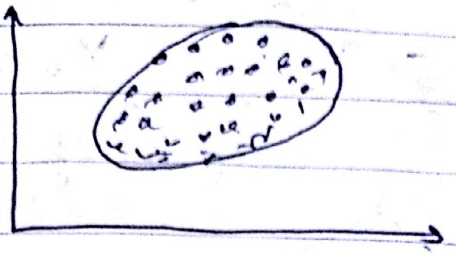
17-OCT-2014

PRINCIPAL COMPONENT ANALYSIS (PCA)



1. Best 0-dim Representation.

Minimising $J(X_0) = \sum_{k=1}^{N} \| X_0 - X_k \|^2$

$= \sum_{k=1}^{N} \| (X_0 - m) - (X_k - m) \|^2$

$= \sum_{k=1}^{n} \| X_0 - M \|^2 - 2(X_0 - m) \sum_{k=1}^{N} (X_k - m) + \sum_{k=1}^{n} \| X_k - m \|^2$

$\sum_{k=1}^{N} (X_k - m)$ → should add upto '0'

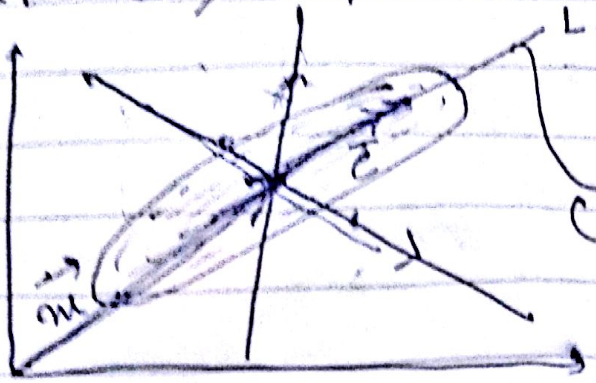
find an 'X_0' that will minimise above expression.

' Here 'm' is assumed to be the mean'

So in order to minimise above we can only change x_0 and it is clearly seen that

$\boxed{X_0 = m}$ is the mini. condition.

2. Best 1-D Representation:-



$X_k = m + a_k \bar{e}$ (any point on this line)

$X_m = \underbrace{a_k}_{\text{scalar}} \bar{e} + \underbrace{m}_{d\ \text{dimensional}}$

In above we know $(\bar{e}$ & $m)$ (same ∀) so for every point we will store 'a_k' (scalar).

$$\underbrace{J(a_1 -- a_n, e)}_{\text{we don't know}} = \sum_{k=1}^{N} \|(m + a_k e) - X_k\|^2$$

$$= \sum_{k=1}^{N} \| a_k e - (X_k - m)\|^2$$

$\hat{e}$ = unit vector

(1)* — $$J(\cdot) = \sum_{k=1}^{N} a_k^2 \underbrace{\|e\|^2}_{1} - 2 \sum_{k=1}^{N} a_k e^T (X_k - m) + \sum_{k=1}^{N} \|X_k - m\|^2$$

└ we want to minimise above by finding a_k & e.

So 2 questions

(1) - what is the best $\bar{e}$.

(2) - what are the best a_n's.

Let us find best a_n's :-

$$\frac{\partial J}{\partial a_k} = 0.$$ [Here we are differentiating only wrt $a_{\boxed{k}}$ → particular]

So

$$2a_k - 2e^T(x_n - m) = 0$$

So

$$\boxed{a_n = e^T(x_n - m)} = (x_n - M)^T e$$

→ simply the Linear projection of $(\overrightarrow{x_n - m})$ over $\vec{e}$. [Dot product]

Now we will find Best $\hat{e}$:→

From (1)* The term $e^T(x_n - m)$ will become a_n for Minimality (ideality)

So

$$J(\cdot) = -\sum_{n=1}^{N} a_n^2 + \sum_{n=1}^{N} \|x_n - m\|^2$$

Since we have to find $\frac{\partial J}{\partial e}$ expand a_k.

$$J(\cdot) = -\sum_{n=1}^{N} e^T (x_n - m)(x_n - m)^T e$$

$$+ \underbrace{\sum_{n=1}^{N} \|x_n - m\|^2}$$ → when we will Diff. This will go to 0.

$$J(\cdot) = -e^T \left[\sum_{n=1}^{N} (x_n - m)(x_n - m)^T \right] e$$

Scatter Matrix { scaled version of Σ (covariance matrix) }

$$\text{Min } J(\cdot) = -e^T S e$$

$$\Rightarrow \text{Maximise } J'(\cdot) = e^T S e$$

We are having here a constraint Maximisation problem subject to constraint $\|e\|_1$

$$e^T e = 1$$

So

$$J(e) = e^T S e + \lambda e^T e + \lambda$$

$$\frac{\partial J}{\partial e} = Se - \lambda e = 0$$

$$\rightarrow \boxed{Se = \lambda e}$$

So perfect 'e' would be a eigen vector, but which one

So That one which corresponds to Maximum eigen value.

(Best $\vec{e}$ = Eigen vector of C with Max. eigen value)

→ Projection That will minimise the error, $J(\vec{e})$ — will be giving the same thing (solⁿ) if we have found the $\vec{e}$, one That Maximises the Between class Scattering.

→ So our Low dimensional feature vectors can be found by

$$E = \begin{bmatrix} | & | & & | \\ e_1 & e_2 & \cdots & e_{10} \\ | & | & & | \end{bmatrix} \rightarrow [\text{Take Top 10 such eigen vectors}]$$

$$\underset{[10\times 1]}{\chi_k} = \underset{10\times d}{E^T} \underbrace{(X_n - M)}_{\text{HDFV}}{}_{d\times 1}$$

χ_k: LDFC

So how many Eigen vectors should we choose, I may $\sum_{\text{D}} \lambda$ = 95% of Total

So Take D of Those

Remember

$$X_{New_K} = E^T (X_{OLD_K} - M)$$

EIGEN FACES:- (samples) — ($N^2 \times 1$ vector, $N \times N$ image)

Faces :- $\Gamma_1, \Gamma_2, \Gamma_3 \dots \Gamma_M$

Mean Face $= \frac{1}{M} \sum_{i=1}^{M} \Gamma_i = \Psi$ (average face)

$$\Phi_i = \Gamma_i - \Psi = (X_K - M)$$ of a sample

So let us define

$$A = [\Phi_1 \ \Phi_2 \dots \Phi_M]$$ [Data matrix of mean sub. faces]

So the Scatter Matrix.

$$S = \sum_{K=1}^{N} \Phi_i \Phi_i^T = \sum (X_K - M)(X_K - M)^T$$

$$S = A A^T$$

Now find eigen vectors of S

$Sv = \lambda v$

Now The problem Here is

$S = AA^T = [10,000 \times 10,000]$ Matrix for a 100×100 image

Suppose

$S_{10K \times 10K} = A_{10K \times 100} \times A^T_{100 \times 10K}$

So we have in 'S' the rank of the matrix S is 100 {rest all eigenvalues are 0} and 100 ⇒ The number of samples. {images}

→ Here we can deal with

$B = S_1 = A^T A \quad [100 \times 100]$

so Let its eigen vectors be

$(A^T A) V = \lambda V$ —(2)

$A^T A$: 100×100

Premultiply above with A

$$[A A^T]Av = \lambda Av$$

$$S\boxed{Av} = \lambda \boxed{Av}$$

↳ 10k x 1 vector

***

So our original $\boxed{e = Av}$

Face Recognition becomes Too easy (solvable) because of above 3 lines

$$\boxed{e = Av}$$

{Guys got a paper out of it}

→ Cuba Project, The community solu"

Surviving Peak oil

Agriculture

- Urban Gardens
- Sustainable Practices
- Land Distributions
- Organic Farming
- Roof top Gardens

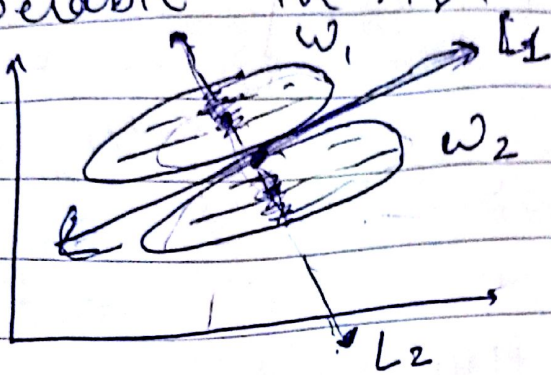
A Large amounts compost prod.

{Bicycles distributed, ∴ No oil}

28-OCT-2014

END-SEM -

- If we use PCA to reduce the dimensionality of samples, then they might not be seperable in Lower Dim. if they were seperable in HD.



S_W S_B —

However in LD there exist a line (like L_2) which avails the info. of seperability just that PCA might not find it

for L_1, $J = |(m+ae) - x||^2$ is minimum but for L_2 is maximum (higher error), so now our idea would be to reduce to LD in order to have higher seperability in LD.

- To have seperability in LD we would want

HD	m_1	m_2	s_1	s_2
LD	$\tilde{m}_1$	$\tilde{m}_2$	$\tilde{s}_1$	$\tilde{s}_2$

$|\tilde{m}_1 - \tilde{m}_2|$ — maximise

$\tilde{s}_1$ and $\tilde{s}_2$ should be as low as possible (in class variance).

$\frac{S_B}{S_W}$ $\quad \tilde{S}_W = \sum S_i$

so our criterian function should look like

$J(L) = \frac{|\tilde{m}_1 - \tilde{m}_2|}{(\tilde{s}_1 + \tilde{s}_2)}$ —— [want to maximise it]

S_W = within class scatter (HD)

$\tilde{S}_W$ = " (LD)

$$S_i = \sum_{k=1}^{N_i} (X_n - M_i)(X_n - M_i)^T$$

$$S_W = \sum_{i=1}^{C} S_i \qquad S_T = S_W + S_B$$

If S_W is the scatter Matrix in HD

Then $\boxed{\tilde{S}_W = W^T S_W W}$ is (scatt. in LD)

where $\vec{W}$ (vector in line)

So we want to Minimise $\tilde{S}_W$

When going to HD we can think of $|m_1 - m_2|$ to be replaced by scatter of Means and we want to maxi. it.

$$S_B = \sum_{i=1}^{C} n_i (M_i - m)(m_i - m)^T$$

[Between class]

for 2 class $S_B = (m_2 - m_1)(m_2 - m_1)^T$

else $S_B = \sum_{k=1}^{c} N_k (m_k - m)(m_k - m)^T$

$\tilde{S}_B = |W^T S_B W|$

we want to Maximise

$$J = \frac{\tilde{S}_B}{\tilde{S}_W}$$, or effectively

we want to Maximise Total scatter, $S_T = S_B + S_W$

↳ { probably the result of PCA }

206 (BishOP)

So our $J = \frac{S_B}{S_W}$

Findat Page

Find $(S_W^{-1} S_B)$ ⟶ eigen vectors line L_2 will be given by it.

$W \Rightarrow S_W^{-1}(m_2 - m_1)$ 2 classes

$y = W^T x$

So above thing is

LDA (Linear discriminant analysis)

or MDA (multi DA)

↳ Proposed by Fischer

∴ = FDA | FLDA | FMDA ..

→ So we can apply this to face Recog. fisher faces vs eigenfaces. (work better)

Consider S_B (d x d) Matrix so →

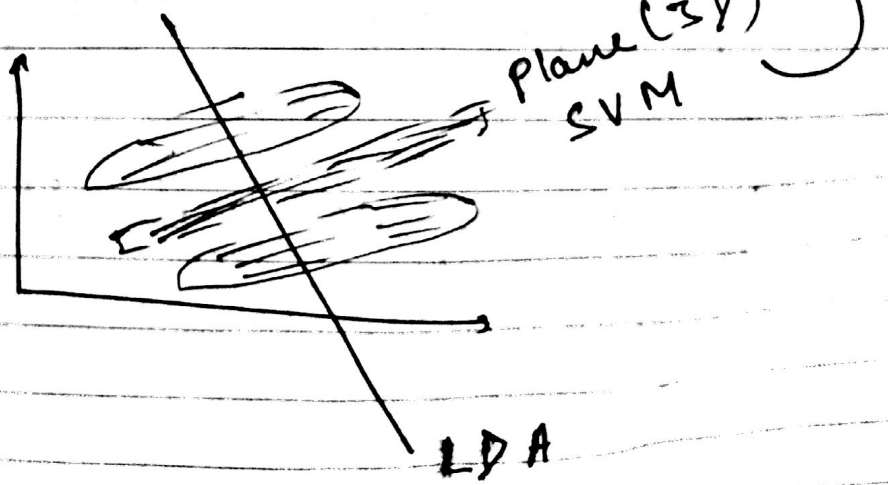
$\begin{bmatrix} \cdot & \cdots & \cdot \\ - & - & - \\ - & - & - \\ - & - & - \end{bmatrix}$ all rows are cR_1 so Rank of the Matrix is '1'.

if $A = B \times C$ or $\leq BC$
The Rank of A is not greater than B or C. ??? (Doubt)

↳ if we have C classes in all we can have utmost (-1 non-zero eigen values of S_B

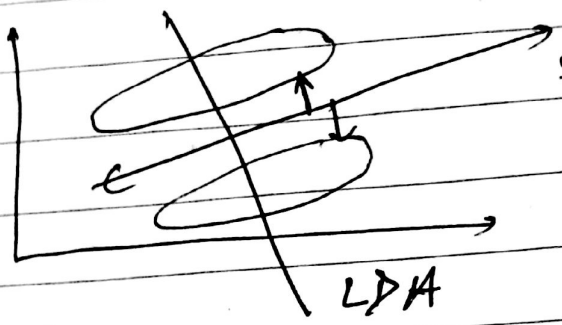
← SVM →

... we want to find the Hyperplane that "MAXIMALLY" seperates the sample, that will be a HP that is perpendicular to line (2D) given by LDA ...



It is somewhat similar to 'Perceptron Learning' but it finds Hyperplane that maxi. sep. the sample.

So Now we do not want to seperate means but to seperate the close samples, Maximize the margin



→ Vapnik - Chervonenkis (VC dimension ——— can we quantify generalisation)

"Huge Breakthrough"

Bound on expected Loss

$$\left[R(\alpha) \underset{\text{classify}}{\leq} R_{train}(\alpha) + \sqrt{f(n)/N} \right]$$

R → Risk (error in classi)

α → asolution

Now this is a HUGE thing to say that my future error is bounded, factor $= \sqrt{f(h)/N}$

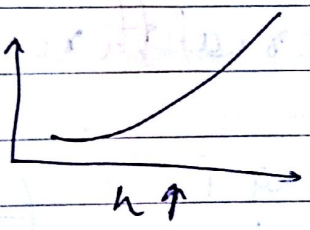
$f(h) = ?$

'h' is the VC dimension

$$f(h) = h + h\log(2N) - h\log h - c$$

$$= h\left(1 + \log\left(\frac{2N}{h}\right)\right) - c$$

overall as $h \uparrow$ $f(h) \uparrow$

infact 

(Log scale)
(less than linear however)

→ for minimising $R(\alpha)$ minimise

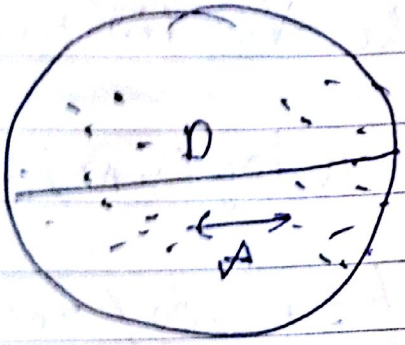
$R_{train}(\alpha) \longrightarrow 0$ (if we can seperate perfectly)

So we need to minimize $\sqrt{\frac{f(h)}{N}}$

⇒ Mini. $(f(h))$ ⇒ min. VC dim (h)

Margin 'Δ' length

as [Δ ↑ h ↓]



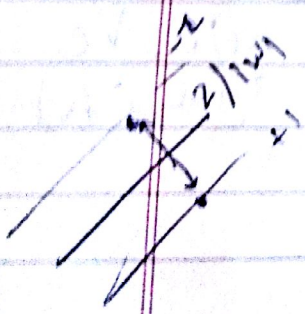
Relative Margin = Δ/D

VC-D, $h \leq \min\{d, \left[\frac{D^2}{\Delta^2}\right]\} + 1$

(Another breakthrough)

so the thing is $\left(\frac{D}{\Delta}\right)$ does not contain 'd'

so for minimising $\frac{D}{\Delta}$ ⇒ Maximise $\frac{\Delta}{D}$, { curse of Higher dimensions has been removed i.e. it does not matter how big 'd' is }



They said this in 1970', no body knew how to use it → 1980's

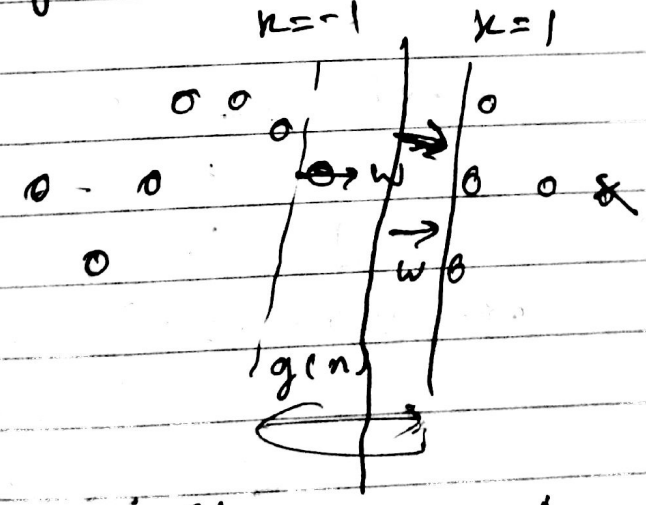
-1990's ⟹ (SVM)

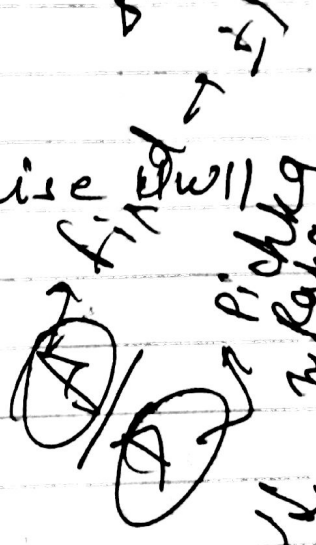
→ so if D is fixed Maximising 'A' will minimise 'h'

31-10-14

- $w^T x + b = \pm \varepsilon$ (margin), but what about higher Dimensions — How to define margin?

- fix $g(x) \geq 1$ and minimise $\|w\|$





minimise $\|w\| \Rightarrow$, $\|w\|^2 = \frac{1}{2} w^T w$

g(x) =

- So we have $\min(\|w\|)$ or $\min(w^T w)$ subject to $d_i(w^T x_i + b) \geq 1 \ \forall d_i$

#1 It's a QP Problem

$$\text{Minimi. } \phi(w) = \tfrac{1}{2} w^T w$$
$$\text{subject to: } d_i(w^T x_i + b) - 1 \geq 0 \ \forall i$$

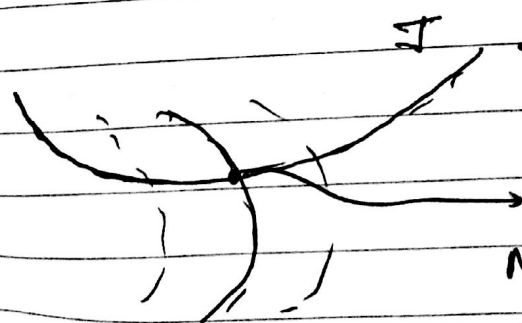
QP solver exists ∴ SVM exist.

equivalently Lagrangian form -

#2
$$\min J(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i d_i (w^T x_i + b) + \sum_{i=1}^{N} \alpha_i$$
$$\text{subject to } \alpha_i \geq 0$$

we want Maximum g_i's with r.t α

→ so Minimize J with respect to W and b, and Maxi w.r.t to α — because of Lagrangian form



$$\frac{\partial J}{\partial d_i} = 0 \ \{\forall \text{dimens} \ \{i \text{ dims}\}\}$$

$$\text{or } \frac{\partial J}{\partial w} = 0$$

{Max flow and Min cut are duals of each other
solve one ⇒ other ∃ - good when one is easy.

Tough: Primer ⟶ Easy: Dual

= So find dual of $J(w, b, \alpha)$

At the optimum -

(1) - $\frac{\partial J}{\partial w} = 0$, (2) $\frac{\partial J}{\partial b} = 0$

a) - $\frac{\partial J}{\partial w} = w - \sum_{i=1}^{N} \alpha_i d_i x_i$

or $w_o = \sum_{i=1}^{n} \alpha_i d_i x_i$ ——(1)

and $d_i = \pm 1$ so w_o is basically a LC of Training samples, only question find α's.

b) - $\frac{\partial J}{\partial b} = 0$, ⇒ $\sum_{i=1}^{N} \alpha_i d_i = 0$ ——(2)

α_i's are Lagrauze Multiplier.

KKT conditions:-
[Karush - Kuhn - Tucker]

(3) = $\boxed{\alpha_i [d_i [w_o^T x_i + b_o] - 1] = 0}$
(for all 'i') — at optimal point

KKT :-

$$\underbrace{\alpha_i}_{(B)}\left\{\underbrace{[d_i(w_o^T x_i + b_o) - 1]}_{(A)}\right\} = 0$$

either $(A) = 0 \rightarrow d_i(w_o^T x_i + b_o) = 1$

∴ They are basically the points on the boundary $k=1$ ——— support vectors

or $(B) = 0$, $\alpha_i = 0$ means that far away samples from the boundry have $\alpha_i = 0$.

So for converting to dual one

$J(w, b, \alpha)$

$$= \sum \alpha_i + \frac{1}{2} w^T w - w^T \sum \alpha_i d_i x_i$$

$$- b \sum \alpha_i d_i$$

{ simply re written the previous J }

At the optimum

#3

$$J(\cdot) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} w_o^T w_o$$

{ will be satisfied }

Now opening w_0

$$J(\cdot) = \sum_{i=1}^{N} \alpha_i - \sum_{i=1}^{N} \alpha_i d_i x_i^T \sum_{j=1}^{N} \alpha_j d_j x_j$$

so we have

$$\#\rightarrow Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j x_i^T x_j$$

{ Now this thing have to be Maximized wrt to α $\because$ w & b are gone }

subject to $\left[\alpha_i \geq 0 \text{ and } \sum_{i=1}^{N} \alpha_i d_i = 0\right]$

→ Dual form

This is also an optimisation problem (QP)

$Q(\alpha) \longrightarrow$ [QP solver] $\longrightarrow (\alpha_i's)$

Now we need to find $g(x) = w_0^T x + b_0$

$$w_0 = \sum_{i=1}^{N} \alpha_i d_i x_i$$ ✓

for any support vector from the data.

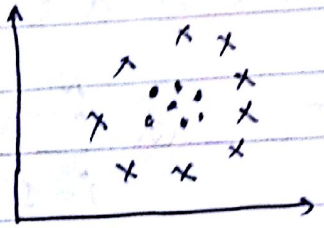
for finding b_0, Take a support vector ($\alpha_i \neq 0$) Then use KKT

$\therefore$ optimum $d_i(w_0^T x_i + b_0) = 1$

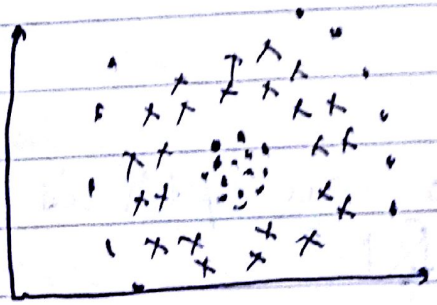
$$b_0 = 1 - w_0^T x_{s+}$$

{It's a Guaranteed optimal solution}

Note →

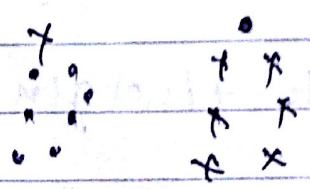


→ Non-Linearly Seperable



→ What about this ?

for Noisy samples:



→ are not Linearly Seperable but we should allow some R(α) Train

So Introduce slack variables $\varepsilon_i \geq 0$

so $\begin{cases} \phi(w) = 1/2\, w^T w \\ d_i(w^T x_i + b) \geq 1 - \varepsilon_i \end{cases} \forall i$

(by allowing $\varepsilon_i > 1$ we can handle Mixing up of samples) — so we have a Training error.

So Minimize $\sum_{i=1}^{N} I(\varepsilon_i > 1)$ → (0 or 1) — Mini it

or Minimise $\left\{ \sum_{i=1}^{N} \varepsilon_i \right\}$ — additional penalties

So Minimise

#5 $\begin{cases} \phi(w) = \frac{1}{2} w^T w + C \sum_{i=1}^{N} \varepsilon_i \\ \text{subject to: } d_i(w^T x_i + b) \geq 1 - \varepsilon_i \\ \varepsilon_i \geq 0, \forall i \end{cases}$

(C → Relative weight)

Its dual form →

#6 $Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j d_i d_j x_i^T x_j$

subject to $0 \leq \alpha_i \leq C \; \forall i$ and

$\sum \alpha_i d_i = 0$ → [additional constraint]

$Q(\alpha) \longrightarrow$ [QP solver] $\longrightarrow \alpha_i$

→ The parameter 'C' controls Relative error between VC dimension & Training error.

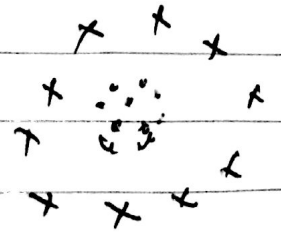
SIS →

- ITER (International Thermo Nuclear Reactor)

2ND-NOV-2014 Non-Linear Boundaries

↳ Good way would be to map them to a ~~Lower~~ Higher Dimension so that samples are linearly seperable.

- So



add an additional dimension 3rd

$x_3 = x_1^2 + x_2^2$

↳ so LD ⟶ | $\phi(x)$ | ⟶ HD

↳ (non-linear Map)

We need not worry about ∆ dimension because Vc Dimension does not depend on it. $\phi(x)$ has to be Non-linear.

so during Traing phase:→ [TRAINING]

svm learning $Q(\alpha) = \sum \alpha_i - \frac{1}{2} \sum\sum \alpha_i \alpha_j d_i d_j \phi(x_i) \cdot \phi(x_j)$

subject to

$0 \le \alpha_i \le C$ & $\sum \alpha_i d_i = 0$

{same methods but do $x_i \to \phi(x_i)$}

8

[TESTING PHASE]

$Label = sign(w_0 \cdot \phi(x_{test}) + b_0)$

$w_0 = \sum_{i=1}^{N} \alpha_i d_i \phi(x_i)$

dot product of svand sample

$\therefore \left[Label = sign\left(\sum_{i=1}^{N} (\alpha_i d_i \phi(x_i) \cdot \phi(x_{test}) + b_0 \right) \right]$

[DURING TESTING TRAINING]

The Thing that is required here is only $\phi(A) \cdot \phi(B)$ so we need not know mapping [but should know the dot product of A & B when they are mapped to HD] —— But How??

- So give the Black Box that perform about thing a name "Kernel"
  Here our kernel (K) gives a scalar when you are inputing it two vectors ie-

$$S = K(x_i, x_j)$$
$$= \phi(x_i) \cdot \phi(x_j)$$

so every where

$$\phi(x_i) \cdot \phi(x_j) = K(x_i, x_j)$$

We have delayed ourselves now see the [MAGIC] :→

1. A simple Quad. kernel.

Let us find $K\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right)$

(Suppose) where $\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1^2 \\ x_1 x_2 \\ x_2 x_1 \\ x_2^2 \end{bmatrix}$

$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \end{bmatrix}$

x_3^2
$x_3 x_4$
$x_4 x_3$
x_4^2

and here $K(X, Y) = (X \cdot Y)^2$

$(x_1 x_3)^2 + 2x_1 x_3 x_4 + (x_4 x_4)$

• also here $K(\alpha, \beta)$ is computationally easier to compute by doing $(\alpha \cdot \beta)^2$ rather than

$\alpha \rightarrow \phi(\alpha) \rightarrow \square$ ⟩ dot ↗ ...

$\beta \rightarrow \phi(\beta) \rightarrow \square$ ⟩

original

2D $K(X, Y) = (X \cdot Y)^2 \Rightarrow \phi(X) =$ (4D) $\begin{bmatrix} x_1^2 \\ x_1 x_2 \\ x_2 x_1 \\ x_2^2 \end{bmatrix}$

2D $K(X, Y) = (X \cdot Y)^3 \Rightarrow \phi(X) =$ (4D) $\begin{bmatrix} x_1^3 \\ x_1^2 x_2 \\ x_1 x_2^2 \\ x_2^3 \end{bmatrix}$

3D $K(X, Y) = (X \cdot Y)^3 \Rightarrow \phi(X) =$ (10D)

$$\begin{bmatrix} x_1^3 \\ x_2^3 \\ x_3^3 \\ x_1^2 x_2 \\ x_2^2 x_1 \\ x_1^2 x_3 \\ x_3^2 x_1 \\ x_2^2 x_3 \\ x_3^2 x_2 \\ x_1 x_2 x_3 \end{bmatrix}$$

KERNELS →

A] ↳ Adding to kernels gives you a new kernel.

$K(X, Y) = K_1(X, Y) + K_2(X, Y)$

$\therefore \phi(X) = \begin{bmatrix} \phi_1(X) \\ \phi_2(X) \end{bmatrix} \rightarrow \left\{ \begin{bmatrix} [\,] \\ [\,] \end{bmatrix} \cdot \begin{bmatrix} [\,] \\ [\,] \end{bmatrix} \right.$

- also here $K(x, \beta)$ is computationally easier to compute by doing $(x \cdot \beta)^2$ rather than

$x \rightarrow \phi(x) \rightarrow$ ☐
$\beta \rightarrow \phi(\beta) \rightarrow$ ☐
} dot ↗ ...

original 2D $K(X, Y) = (X \cdot Y)^2 \Rightarrow \phi(X) =$ (4D) $\begin{bmatrix} x_1^2 \\ x_1 x_2 \\ x_2 x_1 \\ x_2^2 \end{bmatrix}$

2D $K(X, Y) = (X \cdot Y)^3 \Rightarrow \phi(X) =$ (4D) $\begin{bmatrix} x_1^3 \\ x_1^2 x_2 \\ x_1 x_2^2 \\ x_2^3 \end{bmatrix}$

3D $K(X, Y) = (X \cdot Y)^3 \Rightarrow \phi(X) =$ (10D)

$$\begin{bmatrix} x_1^3 \\ x_2^3 \\ x_3^3 \\ x_1^2 x_2 \\ x_2^2 x_1 \\ x_1^2 x_3 \\ x_3^2 x_1 \\ x_2^2 x_3 \\ x_3^2 x_2 \\ x_1 x_2 x_3 \end{bmatrix}$$

KERNELS →

A] ↳ Adding to kernels gives you a new kernel.

$K(X, Y) = K_1(X, Y) + K_2(X, Y)$

$\therefore \phi(X) = \begin{bmatrix} \phi_1(X) \\ \phi_2(X) \end{bmatrix} \rightarrow \begin{bmatrix} [\,] \\ [\,] \end{bmatrix} \cdot \begin{bmatrix} [\,] \\ [\,] \end{bmatrix}$

or → $K(X,Y) = \phi(X) \cdot \phi(Y)$

$= \phi_1(X) \cdot \phi_1(Y) + \phi_2(X) \cdot \phi_2(Y)$

also $(X \cdot Y)^2, (X \cdot Y)^3 - - (X \cdot Y)^p$ are all valid kernels.

A Polynomial kernel

$$K_p(X,Y) = (1 + X \cdot Y)^p$$

$$= 1 + X \cdot Y + (X \cdot Y)^2 + (X \cdot Y)^3 - - - + (X \cdot Y)^p$$

RHS is mapping to extremely HD which have all x_1, x_2^2 - - - many terms.

MERCER'S THEOREM

- Using kernels, we avoid explicit mapping with ϕ.
- So we do not have to know what ϕ is as long as we are sure that there exists one.

So $K(X,Y) = \sum_{i=1}^{\infty} \lambda_i \phi_i(X) \cdot \phi_i(Y)$

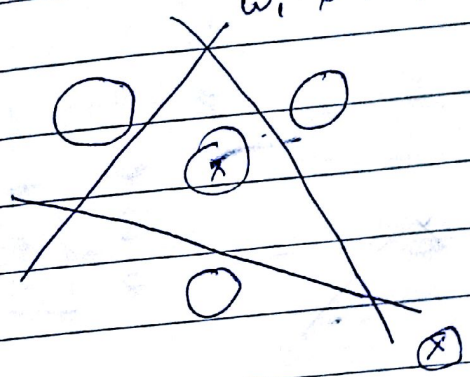
, $\lambda_i > 0$

iff K(X,Y) satisfies above things.
→ then exist ϕ.

7-Nov-2014

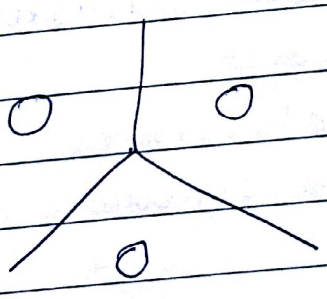
NON-LINEAR MULTICLALASS CLASSIFICRS:-

1. W_1 vs rest W_1



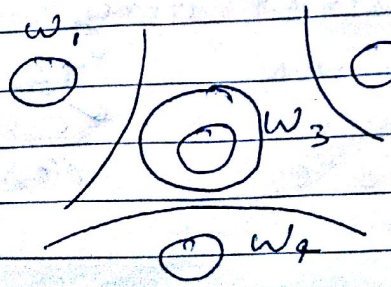
(n) for n classes.

2.



pairwise classifics
$({}^nC_2)$ for n classes.

→ fordisputes we can have Confidence measure approach that will give us probability of a sample belonging to a class, but what about.

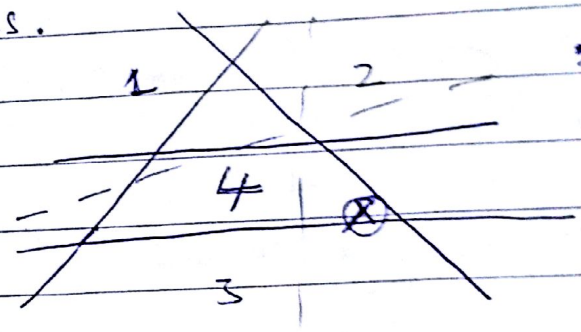


This may possible "SVM", W_1 vs rest W_1

Libraries :-

- Libsvm
- svm lite
- Lib Linear.

- In case of pairwise classifiers find answers for each pair, take the maximum likelihood class.



⇒ classifiers wills say

2	4
2	4
3	4

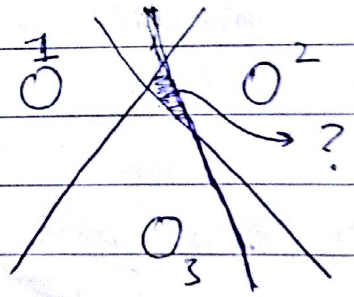
Max.

$4C_2 = 6$

1-2 — 2
1-3 — 3
1-4 — 4 ✓
2-3 — 3
2-4 — 4 ✓
3-4 — 4 ✓

(can have a confidence measure also)

Usually pairwise classifier is good but it can also has situation like



so we can have confidence measures.

for our first strategy

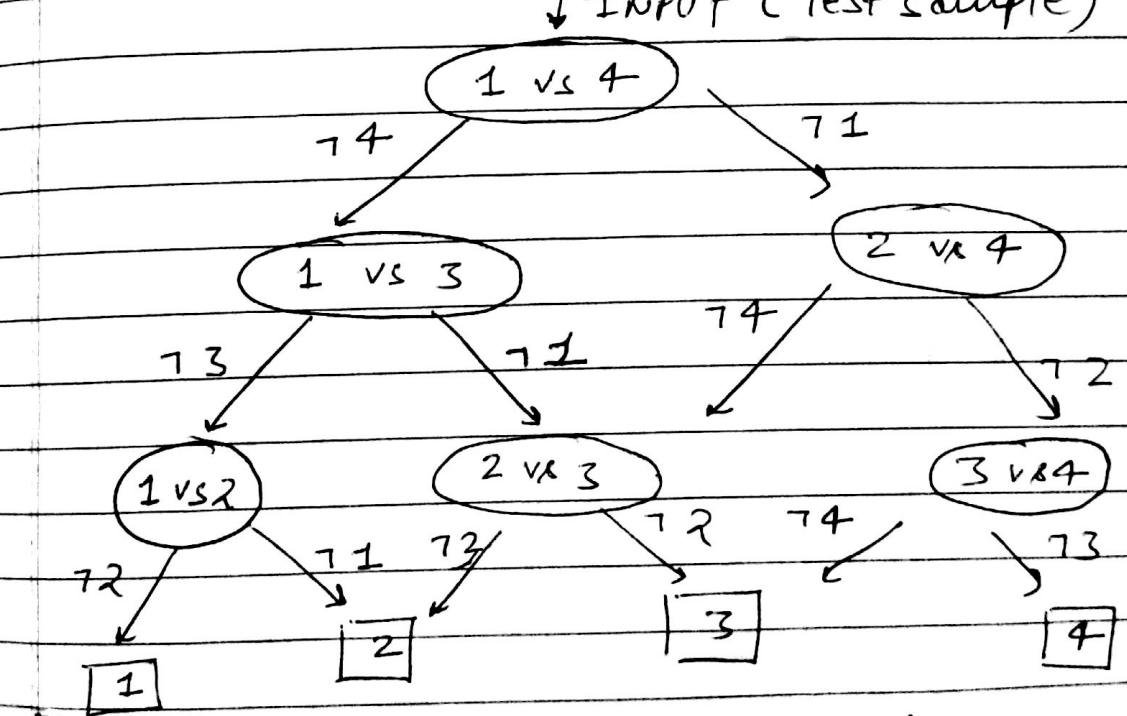
	TRAIN	TEST
TIME	✓O(c)	✓O(c)
SPACE	✓O(c)	O(c)

for second strategy

	TRAINING	TESTING
Time	$O\binom{c}{2}$ ✓	$O\binom{c}{2}$ ✓
space	$O\binom{c}{2}$ ✓	$O\binom{c}{2}$

The ticked measures are quite a lot important.

We can also make decisions in pairwise classifiers like:-



+ {Time complexity is reduced}
≈ O(c)

We have DAG = Directed acyclic graph

But since each edge is a decision so it is a DDAG (Decision directed)

For DDAG

	TRAIN	TEST
Time	$O\binom{C}{2}$	$O(C)$
space	$O\binom{C}{2}$	$O(C)$

→ Time complexity is greatly reduced for testing face from $O(C^2) \rightarrow O(C)$

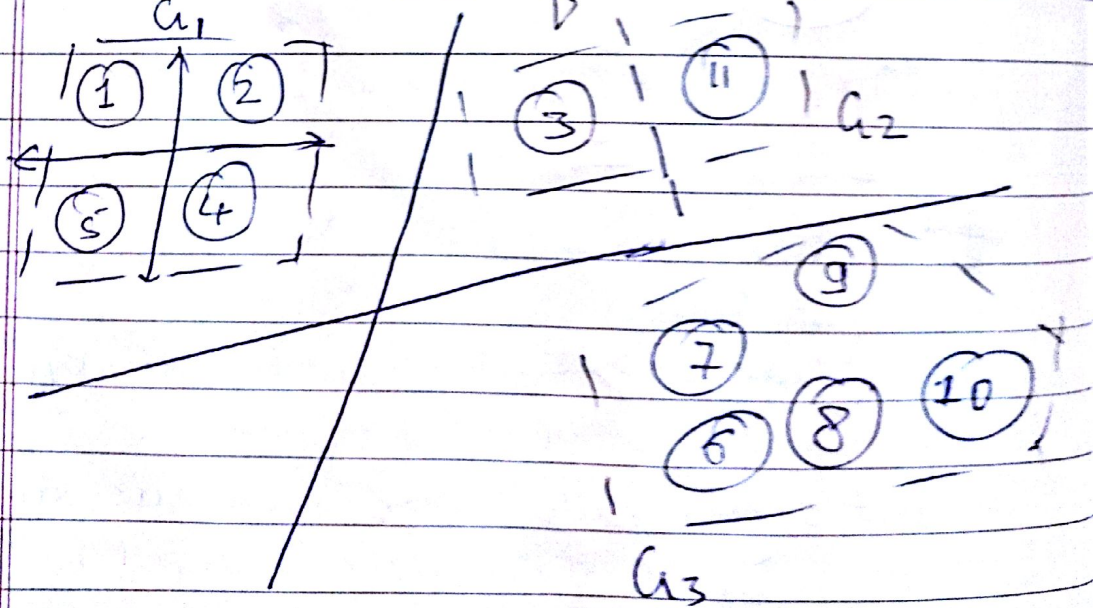
(NEED) for implementing this

1 2 3 4 - - C-1 C

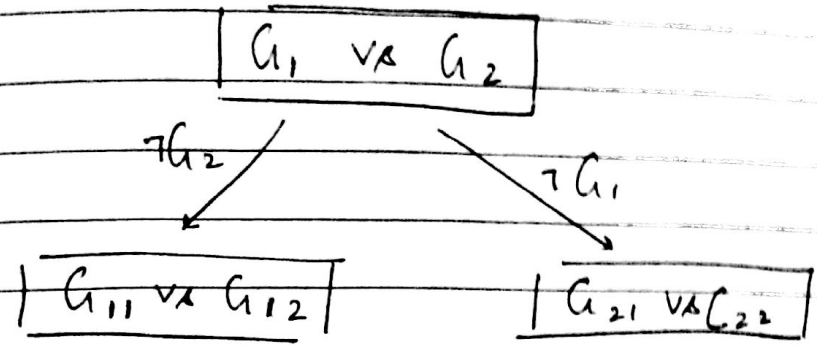
↑L ↑R

Move $L \rightarrow L+1$ or $R \rightarrow R-1$ based on decision of (L vs R)

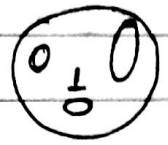
↳ Basically we are eliminating classes based on classifiers, so what if we can eliminate more of them at once.



Basically here we are creating groups and based on decisions we are going into ingroup classification.

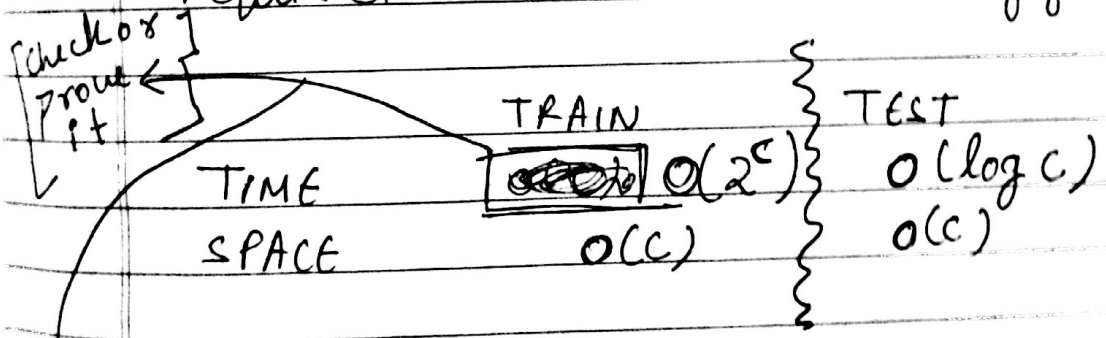


{These are called BDT (Binary decision Tres)}



—But how to group??

↳ How many classifiers would be required for above strategy??



"We can partion given 'C' classes in groups in 2^C possible ways so which partition should we choose.

TRAINIG TIME $\geq O(2^C)$ since we do not know how to choose classifiers Before hand.

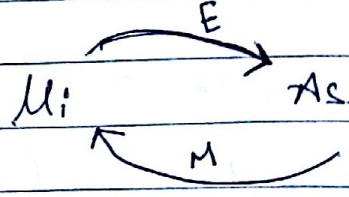
so Like there is a trade off between training & Testing Time, good thing is that Training will be done only once so in practice [like digit recognition Handwriting reco.] this can be preffered over others. {But training time is HUGE!! }

→ [1 class Missed on Clustering]

14-11-14

Expectation - Maximization Algorithms

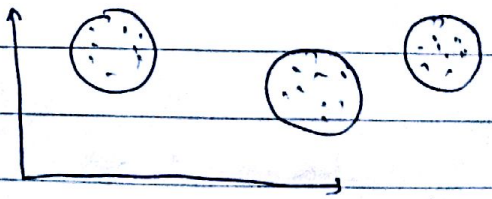
- Like K-Means



└ M = Maximising the likelihood of set of samples coming from Mi

└ E = again think of Mi

{it is like a parameter estimation problem



Like we will have cases Like Variances same, Mean different

---.

[see what happens for k-means]

Generic version of K means → GMM

- all the means are different
- Σ = a diagonal matrix
  {inter class variance = same = 0
  intra " = same }

.. DECISION TREES →

- we have suppose data like Language Now we have to classify people based on language -- attributes that are not quantized but are labels.

Attributes :-
- Numeric
  - Discrete
  - continuous
  - → Till now this only we have read.
- Ordinal } - have order but not exact difference, like - Ranking
- Categorized / Normal
  {color, language}
  Male / Female

Like consider classyfing fruits :-
color - Green, Yellow, Red
size - small, med, Big
shape - Round, square, Long
Taste - sweet, sour, bitter ..

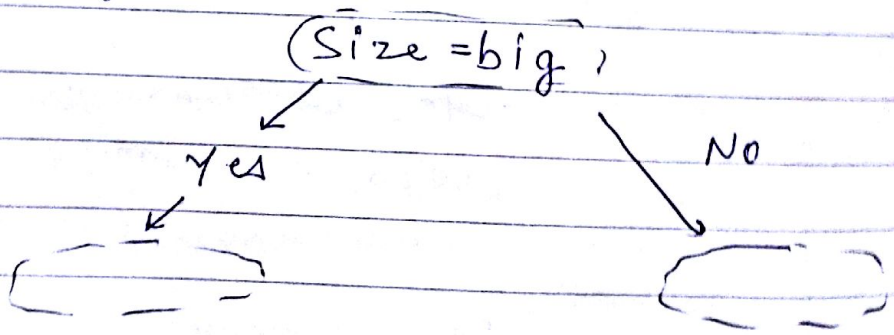
So the problem of classifying by previous in how to compute $x_1 - x_2$?
[Those feature vector differences can't be computed]
↳ Probably we might use Hamming Distance.

fruits = Apples, Mangos, Watermelon, Banana, Grapes...

Que/- 1. A yellow, long fruit
↳ Banana,

2. A round, Green, sweet fruit
↳ Watermelon.

↳ Here we are eliminating the things you can think of it like Decision Trees.



at each point you can ask a binary question.

We have what we call a generic Decision Trees (Need not be binary)
Ask questions till you reach Leaf nodes for all fruits, In such trees you might have to ask same ques. multiple times.

1. How to decide what questions to ask? — at each node (to have splitting).
2. When to stop?
3. How to Label a Leaf-node?

A]→
.. We can define a measure of purity for splits { if more purity } - good..

Like define impurity measure and

$i(N)$, $i(N_L)$, $i(N_R)$

so $\{i(N) - P_L\, i(N_L) - P_R(iN_R)\}$

(IMPURITY Redn)

We have suppose a node that splits into two so

$P(w_1)$ and $P(w_2)$ we have

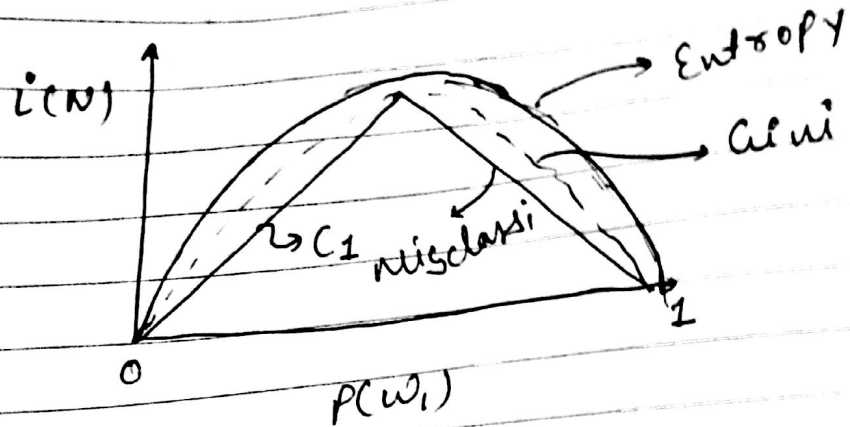
$P(w_1) + P(w_2) = 1$

$0 \le P(w_1) \le 1$

$0 \le P(w_2) \le 1$

also if $P(w_1) = P(w_2) = 0$ { 0 impurity }

$P(w_1) = P(w_2) = 0.5$ { Highest IMPURITY }



for C_1 -

1. Misclassification : $i(N) = \min(P(w_1), P(w_2), \ldots)$

$= 1 - \max_i P(w_i)$

2. Variance / Gini

$i(N) = \sum P(w_i) P(w_j) \quad i \neq j$

Here

$= P(w_1) \cdot P(w_2)$

3. Entropy

$$i(N) = -\sum_i P(w_i) \log_2 P(w_i)$$

(Good question is that for which $i(N)$ is minimised)

{ It's a Good One! }

or [Reduction of Impurity is Maximized]

→ Advantages:
- execution is very fast
- if trained a human can directly analyze / understand

{ so we can explain why we classified or misclassified a sample (can be used in court) ——sometimes very important }

In industry different decision trees might be available on cost. (can have a very complex training algo)

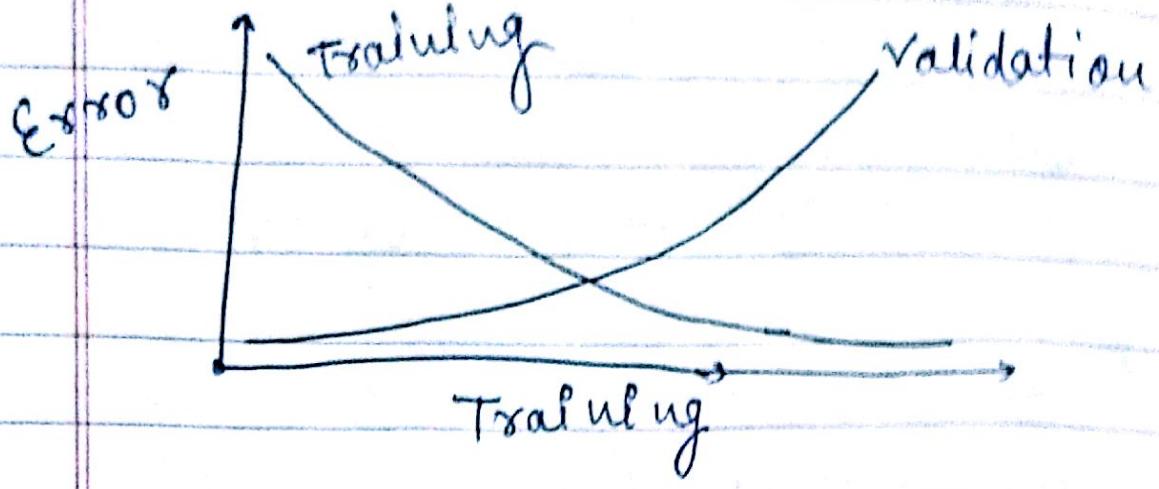
→ We can also have a forest to decide
a forest = { set of Random DecisionTrees }

in some trees, random Noise is introduced.

can be used in deciding if credit-card worthy or not, and at each node a complex question like if income > 10000 can be asked...

Dataset

Training	Testing
validation	



Q) → when to stop?
— when overfitting happens.

21-Nov-2014

.. Suppose we have features that are varying in dimensions [n, m length long]

for example if you try to compare spoken words - { have different lengths ;

Problem - detail , ditail
detaaill

(The length is different, The position of phoneme is different --)

└ Gait Recognition { classify based on how you walk ;

Also we can have a problem, where words are close in spoken format

debatable
detectable

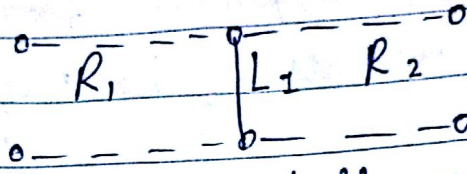
or one measure Edit distance

spectable , frozen
objectable horizon

$$E[i,j] = \begin{cases} \Delta + E[i-1, j-1] & \text{, if } i = j \\ \min \left(E[i, j-1], E[i-1, j] \right) + 1 & \text{, if } i \neq j \end{cases}$$

↳ Dynamic Programming
{optimal substructure property
---}

Litmus Test :-



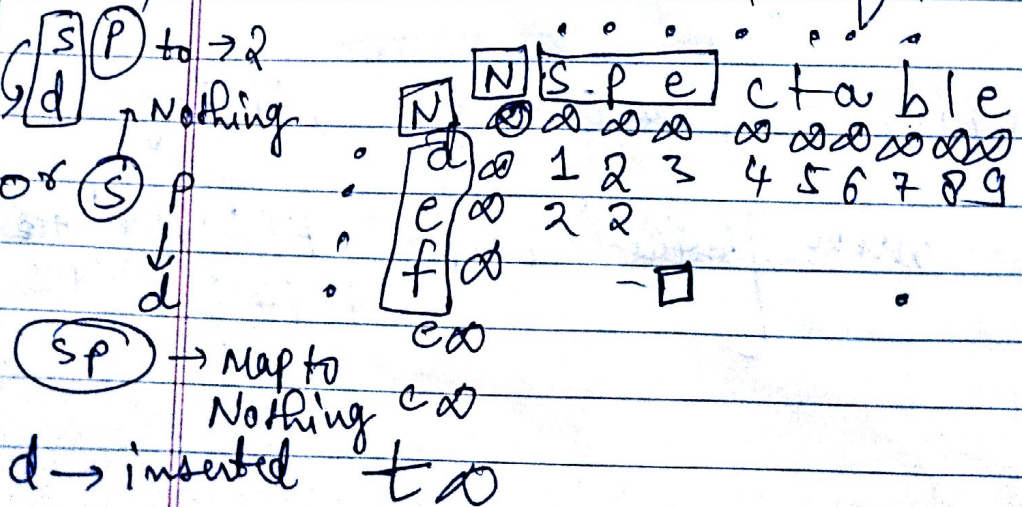
If oracle tells you the value of L_1 then you can find soln of R_1 & R_2 (independently and that will be the global soln)

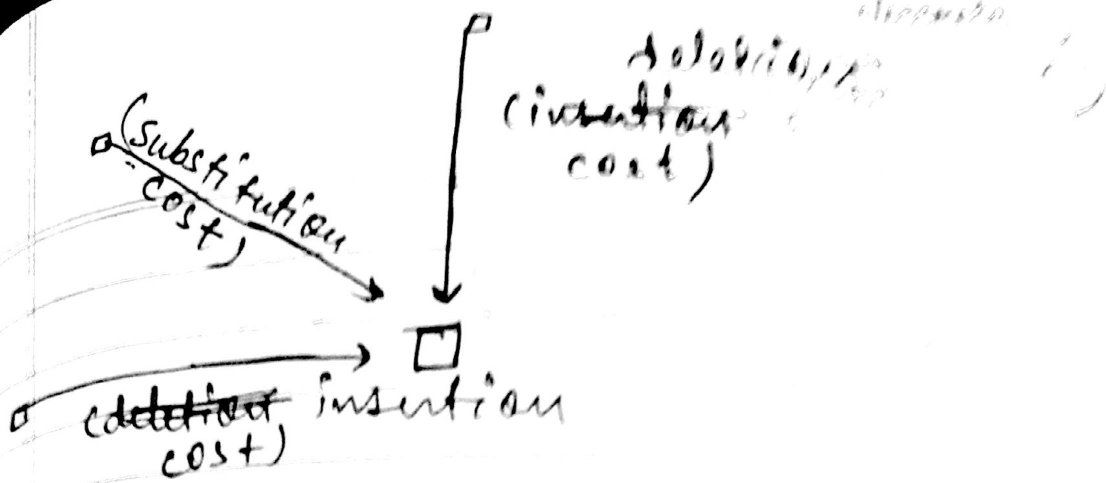
like spectable if t (oracle)
detectable
$\in P_1$ $\in P_2$

↳ Edit distance :-

Insertion cost = I (1)
deletion cost = D (1)
substitution = S (1)

Here we match prefixes.





$$E[i,j] = \min \begin{cases} E[i-1, j-1] + \text{substitu}(i,j) \\ E[i-1, j] + D \\ E[i, j-1] + I \end{cases}$$

To store the path Have a

Trace (i, j) = { whichever is min. } = { Diagonal, Up, Left }

↳ so in case of speech matching or n to m { Dim. mapp }

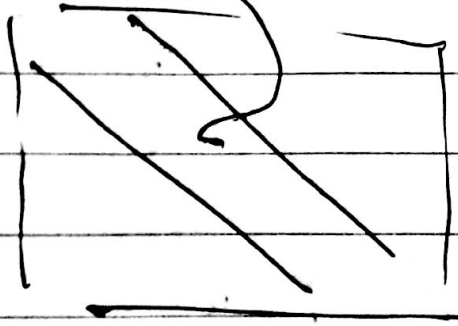
find edit distance { or align

Then { [det] aa [iL] to [det] a [iL] }

So the phonemes are aligned and the costs will be different.

→ Previous method is very popular and is known as DTW (Dynamic Time Warping)

↳ Some optimisation by Sakoe chiba known as Sakoe chiba band ←



The above E(i,j) can be used as a kernel function for SVM's.