Quadratic surfaces – Circles, Parabola

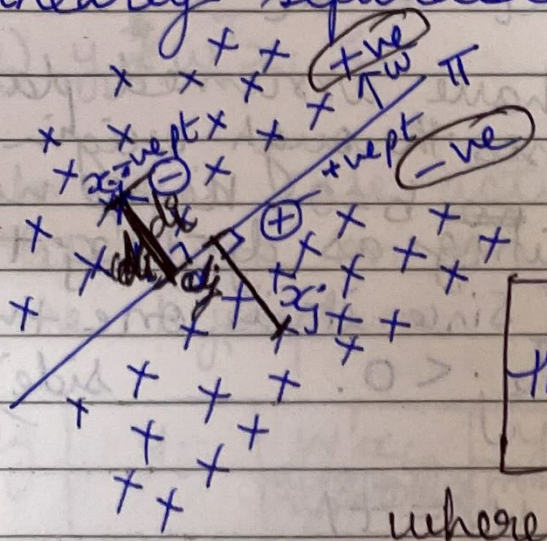
Ch 8 – Logistic Regression

VI: Geometric Intuition of Logistic Regression

Not regression. It is actually a classification technique.

Geometric Intuition:

Assume that classes are almost/perfectly linearly separable (there is a plane that most of the data separates 2 class)



$\Pi(w, b)$, w = normal to Π, b = intercept term

Π (plane) $\Rightarrow w^T x + b = 0$

w: normal to Π; b: intercept

If the plane passes through origin (b=0)

$\therefore w^T x = 0$.

where $x \in \mathbb{R}^d$, $w \in \mathbb{R}^d$

b = scalar, x = vector, w = vector

Assume, classes are almost/perfectly linearly separable.

Given: $D_n = \{+ve, -ve\}$

Find: w & b. Find a plane Π that best separates +ve pts from negative points.

Say $y_i = +1$ for class +ve

$y_i = -1$ for class -ve.

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

$d_i = \frac{w^T x_i}{\|w\|}$ — obtained in linear algebra

We assumed :-
↳ w is normal to the plane
w = unit vector.
$\therefore \|w\| = 1$.
So, $d_i = w^T x_i$

Any, $d_j = w^T x_j$

We have assumed plane passes through origin
So ~~no~~ $b = 0$. Hence not writing as $d_j = w^T x_j + b$.

$d_i \Rightarrow w^T x_i > 0$ (Since it is one +ve side)
And hence, $d_j \Rightarrow w^T x_j < 0$.

Classifier says :-
If $w^T x_i > 0$ then $y_i = +1$
and
$w^T x_j < 0$ then $y_i = -1$

Case 1:
So $y_i * w^T x_i > 0$ is correctly classifying the point.

Case 2:
& $y_i * w^T x_i < 0$ is correctly classified points.

Case 3:
y_i = +ve point.
If x_i is in -ve plane direction.
but x_i is actually +ve. Misclassified

NOTE: multiply

1) $y_i * w^T x_i > 0$

In this case actual label (y_i) and predicted label $\hat{y}_i$ are on the same side of the hyperplane

eg	y_i	$w^T x_i$ (hyperplane distance of x_i from hyperplane)	$y_i w^T x_i$ (signed distance)	classification
	-ve	-ve	+ve	Correctly classified
	+ve	+ve	+ve	" "

2) $y_i * w^T x_i < 0$ (multiply)

If correctly classified $y_i w^T x_i$ is always +ve.

In this case actual label (y_i) and predicted label $\hat{y}_i$ are on the different side of the hyperplane

eg	y_i	$w^T x_i$ (distance of x_i from hyperplane)	$y_i w^T x_i$ (signed distance)	classification
	+ve	-ve	-ve	Incorrectly classified
	-ve	+ve	-ve	" "

If incorrectly classified, $y_i w^T x_i$ is always -ve.

Logistic Regression will predict sigmoid($w^T x_i$) < 0.5 as negative and sigmoid($w^T x_i$) ≥ 0.5 as positive, here threshold is 0.5

$$w^* = \underset{w}{\operatorname{argmax}} \sum_{i=1}^{n} y_i w^T x_i$$

(w: variable; y_i, x_i: fixed) optimization function

V2: Sigmoid Function and Squashing

w^* — optimal w (best hyperplane's perpendicular)

For every plane there are unique perpendicular (w)

optimal w that maximizes sum of $y_i w^T x_i$

$$w^* = \underset{w}{\operatorname{argmax}} \sum_{i=1}^{n} y_i \underbrace{w^T x_i}_{\text{signed distance}}$$

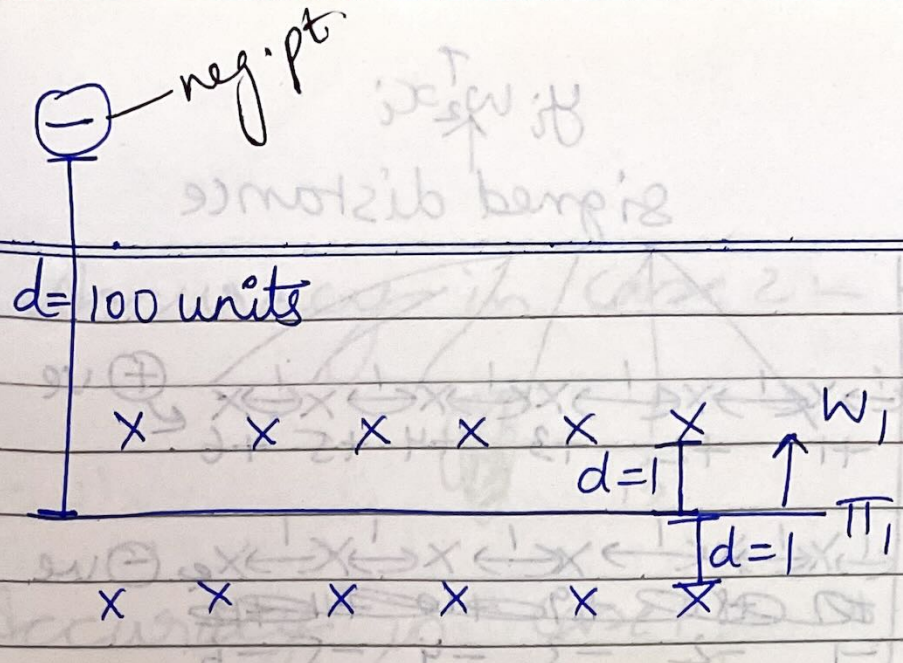
where,

$w^T x_i$ = distance from x_i to Π

w is unit vector.

if signed distance is +ve, plane (Π) defined by w correctly classifies x_i

if signed distance is −ve, plane (Π) defined by w incorrectly classifies x_i



Case 1: π_1 is the class separating hyperplane

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

$+1$

For +ve points, $y_i = +1$, $w_1^T x_i = +1$, $y_i w^T x_i = +1$

For -ve points, $y_i = -1$, $w_1^T x_i = -1$, $y_i w^T x_i = +1$

So for all 12 points $\Rightarrow$ $\sum_{i=1}^{m} y_i w_1^T x_i \Rightarrow$ $+1+1+1+1+1+1$ (= +ve)

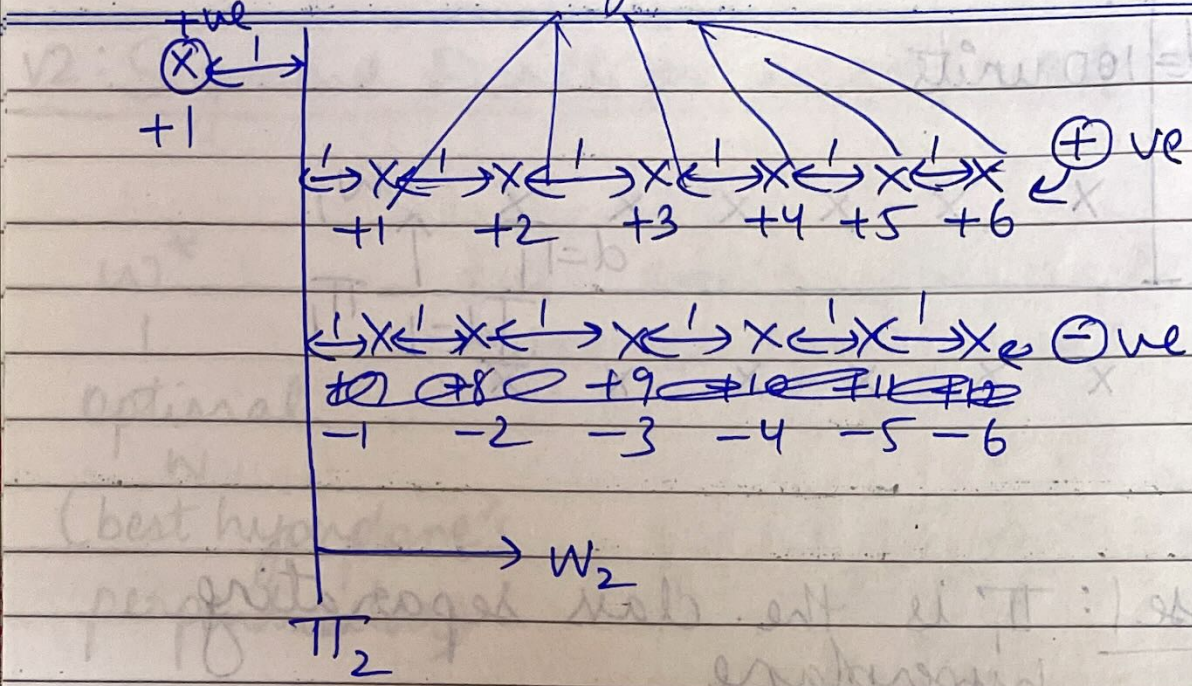
$+1+1+1+1+1+1$ (= -ve)

-100 (for -ve in +ve points area)

$= -88$

$y_i w_2^T x_i$

signed distance



$$\sum_{i=1}^{n} y_i w_2^T x_i$$

Sum all case 2 ~~of~~ Signed distance ⇒

⇒ 6 − 6 + 1

⇒ +1.

But our objective was to find w^* to find maximum signed distance.

$$w^*_{\Pi_1} = -88 < w^*_{\Pi_2} = +1.$$

So Π_2 is a my classifier. but it is doing a terrible job.

Accuracy in case 2:-

$\frac{1}{13}$

Accuracy in case 1:-

$\frac{12}{13}$

So π_1 is better than π_2.

It is happening ∵ we have an extreme outlier point. which is prefering π_2 as the hyperplane.

So, maximizing sum of distance is not outlier prone (outlier can impact)

So, we need to modify the optimization function w^*.

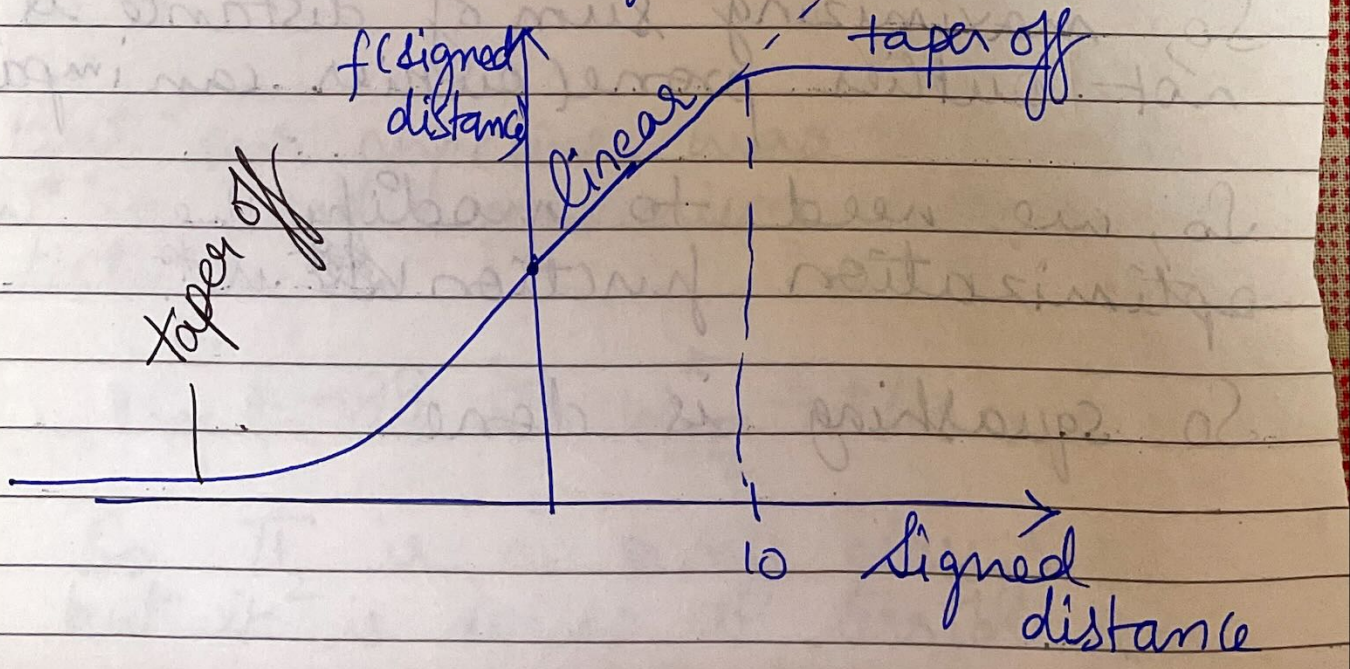
So squashing is done.

Instead of using signed distance, we use:

If signed distance is small use it as it is, but if large, make it a smaller value.

I want to come up with a function of signed distance f(signed distance) which we use when signed distance ↑ beyond a threshold say 10 the f(signed-distance) should taper off.

Below 10(threshold), func(signed-dist) should ↑ with signed distance.



So we should now solve this:-

$$\underset{w}{\text{argmax}} \sum_{i=1}^{n} f(\underbrace{y_i w^T x_i}_{\text{signed distance}})$$

This function → $f(y_i w^T x_i) = \sigma(x)$
This $\sigma(x)$ is sigmoid function.

$$\boxed{\sigma(x) = \frac{1}{1+e^{-x}}}$$ — Sigmoid function.

$0 \leq \sigma(x) \leq 1$ (x = $w^T x_i$)

$\sigma(0) = 0.5$, $\sigma(0)$ means $\sigma(\text{dist}=0) = 0.5$

So this becomes

$$\boxed{w^* = \underset{w}{\text{argmax}}\left(\sum_{i=1}^{n} \sigma(y_i w^T x_i)\right)}$$

☆ we chose σ ∵ it has probabilistic interpretation. dist = 0 means pt. lies on ∏
so prob = 0.5 r/y otherwise prob when it tow opp and prob ↓

σ func. transforms signed distances

So $$\boxed{w^* = \underset{w}{\text{argmax}}\left(\sum_{i=1}^{n} \frac{1}{1+\exp(-y_i w^T x_i)}\right)}$$

— comes from e^{-}

Sigma func. is easy to differentiation and probabilistic interpretation.

It is squashing as it is squashing value/dist$(-\infty, \infty)$ to dist$(0,1)$.

Any argmax or argmin func. needs to be differentiable to solve the optimization problem.

V3: Mathematical Formulation of Objective Function

Writing e^{-x} and $\exp(-x)$ is same.

Actual Optimization problem -

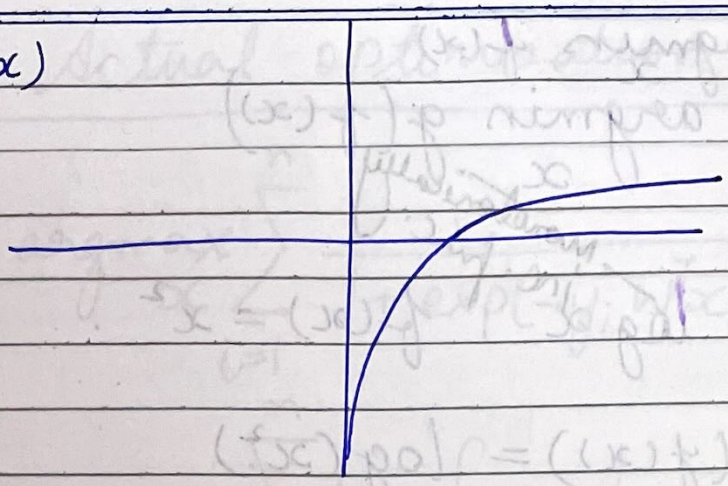
$$W^* = \underset{W}{\operatorname{argmax}} \sum_{i=1}^{n} \frac{1}{1+\exp(-y_i W^T x_i)}$$

Monotonic function Examples

As $x \uparrow$ if $g(x) \uparrow$, such a func. is called monotonically increasing function.

If $x_1 > x_2$ then $g(x_1) > g(x_2)$ then $g(x)$ is said to be monotonically increasing function for $x > 0$.

$\log(x)$



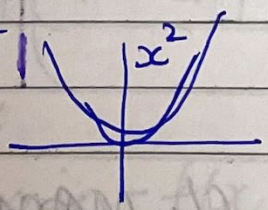
$\log(x)$ is only defined for only $x > 0$, not for negative values.

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

eg Optimization problem :-

$$x^* = \underset{x}{\operatorname{argmin}}\ x^2$$

Best x



$f(x) = x^2$

$x^* = 0$ as this is the min. value.

x^2 is not monotonically increasing function.

~~x^2 has 2 parts :-~~
~~x^2 is monotoni~~

Monotonically increasing when $x > 0$ | $x \uparrow g(x) \uparrow$

" decreasing " $x < 0$ | $x \uparrow g(x) \downarrow$

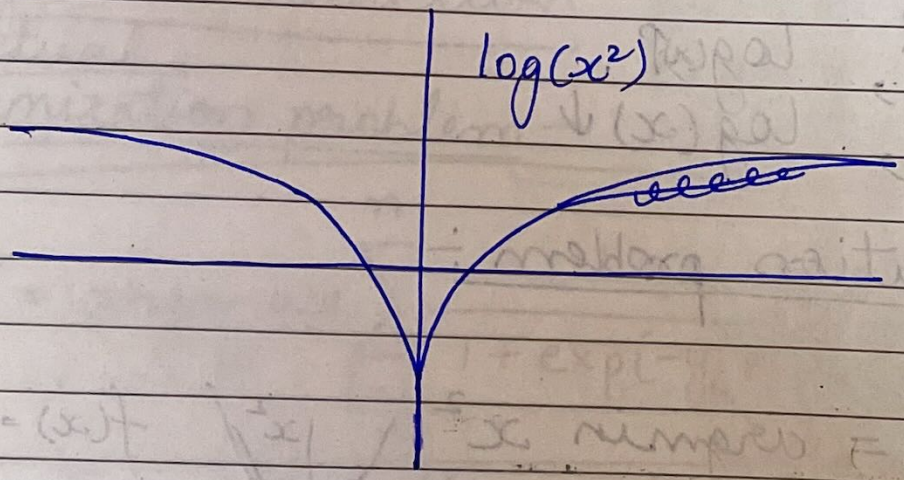
$x^* = \underset{x}{\operatorname{argmin}}\, f(x)$

$x' = \underset{x}{\operatorname{argmin}}\, g(f(x))$

$g(x) = \log x$ (monotonically inc. func.) $\quad f(x) = x^2$.

So $g(f(x)) = \log(x^2)$

Claim $\Rightarrow x^* = x'$ as $g(x)$ is monotonic function.



Property of monotonic function :-

If $g(x)$ is a monotonic function:-

$$\underset{x}{\operatorname{argmin}}\, f(x) = \underset{x}{\operatorname{argmin}}\, g(f(x))$$

OR

$$\underset{x}{\operatorname{argmax}}\, f(x) = \underset{x}{\operatorname{argmax}}\, g(f(x))$$

contd. Actual optimization problem

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

$$w^* = \underset{w}{\operatorname{argmax}} \sum_{i=1}^{n} \log\left\{\frac{1}{1+\exp(-y_i w^T x_i)}\right\}$$

We know $\log\left(\frac{1}{x}\right) = -\log x$.

So,

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

Actual optimization problem

Rule in Optimization:

Maximizing $f(x) = \underset{x}{\operatorname{argmin}} -f(x)$

So, $\underset{w}{\operatorname{arg\,max}} -f(x) = \underset{x}{\operatorname{argmin}} f(x)$

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

↕ geometrically.

LR optimization problem

So in this formulation, we assumed y_i to be either +1 or −1 but not 1 or 0. Not much outlier problem.

Probabilistically,

$$w^* = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{n} -y_i \log p_i - (1-y_i) \log (1-p_i)$$

where $p_i = \sigma(w^T x_i)$

Here $y_i = 1$ or 0:
not $+1$ or -1.

V4: Weight Vector

The optimal w^* is the weight vector.

This w^* is also a d-dim vector.

So
$w^* \in \mathbb{R}^d$ as $x_i \in \mathbb{R}^d$

$w^* = \langle w_1, w_2, \ldots, w_d \rangle$
Also $\langle f_1, f_2, \ldots, f_d \rangle$

If $w^T x_q > 0$ then $y_q = +1$

Why $w^T x_q < 0$ then $y_q = -1$

To find prob. $\sigma(y_i w^T x_i)$ gives prob whether $P(y_i = +1)$ or $P(-1)$

Can't do multiplication of w and x if they are not of same dimension

For every feature there is a weight associated with it.

Interpretation of w^*

Cases :-

1) If w_i is +ve corresponding to f_i;

as $x_{q_i} \uparrow$, then $w_i x_{q_i} \uparrow$

So $\sum_{i=1}^{d} w_i x_{q_i} \uparrow$

Then $\sigma(w^T x_q) \uparrow$

Then $P(y_q = +1) \uparrow$

Case 2 :- If w_i is -ve

If $x_{q_i} \uparrow$, $w_i x_{q_i} \downarrow$

Then $\sum_{i=1}^{d} w_i x_{q_i} \downarrow$

Then $\sigma(w^T x_{q_i}) \downarrow$

Then $P(y_q = +1) \downarrow$

So $P(y_q = -1) \uparrow$ as $P(y_q = +1) = 1 - P(y_q = -1)$

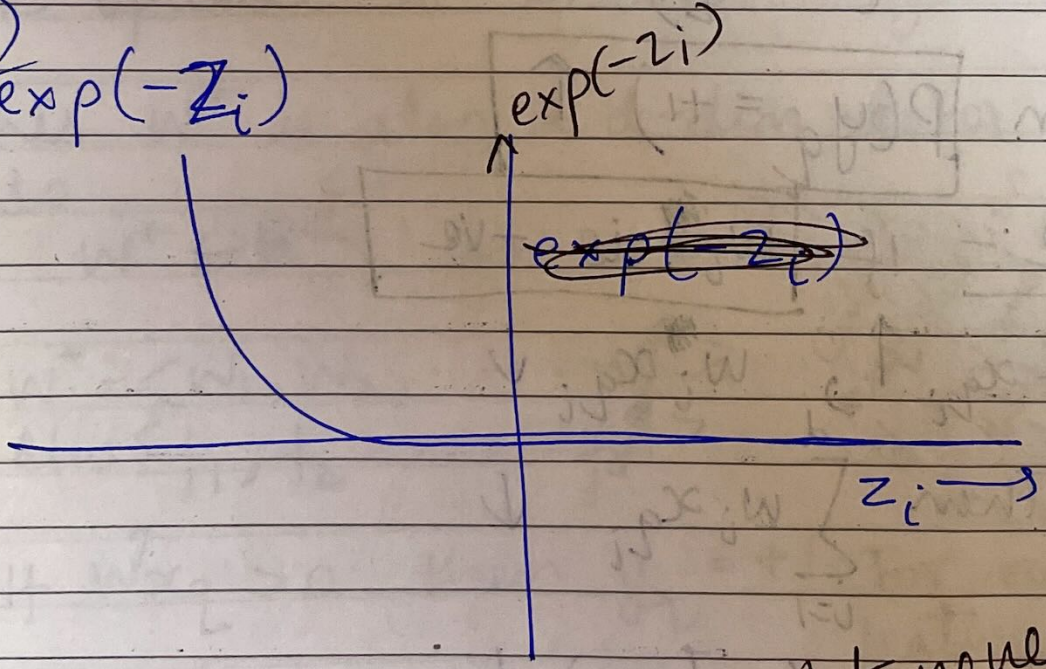
V5: L2 Regularization: Overfitting and Underfitting

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

Let $z_i = y_i w^T x_i$

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

1) $\exp(-z_i)$



$\exp(-z_i) \geq 0$ ∴ plot never crosses x-axis and comes below.

As $\log(1) = 0$
So adding something to $\log(1+\exp(-z_i))$ must be greater than or equal to 0.

2 $\log(1+\exp(-z_i)) \geq 0$

If each z_i is ≥ 0, then the whole term

$$w^* = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{n} \left(\log(1+\exp(-z_i)) \geq 0 \right)$$

Minimal value of $\sum_{i=1}^{n} \log(1+\exp(-z_i))$ is 0.

NOW,

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

Case 1: If z_i is +ve

As $z_i \longrightarrow +\infty$, $\exp(-z_i) \longrightarrow 0$

$\log(1+\exp(-z_i)) \longrightarrow 0$ $\because \log(1) = 0$

If z_i = +ve, all points are correctly classified.

If I pick w s.t.

a) All training points are correctly classified by w

b) $z_i \longrightarrow +\infty$

then best w is found.

Perfect fitting on training data means overfitting.

But what if some points are outliers?

① Then the model is overfitting.

② $w_i \longrightarrow +\infty$ or $-\infty$

ie, w_i are getting very large.

then minima value becomes 0.

We missed one key aspect while constructing the optimization problem. that W is a normal. which means that $W^TW = 1$ but we didn't use it anywhere So we use regularization in optimization.

$$w^* = \underset{w}{\operatorname{argmin}} \left[\underbrace{\sum_{i=1}^{n} \log(1+\exp(-y_i w^T x_i))}_{\text{loss term}} + \underbrace{\lambda w^T w}_{L2 \text{ regularization term}} \right]$$

where $w^T w = \sum_{j=1}^{d} w_j^2$

which can also be written as

$$\|w\|_2^2$$

↳ Square of L_2 norm of W.

We minimize both loss + regularizer.

Regularization term is ~~making~~ avoiding W_i term to go to either $+\infty$ or $-\infty$

There is a tug of war b/w loss & regularizer term.

λ = hyperparameter (not given to us)

If $\lambda = 0$, means no regularizer so, overfit in training data high variance.

If λ = ~~∞~~ large, underfit in training high bias no influence of data. loss term only regularizer will be there.

V6: L1 regularization and Sparsity

$$W^* = \underset{W}{\operatorname{argmin}} \sum_{i=1}^{n} \log(1+\exp(-y_i W^T x_i)) + \lambda \|W\|_2^2$$

$\sum_{i=1}^{n} \log(1+\exp(-y_i W^T x_i))$ — logistic loss

$\lambda \|W\|_2^2$ — L2 regularization

Now, instead of L2 regularization, we can also use L1 regularization.

$$\|W\|_1 = \sum_{i=1}^{d} |W_i|$$

L1 norm

absolute means only +ve value.

Even if $W_i \to +\infty$ or $-\infty$ absolute makes it +ve.

So, $W^* = \underset{W}{\operatorname{argmin}} \left(\text{logistic loss for training data} \right) + \lambda \|W\|$

Does same work as L2

Advantage of using L1 or L2 regulari.

Sparsity means, $W = \langle w_1, w_2 \ldots w_d \rangle$

Solution to LR is said to be sparse if many w_i's ~~or w^*~~ are ~~said to be~~ zero.

Then solution to Logi Reg is said to be sparse.

Advantage of L1 regularization

1) On use of L1 regularization, $w_i = 0$ for all w_i corresponding to less important features.

2) In L2 regularization, w_i's become a small value for less important features but not necessarily zero.

Elastic Net

Using both L1 & L2 regularization.

$$w^* = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i w^T x_i\right)\right) + \lambda_1 \|w\|_1 + \lambda_2 \|w\|_2^2$$

2 hyperparameters :-
$\lambda 1$ and $\lambda 2$.

Zero:— 3 ways to derive logistic regression:—
1) Geometry & simple algebra.
2) Probability -
3) Loss minimization.

V7: Probabilistic Interpretation: Gaussian Naive Bayes + Bernoulli distribution = LR

(Sec. 3.1 Tom Mitchell—@CMU.edu)

From geometry we get,

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

y is either $+1$ or -1

(In Naive Bayes, if features are real valued, then conditional probabilities are Gaussian distributed and $y_i = +1$ or 0 are Bernoulli random variables (like in coin-toss))

From probabilistic derivation, we get :—

$$w^* = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{n} -y_i \log p_i - (1-y_i)\log(1-p_i) + \text{reg}.$$

Logistic Reg = Gaussian Naive Bayes + Bernoulli random variables

where $y_i = +1$ or 0.

and $p_i = \sigma(w^T x_i)$

Case 1: $y_i = +ve$ geom: $y_i = +1$ prob: $y_i = +1$

geom: $\log(1+\exp(-w^T x_i))$

prob: $-1 * \log \dfrac{1}{1+\exp(-w^T x_i)}$ $\left[-\log(x) = \log\left(\frac{1}{x}\right)\right.$

$= \log(1+\exp(-w^T x_i))$ (— of log with reciprocal gets inverted)

(second term is $1 - y_i = 1-1 = 0$ so 2nd term is 0)

Case 2: $y_i = -ve$

geom $= y_i = -1$

prob $= y_i = 0$

geom: $\log(1+\exp(w^T x_i))$

prob: $-(1-0)\log\left[1-\frac{1}{1+\exp(-w^T x_i)}\right]$

(1st term becomes 0)

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

Using $\boxed{-\log(x) = \log\left(\frac{1}{x}\right)}$, we get :-

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

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

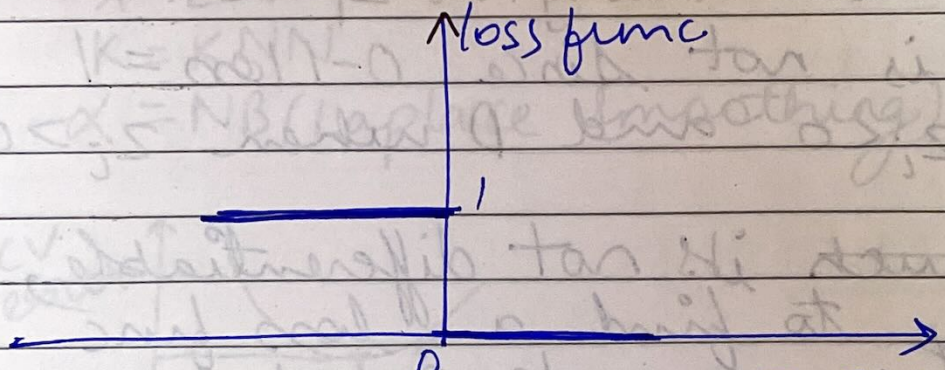
Using $\frac{1}{e^{-x}} = e^x$, we get.

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

V8: Loss Minimization Interpretation

Ideal loss func:-
+1 : Incorrectly classified point
0 : correctly classified point

$$w^* = \underset{w}{\text{argmin}} \left(\text{no. of incorrectly classified points} \right)$$



Such a loss function is called as 0-1 loss func.

$$w^* = \underset{w}{\text{argmin}} \sum_{i=1}^{n} 0\text{-}1 \text{ loss}(x_i, y_i, w)$$

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

To solve optimization in ML we use differentiation in calculus.

But a big issue with selecting this loss func. is it is non differentiable.

For a func. to be differentiable it must be continuous.
There is a discontinuity at $z_i = 0$.
But it is not since 0-1 loss = 1 when $z_i < 0$ and 0 when $z_i > 0$.

Since ~~such~~ its not differentiable, we try to find a loss func. which is close to 0-1 but differentiable.

One such approximation of 0-1 loss is logistic loss. for Logistic Regression.

Another approximation of 0-1 loss would be
* Hinge loss for SVM
* Square loss for Linear Regression
* Exponential loss for Adaboost.

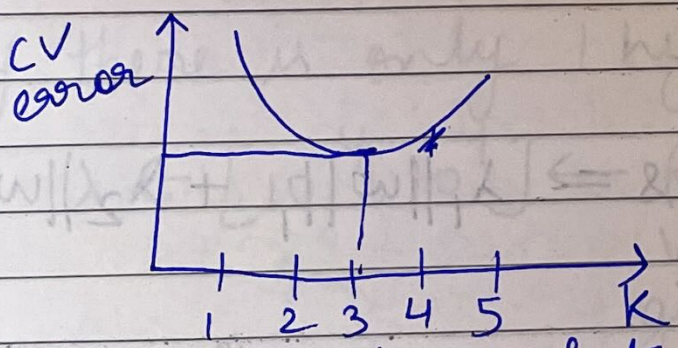
By just changing the loss func slightly you get many different ML models.

V9: Hyperparameter Search: Grid Search and Random Search

λ = hyperparameter (when doing regularization)
If $\lambda = 0$, overfitting, $\lambda = \infty$ underfitting.

So, How to determine the best λ.

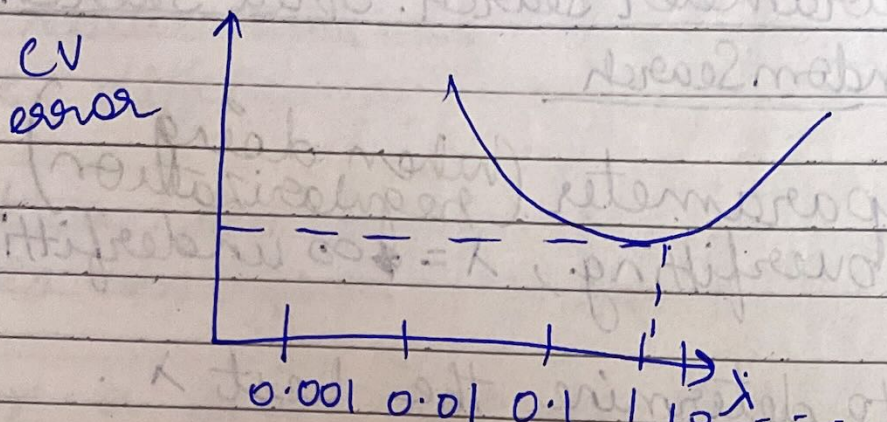
λ = LR
K = KNN
α = NB (Laplace Smoothing)



K in KNN is an integer $\{1, 2, 3 \ldots n\}$
λ in LR is a real number
$\lambda \in \mathbb{R}$, $\lambda = 0.1234$, etc.

Grid Search, it is some what brute force technique.

eg- $\lambda = [0.001, 0.01, 0.1, 1, 10, 100, 10K]$
OR
$\lambda = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$.

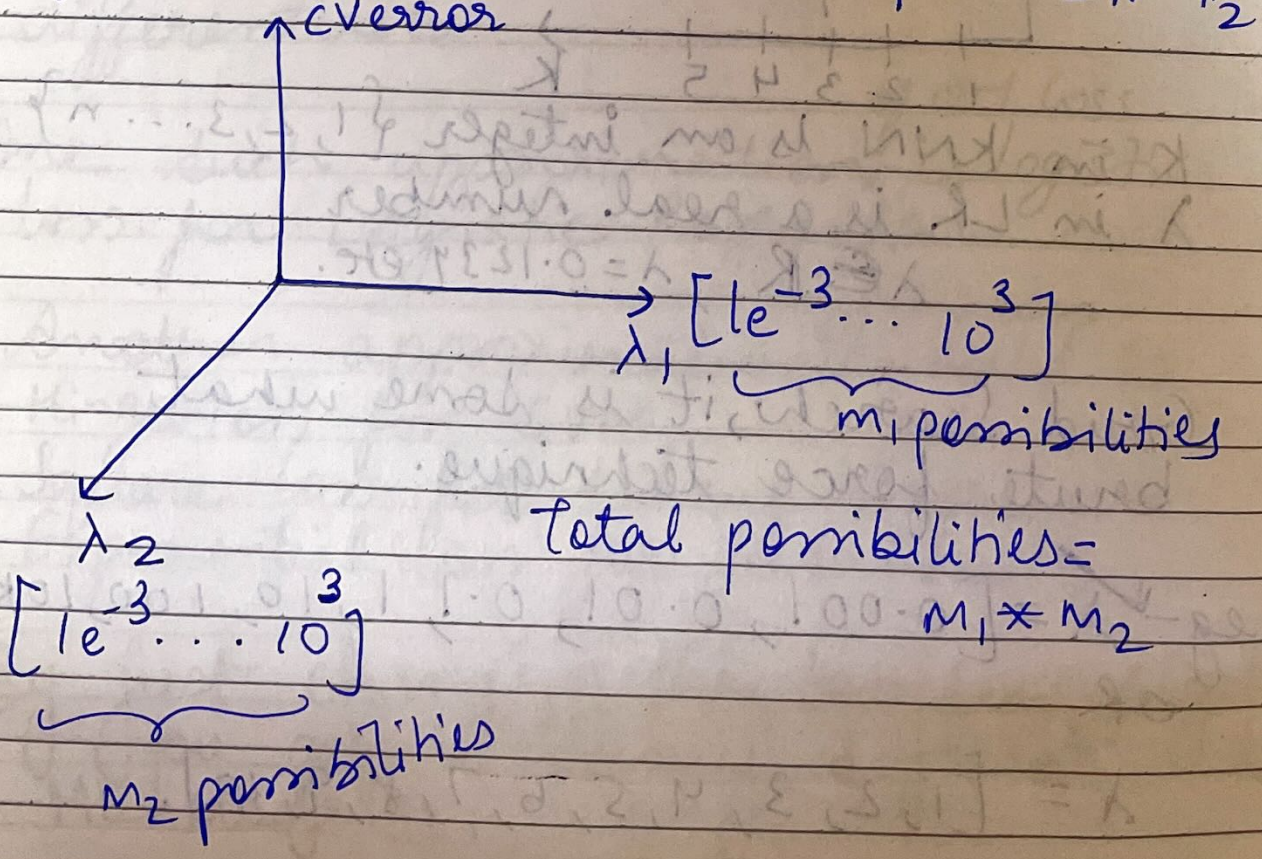


$\lambda = [1e^{-4} \dots 10^{4}]$

10^{-4}

Elastic Net

hyperparameters $= \lambda_1 \|w\|_1 + \lambda_2 \|w\|_2^2$



total possibilities = $m_1 * m_2$

Problem with GridSearch:-

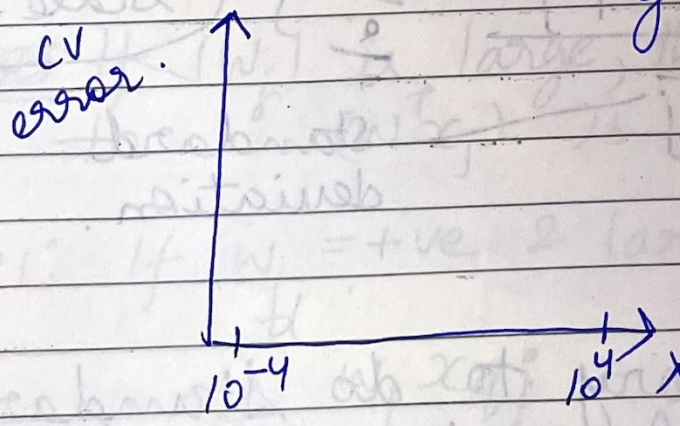
We have to train and compute m_i times where i is the no. of hyperparameter.

So grid search m_i hyperparameter search technique increases exponentially (-lly)
If you have k hyperparameters, You will have to train m^k hyper parameters

Random Search

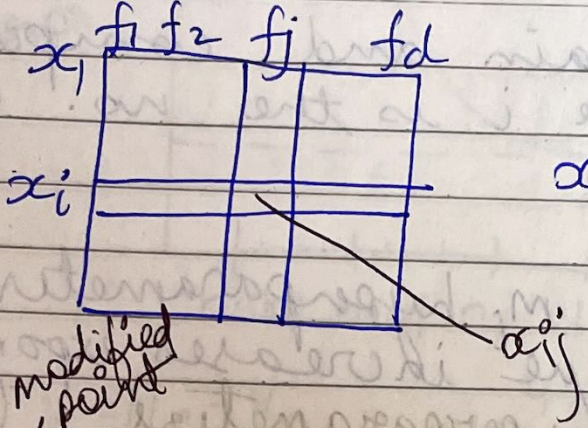
If there is only 1 hyperparameter:-

$\lambda \in [10^{-4}, 10^{4}]$ ← Randomly pick values in the given interval.



This technique is as good as grid search.

V10: Column Standardization



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

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

First we compute μ_j and σ_j of that column and we do Column Standardization. For each column we do standardization as different column could be on a different scale.

$x_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}$ — centering; scale.

σ_j — Standard deviation.

It is compulsary to do Standardi-zation in Logistic Regression.

Feature/Column Standardization is also called Mean centering and Scaling.

Mean centering = $x_{ij} - \mu_j$ sub

Scaling = σ_j division

Probabilistically, Logistic Regression is nothing but Gaussian Naive Bayes + Bernoulli

VII: Feature Importance and Model Interpretability

$f_1, f_2, f_3 \cdots f_j \cdots f_d$ — general feature

$w \rightarrow w_1, w_2, w_3 \cdots w_j \cdots w_d$ (feature weights)

Assumption: All features are independent

If this assumption is true.

We can get feature importance using w_j.

j we can take

LR: $\uparrow$ w_j's to determine feature importance.

$|w_j|$ = absolute value of weight corresponding to feature f_j.

If (w_j) is large, then contribution to $(w^T x_j)$ is large.

Case 1: If w_j = +ve & large.

$$\sum_{j=1}^{d} w_j x_{qj} = w^T x_q$$

It impacts Prob. of query point $P(y_q = 1)$

fraction

Case 2: W_j : -ve and large

$$\sum_{j=1}^{d} W_j x_{qj} \quad , \quad P(y_q = -1) \text{ is large.}$$

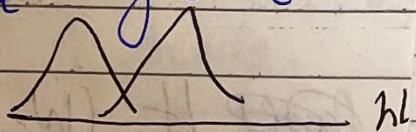
When something is large (+ve or -ve) its impact is large.
So, we can determine the important feature in LR through feature weights

eg → Predict gender : male & female
(+1) (-1)

① (hL) Hair length = $|W_{hL}|$ is large.
Then, W_{hL} = -ve

As $|W_{hL}| \uparrow$; $P(y_q = -1) \uparrow$

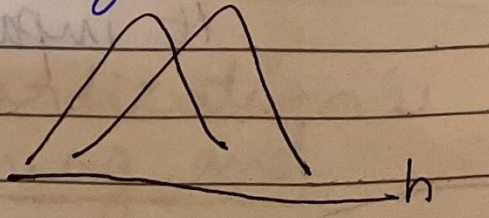
So hL will have a large negative weight value.



② Height ↑ = $|W_h|$ is positive

As $|W_h| \uparrow$; $P(y_q = +1) \uparrow$

W_h will get a medium positive weight.



Model Interpretability = Give reasoning when absolute weight of feature is large.

x_q

$y_q = +1$ or $y_q = -1$

V12: Collinearity of features or Multicollinearity

Feature Importance can be calculated using absolute value of weights of independent features only.

But collinearity/multicollinearity is calculated for non independent features..

f_i, f_j

s.t. if $f_i = \alpha f_j + \beta$

then f_i & f_j are collinear

Multicollinearity

Collinearity among multiple features

if f_1, f_2, f_3 & f_4 s.t.

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

then f_1, f_2, f_3, f_4 are multicollinear.

(All α are constant)

Q Why does $|w_j|$ not be useful as f.I. if features are collinear?

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

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

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

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

$$\tilde{w} \Rightarrow \tilde{w}^T x_q = x_{q_1} + 3x_{q_1} + 3x_{q_3} = 4x_{q_1} + 3x_{q_3}$$

Now weight vector is $= \langle 4, 0, 3 \rangle$

$w^* \Rightarrow$ Original feature vector $= \langle 1, 2, 3 \rangle$
$\tilde{w} = \langle 4, 0, 3 \rangle$

Both weight vector give same classifier

If features are collinear, then weight vector can change arbitrarily.

$\Downarrow$

$|w_j|$ cannot be used as f.I.

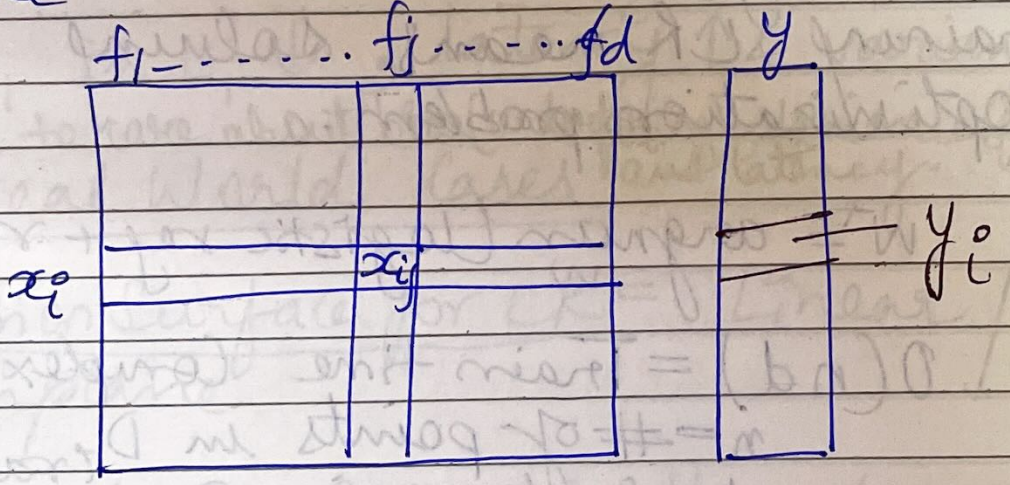
Earlier with $\langle 1, 2, 3 \rangle$, we felt $f_3 > f_2 > f_1$.
But now, $\langle 4, 0, 3 \rangle$ $f_1 > f_3 > f_2$ in terms of F.I.

Determine if features are multicolline -ar or not?

Ans → Perturbation Technique.

Perturbation technique

↑
Shake a little



$x_{ij} + \epsilon$ ← small noise/error from $N(\underset{\mu}{0}, \underset{\sigma}{0.10})$

Before adding noise calculate weight vector and after adding noise again calculate weight vector.

If W_i and $\tilde{W}_i$ differ significantly then data (features) are collinear.

Then you can't use $|w_j|$'s as feature importance.

IMP. Forward feature Importance can be used for any model not just KNN.

V13: Train & Runtime & Space Complexity

Training LR means solving optimization problem.

$$W^* = \underset{W}{\text{argmin}} \{\text{logistic reg} + \text{reg}\}.$$

$O(nd)$ = Train time Complexity.

n = # of points in D_{train}.

d = features in D_{train}.

We just have to store weight vector $W^* = \langle w_1, w_2 \ldots w_d \rangle$ in memory

Runtime Complexity

Time = $O(d)$

d = # of features.

Space = $O(d)$

d = # of feature.

If d is small, the LR is V. good for low latency applications.

w_j's corresponding to imp. features $\neq 0$.

In case of high dimensionality, we can make use of L1 reg. to create sparsity in W so as to reduce calculations during runtime.

As λ_1 in L_1 ↑ ⇒ bias ↑ Underfitting
∴ latency ↓

λ_1 in L_1 ↓ ⇒ latency ↑ & variance ↑

You have to care about bias (underfitting), Variance (overfitting) and latency.

V14: Real World Cases

Decision Surface for LR = Linear Hyperplane

Assumption = Data is linearly or almost linearly separable.

Feature Imp & Interpretability – we can use $|w_j|$ if features are not multicollinear.

Imbalanced data = Upsampling / Downsampling

Outliers = Less impact due to outliers due to sigmoid function. but not completely avoided.

(1) So, first calculate W^* from D_{train}

(2) Calculate $W^{*T} x_i$ on each x_i using W^{*T}. You get distance from plane π to x_i

(3) Remove points which are very far away from π from

D_{train} and you get D'_{train}

(4) Calculate $\tilde{W}^*$ using D'_{train}

So, $\tilde{W}^*$ is the final solution.

Missing values = Imputation method like mean, median, mode.

Multiclass = One vs. Rest ← Typically

(1) Maximum Entropy models → Extension to LR.

(2) Softmax classifier in Deep learning

(3) Multinomial LR.

Similarity Matrix = Regular LR does not work on similarity matrix so we use Kernel & Logistic Regression (Research paper)

Best Case = If data is almost linearly separable and (L1 Regn) Sparsity the LR works best.

Large Dimensionality = chances of linearly separable is high.

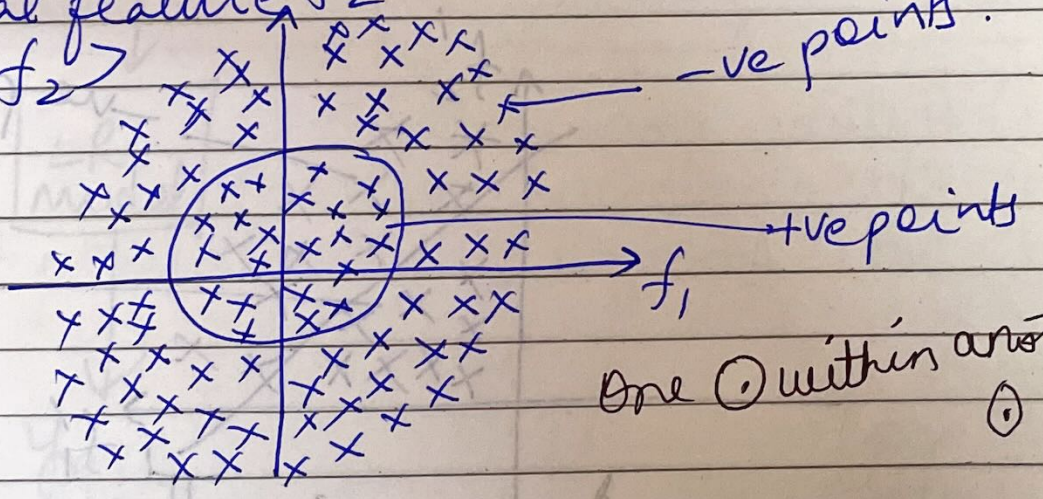
But if you want a low latency system, esp, when d is large then use L1 regularization.

V15: Non-linearly separable data and feature engineering

eg1

Original feature f_2

$\Rightarrow \langle f_1, f_2 \rangle$



Can we use Logistic Regression to separate the classes.

→ Drawing a plane in the space of f_1 and f_2.

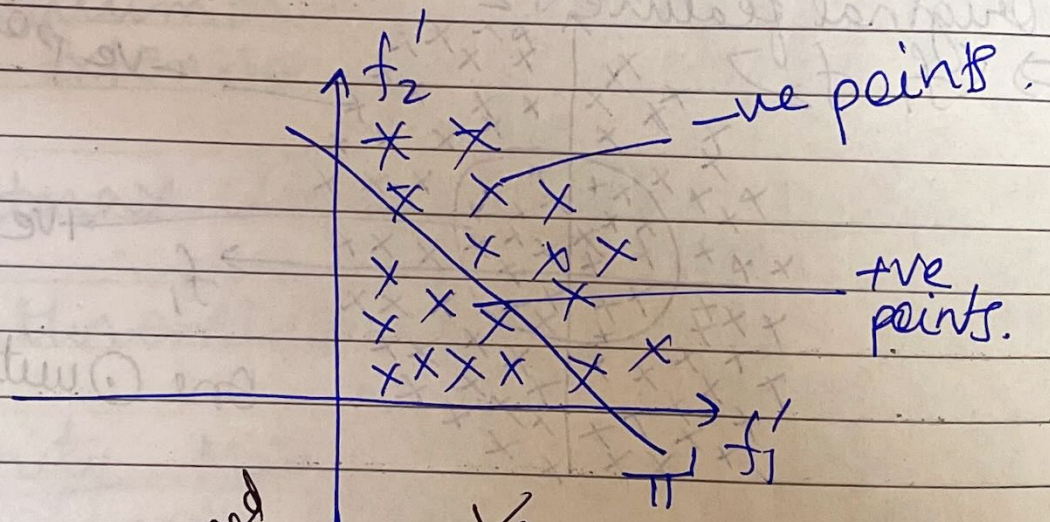
→ Cannot separate +ve and -ve pts by drawing a plane.

So we do feature transformation or feature engineering and we modify the features to a completely different space.

Engineered features: f_1', f_2'

$$f_1' = f_1^2$$

$$f_2' = f_2^2$$



We have engineered features, also we will do same feature engineering on query points i.e.

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

$$x_{i1}' = x_{i1}^2$$

$$x_{i2}' = x_{i2}$$

$x_q = \langle x_{q_1}, x_{q_2} \rangle$

↓ FE/FT

$x'_q = \langle x'_{q_1}, x'_{q_2} \rangle$

$x'^2_{q_1}$ $x'^2_{q_2}$

↓

LR model

↓

y_q.

Q How did you know which transform to apply?

Comes with practice.

These were 2 concentric circles. So we use equation of a circle using geometry.

quadratic

$$f_1^2 + f_2^2 = r$$ (not linear in f_1 & f_2)

But linear in f_1^2 & f_2^2.

So $$f_1' + f_2' = r$$ — linear

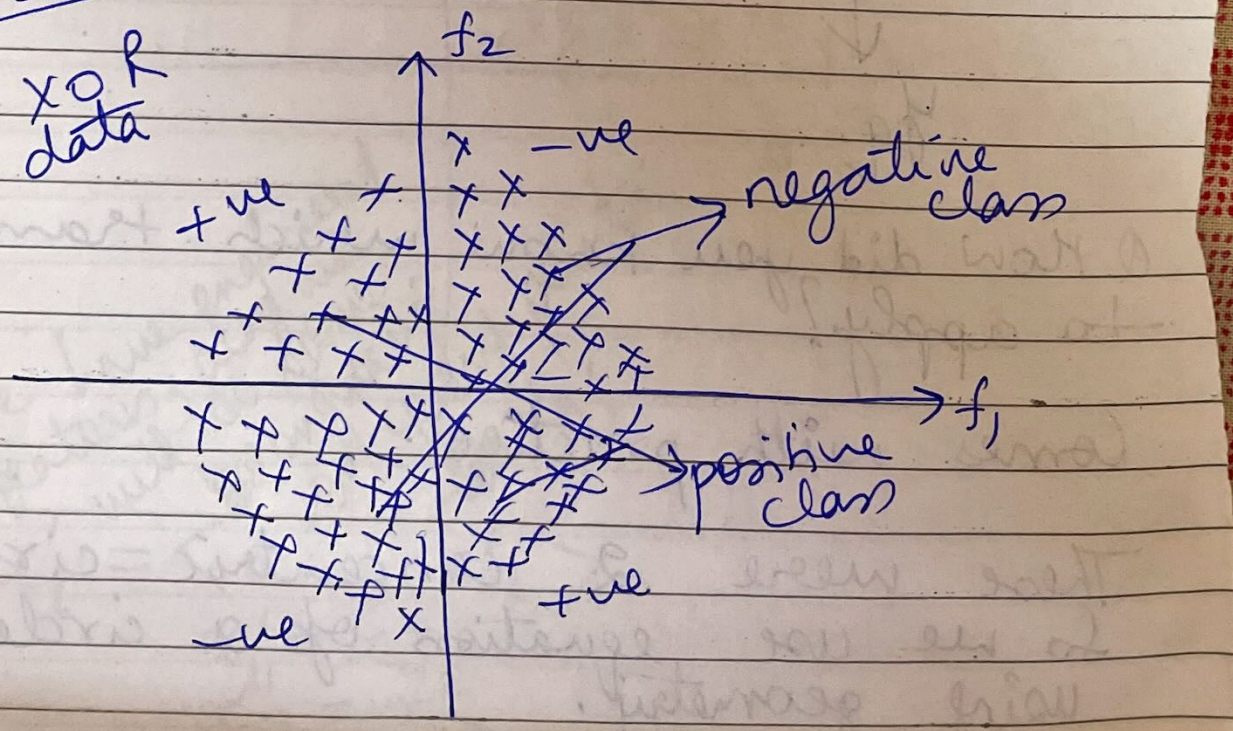
Equation of line :- $ax + by + c = 0$.

Most important aspect of applied AI/ML :-

1) Feature Engineering
2) Bias-variance tradeoff
3) Data Analysis & Visualization technique.

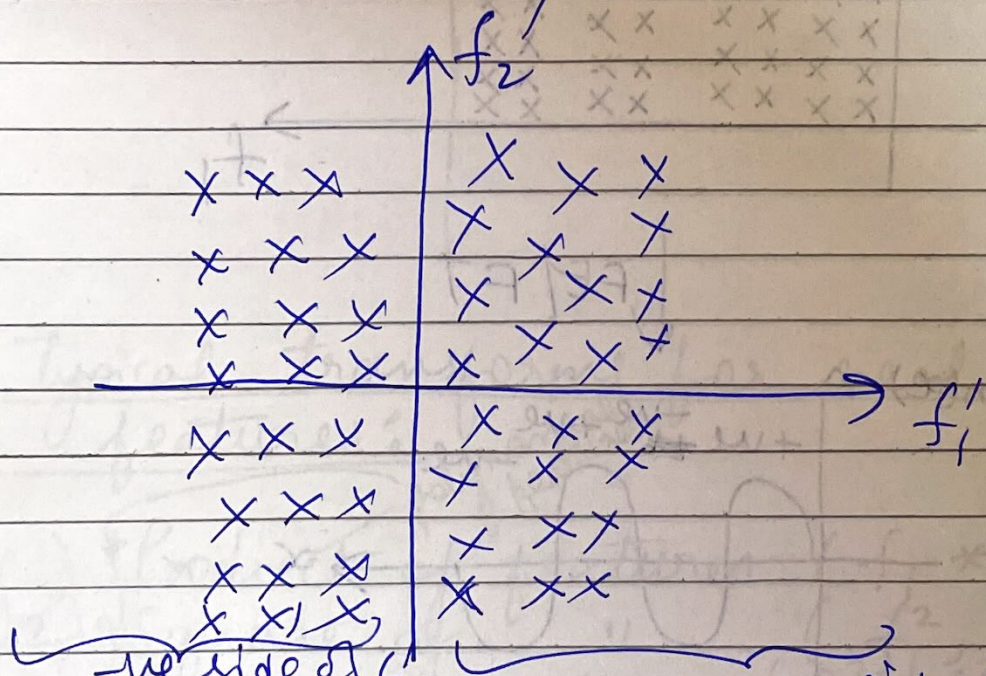
eg 2

XOR data



XOR	0	1
0	0	1
1	1	0

FE/FT

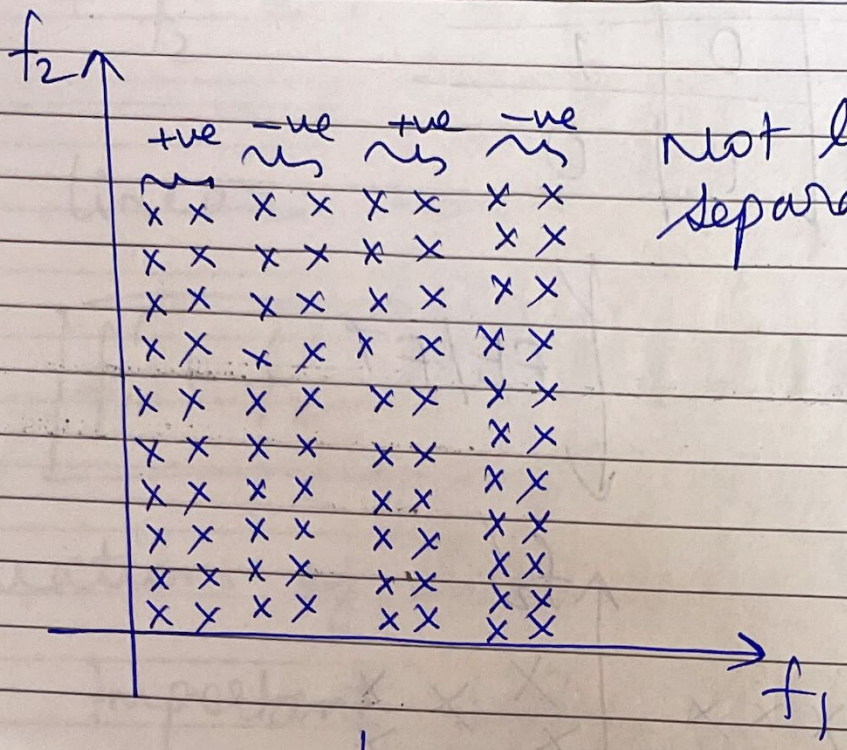


$f_1' = f_1 * f_2$ — FE/FT.

$f_2' = f_2$

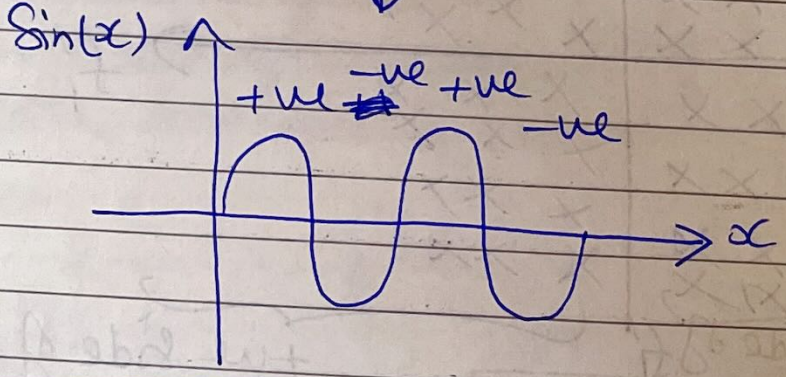
They are now linearly separable.

eg 3



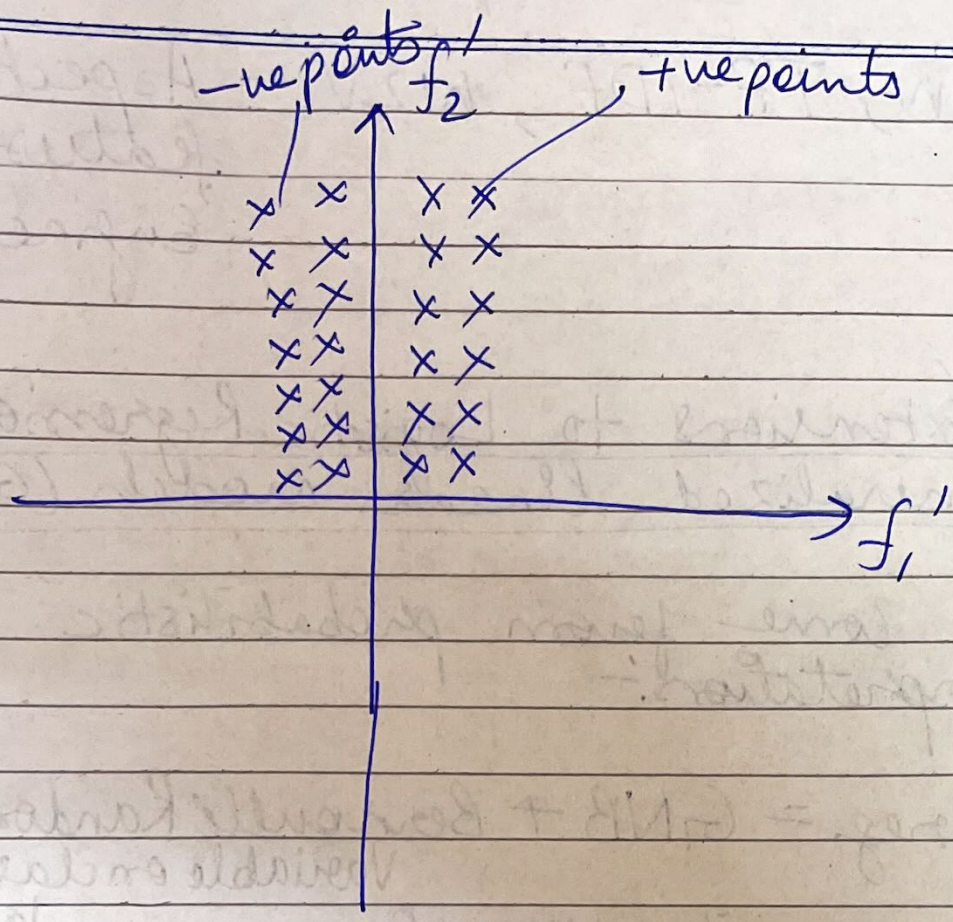
Not linearly separable

FE/FT



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

FE/FT



Typical transforms for real valued features:

polynomial features

Poly words of

1) Product of features $(f_1 * f_2)$
2) Square of " (f_1^2, f_2^2)
3) Cube of " (f_1^3, f_2^3)
4) Trigonometric functions
   $\sin(f_1)$, $\cos(f_1)$
5) Polynomial trigonometric functions:
   $\sin(f_1) * \cos(f_2)$
   $\sin^2(f_1)$
6) Boolean features : OR, AND, XOR
7) $\log(f_1)$, e^{f_1}, etc.

In Sklearn Logistic Regression $C = \frac{1}{\lambda}$

BOW, TF-IDF, W2V — Aspects of feature Engineering.

VII. Extensions to Logistic Regression :- Generalized linear models (GLM)

They come from probabilistic interpretation :-

log reg = GNB + Bernoulli Random Variable on class labels

change it slightly :-

① Multinomial LR :-
Distribution changes from Bernoulli to Multinomial dist.
Useful when you have multiclass classification.

② Linear Regression (Actual Reg. tech.)

$P(y|x) \sim N(\mu, \sigma^2)$ Gaussian assumption $y_i \in \mathbb{R}$

③ Poisson Regression :-
for Poisson distributed.
If you want to predict counts. Say predict how many times a machine fails.
(for count prediction)