$y = mx + C$ errors.

There are two ways to solve this equation

1) Find the best value of $C, m_1, m_2, m_3 \ldots m_n$

$$\hat{\theta} = (X^T \cdot X)^{-1} \cdot X^T \cdot Y$$

Best value that minimizes cost function.

Training set

Target Vector

$$\theta = \left[(X_b)^T \cdot (X_b) \right]^{-1} \cdot X_b^T \cdot y$$

Dot product

I

In normal equation we are finding $\boxed{x^T \cdot x}$

which will result in $n * n$ matrix.

1000,000,000 We are calculating (-1) or inverse of this

$n * n$ matrix. Which will

Python can do require = ~~$O(n^{2.3})$ time to solve.~~

or $\boxed{O(n^3)}$

To calculate in 1 sec If i have matrix of 1000 rows time to compute inverse of 1000×1000 matrix.

100×10^7 sec $= \boxed{100 \text{ sec}}$

$[1000, 1000]$ - soln

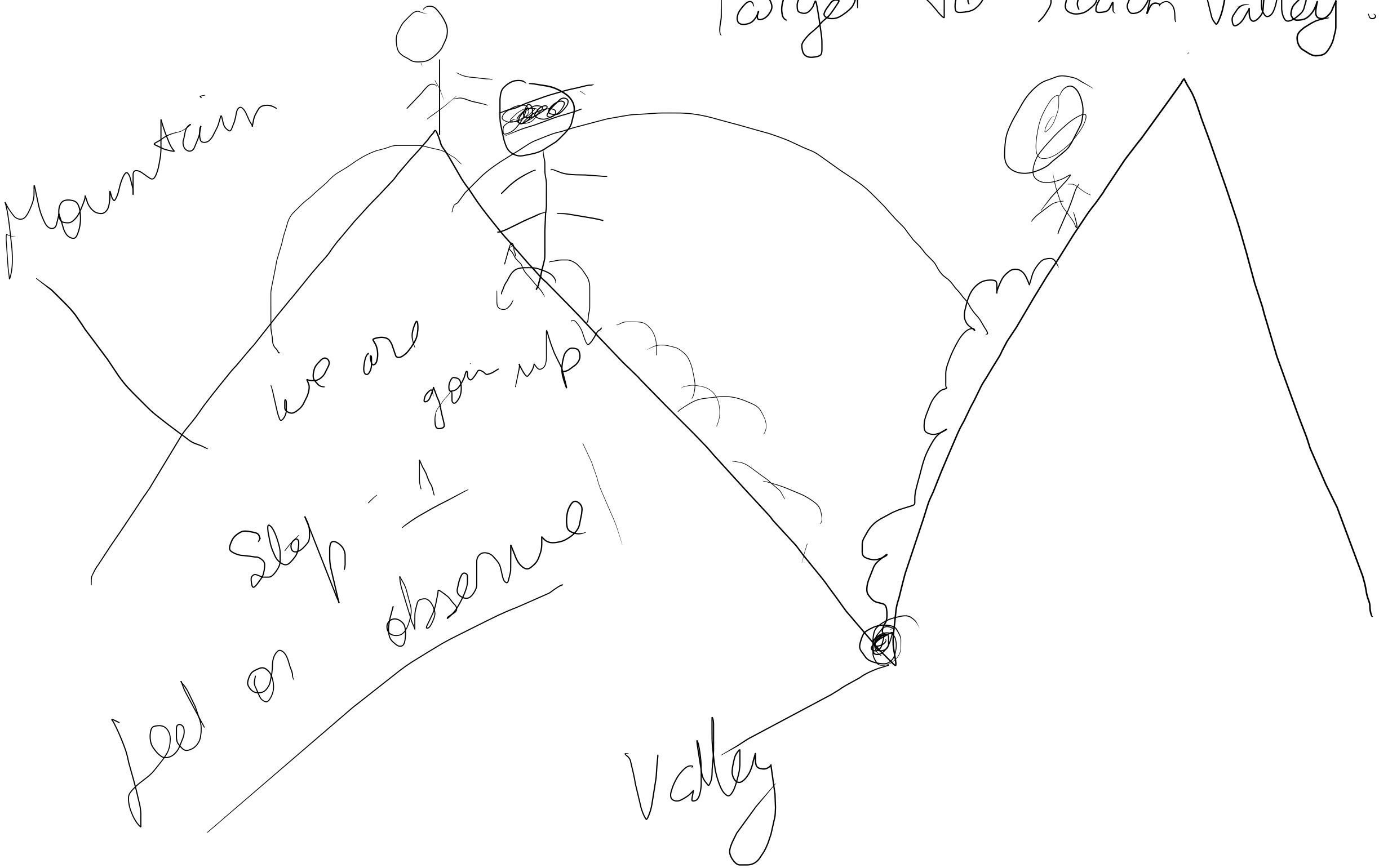
100 sec

2.8

$$\frac{(10{,}000)^3}{10^7} = \frac{(10)^{12}}{10^7} = 10^5 = \boxed{10{,}000} \text{ sec}$$

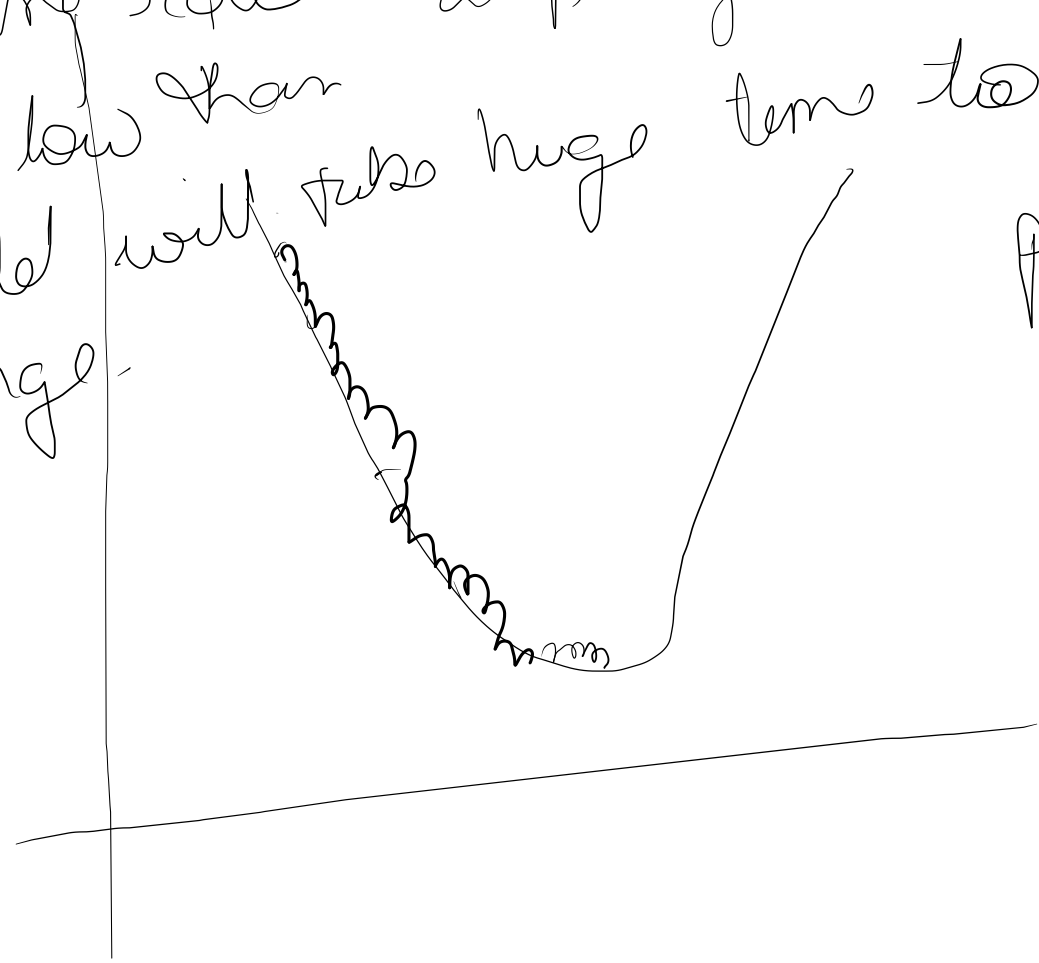
$$\frac{10{,}000}{3600}$$

Target to reach Valley.



We take random α values & start

1) If learning rate is too low than our model will take huge time to converge.

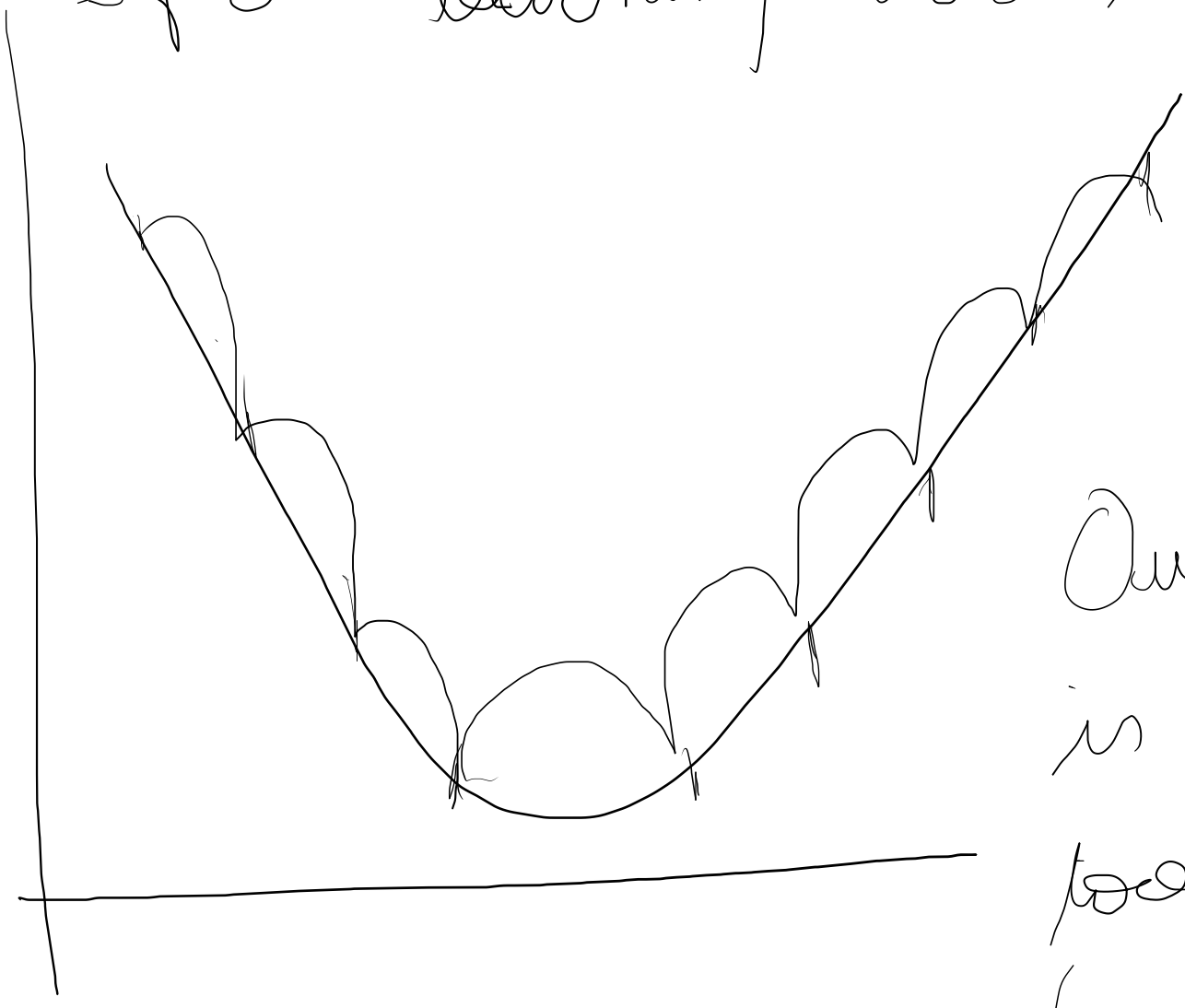


Step size - is also called learning rate

Prob 1) If learning rate is too low

Prob 2) If learning rate is too high :

If our learning rate is too high then?

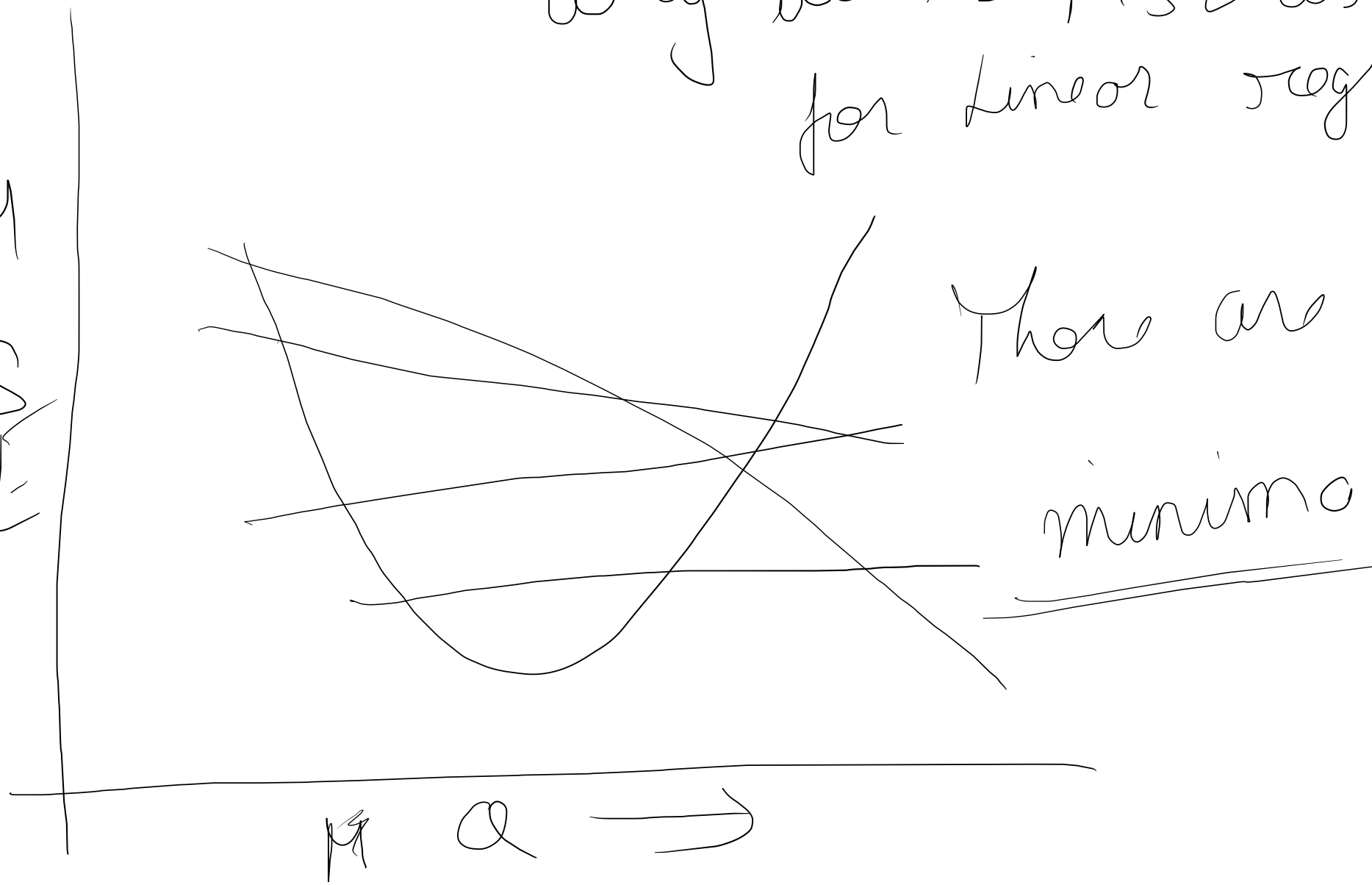


we may never reach minima (actual)

Our learning rate is should be neither too high or too low

(.001) (.01) (.1)

Why we use MSE as Cost function for linear regression



There are no local minima