}$), let its corresponding eigenvalue be λ. So we have:

$$\frac{1}{N}\mathbf{X}\mathbf{X}^T\mathbf{v} = \lambda\mathbf{v}$$

The LHS and RHS results in a vector of dimension Nx1. So we can pre-multiply both sides by $\mathbf{X}^T$ (which is of size DxN). Multiplying that, we get,

$$\begin{aligned}\left(\frac{1}{N}\mathbf{X}^T\mathbf{X}\right)(\mathbf{X}^T\mathbf{v}) &= \lambda\mathbf{X}^T\mathbf{v}\\ \mathbf{S}\mathbf{X}^T\mathbf{v} &= \lambda\mathbf{X}^T\mathbf{v}\end{aligned}$$

as $\mathbf{S} = \frac{1}{N}\mathbf{X}^T\mathbf{X}$. Thus we have found an eigenvector of $\mathbf{S}$ as $\mathbf{X}^T\mathbf{v}$. This is the required eigenvector $\mathbf{u}$.

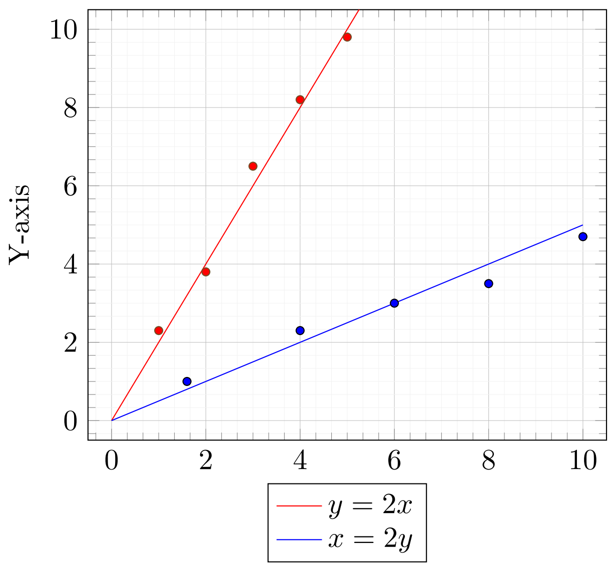
The advantage is that when D > N, it is easier to compute the eigenvector for matrix $\mathbf{M}$ of size NxN as compared to matrix $\mathbf{S}$ as $\mathbf{M}$ is smaller. Also, storing the matrix $\mathbf{M}$ is more space efficient.

Student Name: Siddhant Jakhotiya
Roll Number: 211030
Date: November 16, 2023

Part 1

The described model divides the datasets into different groups (based on the value of z_n) and trains the model parameters for each group separately. It would be useful to capture multiple linear regression models within the same dataset (See Figure 1).

Figure 1



In Figure 1, if we try to learn a linear model for the entire dataset, we'd get very poor results. However, if we split the data into 2 linear models, our model would perform well. This is achieved by segmenting the training examples into regions, each with its own linear model. This is achieved by assigning each $\mathbf{x}_n$ a cluster-id z_n which would denote which part of the dataset it belongs to.

Part 2

The ALT-OPT algorithm could be implemented as follows:

Update equation for W

Given that z_n follows a multinoulli distribution and y_n follows a Gaussian distribution and the value Θ^{old} from the previous iteration, we can model the probability of $\mathbf{x}_n$ belonging to cluster k as follows:

$$P(z_n = k|y_n, \mathbf{x}_n, \Theta^{old}) = \frac{P(y_n|z_n = k, \mathbf{x}_n, \Theta^{old})P(z_n = k|\mathbf{x}_n, \Theta^{old})}{\sum_{j=1}^{K} P(y_n|z_n = j, \mathbf{x}_n, \Theta^{old})P(z_n = j|\mathbf{x}_n, \Theta^{old})}$$

Where, $P(y_n|z_n, \mathbf{x}_n\Theta^{old})$ can be computed as $\mathcal{N}(y_n|\mathbf{w}_{z_n}^T, \mathbf{x}_n, \beta^{-1})$ and $P(z_n = k|\mathbf{x}_n, \Theta^{old})$ is modelled using a multinoulli.

Thus, in each iteration we update z_n as

$$z_n^{new} = \arg\max_k P\left(z_n = k|\mathbf{x}_n, \Theta^{old}\right)$$

Update equation for Θ

$\mathbf{W}$, the parameter of the Gaussian distribution can be maximized by maximizing the log likelyhood of the input.

$$\mathbf{W}^{new} = \arg\max_{\mathbf{W}} \sum_{n=1}^{N}\sum_{k=1}^{K}\Big(P(z_n = k|y_n, \mathbf{x}_n, \Theta^{(old)}) \log P(y_n|z_n = k, \mathbf{x}_n, \mathbf{W})\Big)$$

Thus for each individual $\mathbf{w}_k$, we have the update equation

$$\mathbf{w}_k^{(new)} = \left(\sum_{n=1}^{N} p(z_n = k|y_n, \mathbf{x}_n, \Theta^{(old)})\mathbf{x}_n\mathbf{x}_n^T\right)^{-1}\left(\sum_{n=1}^{N} p(z_n = k|y_n, \mathbf{x}_n, \Theta^{(old)})\mathbf{x}_n y_n\right)$$

π_k would be used to denote the fraction of points belonging to cluster k. This can be updated as

$$\pi_k^{(new)} = \frac{1}{N}\sum_{n=1}^{N} P(z_n = k|y_n, \mathbf{x}_n, \Theta^{(old)})$$

When $\pi_k = 1/K \forall k$, $P(z_n = k|\mathbf{x}_n, \Theta^{old}$ simplifies to $1/K\forall k$. Thus the update equation for z_n becomes

$$P(z_n = k|y_n, \mathbf{x}_n, \Theta^{old}) = \frac{P(y_n|z_n = k, \mathbf{x}_n, \Theta^{old})(\frac{1}{K})}{\sum_{j=1}^{K} P(y_n|z_n = j, \mathbf{x}_n, \Theta^{old})(\frac{1}{K})}$$

$$P(z_n = k|y_n, \mathbf{x}_n, \Theta^{old}) = \frac{P(y_n|z_n = k, \mathbf{x}_n, \Theta^{old})}{\sum_{j=1}^{K} P(y_n|z_n = j, \mathbf{x}_n, \Theta^{old})}$$

The intuition behind the update of $\mathbf{W}$ is that we try to maximize the likelyhood of the observed data. This likelyhood is a measure of how well the model explains the observed data.

The intuition behind the update of π_k is to correctly compute the fraction of points from each dataset, which would influence our future updates of z_n by having a weighted average of $P(y_n)$ for each cluster.

The intuition behind the update of z_n is to compute the most likely cluster that a given test point belongs to, given the predicted output for the point (when it is considered in a given class) weighted by the number of points in the dataset observed from that class.

Introduction to ML (CS771), Autumn 2023
Indian Institute of Technology Kanpur
Homework Assignment Number 2

Student Name: Siddhant Jakhotiya
Roll Number: 211030
Date: November 16, 2023

QUESTION

5

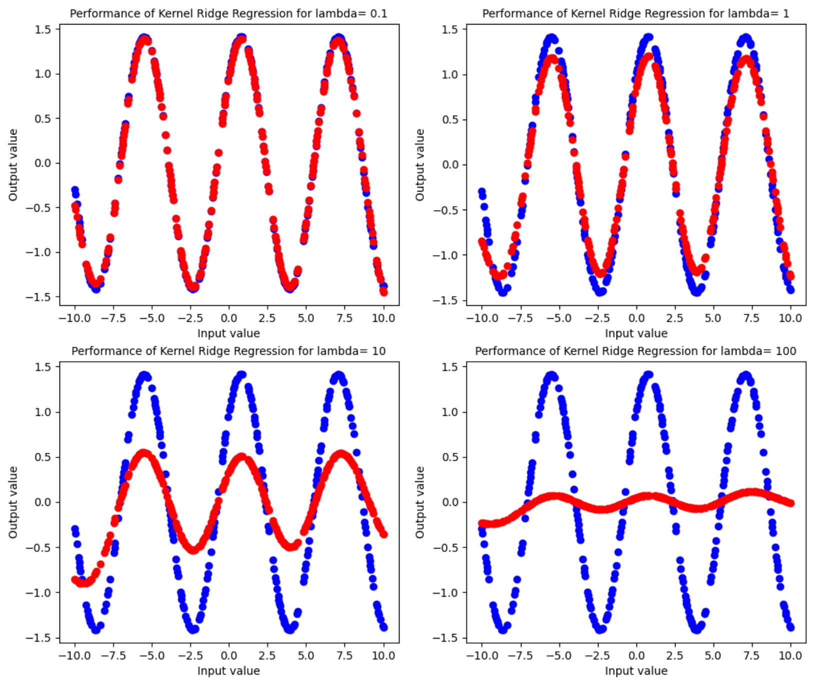
Part 1

The following results are observed for kernel ridge regression with different regularization parameter:

Value of λ	Value of RMSE
0.1	0.03257767
1	0.17030390
10	0.60926716
100	0.91108581

Table 1: Value of RMSE for different values of λ

The plot obtained is:



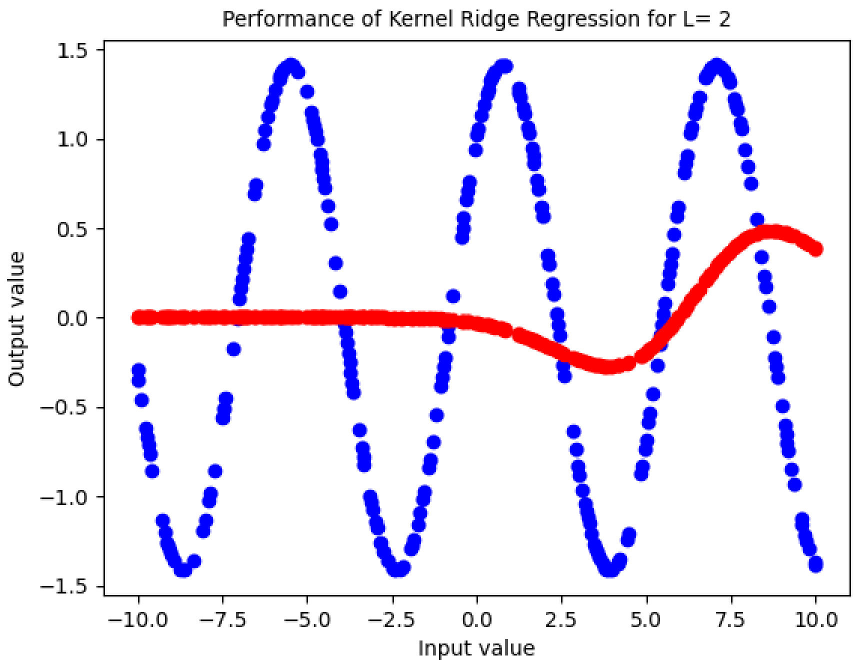
It can be observed that as the regularization is decreased, the model is able to fit the training data increasingly well. It can also be seen that the kernel model is able to capture non-linearity in the data!

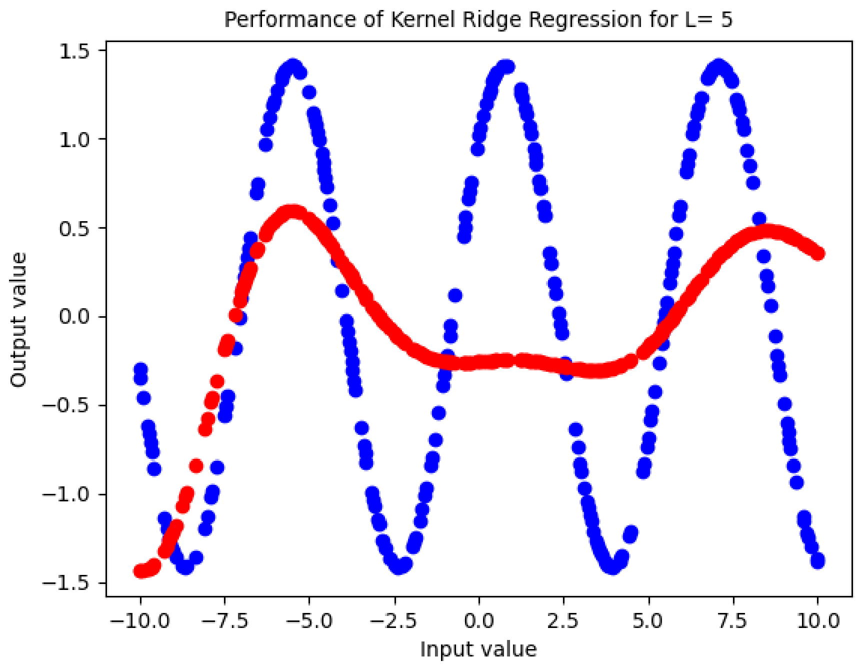
For part 1.2, we used regression with landmark based features. The following RMSE value was obtained for different values of λ and L. It can be seen that as the value of λ increases, we get worse performance, as is expected (since when heavily regularized, the model is not able to

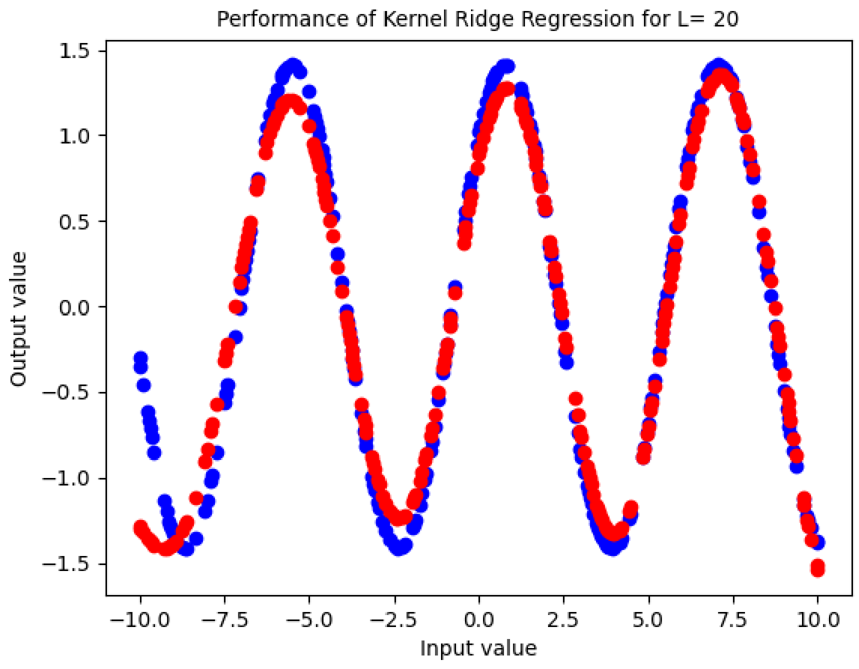
		Value of L				
		2	**5**	**20**	**50**	**100**
Value of λ	**0.1**	0.97458035	0.84410252	0.16109625	0.08257300	05837879
	1	0.96806744	0.92831694	0.36253595	0.25124535	0.15086218
	10	0.96537016	0.91391702	0.80650935	0.62730012	0.46624593
	100	0.97292244	0.95620426	0.93842416	0.88525641	0.84389033

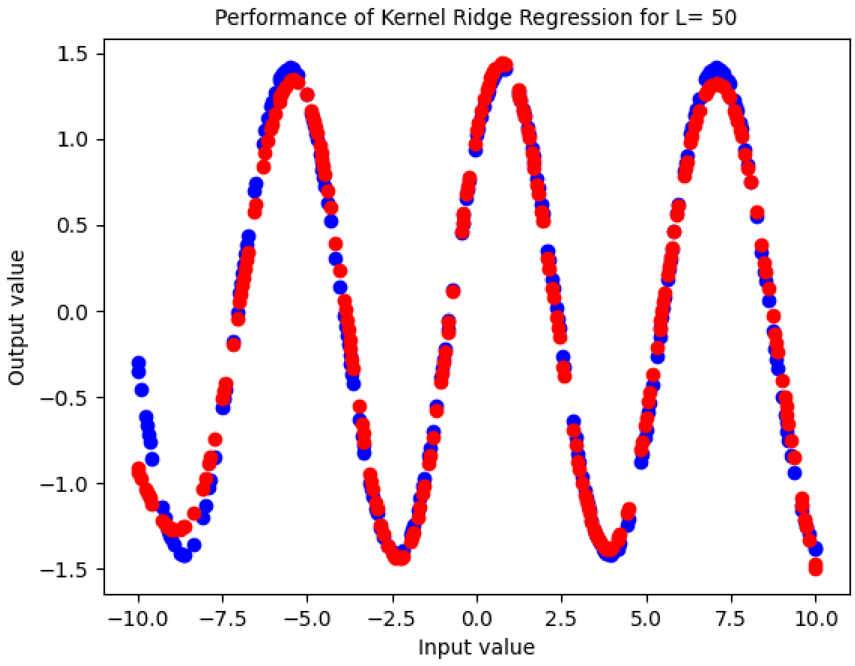
Table 2: Table showing the RMSE value for different values of L and λ

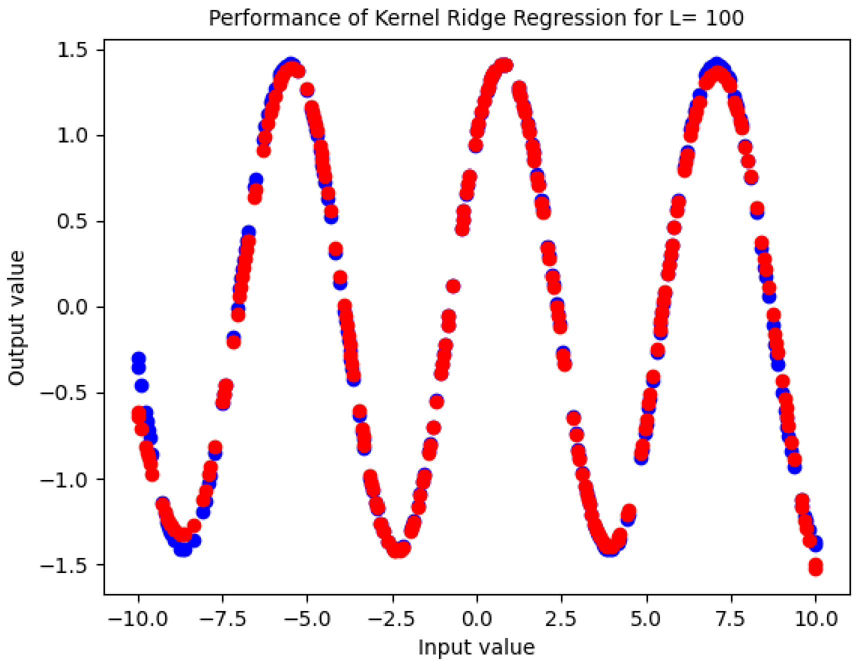
capture trends in the data). Also, for an increasing number of landmark features, the RMSE value decreases. The plots obtained for different values of L are as follows:




Performance of Kernel Ridge Regression for L= 5
Output value
Input value
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

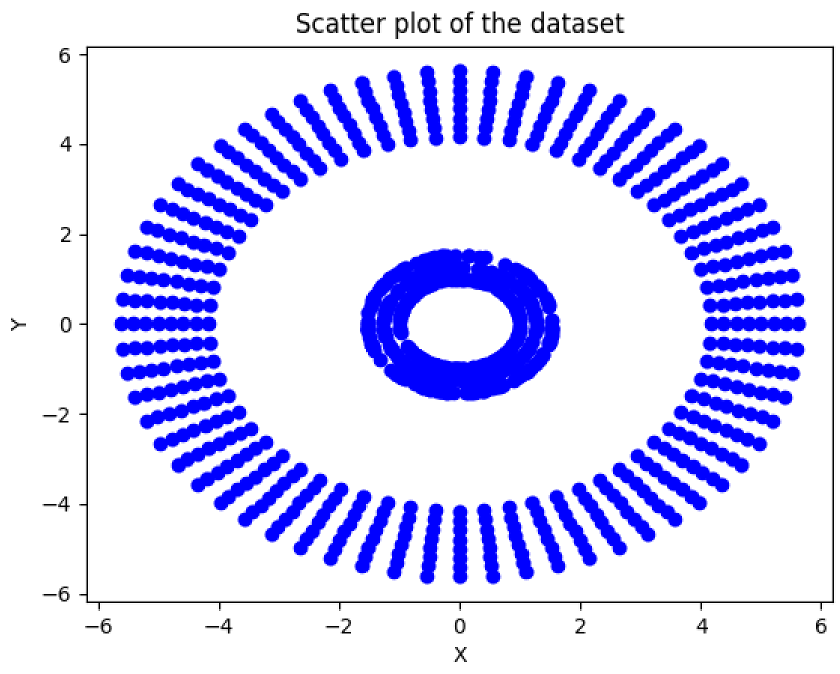

Performance of Kernel Ridge Regression for L= 20
Output value
Input value
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


Performance of Kernel Ridge Regression for L= 50
Output value
Input value
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

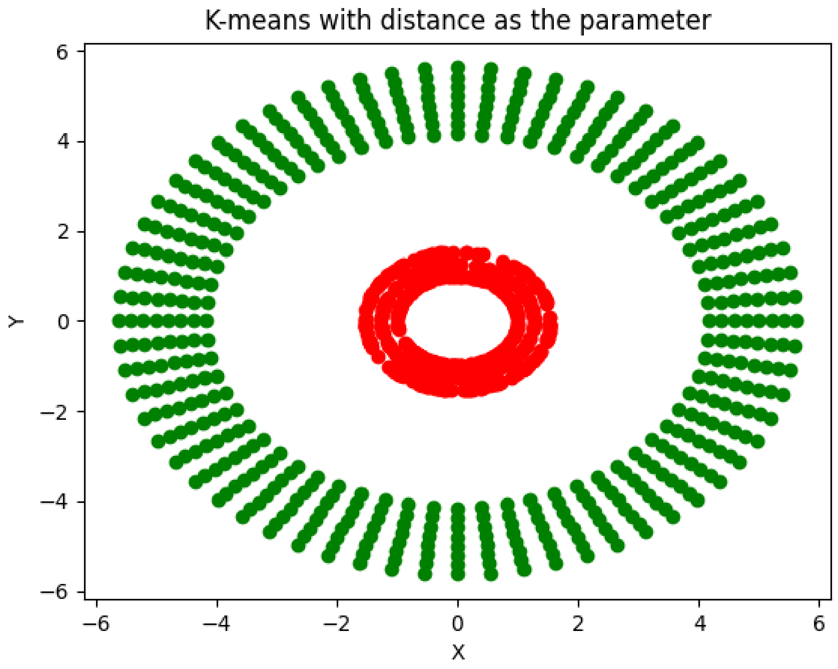

Performance of Kernel Ridge Regression for L= 100
Output value
Input value
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

L = 20 seems good enough to capture the input trend in the data without (potentially) overfitting on the training data by selecting too many training examples as landmark features. For values of $L < 20$, we can see that the model is not able to capture the non-linearity and trends sufficiently well. For $L > 20$, the non-linearity captured increases but the reduction in RMSE value is very small compared to the increase in value of L. L=20 is sort of the "elbow" point if we were to plot RMSE vs L graph.

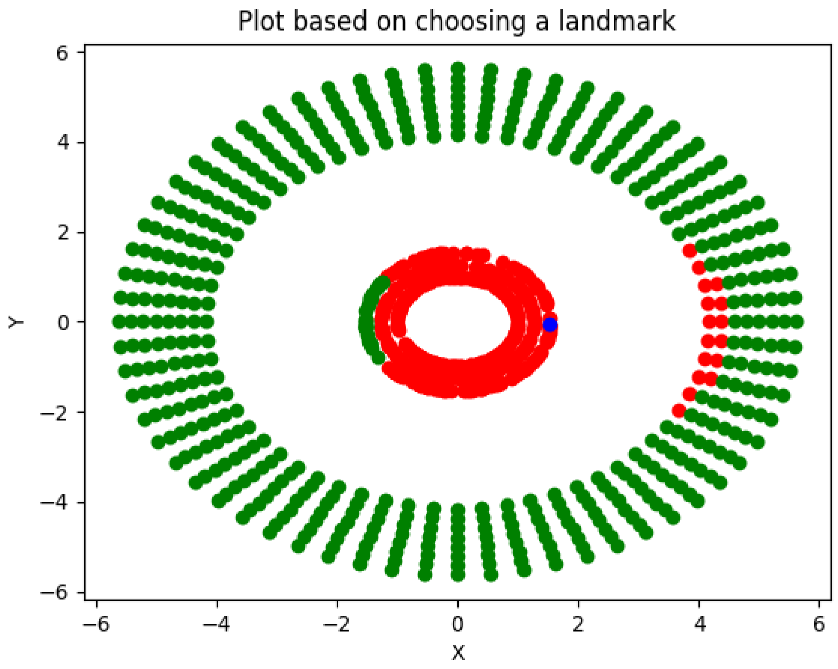
Part 2



It can be observed that the points can be easily divided into 2 rings centred at origin. Thus a very intuitive feature to divide the dataset is to consider the distance from the origin of a point. When we consider the distance r, it would give us 2 very distinct ranges corresponding to the 2 clusters. The plot is attached below (the cluster means are initialized to the first two points in the dataset to avoid randomness). It can be seen that the clustering is accurate!


K-means with distance as the parameter
Y
X
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

Next, we select a random landmark point and compute kernels based on that. The run of this algorithm on ten different iterations is shown below:

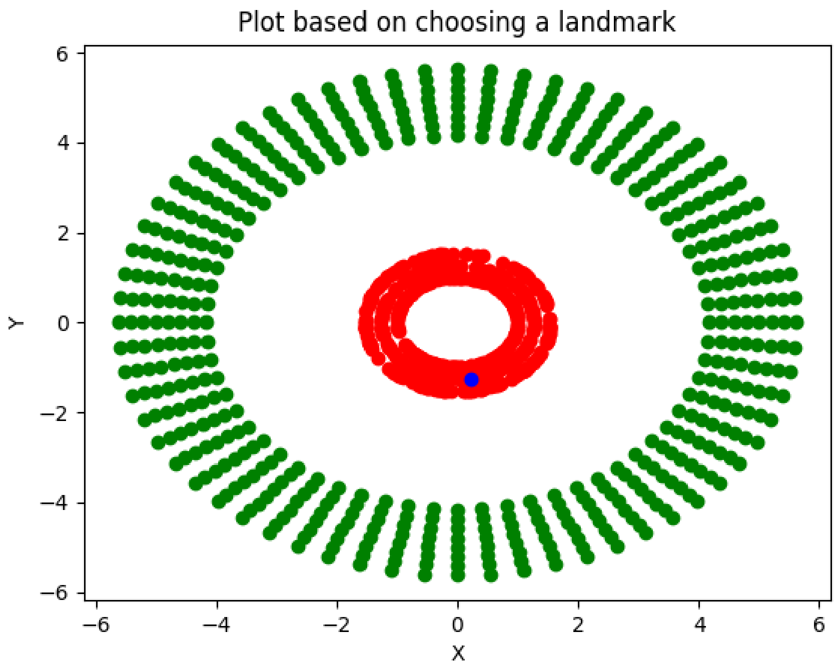


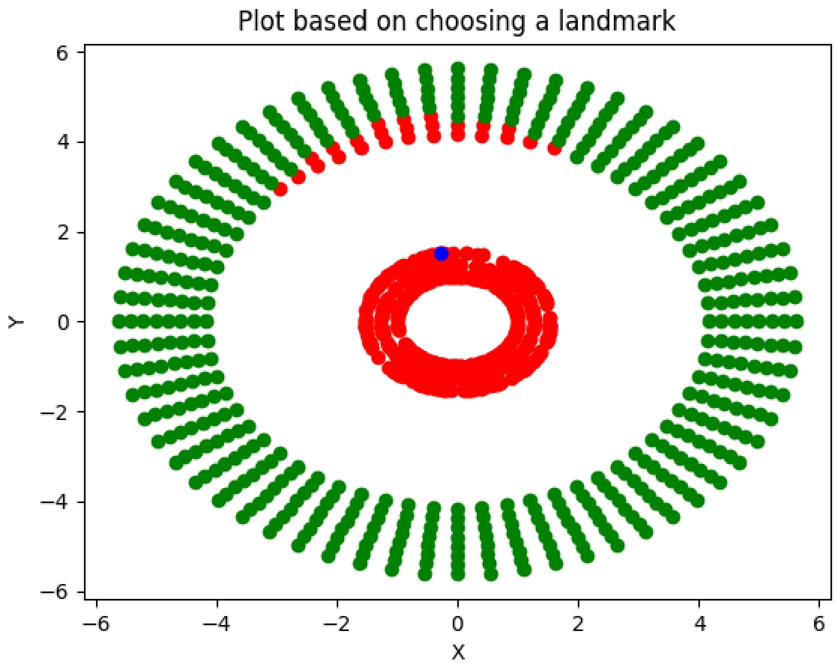
It can be observed that when the landmark point lies closer to the inner region of the inner circle, we get (near) perfect classification. This is due to the fact that we are using the RBF kernel which is computing similarity based on distance from the landmark. It can be oberved that a cluster is roughly centred around the landmakr point. When the point is

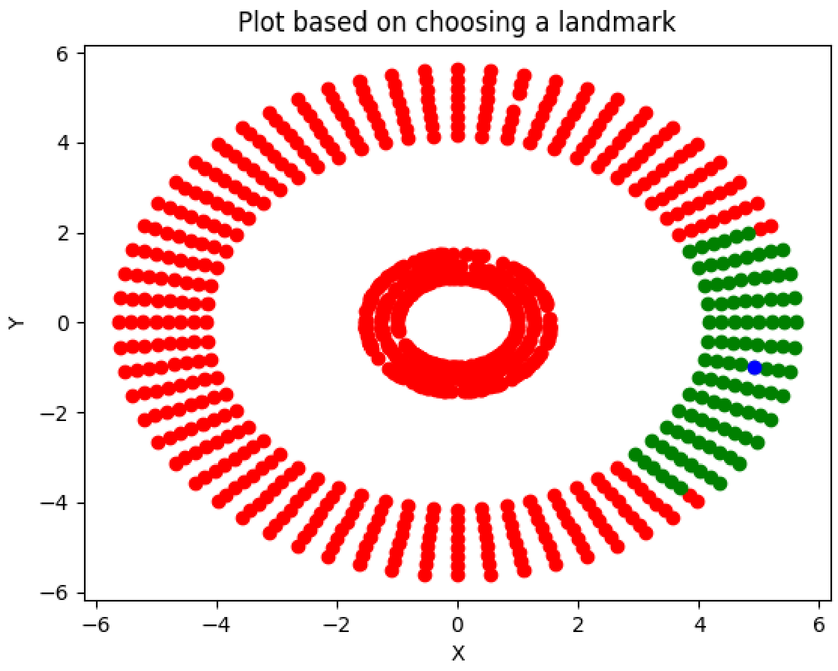
- **On the outer ring**: We get really poor clustering. This is due to the fact that when we compute similarity with this landmark based on Euclidean distance, the points lying on the same ring but diametrically opposite to the landmark have the *least amount of similarity* with the point.

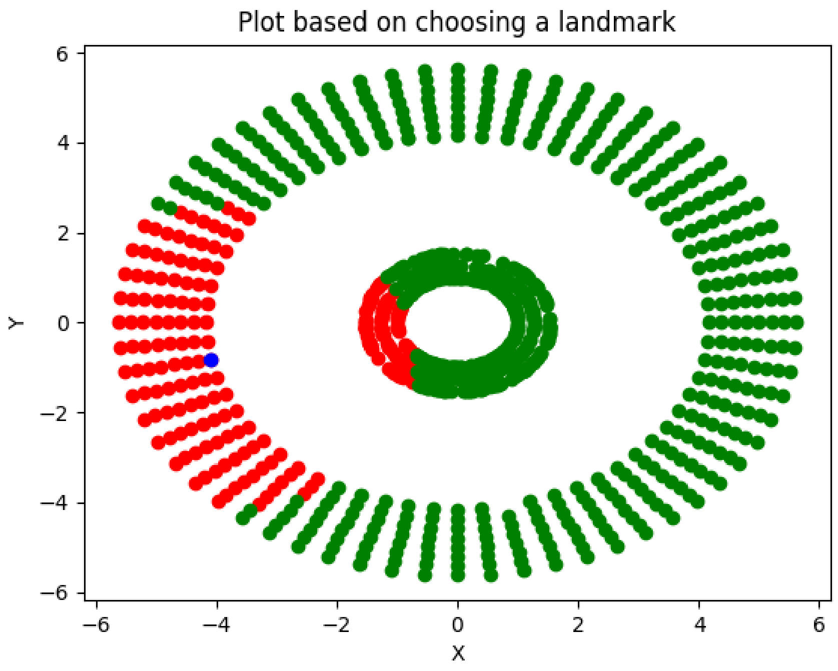
- **In the outer region of the inner ring**: Almost all the points of the inner ring result in a high similarity with the given landmark. But, simultaneously, some points of the outer ring (which are closest to the landmark point) give nearly equal similarity since they are almost at the same distance.

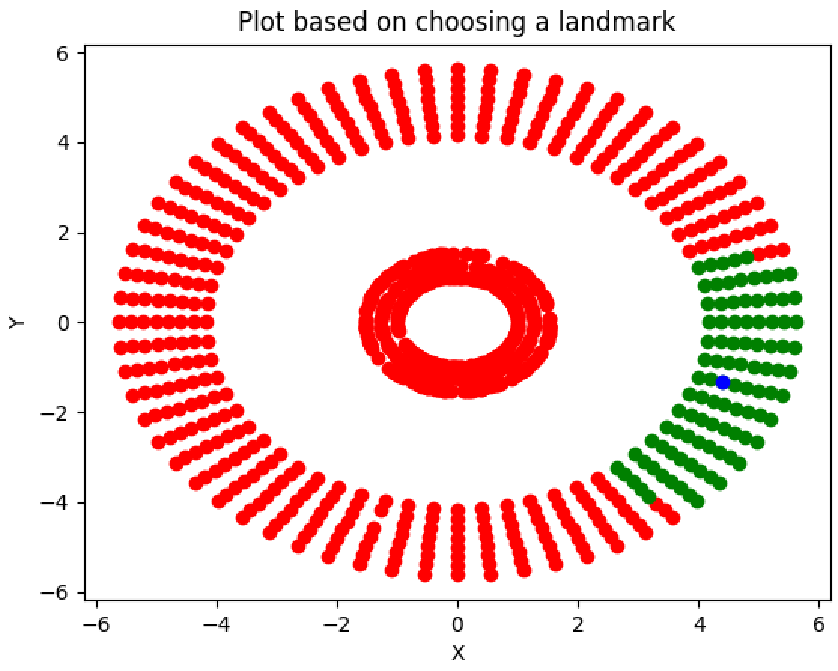
- **In the inner region of the inner ring**: All the points of the inner ring have a comparable similarity with the landmark, which is distinctly smaller than the similarity of the points on the outer ring with the landmark. This results in almost perfect classification.

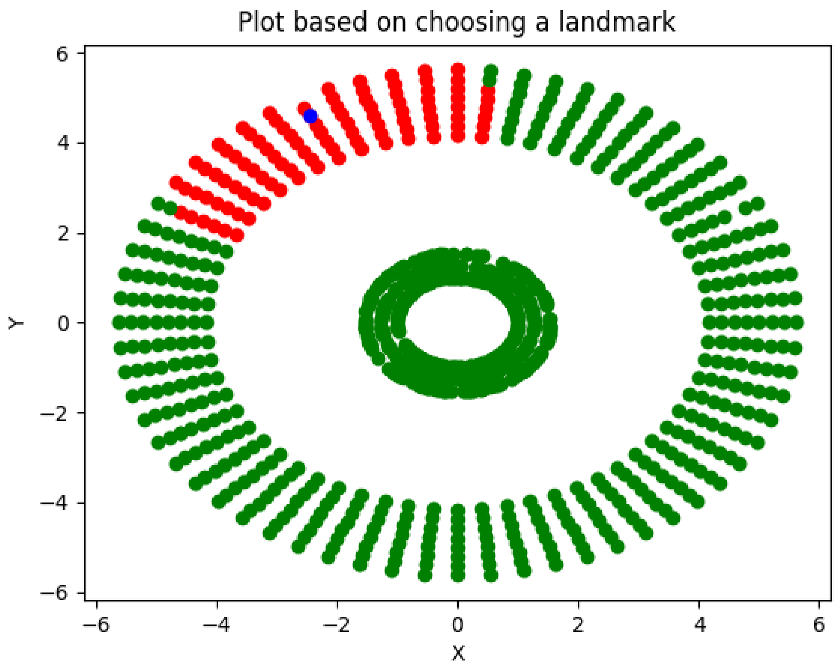

Plot based on choosing a landmark
Y
X
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

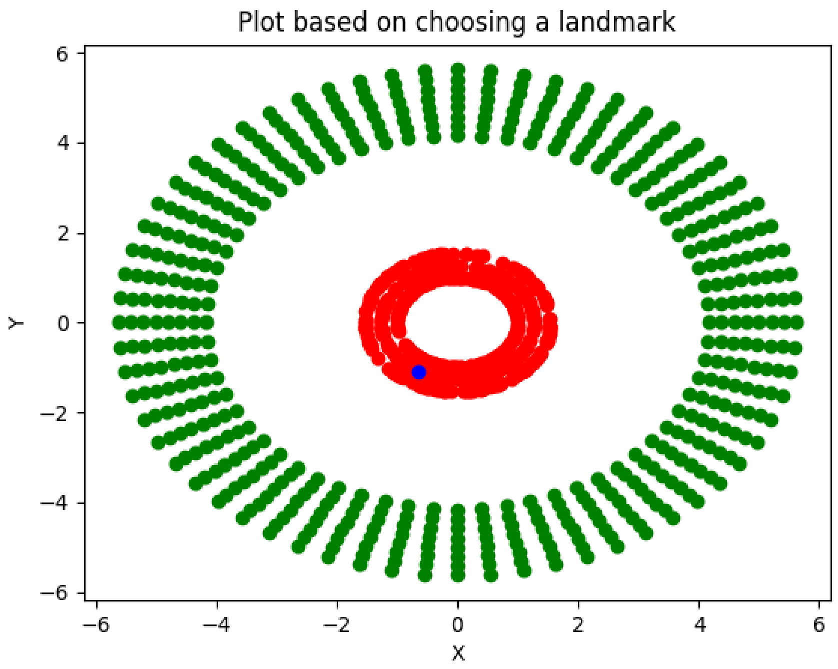

Plot based on choosing a landmark
Y
X
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

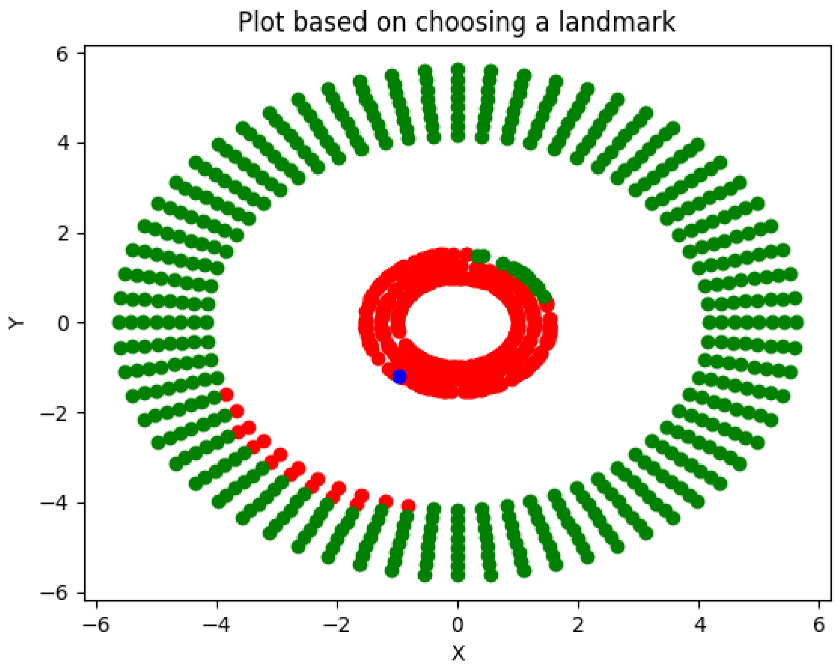

Plot based on choosing a landmark
Y
X
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

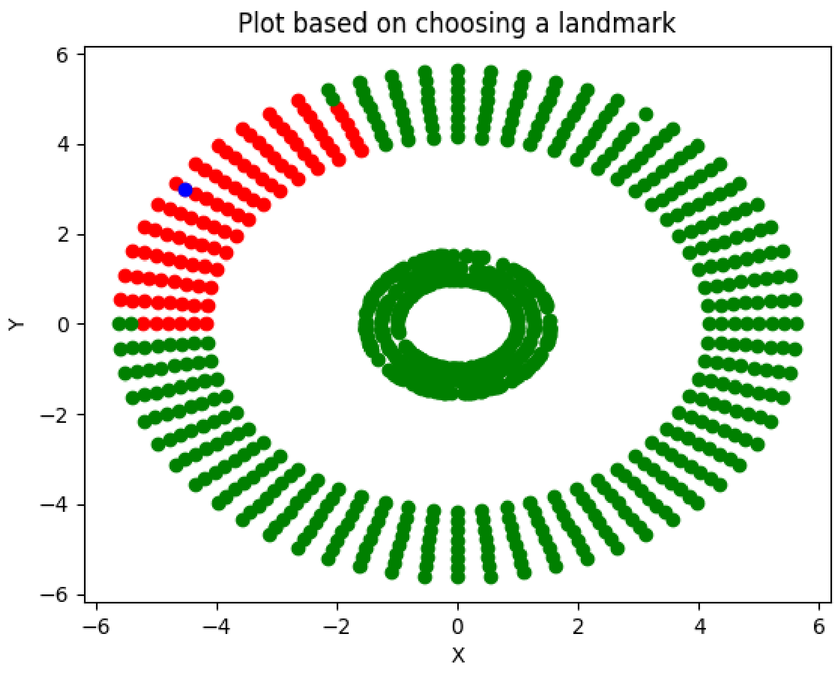

Plot based on choosing a landmark
Y
X
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6


Plot based on choosing a landmark
Y
X
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6


Plot based on choosing a landmark
Y
X
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

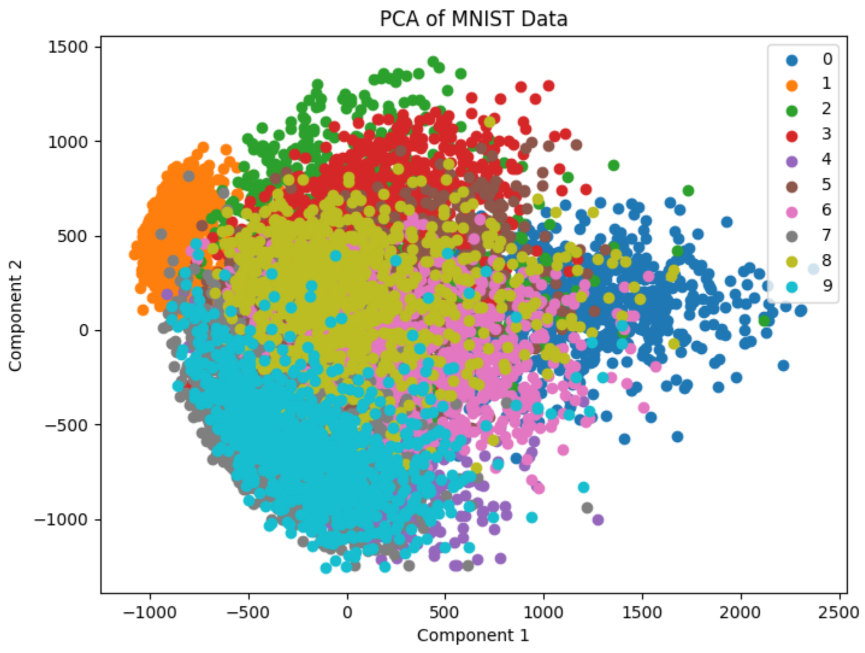

Plot based on choosing a landmark
Y
X
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

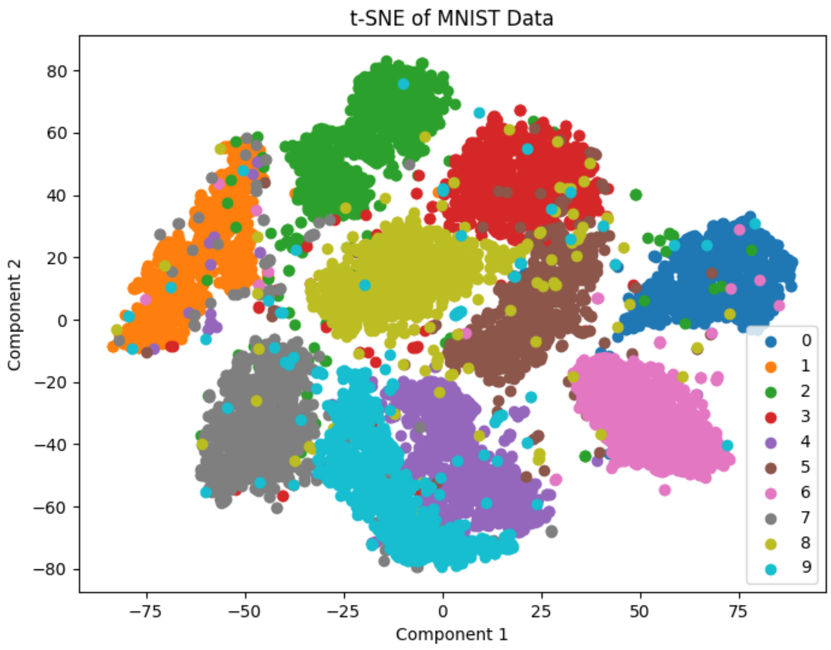

Plot based on choosing a landmark
Y
X
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6



Plot based on choosing a landmark
6
4
2
0
−2
−4
−6
Y
−6
−4
−2
0
2
4
6
X

Part 3 As can be seen from the plots (attached below), PCA has a large overlap between classes whereas t-SNE results in largely disjoint and distinct classes.




t-SNE of MNIST Data
Component 1
Component 2
80
60
40
20
0
−20
−40
−60
−80
−75
−50
−25
0
25
50
75
● 0
● 1
● 2
● 3
● 4
● 5
● 6
● 7
● 8
● 9