LECTURE 1

SUNDAY

DATE: 04/06/23

Lecture 1

$y = a_1x_1 + a_2x_2 + \ldots + a_nx_n + b$

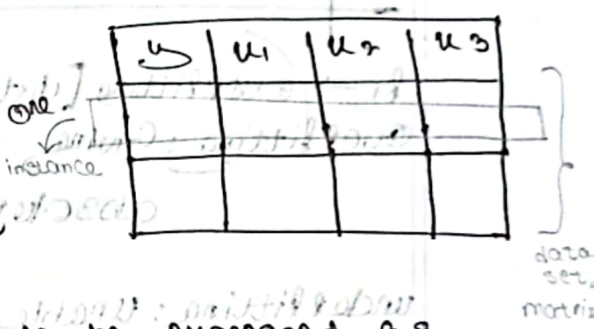
target → y; feature → $x_1, x_2, \ldots, x_n$

Data for ind y is the instance.

feature → column of data set except last one.

instance → rows of data set

label → ~~category~~ last column of data set



target → specific value

feature vector: rows can be expressed as a single vector except label.

~~training~~ ~~predicting~~ ~~data~~ ~~table~~ → The data set from which we learn.

testing ~~tab~~ data → The data set which we test to check whether or not our training data is correct.

*Error made in training data is training error.

*Error made in testing data is testing error.

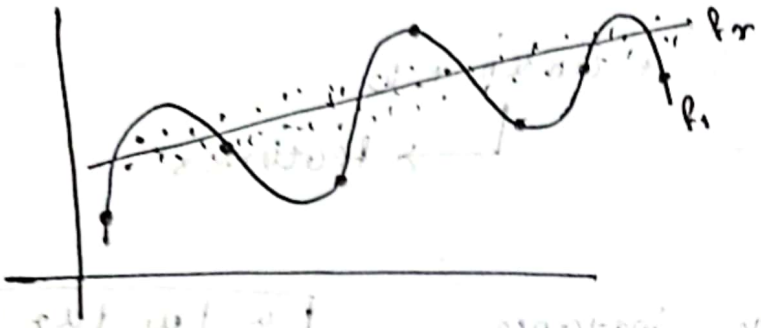
Hyperparameters → free variables which are independent of learning algorithm

validation Data → Testing the training data

Domain → Each feature has a domain

Concept → It is what a learning algorithm tries to learn.

eg. function.



Higher degree polynomial, higher ~~var~~ p number of parameters, more memory used.

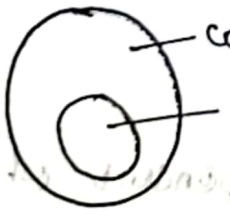
f_1 → overfitting [data is less compared to claims capacity]

overfitting: Giving an explanation that it ~~is~~ is absolutely correct.

underfitting: Unable to give an explanation.

[data is more, but capacity is less]

Hypothesis: A guess / Assumption about the concept.



Inductive bias: Bias produced by hypothesis.

Empirical error: Error ~~is~~ occurs when our prediction does not match with the training data.

(training error / performance error)

$$\text{Error} = \frac{\sum_{i=1}^{m} f(x_i) \neq h(x_i)}{m}$$

Generalization Error: Error of our learning algorithm if we could test it against all the data instances in the domain.

$$\text{Error} = \frac{1 + 0 + 1 + 1 + 0 + 1}{m}$$

$$= \frac{1}{m} \times 1 + 0 + 1 + 0 + 1$$

$\frac{1}{m}$ ↓ weight

$$P(x_1, x_2, \ldots, x_n) = P(x_1)\, P(x_2 \mid x_1) \ldots P(x_n \mid x_1)$$

$$= P(x_1)\, P(x_2)\, P(x_3 \ldots x_n \mid x_1, x_2)$$

We cannot find real-time probability. Thus, cannot find generalization error.

Classification: assigning a label to each data instance.

classification with missing inputs: Some of the information will be ~~as~~ absent. We can yet find out the data using other available informations.

Regression: We don't assign label to data instance like we did in classification.
Estimate a value which is real number

Transcription: converts unstructured data into textual forms
eg: speech to text.

Ranking: Given a search prompt, a search engine generates a list of webpages which need to be sorted for user's convenience.

Clustering: Identify groups of similar entities
eg community detection in social media.

Manifold learning: We plot something which has a surface which need to be learned. Task of reducing feature space intelligently so that learning can occur more efficiently.

Anomaly detection: For any suspicious data point, ↓ which is weird. anomaly detection is needed.

Synthesis and Sampling: A new work is created from past works. (synthesis)
For example: AI is fed with Rabindranath Tagore's songs and asked to write a song. It writes a song similar to that of Tagore's songs.

Imputation of missing values: Sometimes there are missing values in a data set, we can use our wisdom gained through learning to fill-up these values.

Denoising: If a data x is observed to be corrupted $\bar{x}$ due to noisy process. From experience, given $\bar{x}$, we can predict the original clean x.

Probability mass/density estimation: We are just learning $p(x)$ for given instance x.

Supervised learning: Classification, ranking, regression are supervised tasks.

Unsupervised learning: No column for labels present. Clustering and dimensionality reduction are some of the unsupervised learning techniques.

Semi supervised learning: Is ideal when labeled data is expensive. Learn from given labels, utilise similarity measures as in unsupervised learning to make predictions.

Transductive interference: like the semi-supervised learning, we get some labelled & some unlabelled data.
- In case of induction, we learn from training set to make general predictions.
- make no assumption about test data.
- Test data are unlabelled instances.
- Goal: minimize error in test samples only.

* supervised learning, unsupervised learning and semi-supervised learning are inductive interference. We know the data set for learning, but the data set for test is unknown.

LECTURE 2

TUESDAY DATE: 06/06/2

~~Online learning~~:

Offline learning: All the learnings happen before testing.
It learns from the ~~learnings~~ training set before it is tested.

Online learning: training and testing phases are checked.
prediction is made when a data instance is encountered.
Learns from mistake.

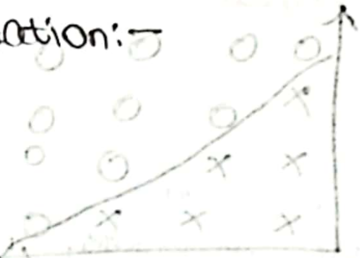
Reinforcement learning: Environment doesn't give any kind of supervision.
Agent gives reward/punishment for an action.
Agent's goal: maximise rewards in future.

Active learning: Carefully choose data points for learning.
goal: get results comparable to the supervised learnings.

Lecture 2

Empirical risk minimization: Error needs to be minimized.

There can be risk of empirical risk minimization:—



(i) Overfit

(ii) underfit

* Hypothesis can introduce bias named inductive bias.
* Inductive bias in ERM is a remedy to overfitting

We have to reduce the search space. From the search space, we have to find the least square.

If we increase data, then error will decrease.

$$Pr(R(h) > \epsilon) \leq 4e^{-\frac{m\epsilon}{4}}$$

- m → no. of data needed for accuracy
- 4 → constant
- h → hypothesis
- $R(h)$ → hypothesis error

$\epsilon = 0.01$

Learning Bound for Finite Inconsistent Hypothesis Classes

$R_S(h)$ → risk

↓ training data

$R(h)$ — generalization / true error

$$R(h) \leq R_S(h) + \sqrt{\frac{\log|H| + \log\frac{2}{\delta}}{2m}}$$

- $|H|$ → no. of hypothesis
- m → no. of data used.

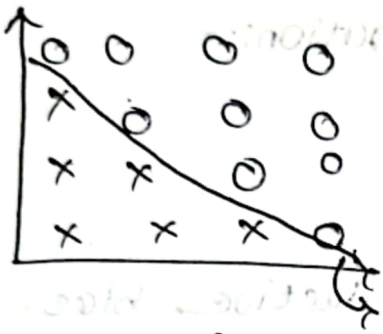
* generalization error cannot be found out ✗
* upper limit of generalization can be determined.

| $|H| \downarrow$, $R(h) \downarrow$ |

$|H|$ → size ছোট ভালো limit দিতে পারে

Lecture 6

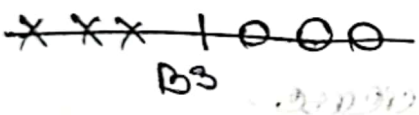
Linear Functions and Affine Functions



X — ~~BS has~~ Diabetes

O — NO ~~BS~~ Diabetes

(Linear models) → 2D (x-y plane)



1D (one plane)

Hyperplane : Line, plane. A plane whose eqn is;—

$$w_1x_1 + w_2x_2 + \dots + w_dx_n + b = 0$$

$f(x_1, x_2) = a_1x_1 + a_2x_2$ (Affine function)

$w = [a_1, a_2]$

$x = [x_1, x_2]$

↳ non-homogenous

non-homogeneous

*Affine function to linear function:— append a column of 1s for in X for b in Y.

$\sum_{i=1}^{d+1} w_i x_i$

*~~Affin~~ Linear functions: ~~Has some~~ Plus something extra (homogenous)

$f(x_1, x_2) = a_1x_1 + a_2x_2 + [b]$

$x = [x_1, x_2, \dots x_n]$

$w_1x_1 + w_2x_2 + \dots + w_nx_n + b$

$\langle w, x \rangle + b$

↓ weight ↓ bias (intercept)

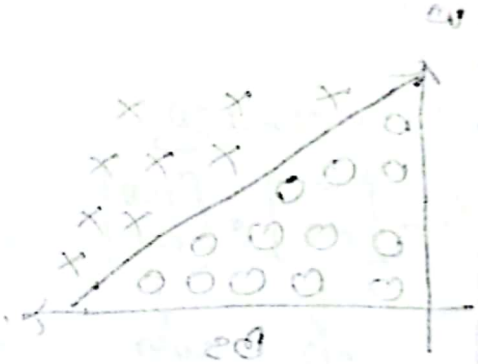
non-homogenous condition

$x = [x_1, x_2, \ldots, x_{n+1}]$

$w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + w_{n+1} x_{n+1}$ (b) (1)

$x_{n+1} = 1$

$w_{n+1} = b$



LECTURE 3

SUNDAY

DATE: 11/06/23

$y = w_1x_1 + w_2x_2 + \dots + w_nx_n + b \times 1 \rightarrow$ linear model

$= \{w.x\}$

$w = \{w_1, w_2, \dots, w_n, b\}$

$x = \{x_1, x_2, \dots, x_n, 1\}$

non-homogenous system

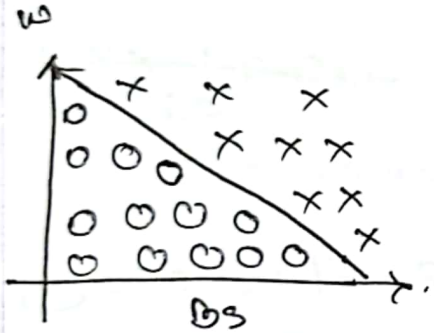
non homogenous coordinate system

univariate linear regression

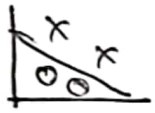
$y = mx + b$

↓

one variable



*linearly seperable data — A line which seperates two data



$\{w.x\} \geq 0 \rightarrow$ X [Diabetes]

$< 0 \rightarrow$ O [No Diabetes]

w_1, w_2, b } To find these, we can use perceptron or gradient descent algorithm

x_1, w_2, b }

$w_1x_1 + w_2x_2 + b \geq 0$

< 0

So,

$\text{sign}(\{w, x\}) \rightarrow$ Hyperplanes and Half spaces

+ve → X [Diabetes]

-ve → O [No Diabetes]

$$\sum_{i=1}^{m} 1 \quad sign(\langle w, x_i \rangle) \neq c_i$$

$x_i = \langle x_{i1}, x_{i2}, \dots, x_{in}, 1 \rangle$,

$w = \langle w_1, w_2, \dots, w_n, b \rangle$

$c_i =$
↳ given

if c_i matches, error = 1

if c_i doesn't match, error = 0

* we cannot use differentiation here.

gradient descent :-

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} - \lambda \begin{bmatrix} \frac{\partial cost}{\partial w_1} \\ \frac{\partial cost}{\partial w_2} \end{bmatrix}$$

(left: new, right: old)

* line এর কাছে গেলে 0 এর কাছে আসে
* line এর নিচে more -ve & line এর উপরে more +ve
* ei error = 0 or 1 কি না (এইটা use করা better না

$sign(w_1 x_{i1} + w_2 x_{i2} + b) = +1$
$= -1$

$c_i = +1$
-1

$$\sum_{i=1}^{m} c_i (w_1 x_{i1} + w_2 x_{i2} + b)$$

sign = +1, c_i = +1, sign * c_i = +1
sign = -1, c_i = -1, sign * c_i = +1
∴ ei sign * c_i

Sigmoid function / Logistic function

~~$w_1x_1 + w_2$~~

$\langle w, x \rangle = w_1x_1 + w_2x_2 + b \qquad w \in \mathbb{R}^{n+1}$

$\downarrow$

$\frac{1}{1+e^{-\langle w, x \rangle}}$

↳ (Diabetes হবে probability কতটা হবে সেটা দেখায়)

$\frac{1}{1+e^{-\langle w, x \rangle}} \geq 0.5 \rightarrow$ Diabetes [output = +1]

$\langle w \cdot x \rangle = 0$

$\downarrow$

$\frac{1}{1+e^{0}} = \frac{1}{2}$

$\langle w \cdot x \rangle = 1 \qquad \frac{1}{1+e^{-1}} \geq 0.5$

$\langle w \cdot x \rangle = -1 \qquad \frac{1}{1+e} < 0.5$

$\log(1 + e^{-y_i \langle w, x_i \rangle})$

$l(h_w(x_i, y_i)) = \log(1 + e^{-y_i \langle w, x_i \rangle}) \rightarrow y_i, h_w(x_i) \in \{-1, 1\}$

$l(h_w, (x_i, y_i)) = -y_i \log h_w(x_i) - (1 - y_i) \log(1 - h_w(x_i)) \rightarrow y_i, h_w(x_i) \in \{0, 1\}$

~~$\sum_{i=1}^{m} y_i (1 + e^{-y_i \langle w, x_i \rangle}) \rightarrow$ Perceptron Learning Rule~~

$\frac{1}{m} \sum_{i=1}^{m} -(y_i \langle w, x_i \rangle) \rightarrow$ Perceptron Learning Rule

$\frac{1}{m} \sum (y - \langle w, x \rangle)^2$

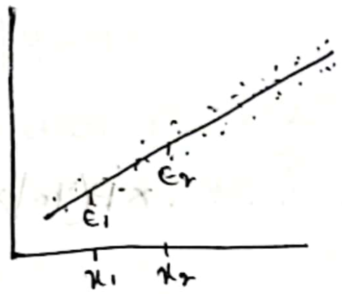
$P(y=1 \mid w, x) = \frac{1}{1+e^{-\langle w, x \rangle}} = h_w(x)$

$P(y=0 \mid w, x) = 1 - h_w(x)$

$a \in \{0, 1\}$

$P(y=a \mid w, x) = h_w(x)^{a} (1 - h_w(x))^{1-a}$

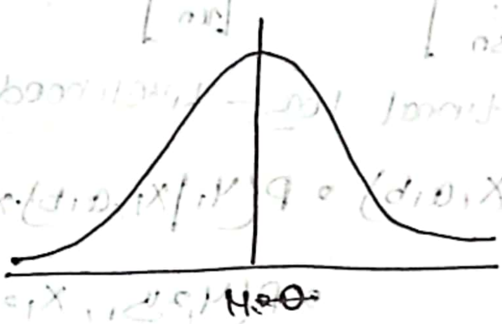
LECTURE 4 TUESDAY DATE: 13/06/23



average error এর জন্য মান না usually.

error

Gaussian Distribution



* error ছোট হওয়ার chance বেশি, বড় হওয়ার chance কম
* μ থেকে error বেশি দূরে যাওয়ার chance কম।
* μ এর কাছেই error বেশি হওয়ার chance আছে।

$\epsilon_i = y_i - \hat{y}_i$

variance ← $\sigma^2 = \frac{(x_1-\mu)^2 + (x_2-\mu)^2 + \dots (x_n-\mu)^2}{n}$

standard deviation ↓ ← $\sigma = \sqrt{\frac{(x_1-\mu)^2 + (x_2-\mu)^2 + \dots (x_n-\mu)^2}{n}}$

average এর থেকে দূরে error

$$P(x \mid N(\mu, \sigma)) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$$

$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$ $X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$

* যত data independent হবে
* $P(A \cap B) = P(A)P(B)$

Conditional log – likelihood

$$P(Y|X,a,b) = P(y_1|x_1,a,b) \times P(y_2|x_2,a,b) \times \dots \times P(y_n|x_n,a,b)$$

$$= P(y_1 = y_1, x_1 = x_1, a, b)$$

$$= P(\epsilon_1 = y_1 - ax_1 - b \mid x_1 = x_1, a, b)$$

$$= P(\epsilon_1|x_1,a,b)\,P(\epsilon_2|x_2,a,b) \dots P(\epsilon_n|x_n,a,b)$$

$$= c e^{-\frac{1}{2}\frac{\epsilon_1^2}{\sigma^2}} \cdot c e^{-\frac{1}{2}\frac{\epsilon_2^2}{\sigma^2}} \dots c e^{-\frac{1}{2}\frac{\epsilon_n^2}{\sigma^2}}$$

$$J = c^n e^{-\frac{1}{2}\left(\frac{\epsilon_1^2}{\sigma^2} + \frac{\epsilon_2^2}{\sigma^2} + \frac{\epsilon_3^2}{\sigma^2} + \dots + \frac{\epsilon_n^2}{\sigma^2}\right)}$$

maximise করতে চাই

$P(x|N(\mu,\sigma)) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$ → ϵ_1

$\mu = \frac{\sum_{i=1}^{m} x_i}{m}$

$\sigma = \sqrt{\frac{\sum_{i=1}^{m}(x_i-\mu)^2}{m}}$

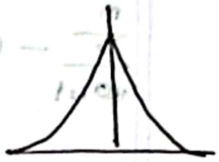
ϵ_1 ↓ c

$$\ln J = \ln c^n e^{-\frac{1}{2}\left(\frac{\epsilon_1^2}{\sigma^2} + \frac{\epsilon_2^2}{\sigma^2} + \frac{\epsilon_3^2}{\sigma^2} + \dots + \frac{\epsilon_n^2}{\sigma^2}\right)}$$

$$\ln J = n \ln c - \frac{1}{2\sigma^2}(\epsilon_1^2 + \epsilon_2^2 + \dots + \epsilon_n^2)$$

$$-\ln J = -n \ln c + \frac{1}{2\sigma^2}(\epsilon_1^2 + \epsilon_2^2 + \dots + \epsilon_n^2)$$

$f(x|\mu,b) = \frac{1}{2b} e^{-\left(\frac{|x-\mu|}{b}\right)}$ → laplace distribution


laplace distribution

$|y - y_i|$

* Error এর কমাবার জন্য আমরা Gaussian Distribution টাই normally use করি

BINARY CLASSIFICATION

$a = \{0, 1\}$

$P(y = a|w,x) = h_w(x)^{a}(1 - h_w(x))^{1-a}$ $P(y = 1|w,x) = \frac{1}{1+e^{-(w \cdot x)}} = h_w(x)$

if $a = 0$,

$P(y = 0|w,x) = 1 - h_w(x)$

if $a = 1$,

$P(y = 1|w,x) = h_w(x)$

$P(Y|X,W) = P(y_1|x_1,w) \cdot P(y_2|x_2,w) \cdots P(y_m|x_m,w)$

X → patient ; W → patient's weight

$= P(y_1 = a_1|x_1,w)$

$J = h_w(x_1)^{a_1}(1 - h_w(x_1))^{1-a_1}\, h_w(x_2)^{a_2}(1 - h_w(x_2))^{1-a_2} \cdots$

$h_w(x_m)^{a_m}(1 - h_w(x_m))^{1-a_m}$

$\log J = \sum_{i=1}^{m} \log\left[h_w(x_i)^{a_i}(1 - h_w(x_i))^{1-a_i}\right]$

$\log J = \sum_{i=1}^{m} \log h_w(x_i)^{a_i} + \log(1 - h_w(x_i))^{1-a_i}$ $a_i = 1$ → $-\log h_w(x_i)$

$-\log J = -\sum_{i=1}^{m} a_i \log h_w(x_i) + (1 - a_i)\log(1 - h_w(x_i))$ $a_i = 0$ → $-(1-a_i)\log(1-h_w(x_i))$

Error কে minimize করতে

cross entropy error/loss [i তম patient এর জন্য যত info, ততটা better হলে তত কম error]

regression | classification → total entropy error
↓ sum of square error | ↙ binary ↘ log multi

$$\sum_{i=1}^{m} -a_i \log(h_w(x_i)) - (1-a_i)\log(1-h_w(x_i))$$ → binary classification

↳ প্রতিটি value a_i এর value এর মাধ্যমে আসবে

MULTINOMIAL CLASSIFICATION

$$\sum_{i=1}^{m} (y_i - \bar{y}_i)$$ → Linear Regression

w_{11} → matrix
↳ color 1 (c_1)
↳ feature 1 of c_1

S = $S_1, S_2, \ldots, S_k$
↳ score for c_1 color (c_1)

$S_1 = w_{11}x_1 + w_{21}x_2 + \ldots + w_{n1}x_n + b_1$

x_1	R
x_2	G

check for color →

c_1	R c_2	B c_3	c_4	c_5
0	1	0	0	0
1	0	0	0	0

$S_2 = w_{12}x_1 + w_{22}x_2 + \ldots + w_{n2}x_n + b_2$

$S_k = w_{1k}x_1 + w_{2k}x_2 + \ldots + w_{nk}x_n + b_k$

$\Rightarrow -c_1 \log(y=1|x_i)) - c_2 \log(y=2|x_i) - c_3\log(y=3|x_i)$
$- \ldots$

$$= \sum_{i=1}^{m} -c_1\log(y=1|x_i) - c_2\log(y=2|x_i) - c_3\log(y=3|x_i)$$

P = Using softmax function:-

S = $S_1, S_2, \ldots, S_k$
$\quad -1 \quad -3 \quad \quad 7$

$$\frac{e^{-1}}{m} + \frac{e^{-3}}{m} + \ldots + \frac{e^{7}}{m} = 1$$

$$m = e^{-1} + e^{-3} + \ldots + e^{7}$$

* softmax function: প্রতিটি new array দিবে (S), new array হবে যেটা এই new array-এ প্রতিটা color এর probability

$$\sum -a_i \log(P(x_i)) - (1-a_i)\log(1-P(x_i)) = ~~P(y_i = a_i|x_i)~~$$

(ONLINE) LECTURE 5

TUESDAY

DATE: 20/09/23

Gradient Descent

ei A value directly $w = (X^T X)^{-1} X^T Y$ দিয়ে বের করা যায়, we are using offline learning.

$$J = (y_1 - \bar{y}_1)^2 + (y_2 - \bar{y}_2)^2 + \dots + (y_m - \bar{y}_m)^2$$

$$J = \sum_{i=1}^{m} (y_i - \bar{y}_i)^2$$

$$\bar{y}_i = a_1 x_{i1} + a_2 x_{i2} + \dots a_n x_{in} + b$$

$$\frac{\partial J}{\partial a_1} = \sum_{i=1}^{m} 2(y_i - \bar{y}_i) \frac{\partial}{\partial a_1}(y_i - \bar{y}_i)$$

$$= \sum_{i=1}^{m} 2(y_i - \bar{y}_i)(-x_{i1})$$

$$\frac{\partial J}{\partial a_2} = \sum_{i=1}^{m} 2(y_i - \bar{y}_i) \frac{\partial}{\partial a_2}(y_i - \bar{y}_i)$$

$$= \sum_{i=1}^{m} 2(y_i - \bar{y}_i)(-x_{i2})$$

$$\frac{\partial J}{\partial b} = \sum_{i=1}^{m} 2(y_i - \bar{y}_i) \frac{\partial}{\partial b}(y_i - \bar{y}_i)$$

$$= \sum_{i=1}^{m} 2(y_i - \bar{y}_i)(-1)$$

* Gradient of J is the direction at which the value of function increases the fastest.

* Gradient এর বিপরীত দিকে গেলে function এর value কমে।

* Gradient এর opposite এ গেলে function এর value কতো কমে সেটা

$$\begin{bmatrix} a_1' \\ a_2' \\ a_3' \\ \vdots \\ b' \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ b \end{bmatrix} - \lambda \nabla J$$

(λ → learning rate)

new weights

$$\begin{bmatrix} a_1' \\ a_2' \\ \vdots \\ a_n' \\ b' \end{bmatrix} = \begin{bmatrix} a_1 + 2\sum_{i=1}^{m} (y_i - \bar{y}_i) x_{i1} \\ a_2 + 2\sum_{i=1}^{m} (y_i - \bar{y}_i)(x_{i2}) \\ \vdots \\ a_n + 2\sum_{i=1}^{m} (y_i - \bar{y}_i)(x_{in}) \\ b + 2\sum_{i=1}^{m} (y_i - \bar{y}_i)(1) \end{bmatrix}$$

learning with gradient descent [for linear regression]

Learning with gradient descent [for linear classification]

$\bar{y}_i = a_1 x_{i1} + a_2 x_{i2} + a_3 x_{i3} + \ldots \ldots + x_{in} + b$

↳ amar value

y_i → given value

→ আমার value output

$\bar{y}_i$ y_i from ↳ given data

maximise { $\bar{y}_i y_i > 0$ → correct
$\bar{y}_i y_i \leq 0$ → incorrect

↓ minimise

* y_i & $\bar{y}_i$ (same sign) → error = 0
* $\bar{y}_i$ & y_i (different sign) → error > 0

$-\bar{y}_i y_i$

$-(-1 * -17) = -17$ → বড় value আমাদের max value নিব

* if $\bar{y}_i$ & y_i have same sign then the error has to be 0. But $-(\bar{y}_i y_i)$ gives a negative value. Thus, we use $\max(-17, 0) = 0$, to get the correct error.

$\max(-17, 0) = 0$ * sub gradient descent!

$\max(-y_i \bar{y}_i, 0)$

$J = \sum_{i=1}^{m} \max(-y_i \bar{y}_i, 0)$ [incorrect গুলোর error]

$\frac{\partial J}{\partial a_1} = \sum_{i=1}^{m} \begin{cases} 0 & y_i \bar{y}_i > 0 \\ -y_i x_{i1} & y_i \bar{y}_i \not> 0 \end{cases}$

$J = \max(0, -y_i \bar{y}_i)$

* $(y_i \bar{y}_i) > 0$, হলে $\frac{\partial J}{\partial a_1}$ এর value ~~0 নিব~~, ~~$-y_i x_{i1}$ নিব~~
* $(y_i \bar{y}_i) \leq 0$ হলে $\frac{\partial J}{\partial a_1}$ এর value ~~$-y_i x_{i1}$ নিব~~ 0 নিব

$\frac{\partial J}{\partial a_2} = \sum_{i=1}^{m} \begin{cases} 0 \\ -y_i x_{i2} \end{cases}$

$J = \sum_{i=1}^{m} \max(0, -y_i \bar{y}_i)$ → Offline learning

$a'_1 = a_1 - \lambda \frac{\partial J}{\partial a_1}$

$J = \max(0, -y_i \bar{y}_i)$ → Online learning

$a'_2 = a_2 - \lambda \frac{\partial J}{\partial a_2}$

$y_i x_i < 0$

$$\frac{\partial J}{\partial a} = -y_i x_{i1}$$

$$\frac{\partial J}{\partial a_2} = -y_i x_{i2}$$

$$\frac{\partial J}{\partial a_n} = -y_i x_{in}$$

$$\frac{\partial J}{\partial b} = -y_i$$

$$a'_1 = a_1 + \lambda y_i x_{i1}$$

$$a'_2 = a_2 + \lambda y_i x_{i2}$$

$$a'_n = a_n + \lambda y_i x_{in}$$

$$b' = b + \lambda y_i * (1)$$

$$w^{t+1} = w^t + \lambda y_i \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{in} \\ 1 \end{bmatrix}$$

*For perceptron algorithm, λ is always 1

Perceptron algorithm:-

input: A training set $(x_1, y_1) \ldots (x_m, y_m)$

initialize: $w^{(1)} = (0, \ldots, 0)$

for $t = 1, 2 \ldots$

if $(\exists i, s.t.\ y_i (w^{(t)}, x_i) \leq 0)$ then

$w^{(t+1)} = w^{(t)} + y_i x_i$ → dot product of $w^{(t)}$ & x_i

else:

output $w^{(t)}$

[incorrect value update]

x_i → 1 number house
1 number feature.

w → weight

x → feature

LECTURE 6

TUESDAY DATE: 04/07/23

CLASSIFIERS

- Naive Bayes Classifier
- Decision Trees
- Boosting

~~P(A∩B) = P(A)P(B|A)~~

A → University open

B → Raining

NAIVE BAYESIAN CLASSIFIER

$$P(B)P(A|B) = P(A \cap B) = P(A)P(B|A)$$

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)} \rightarrow \text{Baye's Theorem}$$

x_1	x_2	x_3	y
a	1	20	OH Y/N

$B = (x_1 = a, x_2 = 1, x_3 = 20)$

$$P(y = yes | x_1 = a, x_2 = 1, x_3 = 20) = \frac{P(y = yes) \{P(x_1 = a, x_2 = 1, x_3 = 20)\} | y = yes)}{P(x_1 = a, x_2 = 1, x_3 = 20)}$$

* Naive Bayesian classifier is used for discrete, categorical data.
* ~~Discrete~~ Images cannot be used directly, even though they are discrete. We need to group.

10 — 20 → child

* একটা column এর data তে ~~group~~ values এর মধ্যে group আসলে ভালো, তবে values same হলে unique হওয়া যাবে না।

$$P(Ph=Yes \mid O=S, T=C, H=h, W=T) = \frac{P(Ph=Yes)\,P(O=S, T=C, H=h, W=T \mid Ph=Yes)}{P(O=S, T=C, H=h, W=T)}$$

$$P(Yes) = \frac{9}{9+5}$$

$$P(O=S, T=C, H=h, W=T \mid Ph=Yes)$$
$$= P(T=S \mid Ph=Yes) + P(T=C \mid Ph=Yes)$$
$$+ P(H=h \mid Ph=Yes) + P(W=T \mid Ph=Yes)$$
$$\Rightarrow \left(\frac{3}{9}\right) * \left(\frac{3}{9}\right) * \left(\frac{3}{9}\right) * \left(\frac{3}{9}\right)$$

$P(A \cap B) = P(A)P(B) \rightarrow$ Absolutely independent

$P(A \cap B \mid C) = P(A \mid C) * P(B \mid C) \rightarrow$ conditionally independent

$$P(O=S, T=C, H=h, W=T) = P(O=S, T=C, H=h, W=T, Ph=Yes) + P(O=S, T=C, H=h, W=T, Ph=No)$$

~~$P(O=S, T=C, H=h, W=T \mid Ph=No) = P($~~

$$P(O=S, T=C, H=h, W=T, Ph=Yes) = P(Ph=Yes)\,P(O=S, T=C, H=h, W=T \mid Ph=Yes)$$

$$P(Ph=Yes \mid O=S, T=C, H=h, W=T) = \frac{P(O=S, T=C, H=h, W=T, Ph=Yes)}{P(O=S, T=C, H=h, W=T, Ph=Yes) + P(O=S, T=C, H=h, W=T, Ph=No)}$$

$$P(Ph=Yes \mid O=S, T=t, H=h, W=T) = \frac{P(O=S, T=t, H=h, W=t \mid Ph=Yes) P(Ph=Yes)}{P(Ph=Yes) P(O=S, T=t, H=h, W=t \mid Ph=Yes) + P(Ph=No) P(O=S, T=t, H=h, W=t \mid Ph=No)}$$

$$P(Ph=No \mid O=S, T=t, H=h, W=T) = \frac{P(O=S, T=t, H=h, W=t \mid Ph=No) P(Ph=No)}{P(Ph=Yes) P(O=S, T=t, H=h, W=t \mid Ph=Yes) + P(Ph=No) P(O=S, T=t, H=h, W=t \mid Ph=No)}$$

Frequent list approach: If probability of event A = 1 and probability of event B = 0, then as per given data, frequent list approach occurs.

* Using more data, we can get a probability of B.

LAPLACE SMOOTHING

Ph = Yes	R	O	S
9	$\frac{3}{9}$	$\frac{6}{9}$	$\frac{0}{9}$
12	$\frac{4}{12}$	$\frac{7}{12}$	$\frac{1}{12}$

↓

Ph = Yes	R	O	S
9	$\frac{3}{9}$	$\frac{6}{9}$	$\frac{0}{9}$
9+3K	$\frac{3+K}{9+3K}$	$\frac{6+K}{9+3K}$	$\frac{0+K}{9+3K}$

* Smoothing করলে কোন value 0 আসবে না।
* ~~কোন value missing আসলে 1 হবে~~
* কোন value missing আসলে value + K হবে (here missing value + all values in table)
* For Laplace smoothing value of K is always 1

	H	T
10	9	1

$$P(H) = \frac{9}{10}, \quad P(T) = \frac{1}{10}$$

$$P(h \mid H=9, T=1) = \frac{P(H=9, T=1 \mid h)\, P(h)}{P(H=9, T=1)}$$

likelihood — prior — prior

$$= \frac{{}^{10}C_1 \left(\frac{1}{2}\right)^9 \left(\frac{1}{2}\right)^1 \left(\frac{1}{2}\right)}{\int_0^1 {}^{10}C_1 (x)^9 (1-x)^1 \, dx}$$

* previous idea (prior) আর new observation (likelihood) নিয়ে, new observation main data তে update করে নিচ্ছি। This is online learning.

LECTURE 7

SUNDAY

DATE: 09/07/23

ENTROPY

H	T	
50	50	→ uncertainty বেশি
90	10	
10	90	

Dice :-

1	2	3	4	5	6	→ outcome
$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	→ 1 is equ-divided amongst all outcomes

Probability Distribution

$\frac{1}{p}$

→ যে event এর chance বেশি, তার p এর value বেশি, $\frac{1}{p}$ এর value ছোট

→ যে event এর chance কম, তার p এর value কম, $\frac{1}{p}$ এর value অনেক বড়

$\log\left(\frac{1}{p}\right)$ → To decrease a very large value of $\frac{1}{p}$ if p is very small

$\Rightarrow -\log(p)$

* $I(X) = -\log P(X)$

↳ self information

Outlook

- $P(\text{Rainy}) = \frac{5}{14}$
- $P(\text{Overcast}) = \frac{4}{14}$
- $P(\text{Sunny}) = \frac{5}{14}$

$$E[\text{Outlook}] = \frac{5}{14}\left(-\log\frac{5}{14}\right) + \frac{4}{14}\left(-\log\frac{4}{14}\right) + \frac{5}{14}\left(-\log\frac{5}{14}\right)$$

(probability of event Rainy; self information of Rainy)

* Entropy is the expected value and is denoted by $H[\text{Outlook}]$

	X		
	a	b	c
Y	$\frac{4}{6}$	$\frac{1}{6}$	$\frac{1}{6}$
Z	$\frac{2}{6}$	$\frac{2}{6}$	$\frac{2}{6}$

~~* Uncertainty of~~

$$H[Y] = \frac{4}{6}\left(-\log\frac{4}{6}\right) + \frac{1}{6}\left(-\log\frac{1}{6}\right) + \frac{1}{6}\left(-\log\frac{1}{6}\right) = 1.25$$

$$H[Z] = \frac{2}{6}\left(-\log\frac{2}{6}\right) + \frac{2}{6}\left(-\log\frac{2}{6}\right) + \frac{2}{6}\left(-\log\frac{2}{6}\right) = 1.58$$

* Uncertainty of Z > Y as H[Z] > H[Y]
* We need small uncertainty so entropy need to be small

~~Rainy~~

DECISION TREE

Overcast
ON 4 Y
* Use entropy as we are sure we can play if Y.

Rainy
3N 2Y

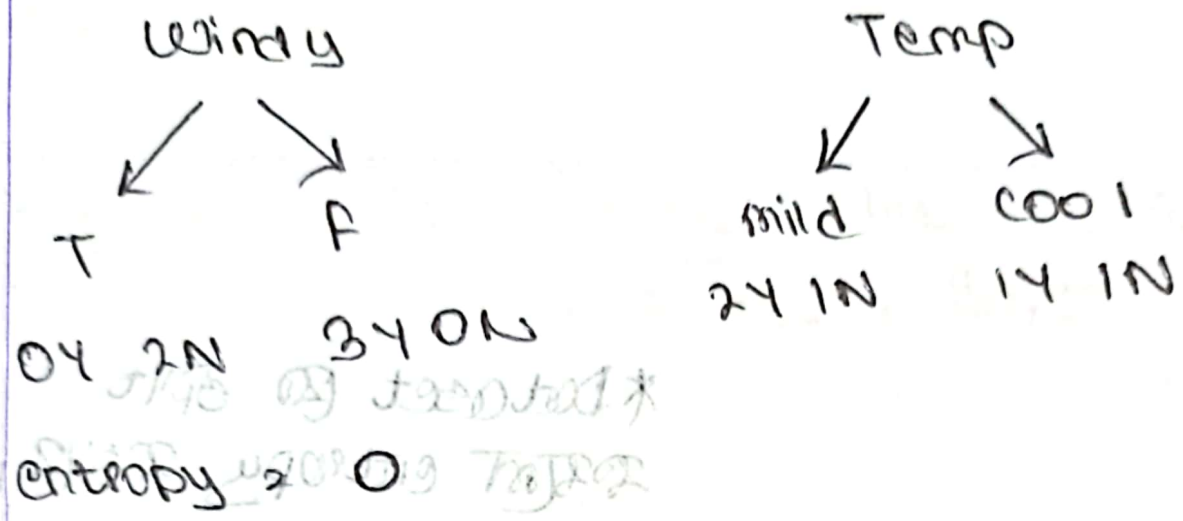
Sunny
3Y 2N

* Same entropy [entropy ≠ 0]

$$\frac{3}{5}\left(-\log\frac{3}{5}\right) + \frac{2}{5}\left(-\log\frac{2}{5}\right)$$

$$= 0.97$$

* We are finding out entropy of playoff with respect to Outlook
* Out entropy of playoff wrt outlook is the smallest compared to Temp, Humidity, Windy.



entropy = 0

LECTURE 10

TUESDAY　　　　　　　　　　　　　　　　DATE: 11/07/23

~~Information gain:~~

Q. Yes	No
9	5

* Dataset কে split করলে entropy কমবে।

overall entropy of dataset

$= \frac{9}{14}\left(-\log\frac{9}{14}\right) + \frac{5}{14}\left(-\log\frac{5}{14}\right)$

$= 0.94$

Outlook

Sunny	Rainy	Overcast ~~Outlook~~
3Y 2N	2Y 3N	4Y 0N
(0.97)	(0.97)	(0)

overall entropy of ~~outlook~~ dataset

$= \left(\frac{5}{14} * 0.97\right) + \left(\frac{5}{14} * 0.97\right) + \left(\frac{4}{14} * 0\right)$

$= 0.69$ → outlook দিয়ে dataset কে split করেছি তাই entropy কমে গেছে

Temp

Hot	Mild	Cool
2Y 2N	4Y 2N	3Y 1N
(1)	(0.92)	(0.81)

overall entropy of dataset

$= \left(\frac{4}{14} * 1\right) + \left(\frac{6}{14} * 0.92\right) + \left(\frac{4}{14} * 0.\right.$

$= 0.911$

* এমন data থাকতে পারে যেখানে different values in a column, so তার entropy smallest হবে. But, এইটা use করা যাবে না. For that reason, we use information entropy.

Information gain = amount of reduction in entropy

= Previous entropy − split_entropy

eg. Information Gain (Outlook) = 0.94 − 0.69

Information Gain (Temp) = 0.94 − 0.911　　OYT

CamScanner

Split Entropy

* If there are many values in a column, then the frequency of that value will be 1. So, split entropy will be highest.

$-\frac{1}{n}\log\frac{1}{n} - \frac{1}{n}\log\frac{1}{n} - \ldots\ldots - \frac{1}{n}\log\frac{1}{n} \ldots\ldots - p_n \log n \leq \log n$

$= n * \frac{1}{n}\log n$

$= \log n$

$$\text{Gain Ratio}(X) = \frac{\text{Information Gain}(X)}{\text{Split Entropy}(X)}$$

* Gain ratio সাধারণত তখনই splitting এ consider করা হয়।

$p_1 + p_2 + \ldots\ldots p_n = 1$

$-p_1\log p_1 - p_2\log p_2 - \ldots\ldots - p_n\log p_n$

$0 \leq -p_1\log p_1 - p_2\log p_2 - \ldots\ldots - p_n\log p_n \leq \log n$

$\log 1 = 0$

$\log x, x \to +$

$\log x, x \to -$

Gini Impurity
↳ index

$p_1^2 + p_2^2 + p_3^2 + \dots + p_n^2$

$= 1 - (p_1^2 + p_2^2 + p_3^2 + \dots + p_n^2)$

$= 1 - \left(\frac{1}{n^2} + \frac{1}{n^2} + \dots + \frac{1}{n^2}\right)$

$= 1 - \frac{1}{n}$ → usually n এর value ২ থাকে

$= \frac{n-1}{n}$

$\lim_{n \to \infty} \frac{n-1}{n} = 1$ [if n is very large, $n \approx n-1$]

$p_1^2 \leq p_1 \mid -p_1^2 \geq -p_1$

$1 - (p_1^2 + p_2^2 + p_3^2 + \dots + p_n^2)$

$\geq 1 - p_1^2 - p_2^2 - p_3^2 - \dots - p_n^2$

$\geq 1 - (p_1 + p_2 + p_3 + \dots + p_n)$

* 0 ≤ Gini Impurity ≤ 0.5

Maximum likelihood estimation

Linear Classification → Perceptron Algo

Linear Regression ~~→ Perceptron Algo~~
- → Gradient Descent
- → Least Square Soln

Naive Bayes's Classifier

Decision Trees

LAB
1

Titanic data –

Supervised learning – classification, ranking, regression

classification – assigning a label to each data instances

Regression – linear → used for continuous values
– logistic → almost linear

Categorical Data – used for prediction
– when we use categorical data it is classification
– Random categories
– eg Old, Yes/No

Ordial value – when one value is better than other

Rent for a house is linearly dependent on all the factors influencing the rent.

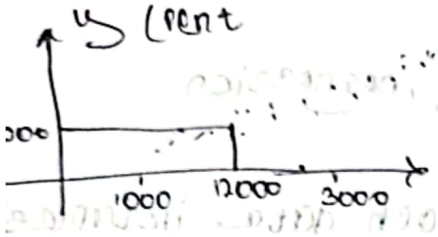
$y = a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 + \dots + a_bx_b + \dots + s + \phi$

↳ linearly dependency [power of all x are 1]

possibilities
no soln
many soln
1 soln

* variable বেশি, equation কম — linear dependent এবং change কম
* variable কম, equation বেশ — no soln
* variable & equation are equal — linear dependent এবং এদের মান আছে,

$y = ax + b$



n = data points / n features / $a_1 a_2 a_3$ weights / variables / parameters

$y_1 = a_1 x_{1,1} + a_2 x_{1,2} + a_3 x_{1,3} + \dots + b_1 * 1$

($x_{1,1}$: Data 1 for 1st Data; $x_{1,2}$: Data 2 for 1st Data; b_1: noise)

A	B	C
a_1	b_1	c_1

1st Data. Data 1 for 1st Data

$y_2 = a_1 x_{2,1} + a_2 x_{2,2} + a_3 x_{2,3} + \dots + b_2 * 1$

$\vdots$

$y_m = a_1 x_{m,1} + a_2 x_{m,2} + a_3 x_{m,3} + \dots + b_m$

$X \rightarrow$ feature matrix $\quad w \rightarrow$ feature vector

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,n-1} \\ \vdots & & & \\ x_{m,1} & x_{m,2} & \dots & x_{m,n-1} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \\ b \end{bmatrix}$$

homogenous

non-homogenous

* for b variable, we have to append a column with 1 which makes the matrix non-homogenous from homogenous.

* This is offline learning

$Y = Xw$

$\Rightarrow Xw = Y$

$\Rightarrow X^T X w = X^T Y$

$\Rightarrow w = (X^T X)^{-1} X^T Y$ → least square sol^n

$\begin{bmatrix} 3 & 6 \\ 2 & 4 \end{bmatrix}$ Determinant = $(3*4) - (6*2) = 0$, so can't invert

* Actual Data এর matrix এ নিয়ে কাজ হচ্ছে

$$\begin{bmatrix} 3+0.003 & 6 \\ 2 & 4+0.003 \end{bmatrix}$$

↓

$$\underbrace{\begin{bmatrix} 3 & 6 \\ 2 & 4 \end{bmatrix}}_{X} + \begin{bmatrix} 0.003 & 0 \\ 0 & 0.003 \end{bmatrix}$$

$$X + 0.003 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

↓ identity matrix

$$X + 0.003 I$$

$$Xw = Y$$

$$\Rightarrow X^T X w = X^T Y$$

$$\Rightarrow (X^T X + \alpha I) w = (X^T Y)$$

Error:-

$y \to$ actual

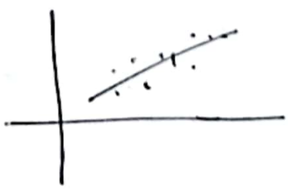
$\bar{y} \to$ approximate

$(y - \bar{y}_1)^2$ OR $|y_1 - \bar{y}_1|$

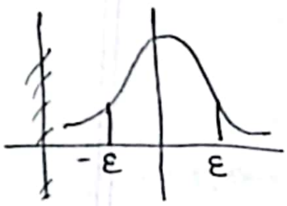
↓ better

$$\frac{(y_1 - \bar{y}_1)^2 + (y_2 - \bar{y}_2)^2 + \dots (y_K - \bar{y}_K)^2}{K}$$

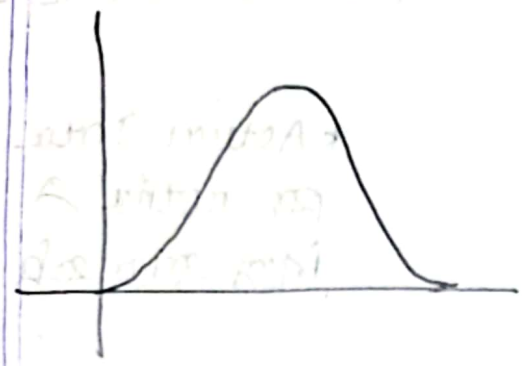
→ mean square error



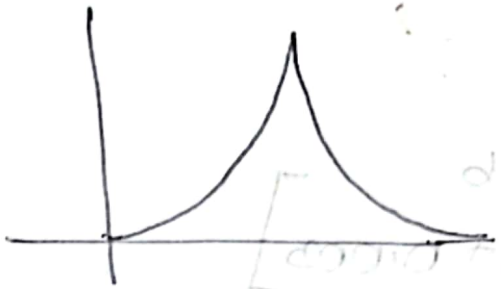
$\varepsilon_1 = y_1 - \bar{y}_1$



+ve & -ve error are of same probability



Gaussian Distribution



Laplace Distribution

LAB 2

SATURDAY

DATE: 10/06/23

$\frac{\sum_{i=1}^{m} (y_i - \bar{y}_i)^2}{m}$ [mean square error (MSE)]

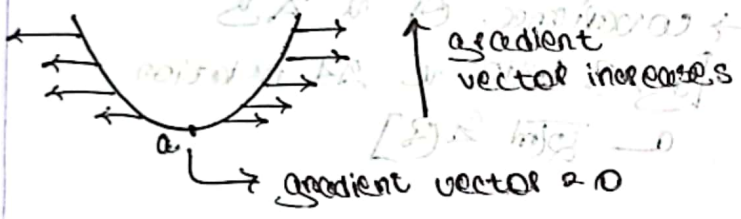
$\bar{y} = ax + b$ → estimated

$\bar{y}_i = ax_i + b$

$J = \sum_{i=1}^{m} (y_i - \bar{y}_i)^2$ → sum of squared error

$J = \sum_{i=1}^{m} (y_i - ax_i - b)^2$

gradient vector:



$\nabla J = \left[\frac{\partial J}{\partial a}, \frac{\partial J}{\partial b}\right]$ → gradient vector, $\Delta x = (\Delta a, \Delta b)$ → change in a & b, → directional derivatives

$= |\Delta x| |\nabla J| \cos\theta$

Total change $= \Delta x \cdot \nabla J$

$= \frac{\partial J}{\partial a} \Delta a + \frac{\partial J}{\partial b} \Delta b$

At maximum/minimum point, $\frac{\partial J}{\partial a} = 0, \frac{\partial J}{\partial b} = 0$

$\frac{\partial J}{\partial a} = \sum_{i=1}^{m} 2(y_i - ax_i - b)(-x_i) = 0$

$\frac{\partial J}{\partial b} = \sum_{i=1}^{m} 2(y_i - ax_i - b)(-1) = 0$

$A \cdot B = |A||B|\cos\theta$

* gradient এর direction এ যখন maximum value of $\cos\theta$ পাওয়া যায়,

* gradient এর opposite direction এ যখন minimum value of $\cos\theta$ পাওয়া যায়,

$$\frac{\partial J}{\partial a} = \sum_{x_i=1}^{m}(x_i y_i - a x_i^2 - b x_i) = 0 \rightarrow \frac{\partial J}{\partial a} = \sum x_i y_i - a\sum x_i^2 - b\sum x_i = 0$$

$$\rightarrow a \sum x_i^2 + b\sum x_i = \sum x_i y_i$$

$$\frac{\partial J}{\partial b} = \sum_{x_i=1}^{m}(y_i - a x_i - b) = 0 \rightarrow \frac{\partial J}{\partial b} = \sum y_i - a\sum x_i - bm = 0$$

Using Cramer's Rule,

$$\rightarrow a\sum x_i + bm = \sum y_i$$

$$a = \frac{\begin{vmatrix} \sum x_i y_i & \sum x_i \\ \sum y_i & m \end{vmatrix}}{\begin{vmatrix} \sum x_i^2 & \sum x_i \\ \sum x_i & m \end{vmatrix}} \Rightarrow a = \frac{m\sum x_i y_i - (\sum x_i)(\sum y_i)}{m\sum x_i^2 - (\sum x_i)^2}$$

→ +ve হলে → x বাড়ে, y↑
-ve → x↓, y↓

→ covariance of x & y
[এর এর মধ্যে x এর relation a হলো ?]

$$b = \frac{\begin{vmatrix} \sum x_i & \sum x_i y_i \\ \sum x_i & \sum y_i \end{vmatrix}}{\begin{vmatrix} \sum x_i^2 & \sum x_i \\ \sum x_i & m \end{vmatrix}} \Rightarrow b = \frac{\sum x_i y_i - (\sum x_i)^2(\sum y_i)}{(\sum x_i^2)(m) - (\sum x_i)^2}$$

~~y = ax + b~~
~~ky = kax + kb~~

~~* gradient descent use করলে কিছু চলে কোথায় যাবে~~

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \\ b \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \\ b \end{bmatrix} + \lambda \begin{bmatrix} \frac{\partial E}{\partial a_1} \\ \frac{\partial E}{\partial a_2} \\ \vdots \\ \frac{\partial E}{\partial a_n} \\ \frac{\partial E}{\partial b} \end{bmatrix}$$

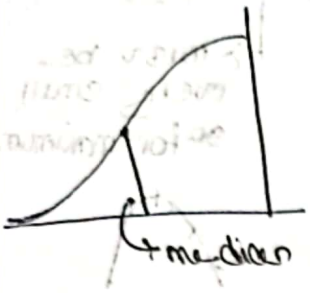
$\lambda \frac{\partial J}{\partial a}$

↳ must be very small so for minimum

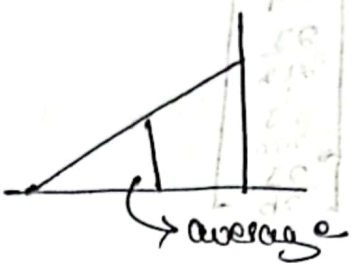
new a_1 b

old a_1 b

LAB 3



এরা graph হলে,
median use করা
better for missing
inputs.
average use করলে
বা value আসে



এরা graph হলে,
average use করা
better for missing
inputs,

$w = (X^T X)^{-1} X^T y$

X → all columns except y
y → survived column

pandas → data frame manipulate করে

~~1 no vector~~

1 hot vector

M	F
1	0
0	1

এভাবে make
করলে 1 হয়
M=1 হয় & F=0
হয় এই row তে

FINAL

LECTURE 11

TUESDAY

DATE: 01/08/23

LEARNER

* Learner: an algorithm which learns from a dataset

$y = x + \epsilon$

↳ error from normal distribution

* A strong learner can find out the hyperplane giving 95% accuracy.

* Realizability assumption: If a hyperplane can divide a dataset ~~into~~ seperating (O and X), it is realizable.

* % of women in the world population → True estimate

* % of women in a state → Sample estimate

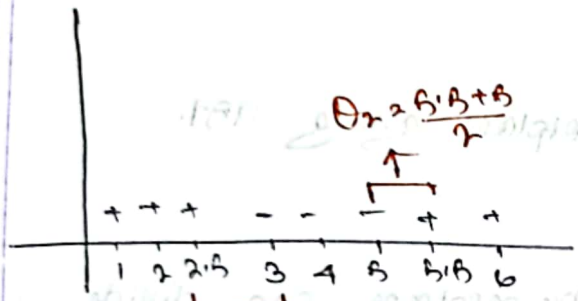
$R_D(h)$ → Error calculated over all data in the world
↓ true

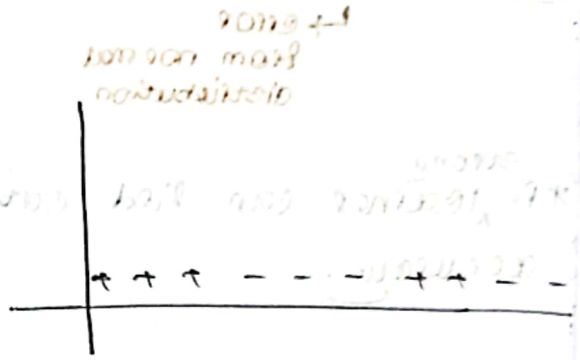
$R_S(h)$ → Error calculated over all data in a given area.
↓ sample

* Weak learner: accuracy $\frac{1}{2}$ (চান্স বেশি)
[for binary classification] error $\frac{1}{2}$ (চান্স কম)

* It is not possible to find out $R_D(h)$ directly.

Interval Function





* Not realizable, ~~so~~ as there is - after θ_2 interval. So, cannot be solved using interval function

$$h_{\theta_1,\theta_2,b}(x) = \begin{cases} -b & \text{if } \theta_1 \le x \le \theta_2 \\ +b & \text{otherwise} \end{cases}$$

$\theta_1 \le x \le \theta_2 = +1$

$-b = +1$

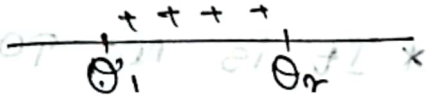
− − − − + + + + − − − −

1 2 3 4 5 6 7 8 9 10 11 12

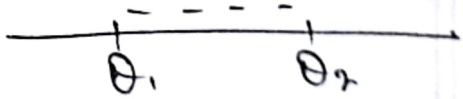
θ_1 θ_2

$h_{4.5,8.5,-1}(x)$

For $-b = +1$



For $-b = -1$



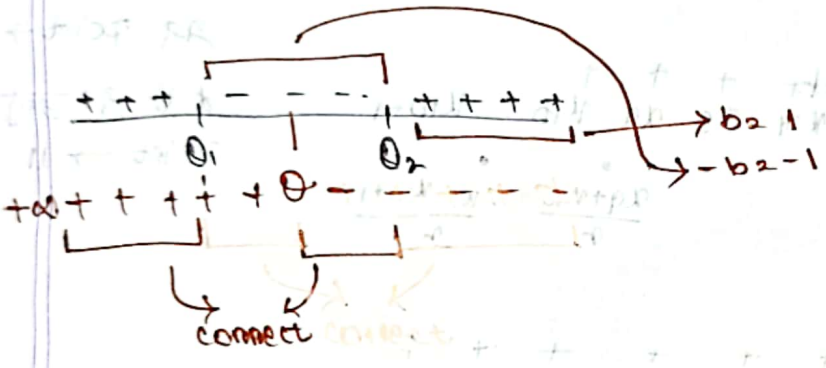
BOOSTING

*Boosting : একটা learner এরপর, তাদের opinion নিয়ে, তাদের weight দিয়ে এবং নিজে decision নিয়ে

$h_{\theta,b}(x) = b \cdot \text{sign}(\theta - x)$

$b \in \{-1, +1\}$

interval function — threshold function



* accuracy is not 50%, so we have to choose θ accurately.

[interval function]

threshold function

$+\infty$ + + + + + + + + + + $\theta = \infty$

$-\infty$ − − − − − − $\theta = -\infty$

− − − − − + + + θ → Here, accuracy is 67% which is poor.
This is weak learner.

* If we use many weak learners, then we can find a better accuracy → Ensemble learning.

* x কে increasingly sort করা লাগবে [sort in descending order]

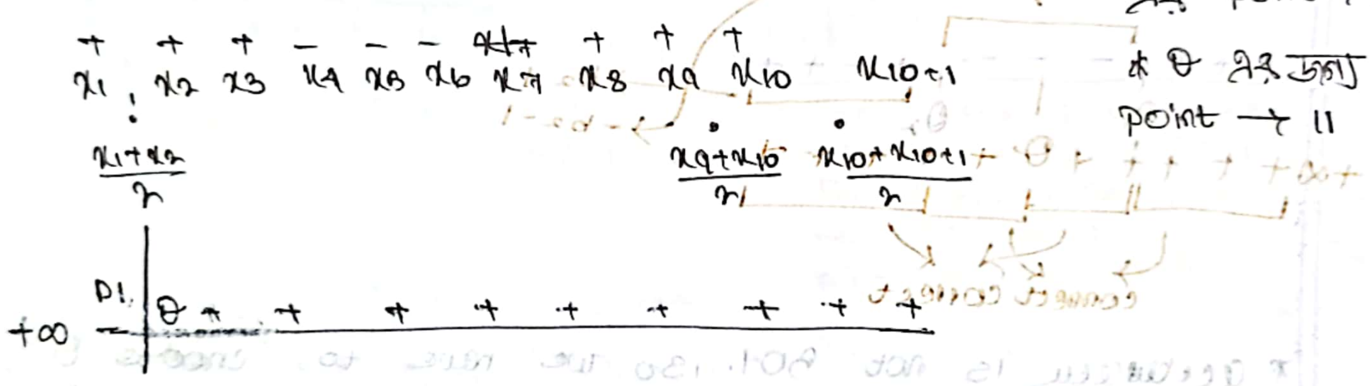
$$\overset{+}{x_1}\ \overset{+}{x_2}\ \overset{+}{x_3}\ \overset{-}{x_4}\ \overset{-}{x_5}\ \overset{-}{x_6}\ \overset{+}{x_7}\ \overset{+}{x_8}\ \overset{+}{x_9}\ \overset{+}{x_{10}}$$

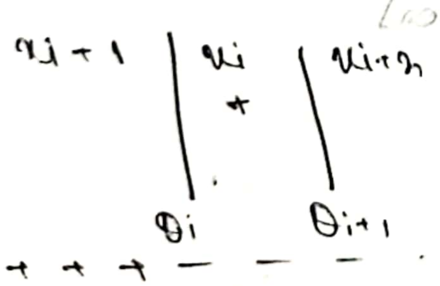
D1 D2

↓

x_i এর error actual error কে কমানোর জন্য গুরুত্বপূর্ণ
important

* Dataset এর point → 10
* θ এর জন্য point → 11





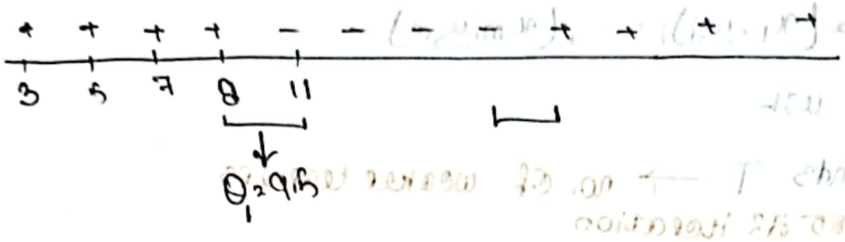
* Actual output of x is +ve and after checking using θ_i, x get −ve. By shifting θ_i to θ_{i+1}, we get +ve for x_i. So, there's no error (D1).

* Actual output of x is +ve and after using θ_i, x gets −ve. By shifting θ_i to θ_{i+1}, we still get −ve for x, so, there's an error (D1)

LECTURE 12

SUNDAY

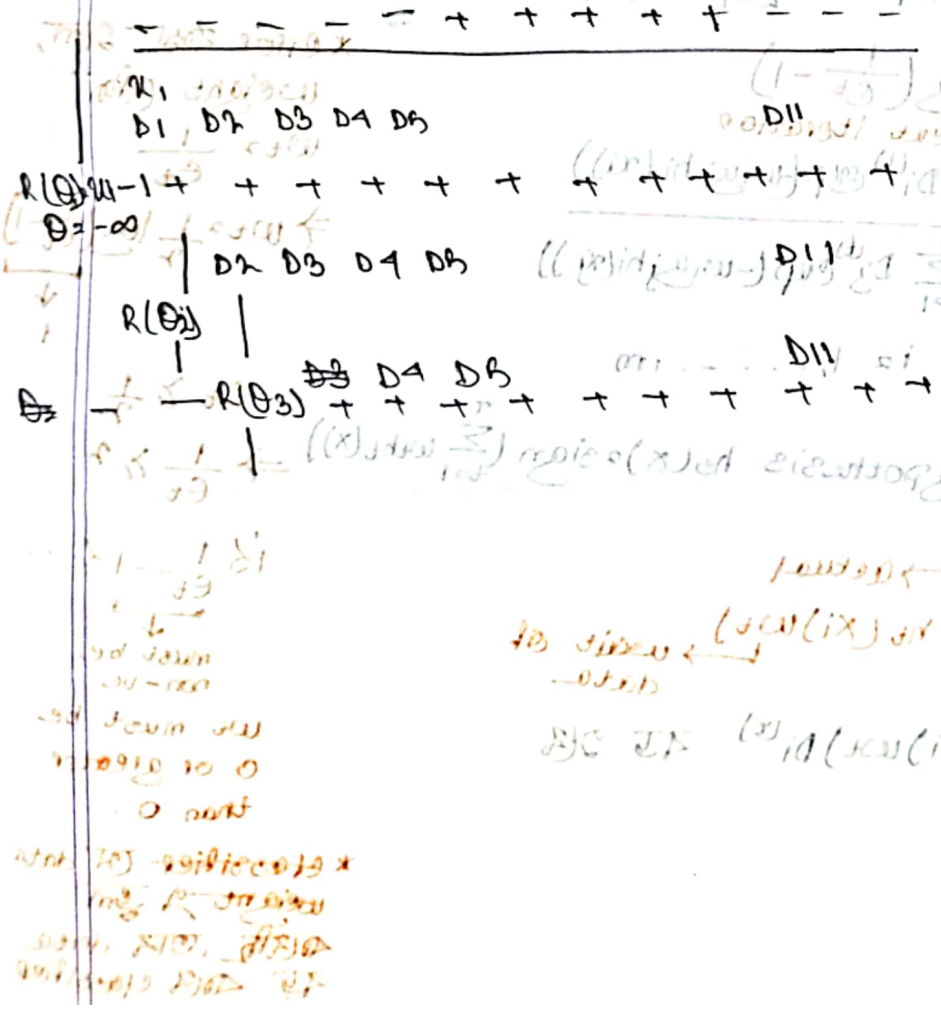
DATE: 06/08/23



For threshold function,

$h_{\theta,b}(x) = b \cdot \text{sign}(\theta - x)$

if $x > \theta$, sign$(\theta - x)$ is −ve.



$\theta_i = -\infty$, $b = 1$ → for threshold

$\theta_i = \frac{x_i + x_{i+1}}{2}$

* To increase the accuracy, we have to find θ_i upto maximum errors are resolved.

ADABOOST

Input:

training set $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$

weak learner WL

number of rounds $T \rightarrow$ no. of weaker learners

initialize $D^{(1)} = \left(\frac{1}{m}, \frac{1}{m}, \ldots, \frac{1}{m}\right)$ → প্রথম বার iteration

for $t = 1, \ldots, T$:

invoke weak learner $h_t = WL(D^{(t)}, S)$

compute $\epsilon_t = \sum_{i=1}^{m} D_i^{(t)} \mathbb{1}_{[y_i \neq h_t(x_i)]}$

let $w_t = \frac{1}{2} \log\left(\frac{1}{\epsilon_t} - 1\right)$

update $D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-w_t y_i h_t(x_i))}{\sum_{j=1}^{m} D_j^{(t)} \exp(-w_t y_j h_t(x_j))}$ → next iteration

প্রথম নাও ← data

for all $i = 1, \ldots, m$

output the hypothesis $h_s(x) = \text{sign}\left(\sum_{t=1}^{T} w_t h_t(x)\right)$

$D_i^{(t+1)} = D_i^{(t)} (-y_i h_t(x_i) w_t)$ → actual → weight of data

Error হলে $(-y_i h_t(x_i) w_t) D_i^{(t)}$ বড় হয়

* error কম হলে, weight বেশি

$w_t = \frac{1}{\epsilon_t}$

$\Rightarrow w_t = \frac{1}{2} \log\left(\frac{1}{\epsilon_t} - 1\right)$ → 1

$\epsilon_t > \frac{1}{2}$

$\Rightarrow \frac{1}{\epsilon_t} > 2$

যাতে $\frac{1}{\epsilon_t} - 1 > \ldots$ → must be non-ve

we must be 0 or greater than 0.

* classifier যে data weight এ ভুল করেছি, তার data বড় করে classified

$$D_i^{(t+1)} = D_i^{t} e^{-y_i h_t(x_i) w_t}$$

$$D_1^{(t+1)} = D_1^{t} e^{-y_1 h_t(x_1) w_t}$$

Total of all D_i = ...

$h_1(11) = +$ $h_2(11) = -$ $h_3(11) = +$

10   13   7

$(+1)(10) + (13)(-1) + (7)(+1) = +4$

↓ sign $h_t(x)$ ↓ weight w_t

$$R_S(h_{AB}) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}_{h_{AB}(x_i) \neq y_i} \leq e^{-2\lambda^2 T}$$

→ error থাকলে $R_S(h_{AB}) \geq 1$

→ $\frac{1}{e^{2\lambda^2 T}}$ [increasing T, decreases $R_S(h_{AB})$]

→ $R_S(h_{AB}) = 1$ (for error)

$f_T(x) = w_1 h_1(x) + w_2 h_2(x) + \ldots\ldots + w_T h_T(x)$ → total error

$f_0(x) = 0$

$f_1(x) = w_1 h_1(x)$

$f_2(x) = w_1 h_1(x) + w_2 h_2(x)$

$$\gamma_t = \frac{1}{m}\sum_{i=1}^{m} e^{-y_i f(x_i)} \quad \gamma_1 \frac{1}{m}$$

$$D_i^{(t+1)} = \frac{e^{-y_i f_t(x_i)}}{\sum_{j=1}^{m} e^{-y_j f_t(x_j)}}$$

$$D_i^{t} = \frac{e^{-y_i f_{t-1}(x_i)}}{\sum_{j=1}^{m} e^{-y_j f_{t-1}(x_i)}}$$

$$D_i^{t} = \frac{\left(e^{-w_t h_t(x_i)}\right) e^{-y_i f_{t-1}(x_i)}}{\sum_{j=1}^{m} e^{-y_j f_{t-1}(x_i)}}$$

$$f_t(x_i) = f_{t-1}(x_i) + w_t h_t(x)$$

$$f_t(x) = w_1 h_1(x) + \ldots\ldots + w_t h_t(x)$$

$f_{t-1}(x_i)$ $\quad$ $f_t(x_i)$

$$e^{-w_t y_i h_t(x_i)} \quad D_i^{t} = \frac{e^{-y_i f_t(x_i)}}{\sum_{j=1}^{m} e^{-y_j f_t(x_j)}}$$

LECTURE 13

TUESDAY

DATE: 08/08/23

SUPPORT VECTOR MACHINE

- binary classification

A → A or not A → cascading classification

B

C

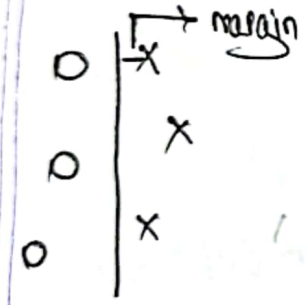
D

[Dataset 2 ভাগে ভাগ করা]

↓

Margin — the minimum distance between datapoint and hyperplane.

SVM — margin কে maximise করে



$\forall i \in [n], y_i(\langle w_i, x_i \rangle + b) \geq 0$

$(w_1x_1 + w_2x_2 + \dots + w_ix_i + b)$

$y_1(w_1x_{11} + w_2x_{12} + \dots + b) = \gamma_1$

$y_2(w_1x_{21} + w_2x_{22} + \dots + b) = \gamma_2$

$y_j(w_1x_{j1} + w_2x_{j2} + \dots + b) = \gamma_j$ ← smallest

$y_m(w_1x_{m1} + w_2x_{m2} + \dots + b) = \gamma_m$

↓

$$y_1\left(\frac{w_1}{\gamma_j}x_{11} + \frac{w_2}{\gamma_j}x_{12} + \dots + \frac{b}{\gamma_j}\right) = \frac{\gamma_1}{\gamma_j} \geq 1$$

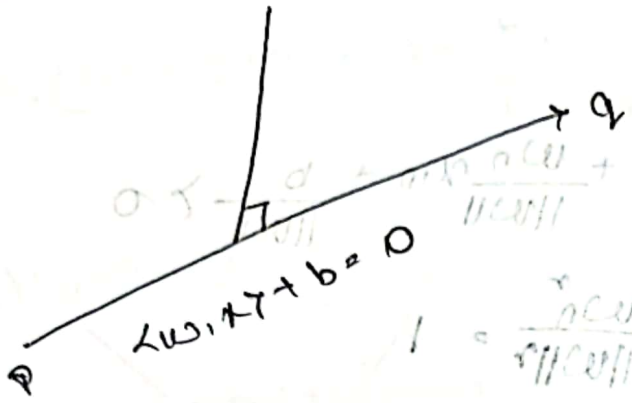
$$y_2\left(\frac{w_1}{\gamma_j}x_{21} + \frac{w_2}{\gamma_j}x_{22} + \dots + \frac{b}{\gamma_j}\right) = \frac{\gamma_2}{\gamma_j} \geq 1$$

$$y_j\left(\frac{w_1}{\gamma_j}x_{j1} + \frac{w_2}{\gamma_j}x_{j2} + \dots + \frac{b}{\gamma_j}\right) = 1$$

$$y_m\left(\frac{w_1}{\gamma_j}x_{m1} + \frac{w_2}{\gamma_j}x_{m2} + \dots + \frac{b}{\gamma_j}\right) = \frac{\gamma_m}{\gamma_j} \geq 1$$

$y_i(\langle w_i, x_i \rangle + b) \geq 1$

* Divide by γ_j (smallest value) ∴ values greater than smallest value would be greater than 1.



$\langle w \cdot p \rangle + b = 0$

$\langle w \cdot q \rangle + b = 0$

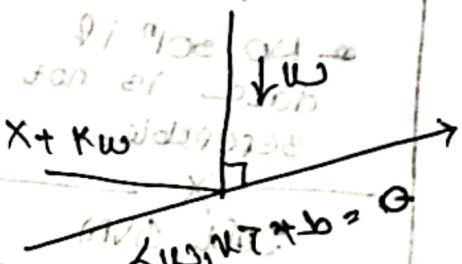
$\overrightarrow{pq} = q - p = (q_1 - p_1, q_2 - p_2, \ldots, q_n - p_n)$

$\langle w \cdot p \rangle = \langle w \cdot q \rangle$

$w_1 p_1 + w_2 p_2 + \ldots + b = w_1 q_1 + w_2 q_2 + \ldots + b$

$w_1 (q_1 - p_1) + w_2 (q_2 - p_2) = 0$

$\langle w \cdot \overrightarrow{pq} \rangle = 0$



$\langle w, x + kw \rangle + b = 0$

$\langle w, x \rangle + \langle w, kw \rangle + b = 0$

$\langle w, x \rangle + \langle w, w \cdot k \rangle + b = 0$

$\langle w, x \rangle + k\|w\|^2 + b = 0$

$k = \frac{-b - \langle w, x \rangle}{\|w\|^2}$

$d^2 = \langle kw, kw \rangle$

$d^2 = k^2 \langle w, w \rangle$

$= k^2 \|w\|^2 = \|w\|^2 \left| \frac{b + \langle w, x \rangle}{\|w\|^2} \right| \Rightarrow d = \frac{|b + \langle w, x \rangle|}{\|w\|}$

$$\|w\| = \sqrt{w_1^2 + w_2^2 + \dots + w_n^2} = 1$$

$$\frac{w_1}{\|w\|} x_{i1} + \frac{w_2}{\|w\|} x_{i2} + \dots + \frac{w_n}{\|w\|} x_{in} + \frac{b}{\|w\|} \geq 0$$

$$\frac{w_1^2}{\|w\|^2} + \frac{w_2^2}{\|w\|^2} + \dots + \frac{w_n^2}{\|w\|^2} = 1$$

* w and x আমরা কোন scale করলে আমরা পাব so that margin is always 1.

$$|\langle w, x \rangle + b| = 1$$

for

$$d = \frac{|\langle w, x \rangle + b|}{\|w\|} = \frac{1}{\|w\|}$$

↓ margin

Hard SVM

- সবার line হতে সবচেয়ে points are always ≥ 1 from it
- points are seperable
- NO soln if data is not seperable

X

Soft SVM

- Gives soln even if data is unseperable
- Points are not greater than 1 so we use $1 - \xi_i$

Soft SVM

$y_i(\langle w, x_i \rangle + b) \geq 1 - \xi_i$

↳ Hard SVM ला violate करेल

$\lambda \|w\|^2 + \frac{1}{m} \sum_{i=1}^{m} \xi_i$

↘ normal ↙ violation

ला minimize करेल

Error কে আমরা max (0, 1 − yi (<w, xi> + b)) এ express করি

max (0, 1 − yi (<w, xi> + b))

(+) 0 : B
↓
error

max 0 : B

$J = \frac{1}{m}\sum_{i=1}^{m}(y_i - \bar{y}_i)^2$ → Batch GD

Batch GD — সব point use করে error বের করি
Minibatch GD — random ভাবে একটা data
Stochastic GD — একটা point use করি

Stochastic GD → [actual gradient পাওয়া যায় না]
← estimated GD

$J = \frac{1}{m} \cdot \boxed{m} \; (y_{17} - \bar{y}_{17})^2$

↓
m দিয়ে multiply করি

* Adv : পুরো data load করা লাগবে না
পুরো loop run করা লাগবে না

Minibatch GD

$J = 10\sum(y_{17} - \bar{y}_{17})^2$

(10 Data used)

LECTURE 14 SUNDAY DATE: 13/08/23

SVM objective function

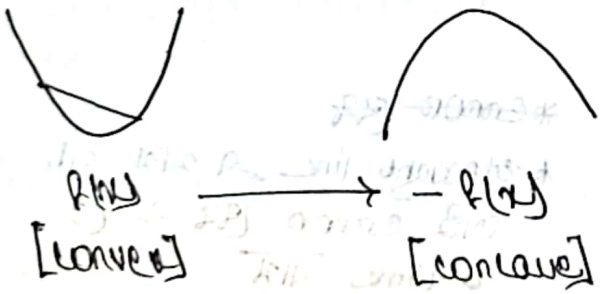
$\|w\|^2$ $\langle w.w \rangle$ $\frac{1}{2} w^T w$

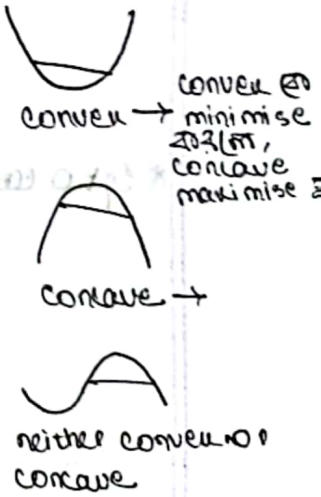
$y_i (w^T x_i + b) \geq 1 \rightarrow$ correct

↓ given ↓ predicted

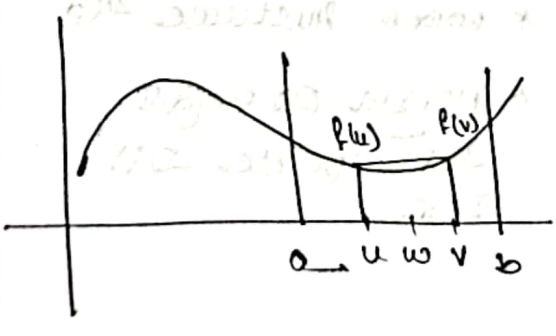
$0 \geq 1 - y_i (w^T x_i + b)$

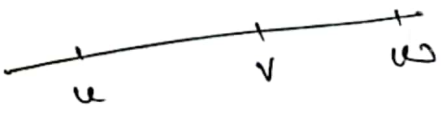
$$\min_{w,b} \frac{1}{2} w^T w + C \sum \max(0, 1 - y_i (w^T x_i + b))$$





* Convex is represented by a domain





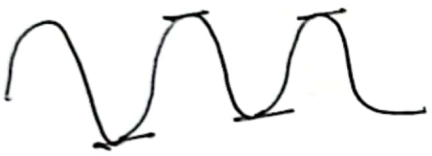
$w = \lambda u + (1-\lambda) v \rightarrow$ convex combination

$= \frac{1}{2} u + \frac{1}{2} v = \frac{u+v}{2}$ [u & v ka mean mean point]

~~$\lambda f(u) \rightarrow \lambda f(u) + (1-\lambda) f(v)$~~

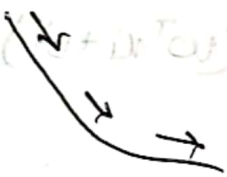
$f(\underbrace{\lambda u + (1-\lambda) v}_{w}) \leq \lambda f(u) + (1-\lambda) f(v)$ ⟶ convex

* Convex না হলে gradient এ problem হয়।



⟶ gradient descent changes

* Batch GD ⟶



* সব data points use করা লাগে, so error বের করতে time লাগে

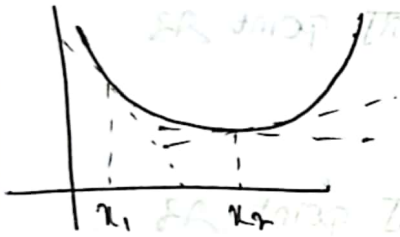
* Stochastic ⟶

* ~~error বের~~
* straight line এ যায় না, তাই error বের করতে time যায়
* প্রতিটা point এর GD বের করি, তাই straight line এ যায় না
* error fluctuate করে
* weight কে অনেক বার update করা যায়।

* An unique tangent line at the point of a function.

* tangent curve touch করতে।

* sub-tangent curve touch করতে কোন কাজে লাগে।



* যেখানে কোন function
কোন tangent থাকে।
so differentiate
করা যায় না।
so, sub gradient descent
use করি।

Sub-gradient descent

$$J^t(w) = \frac{1}{2} w_0^T w_0 + C \cdot N \max(0, 1 - y_i w^T x_i)$$

$$\frac{\partial J}{\partial w_j} = \frac{1}{2}(w_j) + C \cdot N_{max} \begin{cases} 0 & \text{if } y_i w x_i \geq 1 \\ -y_i x_{ij} & \text{else} \end{cases}$$

$w^T x_i$
$= w_1 x_{i1} + w_2 x_{i2} + w_3 x_{i3} + \dots + w_{n+1} x_{n+1}$
$\downarrow$
1

$w^T w$
$= w_1^2 + w_2^2 + \dots + w_{n+1}^2$

~~3rd~~

* SVM এ different learning rate use করি for each iteration

* এবং $\gamma_t = \frac{1}{t}$

* learning rate বেশি: এক point থেকে অন্য point এর jump টা বড়

* learning rate ছোট: এক point থেকে অন্য point এর jump টা ছোট

* time এর সাথে learning rate কমে

$$w_j^{(t+1)} = w^t - \gamma_t \frac{\partial J}{\partial w_j}$$

$$\frac{\partial J}{\partial w_j} = w_j + CN \begin{cases} 0 \\ -y_i x_{ij} \end{cases}$$

$$w_j^{(t+1)} = w_j^t - \gamma_t w_j = w_j(1-\gamma_t) \rightarrow \frac{\partial J}{\partial w_j} = 0$$

$$w_j^{(t+1)} = w_j - \gamma_t (w_j - (CN y_i x_{ij})) \rightarrow \frac{\partial J}{\partial w_j} = -y_i x_{ij}$$

$$= w_j - \gamma_t (w_j) + CN \gamma_t y_i x_{ij}$$

$$= (1-\gamma_t) w_j + \gamma_t CN x_{ij} y_i x_{ij}$$

$$\gamma_t = \frac{\gamma_0}{1+t\gamma}$$

SVM

$$\min_{w,b} \frac{1}{2} w^T w + C \sum \max(0, 1 - y_i (w^T x_i + b))$$

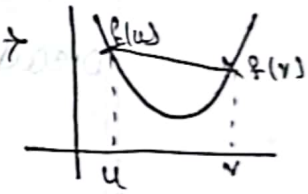
- $\frac{1}{2} w^T w$ → regularization term
  - maximise the margin
- C → hyper-parameter
  - controls trade off bet' large margin & small hinge loss
- $\sum \max(0, 1 - y_i (w^T x_i + b))$ → Empirical loss
  - Hinge loss
  - penalizes weight vectors that make mistakes

$w_0 \leftarrow w$
$w \leftarrow [w_0, b]$
$x_i \leftarrow [x_i, 1]$

* This function is convex in w, b.

* A function is convex when the domain is:—

$$f(\lambda u + (1-\lambda)v) \leq \lambda f(u) + (1-\lambda) f(v) \quad \lambda \in [0,1]$$



$f(x) \geq f(u) + \nabla f(u)^T (x - u)$

* Tangent lies below a convex function

Necessary condition:—
(i) x needs to be minimum
(ii) $\nabla f(x) = 0$

Convex functions

* How to proof convex function:

(i) Definition of convexity

(ii) 2nd derivative is +ve → 1D

(iii) 2nd derivate is +ve semi definite → vector functions

$f(x) = -x$	$f(x) = x^2$	$f(x) = \max(0, x)$

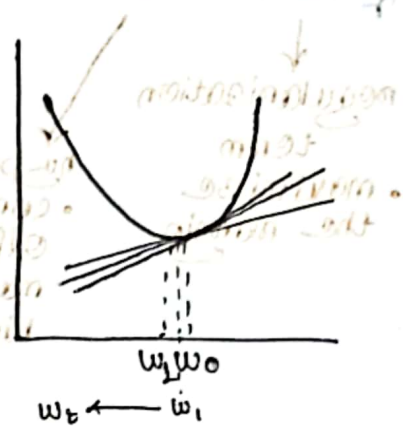
GRADIENT DESCENT

[To solve SVM optimization problem]

* Start with an initial guess, w_0
* Compute gradient of $J(w)$ at w^t.
  $\rightarrow \nabla J(w^t)$
* Update w^t to w^{t+1} by taking a step in the opposite direction.

$$w^{t+1} = w^t - \eta \nabla J(w^t)$$

(η ↓ learning rate)



Disadv: Gradient of SVM requires summing of entire training set → SLOW

STOCHASTIC GRADIENT DESCENT [preferable for huge data]

* A training data is given, $S = \{(x_i, y_i)\}$, $x \in \mathbb{R}^n$, $y \in \{-1, 1\}$
* Initialize $w^0 = 0 \in \mathbb{R}^n$
* For epoch = 1 . . . T:
  1. Pick a random example (x_i, y_i) from S
  2. Repeat (x_i, y_i) to make a full dataset & take derivative of SVM objective at the current w^{t-1} to be $\nabla J^t(w^{t-1})$

$$J^t(w) = \frac{1}{2} w_0^T w + C \cdot N \max(0, 1 - y_i w^T x_i)$$

(Hinge Loss; N → num of training examples)

  3. Update: $w^t \leftarrow w^{t-1} - \gamma_t \nabla J^t(w^{t-1})$ (γ_t → learning rate)
  4. Return w

Adv: Can converge to minimum of J if γ_t is too small.

Stochastic GD

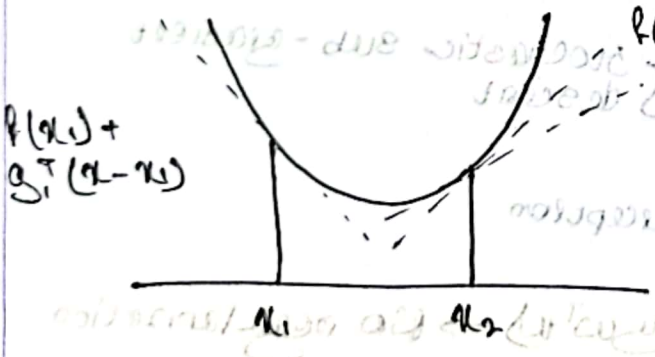
GD

Stochastic GD has many more updates than GD, which are less expensive.

SUB GRADIENTS

A sub-tangent at a point is any line lies below the function at the point.
A sub-gradient is the slope of that line.

$f(y) \geq f(x) + g_1^T(y-x)$ for all y



* f is differentiable at x_1, so there's a tangent at x_1. $\therefore g_1$ is the gradient.
* g_2 and g_3 are subgradients

$J^t(w) = \frac{1}{2}w_0^T w + C \cdot N \max(0, 1 - y_i w^T x_i)$

* solve max
* compute gradient

$$\nabla J^t = \begin{cases} [w_0; 0] & \text{if } \max(0, 1 - y_i w_i^T x_i) = 0 \\ [w_0; 0] - C \cdot N y_i x_i & \text{otherwise} \end{cases}$$

STOCHASTIC SUB-GRADIENT DESCENT FOR SVM

Given a training set $S = \{(x_i, y_i)\}$, $x \in \mathbb{R}^n$, $y \in \{-1, 1\}$

(i) Initialize $w^0 = 0 \in \mathbb{R}^n$

(ii) For epoch = 1 ... T; → need to shuffle for each epoch

(i) For each training example $(x_i, y_i) \in S$

if $y_i w x_i \leq 1$ → learning rate

$w \leftarrow (1 - \gamma_t)[w_0; 0] + \gamma_t C N y_i x_i$

else

$w_0 \leftarrow (1 - \gamma_t) w_0$

(iii) Return w

PERCEPTRON V/S STOCHASTIC SUB-GRADIENT DESCENT

if $y_i w^T x_i \geq 1$

update $w_0 \leftarrow (1 - \gamma_t) w_0$ } stochastic sub-gradient descent

if $y_i w^T x_i \leq 0$

update $w \leftarrow w + \eta y_i x_i$ } perceptron

$L_{perceptron}(y, x, w) = \max(0, -y w^T x)$ → No regularization

$L_{Hinge}(y, x, w) = \max(0, 1 - y w^T x)$ → Regularization (SVM optimizes hinge loss)

* Perceptron and stochastic has similar frame work with different objectives.
* Perception cannot maximise margin width. Can have a margin.

CONVERGENCE AND LEARNING RATES

$$\gamma_t = \frac{\gamma_0}{1 + \frac{\gamma_0 t}{c}} \Rightarrow \gamma_t = \frac{\gamma_0}{1+t}$$

↓

learning rate.

$t \downarrow, \gamma_t \uparrow$

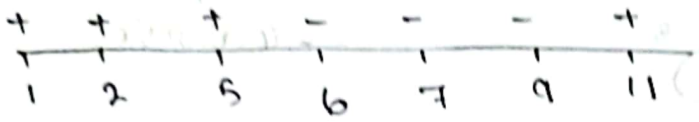
LECTURE 15

SUNDAY

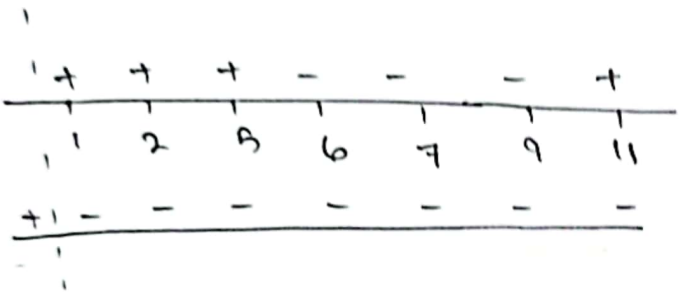
DATE: 20/08/23

Quiz 2 solution

(1,+), (2,+), (5,+), (6,-), (7,-), (9,-), (11,+)

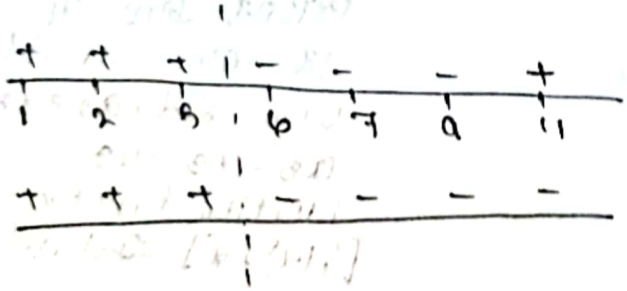


a = 0.5, b = 1



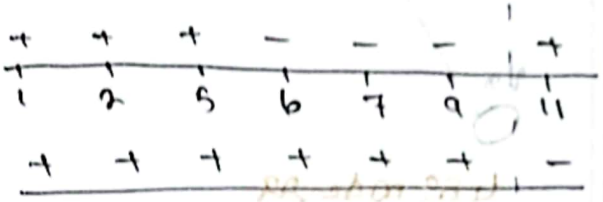
error = 4

a = 5+6/2 = 5.5, b = 1



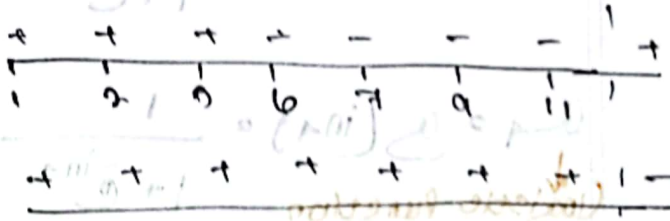
error = 1

a = 9+11/2 = 10, b = 1



error = 4

a = 11.5, b = 1



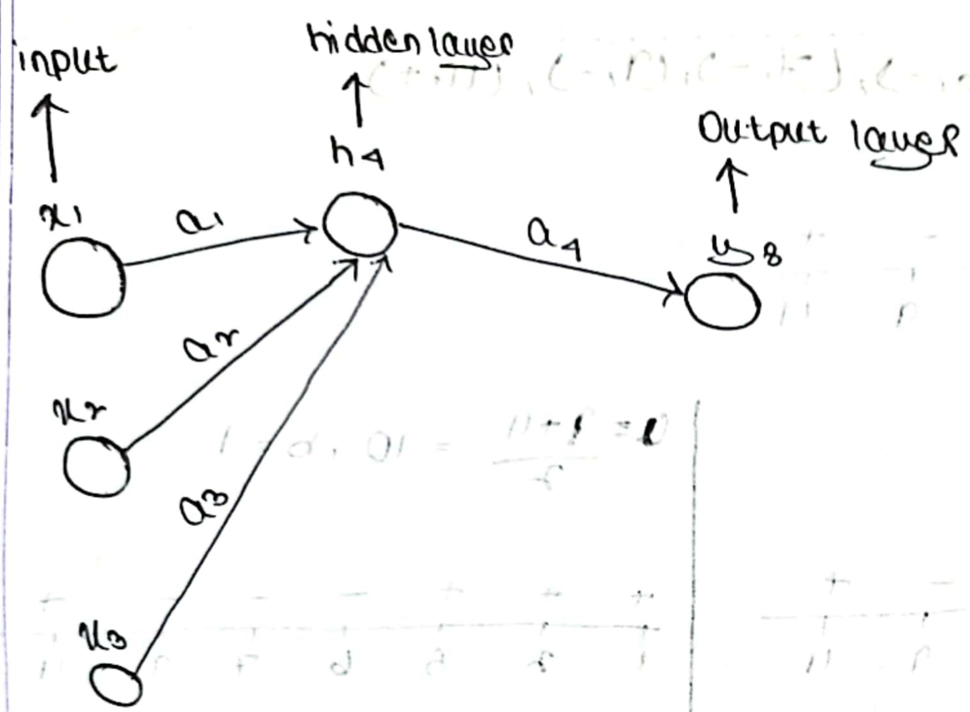
error = 5

possible values of a = 8

minimum error = 1

value of a = 5.5, b = 1 for minimum error

NEUTRAL NETWORK



* Hidden layer থাকতেও পারে, নাও পারে

* a_1, a_2, a_3, a_4 → activation

4 নং node এর activation কে ↑ 4 নং অংশ ধরা হয়;

$$in_4 = w_{1,4}a_1 + w_{2,4}a_2 + w_{3,4}a_3 + b_4 \rightarrow \text{for hidden layer}$$

($a_1 \downarrow x_1$, $a_2 \downarrow x_2$, $a_3 \downarrow x_3$)

* input layer থেকে hidden layer এ যাওয়ার জন্য a_1 এর value x_1 এর value হবে, $a_2 = x_2$, $a_3 = x_3$ এবং identity function [$f(x) = x$] এটা লাগে

$$a_j = g(in_j) = \frac{1}{1+e^{-in_j}}$$

$$a_4 = g(in_4) = \frac{1}{1+e^{-in_4}}$$

↓ logistic function দিয়ে complex activation create করেছি)

$$in_8 = w_{4,8}a_4 \rightarrow \text{for output layer}$$

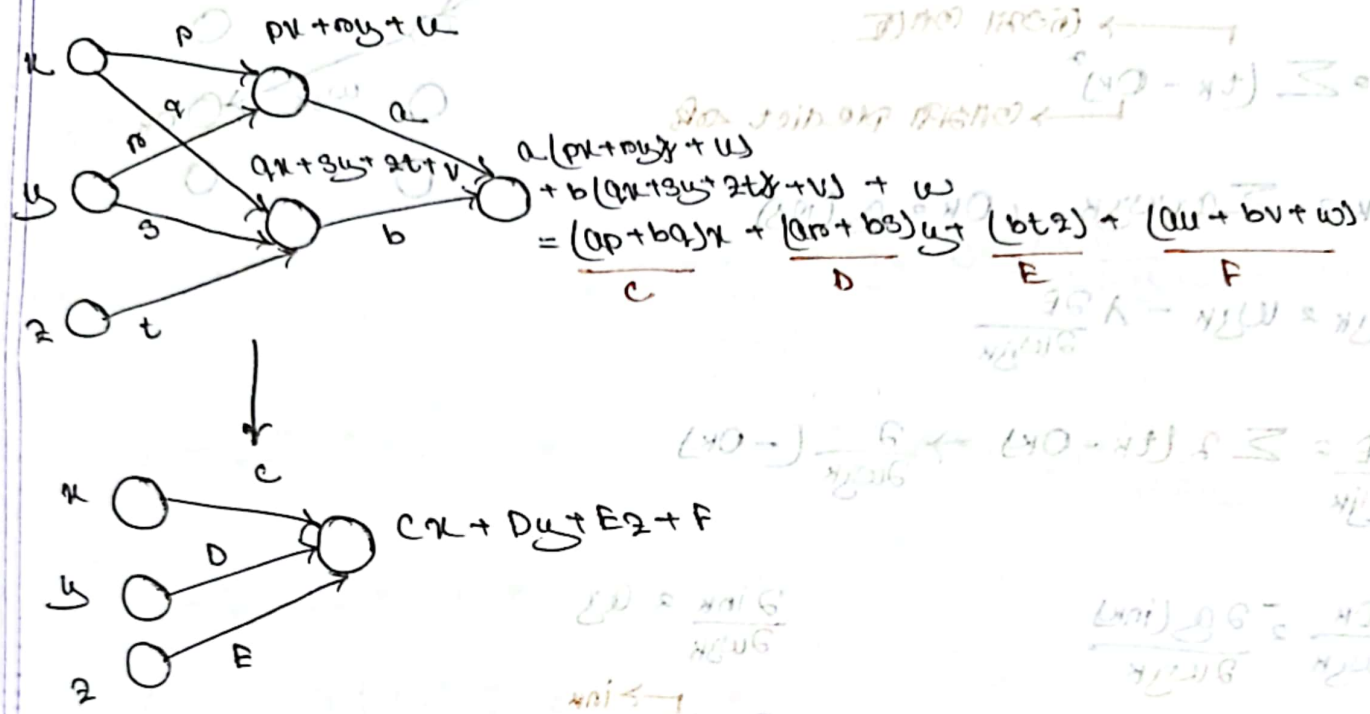
$$\downarrow \frac{1}{1+e^{-in_4}}$$

* Hidden layer থেকে output layer যাওয়ার জন্য $a_4 = \frac{1}{1+e^{-in_4}}$ [logistic regression] এটা লাগে

LECTURE 16

TUESDAY

DATE: 22/08/23



For activation, we use :-

(i) logistic function for binary classification

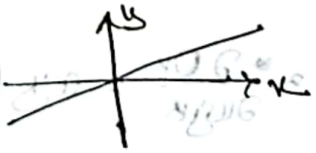
(ii) softmax for multinomial

(iii) identity function for linear

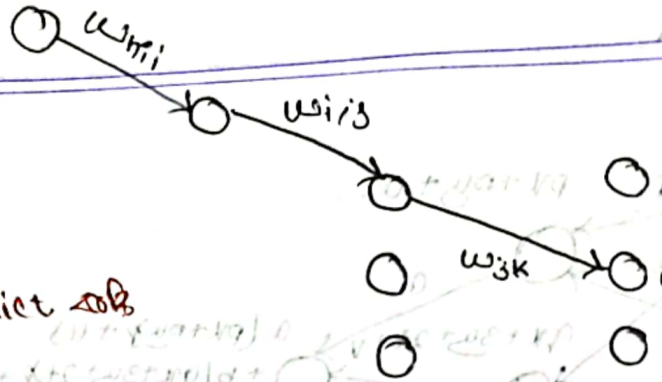
(iv) Threshold function for perceptron

* Neural network is more useful for multiple layers/inputs.

* Identity function; input যা, output তাই



* Identity function এটা, যাতে non-linearity show করে।



BACK PROPAGATION

$E = \sum (t_k - O_k)^2$ → (error) → (actual output)

$in_k = \sum a_j w_{jk}$, $O_k = g(in_k)$

$w_{jk} = w_{jk} - \lambda \frac{\partial E}{\partial w_{jk}}$

$\frac{\partial E}{\partial w_{jk}} = 2(t_k - O_k) \rightarrow \frac{\partial}{\partial w_{jk}} (-O_k)$

$\frac{\partial O_k}{\partial w_{jk}} = \frac{\partial g(in_k)}{\partial w_{jk}}$

$= \frac{\partial in_k}{\partial w_{jk}} \cdot \frac{\partial g(in_k)}{\partial in_k}$

$= a_j g(in_k)(1 - g(in_k))$

$\frac{\partial in_k}{\partial w_{jk}} = a_j$

$\frac{\partial g(x)}{\partial in_k}$ → in_k

$= \frac{\partial}{\partial x} \frac{1}{1+e^{-x}}$

$= \frac{\partial (1+e^{-x})^{-1}}{\partial x}$

$= (-1)(1+e^{-x})^{-2}(-e^{-x})$

$= \frac{e^{-x}}{(1+e^{-x})} \cdot \frac{1}{(1+e^{-x})}$

$= (1 - g(x)) g(x)$

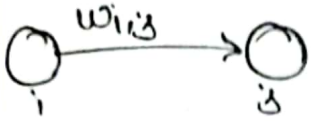
$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial O_k} \cdot \frac{\partial O_k}{\partial w_{jk}} = a_j g(in_k)(1 - g(in_k))$

$w_{jk} = w_{jk} - \lambda \frac{\partial E}{\partial w_{jk}}$

$w_{jk} = w_{jk} - \lambda (2(t_k - O_k)) a_j (1 - g(in_k)) g(in_k)$

$g'(in_k) = (1 - g(in_k)) g(in_k)$, $\Delta_k = (t_k - O_k) g'(in_k)$

$w_{j,k} = w_{j,k} - \Delta_k a_j$



$$E = \sum (t_k - o_k)^2$$

$$\frac{\partial E}{\partial w_{i,j}} = -2 \sum (t_k - o_k) \frac{\partial o_k}{\partial w_{i,j}} = -2 \sum (t_k - o_k) \left(\frac{\partial in_k}{\partial w_{i,j}} g'(in_k) \right)$$

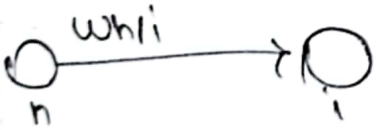
$$\frac{\partial E}{\partial w_{i,j}} = -2 \sum \Delta_k w_{j,k} \frac{\partial a_j}{\partial w_{i,j}} = -2 \sum \Delta_k w_{j,k} \frac{\partial w_{in_j}}{\partial w_{i,j}} g'(in_j)$$

$$= -2 \sum \Delta_k w_{j,k} a_i g'(in_j)$$

$$\frac{\partial E}{\partial w_{i,j}} = -2 a_i g'(in_j) \sum \Delta_k w_{j,k}$$

$$\Delta_j = g'(in_j) \sum \Delta_k w_{j,k}$$

$$w_{i,j} = w_{i,j} + \lambda a_i \Delta_j$$



$$\frac{\partial E}{\partial w_{ni}} = -2 a_n g'(in_i) \sum \Delta_j w_{i,j}$$

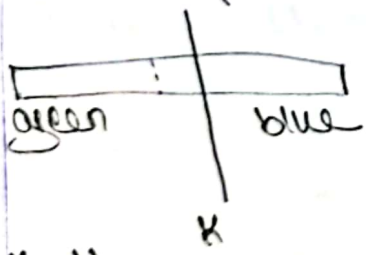
$$\Delta_i = -2 (in_i) \sum \Delta_j w_{i,j}$$

$$w_{n,i} = w_{ni,i} + \lambda a_n \Delta_i$$

K-NEAREST NEIGHBOURS (KNN)

* K এর value দেওয়া থাকবে,
* training data set & testing data থাকবে,
* loop চালাতে হবে from 1 to no. of rows in training data set to find out the distance between training data and test data.
* Then distance গুলো সাজানো sort করতে হবে
* K value অনুযায়ী distance গুলো নিতে হবে (যেমন K = 5)
* এই 5 টা distance এর মধ্যে যার vote বেশি তার under এ test data থাকবে. For
* For example, 3 টা +ve/Yes/1 and 2 টা -ve/No/0 আছে, so test data is +ve.

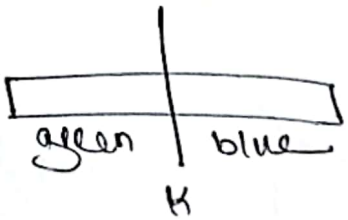
For example:



K = 11

* 7 blue points
* 6 green points

So, test data lies under the blue points.



K = 12

* 6 blue points
* 6 green points

So, either decision নিতে হবে Or randomly একটা pick করে নিতে হবে

$$\text{distance } (d) = \sqrt{\sum_{i=1}^{n} (x_i - q_i)^2}$$

x_i → training data, q_i → test data

LAB 6

→ 2D array কে 1D array বানানো

→ K- সংখ্যক label আছে

→ প্রত্যেক class এর জন্য score বানাতে হবে

$x_1 \; x_2 \; \cdots \; x_n$

$S_1 = w_{11}x_1 + w_{21}x_2 + \cdots + w_{n1}x_n + b_1$

$S_2 = w_{12}x_1 + w_{22}x_2 + \cdots + w_{n2}x_n + b_2$

$\vdots$

S_K

একটা weight array use করলে সব weight same হয়ে যায়।

So, weight এর matrix use করি

→ X হচ্ছে image যাকে flatten করেছি, তার জন্য probability বের করা লাগবে।

$S_1 \quad S_2 \quad \cdots \quad S_K \qquad S = \sum e^{i}$

$\frac{e^{S_1}}{S}, \frac{e^{S_2}}{S}, \cdots, \frac{e^{S_K}}{S}$ → probability

$\hat{Y} = P_1 \quad P_2 \quad \cdots \quad P_K$

↓ prediction

$Y = 0$

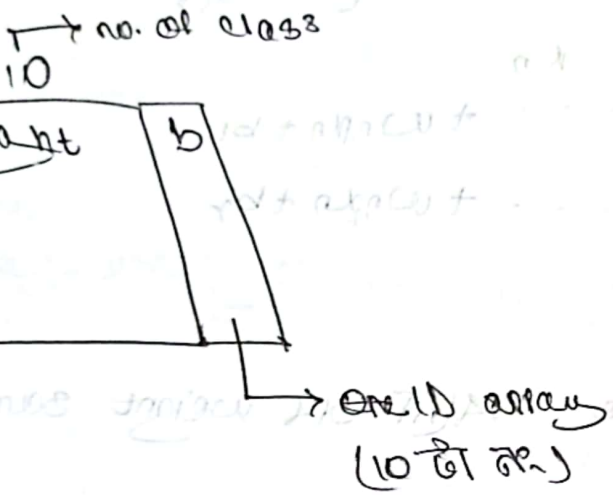
M → total data, K → total label

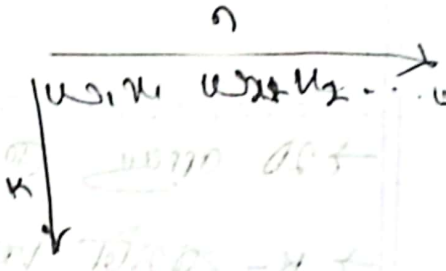
loss = ~~$\sum y_j \log \hat{y}_j$~~ $-\sum_{i=1}^{M} \sum_{j=1}^{K} y_{ij} \log \hat{y}_{ij}$

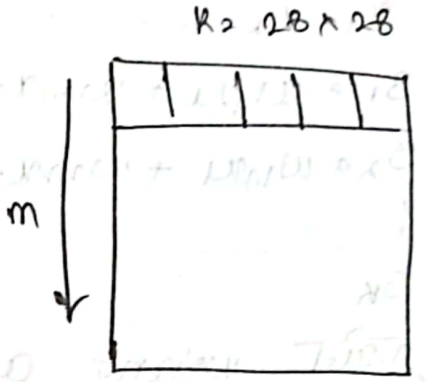
*weights কে update করতে হয়

$$\frac{\partial J}{\partial w_{pij}} = -\sum_{i=1}^{m}\sum_{j=1}^{k} y_{ij}$$

↓ কোন class

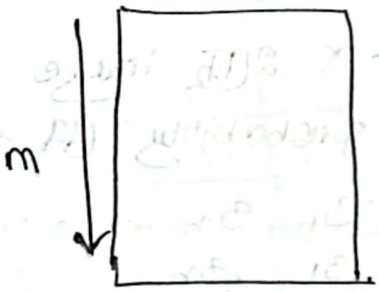






$$w = w - \frac{1}{m} * np.dot(X^T, (pred - one\_hot))$$

$$b = b - \frac{1}{m} * np.sum(pred - one\_hot)$$



আউটপুট prediction

---

* X কে reshape করতে

*label কে one_hot vector বানাতে হয়

Training data : $\{(x_i, y_i)\}$

Test data : x

$$\text{softmax}(x, x_i) = \frac{e^{-x_i^T x}}{S}$$

$$\downarrow$$

$$\frac{e^{-x_1^T x}}{S} + \frac{e^{-x_2^T x}}{S} \cdots + \frac{e^{-x_m^T x}}{S}$$

Data no.
x_1
x_2
.
.
.
x_m

$$\sum_{x_i \in \text{Soft}} \text{softmax}(x, x_i) y_i \gtrless 0 \rightarrow +1$$
$$\rightarrow -1$$

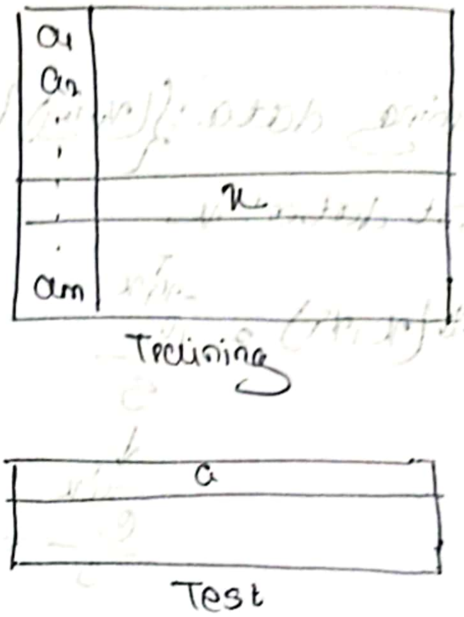
* Data কে centralise করা লাগবে

$a_1 = \frac{a_1 - \mu}{\sigma}$

$a_2 = \frac{a_2 - \mu}{\sigma}$

⋮

$a_m = \frac{a_m - \mu}{\sigma}$



Closest ১টা data ছাড়া বাকি সব data এর weight 0.

$(+1)(0) + (-1)(0) + (+1)(1) + (-1)(1) + (+1)(1) + (+1)(1) + (-1)(1) + .$

$x_1, x_2 \; \cdots \; x_m \rightarrow$ train

$a \rightarrow$ test

$\|x_1 - a\|^2 + \|x_2 - a\|^2 + \cdots + \|x_m - a\|^2$

কম distance যার হবে তার weight এর মানগুলোর বেশি হবে

$-\|x_1 - a\|^2 - \|x_2 - a\|^2 - \cdots - \|x_m - a\|^2 \rightarrow$ score হিসাবে

$\frac{e^{-\frac{1}{2}\|x_1 - a\|^2}}{S} \quad \frac{e^{-\frac{1}{2}\|x_2 - a\|^2}}{S} \quad \cdots \quad \frac{e^{-\frac{1}{2}\|x_m - a\|^2}}{S} \rightarrow$ softmax

$$\frac{e^{-\frac{1}{2}\|x_1 - a\|^2}}{S}$$

$$\|x_1 - a\|^2 = \|x_1\|^2 + \|a\|^2 - 2x_1^T a$$

$\|x_1\|^2$ → centralise করার জন্য norm constant

$\|a\|^2$ → এর মানগুলো same

বাদ দিয়ে দেয়া যায়

$$= -2x_1^T a$$

$$\frac{e^{-\frac{1}{2}\|x_1 - a\|^2}}{S} \Longrightarrow \frac{e^{x_1^T a}}{S}$$

$$\frac{e^{x_1^T a}}{S} \quad \frac{e^{x_2^T a}}{S} \quad \cdots \quad \frac{e^{x_m^T a}}{S}$$

score(1) score(2) ... score(m)

(score(1) x 1) + (score(2) x -1) + ... (score(m) x -1) → যেটা +ve আসবে survived, -ve আসবে not survived.

batch normalization