27.8 LOSS MINIMIZATION INTERPRETATION :-

* From optimization problem

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

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

→ If we want "to build an Ideal optimization model Then we try to minimize the no of incorrectly classified points."

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

→ As the whole "classification" is all about find the correct parameter that minimizes no of incorrectly classified points.

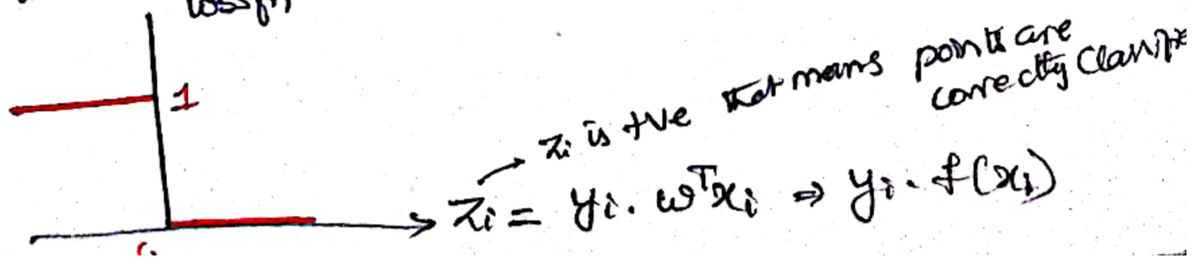
→ If we define a loss function for ex:

+1 : for incorrectly classified

0 : correctly classified.

as we always try to min the loss

So If we take ideal loss fn



loss fn
1
z_i is +ve that means points are correctly classified
$z_i = y_i \cdot w^T x_i \Rightarrow y_i \cdot f(x_i)$

→This is also called as 0-1 loss fn.

~~So If we have a loss fn like This~~

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

So If we have a loss fn like This we are trying to minimize w such that

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

so if given x_i, y_i & w we can compute z_i ~~0-1 loss~~ if we can compute z_i, given the 0-1 loss fn. Such that

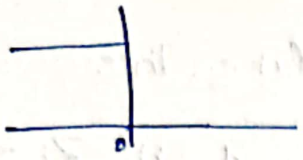
$$\text{0-1 loss fn}(z_i) = \begin{matrix} 1 \text{ if } z_i < 0 \\ 0 \text{ if } z_i > 0 \end{matrix}$$

→ One of the problem with 0-1 loss fn is (as we need to solve optimization problems in ML the functions needs to be differentiable then only we can do some much operations on optimization problem)

from basics we know the function can differentiable if it is ~~on~~ continous only.

as we see 0-1 loss function is discontinous

@ $z_i = 0$

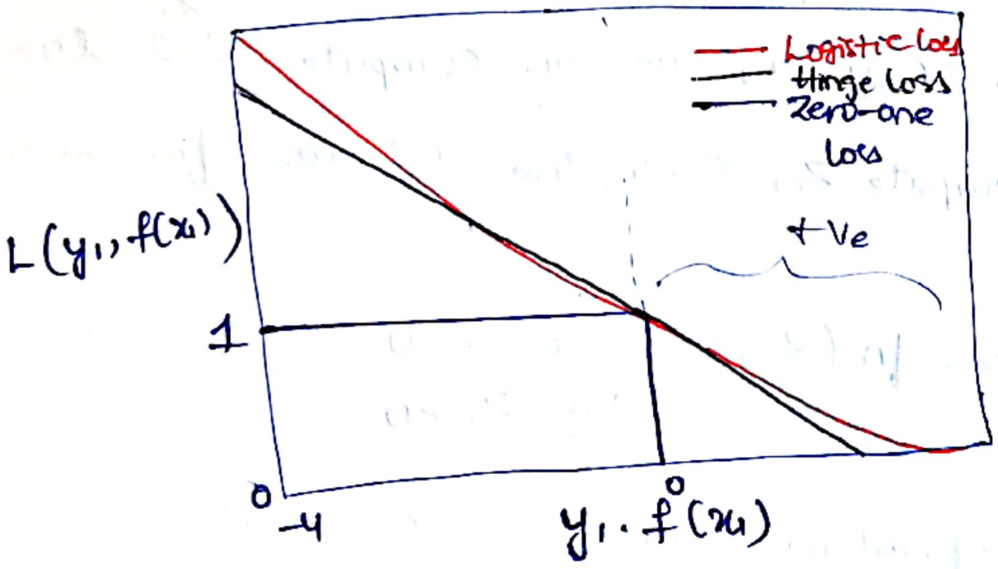


So it is not differentiable.

This is the problem.

So to avoid This problem, as it is not differentiable we will try to approximate it.

we can try many approximations one such approximation is called Logistic loss



→ So when we use Logistic loss as an approximation to zero-one loss we get Logistic Regression.

→ lly when we use Hinge loss as an approximation to zero-one loss we get other algorithms called Support Vector Machines.

→ lly exponential loss $\xrightarrow{\text{od loss}}$ Ada boost

→ squared loss $\xrightarrow{\text{0-1 loss}}$ linear Regression

→ So we need to check every possibility of λ_1 & λ_2 values to get the best hyperparameter.

→ So If There are m_1 values [$< m_1 >$] in the when evaluating λ_1 : 1 hyperparameter. we need to perform m' times of searching across all the values

lly if There are 2 hyperparameters

λ_1, λ_2 : 2 hyperparameters : $m_1 \times m_2$ m^2

lly if 3 hyperparameters

$\lambda_1, \lambda_2, \lambda_3$: 3 hyperparameters : $m_1 \times m_2 \times m_3$ m^3

⋮

if K hyperparameters m^K

So as There are " no of hyperparameter increase the # no of times the model needs to be trained Increases exponentially."

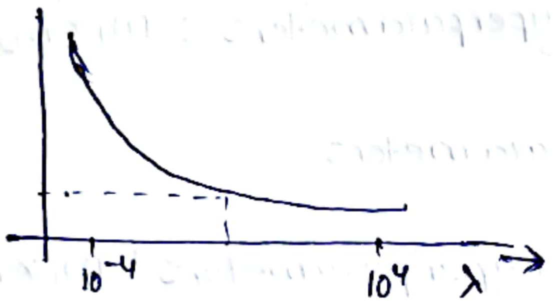
→ Even though. there ~~are~~ we are only 2 hyperparameters in Logistic Regressions there are Cases in deep learning where more than 2 hyperparameters would be needed That's why Gridsearch is not best technique

→ we use another technique which is as good as for Grid Search called Random Search

Random Search

In "Random Search we randomly pick values in the given Interval"

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



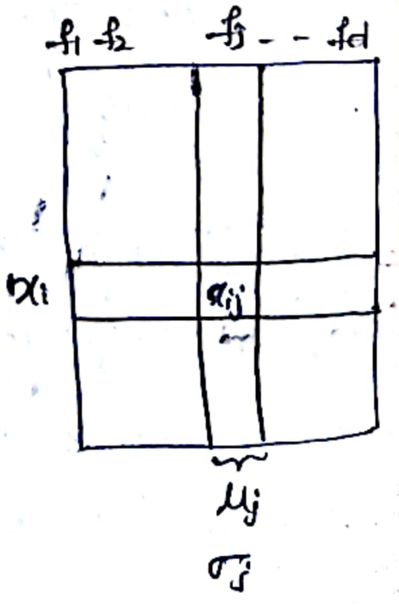
27.10 COLUMN STANDARDIZATION

→ for a dataset given with x_i as a d-dimensional point and with d features.

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

→ In Column Standardization we transform

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



27-9 HYPERPARAMETER SEARCH: GRID SEARCH AND RANDOM SEARCH:-

As for the logistic regression λ is the hyper parameter

and as we seen if $\lambda = 0 \Rightarrow$ overfitting

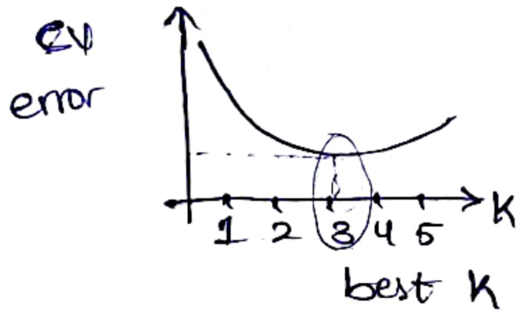
$\lambda = \infty \Rightarrow$ under-fitting.

as in case of

K: for KNN

and α: for NB (Laplace smoothing)

we use Cross Validation to find the best K



lly for Logistic Regression.

But there is a small problem that in

∵ k in KNN is an Integer $\{1, 2, 3, ---N\}$.

But ~~AA~~ the λ in Logistic Regression is a real number

$\lambda \in \mathbb{R}$.

so the values that can accomodate is infinity so

to avoid such problem as it is difficult to get a

Value of λ using General techniques There is called Grid Search technique.

→ **Grid Search :-**

Gridsearch is a brute force technique where people typically uses a grid or bunch of values to search for the best λ value.

Ex:- $\lambda = [0.001, 0.01, 0.1, 1, 10, 100, 1000]$

10 10 10 10 10

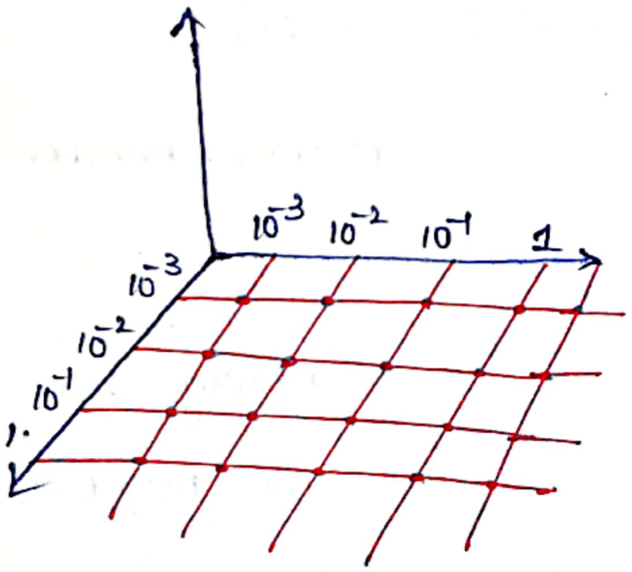
(or) $\lambda = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$

(or)

$\lambda = [10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^{2}, 10^{3}]$

Instead of one λ value if we considered elastic net!

Then $\lambda_1 \|w\|_1 + \lambda_2 \|w\|_2^2$



let

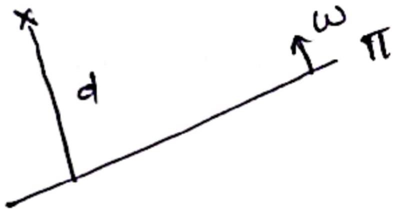
$\lambda_1 = [10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}]$

$\lambda_2 = [10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}]$

→ In KNN since we $\overset{\text{use}}{\wedge}$ distances b/w data points the features could be in different scale.

So whereever we are using distance we need to standardize the features.

→ Similarly in Logistic Regression also It is ~~made~~ mandatory to perform feature standardization before training on your data.



as logistic regression also deals with distances

→ column | feature standardization also called mean centering and scaling.

because

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

→ It is like centering the mean by subtracting

→ scaling.