Introduction to ML (CS771), Autumn 2018
Indian Institute of Technology Kanpur
Homework Assignment Number 3

Student Name: Abhishek Kumar
Roll Number: 18111002
Date: October 21, 2018

QUESTION

1

Learning SVM via Co-ordinate Ascent

We have dual formulation of linear SVM problem as:

$$\begin{aligned}
w &= \sum_{n=1}^{N} \alpha_n y_n x_n \\
L(\alpha) &= \alpha^T 1 - \frac{1}{2}\alpha^T G\alpha \\
\frac{\partial L(\alpha)}{\partial \alpha} &= \Delta L(\alpha) = 1 - G\alpha \\
\alpha &= \arg\max_{C>\alpha>0} \alpha^T 1 - \frac{1}{2}\alpha^T G\alpha
\end{aligned}$$

Now we apply coordinate ascent over it and say we will maximize with respect to δ an addition to α_n(i.e $\alpha's$ are constant per iteration) :

$$\begin{aligned}
\delta^* &= \arg\max_{\delta}\{(\alpha + \delta e_n)^T 1 - \frac{1}{2}(\alpha + \delta e_n)^T G(\alpha + \delta e_n)\} \\
&= \arg\max_{\delta}\{\delta(e_n^T(1 - G\alpha)) - \frac{\delta^2}{2}(e_n^T G e_n) + const.\} \\
&= \arg\max_{\delta}\{\delta\Delta_n L(\alpha) - \frac{\delta^2}{2}(G_{nn}) + const.\} \\
&= \frac{\Delta_n L(\alpha)}{G_{nn}}
\end{aligned}$$

where $\Delta_n L(\alpha)$ is the n^{th} component of gradient $\Delta L(\alpha)$, and $G_{nn} = G[n][n]$.

Now we want to update alpha's only when constrained is satisfied and $\Delta_n L(\alpha) \neq 0$. so if constraints are violated then we will project alpha back to domain. So we can define now projected alpha as:

$$\alpha_n^{new} = \min\{\max\{\alpha_n + \frac{\Delta_n L(\alpha)}{G_{nn}}, 0\}, C\} \tag{1.1}$$

Now our weight update equation can be written as:

$$\begin{aligned}
w^{new} &= \sum_i \alpha_i^{new} y_i x_i \\
&= \sum_i \alpha_i y_i x_i + (\alpha_n^{new} - \alpha_n) y_n x_n \qquad (1.2) \\
&= w + (\alpha_n^{new} - \alpha_n) y_n x_n
\end{aligned}$$

Dual Co-ordinate Ascent Algorithm

1. Given α calculate $w = \sum_i \alpha_i y_i x_i$.
2. While not converged or up to max iterations:
   (a) calculate $\Delta_n L(\alpha)$.
       if $(\Delta_n L(\alpha) \neq 0)$:
       - calculate α_n^{new} using equation 1.1
       - calculate w^{new} using equation 1.2
3. Done Stop.

Note that bias term b does not need to calculated separately you can append a array of ones to $X = [1, x]$ matrix and $W = [b, w]$.
This concludes the answer to above question thank you.

Introduction to ML (CS771), Autumn 2018
Indian Institute of Technology Kanpur
Homework Assignment Number 3

Student Name: Abhishek Kumar
Roll Number: 18111002
Date: October 21, 2018

QUESTION
2

Within and Across

We have:

$$\arg\min_f L_w = \arg\min_f \sum_{n,m} \mathbb{I}(f_n = f_m)||X_n - X_m||^2$$

which can be written as:

$$\arg\min_f L_w = \arg\min_f \sum_{n,m} (1 - I(f_n \neq f_m))||X_n - X_m||^2$$

Negation of minimization problem $\implies$ maximization problem

$$= \arg\max_f \sum_{n,m} (I(f_n \neq f_m))||X_n - X_m||^2 - \sum_{n,m} ||X_n - X_m||^2$$

$$= \arg\max_f \sum_{n,m} (I(f_n \neq f_m))||X_n - X_m||^2$$

which shows that it *implicitly* also maximizes the sum of squared distances between all pairs of points that are in different clusters.

This concludes the answer to the given problem.

Introduction to ML (CS771), Autumn 2018
Indian Institute of Technology Kanpur
Homework Assignment Number 3

Student Name: Abhishek Kumar
Roll Number: 18111002
Date: October 21, 2018

QUESTION

3

Estimating A Gaussian when data is missing

Given,

$$x_n = [x_n^m, x_n^o] \sim \mathcal{N}(x_n|\mu, \Sigma)$$

where x_n^m are missing data features and x_n^o are observed features.
We will consider each missing value as a latent variable (i.e $z^n = x_n^m$)
For such a problem we will use a known properties from MLAPP book section 4.3.1 as follows

$$x_n = \begin{bmatrix} x_n^o \\ x_n^m \end{bmatrix}, \mu = \begin{bmatrix} \mu^o \\ \mu^m \end{bmatrix}, \Sigma = \begin{bmatrix} \Sigma^{oo} & \Sigma^{om} \\ \Sigma^{mo} & \Sigma^{mm} \end{bmatrix}, \Lambda = \Sigma^{-1} = \begin{bmatrix} \Lambda^{oo} & \Lambda^{om} \\ \Lambda^{mo} & \Lambda^{mm} \end{bmatrix}$$

and,

$$\begin{aligned} p(x_n^m|x_n^o, \mu, \Sigma) &= \mathcal{N}(x_n^m|\mu^{m|o}, \Sigma^{m|o}) \\ \mu_n^{m|o} &= \mu^m + \Sigma^{mo}\Sigma^{oo-1}(x_n^o - \mu^o) \\ \Sigma_n^{m|o} &= \Sigma^{mm} - \Sigma^{mo}\Sigma^{oo-1}\Sigma^{om} = \Lambda^{mm-1} \end{aligned} \tag{3.1}$$

We can write CLL(Complete-data Log Likelihood) as following:

$$\begin{aligned} \log(p(X^m, X^o|\Theta)) &= \sum_{n=1}^{N} \log(\mathcal{N}(x_n|\mu, \Sigma)) \\ &= \sum_{n=1}^{N} C_0 - \log(|\Sigma| + tr(x_n x_n^T \Sigma^{-1})) - 2x_n^T \Sigma^{-1}\mu + \mu^T \Sigma^{-1}\mu \\ \mathbb{E}\{CLL\} &= \sum_{n=1}^{N} C_0 - \log(|\Sigma| + tr(\mathbb{E}\{x_n x_n^T\}\Sigma^{-1})) - 2\mathbb{E}\{x_n\}^T \Sigma^{-1}\mu + \mu^T \Sigma^{-1}\mu \end{aligned}$$

Now differentiating expected CLL w.r.t Θ we get closed form solution for parameters as follows:

$$\begin{aligned} \mu &= \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}\{x_n\} \\ \Sigma &= \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}\{x_n x_n^T\} - 2\mathbb{E}\{x_n\}\mu^T + \mu\mu^T \\ &= \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}\{(x_n - \mu)(x_n - \mu)^T\} \end{aligned} \tag{3.2}$$

We can calculate expected values from distribution we got in equation 3.1 as follows:

$$\begin{aligned}
\mathbb{E}\{x_n\} &= \begin{bmatrix}\mathbb{E}\{x_n^o\}\\ \mathbb{E}\{x_n^m\}\end{bmatrix} = \begin{bmatrix}x_n^o\\ \mu_n^{m|o}\end{bmatrix}\\
\mathbb{E}\{x_nx_n^T\} &= \begin{bmatrix}\mathbb{E}\{x_n^o x_n^{oT}\} & \mathbb{E}\{x_n^o x_n^{mT}\}\\ \mathbb{E}\{x_n^m x_n^{oT}\} & \mathbb{E}\{x_n^m x_n^{mT}\}\end{bmatrix}\\
&= \begin{bmatrix}x_n^o x_n^{oT} & x_n^o \mu_n^{m|o^T}\\ \mu_n^{m|o} x_n^{oT} & \Sigma_n^{m|o} + \mu_n^{m|o}\mu_n^{m|o^T}\end{bmatrix}
\end{aligned} \tag{3.3}$$

Since the expectation is over latent variables observed features will remain as it is. Now we can write out EM Algorithm formally.

EM Algorithm

1. Initialize $\Theta_0 = \{\mu_0, \Sigma_0\}$
2. **E-step:** Use equations 3.3 and 3.1 to calculate expected values $\mathbb{E}\{x_nx_n^T\}$ and $\mathbb{E}\{x_n\}$ with $(t-1)^{th}$ iteration parameters.

$$\begin{aligned}
\mu_n^{m|o^t} &= \mu^{m^{t-1}} + \Sigma^{mo^{t-1}}\Sigma^{oo-1^{t-1}}(x_n^o - \mu^{o^{t-1}})\\
\Sigma_n^{m|o^t} &= \Sigma^{mm^{t-1}} - \Sigma^{mo^{t-1}}\Sigma^{oo-1^{t-1}}\Sigma^{om^{t-1}}\\
\mathbb{E}\{x_n\}^t &= \begin{bmatrix}x_n^o\\ \mu_n^{m|o^t}\end{bmatrix}\\
\mathbb{E}\{x_nx_n^T\}^t &= \begin{bmatrix}x_n^o x_n^{oT} & x_n^o \mu_n^{m|o^{tT}}\\ \mu_n^{m|o^t} x_n^{oT} & \Sigma_n^{m|o^t} + \mu_n^{m|o^t}\mu_n^{m|o^{tT}}\end{bmatrix}
\end{aligned}$$

3. **M-step:** Use equations 3.2 to calculate μ and Σ with values obtained in E-step.

$$\mu^t = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}\{x_n\}^t$$

$$\Sigma^t = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}\{x_nx_n^T\}^t - 2\mathbb{E}\{x_n\}^t\mu^{t^T} + \mu^t\mu^{t^T}$$

4. Repeat steps 2-4 till convergence.

This concludes the answer for given question.

Introduction to ML (CS771), Autumn 2018
Indian Institute of Technology Kanpur
Homework Assignment Number 3

Student Name: Abhishek Kumar
Roll Number: 18111002
Date: October 21, 2018

QUESTION 4

Semi-supervised Classification

Given,

$$p(y = k) = \pi_k$$
$$p(x_n|y = k) = \mathcal{N}(\mu_k, \Sigma_k)$$

i.e we have a Gaussian mixture model to estimate.
Let's define the two sets as $x^l \epsilon D_l$ labeled set and $x^u \epsilon D_u$ unlabeled set.

Let also define our latent variable which is labels of unlabeled data as Z_u. Now we can write our CLL as follows:

$$\begin{aligned}\log P(X_l, Y_l, X_u, Z_u|\Theta) &= \log(P(X_l, Y_l|\Theta)P(X_u, Z_u|\Theta)) \\ &= \log(P(X_l, Y_l|\Theta)) + \log(P(X_u, Z_u|\Theta))\end{aligned}$$

We have derived both of the terms in class as follows:

$$\log(P(X_l, Y_l|\Theta)) = \sum_{n\epsilon D_l}\sum_{k=1}^{K} y_{nk}[\log(\pi_k) + \log(\mathcal{N}(x_n^l|\mu_k, \Sigma_k))]$$

$$\log(P(X_u, Z_u|\Theta)) = \sum_{n\epsilon D_u}\sum_{k=1}^{K} z_{nk}[\log(\pi_k) + \log(\mathcal{N}(x_n^u|\mu_k, \Sigma_k))]$$

We can combine these two equations to get this:

$$\log P(X_l, Y_l, X_u, Z_u|\Theta) = \sum_{n=1}^{N}\sum_{k=1}^{K} \zeta_{nk}[\log(\pi_k) + \log(\mathcal{N}(x_n|\mu_k, \Sigma_k))]$$

where $\zeta_{nk} = y_{nk}$ if x_n is labeled otherwise $\zeta_{nk} = z_{nk}$.
We can now take expectation over this CLL w.r.t $P_{Z_u} = P(Z_u|X_u, X_l, Y_l, \Theta)$ as:

$$\begin{aligned}\mathbb{E}_{P_{Z_u}}(CLL) &= \sum_{n=1}^{N}\sum_{k=1}^{K} \mathbb{E}_{P_{Z_u}}(\zeta_{nk})[\log(\pi_k) + \log(\mathcal{N}(x_n|\mu_k, \Sigma_k))] \\ &= \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma_{nk}[\log(\pi_k) + \log(\mathcal{N}(x_n|\mu_k, \Sigma_k))]\end{aligned} \tag{4.1}$$

where,

$$\gamma_{nk} = \begin{cases} y_{nk} & x_n \epsilon D_l \\ \mathbb{E}_{P_{z_u}}(z_{nk}) & x_n \epsilon D_u \end{cases} \tag{4.2}$$

Now we need to calculate $\mathbb{E}_{P_{z_u}}(z_{nk})$ in E-step for which we need to find which we already did in class as follows:

$$\mathbb{E}_{P_{z_u}}(z_{nk}^t) = \frac{\pi_k^{t-1}\mathcal{N}(x_n|\mu_k^{t-1}, \Sigma_k^{t-1})}{\sum_{l=1}^{K}\pi_l^{t-1}\mathcal{N}(x_n|\mu_l^{t-1}, \Sigma_l^{t-1})}\ ,\ \forall n, k \tag{4.3}$$

Now in M-step we can estimate other parameters by taking derivative and equating to 0 as follows (derivation have been done in class):

$$\begin{aligned} N_k &= \sum_{n=1}^{N} \gamma_{nk} \\ \pi_k^t &= \frac{N_k}{N} \\ \mu_k^t &= \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk}^t x_n \\ \Sigma_k^t &= \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk}^t (x_n - \mu_k^t)(x_n - \mu_k^t)^T \end{aligned} \tag{4.4}$$

EM Algorithm

1. **Initialize parameters** Θ as:

$$\begin{aligned} N_k &= \sum_{n \epsilon D_l} y_{nk} \\ \pi_k^0 &= \frac{N_k}{N} \\ \mu_k^0 &= \frac{1}{N_k} \sum_{n \epsilon D_l} y_{nk} x_n \\ \Sigma_k^0 &= \frac{1}{N_k} \sum_{n \epsilon D_l} y_{nk} (x_n - \mu_k^0)(x_n - \mu_k^0)^T \end{aligned}$$

2. **E-step**
   Calculate γ_{nk} from equation 4.2 and 4.3

3. **M-step**
   Calculate all global parameters using equation 4.4

4. If not converged go to step 2.

Note : This initialization might work better as compared to random initialization.
This concludes the answer to this question

Introduction to ML (CS771), Autumn 2018
Indian Institute of Technology Kanpur
Homework Assignment Number 3

Student Name: Abhishek Kumar
Roll Number: 18111002
Date: October 21, 2018

QUESTION

5

Latent Variable Models for Supervised Learning

Given,

$$z_n \sim multinoulli(\pi_1, \pi_2, \pi_3, ..., \pi_k)$$
$$y_n \sim \mathcal{N}(w_{z_n}^T x_n, \beta^{-1}), \quad y_n \epsilon \mathbb{R}, \quad x_n \epsilon \mathbb{R}^D$$
$$Z = \{z_1, z_2, .., z_n\}, \Theta = \{w_1, w_2, .., w_k, \pi_1, \pi_2, ..., \pi_k\}$$

1. What is it doing and why to use it

1. It's a set of linear models (Mixture of Experts).
2. It's non-linear model.
3. Non-linear is approximated as combination of linear models. So its better to use it in cases where the non linearity of data can be captured as set of linear models.

2. ALT OPT

Alternating optimization allows us to take z_n as known every iteration.
Then calculating z_n by maximizing the probability $P(y_n, z_n = k|x_n\Theta)$ makes calculation easy as compared to doing MLE on ILL.

Step 1. Initialize Θ somehow.

Step 2. Calculating z_n.
We can now write:

$$\begin{aligned}\hat{z}_n &= \arg\max_k P(z_n = k, y_n|x_n, \Theta) \\ &= \arg\max_k P(z_n = k|x_n, \Theta)P(y_n|z_n = k, x_n, \Theta) \\ &= \arg\max_k [\pi_k \mathcal{N}(y_n|w_{z_n}^T x_n, \beta^{-1})]\end{aligned}$$

And if π_k was $\frac{1}{K}$ then:

$$\hat{z}_n = \arg\max_k \mathcal{N}(y_n|w_{z_n}^T x_n, \beta^{-1})$$

which selects the class which provides the linear regression output closest to actual y. Its like kmeans clustering iterative step with loss function over y instead of x.

Step 3. Calculating the Θ.

Now to calculate global parameters we will do maximization over CLL.

$$
\begin{aligned}
\hat{\Theta} &= \arg\max_{\Theta} P(Z, Y|X, \Theta) \\
&= \arg\max_{\Theta} \prod_{n=1}^{N} \prod_{k=1}^{K} (P(z_n = k|x_n, \Theta) P(y_n|z_n = k, x_n, \Theta))^{z_{nk}} \\
&= \arg\max_{\Theta} \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} (\log(P(z_n = k|x_n, \Theta) P(y_n|z_n = k, x_n, \Theta)) \\
&= \arg\max_{\Theta} \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} (\log(\pi_k) + \log(\mathcal{N}(y_n|w_k^T x_n, \beta^{-1})))
\end{aligned}
$$

Now we can take derivative of this loss combined with Lagrange constraint on π_k as $\sum_k \pi_k = 1$ with respect to π_k and equate to zero to get it as:

$$
\begin{aligned}
N_k &= \sum_{n=1}^{N} z_{nk} \\
\pi_k &= \frac{N_k}{N}
\end{aligned}
$$

Similarly we can calculate w_k by doing same steps as that of π_k, we get:

$$
\begin{aligned}
&\implies \frac{\partial}{\partial w_k} \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} (-\frac{1}{2} (\beta (y_n - w_k^T x_n)^T (y_n - w_k^T x_n))) = 0 \\
&\implies \sum_{n : x_n \epsilon k} (y_n - w_k^T x_n)^T x_n = 0 \\
&\implies w_k = [X_k^T X_k]^{-1} X_k^T Y_k
\end{aligned}
$$

where, $X_k = \{x_n : z_n = k\}$,$Y_k = \{y_n : z_n = k\}$

Step 4. Repeat steps 2 - 4 till convergence

3. EM Algorithm

Similar to alternating optimization we have z_n as latent variable here. We can write CLL as:

$$
\begin{aligned}
CLL &= \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} (\log(\pi_k) + \log(\mathcal{N}(y_n|w_k^T x_n, \beta^{-1}))) \\
\mathbb{E}_{p_z}(CLL) &= \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{E}_{p_z}\{z_{nk}\} (\log(\pi_k) + \log(\mathcal{N}(y_n|w_k^T x_n, \beta^{-1}))) \\
&= \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk} (\log(\pi_k) + \log(\mathcal{N}(y_n|w_k^T x_n, \beta^{-1})))
\end{aligned}
$$

we can calculate expectation of z_{nk} as follows (its similar to what we did for GMM so skipping steps):

$$
\begin{aligned}
p(z_n, y_n|x_n, \Theta) &\propto p(z_n = k|x_n, \Theta) p(y_n|z_n = k, x_n, \Theta) \\
&\propto \pi_k \mathcal{N}(y_n|w_k^T x_n, \beta^{-1}) \\
E_{p_z}\{z_{nk}\}^{t+1} = \gamma_{nk} &= \frac{\pi_k^t \mathcal{N}(y_n|{w_k^t}^T x_n, \beta^{-1})}{\sum_l \pi_l^t \mathcal{N}(y_n|{w_l^t}^T x_n, \beta^{-1})}
\end{aligned}
\tag{5.1}
$$

To estimate the global parameters we can take derivative of CLL and equate it to 0 to get following(its similar to what we did for GMM so skipping steps):

$$\pi_k = \frac{\sum_{n=1}^{N} \gamma_{nk}}{N} = \frac{N_k}{N} \tag{5.2}$$

Similarly for differentiating w.r.t w_k we get:

$$\begin{aligned}
&\implies \sum_{n=1}^{N} \gamma_{nk}\beta(y_n - w_k^T x_n)x_n = 0 \\
&\implies (Y - Xw_k)^T \Gamma_k X = 0 \\
&\implies w_k = [X^T \Gamma_k X]^{-1} X^T \Gamma_k Y
\end{aligned} \tag{5.3}$$

where Γ_k is a diagonal NxN matrix,

$$\Gamma_k = \begin{bmatrix} \gamma_{1k} & 0 & 0 & ... & 0 \\ 0 & \gamma_{2k} & 0 & ... & 0 \\ . & . & . & ... & . \\ 0 & 0 & ... & & \gamma_{nk} \end{bmatrix}$$

EM steps

1. Initialize the global parameters Θ some way.
2. **E-step**: calculate γ_{nk} using equation 5.1 with $(t-1)^{th}$ iteration parameters.
3. **M-step**: calculate π_k and w_k using equation 5.2 and 5.3 with values obtained in E-step.
4. Repeat steps 2-4 till convergence.

When $\beta \to \infty$

In equation 5.1 : if $\beta \to \infty$

$$\lim_{\beta \to \infty} \mathcal{N}(y_n | w_k^T x_n, \beta^{-1}) \to \begin{cases} \infty & |y_n - w_k^T x_n| \to 0 \\ finitely\ small & otherwise \end{cases}$$

so, Gaussian value for all other classes except for one with closest approximation of y will be very less as compared to closest one. (i.e compared to closest approximation class all other classes will have comparably very small values)

$$\lim_{\beta \to \infty} \gamma_{nk} \to z_{nk} = \begin{cases} 1 & |y_n - w_k^T x_n| \to 0 \\ 0 & otherwise \end{cases}$$

Which means z_{nk} will be 1 for one class and 0 for rest. since the expected value of z_{nk} is same as itself then:
E-step and M-step reduces to that of second and third steps in alternating optimization. So whole EM converts to ALT-OPT.

This concludes the answer to this question.

Introduction to ML (CS771), Autumn 2018
Indian Institute of Technology Kanpur
Homework Assignment Number 3

Student Name: Abhishek Kumar
Roll Number: 18111002
Date: October 21, 2018

QUESTION

6

Programming Problem 1

Using RBF Kernel every where... and All plots are on test cases...

1. **Normal Kernel Ridge Regression**
   Plots:

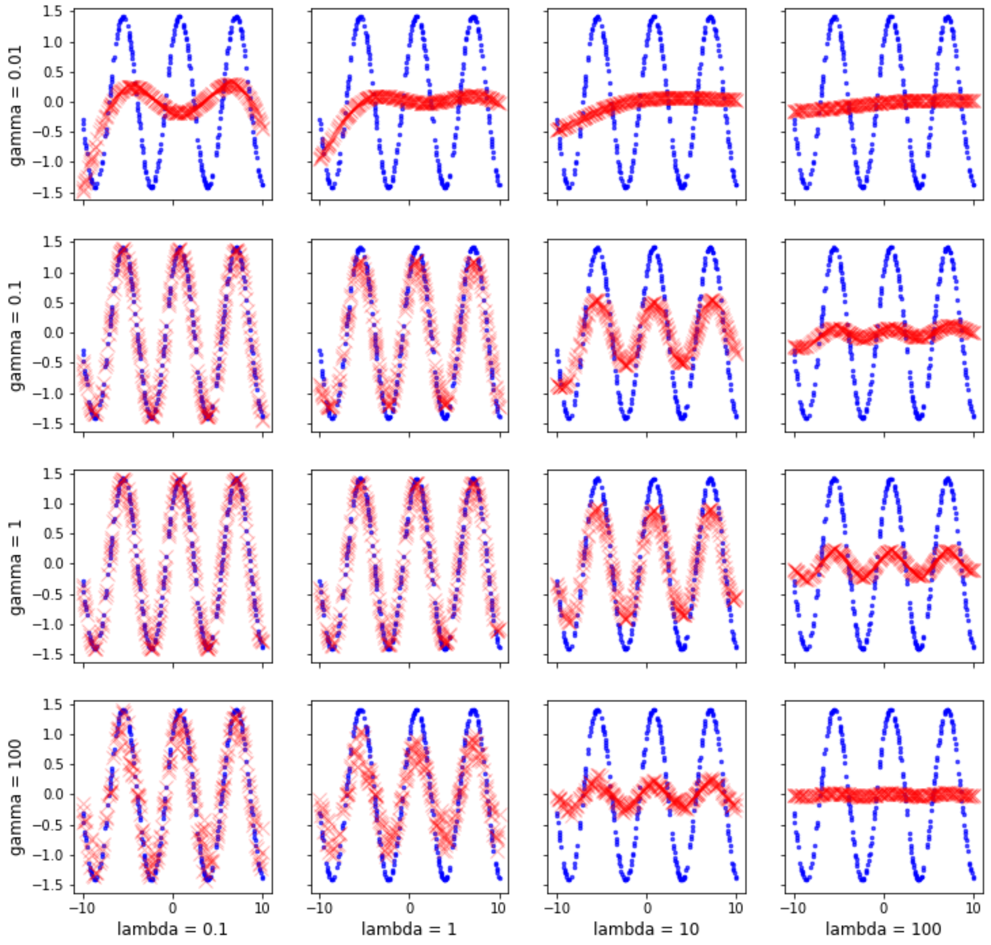

Figure 1: plots for different λ and γ combinations

RMSE Errors:

	$\lambda = 0.1$	$\lambda = 1$	$\lambda = 10$	$\lambda = 100$
$\gamma = 0.01$	14.58078513	14.91538188	15.23692736	15.35359797
$\gamma = 0.1$	0.51509819	2.69274115	9.63335964	14.40553144
$\gamma = 1$	0.1817181	1.01176835	5.82594119	13.16971075
$\gamma = 100$	3.41299967	7.00529538	13.19156093	15.1571786

Table 1: RMSE errors for different λ and γ combinations

Observation : Accuracy decreases with increase in λ due to **under-fitting** and varies non-linearly with γ as the bandwidth increase and decrease changes the distance in transformed Hilbert space and neighbour points may become far or close which changes the neighbour hood consideration of that point.

2. **Landmark Kernel Ridge Regression**
RMSE Errors:

	$L = 1$	$L = 2$	$L = 5$	$L = 20$	$L = 50$	$L = 100$
$\gamma = 0.1$	15.423755	15.356681	12.07004	2.255329	1.862431	1.385910

Table 2: RMSE errors for different λ and γ combinations

plots:

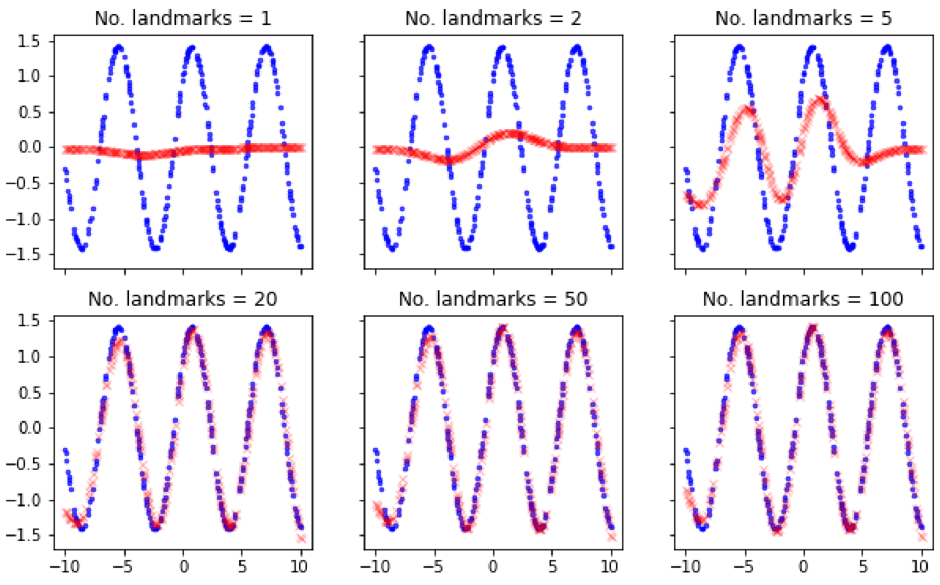

Figure 2: change with number of landmark points

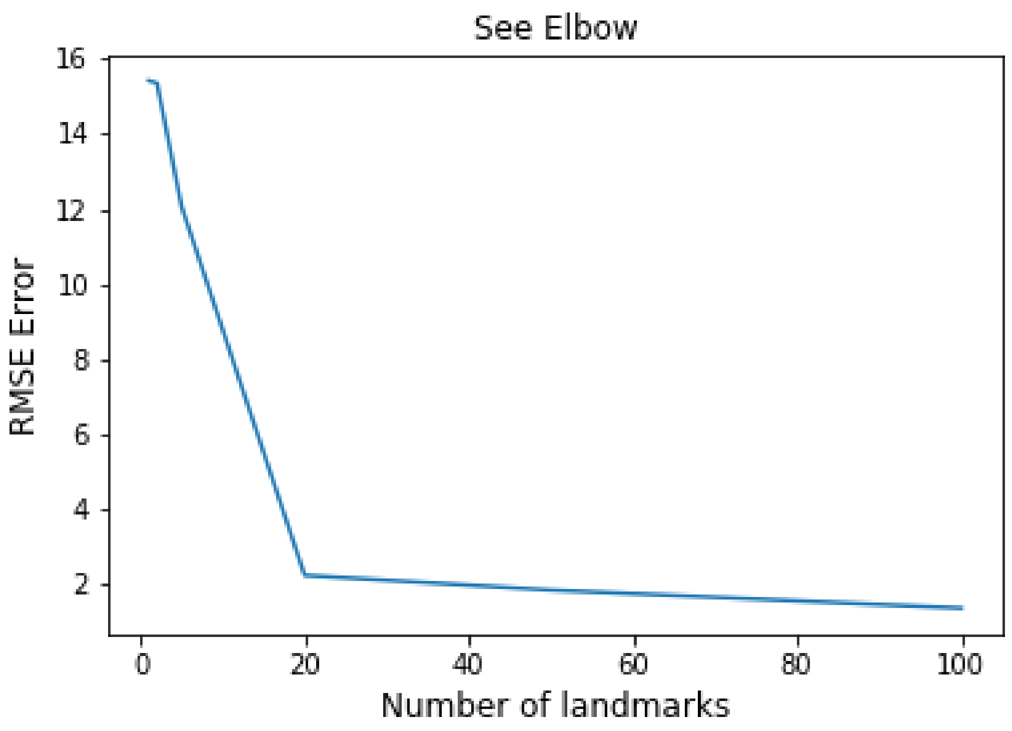

Figure 3: RMSE vs Number of landmark points

Observation : As number of landmark point increases accuracy increases as it gets close to actual kernel matrix. But L = 20 seems the best option as it gives desirable accuracy with least number of landmark points (which is desirable since we don't want to increase our feature dimensions alot).

Programming Problem 2

K-Means Clustering

Visualizing Data

We can see that data is in cicular form with center at $(0, 0)$.

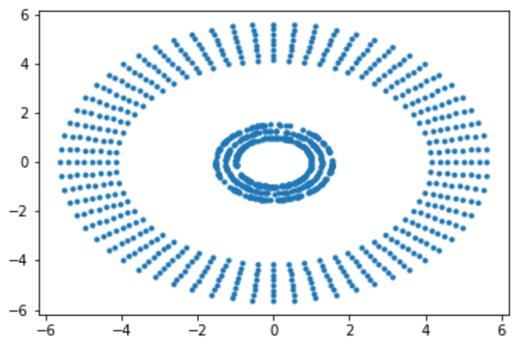

Figure 4: Data provided

What happened with clustering without feature transformation.

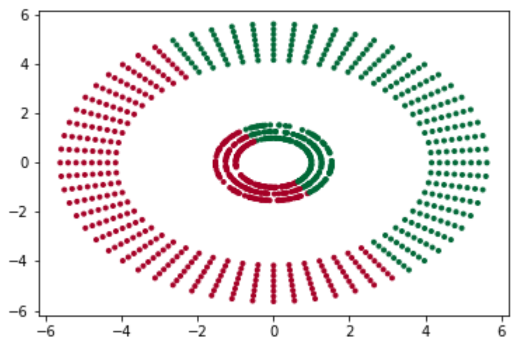

Figure 5: Clustering without transformation

Clearly we cannot use kmeans directly we need to transform features.

Feature Transformations

1. Using Hand-crafted Features

Proposing two transformation of features for such problem:

1. Polar Co-ordinates:
   $r = \sqrt{(x_1^2 + x_2^2)}, \theta = \tan^{-1}(x_2/x_1)$
   $\phi(x) = [r, \theta]$

2. Circle Equation
   $\phi(x) = [x_1^2, x_2^2]$

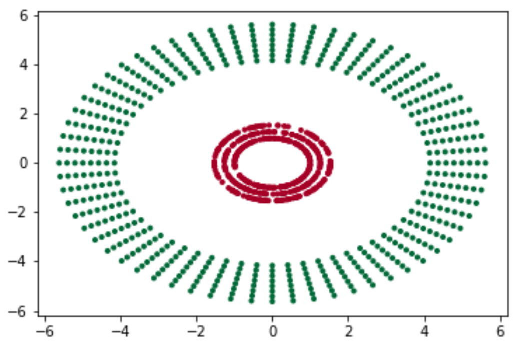
Figure 6: Polar Clustering

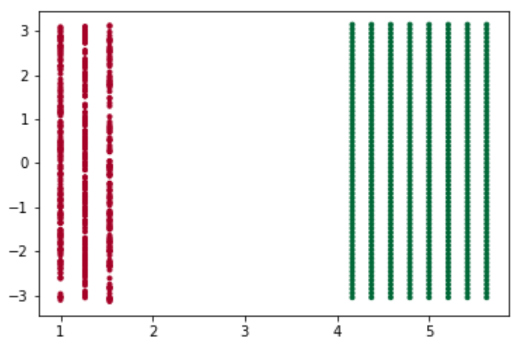
Figure 7: Polar Transformation

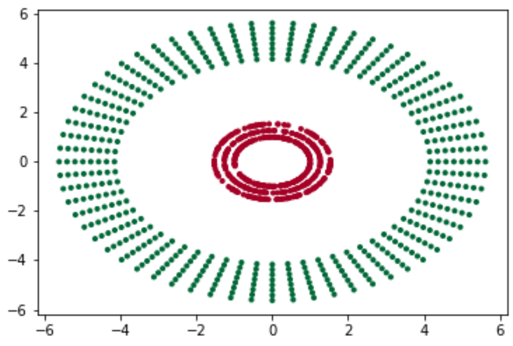
Figure 8: Circle Clustering

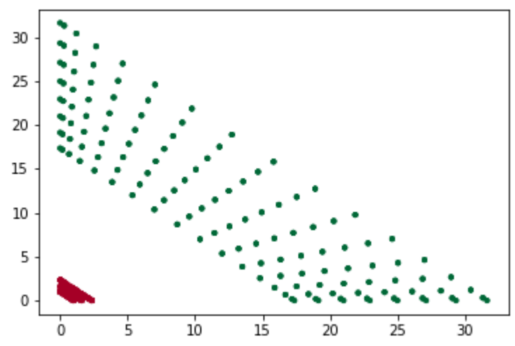
Figure 9: Circle Transformation

We can see why clustering is possible now as the transformed features are separable with good grouping among classes which is desired in kmeans clustering.

But all this required visualizing data but in high dimensions it might not be possible. So we might be able to use kernilization lets see one of such examples below in part 2.

2. Using Landmark Kernels

We will pick just one landmark point with RBF kernel.
A question arises now how will the landmark based kernelized features behave.

We know RBF transforms the input into exponential function of distances between the points. So when only one landkmark is considered the transformed features would be a function of distances from that point. Let's see what we got..

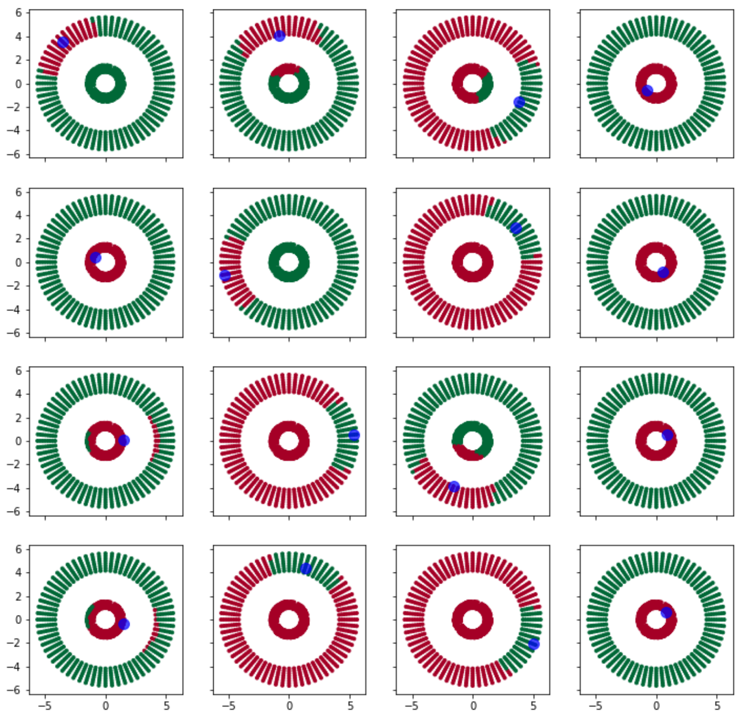

Figure 10: Random Landmark Clustering

As we can see its not always a good clustering. since the data is circular, the landmarks(blue points) that are close to center must separate the data in good clusters since we are using RBF kernel which transforms features into distance from landmarks.In similar way any landmarks away from center are not so good landmark points.

Lets verify our theory by plotting using a landmark at center and a landmark away from center.

1. Landmark at center

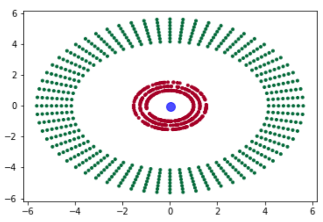

Figure 11: Landmark = Center Clustering

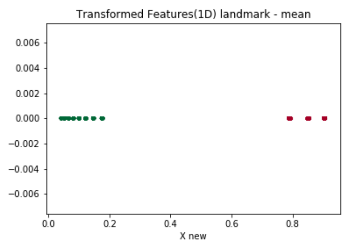

Figure 12: Landmark = Center Transformation

2. Landmark away from center

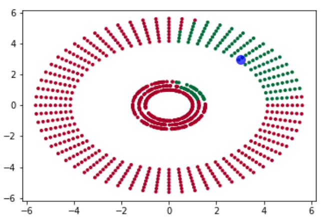

Figure 13: Landmark $\neq$ Center Clustering

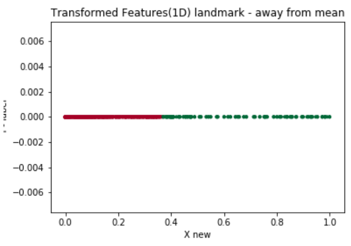

Figure 14: Landmark $\neq$ Center Transformation

This verifies our theory for the given data that because the rbf kernel with landmarks transforms the objects into a Hilbert space where the features are the function of distances from the landmark point center is very good single landmark point for this data.

This concludes the answer for the given problem.