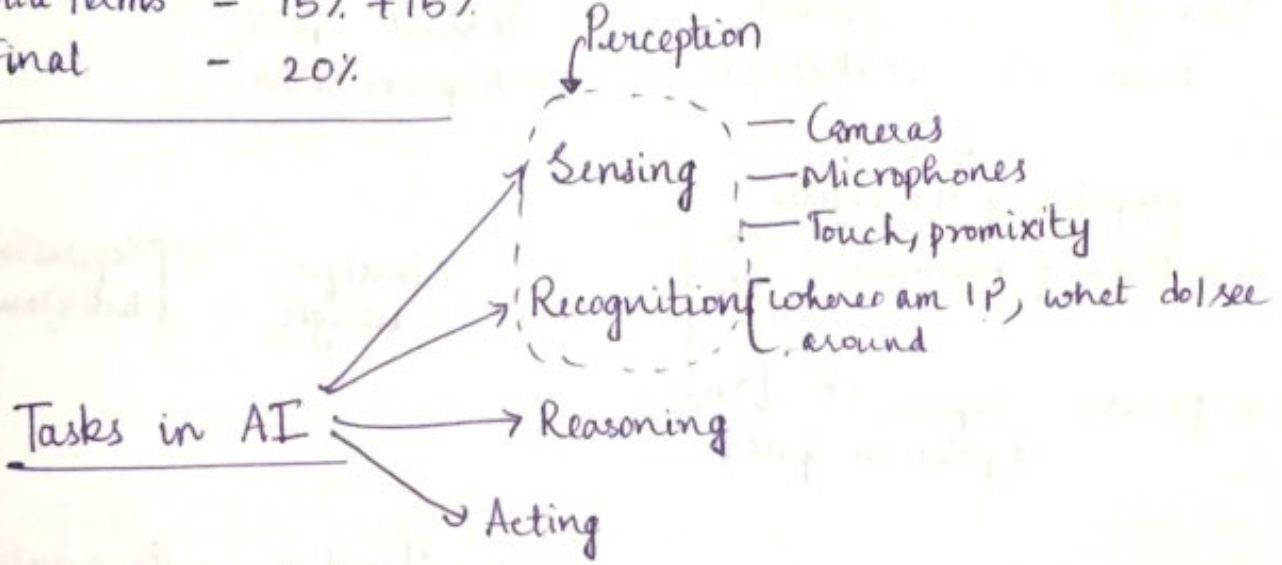
STATISTICAL METHODS IN A.I

Reference Books:

> Pattern Classification – Duda Hart and Stork
> Machine Learning – A Probabilistic Perspective by Kevin Murphy
> Neural Networks – Simon Haykin
> AI – A Modern Approach – Russell and Norvig

Grading Schemes –

Mini Project – 20%
Homeworks – 30%
Mid Terms – 15% + 15%
Final – 20%



Recognition – Identification of a pattern as a member of a category we all know, or familiar with.

"Re" + "Cognize"

associating it with a class of objects } "examples"

Pattern – an entity vaguely defined, that could be given a "name"

→ Same sample might be classified as different classes.

"Class" : collection of "similar" objects
→ defined by class samples

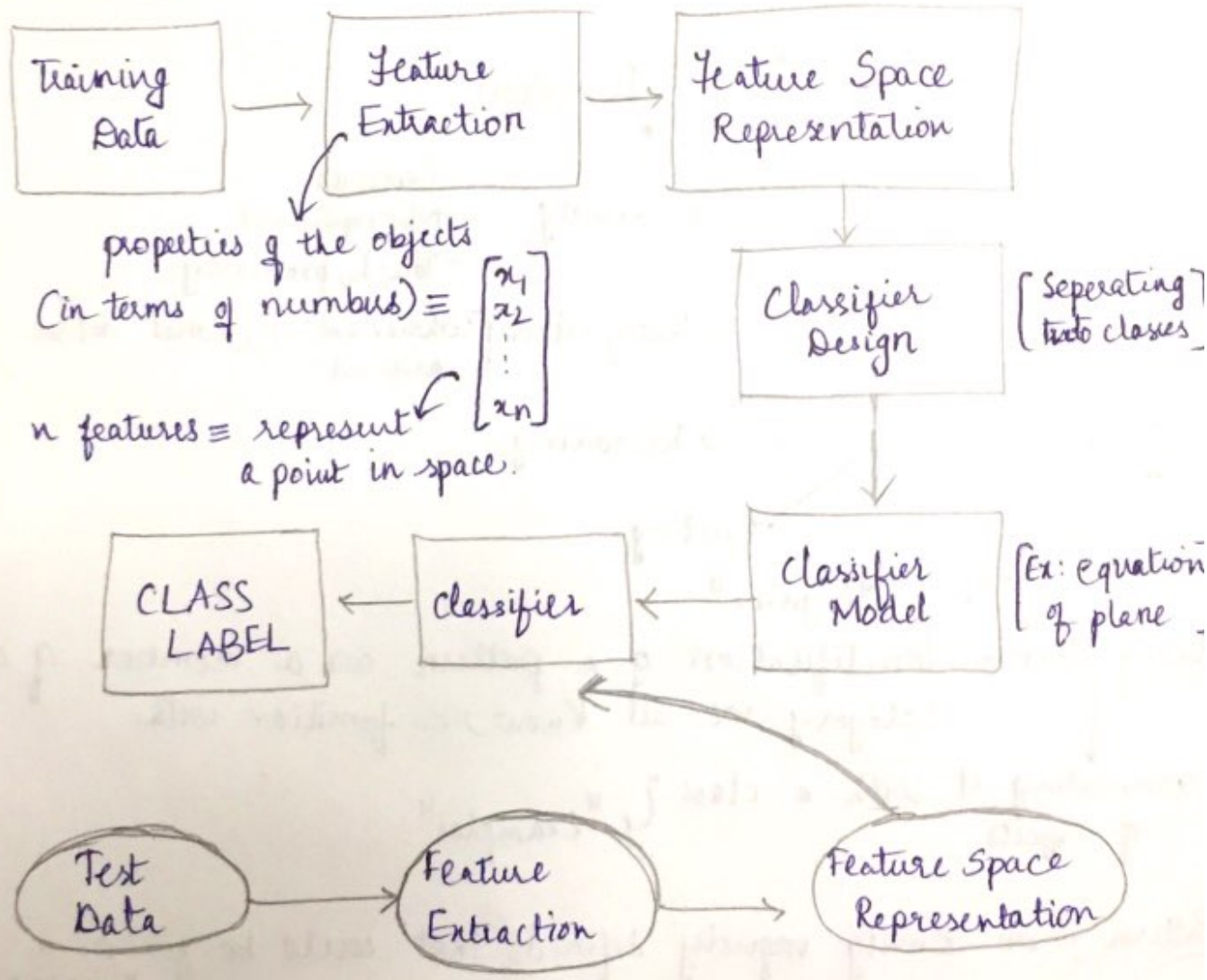
Pattern Recognition – inferring a generality from a few examples

↳ system "learns" to tell whether or not an object belongs to a class.

* class → Inter-class variability
  → Intra-class variability
  ↓
  represented as $[\omega_1, \omega_2, \ldots \omega_n]$

"train/teach the machine with a large dataset"

Pattern Recognition Process :-



* class – modeled by a probability density function, $P(x)$

↓

class-conditional ≡ $P(x/\omega_1)$

Gaussian Distribution ≡ $p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \sim N(\mu, \sigma^2)$

Mathematics :- Revision

→ Linear Algebra :

$f(x) \equiv f(\alpha x + \beta y) = \alpha f(x) + \beta f(y)$ ↷ for a function to be linear

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1$$
$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2$$
$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3$$

↓

Basic Representation ≡ matrices and vector form

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \qquad AX = B$$

A — X (3D-vector) — B

(Norm/Length)

Size of Vector ≡ $X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$ ↷ $\|X\|$

$|x|$ ↷ absolute value.

$$\|x\|_2 = \sqrt[2]{x_1^2 + x_2^2 + x_3^2}$$

$$\|x\|_p = \sqrt[p]{|x_1|^p + |x_2|^p + |x_3|^p + \dots |x_k|^p} \equiv L_p\text{-norm}$$

L_p-norm : $\|x\|_p = \sqrt[p]{\left(\sum_{i=1}^{n} |x_i|^p\right)}$

↓ doesn't have any physical significance.

Total distance covered ≡ L_1-norm ⇒ $\|x\|_1 = \sum_{i=1}^{n} |x_i|$

L_0-norm ≡ $\|x\|_0 = \sum_{i=1}^{n} |x_i|^0$ ↷ number of non-zero dimensions / entries in the vector.

L_∞-norm $\equiv$ $||x||_\infty$ = | Maximum value of the entries |

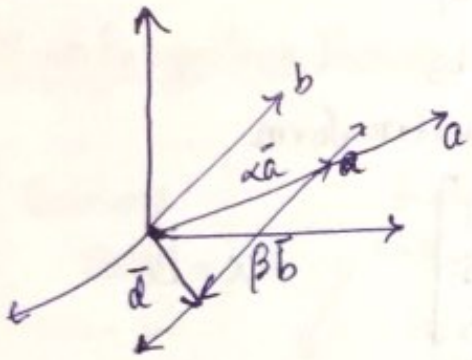
for Ex: $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$

$L_1 = 3$
$L_2 = 2.24$
$L_3 = 2.08$
$L_4 = 2.03$
$\vdots$
$L_{10} = 2.00019$

⇒ Span of a set of vectors –

Linearly independent ≡ when any combination of two vectors donot result in the third vector

For Ex: $c = \alpha a + \beta b$ (c is dependent on a, b)



$\bar{d} = \bar{a}\alpha + \bar{b}\beta$

If the vectors can be used in expressing the space ↓ are linearly independent, then they can span the whole space.

$$M = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix}$$

Det of M = |M| ≠ 0, if rows are linearly independent ↓ form a basis

if Rank ≥ 2, then out of 3, only 2 vectors are linearly independent & the other one is dependent on the first two.

↓ dimensionality of the subspace spanned.

as in feature ⇒ typically, vectors are "column vectors".

⇒ Orthonormal Basis — Normal + Perpendicular basis (linearly independent)

↓ perpendicular (90°) [Dot product of orthogonal vectors = 0]

↳ |Norm| = 1

$a \perp b \equiv a.b = ab\cos\theta = 0$

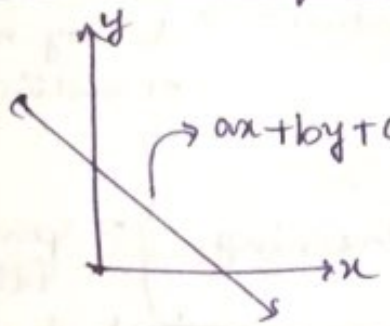
⇒ **DOT PRODUCT** ≡ $\begin{matrix} X \\ \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \end{matrix} \begin{matrix} Y \\ \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \end{matrix} \Rightarrow X.Y \Rightarrow x_1y_1 + x_2y_2 + x_3y_3 = X^TY$ (Inner Product)

$$XY^T = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} [y_1\ y_2\ y_3] = \begin{bmatrix} x_1y_1 & x_1y_2 & x_1y_3 \\ x_2y_1 & x_2y_2 & x_2y_3 \\ x_3y_1 & x_3y_2 & x_3y_3 \end{bmatrix}$$ (outer Product)

⇒ **CROSS PRODUCT** ≡ $(X,Y) = X \times Y = \begin{vmatrix} i & j & k \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{vmatrix}$

$a = \bar{X} \times \bar{Y}$ → perpendicular to both X & Y

⇒ **Representation of Line :-**



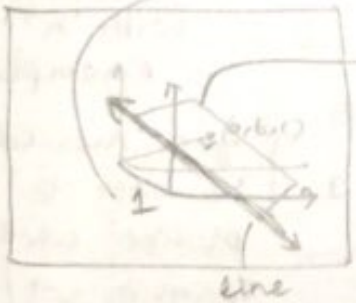
$\begin{bmatrix} a \\ b \\ c \end{bmatrix}$ ω $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ x

is it a vector or a line? [3D vector]

how do you relate both?

$\omega^T x = 0$ ↑ Space (augmented vector)

If ω is scaled, the plane doesn't change and so does the line.



Distance of a point from a plane → Normalise ω and take the dot product.

Linear Transformation ⟺ Matrix Multiplication

↓ by inverting the matrix, we can invert the transformation

⇒ **Eigen Values and Eigen Vectors**

$M = U \Theta V^T$ [λ] eigen values

U, V^T → orthonormal vectors

rows = eigen vectors

physical entity ——— represented as a vector

$x_1 \ldots x_n \Rightarrow$ vectors

with $d \equiv$ dimension

where $x_i \in \mathbb{R}^d$

$\begin{bmatrix} \\ \\ \\ \end{bmatrix}$ d — dimension

observations

physical entities are represented differently in different domains.

Main Problem

For Ex: $x_i \longrightarrow y_i \in \{0,1\}$ (say, spam or not)

$\{-1,+1\}$

$\{1,2,\ldots k\}$ (say, professional, personal, ... etc) → k-class.

$y_i \in R \equiv$ Real, continuous measure.

It is called as "**Regression**"

classification → Can be a binary one or multiclass.

representation of an email $\in \mathbb{R}$

Given, (x_i, y_i), $i = 1, \ldots N$ — y_i: spam or not

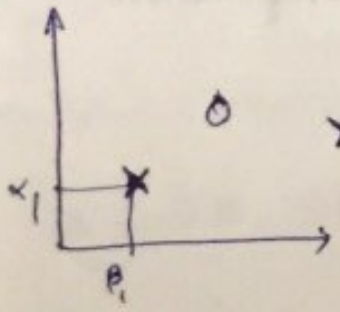
examples for learning/training

[where $x_i \in \mathbb{R}^d$, $y_i \in \{0,1\}$] } ≡ Spam filter to be built with "n" examples

Find a function, $f: x \to y$

So when given a new email, $x \equiv f(x) \to 0$ or 1 [you should be able to predict whether spam or not]

x_1	x_2	x_3	
$\begin{bmatrix} \alpha_1 \\ \beta_1 \end{bmatrix}$	$\begin{bmatrix} \alpha_2 \\ \beta_2 \end{bmatrix}$	$\begin{bmatrix} \alpha_3 \\ \beta_3 \end{bmatrix}$	→ 2 features with different values } can be represented on a plane.
0	1	0	(spam / not spam)



$\longrightarrow f() \equiv f(x) = w^T x + b$

$\begin{bmatrix} x^1 \\ x^2 \end{bmatrix}$

$f(x) = w_1 x^1 + w_2 x^2 + b$

Now the problem is reduced to "finding the value of w_1, w_2, b".

→ Find w, w_2, b ?

↓

OPTIMIZATION problem ≡ Find w that best solves the problem.

↓

we need an "objective function"

Data (x_i, y_i) pairs

↓

Problem ≡ Find best w → which minimizes

↓

Objective function ≡ Error / Loss

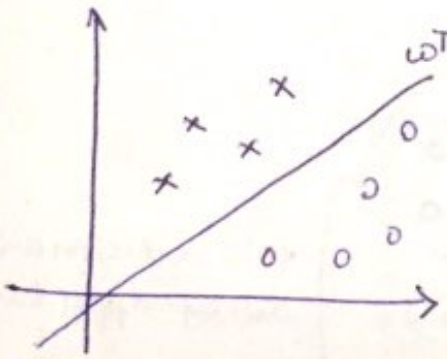
$f(x) = w^T x + b$.

→ Regression

→ Classification.

Changing the loss function can also changes the Optimization Algorithm. ↑ because N>>

Representation ≡



x – Spam

o – Not Spam

Case ①

$f(x) = w^T x + b = 0$.

Spam if $f(x) < 0$

Not Spam if $f(x) \geq 0$

Discriminent function

Case ②:

$f_1(x) = w_1^T x + b_1$

$f_2(x) = w_2^T x + b_2$

$f_1 - f_2 > 0$.

Spam if $f_1(x) > f_2(x)$

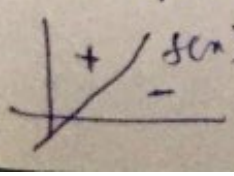
Not Spam if $f_1(x) \leq f_2(x)$

$f_1 - f_2 = (w_1 - w_2)^T x + b_1 - b_2$

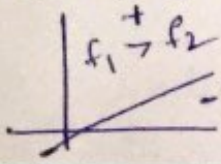
$= w_3^T x + b_3$.

In the 1st Case ≡

$f(x)$ ≡ function discriminates spam from not spam



In the 2nd Case ≡



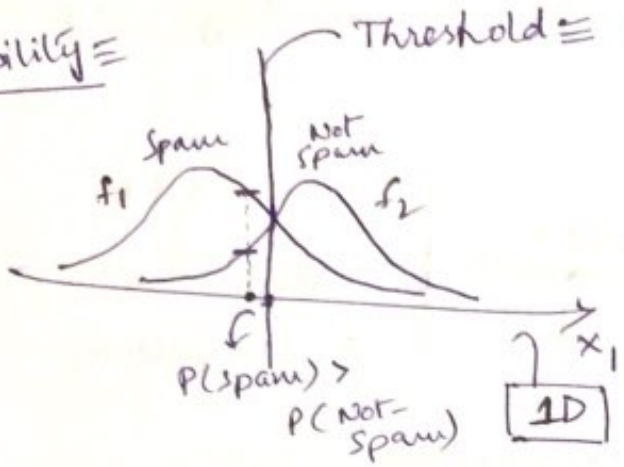
? (when to use this approach)

the second case: Assume $f_1(x) \equiv$ Probability of being spam.

$f_2(x) \equiv$ Probability of being not spam

both the viewpoints are equivalent in this case.

Probability ≡



$P(\text{spam}) = P(\text{Not spam})$

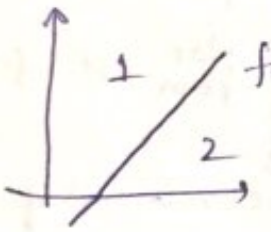
if $x < T \equiv$ Spam

$\geq T \equiv$ Not Spam.

So ≡ Spam

These both viewpoints go in parallel. √ (probabilistic and Non probabilistic Approach)

2-Class classification Problem:

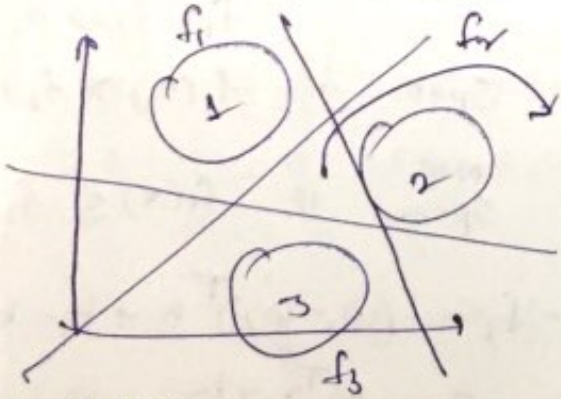


1 if $f(x) > 0$

2 if $f(x) \leq 0$

this decision rule doesn't apply here.

what about a 3-class classification?



what about this sample space.

"No-man"

∴ "multi-class classification problem"

Decision Rule:

1: $f_1 > f_2, f_3$

2: $f_2 > f_1, f_3$

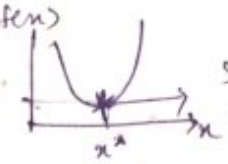
3: $f_3 > f_2, f_1$

Output – Discrete (Not Continuous)

↓ discrete

becomes an optimization problem.

Loss function → Optimization Problem

Optimization → Convex ≡ 

single unique minima.

Optimization → Non-convex

≡ 

maybe the best minima to get

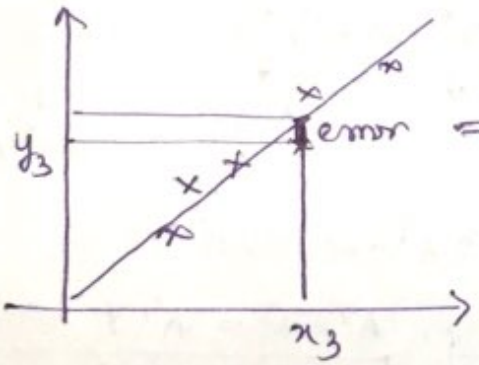
we can solve it.

how to find?
— brute force search.

Regression ≡

$(x_1, y_1) \dots (x_n, y_n)$ where $y_i \in R$.

$f(x) = w^T x + b$



$\Rightarrow \varepsilon_3 = w^T x_3 + b - y_3$

Error ≡ $\Sigma = \sum_{i=1}^{N} \left[y_i^0 - (w^T x_i^0 + b) \right]^2$

predicted value by the function.

$\Downarrow$ min Σ w, b

$\Downarrow$

$\min_w \left(\sum_{i=1}^{N} (y_i - w^T x_i)^2 \right)$

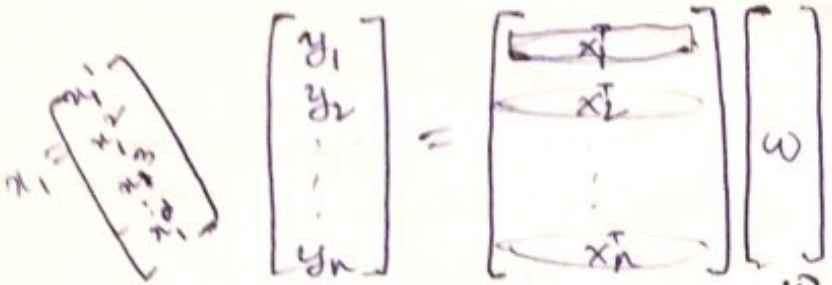
can be +ve / -ve [above or below the line]

$w^T x$ → $\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}^T \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + b$ ↓ How to remove b ↓

$w' = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ b \end{bmatrix}$ $x' = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

x is augmented with 1.

Minimum Square Error (MSE)



$N \times 1$ $\quad N \times d$ $\quad d \times 1$

$(N \times (d+1))$ with the augmentation

(Predicted Y) $\hat{Y} = AW$

$N >>> d$

$$\sum_{i=1}^{N} (y_i^{\circ} - w^T x_i^{\circ})^2 \equiv (\text{in metrix form}) \equiv [Y - AW]^T [Y - AW]$$

Sigma is gone

$(N \times 1)^T \quad (N \times 1)$

$\equiv (1 \times 1)$

To minimize this equation $\Longleftarrow Y^T Y + (AW)^T AW - 2(AW)^T Y$

[Differentiate and equate = 0.] $\Longrightarrow \frac{\partial}{\partial w}\left[Y^T Y + (AW)^T AW - 2(AW)^T Y\right]$

$= \frac{\partial}{\partial w}\left[Y^T Y + W^T A^T AW - 2W^T A^T Y\right] = 0.$

$Y^T Y \to 0 \quad W^T A^T AW \to 2A^T AW \quad 2W^T A^T Y \to 2A^T Y$

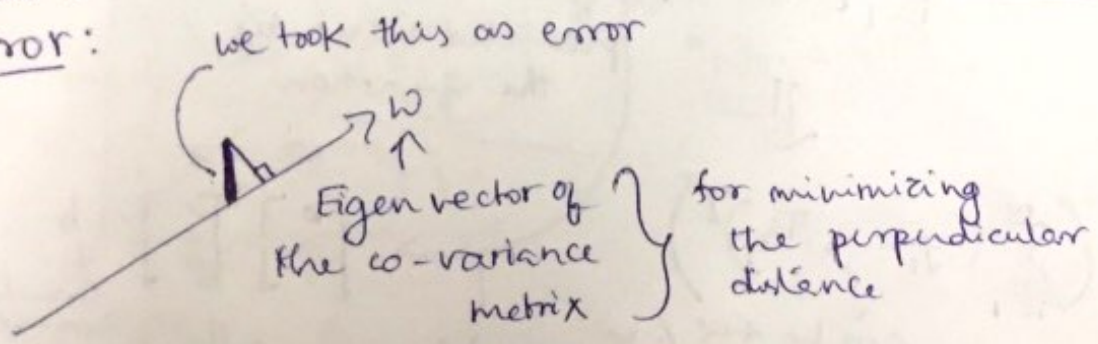
$2A^T AW - 2A^T Y = 0.$

$A^T AW = A^T Y$

$$\Longrightarrow \boxed{W = (A^T A)^{-1} A^T Y}$$

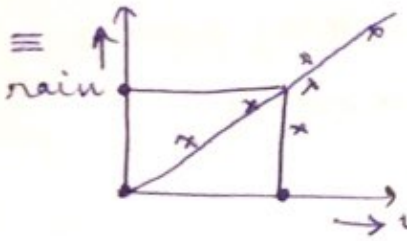
Tom Minka = Metrix Differentiation.

Issues with Error:



→ A Size $\equiv N \times d$ → could be extremely large $\equiv$ "Gradient Descent"

Examples of Regression ≡



$Y = ax + b$

$\Rightarrow x = \begin{bmatrix} x \\ 1 \end{bmatrix}$ (augmented vector)

$\begin{bmatrix} a \\ b \end{bmatrix}^T \begin{bmatrix} x \\ 1 \end{bmatrix}$

$y = w_1 x_1 + w_2 x_2 + w_3$

$\Rightarrow x = \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} \quad \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$ → this even works for non-linear

$Y = f(x) = w^T z$ ← $\begin{bmatrix} x \\ 1 \end{bmatrix}$

$\begin{bmatrix} x^2 \\ x \\ 1 \end{bmatrix} \equiv \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$

$w_1 x^2 + w_2 x + w_3 = 0$

- not restrictive
- due to augmentation (not passing thru origin ✓)
- New vectors ≡ Non linear functions embedded in $w^T x = 0$.

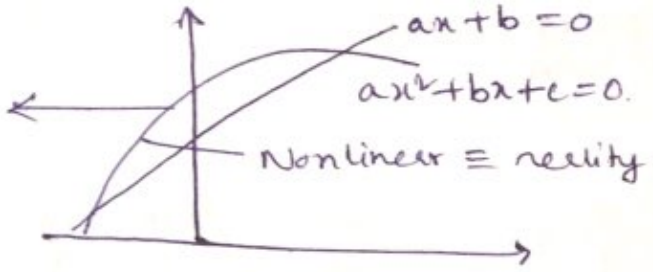
representation of $x \in$

line fitting

can be represented as ≡

$x = \begin{bmatrix} x^2 \\ x \\ 1 \end{bmatrix} \quad w = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$

↳ $w^T x = 0$.



* Main goal:- Find "w", for a function of the form ≡ $w^T x$

Not restrictive

$\Rightarrow Y = Aw \Rightarrow$ To find $w \equiv$

N×1 | N×d | d×1

$w = A^{-1} Y$ (doesn't work. as long as A is not singular and square matrix ($N = d$))

→ $x \to z$ can be mapped

$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_x \Rightarrow \begin{bmatrix} x_1^2 \\ x_1 x_2 \\ x_2^2 \\ x_1 x_3^2 \\ \vdots \end{bmatrix}_z$ these are not independent.

$w = \boxed{(A^T A)^{-1} A^T} Y$ (pseudo inverse of A.)

approach ≡ Minimum Square Error

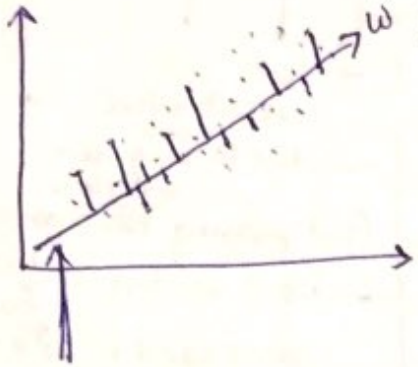
→ Doesn't minimize perpendicular (orthogonal) dist.

* $(A^T A)^{-1}$ ≡ computational storage

— Data comes incrementally (data is always not offline)

w – needs to be incrementally updated.

⇒ Orthogonal Distance Minimizing → find P, a point
→ find w, the direction.



$x_i \Rightarrow i = 1 \ldots N$ (set of points)

Let a point P be the point which has the least square error distance from all the points.

$$E = \sum_{i=1}^{N} |x_i - p|^2 = \sum_{i=1}^{N} [x_i - p]^T [x_i - p]$$

eigenvector for minimum, $\frac{\partial E}{\partial p} = 0$.

from the covariance matrix.

(passing thru the mean)

$$\frac{\partial}{\partial p} \sum_{i=1}^{N} \left(x^T x + p^T p - 2 p^T x_i \right)$$

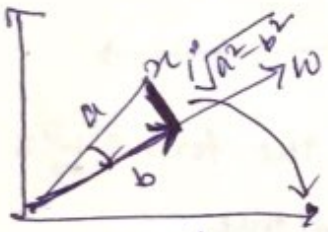
$x^T x \to 0$, $p^T p \to 2p$, $2 p^T x_i \to 2x_i$

$$= \sum_{i=1}^{N} 2p - 2x_i = 0.$$

$$\sum_{i=1}^{N} P = \sum_{i=1}^{N} x_i$$

$$P \cdot N = \sum_{i=1}^{N} x_i$$

$$P = \frac{\sum_{i=1}^{N} x_i}{N}$$



Normalized.

$$w^T w = 1$$

Norm of x's are fixed.

$$\sum_{i=1}^{N} \left(\|x_i\|^2 - (w^T x_i)^2 \right)$$

independent of w.

$$\text{Minimize} \equiv \sum_{i=1}^{N} -(w^T x_i)^2 \quad \text{s.t } w^T w = 1$$

$$\text{Maximize} \equiv \sum_{i=1}^{N} (w^T x_i)^2$$

$(Xw)^T Xw$

$w^T X^T X w$

$(w^T x)^T (w^T x)$

$x^T w w^T x$? wrong.

Max: $w^T X^T X w$

← look at regression.

Maximize $\equiv$ $w^T x^T x w$ st $w^T w = 1$

w is eigen vector of $x^T x \equiv$ PCA (covariance matrix)

(largest) eigen value

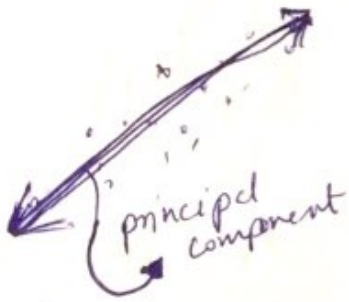
$\Downarrow$ why?

$Ax = \lambda x$

$Max\ (w^T x^T x w)$ $\rightarrow$ (λw)

$w^T \lambda w$.

To Maximize this quantity $w^T \lambda w$, we have to take largest value of λ



$\Downarrow$

Largest eigenvalue

$\max (w^T A w)$

$\Rightarrow w^T \lambda w \equiv \lambda w^T w = \boxed{\lambda}$

(λ: eigen value; $w^T w = 1$)

— Direction is given by w, where w is the eigen vector corresponding to the largest value of $x^T x$.

$\Downarrow$

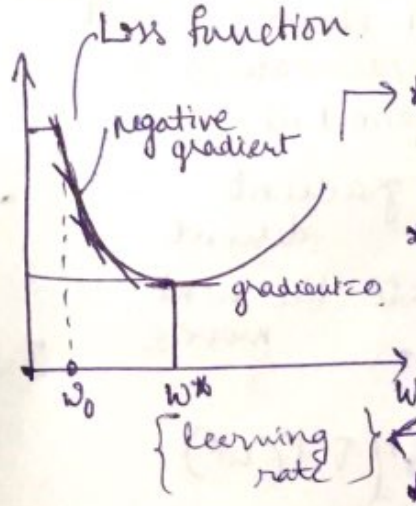
Mainly used in PCA.

Gradient Descent $\Rightarrow$

(x_i, y_i), $i = 1 \ldots N$.

$\sum (y_i - f(x_i))$

loss function = need to be minimized



* Randomly initialize w
* Move on the negative grad $w^{n+1} \leftarrow w^n - \eta \nabla J$
* Repeat steps until you reach termin^n.

* Goal
  - objective
* Data (x, y)
* Optimization
  - closed form Algo
  - Iterative Algo $\rightarrow$ gradient Descent

Given (x_i, y_i), $i = 1, \dots n$.

Find $y = f(x)$ such that it fits the existing (and future) data or minimize a loss/error function.

$f(x) = w^T x$ } Assumed it to be a linear function.

$\rightarrow y_i \sim f(x_i) \quad \forall i$

↙ ↘

R $\{0,1\}$

C

$$\begin{bmatrix} x^1 \\ x^2 \\ x^3 \\ x^4 \end{bmatrix} = w_1 x^1 + w_2 x^2 + w_3 x^3 + w_4 x^4$$

$w^T x \equiv$ Not restrictive

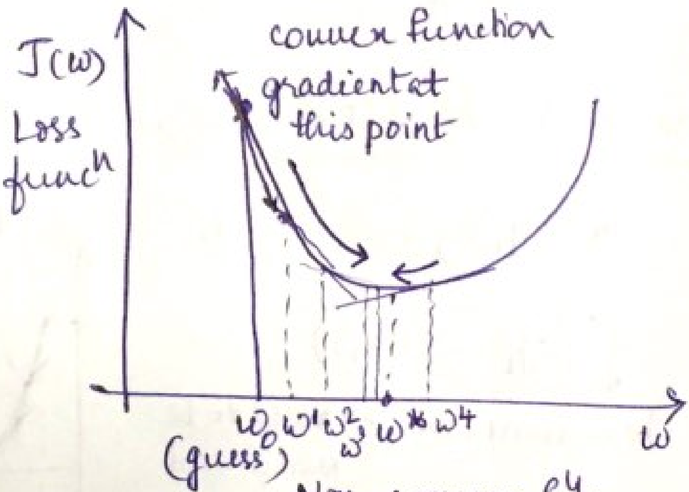
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \curvearrowright \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_1 \\ x_2 \\ 1 \end{bmatrix}$$

2d → 5d

→ Find the w suiting these dimensions/features.

5d → In five dimension, it is linear.

Gradient Descent:



> Make a random guess.

↓

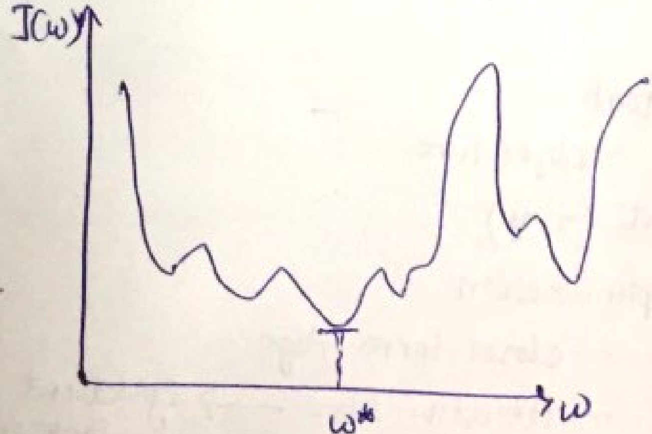
you should either increase w/decrease w to reach optimal w.

> check the gradient descent and calculate the next move ≡

$$w = w^0 - \eta \nabla J(w^0)$$

η = amount of movement.

Non-convex fn



At the minimum;

$\Delta J(w) = 0$ and

w, doesnt change $\longrightarrow$ Algorithm converges

May not reach $\nabla J(w^0) = 0$, we can put an error constant threshold.

In nonconvex function; the minimum (local) that we get maynot be the best one. by using the same equation

$\nabla J(w_0) < 0.001$

changing it doesnt give much of an advantage

What is η(eta)? — Adaptive over time / iterations based on the behavior of algo.

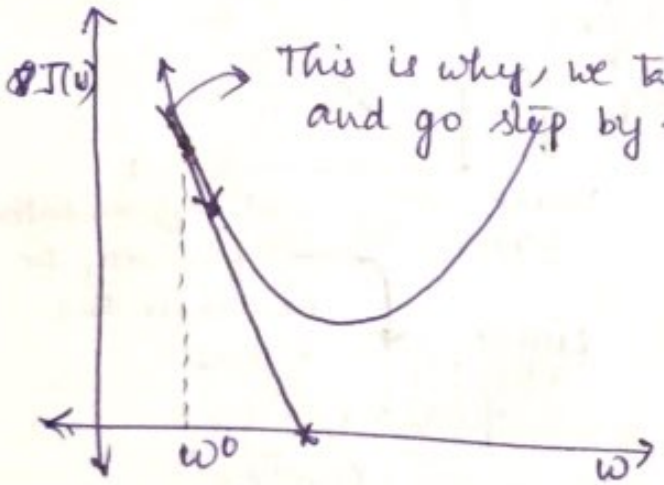
Learning Rate

$w^{n+1} \leftarrow w^n - \eta \nabla J(w^n)$

→ Given is data (x_i, y_i) not the loss function

when we make a move into the negative gradient

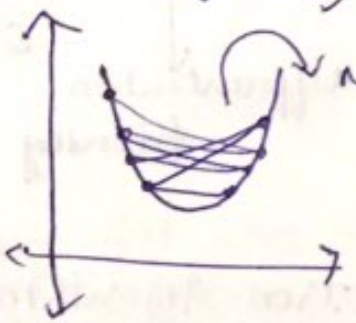
— Assume the function to be a line at the initial stage. and compute the slope.



This is why, we take η and go step by step. ← (if function nature is known, it would have been easy).

⇒ start with a very small η

Reason: (with large η, you take huge leaps (to and fro into positive and negative gradient regions))



⇒ η increased, if direction of w in the prev step and present step are same.

⇒ and if direction changes between steps, η decreased

For Ex:

$\eta \leftarrow 1.5 \times \eta$, if prev dir = present dir

$\eta \leftarrow 0.5 \times \eta$, if direct change.

Taylor series expansion ≡

$x = \omega^{n+1}$
$a = \omega^n$

$$f(x) = f(a) + f'(a).(x-a) + f''(a)(x-a)^2 + \dots$$

$$f(\omega^{u+1}) = f(\omega^u) + [\omega^{u+1} - \omega^u] * f'(\omega^u)$$

negative quantity.

$$\omega^{u+1} \leftarrow \omega^u - \eta \Delta J(\omega^u)$$

⇒ GD for Classification:

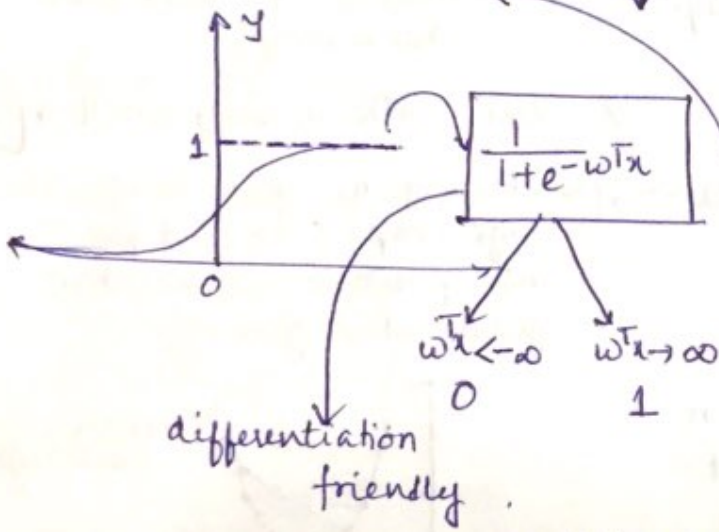
New function value = Old function value − (Positive quantity).
(less than the old one)

» Logistic Regression ≡

» ~~Perceptron Algorithm~~ ≡

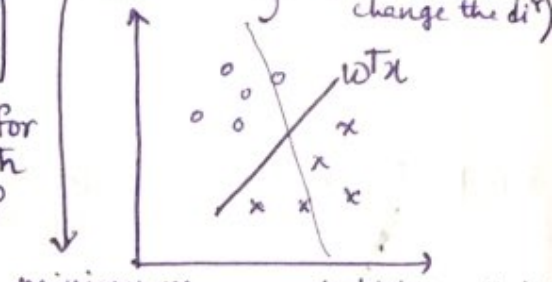
But instead;

[instead of step] → search for smooth step



differentiation friendly.

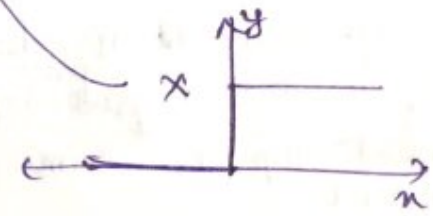
GD approach ≡ ω^0 (take a line) and acch to change the dir



minimum error. ≡ not differentiation friendly to eliminate the errors

Linear classifier.

$f(x) = 1, \omega^T x > 0$
$0, \omega^T x \leq 0.$



(created a new objective func^n)

» Perceptron Algorithm ≡

» $J = \sum \omega^T x$

$x \in$ Misclassified samples

↑ Minimize the no. of such (error) classifications.

Sum over misclassified samples. (instead of over all the samples)

when, Misclassified samples = $\{\phi\}$ ↷ we found the "line".

Logistic Regression ≡ step → smooth step

Perceptron Algorithm ≡ Creating a new objective function

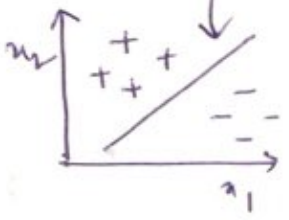
↓

every sample that gets misclassified is added to it

$w^{n+1} \leftarrow w^n + x$?

If the samples are "linearly seperable", ∃w which (can seperate +ve and -ve examples) → perceptron will find that W

Classification ≡ Model of classifier is ≡ Line. [Given this model, we cannot generate an apple]

① 

Boundary line between class A and class B. ← we model the line in the feature space

→ Discriminative classifier/model

② Generative Model (we don't model the boundary ↓ instead model the data) ≡ reduced representation [as in "a point"]

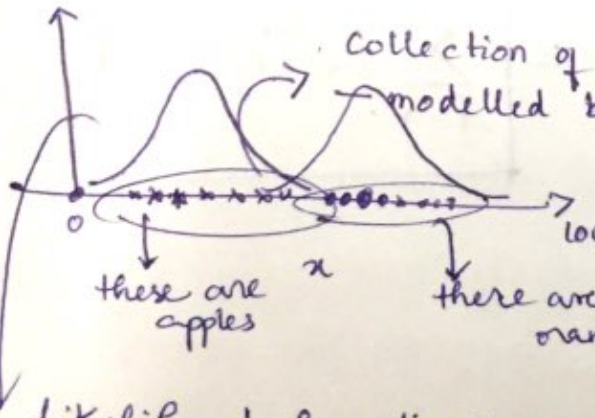
[Ex: model oranges and apples independently.]

↓

For Ex: Given a fruit, how apple-like or orange-like is it?

↓

Model the Classes?



[If I see 10.5, how likely is to classify it as apple]

Likelihood functions = Independently Learning about apples/oranges.

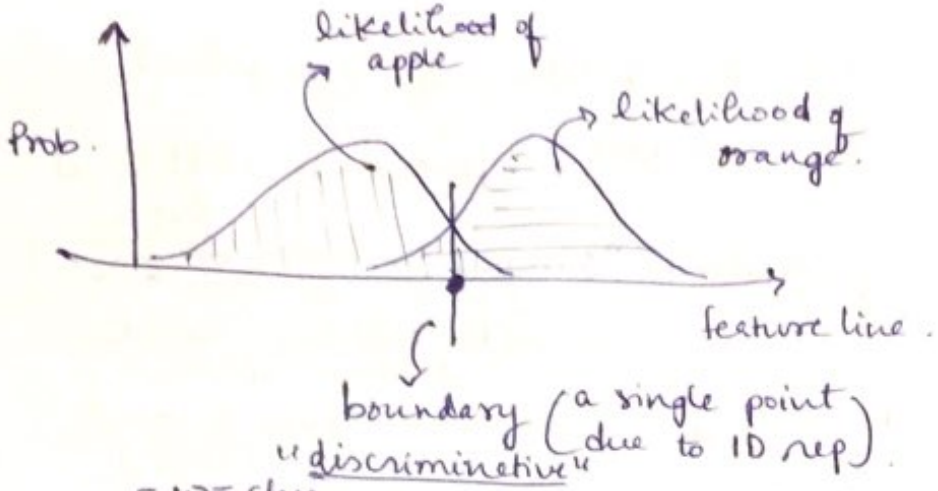
⇒ given density function → we can generate new points [that is why ≡ generative model]

Discriminative classifier ≡ we cannot generate samples as it can lie anywhere over the apple side of the line.

↓ Not rich enough to capture the details

↓ cannot explicitly model apples.

[can only say that anything that lies on one side is apple]



"discriminative"

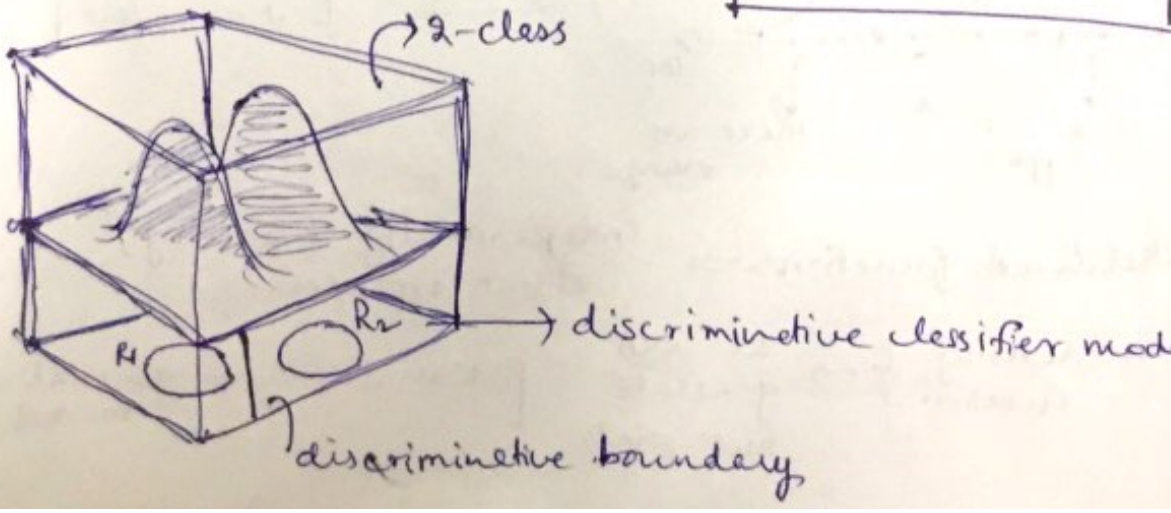
$P(\omega_1/x)$ ($\omega \equiv$ class) ≡ "Probability" that given feature vector, x belongs to the ω_1 class. [$0 < P < 1$]

↓ feature vector

$P(\omega_1/x)$, $P(\omega_2/x)$ ↷ can be compared to determine the class.

[in the case of 2-class problems ≡ $P(\omega_1/x) + P(\omega_2/x) = 1$].

Whereas, if there are c classes ≡ $\sum_{i=1}^{c} P(\omega_i/x) = 1$



P [Probability] ; $P(\omega_1/x)$, $P(\omega_2/x)$.

p [pdf]

$P(\omega_1, x) = P(\omega_1/x) \cdot P(x)$ (or)

↑ Joint Probability

$P(x/\omega_1) \cdot P(\omega_1)$.

$P(A,B) = P(A/B) \cdot P(B)$

(independent events) $= P(A) \cdot P(B)$

$$\therefore P(\omega_1/x) = \frac{p(x/\omega_1) \cdot P(\omega_1)}{p(x)}$$

Posterior prob. → Bayes Theorem.

$\Rightarrow P(\omega_1) \equiv$ Prob. of a class irrespective of given sample.

↳ Prior probability / belief

$\Rightarrow p(x/\omega_i) \Longrightarrow$ likelihood.

$\Rightarrow p(x) \longrightarrow$ how likely is that we'll observe x. "evidence".

Posterior prob $\equiv$ Post observing x, what is the probability $P(\omega_1/x)$

Ex: Disease and Test

$P(D) = 0.001$

$P(+/D) = 0.99$ ↷ If you have the disease, tests will say +, 99% (Accuracy of test)

$$P(D/+) = \frac{0.99 \times 0.001}{P(+)}$$

Prob of having disease, given test came out +ve

Evidence = likelihood that the test come out +ve (irrespective of disease or not) on random person

$P(+) = P(+/D) \cdot P(D) + P(+/ND) \cdot P(ND)$

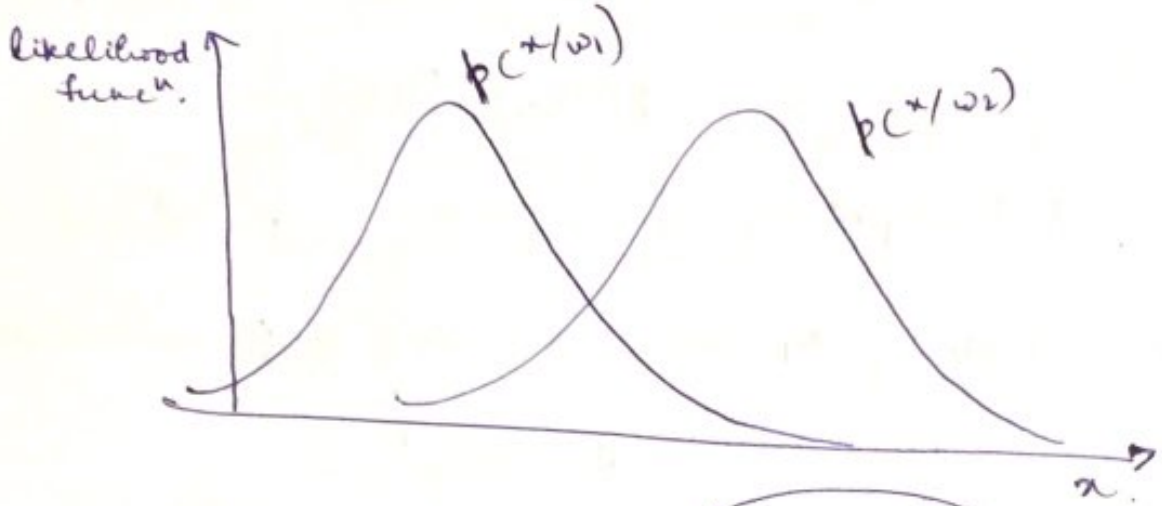
(person with D gets +ve result; person with ND)

$= 0.99 \times 0.001 + 0.01 \times 0.999$

$= 0.01 (0.099 + 0.999) = 1.098 \times 0.01$

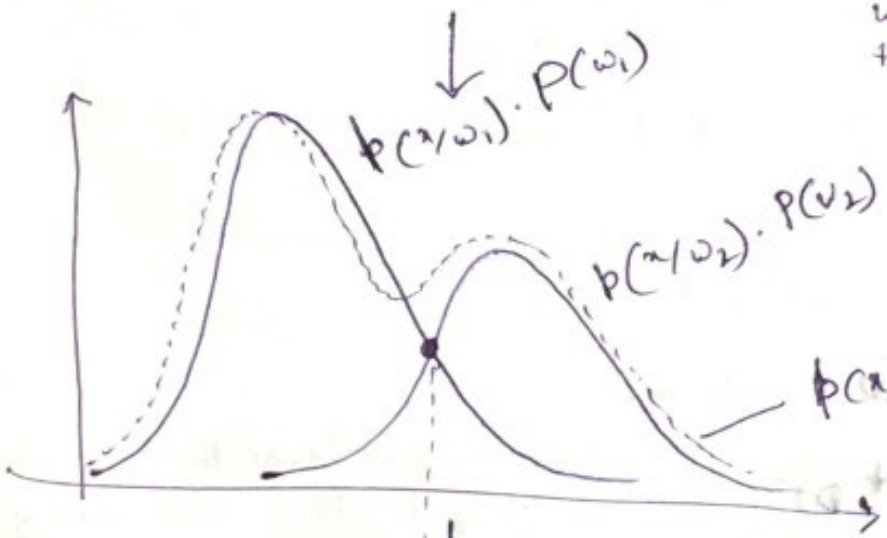
$$P(D/+) = \frac{0.99 \times 0.001}{0.01098} = 0.09\ (9\%).$$

↑ chance of having the disease.

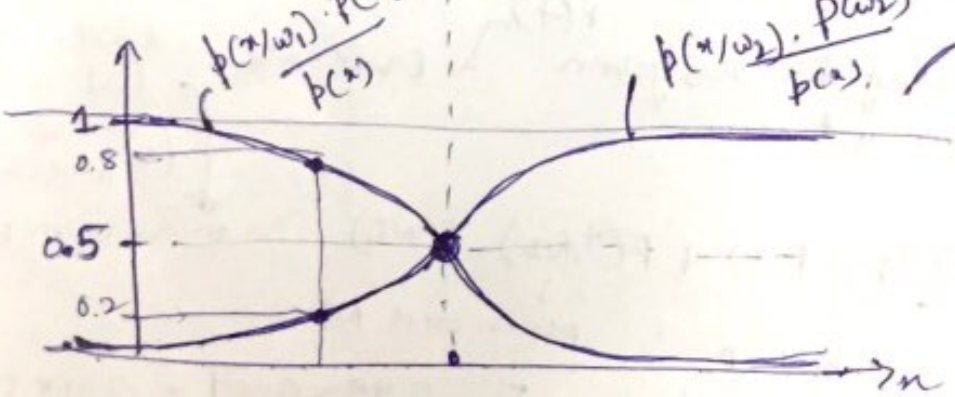


$$P(\omega_1/x) = \frac{p(x/\omega_1) \cdot P(\omega_1)}{p(x)}$$ — constant

↘ one of the curves will be larger than the other.



Normalize ≡ ↓ divide by $p(x)$



add upto 1 always for any x ∈

Bayes Theorem ≡

$$P(\omega_j/x) = \frac{p(x/\omega_j).P(\omega_j)}{p(x)}$$

- $P(\omega_j/x)$ ← after observing x, state of nature ≡ ω_j
- probability that state of nature, ω_j is true.
- Pick ω_j that has maximum prob.

Are all errors equally costly? No

so we define a loss function ≡ $\lambda(\alpha_i/\omega_j)$

- α_i ← action, α_i
- ω_j ← state of nature ≡ class
- → there can be many actions

Loss is not symmetrical,
E_1 if class A is misclassified as class B or
E_2 if class B is misclassified as class A
$E_1 \neq E_2$

Risk, $R(\alpha_i/x)$ → observe x, risk of taking α_i action

(of taking action α_i) $R(\alpha_i/x) = \sum_{j=1}^{c} \lambda(\alpha_i/\omega_j).P(\omega_j/x)$

- $P(\omega_j/x)$ → Prob that ω_j is true
- $\lambda(\alpha_i/\omega_j)$ → loss incurred because of taking α_i ~~to~~ in ω_j state of nature
- c ↓ c classes.

Strategy to pick the best action = based on action plan

Evaluate the action plan?

$\alpha(x)$ — if I see this observation, I'll do this.

$R(\alpha(x)/x).p(x)$

- $p(x)$ ← prob of observing x
- $R(\alpha(x)/x)$ ↑ Risk of following action plan, $\alpha(x)$ on observing x.
- given an x, what action needs to be taken.

"Action plans can be eval. based on risks"

— Integrate over all possible x,

Overall Risk ≡ $R_{\alpha(x)} = \int_x R(\alpha(x)/x).p(x)\,dx$

To Minimise overall risk, by minimize $R(\alpha(x)/x)$ for x.

↑ choosing appropriate / best "$\alpha(x)$"

From Bayes —

$$R(\alpha_i/x) = \sum_{j=1}^{c} \lambda(\alpha_i/\omega_j) \cdot P(\omega_j/x)$$

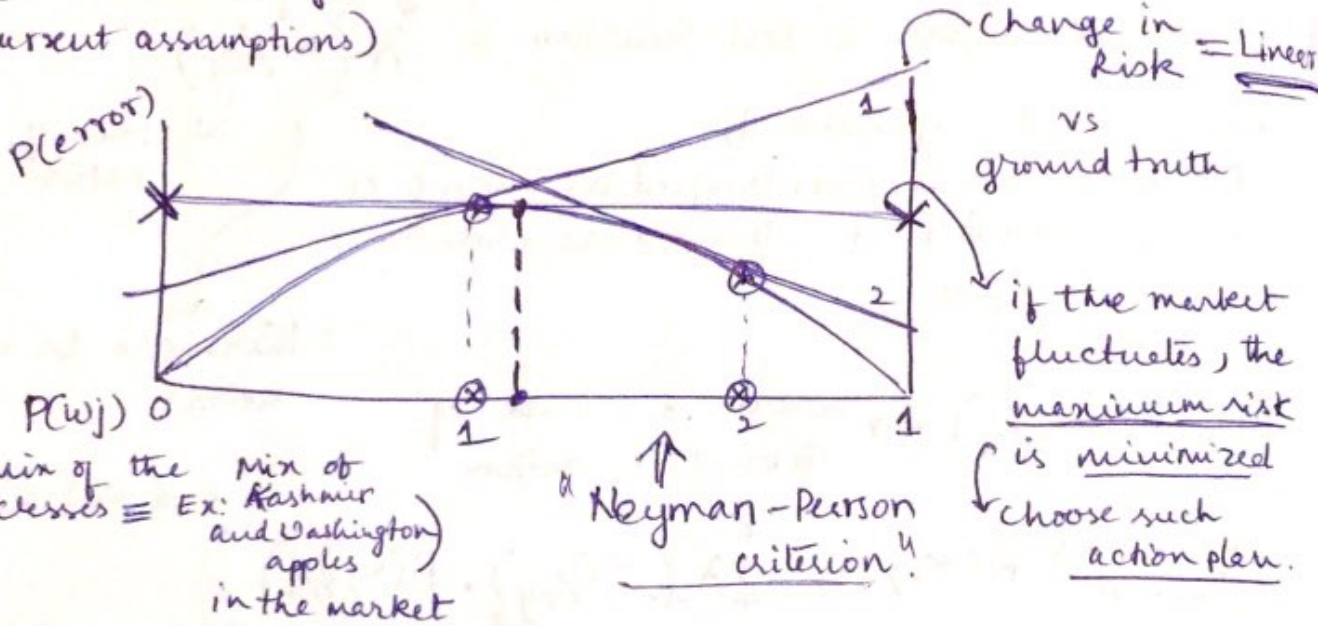
R^* = Bayes Risk.

↖ Bayes Action Plan.

$$R(\alpha(x)/x) = \underset{\alpha_i}{\text{argmin}} \left(R(\alpha_i/x)\right)$$

Risk $\propto$ P(error).

* $P(\omega_j)$ → can change (current assumptions)



$$P(\omega_i/x) = \frac{p(x/\omega_j) \cdot P(\omega_j)}{p(x)}$$

$p(x)$ → need not be computed to decide / compare [independent of ω_j]

$$g_i(x) = p(x/\omega_i) \cdot P(\omega_i)$$

(this is different g) $g_i(x) = \ln p(x/\omega_i) + \ln P(\omega_i)$

To classify, you need to compute $P(\omega_i/x)$, you can either compute $g_i(x)$ or $g_i(x)$ (latter one)

equation of Normal density $\Rightarrow p(x) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} e^{-\frac{1}{2}(x-\mu_i)^t \Sigma_i^{-1}(x-\mu_i)} = p(x/\omega_i)$

(can be ignored) constant

$g_i(x) = -\frac{d}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma_i| - \frac{1}{2}(x-\mu_i)^t \Sigma_i^{-1}(x-\mu_i) + \ln P(\omega_i)$

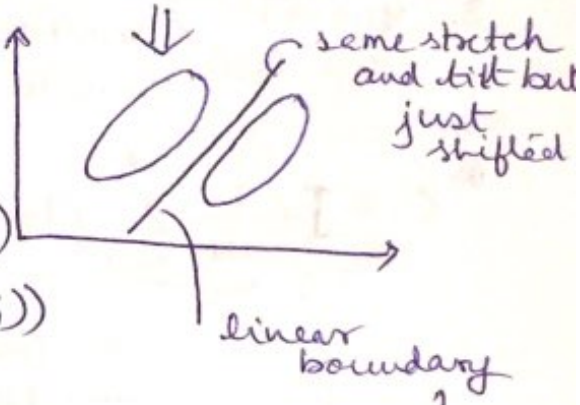
$g_1(x), g_2(x)$ can be computed for the normal densities

* $g_i(x) = -\frac{1}{2}\ln|\Sigma_i| - \frac{1}{2}(x-\mu_i)^t \Sigma_i^{-1}(x-\mu_i) + \ln P(\omega_i)$

Case 2: if $\Sigma_i = \Sigma$ (both the classes)) if Kashmiraud Washington, variance may be same, same covariance matrix

$g_i(x) = -\frac{1}{2}(x-\mu_i)^t \Sigma_i^{-1}(x-\mu_i) + \ln P(\omega_i)$

$= -\frac{1}{2}(x^t \Sigma^{-1} x - 2\Sigma^{-1}\mu_i^t x + \mu_i^t \Sigma^{-1}\mu_i) + \ln(P(\omega_i))$



independent of i, class

$g_i(x) = \Sigma^{-1}\mu_i^t x - \frac{1}{2}\mu_i^t \Sigma^{-1}\mu_i + \ln(P(\omega_i))$ — tends to be "linear"

Discriminant function.

$= w_{i1}^T x + w_{i0}$ } linear function.

In 2-D, decision boundary, would be a linear function (a line)

In 3-D, it'll be a plane

* $\Sigma_i \neq \Sigma$

$\Rightarrow$ if covariance matrix is not the same, then the decision boundary will no more be a linear function but a complex function. (it coulde be an ellipse, hyperbole ...)

Case 3: $\Sigma_i = \Sigma$

$g_i(x) = -\frac{1}{2}(x^t \Sigma^{-1} x - 2\Sigma^{-1}\mu_i^t x + \mu_i^t \Sigma^{-1}\mu_i) + \ln P(\omega_i)$

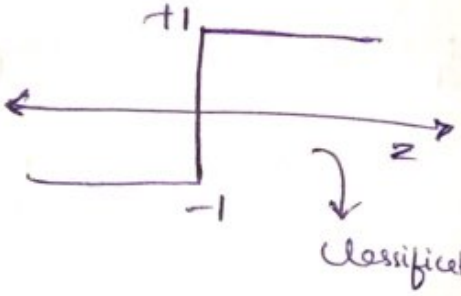
Loss :

In Regression – $R \equiv J = \sum_{i=1}^{N} (y_i - w^T x_i)^2$

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

In Classification – $C \equiv$ 0/1 misclassifications.

Ex: $y_i * g(w^T x)$.

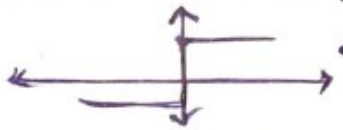


1. $y_i = +1$, $w^T x = 10$. ✓ correct $= 1 = y_i (g(w^T x))$
2. $y_i = -1$, $w^T x = -10$. ✓ correct $= 1 = y_i (g(w^T x))$
3. $y_i = -1$, $w^T x = 10$ × incorrect (misclassified)

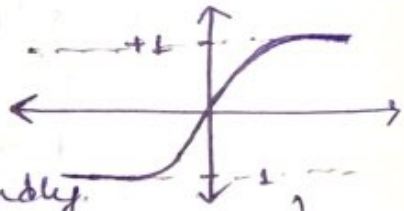
* $C \equiv J = \frac{1}{2N} \sum_{i=1}^{N} (1 - y_i \cdot g(w^T x_i))$

— Good
— Not optimisation friendly.

for this, instead of using

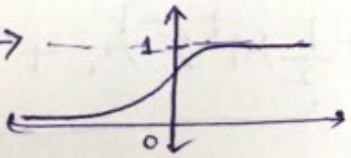


its better to use



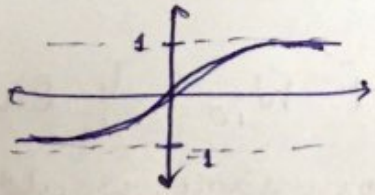
this is (Not Logistic Regression)

optimize it by changing the [g] →



[Logistic Regression]

(or)



Gradient Descent:

1. start with w^0, $n = 0$
2. $w^{n+1} \leftarrow w^n - \eta \nabla J$
   - η → changes, when loss function changes
   - varies over time (iteration) [usually constant]
3. $n \leftarrow n+1$
4. Repeat 2–4 until some convergence criteria is met.

?

— change in w is very very small.
— change in loss is very small.

→ Loss function is not linear but the discriminant func^n is linear.

$$f(y) = f(x) + (y-x)f'(x) + \frac{(y-x)^2}{2} f''(x)$$

operator

approximated $\equiv f(Y) = f(X) + (y-x)^T \nabla f(x) + \frac{1}{2}(y-x)^T H (y-x) \equiv$ H: Hessian matrix, d×d matrix

$$f(w^{n+1}) = f(w^n) + [w^{n+1} - w^n]^T \nabla f(w^n) + \frac{1}{2}(w^{n+1} - w^n)^T H [w^{n+1} - w^n]$$

$$w^{n+1} \leftarrow w^n - \eta \nabla f$$

$$w^{n+1} - w^n = -\eta \nabla f$$

$$f(w^{n+1}) = f(w^n) + (w^{n+1} - w^n)^T \nabla f$$

$\nabla f^T \cdot \nabla f$

only proves that

$$f(w^{n+1}) = f(w^n) - \eta \,(+\text{ve quantity})$$

$\Rightarrow$ GD reduces the loss in each iteration if η is +ve and small

but can we go faster. ?

$$f(w^{n+1}) = f(w^n) - \eta \|\nabla f\|^2 + \frac{\eta^2}{2} \nabla f^T H (\nabla f)$$

Since we want minimum, diff wrt η. $f(w^{n+1})$

$$\frac{\partial}{\partial \eta} f(w^{n+1}) = 0 - \|\nabla f\|^2 + \frac{2\eta}{2} \cdot \nabla f^T H (\nabla f) = 0$$

$$\eta \cdot \nabla f^T \cdot H (\nabla f) = \|\nabla f\|^2$$

$$\eta = \frac{\nabla f \cdot \nabla f^T}{\nabla f^T \cdot H (\nabla f)} = \frac{\nabla f^T \nabla f}{\nabla f^T H \nabla f}$$

$w^{n+1} - w^n \equiv s$

Better update rule than this : $w^{n+1} \leftarrow w^n - \eta \nabla f$?

$$f(w^{n+1}) = f(w^n) + s^T \nabla f + \frac{1}{2} s^T H (s)$$

$$= f(w^n) + s^T \nabla f + \frac{1}{2} s^T H s$$

H – matrix
s – vector

$$\frac{\partial}{\partial s} = 0 \Rightarrow \nabla f + Hs = 0$$

$$s = -H^{-1} \nabla f$$

Better update rule, – Newton iteration (Descent) * (but not used)

↳ Not used → maybe H might be singular. ✓
(inverse doesn't exist)

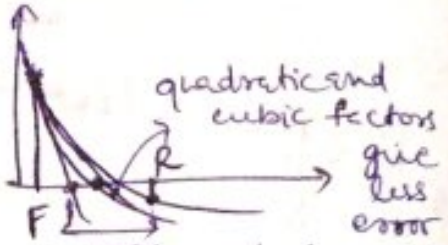
if |H| is very less → inverse can be very sensitive

↓

Not computationally attractive

First of all, how is this better than the first one? GD

→ In the first case, approximating to the first order [GD] gives a smaller value (higher error)



but computational load ↑

For now, constant η. ✓ but we have to handle the loss, J

Perceptron ≡ Sum the loss only over the misclassified samples.

↳ optimizing wrong objective func^n.

objective function.

$$\left[\sum_{x_i \in \Sigma} -y_i \, w^T x_i\right]$$

↳ misclassified.

$w^T x \geq 0, \; y_i = +1$

$w^T x < 0, \; y_i = -1$

Total Error (Amount of misclassification)

Not number, because we are taking $w^T x$ instead of $g(w^T x)$.

↓

update rule changes to ≡ $w^{n+1} \leftarrow w^n + \eta \sum_{x_i \in \Sigma} y_i x_i$

Changes with every step.

↓ so

Not computationally different ← alternative ≡

$$w^{n+1} \leftarrow w^n + \eta \sum_{i=1}^{N} (t_i - o_i) x_i$$

if target = 1, output = 1

$t - o = 0$ (correctly classified)

target output

doesn't need to ~~go~~ misclassified set

go over

Convergence criteria ⇒ Until no class is misclassified

How does this work? (Intuitive Approach)

$w^T x \geq 0$, +1
$w^T x < 0$, −1

① Assume, $t_i = +1$, $o_i = -1$ and $x_i = +ve$.
(Error is present.)

initially $w < 0$ as $x > 0$ & $w^T x < 0$

↓

With these values, w's going to increase. ⇒ increase w

↳ w becomes +ve.

② $t_i = +1$, $o_i = -1$ and $x_i = -ve$

$wTx < 0$

$x < 0$ $w > 0$

$(t - o)x \Rightarrow -ve \rightarrow$ w decreases.

↳ w becomes −ve

11ly for diff values of t & o.

→ If a sample is misclassified, add/subtract it. (based on values of t and o)