

Deep Learning & Neural Networks

Dr. Shahira Ahmed

DS710

Lecture 5





Deep Learning & Neural Networks DS710

**Lecture 5
Optimization**

Dr. Shahira Ahmed



LECTURE OUTLINE

- ◉ Optimization
- ◉ Local vs Global Optima
- ◉ Finding Critical values
- ◉ Derivative
- ◉ Gradient Descent
- ◉ Types of Gradient Descent

- “Optimization” comes from the same root as “optimal”, which means best.
- When you optimize something, you are “making it best”.
- an optimization problem is the problem of finding the **best** solution from all **feasible** solutions.

- Mathematical Optimization is a branch of applied mathematics which is useful in many different fields. Here are a few examples:
 - Manufacturing
 - Production
 - Inventory control
 - Transportation
 - Scheduling
 - Networks
 - Finance
 - Engineering
 - Mechanics
 - Economics
 - Control engineering
 - Marketing
 - Policy Modeling

- Many important applied problems involve finding the best way to accomplish some task.
 - The **minimum** time to make a certain journey.
 - The **minimum** cost for doing a task.
 - The **maximum** power that can be generated by a device.
- Many of these problems can be solved by finding the appropriate function and then using techniques of calculus to find the maximum or the minimum value required.

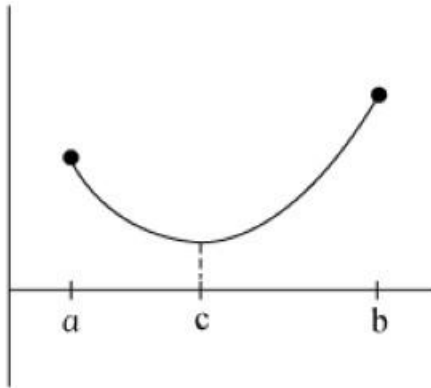
- Your basic optimization problem consists of
 - The **objective function(Fitness)**, $f(x)$, which is the output you're trying to maximize or minimize.
 - **Variables**, x_1 x_2 x_3 and so on, which are the inputs – things you can control.
 - **constraints**

<https://web.stanford.edu/group/sisl/k12/optimization/MO-unit1-pdfs/1.1optimization.pdf>

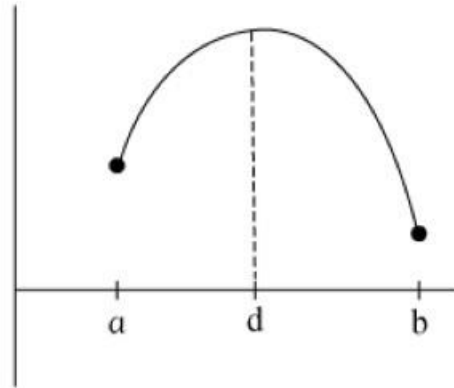
Absolute Extrema

- Absolute extrema represents the highest and lowest points on a curve.
- the highest point (absolute **maximum**) and the lowest point (absolute **minimum**)

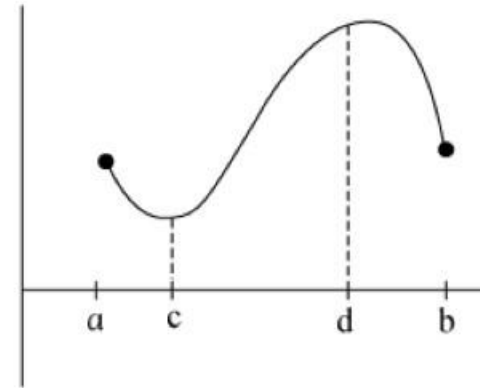
Absolute Minimum



Absolute Maximum

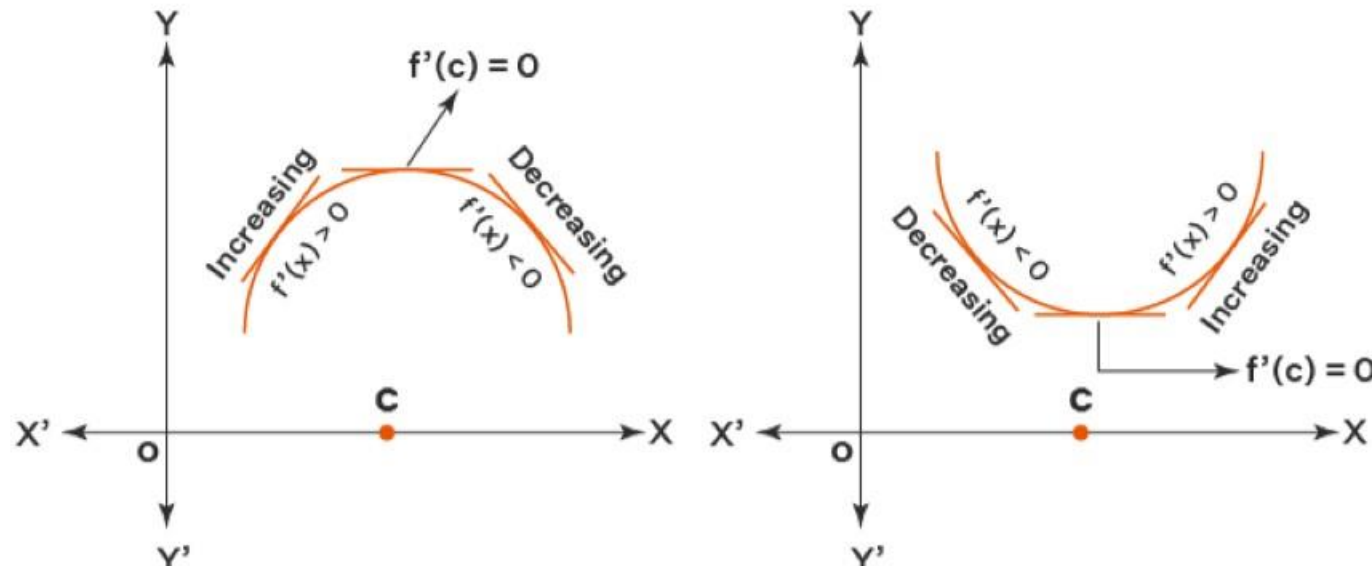


Absolute Minimum and Maximum



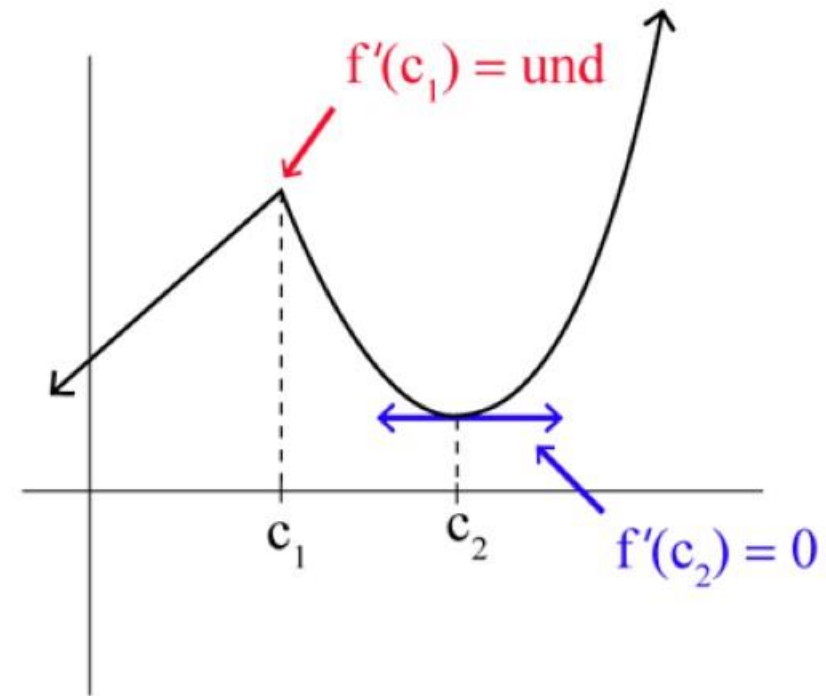
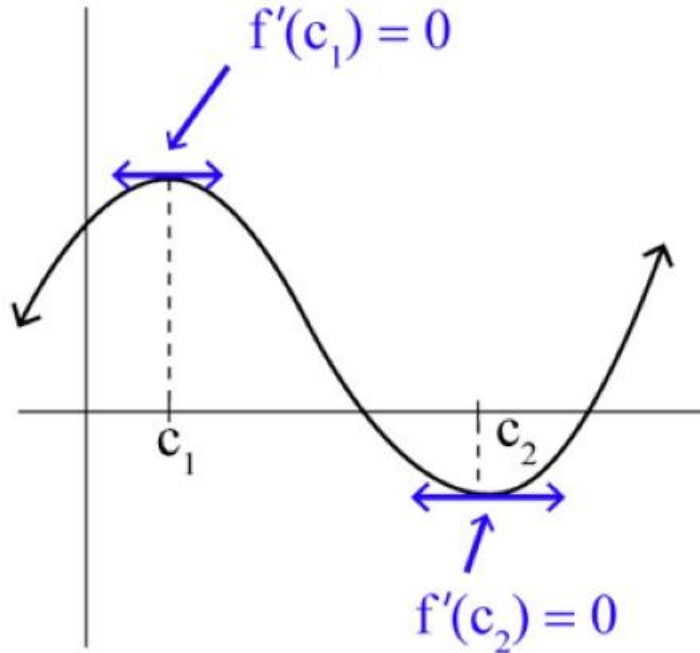
First Derivative Test

- The **first derivative** test helps in finding the turning points, where the function output has a maximum value or a minimum value. For the first derivative test.



Finding critical numbers

- A critical number of a function f is a number c in the domain such that $f'(c) = 0$



- Find the following function's global maximum and global minimum values on the indicated interval.

$$f(x) = 3x^2 - 18x + 5, [0, 7]$$

- First, we find all possible critical numbers by setting the derivative equal to zero.

$$f'(x) = 6x - 18$$

$$6x - 18 = 0$$

$$x = 3$$

Finding Critical Numbers

$$f(x) = 3x^2 - 18x + 5, [0, 7]$$

Now we substitute the critical number and both endpoints into the function to determine absolute extrema.

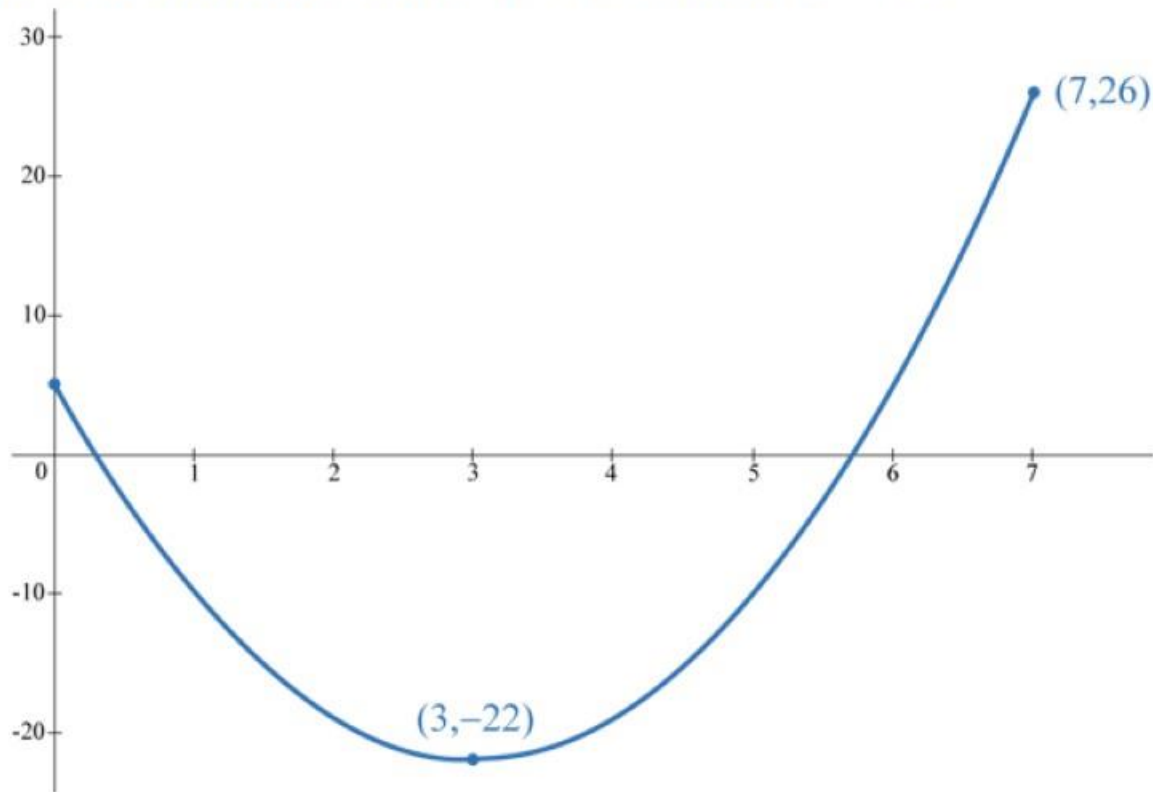
$$f(3) = 3(3)^2 - 18(3) + 5 = -22$$

$$f(0) = 3(0)^2 - 18(0) + 5 = 5$$

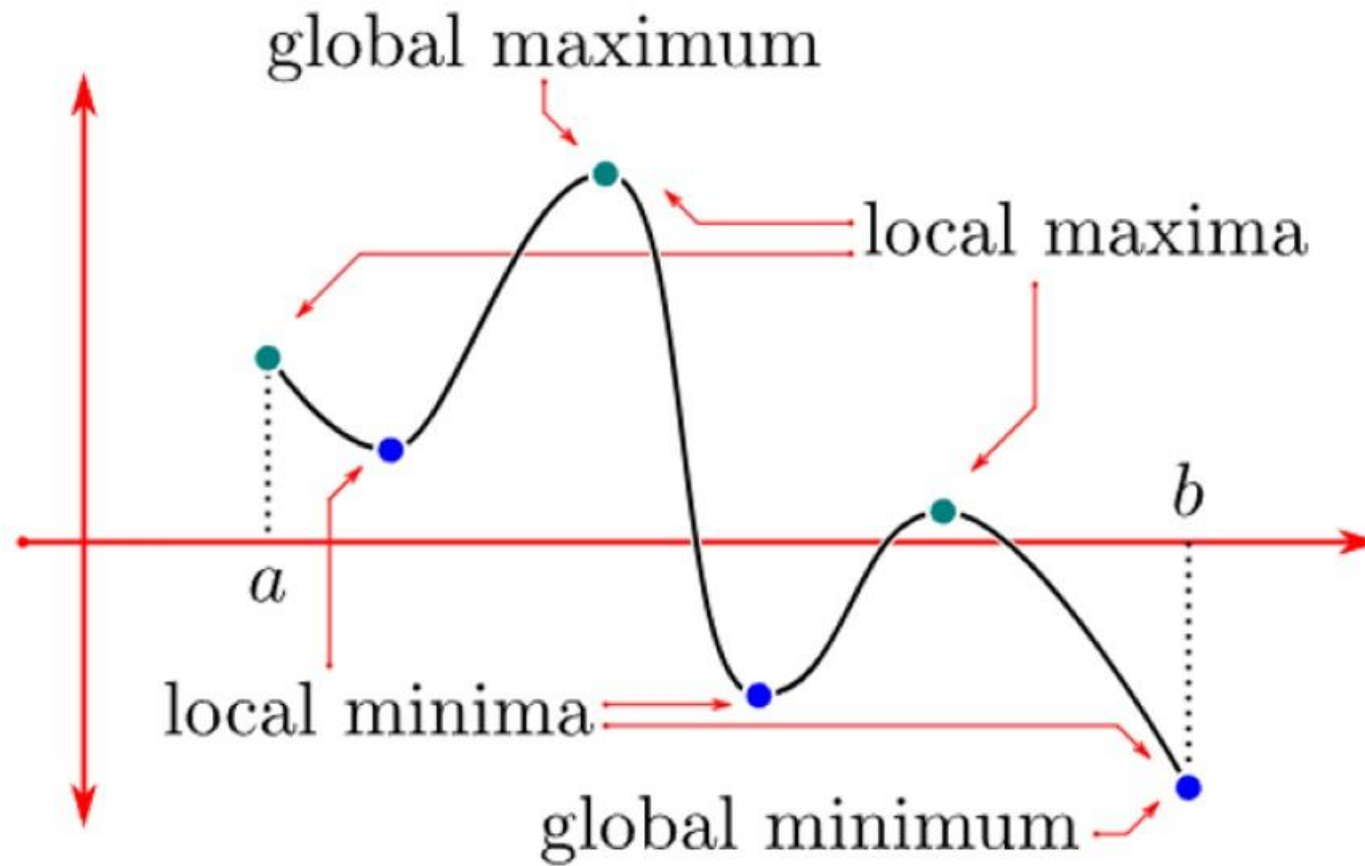
$$f(7) = 3(7)^2 - 18(7) + 5 = 26$$

Finding Critical Numbers

This means the absolute minimum value is -22 and is when $x = 3$
The absolute maximum value is 26 when $x = 7$.



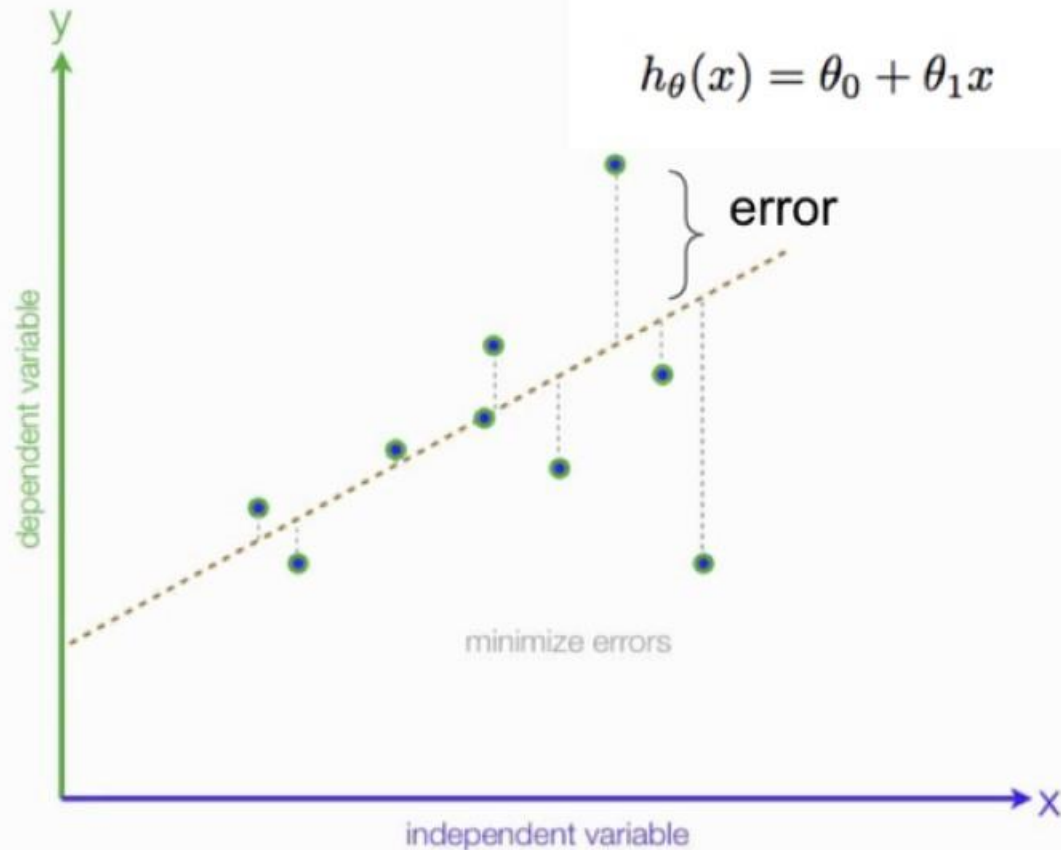
Local and Global Maxima and Minima



What is the Cost Function?

- The goal is to find a set of **weights and biases** that **minimizes** the **cost**. One common function that is often used is the **mean squared error**, which measures the difference between the actual value of y and the estimated value of y (the prediction).

What is the Cost Function?



Hypothesis:

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

Parameters:

$$\theta_0, \theta_1$$

Cost Function:

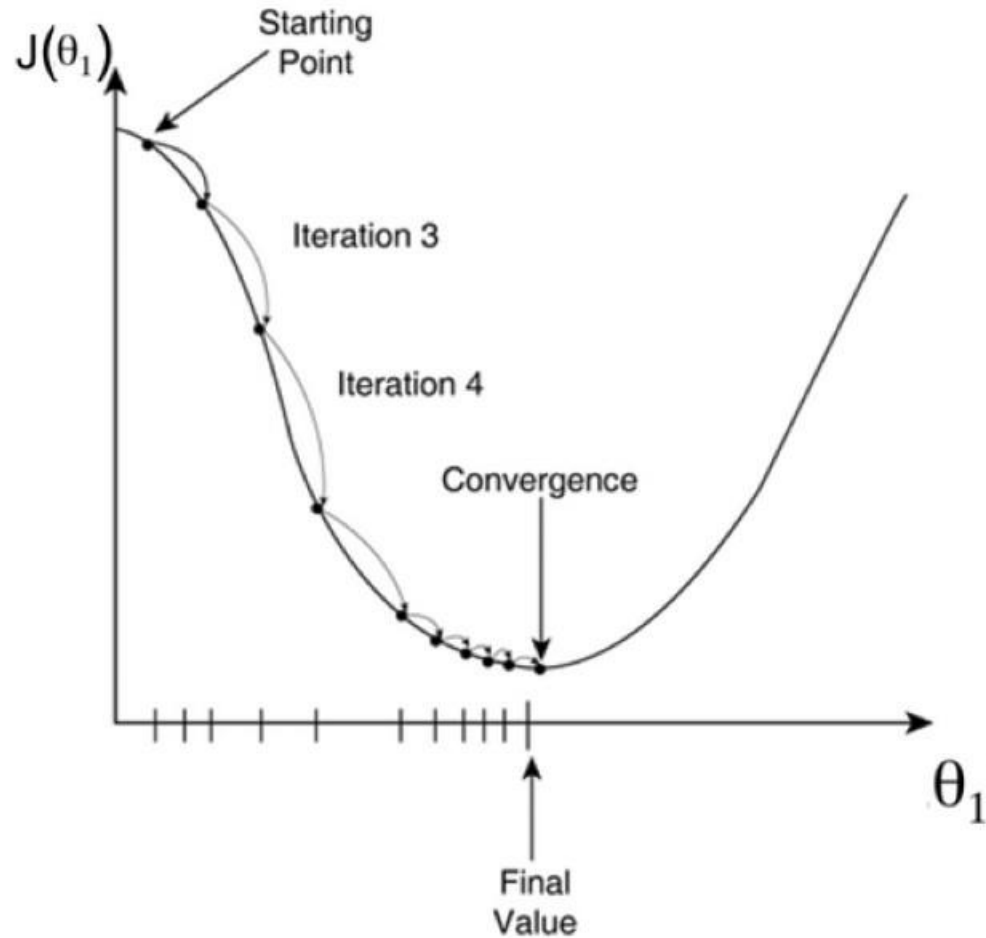
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

Goal:

$$\underset{\theta_0, \theta_1}{\text{minimize}} J(\theta_0, \theta_1)$$

- Gradient descent (GD) is an optimization algorithm, used to find a local minimum/maximum of a given function. This method is commonly used in machine learning (ML) and deep learning (DL) to minimize a cost/loss function (e.g. in a linear regression).
- used also in areas like:
 - control engineering (robotics, chemical, etc.)
 - computer games
 - mechanical engineering

Gradient Descent



Cost Function – “One Half Mean Squared Error”:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

Objective:

$$\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$$

Derivatives:

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})$$

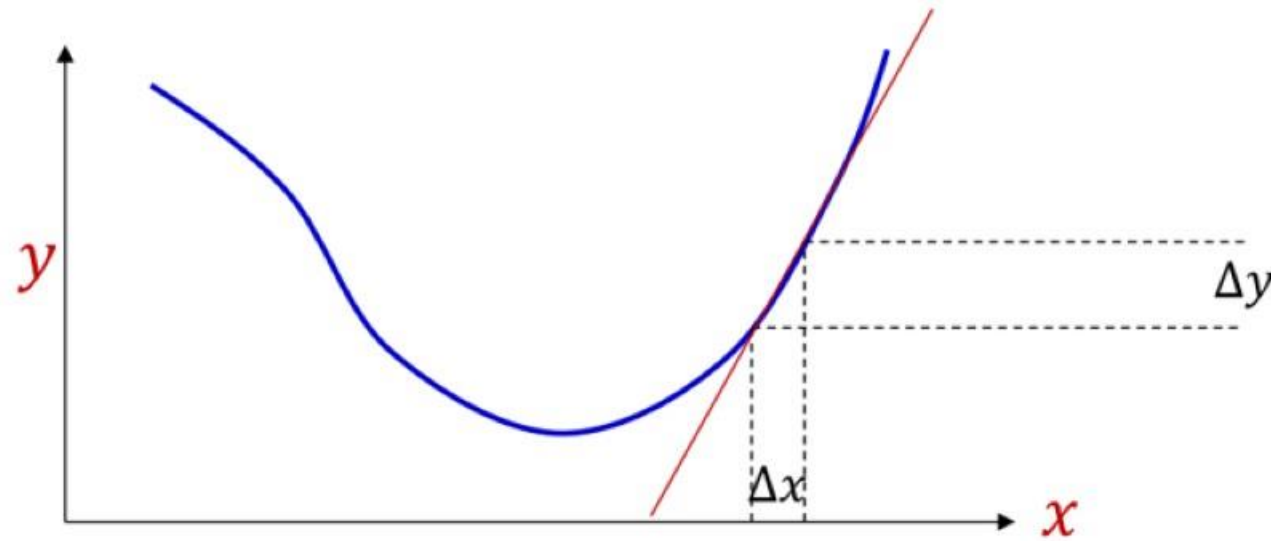
$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$$

- A function has to be:
 - **Differentiable**
 - **Convex**

- **Differentiability:** The function must be differentiable with respect to its parameters. This is because gradient descent relies on computing the gradient, which requires taking derivatives of the function.

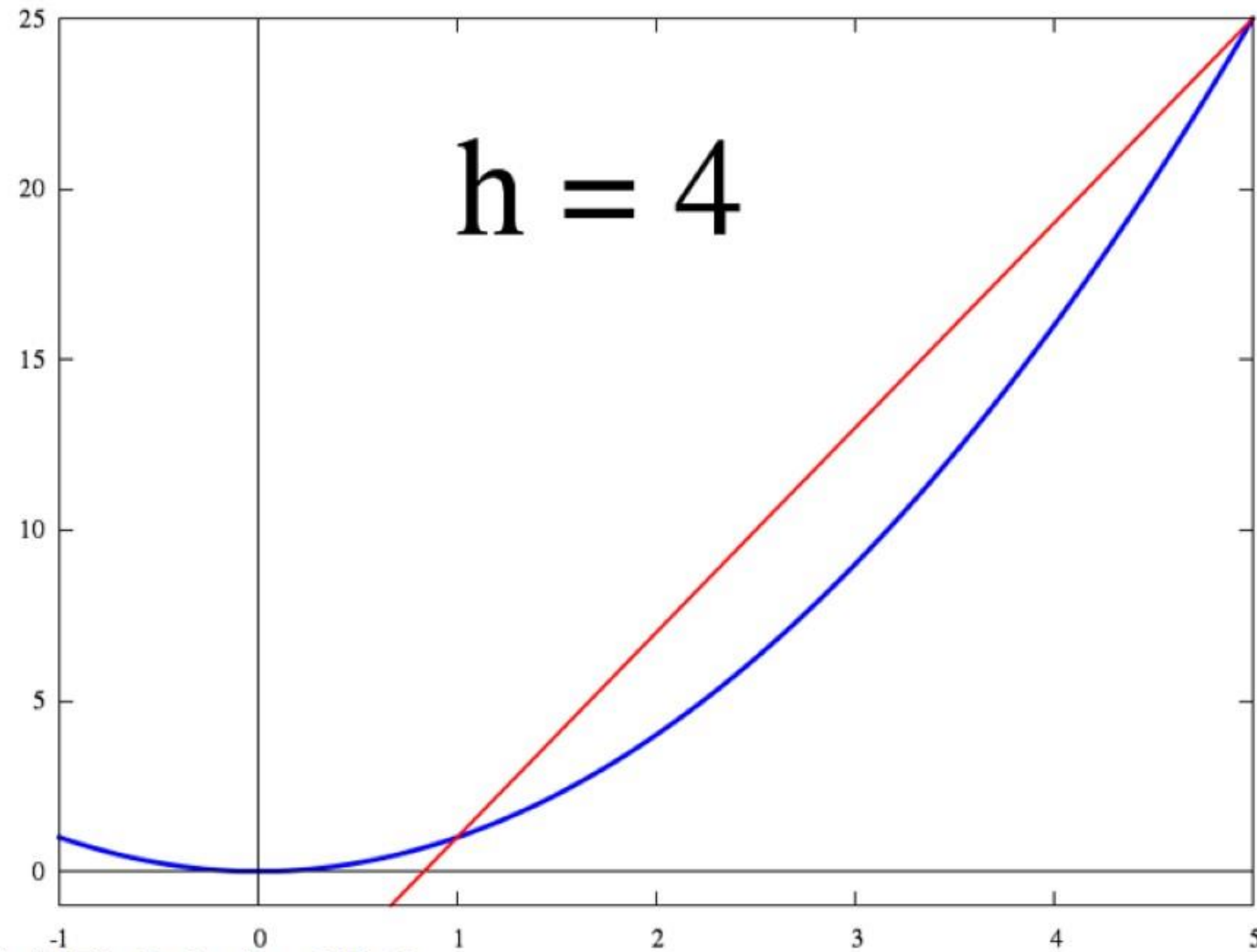
- A maximum or minimum is said to be local if it is the largest or smallest value of the function, respectively, within a given range.
- However, a maximum or minimum is said to be global if it is the largest or smallest value of the function, respectively, on the entire domain of a function.

- The derivative of a function describes the function's instantaneous rate of change at a certain point.
- Another common interpretation is that the derivative gives us the slope of the line tangent to the function's graph at that point. Learn how we define the derivative using limits.

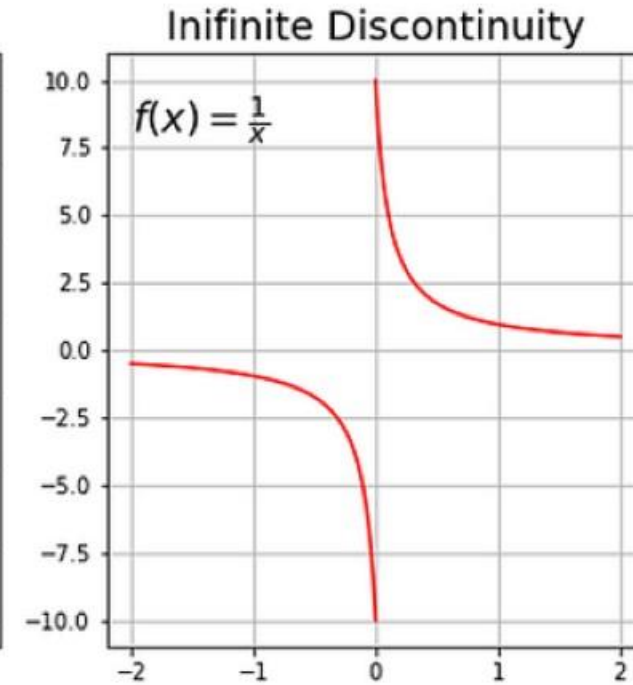
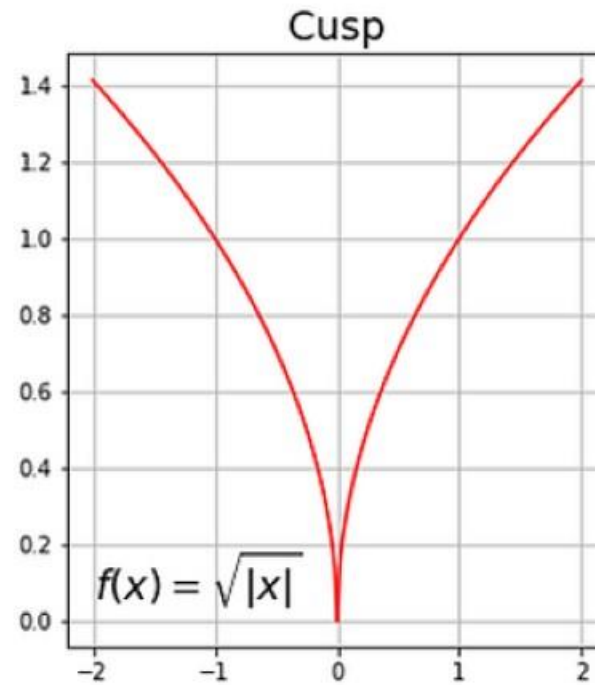
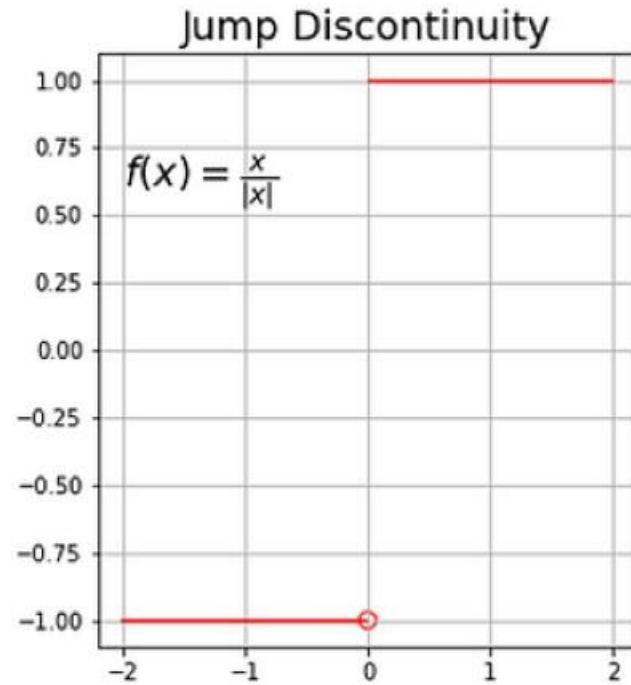


- A derivative of a function at any point tells us how much a minute increment to the argument of the function will increment the value of the function
 - For any $y = f(x)$ expressed as a multiplier α to a tiny increment Δx to obtain the increments Δy to the output
 - $\Delta y = \alpha \Delta x$

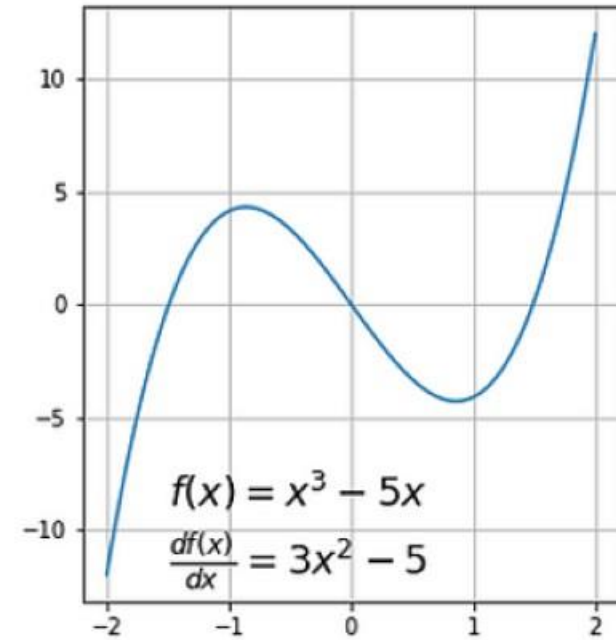
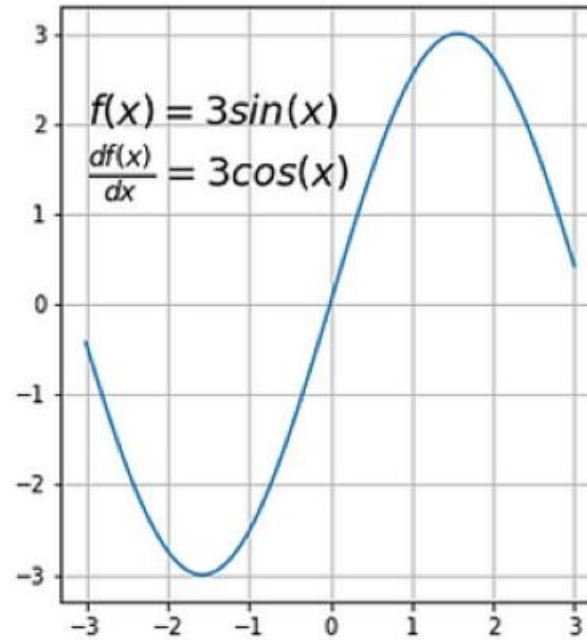
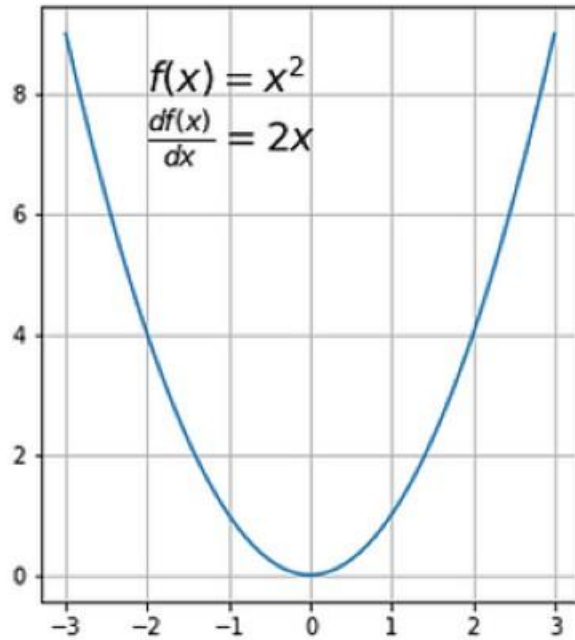
The derivative of a function measures the rate at which the function changes with respect to its independent variable. It represents the slope of the tangent line to the graph of the function at a particular point. The derivative is usually denoted using the prime notation (f') or by using the operator notation (d/dx).



Examples of non differentiable Functions



Examples of non differentiable Functions



Gradient:

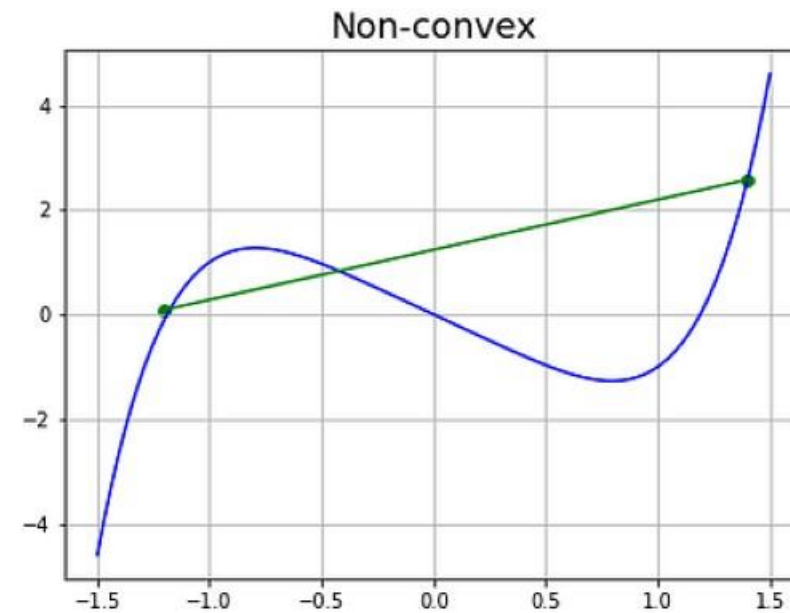
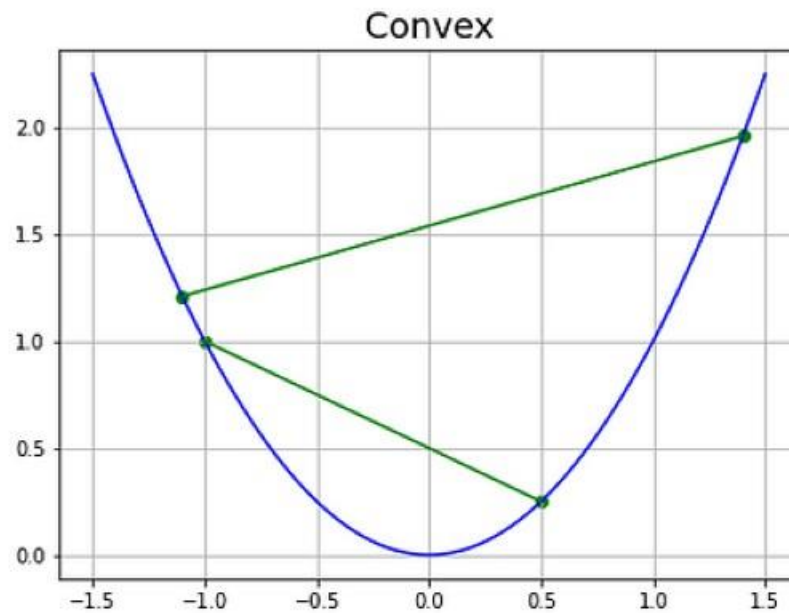
The gradient, also known as the "vector derivative," is a generalization of the derivative for functions of multiple variables..

For a function $f(x_1, x_2, \dots, x_n)$, the gradient is denoted as ∇f or $\text{grad}(f)$. The gradient is a vector that consists of partial derivatives of the function with respect to each variable. Each component of the gradient vector indicates the rate of change of the function concerning a specific variable.

- In summary, the derivative measures the rate of change of a function with respect to a **single variable**, while the gradient measures the direction and magnitude of the steepest ascent of a function with respect to **multiple variables**. The derivative is a **scalar**, while the gradient is a **vector**.

Convex Function

- For a univariate function, this means that the line segment connecting two function's points lays on or above its curve (it does not cross it). If it does it means that it has a local minimum which is not a global one.

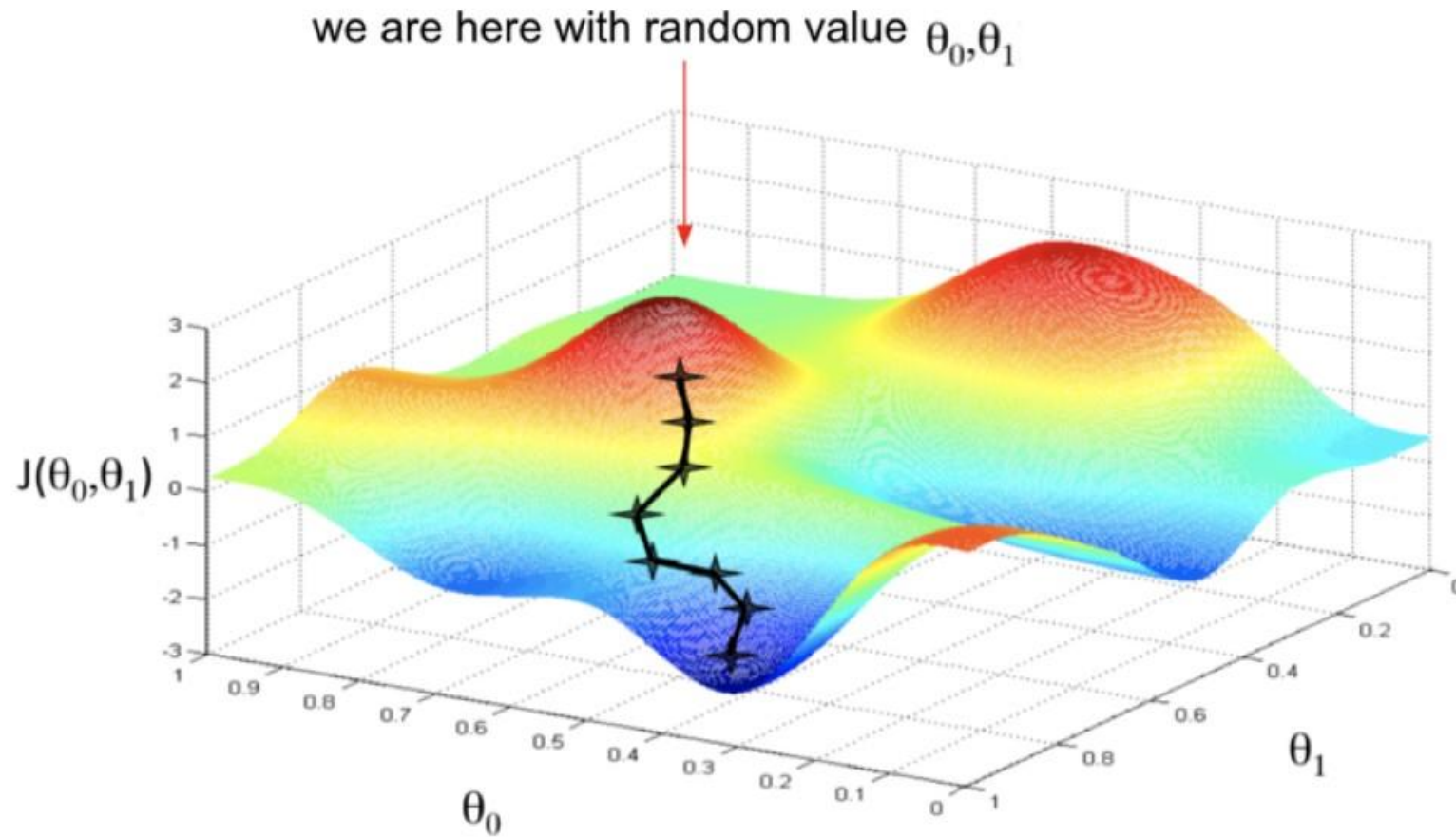


1. choose a starting point (initialization)
2. calculate gradient at this point
3. make a scaled step in the opposite direction to the gradient (objective: minimize)
4. repeat points 2 and 3 until one of the criteria is met:
5. maximum number of iterations reached
6. step size is smaller than the tolerance (due to scaling or a small gradient).

Gradient Descent

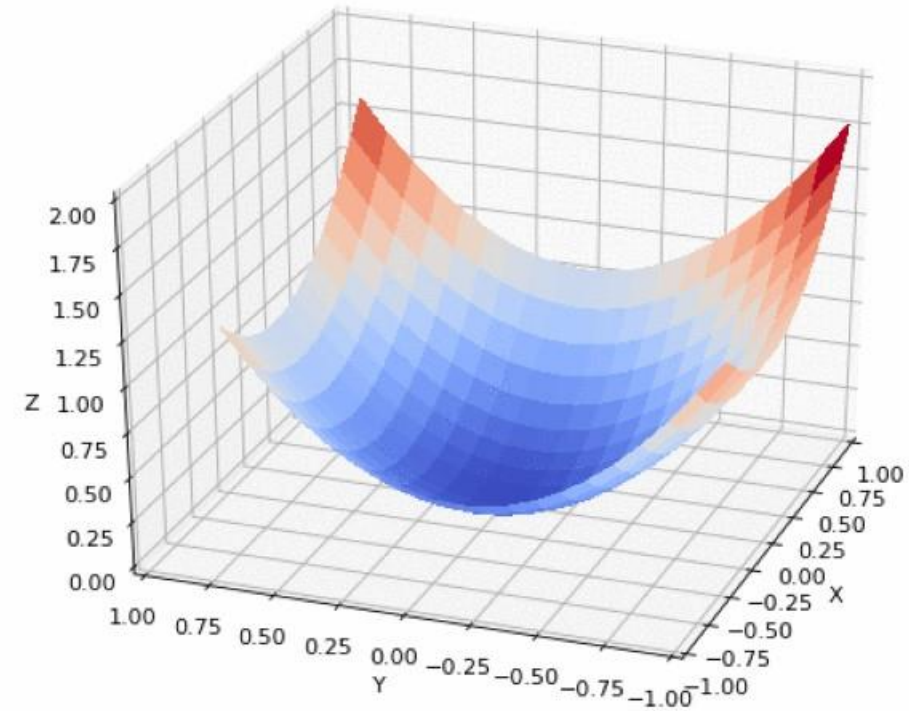
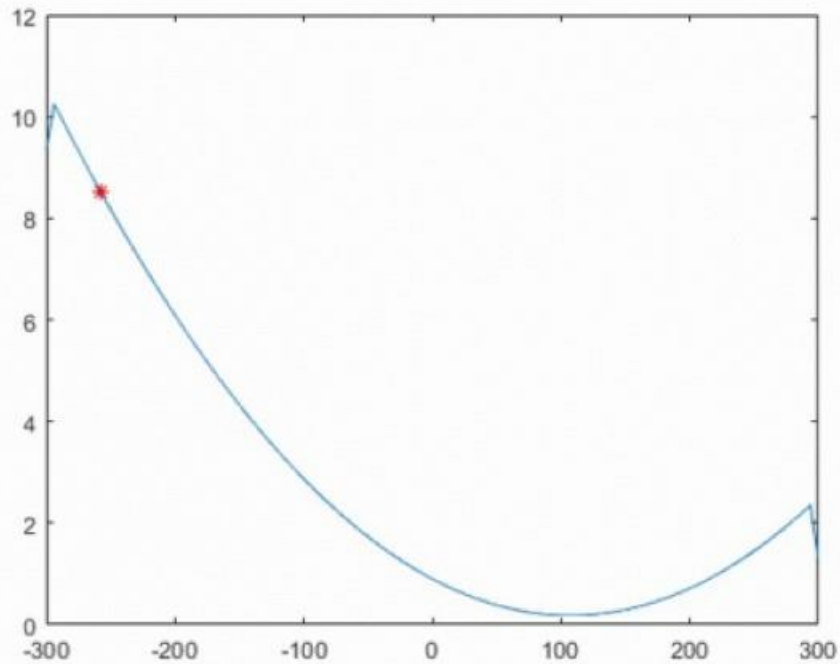


Gradient Descent



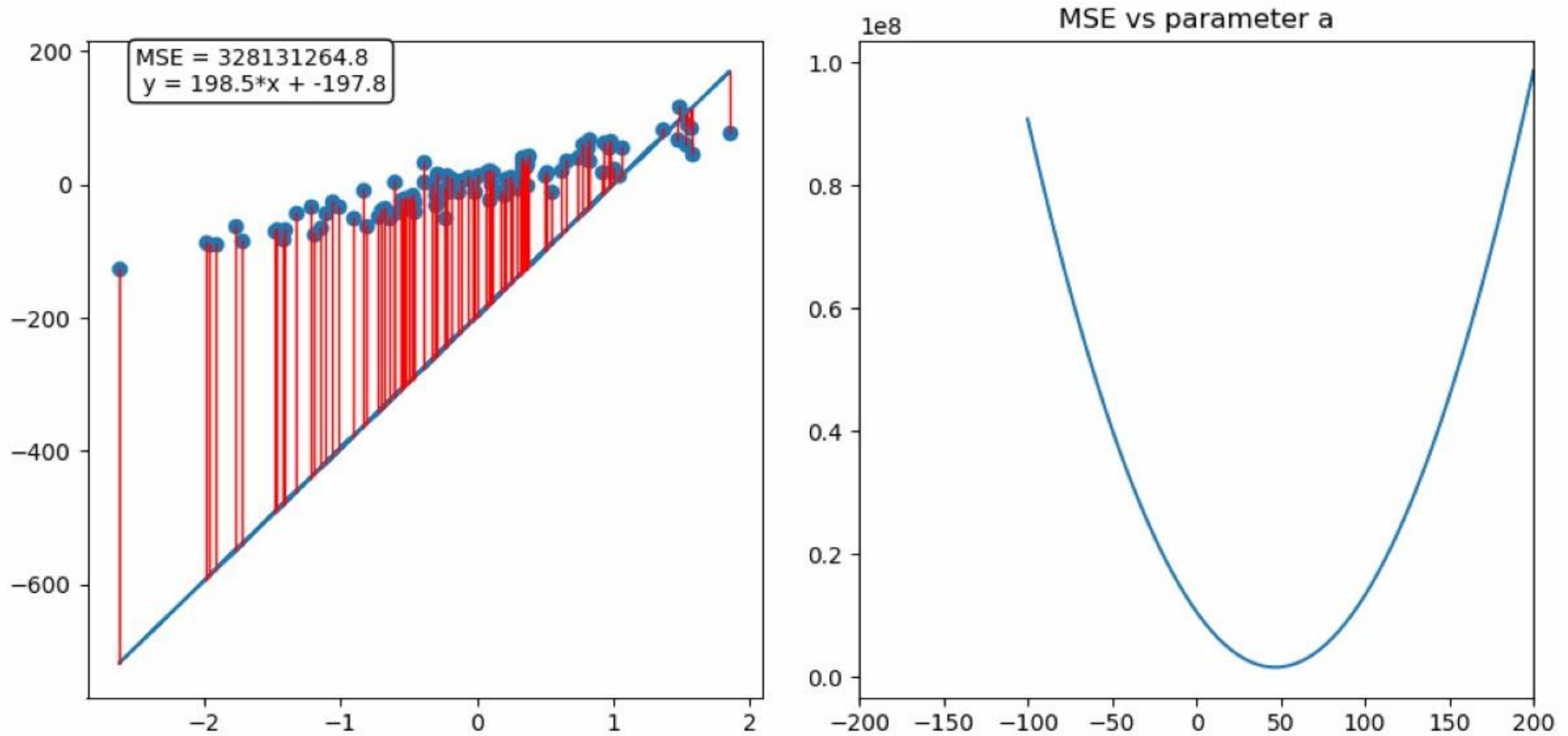
- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum

Gradient Descent

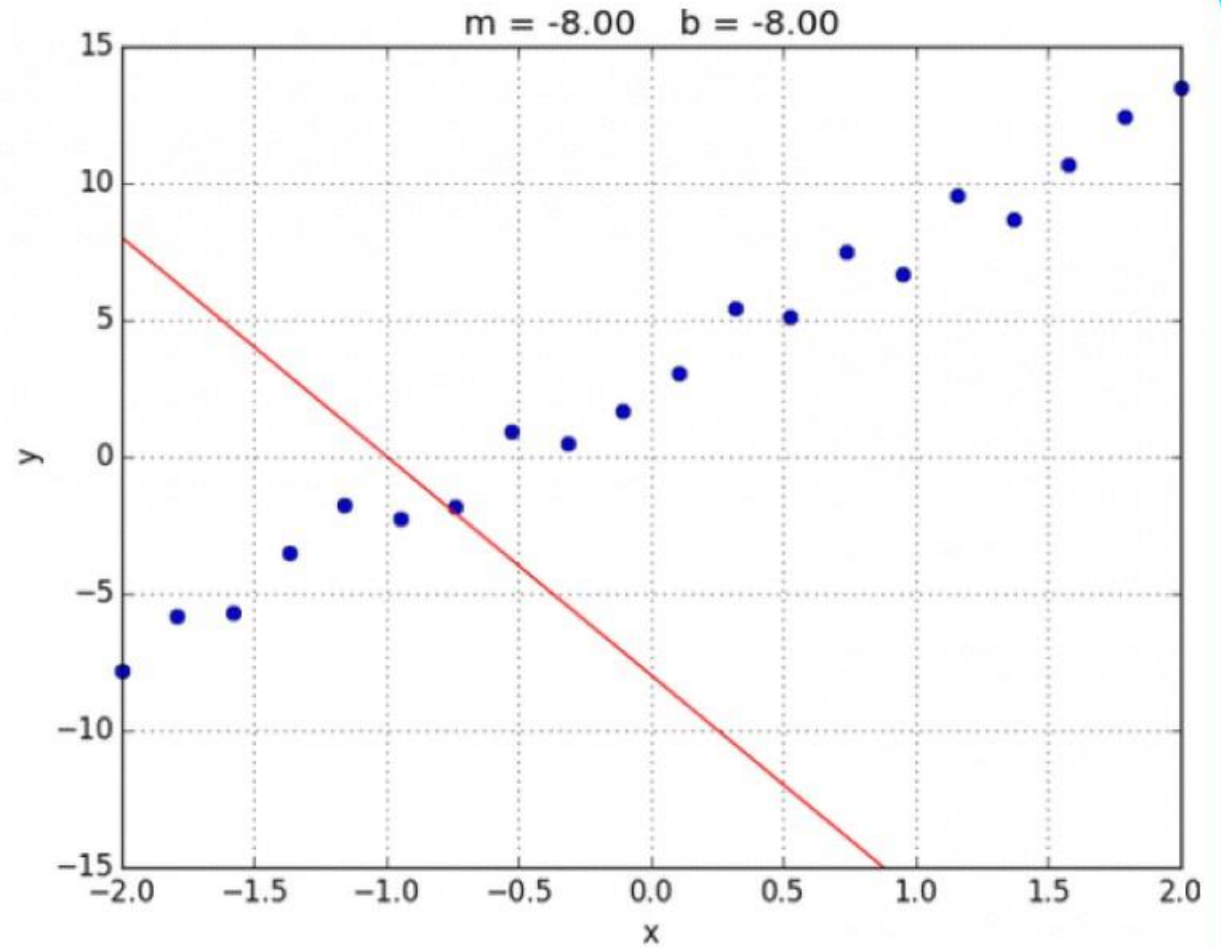
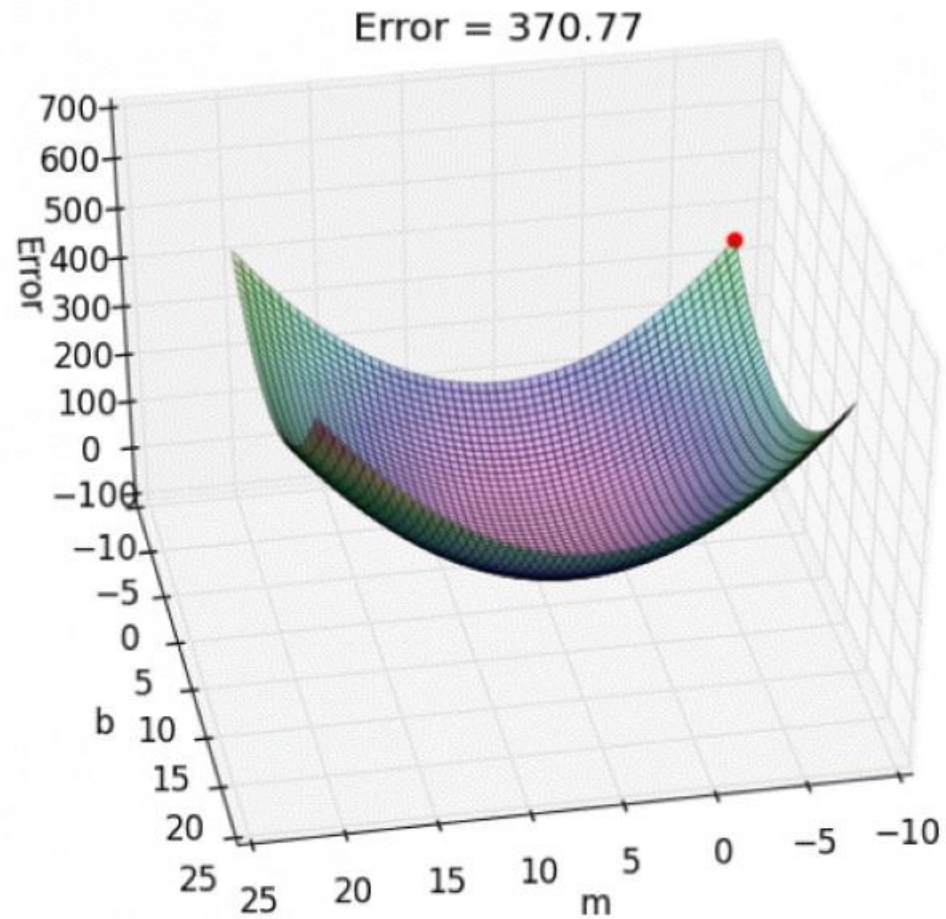


[Gradientdescent Graph GIF - Gradientdescent Graph Lines - Discover & Share GIFs \(tenor.com\)](#)

Gradient Descent

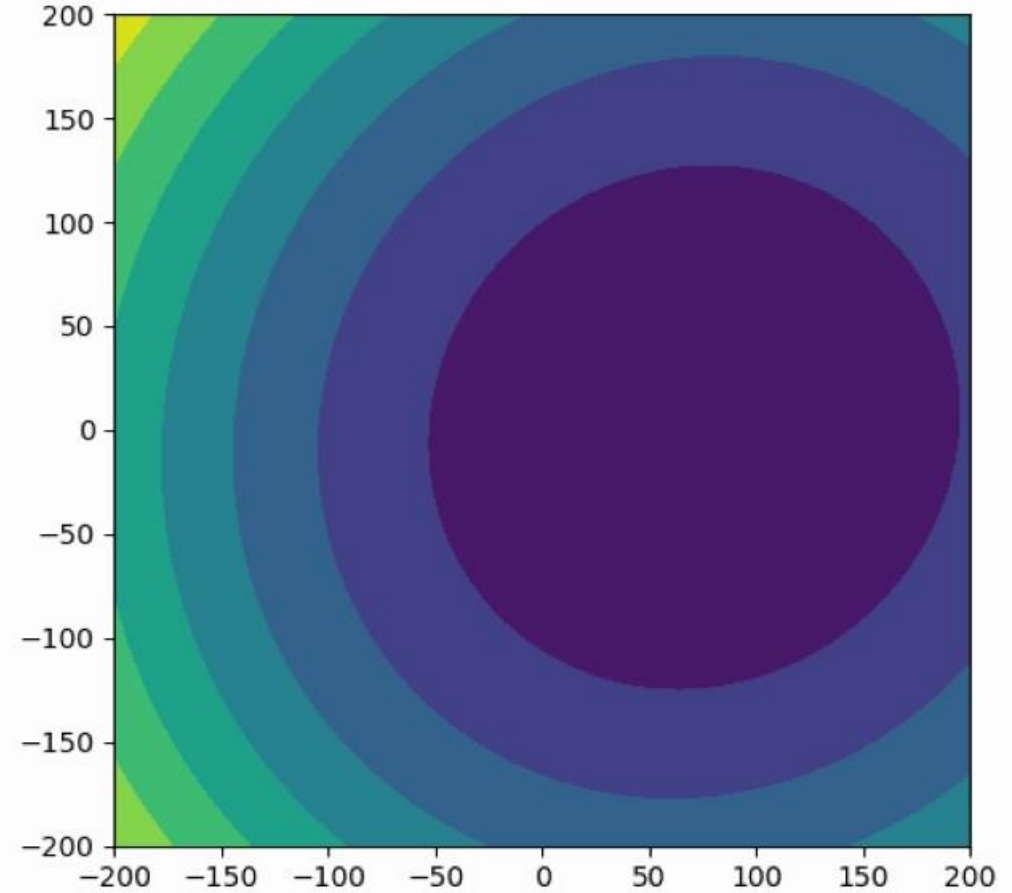
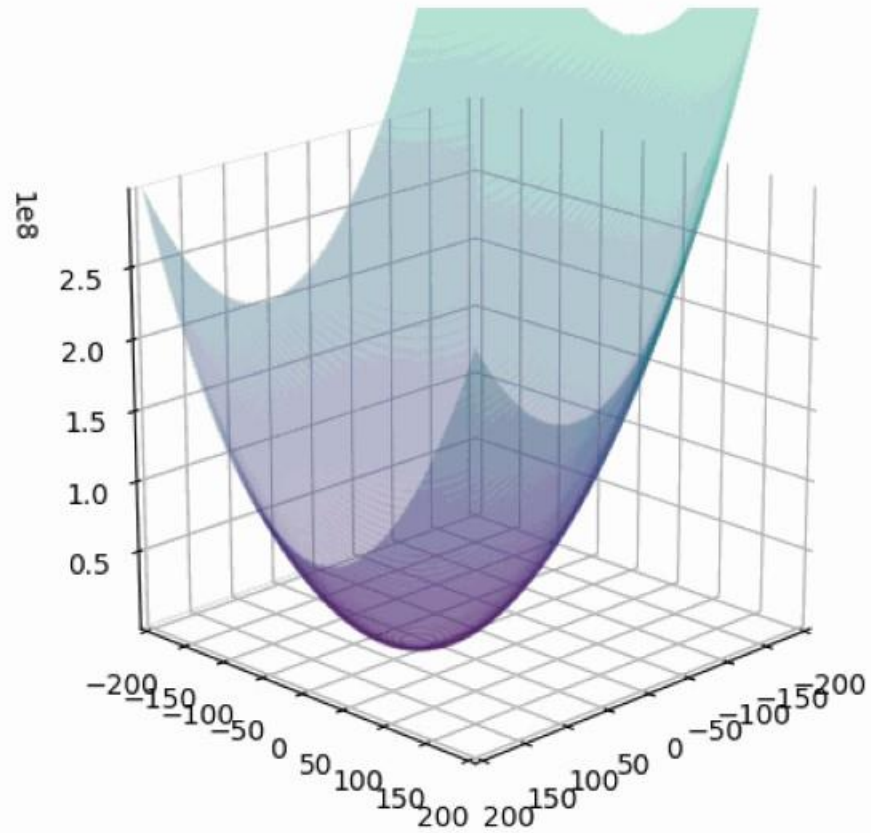


Gradient Descent



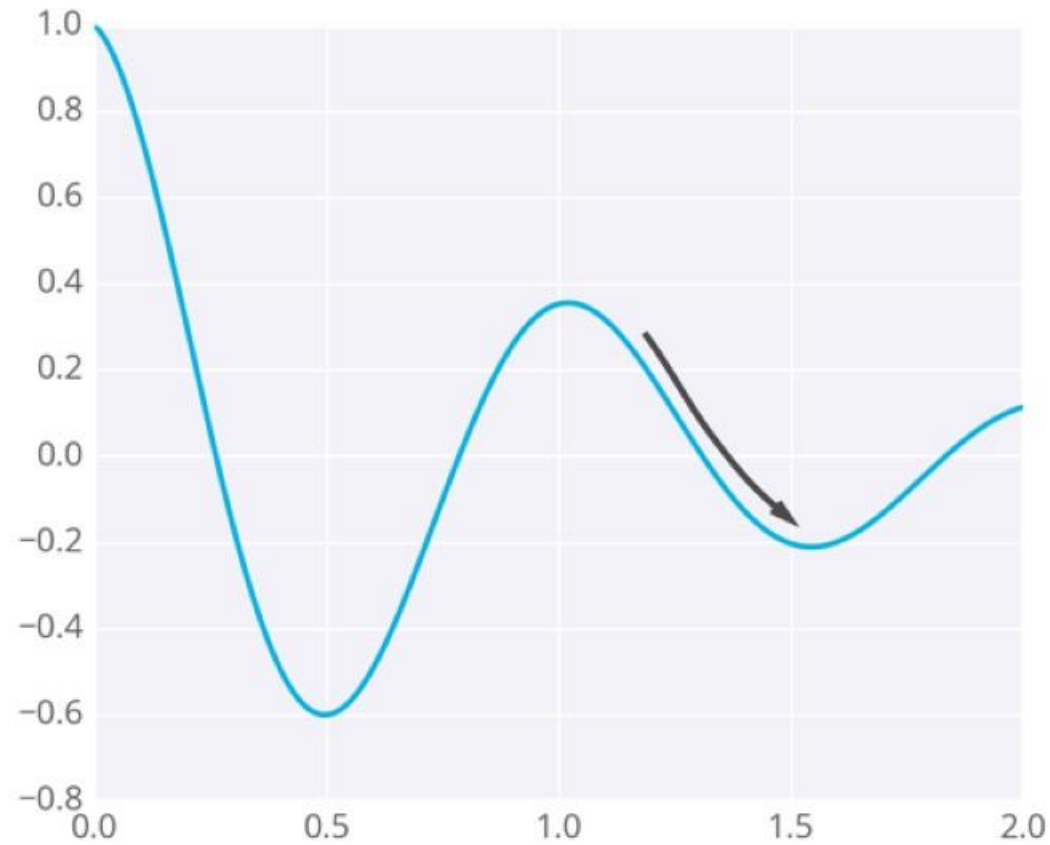
[Gradient Descent in 4 minutes:.. Hey there ! Confused about gradient... | by Shobhit Srivastava | AITS Journal | Medium](#)

Gradient Descent

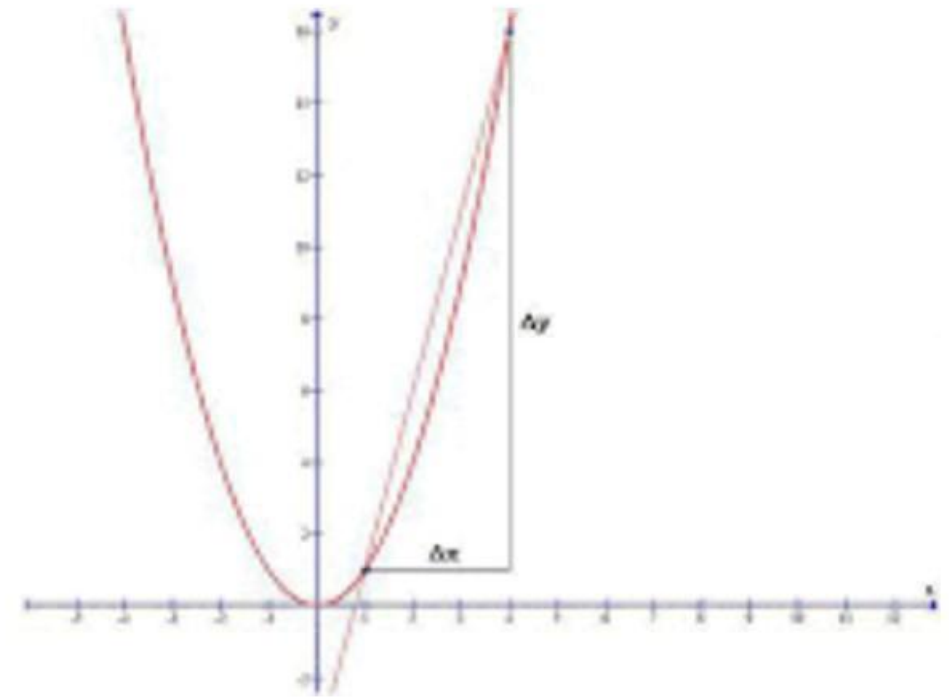


[Stochastic Gradient Descent for machine learning clearly explained | Baptiste Monpezat \(baptiste-monpezat.github.io\)](https://baptiste-monpezat.github.io)

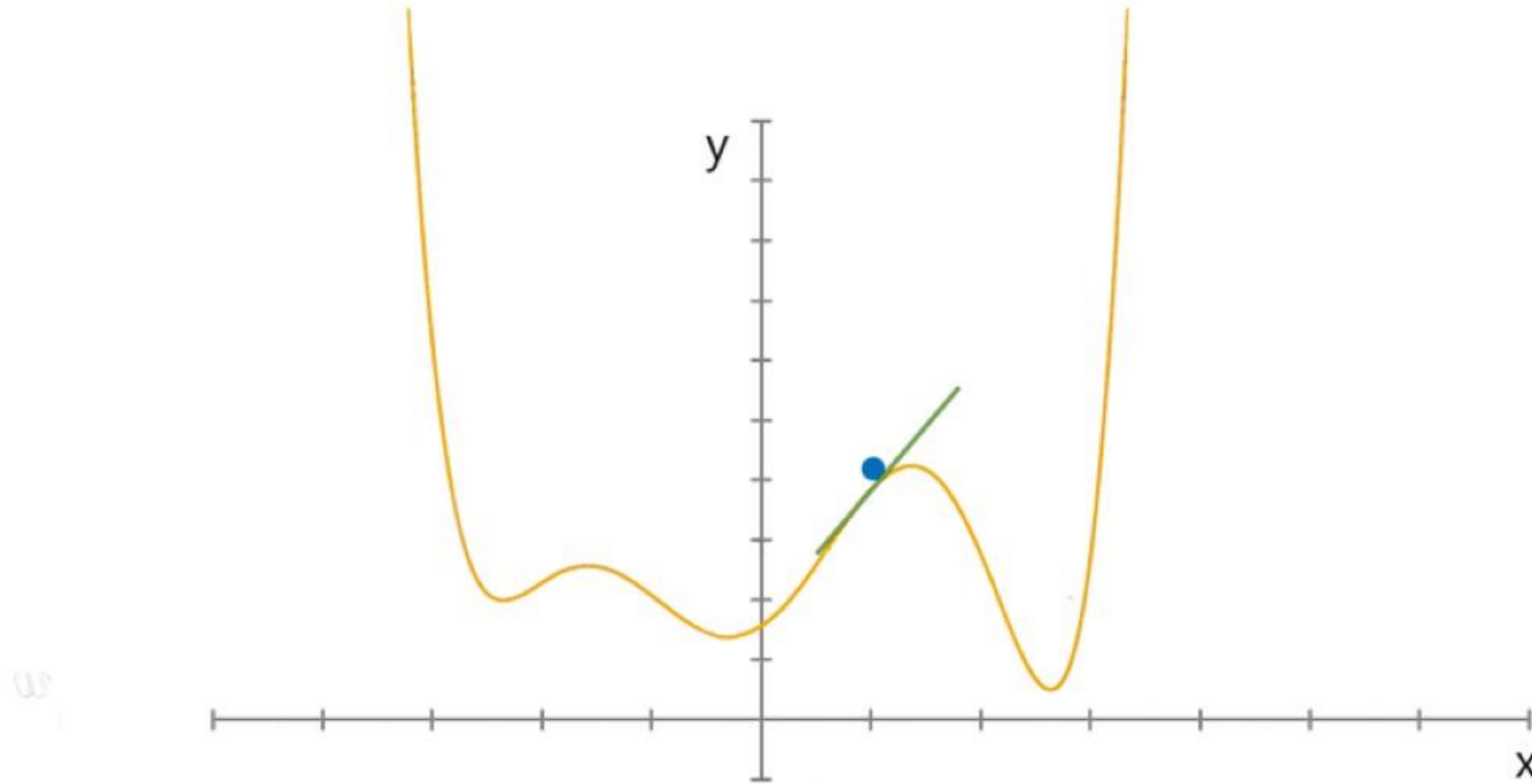
- Gradient descent leading into a local minimum



Derivative



Derivative



Chain Rule



$$f' \circ g \circ h \circ k(x) \times g' \circ h \circ k(x) \times h' \circ k(x) \times k'(x)$$





The Chain Rule

For $F(x) = f(g(x))$

$$F'(x) = f'(g(x)) \cdot g'(x)$$

Derivative of outer function

Derivative of inner function

Chain Rule



Differentiate $f(x) = (4x - 3)^5$

$$\frac{dy}{dx} = \left(\begin{array}{l} \text{Differentiate} \\ \text{outer function} \\ \text{Keep the inside} \\ \text{the same} \end{array} \right) \left(\begin{array}{l} \text{Differentiate} \\ \text{inner function} \end{array} \right)$$

$$f(x) = (4x - 3)^5$$

1. choose a starting point (initialization)
2. calculate gradient at this point
3. make a scaled step in the opposite direction to the gradient (objective: minimize)
4. repeat points 2 and 3 until one of the criteria is met:
5. maximum number of iterations reached
6. step size is smaller than the tolerance (due to scaling or a small gradient).

$$\begin{aligned}\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_{\theta}(x) - y)^2 \\ &= 2 \cdot \frac{1}{2} (h_{\theta}(x) - y) \cdot \frac{\partial}{\partial \theta_j} (h_{\theta}(x) - y) \\ &= (h_{\theta}(x) - y) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^d \theta_i x_i - y \right) \\ &= (h_{\theta}(x) - y) x_j\end{aligned}$$

- For a single training example, this gives the update rule

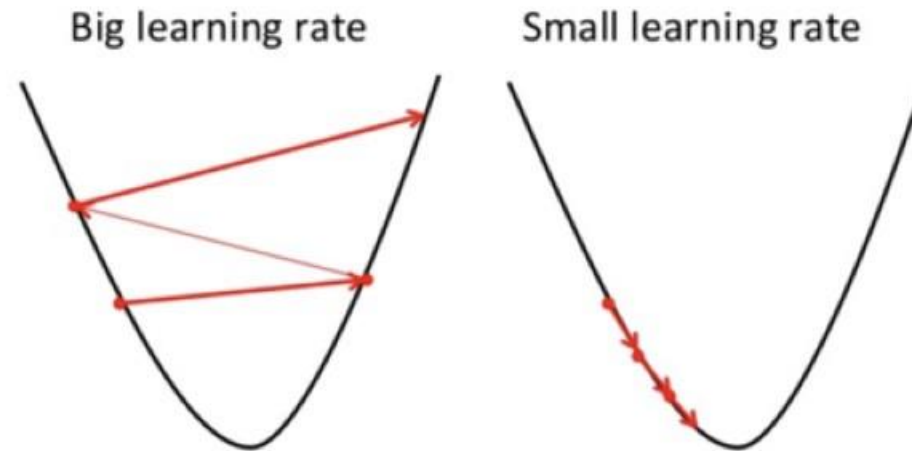
$$\theta_j := \theta_j + \alpha (y^{(i)} - h_{\theta}(x^{(i)})) x_j^{(i)}.$$

Repeat until convergence {

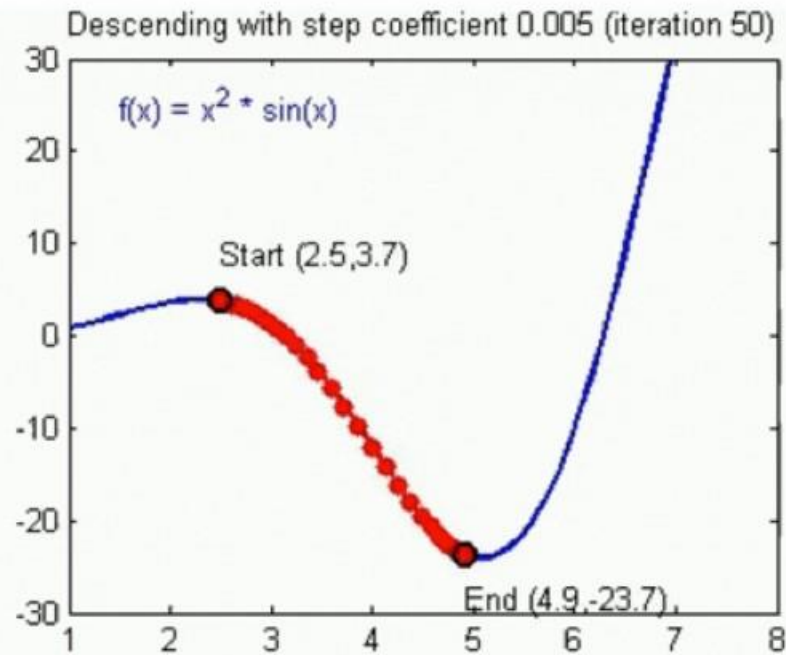
$$\theta_j := \theta_j + \alpha \sum_{i=1}^n (y^{(i)} - h_{\theta}(x^{(i)})) x_j^{(i)}, \text{ (for every } j)$$

}

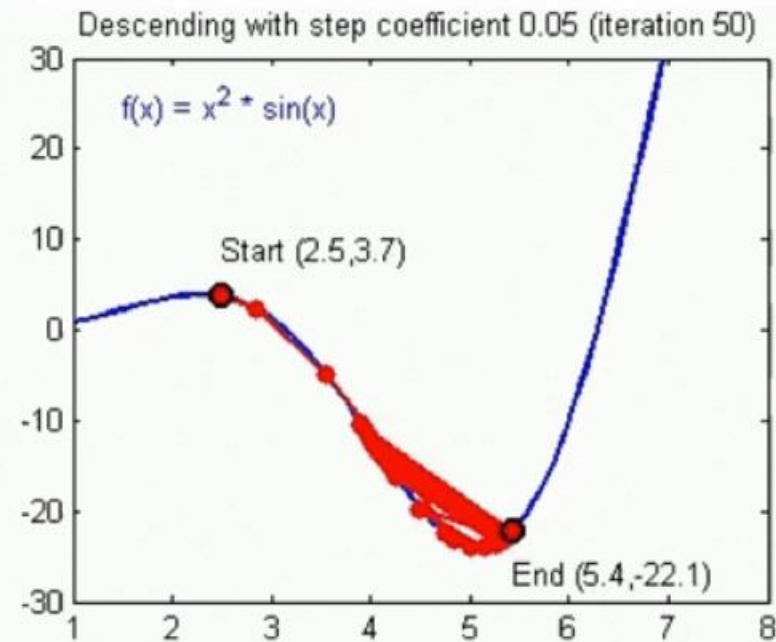
- The learning rate is a hyperparameter in the gradient descent algorithm that controls the step size at each iteration. It determines how much the parameters are updated based on the computed gradients. The learning rate can significantly impact the convergence and performance of the gradient descent algorithm.



Convergence



Divergence



- Choosing an appropriate learning rate is crucial.
- If the learning rate is too high, the algorithm may fail to converge or become unstable.
- If it is too low, the convergence may be slow.
- It is often necessary to experiment with different learning rates or use techniques like learning rate schedules or adaptive learning rate methods (e.g., Adam, RMSprop) to find an optimal or adaptive learning rate that balances convergence speed and stability.

Types of Gradient Descent

1. **Batch** Gradient Descent
2. **Stochastic** Gradient Descent
3. **Minibatch** Gradient Descent

- **Batch** gradient descent, also called vanilla gradient descent, calculates the error for each example within the **training dataset**, but only after all training examples have been evaluated does the model get updated. This whole process is like a cycle and it's called a training **epoch**.

- Advantages:
 - **Guaranteed convergence** to the global minimum for convex problems.
 - **Stable** convergence, as the direction of the gradient is determined by the entire dataset.
- Disadvantages:
 - **Computationally expensive** for large datasets, as it requires the entire dataset to compute each gradient update.
 - Prone to getting stuck in local optima for non-convex problems.

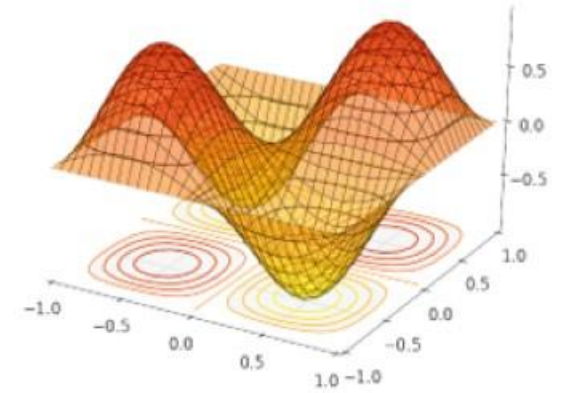
- **Stochastic** gradient descent (SGD) updates the parameters using the gradient computed from a **single randomly** selected data point at each iteration.

- Advantages:
 - **Computationally efficient**, as it uses only one data point to compute each gradient update.
 - Suitable for **large** datasets, as it avoids the need to store the entire dataset in memory.
 - Can escape local optima and explore the parameter space more freely.
- Disadvantages:
 - Convergence can be **noisy** and may **not** reach the global minimum.

- Mini-batch gradient descent is a variation of gradient descent that randomly selects a **subset** (mini-batch) of the training dataset to compute the gradient at each iteration. It updates the parameters based on the average gradient computed from the mini-batch. This approach is a balance between the computational efficiency of stochastic gradient descent and the stability of batch gradient descent.

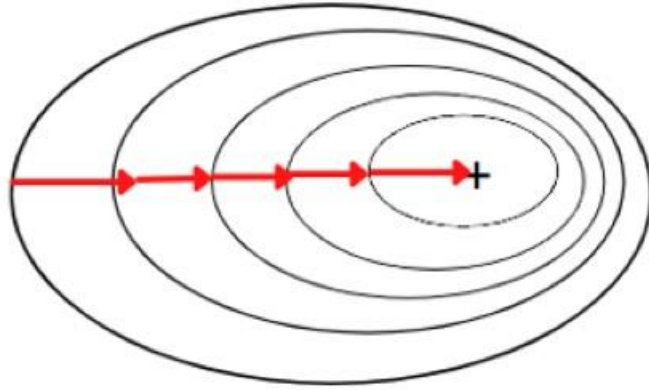
- Advantages:
 - **Faster convergence** compared to batch gradient descent, as it uses a subset of the data.
 - **Less memory-intensive** :it only requires storing gradients for a mini-batch.
 - Suitable for parallel computing, as multiple mini-batches can be processed simultaneously.
- Disadvantages:
 - May not guarantee convergence to the global minimum, for non-convex problems.
 - The learning rate needs to be carefully chosen to balance convergence speed and stability.

STOCHASTIC GRADIENT DESCENT

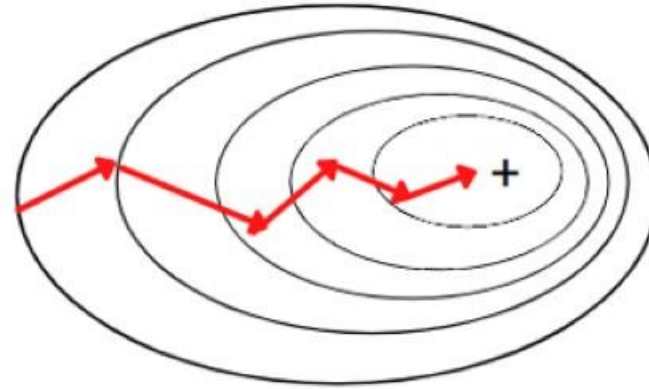
[illegible]

Types of Gradient Descent

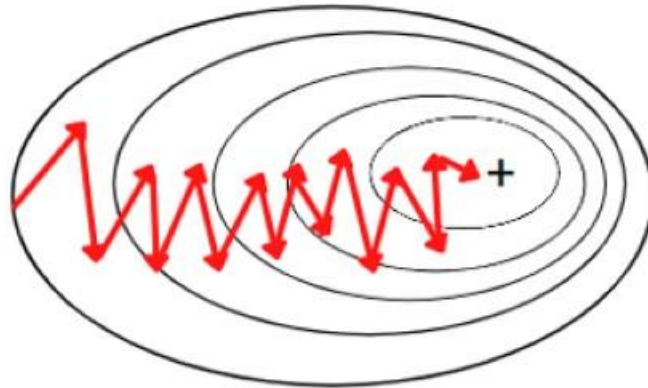
Batch Gradient Descent



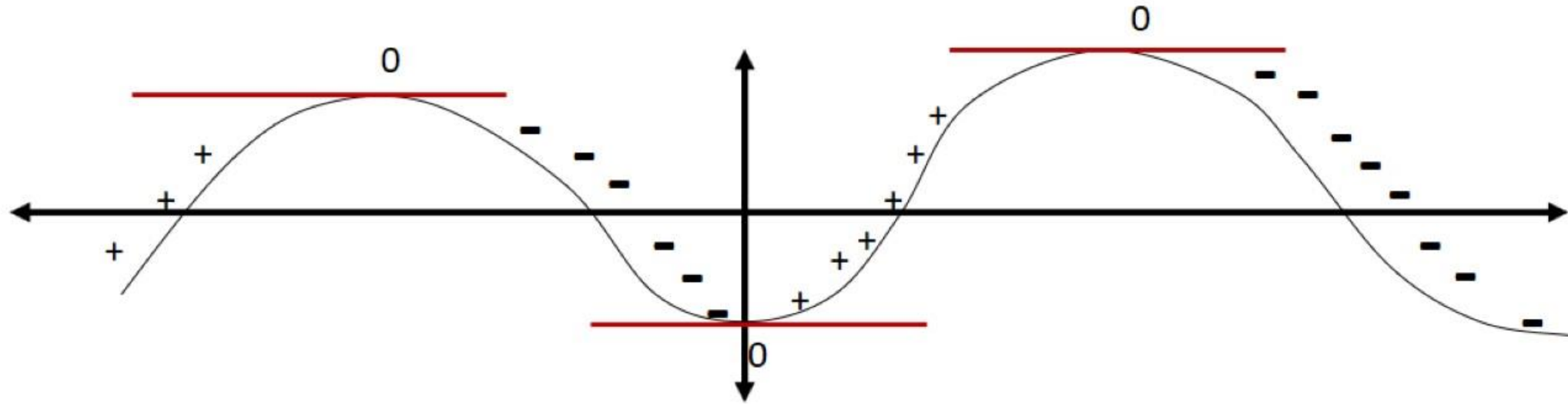
Mini-Batch Gradient Descent



Stochastic Gradient Descent

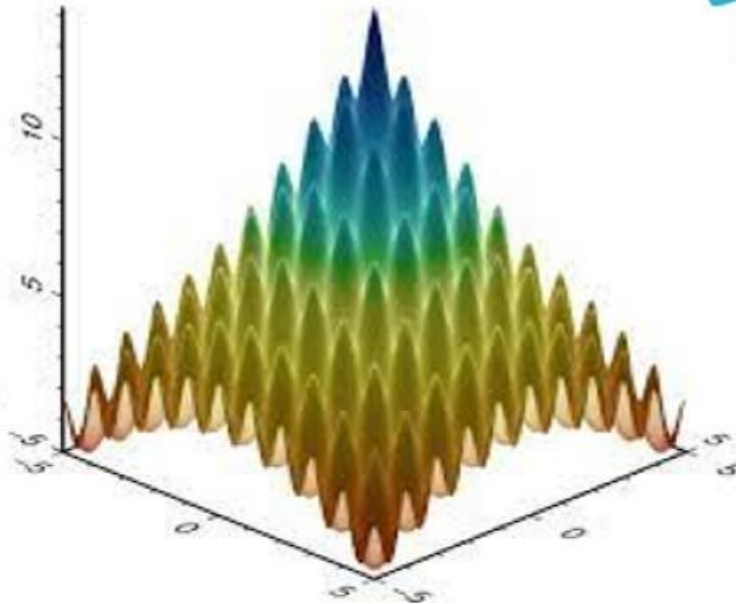
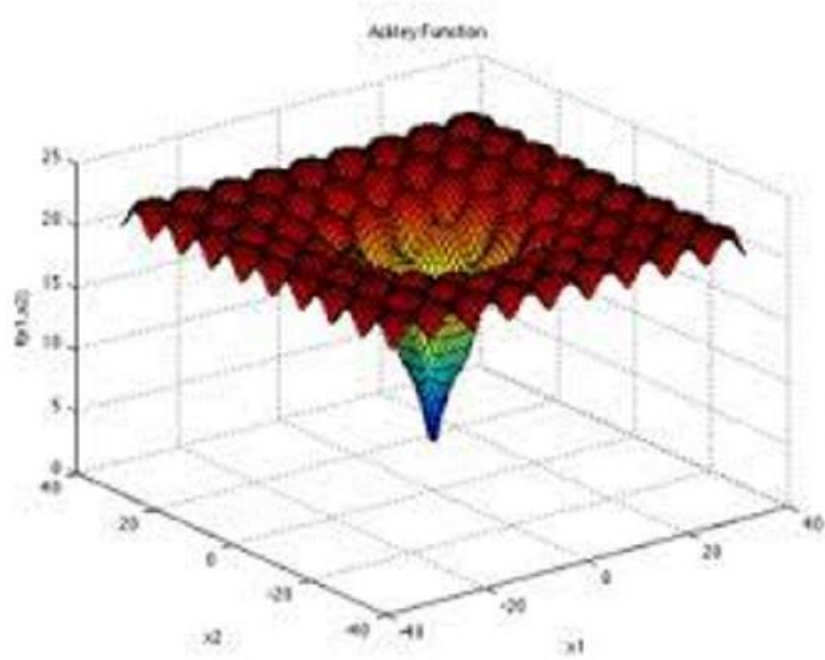


Scalar function of scalar argument



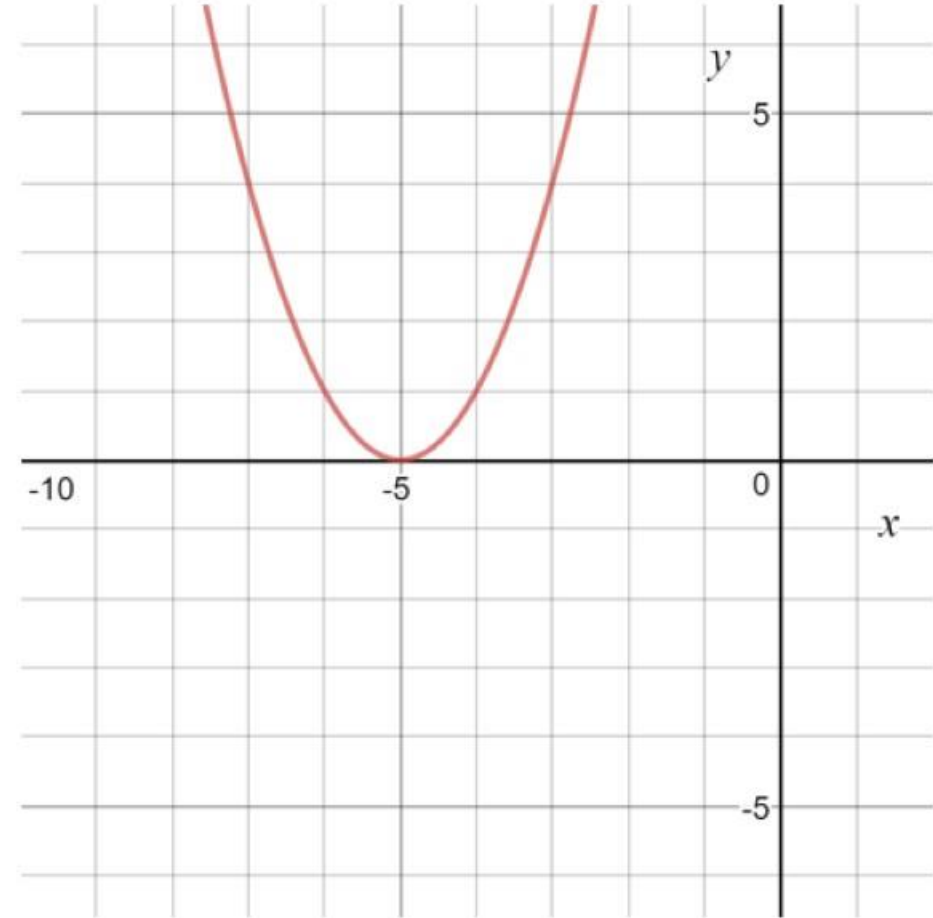
Why?

- Why do we need a gradient descent anyway, if we can just take derivative of function and make it equal to 0, thus finding minimum?



Numerical Example

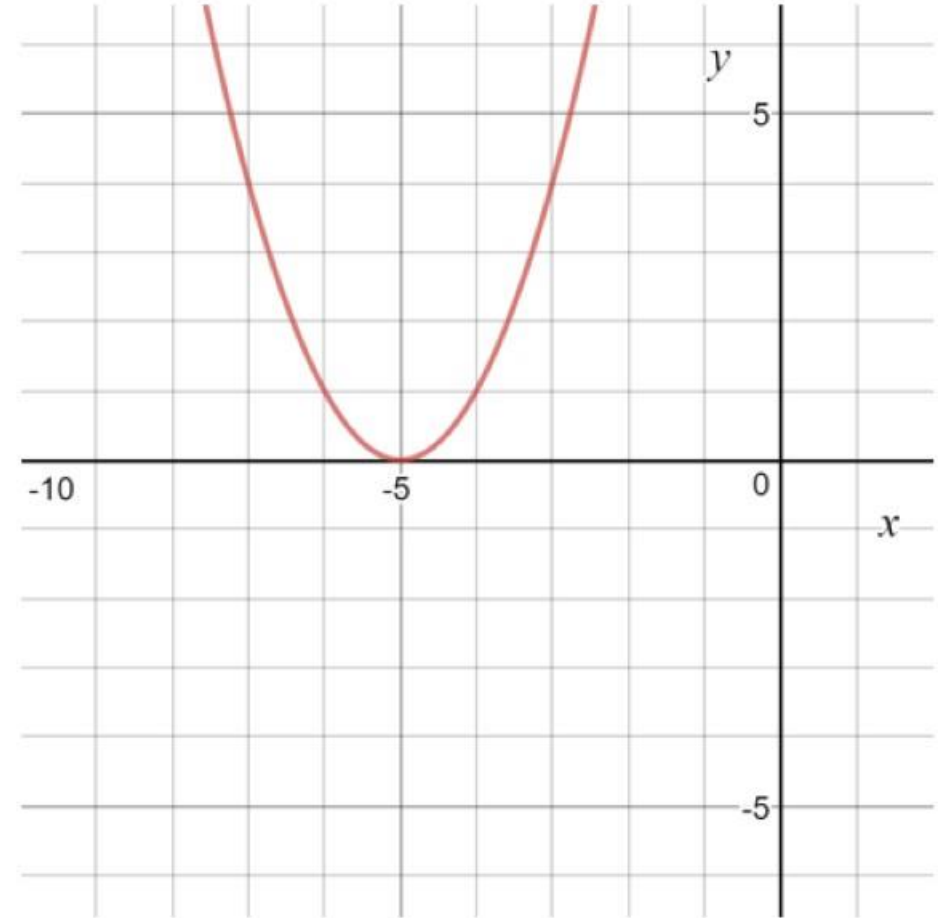
- $y=(x+5)^2$



- Initialization
 - $x_0 = 3$
 - $\alpha = 0.01$
 - $\frac{dy}{dx} = (x+5)^2 = 2 * (x + 5) * 1$

Numerical Example

- Step 1
- $x_1 = x_0 - \alpha * 2 * (x + 5)$
- $x_1 = 3 - 0.01 * 2 * (2 + 5) = 2.84$



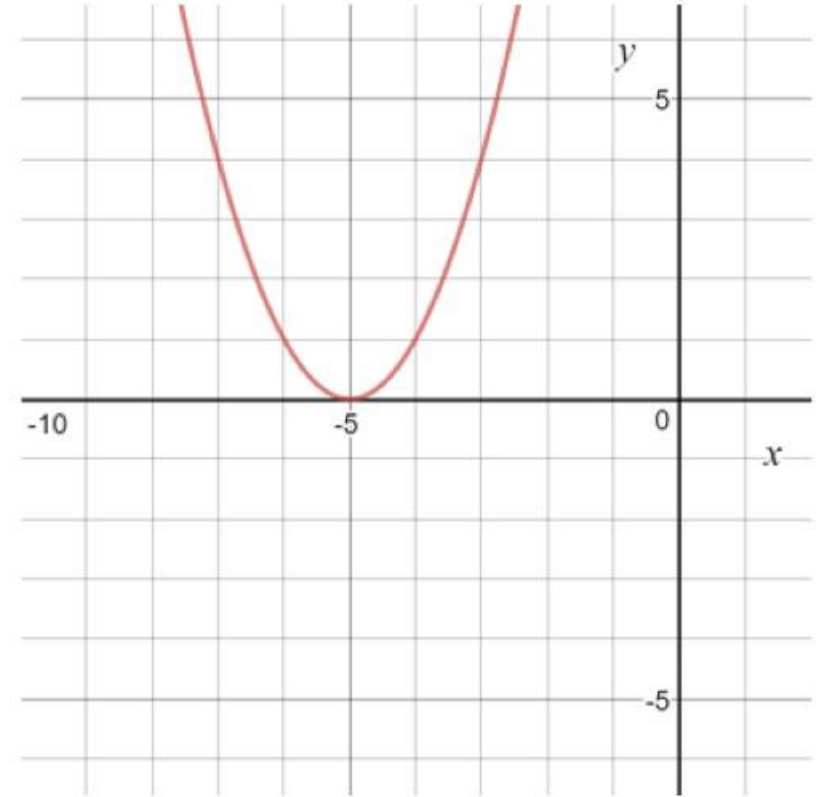
Numerical Example

- Step 1

- $x_1 = x_0 - \alpha * 2 * (x + 5)$
- $x_1 = 3 - 0.01 * 2 * (3 + 5) = 2.84$

- Step 2

- $x_2 = x_1 - \alpha * 2 * (x + 5)$
- $x_1 = 2.84 - 0.01 * 2 * (2.84 + 5) = 2.6832$



Numerical Example

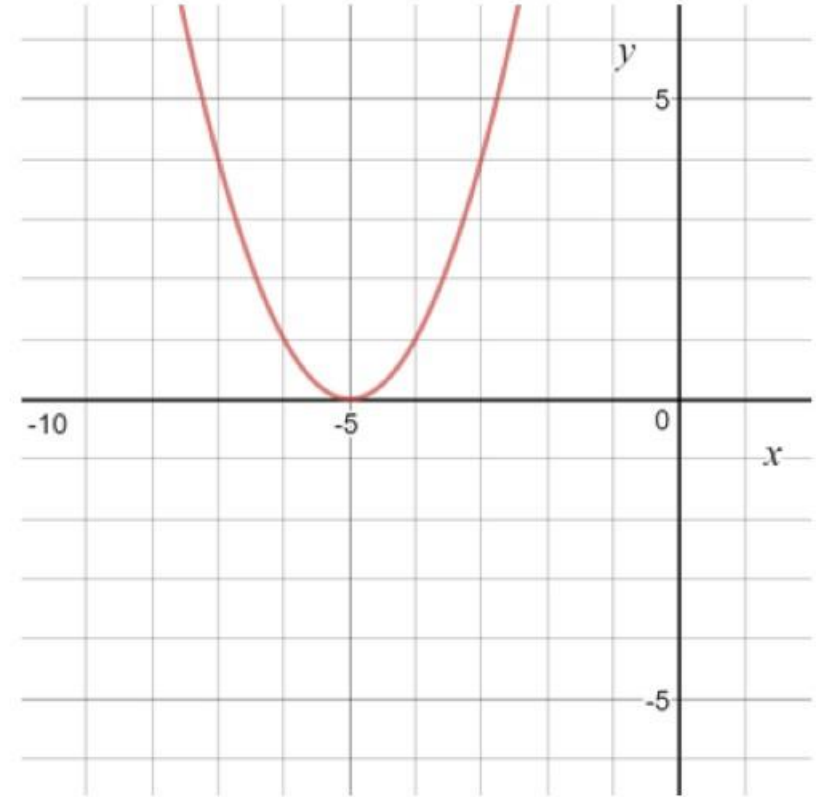
$$\alpha = 1$$

- Step 1

- $x_1 = x_0 - \alpha * 2 * (x + 5)$
- $x_1 = 3 - 1 * 2 * (3 + 5) = -13$

- Step 2

- $x_2 = x_1 - \alpha * 2 * (x + 5)$
- $x_1 = -13 - 1 * 2 * (-13 + 5) = 3$



- Convergence of the objective function: Gradient descent can be stopped when the value of the objective function reaches a certain threshold or when it no longer significantly decreases. This indicates that the algorithm has found a satisfactory solution.
- Maximum number of iterations: Gradient descent can be terminated after a certain number of iterations have been performed.



LECTURE SUMMARY

- ◉ Optimization
- ◉ Derivative
- ◉ Gradient Descent

*Thank
you*

