3/10/2023 Tuesday

Lecture No. 2

→ Two Views of AI.

1. AI agents
2. AI tools.

→ An intelligent agent:

- Perception
- Robotics → performing Actions
- Languages
- Knowledge → Procedural Knowledge
  → Declarative Knowledge
- Reasoning → Decision Making.
- Learning → Overtime Learning
  ↓
  Requires thousands of examples to Train

Note: Few shot Learning
↳ Chat gpt.

→ AI Tools:

- Poverty Prediction
- Saving Energy by Cooling Data Centers.
- Authentication → Deep Face login

Limitation:

— Bias in machine translation
↳ Due to collection of Data from Society.

Society → Data → Predictions.

- fairness in criminal risk assessment
  ↳ Algorithm will decide either to give bail or not on base of previous data.
  Exp: Racisim - Black & whites

Summary:

- AI Agents: Achieving human level intelligence, still very far.
- AI Tools: Need to think carefully about real world concequences.

How do we solve AI Tasks?

Paradigm

Modeling (real world to mathematical object)

Inference
↓ (is a module that answers our questions based on model)

Model Learning
Model without parameters
+
Data
↓ (Learning
Model with parameters

Generalization: When algorithm predicts beyond the data set given to it.
Exp: Heart patients.

Given Data set must be rich also Model / Skeleton must be good

→ Reflex - Based Models

- Most common models in machine learning
- Fully feed-forward (No Back Tracking)

→ State - Based Models

- Exp: Chess, Pac-Man
- Robotics: Motion Planning.
- Types
  - Search Problems: (You control Everything)
  - Markov Decision Processes: Randomness ← Against Nature (e.g Blackjack)
  - Adversarial Games: Against Opponent (e.g Chess)

→ Variable-Based Models:

- Constraints satisfaction Problems
  Hard Constraints e.g Sudoku, Scheduling

- Bayesian Networks
  (Soft Dependencies)

→ Logic-Based Models.

Exp : Virtual Assistant.

6/10/2023 Friday

Lecture No. 3

→ Reflex Models:

=> Machine Learning

- Linear Predictors
- Deep Learning
  - ↳ Loss Minimigator
  - ↳ Stochastic Gradient Descent.

→ Classification Problems → Discrete Output. e.g: Spam Emails

→ Regression Problems.
  ↳ Output continuous.
  e.g: Predicting Temp, House prices.

→ Spam Classification : (App)

Input → x = email message

Output: $y \in \{Spam, Not\ Spam\}$

Objective: Obtain a predictor f

processes and Then categorize, either spam or not (Example)

$x \rightarrow [f] \rightarrow y$

- We need a measure to classify either my objective is good or not.
- Objective function could be a loss; How many mistakes were made, so in order to reduce it.
- Function is basically a model / skeleton of a real world scenario.

→ Binary Classification:

$x \rightarrow [f] \rightarrow y \quad y \in \{+1, -1\}$

↘ Output Discrete

→ Regression (House price prediction)

$x \rightarrow [f] \rightarrow y \in \mathbb{R}$

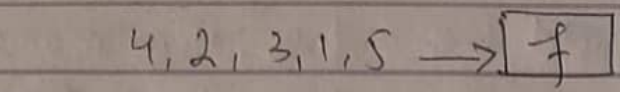
↘ Output Continuous.

→ Multi-class classification:
Output is discrete but classes can be more than one.

$x \rightarrow [f] \rightarrow y \in \{0, 1, 2, 3\}$

Exp: Healthy ← mandatory
Covid
Lung Cancer
Dengue

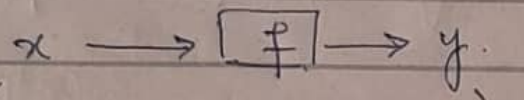
→ Ranking:



Example:

Teams Ranking
University Ranking on the basis of certain criteria.

→ Structured Prediction:



Diff words input

Those words in proper grammatical order.

- Data

Machine Learning

→ Supervised Predictors → (Target output is provided)
→ Unsupervised Predictors → (Target output is not provided)

Exp: (x, y)

x: Input
y: Target Output / Ground truth

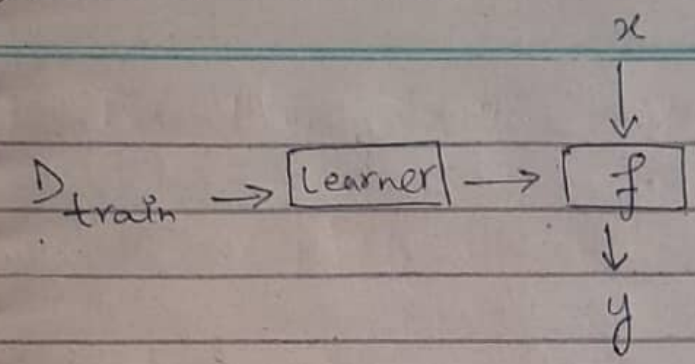
Data Set → List of Examples (Instances / Data points)

$D_{train} = \{("---", +1), ("----", -1), ...\}$

↓

contains set of examples.

→ Loss Minimization Framework



// Based on data we want to construct function f. which requires some parameters. So inorder to find some parameters for predictor functions we need Learner.

// Data is problem dependent.

// Function [f] depends upon features of Data. Function captures These features from Data ~~and~~ to enrich itself.
we focus on relevant features.

- Feature Extraction:

Exp: x = "abc@hotmail.com"

predict y : whether a string x is an email address

ϕ

Feature Name	Feature value.
• Length >10	1 or 0 True False
• fraction of alphabets	0.85
• contains @	1
• ends .com	1

⇒ ϕ → is a function that will extract features.

Data set

	length > 10	fraction of Alphabets	contains @	end.com
1.				
2.				
3.				

- Feature Vectors:

$$x = \text{"\_\_\_\_\_"} = \begin{bmatrix} 1 \\ 0.85 \\ 1 \\ 1 \\ 0 \end{bmatrix} \begin{matrix} \rightarrow \phi_1(x) \\ \rightarrow \phi_2(x) \\ \vdots \\ \rightarrow \phi_5(x) \end{matrix} \in \mathbb{R}^5$$

$$\phi(x) = [\phi_1(x), \phi_2(x), \phi_3(x) \ldots\ldots \phi_d(x)]$$

$$\phi(x) \in \mathbb{R}^d$$

- Weight Vector:

for each features j, we have a real number w_j, · represents contribution of the feature j.

$$w \begin{bmatrix} 1.2 \\ -0.8 \\ 3 \\ 1 \\ 0.1 \end{bmatrix} \in \mathbb{R}^d \quad \therefore d = 5$$

→ weight of feature.

$\phi(x)$ depends on x.
But w doesnot directly depends upon x

. ——— .

10/10/2023 Lecture No. 4 Tuesday

→ Reflex Models.

• Linear Predictors:

$x \longrightarrow [f] \longrightarrow y$

$x \longrightarrow \phi \longrightarrow \phi(x) \in \mathbb{R}^d$

F.E (feature Extractor)

$[\phi_1(x), \phi_2(x), \cdots \phi_d(x)]$

$\phi_i(x) = (\underline{\text{feature Name}}, \text{feature value})$

↓

is imp than feature name as it is required for computation.

→ Feature Vector:

$$\begin{bmatrix} \phi_1(x) \\ \phi_2(x) \\ \vdots \\ \phi_d(x) \end{bmatrix}$$

→ Weight Vector:

$$w \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}$$

Weights / Parameters

Assume

$\phi(x) \in [0, 1]$

$y \in \{+1, -1\}$: Labels

(+1: email Add, -1: Not email Add.)

⇒ Features are Binary

- if weight w_i is +ve, $\phi_i(x) = 1$ (Positive prediction)
  ⇒ +ve class label

- if Weight w_i is -ve, $\phi_i(x) = 1$
  ⇒ -ve class label

Scores

$$\text{Score} = w \cdot \phi(x)$$

$$= \sum_{i=1}^{d} w_i \cdot \phi_i(x)$$

$$w \quad \phi(x) = \begin{bmatrix} 1.2 \\ 0.6 \\ 3 \\ 2.2 \\ 1.4 \end{bmatrix}, \quad \phi(x) = \begin{bmatrix} 1 \\ 0.85 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Score = (1.2)(1) + (0.6)(0.85) + 3(1) + (2.2)(1) + (1.4)(0)

= 4.5

- +ve Value of Score Represent +ve classification (+1).
- Greater the value The more confident we are That it is positive

→ Case of Binary Features:

Let $w = \begin{bmatrix} 1.2 \\ 3 \\ -2.2 \\ 1.4 \end{bmatrix}$ $\phi(x) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$

No contribution ←

+ve feature ~~with~~ +ve weight → +ve classification

+ve feature with -ve weight → -ve classification

→ Linear Predictor:

$\phi(x) \in \mathbb{R}^d$, $w \in \mathbb{R}^d$

- Linear (Binary) Classification.

$f_w(\overset{\text{string input}}{\uparrow x}) = \text{sign}(w \cdot \phi(x))$

parameter ↑

$$= \begin{cases} +1 & \text{if } w \cdot \phi(x) > 0 \\ -1 & \text{if } w \cdot \phi(x) < 0 \\ ? & \text{otherwise. (un decisive)} \end{cases}$$

- Geometeric Interpretation:

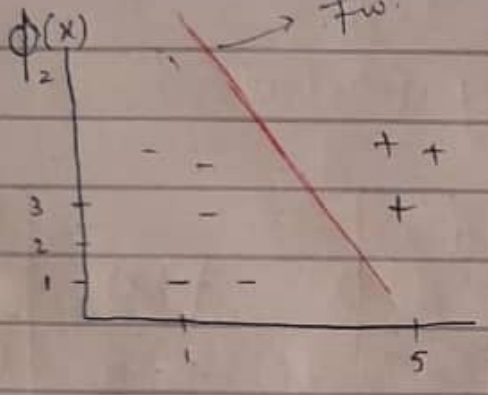
  - f_w is a binary classifier
  - It defines a hyperplane with normal vector w.

$\phi(x) \in \mathbb{R}^2$

Hyperplane

Line
$\phi(x) \in \mathbb{R}^2$

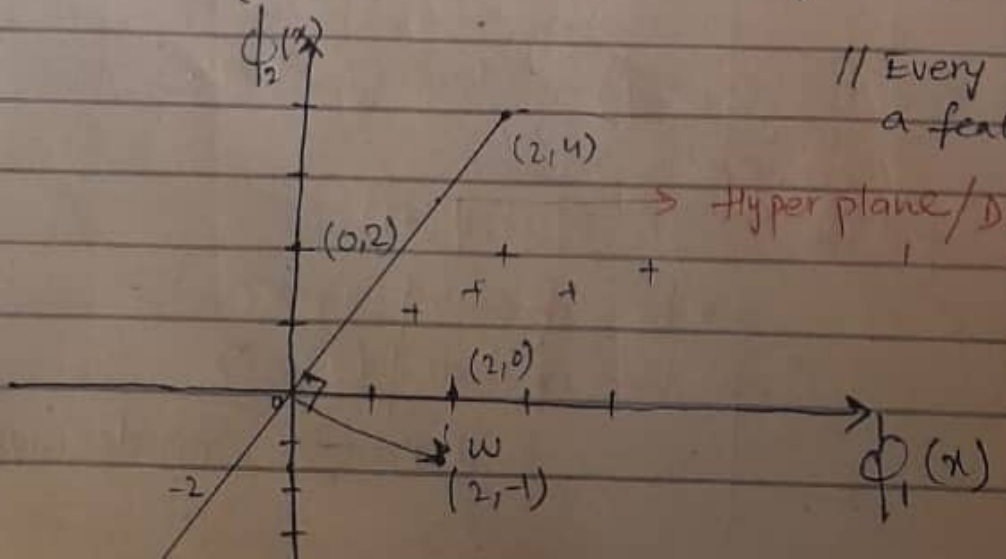
Plane.
$d \geq 3$
$\phi(x) \in \mathbb{R}^d$



$w = [2, -1]$

$\phi(x) \in \{[2,0], [0,2], [2,4]\}$

$D_{train} = \{(x_1, y_1), (x_2, y_2) \dots\}$



// Every point is a feature vector.

→ Hyperplane / Decision Boundary

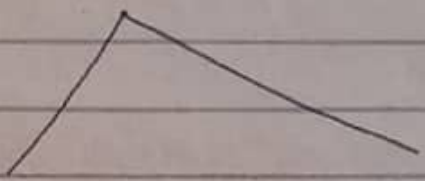
$2(+) + 2(-1)$

$w \cdot \phi(x) = 4 + 0 = 4$

$w \cdot \phi(x) = 0 - 2 = -2$

$w \cdot \phi(x) = 4 - 4 = 0$

→ Loss Minimization Frame work.



Optimization Problem

Model { what we want to compute

Optimization Algorithm (How to compute? (Learning.)

ground truth

Loss (x, y, w)

$y' = f_w(x)$

Loss is function in which we give value, Traget value.

Classification → miss classification matters (los

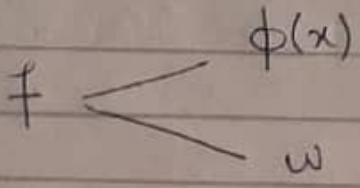
Regression → diff matters (loss).

exp: house price, Age.

17/10/2023 Tuesday

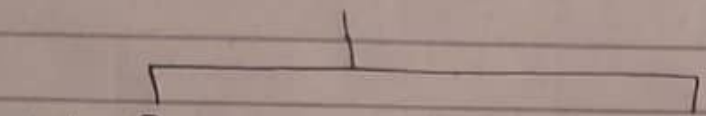
Lecture No. 5

→ Reflex Models: (Cont.)



Score = $w \cdot \phi(x)$

→ Loss Minimization Frame Work.



opt. Problem (Model)

opt. Algorithm. (Learning)

• Loss function: ground Truth

$$Loss(x, y, w) = ?$$

• Margin:

= Score * y

= $w \cdot \phi(x) * y$ → Margin

+ve +ve } → +ve (+1) → classification.

-ve -ve } → +ve (-1)

-ve +ve } → -ve

+ve -ve } → -ve

=> if margin > 0 Then our prediction is correct.

=> if margin < 0 Then our prediction is wrong.

Example :

$$W = [2, -1]$$

$$x = \phi(x) = [2, 0] \quad , y = -1$$

$$y' = f_w(x) = \text{sign}(w \cdot \phi(x))$$
$$= \text{sign}(4 - 0)$$
$$= +ve \Rightarrow +1$$

$$\Rightarrow \text{Margin} = w \cdot \phi(x) * y$$
$$= 4 * (-1) = -4.$$

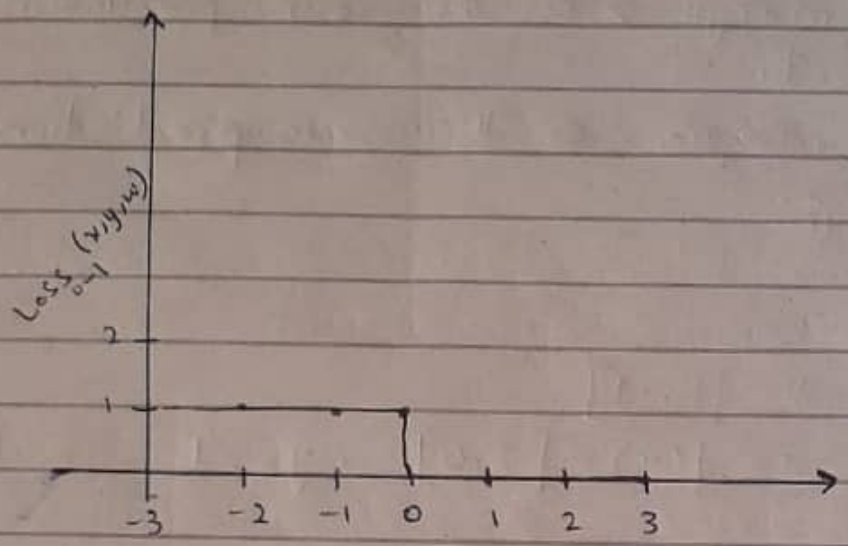
- Loss Function : (Classification Problems).

0—1 Loss

$$\text{Loss}_{0-1}(x, y, w) = 1\left[f_w(x) \neq y \right]$$

$$= 1\left[(w \cdot \phi(x) * y \leq 0 \right]$$

Loss is 1 when prediction is wrong i.e Margin is less than equal to 0. otherwise loss is 0.

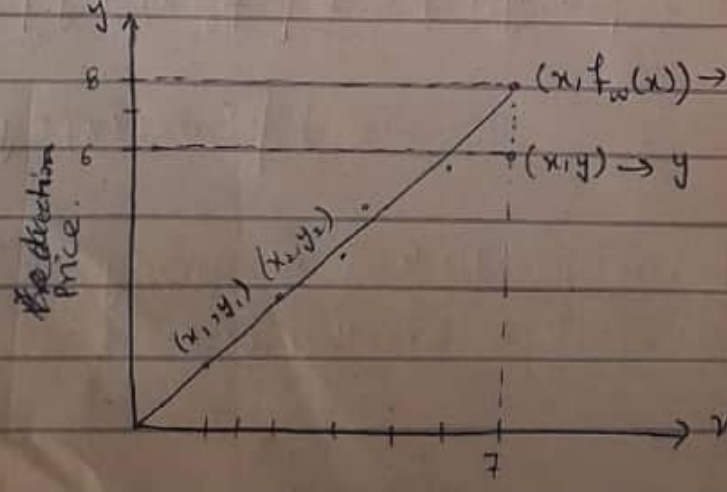


→ Linear Regression:

$y' = y$ can't be achieved
bcz $y \in \mathbb{R}$.

⇒ So 0-1 loss can't be used in Regression Problems.

⇒ Can be used in Classification Problems.
$y' = y$ can be achieved. $y \in$ Discrete.



Size of house

⇒ Residual:

Diff b/w our prediction and ground truth.

$$\text{Residual} = (w \cdot \phi(x) - y) = (f_w(x) - y)$$

• Regression Loss Function:

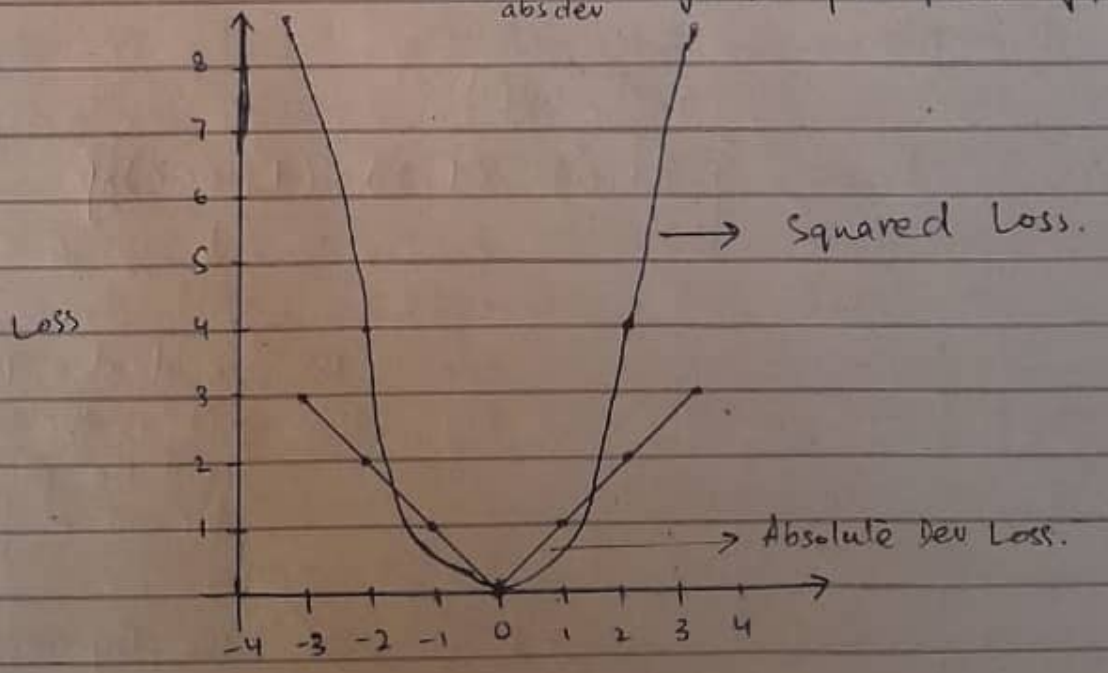
$$f_w(x) = w \cdot \phi(x).$$

– Squared Loss:

$$\text{Loss}_{sq}(x, y, w) = (w \cdot \phi(x) - y)^2$$

– Absolute Deviation Loss:

$$\text{Loss}_{absdev}(x, y, w) = |w \cdot \phi(x) - y|$$



$$\text{Residual} = w \cdot \phi(x) - y.$$

=> Loss Minimization Framework.

Loss $(x, y, \boxed{w})$ → Loss on a single example x

↑

Minimizing Training Loss → D_{train}

Train Loss (w)

$$= \frac{1}{|D_{train}|} \sum_{(x,y) \in D_{train}} \text{Loss}(x, y, w)$$

↑ No. of Training Examples.

Minimizing Train Loss (w) By modifying $w \in \mathbb{R}^d$

Example:

input output.

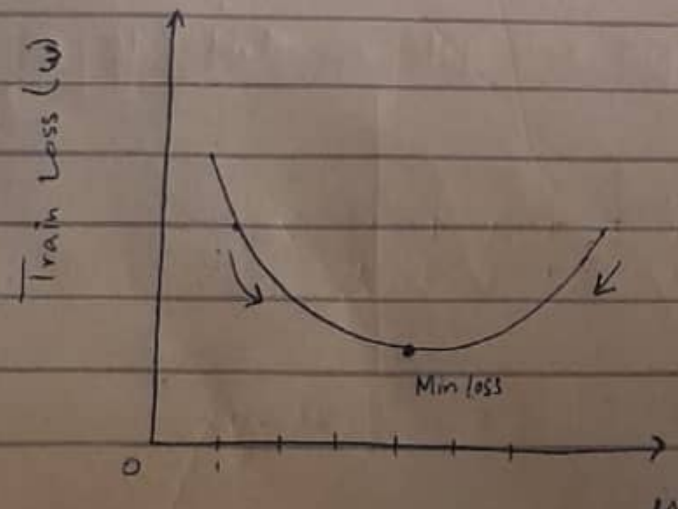
$D_{train} = \{ (1,0), (1,2), \underline{(1,1000)} \}$

↓

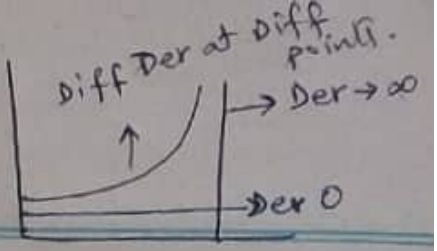
Outliar

Assume Residual = 997

it is too high will be problematic for our model.



→ we try move towards min loss value of w.



. Gradient : / Derivative.

Rate of Change.

∇ Train loss (w)

↳ tells us the direction that increases the lost the most.

→ Gradient Descent : (opt. Algorithm).

Objective: Minimal Train Loss (w)

Initialize $w = [0, 0, \ldots 0]$

For $t = 1, 2, \ldots T$

$w = w - \eta \nabla$ Train Loss (w).

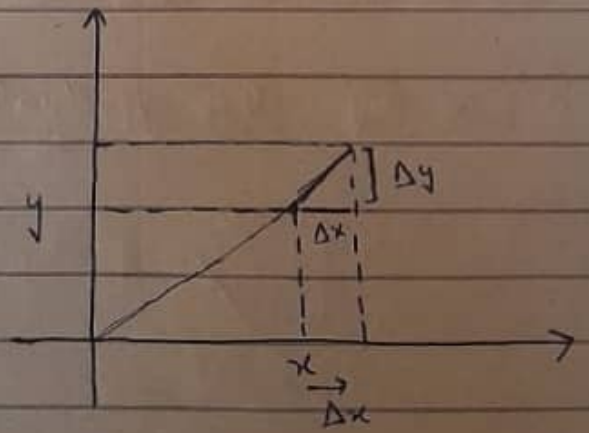
↳ step size ≤ 1

We are iteratively changing the value of w.

[0,1,∞]

[T → no. of iteration

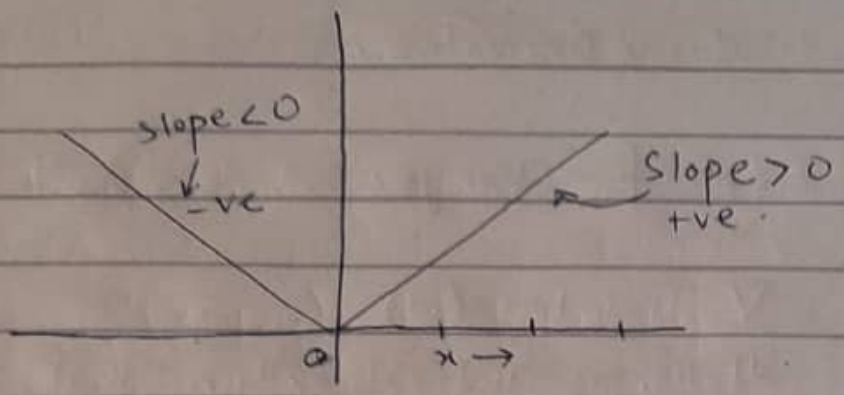
η → eta (step size).]

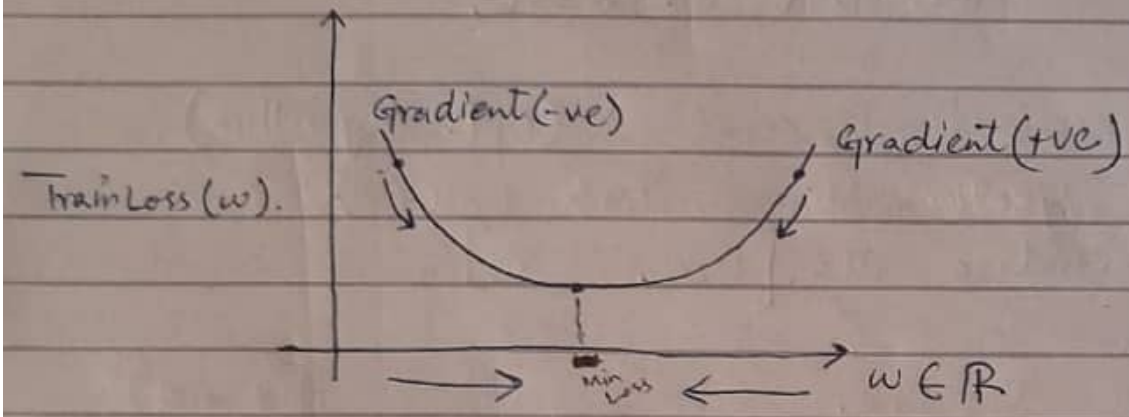


slope = $\frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$

or

Der



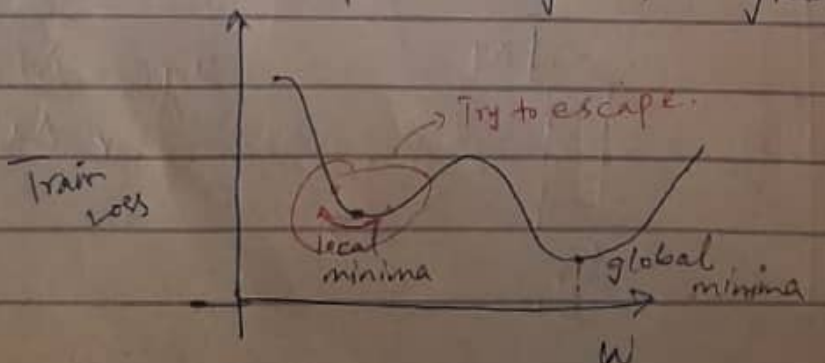


→ Small values of step size will make ~~our~~ Algo slow.

→ Sometimes we ~~take~~ take big steps to escape local minima. After escaping local minima Then we Take small steps to reach Global Minima.

→ Big steps sometimes cause back & forth movement and we can't achieve our Minima Point.

→ At start → step size Big Then gradually decrease to small step size



20|10|2023 Friday

Lecture No. 6.

→ Gradient Descent: (Cont.)

$$\text{Train Loss}(w) = \frac{1}{|D_{train}|} \sum_{i=1}^{|D_{train}|} (w.\phi(x_i) - y_i)^2$$

$$\nabla \text{Train Loss}(w) = \frac{2}{|D_{train}|} \sum_{i=1}^{|D_{train}|} (w.\phi(x_i) - y_i).\phi(x_i)$$

$\because \frac{d}{dx} x^2 = 2x^{2-1} \frac{d}{dx}(x)$

Here, we took derivative of Train Loss to Get Gradient which gives us direction.

→ Batch Gradient Descent:

↳ Slow.

↳ w will update once for all training data. examples

→ Stochastic Gradient Descent:

$w = [0, 0, 0 \ldots]$

For $t = 1, \ldots, T$

For each $(x, y) \in D_{train}$

$w = w - \eta \nabla_w \text{Loss}(x, y, w)$

↳ Fast.

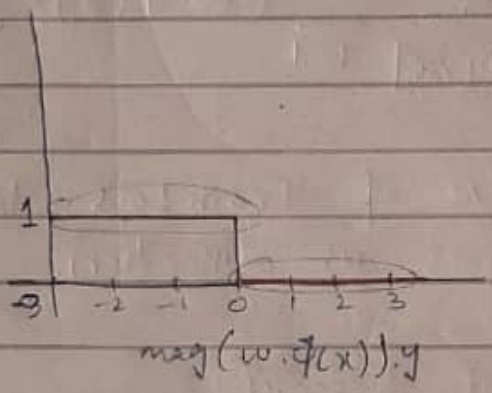
↳ Getting Data on the fly / real time.

For Classification Problems.

$$\text{Loss}_{0-1}(x, y, w) = 1\left[f_w(x) \neq y\right]$$

$$= 1\left[(w \cdot \phi(x))y \leq 0\right]$$

$$w = w - \eta \nabla_w \text{loss}(x, y, w)$$



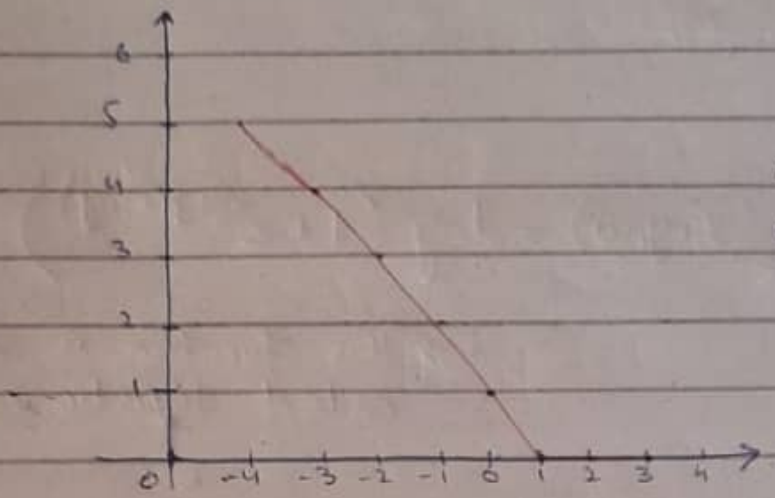
We cannot use $(0-1)$ Loss_{0-1} for classification Problems in Gradient Descent Algo because w will not update.

↳ straight line ∇ is zero. No change.

So we will use another Method.

→ **Hinge Loss:**

$$\text{Loss}_{\text{Hinge}}(x, y, w) = \max\left[1 - (w \cdot \phi(x)y, 0\right]$$



So, Hinge loss can be used in Gradient Descent instead of Loss_{0-1}

$$w = w - \eta \nabla_w \text{loss}(x, y, w)$$

$$\text{Loss}_{\text{Hinge}}(x,y,w) = \max\left[1 - (w \cdot \phi(x))y, 0\right] = \begin{cases} 0 & (w \cdot \phi(x))y \geq \\ 1-(w \cdot \phi(x))y & \text{o/w} \end{cases}$$

↓ otherwise

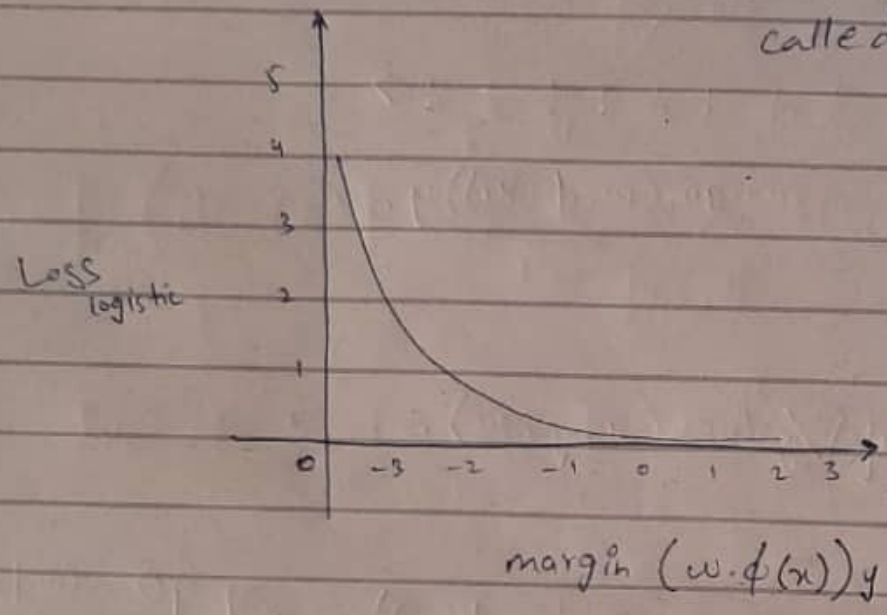
$$= \begin{cases} 0 & (w \cdot \phi(x))y \geq 1 \\ 1-(w \cdot \phi(x))y & \text{otherwise} \end{cases}$$

$$\nabla_w \text{Loss}_{\text{Hinge}}(x,y,w) = \begin{cases} 0 & (w \cdot \phi(x))y \geq 1 \\ -\phi(x)y & \text{otherwise} \end{cases}$$

→ Logistic Loss:

$$\text{Loss}_{\text{logistic}}(x, y, w) = \log\left(1 + e^{-(w \cdot \phi(x) \cdot y)}\right)$$

↳ This function is also called logistic Regression.



- Ground truth (y) → will remain same.
  So it means score ($w \cdot \phi(x)$) will change because of w.

~~or loss will not become~~

24/10/2023 Tuesday.

Lecture No. 7.

→ Python:

// google colab.com.
// python.org.

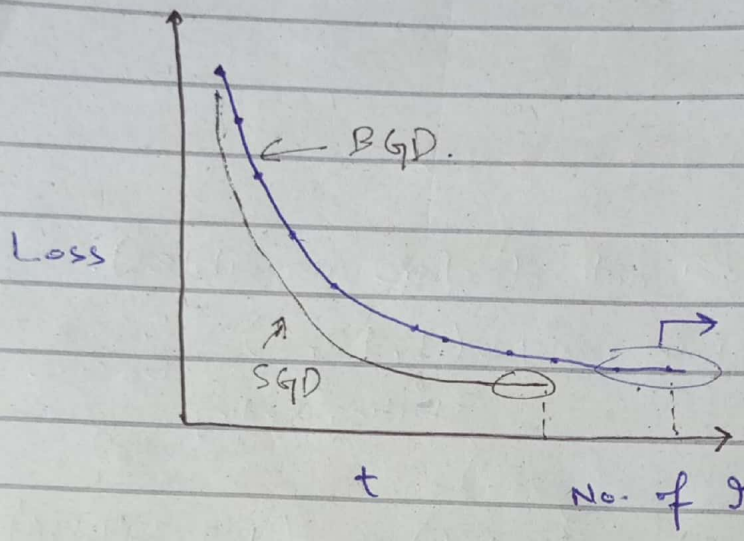
→ Operators
→ Variables (we donot declare variables).
→ Range → for i in range(1, 10, 2) → [1,3,5,7,9]
(1 → starting, 10 → Ending (not included), 2 → step size)
→ list indexing
→ Slicing.
→ Tuple → (1,2,3) + (4,5,6) = (1,2,3,4,5,6)
(+ → concatenation)
→ The + Operator
→ The * Operator
→ Dictionaries
→ Functions.

Note: Fixed formal language. Indentations imp

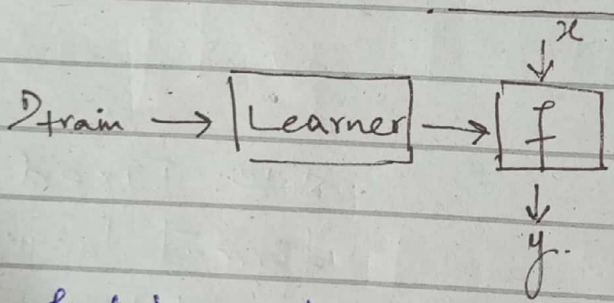
27/10/2023. Friday.

Lecture No. 8

→ Batch Gradient Descent.
→ Stochastic Gradient Descent.

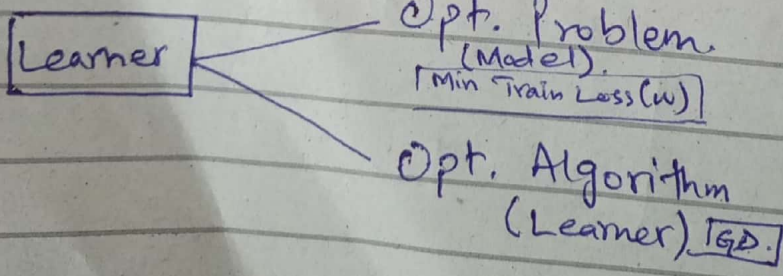


At some point, change in Loss becomes very minimum.



$f_w(x) = w \cdot \phi(x)$ ← Regression

$f_w(x) = \text{sign}(w \cdot \phi(x))$ ← Classification.



→ Regression Example:

Data:

x $\phi(x_1)$ $\phi(x_2)$	y	Sq. Loss.
[1 0]	2	$((w_1 \cdot 1 + w_2 \cdot 0) - 2)^2 = (w_1 - 2)^2$
[1 0]	4	$(w_1 - 4)^2$
[0 1]	-1	$(w_2 - (-1))^2$
[0 1]	?	

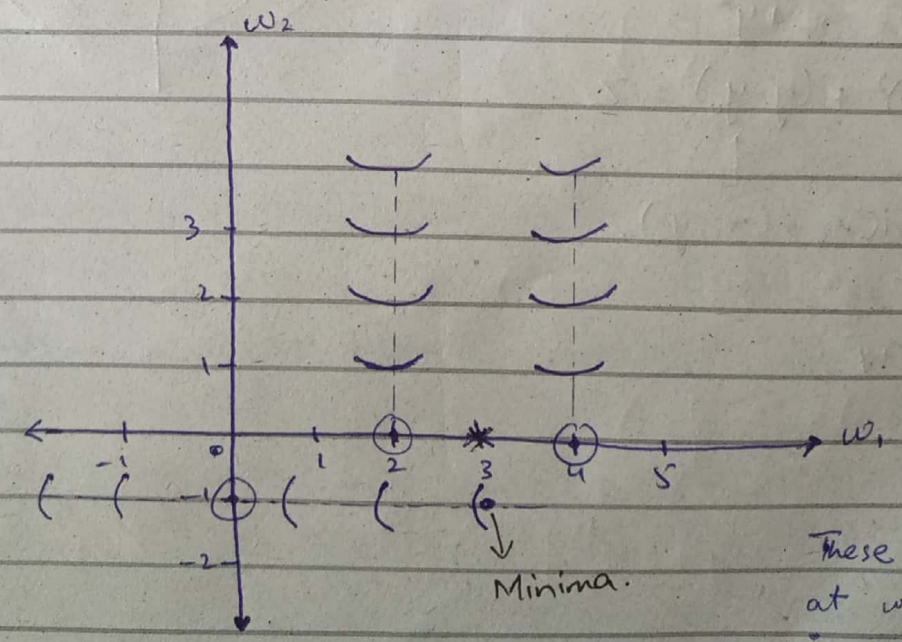
→ Sq Loss:

$$w = [w_1, w_2]$$

$$\text{Loss}(x, y, w) = (w \cdot \phi(x) - y)^2$$

→ Train Loss:

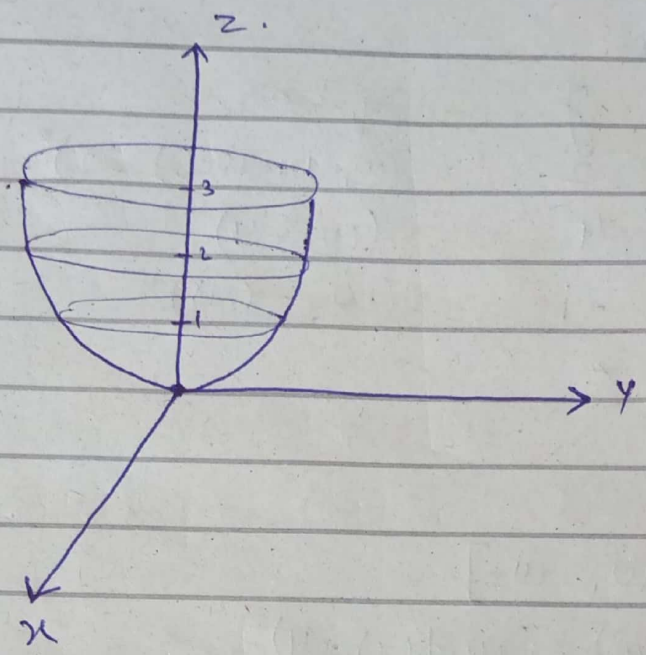
$$\text{Train Loss}(w) = \frac{1}{3}\left[(w_1 - 2)^2 + (w_1 - 4)^2 + (w_2 - (-1))^2\right]$$



These are weights at which Loss is zero for each corresponding example.

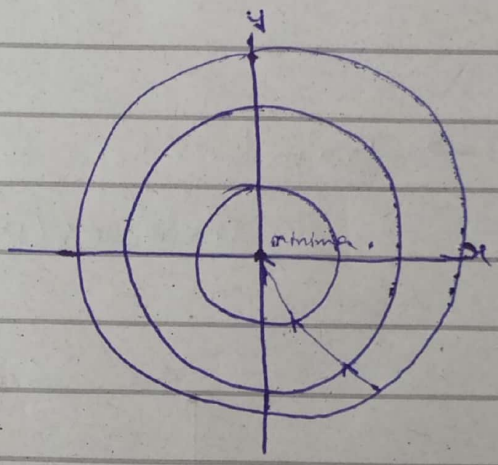
//→ Level curve?

→ Level Curves:



$f(x,y) = x^2+y^2$

$f(x,y) = z.$



$z = x^2+y^2$

$0 = x^2+y^2$

$(x-x_0)^2 + (y-y_0)^2 = z$

Center $= (x_0, y_0)$.

$1 = x^2+y^2$

$2 = x^2+y^2.$

→ Features : $\phi(x)$.

- Critical part of Machine Learning.
- with right features, we can improve the ~~of~~ Algo.

$$f_w(x) = w \cdot \phi(x) \rightarrow \text{score.}$$

Q: Can Linear predictors give us a non-Linear Decision Boundary? (Y/N).

Relationship b/w w and score → Linear

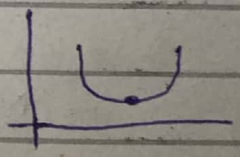
Score is ^Linear in w → Linear

" " $\phi(x)$ → Linear.

Input is it Linear?

No, Input (x) is not linear.

$f_w(x)$, Loss ← Non-linear



- Feature Template:

It is a group of features all compuled in a similar.

Example:

- Length greater than ____
- Last three character equals to ____
- Contains character ____

- Sparsity in Features: (problem).

abc @ hotmail.com.

ends	with	aaa	0
"	"	aab	0
"	"	aac	0
		⋮	⋮
"	"	com	1
		⋮	⋮

One-Hot Representation

m → no. of Char
m^3 → Combinations

Representations:

- Array $[0,0,0,\ldots 1,0,0\ldots]^{m^3}$

- Map $\{$ ends with .com : 1 , Alphabet : 0.85 $]$

↳ Save Memory.

- Hypothesis Class: (HC)

$f_w(x) = w \cdot \phi(x)$

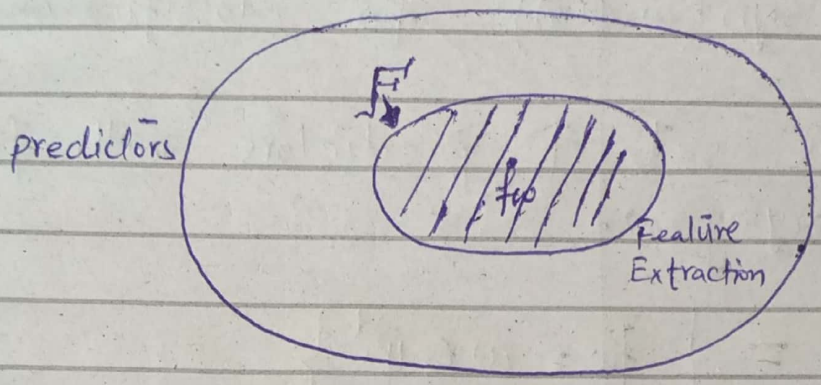
$f_w(x) = \text{sign}(w \cdot \phi(x))$

→ we study the impact of Features on prediction.

HC : It is a set of possible predictors with a fixed $\phi(x)$ and varying w.

$$\mathcal{F} = \{ f_w : w \in \mathbb{R}^d \}$$

Exp



Feature Extraction

$\phi(x)$ → Domain knowledge.

Must be significan

Learner → will choose best/imp features

F.E → $\mathcal{F}$ is based on Domain Knowledge

Learning → $f_w \in \mathcal{F}$ → Based on Data

31/10/2023.

Tuesday

Lecture No. 9.

→ Hypothesis Class:

Predictors:

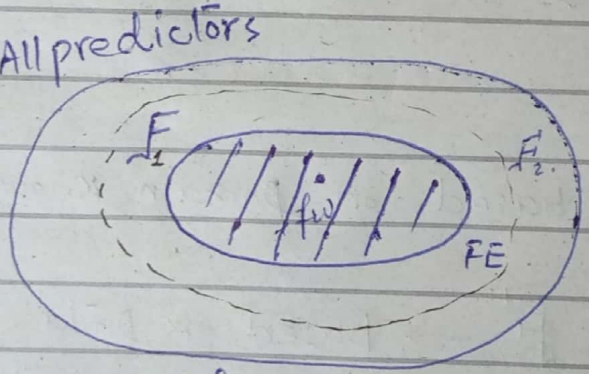
$f_w(x) = w \cdot \phi(x)$ for Regression.

$f_w(x) = \text{sign}(w \cdot \phi(x))$ for classification

- Hypothesis is set of predictors with fixed $\phi(x)$ and varying w.

$$\mathcal{F} = \{ f_w : w \in \mathbb{R}^d \}$$

- Feature Extraction + Learning:



FE → specify the Hypothesis class.
Weights are Learned through data.

- Features are very important, the more good features will result in good predictors.
- Learning also depends upon Data.
  - Data Diversity

Example:

Regression: $x \in \mathbb{R}$, $y \in \mathbb{R}$

Let's Assume $\phi(x) = x$.

// we are assuming input is feature. Not extracting here.

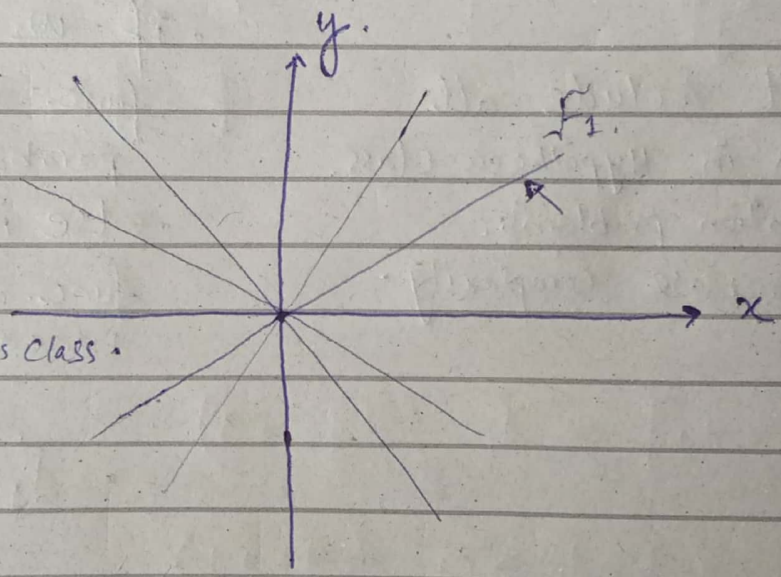
$$F_1 = \{ w_1 x + w_2 x^2 : w_1 \in \mathbb{R}, w_2 = 0 \}$$

↳ Hypothesis class representing Linear Predictors

Assumed as zero for linear predictors To remove non-linear

$\therefore y = mx + c$

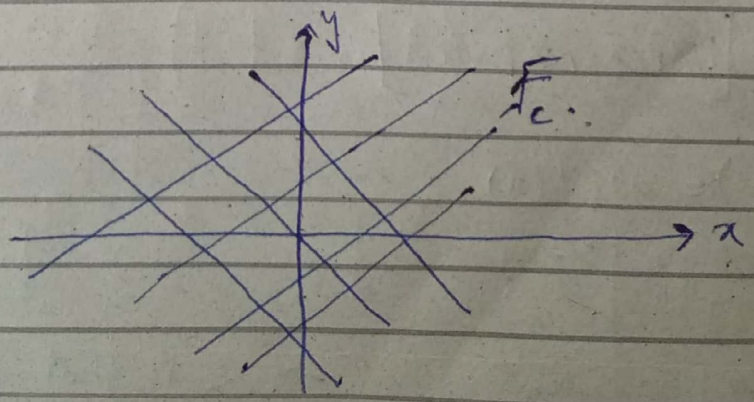
slope, y-intercept



These ← Lines are all possible predictors of F_1 Hypothesis class.

$$\Rightarrow F_c = \{ w_1 x + c : w_1 \in \mathbb{R}, c \in \mathbb{R} \}$$

↳



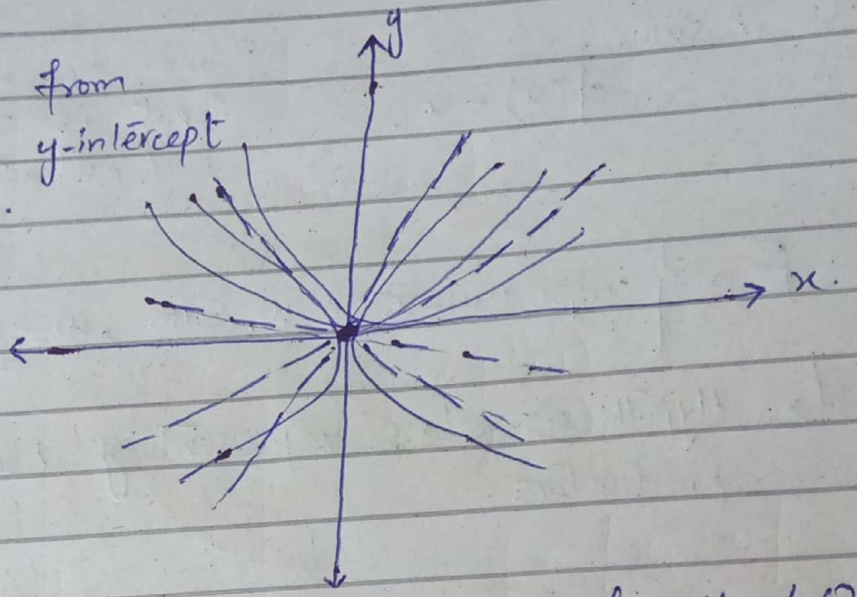
Assignment:

→ Common words.
→ Time lapse → Not a good feature.
→ Question Mark in msg.

- BBA use Balanced Data.
- Logistic Regression.

$$\mathcal{F}_2 = \left\{ w_1 x + w_2 x^2 \; ; \; w_1 \in \mathbb{R}, w_2 \in \mathbb{R} \right\}$$

All passing from origin bez y-intercept (c) is zero.



- if $w_2 \neq 0$. func. will be parabolic.
else if $w_2 = 0$. func. will be Linear. (straight).

- $\mathcal{F}_1 \subset \mathcal{F}_2$

We can't include all Predictors in Hypothesis Class.
→ Computation problems.
→ will increase complexity.

$$\phi(x) = [1, x_1, x_2, x_1^2, x_2^2]$$

$$f_w(x) = \underbrace{w \cdot \phi(x)}_{\text{score.}}$$

→ Linear in w.
→ Linear in $\phi(x)$

→ x is not Linear
↳ $f_w(x)$ is also non-Linear in x.

→ Loss function is also non-Linear.

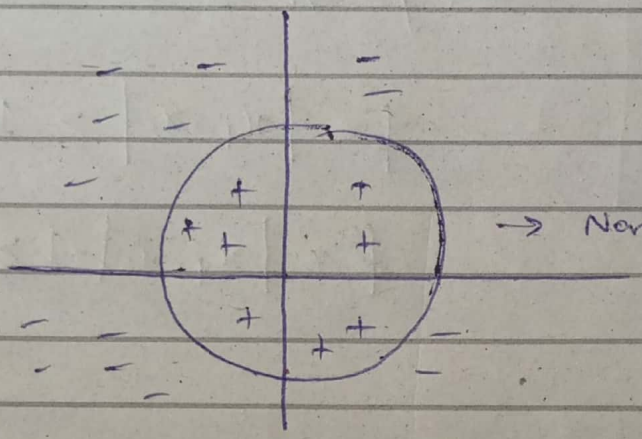
$$w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + 1.$$

Suppose $w_1 = w_2 = 0$

$w_3 = w_4 = 1.$

$$y' = f_w(x) = x_1^2 + x_2^2 + 1.$$

→ similar to circle equation. So decision boundary will be circular shaped.



→ Non-Linear Decision Boundary.

→ NEURAL NETWORKS:

↳ Learns good feature automatically from raw / crude data.

ADV:

Crude Data → NN → [FE] (Learns)

So, ~~focus~~ focus on Learning only.

Example:

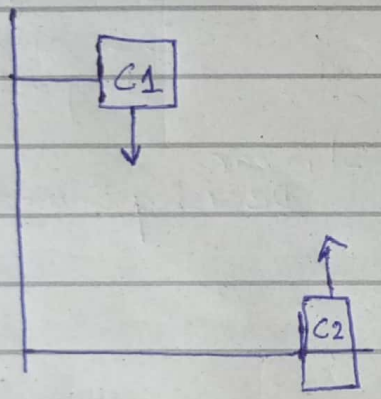
Predicting Car Collisions

Inp: Position of two incoming cars.

$$x = [x_1 \quad x_2]$$

(x_1: car 1, x_2: car 2.)

Output: whether safe ($y = +1$) or collide ($y = -1$).



$$\text{if } |x_1 - x_2| \geq 1$$

$$\text{if } |x_1 - x_2| - 1 \geq 0$$

$$|x_1 - x_1| \geq 1 \quad (y = +1)$$

$$|x_1 - x_2| - 1 \geq 0 \quad (y = +1).$$

$$y' = f_w(x) = \text{sign}(|x_1 - x_2| - 1) \begin{cases} +1 \\ -1 \end{cases}$$

Ex: Data set

x	y
$[1 \quad 3]$	$+1$
$[3 \quad 1]$	$+1$.
$[1 \quad 0.5]$	-1.

1/11/2023. Wednesday.

Lecture No. 10

→ Neural Networks:

↳ Artifical Neural Network

Example:

Input: Position of two incoming cars $x = [x_1, x_2]$

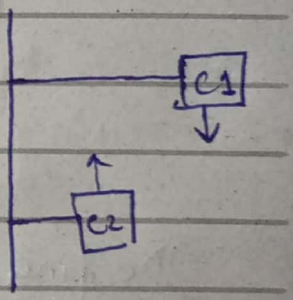
Output: whether safe ($y = +1$) or collide ($y = -1$)

$$y = \text{sign}(|x_1 - x_2| - 1 \geq 0)$$

- if car1 is far Right of car2.

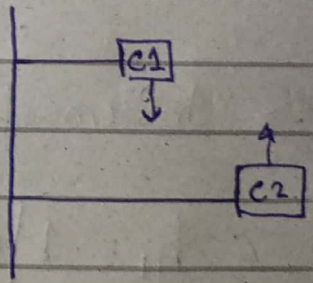
$$h_1 = 1[x_1 - x_2 \geq 1]$$

↳ Indicator Func. / Step Func.



- if car2 is far Right of car1.

$$h_2 = 1[x_2 - x_1 \geq 1]$$



- Cars are safe if atleast one is true.

$$y = \text{sign}(h_1 + h_2).$$

Data set

x	y
$[3 \ 1]$	+1
$[1 \ 3]$	+1
$[1 \ 0.5]$	-1

x	h_1	h_2	y' ← Predicted.
$[3 \ 1]$	1	0	+1
$[1 \ 3]$	0	1	+1
$[1 \ 0.5]$	0	0	-1.

→ Learning Strategy:

Define $\phi(x) = [\underbrace{1}_{\text{Bias}} \quad x_1 \ x_2]$

→ Intermediate hidden subproblems.

$h_1 = 1[V_1 \cdot \phi(x) \geq 0]$ where $V_1 = [-1 \ 1 \ -1]$

From $h_1 = 1[x_1 - x_2 \geq 0]$

$\Rightarrow -1 + x_1 - x_2 \geq 0$

$[-1 \ +1 \ -1]$

$\phi(x) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$

$h_2 = 1[V_2 \cdot \phi(x) \geq 0]$ where $V_2 = [-1 \ -1 \ 1]$

$\phi(x) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$.

Final Prediction

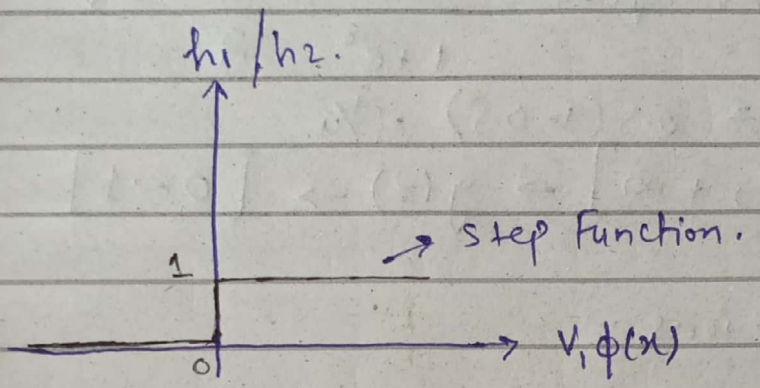
$\because f_w = (w) \cdot \phi(x)$ ($w \to V_1, V_2$)

$f_{V,w}(x) = sign(w_1 h_1 + w_2 h_2)$ $\quad w = [1 \cdot 1]$

Gradient Descent Algo

$w = w - \eta \nabla_w TL$

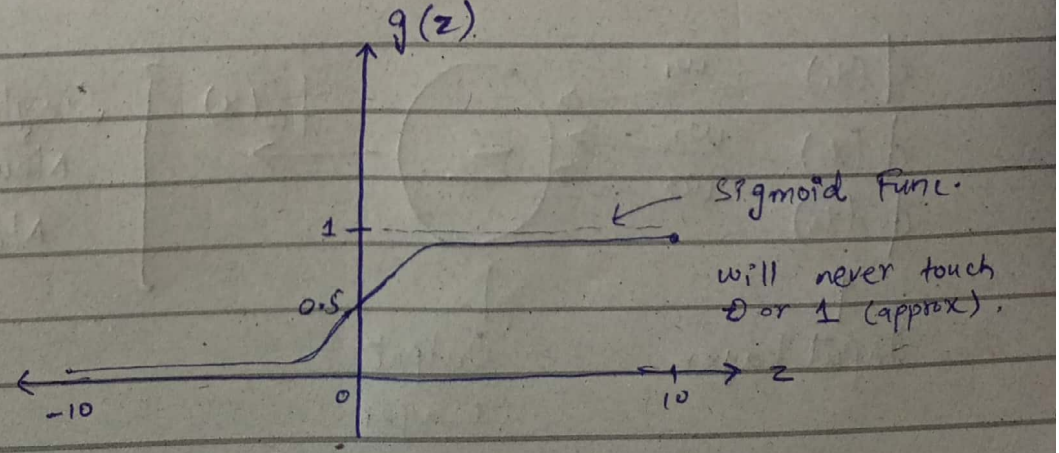
$V = V - \eta \nabla_V TL$



Slope is zero which means gradient/derivati is always zero and it will not update weights. So we cannot use it → (gradient descent). ← step func.

→ Sigmoid Function:

$$g(z) = \frac{1}{1+e^{-z}} \quad , \quad g'(z) = g(z)(1-g(z))$$



Let $z = 10$ $\quad g(z) = \frac{1}{1+e^{-10}} \approx 1$

$\Rightarrow g'(z) = 1(1-1) \approx 0$

Let $z = -10$ $\quad g(z) = \frac{1}{1+e^{10}} \approx \frac{1}{\infty} \approx 0$

$\Rightarrow g'(z) = 0(1-0) \approx 0$

Let $z = 0$, $\quad g(z) = \frac{1}{1+e^{0}} = \frac{1}{2} = 0.5$

$\Rightarrow g'(z) = 0.5(1-0.5) = (1/4)$

$[-\infty, +\infty] \rightarrow g(z) \rightarrow [0 \sim 1]$

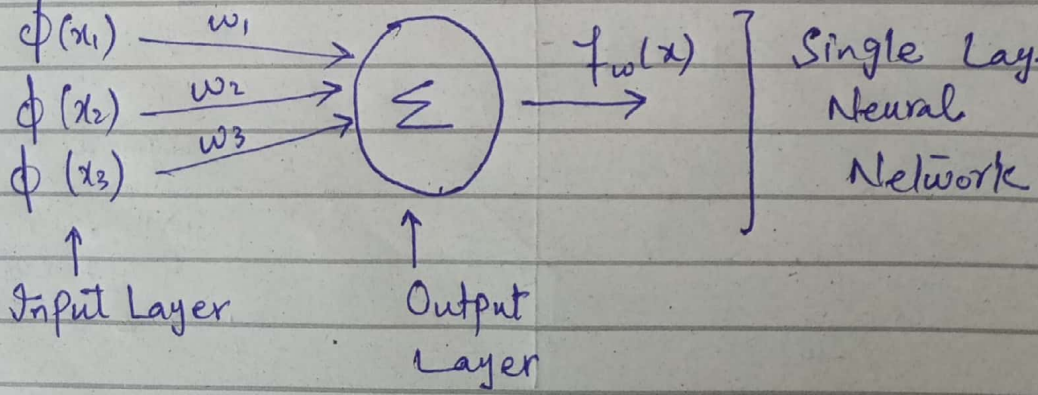
Sigmoid Functions

$$h_1 = \sigma(v_1 \cdot \phi(x))$$

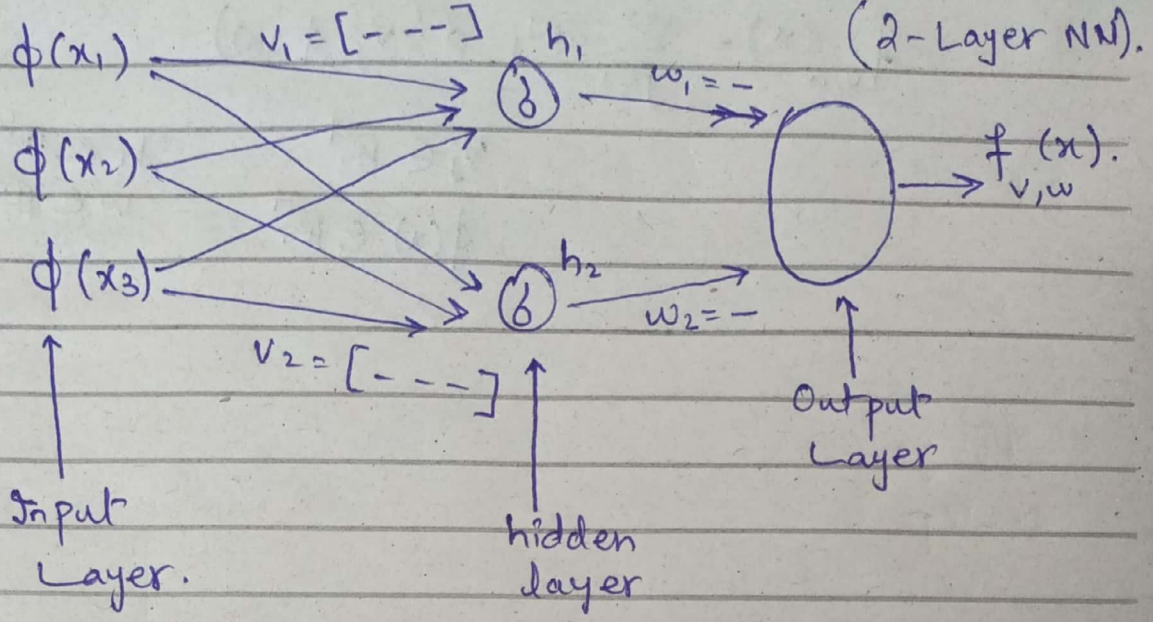
$$h_2 = \sigma(v_2 \cdot \phi(x))$$

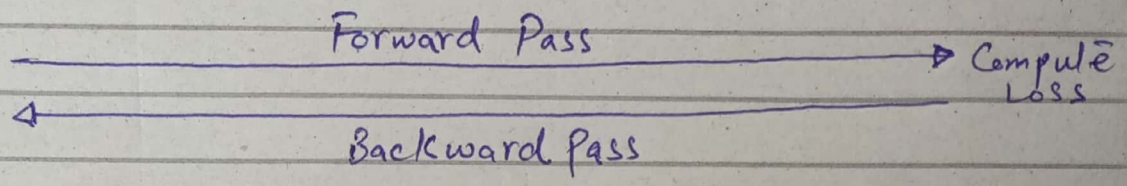
We have discussed in Linear predictors.

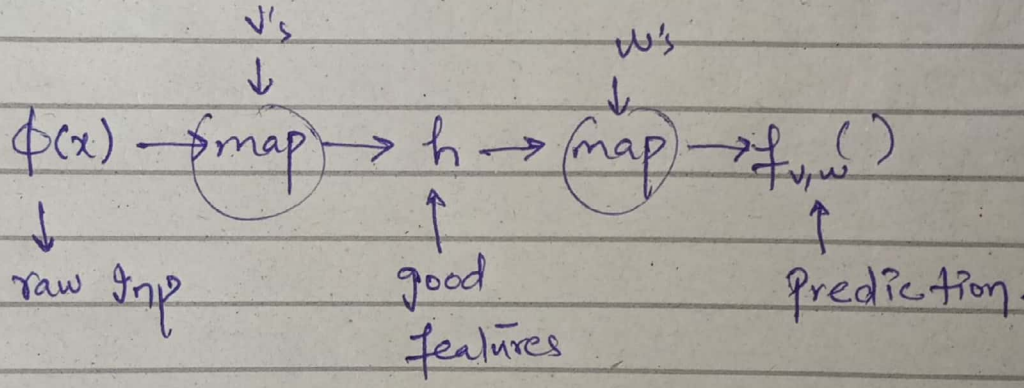
$$f_w = w \cdot \phi(x) = \sum_{i=1}^{d} w_i \phi(x)_i$$



→ Multi-Layer Neural Network.







In Linear Predictors:

$$\frac{1}{|D_{train}|} \sum_{(x,y) \in D_{train}} Loss(x, y, w)$$

$$= (f_w(x) - y)^2$$

$$w \cdot \phi(x)$$

So, in Neural Network

$$Loss(x, y, V, w) = (f_{V,w}(x) - y)^2$$

No. of Hidden units.

$$f_{V,w}(x) = \sum_{j=1}^{k} w_j \cdot \sigma(v_j \cdot \phi(x))$$

$v_j \in \mathbb{R}^d$

$\phi(x) \in \mathbb{R}^d$

$h \in \mathbb{R}^k$

$w \in \mathbb{R}^k$