Support Vector Machines

Nov. 16, 2022

Srinivasan Viswanathan

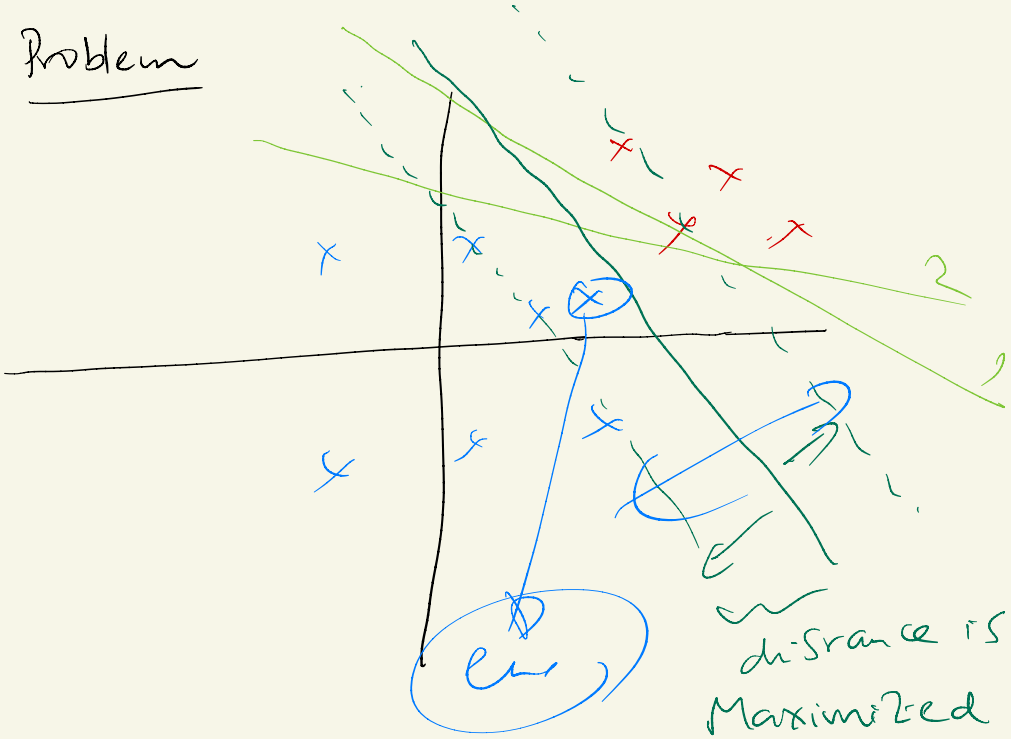
RECAP

- PERCEPTRON
  - finds Any hyper plane
  - Not optimal.

Algorithm

for. niterations
  for each point
    if (misrake)
      $w_{new} = w + (y w x_i)$
  if (NO new misrake) return $w_0 = w_0 + y_i$

Problem



Exercise

→ Distance between line $ax+b=c$ and point (x_1, y_1)

→

$$d = \frac{|\theta x + \theta_0|}{\|\theta\|}$$

Minimize d over x_i's

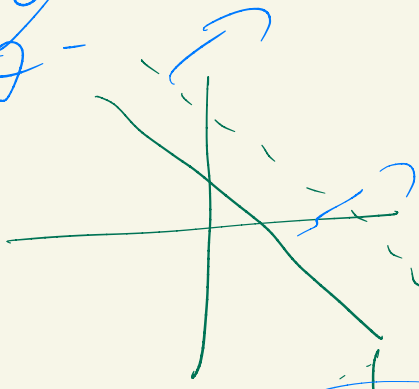
at the same time

Maximize the Margin.

Max [min. d.]

$\theta^T x + \theta_0 = \#$

$\min d = \min \frac{|\theta^T x + \theta_0|}{\|\theta\|}$



$\theta^T x + \theta_0 = 1$ → with our loss of Generality.

$\theta^T x + \theta_0 = 0.$

$d^2 = \frac{1}{\|\theta\|^2}$ ⟵ $d = \frac{1}{\|\theta\|}.$

$$\text{Max } d^2 = \text{Min } \|\theta\|^2$$

$$\text{Cost function} = \|\theta\|^2$$

Constraint

$$y_i(\theta x_i + \theta_0) \geq 1$$

Hinge Loss!

$z = y_i(\theta x_i + \theta_0)$

Hinge Loss:

$1 - z$

$z < 1$

$z \geq 1$

$y_i x_i$

-1

$0 \rightarrow \text{Cost} = 0$

$0 \rightarrow \text{Cost} = 0$

$0 \rightarrow \text{Cost} < 1$

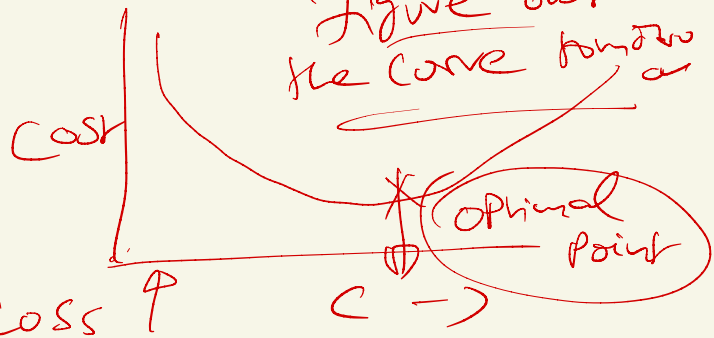
$\text{Cost} = 2$

$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} \text{Loss}$

$+ \frac{1}{2} \cdot \|w\|^2$

CC ⟺ C ⟺

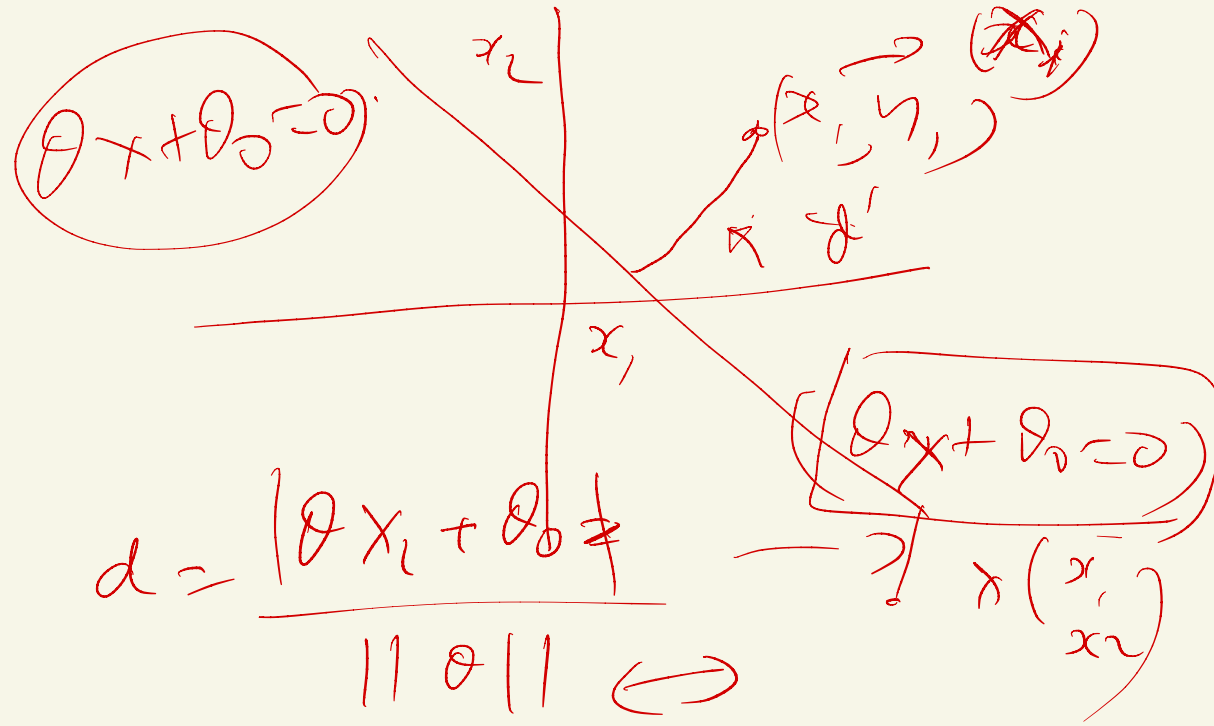
$C \geq \frac{1}{\lambda}$ ⟺ C.



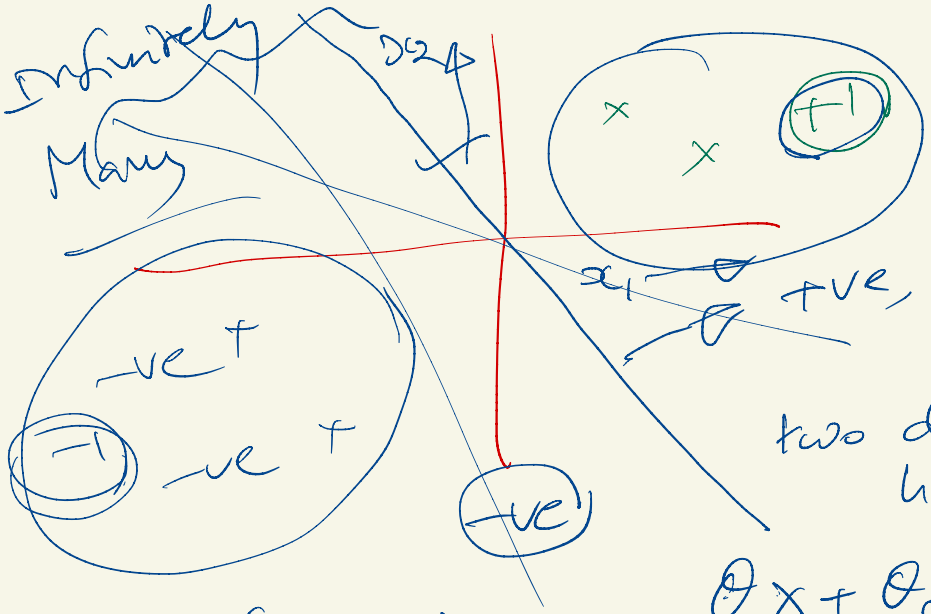
~~high~~

High C → Loss ↑

Low C → Margin ↑



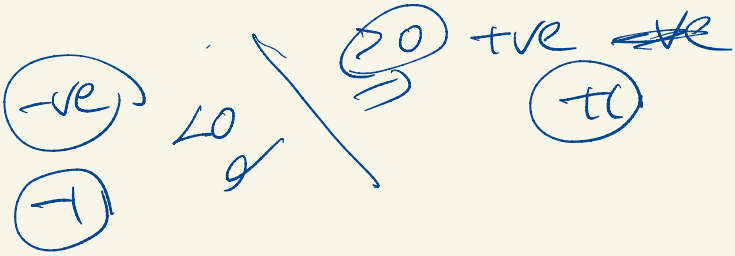
$$d = \frac{|\theta x_i + \theta_0|}{\|\theta\|}$$


Infinitely
Many
x2
x
x
+1
x1
+ve,
-ve
+
-1
-ve
+
+ve
two dimensional
line.
$\theta x + \theta_0 = 0$
$x = (x_1, x_2)$
$y \rightarrow$ Label.

if the points are linearly Separable

Perceptron Algorithm will Converge

to a line / hyperplane

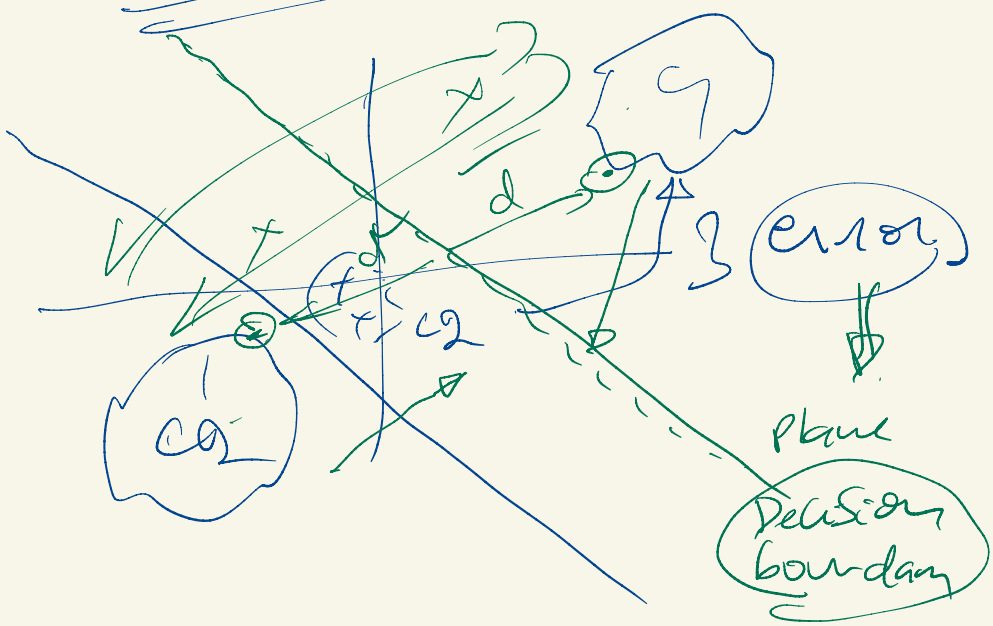


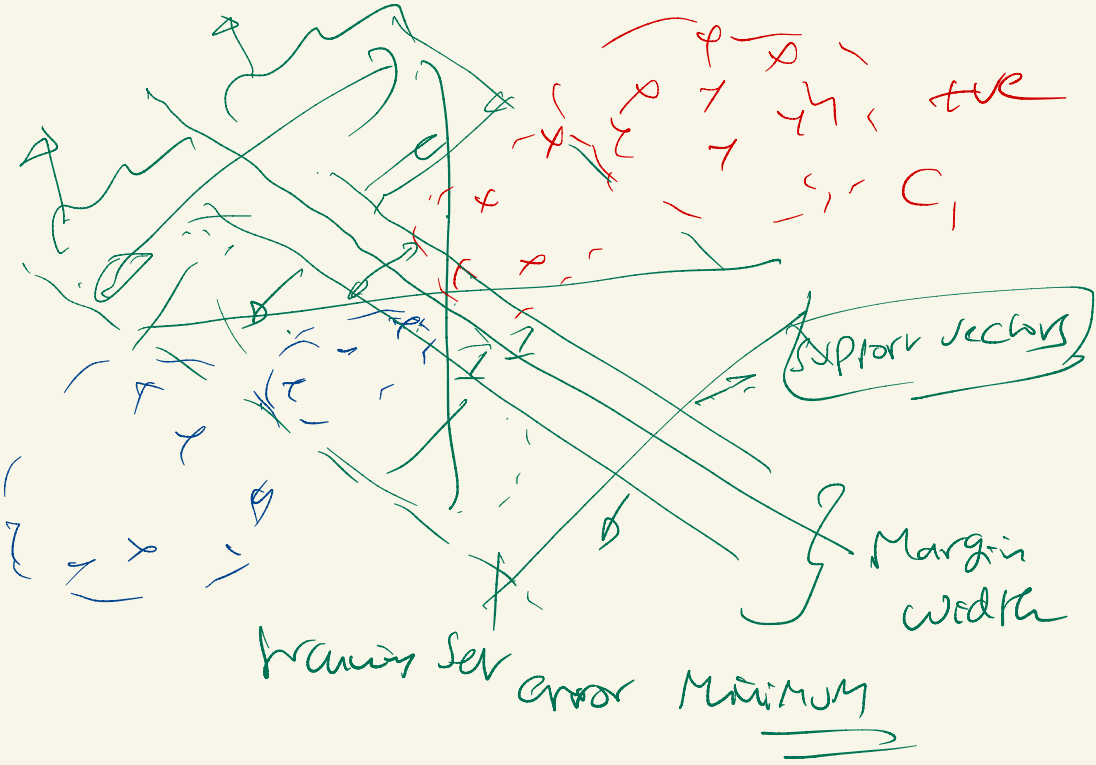
Lab Assignment on Perceptron

$\theta_1 \rightarrow$ *Each any θ_1, θ_0 as Parameters

Perceptron

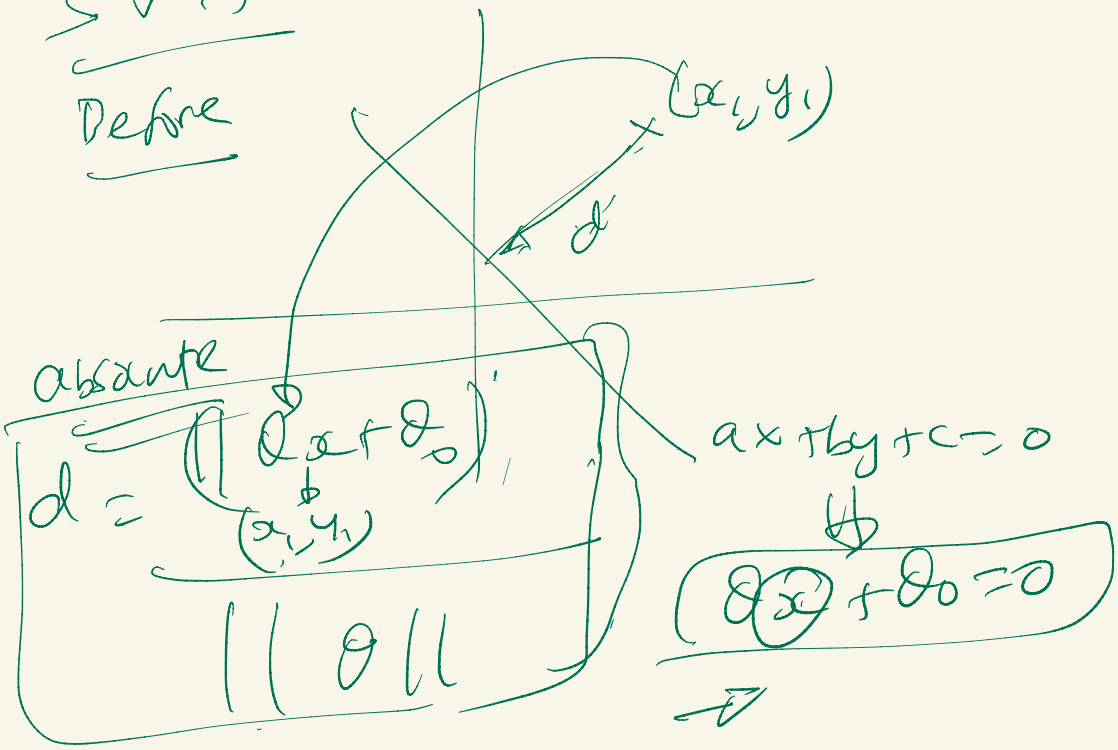
NOT optimal.

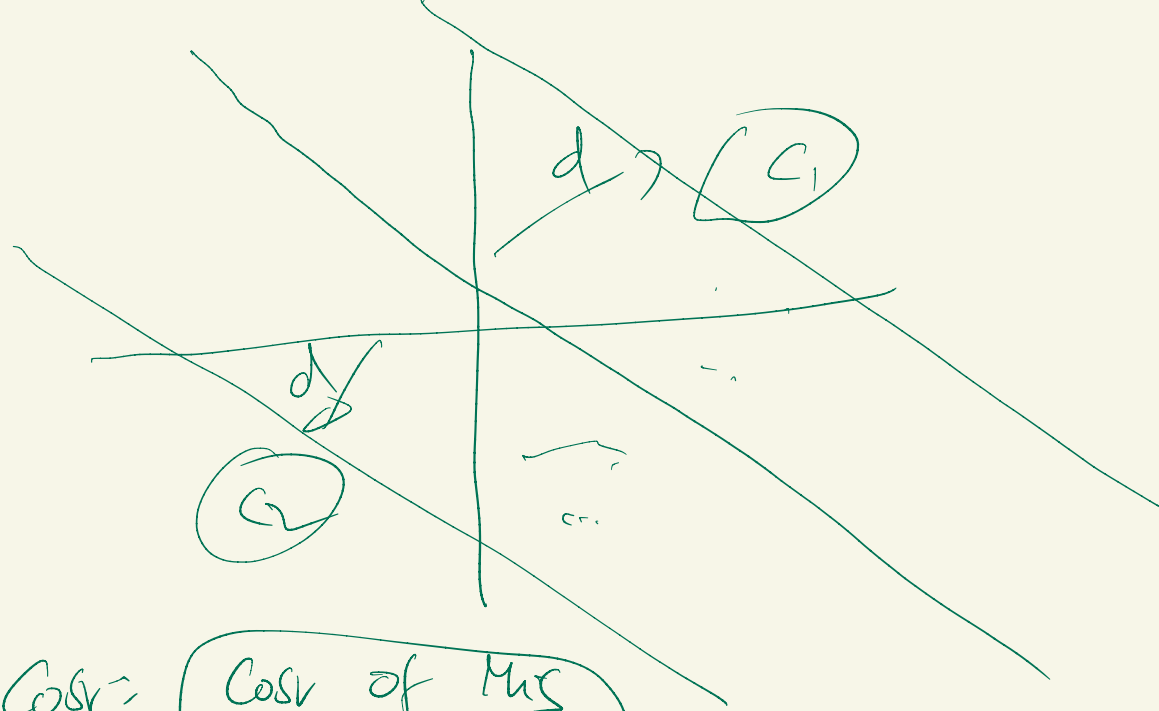



Support vectors
Margin width
Training set error Minimum

SVM

Before


(x_1, y_1)
d
absante
$\theta x + \theta_0$
$d =$
(x_1, y_1)
$\| \theta \|$
$ax + by + c = 0$
$\theta x + \theta_0 = 0$



Cost: Cost of MRS rate

$\text{Max.}[d]$

$d_i \in C_1 \quad \text{Min}\{d\}$

$d \cdot \text{Min}\left[\frac{\theta x + \theta_0}{\|\theta\|}\right]$

$d\left(\frac{\text{Min}|\theta x + \theta_0|}{\|\theta\|}\right) \to \theta$

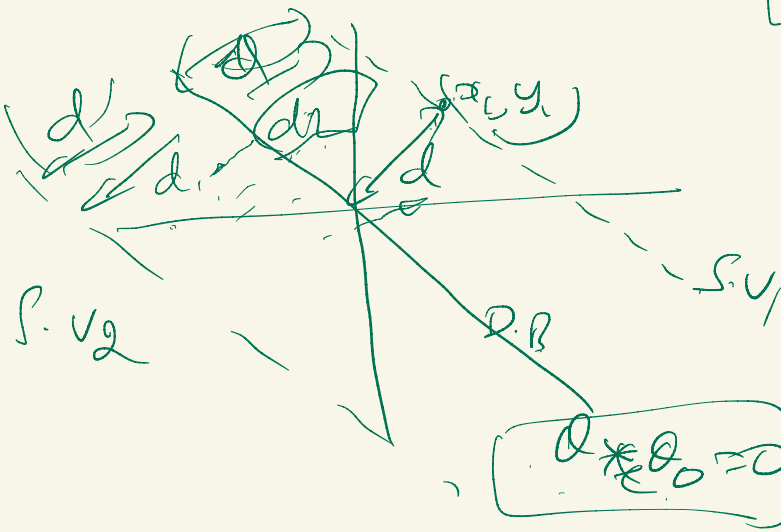
$d = 1 \text{ Min} (\theta x + \theta_0)$

II wll

$d_1 \neq d_2$

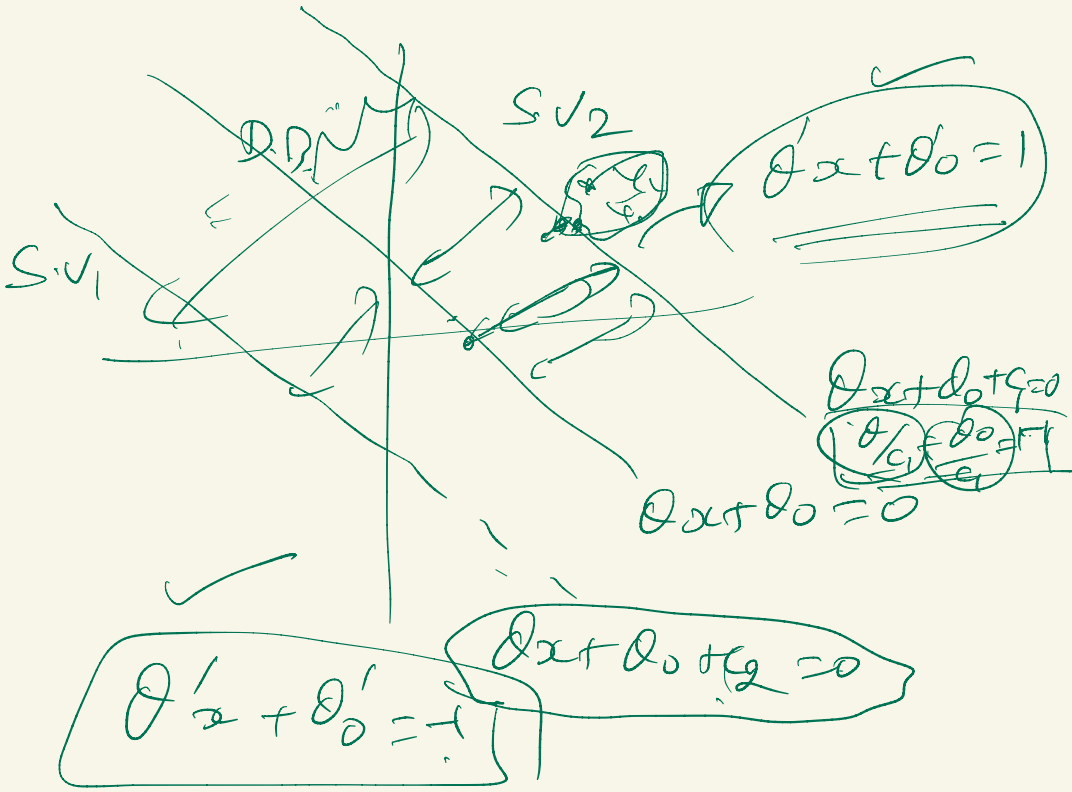
$d_1 > d_2$

$d_1 < d_2$



$d_1 \geq d_2$

$\theta * \theta_0 = 0$


S.V2
S.V1
$\theta' x + \theta_0 = 1$
$\theta x + \theta_0 + c = 0$
$\theta x + \theta_0 = 0$
$\theta' x + \theta_0' = -1$
$\theta x + \theta_0 + c_2 = 0$

Cost of Mistakes

V_1 – Perceptron SVM – motivation ✓

V_2 – 'distance' as a component of cost function & key role in SVM optimization

Min → $\|\theta\|^2$ · $\frac{1}{2}$

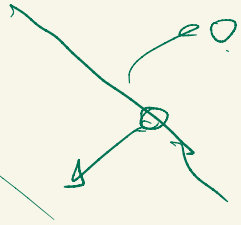
REGULARIZER → generalization

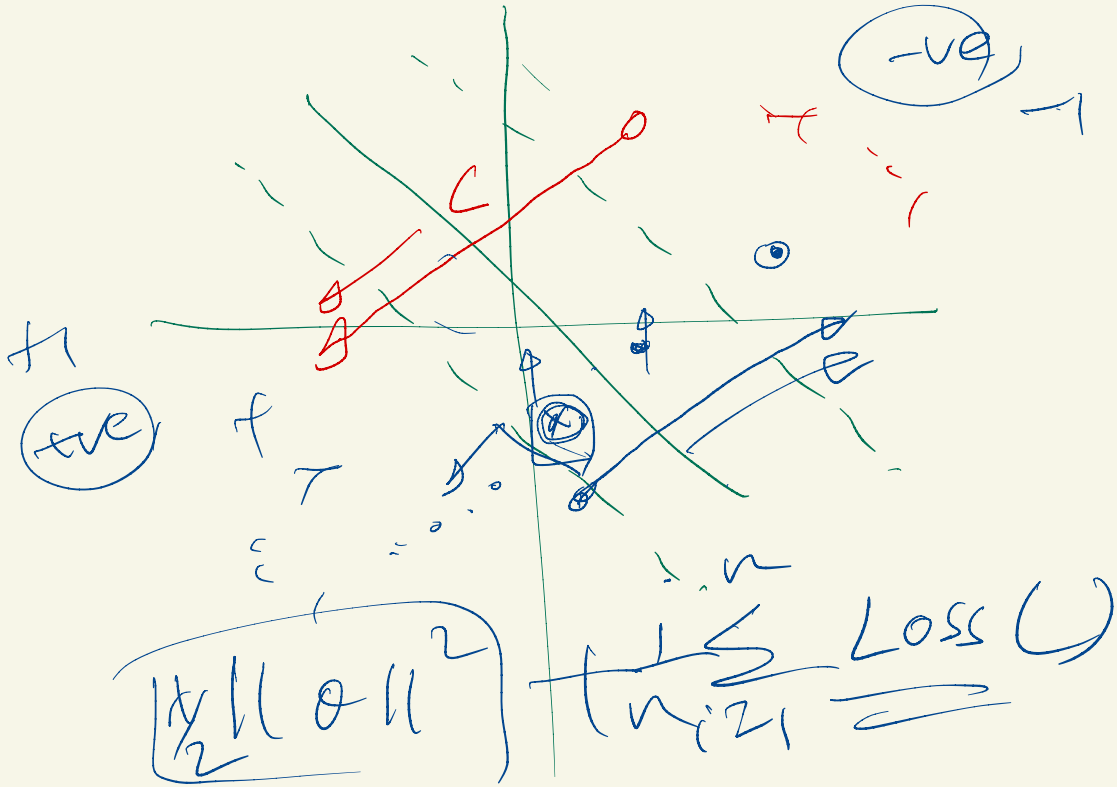
$\theta x_i + \theta_0 = -1$ (

$\rightarrow +1$ (

Cost function for Mistakes

$M = [y_i (\theta x_i + \theta_0) \leq 0]$




-ve
+ve

Hinge loss:

$y_i(\theta x_i + \theta_0) \geq 1$ — no cost

$y_i(\theta x_i + \theta_0) < 1$ — (cost)

$z = y_i(\theta x_i + \theta_0)$

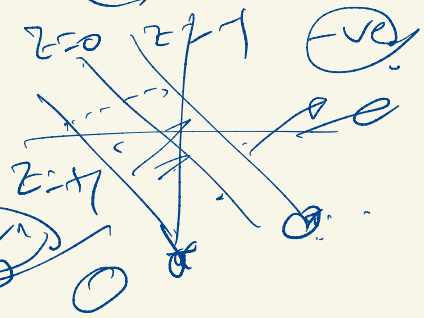
$\Downarrow$

$z = 0$

D.B — z

SV1 — $z = +1$

SV2 — $z > +1$



$$\text{hinge loss} = \begin{cases} 1 - z & z < 1 \\ 0 & z \geq 1 \end{cases}$$

$$Cf = \frac{1}{n} \sum_{i=1}^{n} \text{loss}_f(x_i, y_i)$$

$$z = y_i(x_i \theta + \theta_0)$$

$$1 - z = 1 - y_i(x_i \theta + \theta_0)$$

COST FUNCTION FORSVM

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} \text{Loss function}(x_i, y_i) + \frac{\lambda}{2} \|\theta\|^2$$

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} LF(x_i, y_i) + \frac{\lambda}{2} \|\theta\|^2$$

$$LF(x_i, y_i) = 1 - z \quad z < 1$$

$$= 0 \quad z > 1$$

$$z = y_i(\theta x_i + \theta_0)$$

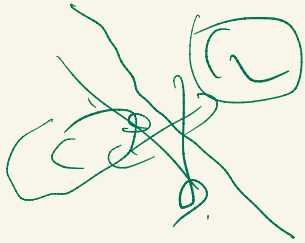
Maximized $(d)^{\circ}$ = where

d is Min. $(\theta \times 100)$

$= d$ if Min. $\frac{1}{\|\theta\|}$

Maximize $= \frac{1}{\|\theta\|}$

Max. $\frac{1}{\|\theta\|}$



$= \frac{1}{\|\theta\|^2}$

Minimum of $\lambda\|\theta\|^2$

Regularizer

Cost of Mistakes

→ Perceptron → Why SVM intuitively?

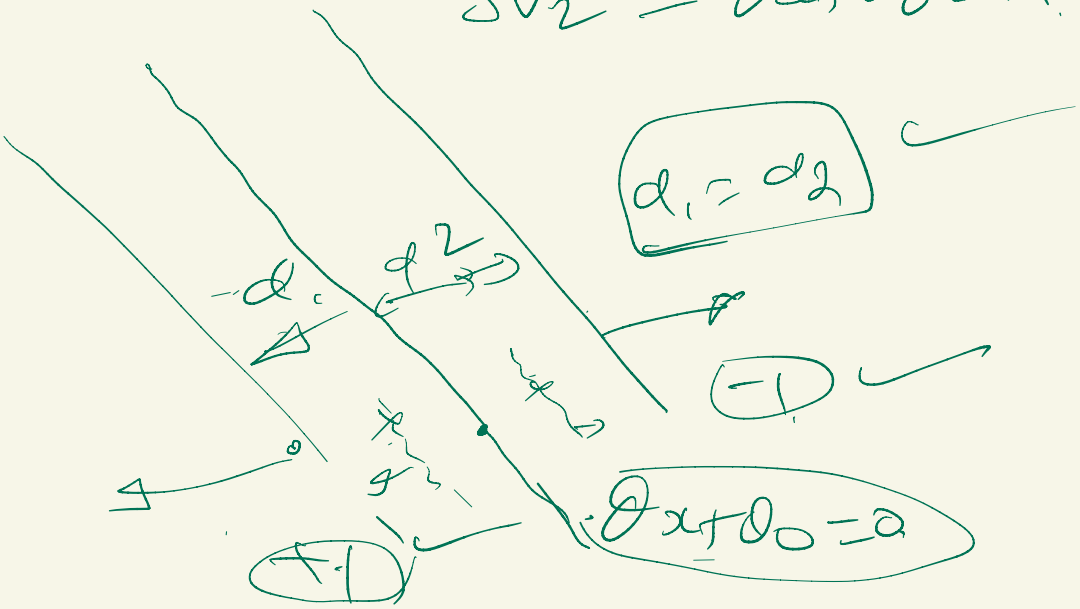
→ SVM in terms of distance

optimization

$$\text{Minimize} \left[\|\theta\|^2 \times \frac{\lambda}{2} \right] \rightarrow \phi$$

Support vectors : $SV_1 - \theta x + \theta_0 = -1$

$SV_2 - \theta x + \theta_0 = +1$



fully Math Problem:

Mini $\frac{1}{K^2}$ $\|w\|^2$ $\left(\frac{\lambda}{2}\right)$

s.t. $y_i(\alpha x_i + \beta_0) \geq 1$

Perceptron Algorithm

Multiple if $y_i(\alpha x_i + \beta_0) \leq 0$

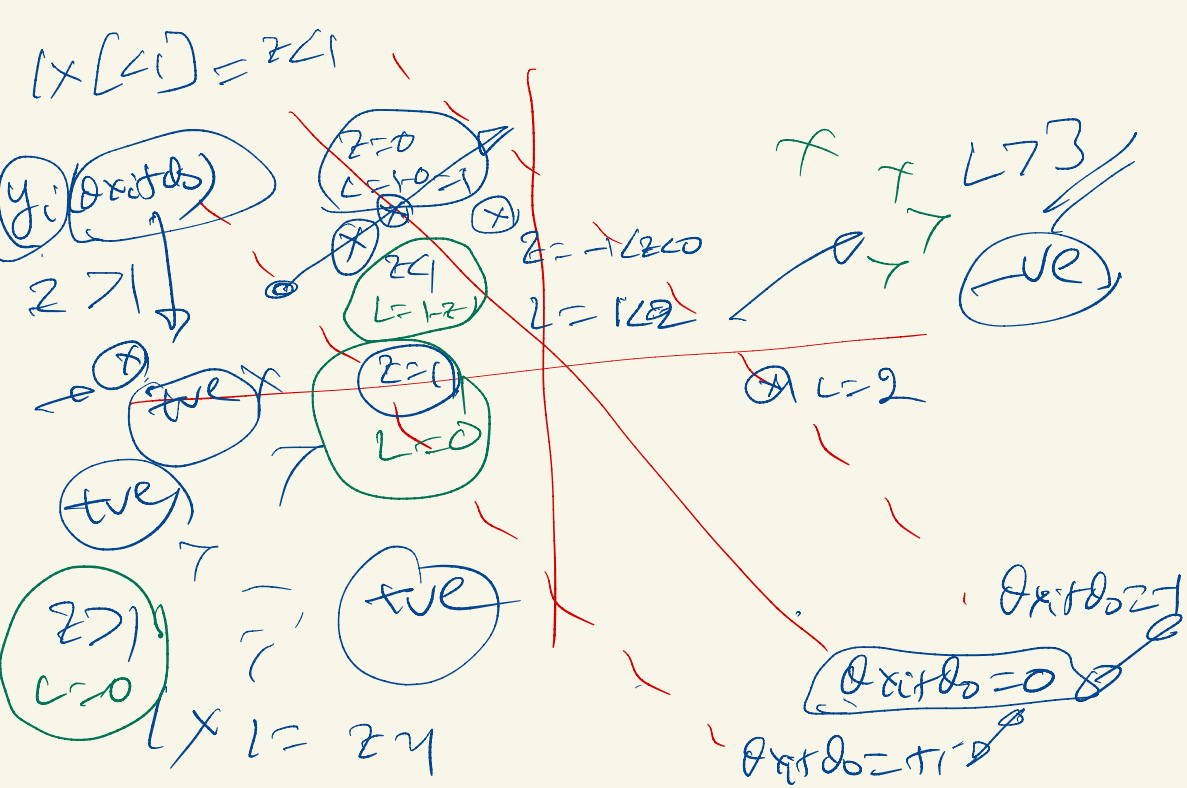
Min. $\|\theta\|^2$

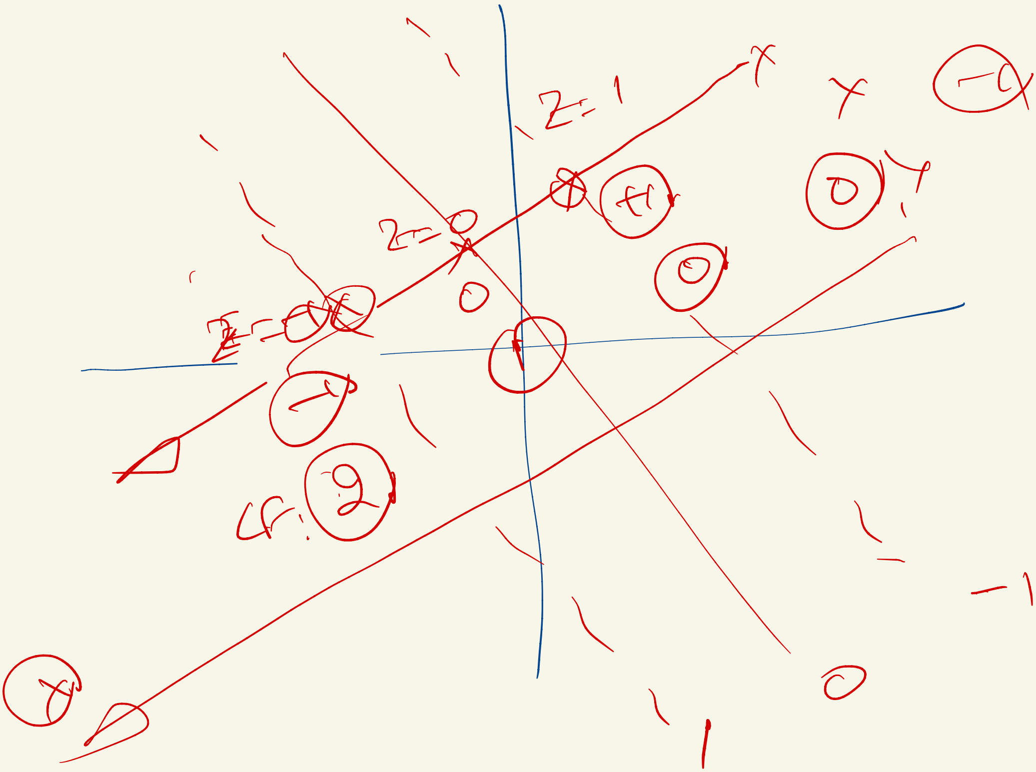
Min. cost function

$z = y_i(\theta x_i + \theta_0)$

"hinge loss"

$H(z) = \begin{cases} 1 - z & z < 1 \\ 0 & z \geq 1 \end{cases}$


z=0
z=1
z=1
z=1
L=0
-ve
-ve
-ve
-ve
L=2
-ve


z=1
z=0

Optimization

- Similar to linear regression Model

- Gradient descent

→ iterate

$$\theta = \theta - \gamma [\text{gradient value}]$$

until cost & θ converges

& Max iterations

Accuracy of the Model

Testing set → 'm' no. of points.

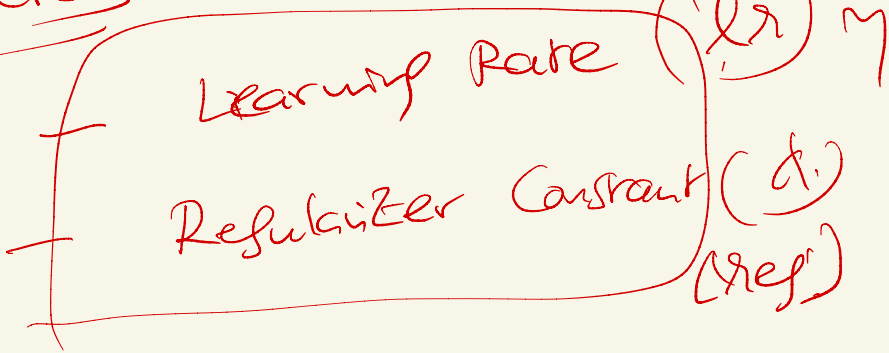
(x_i, y_i) $i = 1 \dots m$

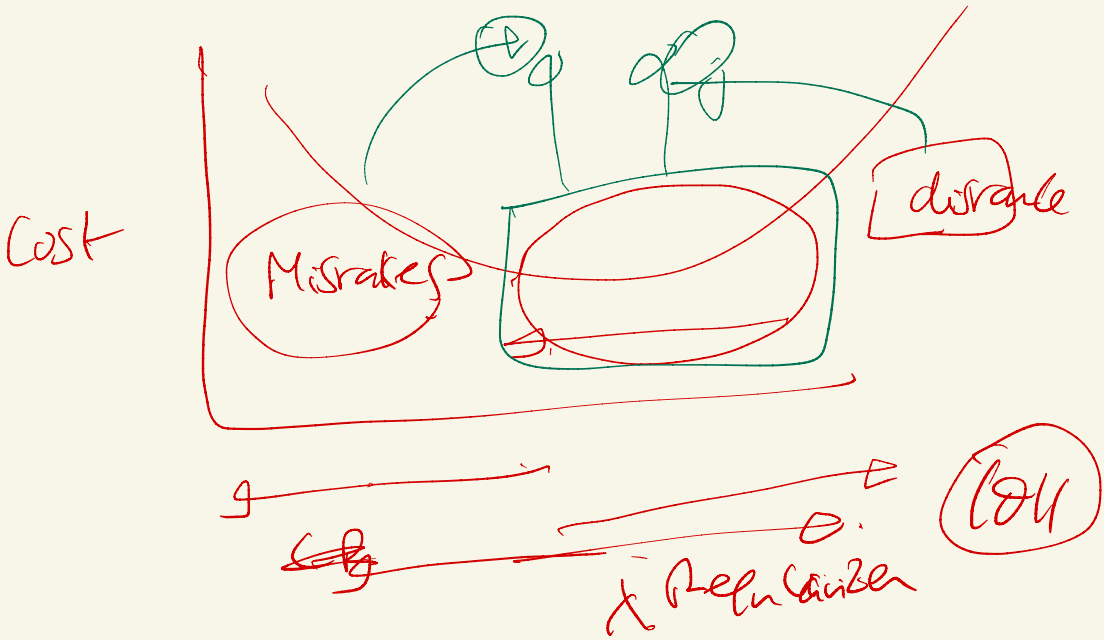
Total Produced (x_i) correct $\leq m$

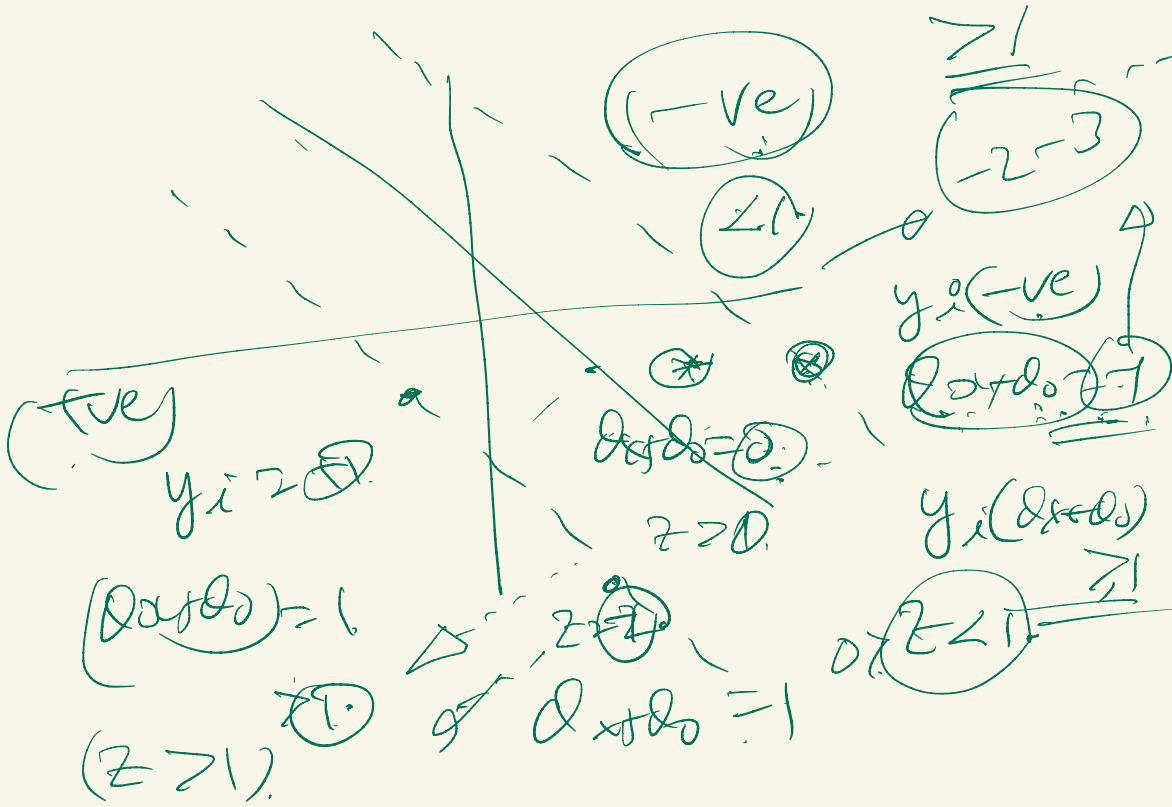
'm' = Total correct in the Testing set

Accuracy of the Model increases

Hyper Parameters in this case




Cost
Migrate
disrupt
IOU


(-ve)
z<1
z>1
-z-3
yi(-ve)
(+ve)
yi>0
(α₀x+α₀)=1
α₀x+α₀=0
z>0
yi(α₀x+α₀)
≥1
(α₀x+α₀)=1
z=1
(z>1)
α₀x+α₀=1
(z<1)

SVM gradient descent

V_1 — Motivation for SVM algorithm

V_2 — distance cost function

V_3 — Hinge loss + overall cost function + Accuracy of the model

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} \text{loss}(x_i, y_i) + \frac{\lambda}{2} \cdot \|\theta\|^2$$

$$\text{loss}(x_i, y_i) = \begin{cases} 1 - z & z < 1 \\ 0 & z \geq 1 \end{cases}$$

$$z = y_i(x_i\theta + \theta_0) \quad \}$$

$$\frac{\partial J}{\partial \theta} = \frac{\partial}{\partial \theta}\left[\frac{1}{n}\sum_{i=1}^{n} \text{Loss}f(x_i, y_i)\right] + \frac{\partial}{\partial \theta}\left[\frac{\lambda \|\theta\|^2}{2}\right]$$

$$2 \cdot \frac{\lambda}{2} \cdot \|\theta\| = \lambda \|\theta\|$$

$$\|\theta\| = \sqrt{np.dot(\theta, \theta)}$$

$$np.linalg.norm(\theta)$$

$$\frac{\partial J}{\partial \theta} = \frac{\partial}{\partial \theta}\left[\frac{1}{n}\sum_{i=1}^{n} 1 - [y_i(\theta x_i + \theta_0)]\right]$$

$x = [1\ x_i]$

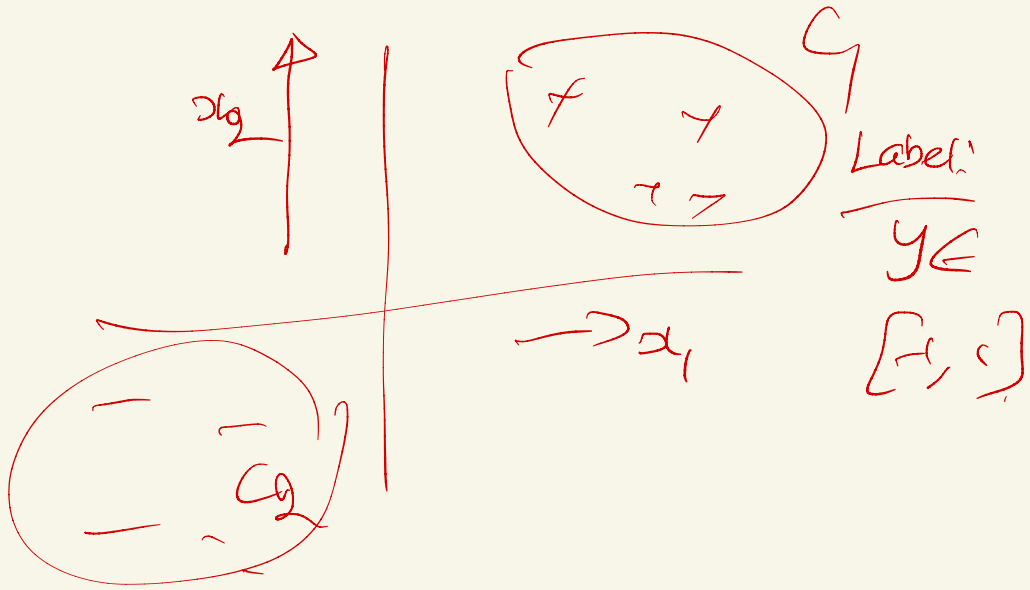
$\theta = [\theta, \theta_0]$

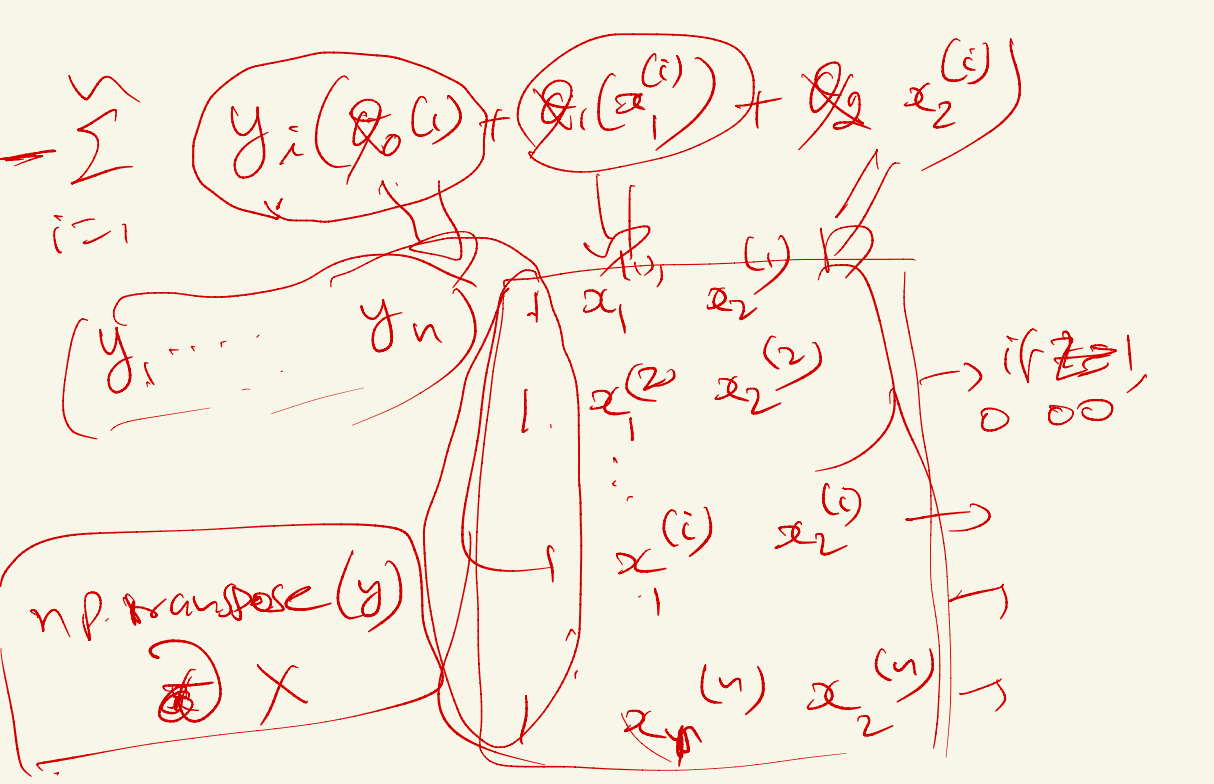
$$= \frac{1}{n}\sum_{i=1}^{n} - y_i(x_i)$$

i is the point.

$\theta_1\ \theta_2\ \theta_0$

$$\nabla \quad y_A = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}_{n\times 1} \quad \underbrace{\begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} \\ \vdots & & \\ 1 & x_1^{(n)} & x_2^{(n)} \end{bmatrix}}_{n\times 3}$$


x_2
x_1
C_1
C_2
Label:
$y \in$
$[-1, 1]$

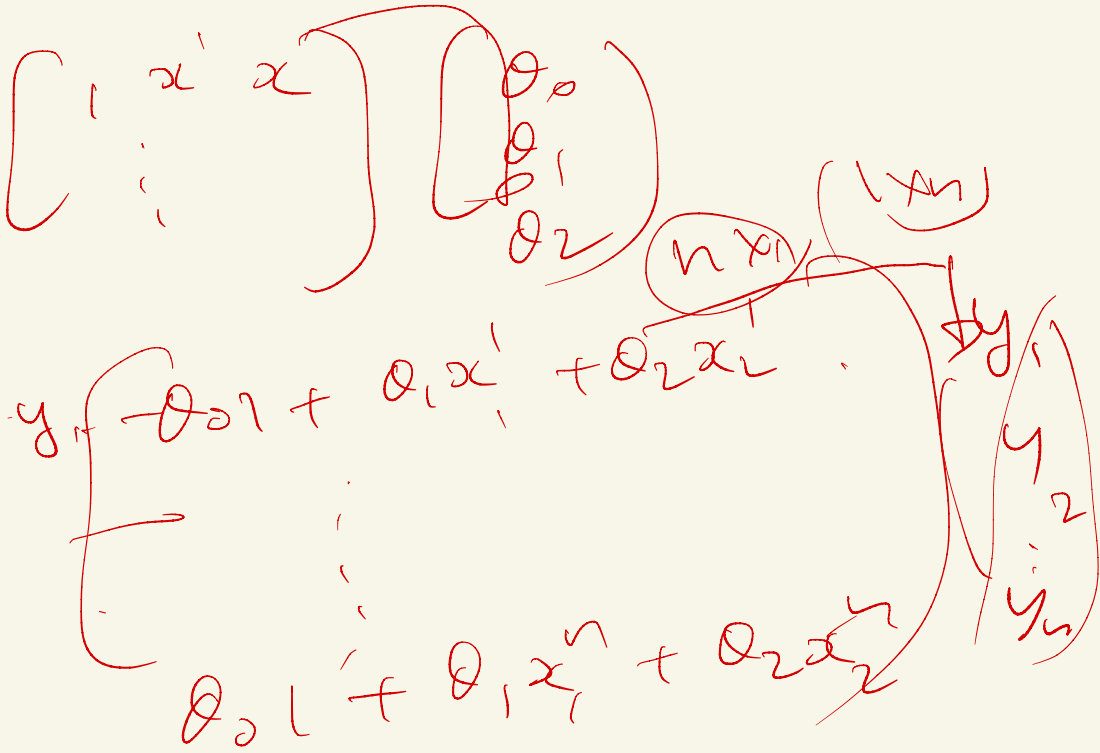

$\sum_{i=1}^{n}$
$y_i(\theta_0 x_0^{(i)} + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)})$
$y_1 \ldots y_n$
$x_1^{(1)}$ $x_2^{(1)}$
$x_1^{(2)}$ $x_2^{(2)}$
$x_1^{(i)}$ $x_2^{(i)}$
$x_n^{(n)}$ $x_2^{(n)}$
np.transpose(y)
X

$$\frac{\partial J}{\partial \theta} = -\frac{1}{n}\left[\text{np.transpose}(y)\, a X\right]$$

Assumption all the hinge losses are > 0:

how do we do this if there are NO losses for some points

i.e. $\text{Loss}(x_i, y_i) = 0$ ✗ (1-2)



V1 – Perceptron weakness Motivation for SVM

V2 – Motivation for distance function SVM intuition &

V3 – Hinge loss Cost function.

V4 – Gradient Calculation for J(θ) including numpy implementation this

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} \text{loss}(x_i, y_i) + \frac{\lambda}{2} \cdot \|\theta\|^2$$

$$\text{loss}(x_i, y_i) = \begin{cases} 1 - z & z < 1 \\ 0 & z \geq 1 \end{cases}$$

$z = y_i(x_i\theta + \theta_0)$ →

$(y_i \times \theta_i)$

$x' = [1, x]$

$\theta = [\theta_0, \theta]$

$$\frac{1}{n}\sum_{i=1}^{n} \text{Loss}(x_i, y_i)$$

$$z = y_i(\theta x_i + \theta_0)$$

$$\boxed{z = y_i \theta x_i}$$

$$\sum_{i=1}^{n} (y_i \theta x_i)$$

$$\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix}$$

$y_1 \underline{\theta^T x_1} + y_2 \underline{\theta^T x_2} + \cdots$

$y_1 \left[\theta_0 + \theta_1 x_1^{(1)} + \theta_2 x_2^{(2)}\right] + \cdots$

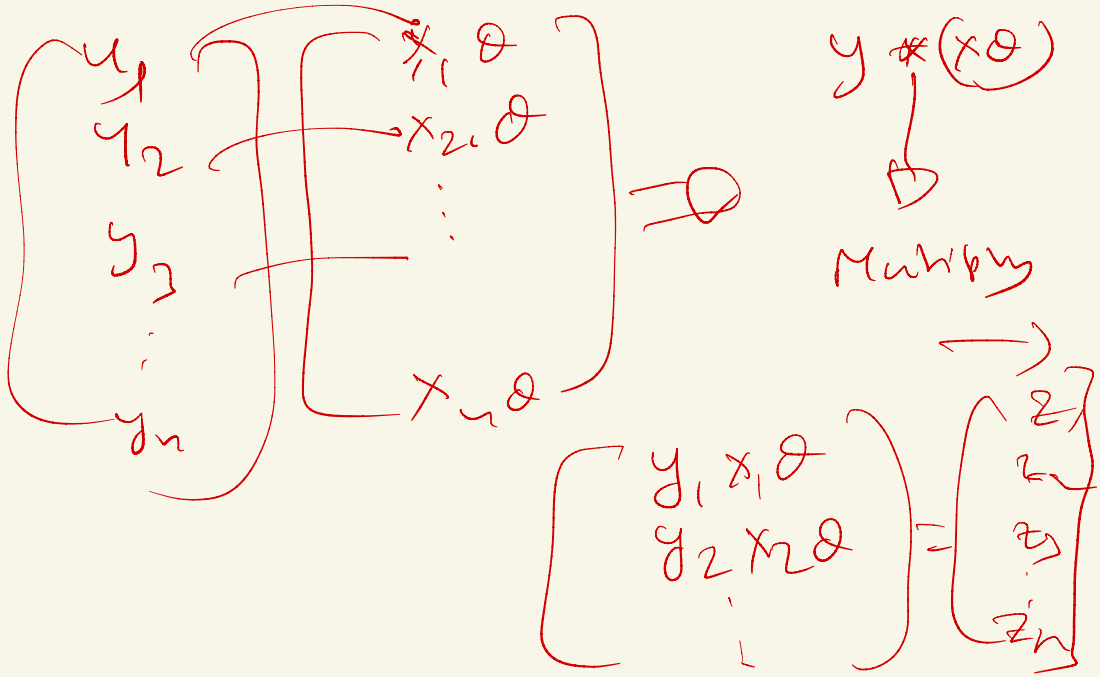
$+ \left(y_j \theta_0 + \theta_1 x_1^{(j)} + \theta_2 x_2^{(j)}\right)$

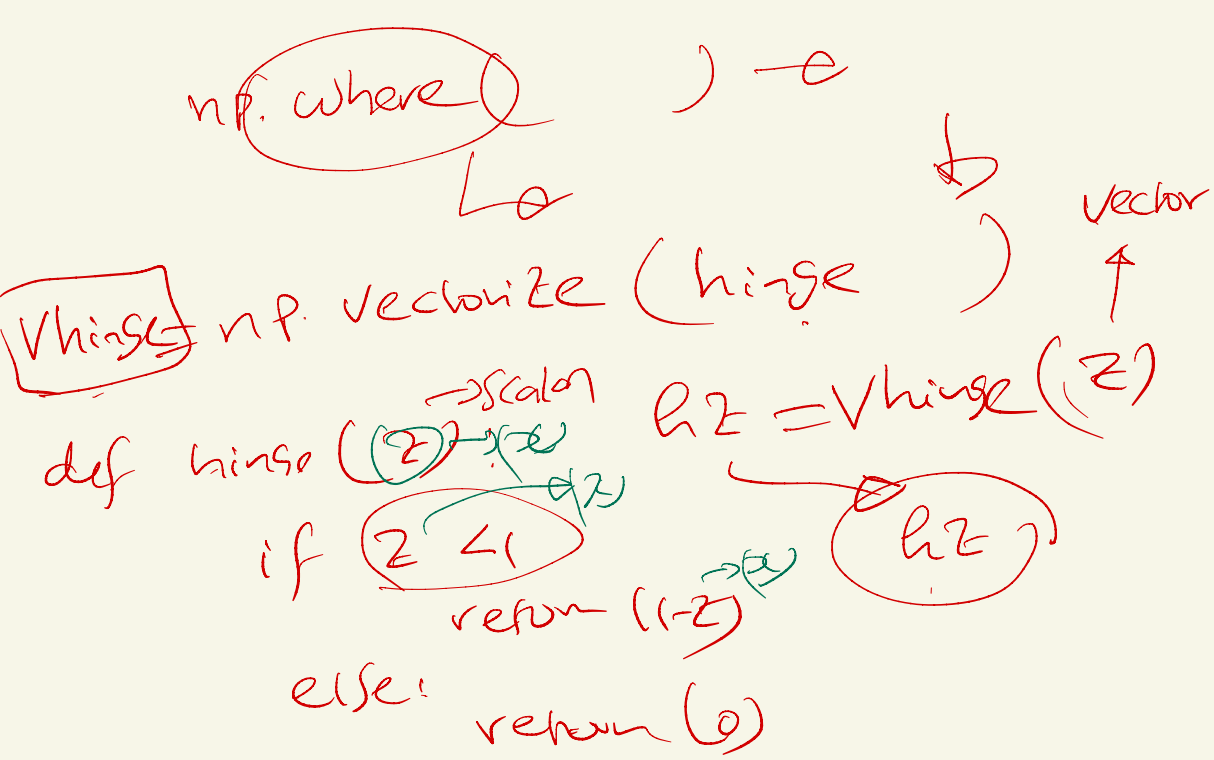
$\rightarrow (X \cdot \theta)$

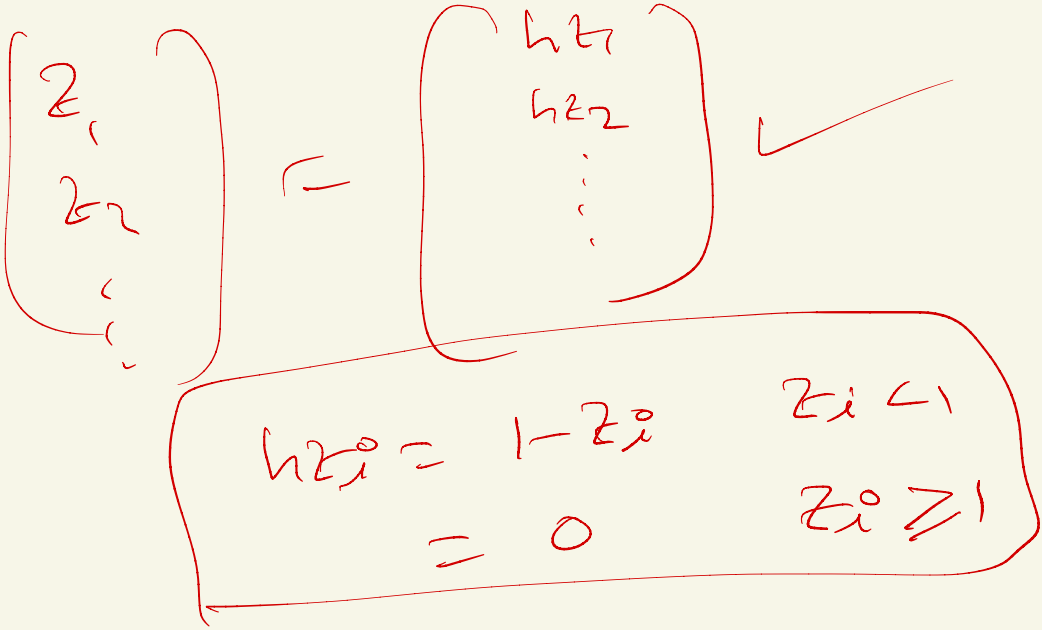
Multiply

$\begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} \\ \vdots & & \\ 1 & x_1^{(j)} & x_2^{(j)} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}$

rows


y * (x θ)
Multiply


np.where()
np.vectorize(hinge)
vhinge
vector
def hinge(z):
scalar
if z<1:
return (1-z)
else:
return(0)
hz = vhinge(z)
hz


z_1
z_2
h_{z_1}
h_{z_2}
$h_{z_i} = 1 - z_i \quad z_i < 1$
$= 0 \quad z_i \geq 1$

hz.mean

$\frac{1}{n} \sum loss(x_i, y_i)$

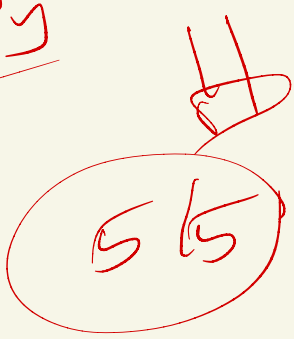
$hz.mean + \left(\frac{\lambda}{2}\right) . np.dot(\theta, \theta)$

$\frac{\lambda}{2} \|\theta\|^2$

Good Luck

SUM.Py

Submission



Points !!

V. Good