ML
- Classification → Yes/No
- Regression → how much

Linear Regression

eg. price of house based on size

x-axis → size y-axis → price

best fitting line ⇒ Linear Regression Model

How to find line

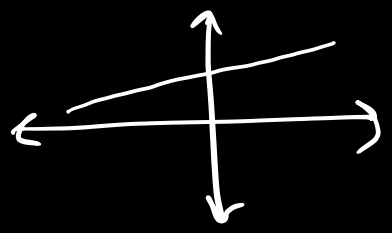
Select a random line and find error

move the line

$y = w_1 x + w_2$

$w_1 \rightarrow$ slope

$w_2 \rightarrow$ y-intercept



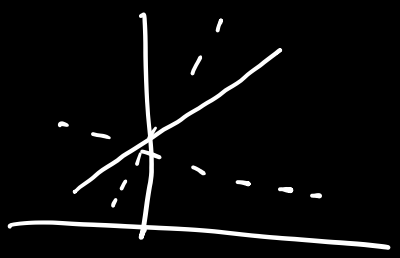
$$w_1 = \frac{\text{increase in vertical}}{\text{increase in horizontal}}$$

steepness ⇒ slope

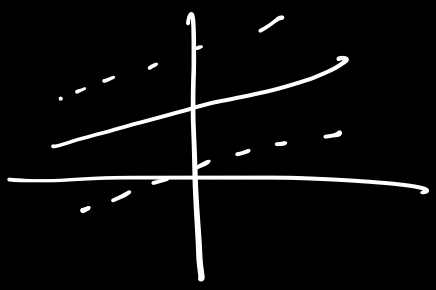
$w_1 = \Delta y / \Delta x$

We can move the line by change in w_1 and/or w_2

We can increase the slop or decrease the slop

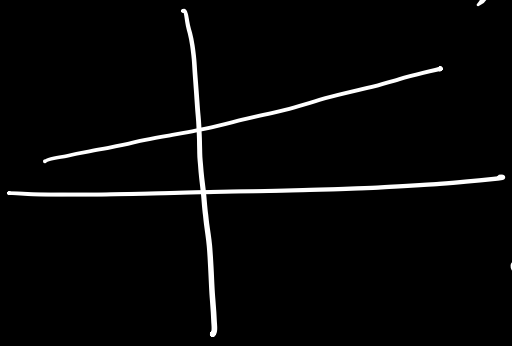


By we an increase w_2 or decrease w_2



Absolute Trick

(P,Q)



We want to move line towards (P,Q)

$$y = w_1 x + w_2$$

w_2 to be increased by 1 (y-intercept)

w_1 also to be increased by p

→ apply learning rate

$$y = (w_1 + p\alpha)x + (w_2 + 1.\alpha)$$

If point is below the line

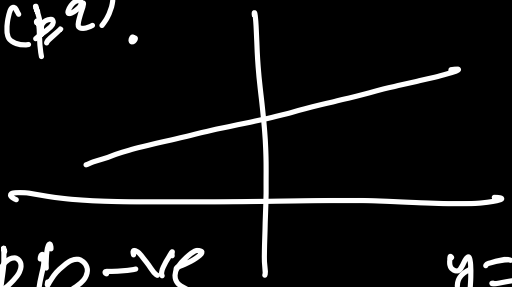
$$y = (w_1 - p\alpha)x + (w_2 - \alpha)$$

Why subtract

If we reduce w_2 line moves below

If we reduce w_1 line rotates in lower direction

If the point is left of y-axis (p<0).



$y = w_1 x + w_2$

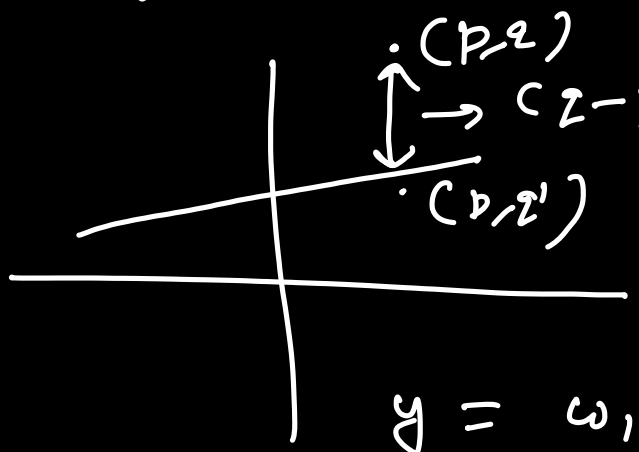
Now p is -ve

$$y = (w_1 + p\alpha)x + (w_2 + \alpha)$$

Here still add but since $p < 0$ slope is reduced

That's why we use p

Square Trick



$$y = w_1 x + w_2$$

$$p\alpha(q-q') + \alpha(q-q')$$

This helps to move line faster in the direction in comparison to constant move α

$$y = \left(w_1 + p(q-q')\alpha\right)x + w_2 + (q-q')\alpha$$

If point is below line

$q-q' < 0 \Rightarrow$ subtraction

No need of different rules

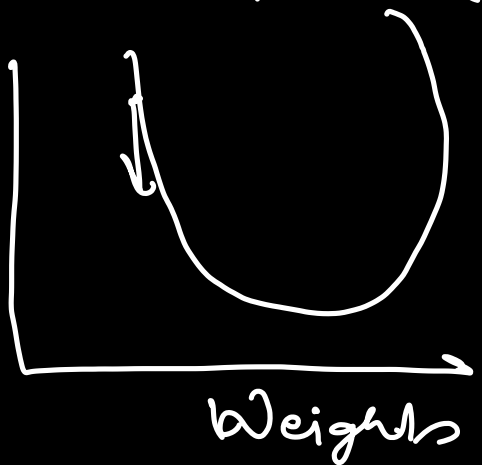
Linear Regression

Draw a random line
Compute error
Move line in one direction to see error
if error reduces then keep going
else change direction

To Minimize the error
we use Gradient Descent

Find the direction that can reduce the error

Error Function
-ve gradient of error function



Since gradient gives the direction increases most so we take -ve direction

$$w_i = w_i - \alpha \frac{\partial \text{Error}}{\partial w_i}$$

Now we have Errors

→ Mean Absolute Error

→ Mean Squared Error

Mean Absolute Error

$$\text{Error} = \frac{1}{m} \sum_{m} |y - \hat{y}|$$

Mean Squareal Error

$$\text{Error} = \frac{1}{2m} \sum_{m} (y - \hat{y})^2$$

1/2 is convenience

since no impact on minimization

So for Linear Regression

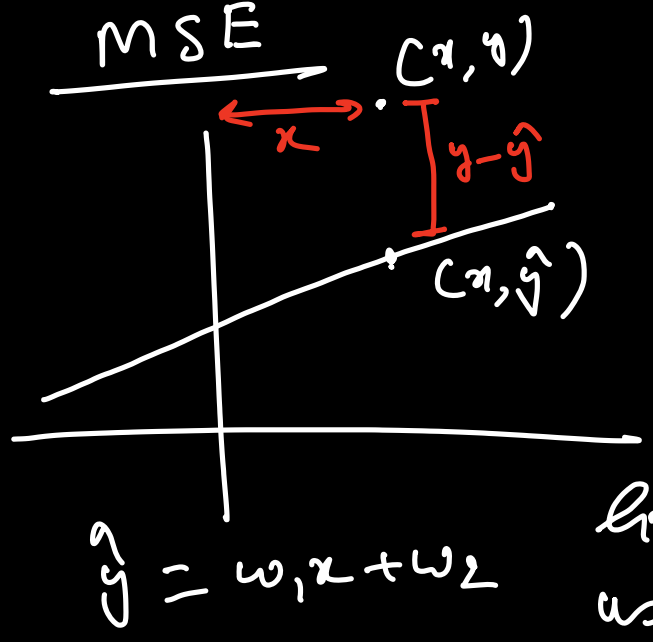
Absolute Trick
Square Trick

Minimize Error Function

Both are same

Gradient Descent on MAE
⇒ Absolute Trick (move constant)

Gradient Descent on MSE
⇒ Square Trick
(move proportionally)



$$E = \frac{1}{2m}\sum (y-\hat{y})^2$$

$$= \frac{1}{2}(y-\hat{y})^2$$

Gradient Descent uses derivative

$$E = \frac{1}{2}(y-\hat{y})^2$$

and $\hat{y} = w_1 x + w_2$

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_1} = -(y-\hat{y}) \cdot x$$

$$\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_2} = -(y-\hat{y}) \cdot 1$$

Gradient Step

$$w_i \rightarrow w_i - \frac{\partial}{\partial w} \text{Error}$$

$$w_1 = w_1 + (y - \hat{y}) \cdot x$$

$$w_2 = w_2 + (y - \hat{y})$$

{ earlier $x = 5$, $y = 2$ and learning rate

By MAE

$$E = |y - \hat{y}|$$

$$\hat{y} = w_1 x + w_2$$

$$\frac{\partial E}{\partial w_1} = \pm 1 \cdot x = \pm x$$

$$\frac{\partial E}{\partial w_2} = \pm 1 \cdot 1 = \pm 1$$

$$w_1 = w_1 \pm x$$

$$w_2 = w_2 \pm 1$$

* learning rate

lalhat's better MAE or MSE

MSE is a quadratic function and quadratic function has minimum in the middle

Now if we have many variables

$$y = w_1 x + w_2 x + w_3 x + \cdots + w_n$$

$$= \sum_{i=1}^{n-1} w_i x_i + w_n$$

$x_1, x_2, \ldots x_{n-1}$ are features

We can use MSE and use Gradient Descent of MAE

Polynomial Regression

Consider a polynomial of higher degree

$$\hat{y} = w_1 x^3 + w_2 x^2 + w_3 x + w_4$$

Now compute MSE and Gradient Descent

Regularization

works for both Regression and Classification

Useful technique to improve model

During training model takes error and minimize it

The problem is during minimization we can reduce so much that it overfits

$$M1 \rightarrow 3x_1 + 4x_2 + 5 = 0$$

$$M2 \rightarrow 2x_1^3 - 2x_1^2 x_2 - 4x_2^3 + 3x_1^2 + 6x_1 x_2 + 4x_2^2 + 5 = 0$$

M1 is simple may have some error (E1)

M2 is complex may have less error (E2)

$\longrightarrow$ Now consider the complexity of the model to compute error

$E1 = E1 + |3| + |4|$ (without constant)

$E2 = E2 + |2| + |-2| + |-4| + |3| + |6| + |4|$

Adding absolute values of co-efficients to error term

$\Rightarrow$ L1 - Regularization

$\Rightarrow$ Error in E1 is less than E2

So E1 is less $\Rightarrow$ M1 is better

L2 - Regularization

Add squares of the co-efficients instead of absolute values.

So, with L1, L2 we punish the complex models

Simple Vs Complex Models

Medical Model

Requires low error

Sending Rocket on Moon

OR If Complex

Video Recommender or other similar

Requires simple

Now how much to punish

↳ λ parameter

$$E1 = E1 + \lambda(\sum \text{co-eff})$$

$$E2 = E2 + \lambda(\sum \text{co-eff})$$

Now λ can decide punishment to complexity

L1 / L2 Regularization

L1	L2
Computationally Inefficient (unless data is sparse)	Computationally Efficient (since GD can be easily applied due to derivative)
Sparse Outputs	Non-sparse outputs
Feature Selection	No-Feature Selection