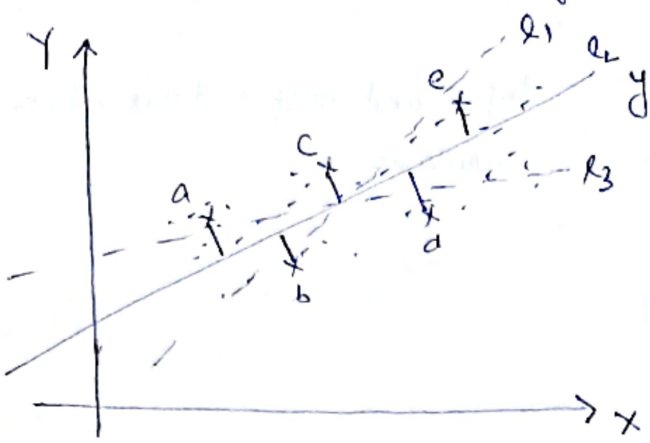
Linear Regression



slope

$y = mx + c$ - best fit line

↳ intercept

$e = y - y'$

$\min(e)$ $\quad \frac{\partial e}{\partial m} = \frac{\partial e}{\partial c} = 0$

that value of m, c.

We will choose a line which is at a min distance from points a, b, c, d, e. $(d_a + d_b + d_c + d_d + d_e) \rightarrow$ minimum

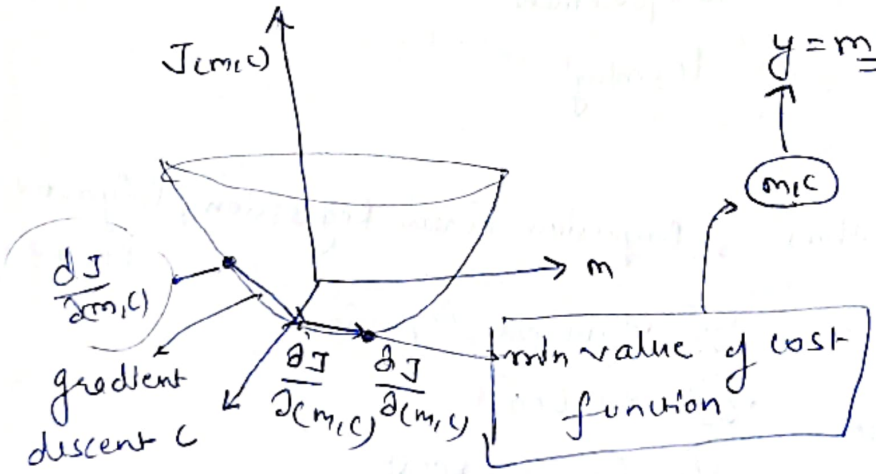
$\hat{y} = mx + c$

$$\text{Cost function / error fn} = J(m,c) = \frac{1}{2N} \sum_{i=1}^{N} (\hat{y} - y)^2$$

↳ Real value

$2N$ ↓ Number of all the points

→ minimize this error



$y = mx + c$

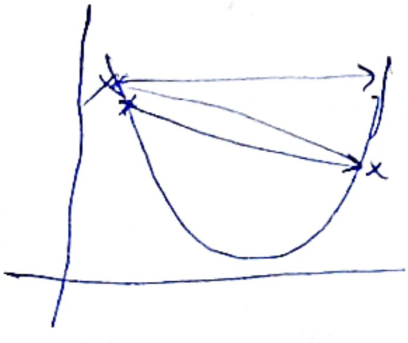
m, c

$\frac{\partial J}{\partial (m,c)} = 0$

↓ global minima

Convergence theorem

$m - \left(\frac{\partial m}{\partial m_2}\right) \times \alpha$ → slope, → learning Rate, → small.



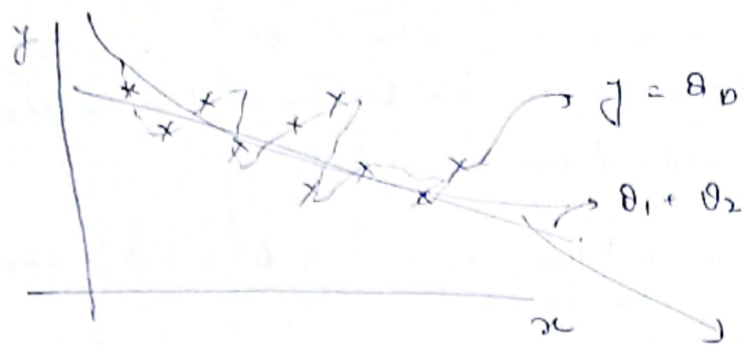
if $\alpha \uparrow\uparrow$

Converging gradient problem

$\alpha \downarrow\downarrow$

↳ slow

Lasso | Ridge | Elasticnet | L1 L2 Regularizations



$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \cdots$ → high order polynomial eqn

$\theta_1 + \theta_2 x = y$ → l_1 → underfit

l_2 → overfit

$y = \theta_0 + \theta_1 x + \theta_2 x^2$ → l_3 → balanced

l_3 is a balance b/w l_1 & l_2

In order to obtain a balanced fit we need regularization.

$$\text{mean sq error} = \frac{1}{n} \sum_{i=1}^{n} (y_i - y_{pred})^2$$

$$y_{pred} = h_\theta(x_i)$$

$$mse = \frac{1}{n} \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2 + \left(\lambda \sum_{i=1}^{n} \theta_i^2\right)$$

Ridge → L_2 Regularisation

b/c we are using square

if $\lambda \uparrow\uparrow$ → horizontal line

λ (slope)^2

This lambda can help define the limits for θ.

By adding this value we are trying to ensure that the model gets penalised whenever we have a higher value of θ in our predictions.

It tend to reduce the slope steepness. With unit increase in x there is high increase in y.

$$mse = \frac{1}{n} \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2 + \lambda \sum_{i=1}^{n} |\theta_i|$$ → L_1 Regularisation

no square

Lasso

* overfitting - high variance, ~~high bias~~
* balanced model - low bias and low variance

helps us to do feature selection

$$y = m_1x_1 + m_2x_2 + m_3x_3 + m_4x_4 + c$$

$\lambda |m_1 + m_2 + m_3 + m_4|$ → unimportant or bias feature will automatically get reduced near to zero

let's say fit with m_4 is very less and here x_4 will also be a useless feature

In case of lasso regression a slope can reach zero and can render a feature completely useless and in case of ridge the slope may reduce but will never get to zero. Elastic net uses both lasso and ridge together to include the impact & benefits of both the varieties.

* Lasso works best when we have a lot of useless features. as it eliminates all the insignificant ones and keep only the most significant ones

* Ridge regression works best when we have a lot of useful variables i.e when all the variables are significant ones.

* when we have a model that include millions of parameters. and we don't know whether which variables are useful and which are not.

$$mse = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2 + \lambda\sum|\theta_i| + \lambda_2\left[\sum(\theta_i^2)\right]$$ elastic net regression

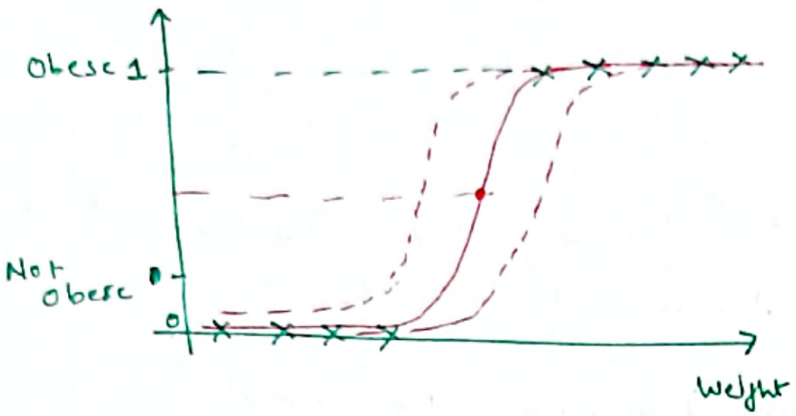
↓ very good when there exist a lot of correlated parameters.

$\lambda_1 = 0$ $\lambda_2 \neq 0$ Lasso

$\lambda_1 \neq 0$ $\lambda_2 = 0$ Ridge

$\lambda_1 = 0$ $\lambda_2 = 0$ mse

* Logistic Regression



Binary classification

* We can also do this classification through linear regression but that will have a high error rate and is prone to outliers. This may not work effectively. Hence sigmoid function is used for classification.

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

* Logistic regression is applied to a classification problem that is linearly separable, can be divided with the help of straight line.

Cost function = $\max \sum_{i=1}^{n} y_i (w_i^T x_i)$ → if I want a linearly separable line

Classification by linear regression

only parameter to be updated

y_i → +1/-1

w_i^T → distance of point from plane

x_i → the point

If an outlier exists then this will not be possible

Sigmoid fn → logistic (Maximum likelihood)

$$\text{Cost fn} = \max \sum_{i=1}^{n} f(y_i w_i^T x_i)$$

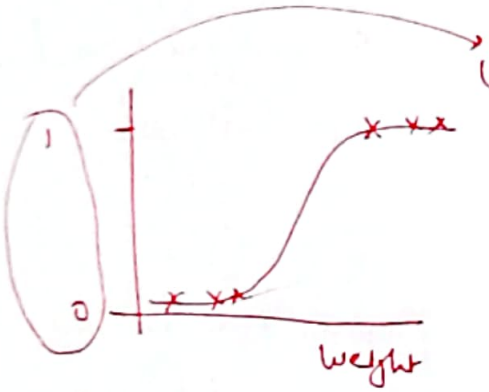
$$f(x) = \frac{1}{1+e^{-x}} : (0-1)$$ → removes the effect of outliers in linearly separable algorithms.

- No multicollinearity in data
- Binary & dichotomous classification
- Linearly separable data – log odds
- Large sample

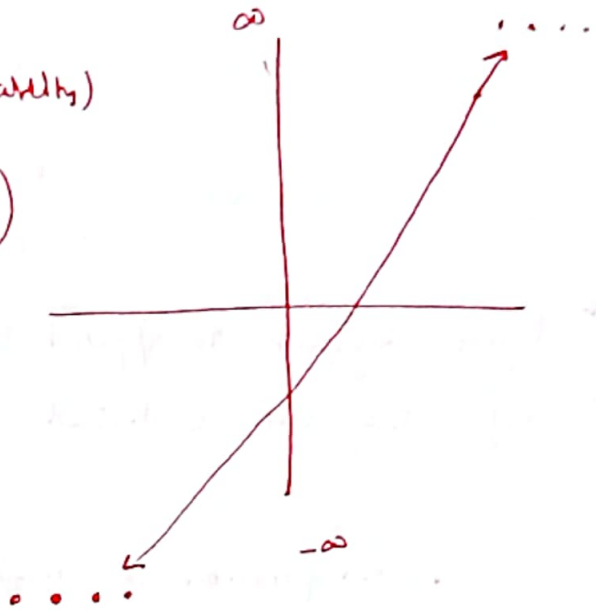
– logistic regression

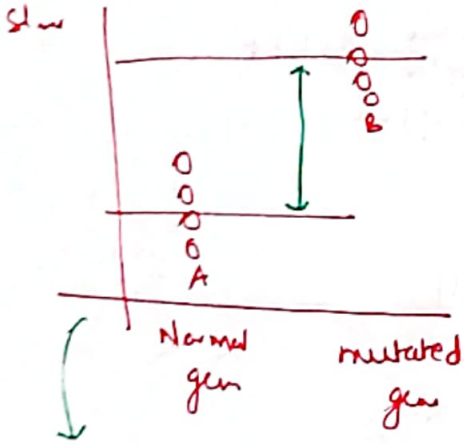
weight is the continuous variable predicting obese. There can be a categorical variable leading to this prediction as well.

Logistic regression is one type of generalised linear model.



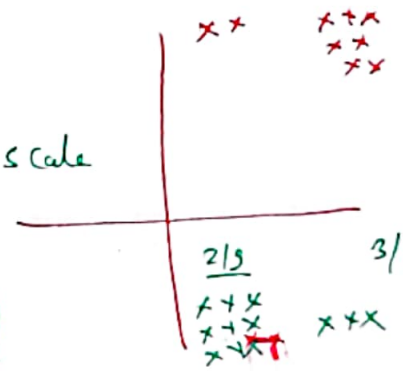
$\log(\text{odds of probability}) = \log\left(\frac{p}{1-p}\right)$





$\text{size} = \text{mean normal}_A \times B_1 + (\text{mean}_B - \text{mean}_A) \times B_2$

for logistic regression - transform this to a log scale



$\log(\text{odds gene normal}) = \log(2/9)$

$\log(\text{odds gene mutated}) = \log(7/3)$

$\Rightarrow \text{size} = \log(\text{odds gene normal}) \times B_1 + (\log(\text{odds gene mutated}) - \log(\text{odds gene normal})) \times B_2$