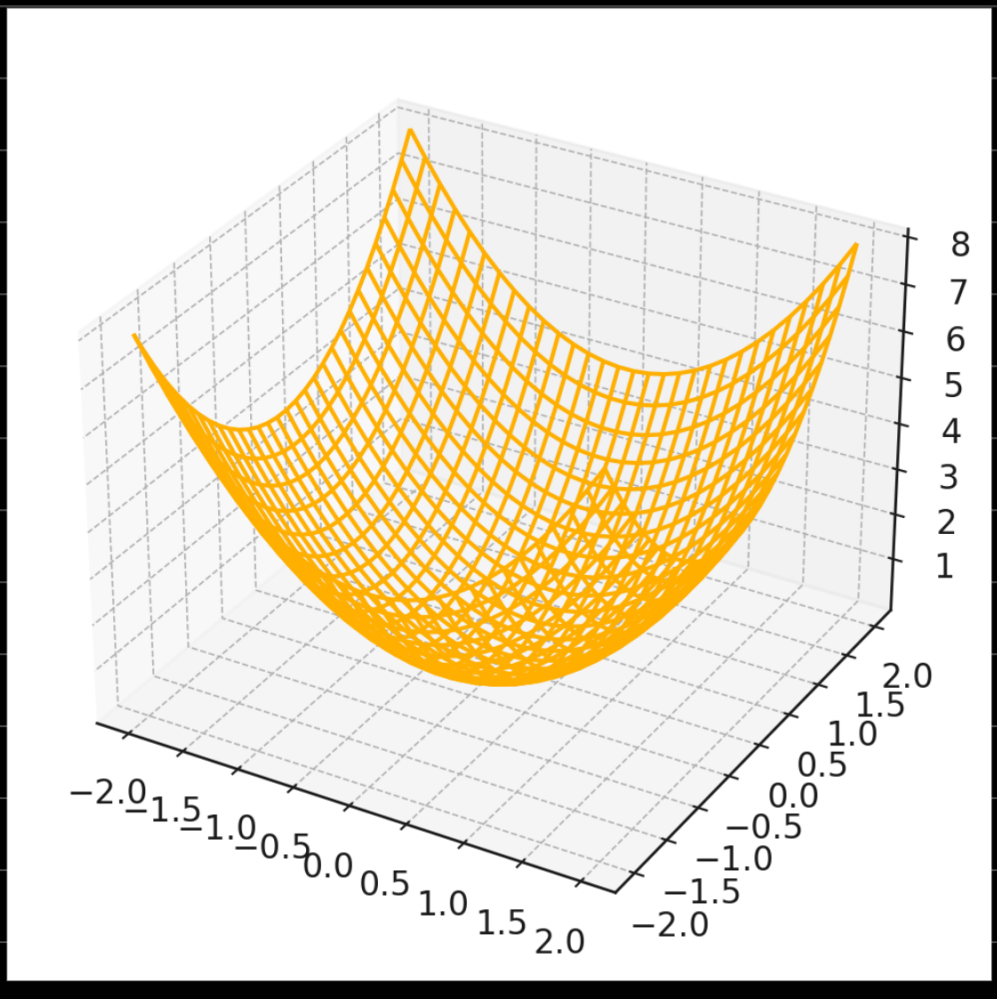
Gradient Descent

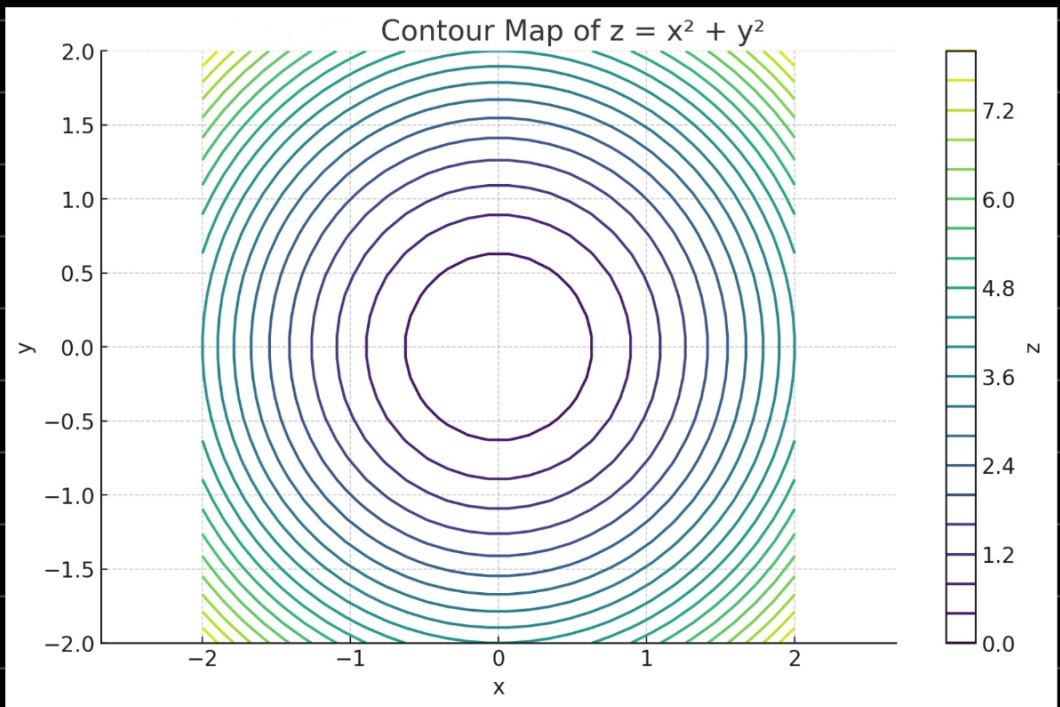
* **Contour Map / Level Curve** : In gradient descent, a contour map is a visual representation of the cost or objective function being optimized. It's particularly useful in understanding how the algorithm navigates through the parameter space to find the minimum of the function.

In gradient descent, the goal is to find the minimum of the cost function. On a contour map, the minimum corresponds to the centre of the deepest valley. The gradient descent algorithm starts from some initial point on the contour map and iteratively moves towards the direction of steepest descent (negative gradient) until it reaches the minimum

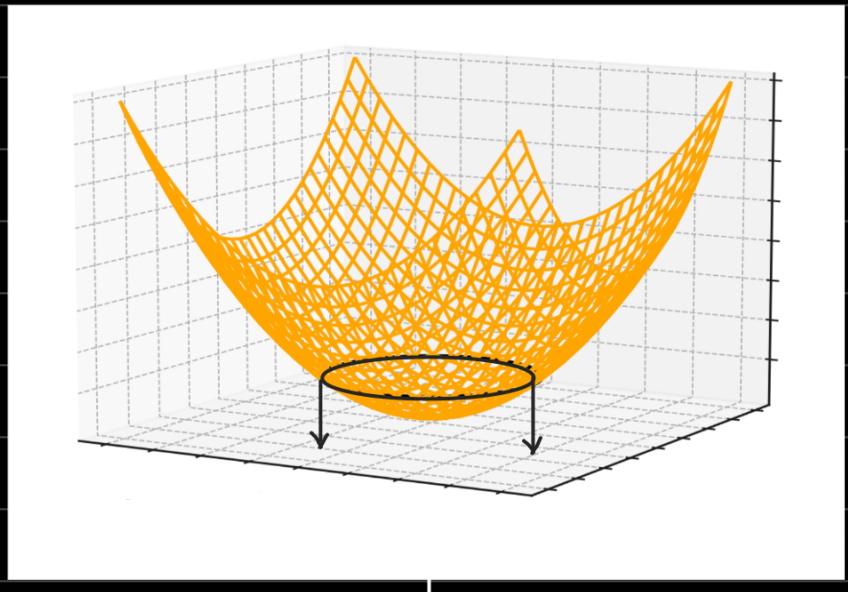
- Plot of some cost function $z = x^2 + y^2$:



Contour lines of $z = x^2 + y^2$:

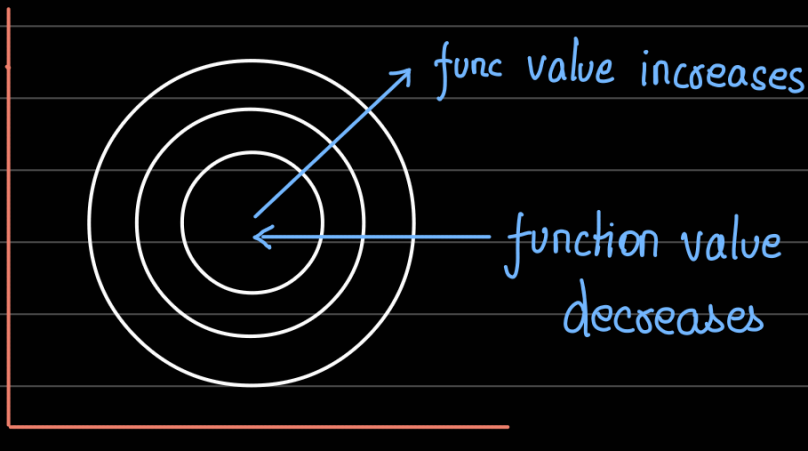


Eg → $x^2 + y^2 = 1$ →

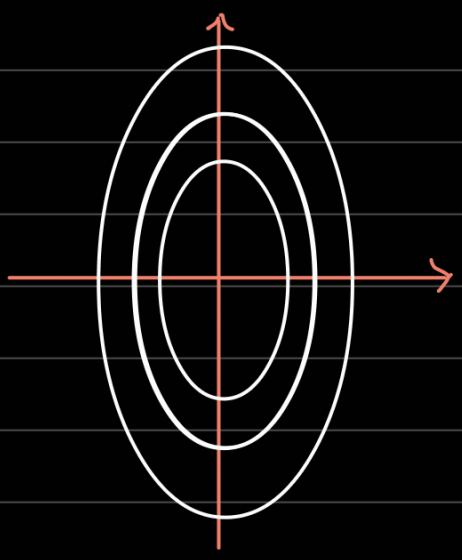


Projection on X-Y Plane

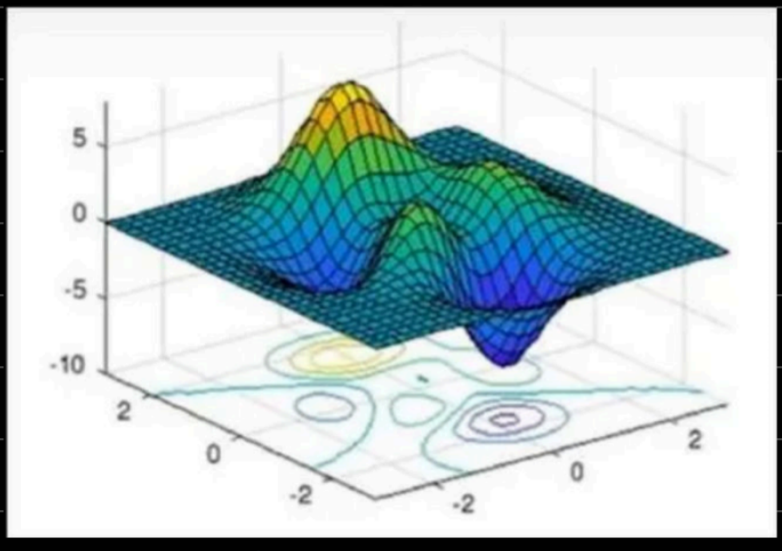
- The value of function increase as we go away from centre.
- Similarly, function value decreases as we go towards centre.



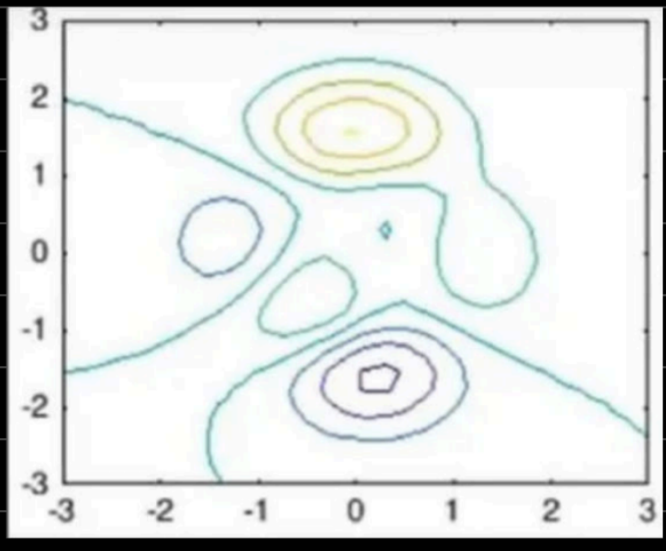
Example: Level curve of $3x^2+y^2$



* Example Function:

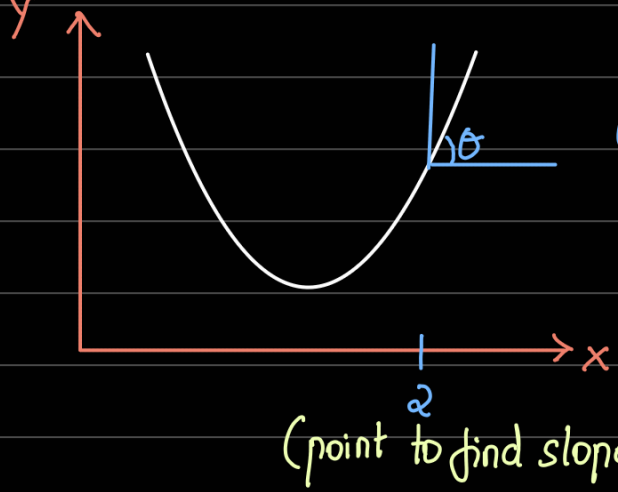


FUNCTION PLOT



CONTOUR MAP

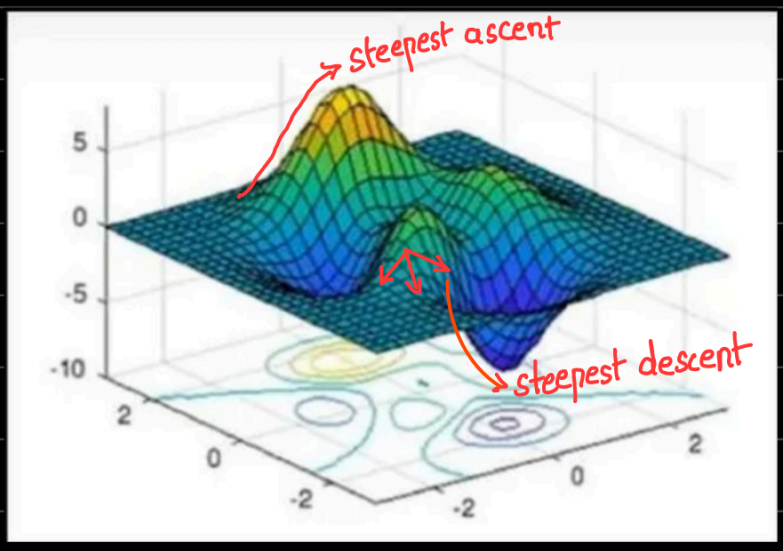
* Finding slope on 2-D curve →



$\theta = \frac{dy}{dx}$

In 2-D, we only need the point at which we want to find slope.

✳ Finding slope on 3-D curve



In 3-D curve, alone point is not sufficient to find slope. We also need direction where we want to find slope.

Gradient → It is a vector (object with both magnitude and direction) that points in the direction of steepest ascent (max slope) of the function.

Note → Gradient is used to find direction of steepest slope. After getting gradient of the function, we can get rate of change of a function in given direction.

✳ Rate of change of a function in Given Direction.

Let ∇f be the gradient (max slope)
Let u be the direction of interest (at max slope direction is always 0 because gradient and max slope face at same direction)

Hence $\frac{\partial f}{\partial u} = \nabla f \cdot \hat{u}$

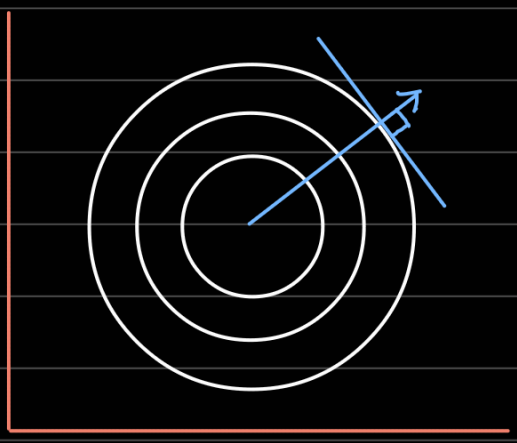
$= |\nabla f|\,|\hat{u}| \cos\theta$

+1 (steepest ascent) $\theta = 0°$

-1 (steepest descent) $\theta = 180°$

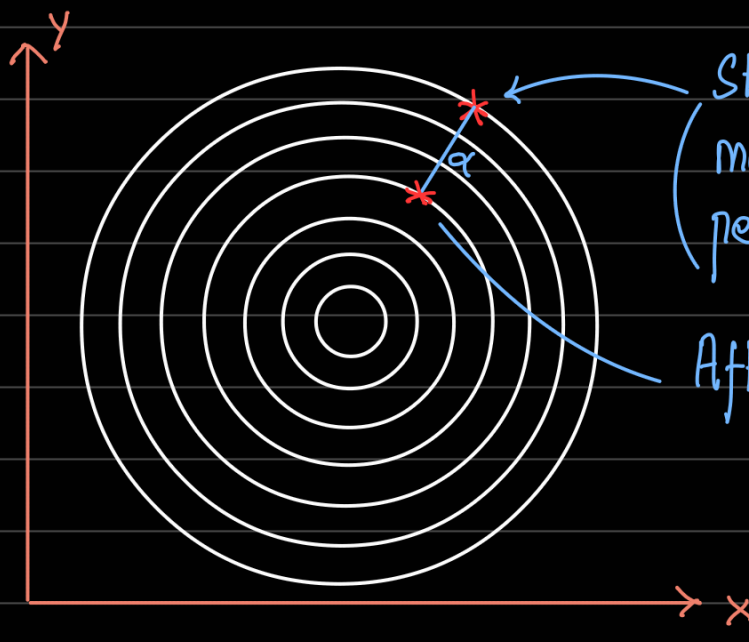
at $\cos\theta = +1 \Rightarrow |\nabla f|\,|\hat{u}| \times 1$ (at gradient) (steepest ascent)

at $\cos\theta = -1 \Rightarrow |\nabla f|\,|\hat{u}| \times -1$ (opp. to gradient) (steepest descent)



Direction of gradient is always perpendicular to level curve

* At any given point gradient gives the direction of Maximum slope in function



Starting point $\begin{bmatrix} x \\ y \end{bmatrix}^{old}$

moving α in the direction perpendicularly opposite to gradient.

After moving α, we get $\begin{bmatrix} x \\ y \end{bmatrix}^{new}$

$$\underbrace{\begin{bmatrix} x \\ y \end{bmatrix}^{old} - \begin{bmatrix} x \\ y \end{bmatrix}^{new}}_{\text{Change}} = \alpha \quad (-\nabla f)$$

Change = moved by α * Opposite to gradient

$$\begin{bmatrix} x \\ y \end{bmatrix}^{new} = \begin{bmatrix} x \\ y \end{bmatrix}^{old} - \alpha(\nabla f)$$

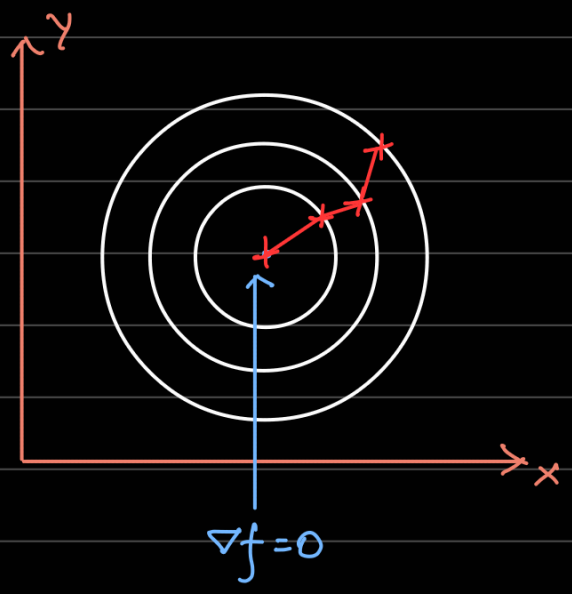
α: Learning Rate

∇f: gradient at old point

When gradient becomes 0, i.e. $\nabla f = 0$

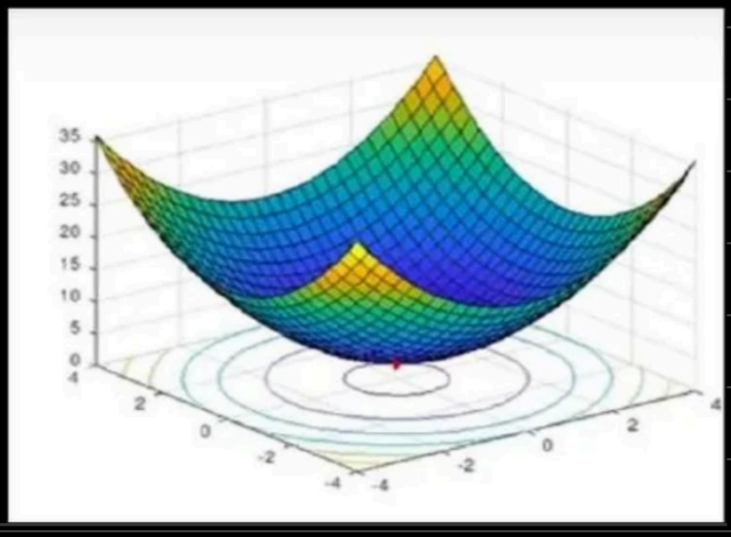
$$\begin{bmatrix} x \\ y \end{bmatrix}^{new} = \begin{bmatrix} x \\ y \end{bmatrix}^{old}$$

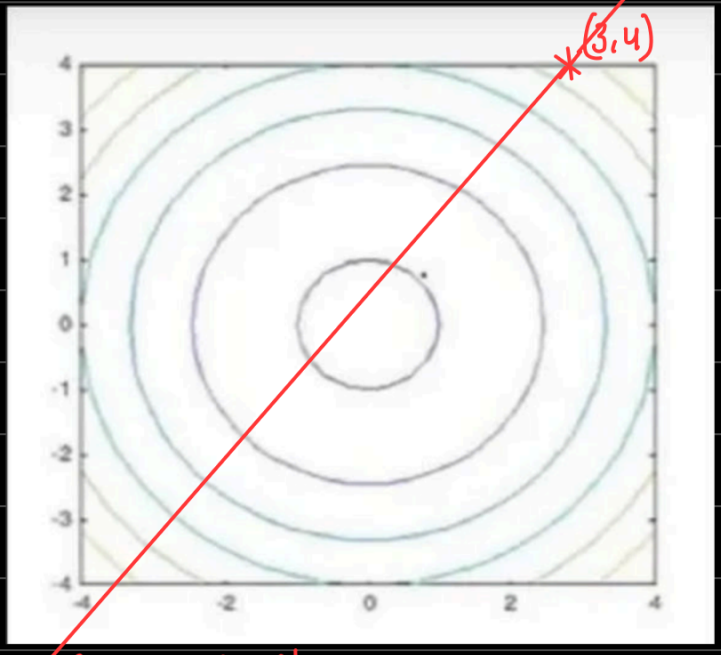
(Algorithm will stop updating)



Example → $f(x,y) = x^2 + y^2 + 4$

$f(0,0) = 4$ (required for minimum gradient $\nabla f \approx 0$)





Scenario I →

let $\alpha = 2$

Starting point $\begin{bmatrix} 3 \\ 4 \end{bmatrix}$

$f(3,4) = 9 + 16 + 4$

$= 29$

$$\begin{bmatrix} x \\ y \end{bmatrix}^{new} = \begin{bmatrix} 3 \\ 4 \end{bmatrix} - 2 \begin{bmatrix} df/dx \\ df/dy \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \end{bmatrix}^{new} = \begin{bmatrix} 3 \\ 4 \end{bmatrix} - 2\begin{bmatrix} 2x \\ 2y \end{bmatrix}$$

$$= \begin{bmatrix} 3 \\ 4 \end{bmatrix} - \begin{bmatrix} 12 \\ 16 \end{bmatrix} \Rightarrow \boxed{\begin{bmatrix} x \\ y \end{bmatrix}^{new} = \begin{bmatrix} -9 \\ -12 \end{bmatrix}}$$ 1st iter

$$f(-9,-12) = 81 + 144 + 4$$
$$= \underline{229}$$

Second Iteration →

$$\begin{bmatrix} x \\ y \end{bmatrix}^{new} = \begin{bmatrix} -9 \\ -12 \end{bmatrix} - 2\begin{bmatrix} -18 \\ -24 \end{bmatrix}$$

$$= \begin{bmatrix} -9 \\ -12 \end{bmatrix} + \begin{bmatrix} 36 \\ 48 \end{bmatrix} \Rightarrow \boxed{\begin{bmatrix} x \\ y \end{bmatrix}^{new} = \begin{bmatrix} 27 \\ 36 \end{bmatrix}}$$ 2nd Iter

$$f(27,36) = 2029$$

Here, our algorithm is diverging (overshooting) due to large value of α (learning rate)

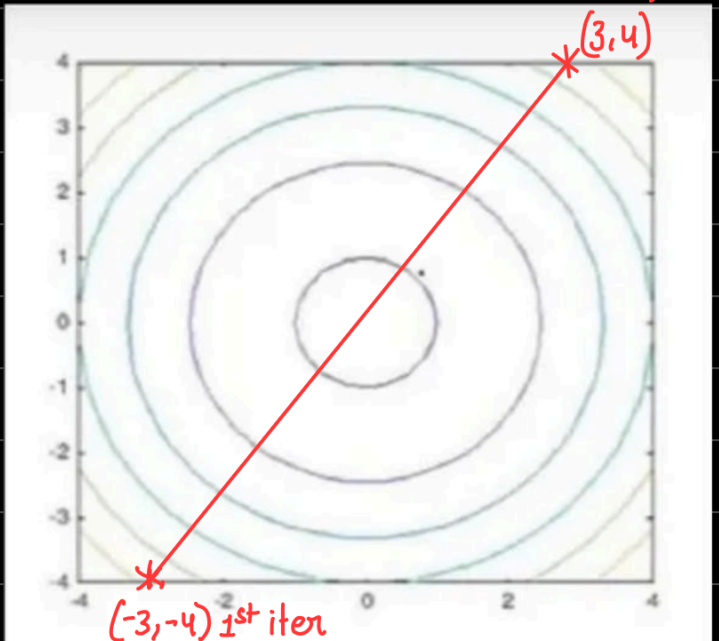
Scenario II →

let $\alpha = 1$

Starting point $\begin{bmatrix} 3 \\ 4 \end{bmatrix}$

$$f(3,4) = 9 + 16 + 4$$
$$= 29$$

$$\begin{bmatrix} x \\ y \end{bmatrix}^{new} = \begin{bmatrix} 3 \\ 4 \end{bmatrix} - 1\begin{bmatrix} df/dx \\ df/dy \end{bmatrix}$$



$$\begin{bmatrix} x \\ y \end{bmatrix}^{new} = \begin{bmatrix} 3 \\ 4 \end{bmatrix} - 1 \begin{bmatrix} 2x \\ 2y \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \end{bmatrix}^{new} = \begin{bmatrix} 3 \\ 4 \end{bmatrix} - \begin{bmatrix} 6 \\ 8 \end{bmatrix}$$

$$\boxed{\begin{bmatrix} x \\ y \end{bmatrix}^{new} = \begin{bmatrix} -3 \\ -4 \end{bmatrix}} \quad 1^{st} \text{ iter}$$

$$y(-3,-4) = 9+16+4 = \underline{\underline{29}}$$

Second iteration →

$$\begin{bmatrix} x \\ y \end{bmatrix}^{new} = \begin{bmatrix} -3 \\ -4 \end{bmatrix} - \begin{bmatrix} -6 \\ -4 \end{bmatrix}$$

$$\boxed{\begin{bmatrix} x \\ y \end{bmatrix}^{new} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}} \quad 2^{nd} \text{ iter}$$

$$y = \underline{\underline{29}}$$

Here, our algorithm is oscillating.

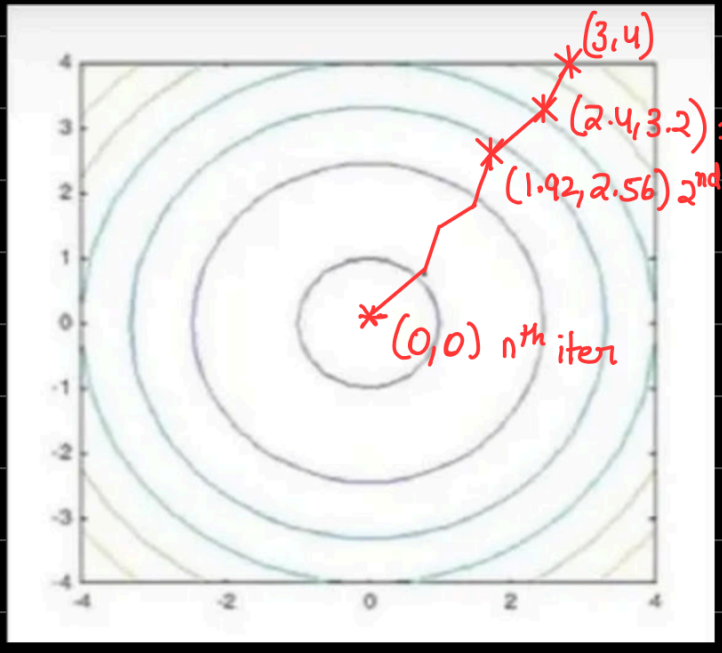
Scenario III →

let $\alpha = 0.1$

Starting point $\begin{bmatrix} 3 \\ 4 \end{bmatrix}$

$$f(3,4) = 9+16+4 = 29$$

$$\begin{bmatrix} x \\ y \end{bmatrix}^{new} = \begin{bmatrix} 3 \\ 4 \end{bmatrix} - 0.1 \begin{bmatrix} df/dx \\ df/dy \end{bmatrix}$$



$$\begin{bmatrix} x \\ y \end{bmatrix}^{new} = \begin{bmatrix} 3 \\ 4 \end{bmatrix} - 0.1 \begin{bmatrix} 2x \\ 2y \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \end{bmatrix}^{new} = \begin{bmatrix} 3 \\ 4 \end{bmatrix} - 0.1 \begin{bmatrix} 6 \\ 8 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \end{bmatrix}^{new} = \begin{bmatrix} 2.4 \\ 3.2 \end{bmatrix}$$ 1st iter

$$y = \underline{20}$$

Second iteration →

$$\begin{bmatrix} x \\ y \end{bmatrix}^{new} = \begin{bmatrix} 2.4 \\ 3.2 \end{bmatrix} - 0.1 \begin{bmatrix} 4.8 \\ 6.4 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \end{bmatrix}^{new} = \begin{bmatrix} 1.92 \\ 2.56 \end{bmatrix}$$ 2nd iter $\quad y = \underline{14.24}$

After n^{th} iteration with $x=0, y=0$ →

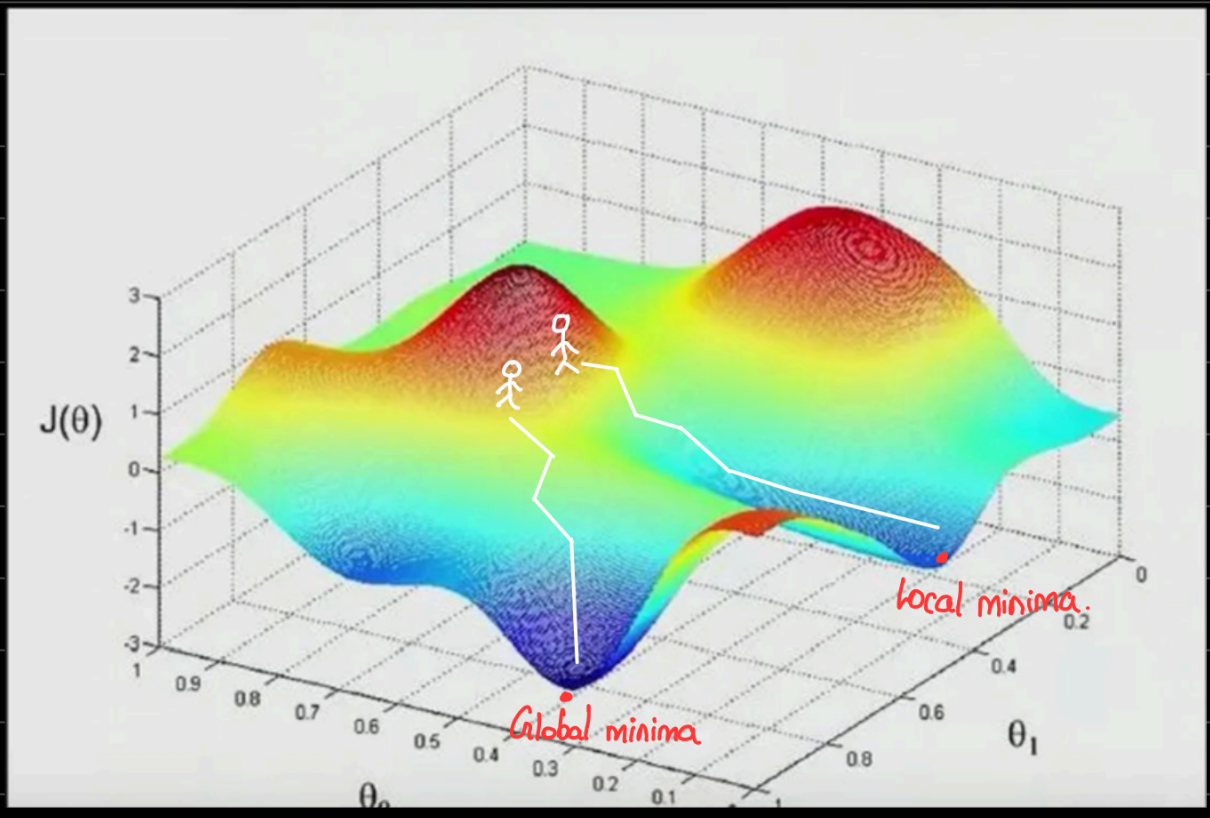
$$\begin{bmatrix} x \\ y \end{bmatrix}^{new} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$y = \underline{4}$ (Reached required value)

We can say our algorithm converged to required point but very slowly.

* Gradient Descent Algorithm can →
1) Diverge
2) Oscillate (Not Diverging nor Converging)
3) Converge Slowly.

* Drawback of gradient descent →



Difference in starting position can lead to different minima than required.

OR

Gradient descent can get stuck in local minima if it starts from an initial point that is far / different from global minima. Once, it reaches a global minimum, the gradient (∇f) becomes zero, & algorithm stops updating, even if a better solution exists elsewhere in the parameter space.

** In linear Regression → Contour Map / level curve / Error Surface only have one optimum point / Global minima. So, we can start from any point and will reach global minima.

So, this drawback is not applicable on linear Regression.

$$\begin{bmatrix} x \\ y \end{bmatrix}^{new} = \begin{bmatrix} x \\ y \end{bmatrix}^{old} - \alpha (\nabla f)$$

α: Learning rate.

∇f: gradient at old point.

Similarly,

$$\begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \end{bmatrix}^{new} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \end{bmatrix}^{old} - \alpha \nabla J(\theta)$$

* $J(w) = \sum \varepsilon_i^2 = (Y - Xw)^T (Y - Xw)$

$$= Y^T Y - Y^T X w - w^T X^T Y + w^T X^T X w$$

$$\nabla J(w) = -2X^T Y + 2X^T X w$$

$$\nabla J(w) = 2X^T (Xw - Y)$$

$$w^{new} = w^{old} - \alpha \nabla J(w)$$

$$w^{new} = w^{old} - \alpha \left[2X^T (X w^{old} - Y) \right]$$

Question 1

If $\phi(x, y, z) = 3x^2y - y^3z^2$, find $\nabla\phi$ (or grad ϕ) at the point $(1, -2, -1)$.

$$\phi(x,y,z) = 3x^2y - y^3z^2$$

$$\frac{\partial \phi}{\partial x} = 6xy \qquad \frac{\partial \phi(1,-2,-1)}{\partial x} = -12$$

$$\frac{\partial \phi}{\partial y} = 3x^2 - 3y^2z^2 \qquad \frac{\partial \phi(1,-2,-1)}{\partial y} = -9$$

$$\frac{\partial \phi}{\partial z} = -2y^3z \qquad \frac{\partial \phi(1,-2,-1)}{\partial z} = -16$$

$$\nabla \phi(1,-2,-1) = (-12, -9, -16)$$

Question 2

Given the Log-Cosh loss function for the i-th data point:

$$\ell(\hat{y}^{(i)}, y^{(i)}) = \log\left(\cosh\left(\hat{y}^{(i)} - y^{(i)}\right)\right),$$

where

$$\hat{y}^{(i)} = w^T x^{(i)} = w_0 + w_1 x_1^{(i)} + \cdots + w_n x_n^{(i)},$$

and $y^{(i)}$ is the true value of the i-the data point.

Compute the gradient of the Log-Cosh loss function with respect to w_j.

$$\nabla l_w = \begin{bmatrix} \partial l/\partial w_0 \\ \partial l/\partial w_1 \\ \partial l/\partial w_2 \\ \vdots \\ \partial l/\partial w_n \end{bmatrix}$$

$$\frac{\partial l}{\partial w_0} = \frac{1}{\cosh(\hat{y}^i - y^i)} \times \sinh(\hat{y}^i - y^i)\,(1) = \tanh(\hat{y}^i - y^i)$$

$$\frac{\partial l}{\partial w_1} = \frac{1}{\cosh(\hat{y}^i - y^i)} \times \sinh(\hat{y}^i - y^i)(x_1^i)$$

$$= \tanh(\hat{y}^i - y^i)(x_1^i)$$

$$\nabla l_w = \begin{bmatrix} \tanh(\hat{y}^i - y^i) \\ \tanh(\hat{y}^i - y^i)\, x_1^i \\ \tanh(\hat{y}^i - y^i)\, x_2^i \\ \vdots \\ \tanh(\hat{y}^i - y^i)\, x_n^i \end{bmatrix}$$

Problem:
Consider the following loss function for linear regression:

$$L(w) = \sum (y_i - (w_0 + w_1 x_i))^2$$

Where y is the vector of actual data points, and X is the design matrix with each row containing the values of the input data points for the prediction. w_0 and w_1 are the parameters.
Task: Calculate the gradient of the loss function with respect to the parameter vector $\mathbf{w}$.
Options:

1. (a) $2Y^T(X^TXw - Y)$
2. (b) $2X^T(Xw - Y)$
3. (c) $Y^T(Xw - Y)$
4. (d) $X^T(Xw - Y)$

$$L(w) = \sum (y_i - w_0 - w_1 x_i)^2$$

$$\frac{\partial L(w)}{\partial w_0} = 2 \sum (y_i - w_0 - w_1 x_i)(-1) = 0$$

$$= \sum_1^N \varepsilon_i = 0$$

$$\frac{\partial L(w)}{\partial w_1} = 2 \sum \varepsilon_i * (-x_i) = 0$$

$$= \sum \varepsilon_i * (-x_i) = 0$$

Consider the following loss function:

$$L(w) = \sum_{i=1}^{n} \log(1 + w^T x_i),$$

where w is the weight vector and x_i represents the feature vector for the i-th data point.
What is the gradient $\nabla L(w)$?

$$L(w) = \sum_1^N \log(1 + w_1 x_1 + w_2 x_2 + \cdots w_n x_n)$$

$$\frac{\partial L(w)}{\partial w_1} = \sum_{1}^{N} \frac{-1}{1 + w_1 x_1 + w_2 x_2 + \cdots w_n x_n} * (x_1)$$

$$\vdots$$

$$\frac{\partial L(w)}{\partial w_n} = \sum_{1}^{N} \frac{1}{1 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n} * (x_n)$$

Question 1

Suppose we run gradient descent with a fixed learning rate of $\alpha = 0.1$ to minimize the 2D function:

$$f(x, y) = 5 + x^2 + y^2 + 5xy$$

Given that the starting point is $x^{(0)} = 1$ and $y^{(0)} = 2$, what will be the next guess $x^{(1)}, y^{(1)}$ after one iteration of gradient descent?

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} - \alpha \begin{bmatrix} \partial f/\partial x \\ \partial f/\partial y \end{bmatrix}$$

$$\frac{\partial f}{\partial x} = 2x + 5y = 12$$

$$\frac{\partial f}{\partial y} = 2y + 5x = 9$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} - 0.1 \begin{bmatrix} 12 \\ 9 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} - \begin{bmatrix} 1.2 \\ 0.9 \end{bmatrix} = \underline{\underline{\begin{bmatrix} -0.2 \\ 1.1 \end{bmatrix}}}$$

Question 2

Consider the following loss function for a set of input vectors $x_i \in \mathbb{R}^d$ and a weight vector $w \in \mathbb{R}^d$:

$$L(w) = \sum_{i=1}^{n} \log(1 + w^T x_i)$$

(a) Derive the gradient $\nabla L(w)$.
(b) Write down the gradient descent update rule for minimizing $L(w)$, using a learning rate η.

$$\begin{bmatrix} w_1^{new} \\ w_2^{new} \\ \vdots \\ w_n^{new} \end{bmatrix} = \begin{bmatrix} w_1^{old} \\ w_2^{old} \\ \vdots \\ w_n^{old} \end{bmatrix} - \eta \begin{bmatrix} \frac{x_1}{\sum w^T x_i} \\ \frac{x_2}{\sum w^T x_i} \\ \\ \frac{x_n}{\sum w^T x_i} \end{bmatrix}$$

Let

$$F(x) = w_0 + \sum_{j=1}^{d} w_j x_j \quad \text{and} \quad L(y^i, F(x^i)) = \frac{1}{1 + \exp(y^i F(x^i))}$$

be the cost function to be minimized. Here, d represents the number of features, x_j represents the j-th feature of the input x, y is the target variable, and i represents the index of the training sample. Suppose you use gradient descent to obtain the optimal parameters w_0 and w_j. Give the update rules for these parameters.

$$F(x) = w_0 + \sum_{1}^{d} w_j x_j$$

Loss function $L(y^i, F(x^i)) = \dfrac{1}{1 + e^{y_i F(x^i)}}$

$$\frac{\partial L}{\partial w_0} = \boxed{\frac{\partial L}{\partial F(x)}} \cdot \boxed{\frac{\partial F(x)}{\partial w_0}}$$

$\boxed{\frac{\partial F(x)}{\partial w_0}} \rightarrow 1$

$\boxed{\frac{\partial L}{\partial F(x)}} \rightarrow -y\, L(y, F(x))\,(1 - L(y, F(x)))$

$$\frac{\partial L}{\partial w_0} = -y\,L(y, F(x)) \cdot (1 - L(y, F(x))$$

$$\frac{\partial L}{\partial w_j} = \boxed{\frac{\partial L}{\partial F(x)}} \cdot \boxed{\frac{\partial F(x)}{\partial w_j}} \rightarrow x_j$$

$$-y\,L(y, F(x))\,(1 - L(y, F(x))$$

$$\frac{\partial L}{\partial w_j} = -y\,L(y, F(x)) \cdot (1 - L(y, F(x))(x_j)$$

* Update Rules :

$$w_0^{new} = w_0^{old} + \eta \;\; y\,L(y, F(x))(1 - L(y, F(x)))$$

$$w_j^{new} = w_j^{old} + \eta \;\; y\,x_j\,L(y, F(x))(1 - L(y, F(x)))$$