

$$J_R = J + \lambda \|w\|^2 + 1000w_3^2 + 16w_4^2$$

↓
small.

(to remove a lot of dimensions/features)

— computationally good at the test time.

$w \rightarrow$ sparse

(lot of elements = 0,
express 3 features)

Find w as sparse as possible.

Can use L_1 norm.

$\checkmark L_2$ norm (easier) \Rightarrow this to make it small

$$\underline{(\|w\|^2)}$$

$\Rightarrow (x_i, y_i)_{i=1, \dots, N}$ (Data)

$x_i \in \mathbb{R}^d$, $\boxed{\text{Nord (large)}}?$ (mostly N).

What if d is large \rightarrow start with a large X (d measurements) (but not optimal).

* Feature Selection \equiv subset selection

(given d features, find a subset of those features reducing the cost). \rightarrow Backtracking

— No efficient solution. (polynomial)

$\overset{\text{not used}}{\equiv}$ because we work with large set of dimensions.

Dimensionality
↑ Reduction.

* Feature Extraction \equiv

$z = Ux$ \rightarrow initial features $\equiv d$

$k \times 1$ $\underbrace{k \times d}_{d \times 1}$

new K features

linear combination
of d features

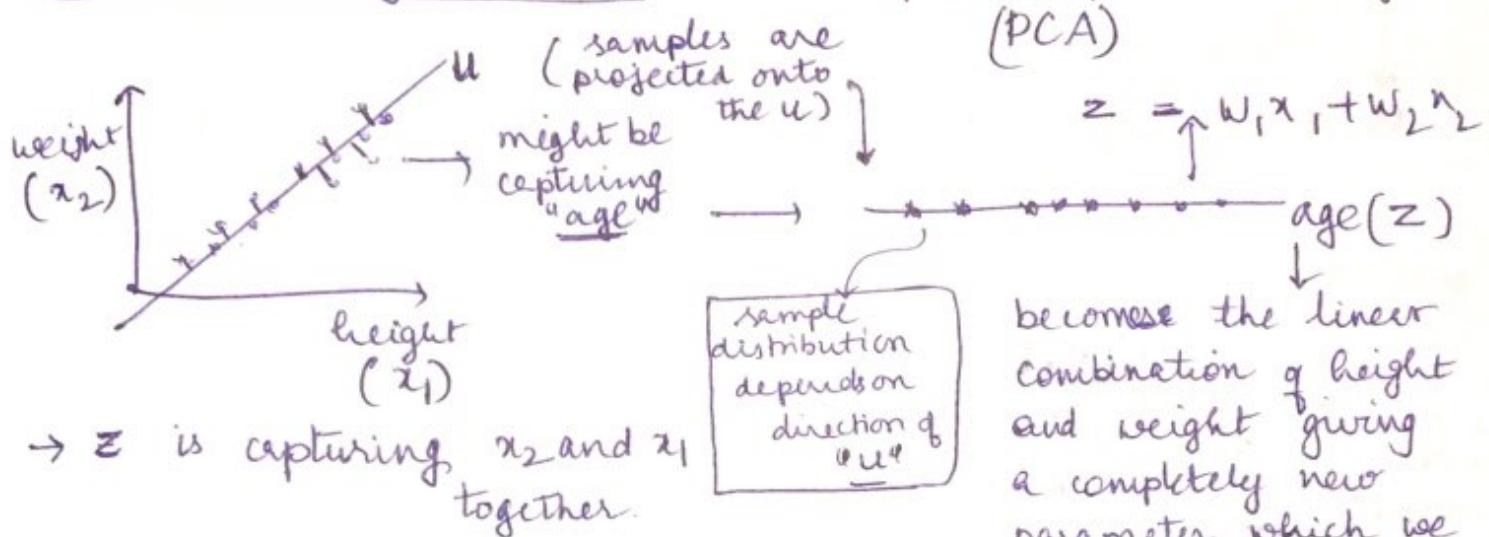
$[k < d]$

doesn't reduce
the cost
[because we
utilize the
combination of
all the features]

\hookrightarrow may be useful for low computational devices

Feature extraction - lower computational cost
+
lower dimensional representation.

Dimensionality Reduction = Principal Component Analysis



" u = direction where maximum variance is preserved."

New mean of the data (projections of the samples)

$$\bar{z} = \frac{1}{N} \sum_i z_i = \frac{1}{N} \sum_i u^T x_i$$

↓ independent of i .

$$= u^T \left(\frac{1}{N} \sum_i x_i \right)$$

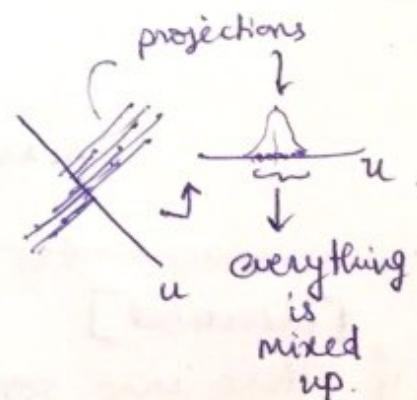
$$= u^T \mu$$

If the line is this way, so our objective is to maximize the variance;

$$\text{Max } \frac{1}{N} \sum_{i=1}^N (z_i - \bar{z})^2$$

$$\frac{1}{N} \sum_{i=1}^N (u^T x_i - u^T \mu)^2$$

$$\frac{1}{N} \sum_{i=1}^N (u^T x_i - u^T \mu)(x_i^T u - \mu^T u)$$



$$= \frac{1}{N} \sum_{i=1}^N u^T [x_i - \mu] [x_i - \mu]^T u$$

$$= u^T \underbrace{\sum_{i=1}^N u}_{\text{covariance matrix}} \text{, such that } u^T u = 1$$

↓
covariance matrix = $d \times d$ matrix

(Solution = an eigen vector)

u is the eigen vector of Σ , corresponding to the largest eigen value, λ .

eigen vectors - orthogonal

eigen vector matrix!

$U = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_K \end{bmatrix}$ eigen
largest value, vector = u_1

such that when projected onto this, variance is maximized.

K such directions onto which samples can be projected.

Eigen Face =

↓ concatenated
{learning a feature}

$\begin{array}{c} \square \quad \square \\ \parallel \quad \parallel \end{array} \dots \quad N = 1000$ → $\begin{array}{c} \square \quad \square \\ \parallel \quad \parallel \end{array} \dots \quad 100 \equiv 10^4$ features
N faces of people.

$$\sum_{i=1}^{1000} (x_i - \mu)(x_i - \mu)^T \equiv \begin{array}{c} \square \quad \square \\ \parallel \quad \parallel \end{array} \dots \quad 100 \text{ eigen vectors (Non zero)}$$

$$\mu = \frac{1}{N} \sum x_i$$

rearrange back as a rectangle.

\square (also looks like a face but blurred)

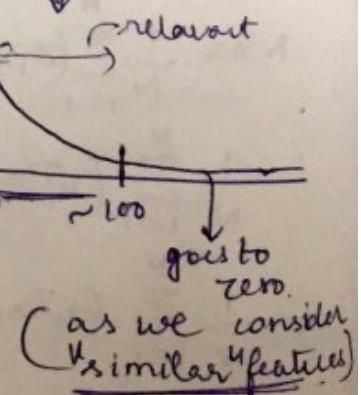
[Identical]

If there were correct assignment (height & weight)
→ we would have got single α_1 and then $\alpha_2 \dots = 0$.

but not the case here due to redundant, multiple features.

$$z = \begin{bmatrix} x^T u_1 \\ x^T u_2 \\ \vdots \\ x^T u_{100} \end{bmatrix} \alpha_1 \alpha_2 \dots \alpha_{100}$$

these features actually represent you.



These 100 scalars — enough to reconstruct any face.

$$X \begin{bmatrix} \text{new} \end{bmatrix} = \sum_{i=1}^{100} \alpha_i u_i \quad \begin{array}{l} \xrightarrow{\text{global basis for faces. (we have 100 us)}} \\ \left[\text{common for everybody} \right] \end{array}$$

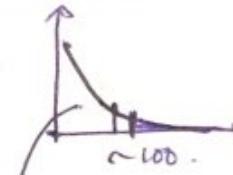
signature of your face.

$$X = \sum_{i=1}^{10000} \alpha_i u_i = \alpha_1 u_1 + \alpha_2 u_2 + \dots + \alpha_{100} u_{100} + \dots + \alpha_{10000} u_{10000}$$

changes to

$$= \sum_{i=1}^{100} \alpha_i u_i \rightarrow \text{all you need is these 100 values.}$$

can be discarded because \equiv



~ compression (within minimal loss)

(basis = eigenvector).

KLT (eigenvector basis).

this is data dependent (if it is faces, then one set of u_i 's)

is
one set of u_i 's
else some other set of u_i 's.

→ domain is narrow,

then we can learn the basis.

For Eu: person specific.

decay effect
 \Rightarrow "only for similar features not for scattered parameters".

Other Defⁿ → Find u such that we get minimum reconstruction error.

in this case, $N(10^3) < d(10^4)$, $\sum_{d \times d}$.

$$X_{N \times d}, \quad X X^T = N \times N \quad \boxed{X X^T u = \lambda u} \quad \text{--- (1)}$$

$$X^T X = d \times d \quad \boxed{X^T X v = \lambda v} \quad \text{--- (2)}$$

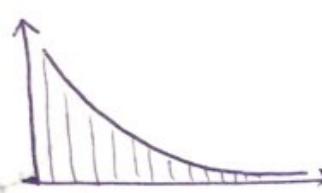
Can we compute eigen vector of one from the other?

$$x^T x \underset{u}{\underset{\downarrow}{\sim}} (x^T u) = \lambda \underset{v}{\underset{\downarrow}{\sim}} (x^T u)$$

$$x^T x v = \lambda v$$

$\Rightarrow v$ can be computed from u
($d \times d$)

while considering eigen values;



$$\rightarrow \frac{\sum_{i=1}^K \lambda_i}{\sum_{i=1}^d \lambda_i}$$

$$N \times N \rightarrow u \in 1000$$

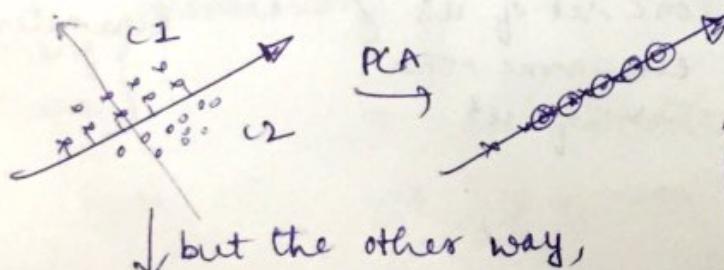
$$\left\{ \begin{array}{l} \downarrow \\ x^T u \\ \downarrow \\ v = x^T u \end{array} \right.$$

Computationally attractive.
better than $[d \times d]$

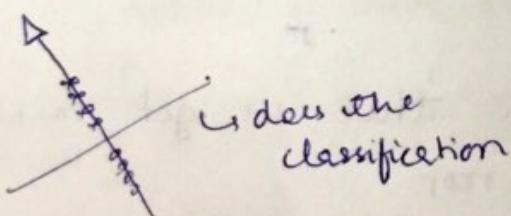
$> 95\%$ if it is 100%
(we are picking all the values)

can have nonlinear schemes.

If there are 2 classes;



but the other way,



uses the classification

"supervised" — uses class information

LDA / Fisher.

computed and derived
from data

PCA

"unsupervised"

class information is
not taken into
account.

works for compression

PCA fails miserably
in the case of
"classification" problems

use "supervised"

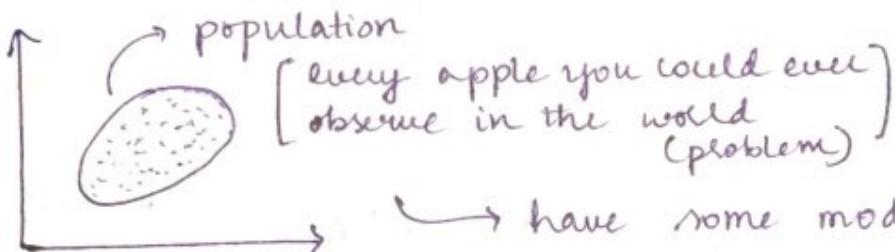
$$z = \underline{w}^T x \text{ (linear)}$$

↓

can be replaced by nonlinear (Non-Linear Dimensionality Reduction)

Parameter Estimation

→ after learning how apples and oranges look like, how probable that my understanding is correct?



→ have some model in mind

Ex: distribution of apples is an ellipse/ fitting line.

Model: Line.

↳ Goal - estimate w

* Params:
 $\langle w_0, w_1, w_2 \rangle^T$

Population → not observable

(you hardly have the full population in any problem, if you have, then the classification becomes easy).

Randomly pick up some samples [if I want apples from gachibowli, go to all shops in gachibowli]

* Any sample has same probability of getting picked

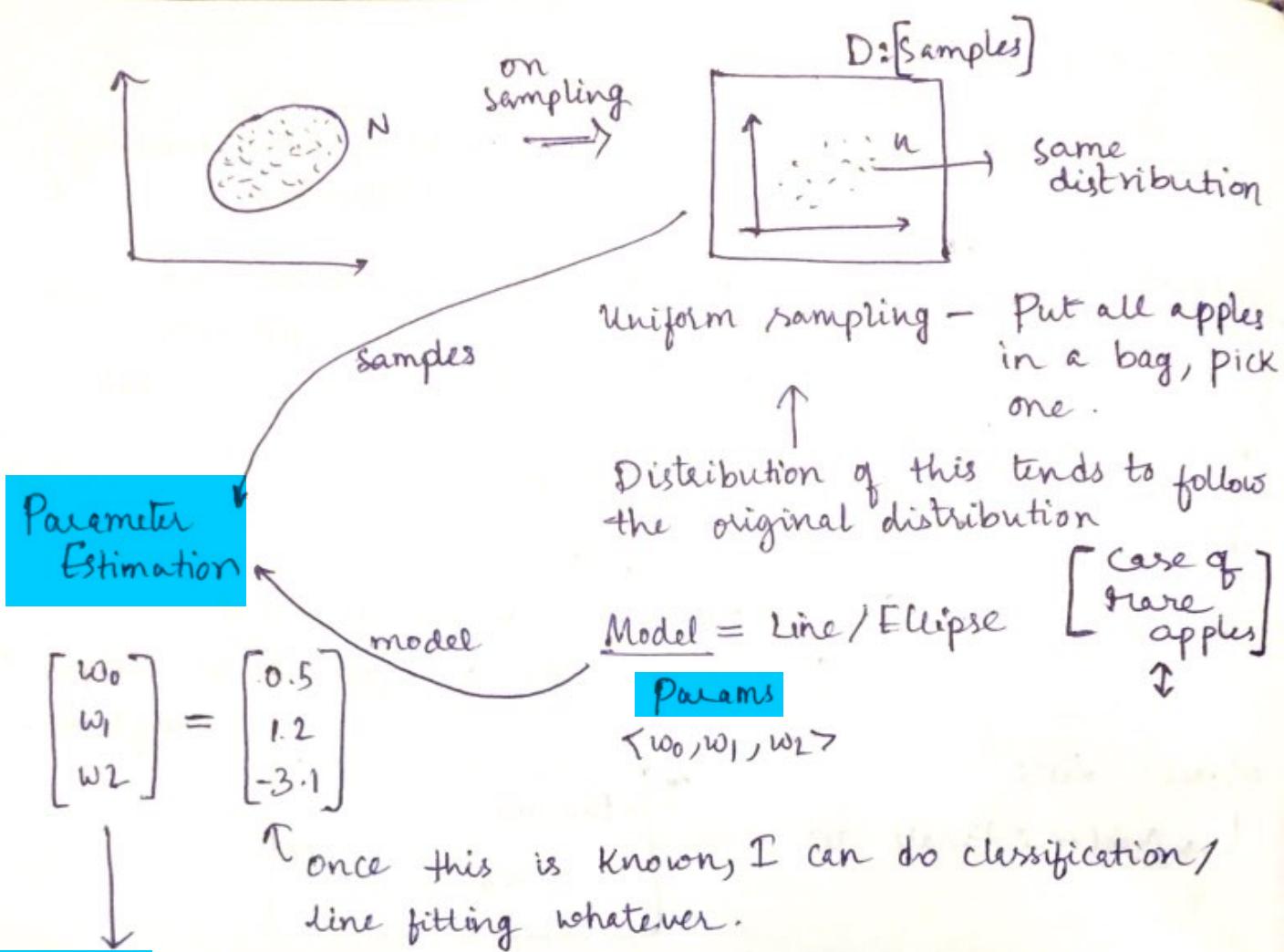
(pick uniformly)

⇓ "Uniform Sampling".

* Not distribution of samples is uniform but "sampling process" is uniform

→ Out of a bag of n^4 apples

uniform sampling = probability of picking an object will always be same - $1/n$



once this is known, I can do classification / line fitting whatever.

goal: Finding out the values of w_0, w_1, w_2 of the population, but we can observe only samples.

Two approaches:

1. Compute $\hat{\theta}$ that best explains D (Samples - whatever we observe)

(vector) parameters to be estimated

population \mapsto parameters corresponds to samples

$\left[\begin{smallmatrix} \text{frequentist} \\ \text{approach} \end{smallmatrix} \right]$ MLE

set of training samples.

Called as maximum likelihood estimate (MLE)

$$\hat{\theta} = \arg \max_{\theta} P(D|\theta)$$

[bayesian approach] find that θ that maximizes $P(\theta|D)$

2. Use bayes theorem to compute $\hat{\theta}$, ("case of prior belief")

$$P(\theta|D) = \frac{P(D|\theta) \cdot P(\theta)}{P(D)}$$

would also be
a density
function

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} P(D|\theta) \cdot P(\theta)$$

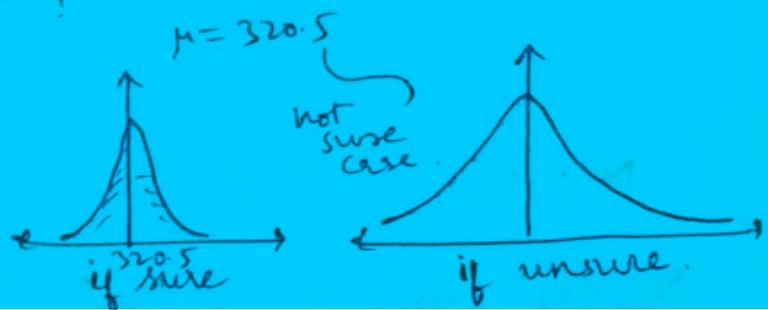
(density)
(density)

MAP
Maximum Apriori Estimate
prior belief is being combined with what you observe.

$\theta \Rightarrow \mu \equiv \text{mean of apples (if normal distribution)}$

$P(\theta) = ?$

In bayesian world,
everything is a
density function,
(nothing is a scalar)



MLE =

$$\underset{\theta}{\operatorname{argmax}} P(D|\theta) = \prod_{k=1}^n P(x_k|\theta) \quad \text{each sample.}$$

$$\text{Taking log} \Rightarrow \sum_{k=1}^n \log(P(x_k|\theta))$$

differentiate w.r.t θ ,

$$\nabla_{\theta} l = \sum_{k=1}^n \nabla_{\theta} \ln P(x_k|\theta) = 0 \rightarrow \text{zero vector.}$$

θ is not
a scalar, but a
vector

Model = Normal Distribution.

Case 1: $\mathcal{N}(\mu, \Sigma)$ \rightarrow can we estimate μ ?

Likelihood: $\ln P(x_k|\mu) = ?$

$$\Rightarrow \ln L(\dots) = -\frac{1}{2}(x_k - \mu)^T \Sigma^{-1} (x_k - \mu)$$

independent of θ

$$\begin{aligned} \nabla_{\mu} \ln(P(x_k|\mu)) &= 0 - \frac{1}{2} (-1) \Sigma^{-1} (x_k - \mu) \\ &= \Sigma^{-1} (x_k - \mu) \end{aligned}$$

Normal distrib:

$$e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)} =$$

$$\frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} =$$

$$(2\pi)^{-d/2} |\Sigma|^{-1/2} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}$$

$$\nabla_{\mu} \ln(\downarrow) = \Sigma^{-1}(x_k - \mu) \Rightarrow * \nabla_{\mu} \ln(P(x_k | \mu)) = \Sigma^{-1}(x_k - \mu)$$

$$\hat{\theta}_{MLE} \Rightarrow \sum_{k=1}^n (\Sigma^{-1}) (x_k - \mu) = 0.$$

covariance $\equiv \Sigma$ → gives an estimate
of the world mean

$$\hat{\mu}_{MLE} = \sum_{k=1}^n (\Sigma^{-1}) (x_k - \hat{\mu}) = 0.$$

$$\sum_{k=1}^n x_k - \sum_{k=1}^n \hat{\mu} = 0.$$

$$\hat{\mu}_{MLE} = \frac{1}{n} \sum_{k=1}^n x_k$$

Step 2:

$$N(\mu, \sigma^2) \quad N(\theta_1, \theta_2)$$

$$\nabla_{\theta_2} \ln(\mu_k / \theta_2) = \left[\begin{array}{c} \nabla_{\theta_1} \\ \nabla_{\theta_2} \end{array} \right] \rightarrow \text{same as before.}$$

$$-\frac{1}{2\theta_2} + \frac{(x_k - \theta_1)^2}{2\theta_2^2}$$

$$\sum_{k=1}^n -\frac{1}{2\theta_2} + \frac{(x_k - \theta_1)^2}{2\theta_2^2} = 0.$$

$$\left\{ \frac{1}{2\theta_2^2} \sum_{k=1}^n (x_k - \theta_1)^2 = \frac{n}{2\theta_2} \right.$$

multiply both sides by $(\frac{2\theta_2}{n})$

$$\hat{\theta}_{2MLE} = \frac{1}{n} \sum_{k=1}^n (x_k - \hat{\theta}_1)$$

$$= \frac{1}{n} \sum_{k=1}^n (x_k - \hat{\mu})^2$$

for normal densities
 \therefore mean of population = mean of samples *

As the sampling changes, $\hat{\mu}$ will change
 (take another 100 samples) (will move around)

What is the distribution of $\hat{\mu}$?

↓
 what will be μ of $\hat{\mu}$? Will it converge
 to the actual μ ?

Expected value of $\hat{\mu}$.

$$E(\hat{\mu}) = \mu \text{ (actual } \mu \text{ of popl"}^n \text{)} \quad \begin{array}{l} \text{no. of samples that u} \\ \text{take} \end{array} \quad \begin{array}{l} \text{when } n \text{ increases,} \\ \hat{\mu} \text{ changes very} \\ \text{little.} \end{array}$$

(average)

$$E(\hat{\sigma}^2) \neq \sigma^2 \quad (\text{variance is diff})$$

(average)

actually,

$$E(\hat{\sigma}^2) = \frac{n-1}{n} \cdot \sigma^2 \quad (\text{slightly smaller than } \sigma^2)$$

* MLE, $\hat{\sigma}^2$ is a biased estimate. (we have estimated a smaller one)

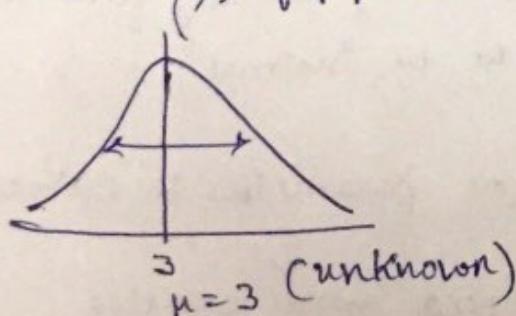
MLE, $\hat{\mu}$ is unbiased estimate.

$\xrightarrow{\text{u-MLE}}$

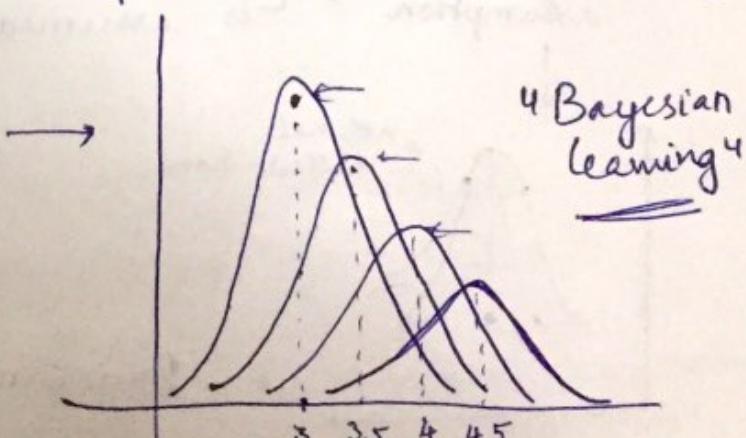
$$\hat{\sigma}^2 = \frac{1}{n-1} \sum_{k=1}^n (x_k - \hat{\mu})^2$$

→ increase slightly becomes.
 ↴ unbiased estimate.

MAP - $\xrightarrow{\text{M of popl"}^n = 3}$



prior belief = 4.5, → it moves towards 3

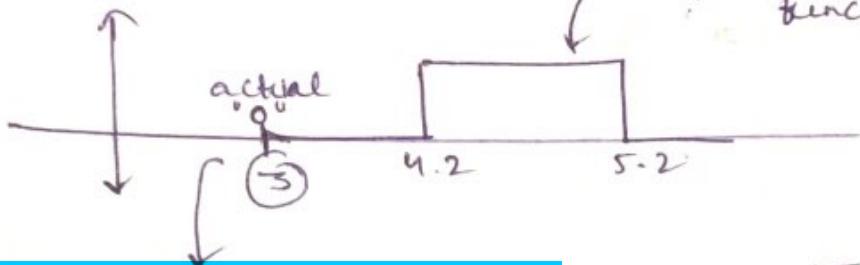


→ as you start observing more and more samples, μ goes toward [actual μ].

If prior belief is this way ↗

no matter how much data is observed, this would at max converge to 4.2

uniform density function



If your prior belief is 0, at a particular point

then posterior at that point is also 0 ↗

and you'll never reach 3 in this case.

- * If your prior belief is bad, you'll never learn.
 - Be slightly skeptical about your beliefs ↗
 - "Wise Man is never confident"

Bayesian Parameter Estimation ≡

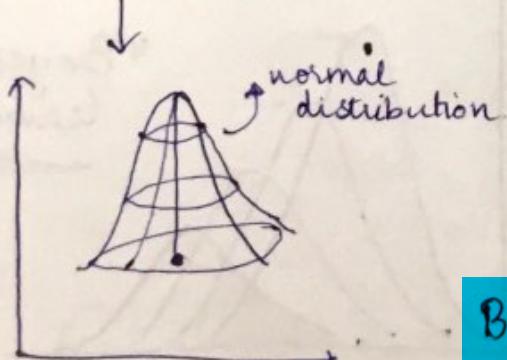
Recall:

$$p(w_i/x) = \frac{p(x/w_i)p(w_i)}{p(x)} \quad \begin{matrix} \rightarrow \text{observing } x, \text{ such that we're talking} \\ \text{about apples, } w_i \end{matrix}$$

generative model.

to compute $p(x/w_i)$ → we use 2 approaches where the

strong assumption ← [nature of probability density function is assumed to be "normal",



more parameters to estimate

↑
need more samples.

Bayesian decisions ≡ optimal

↓
based on the assumed nature of the function.

Bayes doesn't work well because our assumptions are or estimates are not always correct.

In the case of 4000 parameters,

$\mu_{[4000]}$, covariance matrix $[4000 \times 4000]$ → we will be need a minimum of 80 million samples to estimate.

MLE:

$$D = \{x_1, x_2, \dots, x_n\}$$

$$p(x) \sim N(\mu, \sigma^2)$$

$$p(\theta) = \underset{\theta}{\operatorname{argmax}} P(D|\theta)$$

$$= \underset{\theta}{\operatorname{argmax}} \prod_{k=1}^n p(x_k | \theta)$$

$$\begin{aligned} \hat{\mu}_{MLE} &= \frac{1}{n} \sum_{k=1}^n x_k \\ \hat{\sigma}_{MLE}^2 &= \frac{1}{n} \sum_{k=1}^n (x_k - \hat{\mu})^2 \end{aligned} \quad \left. \right\} MLE \text{ estimates of } \mu, \sigma^2$$

MAP:

$$\hat{\theta}_{MAP} = \underset{\theta}{\operatorname{argmax}} P(D|\theta) \cdot P(\theta)$$

$P(\theta|D)$ → θ -random variable [density function].
 $P(\theta)$ → could be a density function.

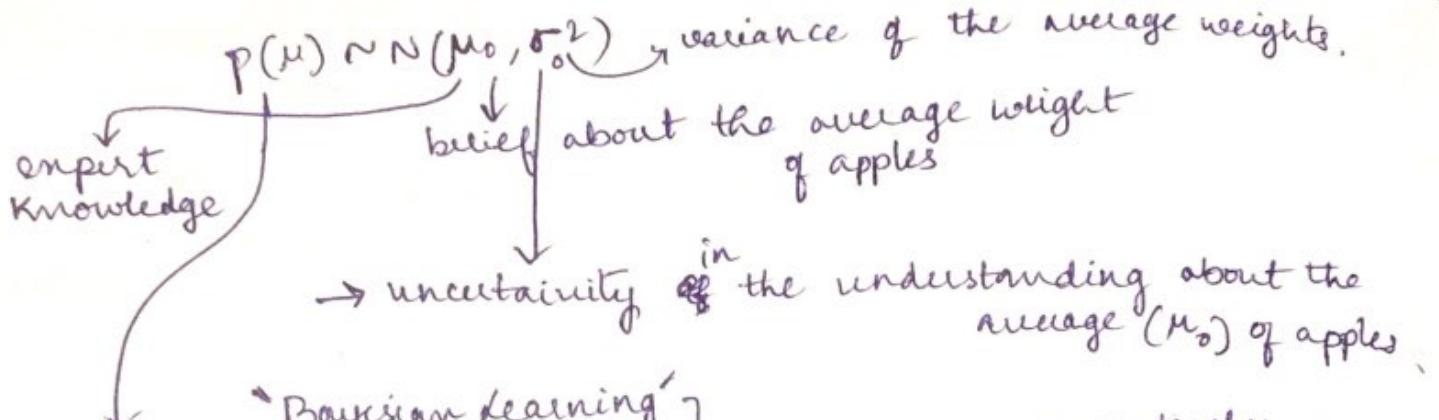
density functions of parameters → posterior → maximum [density function] of the probs.

$$P_\theta(x|w_i) \sim N(\mu, \sigma^2)$$

$$P(\mu) \sim N(\mu_0, \sigma_0^2) \quad [\text{Prior belief}]$$

$$P(\mu|D) = P(D|\mu) \cdot P(\mu).$$

x = weights of apples. → it should be estimated from samples "prior info".
Let $N(\mu, \sigma^2)$
 $\mu = 250 \text{ g}$ $\sigma = 50$ (absolute) - not known



On observing more and more apples, the Σ (actually, estimated μ) (samples) becomes lesser and lesser.

Case 1: $P(x/\mu) = N(\mu, \sigma^2)$, σ^2 is known

$$P(\mu) \sim N(\mu_0, \sigma_0^2) \quad [\text{prior understanding of } \mu]$$

$$P(\mu/D) = \frac{P(D/\mu) \cdot P(\mu)}{\int P(D/\mu) \cdot P(\mu) d\mu} \rightarrow \text{likelihood of observing data set,} \\ \sim \text{product of samples.}$$

$$\int P(D/\mu) \cdot P(\mu) d\mu \rightarrow \text{constant.}$$

$$= \propto \prod_{k=1}^n P(x_k/\mu) \cdot P(\mu)$$

both are normal
density fns.

Posterior
(Normal density).

Reproducing densities.

{Posterior \sim Prior \sim Normal}

shape remains the same

$$= \propto \prod_{k=1}^n \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2} \left(\frac{x_k - \mu}{\sigma} \right)^2} \cdot \frac{1}{\sqrt{2\pi}\sigma_0} e^{-\frac{1}{2} \left(\frac{\mu - \mu_0}{\sigma_0} \right)^2}$$

observations

prior.

$$P(\mu_D) = \frac{1}{\sqrt{2\pi}\sigma_n} e^{-\frac{1}{2} \left(\frac{\mu - \mu_n}{\sigma_n} \right)^2}$$

understanding of μ after observing samples.

$P(\mu_D) \Rightarrow (\mu_n, \sigma_n)$ = updated values

$$\mu_n = \frac{n\sigma_0^2}{n\sigma_0^2 + \sigma^2} \hat{\mu}_n + \frac{\sigma^2}{n\sigma_0^2 + \sigma^2} \mu_0$$

↓ ↓

MLE estimate prior
"observation"

$\frac{n\sigma_0^2}{n\sigma_0^2 + \sigma^2}$

What happens when n increases?

(more and more samples) $\rightarrow \sigma$ (actual variance)
if $\sigma \uparrow$, you need more n .

$$\frac{n\sigma_0^2}{n\sigma_0^2 + \sigma^2} \rightarrow \text{goes to 1}$$

[goes from $\mu_0 \rightarrow \mu_n$]

$$\frac{\sigma^2}{n\sigma_0^2 + \sigma^2} \rightarrow \text{decreases}$$

$$\downarrow \sigma_n^2 = \frac{\sigma_0^2 \sigma^2}{n\sigma_0^2 + \sigma^2} \quad [\text{uncertainty after observing } n \text{ samples}]$$

more and more you observe, lesser will be the uncertainty.

become sure about the distribution.
[uncertainty reduces]

- Parameter update can be done per sample also.
~ batch update.

gives same convergence

Recursive \equiv

$$P(D^n | \theta) = P(x_{n|\theta}) \cdot P(D^{n-1} | \theta)$$

$$\begin{aligned} P(x|D) &= \int P(x|\mu) \cdot P(\mu|D) d\mu. \\ \text{real estimate.} &\sim N(\mu_n, \sigma_f^2 + \sigma_n^2) \end{aligned}$$

↑ ↑
Known uncertainty uncertainty in μ

plugging in the uncertainty

[true bayesian]

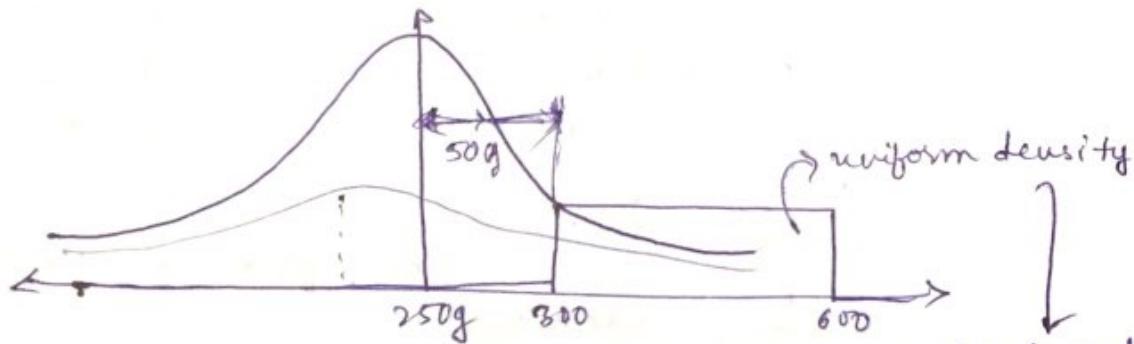
Limiting case, more samples, $\sigma_n \rightarrow 0$. \Rightarrow MAP estimate will be fine.

$$\sim N(\mu_n, \sigma^2)$$

(prior belief)

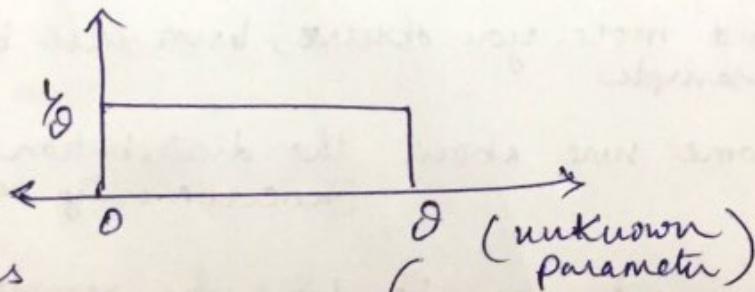
In bayesian, whatever may be the density function, it will converge upon observing more and more samples.

except for uniform density function. (prior belief)



Ex: Special Case -

$$\Rightarrow P(x|\theta) \sim U(0, \theta) \begin{cases} \frac{1}{\theta}, & 0 \leq x \leq \theta \\ 0, & \text{otherwise} \end{cases}$$



Observed 4 numbers

$$D = \{4, 7, 2, 8\}$$

you cannot observe these samples if

$$\theta = 5$$

$$P(7|\theta) = 0$$

what should be θ .

$$P(4|\theta) \cdot P(7|\theta) \cdot P(2|\theta) \cdot P(8|\theta)$$

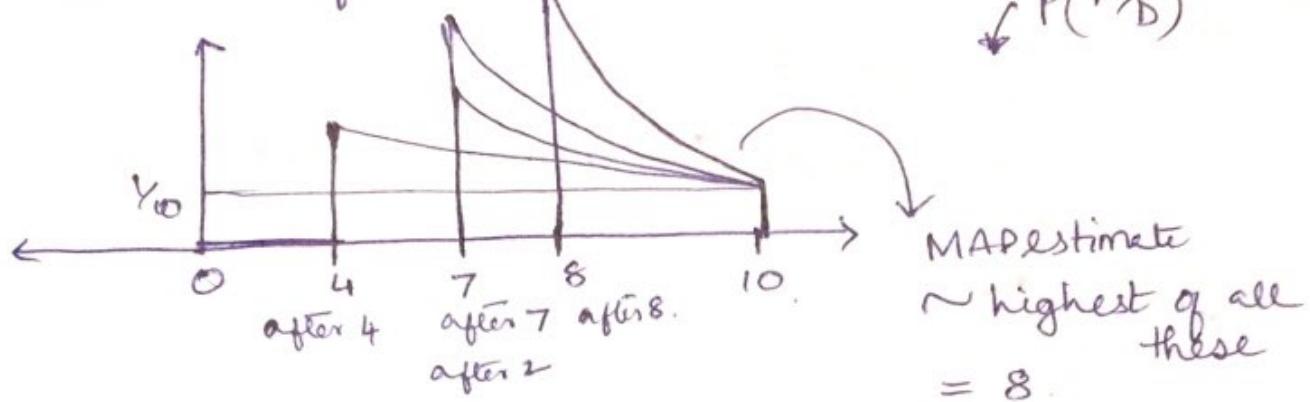
Most Likelihood value of $\theta \equiv 8$

if $\theta = 7.9 \sim \text{product} = 0$

$$\underset{\approx}{\text{Max}} \quad \underline{\theta = 8} \sim \left(\frac{1}{8}\right)^4 \checkmark$$

$$\theta = 10 \sim \left(\frac{1}{10}\right)^4$$

After observing 4, δ value should be greater than 4
 likelihood of δ being less than 4 = 0.



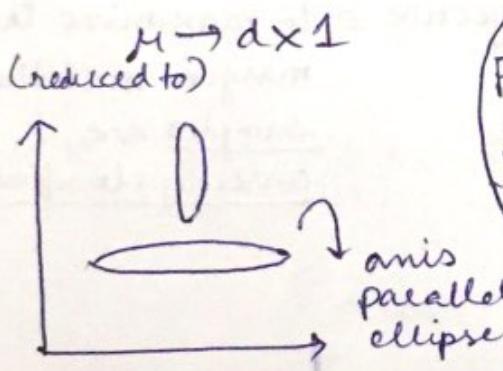
After observing 4, MLE = 4

" 7, MLE = 7

2, MLE = 7

8, MLE = 8.

Naive Bayes Classifier \equiv Features are independent of each other.
 Assume that Σ (covariance) \equiv diagonal matrix



$$\Sigma \quad [\begin{matrix} & & 0 \\ & \ddots & \\ 0 & & \end{matrix}]_{d \times d}$$

off diagonal = 0.
 — correlations of the covariances.

$$P(x) = P(x_1) \cdot P(x_2) \cdots P(x_d)$$

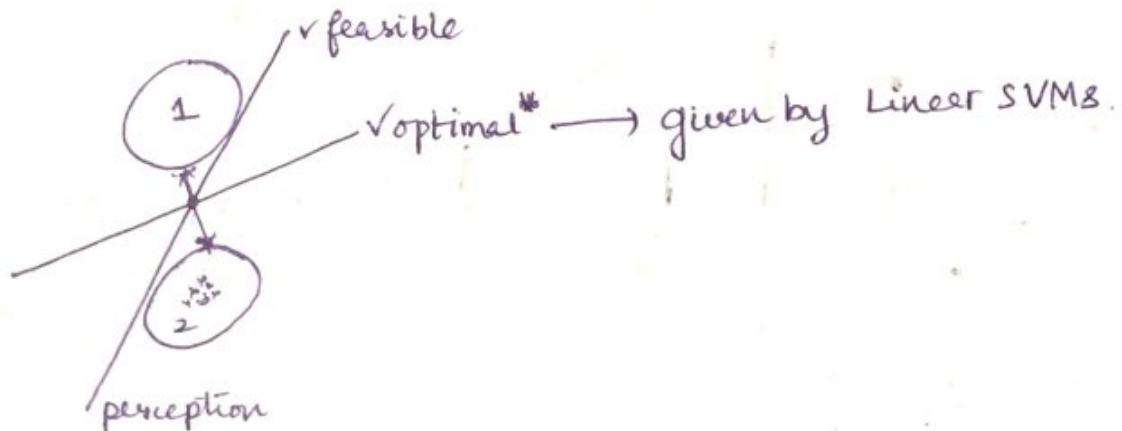
this would cause the no. of parameters to estimate to come down.

in the case of apples; { colour, weight } can be modelled separately.

$$P(x) = P(x_1) \cdot P(x_2) \cdots P(x_d)$$

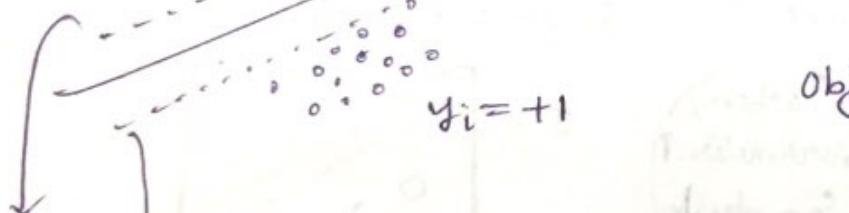
can be density functions (need not be of the same form).

Hyperplane \rightarrow such that the margin between classes is large \leftrightarrow desirable.



Linear SVMs — Support Vector Machines. \equiv max margin

$$y_i = \begin{cases} -1 & \\ 0 & \\ +1 & \end{cases} \quad \begin{array}{l} w^T x + b = -1 \\ w^T x + b = 0 \\ w^T x + b = +1 \end{array} \quad \left. \begin{array}{l} +1, -1 \text{ can be any quantity} \\ \downarrow \\ \text{but equal margin} \end{array} \right\}$$



distance from the origin $= \frac{b+1}{\|w\|}$

distance from the origin $= \frac{b-1}{\|w\|}$

$$\frac{b+1}{\|w\|} - \frac{(b-1)}{\|w\|} = \frac{2}{\|w\|} \equiv \text{margin (has to be maximized)}$$

$$w^T x_i + b \geq +1, y_i = +1$$

$$w^T x_i + b \leq -1, y_i = -1$$

Combining both, we can write

$$\Rightarrow y_i (w^T x_i + b) \geq 1 \quad \forall i$$

$\frac{2}{\|w\|}$ has to be maximized, \rightarrow can be converted to minimization problem \Rightarrow

$$\text{Min } \frac{1}{2} w^T w.$$

$$\text{Min } \frac{1}{2} w^T w$$

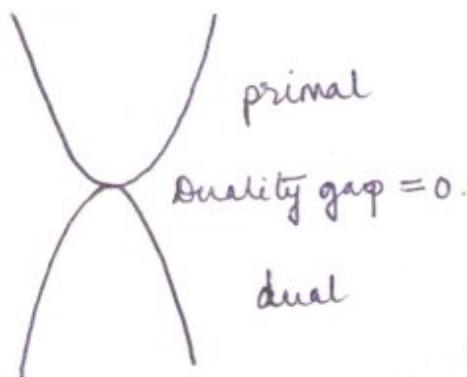
$y_i (w^T x_i + b) \geq 1$

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

\Rightarrow Primal objective.
Hard Margin SVM

Advantage of SVMs = Convex optimisation problems. can be solved using SVMs.
 ↗ solved using gradient descent*
 [not worrying about local minima and all]

optimisation problem —



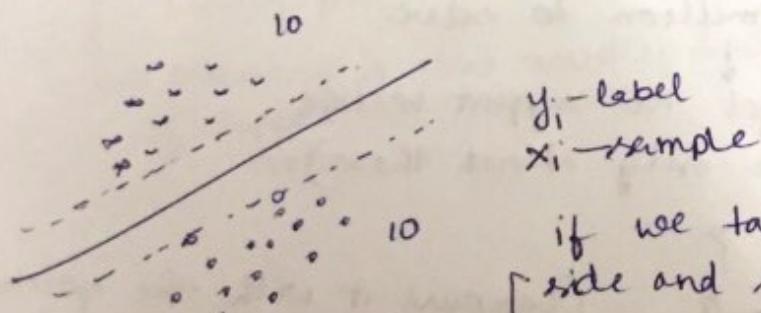
Dual \equiv Max $\sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j x_i^T x_j$ N , no. of samples.

$$\sum_{i=1}^N \alpha_i y_i = 0 \quad ; \quad \alpha_i \geq 0$$

$$w = \sum_{i=1}^N \alpha_i y_i x_i$$

(quadratic programming - Not trivial)

max margin separating hyperplanes. \rightarrow always get the same solution using SVMs



if we take 10, 9 samples on each side and still get the same sol'n
 then utility of that sample = 0.
 $(\alpha_i = 0)$

So, do all the points have $\alpha_i = 0$?

No, not for the case of boundary points, they decide the boundary margin.

$\alpha = 0$ points \Rightarrow dont really matter in the solution

α vector - [sparse = for most of the samples]
values of α are zero

SVMs \rightarrow return non-zero " α "s \rightarrow can be used to calculate optimal "w"

$$\max \left[\sum_i^N \alpha_i - \frac{1}{2} \sum_{i,j}^N \alpha_i \alpha_j y_i y_j x_i^T x_j \right]$$

$$w = \sum \alpha_i y_i x_i$$

if no. of Nonzero " α 's increase \leftrightarrow difficult problem

sum it over the non-zero α 's

SV \equiv Support vectors (with nonzero α)
decide the boundary.

$$w = \sum_{i \in SV} \alpha_i y_i x_i$$

(set of support vectors)

if 1 million samples \rightarrow have 100 support vectors so,

do we only care about 100 support vectors?

No, we process all the samples, when we start
 \rightarrow we dont know which are useful and which are not, use the million to solve

we get 100 support vectors
(we care only about these)

for testing,

$$\rightarrow w^T x + b \leq 0 \quad (\text{compare it with the optimal first line})$$

$$\rightarrow \underbrace{\sum \alpha_i y_i x_i^T x}_w + b \leq 0$$

Computational complexity \rightarrow depends on support vectors.
[if w is not precomputed]

$$(w)^T x = \left(\sum \alpha_i y_i x_i \right)^T x = \sum \underbrace{\alpha_i y_i}_{\text{these are scalars.}} \underbrace{x_i^T x}_{\text{dot product}}$$

Samples come into the problem - as "dot product"

$$(x_i^T \cdot x_j)$$

Scalar independent of the dimension of vector.

Number of Support Vectors \Rightarrow

Error on the future value - (performance on the test data)

training set test set

\rightarrow leave one out (Take 19 and train on them and test on the one out of 20)

Upper bound on the test error = $\left\{ \frac{\#SV}{N} \right\}$.

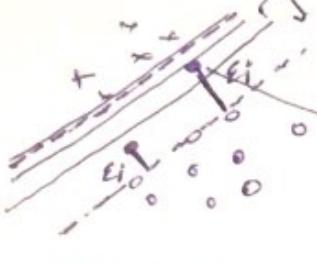
Doubt -

why take all the support vectors if all we need is 2 SVs to define the boundary?

this is 2 D data, (simplified) Data in reality is multidimensional.

pruning is also usually done.
of support vectors.

if support vector is left out, then we might get error.

 boundary has to be this way. C hard margin)
 \Leftrightarrow linear separability (\Leftrightarrow have to include E_i else there won't be any sol'n possible)
 might be an outlier.
 $\{E_i \text{ has to be small.}\}$

- Modify =

$$\left\{ \begin{array}{l} w^T x_i + b \geq +1 - E_i \\ y_i = +1 \end{array} \right.$$

allows some of the samples ~~to~~ that are not linearly separable.

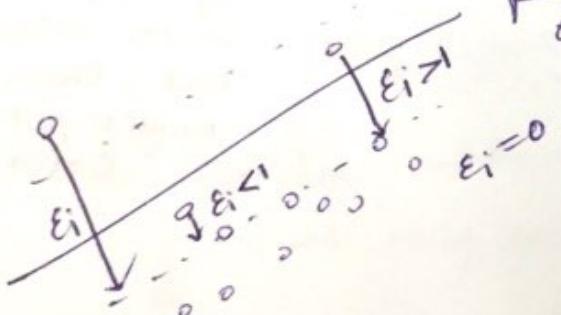
\Rightarrow $y_i(w^T x_i + b) \geq 1$ → might not give solutions to all the samples. (linear inseparable)

modify $\Rightarrow y_i(w^T x_i + b) \geq 1 - E_i \rightarrow$ many samples will have $E_i, \text{ non-zero}$

* allowing violations with penalty.
 $\min \left\{ \frac{1}{2} w^T w + \sum_{i=1}^N E_i \right\}$
 E_i has to be minimized also. else any line is possible.

E_i (allowing a penalty)

penalty constraints on the samples.



* **Soft Margin** "C-constant"

$$\min \frac{1}{2} w^T w + C \sum_{i=1}^N E_i$$

neither large nor small

C has to be fixed

searched over a range.

Tuning type of penalty (power)

$$\min \frac{1}{2} w^T w + C \sum_{i=1}^N E_i^K \quad K=1, 2,$$

L1 SVM

L2 SVM

Tuning no. of samples incurring penalty \rightarrow "consider it to be an outlier"
 Primal \leftrightarrow Dual.

New objective for Primal -

$$\frac{1}{2}w^T w - \sum_{i=1}^N \alpha_i [y_i(w^T x_i + b) - 1] = J(w, b, \alpha)$$

$\alpha_i = 0$
 {beyond the plane}

(constraint) is added.

$=$ zero
 [on the wall]

$w^T w$
 or they become
 indistinguishable.

$$\Rightarrow \frac{\partial J}{\partial w} = 0$$

$$\Rightarrow \frac{\partial J}{\partial b} = 0.$$

$$\Rightarrow w - \sum_{i=1}^N \alpha_i y_i x_i = 0$$

$$\Rightarrow \sum_{i=1}^N \alpha_i y_i = 0.$$

$$* w = \sum_{i=1}^N \alpha_i y_i x_i$$

$$\begin{aligned} J(w, b, \alpha) &= \frac{1}{2} \left[\sum_i^N \alpha_i y_i x_i \right]^T \left[\sum_j^N \alpha_j y_j x_j \right] \\ &\quad \underbrace{\sum_{i=1}^N \alpha_i y_i w^T x_i}_{\text{can be written as}} - \underbrace{\sum_{i=1}^N \alpha_i y_i b}_{0} + \underbrace{\sum_{i=1}^N \alpha_i}_{\because \sum_{i=1}^N \alpha_i y_i = 0} \\ &= \sum_{i=1}^N \alpha_i y_i x_i^T w \\ &= \sum_{i=1}^N \alpha_i y_i x_i^T \sum_{j=1}^N \alpha_j y_j x_j \end{aligned}$$

$$= \frac{1}{2} \left[\sum_i^N \alpha_i y_i x_i^T \right] \left[\sum_j^N \alpha_j y_j x_j \right] - \sum_{i=1}^N \alpha_i y_i x_i \sum_{j=1}^N \alpha_j y_j x_j + \sum_{i=1}^N \alpha_i$$

Maximisation problem =

$$\sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \alpha_i y_i x_i^T \sum_{j=1}^N \alpha_j y_j x_j$$

$$= \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j x_i^T x_j \quad (\underline{\text{Dual}})$$

$\alpha = 0$ SV, SV
 $\alpha \neq 0 : \alpha \neq 0$
 $\alpha \neq 0 : \alpha \neq 0$
boundary

→ objective functions?

Kernels and Non Linear SVMs =

Linear SVMs =

$$\sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$[\mathbf{w}] = d$ (dimension)
 $[\alpha] = (N)$
 (sparse set)

$$\alpha_i > 0, \quad \sum_{i=1}^N \alpha_i y_i = 0.$$

$$\mathbf{w} = \sum_{i=1}^N \alpha_i y_i \mathbf{x}_i$$

$$\mathbf{w}^T \mathbf{x} = \sum_{i=1}^N \alpha_i y_i \mathbf{x}_i^T \mathbf{x} \quad \begin{matrix} \rightarrow d \times \#SV \\ O(d) \end{matrix}$$

Linear dimensionality reduction:-

$$\mathbf{x} \rightarrow \mathbf{x}' = M\mathbf{x} \quad \begin{matrix} \Rightarrow \text{computation easier} \\ \Rightarrow \text{useless information is thrown away} \end{matrix}$$

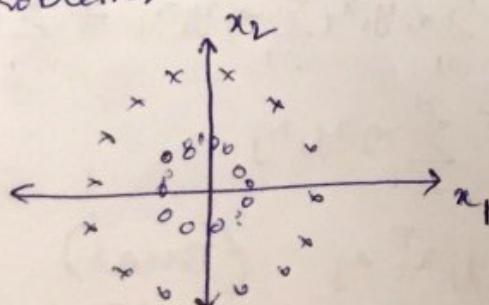
↓
can be nonlinear.

Dimensionality reduction from

$$\mathbf{x} \rightarrow \underline{\Phi(\mathbf{x})}$$

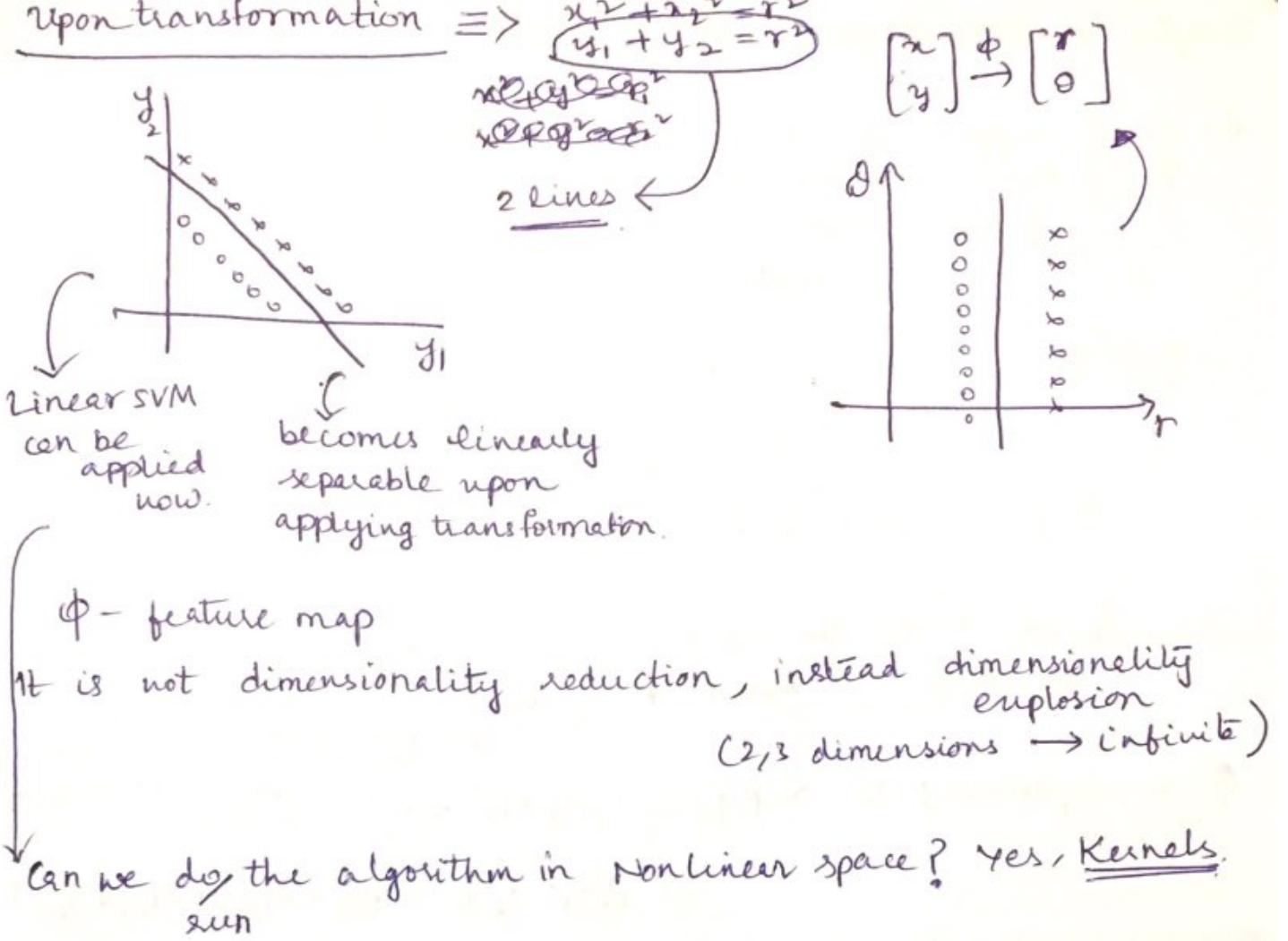
↑ mapping (linear nonlinear)
 linear
 non separable
 problem.

Linear separable → but not linear.
 problem;



Φ = transformation (No inc/dec)

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \xrightarrow{\Phi} \begin{bmatrix} x_1^2 \\ x_2^2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$



Let 2 points,

$$p = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} \quad q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \Rightarrow p^T q = p_1 q_1 + p_2 q_2$$

applying ϕ $\phi(p) \equiv$

$$\begin{bmatrix} p_1 \\ p_2 \end{bmatrix} \xrightarrow{\phi} \begin{bmatrix} p_1^2 \\ p_2^2 \\ \sqrt{2} p_1 p_2 \end{bmatrix}$$

Now on $q \rightarrow \phi(q) \equiv$

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \xrightarrow{\phi} \begin{bmatrix} q_1^2 \\ q_2^2 \\ \sqrt{2} q_1 q_2 \end{bmatrix}$$

$$\phi(p)^T \cdot \phi(q) = p_1^2 q_1^2 + p_2^2 q_2^2 +$$

$$\sqrt{2} p_1 q_1 \sqrt{2} p_2 q_2$$

$$= p_1^2 q_1^2 + p_2^2 q_2^2 + 2 p_1 p_2 q_1 q_2$$

$$= (p_1 q_1 + p_2 q_2)^2$$

$$= (p^T q)^2 \equiv K(p, q)$$

Kernel

Computing dot product in feature space

\Leftrightarrow Computing dot product in input space with square

\sim depends on ϕ

gives dot product in 3D

Samples in SVM come as dot products. (algorithm ^{dual} = $\max_{\alpha} \sum_i \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_j$)

for this $\Phi = \{$ suppose if we put square over there (in SVMs then it means we have solved it in the obj funcn) & new complex feature space

→ Homogeneous kernels

$$\Phi(\mathbf{p})^T \Phi(\mathbf{q}) = (\mathbf{p}^T \mathbf{q})^2 = K(\mathbf{p}, \mathbf{q})$$

"Kernel function"

$$(\mathbf{p}^T \mathbf{q})^2 \rightarrow \Phi = \begin{bmatrix} p_1^2 \\ p_2^2 \\ \vdots \\ \sqrt{2}p_1 p_2 \end{bmatrix}$$

$$K(\mathbf{p}, \mathbf{q}) = \Phi(\mathbf{p})^T \Phi(\mathbf{q})$$

If we know the Kernel function, then we don't care about Φ (feature map)

→ Polynomial kernels

Φ corresponding to $K(\mathbf{p}, \mathbf{q}) = (\mathbf{p}^T \mathbf{q} + 1)^2$ ~ equivalent to mapping to a 6

$$\equiv p_1^2 q_1^2 + p_2^2 q_2^2 + 2p_1 q_1 + 2p_2 q_2 + 2p_1 p_2 q_1 q_2 + 1$$

$$\begin{bmatrix} p_1 \\ p_2 \end{bmatrix} \xrightarrow{\Phi} \begin{bmatrix} p_1^2 \\ p_2^2 \\ \sqrt{2}p_1 \\ \sqrt{2}p_2 \\ \sqrt{2}p_1 p_2 \\ 1 \end{bmatrix} \sim 6 \text{dimensional feature space}$$

Kernels can be anything ~

$$\text{Ex: } K(\mathbf{x}, \mathbf{y}) = e^{-\gamma \mathbf{x}^T \mathbf{y}}$$

Equivalent to infinite dimensions.

$$K(\mathbf{x}, \mathbf{y}) = \tanh(\mathbf{x}^T \mathbf{y})$$

Independent of dimensions, dot product is a "scalar".

Kernel :-

- * Power of Non-linearity
- * Computation efficiency of linear

$$\Sigma x_i - \frac{1}{2} \sum \sum x_i x_j y_i y_j \underline{\mathbf{x}_i^T \mathbf{x}_j}$$

↓
can be replaced according to kernel

2 feature maps \rightarrow corresponding to some kernel.

$$\text{For } \Phi = \begin{bmatrix} p_1^2 \\ p_2^2 \\ p_1 p_2 \\ p_2 p_1 \end{bmatrix} \quad \Phi_1 \rightarrow \begin{bmatrix} p_1^2 \\ p_2^2 \\ p_1 p_2 \\ \sqrt{2} p_1 p_2 \end{bmatrix} \quad \Phi_2 \rightarrow \begin{bmatrix} p_1^2 \\ p_2^2 \\ \sqrt{2} p_1 p_2 \end{bmatrix}$$
$$\Phi_1 \cdot \Phi_2 \rightarrow K(p, q) = \underline{(p^T q)^2}$$

Φ = feature map

$K(\cdot, \cdot) =$

$$(x_i^T x_j)^d, (x_i, x_{j+1})^d, \\ e^{-\gamma x_i^T x_j}$$

(exponential kernel)

Kernel Matrix

$$K = [K(x_i, x_j)]$$

it is symmetric, square matrix
of dimensions $N \times N$

$(N \equiv \underline{\text{samples}})$

\Rightarrow if K is PSD, $\exists \Phi$

+ve semi definite.

\Rightarrow (random Φ doesn't give a PSD. K)

Kernel computation
 \approx small computation

* Not all the kernel functions are valid \Rightarrow

rbf Kernel - most popular Kernel
parameter γ (right one has to be selected).

Kernel can be defined on programs also, not only on the feature representation of an entity

$K(P_1, P_2), \dots$

entity of interest = program, tree,

graph, string,

which have to be classified.

Not needed.

equivalent to "similarity function"

can come up with own Kernel functions.

$x \downarrow$ entity that can be compared.
 $\phi \equiv K$

↑
 only need kernel computation is needed, not ϕ computation.

$$\Rightarrow K_3(\cdot) = K_1(\cdot) + K_2(\cdot)$$

if K_1, K_2 are valid, then K_3 is valid.

\Rightarrow linear combination is also a valid Kernel of valid Kernels

$$K = \sum_{i=1}^p \alpha_i K_i(\cdot)$$

↑ ↑ ↑
 valid valid valid.

$$\begin{aligned} & \Rightarrow \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \underbrace{K(x_i, x_j)}_{\Phi^T(x_i) \Phi(x_j)} \\ &= \sum_{i=1}^N \alpha_i - \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \underbrace{K_{ij}}_{\text{Kernel matrix}} \end{aligned}$$

while testing $\rightarrow \text{Sign}(w^T x + b)$ \checkmark (linear) $\xrightarrow{\text{Kernel matrix} = \text{precompute Kernel matrix}}$

$\text{sign}\left(\sum_{i=1}^N \alpha_i y_i \underline{K(x_i, x)} + b\right) \equiv \boxed{\text{Nonlinear SVM}}$

that is why never ^{psl} compute w , computing kernels for support vectors.

\Rightarrow [you need to carry $\{\alpha_i, x_i\}$ after training] $\xrightarrow{\text{unlike perception where only } w \text{ is used for testing}}$

* SVMs became non linear with the help of Kernels

adds storage complexity.

$w^T x + b$ → for linear, is fine but
 for non-linear, you have to store $\{(x_i, y_i)\}$
 \Rightarrow training data $\rightarrow \phi(x)$ } if we have feature map, then
 no need of support vectors.
 test data $x \rightarrow \phi(x)^T w$.

→ Kernel Matrix $(N, N) \equiv$ killer of this problem (SVM)
 ↓
 struggle with large data.
 memory space complexity. ($N \ggg$)

→ PCA: limitations

→ It is linear (KPCA) ≡ Kernel PCA.

→ LDA. ↑
 using kernels.

Linear PCA ≡
 → given x_1, \dots, x_m examples, $x_i \in \mathbb{R}^N$ (N dimensional)
 centered data.

Compute the covariance matrix, $C = \frac{1}{M} \sum_{i=1}^M x_i x_i^T$, C has $N \times N$ dimensions, $\min(N, M) \in \mathbb{N}$

Compute eigen vectors and project a new sample
 require explicit computation of C^{-1} on $x^T v$.

This is linear, we should Kernelize this.

if $\sum_{i=1}^M \phi(x_i) \phi(x_i)^T$ → we get $\infty \times \infty$ covariance matrix → gives ∞ eigen vectors
 put ϕ all the samples

$$C = \frac{1}{M} \sum_{i=1}^M \phi(x_i) \phi(x_i)^T \rightarrow C v = \lambda v$$

$$v = \frac{1}{\lambda} \sum_{i=1}^M \phi(x_i) \phi(x_i)^T v$$

$$\lambda v = \frac{1}{M} \sum_{i=1}^M \phi(x_i) \phi(x_i)^T v$$

scalar

$$= \frac{1}{\lambda} \sum_{i=1}^M \frac{\phi(x_i) \cdot \phi(x_i)^T \cdot v}{x_i} \quad \text{scalar}$$

KPCA :- $v = \sum_{i=1}^M \alpha_i \phi(x_i)$

PCA in the feature space?

calculating
after eigenvector, $\Rightarrow v^T x$
project.

projection can be
computed without v

$$v \text{ (linear PCA)} = \sum_{i=1}^M \alpha_i \phi(x_i)^T \phi(x)$$

$$= \sum_{i=1}^M \alpha_i K(x_i, x) \quad \text{Kernel.}$$

can be rbf, ...
any kernel

linear combination of
kernels of x_i, x .

non linear surface - $\left[\text{sample}(M) \alpha_s \rightarrow \text{independent of the feature space.} \right]$
to find we have to find only M

1 set of $\alpha_s \rightarrow 1$ eigen vector.

$$CV = \lambda V$$

$$\left(\sum_{j=1}^M \phi(x_j) \phi(x_j)^T \right) \sum_{i=1}^M \alpha_i \phi(x_i) = \lambda \sum_{i=1}^M \alpha_i \phi(x_i)$$

get rid of ϕ , because we know only K . — we have odd no.
of ϕ s on both sides, so multiply.

- on multiplying $\phi(x_k)^T$ on both the sides.

$$\phi(x_k)^T \sum_{j=1}^M \phi(x_j) \phi(x_j)^T \sum_{i=1}^M \alpha_i \phi(x_i) = \lambda \phi(x_k)^T \sum_{i=1}^M \alpha_i \phi(x_i)$$

$\cancel{x_k}$

[Trick: Replace $\phi^T\phi$ with K]

— moving ' ϕ 's across summations
will not have an impact

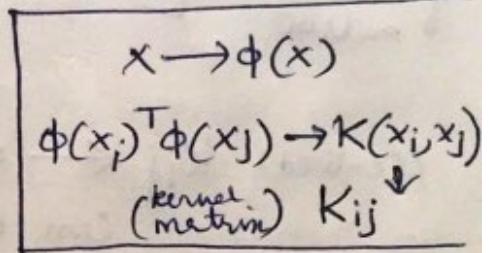
$$\frac{1}{M} \sum_{j=1}^M \phi(x_k)^T \phi(x_j) \sum_{i=1}^M \alpha_i \phi(x_j)^T \phi(x_i) = \lambda \sum_{i=1}^M \alpha_i \phi(x_k)^T \phi(x_i) \quad \forall x_k$$

$$\Rightarrow \frac{1}{M} \sum_{j=1}^M K_{kj} \sum_{i=1}^M \alpha_i K_{ji} = \lambda M \sum_{i=1}^M \alpha_i K_{ki}$$

\downarrow

α_K (one of M samples)
 $\sum_j K_{kj} \rightarrow$ row of kernel matrix.
 k^{th} sample row

$K \bar{\alpha}$ (\prec vector)



$$K \cdot K \bar{\alpha} = \lambda' M K \bar{\alpha}$$

$$K \bar{\alpha} = \lambda' \bar{\alpha}$$

\downarrow

$M \times N$ $\frac{N \times 1}{M}$ Max, M eigen vectors possible

→ $\bar{\alpha}$ is eigen vector of the Kernel matrix, $K_{M \times N}$

→ Nonlinear projection is possible with this new, KPCA if there exists a right Kernel.

Algo:

- Compute Kernel matrix
- Compute Eigen vectors of Kernel matrix
- Each eigen vector has one projection. $(\sum_i \alpha_i K(x_i, x))$

But there's a limitation —

Assumption: data is centered, but in ϕ space?
 ↓
 we assumed
mean is
always
subtracted.

$$\phi(x_i) - \frac{1}{M} \sum_{i=1}^M \phi(x_i)$$

$M_{\phi(x)}$

$$K_{mn} = \phi(x_m)^T \phi(x_n)$$

\downarrow

m, nth
element
of new K
matrix

$$\left[\phi(x_m) - \frac{1}{M} \sum_{i=1}^M \phi(x_i) \right]^T \left[\phi(x_n) - \frac{1}{M} \sum_{i=1}^M \phi(x_i) \right] \approx \phi^T \phi$$

(centered) $K_{ij} \leftarrow \text{original}(K_{ij})$
 can be
derived
from

→ Given x_1, \dots, x_M , $x_i \in \mathbb{R}^N$

→ Compute kernel matrix $K'_{M \times M}$

→ Compute Kernel matrix of centered K . [Centered data]

→ Find α by;

$$K\bar{\alpha} = \lambda \bar{\alpha} \quad (\text{after computing } \alpha, \text{ we})$$

→ For each eigen vector α^k , $(\text{cannot throw away } x)$

$$\sum_{i=1}^M \alpha_i^k K(x_i, x) \rightarrow \text{projection}$$

Kernel LDA > Kernel PCA > LDA > PCA⁴

→ Kernelised variations are much better.

$$\begin{array}{ccc} \text{Linear PCA} & \longrightarrow & \text{KPCA} \\ x^T v_i & & \sum_{i=1}^M \alpha_i K(x_i, x) \\ (\text{linear projection}) & & (\text{projection in higher dimensional space}) \\ & & \text{Non linear} \end{array}$$

LDA \Rightarrow

- After projection, samples within classes should come closer to mean of the respective classes.
- ① within class scatter — minimized
 - ↓ should become compact
 - and well separated.
 - ② between classes scatter — maximized

A, B — 2 classes \Rightarrow

$$(\text{between}) S_B = [M_A - M_B][M_A - M_B]^T \quad M_A \xleftarrow{\text{man}} M_B$$

$$(\text{within}) S_W = S_A + S_B$$

$$= \sum_{i=1}^{N_A} [x_i - M_A][x_i - M_A]^T + \sum_{i=1}^{N_B} [x_i - M_B][x_i - M_B]^T$$

$$\Rightarrow \max_u \begin{bmatrix} u^T S_B u \\ u^T S_W u \end{bmatrix} \quad \begin{matrix} \text{eigen vector.} \\ \Rightarrow \end{matrix} \quad \begin{matrix} A \\ \text{x x x x} \end{matrix} \quad \begin{matrix} B. \\ \text{o o o} \end{matrix}$$

$$\downarrow \quad \max_u [u^T S_B u] \quad \text{such that} \quad u^T S_W u = 1$$

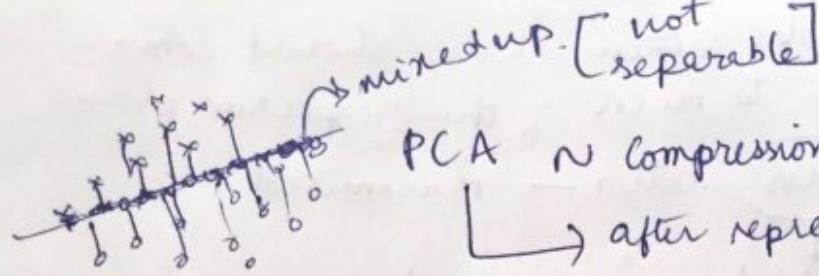
converted to \rightarrow

$$\max_u \left[\frac{1}{2} u^T S_B u - \frac{\lambda}{2} (u^T S_W u - 1) \right]$$

$$\Rightarrow S_B u = \lambda S_W u$$

$$S_W^{-1} S_B u = \lambda u$$

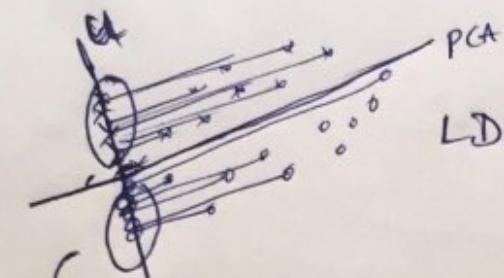
PCA doesn't know class labels \equiv



LDA is not always orthogonal to PCA \Downarrow

PCA \sim compression friendly

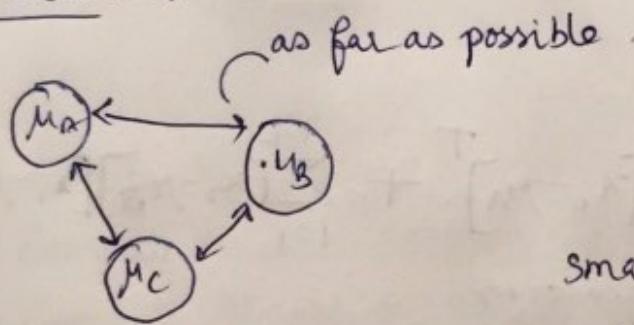
\Downarrow after representation, loss of info is small.



LDA \rightarrow after projection, separability is high.

nicely classified. [linearly separable]

for 3 classes:-



small sample problem.

Problem with LDA \Rightarrow $S_W^{-1} S_B u = \lambda u$

1.) can be resolved: ϵ (small) \Downarrow computationally hectic.

$S_W \leftarrow S_W + \rho I$ (regularise matrices) \Downarrow may not be full rank matrix because, samples < dimensions $[M < N]$.

upon doing it,

rank will increase

changes the singularity.

2) $\text{PCA} \rightarrow \text{LDA}$

dim: 65536
Samples: 520

\Rightarrow apply PCA and reduce it below 520,
now it won't have any inverse
problem

$$3) f_W = \sum_{i=1}^M \sum_{j=1}^M [x_i - \bar{x}_i][x_i - \bar{x}_j]^T$$

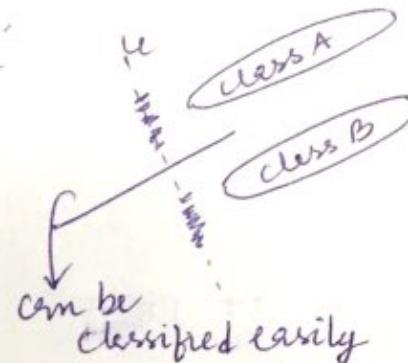
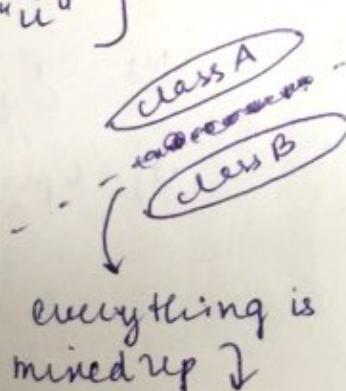
new
definition
of S_W

Generalised Eigen Vector problem — projected onto a plane.

\Rightarrow linear dimensionality reduction techniques:

Unsupervised technique.
 $y = u^T x_i$
 \uparrow
Goal: to find "u"
 \uparrow
PCA

Supervised technique
 \uparrow
LDA
 \uparrow
class labels are needed



$\Sigma B_A \equiv$

* classes are compact and well separated.
 within class between class.

S_W S_B
 \uparrow \uparrow
 has to be minimized. has to be maximized

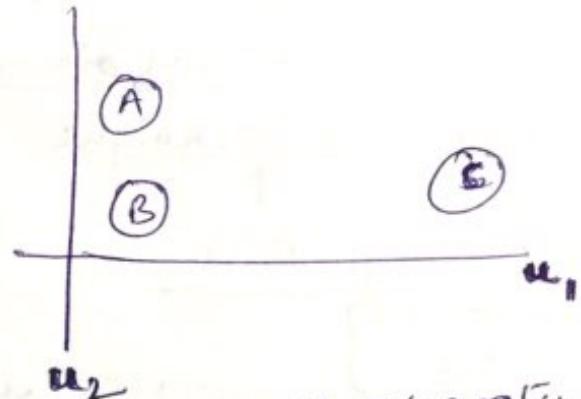
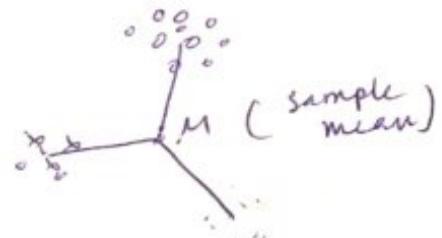
For n-classes :-

$$S_W = S_A + S_B + S_C \dots S_C$$

$$= \sum_{i=1}^C \sum_{j=1}^{N_i} [x_j - \mu_i] [x_j - \mu_i]^T$$

$$S_B = \sum_{i=1}^C [\mu_i - \mu] [\mu_i - \mu]^T$$

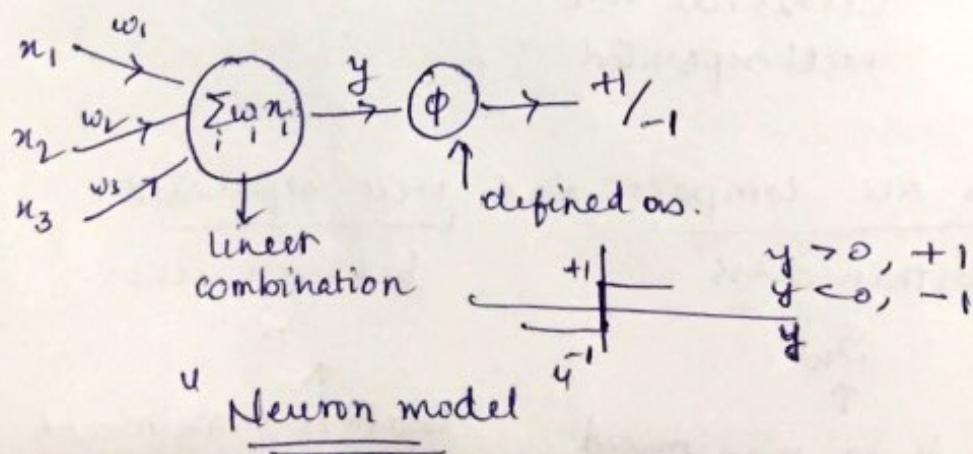
$$\text{better way} = \sum_{i=1}^C \sum_{j=1}^C [\mu_i - \mu_j] [\mu_i - \mu_j]^T$$

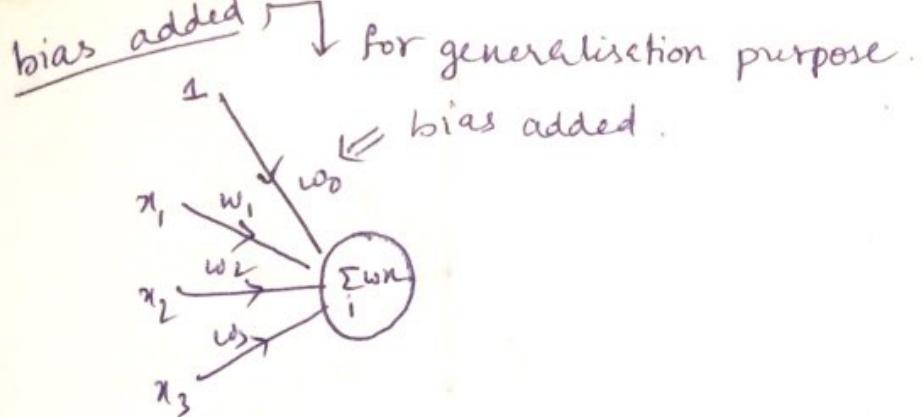


w_1 — separates
C from
A & B
 w_2 — separates
A & B

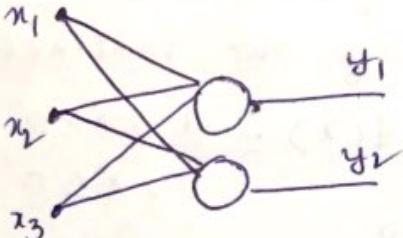
* Neural Networks =

• Previous algorithms = $\underset{\uparrow}{\text{dot product}} \rightarrow \text{sign}(w^T x)$





$\rightarrow x \xrightarrow{(3D)} y \xrightarrow{(2D)}$ $\phi?$



↑
Single layer
networks

Regression =

$\phi =$ linear neuron

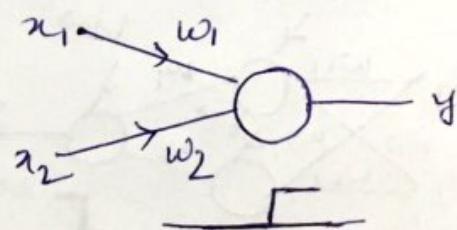
Classification =

Sigmoid, $\phi =$

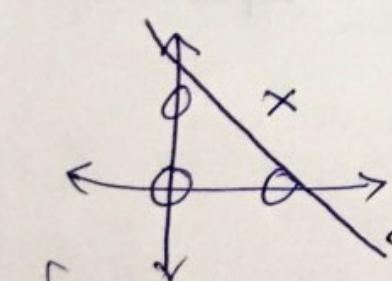
Network design =

Logic - AND

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

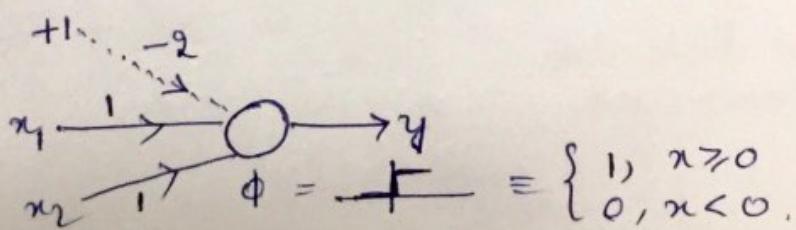


$$w_1, w_2 = ?$$



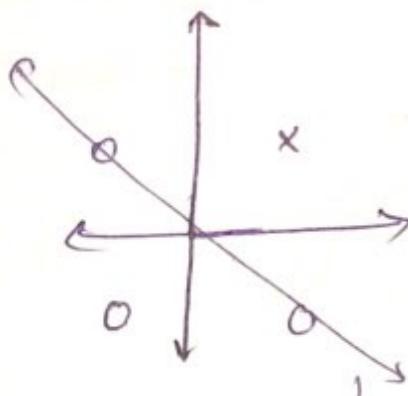
for this line, we need bias also
change the non-linearity \rightarrow to
classify

introducing bias
into the network.



if AND in terms of $-1, 1 \Rightarrow$

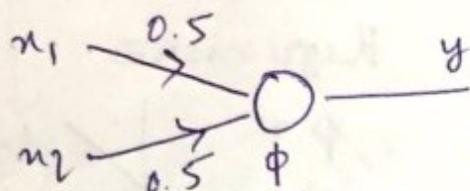
x_1	x_2	y
-1	-1	-1
-1	1	-1
1	-1	-1
1	1	1



Line without bias

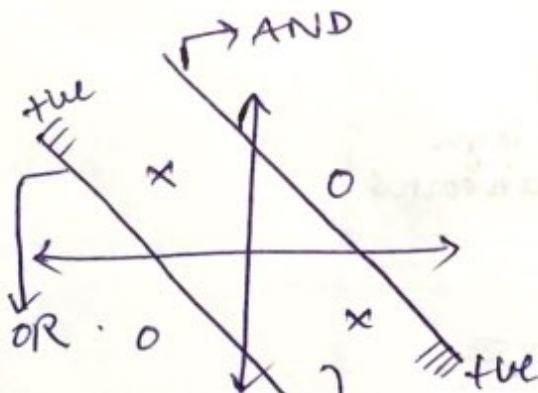
change the nonlinearity

$$\rightarrow \phi(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0. \end{cases}$$

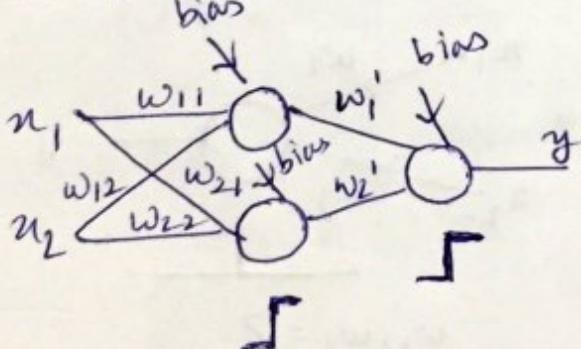


XOR \equiv

x_1	x_2	y
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1

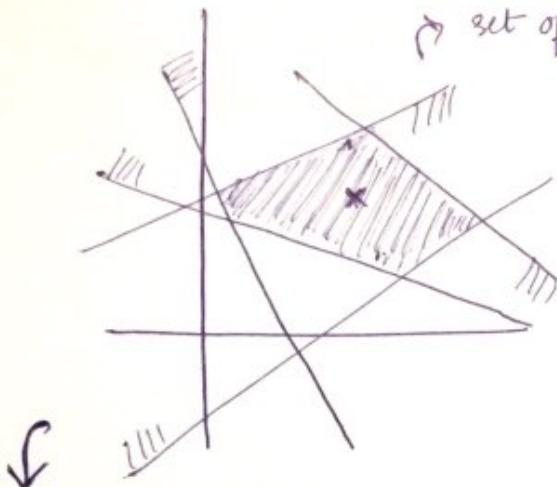


\rightarrow cannot be solved with a linear classifier!



visually represented as

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$



→ set of 2D lines on a plane

Non linearities
↓
getting converted
to +1/0.s

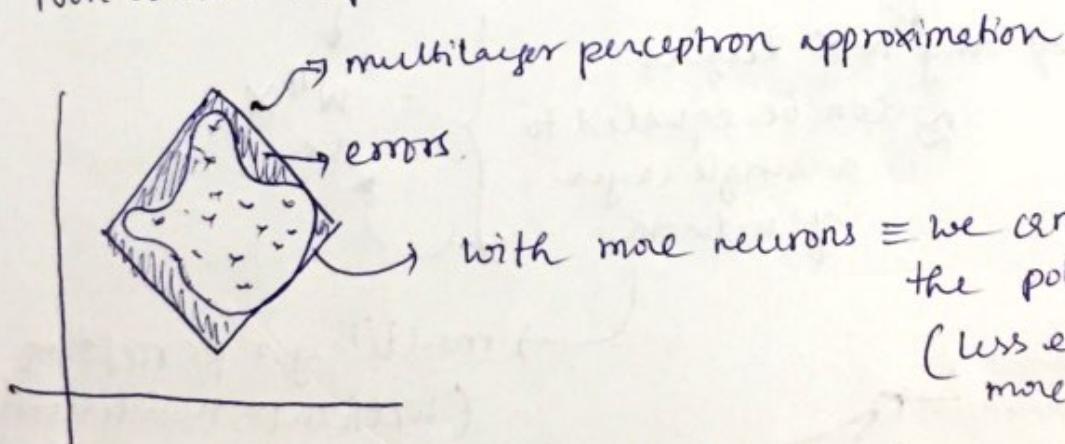
Multilayer Perception

↓
Singlelayer perception - only applicable for linearly separable data.

single layer → Multilayer perception → will be able to
[half planes] [can classify any convex] classify any fig.
regions existing in the space.

⇒ with more layers → we can get arbitrary shapes.
(multiple convex polygons).

Non convex shape



with more neurons = we can increase the polygon sides (less error and more efficient)

→ increasing neurons in the middle hidden layer → ↑ sides of polygon (but still a convex shape)

but width increase in depth
arbitrary shapes

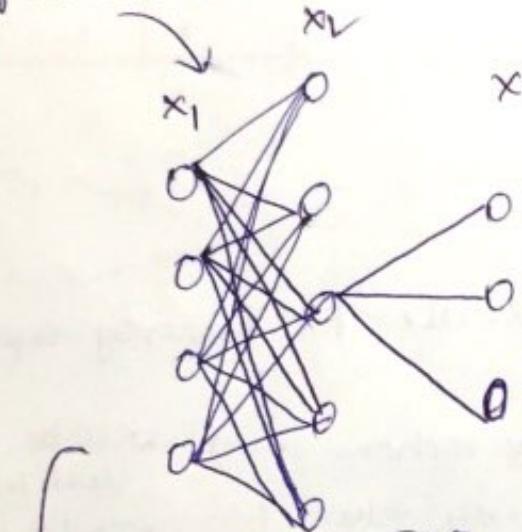
can approximate any set (positive negative)

you will not get a non convex shape.

you are given only the examples [what is inside/outside] → decide the boundary/shape (approximate).

multi Layer Perception:

fully connected networks



4×5 5×3 .

↓ weights have to be learned

$$x_2 = w x_1 \downarrow \\ 5 \times 4$$

$$x_3 = w^T x_2 \downarrow \\ 3 \times 5$$

$$x_3 = \underbrace{w^T}_{w x_1} x_2$$

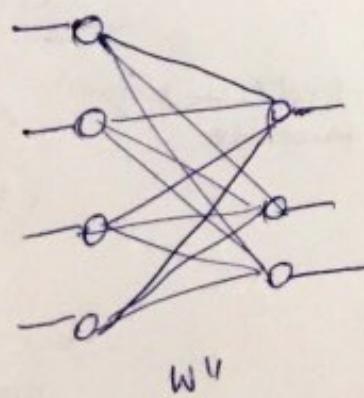
$$= \underbrace{w^T w x_1}_{w x_1}$$

$$= \underbrace{w'' x_1}_{3 \times 4}$$

⇒ any two layers
≈ can be equated to
a single layer
of network

multilayer perception
(without nonlinearity)

≡ single layer
perception



w^U

= This is working because of the absence of non-linearity
but when we introduce non-linearity;

$$x_2 = \phi(wx_1)$$

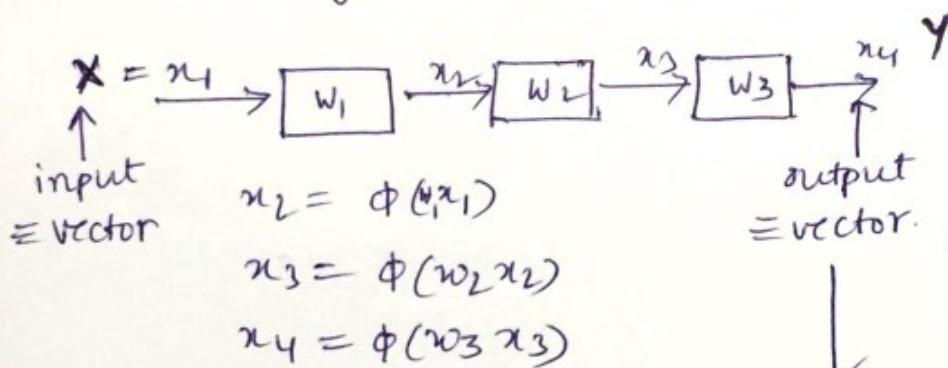
$$x_3 = \phi(w'x_2)$$

$$x_3 = \underbrace{w''x_1}_{= w'''x_1} \times \text{Not possible}$$

$$\phi \equiv \begin{cases} \text{---} \\ \text{---} \\ \text{---} \end{cases}$$

Back Propagation \equiv "for learning weights"

neural network layers \rightarrow as computational blocks.



All the non-linearities \equiv need across layers $\not\equiv$ not be the same

if sample is in class 2

$$\text{Output layer} \equiv \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$w_1 \leftarrow w_1 - \eta \frac{\partial E}{\partial w_1} \quad \text{matrix} \rightarrow \text{matrix}$$

iff of loss w.r.t w_1

$$w_2 \leftarrow w_2 - \eta \frac{\partial E}{\partial w_2}$$

$$w_3 \leftarrow w_3 - \eta \frac{\partial E}{\partial w_3}$$

We have to compute, $\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \dots \equiv$ to update and improve the solution using gradient descent.

\downarrow
we use "chain rule"

We can compute $\frac{\partial E}{\partial w_3}, \frac{\partial E}{\partial z_3}$ directly, but

$$\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial x_3} \cdot \frac{\partial x_3}{\partial w_2} \quad [\text{chain rule}]$$

partial derivatives of output with respect to ^{and} parameters
should be computable.

(input)

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial x_3} \cdot \frac{\partial x_3}{\partial x_2} \cdot \frac{\partial x_2}{\partial x_1}$$

"Chain rule"

then we can apply chain rule to compute E w.r.t any of the parameters.