Gradient Ascent / Descent

So far we have

$\theta = (X^T X)^{-1} X^T Y$ (Closed form sol^n)

→ we can't use this in every case
  - matrix may be too large to compute X^T
  - too many dimentions D
  - Inverse may not even exist

→ we need alternate method to compute a global sol^n

∴ Gradient based learning.

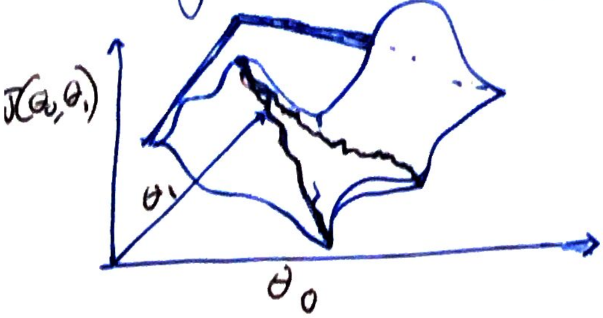
* → we comput gradient of eq^n wrt each parameter in order to move our current parameter's value in that direction

$$\theta_j \mathrel{+}= \frac{\partial J}{\partial \theta_j}$$

we do this iteratively until we converge to a minima/maxima.

we can do this to find values of params to minimize or maximize some f^n

- gradient is synonymous with slope



(vector)

gradient can be thought of as a way to go towards minima/maxima

$\frac{\partial J}{\partial \theta_0}$ $\frac{\partial J}{\partial \theta_1}$ Keep on changing till we reach a min/max

→ want to minimize? go downhill!

→ maximize? go uphill!

$$\frac{dJ}{d\theta} = \begin{bmatrix} \frac{dJ}{d\theta_0} \\ \frac{dJ}{d\theta_1} \\ \frac{dJ}{d\theta_2} \\ \vdots \\ \frac{dJ}{d\theta_D} \end{bmatrix}$$

$$\theta = \theta - \frac{\partial J}{\partial \theta}$$