

ChatGPT ▾

[Upgrade to Go](#) ×[↑ Share](#)

...

gradient descent

Here's a **simple, clear example** of how **Gradient Descent** is used to compute weights for **Linear Regression**.

✅ Example: Compute weights using Gradient Descent

We want to fit a line:

$$\hat{y} = wx + b$$

Given dataset:

| x | y |
|---|---|
| 1 | 2 |
| 2 | 3 |
| 3 | 4 |

Step 1: Initialize parameters

Let:

```
ini  
  
w = 0  
b = 0  
learning_rate = 0.01
```

Step 2: Define formulas

Prediction

$$\hat{y}_i = wx_i + b$$

Loss (MSE)

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

Gradients

$$\frac{\partial L}{\partial w} = \frac{2}{n} \sum_i x_i (\hat{y}_i - y_i)$$

$$\frac{\partial L}{\partial b} = \frac{2}{n} \sum_i (\hat{y}_i - y_i)$$

Step 3: Perform Gradient Descent Update

$$w := w - \alpha \frac{\partial L}{\partial w}$$

$$b := b - \alpha \frac{\partial L}{\partial b}$$

✓ Let's do 1 full iteration (epoch)

Dataset:

ini

 Copy code

$x = [1, 2, 3]$

$y = [2, 3, 4]$

$n = 3$

Predictions (initial $w=0$, $b=0$)

 Copy code

$\hat{y} = [0, 0, 0]$

Errors ($\hat{y} - y$)

css

 Copy code

$[-2, -3, -4]$

Compute gradients

Gradient wrt w :

$$\frac{2}{3}[(1)(-2) + (2)(-3) + (3)(-4)]$$

Calculate inside:

bash

 Copy code

```
(1)(-2) = -2  
(2)(-3) = -6  
(3)(-4) = -12  
sum = -20
```

$$\frac{2}{3}(-20) = -13.33$$

Gradient wrt b:

$$\frac{2}{3}(-2 - 3 - 4) = \frac{2}{3}(-9) = -6$$

Step 4: Update w and b

Learning rate = 0.01

Update w:

$$w = 0 - 0.01(-13.33) = 0.1333$$

Update b:

$$b = 0 - 0.01(-6) = 0.06$$

Updated parameters after 1 Gradient Descent step:

ini

 Copy code $w = 0.1333$ $b = 0.06$

Want full code version?

I can give you:

☒ Python from scratch

or

☒ Numpy version

or

☒ Vectorized version

Just tell me which one you want!



decision tress entropy example

Here is a **simple, clear example** of how **entropy** is calculated in **Decision Trees (ID3 / C4.5)**.



Decision Tree — Entropy Example

Suppose you have the following dataset:

| Weather | Play? |
|----------|-------|
| Sunny | No |
| Sunny | No |
| Overcast | Yes |
| Rainy | Yes |
| Rainy | Yes |
| Rainy | No |

We want to calculate the **entropy of the target column ("Play?")**.

Step 1: Count class frequencies

Target = **Play?**

```
ini
```

[Copy code](#)

```
Yes = 3
```

```
No = 3
```

```
Total = 6
```

Step 2: Entropy formula

$$Entropy = - \sum p_i \log_2(p_i)$$

Where p_i = class probability.

Step 3: Compute probabilities

$$p(Yes) = 3/6 = 0.5$$

$$p(No) = 3/6 = 0.5$$

Step 4: Apply entropy formula

$$Entropy = -[0.5 \log_2(0.5) + 0.5 \log_2(0.5)]$$

We know:

$$\log_2(0.5) = -1$$

So:

$$Entropy = -[0.5(-1) + 0.5(-1)]$$

$$Entropy = -[-0.5 - 0.5]$$

$$Entropy = 1.0$$

✓ Entropy of the entire dataset = 1.0

This means:

- Dataset is **perfectly impure**
- All classes are **equally mixed**
- Decision Tree must **split** to reduce uncertainty

★ Want a split example (e.g., entropy of "Weather" attribute)?

I can show you **entropy of Sunny / Overcast / Rainy** and also the **Information Gain** for the split.



yes please

Great — let's compute **attribute entropies** and the **Information Gain** for splitting on **Weather**.

We use the same dataset:

| Weather | Play? |
|----------|-------|
| Sunny | No |
| Sunny | No |
| Overcast | Yes |
| Rainy | Yes |
| Rainy | Yes |
| Rainy | No |

We already computed:

$$\text{Entropy}(\text{parent}) = 1.0$$

Step 1: Split the data by Weather

1) Sunny

Rows:

| Sunny | No |
|-------|----|
| Sunny | No |
| Sunny | No |

Counts:

ini

 Copy code

Yes = 0

No = 2

Entropy:

$$\text{Entropy}(\text{Sunny}) = -[0 \log_2(0) + 1 \log_2(1)] = 0$$

(Reason: all samples are "No" → perfectly pure)


2) Overcast

Overcast

Yes

Counts:

ini

 Copy code

Yes = 1

No = 0

Entropy:

$$\text{Entropy}(\text{Overcast}) = 0$$

(Reason: perfectly pure)

3) Rainy

| Rainy | Yes |
|-------|-----|
| Rainy | Yes |
| Rainy | No |

Counts:

```
ini
```

[Copy code](#)

```
Yes = 2
No  = 1
Total = 3
```

Probabilities:

$$p(Yes) = 2/3, \quad p(No) = 1/3$$

Entropy:

$$Entropy(Rainy) = - \left(\frac{2}{3} \log_2 \frac{2}{3} + \frac{1}{3} \log_2 \frac{1}{3} \right)$$

Compute:

$$\frac{2}{3} \log_2 \frac{2}{3} \approx 0.6667 \times (-0.585) = -0.39$$

$$\frac{1}{3} \log_2 \frac{1}{3} \approx 0.3333 \times (-1.585) = -0.53$$

$$Entropy(Rainy) = -(-0.39 - 0.53) = 0.92$$



Step 2: Compute Weighted Entropy After Split

Weights = proportion of rows.

makefile

Copy code

Sunny: $2/6 = 0.33$

Overcast: $1/6 = 0.17$

Rainy: $3/6 = 0.50$

Weighted entropy:

$$Entropy_{after} = 0.33(0) + 0.17(0) + 0.50(0.92)$$

$$Entropy_{after} = 0.46$$



Step 3: Information Gain

$$IG(Weather) = Entropy(parent) - Entropy_{after}$$

$$IG = 1.0 - 0.46 = 0.54$$



Final Results

| Category | Entropy |
|----------|---------|
| Sunny | 0 |
| Overcast | 0 |

| Category | Entropy |
|-------------------------|-------------|
| Rainy | 0.92 |
| Weighted total | 0.46 |
| Information Gain | 0.54 |

Interpretation

- **Weather is a very good splitting attribute**, because:
 - It reduces entropy from **1.0** → **0.46**
 - Information Gain = **0.54** (high)
 - Sunny + Overcast become perfectly pure.

If you want, I can now show:

✓ Information Gain for other attributes (Temperature, Windy, etc.)

or

✓ Build the **entire decision tree** step-by-step.

Just tell me!