① Types of learning: supervised and unsupervised

② Types of problems Regression Classification and clustering

③ Cost function, optimization - Gradient Descent algorithm.

④ Sampling, decision boundary, under-fitting and overfitting of models

⑤ Bias-Variance tradeoff

(Supervised).

Regression: Linear, Regularization problem - Ridge and Lasso.

Classification - Logistic Regression

(Unsupervised)

Intro-to-Artificial Neural Network

Model validation and selection

↳ Accuracy

Confidence interval

Confusion Matrix

Precision, Recall

Machine Learning (ML).

Types of learning

Supervised — Unsupervised

Regression — Classification

Accuracy
→ Matrix
interval

Machine Learning : Learning from Data

it is about predicting the future based on the past. ML algorithm's attempt to optimize along a certain dimension ; i.e minimize error or Maximize the likelihood of their prediction
→ optimizing an error/cost function.

Full Dataset.

Training set — Testing set

Training set — Validation set.

Testing set

train
Evaluate
predict.

training data
↓
train ML Algo.
↓
Model

(if not ok.) ← Evaluate. (OK) =

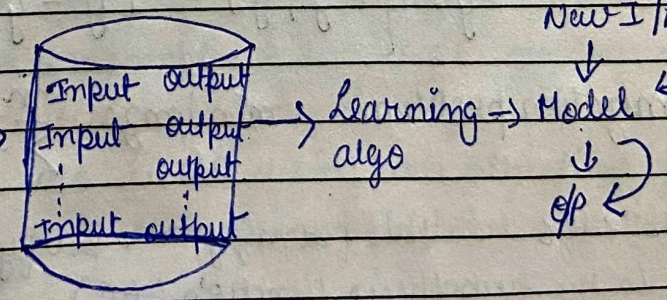
Input data → ML Algo → prediction

training is a process of making the system able to learn.
→ training and testing set come from same distribution.

① Supervised and Unsupervised:

Supervised.
→ Training Data
→ Both Input and outputs are known
→ Classification
→ Naive Bayes Algo.

Unsupervised. (Unlabelled data).
→ only Inputs
→ forms cluster's
→ Clustering, K-mean
eg: same cluster's,



Supervised learning eg; election Exit poll. climate.

Training data.

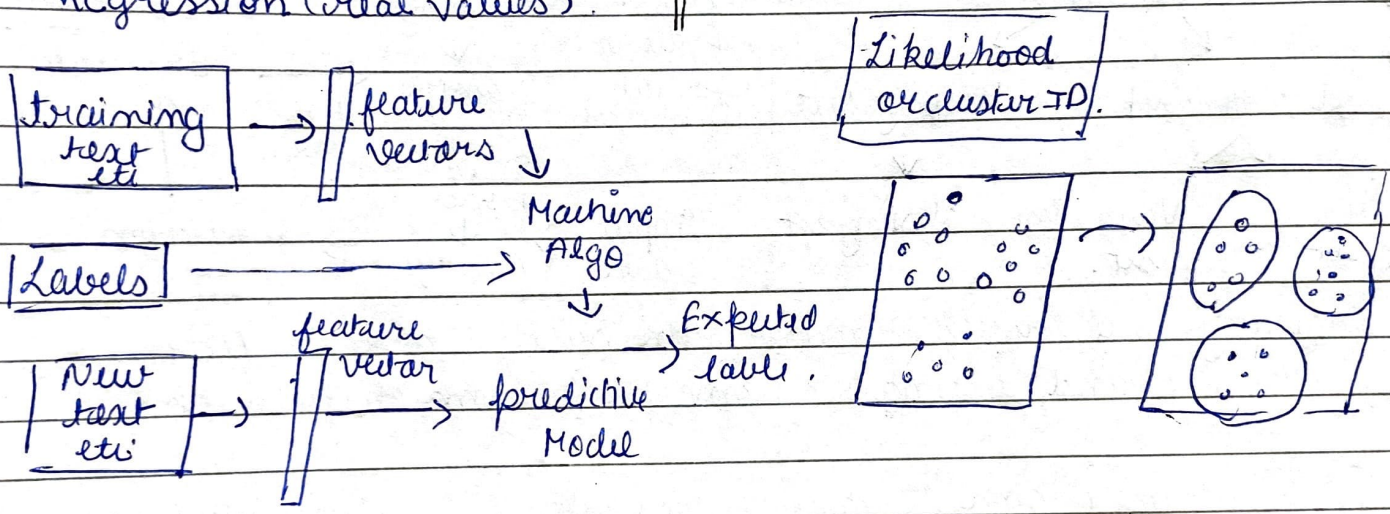
(Labeled data)

supervised		unsupervised
Classification	Regression	PCA
Logistic Regression	Linear	K-mean clustering
Naive Bayes	Ridge	Hierarchical clustering
Linear Discriminat (LDA)	Lasso	DB Scan.

→ KNN, Decision trees, Random forest

→
- Prediction
- → Classification (discrete labels), Regression (real values).

→
- Clustering
- $P(\bar{E})$ distribution estimation
- Finding association (features)
- Dimension reduction.



② Key Concepts (cost function, optimization) etc:

Loss function: $l : y \times y \to R_+$, measures the difference between a prediction and an actual output.

Eg: $\{(\hat{y}, y) = (\hat{y} - y)^2\}$.

ML optimization problem: $\underset{\theta}{\text{minimize}} \sum^{m} l(h_\theta(x, y))$

Every ML algo has this form, just specify
→ what is the hypothesis function (h_θ)
→ what is loss function (l)
→ How do we solve the optimization problem.

Loss function V.S Cost function

→ The loss function computes the error for a single training eg, while the cost function is the average of the loss functions of the entire training set.

→ if we have m training data like this $\{(x_1, y_1), (x_2, y_2) \ldots (x_m, y_m)\}$

y_1 = output of the model for training eg x_1

y_2 = expected output / true value for training eg x_1

→ Loss function $L(y_1, y_2)$ to defines the error/difference between y_1 and y_2 for the single training eg x_1.

→ Loss function refers to an error in model output for an individual sample.

* If we want to find loss over all the training eg. present in a training-set, we refer to it as the (cost function). i.e total or average loss over all training examples.

Key differences.

feature	Loss	Cost
Definition	Measures error for a single data point	Measures error (avg) over the entire dataset.
Scope	Individual sample	whole dataset
Usage	Evaluates single instance performance	used for optimization and parameter updates.
Example	MSE for one sample	MSE for all samples.

Loss / Error / Cost - objective function.

The more general scenario is to define an objective function first, which we want to optimize. This objective function could be to:

1. Min a (MSE), cost or loss function
2. Max the posterior probabilities (eg. Naive Bayes)
3. Max a fitness function (genetic programming)
4. Max the total reward / value function (Reinforcement learning)
5. Max information gain / Min child node impurities.
6. Max -log-likelihood.

Supervised learning (Regression):

Regression: Relationship b/w one dependent variable (y) and explanatory variable(s).

Models

→ it use the equation to set up a relationship of one or more numerical or categorial independent (Explanatory) variables.

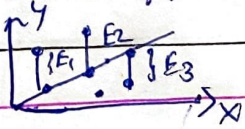
→ mainly for the prediction and Estimation of (Continous numeric variables).

Regression Model
- 1 Explanatory
  - Simple
    - Linear
    - Non linear
- 2+ Explanatory
  - Multiple
    - linear
    - non linear.

Linear Equation:
A linear relationship between the predictor / independent variable (x) and the response / dependent variable (Y) is assumed. $y = \beta_0 + \beta_1 * x$ or $y = mx + c$

Least square: 'Best fit' means difference between the actual y value and predicted $\hat{y}$ values are a min.
→ positive differences off-set negative. (So square errors!)

$$\sum (y_i - \hat{y})^2 = \sum (E_i)^2$$

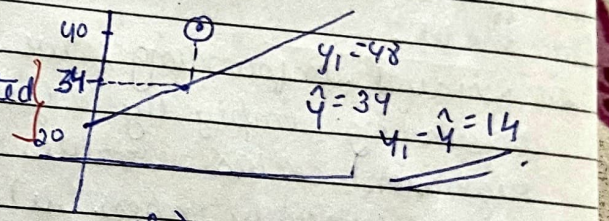


Assumptions for linear regression:

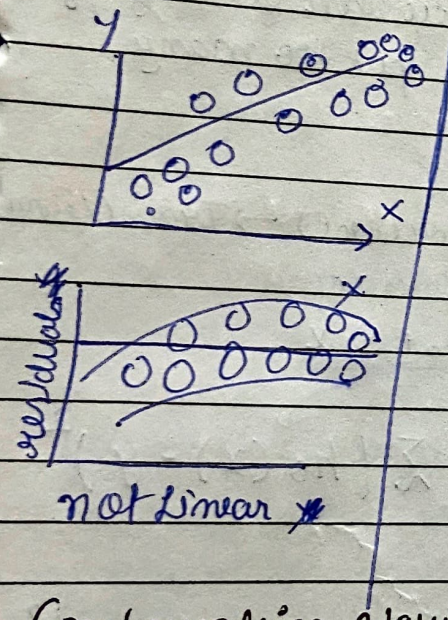
- Linearity : x and y must be linear.
- Independence of error
- Normality of error
- Equal Variance.

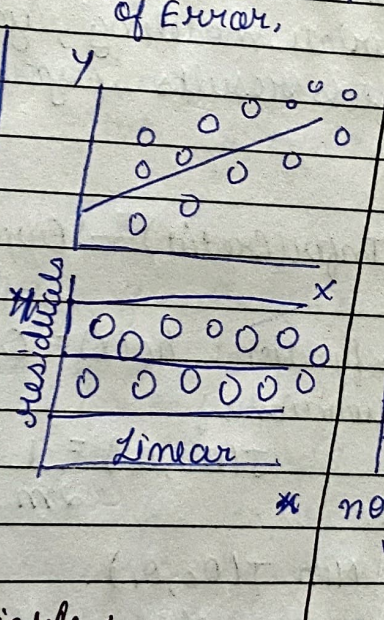
Residual Analysis. $e = y_i - \hat{y}$.

Residual = Observed − predicted

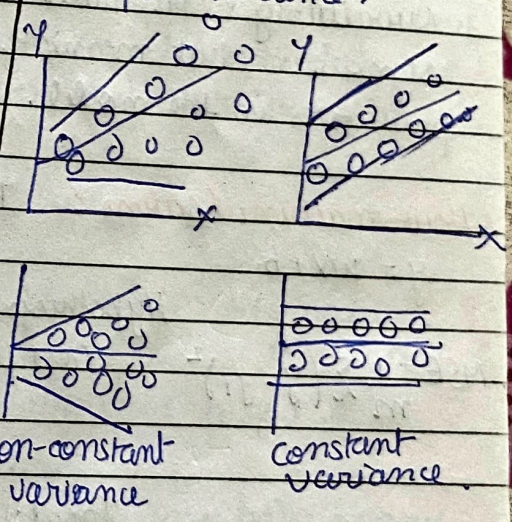


(a) Linearity and Independence of Error,

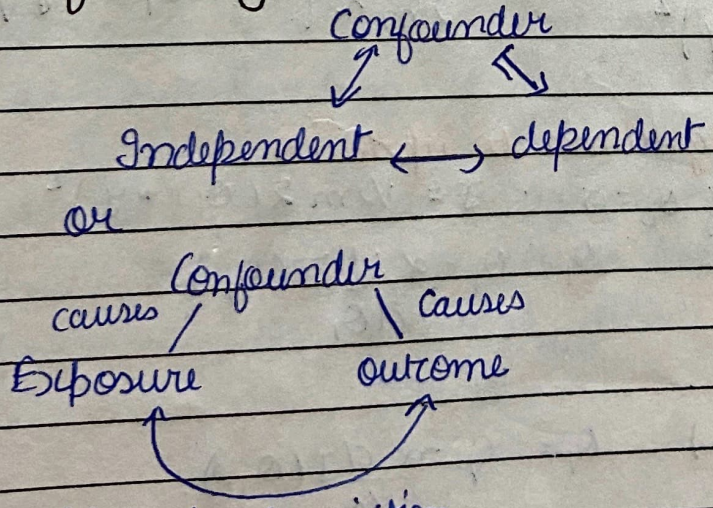




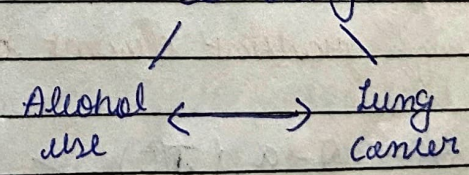
(b) Equal variance.



Confounding Variable:



eg:



Multivariate Regression Pitfalls

1. **Multicollinearity**: two variables that measure the same thing or similar things (eg weight and BMI) are both included in a multiple regression model, they will, in effect cancel each other out and generally destroy your model.
2. **Residual confounding**: we cannot completely wipe out confounding simply by adjusting for variables in multiple regression unless variables are measured with zero error (usually impossible)
3. **Overfitting**: In multivariate modeling, you can get highly significant but meaningless results if you put too many predictors in the model.

How machine Learns: Infer(Predict) → Error(cost) → Train(Learn)

$y = wx + b$

Hypothesis function: $h_\theta(x) = \theta_0 + \theta_1 x$

$MSE = \frac{1}{m}\sum(\hat{y} - y_i)^2$ Cost function:

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum(h_\theta(x) - y)^2$$

Goal: Min $J(\theta_0, \theta_1)$

if $\theta_0 = 0$ $\quad J(\theta_1) = \frac{1}{2m}\sum(\theta_1 x - y)^2$ $\quad$ else; $= \frac{1}{2m}\sum(\theta_0 + \theta_1 x - y)^2$

Using gradient descent algorithm to update θ:

if $\theta_0 = 0$: $J = 1/2m\,\Sigma(\theta_1 x - y)^2$

$\theta := \theta - \alpha \frac{d}{d\theta} J(\theta)$ $\quad$ $\theta_1 = \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$

if $\theta_0 \neq 0$

$\theta_0 = \theta_0 - \alpha \frac{d}{d\theta_0} J(\theta_0, \theta_1)$ $\quad$ $\theta_1 = \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1, \theta_2)$

$= \theta_0 - \alpha \frac{1}{m}\sum(h_\theta(x) - y)$ $\quad$ $\theta_1 = \theta_1 - \frac{\alpha}{m}\sum(h_\theta(x) - y)\cdot x$

Performance Metrics:

1. Mean squared Error (MSE) Measures the avg. squared difference between actual and predicted value.

$$MSE = \frac{1}{n} \sum (y_i - \hat{y})^2$$

2. Root Mean Squared Error (RMSE):

$RMSE = \sqrt{MSE}$ gives an interpretable error in the original unit.

3. Mean Absolute Error (MAE):

$$MAE = \frac{1}{n} \sum |y_i - \hat{y}_i|$$ the avg. absolute difference between actual and predicted values.

4. R-Squared (R^2)

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

$SS_{res} = \sum (y_i - \hat{y})^2$ (Residual Sum of Squares)

$SS_{tot} = \sum (y_i - \hat{y}.)^2$ (total sum of squares).

More about Gradient Descent:

Linear relationship: $y = a_0 + a_1 x$

{ y = target } ; { a_0 and a_1 are model parameters }

{ x = feature }

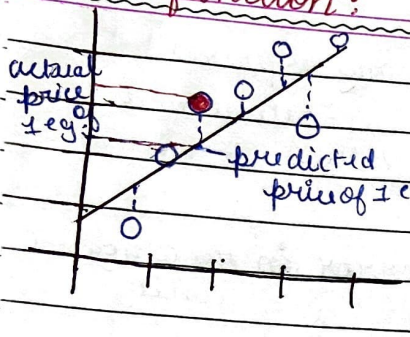
As per our linear regression model, we need to fit a straight line to it $y = a_0 + a_1 x_1$

depending on values of a_0 and a_1, we can have many possibilities, which look promising.

We need to settle the case on a value of parameters a_0 and a_1 corresponding to which straight line fits best to data.

→ For this we need to agree on a metric to judge best fit and then we can choose that straight line which performs best on that metric → { cost function } to resolve this.

Cost function:



Error Term (e): $y_{actual} - y_{predicted}$.

$e_1 = y_{actual} - y_{predicted}$.

for i^{th} term

$e_i = y_{predicted}^{i} - y_{actual}^{i}$

Now e_i can be +ve or -ve depending on the value of y_{actual} and predicted so square e_i

So now;

$$\text{Cost function } (J) = \frac{1}{2m}\left(e_1^2 + e_2^2 + e_3^2 + \dots + e_m^2\right)$$

$$= \frac{1}{2m}\sum\left(y_i - \hat{y}\right)^2 \quad \text{where } m = \text{no of example}$$

$$= \frac{1}{2m}\sum\left(a_0 + a_1 x_1^{(i)} - \hat{y}_{(actual)}\right)^2 \quad \left\{y_1 = a_0 + a_1 x_1 \text{ predicted}\right\},$$

Clearly, cost function (J) is a function of parameter space $a = (a_0, a_1)$.

→ The best fitting model would be the one which minimizes our error metric which is cost function.

also (lesser the distance between is our model & between straight line and datapoints.

→ How to minimize cost function?

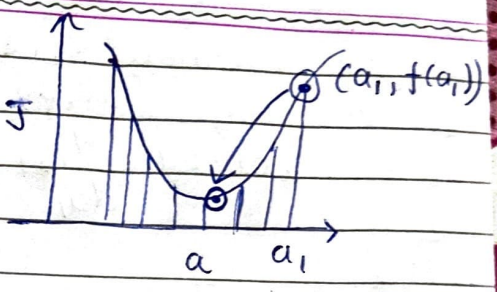
↳ As we have seen cost function (J) is a function of parameter space $\alpha = (a_0, a_1)$;

we will keep changing a_0 and a_1 till we find a combination where cost function is minimized.

→ for this Gradient Descent Algorithm is used.

Gradient Descent Algorithm :

Assume a function $J = f(a)$



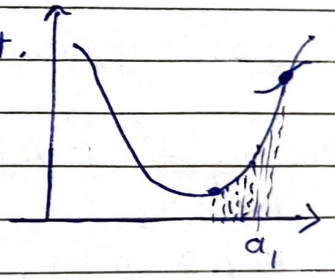
this curve represents the values of J will assume for different values of 'a'.

we are at $(a_1, f(a_1))$ as of now, and we want to reach at the minimum of this function.

→ How can we reach the minimum starting from a_1?

For this we will find slope at this point.

→ $\frac{dJ}{da}$ at a_1 or $f'(a_1)$.



Lets move a step 'α' in this direction to reach $a_1 - \alpha * f'(a_1)$

→ 'α' is a small fixed quantity in the range of 0.01. therefore the size of our steps is dependent on the slope $f'(a_1)$.

→ Higher the value of slope (which occurs away from minima) larger the steps and visa-versa.

→ As we move closer to minimum, the slope decreases and hence our steps towards minimum keeps on getting smaller. at minimum, slope becomes zero.

→ We preform multiple iteration of this Gradient Descent Algorithm. The number of iteration's preformed depends upon

→ nature of function

→ 'α' (learning rate)

→ and where we start from (a_1) in this case.

Step 1: Calculate $\frac{d(J)}{da_0}$ at current value of parameter a_0

Calculate $\frac{d(J)}{da_1}$ at current value of parameter a_1

Step 2:

$$(new)\ a_0 = a_0 - \alpha\left(\frac{dJ}{da_0}\right)$$

$$(new)\ a_1 = a_1 - \alpha\left(\frac{dJ}{da_1}\right)$$

Step 3: update cost function J with new (a_0 and a_1)

~~Repeat step 1~~

One important thing about gradient desient is learning rate (α):

Keep α small, its value is kept around 0.01 so that it neither makes algorithm too slow nor does it fail to converge.

→ Learning Rate (α) is a hyperparameter for this model even though it doesn't directly affect model like parameters a_0 and a_1 do but it can impact the performance of our model and we need to be cautious about choosing model hyperparameters.

Logistic Regression: (Overview)

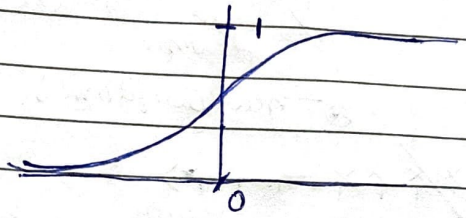
We often use linear regression to generate predictions for (numerical continuous) data eg: height, weight etc.

But what if data can not be expressed in number eg: color, gender or job role etc.

→ Categorical data consists of discrete values that describe an object. Logistic Regression is a great candidate for ML problems that involve binary classification, which determines the P(E) that each data entry can be classified into one of two different categories.

Logistic regression uses the sigmoid function, also known as the logistic function, to map a linear combination of the input features into a range of 0 to 1.

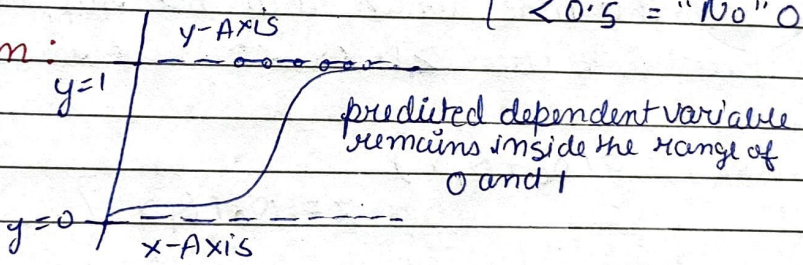
$$f(x) = \frac{1}{1 + e^{-x}}$$



→ We use (Decision Boundary) to determine at which point we will assign each data entry to each category. if (DB) is 0.5

{ >= 0.5 = 1 }
{ <0.5 = "No" 0. }

Logistic Regression:



→ technique for classification.
→ probabilistic view of classification.
→ dependent variable is binary rather than continuous and it can also be applied to ordered categories (ordinal data).

$$(MLE) - \ln\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 X$$

The term odds (odds of winning or affecting)
In logistic regression, the dependent variable is a logit, which is the natural log of the odds, that is,

$$\text{odds} = \frac{P}{1-P} \qquad \log(\text{odds}) = \text{logit}(P) = \ln\left(\frac{P}{1-P}\right)$$

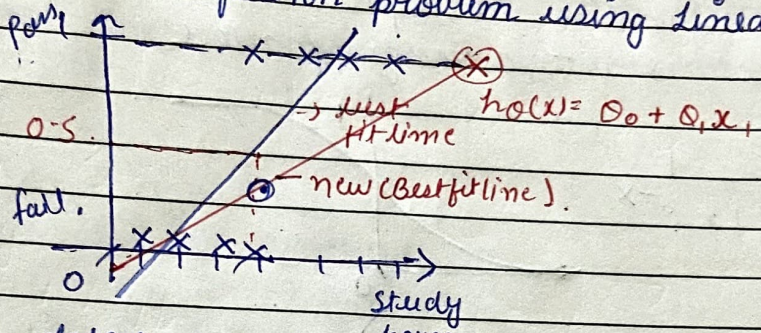
Logistic Regression Indepth Maths :

Linear Regression v/s Logistic Regression :

→ Can we solve classification problem using Linear Regression

No;

Why?



① As we will detect any (outlier) in the graph the best fit line will also change and this will lead to mis-value reading, Also in linear regression there will be a straight line which will not support binary output.

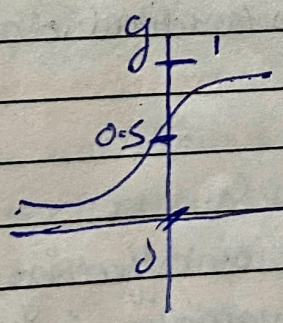
② output can be >1 or <0. also in some case.

$$h_\theta(x) = g(\theta_0 + \theta_1 x)$$

$$\left\{ g = \frac{1}{1+e^{-z}} \; ; \quad z = \theta_0 + \theta_1 x \right\}$$

$$\left\{ h_\theta(x) = \frac{1}{1+e^{-(\theta_0 + \theta_1 x_1)}} \right\} \text{ - hypothesis in (case of Logit.}$$

if



$$\left\{ \begin{array}{l} z \geq 0 \\ g(z) > 0.5 \text{ --- 1.} \end{array} \right\}$$

$$\left\{ \begin{array}{l} z < 0 \\ g(z) < 0.5 \text{ --- 0.} \end{array} \right\}$$

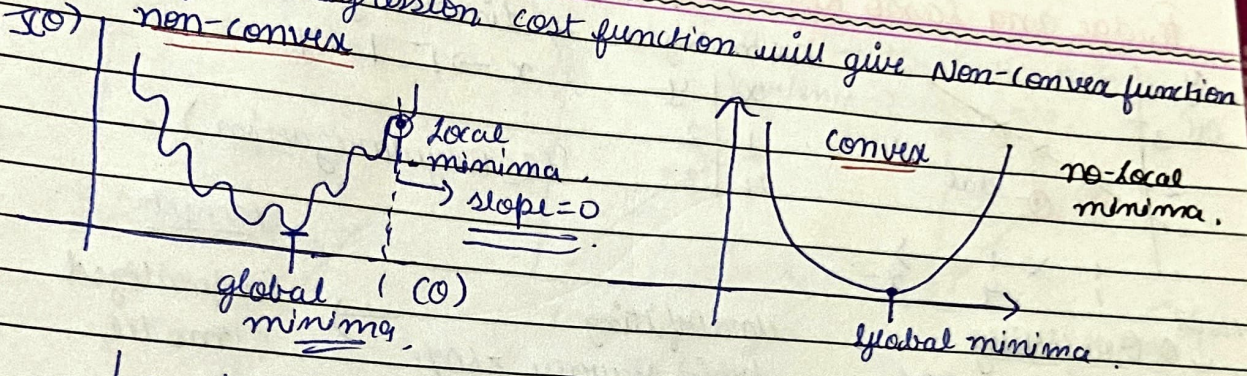
Cost function of Logistic Regression:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum \left(h_\theta(x)' - y^{(i)} \right)^2$$

$h_\theta(x)' = \frac{1}{1+e^{-z}}$

$\Rightarrow h_\theta(x) = \frac{1}{1+e^{-(\theta_0 + \theta_1 x)}}$

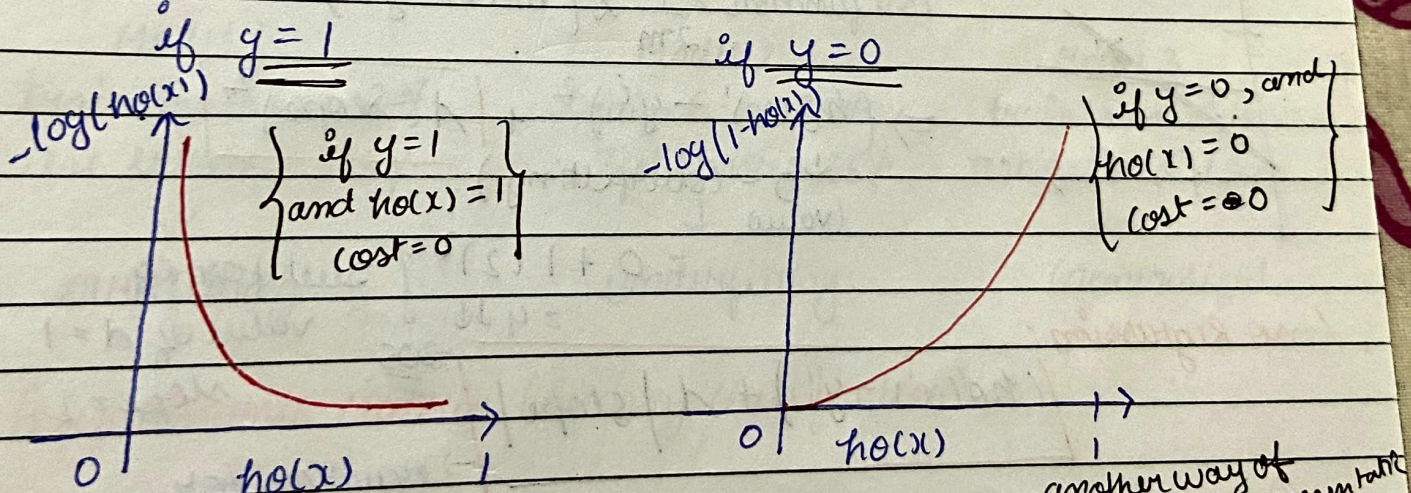
In Logistic Regression cost function will give Non-convex function



↳ to make the funtion convex

$$\text{Cost}\{h_\theta(x^{(i)}), y\} = \begin{cases} -\log(h_\theta(x)) & \text{if } y=1 \\ -\log(1-h_\theta(x)) & \text{if } y=0 \end{cases}$$

y ki value will always be 0 or 1.



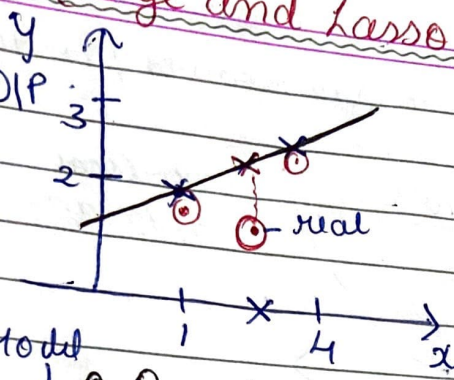
another way of representation

$$\text{Cost}\{h_\theta(x), y\} = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))$$

Convergence Repeat:

$$\left\{ \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \right\} \rightarrow \text{tells global minima.}$$

Ridge and Lasso Regression (overview)



trainingset ⇒ Linear Regression

x	y
1	2
4	3

$x \rightarrow [\ ] \rightarrow y$.

(L_2 Regularization).

Model → Overfitting → target

Train Accuracy = 90% ⇊ Test Accuracy = 70%

{ Low Bias, High Variance }

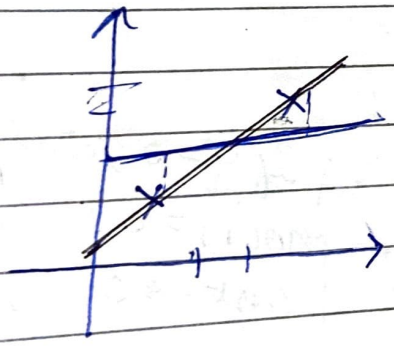
Underfitting → Generalized model needs

Train Accuracy = 60%

Test Accuracy = 62%

{ ~~Low~~ High Bias, ~~Low~~ Variance High } - { Low Bias and Low Variance.

To prevent overfitting we use Ridge Regression:



$$\text{cost function} = \frac{1}{2m}\sum (h_\theta(x)^i - y^i)^2$$

$$\Rightarrow (h_\theta(x)^i - y^i)^2 + \boxed{\lambda(\text{slope})^2}$$

↓0 - (overfitting).

(value)

$= 0 + 1(2)^2$ } Just for eg value of $\lambda = 1$ slope = 2.

$= 4$ ⇊.

Lasso Regression:

$$\boxed{(h_\theta(x^i) - y^i) + \lambda|\text{slope}|} \approx 205.$$

→ equation for lasso.

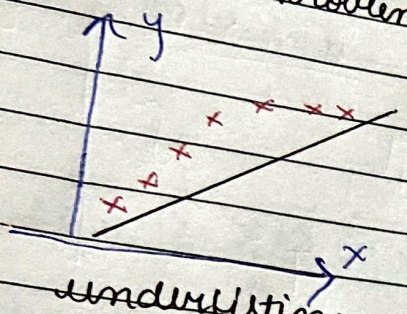
→ ① Overfitting prevented

→ ② Helps in feature selection.

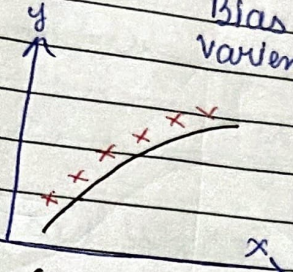
(L1 ~~Regression~~ Regularization)

Bias and Variance : (overview).

Regression Problem :

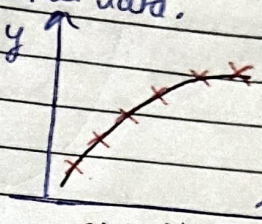


underfitting.
(TE ↑)
High Bias
{ High variance
Low variance }
↓ Both can be possible.



Bias → Training Data
Variance → Test data.

Generalized.
TE(↓)
{ Low Bias
Low variance }



overfitting
(TE ↓) ≈ 0
{ Low Bias.
High Variance }.

Classification Problem : = { Confusion Matrix }.

Model 1	Model 2	Model 3
training error 2%.	train E ≈ 25%	train Error < 10%
test error = 25%.	Test Error ≈ 26%	test error < 10%
↓	↓	↓
overfitting.	underfitting	Generalized.

Performance Matrix (Binary classification).