Deep Learning Basics

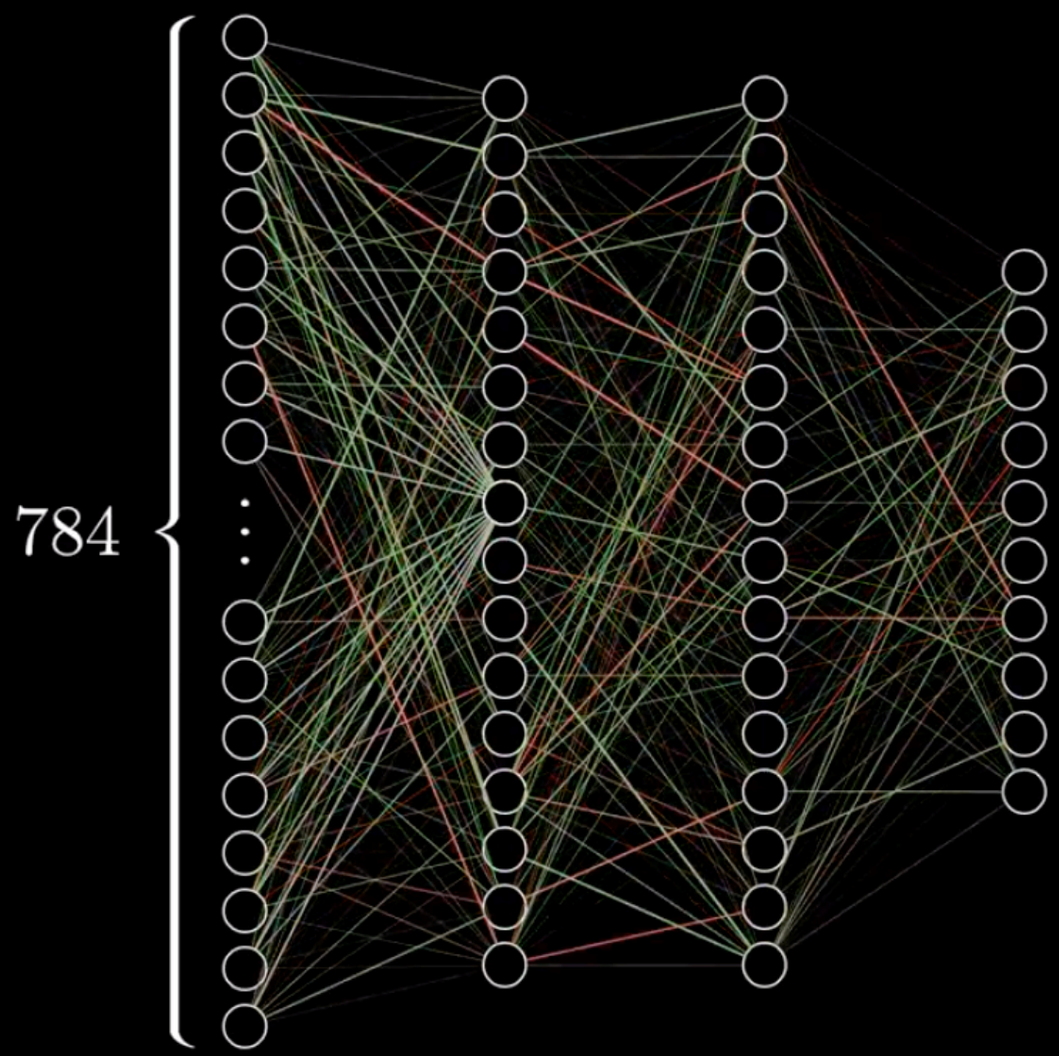
3: Training Neural Networks

Francis Steen and Xinyu You
Red Hen Lab
August 2019

A. Loss function

How well does the neural network predict

Learning: Finding the right weights and biases



$$784 \times 16 + 16 \times 16 + 16 \times 10$$
weights

$$16 + 16 + 10$$
biases

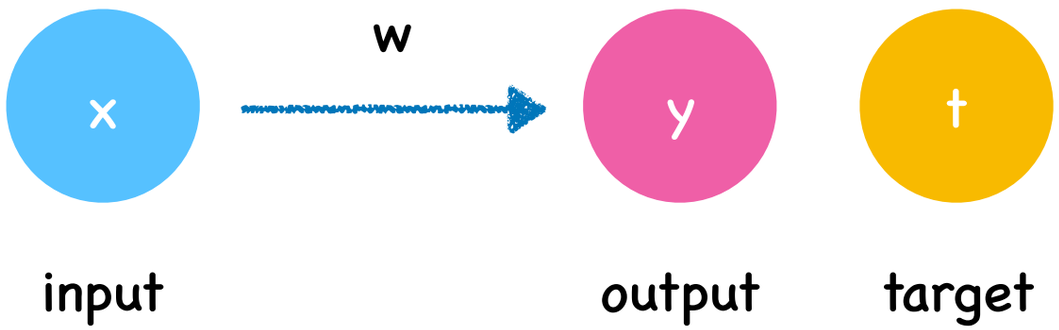
$$13{,}002$$

Learning $\rightarrow$ Finding the right weights and biases

Loss function

How **different** between the **target** value and predict **output** value? We use a loss function to measure it.

A simple neural network example:

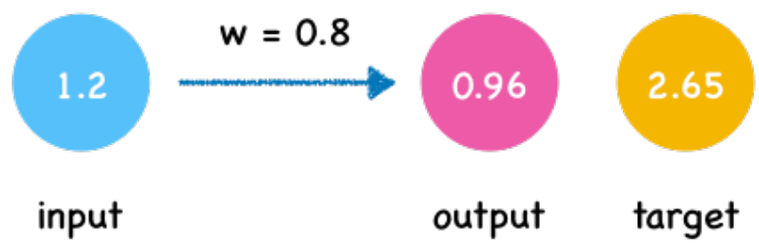


input x	target t
1.2	2.65
3.4	7.50
2.8	6.17
7.3	16.09

Loss function

1. First, we randomly set the value of w to 0.8.

```
w = 0.8
def neural_netwok_predict(x, w):
    y = w * x
    return y
```



2. Get the predict output.

3. Calculate the difference between the actual value y and the target value t

raw error = actual value - target value

```
def raw_error(y, t):
    return y - t
```

The Problem with raw error

Suppose we have a set of data.

Node	Actual value by network	Target training values
1	0.6	0.5
2	0.6	0.7
3	0.7	0.85
4	0.95	0.8

The Problem with raw error

If we calculate raw error to measure the performance of neural network.

Node	Actual value by network	Target training values	Error
1	0.6	0.5	0.1
2	0.6	0.7	-0.1
3	0.7	0.85	-0.15
4	0.95	0.8	0.15

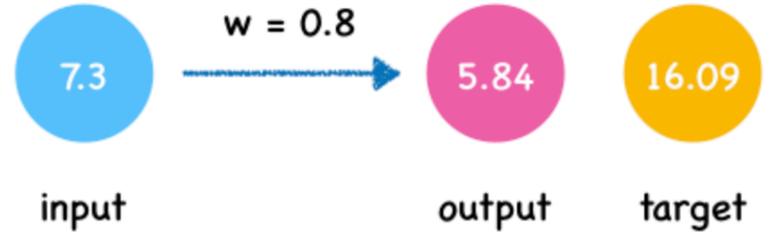
$$L = 0.1 + (-0.1) + (-0.15) + (0.15) = 0$$

How to solve this?

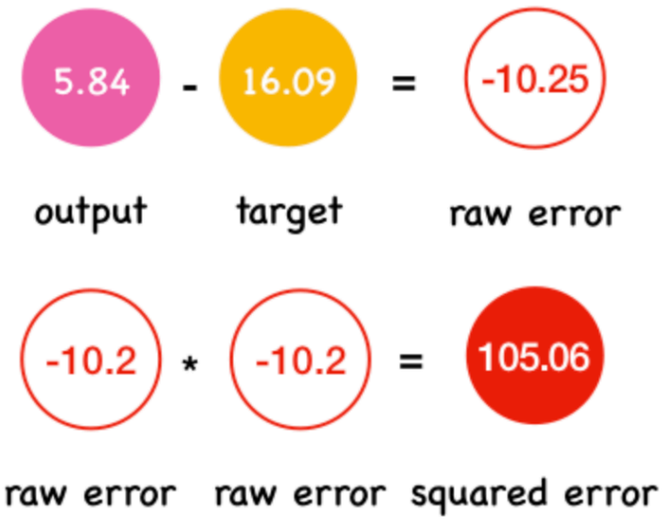
Mean Squared Error

The form of mathematical expressions of MAE:

$$L = \frac{1}{n}\sum_{i=1}^{n}(y_i - t_i)^2$$



Calculate the difference between the actual value y and the target value t using MSE:




```python
def squared_error(y, t):
    error = row_error(y, t)
    return error**2
```

```python
def mean_squared_error(errors):
    MSE = 1/len(errors) * sum(errors)
    return MSE
```

The result of MSE

Node	Actual value by network	Target training values	Error
1	0.6	0.5	0.1
2	0.6	0.7	-0.1
3	0.7	0.85	-0.15
4	0.95	0.8	0.15

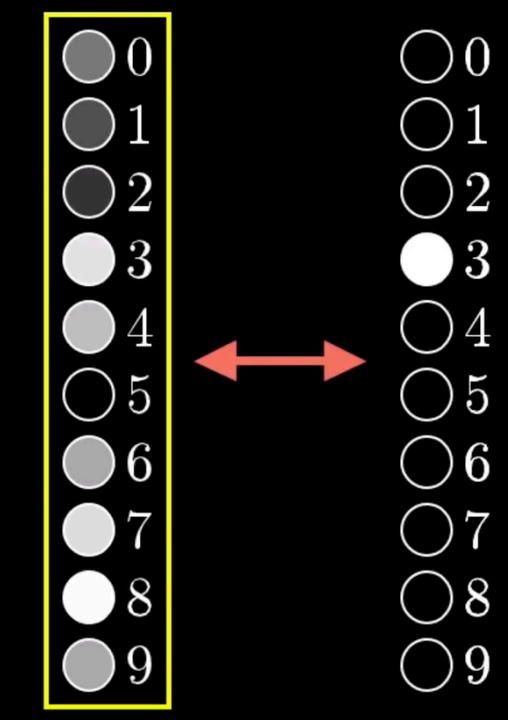
$$L = ((0.1)^2 + (-0.1)^2 + (-0.15)^2 + (0.15)^2)/4 = 0.016$$

The Example

Cost of 

$$
\begin{aligned}
&(0.43 - 0.00)^2 + \\
&(0.28 - 0.00)^2 + \\
&(0.19 - 0.00)^2 + \\
&(0.88 - 1.00)^2 + \\
&(0.72 - 0.00)^2 + \\
&(0.01 - 0.00)^2 + \\
&(0.64 - 0.00)^2 + \\
&(0.86 - 0.00)^2 + \\
&(0.99 - 0.00)^2 + \\
&(0.63 - 0.00)^2
\end{aligned}
$$

What's the "cost" of this difference?



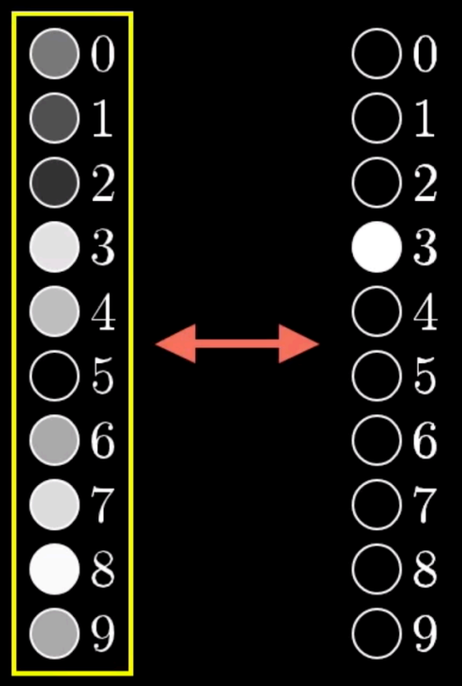
Utter trash

The Example

Cost of 

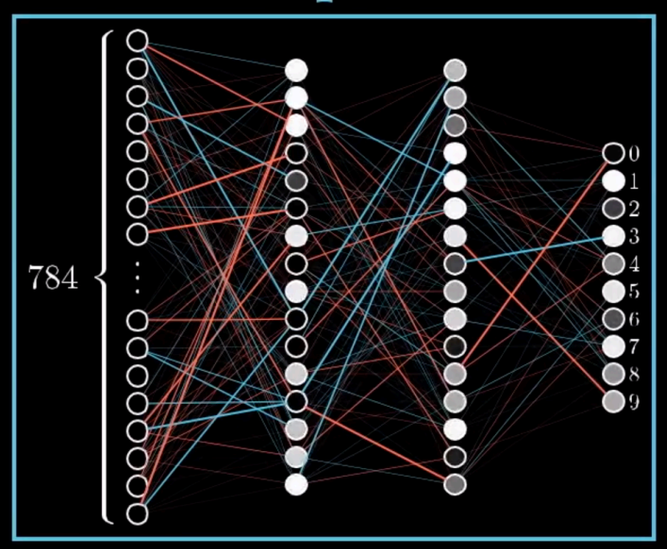
$$
\left.\begin{array}{rl}
0.1863 \leftarrow & (0.43 - 0.00)^2 + \\
0.0809 \leftarrow & (0.28 - 0.00)^2 + \\
0.0357 \leftarrow & (0.19 - 0.00)^2 + \\
0.0138 \leftarrow & (0.88 - 1.00)^2 + \\
0.5242 \leftarrow & (0.72 - 0.00)^2 + \\
0.0001 \leftarrow & (0.01 - 0.00)^2 + \\
0.4079 \leftarrow & (0.64 - 0.00)^2 + \\
0.7388 \leftarrow & (0.86 - 0.00)^2 + \\
0.9817 \leftarrow & (0.99 - 0.00)^2 + \\
0.3998 \leftarrow & (0.63 - 0.00)^2
\end{array}\right\} 3.37
$$

What's the "cost" of this difference?



Utter trash

The Example



Cost: 5.4

Cost function

Input: 13,002 weights/biases

Output: 1 number (the cost)

Parameters: Many, many, many training examples

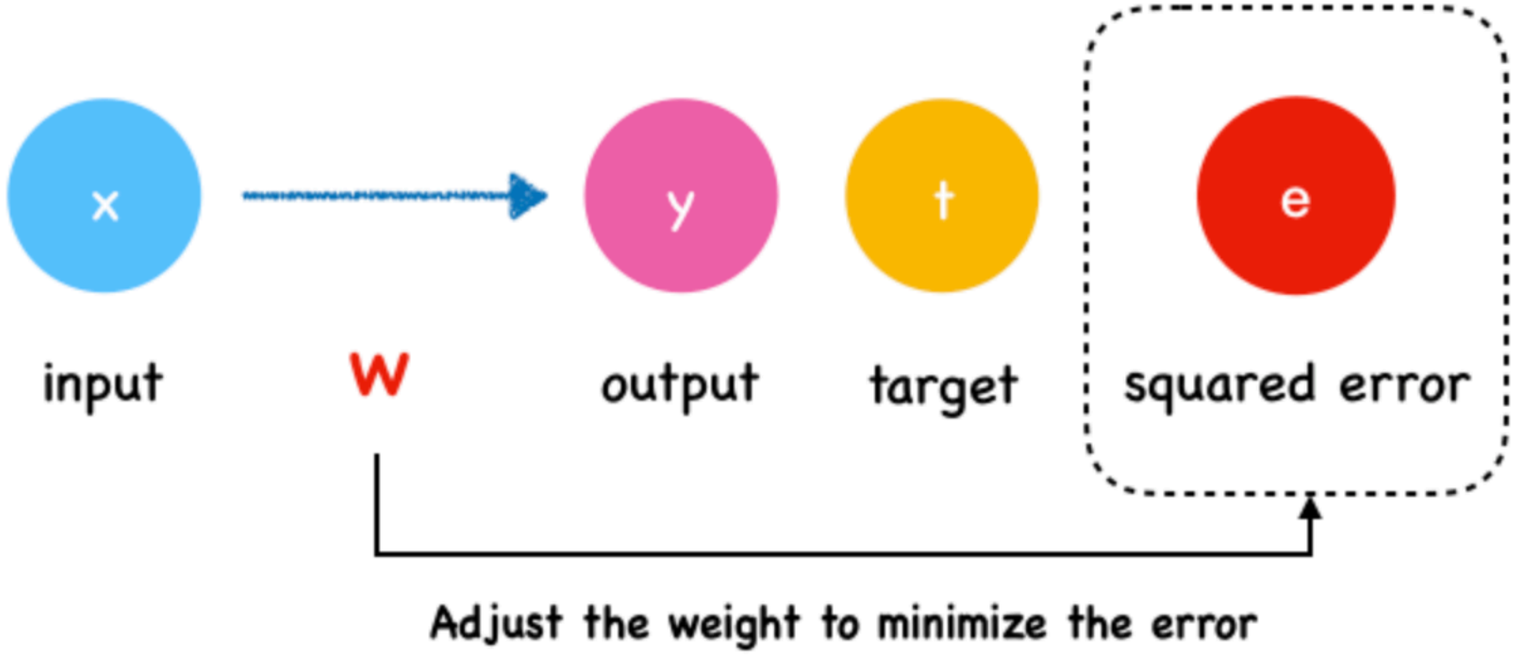
Hard to compute the result directly

A. Gradient Descent

Learning to minimize error

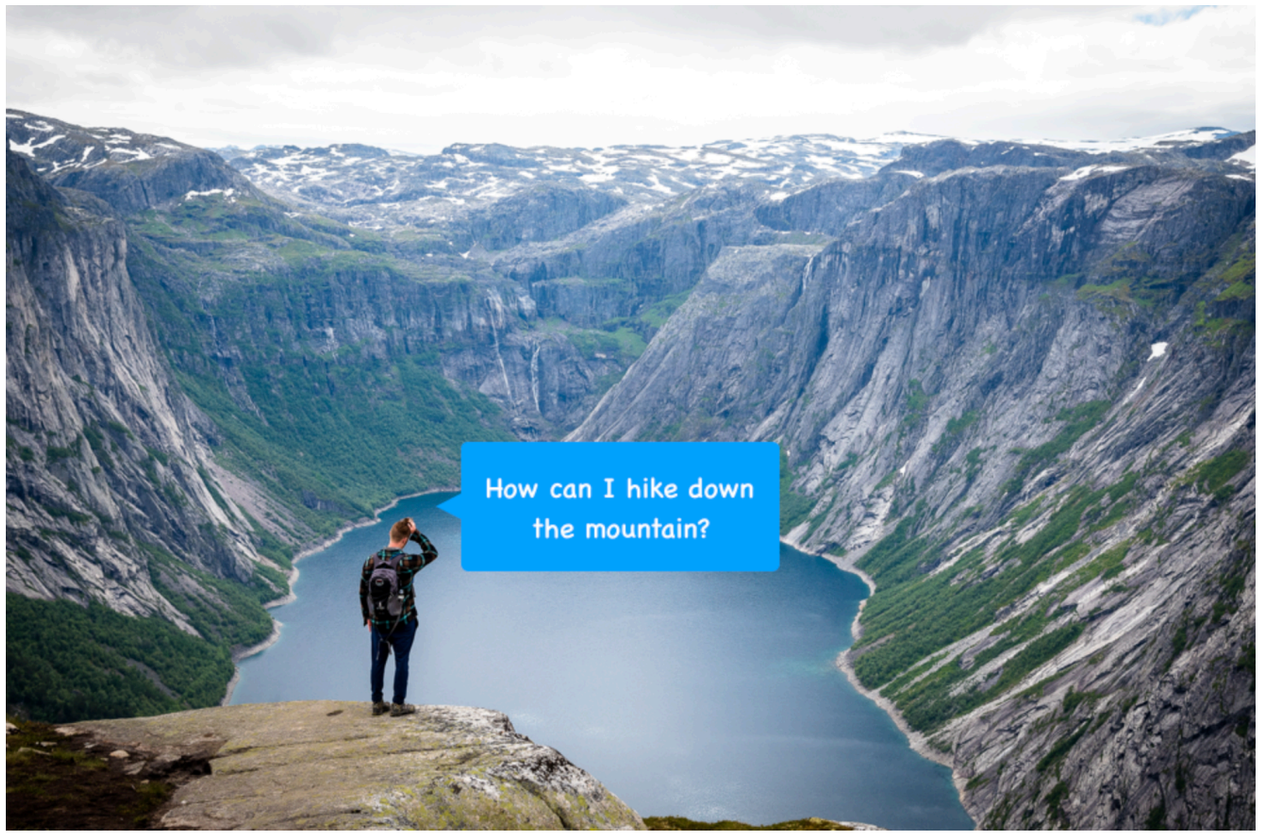
The Goal of neural network

The goal of neural network is adjust the weight to minimize the output of loss function.



What is Gradient Descent?

The classic mountaineering example



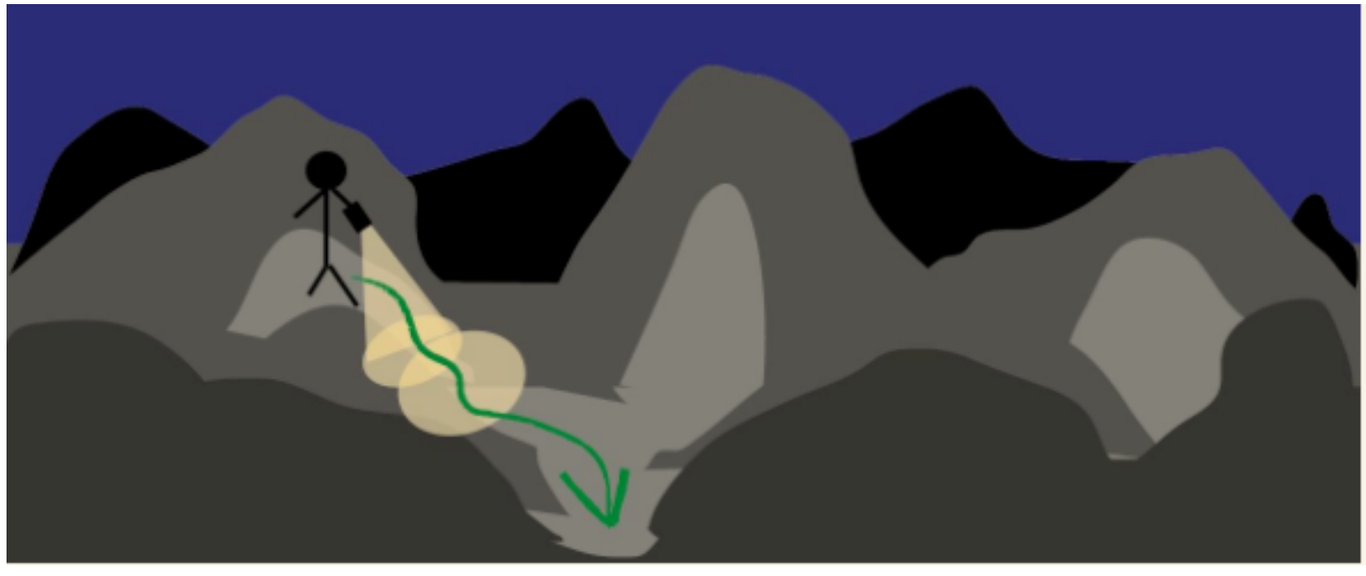
Suppose you are at the peek of a mountain and need to reach a lake which is in the valley of the mountain.

It's on dark nights and you have zero visibility to see where you are headed. The only thing you have is a torch to see a short distance.

So, what approach will you take to reach the lake? Take a few minutes to think about this.

What is Gradient Descent?

The classic mountaineering example



You may use the torch to look around the ground beneath your feet.

Then you can figure out where it is going downward and take small steps in that direction.

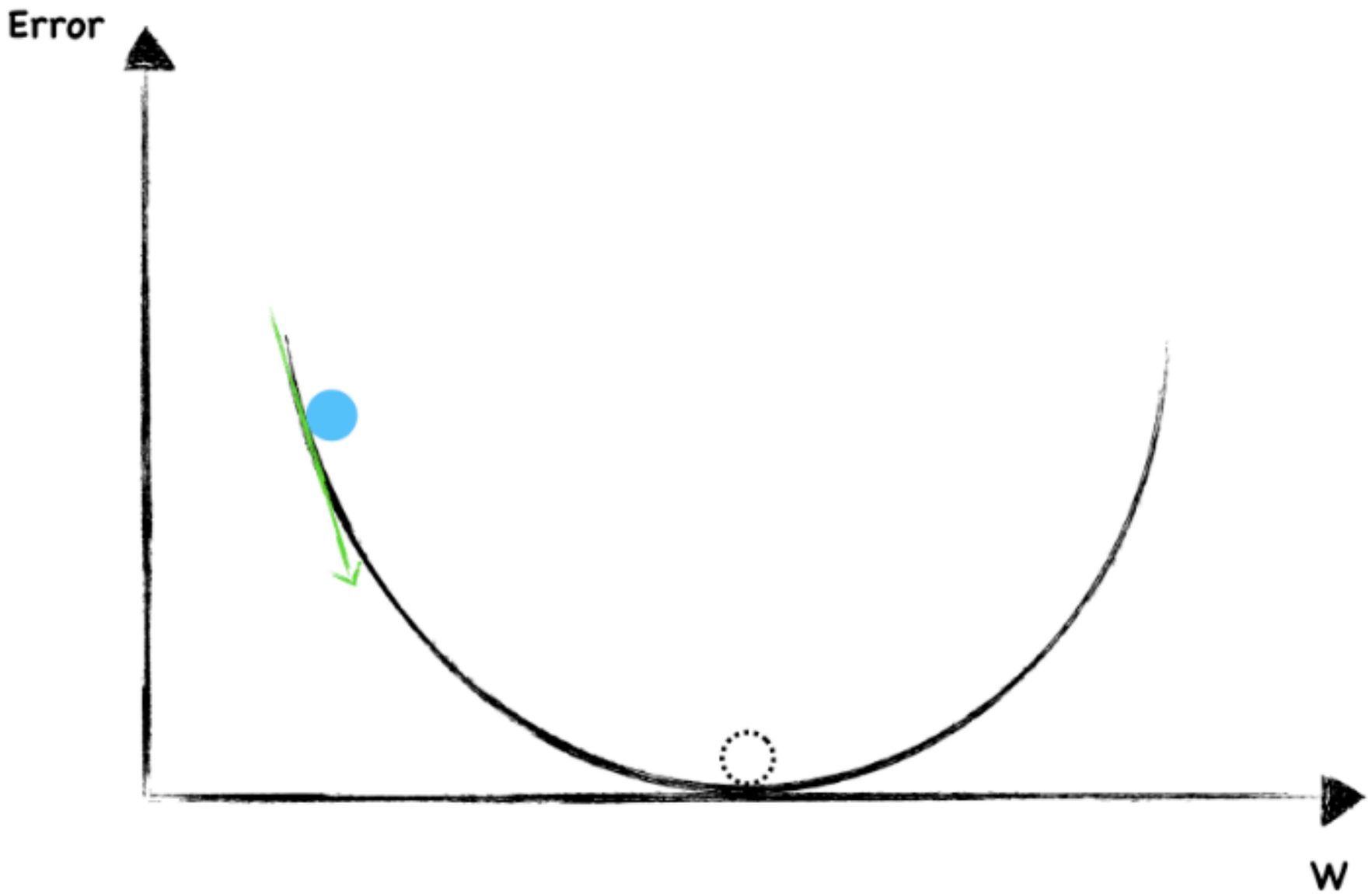
In this way, you keep following the path that is descending, step by step, you’re likely to reach the lake at the valley.

Simple Example

Here we still begin with a simple neural network.

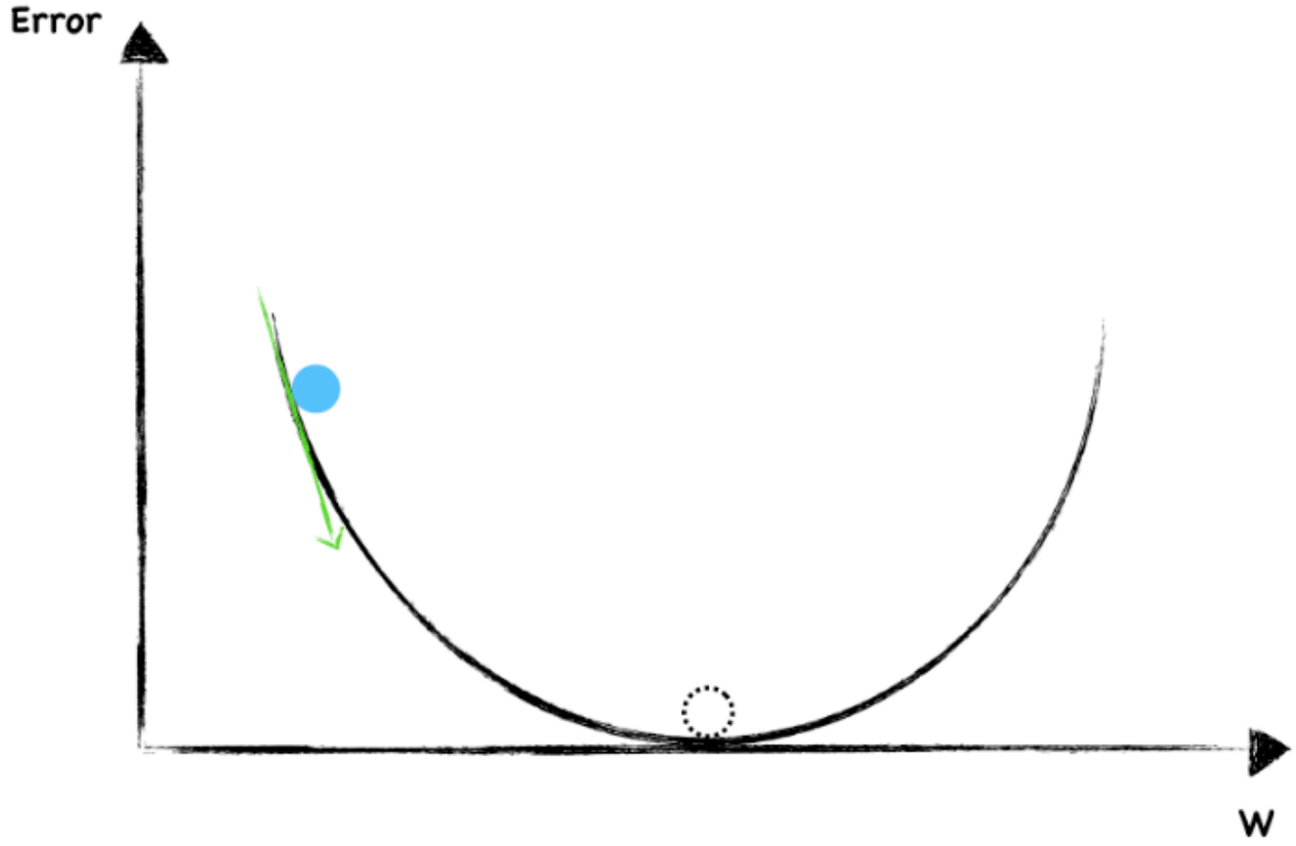
Simple Example

Say the following graph shows a function where y-axis is the error, we are going to find the right w to minimize it.



Simple Example

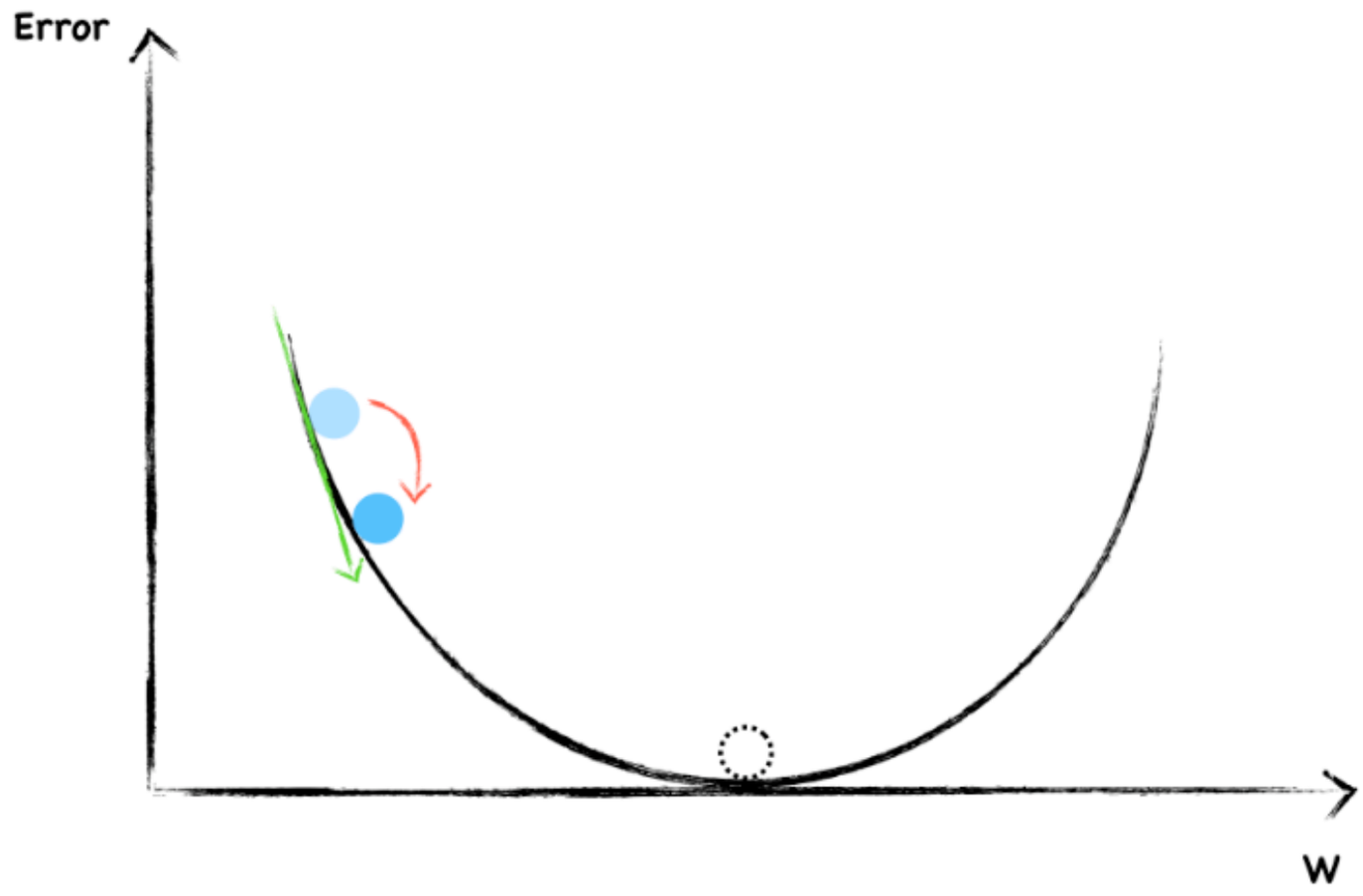
Say the following graph shows a function where y-axis is the error, we are going to find the right w to minimize it.



The dotted circle is where error equal to 0 and that's where we want to be as close as possible.

Simple Example

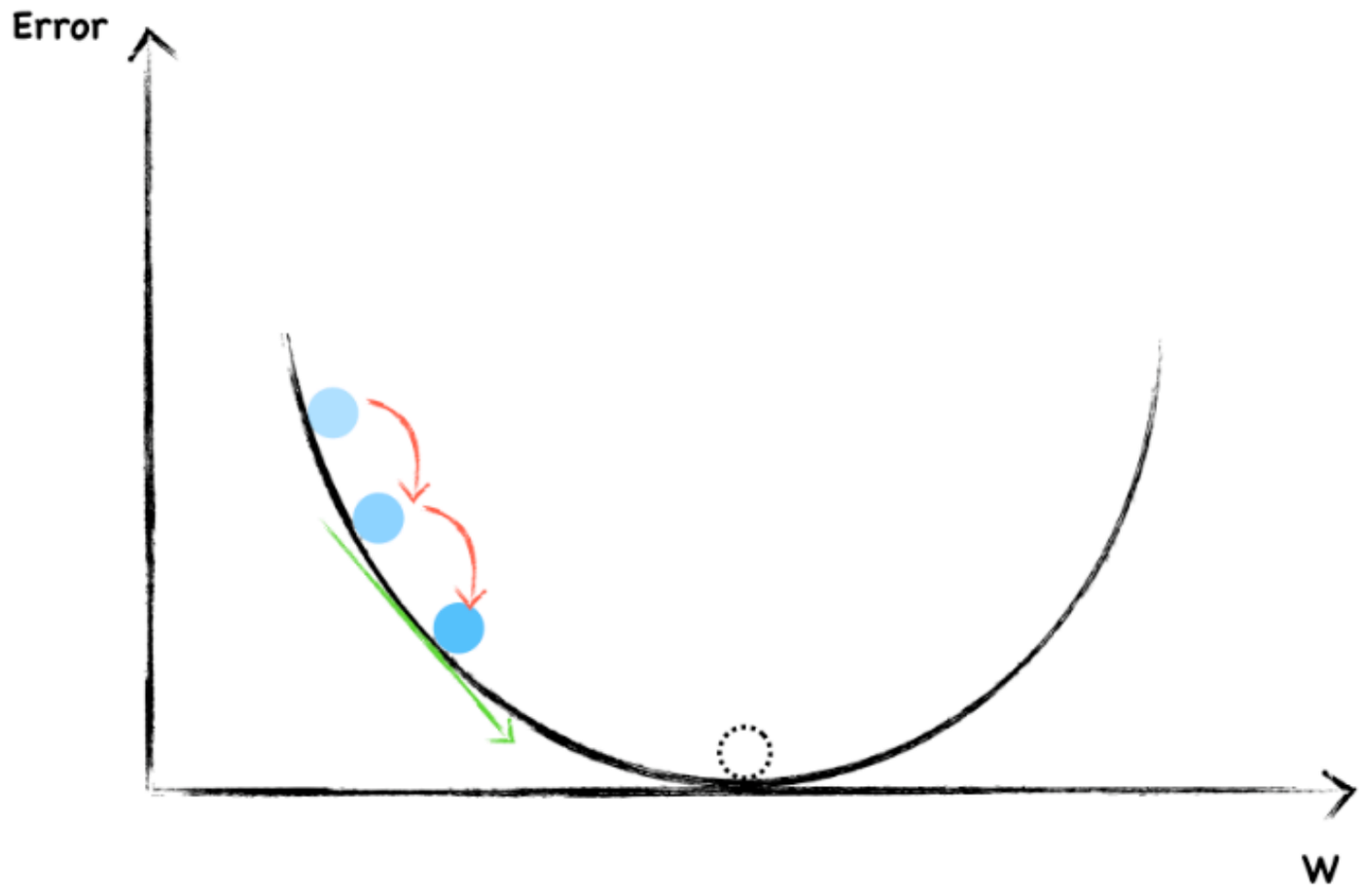
To start, we randomly choose a value of w. The blue dot represent current weight and error.



Now, let's look around to see which direction should we go. The slope help us to do that. This is a negative gradient, so we increase the w a little.

Simple Example

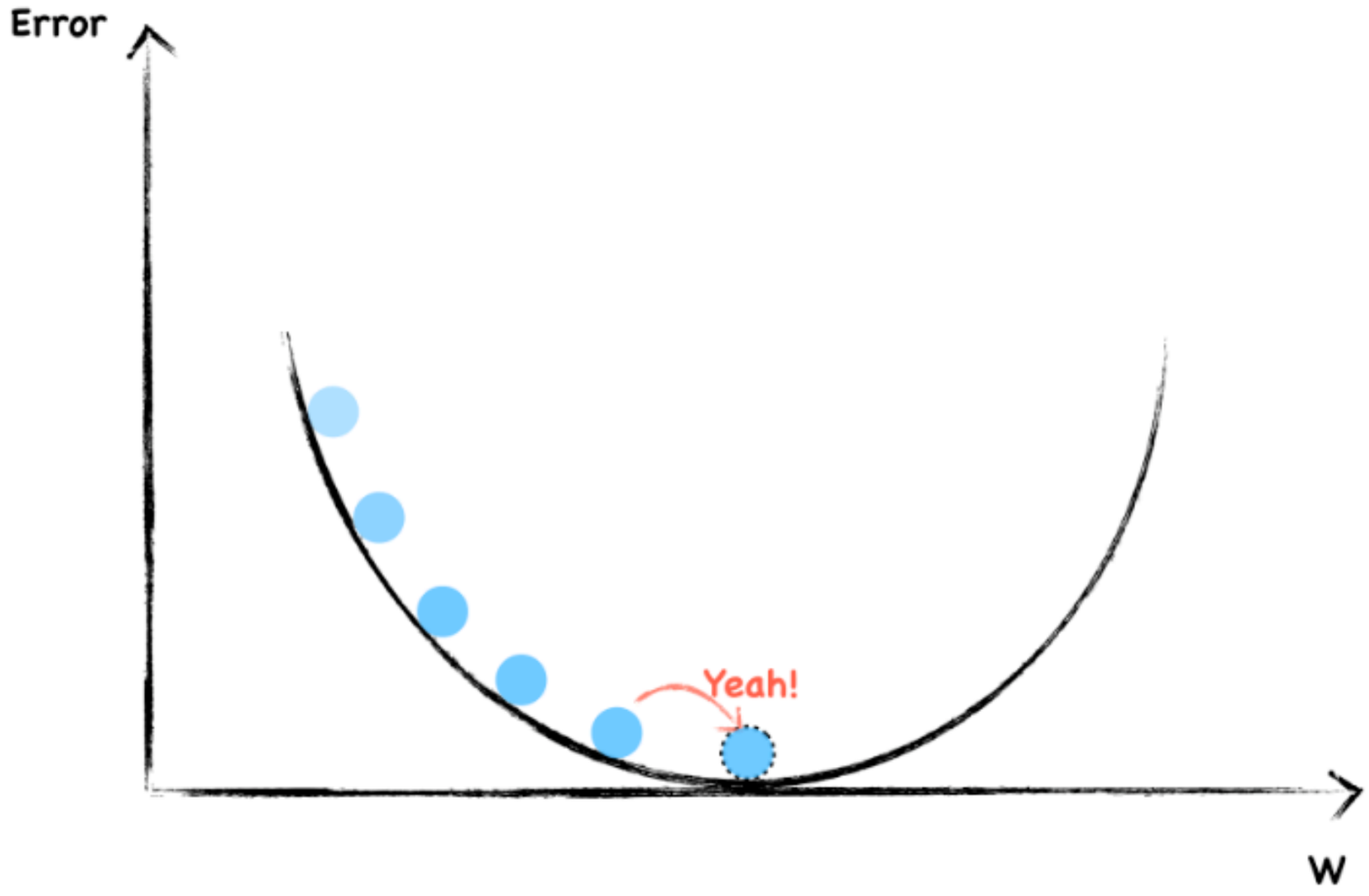
Again, this is a negative slope, we continue increase w.



Each time, we adjust the weight the same size step by step. In this way, if we're lucky, we're probably arrived at the bottom.

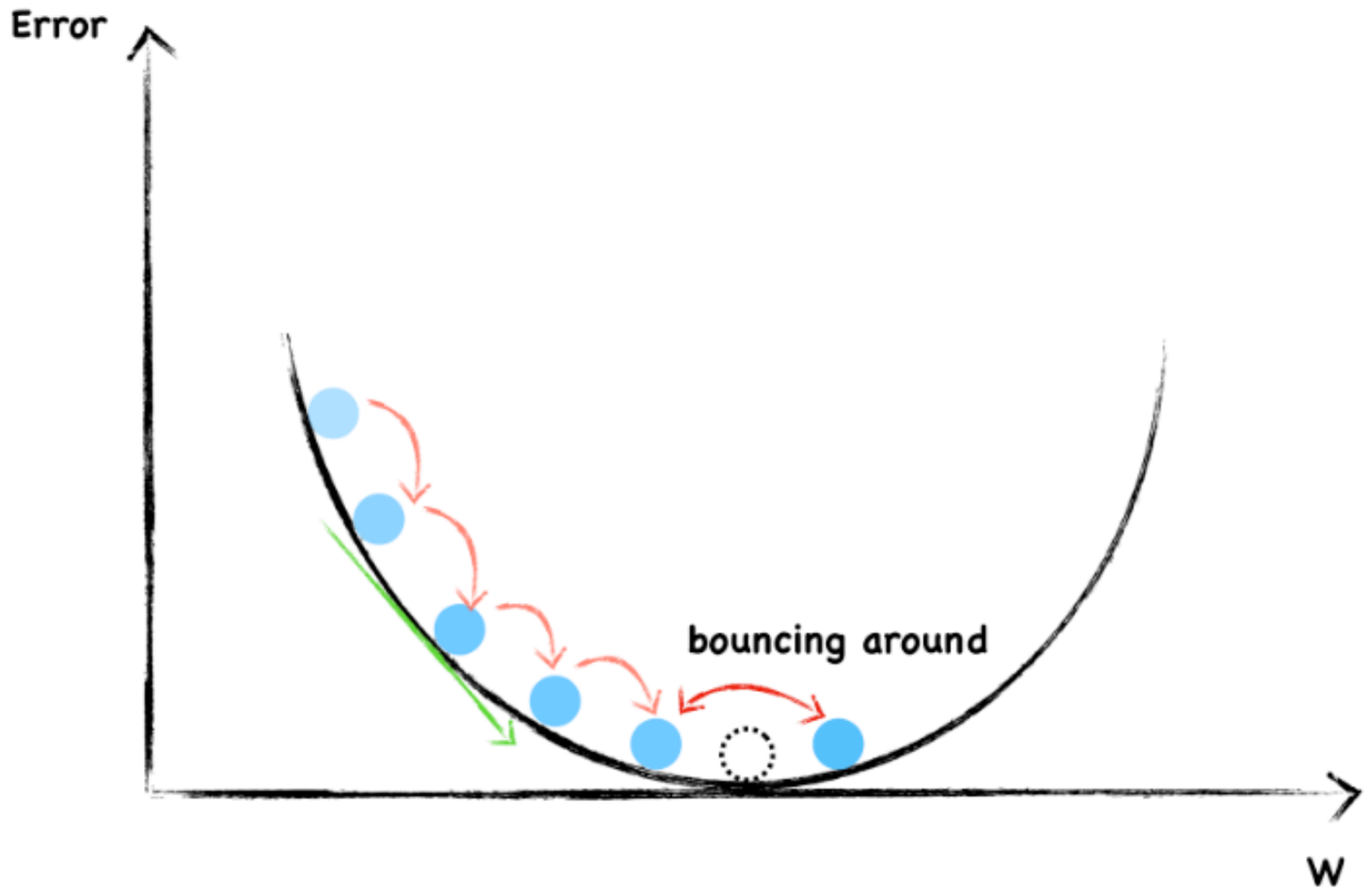
Simple Example

Each time, we adjust the weight the same size step by step. In this way, if we're lucky, we're probably arrived at the bottom.



Avoid Overshooting

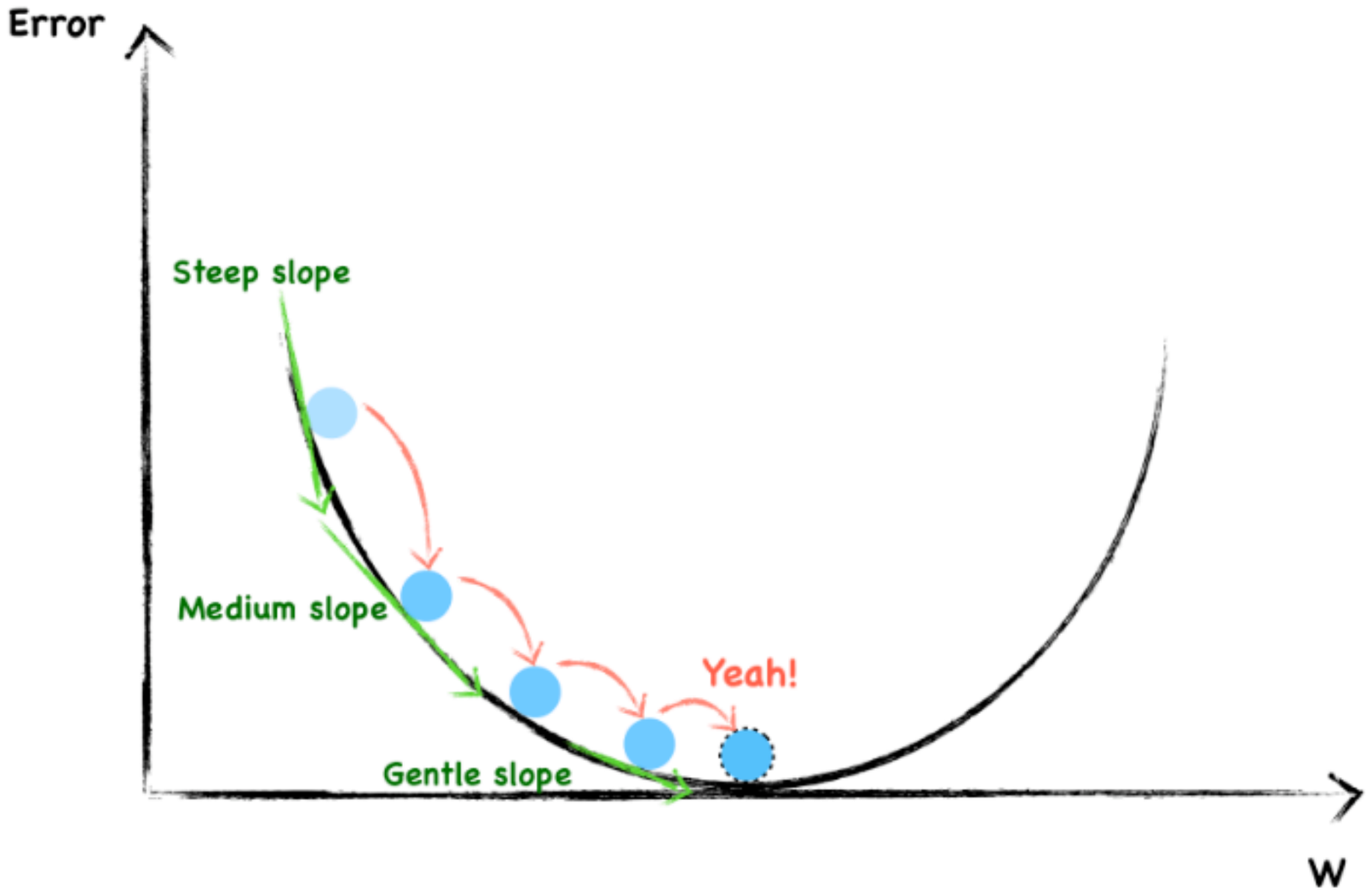
However, we are likely to overshooting the minimum and forever bouncing around it.



To avoid overshooting, what should we do?

Avoid Overshooting

Well, as you can see, in order not to overshooting through the minimum, we have to take smaller and smaller step towards the end.



Update Weight

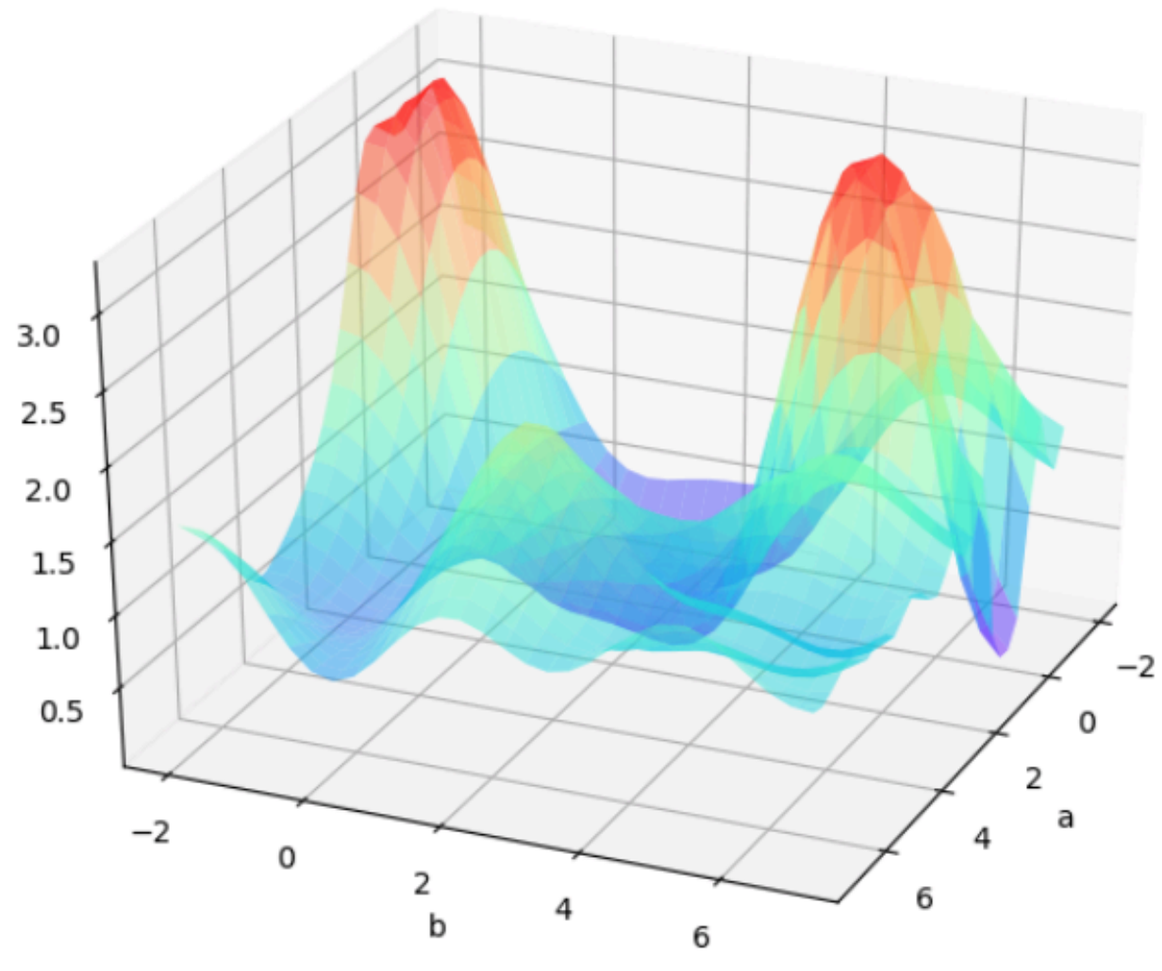
new weight = old weight - learning rate * slope

On the whole, we randomly choose a weight and then changed it in a direction opposite to the gradient.

Besides, as we get close to the local bottom, we change the step size according to how steep the slope is.

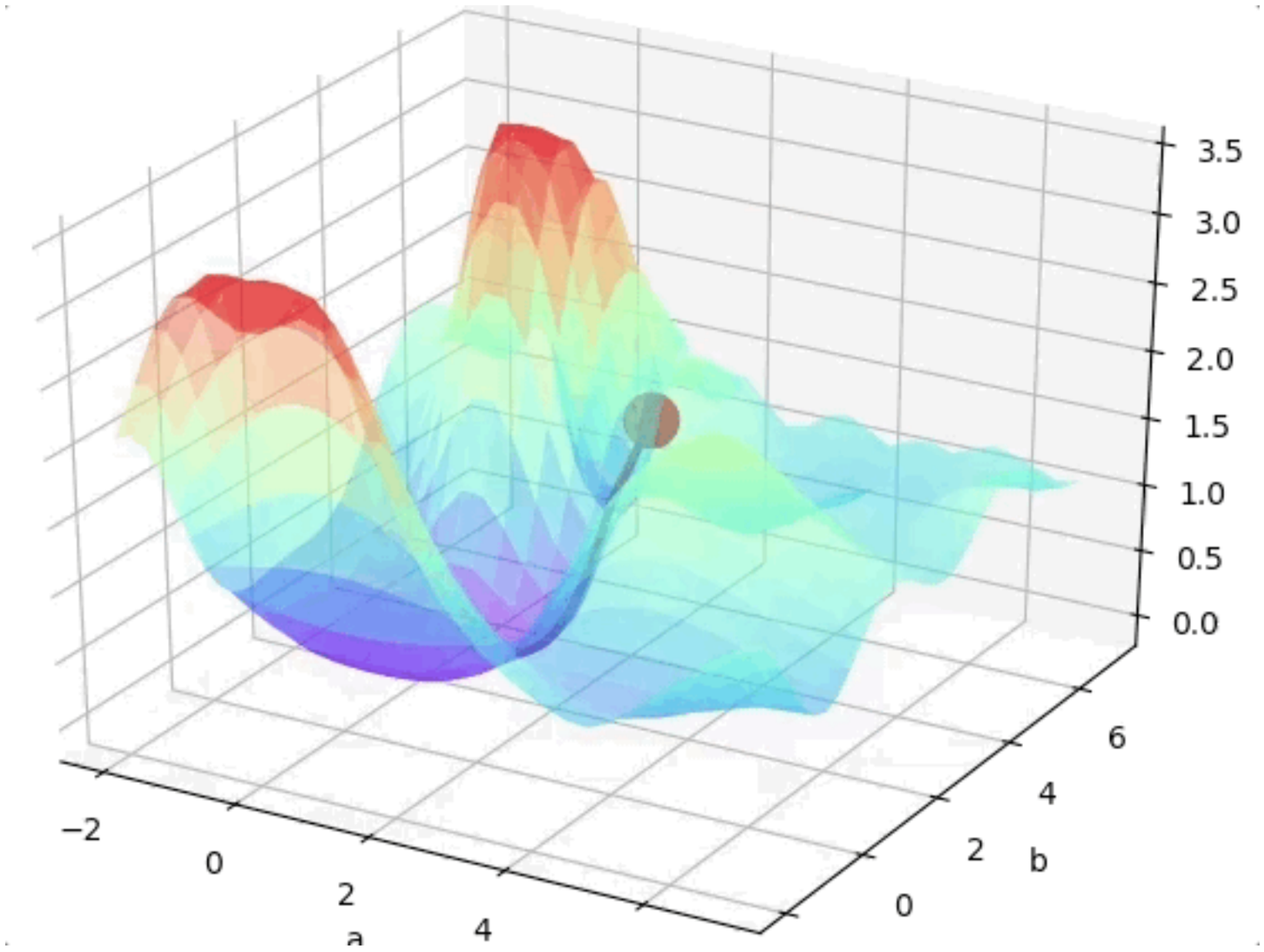
Challenges: Local Minima

For a two layers neural network with two parameters. The loss function may looks like:



Challenges: Local Minima

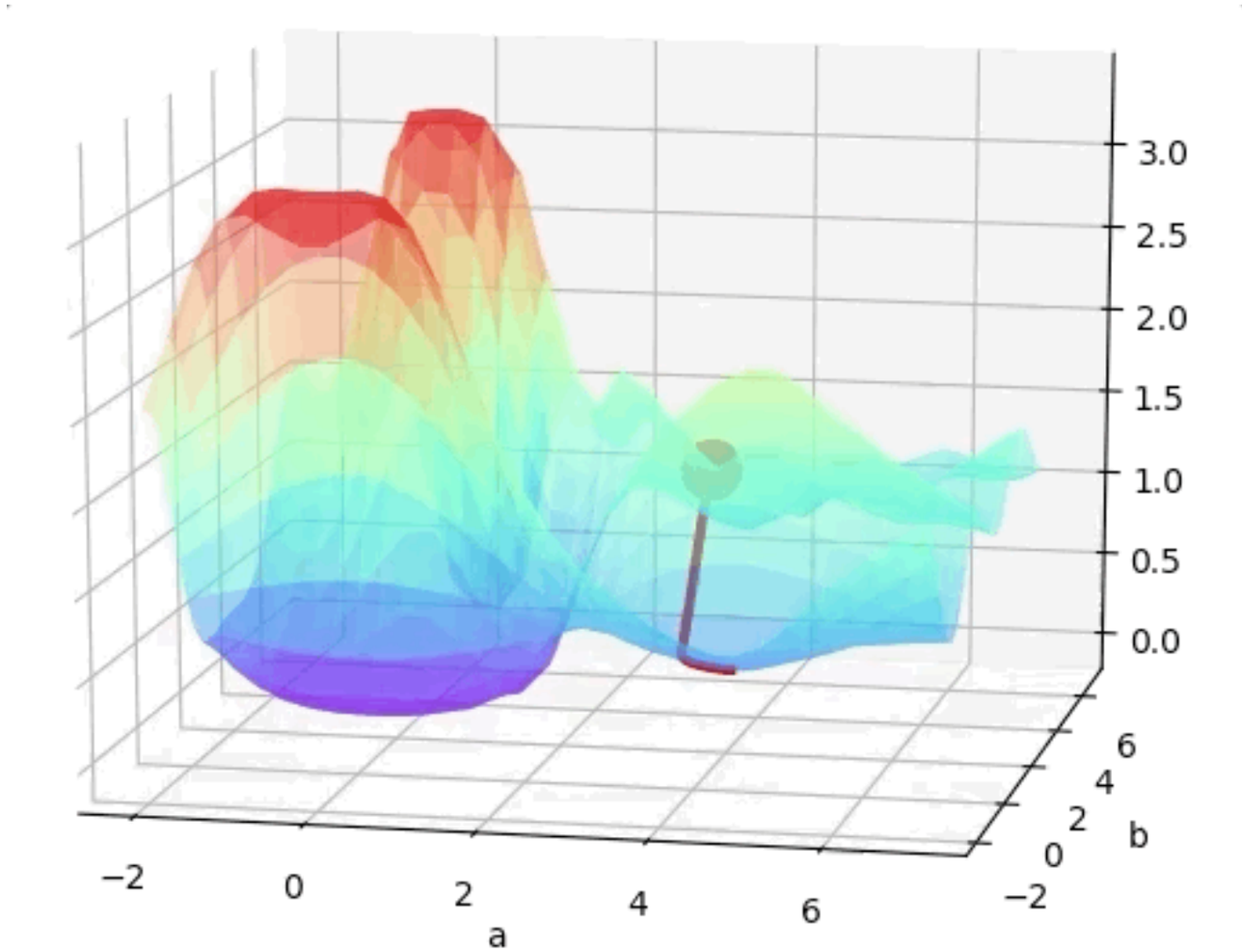
If we are lucky we may reach the global minimum from a random initial point.



Click to play.

Challenges: Local Minima

However, we may come across to a local minimum.



Click to play.

Challenges: Local Minima

One way to solve this problem is run the algorithms multiple times with different initial weights and then choose the the best minimum.