Neural Network Assignment 04 (Tutor Redion)

Priyanka Upadhyay (2581714) - s8prupad@stud.uni-saarland.de
Gopal Bhattrai (7013547) - gobh00001@stud.uni-saarland.de

Problem 01

(a) $z = w^T x + b$, $y = \sigma(z)$, $L = -t \log(y) - (1-t)\log(1-y)$

$\therefore h_\theta(x) = \sigma(w^T x + b)$

$$\text{Cost} = \begin{cases} -\log h_\theta(x) & , t = 1 \\ -\log(1 - h_\theta(x)) & , t = 0 \end{cases}$$

Rewritten Cost f?

✓ $$L = -\frac{1}{2N}\left[\sum_{i=1}^{N} t^{(i)} \log\left(\sigma(\underbrace{w^T x^{(i)} + b}\right)\right) + (1 - t^{(i)}) \log\left(1 - \sigma(w^T x^{(i)} + b)\right)\right]$$

b) If we introduced the $\{-1, 1\}$ instead of $\{0, 1\}$, we just need to change the deciding factors $\{t$ and $1-t\}$ accordingly. We can do two things:

i) Either encode all -1's with 0's and use the same above Cost f^n.

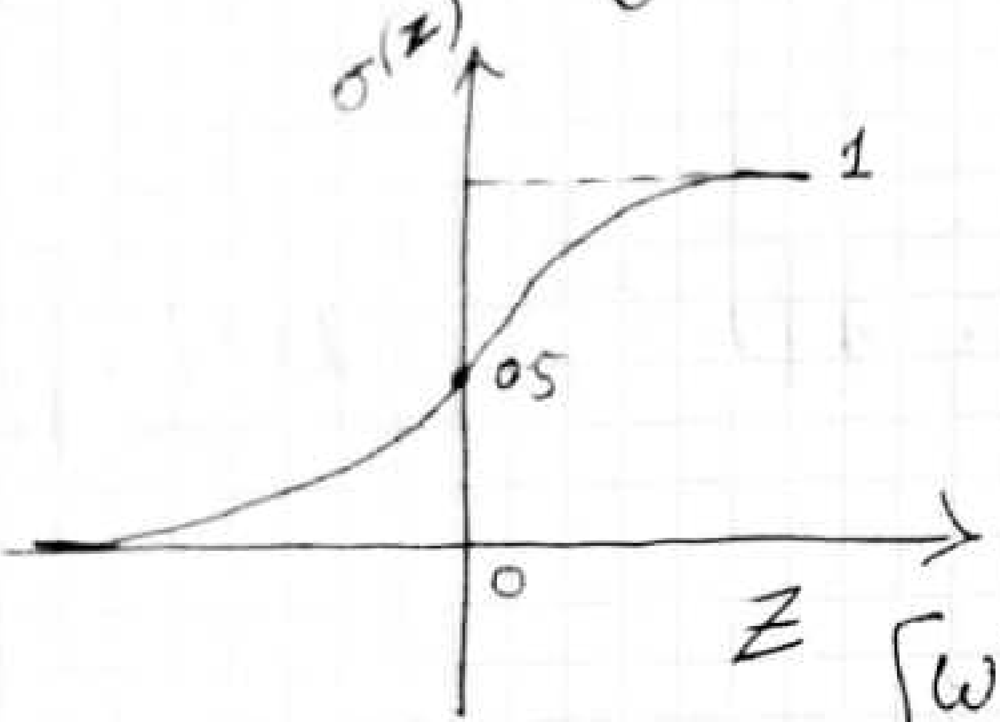
ii) Change the Cost f^n a bit accordingly:

$$\text{Cost} = \begin{cases} -\log h_\theta(x) \Rightarrow t = 1 \\ -\log(1 - h_\theta(x)) \Rightarrow t = -1 \end{cases}$$

$$L = -\frac{1}{2N}\left[\sum_{i=1}^{N} \frac{(1+t^{(i)})}{2} \log\left(\sigma(w^T x^{(i)} + b)\right) + \frac{(1-t^{(i)})}{2} \log\left(1 - \sigma(w^T x^{(i)} + b)\right)\right]$$

c) $y(x) = 1[z(x) > 0.5]$

$$= \begin{cases} 1 & , \sigma(\omega^T x + b) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$



For $\sigma(z) > 0.5$,

Z must be > 0.

i.e. $[\omega^T X + b > 0]$

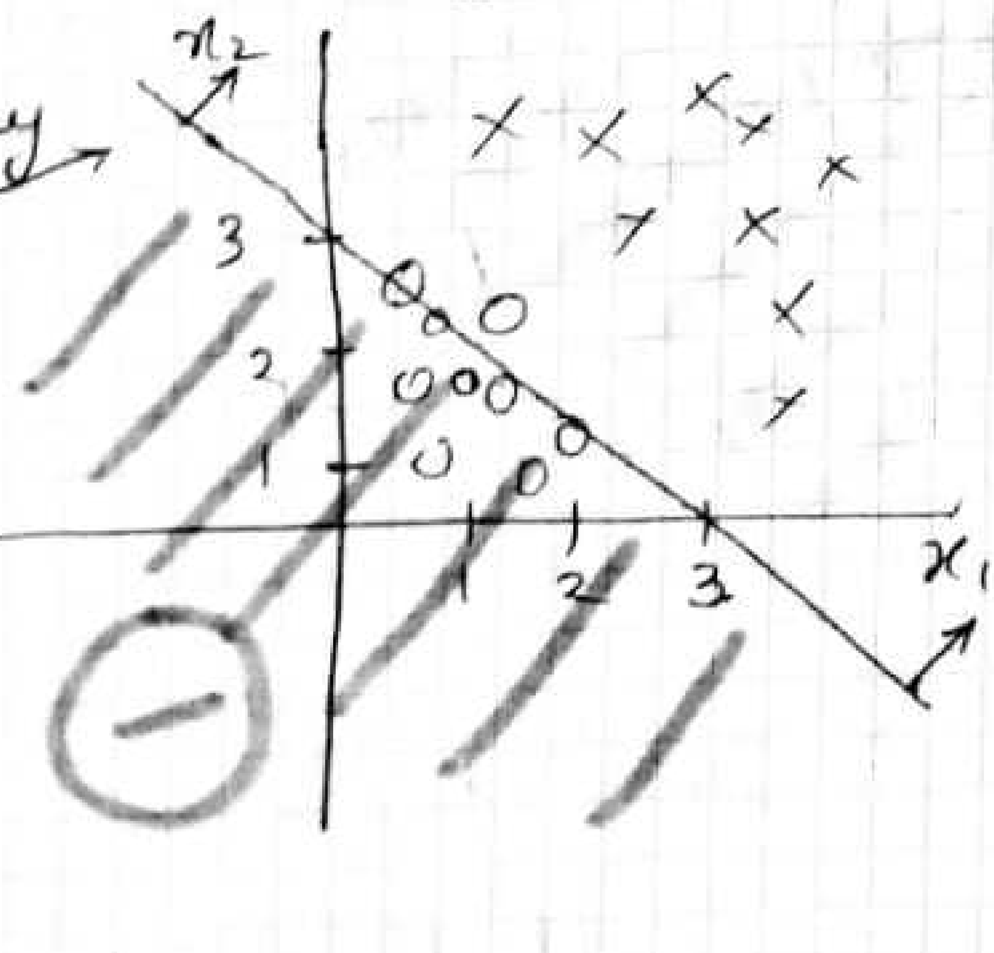
$[\omega^T X + b = b + \omega_0 x_0 + \omega_1 x_1 + \omega_2 x_2 + \dots + \omega_n x_n]$

Linear Comb^n of Inputs. $x_0 = 1$

So if this Linear Comb^n > 0, it outputs 1 else Zero. [Just like Linear Regression]

that is nothing but a linear decision boundary

Say →



And Say My $\omega = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$ → (I found by GD or SGD)

then. $\omega^T X = -3 * 1 + x_1 * 1 + x_2 * 1$

$= -3 + x_1 + x_2 > 0$

$\boxed{= x_1 + x_2 > 3}$

Comprise to Linear Decision boundary.

Problem-2.9 b = intercept term, $\vec{w}$ = weight vector
x_i = input data points, y_i = class label
for SVM $Y = -1$ or 1 (rather than $1, 0$)

Decision Rules are :-

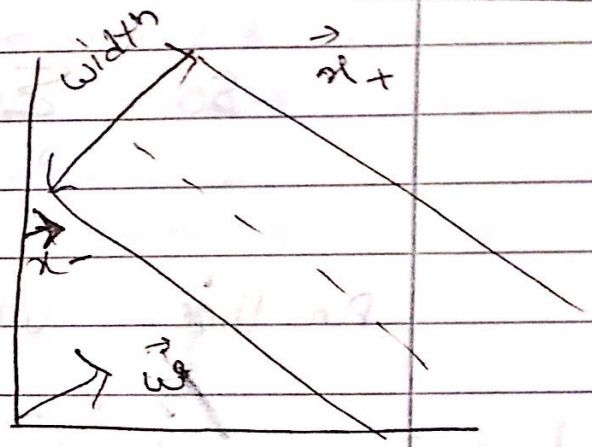
$$\vec{w} \cdot \vec{x} + b \geq 0$$

if $y = 1 \Rightarrow \vec{w} \cdot \vec{x} + b = 1$
$y = -1 \Rightarrow \vec{w}\vec{x} + b = -1$ } → (1)

→

$\vec{x_-}$ vector of class $y = -1$

$\vec{x_+}$ vector of class $y = 1$



$$\text{width} = (\vec{x_+} - \vec{x_-}) \frac{\vec{w}}{\|\vec{w}\|}$$

$$= \frac{\vec{x_+} w - \vec{x_-} w}{\|\vec{w}\|}$$

from eqn (1) $\quad \frac{1-(-1)}{\|w\|} = \frac{2}{\|\vec{w}\|}$ Ans

Reference: nlp.stanford.edu → SVM and MIT Lecture

(2.b) Plug in margin of width for x, since that is what we want to optimize.

$$L = \frac{2}{\|\vec{w}\|^2} - \sum_i \alpha_i \left[y_i (\vec{w} \cdot \vec{x}_i + b) - 1 \right]$$

$\left[\frac{2}{\|w\|^2} \text{ or } \frac{1}{2}\|w\|^2 \right]$

To find out the maximize of our marzin, take the derivative of L w.r.t w

and $\|\vec{w}\| = \vec{w}^T \vec{w}$ ~~$\left(\frac{\partial}{\partial w} \|\vec{w}\| = 2w \right)$~~

$$\rightarrow \frac{\partial L}{\partial w} = \frac{2w}{2w} - \sum_i \alpha_i \left[y_i (\vec{x}_i + \text{[illegible]}) - 0 \right]$$

→ Set $\frac{\partial L}{\partial w} = 0$

→ ~~$w \sum \alpha_i \left[y_i (\vec{w} \vec{x}_i + b) \right]$~~

$$\boxed{w = \sum_i \alpha_i y_i \vec{x}_i}$$

→ derivative of L w.r.t b

$$\Rightarrow \frac{\partial L}{\partial b} := \sum_i \alpha_i y_i$$

So $\sum_i \alpha_i y_i = 0$

Put these value into main L function:

$$L_{max} = \frac{1}{2} \vec{w}^T \vec{w} - \sum_i \alpha_i \left[y_i (\vec{w} \cdot \vec{x}_i + b) - 1 \right]$$

$$\Rightarrow \frac{1}{2} \left[\left(\sum_i \alpha_i y_i \vec{x}_i \right) \left(\sum_j \alpha_j y_j \vec{x}_j \right) \right] - \left[\left(\sum_i \alpha_i y_i \vec{x}_i \right) \left(\sum_j \alpha_j y_j x_j \right) - b \sum_i \alpha_i x_i y_i + \sum_i \alpha_i \right]$$

$L_{max} \Rightarrow \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \vec{x_i} \vec{x_j}$

Ans

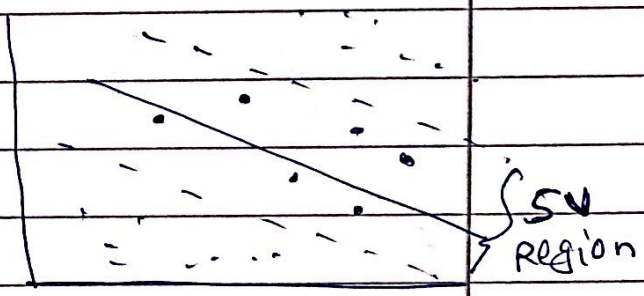
And max margin will become :-

$$\sum_i \alpha_i x_i y_i \vec{x} + b \geq 0$$

; Reference: MIT Lecture

(2.c) α_i determines whether or not $\vec{x_i}$ is a support vector (SV)

→ When $\alpha_i > 0 \rightarrow \vec{x_i}$ is a support vector
→ When $\alpha_i = 0 \rightarrow \vec{x_i}$ is not a support vector



(2.d) • Yes, when data are not seperable in the Lower dimension then maximized function uses Higher Dimension to seperate / Classify the data using Non Linearity.

Non Linear Data

In this approach SVM uses the kernal function $K(x_i, x_2)$

• Kernal function ~~uses~~ transform our Suppor vector Classifier into Higher Dimension

Problem →

(3.a) The domain Boundry of Logistic Regression is

$$\vec{w} = \sum_{n=1}^{m} \beta_n \vec{x}_n \rightarrow \text{Linear}$$

- So value of $\vec{w}$ is depend on weight of suppor vector $\vec{x}_n$.
- kernal function (maps) are points in $\vec{x}$ in Higher dimension and w is optimize by kernal function.

$$k(x, x_i) \text{ in } w$$

$$\vec{w} = \sum_{n=1}^{m} \beta_n k(x, x_i) \quad -(1)$$

and in 4.2.b, $\vec{w} = \sum_{i} a_i y_i x_i \quad -(2)$

Since for KLR, $\beta_n \neq 0$, it has $O(N^3)$ whereas SVM has a complexity $O(N^2K)$ and this way KLR will work for Non-Linear decision Boundry.

(3.b)
- KLR is complex and more expensive KLR has complexity of $O(N^3)$ where as SVM has complexity of $O(N^2K)$ [K = no. of sv points].
- KLR, all $\alpha_i \neq 0$ So you can not compress Data Like you do in SVM [where SVM only sv points have $\alpha \neq 0$]

~~KLR~~