$\therefore P(M_3/D) = \frac{P(D/M_3)P(M_3)}{P(M_2)} = \frac{0.01 \times 0.5}{0.02}$

Naive Bayes Algorithm

↳ Naive Bayes is a conditional probability model.

↳ It is called Naive because simple sophisticated model.

↳ Bayes became based on Bayesian model.

Suppose we have $x = (x_1, x_2 \cdots x_n) \rightarrow$ vectors
↳ feature (d-dimension)

$P(C_k | x_1, x_2 \cdots x_n)$

↳ $k \rightarrow$ can be multiclass
$k \rightarrow$ binary when $k = 2$.

$P(C_k|x) = \frac{P(C_k)P(x|C_k)}{P(x)}$

$P(C_1|x)$, $P(C_2|x)$ → Let suppose, this C_2 has highest prob. So x belongs to class C.

$\vdots$

$P(C_k|x)$

prob that x belongs to class C_k

For this eqn $P(x) \rightarrow$ Constant.

So, increase or decrease in probability depends on $P(C_k)P(x|C_k) = P(x \cap C_k) = P(x, C_k)$

Now, $P(C_k | x_1, x_2 \cdots x_n) = P(x_1, x_2 \cdots x_n, C_k)$ [$P(A \cap B) = P(B \cap A)$]

$= P(x_1 | x_2 \cdots x_n, C_k) * P(x_2, \cdots x_n, C_k)$

↳ defn from Conditional probability

$P(A|B) = P(A,B)$ & $P(B)$

Similarly, $P(x_2, x_3 \cdots x_n | C_k) = P(x_2 | x_3, x_4, \cdots x_n, C_k)$

↳ At last after continuing this same fashion

$P(C_k, x) = P(x_1|x_2 \cdots x_n, C_k) P(x_2|x_3 \cdots x_n, C_k) \cdots P(x_{n-1}|x_n, C_k) P(x_n|C_k) P(C_k)$

are we able to compute this?

$P(x_1 | x_2, x_3 \cdots x_n, C_k)$

↳ $P(w = 100 | hc = black, height = 5m, C_k = India)$

↳ It is called naive because it is conditional independence.

$P(A|B) = P(A)$ independent

$P(A|B,C) = P(A|C)$ conditional independent

So, here each feature is conditional independence.

$\therefore P(x_1 | x_2 \cdots x_n, C_k) P(x_2 | x_3 \cdots x_n, C_k)$

$P(x_1|C_k)$ (conditional independence) $P(x_2|C_k)$ "similarly for all

So, $P(C_k | x_1, x_2 \cdots x_n) \propto P(C_k) P(x_1|C_k) P(x_2|C_k) \cdots$

$\propto P(C_k) \prod_{i=1}^{n} P(x_i|C_k)$

$P(C_k | x_1, x_2 \cdots x_n) = \frac{1}{Z} P(C_k) \prod_{i=1}^{n} P(x_i|C_k)$

$P(x)$

Toy example train and test stages.

	outlook (f_1)	temp (f_2)	humidity f_3	wind f_4	Class (C) Play=Yes / Play=No
Day 1	Sunny	Hot	High	weak	No
Day 2	Overcast	Mild	Normal	strong	Yes
Day 3	Rain	cool	High	weak	No
⋮	⋮	⋮	⋮	⋮	⋮

n

This is the table given and we have to apply naive bayes on this.

This is a binary classification with feature f_1 f_2 f_3 f_4 all are categorical

to find $P(\text{class play=Yes} | x_q)$ $P(\text{class play=no} | x_q)$

any one which have high value x_q will belong to that class.

from Naive Bayes Theorem.

$$P(c|f_1, f_2, f_3, f_4) = P(f_1|c) * P(f_2|c) * P(f_3|c) * P(f_4|c) * P(c)$$

x_1

For the whole train we can see that $P(C=Yes) = \frac{9}{14}$, $P(C=No) = \frac{5}{14}$

P(outlook | class = yes, no)

outlook =	Frequency: play=yes	play=no	probability in class: play=yes	play=no
sunny	2	3	2/9	3/5
overcast	4	0	4/9	0/5
Rain	3	2	3/9	2/5
	total=9	total=5		

Similarly for P(temperature/class = yes, no) and P(humidity = / class yes, no)

Time Complexity : $O(nd c)$

Space = $O(d * c)$

$P(\text{class play=yes} | x') \propto P(\text{sunny}|\text{class play=yes}) * P(\text{cool}|\text{class play=yes}) * \cdots * P(\text{class=yes})$

$\propto 2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0053$

$P(\text{class play=no} | x') = P(\text{High}|\text{class play=no}) * P(\text{sunny}|\text{class play=no}) * \cdots P(\text{No})$

$= 4/5 \times 3/5 \times 3/5 \times 1/5 \times 5/14 = 0.021$

Now $P(\text{Yes}|x') < P(\text{No}|x')$

$\therefore x'$ belongs to x

Scattered like coin naive bayes blog.

* Naive Bayes is much better in terms of space-complexity.

Space C $\begin{cases} O(d*c) \rightarrow \text{KNN} \\ O(nd) \rightarrow \text{log R} \end{cases}$ ↓ less

Naive Bayes on Text data

for text data Naive Bayes works very well, at least better than all algorithm except Deeplearning.

Naive is considered to be a benchmark (baseline) for other algorithm.

Suppose, if acc. on naive bay = 98% → simple model

R.F ——— Deeplearning = 98.5% → complex one.

for classification of Spam, not spam naive bayes works very well

Suppose we have text data with class labels (binary)

text	class
text1 →	0
text2 →	1
:	0
	1
	0

Task: $P(y_q=1 | \text{text}_q)$

$P(y_q=0 | \text{text}_q)$

→ Here we first do text cleaning only. text → stopwords, stemming, n-gram } text preprocessing

output $\{w_1, w_2, w_3 \cdots w_d\}$ → Binary Bow

check if word is present or not don't care for repeatation.

$P(y=1|\text{text}) = P\{y=1 | w_1, w_2, w_3 \cdots w_d\}$ (features)

$P(y=1) * P(w_1|y=1) * P(w_2|y=1) \cdots P(w_d|y=1)$

$P(y=1)$: class prior; $P(w_d|y=1)$: likelihood

$P(y=1|\text{text}) \propto P(y=1) * \prod_{i=1}^{d} P(w_i|y=1)$

f1	
	0
	1
	0
	1
	0

train data

$P(y=1) = \frac{\text{train points with } y=1}{\text{total train points}}$

$P(y=0) = \frac{\text{train points with } y=0}{\text{total train points}}$

$P(w_i|y=1) = \frac{\text{train data pts with (contain } w_i \text{ and } y=1)}{\text{train data points with } y=0}$ → some particular word.

f4	
	1
	1
	1
	0
	0
	1

rearrange train data with y=1 and 0.

Laplace Additive Smoothing

we have that while in train { $P(y=1)$; $P(y=0)$ → class priors
$P(w_1|y=1)$ $P(w_1|y=0)$
$P(w_2|y=1)$ $P(w_2|y=0)$ → likelihoods.
$P(w_n|y=1)$ $P(w_n|y=0)$

Test :- $text_q = (w_1, w_2, w_3, w')$ — set of words in training data

what is the probability that a test review these words has class 0 or 1. we need to find.

* if w' is not present in $\{w_1, w_2, w_3 \cdots w_n\}$

So, $P(y=1|text_q) = P(y=1|w_1, w_2, w_3 \cdots w')$

$= P(y=1) * P(w_1|y=1) * P(w_2|y=1) \cdots * P(w'|y=1)$

Since this word is not present in training data.

So, if we ignore this word then this would not be correct, because at this moment $P(w_i|y=1)$ will become $P(w_i|y=0)$ → which is not good.

if we put $P(w_1|y=1)=0$ or $P(w_7|y=1)=0$ then whole probability will become 0. which is also not good.

$$P(w'|y=1) = \frac{P(w', y=1)}{P(y=1)}$$

$= \frac{\text{train pts where that } w' \text{ occurs } \& y=1}{\text{train pts where } y=1}$

$n_1 \rightarrow$ train pts where $y=1$

$= \frac{0}{n_1} = 0 \rightarrow$ not good

So, to escape these conditions we use Laplace smoothing (Additive smoothing)
↳ not Laplacian smoothing.

$$P(w'|y=1) = \frac{0+\alpha}{n_1+\alpha k}$$

1 - present
0 - not present

k = diff values that w' can take i.e. $k=2$ → since binary

Suppose $P(w'|y=1) = \frac{0+\alpha}{100+2\alpha}$ ($k=2$)

let $n_1 = 100$

Case 1 :- $\alpha = 1 = \frac{1}{102} \neq 0$

↳ very small probability

Case 2 :- $\alpha = 1000$

$$P(w'|y=1) = \frac{0+1000}{100+2000} \approx \frac{1000}{2000} \approx \frac{1}{2}$$

$$P(w'|y=1) = P(w'|y=0) = \frac{1}{2}$$

So, with increase in α to high value probability reaches to $\frac{1}{2}$ which is also correct to some extent.

↳ So, with Laplace smoothing.

↳ this can be applied to prob. of words which is present in train data also.

$$P(w_i|y=1) = \frac{\text{data pts with } w_i \text{ \& } y=1 + \alpha}{\text{data pts } y=1 + \alpha k}$$

for all word

present in training data

when $\alpha = 1$ ↳ called add one smoothing

$$P(w_i|y=1) = \frac{2+\alpha}{50+2\alpha} = \frac{2}{50} \rightarrow \alpha = 0$$

$$\alpha = 1 = \frac{2+1}{50+2} = \frac{3}{54}$$

$$\alpha = 100 = \frac{102}{250} \rightarrow \alpha = 1000 \approx \frac{1}{2}$$

So, when $\alpha \uparrow$ likelihood probability reaches to uniform distribution

So, when n_1 is small → less confidence in the prob.

when $\alpha \uparrow \Rightarrow$ Confidence increases.

Log probability for numerical stability.

We know that

$$P(y=1 \mid w_1, w_2, \cdots w_d) = P(y=1) * P(w_1|y=1) * P(w_2|y=1) * \cdots * P(w_d|y=1)$$

(0 to 1) (0 to 1) (0 to 1) ...

Then the overall probability will be very small.

Suppose if $d = 100$.

that python will find some numerical underflow. and might round off the digits.

So, to avoid this log probability is used for numeric stability

$$\log\left[P(y=1 \mid w_1, w_2 \cdots w_d)\right) \overset{\log}{=} \left(P(y=1) * \prod_{i=1}^{d} P(w_i|y=1)\right)$$

$$\log\left[P(y=0 \mid w_1, w_2 \cdots w_d)\right) = \log\left(P(y=0) * \prod_{i=1}^{d} P(w_i|y=0)\right)$$

log fn, $-x \uparrow$; $\log x \uparrow$

↳ monotonic fn. and log can easily convert multiply value to sum.

$$\log\left(P(y=1)\right) + \sum_{i=1}^{d} \log\left(P(w_i|y=1)\right) \qquad \log(a*b) = \log a + \log b.$$

Bias and Variance tradeoff.

We know that when there is high bias, there is underfitting

— variance, — overfitting

↳ training data change, model changes drastically.

So, here, in naive bayes, High bias and variance will depend on α.

Case 1: when $\alpha = 0$.

↳ $P(w_i|y=1) = \dfrac{\text{train data pts } w_i \text{ occur \& } y=1}{\text{train pts with } y=1}$

Suppose. $n = 2000$ datapoints
↳ 1000 +ve
↳ 1000 -ve

$= \dfrac{2}{1000}$ → word w_i occurs only 2 times ⇒ rare
↳ +ve data pts

{ words that are rare
↳ $P(w_i|y=1)$ } → overfitting

Since w_i occurs only 2 times in 2000 data

So, if we remove those 2 rare words, there will be small change in dataset.

But prob $\left(w_i|y=1\right) \xrightarrow{\text{shift}} \frac{2}{1000}$ to $\frac{0}{1000}$

model change

$= 0$

huge shift in probability

So, this is the case of ~~overfitting~~ overfitting.

Case 2: α is very large $\therefore \alpha = 1000$

$$P(w_i|y=1) = \frac{2 + 1000}{1000 + 2000} \simeq 1/2$$

$k = 2$

So, for all w_i

$$P(w_i|y=1) \simeq P(w_i|y=0) \simeq 1/2$$

$$\Downarrow$$

$$P(y_q=1 \mid x_q) \simeq P(y_q=0 \mid x_q) \simeq 1/2$$

this is the case of underfitting.

So, since α = large

$$P(y=1 \mid \overbrace{w_1, w_2 \cdots w_d}^{x_q}) = P(y=1) * \prod_{i=1}^{d} P(w_i|y=1)$$

$\simeq 1/2$ for all word

$$P(y=0 \mid w_1, w_2 \cdots w_d) = P(y=0) * \prod_{i=1}^{d} P(w_i|y=0)$$

$\simeq 1/2$

So, here the prob. will only depend on these values.

Suppose: $n = 1000$ $\quad n_1 > n_2$ $\quad$ so $P(y=1) > P(y=0)$

$n_1 = 700$ ↳ ↳ the +ve

$n_2 = 300$ $\quad$ for every case this will hold true.

$$\text{So } P(y=1 \mid w_1 \cdots w_d) > P(y=0 \mid w_1 \cdots w_d)$$

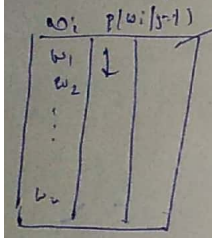
this is the case of underfitting.

finding α in naive Bayes and k in knn can be done by 10 fold cross validation.

↳ Hyperparameter tuning.

Feature importance and interpretability.

As we have seen in knn that important features can selected by forward/Backward feature selection,



Sort w_i's based on $P(w_i|y=1)$ decreasing order.

So, here words with high prob. would be a good feature to consider followed by next.

w_i's which have high value of

$P(w_i|y=1)$ ⟶ important words/features in determining that point $\in$ +ve class.

$P(w_i|y=0)$ → high

↳ w_i is important word/feature -ve class

feature importance is determined/obtained directly from the model in naive bayes

Interpretability:-

$x_q \rightarrow y_q = 1$

↳ $\{w_1, w_2 \dots w_d\}$

↳ 9 on concluding $y_q = 1$ because x_q contains words w_3 w_8 w_{10} (let say) which have a high value of $P(w_3|y=1)$, $P(w_{10}|y=1)$ $P(w_8|y=1)$

↳ $y_q \neq 0$ because $P(w_3|y=0)$ is very small.

Imbalanced data

We have $n \rightarrow n_1$ +ve
↳ n_2 -ve

when, $n_1 >> n_2$ or $n_2 >> n_1$ ⟹ we have imbalanced data

Suppose, 90% of train pts are +ve. → $n_1/n = 0.9$
10% of train pts are -ve → $n_2/n = 0.1$

$P(y=1|w_1, w_2 \dots w_d) = P(y=1) \prod_{i=1}^{d} P(w_i|y=1)$ ⟹ let $1/2$.

$P(y=0|w_1, w_2 \dots w_d) = P(y=0) \prod_{i=1}^{d} P(w_i|y=0)$ ⟹ let $1/2$

$P(y=0)$ → $n_2/n = 0.1$

imbalanced data

① class priors → majority/dominant class has an advantage

this can be solved by.

① Upsampling or downsampling.

↳ $n_1 = n_2$ $\quad P(y=1) = P(y=0) = 1/2$

② drop $P(y=1)$ & $P(y=0)$

$P(y=1) = P(y=0) = 1$

these two process is not often used in imbalanced data.

③ modified NB to account for class-imbalance
↳ not often used

imbalanced data

majority (+ve) $n_1 = 900 \rightarrow P(w_i|y=1) = \frac{0 \text{ to } 900}{900}$

minority (-ve) $n_2 = 100 \rightarrow P(w_i|y=0) = \begin{cases} \frac{0 \text{ to } 100}{100} \rightarrow \text{small} \\ \frac{\text{else}}{900} \end{cases}$

100 → minority
↳ α impacts more

other. ⟶

when laplace smoothing is used → majority ↳ α impacts less

Min(-ve) — Maj(+ve)

$P(w_i|y=1) = \frac{2}{10} = 2\%$ — same → $\frac{18}{900} = 2\%$ ← less effect in large

let assume $\alpha = 10$

$\frac{12}{120} = \frac{2+10}{100+20}$ ← more effect in large

$\frac{18+10}{900+20} = \frac{28}{920} = 3.04\%$

Tricks ⇒ Choose α_{true}, α_{false} different according to need.

Outliers

We have seen that if a word is not present in training data then through Laplace smoothing we can handle that with a reasonable value of α

Note: An outlier is that word is occured few times in training set.

{ $w_1, w_2, w_3 \cdots w_n$ }

↓ occured very few times so an outlier.

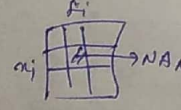
So in this case best way is to remove that word from dataset and don't consider it in any calculation.

Missing-Values :-

① Text data :- Test :- { $w_1, w_2 \cdots w_n$ }

↳ there is no case of missing data in test data

② Categorical features :-



$f_i \in \{a_1, a_2, a_3\}$

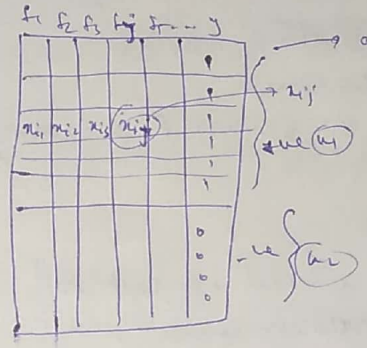
↳ when we have categorical feature then consider NAN as a category

$f_i \in \{a_1, a_2, a_3, NAN\}$

③ Numerical feature.

↳ in numerical feature ↳ do imputation → Simple mean/median/mode or Gaussian NB

Handling Numerical features (Gaussian NB)



arrange the train data with respect to +ve class

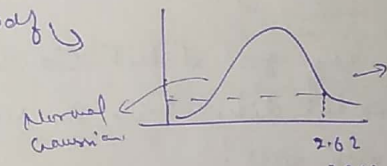
$P(y_i=1 \mid x_{i1}, x_{i2}, x_{i3} \cdots, x_{ij}, \cdots x_{id}) = P(y_i=1) \prod_{j=1}^{d} P(x_{ij} \mid y_i=1)$

↳ likelihood

class prior $= \frac{n_1}{n_1+n_2}$

So, how can we know the prob. of x_{ij}, this can be done by Gaussian Naive Bayes.

make a pdf ↳



→ obtain this prob. from the pdf of f_j (column) in D' → train data

Assume :- f_j (feature j) in D' → Gaussian dist with $N(\mu_j^1, \sigma_j^1)$

↳ +ve train

Assume f_j in D'' → Gaussian dist $N(\mu_j^0, \sigma_j^0)$

↳ -ve train

$P(w_i|y=1)$

↳ Bernoulli NB → 0 or 1

Multinomial NB

Gaussian NB → Likelihood we have.

Multinomial class classification

Naive Bayes can do multi-class classification. like it does binary classification.

Compare probs:
- $P(y_i=1 \mid w_1, w_2 \cdots w_d)$
- $P(y_i=0 \mid w_1, w_2 \cdots w_d)$
- $P(y_i=2 \mid w_1, w_2 \cdots w_d)$
- ⋮
- $P(y_i=c-1 \mid w_1, w_2 \cdots w_d)$

Suppose we have c class.

⇒ Suppose $P(y_i=6 \mid w_1, w_2 \cdots w_d)$ is the largest

So, $y_i = 6$ is label for $x_i = \{w_1, w_2 \cdots w_d\}$

Similarity matrix or distance matrix :-

Since, Naive Bayes doesn't works on distance based, so can't use distance matrix.

→ only and probabilistic approach.

Distance/Similarity matrix is well used by KNN.

Large Dimensionality.

Naive bayes is very good at text classification and since text classification ⇒ high dim. vectors, so, Naive Bayes can handle large Dimensions quite well.

But. when dimensions are large

$$P(y=1 \mid w_1, w_2 \cdots w_d) = P(y=1) \prod_{i=1}^{d} P(w_i \mid y=1)$$

d → dimensions

↳ prob. will decrease to very low.

So we use log probability here.

To save this can cause numerical stability.

Best and Worst Cases

① When Conditional independence of feature ← assumption → true
- ↳ NB perform well.
- ↳ false → degrade.

Here, some words like good, great, phenomenal are dependent but that is fine.

② For text classification → email spam, review polarity
- ↳ NB is best.

③ For categorical feature ↳ NB seldom used.

④ Interpretable, feature importance
- ↳ for this runtime complexity → low
- NB → train time complexity → low
- run time space → low

⑤ easily overfit if you don't do laplace smoothing.

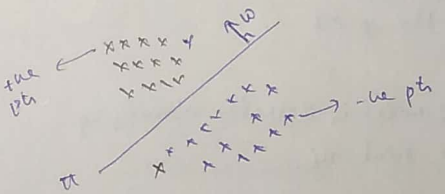
Logistic Regression

Geometric intuition of logistic Regression.

↳ Logistic Regression is a classification technique.

NB was probabilistic based

LR is geometric based



Here, logistic regression makes an assumption, that the data is linearly seperable or almost linearly seperable.

for 2D
↳ $y = mx + c$

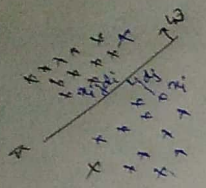
for nD
↳ $w^T x + b = 0$

if hyperplane π passes through origin $b=0$ ∴ $w^T x = 0$

↓

where $x \in R^d$ $w \in R^d$ ↳ vector, $b \in R^1$ ↳ scalar

Assumption of LR :- Classes are almost/perfectly linearly seperable.



$\pi = w^T x + b$

given $D_n = \{+ve, -ve\}$

Task :- find π (plane) that best seperates the pts from -ve pts

Here, $y_i = +1 \rightarrow +ve$ pt

$-1 \rightarrow -ve$ pt

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

distance of point x_i from the plane $= d_i = \frac{w^T x_i}{\|w\|}$ (w normal to the plane)

if $\|w\|$ is unit vector $= 1$

$d_i = w^T x_i$

$d_j = w^T x_j$

classifier

if $w^T x_i > 0$ then $y_i = +1$

if $w^T x_i < 0$ then $y_i = -1$

Case 1 (+ve pt.)

$y_i * w^T x_i > 0 \rightarrow$ if $y_i = +1 \rightarrow$ +ve pt

$w^T x_i > 0 \rightarrow$ classifier is saying it is +ve pt.

↓ this mean w is correctly classifying the pts.

Case 2 (-ve pt)

$y_i = -1$: -ve pts

$w^T x_i < 0 \Rightarrow$ LR concluding that x_i is a -ve pt

$y_i w^T x_i > 0 \Rightarrow$ w is correctly classifying the point.

So, when both the +ve & -ve pts

$y_i w^T x_i > 0 \Rightarrow$ the LR model is correctly classifying the point x_i.

Case 3 :- $y_i = +1$ (+ve pts)

$w^T x_i < 0 \Rightarrow$ LR is saying x_i is -ve class

$y_i w^T x_i < 0 \Rightarrow y_i = +1$, LR $= -1$ (misclassified)

Case 4 :- $y_i = -1 \Rightarrow$ -ve class

$w^T x_i > 0 \Rightarrow$ LR is saying x_i is +ve pt

↳ misclassified $y_i w^T x_i < 0$

A classifier is good if

↳ model gives minimum misclassification

or

maximum correctly classified pts.

$y_i w^T x_i > 0$

So, max $\sum_{i=1}^{n} y_i w^T x_i \rightarrow$ here $(x_i, y_i) \Rightarrow$ are fixed in train D_n.

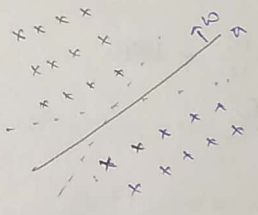
only w is variable.

We have to find such plane(w) such that this value is max^n.

$w^* = \arg\max \left(\sum_{i=1}^{n} y_i w^T x_i \right) \Rightarrow$ math optimization problem.

find w with max^n the value $y_i w^T x_i$ among all w.

Sigmoid function Squashing.



till now we looked for best optimized w^* when $w^* = \arg\max \sum_{i=1}^{n} y_i w^T x_i$

this can be any random separating plane.

$\arg\max \sum_{i=1}^{n} y_i w^T x_i$ ← this is the signed distance

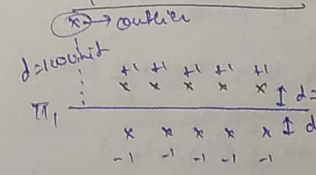
↳ y_i can be $+1$ or -1

$w^T x_i \rightarrow$ dist from x_i to π.

$y_i w^T x_i \rightarrow$ +ve $\rightarrow \pi$ as defined by w correctly classifies x_i.

↳ -ve $\rightarrow$ incorrectly classifies x_i.

But Suppose a situation,



Task :- find a best $\pi(w)$ max^{sum} of signed distance

each point is at a dist of 1 unit from plane except the outlier.

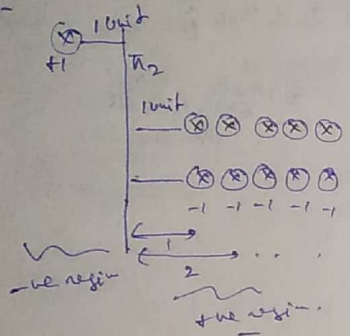
Case1 :- π_1 is my seperator,

So, $\sum_{i=1}^{n} y_i w_1^T x_i$ = 1+1+1+1+1+1 → +ve
1+1+1+1+1+1 → -ve
-1 ∞ → -ve outlier

= -90.

Since the plane π_1 was looking a good seperator but due to one outlier, $\sum_{i=1}^{n} y_i w_1^T x_i < 0$ this becomes one.

Case2 :-



Consider plane π_2

$\sum_{i=1}^{n} y_i w_2^T x_i$
= 1+2+3+4+5 → +ve
-1-2-3-4-5 → -ve
+1 → outlier
= +1 > 0

plane π_1 was good seperator but due to one outlier plane π_2 ~~was~~ better than π_1.

In Π_1 Correctly classified points was = 10/11
— π_2 — = 6/11

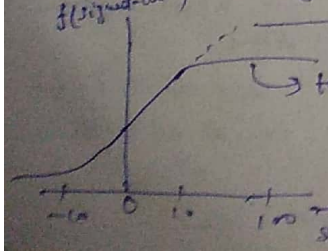
intuitively :- π_1 is better the π_2

So, one single outlier pt. is changing the model here, which is not good.

So, mon-sum of signed-dist. is not outlier prove.

So, **Squashing** technique is used.

Idea :- instead of using signed distance



if signed-distance is small :- use it as it is
large :- make it smaller.

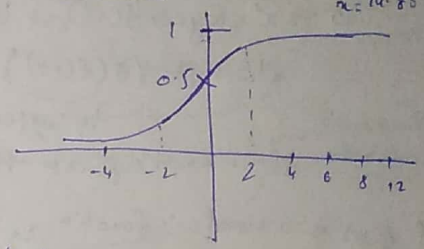
So instead of using $\text{argmax} \sum_{i=1}^{n} y_i w^T x_i$ directly, we will use.
$\text{argmax} \sum_{i=1}^{n} f(y_i w^T x_i)$

So, we will use $f(y_i w^T x_i)$ such that for some fixed extent it signed dist will increase linearly, but if it passes that threshold value it started tampering off → to make value reduced from original value.

So, for this, Sigmoid function is used.

Sigmoid fn :- $\sigma(x)$

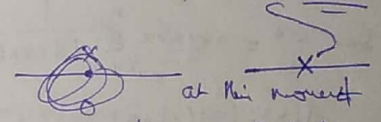
$$\sigma(x) = \frac{1}{1+e^{-x}}$$



Here in sigmoid function,
when signed dist. is small
from -2 to 2 we can see that
graph is linear, after that it tampered off for higher values.

$\sigma(x) = \frac{1}{1+e^{-x}}$
↓
max = 1
min = 0

So, when signed dist = 0.



at this moment
it will predict $\hat{y}_i = 0.5$,
since it can belong to any of the two class.

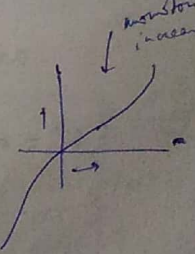
$$w^* = \text{arg max} \sum_{i=1}^{n} \sigma(y_i w^T x_i)$$

$$w^* = \text{arg max} \sum_{i=1}^{n} \frac{1}{1+\exp(y_i w^T x_i)}$$ → less impacted by outlier.

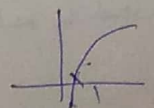
This function is easily differentiable.

Mathematical formulation of Objective function.

We know that when $g(x)$ → is monotonic function.
Then, when $x \uparrow$ $g(x) \uparrow$
if $x > x_1$ then $g(x_2) > g(x_1)$



$\log(x)$ is also a monotonic ↑ fn when $x > 0$



$x \uparrow$, $\log(x) \uparrow$
$x \downarrow$, $\log(x) \downarrow$

$\arg\min \sum_{i=1}^{n} \log(1+\exp(-y_i w^T x_i))$

suppose $\frac{1}{\infty}$ is not there then $\log e = 1$.

$\arg \sum_{i=1}^{n} -y_i w^T x_i$

$\arg\max \sum_{i=1}^{n} y_i w^T x_i$

$w^* = \arg\min \sum_{i=1}^{n} \log(1+\exp(-y_i w^T x_i))$ → geometrical approach

↳ $+1$ or -1

$w^* = \arg\min \sum_{i=1}^{n} -y_i \log p_i - (1-y_i)\log(1-p_i)$ → probabilistic method.

↳ y_i is $+1$ or 0

↳ $p_i = \sigma(w^T x_i)$

Weight vector

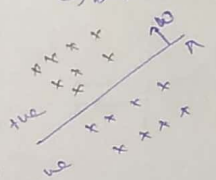
optimized weight vector ← $w^* = \arg\min \sum_{i=1}^{n} \log(1+\exp(-y_i w^T x_i))$

↳ best weight vector

weight vector

$w^* = \langle w_1, w_2, w_3 \ldots w_d \rangle$

$\in R^d$ since $x_i \in R^d$ → d feature.



$w^* = \langle w_1, w_2, w_3 \ldots w_d \rangle$

$f_1\ f_2\ f_3 \ldots f_d$

every feature have some weight associated with it.

Decision: $x_q \to y_q$ { if $w^T x_q > 0$ then $y_q = +1$; if $w^T x_q < 0$ then $y_q = -1$ } → checks if query point belongs to which label.

probabilistic ↳ $\sigma(w^T x_q) = P(y_q = +1)$

↳ 0 to 1

Interpretation of w:

Case 1: if $w_i = +ve$, $x_{qi} \uparrow \to (w_i x_{qi}) \uparrow \to \sum_{i=1}^{d} (w_i x_{qi}) \uparrow \Rightarrow \sigma(w^T x_q) \uparrow$ (since σ is monotonic function)

$\therefore P(y_q = +1) \uparrow$.

Case 2 :- if $w_i = -ve$;

$x_{ij} \uparrow \Rightarrow (w_i x_{ij}) \downarrow \longrightarrow \left(\sum_{i=1}^{d} w_i x_{ij}\right) \downarrow \longrightarrow \sigma(w^T x_j) \downarrow$

$P(y_j = -1) \uparrow \longleftarrow P(y_j = +1) \downarrow$

Since, $P(y_j = 1) = 1 - P(y_j = -1)$

So, we see that when w_i is +ve and x_{ij} is increasing
Then $P(y_j = +1) \uparrow$

and vice, $w_i = -ve$ and x_{ij} is increasing
then $P(y_j = -1) \downarrow$

L2 Regularization Overfitting and Underfitting

Here we have, $w^* = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{n} \log(1 + \exp(-y_i w^T x_i))$

let $z_i = y_i w^T x_i$

$= \underset{w}{\operatorname{argmin}} \sum_{i=1}^{n} \log(1 + \exp(-z_i))$

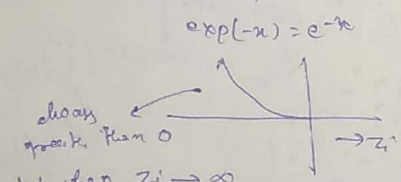
So, $\sum_{i=1}^{n} \log(1 + \exp(-z_i))$ ($\exp(-z_i) \geq 0$)

we know that $\log 1 = 0$

So, $\log(1 + \exp(-z_i)) \geqslant 0$

and the minimum this value
$z_i \to \infty$
and $\exp(-z_i) \to 0 \Rightarrow \log(1) = 0$.

we know, $\exp(-x) = e^{-x}$



but when $z_i \to \infty$
$e^{z_i} \to 0$.

So, minimal value of $\sum_{i=1}^{n} \log(1 + \exp(-z_i))$ is "0".
This occurs when $z_i \to \infty$ for all i

Since, $z_i = y_i w^T x_i \to +ve$ $\quad D = \{(x_i, y_i), i = 1 \text{ to } n\}$ (fixed value from data set)
Correctly classified

$z \to \infty$ ↳ modify my w in such a way that each $z_i \to \infty$

⇒ if I pick any w such that
(a) all training pts are correctly classified
(b) $z_i \to \infty$
Then we find best w

↳ then my model is more prone to "overfitting".

Since, $z_i = y_i w^T x_i \to y_i$ and x_i are fixed.

$z_i \propto w^T$

$z_i \to \infty$
$w^T \to \infty \Rightarrow w \to \infty$

So $w_i \to \infty$ or $w_i \to -\infty$ $\not\Rightarrow$ minima tends to 0.

w is a normal to plane.

$w^T w = 1$

Regularization

$w^* = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{n} \log(1 + \exp(-y_i w^T x_i)) + \lambda w^T w$

(loss term; regularization term)

$= \lambda \|w\|_2^2$; $\lambda \sum_{j=1}^{d} w_j^2$; $\|w\|_2$: L2 norm of w.

$\left\{ \sum_{i=1}^{n} \log(1 + \exp(-z_i)) + \lambda \sum_{j=1}^{d} w_j^2 \right\}$

$\log(1 + \exp(-z_i)) \to 0$; $\lambda \sum w_j^2$: v.v large

There is a tradeoff b/w these two values and this to kept the value in between {0 to v.v. large}.

Here λ is hyperparameter in LR.

when $\lambda = 0 \Rightarrow$ overfitting to the training data $\Rightarrow$ high variance
$\lambda = \text{large} \Rightarrow$ underfit $\Rightarrow$ high bias

We are introducing regularization because we don't want values $w_i \to \infty$, $w_i \to -\infty$ or overfit

we use $\lambda \|w\|^2 \to L_2$-regularization
L_2 norm here

L1 regularization and sparsity.

see how this optimized vector

$$w^* = \arg\min_w \sum_{i=1}^{n} \log\left(1+\exp(-y_i w^T x_i)\right) + \lambda \|w\|_2^2$$

(z_i = $y_i w^T x_i$; logistic loss; L_2-regularization)

We have, an alternative to L_2-regularization, $\longrightarrow$ L_1 regularization.

L_1 regularization is basically. $\|w\|_1 = \sum_{i=1}^{d} |w_i|$ → sum of w_i ↳ absolute value.

$$w^* = \arg\min_w (\text{logistic loss for training data}) + \lambda \|w\|_1$$

(λ → hyperparameter; $\|w\|_1$ → L_1 regularization)

This will avoid $w_i \to \infty$ or $\to -\infty$

Sparsity:- $w = \langle w_1, w_2, w_3, \ldots, w_d \rangle$

{ solution to LR is said to be sparse if many w_i's are zero }

↓ w^*

↳ So, if we use L_1 reg in LR, all the unimportant or less-important features become zero. but in case of L_2 regularization it becomes small but not necessarily zero.

elastic-net:-

↳ the concept of this is use L_1 and L_2 regularization simultaneously.

$$w^* = \arg\min_w \sum_{i=1}^{n} \log\left(1+\exp(-z_i)\right) + \lambda_1 \|w\|_1 + \lambda_2 \|w\|_2^2$$

↳ we can find easily the two hyperparameters λ_1 and λ_2.

Probabilistic Interpretation Gaussian Naive Bayes

when model was Naive Bayes

↳ when for real valued features ① ↳ Gaussian Distribution was used

② $y_i = +1$ or 0

↳ Bernoulli random Variable.

So, for probabilistic Interpretation, (HW2) following assumption need to make

→ Y is boolean governed by a Bernoulli distribution.

→ $X = (x_1, x_2 \ldots x_n)$ is a continuous random variable.

→ For each X_i, $P(x_i | y=y_k)$ is a Gaussian distribution of the form $N(\mu_{ik}, \sigma_i)$

→ For all i and $j \neq i$, x_i and x_j are Conditionally independent

Logistic Regression = GNB + Bernoulli ← Sum of these two.

$P(x_i|y_i)$ $\quad$ $y_i \sim$ Bernoulli.

for geometry ↳ $w^* = \arg\min_w \sum_{i=1}^{n} \log\left(1+\exp(-y_i w^T x_i)\right)$ + regularization

(y_i ∈ +1 or −1)

for probabilistic approach ↕ Same

$w^* = \arg\min_w \sum_{i=1}^{n} -y_i \log p_i - (1-y_i)\log(1-p_i)$ + regularization

(y_i ∈ 0 or 1) $\quad$ when $p_i = \sigma(w^T x_i)$

Case1:- $y_i = +ve$ $\quad$ geo :- $y_i = +1$ $\quad$ prob $y_i = +1$

geo :- $\log\left(1+\exp(-w^T x_i)\right)$ $\quad$ rejecting regularization

prob :- $-1 \cdot \log\left(\frac{1}{1+\exp(-w^T x_i)}\right)$ $\quad$ $(-\log(x) = \log(1/x))$

$= \log\left(1+\exp(-w^T x_i)\right)$

Case2:- $y_i = -ve$; $y_i = -1$ ← geo $\quad$ $y_i = 0$ ← prob

geo :- $\log\left(1+\exp(w^T x_i)\right)$

prob :- $-1 \cdot \log\left(1 - \frac{1}{1+\exp(-w^T x_i)}\right)$

$-1 \cdot \log\left(\frac{\exp(-w^T x_i)}{1+\exp(-w^T x_i)}\right)$

$\log\left(\frac{1+\exp(-w^T x_i)}{\exp(-w^T x_i)}\right)$

$= \log\left(1 + \frac{1}{\exp(-w^T x_i)}\right)$ $\quad$ $\left[\frac{1}{e^x} = e^{-x}\right]$

$= \log\left(1+\exp(w^T x_i)\right)$

Loss minimization interpretation

We have,

$$w^* = \arg\min_w \sum_{i=1}^{n} \log(1+\exp(-y_i w^T x_i))$$

$z_i = y_i w^T x_i \quad f(x_i) = w^T x_i$

$z_i = y_i f(x_i)$

We want our model to have minimum number of incorrectly classified points.

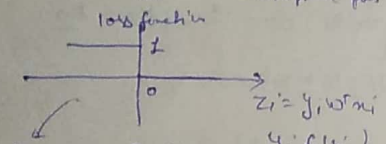
for that in loss minimization

we assume $+1$ = incorrectly classified points

0 = correctly classified points

$w^* = \arg\min_w$ (number of incorrectly classified pts)

{ we want to do min-loss, non-profit



this is the 0-1 loss function.

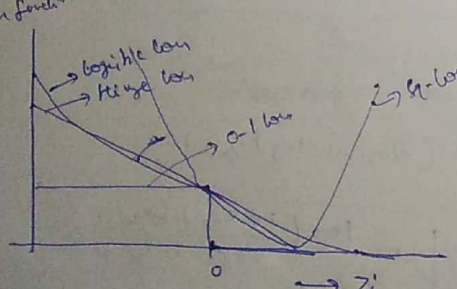
when $z_i > 0 \rightarrow$ loss is 0 $\Rightarrow$ max^m no. of correctly classified pts.

$z_i < 0 \rightarrow$ loss is 1 $\Rightarrow$ max^m no. of incorrectly classified pts.

But here, there is a discontinuity when $z=0$.

ideal ↳ $w^* = \arg\min_w \sum_{i=1}^{n} 0\text{-}1\ loss(x_i, y_i, w)$

$$0\text{-}1\ loss(z_i) = \begin{cases} 1 & \text{if } z_i < 0 \\ 0 & \text{if } z_i > 0 \end{cases}$$



So, for continuity in graph, we can take hinge loss plot or logistic loss or any similar kind of plot.

Take logistic loss.

when $z_i > 0 \rightarrow$ it will be near zero

when $z_i > 4$

$z_i < 0 \rightarrow$ it will not exactly 1 by guess but this.

loss-minimization interpretation.

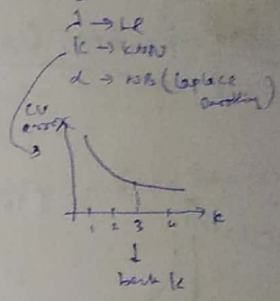
↳ loss-fn → logistic loss → LR
hinge-loss → SVM
expo-loss → AdaBoost
Square-loss → Linear Reg.

when loss function changes or model changes

Hyperparameters and random search.

$\lambda \rightarrow$ hyperparameter when $\lambda = 0 \Rightarrow$ overfitting

$\lambda = \infty \Rightarrow$ underfitting



k in KNN is a set of integers $\{1, 2, 3 \cdots n\}$

whereas in λ in LR is a real number

λ can be any values.

$\lambda \in R$ { $\lambda = 0.123$, $\lambda = 0.2256$ ⋮

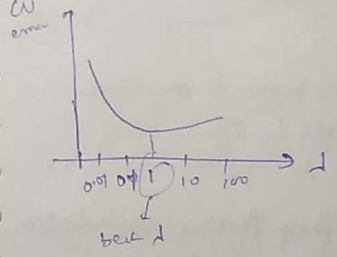
So, to get approximately correct λ

Grid Search (Brute Force) → technique is used.

we check for multiple value of λ in a multiple of 10

as $\lambda = [0.001, 0.01, 0.1, 1, 10, 100, 1000]$ (×10 between each)

or $\lambda = [1, 2, 3, 4, 5 \cdots 10]$

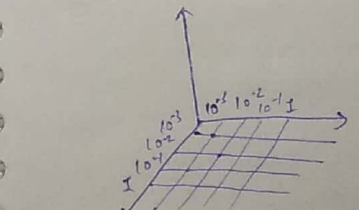


we then plot the graph b/w CV-error and λ, and the value corresponding to lowest CV error will be taken as best λ.

Elastic Net :- $\lambda_1 \|w\|_1 + \lambda_2 \|w\|_2^2$

$\lambda_1 = [10^{-3}, 10^{-2} \cdots 10^{3}]$ (n_1)

$\lambda_2 = [10^{-3}, 10^{-2} \cdots 10^{3}]$ (n_2)



for each pair of d_1 and d_2 CV will be plotted here, to get the best (λ_1 and λ_2)

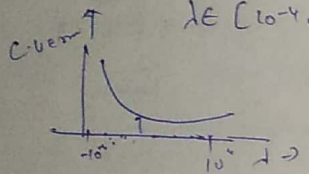
So, in Grid search, as the no. of hyperparameter increases, the time of model needed to train increases exponentially.

λ_1 : 1 hyperparameter → n_1 train time
$\lambda_1+\lambda_2$: 2 hp ——— → $n_1 \times n_2$ train time
$\lambda_1 \lambda_2 \cdots \lambda_{10}$ ——— → $n_1 \times n_2 \times \cdots n_{10}$ train time.

So, as $k \uparrow$ $n^k \uparrow$.

So, Random search is used.
↳ It is almost as good as grid search.
It ~~can be~~ perform better even if hyperparameter is large.

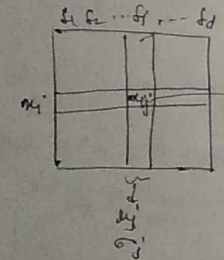
$\lambda \in [10^{-4}, 10^4]$ ← random pick value in the given interval and plot the graph.



[GridSearchCV
RandomizedSearchCV

$\lambda \in [10^{-4}, 10^4]$
$\lambda \in$

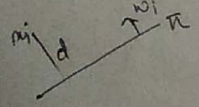
Column/feature standardization.



$$x_{ij}' = \frac{x_{ij} - \mu_j}{\sigma_j} \rightarrow \text{standardization.}$$

if features have different scales then it can affect the model, so to keep the data to same scale standardization is used.

So, in Logistic Regression it is "mandatory" to perform feature standardization before training on your data, because like knn, LR is also distance based.



Feature importance and model interpretability.

We know that Logistic regression is a combination of GaussianNaive Bayes and Bernoulli

f_1 f_2 ... f_j ... f_d
↓ ↓ ↓ ↓
$w \rightarrow w_1, w_2 \quad w_j \quad w_d$

all features have some weight associated.

Assume all the features are independent (Naive Bayes)

In knn → feature - importance is determined by forward/backward feature sel^n.
In NaiveBayes → $P(x_i | y=+1) \rightarrow$ high prob corresponding to feature can have of feature ~~independence~~ importance

So in LR → w_j's → weights corresponding to feature determine feature Importance.

$|w_j| \Rightarrow$ absolute value of weight corresponding to f_j

when $|w_j| \uparrow$; $(w^T x_q) \uparrow$

Case1 :- if w_j = +ve & large ; $\sum_{j=1}^{d} w_j \cdot x_{qj} = w^T x_q$
↳ $P(y=+1) \uparrow$

Case2 :- if w_j = -ve & large ; $\sum_{j=1}^{d} w_j x_{qj}$ $P(y=-1) \uparrow$

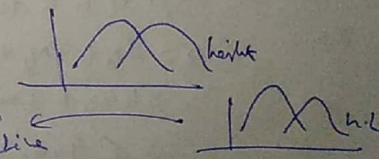
So, if $|w_j| \uparrow$ → pred corresponding to +1 or -1 increases.

example: predict gender : male & female
(+1) (-1)

feature → hair-length : $|w_{hl}|$ is large
↳ since most of girls have large hair length so, it has high weight.

So, $w_{hl} \uparrow$; $P(y=-1) \uparrow$

$\Rightarrow$ height $\uparrow$; $P(y=+1) \uparrow$
↑ value.



But hair length is more effective for girls.

Model Interpretability.

$x_q \rightarrow y_q = +1 \rightarrow$ top-features
$\rightarrow y_q = -1$

↓ hair-length height ↑ ⇒ this man boy.

Similarly, we can find 5 most imp feature out of 100 features.

Collinearity of features

feature importance w.r.t weight vector $|w_i|$ is only good when our features are independent.

if features are collinear or Multicollinear then weight vector doesn't hold good.

f_i, f_j

Such that $f_i = \alpha f_j + \beta$

↳ then f_i & f_j are collinear

Multicollinearity: features

if f_1 f_2 f_3 f_4 such that

$f_1 = \alpha_1 + \alpha_2 f_2 + \alpha_3 f_3 + \alpha_4 f_4$

then f_1, f_2, f_3 & f_4 are said to be multicollinear.

Q why does $|w_i|$ not be useful as f.I if features are collinear.

Suppose. $D = \langle x_i, y_i \rangle_{i=1}^{n}$

$w^* = \langle 1, 2, 3 \rangle$: $x_q = \langle x_{q_1}, x_{q_2}, x_{q_3} \rangle$
f_1 f_2 f_3

$w^{*T} x_q = x_{q_1} + 2x_{q_2} + 3x_{q_3}$

if $f_2 = 1.5 f_1 \Rightarrow f_1$ & f_2 are collinear

then, $w^T x_q = x_{q_1} + 3x_{q_1} + 3x_{q_3}$
$= 4x_{q_1} + 3x_{q_3}$
$= \langle 4, 0, 3 \rangle$

So, $w^* = \langle 1, 2, 3 \rangle$ → not imp.

$\hat{w} = \langle 4, 0, 3 \rangle$
↳ most imp

So for the same feature when two features are collinear, important feature changes drastically,
i.e $w^* \rightarrow$ 3 is an imp. feature
i.e $\hat{w} \rightarrow$ 4 is an imp feature

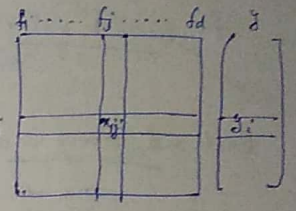
So, if features are collinear
⇓
weight vector can change arbitrarily.
⇓
So ← $|w_i|$ can't be used as f.I.

So, before proceeding further, check if there is colinearity in features or not.

One way to check is perturbation technique

↳ Here relative some error (noise) is added to each column.

$x_{ij} + \varepsilon$
↳ small noise.



Now, if.

before perturbation : $w = \langle w_1, w_2, w_3 \ldots w_j \ldots w_d \rangle$

after perturbation → $\hat{w} = \langle \hat{w}_1, \hat{w}_2, \ldots \hat{w}_j \ldots w_d \rangle$

{ if w_i & $\hat{w}_i$ differ significantly then your features are collinear }

⇓

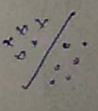
So, can use $|w_i|$'s as f.I.

Or we can use forward feature selection technique.

~~Train & Runtime Space & time complexity.~~

Real world Case.

In logistic Regression we make an assumption that data is almost linearly separable



For imbalanced data: on sampling or down sampling is used

Outliers ↳ less effect because of $\sigma(x)$ function.
↳ Calculate w^* using D_{train}
↳ $x_i \rightarrow w^{*T} x_i$: find dist from π to x_i
↳ remove pts which are very far away from π for $D_{train} \rightarrow D'_{train}$
↳ again find $\hat{w}^*$ for D'_{train}
↳ find cols.

For missing values :- imputation → mean, median
↳ mode is used.

for multiclass class^{n} :- One Vs Rest is used
or
{ extention to LR → Softmax classifier
↳ multinomial LR is used.

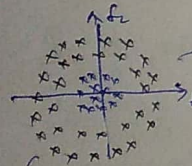
for similarity matrix ↳ extention to LR → (kernel LR → SVM is used)

Best and worst Case :- when dataset is almost linearly separable
↳ and we need a low-latency system (Low reg)
then LR is used because it is very fast to train.

for Large d :- d is large { chance of data is linearly separable is high }

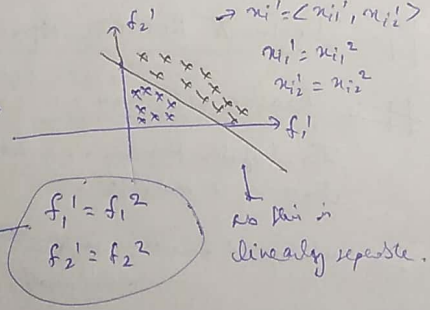
Non-linearly separable data and Feature Engineering.

Cases we don't have linearly separable data then, we use feature Engineering to get good result by LR.
How to use feature engineering comes from experience.



→ Here this data is not linearly separable.
{ drawing a plane in the space of f_1 & f_2 can't separate the +ve & -ve pts }

↳ So we use Feature Engineering →



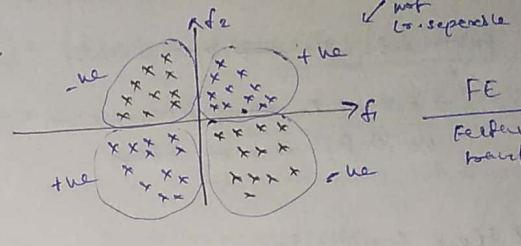
$x_i \langle x_{i1}, x_{i2} \rangle$

feature transformation → $f_1' = f_1^2$, $f_2' = f_2^2$

So this is linearly separable.

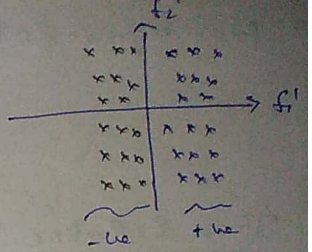
Here $f_1^2 + f_2^2 = r$ (f_1 → f_1', f_2 → f_2')
not linear in f_1 & f_2
but linear in f_1^2 & f_2^2

So, the most important aspects of applied ML/AI
① Feature Engg → VVI → depends on your skill/creativity.
② Bias-Variance
③ Data Analysis & Visualization



not linearly separable

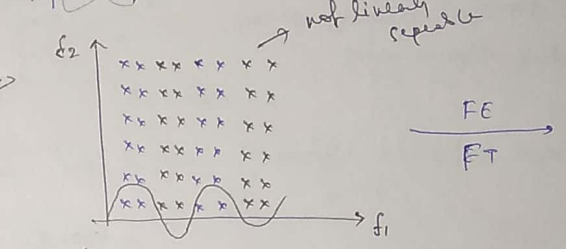
FE
Feature transformation



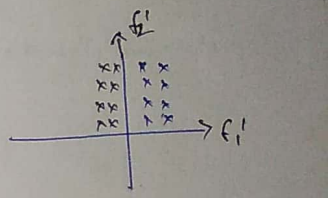
If it's like XOR

XOR	0	1
0	0	1
1	1	0

$f_1' = f_1 * f_2$
$f_2' = f_2$



FE
FT



↳ but this data can be separated linearly by sine curve.

$f_1' = \sin(f_1)$
$f_2' = f_2$

⇒ Typical transforms for real valued features.

① $f_1 * f_2$, $f_1^2, f_2^2, f_2^3, f_1^3 \ldots$
polynomial feature

② Trigonometric :- $\sin(f_1)$; $\cos(f_1)$
$\sin(f_1) * \cos(f_2)$, $(\sin(f_1))^2$

③ Boolean feature :- OR, AND, XOR

④ $\log(f_1)$, e^{f_1}

Extentions to Generalized linear model

extention to logistic-regression — Generalized linear model (GLM)
→ log-reg → GNB + Bernoulli ← prob. interpretation
→ Multinomial - LR → GNB + multinomial distribution ← multiclass classification
→ linear regression → $P(Y|x) \sim N(\mu, \sigma^2)$ → regression $y_i \in R$