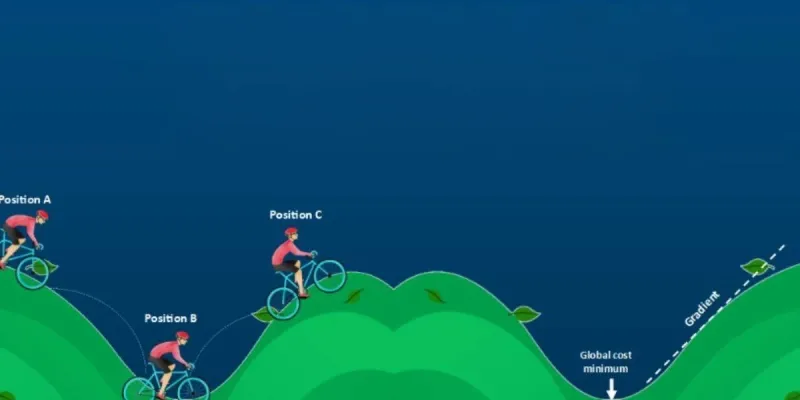
**Introduction to Gradient Descent**

Gradient descent is a crucial optimization algorithm in machine learning used to minimize a cost function

**Gradient Descent** is an algorithm used to find the best solution to a problem by making small adjustments in the right direction. I**t’s like trying to find the lowest point in a hilly area by walking down the slope, step by step, until you reach the bottom.**

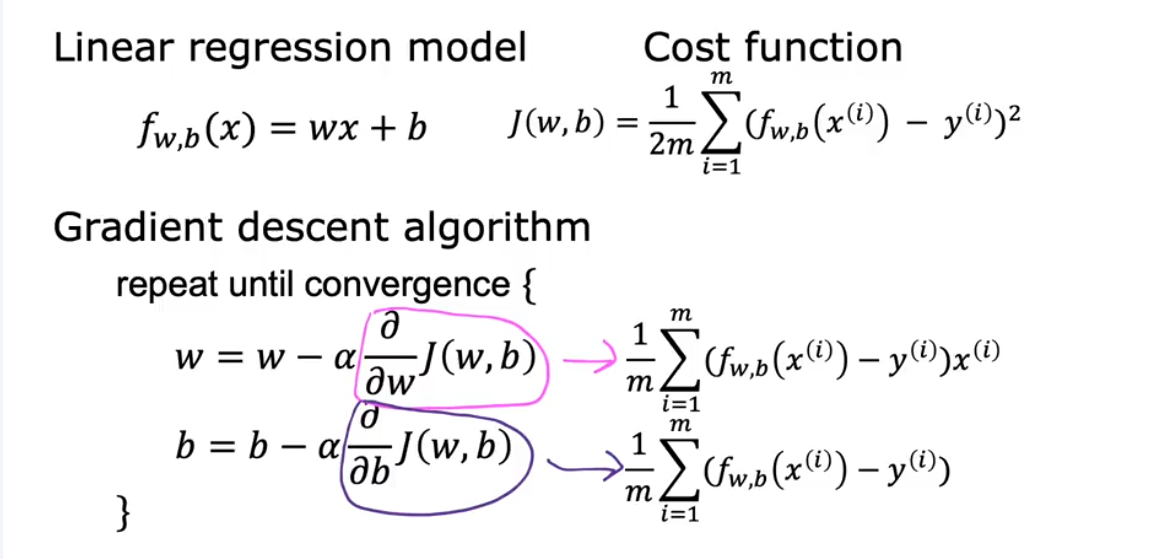


Imagine you're at the top of a hill and your goal is to find the lowest point in the valley. You can't see the entire valley from the top, but you can feel the slope under your feet.

1. **Start at the Top:** You begin at the top of the hill (this is like starting with random guesses for the model's parameters).
2. **Feel the Slope:** You look around to find out which direction the ground is sloping down. This is like calculating the **gradient**, which tells you the steepest way downhill.
3. **Take a Step Down:** Move in the direction where the slope is steepest (this is adjusting the model's parameters). **The bigger the slope, the bigger the step you take.**
4. **Repeat:** You keep repeating the process — feeling the slope and moving downhill — until you **reach the bottom of the valley (this is when the model has learned and minimized the error**).

The key idea is that, just like walking down a hill, Gradient Descent moves towards the "bottom" or **minimum** of the loss function, which represents the error in predictions.

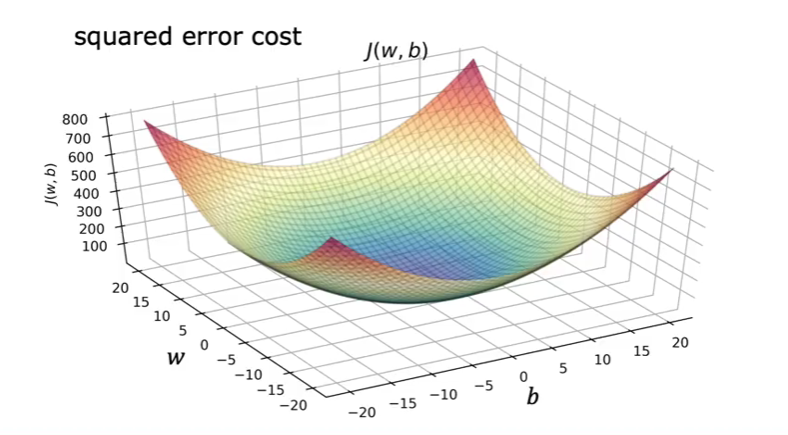
Moving in opposite direction of the gradient allows the algorithm to **gradually descend towards lower values of the function and eventually reaching to the minimum of the function. These gradients guide the updates ensuring convergence towards the optimal parameter values.** Gradual steps used in descent is done by defining **learning rate**.



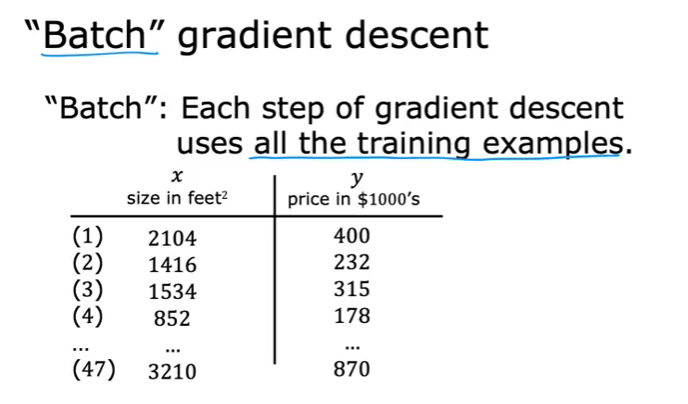
Gradient Descent Algorithm

A math equation with black text

AI-generated content may be incorrect.







**What is Gradient Descent?**

Gradient Descent is an iterative algorithm that **adjusts the model's parameters** to minimize the **cost (or error)**.

for epoch in range(epochs):

y\_pred = theta0 + theta1 \* X # prediction

error = y\_pred - y

grad0 = (1/m) \* sum(error)

grad1 = (1/m) \* sum(error \* X)

theta0 -= alpha \* grad0

theta1 -= alpha \* grad1

**What is Learning Rate?**

Learning rate is a important hyperparameter in gradient descent that controls how big or small the steps should be when going downwards in gradient for updating models parameters. It is essential to determines how quickly or slowly the algorithm converges toward minimum of cost function.

The learning rate, often represented by the symbol alpha (α), is like the size of the steps you take when trying to find the lowest point in a valley. If your steps are too small, you’ll take a long time to reach the bottom, which means your learning process will be slow. On the other hand, if your steps are too big, you might overshoot the bottom and end up further away from it, making it hard to find the right path. So, finding the right learning rate is crucial for efficiently reaching the best solution!

