Gradient Descent Made Simple: Study Time vs. Grade

1. Introduction

This document shows a use case calculation of gradient descent, a important method in machine learning for improving model predictions, for the aim of learning the material.

It focuses on how the derivative of an error function makes adjustments to model parameters, such as weights in a neural network SO the focus is more the mathematical details.

Using an example of predicting a student's grade based on hours studied, we will show the process with calculations, updates, and visualizations.

The content aligns with the topic '2.1 Calculus in Machine Learning'.

2. Gradient Descent Overview

Gradient descent is a method to make a model better.

It adjusts the model step-by-step to reduce errors in predictions.

In our example, we predict grades using a straight line:

$$\hat{y} = b + m \cdot x$$

where:

- $\hat{y}$: predicted grade
- x: hours studied
- b: starting point of the line (intercept)
- m: steepness of the line (slope)

Our goal is to find the best b and m to make predictions close to actual grades.

3. Error Functions

An error function (E) shows us how wrong a model's prediction is compared to the true value.

So, it measures the difference between the prediction and the reality.

To calculate this it needs two inputs: The predicted value (y1) and the actual value (y).

The model uses the gradient (derivative) of the error to adjust its weights, so it can perform better next time.

In supervised learning there are two main types of error functions.

Both correlate to the two types of neural networks:

- Regression: This predicts continuous values
- Classification: This categorizes data into discrete labels.

We would like to predict a students grade based on the hours studied.
We will make use of regression loss functions.
This means we can choose about two types of error functions:

- Mean Squared Error (MSE): Finds the average of squared differences between the predicted and actual values.
- Mean Absolute Error (MAE): Calculates the average magnitude of the differences between predicted and actual values.

For our example we will go ahead and use the MSE.

4. Derivatives and Gradients

The derivative of an error function tells us how sensitive the error is to changes in a variable, typically a weight or bias in the model.

You can calculate the error function E(w) by using the following the formula with respect to the weight w:

- dE/dw which means, "the rate of change of E with respect to w"

This formula is used when E is a function of one variable, so E=f(w).
It assumes that all variables depend only on W.

- $\partial E/\partial w$ which means, "how E changes with w"

This formula is used when E is a function of multiple variables.

So, it measures how E changes when only w changes, holding the other variables constant

Both terms are very closely related, but are not the same:

- Derivative is for functions with one input
- So, it applies to single-variable functions, such as f(x).
- It will also produce one output which will tell you the slope of function at a point
- Df/dx
- A gradient is for functions with multiple inputs.
- It applies to multi-variable functions, like f(x,y,z..)
- It will produce a vector of partial derivatives as an output which tells you the direction and rate of the steepest increase
- $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \ldots\right]$

5. Example Dataset

Here is our data:

Hours Studied (x)	Grade (y)
1	2.1
2	2.9
3	4.2

We set $m = 1$ for now and focus on finding the best b using gradient descent.
We'll start with using the following data points:

- Initial weight, w = 0
- Learning rate, $\eta = 0.1$

We will work through how a neural network adjusts its weight using the derivative of the error function, based on our data.

6. Initial Predictions and Errors

We start with $b = 0$ and $m = 1$.
The predictions are:

$$\hat{y}_1 = 0 + 1 \cdot 1 = 1$$
$$\hat{y}_2 = 0 + 1 \cdot 2 = 2$$
$$\hat{y}_3 = 0 + 1 \cdot 3 = 3$$

Errors (differences between actual and predicted grades):

$$2.1 - 1 = 1.1$$
$$2.9 - 2 = 0.9$$
$$4.2 - 3 = 1.2$$

Square the errors:

$$1.1^2 = 1.21$$
$$0.9^2 = 0.81$$
$$1.2^2 = 1.44$$

Total error (sum of squared errors):

$$L(b) = 1.21 + 0.81 + 1.44 = 3.46$$

Why Square Errors?
- Squaring makes all errors positive so they don't cancel out.
- It gives more weight to bigger errors.
- It makes the error function smooth, which helps gradient descent work well.
- It matches common methods used in math.

7. Error Function Derivation

The error function (called the loss function) is:

$$L(b) = \sum_{i=1}^{n} (y_i - (b + mx_i))^2$$

This measures the total squared error for all data points.

For our data:

- Point 1: $y_1 = 2.1$, $x_1 = 1$, error $= 2.1 - (b + 1) = 1.1 - b$
- Point 2: $y_2 = 2.9$, $x_2 = 2$, error $= 2.9 - (b + 2) = 0.9 - b$
- Point 3: $y_3 = 4.2$, $x_3 = 3$, error $= 4.2 - (b + 3) = 1.2 - b$

Square each error:

$$L(b) = (1.1 - b)^2 + (0.9 - b)^2 + (1.2 - b)^2$$

Expand each term:

- $(1.1 - b)^2 = 1.21 - 2.2b + b^2$
- $(0.9 - b)^2 = 0.81 - 1.8b + b^2$
- $(1.2 - b)^2 = 1.44 - 2.4b + b^2$

Add them:

$$\begin{aligned} L(b) &= (1.21 + 0.81 + 1.44) + (-2.2b - 1.8b - 2.4b) + (b^2 + b^2 + b^2) \\ &= 3.46 - 6.4b + 3b^2 \\ L(b) &= 3b^2 - 6.4b + 3.46 \end{aligned}$$

This is the error function used in the plots.

To reduce the error, we find the slope of the error function (called the gradient):

$$\frac{dL}{db} = -2 \sum_{i=1}^{n} (y_i - (b + mx_i))$$

For our data with $m = 1$:

$$\begin{aligned} \frac{dL}{db} &= -2\big[(2.1 - (b + 1)) + (2.9 - (b + 2)) + (4.2 - (b + 3))\big] \\ &= -2\big[(1.1 - b) + (0.9 - b) + (1.2 - b)\big] \\ &= -2(3.2 - 3b) = -6.4 + 6b \end{aligned}$$

This section explains how we find the gradient $\frac{dL}{db}$.

The error function is:

$$L(b) = \sum_{i} (y_i - (b + mx_i))^2$$

- y_i: actual grade - b: intercept - m: slope - x_i: hours studied - Sum over all data points (1, 2, 3).

Use the Chain Rule:

The error function is a sum of squared errors.

For each point, the error is $(y_i - (b + mx_i))^2$.

Let $u_i = y_i - (b + mx_i)$, so the error is u_i^2.

The derivative of u_i^2 with respect to b is:

$$\frac{d}{db}(u_i^2) = 2u_i \cdot \frac{du_i}{db}$$

Find $\frac{du_i}{db}$:

$$u_i = y_i - (b + mx_i)$$

$$\frac{du_i}{db} = -1$$

(y_i and mx_i don't change with b).

Combine:

$$\frac{d}{db}(y_i - (b + mx_i))^2 = 2(y_i - (b + mx_i)) \cdot (-1) = -2(y_i - (b + mx_i))$$

Sum over all points:

$$\frac{d}{db}L(b) = \sum_i [-2(y_i - (b + mx_i))]$$

$$\frac{dL}{db} = -2\sum_i (y_i - (b + mx_i))$$

This is the gradient used in gradient descent.

8. Gradient Descent Process

In a neural network you can adjustment the weight to minimize the error using a method called gradient descent.

The derivative of the error function tells us how much and in which direction each weight affects the total error.

The gradient shows how the error changes when we change b.

If the gradient is positive, increasing b makes the error bigger, so we decrease b.

If the gradient is negative, increasing b makes the error smaller, so we increase b.

When the gradient is zero, the error is at its smallest.

This happens at:

$$-6.4 + 6b = 0$$

$$6b = 6.4 \implies b = \frac{6.4}{6} = \frac{16}{15} \approx 1.0667$$

This $b \approx 1.0667$ gives the smallest error (about 0.0467), as shown in Section 9.

Gradient descent updates b to reduce the error:

$$b_{\text{new}} = b_{\text{old}} - \alpha \cdot \frac{dL}{db}$$

where α is the learning rate and $\frac{dL}{db} = -6.4 + 6b$.

For our data, start with $b = 0$ and $\alpha = 0.1$:

$$\frac{dL}{db} = -6.4 + 6 \cdot 0 = -6.4$$
$$b_{\text{new}} = 0 - 0.1 \cdot (-6.4) = 0.64$$

New predictions:

$$\hat{y}_1 = 0.64 + 1 = 1.64 \quad (\text{error: } 2.1 - 1.64 = 0.46)$$
$$\hat{y}_2 = 0.64 + 2 = 2.64 \quad (\text{error: } 2.9 - 2.64 = 0.26)$$
$$\hat{y}_3 = 0.64 + 3 = 3.64 \quad (\text{error: } 4.2 - 3.64 = 0.56)$$

New total error:

$$L = 0.46^2 + 0.26^2 + 0.56^2 \approx 0.2116 + 0.0676 + 0.3136 = 0.5928$$

This is better than the initial error of 3.46.
We repeat the process.
For $b = 0.64$:

$$\frac{dL}{db} = -6.4 + 6 \cdot 0.64 = -2.56$$
$$b_{\text{new}} = 0.64 - 0.1 \cdot (-2.56) = 0.896$$

Each step makes the error smaller, getting closer to $b \approx 1.0667$.
The learning rate α is important:

- Too big ($\alpha = 0.5$): $b_{\text{new}} = 3.2$, which overshoots and increases the error.
- Too small ($\alpha = 0.001$): $b_{\text{new}} = 0.0064$, which is slow.
- Stop when the gradient is small or the error doesn't change much.

This method keeps adjusting b until the error is as small as possible.
Gradient descent steps:

1. Find the gradient (slope of the error).
2. Update b using the learning rate.
3. Calculate the new error.
4. Stop when the error doesn't decrease much.

Formula weight update rule is:

$$w \leftarrow w - \eta \cdot \left(\frac{\partial E}{\partial w} \right)$$

Here:

- w: the weight in the network
- η: learning rate (a small positive number)
- E: error function

- $\partial E/\partial w$: partial derivative of error with respect to weight

We will predict the best weight by using a simple linear model → y1 = w · x
1st data point ⇒ x = 1 and y = 2.1. So, the prediction is → y1 = 0 · 1 = 0
2nd data point ⇒ y1 = 0.21 · 2 = 0.42
3rd data point ⇒ y1 = 0.706 · 3 = 2.118
This is the set formula to calculate MSE:

$$E = \frac{1}{2n}\sum_{n}(y - \text{y1})^2$$

1st data point ⇒ y = 2.1 and y1 = 0. So, the MSE will be: $\frac{1}{2}(2.1-0)^2 = \frac{1}{2}(4.41) = 2.205$
2nd data point ⇒ $\frac{1}{2}(2.9-0.42)^2 = \frac{1}{2}(6.1504) = 3.075$
3rd data point ⇒ $\frac{1}{2}(4.2-2.118)^2 = \frac{1}{2}(4.33)$
The formula to compute the derivative of the error with respect to w:

$$\frac{dE}{dw} = -x(y - \text{y1})$$

1st data point ⇒ $-1(2.1-0) = -2.1$
2nd data point ⇒ $-2(2.9-0.42) = -2(2.48) = -4.96$
3rd data point ⇒ $-3(4.2-2.118) = -3(2.082) = -6.246$
The formula for this is:

$$w = w - \eta \cdot \left(\frac{dE}{dw}\right)$$

1st data point ⇒ $0 - 0.1 \cdot (-2.1) = 0.21$ ⇒ meaning the new weight is 0.21
2nd data point ⇒ $0.21 - 0.1 \cdot (-4.96) = 0.21 + 0.496 = 0.706$ ⇒ meaning the new weight is 0.706
3rd data point ⇒ $0.706 - 0.1 \cdot (-6.246) = 0.706 + 0.6246 = 1.3306$ ⇒ Final new weight is 1.33
The update rule in gradient descent is:

$$w = w - \eta \cdot \left(\frac{dE}{dw}\right)$$

Here, dE/dw, is the derivative of the error function.
So, the derivative controls the direction

- if dE/dw ¿ 0 then the weight is too high, so we decrease it
- If dE/dw ¡ 0 then the weight is too low, so we increase it.

It tells us which way the error is sloping and we move in the opposite direction to reduce the error
The size of the derivative controls how big the step is.

- A large derivative means we are from the optimal weight, so you take a big step
- A small derivative means we are close to the minimum, so you take a small step

Example based on x = 2:

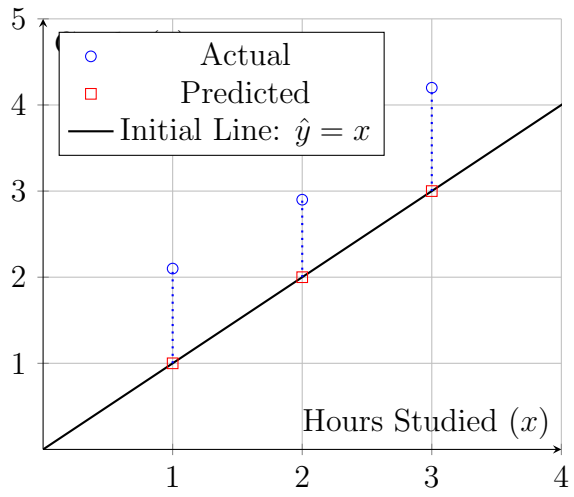
- Prediction was 0.42
- Error was 2.48
- Derivative was -4.96
- Update (w) = 0.706

This shows that:

- The derivative was negative → weight increased
- The derivate is large → weight changed a lot (0.21 → 0.706)

9. Visualizations

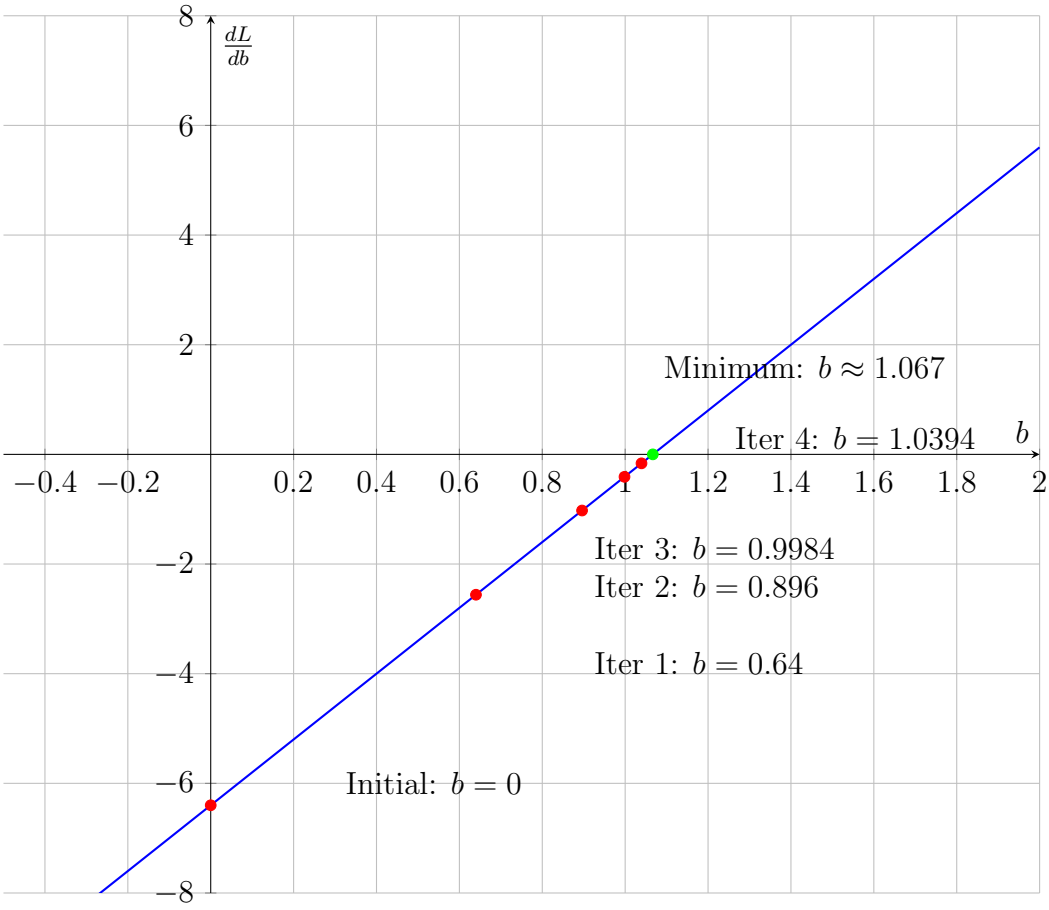
This plot shows:
- Blue circles: actual grades.
- Red squares: predicted grades with $b = 0, m = 1$.
- Black line: initial line $\hat{y} = x$.
- Dotted lines: errors between actual and predicted grades.



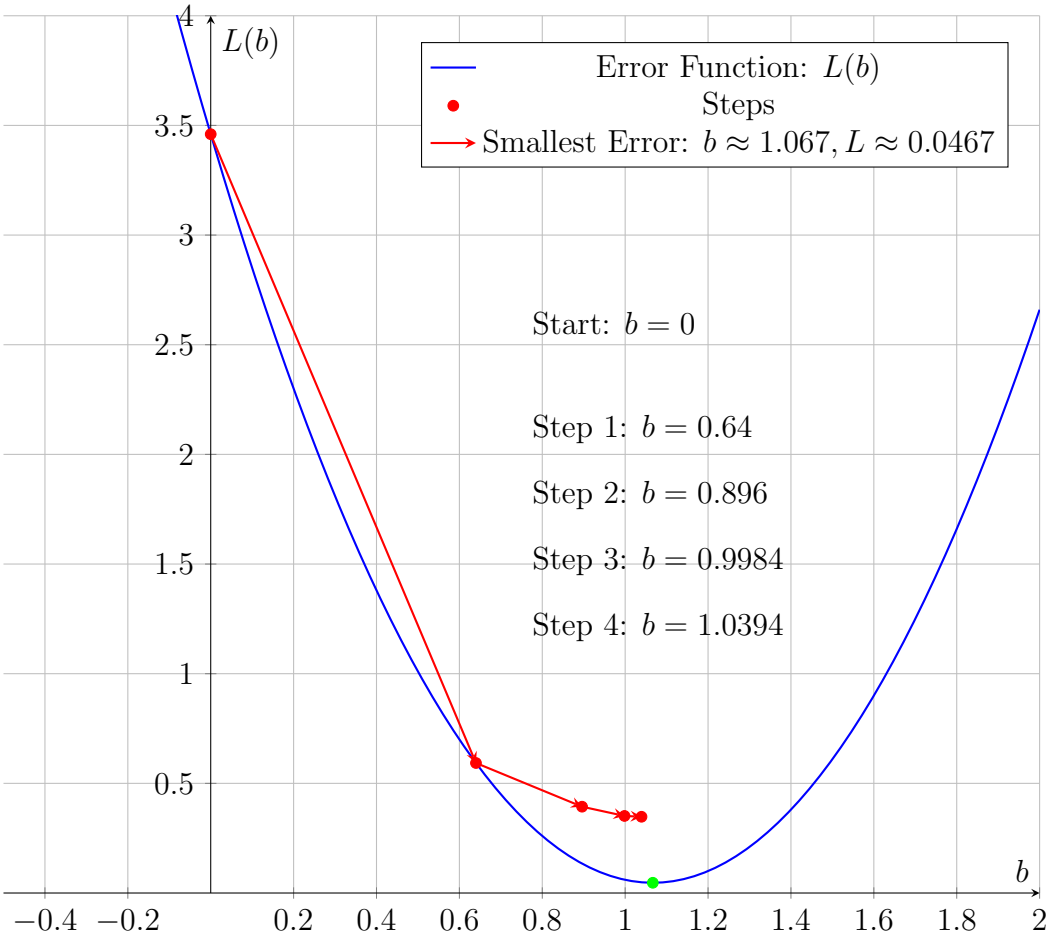
This plot shows the gradient:

$$\frac{dL}{db} = -6.4 + 6b$$

It shows how the gradient changes with b.
A negative gradient means increase b.
A positive gradient means decrease b.
The gradient is zero at $b \approx 1.0667$, where the error is smallest (0.0467).



This plot shows the error function $L(b) = 3b^2 - 6.4b + 3.46$.
Red dots show b values at each step.
Arrows show how b moves toward the smallest error.



10. Optional Slope Calculation

To find the best m, use:

$$m = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$

For our data, $m \approx 1.05$.

11. Final Results

x	y	Prediction (y1 = 1.33)	Error (y - y1)
1	2.1	$1.33 \cdot 1 = 1.33$	0.77
2	2.9	$1.33 \cdot 2 = 2.66$	0.24
3	4.2	$1.33 \cdot 3 = 3.99$	0.21

- The final weight after one round of learning was, w = 1.33. This means the following:
- The model has learned that for each additional of hour of study, the grade increases by 1.33
- So, the model is predicting $\rightarrow$ y1 = $1.33 \cdot$ x
- The predictions are reasonably close, but there is still some error.

- We can calculate the total mean squared error to get more insight into this.

Total mean squared error (TMSE) formula is as follows:

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

For our example, the formula is as follows:

$$\frac{1}{3}\sum_{3}(y_i - \hat{y}_i)^2 = \frac{1}{3}(0.77^2 + 0.24^2 + 0.21^2) = \frac{1}{3}(0.593 + 0.058 + 0.044) = \frac{1}{3}(0.695) \approx 0.231$$

- This shows that the error is getting smaller but not yet minimal.
- A good model will have a TMSE value closer to or has a value of zero.
- In this example we only ran it once, so we might get an even better TMSE value when we run it more.

12. Summary

- Gradient descent finds the best model settings to reduce errors.
- The gradient shows how to change b to lower the error.
- The learning rate controls step size.
- Keep updating until the error is small.