**Introduction to ML (CS771), Autumn 2020**
**Indian Institute of Technology Kanpur**
**Homework Assignment Number 2**

*Student Name:* Musale Krushna Pavan
*Roll Number:* 20111268
*Date:* November 27, 2020

# QUESTION 1

Given logistic regression model with where $y_n$ in $\{0, 1\}$ with no regularizer. Then the loss function is

$$\mathcal{L}(\boldsymbol{\omega}) = -\sum_{n=1}^{N} (y_n \boldsymbol{\omega}^T \mathbf{x_n}) - \log(1 + exp(\boldsymbol{\omega}^T \mathbf{x_n})) \tag{1}$$

second-order optimization based update

$$\boldsymbol{\omega}^{(t+1)} = \boldsymbol{\omega}^{(t)} - (\mathbf{H}^{(t)})^{-1} \mathbf{g}^{(t)} \tag{2}$$

Where $\mathbf{H}$ is hessian and $\mathbf{g}$ denotes the gradient

Need to show that the second-order optimization is equal to

$$\boldsymbol{\omega}^{(t+1)} = \arg\min_{\boldsymbol{\omega}} \sum_{n=1}^{N} \gamma_n^{(t)} (\hat{y}_n^{(t)} - \boldsymbol{\omega}^T \mathbf{x_n})^2$$

$$\implies \sum_{n=1}^{N} 2\gamma_n^{(t)} (\hat{y}_n^{(t)} - \boldsymbol{\omega}^T \mathbf{x_n}) \mathbf{x_n} = 0$$

$$\sum_{n=1}^{N} \gamma_n^{(t)} (\hat{y}_n^{(t)} \mathbf{x_n} = \sum_{n=1}^{N} \gamma_n^{(t)} \mathbf{x_n} \mathbf{x_n^T} \omega$$

$$\boldsymbol{\omega}^{(t+1)} = (\sum_{n=1}^{N} \gamma_n^{(t)} \mathbf{x_n} \mathbf{x_n^T})^{-1} \sum_{n=1}^{N} \gamma_n^{(t)} \hat{y}_n^{(t)} \mathbf{x_n}$$

Now considering the the equation (1)

$$\mathbf{g} = \frac{\partial \mathcal{L}(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}} = -\sum_{n=1}^{N} (y_n \mathbf{x_n}) - \frac{\exp(\boldsymbol{\omega}^T \mathbf{x_n})}{1 + \exp(\boldsymbol{\omega}^T \mathbf{x_n})} \mathbf{x_n}$$

$$= -\sum_{n=1}^{N} (y_n \mathbf{x_n}) - \sigma(\boldsymbol{\omega}^T \mathbf{x_n}) \mathbf{x_n} \qquad \text{where } \sigma(\mathbf{x}) = \frac{1}{1 + \exp(-x)}$$

$$\mathbf{H} = \frac{\partial \mathcal{L}^2(\boldsymbol{\omega})}{\partial^2 \boldsymbol{\omega}} = \frac{\partial \mathbf{g}(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}} = \frac{\partial}{\partial \boldsymbol{\omega}} [ -\sum_{n=1}^{N} (y_n \mathbf{x_n}) - \sigma(\boldsymbol{\omega}^T \mathbf{x_n}) \mathbf{x_n} ]$$

$$= \sum_{n=1}^{N} \sigma(\boldsymbol{\omega}^T \mathbf{x_n}) \sigma(1 - \boldsymbol{\omega}^T \mathbf{x_n}) \mathbf{x_n} \mathbf{x_n^T}$$

Now substituting $\mathbf{g}$ and $\mathbf{H}$ in equation (2)

$$\begin{aligned}
\boldsymbol{\omega}^{(t+1)} &= \omega^{(t)} + ( \sum_{n=1}^{N} \sigma(\boldsymbol{\omega}^{(t)^T}\mathbf{x_n})\sigma(1-\boldsymbol{\omega}^{(t)^T}\mathbf{x_n})\ \mathbf{x_n}\mathbf{x_n^T}\ )^{-1} \sum_{n=1}^{N} (y_n\mathbf{x_n}) - \sigma(\boldsymbol{\omega}^{(t)^T}\mathbf{x_n})\mathbf{x_n} \\
&= ( \sum_{n=1}^{N} \sigma(\boldsymbol{\omega}^{(t)^T}\mathbf{x_n})\sigma(1-\boldsymbol{\omega}^{(t)^T}\mathbf{x_n})\ \mathbf{x_n}\mathbf{x_n^T}\ )^{-1} \\
&\quad [\ (\sum_{n=1}^{N} \sigma(\boldsymbol{\omega}^{(t)^T}\mathbf{x_n})\sigma(1-\boldsymbol{\omega}^{(t)^T}\mathbf{x_n})\ \mathbf{x_n}\mathbf{x_n^T})\boldsymbol{\omega}^{(t)} + \sum_{n=1}^{N} (y_n\mathbf{x_n}) - \sigma(\boldsymbol{\omega}^{(t)^T}\mathbf{x_n})\mathbf{x_n}\ ] \\
&= ( \sum_{n=1}^{N} \sigma(\boldsymbol{\omega}^{(t)^T}\mathbf{x_n})\sigma(1-\boldsymbol{\omega}^{(t)^T}\mathbf{x_n})\ \mathbf{x_n}\mathbf{x_n^T}\ )^{-1} \\
&\quad [\ \sum_{n=1}^{N} [\sigma(\boldsymbol{\omega}^{(t)^T}\mathbf{x_n})\sigma(1-\boldsymbol{\omega}^{(t)^T}\mathbf{x_n})]\ [\mathbf{x_n^T}\boldsymbol{\omega}^{(t)} + \ \frac{y_n - \sigma(\boldsymbol{\omega}^{(t)^T}\mathbf{x_n})}{\sigma(\boldsymbol{\omega}^{(t)^T}\mathbf{x_n})\sigma(1-\boldsymbol{\omega}^{(t)^T}\mathbf{x_n})}\ ]\mathbf{x_n}\ ]
\end{aligned}$$

Let

$$\gamma_n^{(t)} = \sigma(\boldsymbol{\omega}^{(t)^T}\mathbf{x_n})\sigma(1-\boldsymbol{\omega}^{(t)^T}\mathbf{x_n})$$

$$\hat{y}_n^{(t)} = \mathbf{x_n^T}\boldsymbol{\omega}^{(t)} + \ \frac{y_n - \sigma(\boldsymbol{\omega}^{(t)^T}\mathbf{x_n})}{\sigma(\boldsymbol{\omega}^{(t)^T}\mathbf{x_n})\sigma(1-\boldsymbol{\omega}^{(t)^T}\mathbf{x_n})}$$

Then the

$$\boldsymbol{\omega}^{(t+1)} = ( \sum_{n=1}^{N} \gamma_n^{(t)}\mathbf{x_n}\mathbf{x_n^T}\ )^{-1} \sum_{n=1}^{N} \gamma_n^{(t)}\hat{y}_n^{(t)}\mathbf{x_n}$$

from the equation of $\gamma_n$ we can make observe that the points near the boundary have more weight than the points far away from the boundary.

**Introduction to ML (CS771), Autumn 2020**
**Indian Institute of Technology Kanpur**
**Homework Assignment Number 2**

*Student Name:* Musale Krushna Pavan
*Roll Number:* 20111268
*Date:* November 27, 2020

**QUESTION**

# 2

My solution to problem 2
Suppose that there is a transformation function $\phi$ that convert $x$ to its huber space.
And let there is kernal function $k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$

We know that the weight vector in the perceptron is in the form $\omega = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x_n}$

Now kernalized algorithms for SGD

1. Initialize $w = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x_n}$ and $\alpha_i = 0 \quad \forall i$ and set $\eta_t = 1$ – Initialization
2. Pick a point $(\mathbf{x_i}, y_i)$ randomly
3. $$\hat{y} = sign(\ (\sum_{n=1}^{N} \alpha_n y_n \phi(\mathbf{x_n}))^T \phi(\mathbf{x_i})\ )$$
$$\hat{y} = sign(\ \sum_{n=1}^{N} \alpha_n y_n \phi(\mathbf{x_n})^T \phi(\mathbf{x_i})\ )$$
$$\hat{y} = sign(\ \sum_{n=1}^{N} \alpha_n y_n\ k(\mathbf{x_n}, \mathbf{x_i})\ ) \quad \text{– mistake finding}$$
4. If $\hat{y} \neq y_i$ then $\alpha_i += 1$ – update if mistake
5. If not converged go to step 2.

**Introduction to ML (CS771), Autumn 2020**
**Indian Institute of Technology Kanpur**
**Homework Assignment Number 2**

*Student Name:* Musale Krushna Pavan
*Roll Number:* 20111268
*Date:* November 27, 2020

# QUESTION 3

My solution to problem 3
As sometimes its costs more to wrongly classify as negative than positive we can incorporate this information in the SVM equation as follows:

$$\mathcal{L}(\boldsymbol{\omega}, b, \boldsymbol{\xi}) = \min_{\boldsymbol{\omega}, b, \boldsymbol{\xi}} \frac{\|\boldsymbol{\omega}\|^2}{2} + \sum_{n=1}^{N} \mathcal{C}_{y_n} \xi_n$$

$$\text{such that } y_n(\boldsymbol{\omega}^T \mathbf{x_n} + b) \geq 1 - \xi_n \text{ and } \xi_n \geq 0, \forall n$$

Using legrangian method to solve the constraint optimization problem. whose dual can be formed as following

$$\mathcal{L}(\boldsymbol{\omega}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \max_{\boldsymbol{\alpha} \geq 0, \boldsymbol{\beta} \geq 0} \min_{\boldsymbol{\omega}, b, \boldsymbol{\xi}} \frac{\|\boldsymbol{\omega}\|^2}{2} + \sum_{n=1}^{N} \mathcal{C}_{y_n} \xi_n + \sum_{n=1}^{N} \alpha_n (1 - \xi_n - y_n(\boldsymbol{\omega}^T \mathbf{x_n} + b)) - \sum_{n=1}^{N} \beta_n \xi_n \tag{3}$$

$$= \max_{\boldsymbol{\alpha} \geq 0, \boldsymbol{\beta} \geq 0} \min_{\boldsymbol{\omega}, b, \boldsymbol{\xi}} \frac{\boldsymbol{\omega}^T \boldsymbol{\omega}}{2} + \sum_{n=1}^{N} \mathcal{C}_{y_n} \xi_n + \sum_{n=1}^{N} \alpha_n (1 - \xi_n - y_n(\boldsymbol{\omega}^T \mathbf{x_n} + b)) - \sum_{n=1}^{N} \beta_n \xi_n \tag{4}$$

Now for solving for $\boldsymbol{\omega}, b, \boldsymbol{\xi}$ we partially derivate $\mathcal{L}(\boldsymbol{\omega}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ with each of the $\boldsymbol{\omega}, b, \boldsymbol{\xi}$ and equating to 0
First derivating $\mathcal{L}$ with $\boldsymbol{\omega}$ and equating to 0

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\omega}} = 0 \tag{5}$$

$$\boldsymbol{\omega} - \sum_{n=1}^{N} \alpha_n y_n \mathbf{x_n} = 0 \tag{6}$$

$$w = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x_n} \tag{7}$$

Now derivating $\mathcal{L}$ with $b$ and equating to 0

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \tag{8}$$

$$\sum_{n=1}^{N} \alpha_n y_n = 0 \tag{9}$$

Now derivating $\mathcal{L}$ with $\xi_n$ and equating to 0, which is similar $\forall n$

$$\frac{\partial \mathcal{L}}{\partial \xi_n} = 0 \tag{10}$$

$$\mathcal{C}_{y_n} - \alpha_n - \beta_n = 0 \tag{11}$$

$$\mathcal{C}_{y_n} \geq \alpha_n \qquad \text{using } \beta_n \geq 0 \tag{12}$$

By substituting $w = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x_n}$ in the Lagrangian, we get dual as

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\beta}) &= \max_{\boldsymbol{\alpha} \geq 0, \boldsymbol{\beta} \geq 0} \ \frac{\boldsymbol{\omega}^T \boldsymbol{\omega}}{2} + \sum_{n=1}^{N} \mathcal{C}_{y_n} \xi_n + \sum_{n=1}^{N} \alpha_n (1 - \xi_n - y_n(\boldsymbol{\omega}^T \mathbf{x_n} + b)) - \sum_{n=1}^{N} \beta_n \xi_n \\
&= \max_{\boldsymbol{\alpha} \geq 0, \boldsymbol{\beta} \geq 0} \ \frac{1}{2} (\sum_{n=1}^{N} \alpha_n y_n \mathbf{x_n})^T (\sum_{n=1}^{N} \alpha_n y_n \mathbf{x_n}) + \sum_{n=1}^{N} \mathcal{L}_{y_n} \xi_n + \sum_{n=1}^{N} \alpha_n - \sum_{n=1}^{N} \alpha_n \xi_n \\
&- \sum_{n=1}^{N} \alpha_n y_n (\sum_{m=1}^{N} \alpha_m y_m \mathbf{x_m})^T x_n + \sum_{n=1}^{N} \alpha_n y_n b - \sum_{n=1}^{N} \beta_n \xi_n
\end{aligned}$$

using $\mathcal{C}_{y_n} - \alpha_n - \beta_n = 0$ and $\sum_{n=1}^{N} \beta \geq = 0$ we get

$$\mathcal{L}(\boldsymbol{\alpha}) = \max_{\alpha_n \leq \mathcal{C}_{y_n}} \ \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n,m=1}^{N} \alpha_n \alpha_m y_n y_m \mathbf{x_m}^T \mathbf{x_n} \tag{13}$$

When we compare the above equation with the standard SVM equation there is only difference in the condition of $\alpha_n \leq \mathcal{C}_{y_n}$. This specifies that the 2 supporting vectors for the need not be same distance form the decision boundary can used to give more weight on negatively classify on appropriate setting of $\mathcal{C}_{y_n}$.

**Introduction to ML (CS771), Autumn 2020**
**Indian Institute of Technology Kanpur**
**Homework Assignment Number 2**

*Student Name:* Musale Krushna Pavan
*Roll Number:* 20111268
*Date:* November 27, 2020

# QUESTION 4

Now making the above batch k-means to online.
Consider the following initialization:
let the data points be $\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, ..., \mathbf{x_n}$
There be $K$ clusters where $K \in 1, 2, ...K$
Means of the cluster $k$ is randomly initialized be $\boldsymbol{\mu}_k$.
Number of nodes in cluster $k$ be $\eta_k = 0 \; \forall k \in \{1, 2, 3..k\}$
Cosider a point random point $\mathbf{x_n}$
(1) Assigning $\mathbf{x_n}$ "greedily" to the "best" cluster by solving the following function.

$$\mathbf{z_n} = \underset{k \in 1,2,..K}{\arg\min} \|\mathbf{x_n} - \boldsymbol{\mu_k}\|^2$$

$\mathbf{z_n}$ is the one hot encoding whose values is 0 or 1. 1 is at the index where the point is assigned.
The loss is given by

$$\begin{aligned} \ell(\boldsymbol{\mu}, \mathbf{x_n}, \mathbf{z_n}) &= \sum_{k=1}^{K} z_{nk} \|\mathbf{x_n} - \boldsymbol{\mu_k}\|^2 \\ &= \|\mathbf{x_n} - \boldsymbol{\mu_k}_n\|^2 \qquad \text{when assigned to } k^{\text{th}}\text{cluster.} \end{aligned}$$

Now the gradient $\mathbf{g}$ of the loss function is given by:

$$\begin{aligned} \mathbf{g} &= 2(\boldsymbol{\mu_k}_n - \mathbf{x_n}) \\ \mathbf{g^{(t)}} &= 2(\boldsymbol{\mu_k^{(t)}}_n - \mathbf{x_n}) \end{aligned}$$

Update rule according to stocastic gradient decent is :

$$\begin{aligned} \boldsymbol{\mu_k^{(t+1)}}_n &= \boldsymbol{\mu_k^{(t)}}_n - n_t \mathbf{g^{(t)}} \qquad n_t \text{ is learning rate.} \\ &= \boldsymbol{\mu_k}_n^{(t)} - n_t 2(\boldsymbol{\mu_k^{(t)}}_n - \mathbf{x_n}) \\ &= (1 - 2n_t)\boldsymbol{\mu_k}_n^{(t)} + (2n_t)\mathbf{x_n} \qquad ---(1) \\ \eta_k &= \eta_k + 1 \end{aligned}$$

Suggesting a good choice of the step size.
In general when we assign a new point$x_n$ to a cluster the mean $\boldsymbol{\mu_n}_k$ and if the cluster already has $\eta_k$ points in it then the updated mean will be

$$\boldsymbol{\mu_k}'_n = \frac{\boldsymbol{\mu_k}_n \eta_k + \mathbf{x_n}}{\eta_k + 1} \qquad ---(2)$$

By comparing eqn (1) and (2) we get the learning rate

$$n_t = \frac{1}{2(\eta_k + 1)}$$

The above learning rate also gives the intuition that when there less points in the cluster it has larger steps to update the mean and similarly when there are more points in the cluster it has very small steps to update the mean, which should be the general case of updating.

**Introduction to ML (CS771), Autumn 2020**
**Indian Institute of Technology Kanpur**
**Homework Assignment Number 2**

*Student Name:* Musale Krushna Pavan
*Roll Number:* 20111268
*Date:* November 27, 2020

# QUESTION 5

let the data points be $\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, ..., \mathbf{x_n}$ There be $K$ clusters where $K \in 1, 2, ...K$ let $\boldsymbol{\phi}$ be a mapping function that maps to hilbert space. and kernal $\mathbf{k}(\mathbf{x_n}, \mathbf{x_m}) = \boldsymbol{\phi}(\mathbf{x_n})^T \boldsymbol{\phi}(\mathbf{x_m})$
Then the cluster mean in hilbert space is given by $\boldsymbol{\phi}(\boldsymbol{\mu}_k) = \frac{1}{|C_k|} \sum_{n:z_n=k} \boldsymbol{\phi}(\mathbf{x_n})$
Algorithm:
(1) Let randomly assign each point to one of the cluster

$$\mathbf{z_n} = random(k) \qquad \forall n \; k \in 1, 2, ...K$$

(2) Update the each point to its nearest cluster in the hilbert space
Consider:

$$\begin{aligned}
&\|\boldsymbol{\phi}(\mathbf{x_n}) - \boldsymbol{\phi}(\boldsymbol{\mu}_k)\|^2 \\
&\qquad = (\boldsymbol{\phi}(\mathbf{x_n}) - \boldsymbol{\phi}(\boldsymbol{\mu}_k))^T (\boldsymbol{\phi}(\mathbf{x_n}) - \boldsymbol{\phi}(\boldsymbol{\mu}_k)) \\
&\qquad = (\boldsymbol{\phi}(\mathbf{x_n})^T - \boldsymbol{\phi}(\boldsymbol{\mu}_k)^T)(\boldsymbol{\phi}(\mathbf{x_n}) - \boldsymbol{\phi}(\boldsymbol{\mu}_k)) \\
&\qquad = \boldsymbol{\phi}(\mathbf{x_n})^T \boldsymbol{\phi}(\mathbf{x_n}) - 2\boldsymbol{\phi}(\mathbf{x_n})^T \boldsymbol{\phi}(\boldsymbol{\mu}_k) - \boldsymbol{\phi}(\boldsymbol{\mu}_k)^T \boldsymbol{\phi}(\boldsymbol{\mu}_k) \\
&\qquad = \boldsymbol{\phi}(\mathbf{x_n})^T \boldsymbol{\phi}(\mathbf{x_n}) - 2\boldsymbol{\phi}(\mathbf{x_n})^T (\frac{1}{|C_k|} \sum_{m:z_m=k} \boldsymbol{\phi}(\mathbf{x_m})) \\
&\qquad\qquad - (\frac{1}{|C_k|} \sum_{m:z_m=k} \boldsymbol{\phi}(\mathbf{x_m}))^T (\frac{1}{|C_k|} \sum_{m:z_m=k} \boldsymbol{\phi}(\mathbf{x_m})) \\
&\qquad = \boldsymbol{\phi}(\mathbf{x_n})^T \boldsymbol{\phi}(\mathbf{x_n}) - 2\frac{1}{|C_k|} \sum_{m:z_m=k} \boldsymbol{\phi}(\mathbf{x_n})^T \boldsymbol{\phi}(\mathbf{x_m}) - \frac{1}{|C_k|^2} \sum_{\substack{i,j:z_i=k \\ z_j=k}} \boldsymbol{\phi}(\mathbf{x_i})^T \boldsymbol{\phi}(\mathbf{x_j}) \\
&\qquad = k(\mathbf{x_n}, \mathbf{x_n}) - \frac{2}{|C_k|} \sum_{m:z_m=k} k(\mathbf{x_n}, \mathbf{x_m}) + \frac{1}{|C_k|^2} \sum_{\substack{i,j:z_i=k \\ z_j=k}} k(\mathbf{x_i}, \mathbf{x_j})
\end{aligned}$$

Now updating each point to its nearest cluster as

$$\mathbf{z_n} = \underset{k \in 1,2,..K}{\arg\min} \; k(\mathbf{x_n}, \mathbf{x_n}) - \frac{2}{|C_k|} \sum_{m:z_m=k} k(\mathbf{x_n}, \mathbf{x_m}) + \frac{1}{|C_k|^2} \sum_{\substack{i,j:z_i=k \\ z_j=k}} k(\mathbf{x_i}, \mathbf{x_j})$$

We do not compute as mean as its is infinite dimentional(kernal)
(3) If not converge i.e the cluster assignments does not change then go to step 2.

Instead of storing the cluster mean as in the case of normal k-means, in kernal-K means we store which point belongs to which cluster.

Finally, assuming each input to be D-dimensional in the original feature space, and N to be the number of inputs. The cost of input to clusters mean distance calculation
(1) standard K-means just $\|\mathbf{x_n} - \boldsymbol{\mu}_k|^2 \;\; \forall k$ is $O(KD)$
(2) Kernal k-means it is $O(D + DN + D(N - K + 1)(N - K)) = O(DN^2 + DNK + DK^2) = O(DN^2)$ when $N >> K$ from the update equation we got.

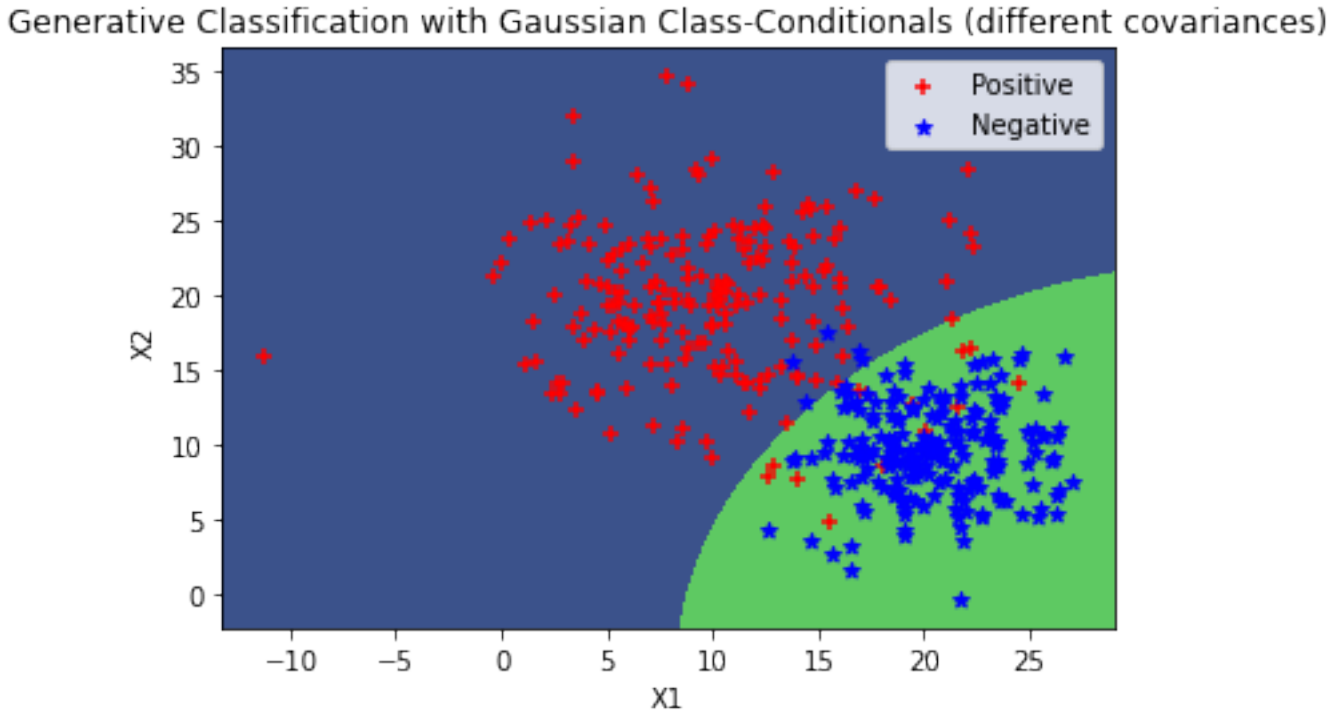
**Introduction to ML (CS771), Autumn 2020**
**Indian Institute of Technology Kanpur**
**Homework Assignment Number 2**

*Student Name:* Musale Krushna Pavan
*Roll Number:* 20111268
*Date:* November 27, 2020

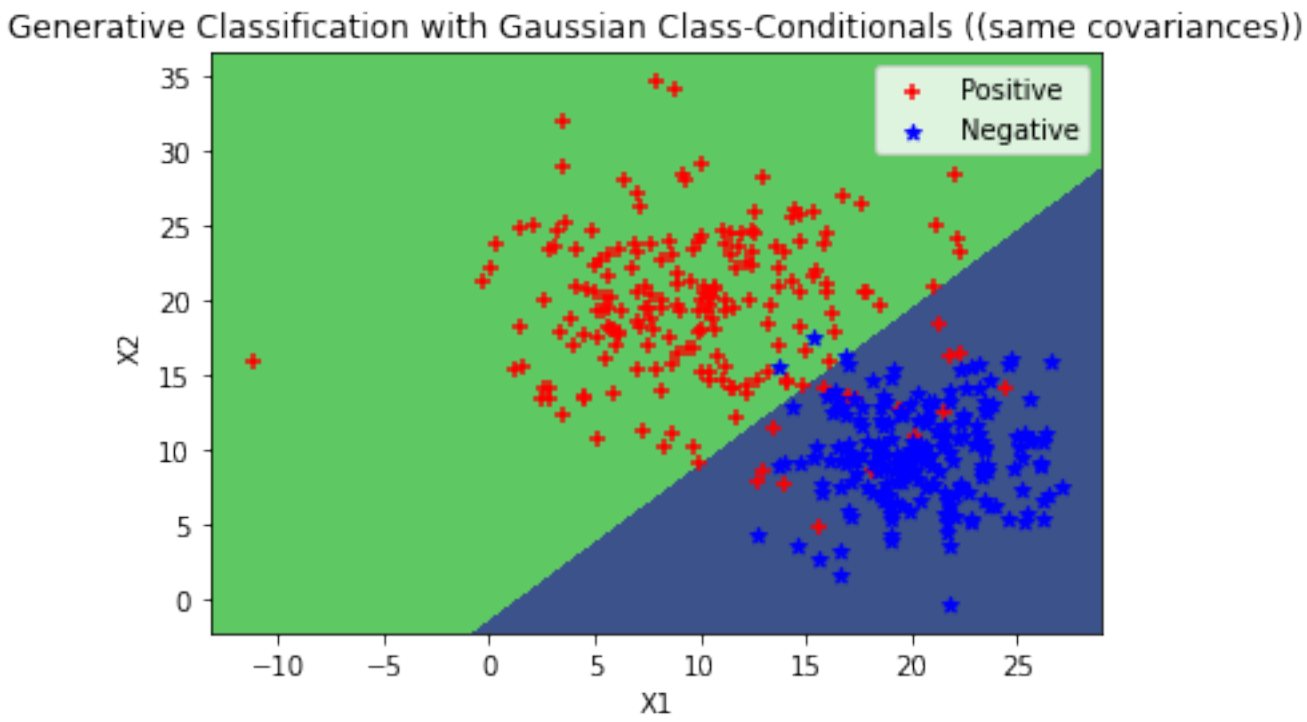
**QUESTION**
# 6

## 1 Part1 - binclass.txt

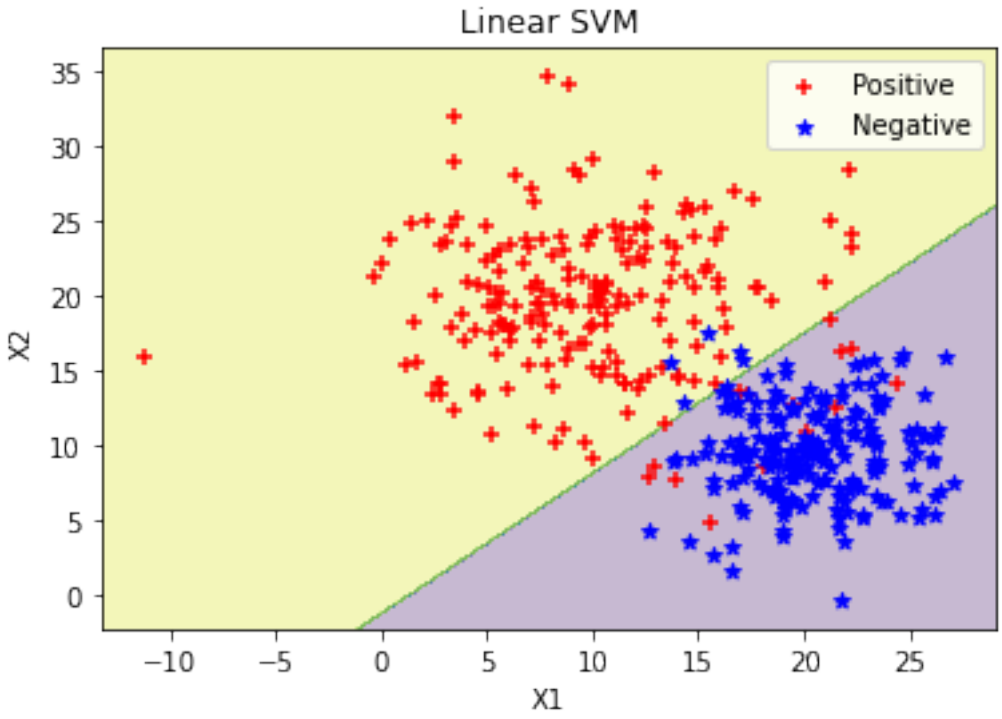
The dataset that is used is in binclass.txt
with gaussian class-conditional distributions with different covariance matrices for positive and negative class examples the decision boundary that we get is as follows




with gaussian class-conditional distributions with same covariance matrices for positive and negative class examples the decision boundary that we get is as follows

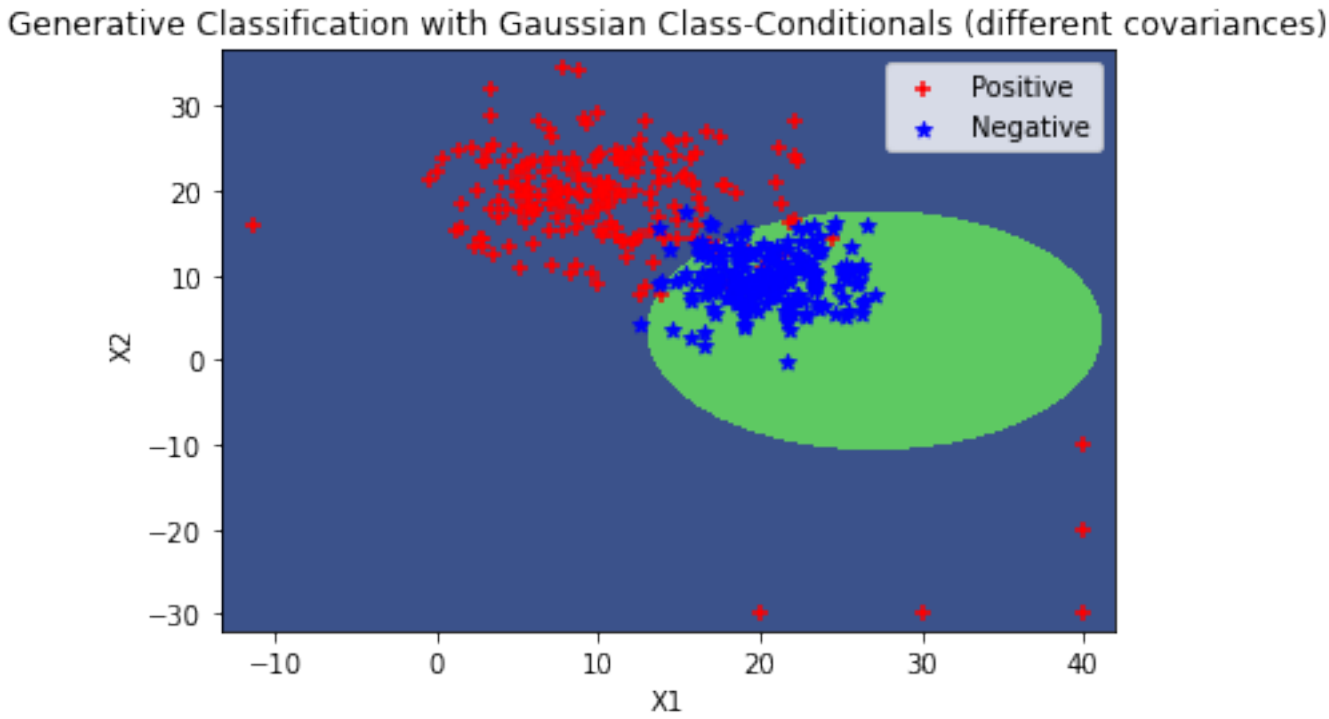



with SVM (from scikit-learn)

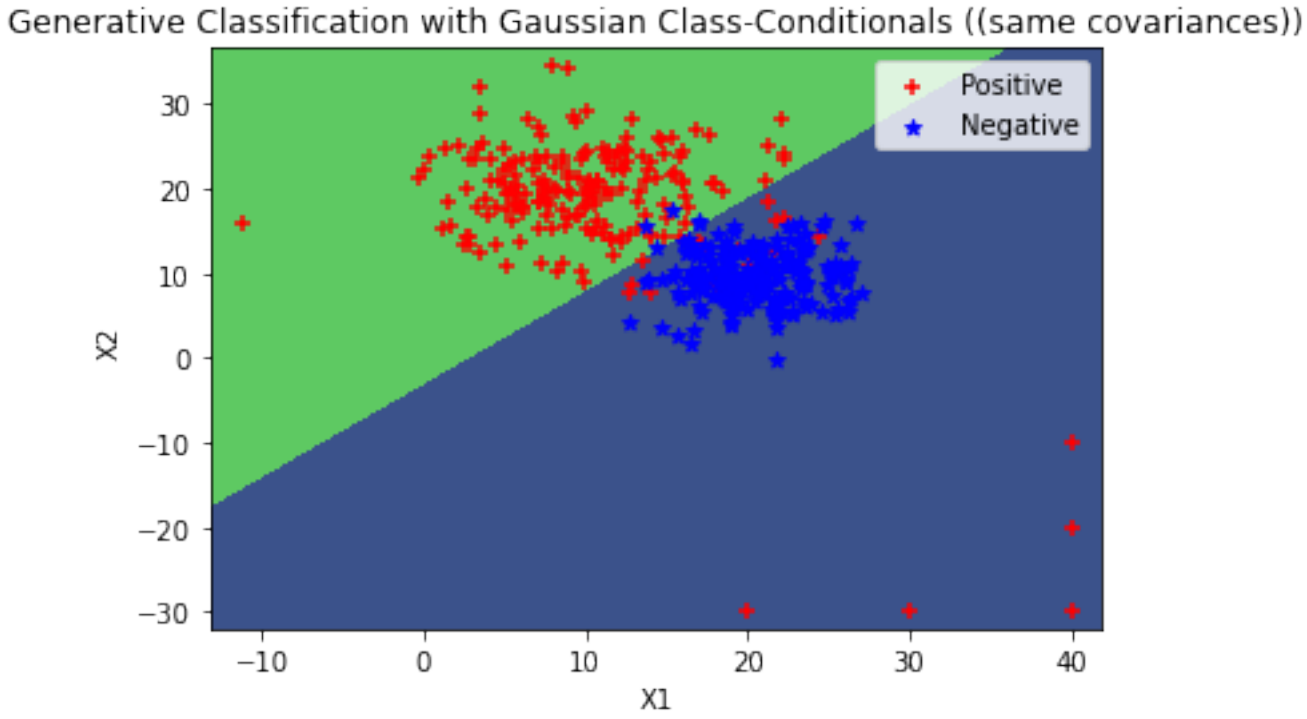



## 2 Part2

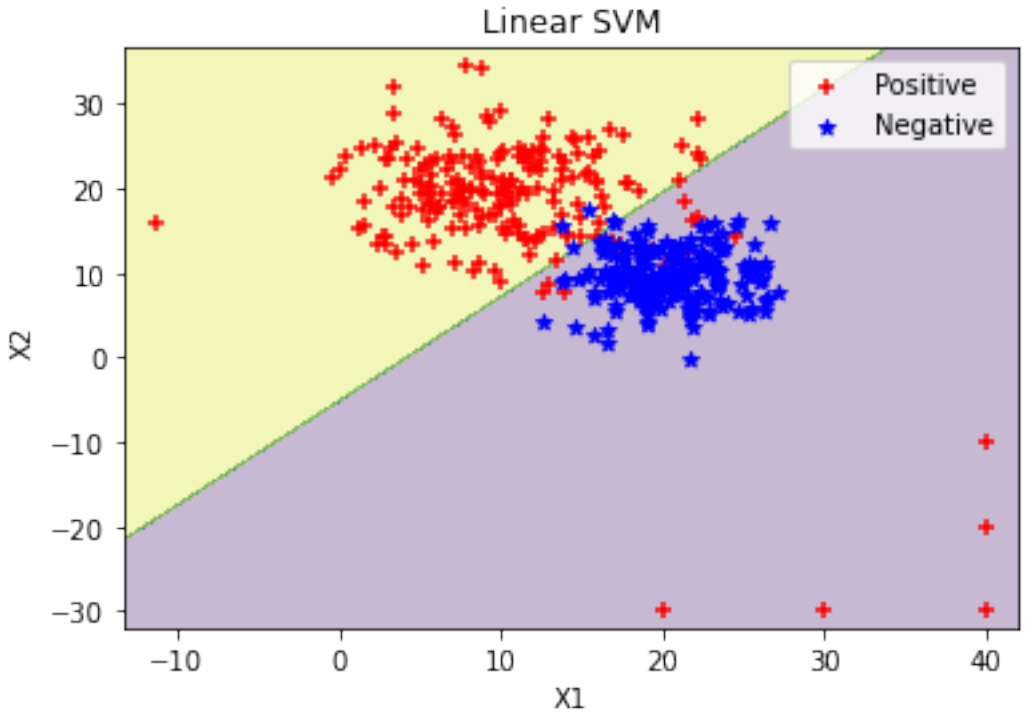
The dataset that is used is in binclassv2.txt
with gaussian class-conditional distributions with different covariance matrices for positive and negative class examples the decision boundary that we get is as follows




with gaussian class-conditional distributions with same covariance matrices for positive and negative class examples the decision boundary that we get is as follows




with SVM (from scikit-learn)




Observation:
First dataset: All the 3 decision boundaries are equally good
Second dataset: We see that there are some outliers and out generative classifier with different covariance was trying to overfit the data, we also have to keep in mind that we are using MLE estimation which is prone to overfit. But the other 2 models are good.
In general SVM with kernal is good as it can learn non linear boundaries with maximum margin.