



مجموعة جامعة
الجميع الذكية
للتعليم القابضة

MIDOCEAN
UNIVERSITY
BY EVERYONE'S SMART UNIVERSITY

Mathematical Foundations For Data Science and AI

Lecture 04

AI102

Calculus; Gradient Descent and Optimization

By
Dr. Khaled ELMenshawy

www.midocean.ae

Lecture

04

LEARNING OUTCOMES

- Understanding Gradient Descent.
- Understanding learning rate.
- Understanding gradient descent in two functions.
- Understanding how to use gradient descent in optimization.
- Understanding Log-Loss.

Hard to optimize function:

$$f(x) = e^x - \log(x)$$

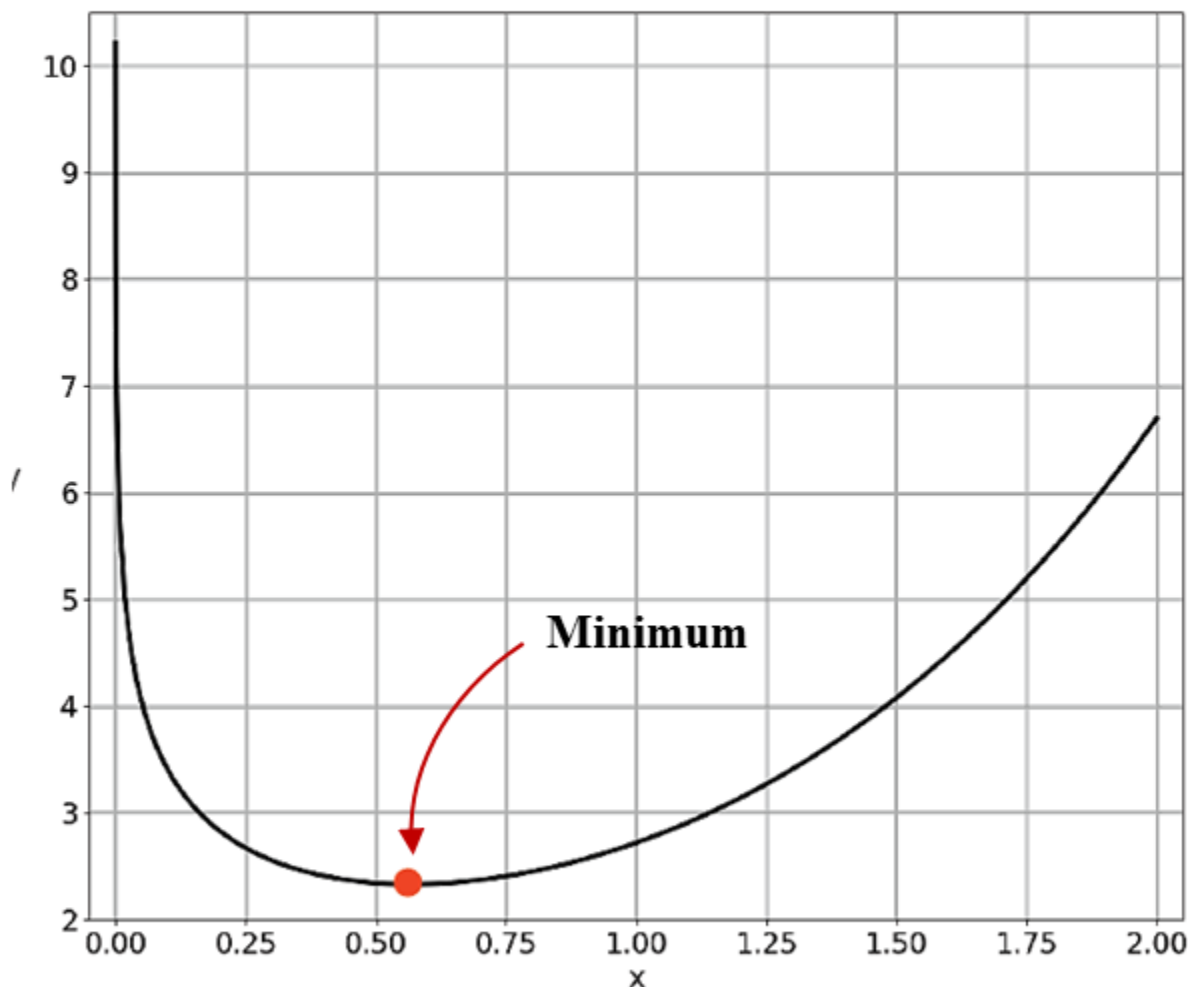
$$f'(x) = e^x - \frac{1}{x}$$

To get the minimum value, let $f'(x) = 0$

$$e^x - \frac{1}{x} = 0$$

$$e^x = \frac{1}{x}$$

Hard to solve analytically.

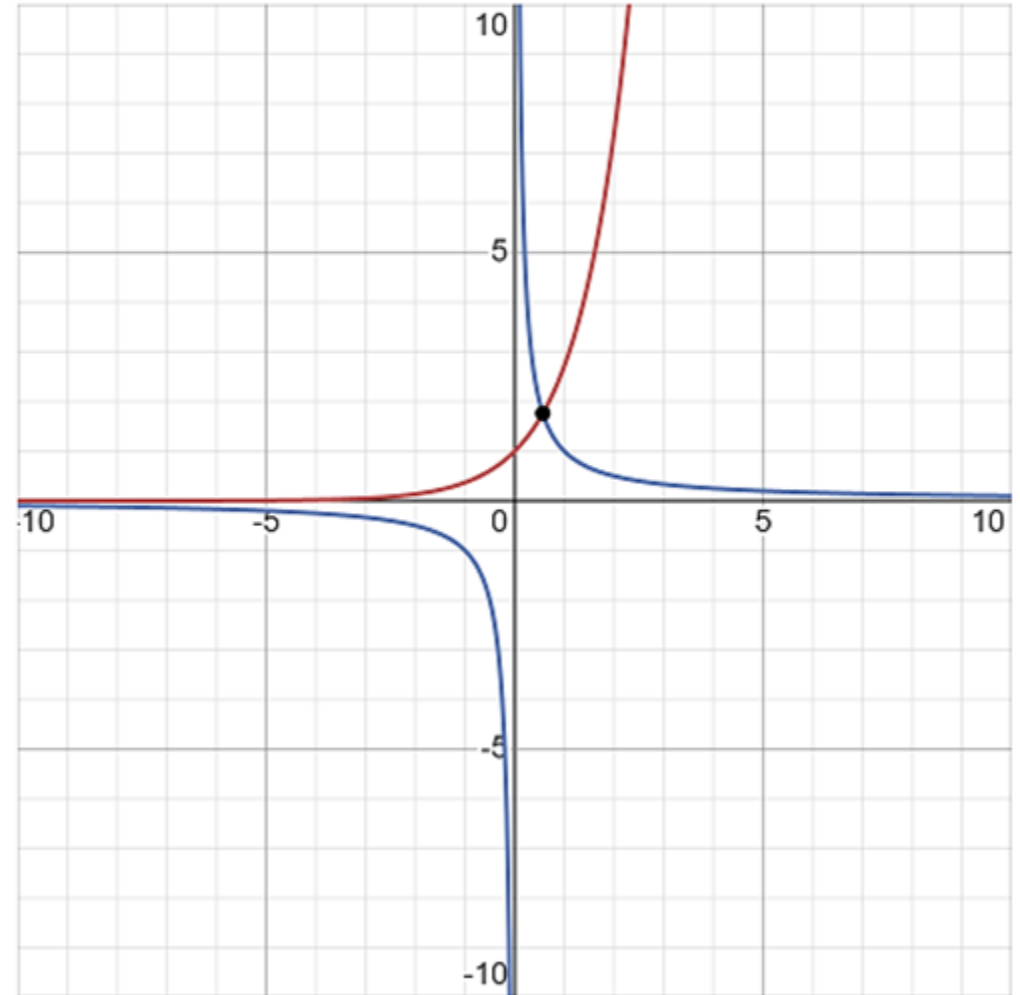


The solution is:

$$x = 0.5671 \dots$$

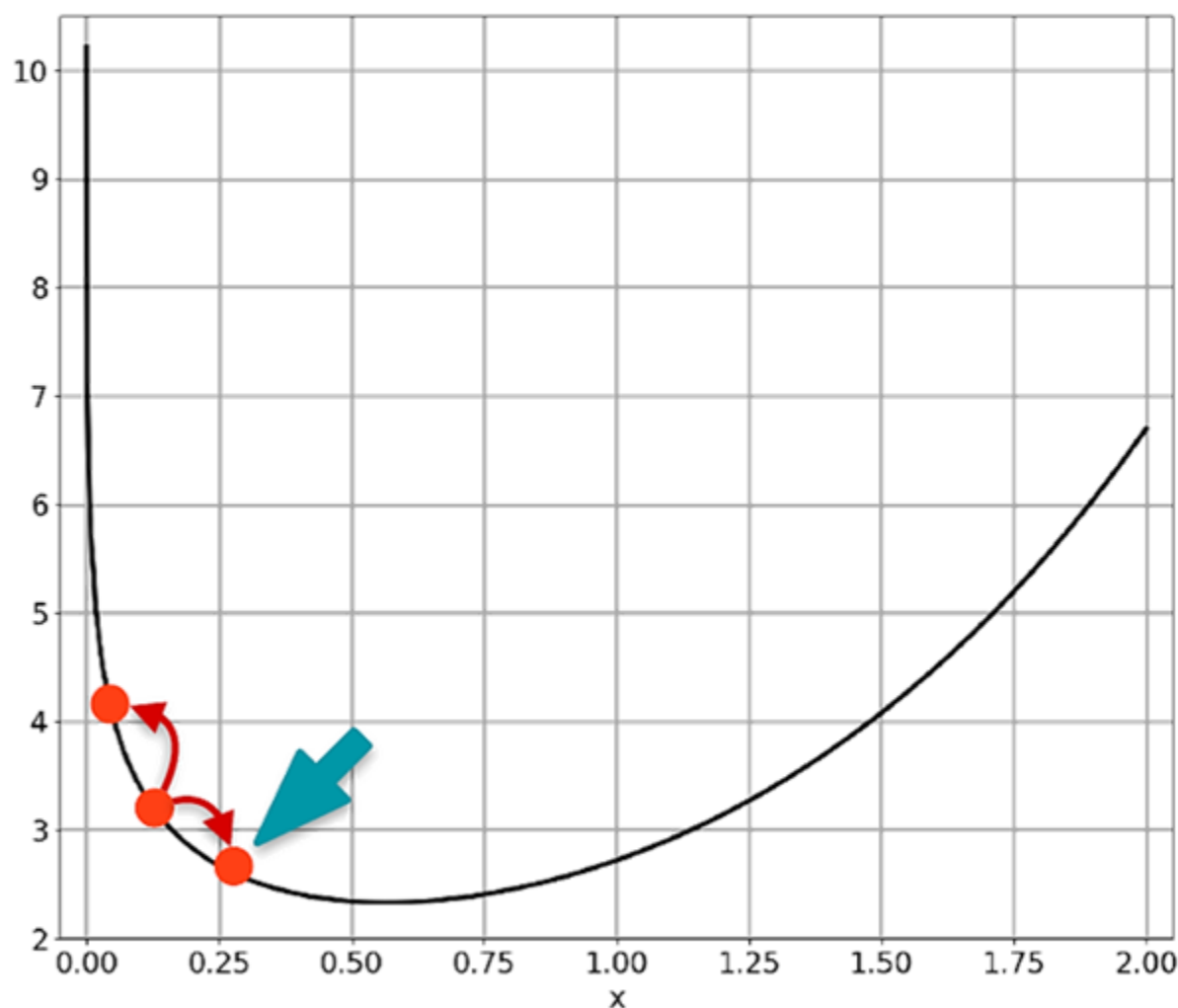
Known as the Omega constant (Ω), it is a mathematical constant defined as the unique real number that satisfies the equation:

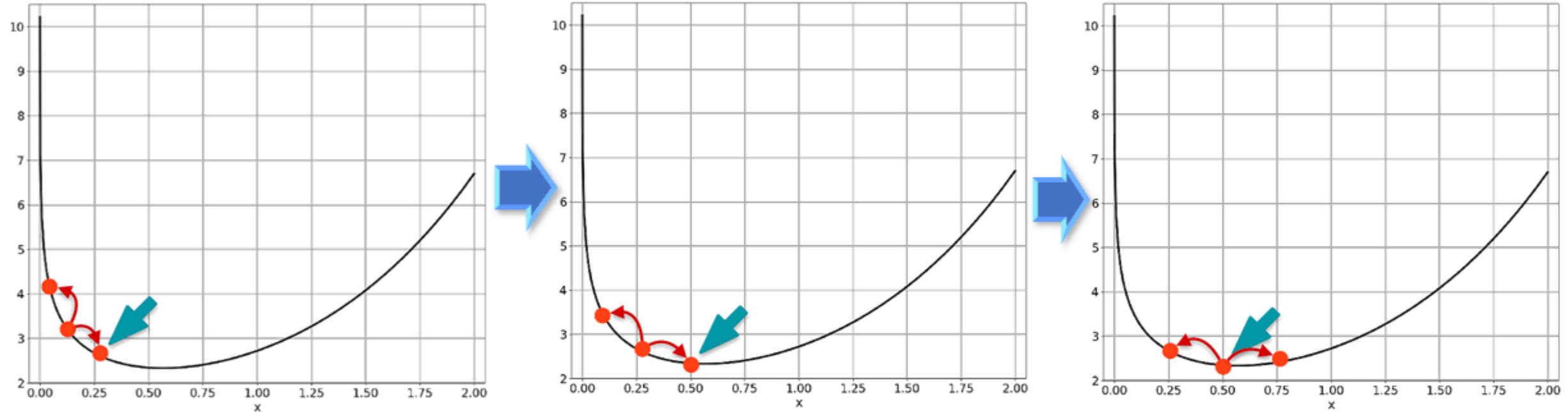
$$xe^x = 1$$



Here is another method:

Let's pick a random value, Now, let's give it a try and move to the left by a little bit, and move to the right by a little bit.





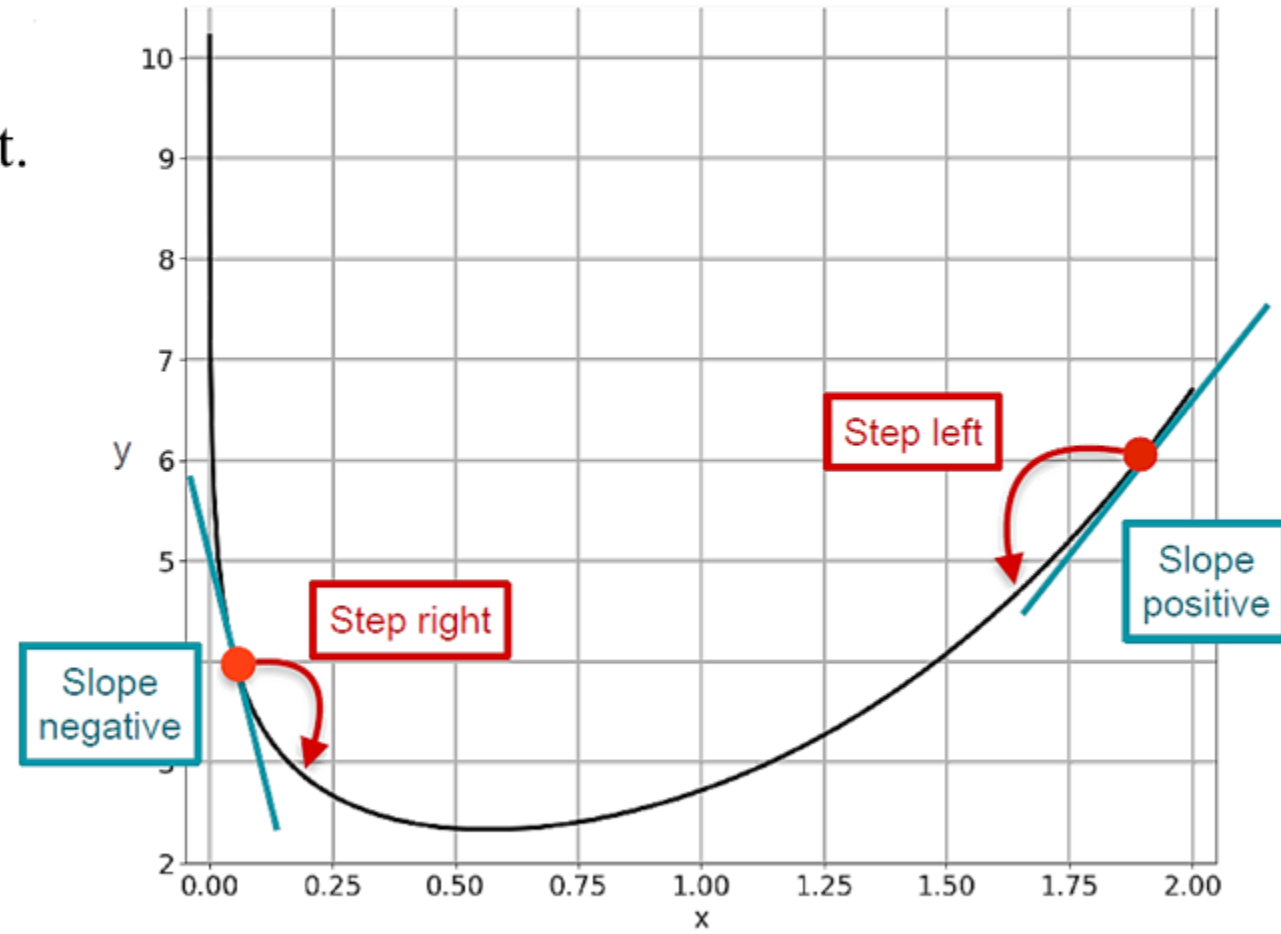
So, we move in both directions and check to see which point is smaller.

Which point we should choose?

When the slope is negative, move to the right.

When the slope is positive, move to the left.

At a point where the **slope is negative**, we need to **add a small value to the coordinate** of the point. Whereas At a point where the **slope is positive**, we need to **subtract a small value from the coordinate** of the point.



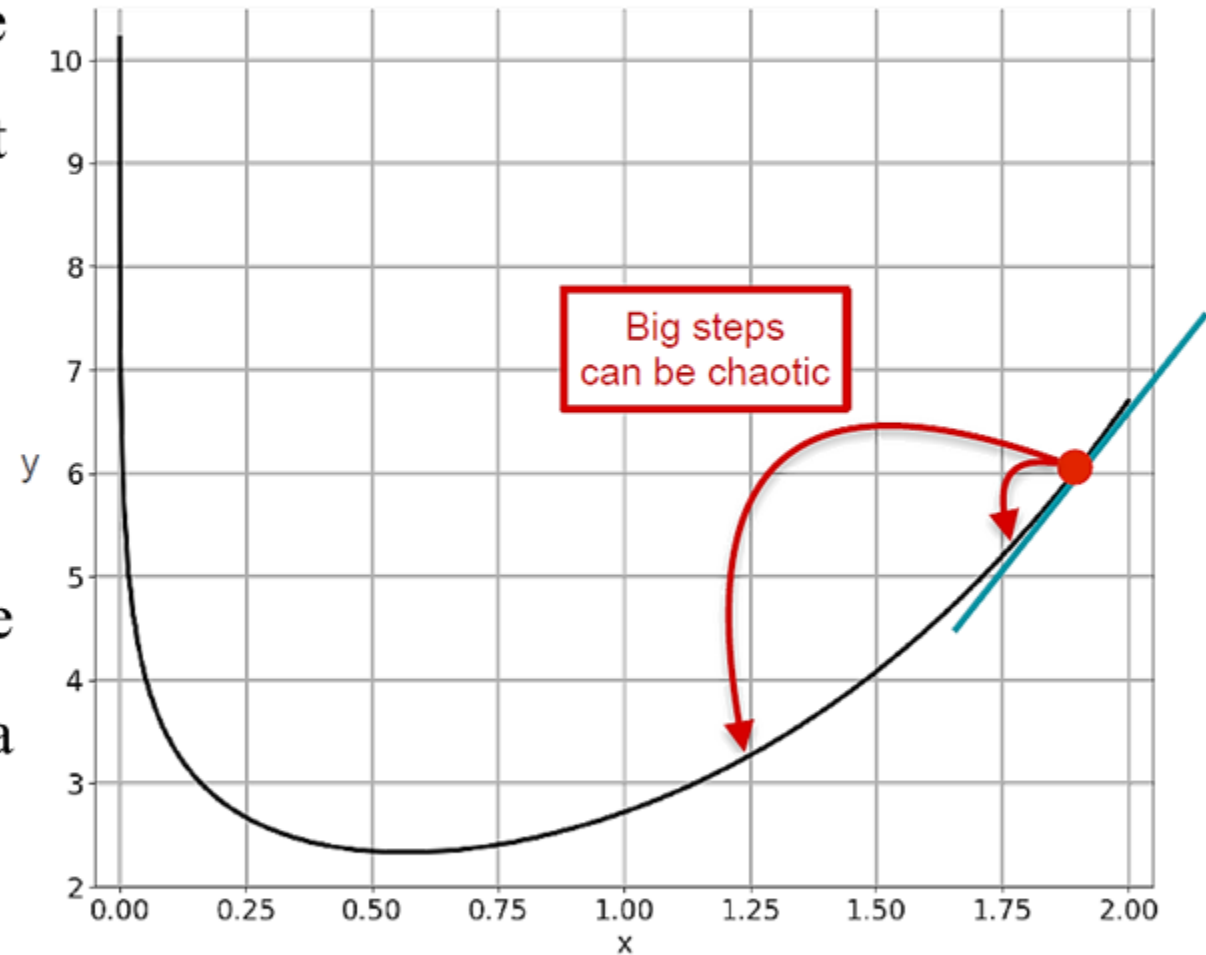
If we want a new point that is closer to the minimum, then that should be the old point minus the slope.

$$\text{New Point} = \text{Old Point} - \text{Slope}$$

$$x_{\text{new}} = x_{\text{old}} - f'(x_{\text{old}})$$

To take small steps so that we do not miss the minimum point, simply multiply the slope by a small number called the “*Learning Rate*”.

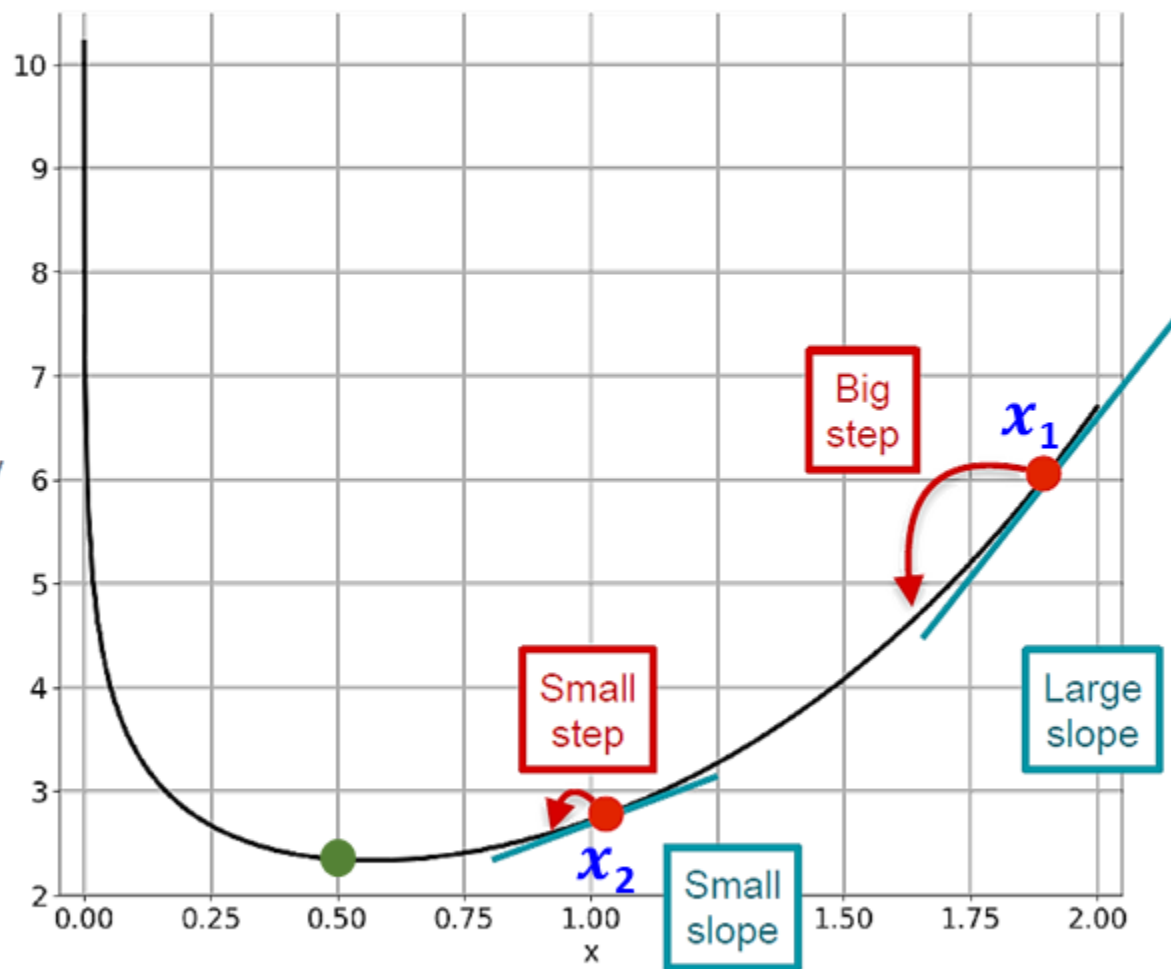
$$x_{\text{new}} = x_{\text{old}} - \alpha f'(x_{\text{old}})$$



Gradient Descent

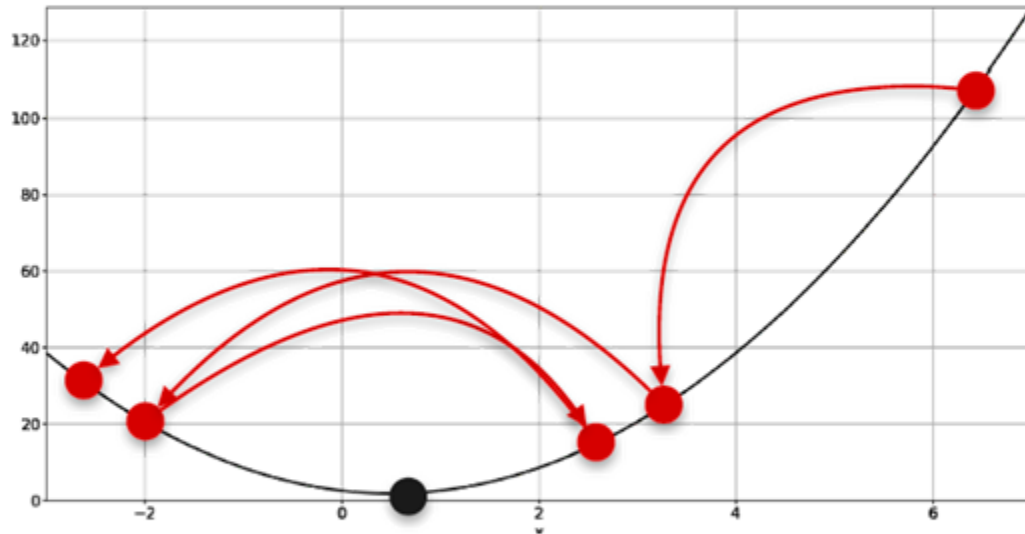
At x_1 the steep is large, so the slope is a big value, and so the step to the next point will be big step. At x_2 the steep is low, so the slope is a small value, and so the step to the next pointy will be small step.

$$x_{new} = x_{old} - \alpha f'(x_{old})$$



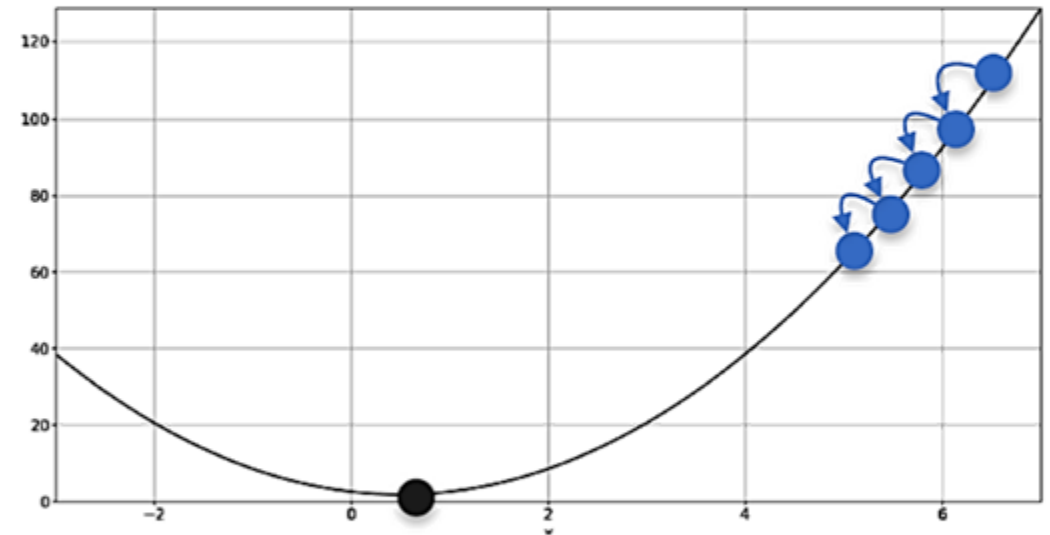
Finding a suitable learning rate

Too large



If the learning rate is too large, we may miss the minimum and never find it because the steps are too big.

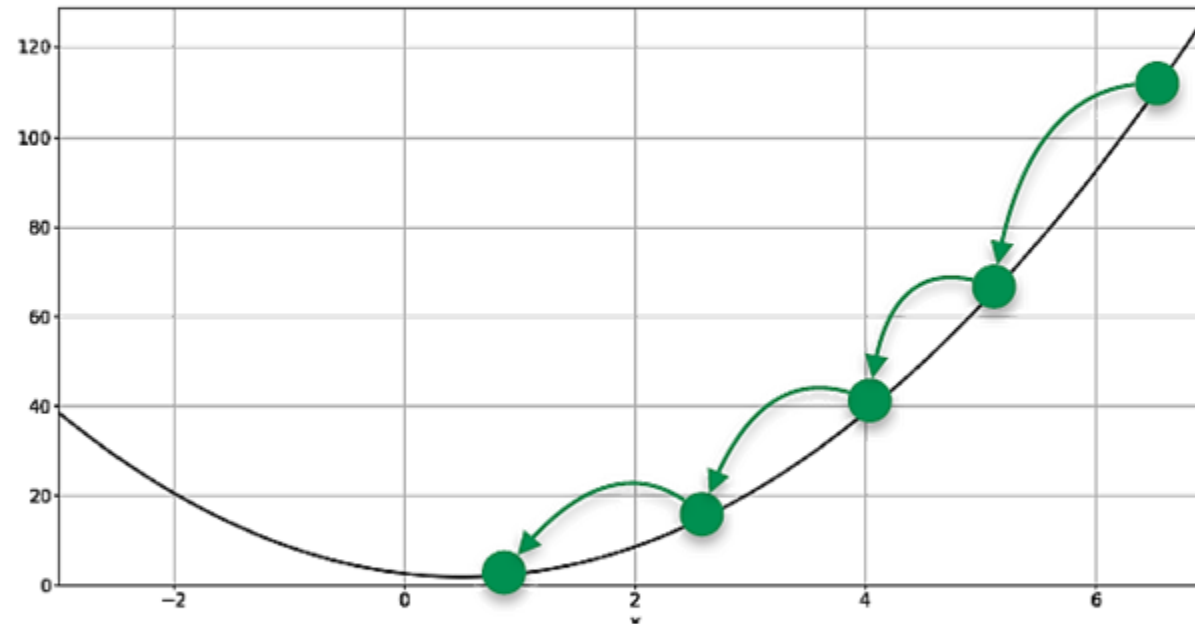
Too small



If the learning rate is too small, it takes forever to reach the minimum or perhaps never reach it because the steps are too small.

Gradient Descent

Just right



The Learning rate has to be just right.

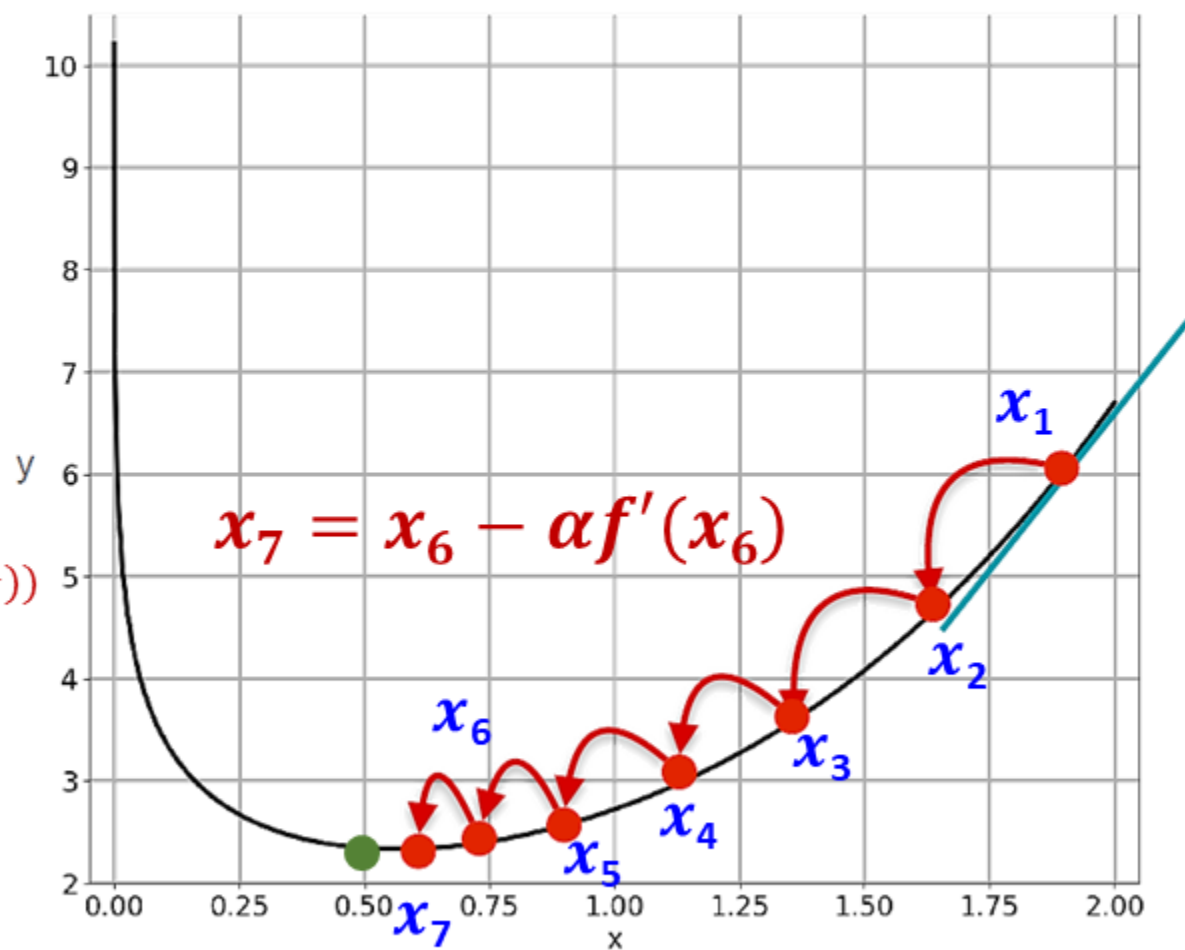
Gradient Descent:

Target: Finding the minimum of $f(x)$

Step (1): Define the learning rate (α) and choose a starting point (x_0)

Step (2): Update $x_{kn} = xk_{(n-1)} - \alpha f'(xk_{(n-1)})$

Step (3): Repeat (Step 2) until the minimum point is reached



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

Starting:

$(x) = 0.05$ and learning rate $(\alpha) = 0.005$

$$f'(0.05) = -18.9$$

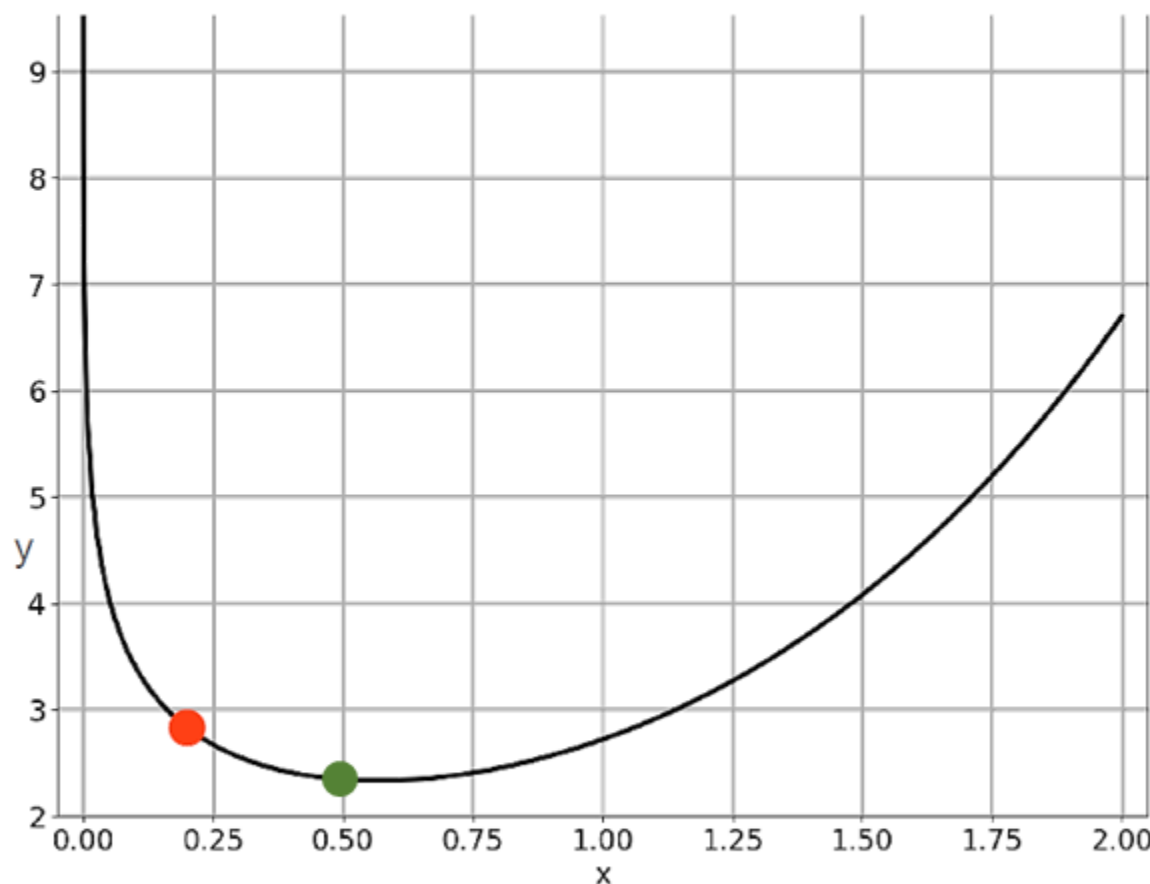
$$x_{new} = 0.05 - 0.005f'(0.05)$$

$$x_{new} = 0.1447 \rightarrow f'(0.1447) = -5.7552$$

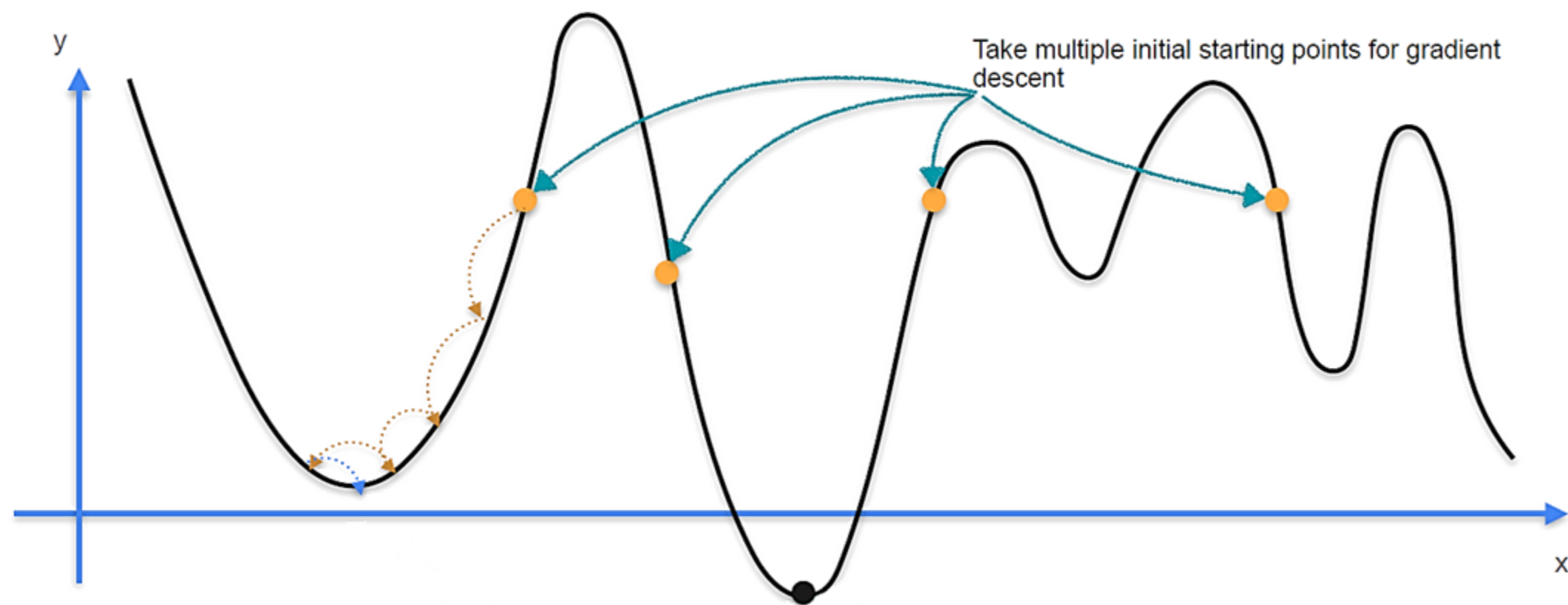
$$x_{new} = 0.1447 - 0.005f'(0.1447)$$

$$x_{new} = 0.1735$$

Repeat



Gradient Descent



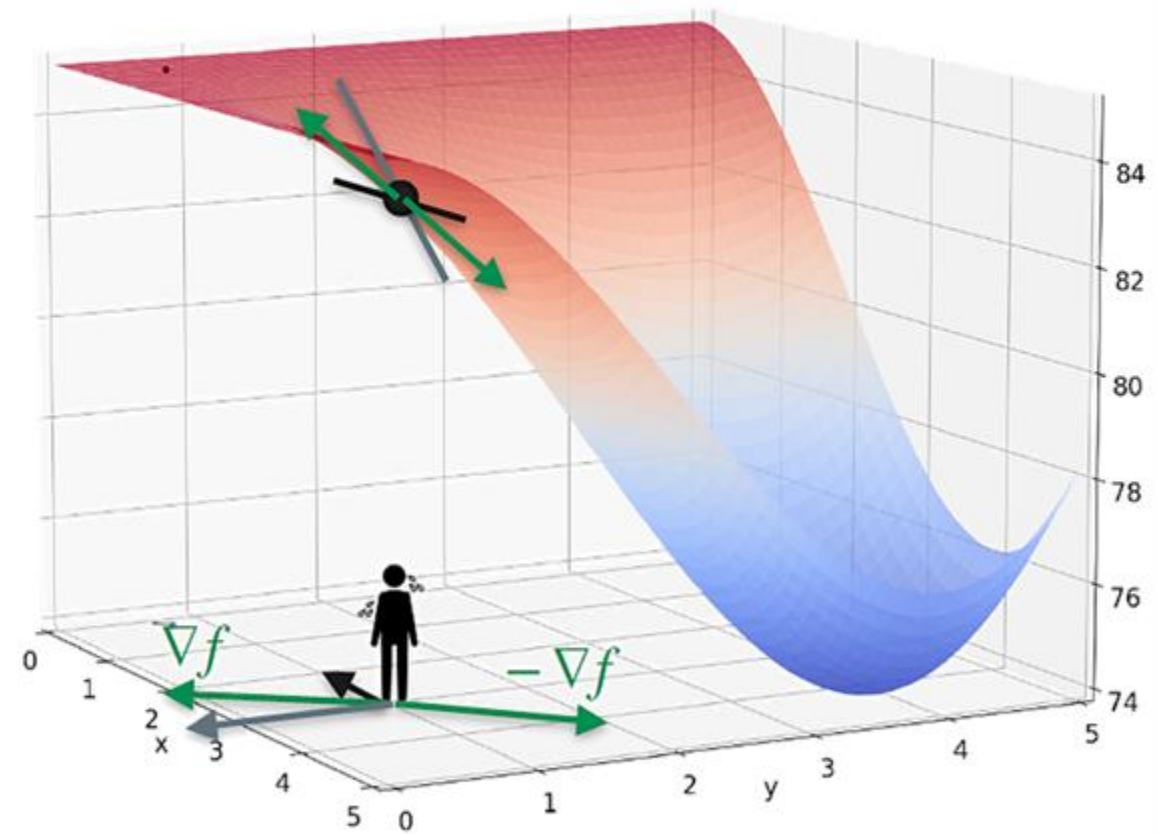
Run the gradient descent algorithm many times with many different starting points .Increase the chances of reaching the minimum point.

Initial position: (x_0, y_0)

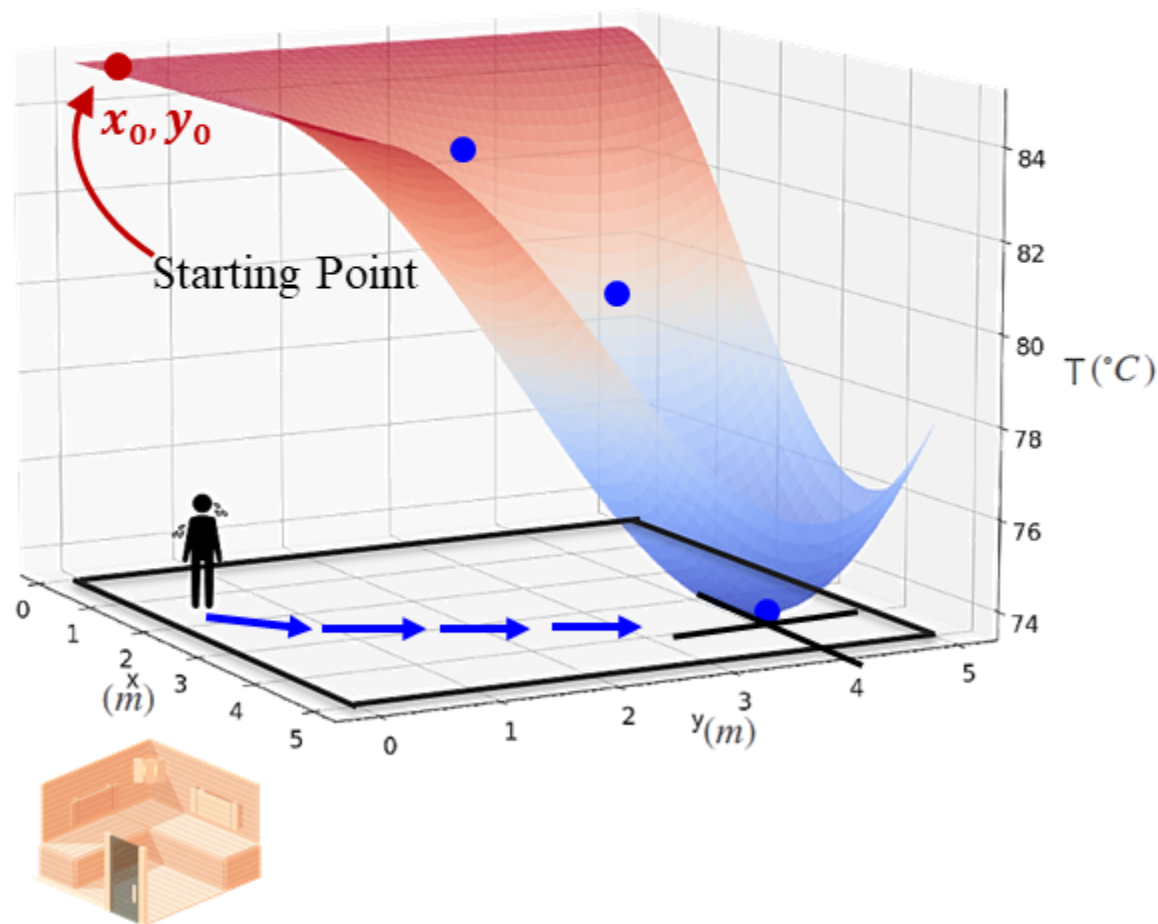
Direction of Gradient Ascent: ∇f

Direction of Gradient Descent: $-\nabla f$

$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = \begin{bmatrix} x_{old} \\ y_{old} \end{bmatrix} - \alpha \nabla f(x_{old}, y_{old})$$



$$T = f(x, y) = 85 - \frac{1}{90} x^2 (x - 6) y^2 (y - 6) \quad \text{Let } x_0 = 0.5, y_0 = 0.6, \text{ and } \alpha = 0.05$$



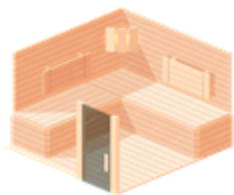
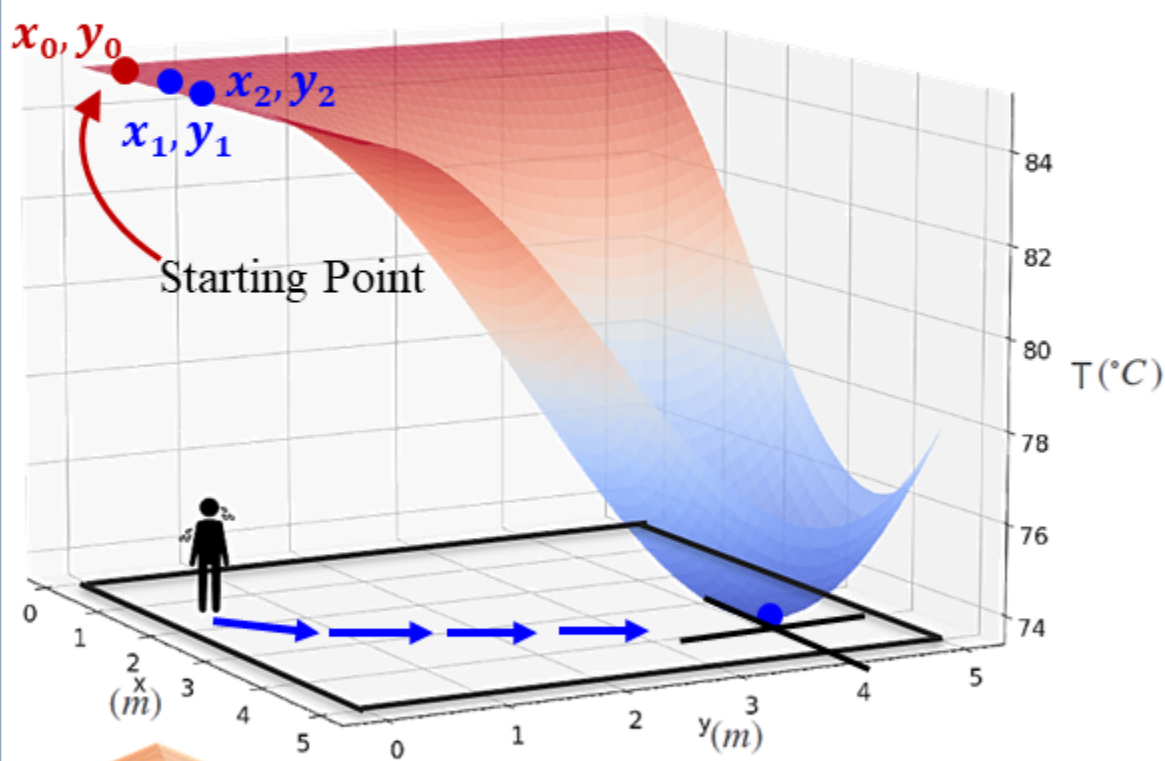
$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} -\frac{1}{90} x (3x - 12) y^2 (y - 6) \\ -\frac{1}{90} x^2 (x - 6) y (3y - 12) \end{bmatrix}$$

$$\nabla f(0.5, 0.6) = \begin{bmatrix} 0.1134 \\ -0.0935 \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} - 0.05 \nabla f(0.5, 0.6)$$

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} 0.5057 \\ 0.6047 \end{bmatrix}$$

$$T = f(x, y) = 85 - \frac{1}{90} x^2 (x - 6) y^2 (y - 6) \quad \text{Let } x_0 = 0.5, y_0 = 0.6, \text{ and } \alpha = 0.05$$



$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} -\frac{1}{90} x (3x - 12) y^2 (y - 6) \\ -\frac{1}{90} x^2 (x - 6) y (3y - 12) \end{bmatrix}$$

$$\nabla f(0.5057, 0.6047) = \begin{bmatrix} -0.1162 \\ -0.0961 \end{bmatrix}$$

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} - 0.05 \nabla f(0.5057, 0.6047)$$

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} 0.5115 \\ 0.6095 \end{bmatrix}$$

Repeat

Gradient Descent:

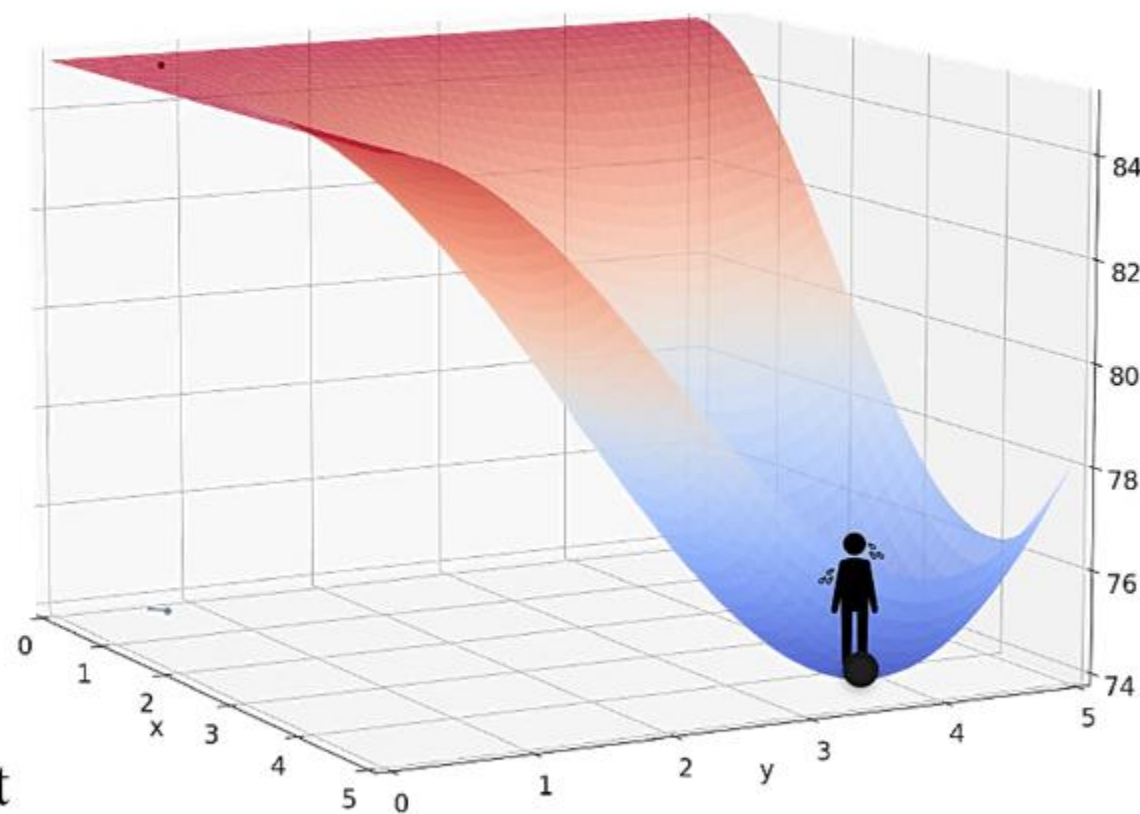
Target: Finding the minimum of $f(x)$

Step (1): Define the learning rate (α) and choose a starting point (x_0, y_0)

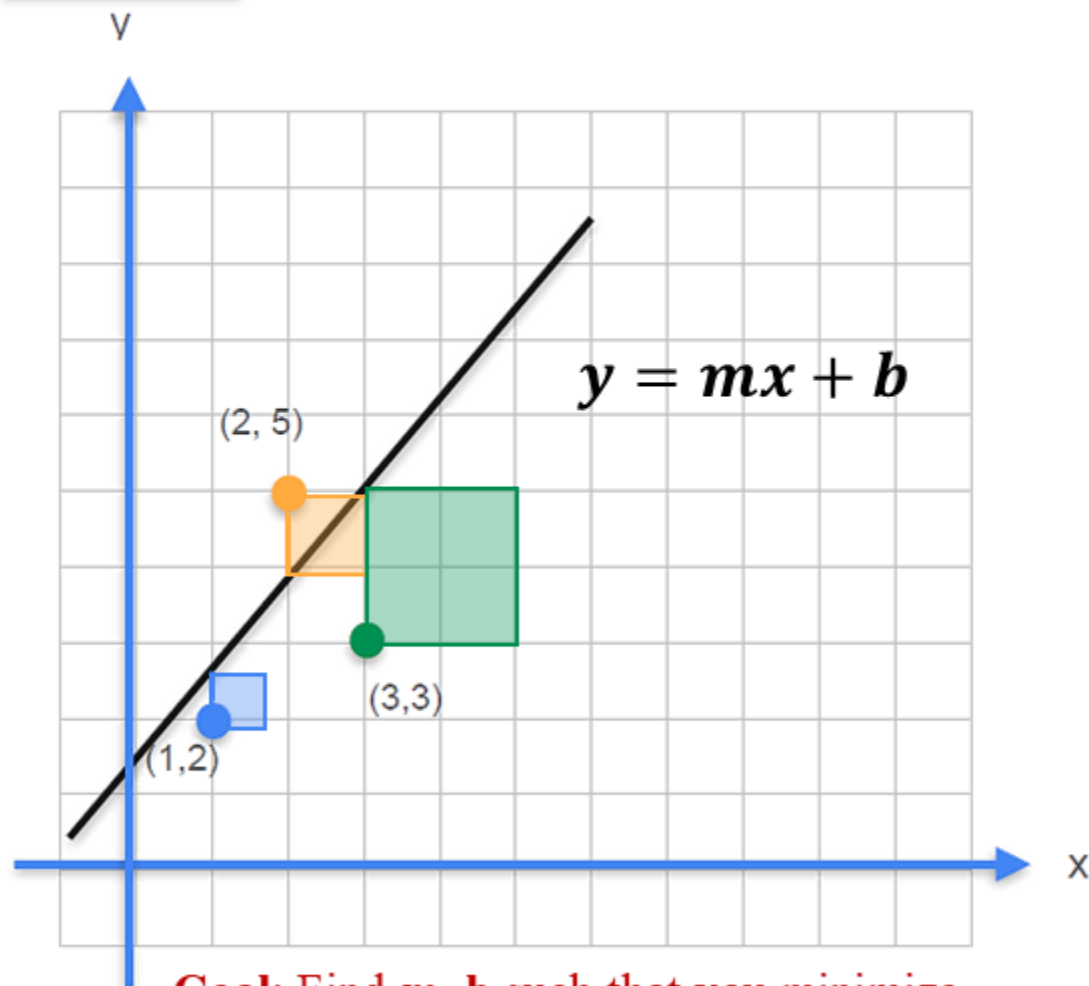
Step (2): Update

$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = \begin{bmatrix} x_{old} \\ y_{old} \end{bmatrix} - \alpha \nabla f(x_{old}, y_{old})$$

Step (3): Repeat (Step 2) until the minimum point (x, y) is reached



Gradient Descent



Goal: Find m , b such that you minimize the sum of squares cost.



(1, 2)



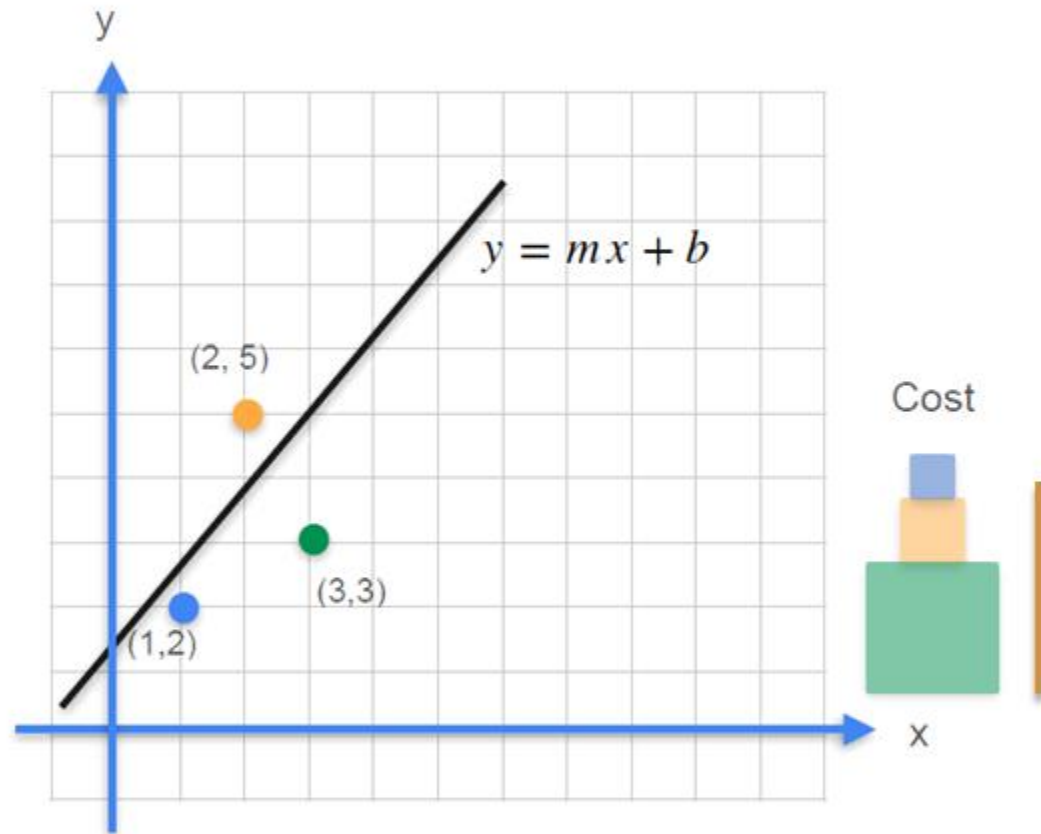
(2, 5)



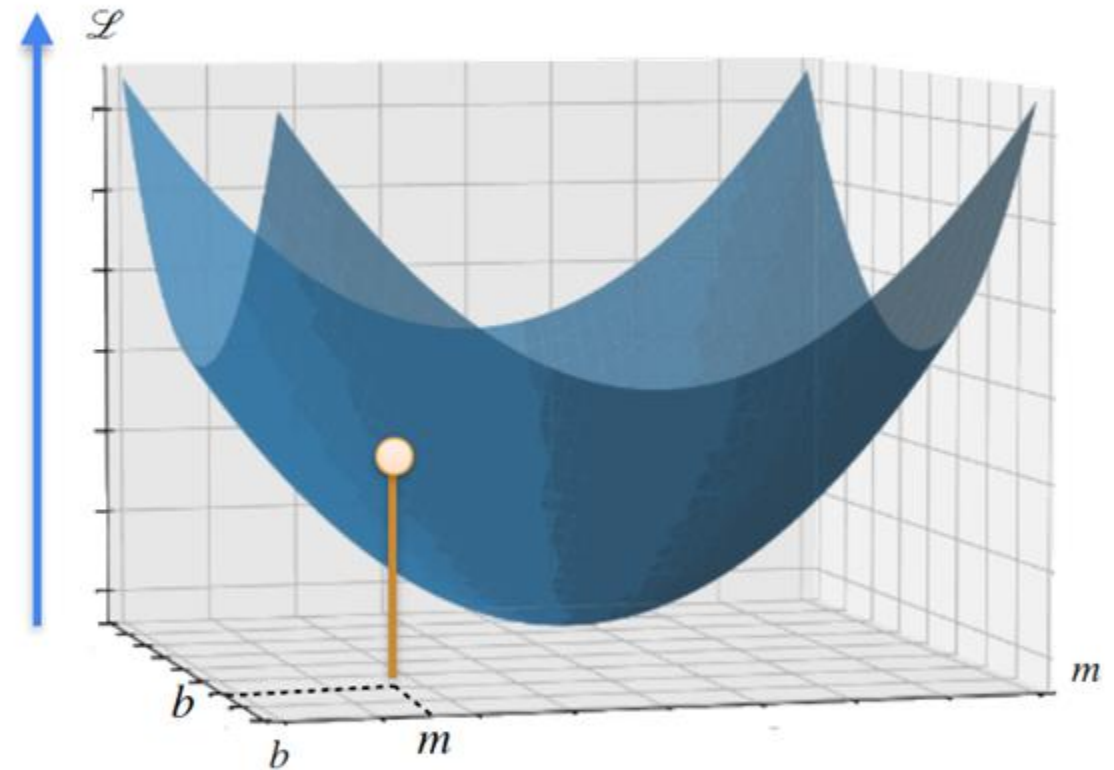
(3, 3)

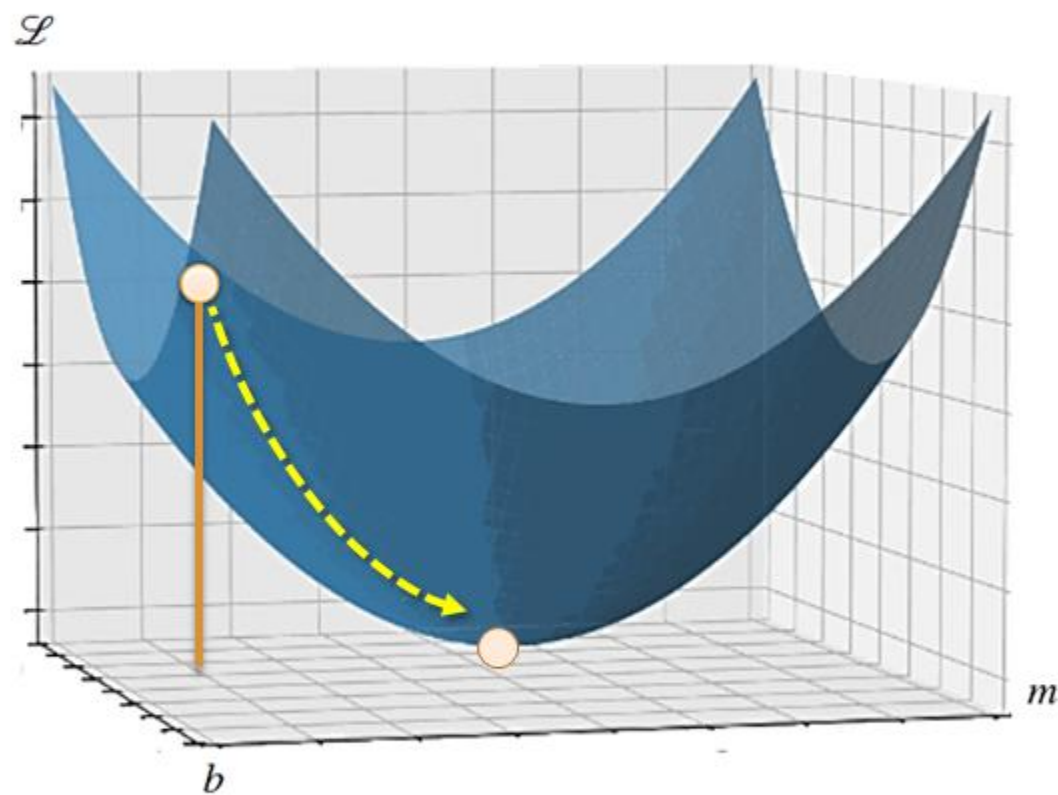
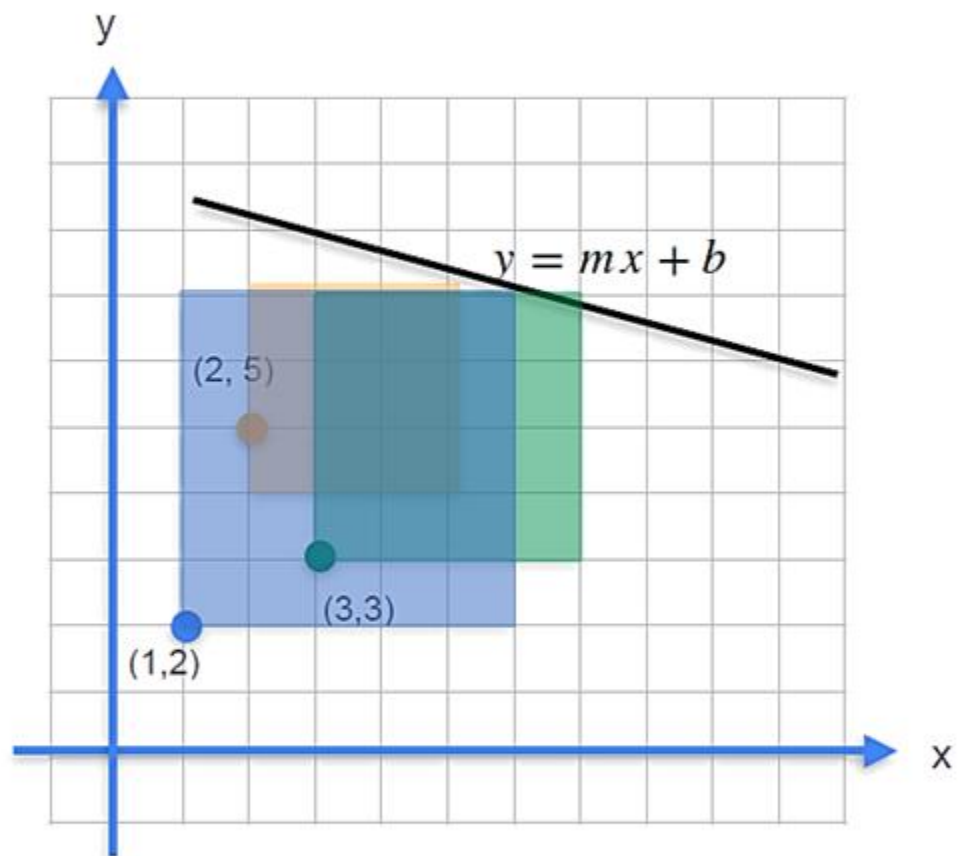
Finding the optimal place for a fiber line connection that goes in a straight line in such a way that you reduce the total cost of connecting to the three power lines by connecting a wire to touch the fiber line, but the wire needs to be parallel to the y-axis. And the cost for connection is the square of the length of the wires.

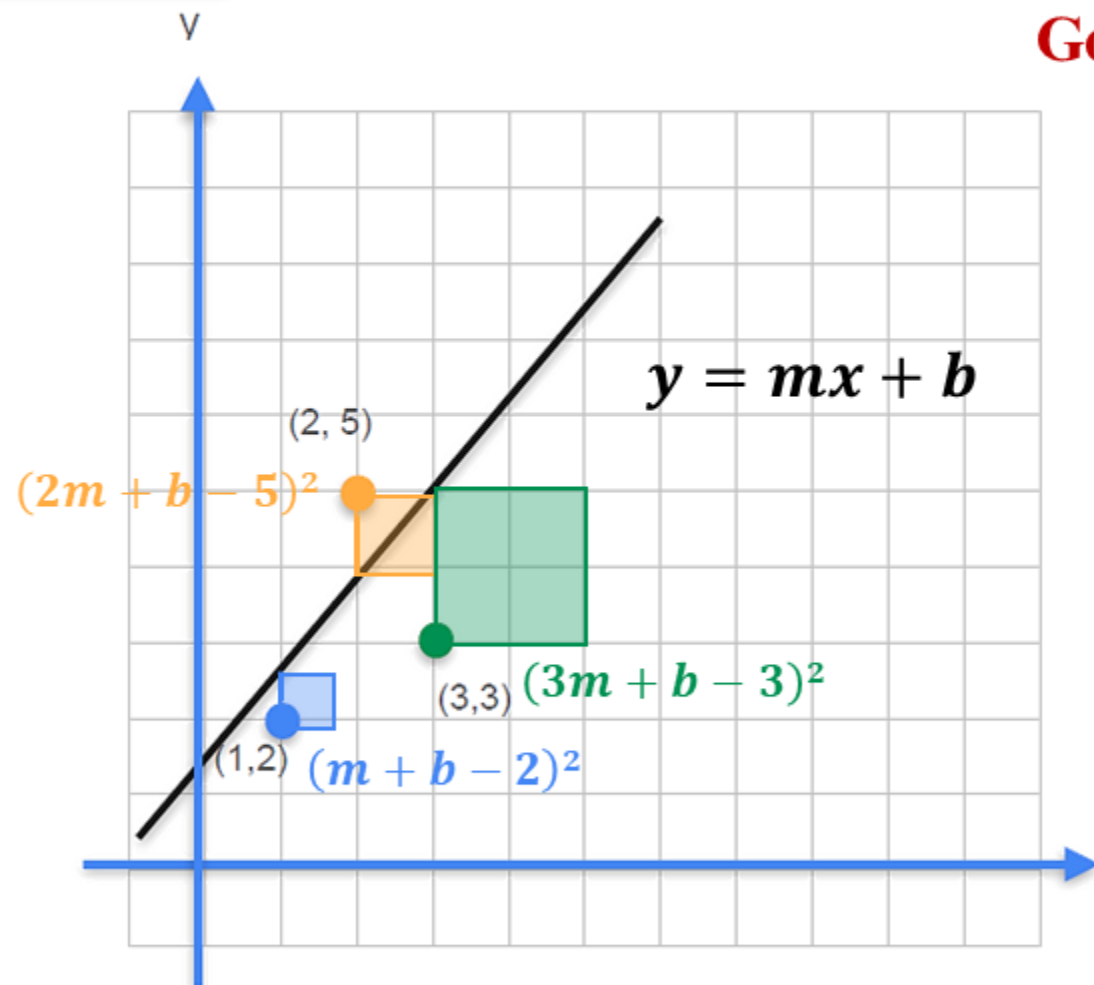
Gradient Descent



Square loss







Goal: Find m , b to minimize the sum of squares cost.

$$E(m, b) = 14m^2 + 3b^2 + 38 + 12mb - 42m - 20b$$

$$E(m, b) = \text{where } (m) = \frac{1}{2}, \text{ and } (b) = \frac{7}{3}$$

$$\therefore E(m, b) \cong 4.167$$

$$\nabla E = 14m^2 + 3b^2 + 38 + 12mb - 42m - 20b$$

Start with (m_0, b_0) , and define the learning rate (α)

Iterate:

$$(m_{New}, b_{New}) = (m_{Old}, b_{Old}) - \alpha \nabla E(m_{Old}, b_{Old})$$

Gradient Descent



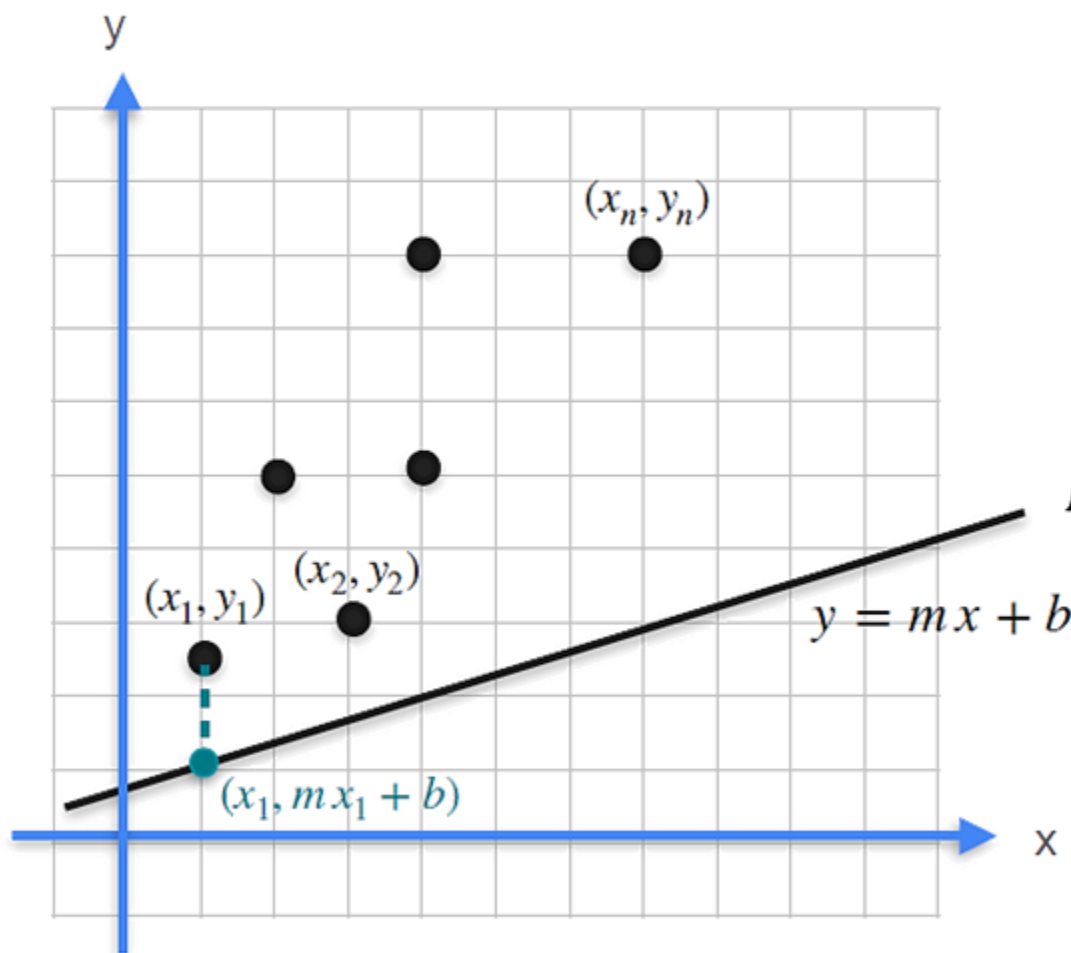
TV advertisement
budget



Number of sales

TV Budget	Sales
230.1	22.1
44.5	10.4
17.2	9.3

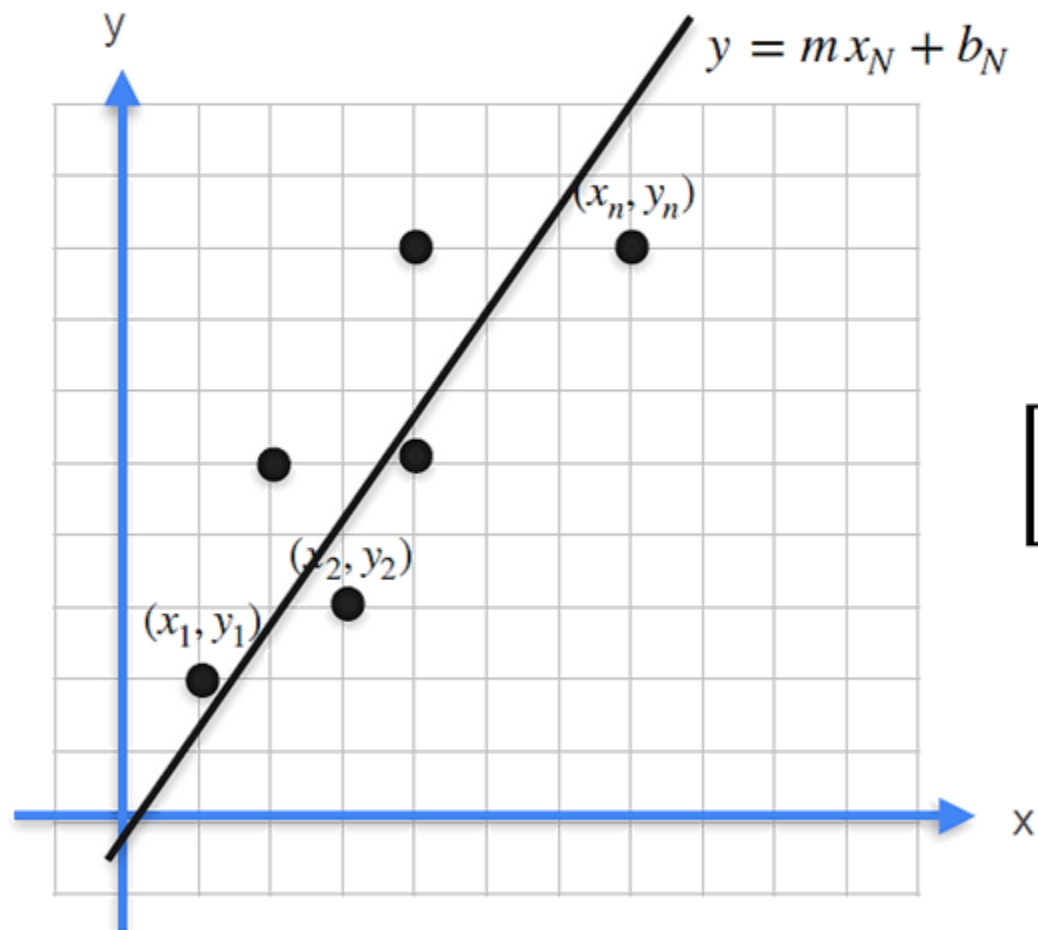
Goal: *Predict sales in terms of TV budget*



$$\text{Loss } (L) = (mx_1 + b - y_1)^2$$

$$L = \frac{1}{n} [(mx_1 + b - y_1)^2 + \dots + (mx_n + b - y_n)^2]$$

$$\begin{bmatrix} m_1 \\ b_1 \end{bmatrix} = \begin{bmatrix} m_0 \\ b_0 \end{bmatrix} - \alpha \nabla L(m_0, b_0)$$



$$\begin{bmatrix} m_N \\ b_N \end{bmatrix} = \begin{bmatrix} m_{N-1} \\ b_{N-1} \end{bmatrix} - \alpha \nabla L(m_{N-1}, b_{N-1})$$

We are going to toss a coin 10 times and we're going to look at the results. If the results are seven heads followed by three tails, then we win. If they're not, then we don't win.



We must pick the coin we want to use in the game from 3 coins and it can be a biased coin.

Coin 1



70% 30%

Coin 2



50% 50%

Coin 3



30% 70%

Which of the three coins would you choose to maximize your chances of winning?



											Total	
Coin (1)	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.3	0.3	0.3	$0.7^7 \times 0.3^3$	0.00222
Coin (2)	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	$0.5^7 \times 0.5^3$	0.00097
Coin (3)	0.3	0.3	0.3	0.7	0.7	0.7	0.7	0.7	0.7	0.7	$0.3^3 \times 0.7^7$	0.00008

Coin (1) is the best coin for winning.

Pick a coin and say that the probability of landing in heads is p and the probability of landing in tails is $(1 - p)$ because they're supposed to equal to one.

$$\text{Chance of winning } f(p) = p^7(1 - p)^3$$

Our goal is to maximize of $f(p)$

Product Rule

Chain Rule

$$\begin{aligned}\frac{d}{dp}(p^7(1-p)^3) &= \frac{d(p^7)}{dp}(1-p)^3 + p^7 \frac{d((1-p)^3)}{dp} \\&= 7p^6 \times (1-p)^3 + p^7 \times 3(1-p)^2 \times \frac{d}{dp}(1-p) \\&= 7p^6 \times (1-p)^3 + p^7 \times 3(1-p)^2 \times (-1) \\&= p^6(1-p)^2 [7(1-p) - 3p] \\&= p^6(1-p)^2 [7 - 10p] = 0\end{aligned}$$

$p = 0$ $p = 1$ $p = 0.7$

$$\log f(p) = \log(p^7 (1-p)^3) = \log(p^7) + \log((1-p)^3)$$

$$= 7 \log(p) + 3 \log(1-p) = F(p)$$

$$\frac{dF(p)}{dp} = \frac{d}{dp} (7 \log(p) + 3 \log(1-p)) = 7 \frac{1}{p} + 3 \frac{1}{1-p} (-1)$$

$$= \frac{7(1-p) - 3p}{p(1-p)} = 0$$

$$= 7(1-p) - 3p = 0$$

$$p = 0.7$$

 $-F(p)$

→ Log-Loss

When to use Log-Loss in Machine Learning?

Why couldn't we simply solve using the derivative as previously?

Using product and chain rules for more than four functions will be very messy and hard to work out.

Another reason why we use logarithms is because the product of tiny things is tiny. Imagine if you have a probability that is the product of 1,000 probabilities. That's the product of 1,000 numbers, between 0 and 1. That number could be really small. On the other hand, if it's the logarithm, the logarithm of a very small number is a very large negative number and computers can handle those numbers.

Function	Derivatives
$f(x) = \sin(x)$	$f'(x) = \cos(x)$
$f(x) = \cos(x)$	$f'(x) = -\sin(x)$
$f^{-1}(x)$	$\frac{d}{dx}f^{-1}(x) = \frac{1}{f'(x)}$
$f(x) = e^x$	$f'(x) = e^x$
$f(x) = \log(x)$	$\frac{d}{dx}\log(x) = \frac{1}{x}$
$f(x) = cg(x)$	$f'(x) = cg'(x)$
$f(x) = h(x) + g(x)$	$f'(x) = h'(x) + g'(x)$
$f(x) = h(x) \times g(x)$	$f'(x) = g(x)h'(x) + h(x)g'(x)$
$f(g(h(x)))$	$\frac{d}{dx}f(g(h(x))) = \frac{df}{dg} \times \frac{dg}{dh} \times \frac{dh}{dx}$

Method	Rule
Square Loss function	$x = \frac{a_1 + a_2 + a_3 + \dots + a_n}{n}$
The Gradient of $f(x, y)$	$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} 2x \\ 2y \end{bmatrix}$
Gradient Descent	$x_{new} = x_{old} - \alpha f'(x_{old})$

