HyperionDev

Calculus Fundamentals

Task

Visit our website

Introduction

Calculus is the study of change in mathematical functions. For those of you who remember anything from calculus in high school, you will remember that the derivative of a function gives you the instantaneous gradient of that function. This is the driving force behind the ‘learning’ part of machine learning. Most models used in machine learning use an error function. If we can find the instantaneous gradient of that error function, we can get our code to move down that error function to find the optimal parameters.

A note on notation

Before we dive into derivatives, error functions, and instantaneous gradients, let’s cover some basics.

Summation

Now, what about this unusual Greek symbol Σ? This is the capital letter Sigma (Σ), the eighteenth letter of the Greek alphabet. It corresponds to the letter 'S' in the English alphabet.

But we are not here to learn Greek, are we? In mathematics, this ‘S’ stands for sum, of course!

Let’s look at an example. Let’s say we have a set of values $x = \{5, 10, 3\}$. Then, we say

$$\sum_{i=0}^{n} x_i = 18$$

where n denotes the number of items in x. This is just all the values added up. The bottom of the $\sum$ tells us two things: we have an index i, which we begin with a value of 0. Essentially, the bottom part tells us how to start the summation.

The top part of $\sum$ tells us where to stop. In this case, we stop when $i = n$.

If we say

$$\sum_{i=0}^{n} x_i + 2$$

This gives us a value of 24. This is because we just add 2 to each element in x. To get to the value of 24 we did the following:

Step 1: Denote the original sequence of numbers as S = {s1, s2, s3, . . . , sn} where n is the number of the elements.

Step 2: Apply the transformation by adding 2 to each element of S. The new sequence becomes S = {s1 + 2, s2 + 2, s3 +2, . . . , sn +2}

Step 3: We can apply the sum of the sequence. We will be using the example set here: S = {5 + 2, 10 + 2, 3 +2}

Therefore the value is (5 + 2) + (10 + 2) + (3 + 2) = 24

Like all good maths, this can be translated directly to code.

Let's look at how the first example is translated into code:

$$\sum_{i=0}^{n} x_i$$

```
# Our list of values
x = [5, 10, 3]

# Sum of all values
sum_total = 0

# We denote n as the number of elements in x
n = len(x)

# Sum values
for i in range(n):
    sum_total += x[i]

print(f"Our sum total is {sum_total}")
```

Output:

```
Our sum total is 18
```

For the second example, the same values for `x`, `n` and `sum_total` will be used.

$$\sum_{i=0}^{n} x_i + 2$$

```python
# Reset our sum_total back to 0
sum_total = 0
# Add 2 to each value then sum values
for i in range(n):
    sum_total += x[i] + 2

print(f"Our sum total is {sum_total}")
```

Output:

```
Our sum total is 24
```

Use this to your advantage!

Function notation

I am sure you have heard of functions before, we make them all the time in Python. In mathematical notation, it is basically the same concept. A function is defined as something that takes an input and returns an output.

Let's say we want to define a function f that takes an input x and returns double the input. In mathematical notation, we would use something like:

$$f(x) = 2x.$$

In mathematics, $2x$ is just the lazy man's way of saying 2 multiplied by x, which just means $2 \times x$.

So, using this definition of f, let's find out what double 16 is:

$$f(16) = 2(16)$$

(The brackets here just mean that we are multiplying 2 by 16. We use this notation because we are replacing x with 16)

$$f(16) = 32.$$

Let’s see what this looks like in Python:

```python
def f(x):
    return 2 * x

print(f(16))
```

Output:

```
32
```

Limit notation

Limits are an important concept in calculus. Calculus is simply the study of change in mathematics, limits describe how functions change as a specific variable approaches a specific value.

Let’s look at a simple example. Let’s define a function:

$$f(x) = \frac{1}{x}.$$

Let’s say that we want to see what happens to the output of the function as x gets closer and closer to 2:

$$\lim_{x \to 2} f(x) = f(2) = \frac{1}{(2)} = 0.5 .$$

This is read as the limit of $f(x)$ as x approaches 2 is 0.5. Another way to say this is as x gets infinitely closer to 2, $f(x)$ gets infinitely closer to 0.5.

Well, this is fine, but how is this useful? Why don’t we just use $f(2)$?

To answer that, let’s look at $f(0)$:

$$f(0) = \frac{1}{(0)}.$$

Well, this is not good. Anything divided by 0 is undefined: the value does not exist!

However, if we attach a limit to this, we can mathematically say this:

$$\lim_{x \to 0} f(x) = \lim_{x \to 0} \frac{1}{x}$$

This is where we start using our words to reason things out. We know that we are looking at values as x approaches 0.

So let's look at this:

x	f(x)
2	0.5
1	1
0.5	2
0.1	10
0.0001	10 000

Well, it looks like, as x approaches 0, $f(x)$ gets larger. So how large does $f(x)$ get when x gets infinitely close to 0? Infinitely large! Yes, so you are correct in saying that it is infinity.

$$\lim_{x \to 0} f(x) = \lim_{x \to 0} \frac{1}{x} = \infty$$

Note that the symbol '∞' represents infinity.

Okay, but infinity does not help us at all. We cannot program that into our computers.

Now, let's get more complicated:

$$f(x) = \lim x \to 0 \frac{x}{x}.$$

Well, the denominator would evaluate to 0, and the numerator would evaluate to 0. So, what does $\frac{0}{0}$ evaluate to? In limits, everything divided by 0 would give infinity, but 0 divided by everything would give 0. So is it infinitely large, or is it 0? It is neither!

Those eagle-eyed readers will notice one other thing: $\frac{x}{x}$ is always 1! But how can we prove this is the correct answer? Same as earlier, let's inspect these values:

x	f(x)
2	1
1	1
0.5	1
0.1	1
0.0001	1

So it looks like it remains the same as x gets infinitely closer to 0. This observation leads us to the concept of a derivative, which helps us understand how a function behaves as its inputs change.

Now that we have established a language for expressing complex ideas using mathematical notation, let's explore derivatives.

What is a derivative?

In the previous section, we observed how the function $f(x)$ changes as x approaches zero. The constant behaviour we noticed hints at a broader concept in mathematics called a derivative.

A derivative represents the rate at which a function changes at any given point. It is a fundamental concept in calculus, describing how a function's output value changes as its input value changes. The derivative of $f(x)$ with respect to x is denoted as $f'(x)$, $\frac{df}{dx}$ or $\frac{d}{dx}f(x)$. This can be read as the gradient of f with respect to x. In other words, given a specific x, what is the instantaneous gradient of the function f?

Say we have y = f(x). In simpler terms, this just means that we have y, which is obtained by applying a function *f* to a variable *x*. The gradient of *f* at exactly point *x* is given as:

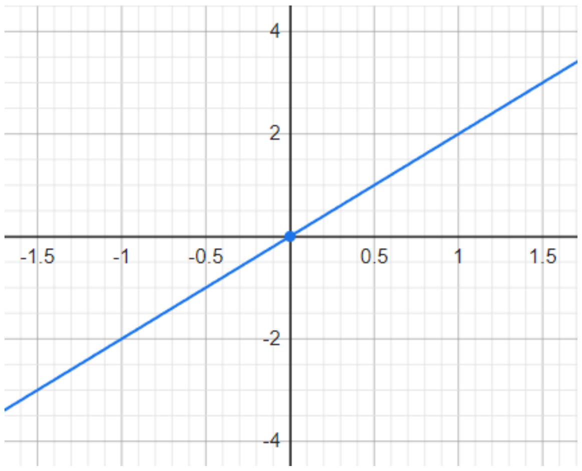
$$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

So, now what? How does this apply to machine learning? What do all of these symbols even mean? Let’s start with a simple, intuitive explanation of what a derivative is before we dive deeper into the equation itself.

Intuitive explanation

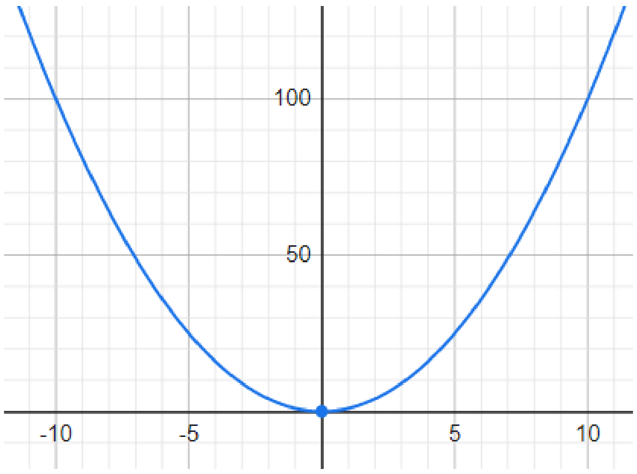
We will start by looking at the slope of a straight line, which offers an easy way to understand how a function changes with input.

Slope of a line: For a straight line, the derivative is simply the slope of the line, which is constant. Let's work with the function $f(x) = 2x$. Graphically, this function looks like this:



You can check that for any input x, the output y is twice that number; for example $f(1) = 2(1) = 2$. Therefore, the derivative of $f(x)$ is 2. Mathematically, we write $f'(x) = 2$.

Slope of a curve: For a curve, the slope is not constant as it is for a straight line. Let's look at the function $f(x) = x^2$. Graphically, this function looks like this:



You can check that for any input x, the output y is the square of that input. For example, $f(5) = 5^2 = 25$ and $f(10) = 10^2 = 100$. We can see that for a curve, the rate at which the function's output value changes is not the same at different points along the curve. Therefore, the derivative is **not constant**.

The derivative of $f(x) = x^2$ is $f'(x) = 2x$ (we will see how to calculate this derivative shortly). $f'(x)$ tells us how the curve is changing at that **exact** point. For example:

- At $x = 5$, we have $f'(5) = 2(5) = 10$. This means the slope of the function is 10 at this point.
- At $x = 10$, we have $f'(10) = 2(10) = 20$. This means the slope of the function is 20 at this point.

Formal definition

Remember the equation for derivation? If not, it is the one we introduced earlier in this section, the one we said we would revisit after going over the intuition behind a derivative:

$$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

This formula gives you the instant gradient of any function f at any point x.

This is what we call the first-order derivative. For now, we will just call it the derivative, as you do not need to know anything more than this. For a more detailed explanation of this equation, you can watch this informative **video**.

Let's look at the numerator. As h approaches 0, $f(x+h) - f(x)$ becomes 0. The same applies to the denominator. This gives us one of those classic $\frac{0}{0}$ situations, which means that there should be something useful here!

Next, let's look at the formula for gradients. Because we denote y as $f(x)$ (because y is the output of this function), we will use this to denote our y value. To calculate gradient, we need two points. Let's say we have any point x, which gives us $f(x)$ when we put it into the function. Now, we define another point $(x+h)$, where we just add a value h onto our x value. This will give us $f(x+h)$. The gradient of these two points, according to the **equation for gradient**, is:

$$m = \frac{f(x+h) - f(x)}{(x+h) - x} = \frac{f(x+h) - f(x)}{h}$$

Because we want to find the instantaneous gradient, this means that the difference in x values (which is h) must get smaller and smaller, until it is infinitely small. Hence, we have the equation:

$$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

The derivative of a function gives another function. This derivative function, $\frac{df}{dx}$ gives the instantaneous gradient of f at the value of x. There are a lot of rules when it comes to differentiation. In this lesson, we will look at the power rule. The other rules are beyond the scope of this task. If you are interested in learning about them, you can begin **here**.

The power rule

The power rule states that if you have a function of the form $f(x) = x^n$, where n is any real number (that is, any number that is positive, negative, whole, or fractional), the derivative of that function is:

$$f'(x) = nx^{n-1}$$

How this works is:

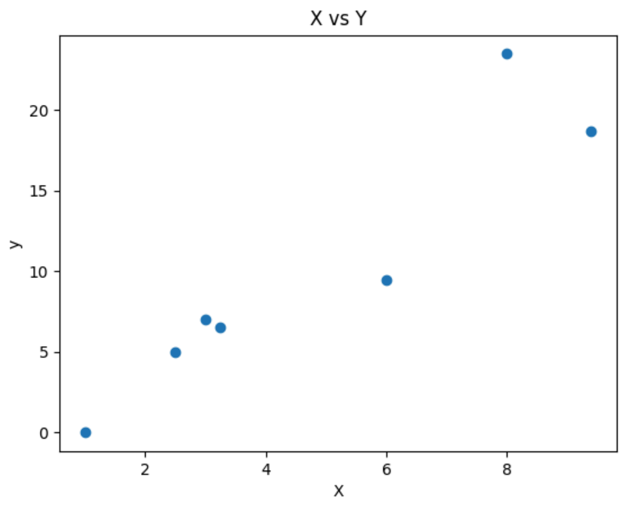
1. We identify the exponent n, which is the power to which x is raised.
2. Multiply by the exponent, which means we bring the exponent n in front of x.
3. Subtract 1 from the exponent.

Let's apply these steps to the function $f(x) = x^2$ to calculate its derivative. The first step is to identify the exponent, which is 2. Next, we bring the exponent in front of x so we have $2x^2$. The last step is to subtract 1 from the exponent: $2x^{2-1} = 2x^1 = 2x$. And that is our derivative.

$$f'(x) = 2x^{2-1} = 2x$$

A simple line

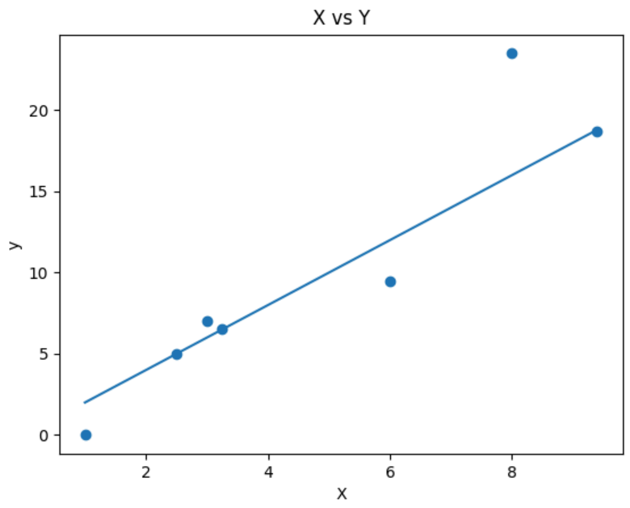
Let's look at a practical example. Let's say we are given the following data:



We can kind of guess that we can model the data with a line of best fit. For the sake of simplicity, we will just assume that this line starts at the origin (i.e. at x = 0 and y = 0).

$$f(x) = mx$$

where *f* is our model, *x* is our input, and *m* is our model parameter. The idea is that we need to find an *m* that will give us a model *f* that, when given an input *x*, will give us a good prediction of the data. This will probably look something like this:



Next, let's see how derivatives are used to find the optimal parameters that minimise the error function.

The error function

An error function, also known as a loss function, measures how well a machine learning model predicts a target value. It quantifies the difference between the model's predicted output and the actual, correct output. By minimising this error function, the model can be improved to make more accurate predictions.
For continuous data, we will use the **mean squared error** (MSE) function. Remember that the formula for this is:

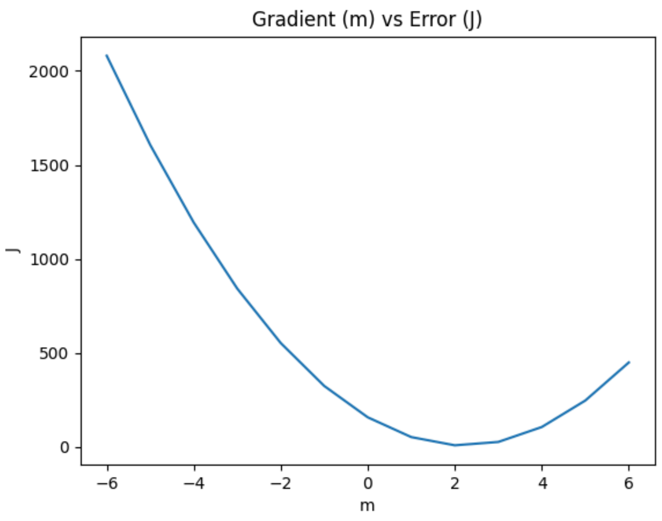
$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Where y_i is the actual observed value and $\hat{y}_i$ is the predicted value for the i-th observation. Let's look at a slightly different version of this formula:

$$J(m) = \frac{\sum_i (y_i - f(x_i; m))^2}{n}$$

What does all of this mean? Let's start with the numerator. In English, this translates to "the sum over the index of *i* in the data, of y_i minus $f(x_i)$ (a.k.a our model's prediction $\hat{y}_i$) given a specific m, squared". The denominator, n, just refers to the number of data points. So, $J(m)$ is simply MSE expressed differently.

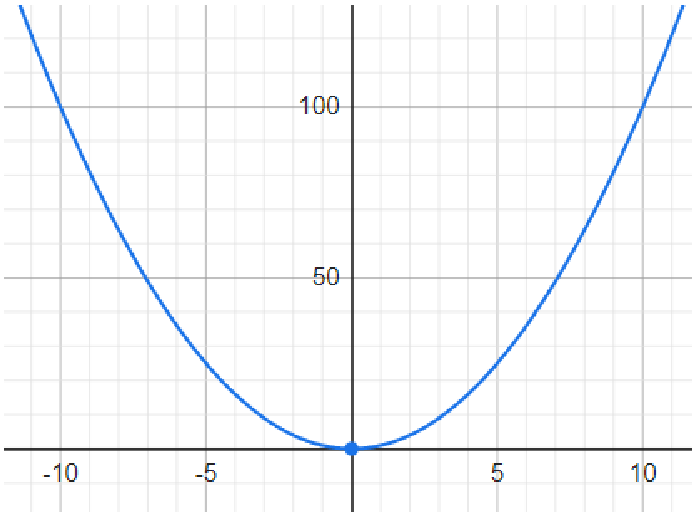
So, we denote our error as *J*. That means now we have the error function we were looking for. Let's see what it looks like. Firstly, we must be mindful that $J(m)$ depends on *m*, which makes *J* the dependent variable, and *m* the independent variable.



Now, what does this mean? Well, it looks like the lowest point of our curve is near *m=2*. This lowest point represents the lowest value of *J*. Remember, our objective is to minimise our model's error, J. So, this means that 2 is our target value for our model parameter m. But how will the computer know this? The answer lies in using derivatives to find **turning points**.

Turning points

Let's go back to the function $f(x) = x^2$ which looks like this:

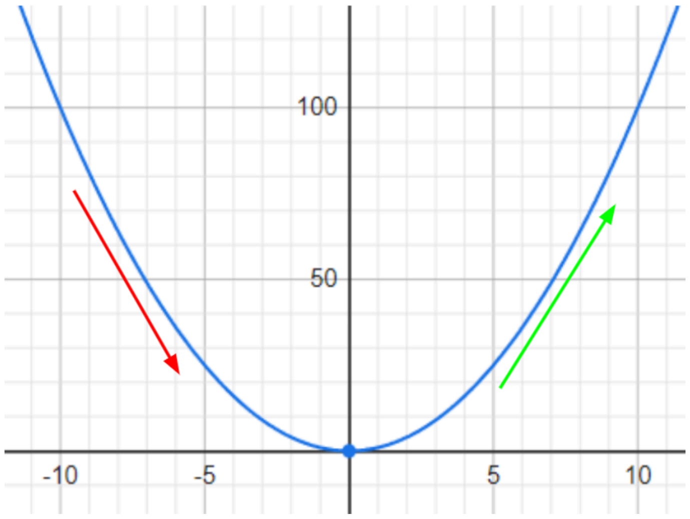


We can see the lowest point of this curve is at *x = 0*. What is the gradient of the curve at this point? We can use the first derivative of our function, $f'(x) = 2x$, to figure this out.

$$f'(0) = 2(0) = 0$$

So we see that the gradient at the lowest point of this curve is 0. This is because that point of the curve is a turning point and the gradient at a turning point is always zero. How do you identify a turning point?

In the figure below, notice that to the left of *x = 0*, the function is decreasing (red arrow) and to the right of x = 0, the function is increasing (green arrow). The turning point on a curve is where the function changes direction, going from decreasing to increasing or from increasing to decreasing.



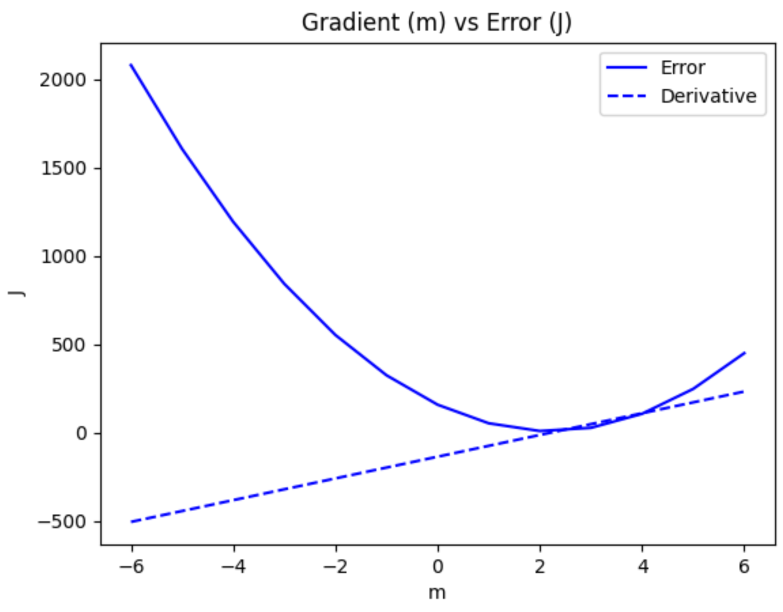
So, for a U-shaped curve or a U-shaped segment of a more complex function, we can identify the turning point, and consequently the lowest point, by finding where the derivative equals zero.

Using calculus to minimise error

While finding the derivative of the error function may seem more complex than applying the basic power rule, don't worry—we've got you covered. Instead of diving into the detailed derivation, we'll focus on the key concepts you need to know. If you're curious about the in-depth steps, feel free to check out **this great guide on how to derive the error function**. Here's the derivative of the error function:

$$\frac{dJ}{dm} = \frac{\sum_i -2x_i(y_i - f(x_i; m))}{n}$$

Let's see what this looks like:



The dotted line represents the derivative (or gradient), showing how the slope of our error function changes for different values of *m*. For example, when *m* is -6, the dotted line shows us that the gradient of the error function is -500. According to this dotted line, what is the gradient of our error function when *m* is 2? The gradient is zero, indicating that we have reached a critical turning point. This is the point of minimum error in this context. So, our algorithm will minimise the error by searching for this point where the error function's gradient is zero.

To visualise this, you can imagine this algorithm as a ball rolling down this error curve. The error function goes from having a negative gradient (the ball rolling downhill) to a positive gradient (the ball rolling uphill). The ball, always searching for the lowest point, will settle when the gradient is 0 at the turning point.

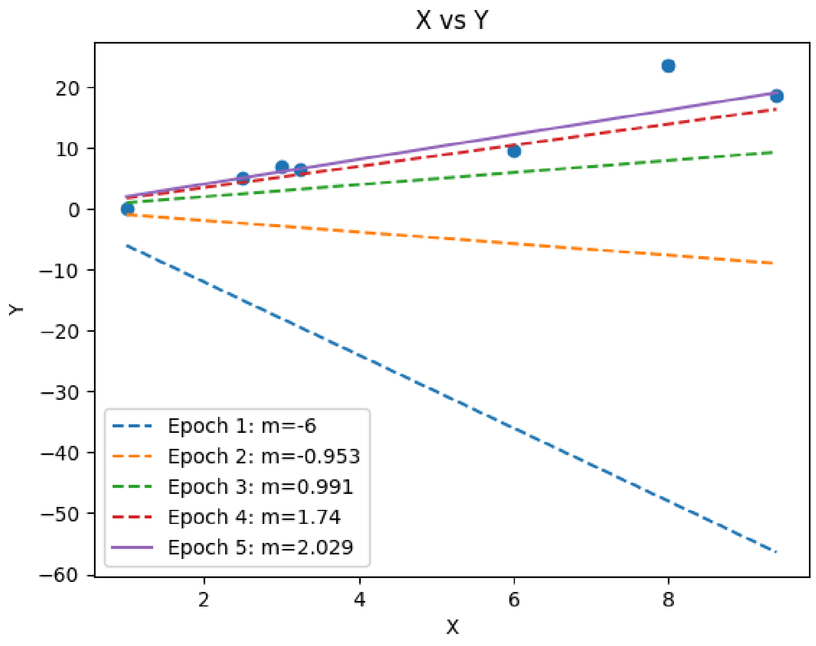
Gradient descent

Gradient descent is the process of rolling the ball down the hill. The formula for gradient descent is:

$$m' = m - \alpha \left(\frac{dJ}{dm} \right)$$

In English, this just means "the current gradient minus alpha times the derivative of the error function". Wait, no one mentioned an alpha?! No need to stress, this is just the symbol we use to define something called the "learning rate". You can think of the learning rate as the parameter that controls how far the ball moves at each step. A small learning rate is like taking small, cautious steps, which means the ball moves very slowly down the hill. A large learning rate is like taking big leaps, where the ball moves faster.

Let's say we want to fit a line to a set of data points (the blue dots in the image below). Our model will iterate through the data multiple times, updating the gradient with each pass. Each complete pass through the dataset is referred to as an epoch. In the image below, we can see a plot of the linear regression model's progress over five epochs



You can see the gradient of the line improving with each epoch, and notice how the difference between m values becomes smaller as the model converges (settles on the best fit). The gradient descent expression we saw earlier is crucial for achieving this. It consists of the following components:

- m' is the new value of the slope after the update (after the ball rolls).
- m is the current value of the slope before the update (before the ball rolls).
- α is the learning rate which controls how big each update step is.
- $\frac{dJ}{dm}$ is the gradient of the cost function J with respect to m, showing how much the cost changes for small changes in m. It points in the direction of the steepest increase in error, and we subtract it to move in the direction that decreases the error.

Think of the ball as starting at some point on the hill, where the model has an initial slope m. For each epoch, the gradient $\frac{dJ}{dm}$ is calculated, which tells us which direction the ball should roll to reduce the error. The learning rate α determines how far the ball rolls along this direction leading to a new point on the hill where the updated slope m' is found. This process is repeated, updating m after each iteration until the error is minimised (when the ball finally reaches the bottom of the hill).

Instructions

Explore the examples in your folder before proceeding to the practical tasks.

Practical task 1

1. Create a file called **polynomial.py**.
2. Write Python code applying the power rule to a polynomial with one term. The program should prompt the user to enter:
   - the coefficient,
   - the power of a polynomial term, and
   - the x-coordinate of the point where the user would like to calculate the gradient.
3. Apply the power rule to find the derivative, and then output the result.

   For example:

   ```
   Enter the coefficient of x: 2
   Enter the power to raise x: 3
   Enter the x-coordinate of the point where you would like to calculate the
   gradient: -4
   ```

   Expected Output:

   ```
   "The derivative of the polynomial 2x³ is 6x². The gradient at the point x
   = -4 is 96."
   ```

Practical task 2

Open **calculus.py**. This file should perform linear regression, similar to what is described in the lesson. You will notice that all of the functions have been left blank. It is your job to fill these functions in.

1. `model(x, m)` should be $f(x)$
2. `error_function(m)` should be $J(m)$
3. `derivative(m)` should be $\frac{dJ}{dm}$
4. `update_step(m)` should return a new `m`, given a specific `learning_rate`

Hint: maths translates surprisingly well to code. The $\sum$ symbol, for example, is just a for loop that adds everything together.

Important: Be sure to upload all files required for the task submission inside your task folder and then click "Request review" on your dashboard.

Share your thoughts

Please take some time to complete this short feedback **form** to help us ensure we provide you with the best possible learning experience.

Reference list

IEEE. (1993). IEEE Standards Collection: Software Engineering. IEEE Standard 610.12-1990.