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

Student Name: Debanjan Chatterjee
Roll Number: 20111016
Date: November 27, 2020

QUESTION 1

The given loss function is defined as follows:

$$\mathcal{L}(\mathbf{w}) = -\sum_{n=1}^{N} \left(y_n \mathbf{w}^T \mathbf{x}_n - \log\left(1 + \exp\left(\mathbf{w}^T \mathbf{x}_n\right)\right)\right) \tag{1}$$

Therefore, the gradient of the above loss function will be:

$$g = \frac{\partial \mathcal{L}}{\partial \mathbf{w}} = -\sum_{n=1}^{N} \left(y_n \mathbf{x}_n - \frac{\exp\left(\mathbf{w}^T \mathbf{x}_n\right) \mathbf{x}_n}{1 + \exp\left(\mathbf{w}^T \mathbf{x}_n\right)} \right) \tag{2}$$

Let $\mu_n = \frac{\exp\left(\mathbf{w}^T \mathbf{x}_n\right)}{1+\exp(\mathbf{w}^T \mathbf{x}_n)}$. Then the gradient becomes-

$$g = -\sum_{n=1}^{N} \left(y_n \mathbf{x}_n - \mu_n \mathbf{x}_n \right) = -\sum_{n=1}^{N} \left(y_n - \mu_n \right) \mathbf{x}_n = X^T (\boldsymbol{\mu} - \mathbf{y}) \tag{3}$$

The Hessian of the loss function will be as follows:

$$H = \frac{\partial^2 \mathcal{L}}{\partial \mathbf{w}^2} = \frac{\partial g^T}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}} \left(-\sum_{n=1}^{N} \left(y_n - \mu_n \right) \mathbf{x}_n^T \right) \tag{4}$$

In (4), μ_n is the term containing $\mathbf{w}$ terms, so the Hessian is equivalent to finding out the derivative with respect to $\mathbf{w}$ of the expression $\sum_{m=1}^{N} \mu_n \mathbf{x}_n^T$

$$\frac{\partial \mu_n}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}} \left(\frac{\exp\left(\mathbf{w}^T \mathbf{x}_n\right)}{1 + \exp\left(\mathbf{w}^T \mathbf{x}_n\right)} \right) = \frac{\exp\left(\mathbf{w}^T \mathbf{x}_n\right) \mathbf{x}_n}{1 + \operatorname{erp}\left(\mathbf{w}^T \mathbf{x}_n\right)} + (-1) \frac{\left(\operatorname{erp}\left(\mathbf{w}^T \mathbf{x}_n\right)\right)^2 \mathbf{x}_n}{\left(1 + \operatorname{erp}\left(\mathbf{w}^T \mathbf{x}_n\right)\right)^2} \tag{5}$$

Therefore,

$$\frac{\partial \mu_n}{\partial \mathbf{w}} = \frac{\exp\left(\mathbf{w}^T \mathbf{x_n}\right) \mathbf{x_n}}{\left(1 + \exp\left(\mathbf{w}^T \mathbf{x_n}\right)\right)^2} \tag{6}$$

Thus, the Hessian will become:

$$H = \sum_{n=1}^{N} \frac{exp\left(\mathbf{w}^T \mathbf{x}_n\right) \mathbf{x}_n}{\left(1 + \exp\left(\mathbf{w}^T \mathbf{x}_n\right)\right)^2} \mathbf{x}_n^T = \sum_{n=1}^{N} \mu_n \left(1 - \mu_n\right) \mathbf{x}_n \mathbf{x}_n^T = X^T S X \text{ where } S = \operatorname{diagonal}\left(\mu_n \left(1 - \mu_n\right)\right) \tag{7}$$

Now, the iteration t of a second-order optimization based update is written as:

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \left(H^t\right)^{-1} \left(g^t\right) = \mathbf{w}^t - \left(X^T S^t X\right)^{-1} \left(X^T \left(\boldsymbol{\mu}^t - \mathbf{y}\right)\right) \tag{8}$$

$$\mathbf{w}^{t+1} = \left(X^T S^t X\right)^{-1} \left[\left(X^T S^t X\right) \mathbf{w}^t - \left(X^T \left(\boldsymbol{\mu}^t - \mathbf{y}\right)\right)\right] \tag{9}$$

$$\mathbf{w}^{t+1} = \left(X^T S^t X\right)^{-1} X^T \left[\left(S^t X\right) \mathbf{w}^t - \left(\boldsymbol{\mu}^t - \mathbf{y}\right)\right] \tag{10}$$

$$\mathbf{w}^{t+1} = \left(X^T S^t X\right)^{-1} X^T S^t \left[X\mathbf{w}^t - \left(S^t\right)^{-1} \left(\boldsymbol{\mu}^t - \mathbf{y}\right)\right] \tag{11}$$

Let, $\mathbf{z}^t = X\mathbf{w}^t - \left(S^t\right)^{-1} \left(\boldsymbol{\mu}^t - \mathbf{y}\right)$

$$\mathbf{w}^{t+1} = \left(X^T S^t X\right)^{-1} X^T S^t \mathbf{z}^t \tag{12}$$

Now, considering the squared loss function:

$$\mathbf{w}^{t+1} = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{n=1}^{N} \gamma_n^t \left(\dot{y}_n^t - \mathbf{w}^{t^T} \mathbf{x}_n\right)^2 \tag{13}$$

If we take the derivative of $\mathbf{w}^{(t+1)}$ with respect to $\mathbf{w}$ and equate to zero , we get the weight vector as follows:

$$\mathbf{w}^{t+1} = \left(\sum_{n=1}^{N} \gamma_n^t \mathbf{x}_n \mathbf{x}_n^T\right)^{-1} \left(\sum_{n=1}^{N} \gamma_n^t \hat{y}_n^t \mathbf{x}_n\right) = \left(X^T \gamma^t X\right)^{-1} X^T \gamma^t \mathbf{y} \tag{14}$$

Therefore, considering (2), we can rewrite (1) as follows:

$$\mathbf{w}^{t+1} = \underset{\mathbf{w}}{\operatorname{argmin}} \left(\mathbf{z}^t - X\mathbf{w}^t\right)^T S^t \left(\mathbf{z}^t - X\mathbf{w}^t\right) \tag{15}$$

$$\mathbf{w}^{t+1} = \underset{\mathbf{w}}{\operatorname{argmin}} \sum^{N} S_{\mathrm{nn}}^t \left(z_n^t - \mathbf{w}^{t^T} \mathbf{x}_n\right)^2 \tag{16}$$

Therefore, $\gamma_n = S_{nn}$ which is the importance of nth training example and $y_n = z_n$ which denotes the modified real valued labels.

In the above expression γ_n makes sense intuitively because, since it is a importance weighted regression problem, it associates a importance γ_n with the input data points. Now, say, a particular data point has a high chance of being a outlier, for such a case, the associated γ_n will be very low, thus even for a large error between the prediction and the actual label for that data point, the model will be affected very little. Thus, leading to the learning of a better model., especially in cases where the input data points might have a large number of outliers.

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

Student Name: Debanjan Chatterjee
Roll Number: 20111016
Date: November 27, 2020

QUESTION 2

We know the form of update expression(ignoring the bias) for perceptron algorithm is,

$$\mathbf{w} = \mathbf{w} + y_n \mathbf{x}_n \tag{17}$$

and the weight vector learned by the perceptron algorithm can be given by:

$$\mathbf{w} = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n \tag{18}$$

If we want to make the perceptron algorithm learn non-linear boundaries we need to transform $\mathbf{x}$ using a feature map ϕ, as shown below

$$\mathbf{w} = \sum_{n=1}^{N} \alpha_n y_n \phi(\mathbf{x}_n) \tag{19}$$

Now, the prediction y_* for a test input $\mathbf{x}_*$ will be:

$$y_* = \text{sign}(\mathbf{w}^T \phi(\mathbf{x}_*)) = \text{sign}(\sum_{n=1}^{N} \alpha_n y_n \phi(\mathbf{x}_n)^\top \phi(\mathbf{x}_*)) = \text{sign}(\sum_{n=1}^{N} \alpha_n y_n k(\mathbf{x}_n, \mathbf{x}_*)) \tag{20}$$

where, k is the kernel with feature map ϕ.

Thus the modified perceptron algorithm using kernels will be, note that t represents the tth iteration, and η_t represents the corresponding learning rate:

Step 1: Initialize $\alpha_n = 0$ for all data points, i.e. $n = 1, 2, ..., N$, using this in (19), set $\mathbf{w} = \mathbf{w}^{(t)}, t = 0, \eta_t = 1 \forall t$

Step 2: Randomly pick a $(\mathbf{x}_m, y_m)$ pair. (since we are using stochastic gradient descent)

Step 3: If the current w makes a **mistake** on $(\mathbf{x}_m, y_m)$, i.e,

$$y_m y_{m*} \leq 0 \tag{21}$$

Using (20) we can perform further simplification, note that in the following steps, the $sign()$ function has been ignored, as it will be inconsequential to the condition check:

$$y_m \mathbf{w}^{(t)T} \phi(x_m) \leq 0 \tag{22}$$

$$y_m \sum_{n=1}^{N} \alpha_n^{(t)} y_n k(\mathbf{x}_n \mathbf{x}_m) \leq 0 \tag{23}$$

(23) will be the mistake condition for the algorithm.

If (23) is satisfied, i.e. the prediction was a mistake, we need to increment α_m by 1. Therefore, we perform,

$$\alpha_m^{t+1} = \alpha_m^t + 1 \tag{24}$$

Update $w^{(t+1)}$, by plugging the updated value of $\alpha_m^{(t+1)}$ in the expression (19). This will be the update expression of the algorithm.

Step 4: If not converged, go back to **Step 2**

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

Student Name: Debanjan Chatterjee
Roll Number: 20111016
Date: November 27, 2020

QUESTION 3

For the modified SVM, the objective function is defined as follows:

$$\min_{\mathbf{w},b,\xi} f(w,b,\xi) = \frac{\|\mathbf{w}\|^2}{2} + \sum_{n=1}^{N} C_{y_n}\xi_n \text{subject to} y_n\left(\mathbf{w}^T\mathbf{x}_n + b\right) \geq 1 - \xi_n \text{and} \xi_n \geq 0 \forall n \tag{25}$$

The Lagrangian problem of this modified SVM is as follows:

$$\begin{aligned}\min_{\mathbf{w},b,\xi}\max_{\alpha\geq 0,\beta\geq 0} L(\mathbf{w},b,\xi,\alpha,\beta) = \frac{\|\mathbf{w}\|^2}{2} &+ C_{+1}\sum_{n:y_n=+1}\xi_n + C_{-1}\sum_{n:y_n=-1}\xi_n \\ &+ \sum_{n=y_n=+1}\alpha_{n_{(+1)}}\left(1 - \left(y_n\left(\mathbf{w}^T\mathbf{x}_n + b\right)\right) - \xi_n\right) \\ &+ \sum_{n:y_n=-1}\alpha_{n_{(-1)}}\left(1 - \left(y_n\left(\mathbf{w}^T\mathbf{x}_n + b\right)\right) - \xi_n\right) \\ &- \sum_{n=y_n=+1}\beta_{n_{(+1)}}\xi_n - \sum_{n=y_n=-1}\beta_{n_{(-1)}}\xi_n\end{aligned} \tag{26}$$

where C_{+1} denotes the cost of mis-clasifying the positive examples and C_{-1} denotes the cost of mis-classifying the negative examples, and the Lagrangian multipliers $\alpha_{n_{(+4)}}$ and $\beta_{n_{(+1)}}$ are associated with positive examples i.e. $\forall y_n = +1$ and $\alpha_{n_{(-1)}}$ and $\beta_{m_{(-1)}}$ are associated with negative examples i.e. $\forall y_n = -1$

The derivatives of the primal variables are as follows :

$$\frac{\partial\mathcal{L}}{\partial\mathbf{w}} = \mathbf{w} - \sum_{n=y_n=+1}\alpha_{n_{(+1)}}y_n\mathbf{x}_n - \sum_{n:y_n=-1}\alpha_{n_{(-1)}}y_n\mathbf{x}_n = 0 \tag{27}$$

$$\therefore \mathbf{w} = \sum_{n=y_n=+1} a_{n_{(+1)}}y_n\mathbf{x}_n + \sum_{n=y_n=-1} a_{n_{(-1)}}y_n\mathbf{x}_n = \sum_{n=1}^{N}\alpha_n y_n\mathbf{x}_n \tag{28}$$

$$\frac{\partial\mathcal{L}}{\partial b} = \sum_{n:y_n=+1}\alpha_{n_{(+1)}}y_n + \sum_{n:y_n=-1}\alpha_{n_{(-1)}}y_n = \sum_{n=1}^{N}\alpha_n y_n = 0 \tag{29}$$

$$\frac{\partial\mathcal{L}}{\partial\xi_n}(\text{ when } y_n = +1) = C_{+1} - \alpha_{n_{(+1)}} - \beta_{n_{(+1)}} = 0 \tag{30}$$

$$\because \beta_{n_{(+1)}} \geq 0, \therefore \alpha_{n_{(+1)}} \leq C_{+1} \tag{31}$$

$$\frac{\partial \mathcal{L}}{\text{if}} (\text{ when } y_n = -1) = C_{-1} - \alpha_{n-1} - \beta_{n-1} = 0 \tag{32}$$

Now, on substituting the primal variables we get the dual problem as follows:

$$\max_{\alpha_n} \left(\sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n,m=1}^{N} \alpha_m \alpha_n y_m y_n x_m^T x_n \right) \tag{33}$$

$$\text{such that } \sum_{n=1}^{N} \alpha_n y_n = 0 \text{ and } 0 \leq \alpha_{n_{(+1)}} \leq C_{+1} \text{and } 0 \leq \alpha_{n_{(-1)}} \leq C_{-1} \forall n = 1 \dots N$$

Now for the last part of the question, in a standard SVM dual problem, since there is no class imbalance problem, there will be only one C value for both positive and negative samples, because we can penalize the model equally on mis-classification on either sample type.

However, in case of SVM with class imbalance dual problem, we need different values for C, i.e. C_{+1} and C_{-1} for positive and negative classes respectively. Whichever class has a lower number of samples, we need to set the corresponding C value higher in the same proportion as the ratio of the samples, in order to penalize the model more on mis-classifying the minority class. For example, if lets say, the ratio of the number of positive to negative samples is $1 : 2$, we need to set C_{+1} as 2, and C_{-1} as 1, in order to keep the model unbiased towards any class.

Student Name: Debanjan Chatterjee
Roll Number: 20111016
Date: November 27, 2020

QUESTION 4

Step 1: Assigning the data point x_n greedily to the best cluster

Randomly select an example $\mathbf{x}_n$ where $n \in \{1, 2, \ldots N\}$ assuming N to be the total number of data examples. Assign x_n to the closest cluster $^n\mathrm{k}''$ by finding the smallest of all distances between x_n and all cluster centers μ_k where $k \in \{1, 2, \ldots .K\}$ assuming K is the total number of clusters using the following:

Predicted cluster for point $\mathbf{x}_n$ be C_k

$$C_k = \left\{ n : k = \underset{k}{\operatorname{argmin}} \left\| \mathbf{x}_n - \boldsymbol{\mu}_k \right\|^2 \right\} \tag{34}$$

Step 2: The SGD-based cluster mean update equation

We increment n_k by 1 every time a point $\mathbf{x}_n$ is assigned to cluster k i.e

$$n_k = n_k + 1 \text{ where } n_k \text{ is size of cluster k} \tag{35}$$

Since we are using single single data point x_n and assigning it to a cluster k, we only need to update the k^{th} cluster mean as all other cluster means will remain same. So our Loss function is as follows:

$$\mathcal{L} = \sum_{n=1}^{n_k} z_{nk} \left\| \mathbf{x}_n - \boldsymbol{\mu}_k \right\|^2 \text{ where } z_{nk} = 1 \text{ as } \mathbf{x}_n \text{ belongs to cluster k} \tag{36}$$

Therefore the loss function can be re-written as:

$$\mathcal{C} = \sum_{n=1}^{n_k} \left\| \mathbf{x}_n - \boldsymbol{\mu}_k \right\|^2 \tag{37}$$

We need to minimize the loss function with respect to μ_k. Therefor the gradient with respect to μ_k will be :

$$g_{\mu_k} = \frac{\partial \mathcal{L}}{\partial \mu_k} = -\sum_{n=1}^{m_k} 2 \left(\mathbf{x}_n - \boldsymbol{\mu}_k \right) == -2 \left(\sum_{n=1}^{n_k} \mathbf{x}_n - \sum_{n=1}^{n_k} \boldsymbol{\mu}_k \right) \tag{38}$$

Therefore,

$$g_{\mu_k} = -2 \left(\mu_k \left(n_k - 1 \right) + \mathbf{x}_n - n_k \boldsymbol{\mu}_k \right) = -2 \left(\mathbf{x}_n - \boldsymbol{\mu}_k \right) \tag{39}$$

The update equation for SGD will be :

$$\boldsymbol{\mu}_k^{t+1} = \boldsymbol{\mu}_k^t - \eta_t g_{\mu_k}^t \tag{40}$$

where η_t is the learning rate at time t.

Therefore, the update equation on substituting (39) is:

$$\boldsymbol{\mu}_k^{t+1} = \boldsymbol{\mu}_k^t + \eta_t \left(\mathbf{x}_n - \boldsymbol{\mu}_k^t\right) \tag{41}$$

This update equation makes sense intuitively because, the updated mean of the cluster to which x_n was assigned must consider the mean of all the ${}^n n_k{}^n$ examples present in that cluster (including point $\mathbf{x}_n$). Also the update on the mean (i.e. the cluster center) is controlled by choosing an appropriate learning rate n_t.

A good choice of learning rate will be as follows :

We should set the learning rate $\eta_t = \frac{1}{2n_k}$, because when we set the above learning rate, the update equation in SGD becomes:

$$\boldsymbol{\mu}_k^{t+1} = \boldsymbol{\mu}_k^t + 2\left(\frac{1}{2n_k}\left(\mathbf{x}_n - \boldsymbol{\mu}_k^t\right)\right) = \boldsymbol{\mu}_k^t + \frac{1}{n_k}\left(\mathbf{x}_n - \boldsymbol{\mu}_k^t\right) \tag{42}$$

The above update equation is exactly same as obtaining the new average of all data points in the k^{th} cluster to which our example $\mathbf{x}_n$ was assigned.The learning rate is good because, as more and more data points get assigned to a cluster l.e. the cluster size increases (n_k increases) the learning rate $\frac{1}{2n_k}$ decreases as we move closer to the exact solution of $\boldsymbol{\mu}_k$.

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

Student Name: Debanjan Chatterjee
Roll Number: 20111016
Date: November 27, 2020

QUESTION 5

The kernel K-Means algorithm is defined as follows:

Input: Specify number of clusters K, N input data points, where each data-point has D dimensions.

Step 1: Selecting K data points without replacement for the initializing clusters: $\pi_1, \ldots, \pi_k$.

Step 2: Compute the Kernel matrix $\mathbf{K}$, where the $(i,j)th$ entry represents $k(a_i, a_j)$, where k denotes the kernel and ϕ is implicit feature map. Also note:

$$k(a_i, a_j) = \phi(a_i) \cdot \phi(a_j) \text{where} i = 1 \ldots N \text{and}$$

Difference: For standard k-means we do not need this step instead we initialize the mean of the K clusters, by the previously assigned data-point itself, as each cluster currently has only one data-point.

Step 3: For each data point $a_i$and for each cluster compute the distance of the data-point from the cluster center in the transformed space using:

$$D\left(\{\pi_c\}_{c=1}^{K}\right) = \sum_{c=1}^{K} \sum_{\mathbf{a}_i \in \pi_c} \left\|\phi\left(\mathbf{a}_i\right) - \phi(\mathbf{m}_c)\right\|^2, \text{ where } \mathbf{m}_c = \frac{\sum_{\mathbf{a}_i \in \pi_e} \phi\left(\mathbf{a}_i\right)}{|\pi_c|} \tag{43}$$

where,

c^{th} cluster is denoted by π_c
$\phi(\mathrm{m}_c)$ - denotes the mean of ϕ mappings of the data-points assigned to the cluster π_c
$\phi\left(a_i\right)$- denotes the data point a_i in transformed space.
Now,

$$||\phi(\mathbf{a}_i) - \phi(\mathbf{m}_c)||^2 = \phi\left(\mathbf{a}_i\right) \cdot \phi\left(\mathbf{a}_i\right) - \frac{2\sum_{\mathbf{a}_j \in \pi_e} \phi\left(\mathbf{a}_i\right) \cdot \phi\left(\mathbf{a}_j\right)}{|\pi_c|} + \frac{\sum_{\mathbf{a}_j, \mathbf{a}_i \in \pi_c} \phi\left(\mathbf{a}_j\right) \cdot \phi\left(\mathbf{a}_i\right)}{|\pi_c|^2} \tag{44}$$

Therefore,

$$||\phi(\mathbf{a}_i) - \phi(\mathbf{m}_c)||^2 = \phi\left(\mathbf{a}_i\right) \cdot \phi\left(\mathbf{a}_i\right) - \frac{2\sum_{\mathbf{a}_j \in \pi_e} \phi\left(\mathbf{a}_i\right) \cdot \phi\left(\mathbf{a}_j\right)}{|\pi_c|} + \frac{\sum_{\mathbf{a}_j, \mathbf{a}_i \in \pi_c} \phi\left(\mathbf{a}_j\right) \cdot \phi\left(\mathbf{a}_i\right)}{|\pi_c|^2} \tag{45}$$

we can also rewrite it as,

$$||\phi(\mathbf{a}_i) - \phi(\mathbf{m}_c)||^2 = k(\mathbf{a}_i, \mathbf{a}_i) - \frac{2\sum_{\mathbf{a}_j \in \pi_e} k(\mathbf{a}_i, \mathbf{a}_j)}{|\pi_c|} + \frac{\sum_{\mathbf{a}_j, \mathbf{a}_i \in \pi_c} k(\mathbf{a}_j, \mathbf{a}_i)}{|\pi_c|^2} \tag{46}$$

Use the values of $k(.,.)$ directly from the kernel matrix $\mathbf{K}$ pre-computed in Step 2.

Difference: For standard K-means algorithm, the distance between the data-points between the data points and means of the clusters are computed as:

$$D\left(\{\pi_c\}_{c=1}^{K}\right) = \sum_{c=1}^{K} \sum_{\mathbf{a}_i \in \pi_c} \|\mathbf{a}_i - \mathbf{m}_c\|^2 \tag{47}$$

where, m_c is already computed, during initialization for first iteration, and recomputed at the end of every iteration henceforth.

Step 4: Assign data point to that cluster, whose distance is minimum, i.e,

$$\text{Find } \pi^*(\mathbf{a}_i) = \arg\min_c \left(\|\phi(\mathbf{a}_i) - \phi(\mathbf{m}_c)\|^2\right) \tag{48}$$

.

Step 5: For all the clusters: $c = 1 \ldots K$:

$$\text{Update cluster } \pi_c = \{\mathbf{a}_i \mid \pi^*(\mathbf{a}_i) = c\} \tag{49}$$

Difference: For standard k-means algorithm, we need to re-compute the mean of the K clusters and store them, but since we are using kernel k with an infinite dimensional feature map (e.g., an RBF kernel) ,we cannot store the cluster means in the kernel-induced feature space. For standard k-means clustering recomputing and storing the means simplifies the computation cost at Step 3.

Step 6: If converged, i.e no data points are re-assigned, terminate the algorithm and output the K clusters, else go to **Step 3**

Now, to answer the last part of the question, we need to examine the expressions of Step 3.

For standard K-means algorithm: cost of computation of distance of input to mean of clusters $\|\mathbf{a}_i - \mathbf{m}_c\|^2$ is $\mathcal{O}(D)$, since the cost of finding the Euclidean distance between two D-dimensional vectors is of the order of D.

Now for **kernel K-means algorithm**: The cost of performing $||\phi(\mathbf{a}_i) - \phi(\mathbf{m}_c)||^2$ would be $\mathcal{O}(|\pi_c|^2)$, where $|\pi_c|$ represents the number of data points in the cluster π_c. Also note, that since we use the pre-computed Kernel matrix K, the cost of computing $k(i,j)$ becomes $\mathcal{O}(1)$. In worst case, the cluster size can go upto the order of $\mathcal{O}(N)$, so in that case the worst case time complexity for performing input to cluster mean distance calculation will be $\mathcal{O}(N^2)$

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

Student Name: Debanjan Chatterjee
Roll Number: 20111016
Date: November 27, 2020

QUESTION 6

All the necessary plots, have been included (see next page). Now, coming to the plots, firstly lets look at the Generative model with different sigma values, in this case the decision boundary is quadratic. It seems to have achieved a better training accuracy, however, it might have also be a case of over fitting, as some of the red data points lying on the edge of the graph (for part 2), might actually be outliers, and hence this model might perform poorly for the test dataset.

Both the generative model with same sigma values as well as the SVM classifier, has a linear decision boundary which seems to have classified the data points evenly. On the surface, this seems to have not over-fitted on the training data, as they have not manipulated the decision boundary to correctly classify the probable outliers (part 2). In conclusion, the SVM model and the generative model with same sigma seem to be the better model, because it has not over-fitted.

Software used for SVM: **scikit-learn**

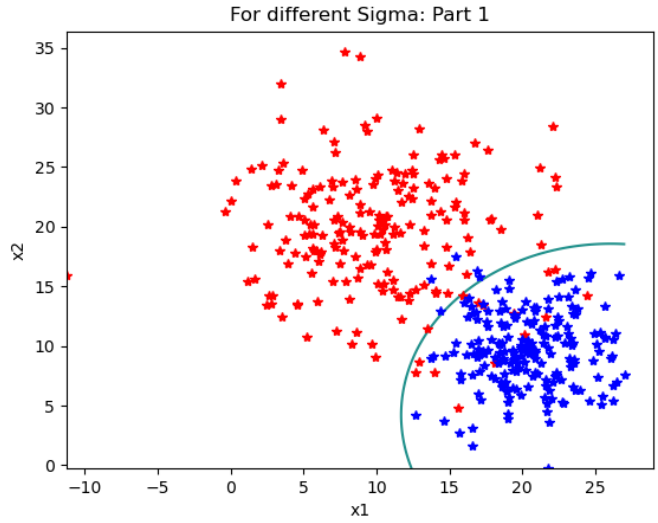

Figure 1: Plot for generative model with different sigma: part 1

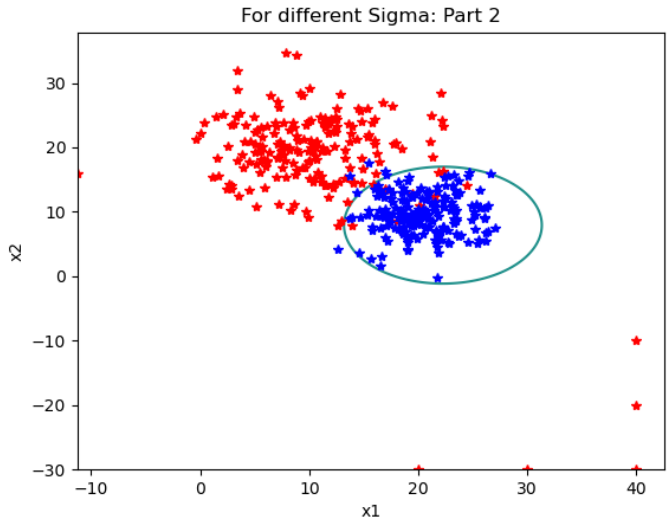

Figure 2: Plot for generative model with different sigma: part 2

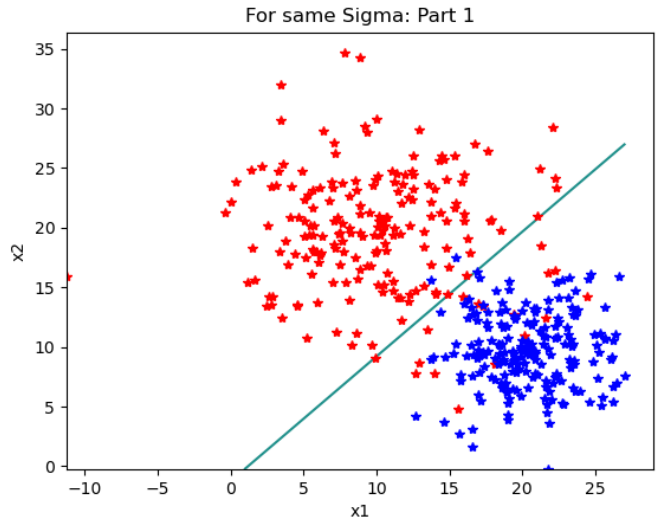

Figure 3: Plot for generative model with same sigma: part 1

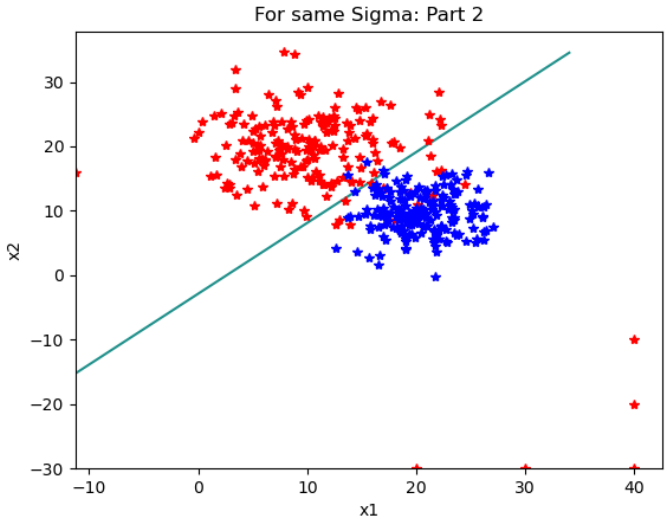

Figure 4: Plot for generative model with same sigma: part 2

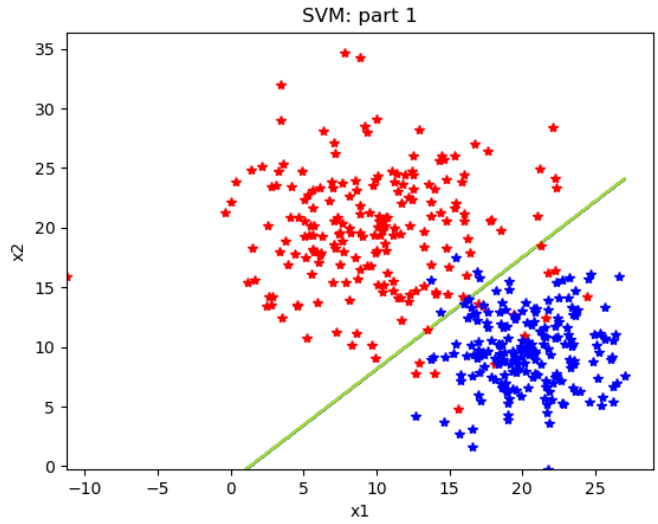

Figure 5: Plot for SVM: part 1

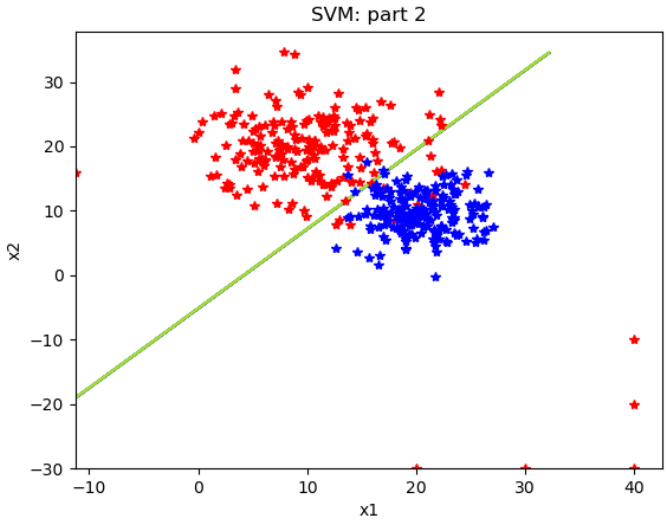

Figure 6: Plot for SVM: part 2