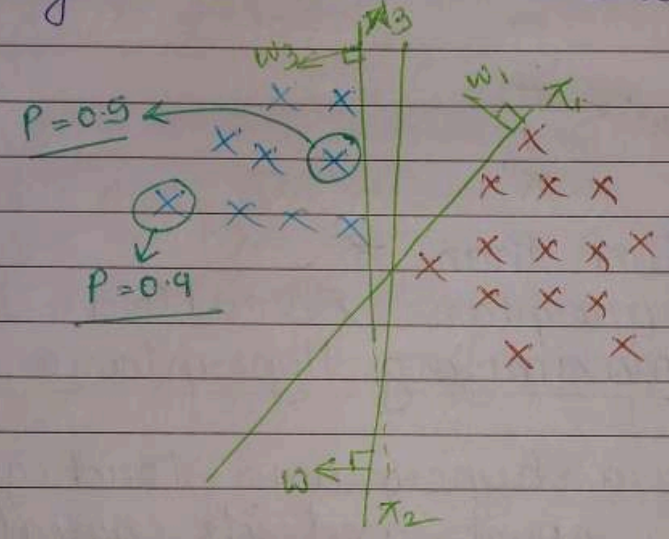
SVM

36.1

SVM → popular ML → classification
→ regression

Geometric Intuition

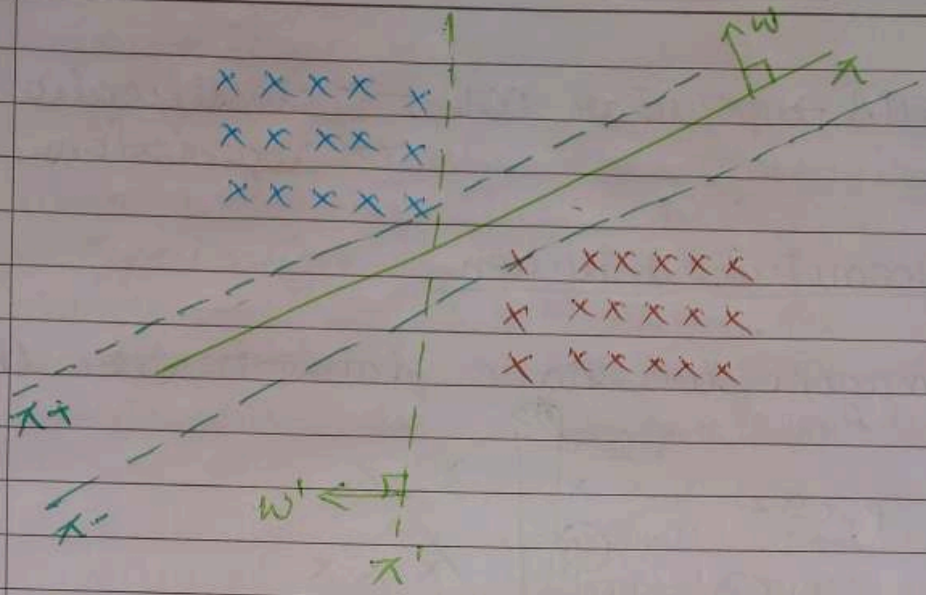
Imagine we have dataset like this:



There are multiple hyperplanes separting the two classes of points

Consider the plane π_3. We see that the positive points are very close to that plane.
We have seen in Logistic Regression that if a point is close to hyperplane, then its probability of belonging to that class is 0.5.

We want a hyperplane which separates +ve points from -ve points as far/widely as possible. This is the core idea of SVM



π is better than π'.
Such a hyperplane is called
Margin Maximizing Hyperplane.

Let π+ be a hyperplane touching
a positive point and its parallel to π
Illy π- touches a -ve point parallel to π

Hence
π+ and π- are parallel to π
π+ and π- are parallel to each other.

dist(π+, π-) = margin

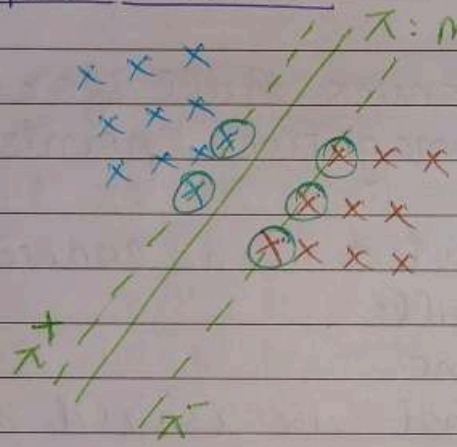
We want to maximize this.
And wider this gap, the better it is

SVM attempt to find the hyperplane
that maximizes the margin

*** If I margin is high, the chance that points would be misclassify would reduce; means my generalization error would reduce (improve)

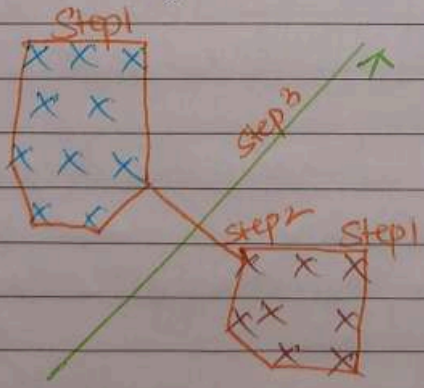
As Margin ↑; Generaliza^{n} Accuracy ↑

Support Vector



Points through which π^+ or π^- pass through are called Support Vectors

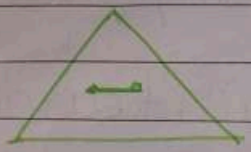
Alternative geometric interpretation of SVM



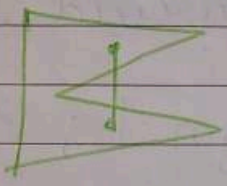
We have to draw a convex hull

Its a convex polygon where all points are either in it or on the polygon

What is convex?
Line between two points in a polygon should lie within the polygon



Convex Polygon



~~No!~~ Non Convex Polygon

① Construct a convex hull for positive and negative points separately
② find the shortest line connecting both the hulls.
③ Bisect the line.
* The plane that we would get is the marginal max plane

36.2 Mathematical Formulation of SVM

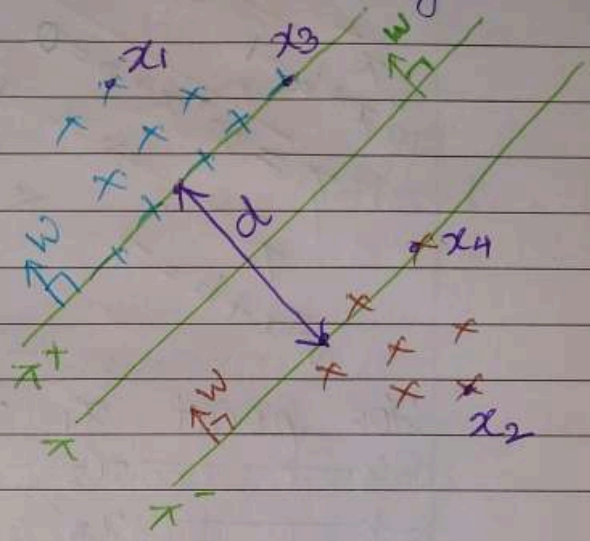
We have to find π with margin maximization

π : margin-max

$\pi : w^T x + b = 0$

π, π^+, π^-

Since ~~w, w^+, w^-~~ are parallel to each other, w will be $\perp$ to all of them



$\pi : w^T x + b = 0$

$\pi^+ : w^T x + b = 1$ } We will see later

$\pi^- : w^T x + b = -1$ } what it is

We did not say $w^T w = 1$; i.e. not a unit vec.

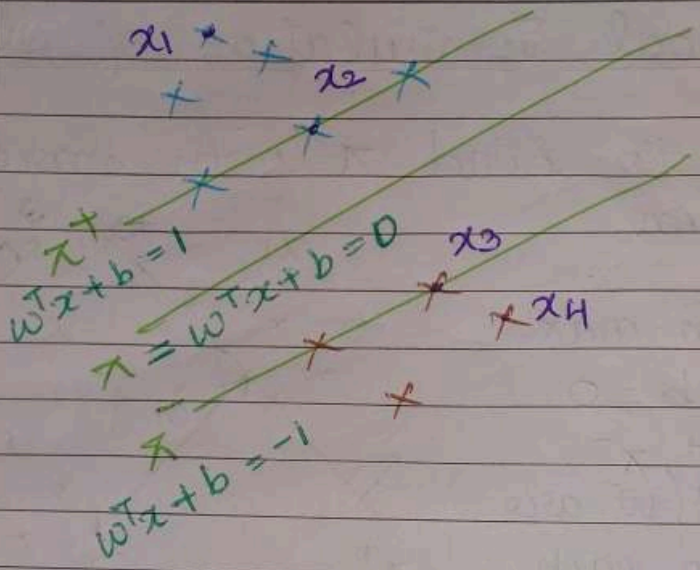
Margin : $\boxed{d = \frac{2}{\|w\|_2}}$

We want to find

$$w^* b^* = \underset{w,b}{\text{argmax}} \left\{ \frac{2}{\|w\|_2} \right\}$$

such that

For point x_i,
$(w^T x_i + b) * y_i \geq 1$
For point x_3,
$y_i (w^T x_i + b) = 1$
For point x_2
$y_i (w^T x_i + b) < -1$
For point x_4
$y_i (w^T x_i + b) = +1$
(-1) (-1)

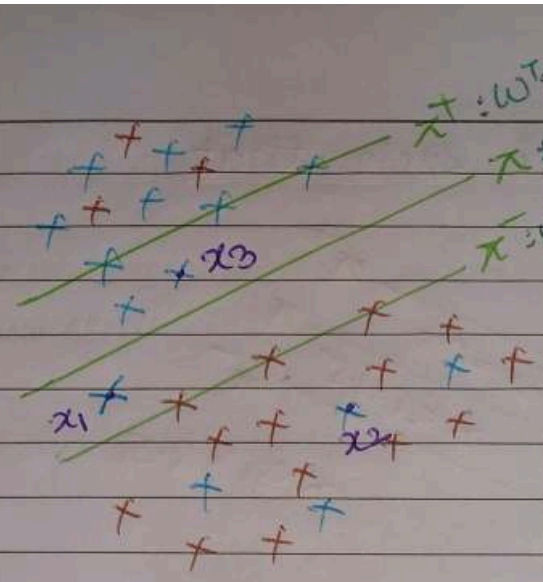


For point x_1 ; $y_i(w^Tx_i+b) > 1$

~~~ x_2 ; $y_i(w^Tx_i+b) = 1$

~~~ x_3 ; $y_i(w^Tx_i+b) = -1$
(-1) (-1)

~~~ x_4 ; $y_i(w^Tx_i+b) \cdot \leftrightarrow + > 1$
(-1) (less than -1) (Resultant product should be >1)

Our problem (optimizaⁿ) is

Constraint opt prob.

$$(w^*, b^*) = \arg\max_{w,b} \frac{2}{\|w\|}$$

such that $y_i(w^Tx_i+b) \geq 1$ for all x_i

The constraint is that all points should lie either in +ve or -ve region. Hence its not 1 constraint but 'n' constraints

Problem with this: Works well on data which is linearly separated.
~~~



Here data is not linearly separated. Hence we wont be able to find w^*, b^* for this dataset.

Hence our optimizaⁿ problem is called as Hard Margin SVM because it has a constraint of being linearly separable

Can we modify the problem to suffice almost linearly separable problem?

Lets assume that point x_i is halfway between π and π^-.

$\therefore y_1 (w^T x_1 + b) = -0.5$ ... Its negative because y_1 is positive class

$\therefore y_1 (w^T x_1 + b) = 1 - (1.5)$

↳ Let this be called ξ_1

For point x_2 ; $y_2 (w^T x_2 + b) = 1.5$
assuming its halfway past π^-

$\therefore y_2 (w^T x_2 + b) = 1 - (2.5)$

↳ Let this be ξ_2

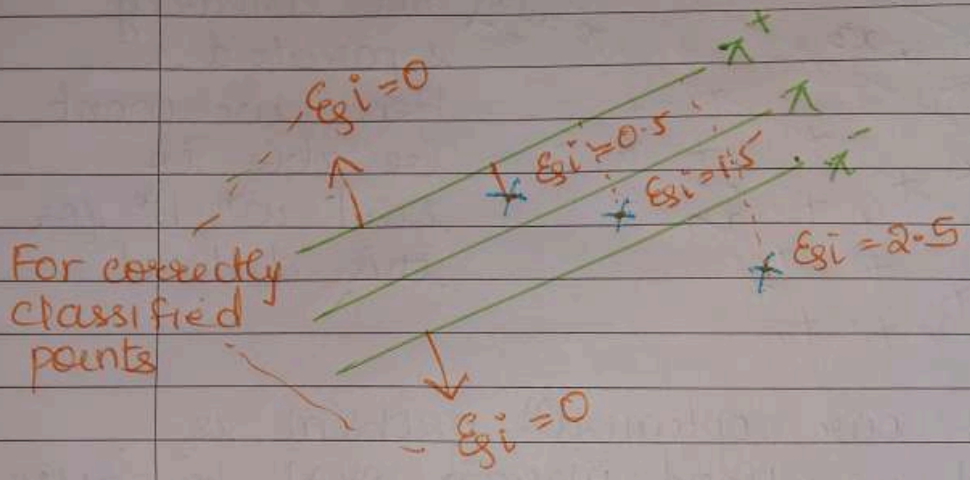
For point x_3; its halfway between π and π^+

$\therefore y_3 (w^T x_3 + b) = 0.5$

$= 1 - (0.5)$

↳ ξ_3

We define ξ_i such that



$\xi_i = 0$ for all correctly classified point
$\xi_i > 0$ for misclassified points depending on how far they are from the plane of the respective class.

For every x_i, we have ξ_i such that

$\xi_i = 0$ if $y_i(w^T x_i + b) \geq 1$
i.e. correctly classified points as per x^+ and x^-

$\xi_i > 0$ and is equal to some units of distance away from correct hyperplane (x^+ or x^-) in incorrect direction.

It is not exactly the distance because w is not a unit vector

Now, we know that

$$\arg\min (f(x)) = \arg\max \left(\frac{1}{f(x)}\right) \text{ where } f(x) \neq 0$$

$$\therefore \arg\max \left(\frac{2}{\|w\|_2}\right) = \arg\min \left(\frac{\|w\|_2}{2}\right)$$

Using ξ_i, we can write optimizaⁿ prob.

$$(w^*, b^*) = \underset{w,b}{\arg\min} \left[\left(\frac{\|w\|}{2}\right) + C \cdot \frac{1}{n} \sum_{i=1}^{n} \xi_i\right]$$

such that $y_i (w^T x_i + b) \geq (1 - \xi_i)$ for each i and $\xi_i \geq 0$

for all misclassified points, we can write like this. As we have seen for x_i: $(y_i)(w^T x_i + b) = 1 - (1 \cdot 5)$

For correctly classified points, $\xi_i = 0$

We want to minimize errors ~~i.e. misclass~~ i.e. misclassification

Now $\frac{1}{n} \sum_{i=1}^{n} \xi_i \Rightarrow$ Average distance of misclassified points from correct πs

We want to minimize this distance, hence we have added it to the term for minimizaⁿ

__/__/__

What is C?

C is the hyperparameter

Regularizaⁿ

Loss

$$(w^*, b^*) = \underset{w,b}{\text{argMin}} \frac{\|w\|}{2} + C \cdot \frac{1}{n} \sum_{i=1}^{n} \xi_i$$

$\frac{1}{\text{margin}}$ — average dist of misclass points

constraints: such that $y_i(w^T x_i + b) \geq 1 - \xi_i \quad \forall i$

and $\xi_i \geq 0$

This formulaⁿ is called Soft Margin SVM

Hence we have

$$\underset{w}{\text{Min}} \; C \cdot \text{Loss} + \text{Regularizaⁿ}$$

As C ↑ ; the tendency to make mistakes on D_{Train} ↓

⇒ overfit ⇒ High Variance

*** comment on video

If C is high, the loss term would dominate the regularization term. In such case, even though the optimization problem tries to minimize the sum of the loss and the regularization term, as the regularization term is negligible when compared to the loss term, the optimization becomes only minimizing the loss term as much as possible, thereby there are chances for ensuring no misclassificaⁿˢ, leading to Overfitting

As $C \downarrow$; its underfit $\Rightarrow$ High-Bias

C behaves exactly opposite to λ in Logistic Regression.

36.3 Why we take values +1 and -1 for Support Vector Machines.

Because, we are not saying $\|w\|_2 \neq 1$
Its any vector apart from unit vector.
Our task is to maximize the margen

Assume, I take

$\pi^+ : w^T x + b = k$ } $k > 0$
$\pi^- : w^T x + b = -k$ }

Why are we taking $+k$ and $-k$ & not k_1 and $-k_2$? Because we want both π^+ and π^- to be equally far away
i.e. dist (π^+, π) $\cong$ dist (π^-, π)

$\therefore$ Margen $= \frac{2k}{\|w\|}$ We know that if

$k = 8$; margen $= \frac{8}{\|w\|}$

It would change as per k.
So k doesnt carry much significance
$\therefore$ We are taking +1 and -1 as way of convinience

From algebraic point,

Lets say: $\pi^+ : w^T x + b = k$

$$\Rightarrow \left(\frac{w^T}{k}\right)x + \left(\frac{b}{k}\right) = 1$$

$$\Rightarrow (w')x + (b') = 1$$

Here we can divide by k, because w is not a unit vector.

Reason to take +1 and -1 : To simplify the math.

36.4 Loss Function (Hinge Loss) based Interpretation

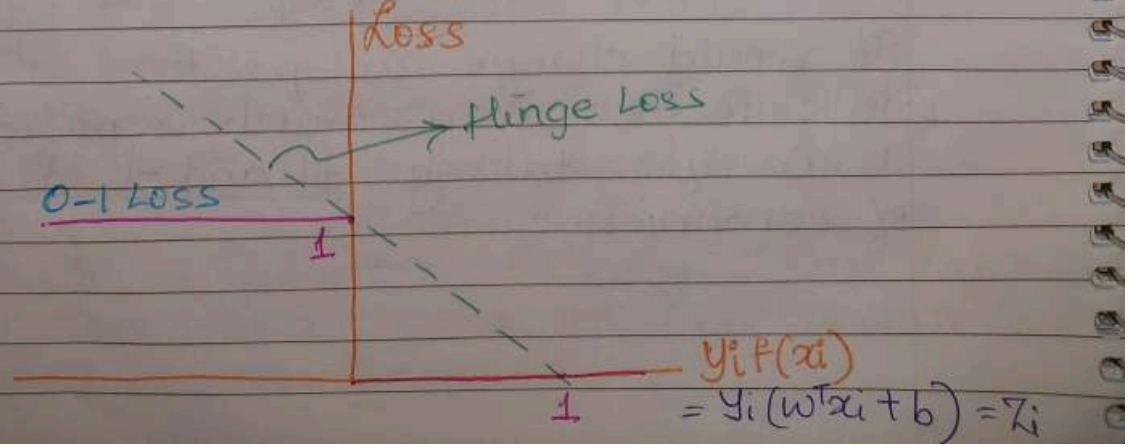
When we study Logistic Regression,

Logistic Regression = Logistic Loss + Regularizaⁿ

Logistic Loss: equivalent to 0-1 Loss

Linear Reg: Linear Loss + Regularizaⁿ

SVM: Hinge Loss + Regularizaⁿ

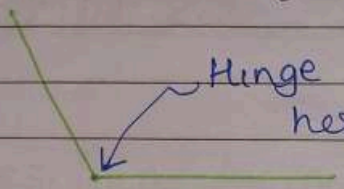


__/__/____

Let $(w^T x_i + b) y_i = z_i$

$z_i > 0$: x_i is correctly classified } 0-1 Loss
$z_i < 0$: x_i is incorrectly classified }

Hinge loss is a straight line from $-\infty$ to 1; after that it becomes 0.



But there are some optimization

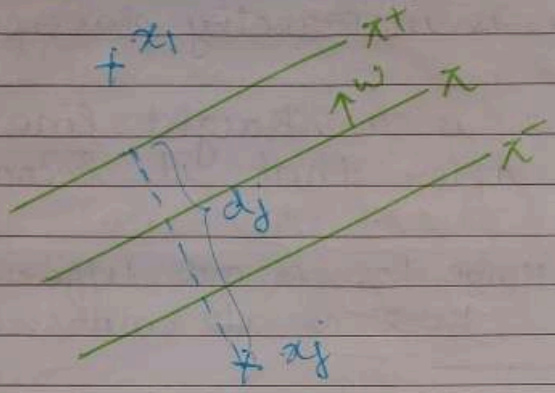
∴ Hinge Loss : $z_i \geq 1$; hinge loss = 0
$z_i < 1$; ——— = $1 - z_i$

One way to simplify the def
Hinge Loss : $\max(0, 1 - z_i)$

Case 1 : $z_i \geq 1$;
$1 - z_i$ is negative value.
∴ $\max(0, 1 - z_i) = 0$

Case 2 : $z_i < 1$
∴ $(1 - z_i) > 0$
∴ $\max(0, 1 - z_i) = (1 - z_i)$

Geometric Intuition



x_i is correctly classified : $\xi_i = 0$
x_j is incorrectly
Let the distance of x_j from $\pi^+ = d$

$$d_j = 1 - y_j(w^T x_j + b)$$

value would be -ve as x_j is on opp side of w direction

d_j is distance of misclassified positive pt

$$d_j = 1 - \underbrace{y_j}_{+ve}\underbrace{(w^T x_j + b)}_{-ve} = x_j(1 - z_j)$$

$\therefore$ d_j would be +ve

We know
ξ_j = dist to x_j from $\pi^+ = d_j = 1 - z_j$
$\therefore$ $\xi_j = 1 - z_j$

$\therefore$ $\xi_j = 0$ → correctly classified
$\therefore$ $\xi_j = 1 - z_j$ → misclassified

$\therefore \underline{\text{Max}(0, 1-z_i) = \xi_i}$

Soft SVM formula^n :

(I) $\underset{w,b}{\text{Min}} \left[\frac{\|w\|}{2} + C \sum_{i=1}^{n} \xi_i \right]$ such that $(1 - y_i(w^T x_i + b)) \geq \xi_i$, $\xi_i \geq 0$.

Loss Min formula^n

(II) $\underset{w,b}{\text{Min}} \sum_{i=1}^{n} \text{max}\left(0, (1 - y_i(w^T x_i + b))\right) + \lambda \|w\|^2$

Lets prove that I and II are same

1. $\frac{\|w\|}{2}$ is same as $\|w\|^2$

   Since $\|w\| > 0$, Minimizing $\|w\|$ is similar to Minimizing $\|w\|^2$
   We can put constant to $\|w\| \Rightarrow \frac{\|w\|}{2}$ or remove it.

2. We have hyperparameter C & λ
   C → multiplied to Loss
   λ → ——— Regularizer

Soft SVM	Loss Min	Behaviour
As C ↑	As λ ↓	Overfit
As C ↓	As λ ↑	Underfit

Typically we call it $C \rightarrow$ Multiplied with Loss func^n & $\lambda \rightarrow$ ~~~~~ with Regularizer. And the behaviour is inverted.

36.5 Dual form of SVM formulation

$$\min_{w,b} \frac{1}{2}\|w\| + C\sum_{i=1}^{n} \xi_i$$

such that

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

$$\& \ \xi_i \geq 0$$

This formulation is Soft Margin SVM
Also called as Primal formula^n of SVM

This Primal is equivalent to other formula^n:

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

such that $C >= \alpha_i >= 0$

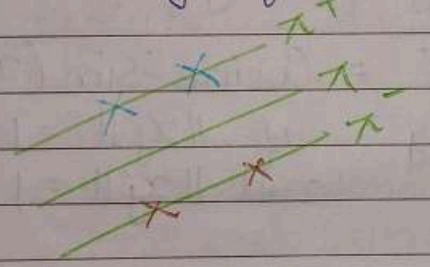
and $\sum_{i=1}^{n} \alpha_i y_j = 0$

→ called Dual form of SVM

⇒ For every x_i, we have corresponding α_i
⇒ $x_i^T x_j$ only occur in product form.
⇒ x_i occur only in form of $x_i^T x_j$
⇒ We want that given query point, we want to compute $(w^T x_q + b)$.
If its positive / negative determines the class label.

$$\therefore f(x_q) = \underbrace{\sum_{i=1}^{n} \alpha_i y_i x_i^T x_q}_{w^T x_q} + b$$

⇒ $\alpha_i > 0$ only for support vectors



$\alpha_i > 0$ for these points

$\alpha_i = 0$ for non support vectors

$$f(x) = \underbrace{\sum_{i=1}^{n} \alpha_i y_i x_i^T x_q}_{\text{Support Vec: } \alpha_i > 0} + b$$

Non ——— : $\alpha_i = 0$; whole term becomes 0

∴ To compute $f(x)$; only thing that matters are Support Vector
Hence the name Support Vector Machine

The Non Support Vector dont really change the class

Hence, the formulation

$$\underset{\alpha_i}{\text{Max}} \left\{ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \right\}$$

such that $\alpha_i \geq 0 \longrightarrow \alpha_i = 0$ for Non SV
$\alpha_i > 0$ for S.V.

and $\sum_{i=1}^{n} \alpha_i y_i = 0$

We also said that x_i occur in form of $x_i^T x_j$. Why is that imp?

$x_i^T x_j = \underbrace{x_i \cdot x_j}_{\text{Dot prod}} = \text{Cosine-Sim}(x_i, x_j)$ if $\|x_j\| = 1$ and $\|x_j\| = 1$

We get the similarity between x_i and x_j only in Dual form
Often times the similarity is written as $K(x_i, x_j)$
where K = Kernel function

$K(x_i, x_j)$ = Similarity function between x_i and x_j

36.6 Kernel Trick

Dual formulation of SVM

$$\max_{\alpha} \left[\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boxed{x_i^T x_j} \right]$$

$\rightarrow sim(x_i, x_j)$

such that $\sum_{i=1}^{n} \alpha_i y_j = 0$

and $\alpha_i \geq 0$

Also called $K(x_i, x_j)$ Kernel function

The most imp idea in SVM is the Kernel Trick

If we use $x_i^T x_j \rightarrow$ linear SVM

If we replace it by $K(x_i, x_j) \rightarrow$ Kernel SVM

Linear SVM :- We try to find Margin Max hyperplane in space of x_i

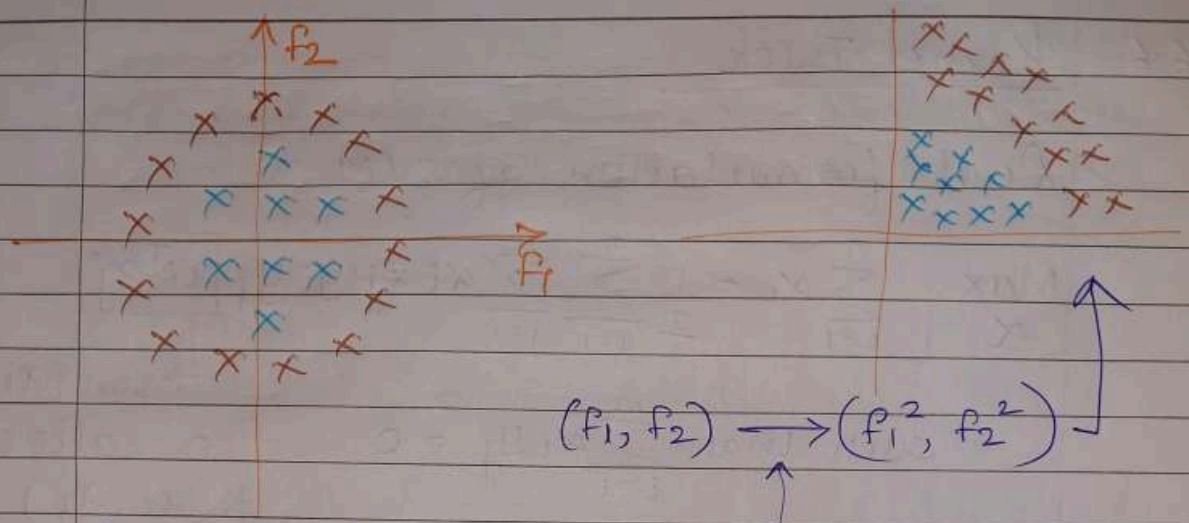
Logistic Regression: Minimize Logistic loss in space of x_i

World changing idea in area of SVM is Kernel function

Using linear SVM, we find a hyperplane.

But it cannot solve all kinds of datapoints

340



In this dataset

Linear SVM : fail
Logistic Regression : fail

Logistic Regression + Feature Transformaⁿ = Succeed

Kernel SVM : Will find right Kernel = Succeed.

There is a difference between feature transformaⁿ & Kernalizaⁿ.

Hence,
Kernalizaⁿ ⇒ Enables SVM handle Non linearly separable dataset.

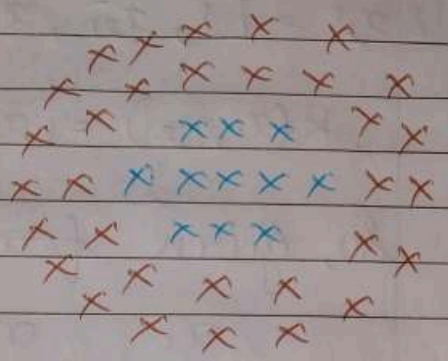
36.7 # POLYNOMIAL KERNEL

In Logistic Regression, we transformed the features into

$(f_1, f_2) \longrightarrow (f_1^2, f_2^2)$

Feature Transformation ↓

Logistic Reg



How does kernelization Handles this?

Given two datapoints x_1 and x_2, the polynomial Kernel is defined as

$K(x_1, x_2) = (x_1^T x_2 + c)^d \Rightarrow$ c and d are constants

Eg: $K(x_1, x_2) = (1 + x_1^T x_2)^2 \Rightarrow c=1$ and $d=2$

Consider $x_1 = \langle x_{11}, x_{12} \rangle$ and $x_2 = \langle x_{21}, x_{22} \rangle$ $\Rightarrow d=2$

$K(x_1, x_2) = (1 + x_{11} \cdot x_{21} + x_{12} \cdot x_{22})^2$

$= 1 + x_{11}^2 x_{21}^2 + x_{12}^2 x_{22}^2$
$+ 2 \cdot x_{11} x_{21} x_{12} x_{22}$
$+ 2 \cdot x_{11} \cdot x_{21}$
$+ 2 \cdot x_{12} \cdot x_{22}$

We can split this into two vectors

Let $d' = 6$

$x_1' = [1, x_{11}^2, x_{12}^2, \sqrt{2}x_{11}x_{12}, \sqrt{2}x_{11}, \sqrt{2}x_{12}]$

& $x_2' = [1, x_{21}^2, x_{22}^2, \sqrt{2}x_{21}x_{22}, \sqrt{2}x_{21}, \sqrt{2}x_{22}]$

$\therefore K(x_1, x_2) = (x_1')^T (x_2')$

So, given ~~features~~ datapoints

$x_1 \quad x_2$

$\Downarrow$ feature transformation

$x_1' \quad x_2'$

and then we can perform $(x_1')^T x_2'$

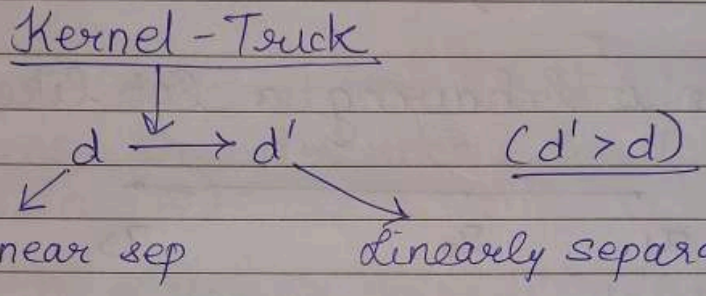
What Kerneliza^n does internally is

$d \xrightarrow[\text{Transformation}]{\text{Feature}} d'$
(internally)

Here, in our eg. $d = 2$ and $d' = 6$
such that $d' > d$

So, we went from $d = 2$ to $d = 6$ and in 6-D data, we have a quadratic term equivalent to $(f_1, f_2) \Rightarrow (f_1^2, f_2^2)$

There is a theorem called Mercer's Theorem:
It says that Kernel trick takes d-dim data which is not linearly separable and transforms it into d'-dim which is linearly separable.



The explicit feature transformation in Logistic Regression is replaced by Kernelization trick (implicit feature Xforma^n)

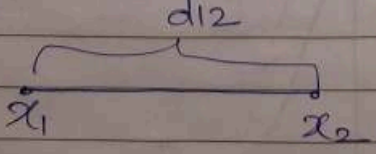
Challenge: What is the right Kernel.

36-8 RBF (Radial Basis func^n) Kernel

SVM: Most popular / general-purpose RBF

Given two datapoints x_1 and x_2;

$$K_{RBF}(x_1, x_2) = \exp\left\{\frac{-\|x_1 - x_2\|^2}{2\sigma^2}\right\}$$



$$d_{12}^2 = \|x_1 - x_2\|^2$$

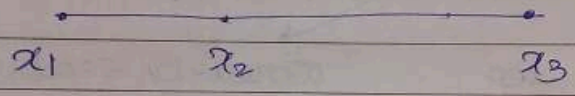
σ = Hyperparameter

$$K(x_1, x_2) = \exp\left(\frac{-d_{12}^2}{2\sigma^2}\right)$$

where $d_{12} = \|x_1 - x_2\|^2$

As $d_{12} \uparrow$; $K(x_1, x_2) \downarrow$

This is behaving a lot like Similarity



① $K(x_1, x_2) > K(x_1, x_3)$

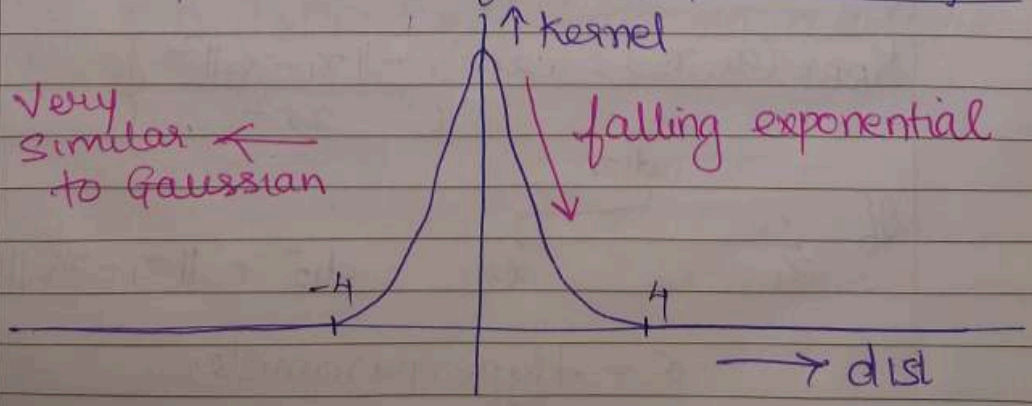
② Impact of σ (for a constant d_{12})

$\sigma = 1$; $K(x_1, x_2) = \exp\left(\frac{-d_{12}^2}{2}\right)$

Case1 : If $d_{12} = 0$

$\therefore K(x_1, x_2) = \exp(0) = 1$

As $d_{12} \uparrow$ in positive / negative direction the ~~exp~~ K value falls exponentially



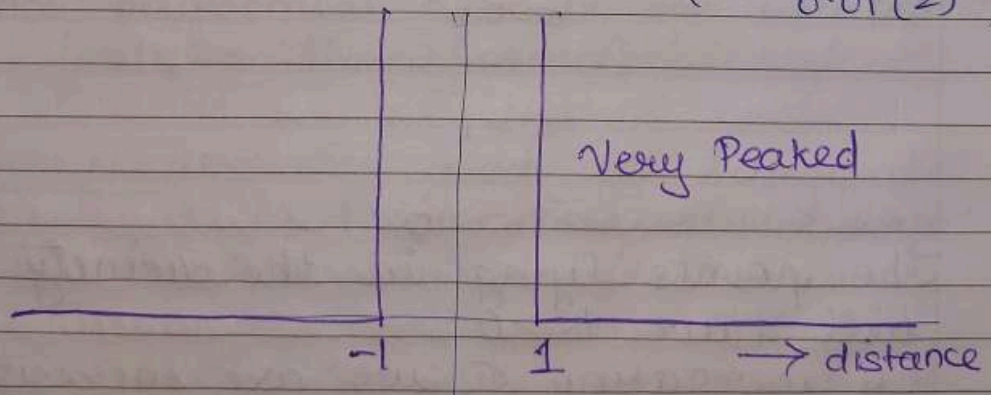
Kernel value is equivalent to similarity
As the distance reduces, the similarity value increases.

When $d_{12} = 0.5$, we get high Kernel value.
When d_{12} is high, the similarity reduces down to 0.

If $K(x_1, x_2) = 0$; it means similarity = 0,
It means the distance should be high ($d_{12} = \infty$)

Intuitively; Kernal ⟶ Similarity
distance ⟶ Dissimilarity

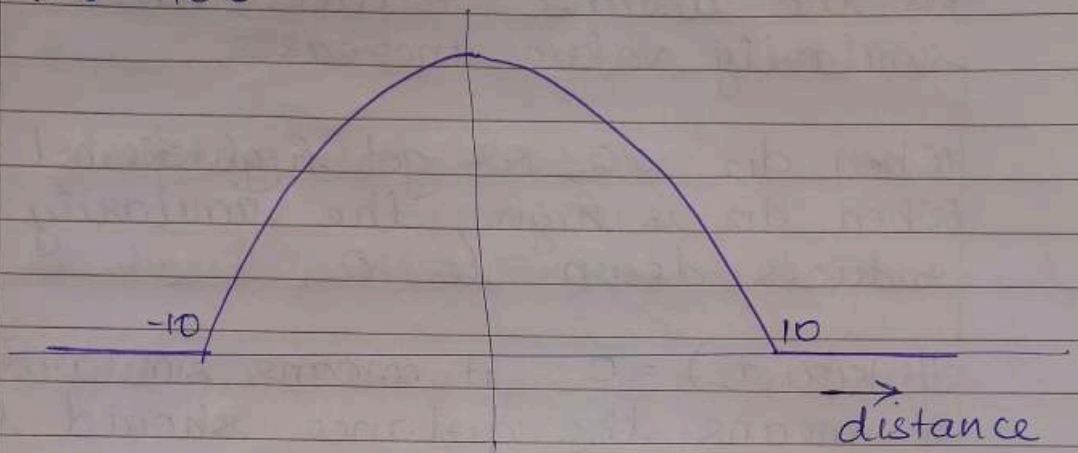
$\sigma = 0.1$
$\therefore \sigma^2 = 0.01$ $\therefore K(x_1, x_2) = \exp\left(\frac{-dist(x_1, x_2)}{0.01(2)}\right)$



When $d > 1$; $K = 0$

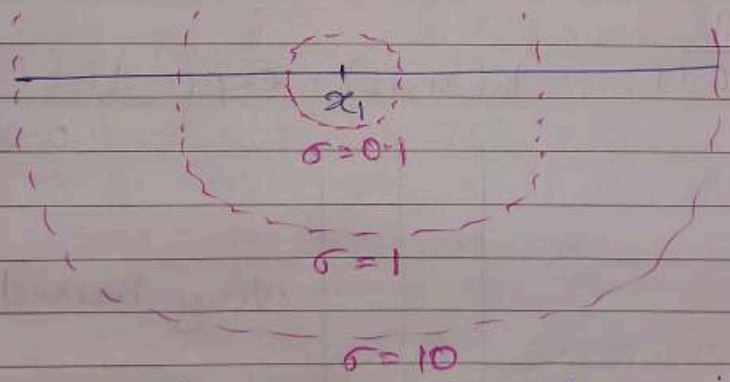
$\sigma = 10$

$\therefore \sigma^2 = 100$



When $d_{12} > 10$; $K = 0$

When $\sigma \uparrow$, the vicinity of points having $K > 0$ increases



The points lying in the vicinity will have $K > 0$.

By increasing σ, we are increasing vicinity

Similarity and distance are opposite to each other

If we observe, this is similar to K-NN

There as well, we take a large value of K where we want to increase the distance of neighbourhood.
The K reduces if the neighbourhood is closer.

$\sigma = 0.1 \Rightarrow$ Very small K in K-NN
$\sigma = 10 \Rightarrow$ Large K in K-NN

K-NN and RBF-Kernel

$\sigma \uparrow$ in RBF = $K \uparrow$ in K-NN

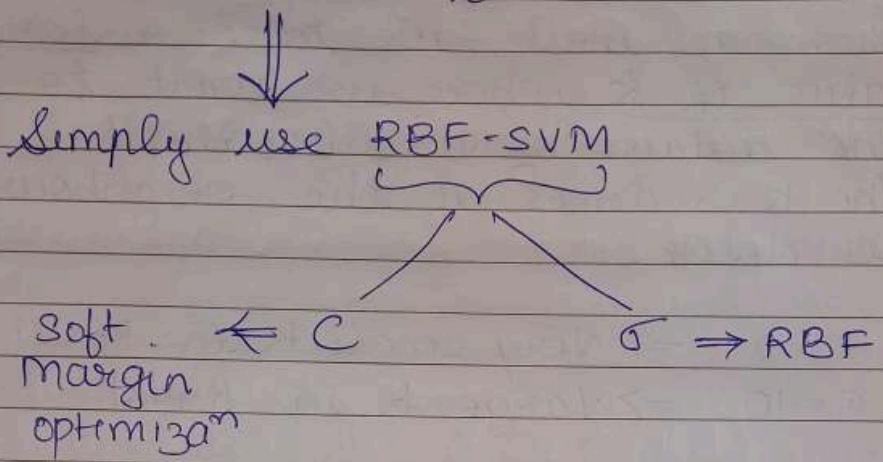
Interview Q: ① How is SVM related to K-NN?
② How is SVM related to Logistic Reg?
③ How polynomeal kernels are related to feature transformaⁿˢ

K-NN are not good for large dataset since we had to store all points for Locality Sensetive Hashing.

RBF-SVM: We just need to have the support vectors and corresponding α_i

K-NN: Lot of time: Time Compl: $O(nd)$
Hence RBF-SVM is nice approximaⁿ to K-NN

Hence, if we dont know the "Best" Kernel to use



We have to do research on both C & σ i.e. we have to do Grid Search OR Random Search on both C & σ.

36-9 Domain Specific Kernels

Kernel trick is similar to the Feature Transformation.

And we know that feature Xformaⁿ is domain specific and its the Art. There are lot of domain specific kernels in ML.

→ String Kernels → Text Classificaⁿ
→ Graph →
If we dont have domain specific kernels, we can use RBF

If we want to classify, we can check for any available domain specific kernels.

Eg:
Amazon food reviews → String classificaⁿ

Feature Transformaⁿ : hard Part in ML → Partially replaced by finding the right Kernel

If we dont find any, RBF can be used for Backlog.

36.10 Train & Run Time Complexity of SVM

We can train SVM using S.G.D but there are better algorithms → Sequential Minimal Optimizaⁿ (SMO)

LibSvm : Best libraries of training SVM.

Training Time Complexity : ~ $O(n^2)$ for Kernel SVM

There was a more optimized algo published around 2007 :- $O(nd^2)$ if $d<n$

If n is large → $O(n^2)$ is very very large & hence most datascientists donot use SVM when n is large.

n is Large → Internet applicaⁿ (Billions of data)

__/__/____

Run Time Complexity

$$f(x_q) = \sum_{i=1}^{n} \alpha_i y_i K(x_i, x_q) + b$$

$\alpha_i = 0$; for non support vectors

Hence the runtime complexity depends on the number of Support Vectors.

If #S.V = K; then
RunTime Complexity : $O(kd)$

The number of S.V. could be from 1 to n. i.e $1 \leq K \leq n$

Imagine if K is high.
Lets say $n = 100K$ and $K = 1000$

∴ RunTime Compl : $O(1000d) \Rightarrow$ high

Comparing to Logistic Regression
RunTime Compl : $O(d)$
Now $d \ll K$
Hence Logistic Regression has much smaller RunTime Complexity than SVM

And we dont have any means to control the Support Vectors

36.11 **nu-SVM**

We had studied till now was **C-SVM**

C-SVM → Original Formula^n ; $C \geq 0$
⇓
Hyperparameter

There is an alternative formula^n of SVM
nu-SVM : Hyperparameter ⇒ nu

$\therefore 0 \leq nu \leq 1$

and nu ≥ fraction of errors
nu ≤ fraction of support Vectors.

Support if I want to build SVM on D_{Train} and
① If I want less than 10% error ⇒ nu = 0.1
② ⇒ 1% ⇒ nu = 0.01

When we say
nu = 0.01 ⇒ % of errors = 1% (Max)
⇒ # S.V.S ≥ 1% of n

Hence, in a way, its very useful since its controlling the number of errors

But for RunTime Complexity, we want fewer SVS ; unfortunately nu-SVM doesnt control that.

But, nu-SVM is one way to get
→ no of Support Vectors
→ no. of Misclassified points in D_{Train}

36.12 SVM - Regression

In classifica^n; $y_i \in \{+1, -1\}$

When SVM Regression is used,
Math. formula^n:

Linear form of SVR

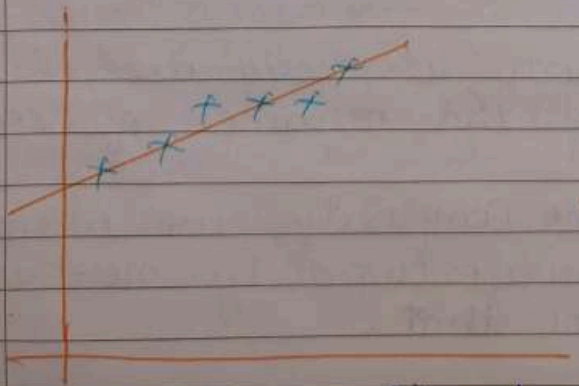
$$\min_{w} \frac{1}{2}\|w\|^2$$

such that $y_i - (w^T x_i + b) \leq \epsilon$
and $(w^T x_i + b) - y_i \leq \epsilon \quad \forall i$
and $\epsilon \geq 0$ → Hyperparameter

We can kernelize the formula^n to fit any shape.

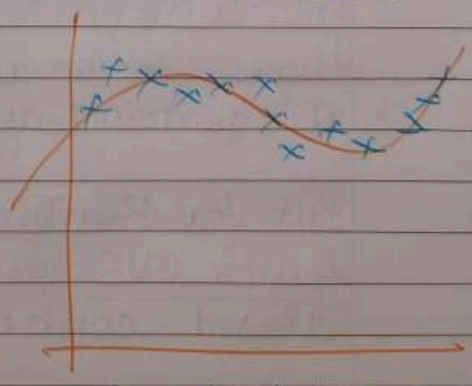
Because

LR - SVR



Will be able to fit only Planes

Kernel SVR

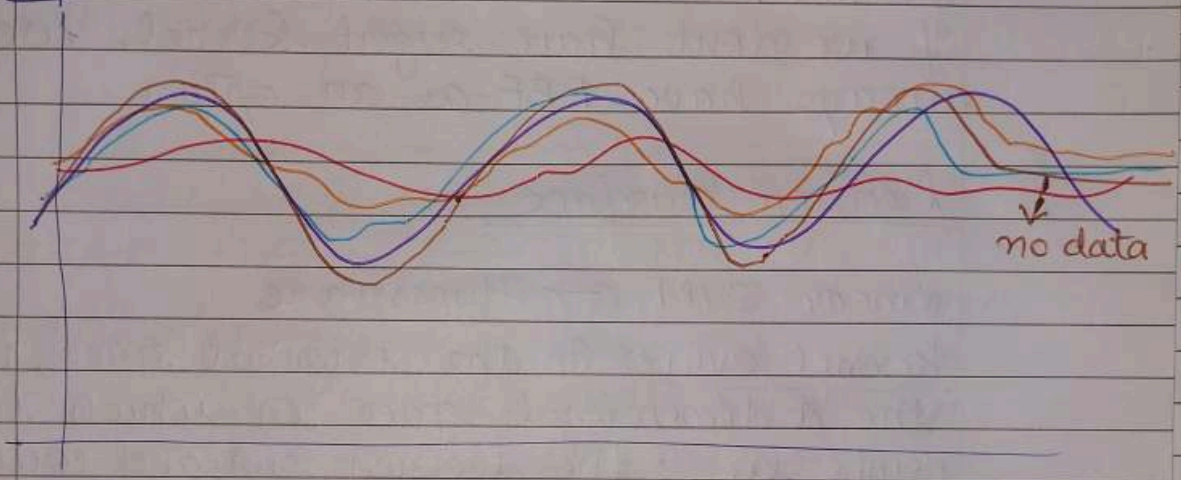


would fit all shapes
ϵ: hyperparam

If ϵ is ↓ ; errors are low on D_{Train}
; overfitting ↑

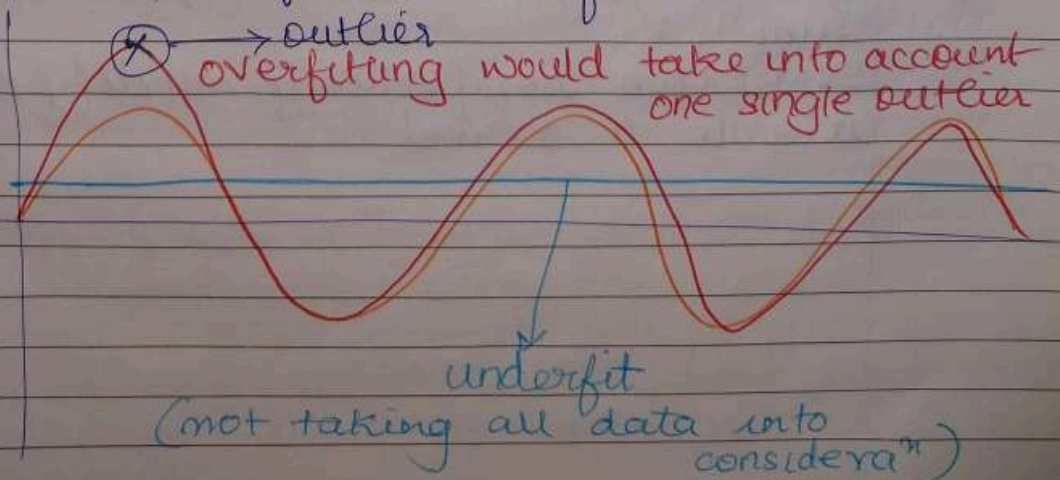
If ϵ is ↑ ; errors are high on D_{Train}
; underfit ↑

Eg:



True Curve
$\epsilon = 0.01$ → overfit
$\epsilon = 0.1$
$\epsilon = 1.0$
$\epsilon = 2.0$ → underfit

In SVM, overfit & underfit means



36.13 Cases : (SVM)

1 Feature Engineering and feature Transformations :
Replaced by Kernel design,
Its all about finding the right Kernel.
If we dont have right kernel, we always have RBF as an option

2 Decision Surface :
Linear SVM :→ Hyperplane
Kernel SVM : In the original space, i.e. the d-dimensional space in which the points lie ; the decision surface could be a non linear surface.
Using Kernel trick, we can convert $d \rightarrow d'$ dimensions ; by which the decision surface changes from Non Linear → Linear.

(3) If we are given Similarity / distance func^n :
SVMs can comfortably modify / convert it into a Kernel.

(4) Big Challenge with SVM is about Kernel Interpretability and feature importance

→ Now way to get F.I. directly from SVM
  → we can get using Forward Feature selection
→ FI and interpretability can be thought of exactly like Logistic Regression

(5) Outliers

→ have very little impact on model
→ Note that if we using RBF with small σ; it behaves similar to K-NN with small K.
Since K-NN with small K can be impacted with outliers.
SVN with σ & small can also be impacted
→ In general, impact is very small as only Support Vectors matter.

(6) Bias Variance Tradeoff

As C ↑ ; we would Overfit ⇒ High Variance
C ↓ ; ——— underfit ⇒ High Bias
We have Grid Search / Random Search techniques to find optimal C.

In RBF-SVM we have two Hyperparameters C and σ
↑ (Loss) ↑ (RBF)

(7) What about Large d

→ Very Good for SVM

(8) Best Cases for SVM

① If we have the right Kernel

(9) Worst Case

① 'n' is large.
→ Train Time is high
→ Not used in Internet Based App

② 'k' is large
→ Means #SVs is large
→ Cannot have Low latency
→ Time to compute $f(x_q)$ is large

Typically when 'n' is high, i.e. millions / Billions, people end up using Logistic Regression with feature transforms.

357

37.1 Decision Trees

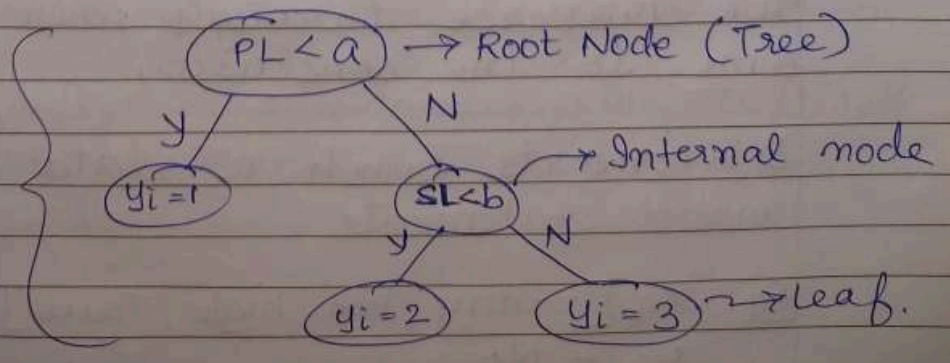
KNN → Instance Based Method ⇒ We are given data points and we coork on that
N.B. → Probabilistic Method
Log Regression
Linear Reg.
SVM → Kernel Based.
} inspired by ideas of Geometry, hyperplane

Thus, we have seen 3 types of models.
4th Model: Decision Trees.
↳ nested if...else condition

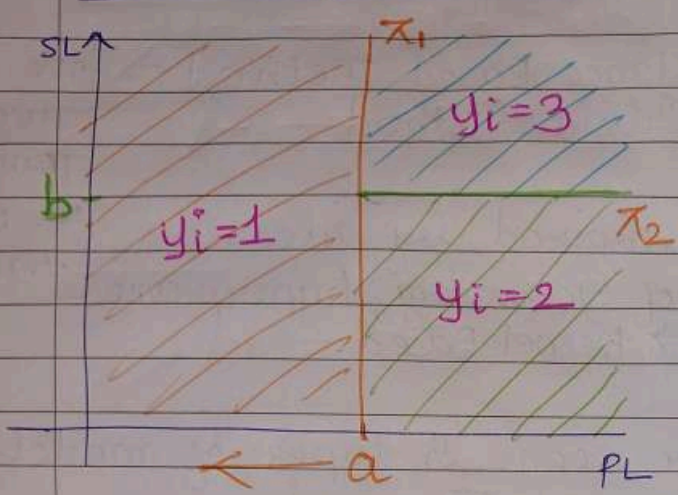
Eg: EDA: Iris dataset
where $y_i = \{1,2,3\} \Rightarrow$ {setosa, Virginica, Versicolor}
and feature = {SL, SW, PL, PW}

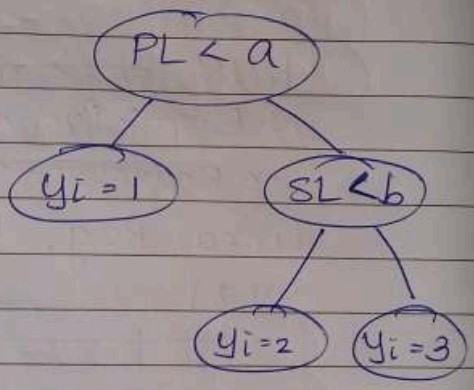
We formulated as:
if (PL < 5) } Simple condition
then $y_i = 1$ }

If my model can be written as nested if else conditions, then we can draw the decision Tree



Geometric Intuition





There would be a π for every decision

V. Imp: All hyperplanes are axes parallel in a D-T.
(parallel to either X/Y axis)

D-T: Set of axis parallel hyperplanes which durde the space into hyperaiboids

37-2 Sample D-T.

Eg: Play Tennis
All features are categorical and $y_i \in$ Binary
We traverse through path to come to the conclusion

D-T converts bunch of data into insert statements

Eg: x_q [sunny, hot, high, True]
$\therefore y_q$ = No

37.3 Building D.T. (Entropy)

Entropy

→ concept of information Theory, Physics

Suppose we are given r.v. y which can take 'k' value

Entropy:

$$H(y) = -\sum_{i=1}^{k} P(y_i) \log_b (P(y_i))$$

where $b = 2$ or

$b = e = 2.718$

Considering the Tennis example.

$$H(y) = -\frac{9}{14} \log\left(\frac{9}{14}\right) - \frac{5}{14} \log\left(\frac{5}{14}\right) = 0.94$$

$P(y+)$ $P(y-)$

= No of +ve pts by total pts

= % of +ve pts = % of -ve points

Properties

Assume that y can take two values $(y+, y-)$

Two categories only.

Two classes.

Case 1:

D → $y+ \rightarrow 99\%$, $y- \rightarrow 1\%$

$H(Y) = -0.99 \log(0.99) - 0.01 \log(0.01)$

$= 0.0801$

Case 2:

D → $y+ \rightarrow 50\%$, $y- \rightarrow 50\%$

$H(Y) = -0.5 \log(0.5) - 0.5 \log(0.5)$

$= 1$

Case 1:

D → $y+ \rightarrow 0\%$, $y- \rightarrow 100\%$

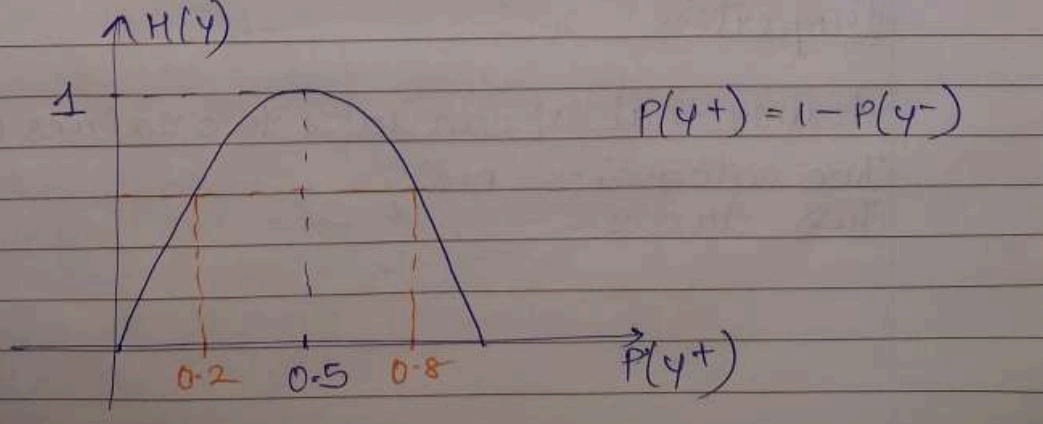
$H(Y) = 0$

From this we understand

⇒ If the values of class are equiprobable i.e. case 2; then we get Max Entropy.

⇒ If one class dominates over other, then entropy is 0

Hence Plot looks like:



Curve is Symetric
$\therefore H(0.2) = H(0.8)$

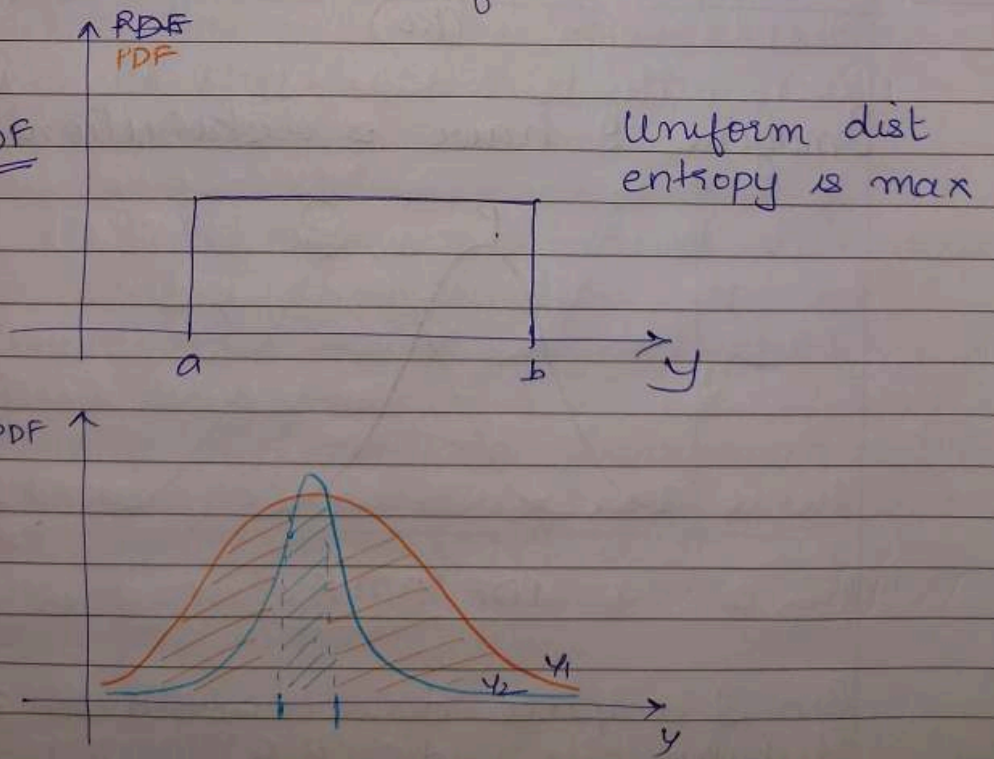
$H(0.2) = -0.2 \log(0.2) - 0.8 \log(0.8)$
$H(0.8) = -0.8 \log(0.8) - 0.2 \log(0.2)$

For multi-class r.v.
$Y \rightarrow y_1, y_2 \ldots y_K$

if its equiprobable $\rightarrow$ entropy is max

if $y_1 \rightarrow$ most probable. } entropy is min
$y_2, y_3 \ldots y_K \rightarrow 0$ }

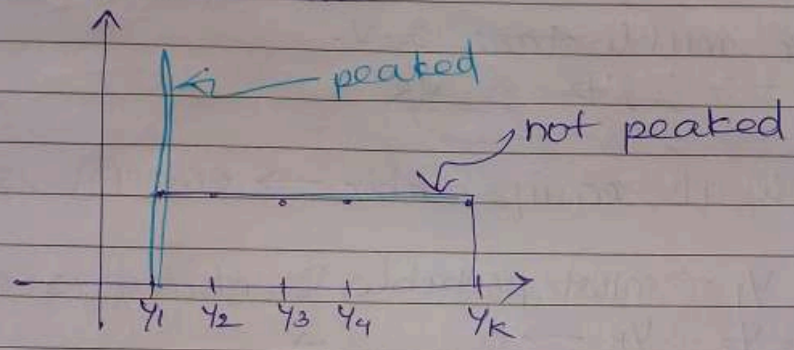
Lets draw PDF of the same



$H(y_2) < H(y_1)$

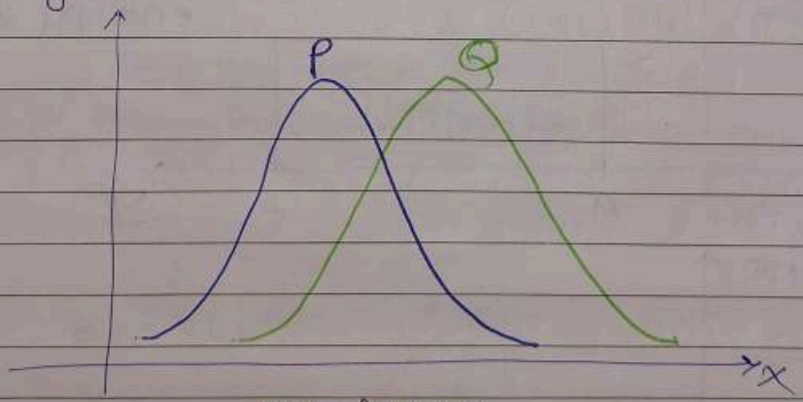
The more peaked the distribution is; ~~less~~ we would see more points in ▨ region. We would see less points in spread region ▨

For discrete case



37.4 Kullback-Leibler divergence (KL)

Imagine I have 2 distributions

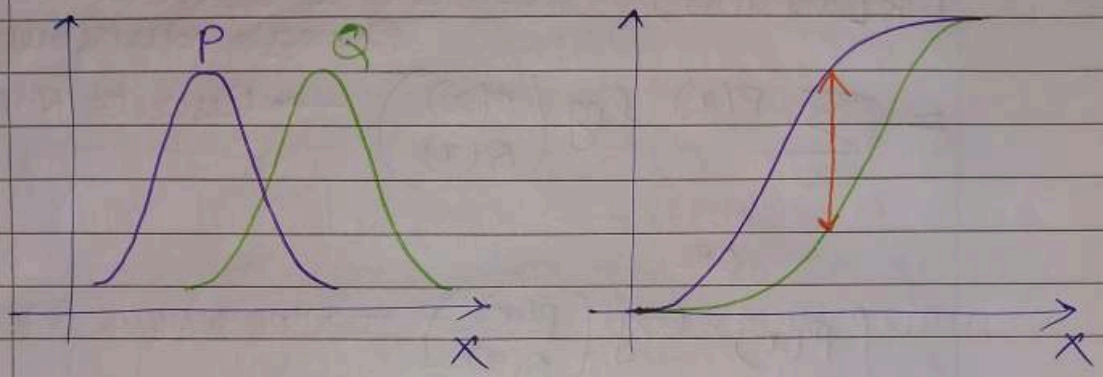


PDF/PMF

Can I compute distance between 2 distributions : dist(P, Q)

One Idea: K-S-Statistics

We have already studied this.
It gives the Supremum or the maximum gap between the CDF of two distributions



K-S Statistics $= \sup(|P'(x) - Q'(x)|)$

The more closer (similar) two func^{ns}/ distributions are, the more overlapping would be their CDF and the gap will be smaller.

In all ML algo we studied till now, we require the function to be differentiable.

We cannot differentiate Supremum

$\sup(|P'(x) - Q'(x)|)$

differentiable

Not ~

The moment we cannot differential K-S Stat, we cannot use it as Loss func^{n} in Optimiza^{n} Based Method.

Hence, I would need a distance measure which is differentiable

Info-Theoretic Measure

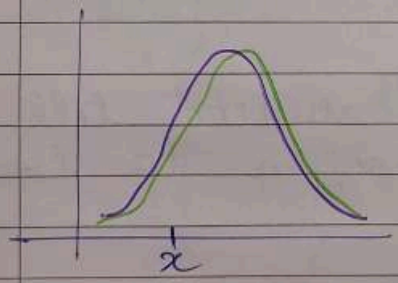
$D_{KL}(P \| Q)$ → KL distance between two Probab distributions

$= \sum_{x} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$ → Discrete R.V

OR

$\int P(x) \log\left(\frac{P(x)}{Q(x)}\right)$ → continuous R.V

Case 1:
Distributions P and Q are very similar



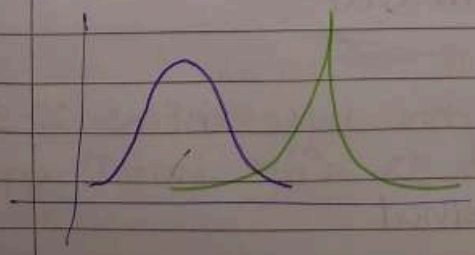
So, $P(x)$ would be very similar to $Q(x)$

$\therefore \frac{P(x)}{Q(x)} \approx 1$

$\therefore$ log would be 0

Hence D_{KL} would be very small.

Case 2: P and Q are very different



D_{KL} is very Large

__/__/__

Why there is log ?

KL divergence is also called Relative Entropy

$$D_{KL}(P||Q) = \sum_x P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

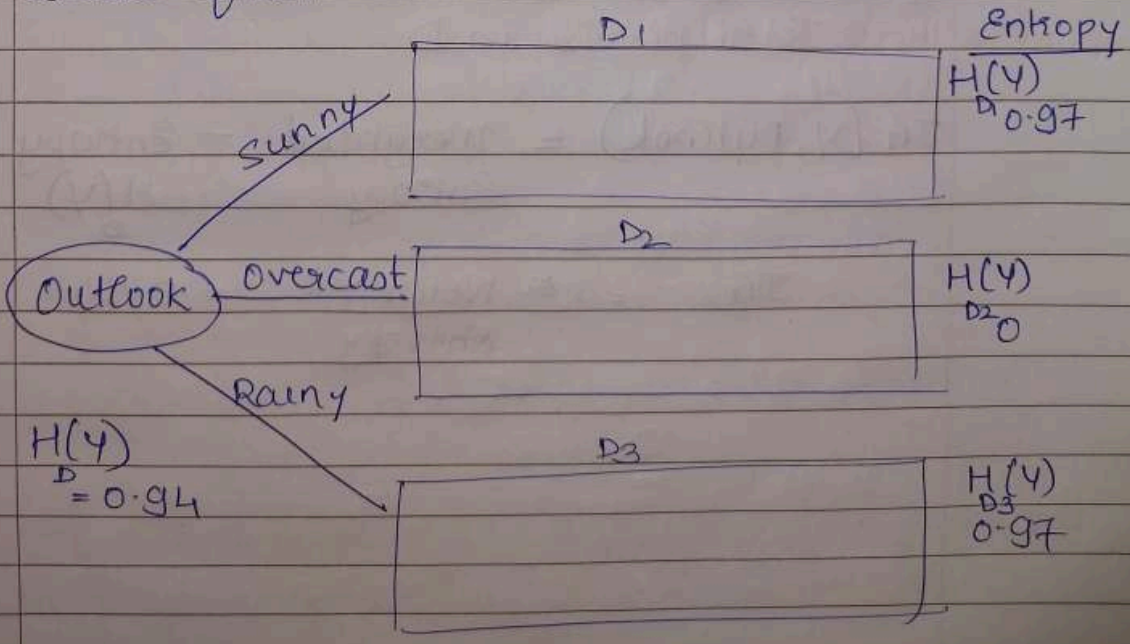
$$= \sum_x P(x) \log(P(x)) - \sum_x P(x) \log(Q(x))$$

This term looks very similar to Entropy. Hence we have the log term

Now the whole D_{KL} term is differentiable

37.5 Information Gain

Consider the Tennis dataset and DT. built from that



① We calculated the Entropy of entire dataset with respect to "Outlook". $H(Y) = 0.94$

② Then we ~~calculated~~ split the dataset D into 3 datasets as per the class of Outlook

③ We calculated the Entropy of each sub dataset.

④ Calculate the average entropy. (also called Weighted Entropy)

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

(D_1, D_2, D_3)

Giving the weight as per the number of points

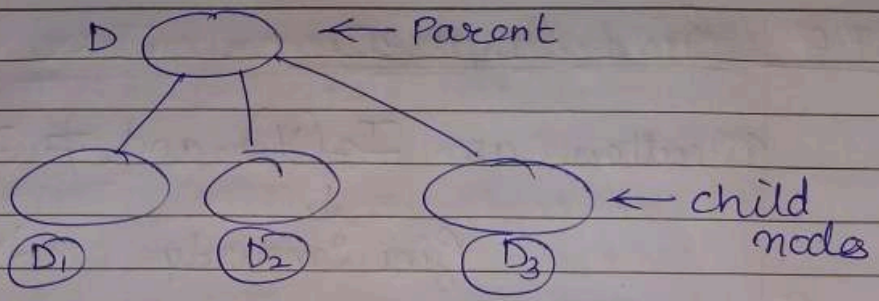
Entropy of D_1

$5 \rightarrow$ No. of points in D_1

$14 \rightarrow$ Total points in D

$$IG(Y, \text{Outlook}) = \text{Weighted Entropy} - \text{Entropy of } H_D(Y)$$

$$IG = \text{Weighted Entropy} - 0.94$$



$$\left(-H_D(Y)\right) + \left(\frac{|D_1|}{|D|} * H_{D_1}(Y) + \frac{|D_2|}{|D|} * H_{D_2}(Y) + \frac{|D_3|}{|D|} H_{D_3}(Y)\right)$$

← Entropy of Y → ← Weighted Entropy →

∴ ~~IG (Y~~

∴ To sum it up.

If r.v. $Y \xrightarrow[\text{broken into}]{\text{variable}} Y_1, Y_2 \ldots Y_k$.

$$\therefore IG(Y, \text{Variable}) = \left[\sum_{i=1}^{k} \frac{|D_i|}{|D|} * H_{D_i}(Y)\right] - H_D(Y)$$

37.6 Gini Impurity

Written as $I_G(Y)$ and $\neq$ $IG(Y)$

$I_G(Y)$ → Gini Impurity

$IG(Y)$ → Info. Gain

For $y \rightarrow y_1, y_2, \ldots y_k$.

$$I_G(y) = 1 - \sum_{i=1}^{k} (P(y_i))^2$$

Assume that a r.v. y takes two classes $y+$ and $y-$

Case 1:

$P(y+) = 0.5$

$P(y-) = 0.5$

$I_G(y) = 1 - (0.25 + 0.25)$

$= 0.5$

$H(y) = 1$

Case 2:

$P(y+) = 1$

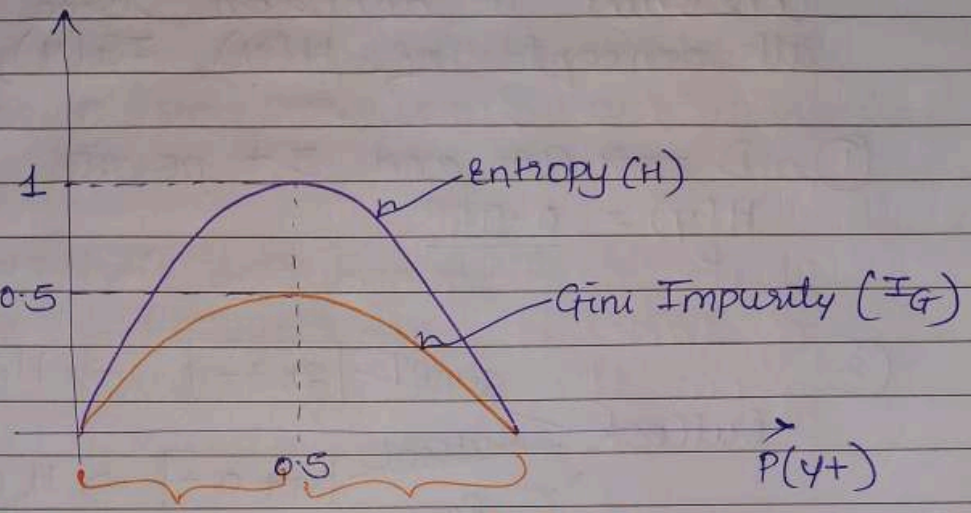
$P(y-) = 0$

$I_G(y) = 1 - (1 + 0)$

$= 0$

$H(y) = 0$

Assume we have a 2-category case (Y+ and Y-) ; P(Y+) = 1 - P(Y-)



As P(Y+) increases, Both H(Y) and $I_G(Y)$ increases at a different rate

As P(Y+) decreases Both H(Y) and $I_G(Y)$ decreases

Q: Why are we studying Entropy?

Consider the formula of Entropy

$$H(Y) = -P(Y+)\log(P(Y+)) - P(Y-)\log(P(Y-))$$

In this case, we are computing log which is more time consuming.

$$I_G(Y) = 1 - (P(Y+)^2 + P(Y-)^2)$$

We are not computing log.
Hence $I_G(Y)$ is more computationally efficient than H(Y).

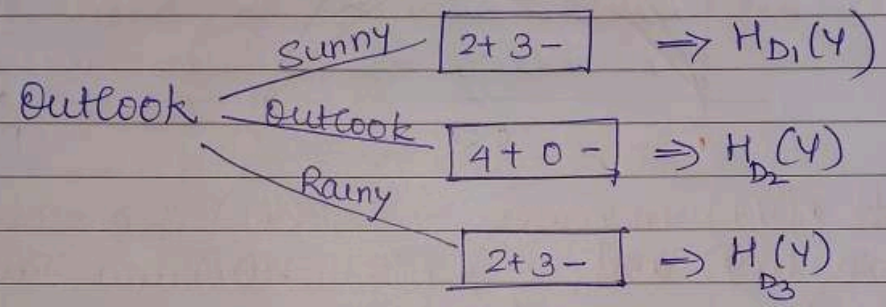
37.7 Building a Decision Tree

Construct a decision tree using all concepts of H(Y), IG(Y), $I_G(Y)$

(1) D ⟶ 9+ and 5+ points

$H_D(Y) = 0.94$

(2)



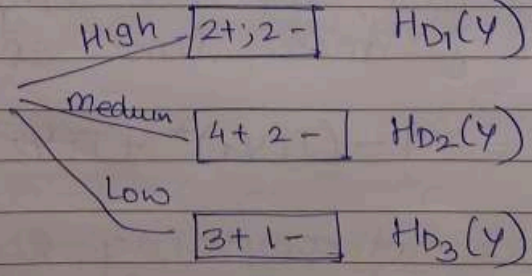
Weighted Entropy = 0.69

Difference of Entropy at this level with entropy at above level

= 0.94 − 0.69

= 0.25 = IG(Y)

(3) Breaking D w.r.t Temperature



And we can find IG at well

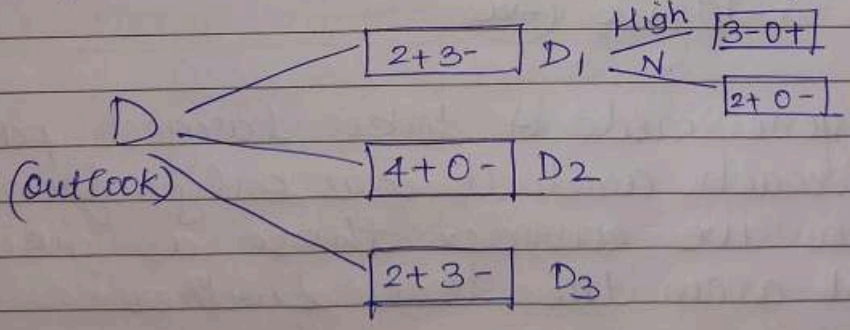
(4) Similar process I can do for Humidity and Windy as well

Hence, we performed following steps:
→ choosing Root Node and we computed details.

IG(Y, Outlook) = 0.25
IG(Y, Temperature) =
IG(Y, Humidity) =
IG(Y, Windy) =
} Pick feature with Max IG

Basically IG = entropy at Parent level − Entropy at child Level

Lets say "outlook" has max entropy
Now, we break dataset into following:



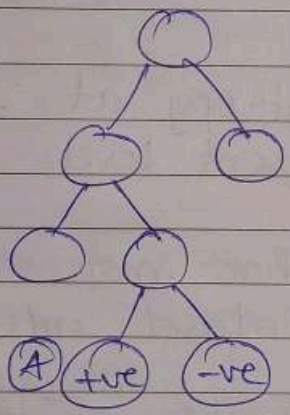
Consider D_2 → It has only +ve points.
Such a node is called PURE Node.
because we stop the computation and we assign y_i = +ve

Hence, we recursively break each node using IG as criteria.

We start with root node and till the time we get a pure node, we continue the traversal

OR

If we can't grow the tree because of lack of points, then we wont grow the node further.



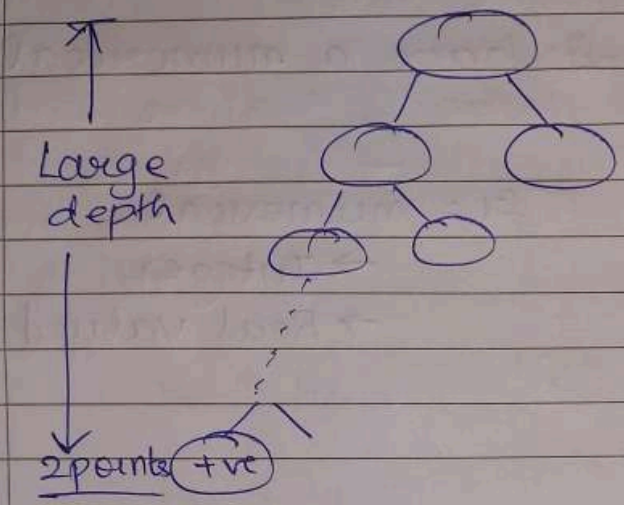
Imagine node A takes largest path to reach and it has only few points, then we consider those as noise and dont grow the tree further.

Hence, we stop computing (growing tree) if:

1. Pure node
2. Few points at a node
3. If we are too deep

373

Imagine $n = 10k$. and we contruct a tree such that



Hence as depth of tree increases, the chances of overfitting also increases

If depth is small ⇒ underfit.

In D.T. :- hyperparameter ⇒ depth of tree

37.8 Splitting Numerical features

We have learnt to

Construct a D.T :- Splitting a Node. → IG

Imp step

We have seen that we use/prefer Gini Impurity over Entropy because:-

→ Its computationally efficient
→ It has almost same behaviour as Entropy

Till now, we have seen discrete R.V or categorical feature

Lets assume I have a numerical feature

f_1	y
2.2	1
2.6	1
3.5	0
3.8	0
4.6	1
5.3	0

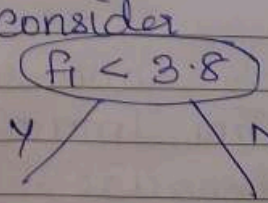
f_1 : numerical
→ Integer
→ Real valued

Here the split based on categorical feature is not intuitively. It is like, every point has a distinct value ⇒ each point would be a category.

Steps we take:

1. Sort the numerical features in ascending order
2. Then build the ~~decion~~ D.T condition as per the values

Eg: $f_1 < 2.2$
$f_1 < 2.6$
$f_1 < 3.5$
$f_1 < 3.8$
$f_1 < 4.6$
$f_i < 5.6$



There are only two values (Yes/No)

Here, I would have atmost n-possible varia^{ns}. like here we had 6 varia^{ns}. For, numerical features; we can have $f_i < T_i$ for each feature

Numerical features:

$f_1 < T_1$ → D_1, D_2

$f_2 < T_2$ → D_1, D_2

$f_3 < T_3$ → D_1, D_2

⋮

$f_n < T_n$ → D_1, D_2

We evaluate all of them. We check all possible splitting criteria and we pick the feature with Max IG.

⇒ Super Time Consuming
⇒ Non Trivial Computation

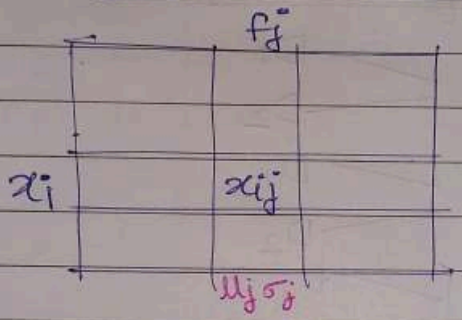
There are hacks/computational tools around it to minimize the complexity

37.9 Feature Standardization

We have learnt that

Logistic Regression, KNN, SVM } Feature Std.

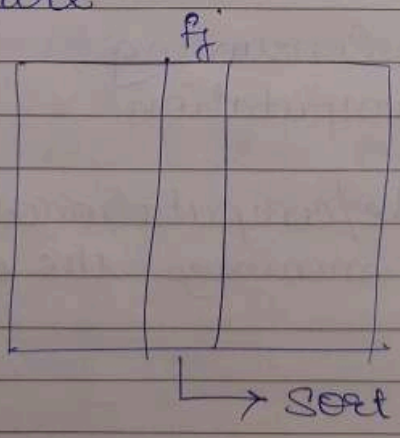
We used this because we were computing some form of distance



$$x'_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}$$

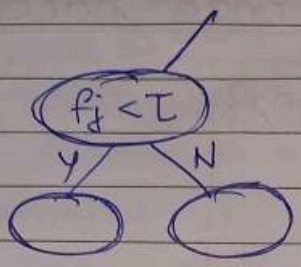
In case of Decision Trees;
It is not a distance based method.

Suppose if I have a real valued feature



We would check for each of values as potential Threshold

Hence, as part of D.T we would traverse the path.

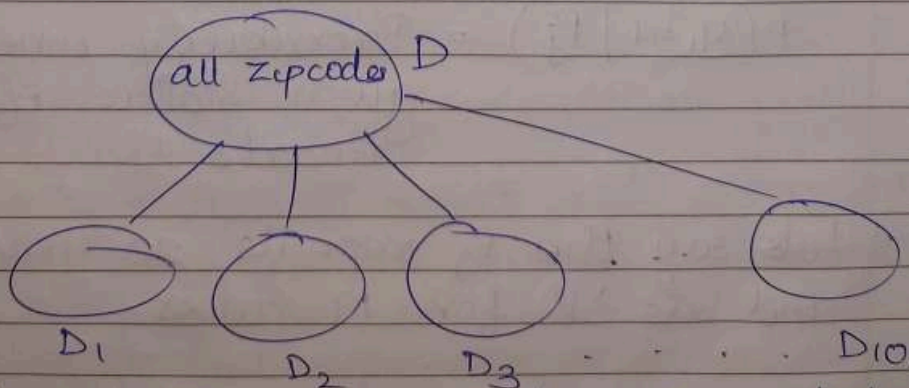


Hence, In D.T. we donot care about actual values of features and hence, we donot need to perform Feature Standardizaⁿ.

37.10 Building a D.T: Categorical features with many possible values.

Eg: Pincode / Zipcode
We have 1000s of categories.
We cannot really compare two zipcodes: Eg: 94087 and 94089.
So, every zipcode is a category.

∴



Lot of categories and every dataset would be extremely small.

This happens a lot in ML.

Hack: Feature Engg hack.

Pincode	$y_i = \{0,1\}$
P_1	
P_2	
P_3	
P_1	
P_2	
⋮	

Hack would be to convert the categorical feature → numerical feature.

① Assign a value y_i to a zipcode.

② Calculate Probability as

$P(y_i=1 \mid P_j)$ = Probability when $y_i=1$ given value of Pincode P_j

Lets say the P_j repeats 20 times and its '1' for 19 times

∴

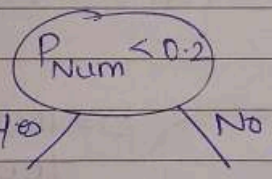
$$P(y_i = 1 | P_j) = \frac{\#\, y_i = 1 \text{ and } P_j}{\#\, P_j}$$

$$P(y_i = 1 | P_j) = \frac{19}{20}$$

So, we would convert

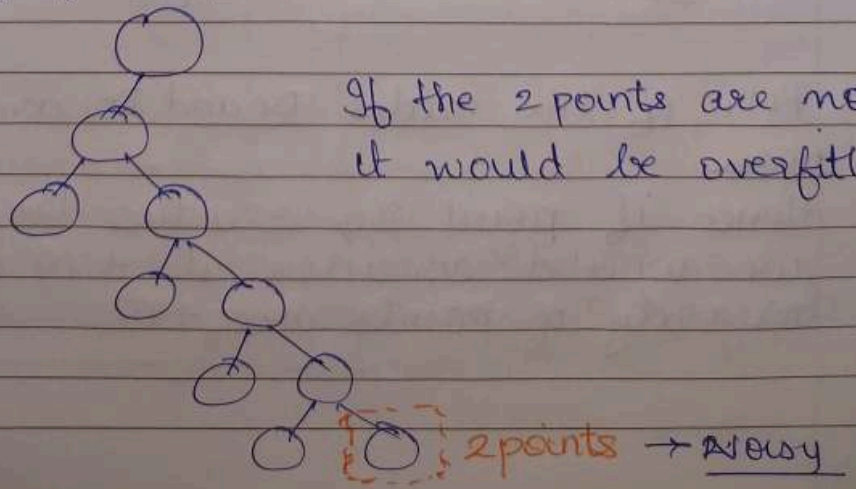
P_j (Pincode) ⟶ Numerical feature (eg: Probability).

Lets say 2 Pincode have same Numerical feature = 0.2.
It would be easier to form a D.T.



37.11 Overfitting and Underfitting

As depth ↑, the possibility of having very few points at leaf node ↑.



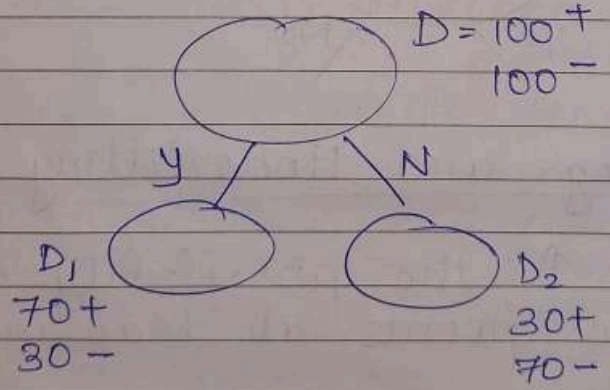
If the 2 points are noisy, it would be overfitting

As depth ↑ ; the interpretability of model ↓
Since there are too many if..else conditions, the interpretability decreases, it becomes hard to understand

```
if ( ) { }
else if ( ) { }
else if ( ) { }
else
  { if ( ) { }
    else { }
  }
```

} Very hard to understand

At the same time, lets say
depth = 1



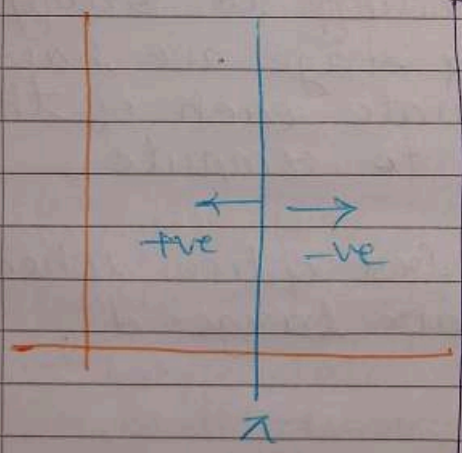
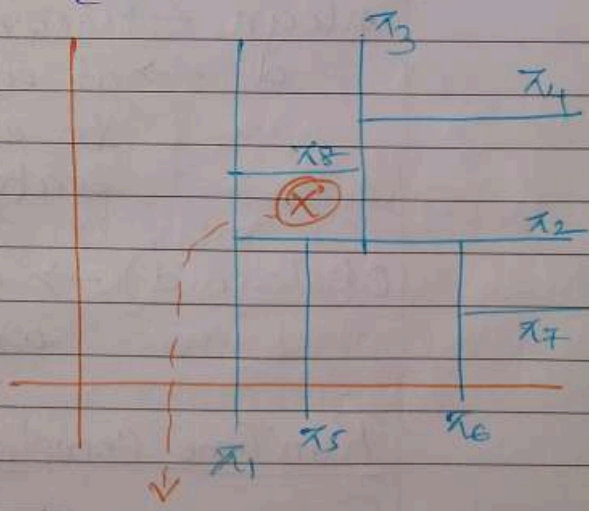
Both of the nodes D_1 and D_2 are not Pure
~~pure~~
Hence if point x_q reaches D_1, we would consider it +ve as majority of points are +ve.

Similarly, if it reaches D_2, it would be classified as -ve.

Hence the tree with depth = 1 is called as Decision Stump.

~~So, we~~ So, when we dont have pure node, the mayority points in leaf nodes are considered.

Hence we want to determine right depth → cross Validaⁿ

Too shallow depth ⇒ underfit	Too deep depth = Overfit
	
Very few Hypercubes & Hyperspaces	It has only one point which may be a noise Too many Hypercubes & Hyperspaces

37.12 Train & Run Time Complexity

Train Time Complexity:

$\sim O(n \log n * d)$

where n = # points in DTrain
d = dimensionality

We may think that the numerical features may take long time to compute as there is Threshold associated with it.
But there are certain algo methods to take care of the same.

$n \log n$ → corresponding to sorting.
d → at every stage we have to evaluate each of the feature to compute.

$O(n \log n\, d)$ → not best option when we have large 'd'.

Run Time Complexity:

We have to save DTree. We convert DTree into nested if... else condition which is very much space efficient

Space Complexity: Num of Internal nodes + Leaf nodes

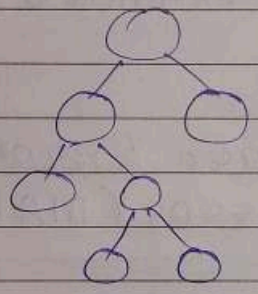
Space Complexity : O(nodes)

Typically we dont train decision Trees of more than depth 5 ; Atmost 10 on very massive datasets .

Hence Run Time Space Complexity : Reasonable.

Run-Time Time Complexity :

While ~~compl~~ computing $x_q \rightarrow y_q$.



Max time taken is the max num of traversals ..
= K
= max depth of any Leaf node

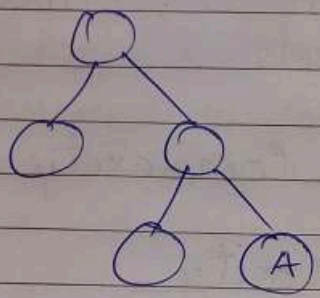
Hence Run Time Complexity is very reasonable

Hence D.T. is super useful during
→ Large data
→ small dimensionality
→ Low Latency .

We would see variants of the same like → Random Forest
→ GBDT.
which are very popular in internet applications

37.13 Regression Using D.T.

We have seen D.T. for classificaⁿ

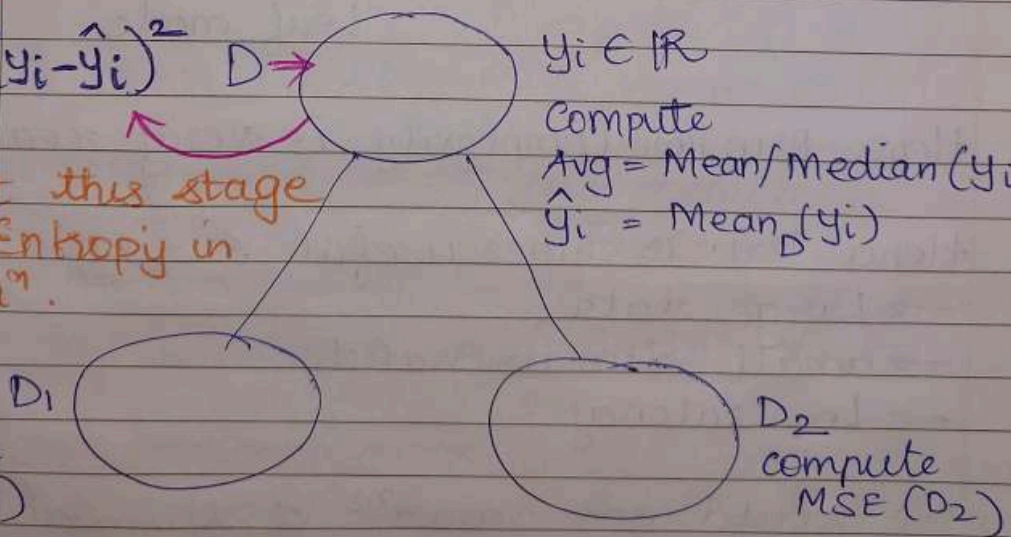


When the control reaches A, it checks the majority class, and assings y_q to x_q.

It can be easily extended to Regression

We use Mean Square Error (MSE) or Median Absolute Error (MAE).

$$MSE(D) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$



MSE at this stage is like Entropy in classificaⁿ.

Compute MSE(D_1)

Compute MSE(D_2)

The feature which gives least MSE would be considered

~~Re~~

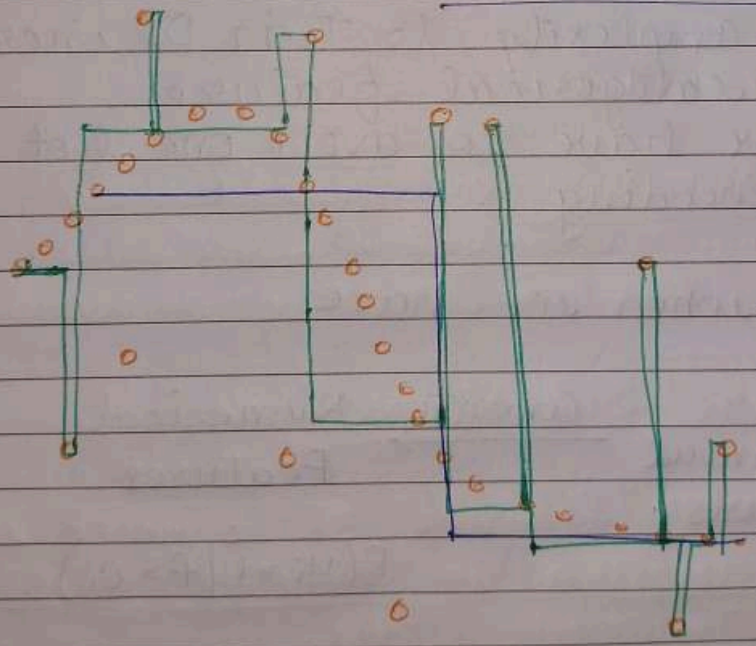
Classificaⁿ ⇒ Pick a feature ⇒ Reduce which Entrophy (Min Entropy = 0)
fi

Regression ⇒ Reduce MSE ⇒ Pick feature.

And

$$MSE_D(y_i) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$MAE = MAD = \text{Median}(|y_i - \hat{y}_i|)$$



o → data
— → max depth = 2 → underfit
— → max depth = 5 → overfit

37.14 Cases

① Imbalanced data

→ We HAVE TO Balance data
→ Upsampling / downsampling
→ Imbal. data impacts the entropy calculations / MSE calculaⁿ.

② Large dimensionality

→ We have to evaluate each of features at each node
→ Time complexity to Train DT increases.
→ For categorical feature.
→ we have to avoid one hot encoding.

Hence when we have.

Lot of categorical features → Converts → Numerical Features

$P(y_i = 1 | f = c_i)$

③ Similarity Matrix

→ D.T. wont work
→ D.T. need features explicitely
→ otherwise we cannot find entropy

(4) Multiclass Classification

→ need not do One Vs Rest (OVR) for D.T.
→ Entropy, IG, I_G can be computed when we have more than 1 category
→ It is straight forward.
→ D.T. naturally can be extended.

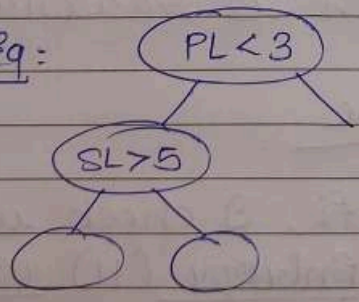
(5) Decision Surface

→ non linear decision surfaces
→ axes parallel hyper cuboid
→ Each of the internal node has a plane corresponding to it.

(6) Features Interaction

→ Possible in decision Trees.
→ As we have two features interacting with each other.

Eg:



My condition would be:

(PL<3) and (SL>5)

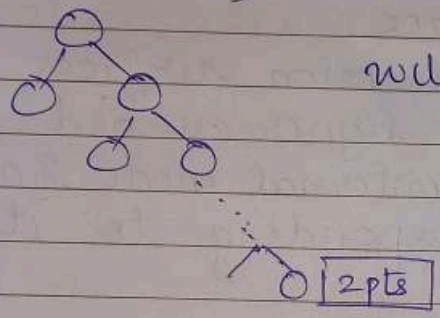
↳ Two features interacts.

→ Feature interac^n come as an added inbuilt benefit for D.T.
→ In Logistics Regression, we didnt have explicit feature interac^n.

→ In LR, we had feature transformaⁿ
Eg: $(f_i * f_j)$ ~~OR~~ ; (f_i AND f_j)

⑦ Outliers

→ As depth ↑ ; outliers would impact will make Tree unstable



⑧ Interpretability:

→ D-T are very interpretability.
→ very much readable
→ when depth is reasonable

⑨ Feature Importance

→ For every feature f_i, I check what is reduction in Entropy (H) or Gini Impurity (I_G).

→ I can add up (Normalized sum) all the reductions in H due to f_i

→ Sklearn → f_i has F.I- associated with it