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

Student Name: Anshuman Barnwal
Roll Number: 200259
Date: November 16, 2023

QUESTION 1

K-means objective function and gradient for a single sample x_n would be:

$$\mathcal{L} = \sum_{k=1}^{k} z_k \|x_n - \mu_k\|^2 \qquad \frac{\partial \mathcal{L}}{\partial \mu_k} = 2(\mu_k - x_n)$$

1. Assigning best cluster to sample x_n by $\mu_n = \arg\min_{\mu_k} \|x_n - \mu_k\|^2$, i.e. the closest mean to the sample.

2. We could now optimize only for this particular mean, because $\mathcal{L}$ would have contributions from $z_k \ni \mu_k \neq \mu_n$ as 0, giving update equation as:

$$\begin{aligned}\mu_k^{(t+1)} &= \mu_k^{(t)} - \eta \frac{\partial \mathcal{L}}{\partial \mu_k} \\ &= \begin{cases} \mu_k^{(t)} - \eta \cdot 2(\mu_k^{(t)} - x_n) & \mu_k = \mu_n \\ \mu_k^{(t)} & else \end{cases}\end{aligned}$$

   This implies only the best cluster mean changes during this iteration. Intuitively, such an update makes sense because once alloted a mean, the sample doesn't affect or contribute to other means. Also the update on the alloted mean makes sense as it can be re-written as $(1-2\eta)\cdot\mu_k^{(t)} + 2\eta \cdot x_n$, which is of the form of a rolling weighted average, that "pushes" the mean towards the sample x_n, leading to lesser loss value for this sample.

3. This loss optimization is no different than any other SGD optimization techniques, therefore general methods of deciding step size η should work. Adaptive learning rate, Adam should be therefore used. Now the update rule would transform to

$$\begin{aligned}\eta_t &= C \cdot \begin{bmatrix} \frac{1}{\sqrt{\epsilon + \sum_{\tau=1}^{t}(g_1^{(t)})}} \\ \vdots \end{bmatrix}_{\mathtt{n}\times 1} \\ m^{(t)} &= m^{(t-1)} + \eta_t \odot \frac{\partial \mathcal{L}}{\partial \mu_n} \\ \mu_n^{(t+1)} &= \mu_n^{(t)} - m^{(t)}\end{aligned}$$

   Where $\odot$ is element-wise scaling, and $\mathtt{n}$ is dimension of x_n This choice leads to taking care of making larger steps at start and slowing down when nearing convergence, also it takes care of directions which need more movement by tracking gradient sum in all directions. Here C can be made as $\min_{i,j}\|\mu_i - \mu_j\| * \beta$ to take into account distance scaling per step, where β may be some other constant like 0.01.

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

Student Name: Anshuman Barnwal
Roll Number: 200259
Date: November 16, 2023

QUESTION 2

Given training dataset $\mathcal{X} = \{x_n \in \mathbb{R}^D, y_n \in \{-1, +1\}\}$, we need a projection model that uses a projection direction $w \in \mathbb{R}^D$ and

1. Distance between means of inputs are as large as possible
2. Inputs within each class become as close to each other as possible

Let both cluster means be:

$$\phi_+ = \frac{1}{N_+} \sum_{n}^{y_n=+1} x_n \qquad \phi_- = \frac{1}{N_-} \sum_{n}^{y_n=-1} x_n$$

The projection model is of form $z_n = g(x_n)$, where the projection technique might be a general $g(x_n) = \langle w, x_n \rangle = w \cdot x_n = w^T \mathbf{A} x_n$. Defining this way means projection is distributive over addition, so mean of projections is the same as projection of the mean.

Distance between class means can be written as:

$$d(w, \mathcal{X}) = \left\| w \cdot \phi_+ - w \cdot \phi_- \right\| \tag{1}$$

Making a cluster's samples close to each other is equal to reducing their variance, which again is equivalent to reducing a sample's distance from the cluster mean. Variance within both clusters can be written as:

$$v(w, \mathcal{X}) = \frac{1}{N_+} \sum_{n}^{y_n=+1} (w \cdot x_n - w \cdot \phi_+)^2 + \frac{1}{N_-} \sum_{n}^{y_n=-1} (w \cdot x_n - w \cdot \phi_-)^2 \tag{2}$$

Designing an objective using above both parts, such that $d(w, \mathcal{X})$ increases and $v(w, \mathcal{X})$ decreases is:

$$\mathcal{L}(w, \mathcal{X}) = v(w, \mathcal{X}) - d(w, \mathcal{X}) \tag{3}$$

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

Student Name: Anshuman Barnwal
Roll Number: 200259
Date: November 16, 2023

QUESTION 3

Let $X_{(N\times D)}$ be the dataset that we want to apply PCA on. Also given $D > N$.
We're given the eigen vectors of matrix $\frac{1}{N}XX^T$, so let u be one of the eigen vectors:

$$\begin{aligned} XX^Tu &= \lambda u \\ X^T(XX^Tu) &= X^T\lambda u \\ X^TX(X^Tu) &= \lambda(X^Tu) \\ X^TXw &= \lambda w \end{aligned}$$

By this modification, we can see that eigen-vector w of matrix $S = \frac{1}{N}X^TX$ can be obtained by transforming u with X^T.

This method is quite effective way to calculate eigen vectors in case of $D > N$ because eigen decomposition of a $d \times d$ matrix is a computationally heavy task dependant on d. Eigen value decomposition of $(X^TX)_{(D\times D)}$ is much more expensive than that of $(XX^T)_{(N\times N)}$. The best current complexity of Eigenvalue decomposition is $\mathcal{O}(n^3 + n^2lg^2(n))$ as shown here.

The only extra computation required after eigen-vector matrix V is obtained as $XX^T = V^T\Lambda V$ by transforming it with X^T as shown in above calculation, which would is just a matrix multiplication of naive complexity $\mathcal{O}(n^3)$.

Final calculation comparison would be:

- From X^TX, eigen value decomposition of $D\times D$, taking complexity of $\mathcal{O}(D^3+D^2lg^2(D))$.
- From XX^T, eigen value decomposition of $N\times N$, followed by matrix multiplication, taking complexity of $\mathcal{O}(N^3 + N^2lg^2(N) + N^2D)$

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

Student Name: Anshuman Barnwal
Roll Number: 200259
Date: November 16, 2023

QUESTION 4

A probabilistic linear regression model yields a distribution per input, here it's given by:

$$p(y_n|x_n, w) = \mathcal{N}(y_n|w^T x_n, \beta^{-1})$$

The proposed modelling technique is to define a latent variable $z_n \in \{1, 2, \cdots K\}$ corresponding to each x_n where each value of z_n corresponds to a different probabilistic linear regression model, i.e. a different $z_n = k$ corresponds to w_k.

1. This method is more powerful than standard probabilistic linear model because it models multiple linear regressions simultaneously and allots each sample to one of them; simulating what a regression decision tree might perform, leading to a non-linear model overall.

2. Optimizing this through ALT-OPT implies we're solving for Complete Data Log Likelihood; we want to perform hard guesses of Z, and use that to optimize the weights $\Theta = \{\{w_1, \cdots w_K\}, \{\pi_1, \cdots \pi_K\}\}$.

   (a) For each n, compute the most probable value of z_n via the *conditional posterior*

   $$\hat{z}_n = \arg\max_k p(z_n = k|\hat{\Theta}, y_n) = \arg\max_k \frac{p(z_n = k|\Theta)p(y_n|z_n = k, \Theta)}{\sum_{l=1}^K p(z_n = l|\Theta)p(y_n|z_n = l, \Theta)} \quad (4)$$

   $$= \arg\max_k \pi_k \mathcal{N}(y_n|w_k^T x_n, \beta^{-1}) \quad (5)$$

   Where denominator of 4 is a constant across optimization. Now use this $\hat{z}_n$ as the one-hot encoded vector z_n where $z_{nk} = 1$ if $\hat{z}_n = k$ else $z_{nk} = 0$.

   (b) To compute Θ now, we can directly utilize *MLE* form

   $$\Theta_{MLE} = \arg\max_\Theta log(p(Y, \hat{Z}|\Theta))$$

   $$= \arg\max_\Theta \sum_{n=1}^N log\left(p(y_n, z_n|\Theta)\right)$$

   Now using the fact that z_n is a one-hot encoded vector, i.e. alloting the whole contribution of this sample from a particular $z_n = k$ value

   $$\Theta_{MLE} = \arg\max_\Theta \sum_{n=1}^N \sum_{k=1}^K z_{nk} \left[log\pi_k + log\mathcal{N}(y_n|w_k^T x_n, \beta^{-1})\right] \quad (6)$$

   Consider the constraint $\sum_{k=1}^K \pi_k = 1$, create the lagrangian.

   $$\arg\max_\Theta \sum_{n=1}^N \sum_{k=1}^K z_{nk} \left[log\pi_k + log\mathcal{N}(y_n|w_k^T x_n, \beta^{-1})\right] + \lambda(1 - \sum_{k=1}^K \pi_k) \quad (7)$$

Taking derivative w.r.t. each w_k and π_k and equating to 0, and let $N_k = \sum_{n=1}^{N} z_{nk}$, which is equivalent to number of samples corresponding to $\hat{z}_n = k$, we obtain

$$\pi_k = \frac{N_k}{N} \tag{8}$$

$$w_k = (X_k^T X_k)^{-1} X_k^T y_k \tag{9}$$

where $\{X_k, y_k\}$ is the dataset corresponding to $\hat{z}_n = k$ for all $k \in \{1, \cdots K\}$.

For the case when $\pi_k = \frac{1}{K} \; \forall \; k$, update of z_n reduces to

$$\hat{z}_n = \arg\max_k \; exp\left(\frac{-\beta}{2}(y_n - w_k^T x_n)^2\right)$$

This update implies all gaussians are similar in scale.

The above updates of 5, 8 and 9 make intuitive sense because once we've alloted z_n for every x_n, in sense our dataset gets clustered over the latent variable space, and within each cluster, the weights w_k are of standard linear regression, and priors pi_k reflect the number of samples per cluster. Also deciding z_n per x_n itself takes into consideration the priors as well as the closeness of outputs to estimates per latent variable.

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

Student Name: Anshuman Barnwal
Roll Number: 200259
Date: November 16, 2023

QUESTION

5

Solution to Part 1

1.1

Kernelizing ridge regression yields as shown here

$$w = \Phi(\Phi^T\Phi + \lambda \mathbf{I}_n)^{-1}\mathbf{y} \tag{10}$$

Where Φ is matrix having each data sample along rows being expanded to kernel space. Therefore prediction is done using:

$$y_* = w^T x_* = \mathbf{y}^T(\Phi^T\Phi + \lambda \mathbf{I}_n)^{-1}\Phi^T\Phi(x_*) \tag{11}$$

$$= \mathbf{y}^T(\mathcal{K} + \lambda I_n)^{-1}\mathcal{K}(x_*) \tag{12}$$

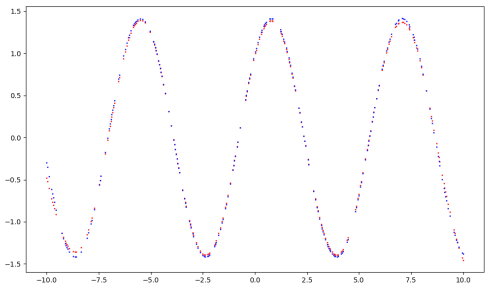
As this problem has x_n being a scalar, $x_i^T x_j = x_i \times x_j$, and

$$\Phi(x_i)^T\Phi(x_j) = k(x_i, x_j) = exp(-\gamma(x_i - x_j)^2)$$

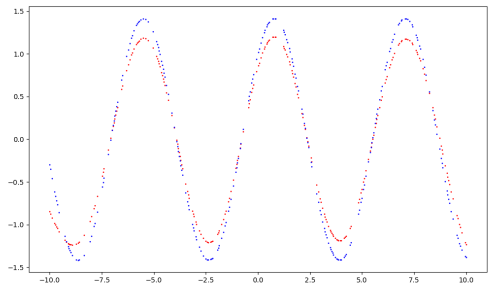
Results

By keeping $\gamma = 0.1$

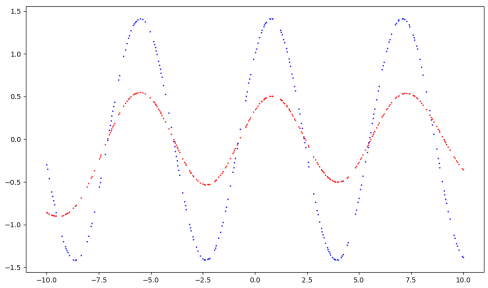
Lambda	RMSE
0.1	0.033
1	0.170
10	0.609
100	0.911



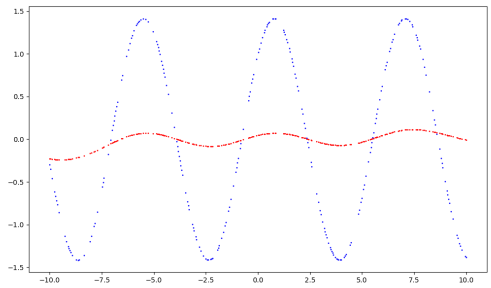
(a) Lambda=0.1



(b) Lambda=1



(c) Lambda=10



(d) Lambda=100

Figure 1: Kernel Ridge Regression figures

1.2

Landmark ridge regression involves recreating the feature vector per sample by:

1. Define a set of L landmarks out of the input dataset by uniform sampling
2. Create new feature vector per input by:

$$\tilde{x}_n = \begin{bmatrix} k(z_1, x_n) \\ k(z_2, x_n) \\ k(z_3, x_n) \\ \vdots \\ k(z_L, x_n) \end{bmatrix}$$

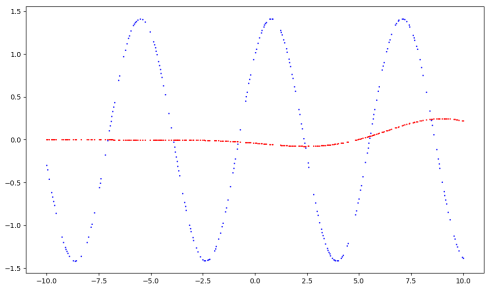
3. Use the new feature vector instead of the original for furthur modelling.

Results

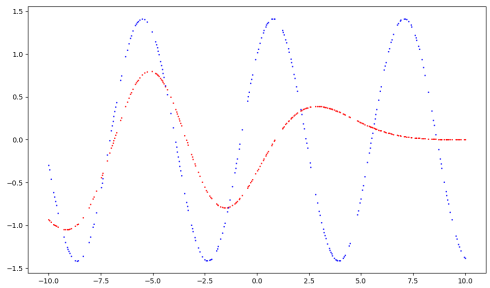
By keeping $gamma = 0.1$ and $\lambda = 0.1$

L	RMSE
2	0.964
5	0.872
20	0.264
50	0.085
100	0.067

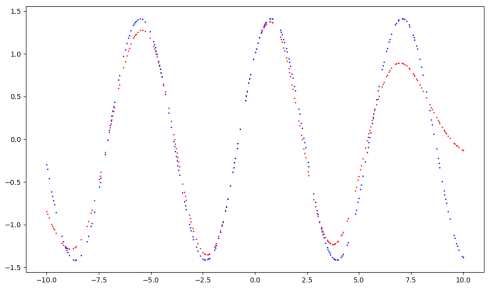
Clearly, change in RMSE from 20 to 50 is much higher than 50 to 100, after which it's nearly the same. So $L = 50$ is good enough number of landmarks.



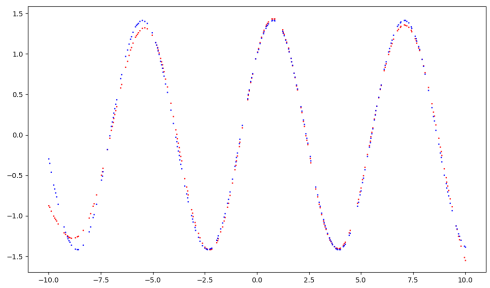
(a) L=2



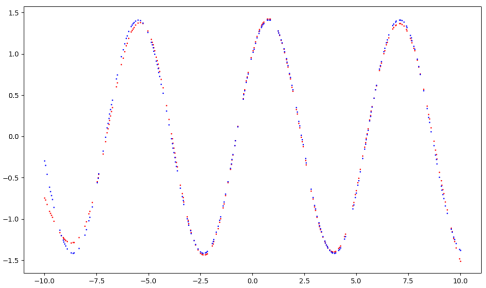
(b) L=5



(c) L=20



(d) L=50



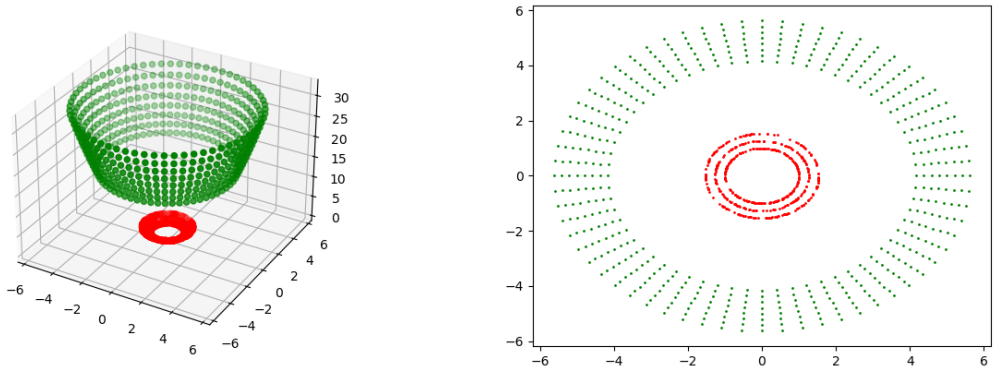
(e) L=100

Figure 2: Landmark Ridge Regression figures

Solution to Part 2

2.1

Upon analyzing the original dataset, a good hand-crafted feature would be the radial distance of data samples from origin. Therefore creating a third feature as $\|x_n\|_2^2$, we obtain the following results:



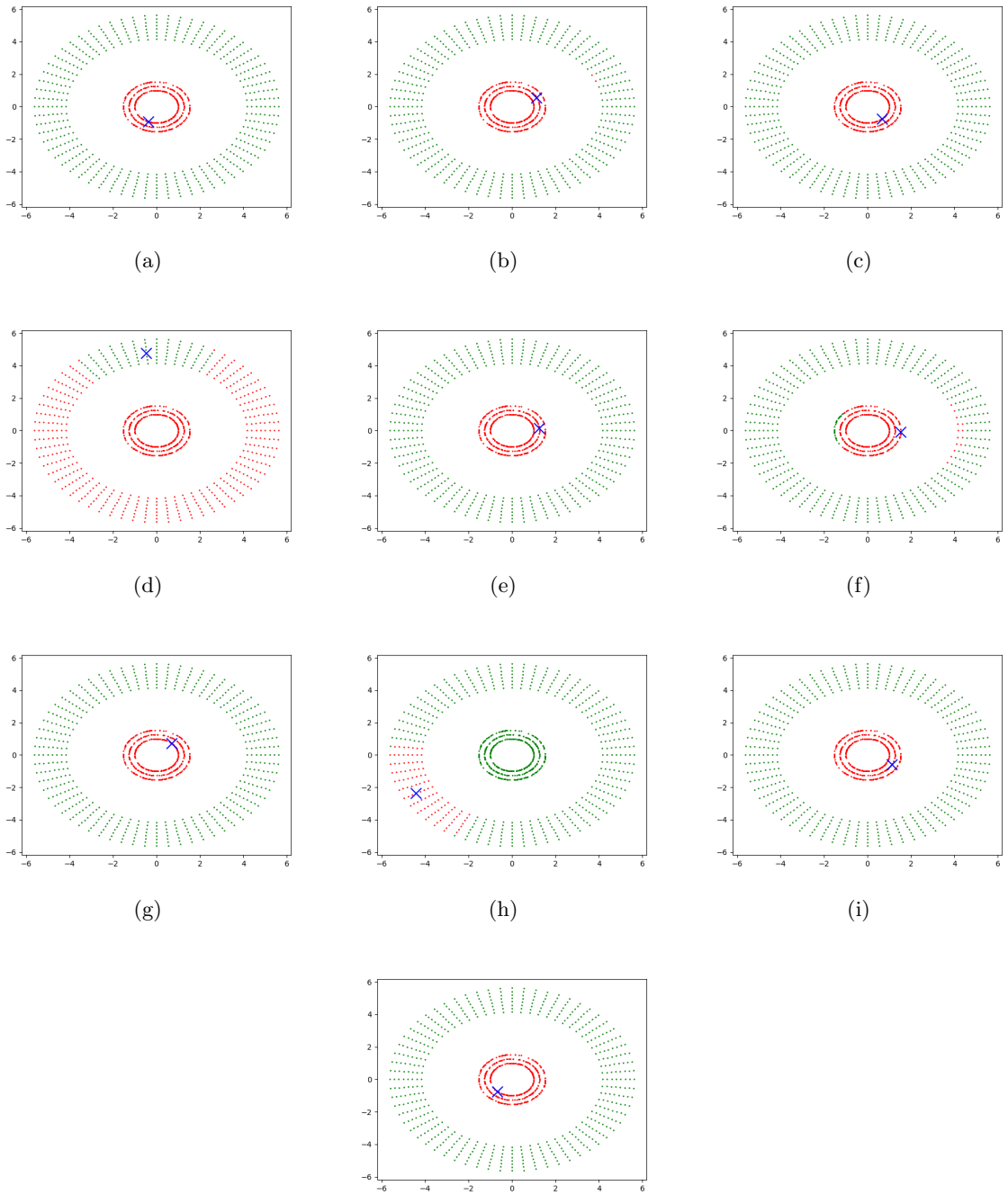
(a) "Kernelized" Space where KMeans is used

(b) Original Space

Figure 3: Hand-crafted features for linear K-Means

2.2

Using one landmark and kernelizing the whole dataset on that basis, using the kernelized features for k-means, we obtain the following:



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j)

Figure 4: Landmark kernelized features for linear K-Means

As observed in the above figures, the landmark decides the new features. Plotting the new feature in the z-axis makes it clear how clustering happens furthur:

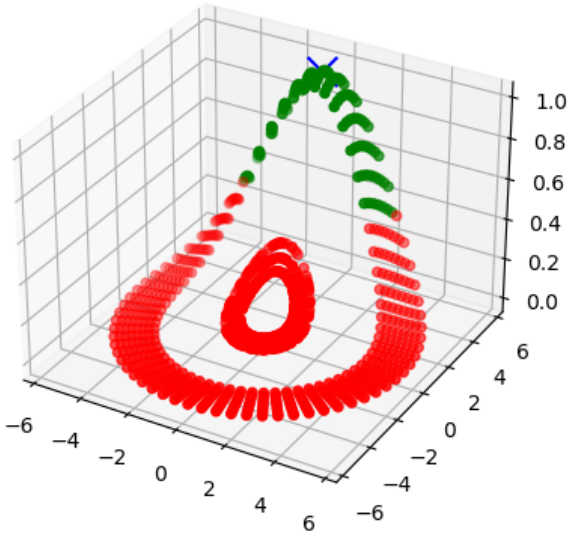


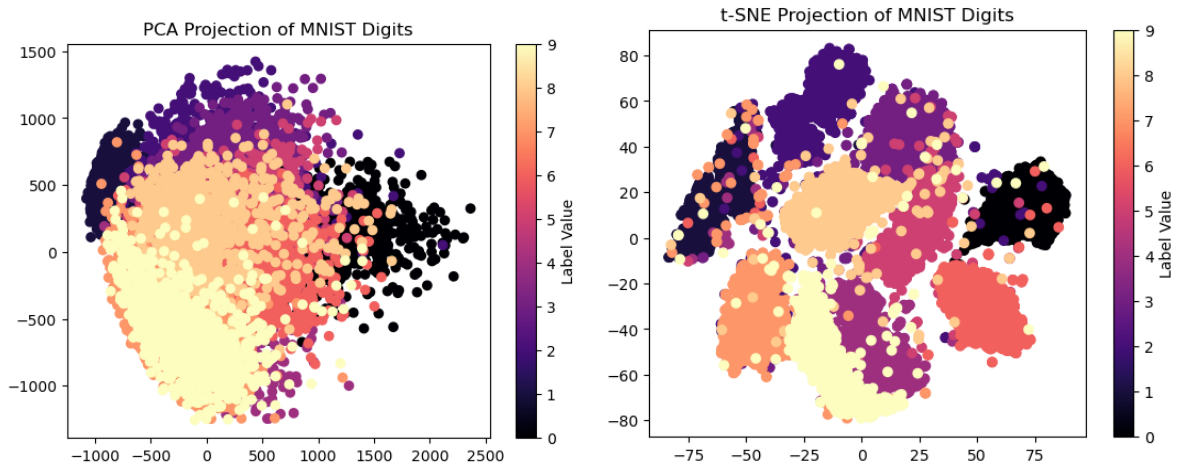
Figure 5: 3D view of 4(d)

Clearly the nearness to landmark leads to a higher value towards 1 in the z-axis, while being furthur leads to a lower value towards 0. Therefore if the landmark is by chance chosen near center of the inner disk, all of that disk being near to the landmark gains a value close to 1, as seen in 4(a).

Solution to Part 3

Visualizing data using `sklearn` is quite straightforward and is treated as a black-box during usage.

The difference observed between PCA and tSNE is that tSNE preserves the structure of clusters formed by digits in the higher dimensions even after the projection, and doesn't just project along the highest variance axes like PCA does, therefore the color clusters seem coherent and not flushed out in tSNE unlike PCA. Thus tSNE is a better manifold projection visulation technique.



(a) PCA (b) tSNE

Figure 6: Projection and Visualization techniques for multi-dimensional inputs