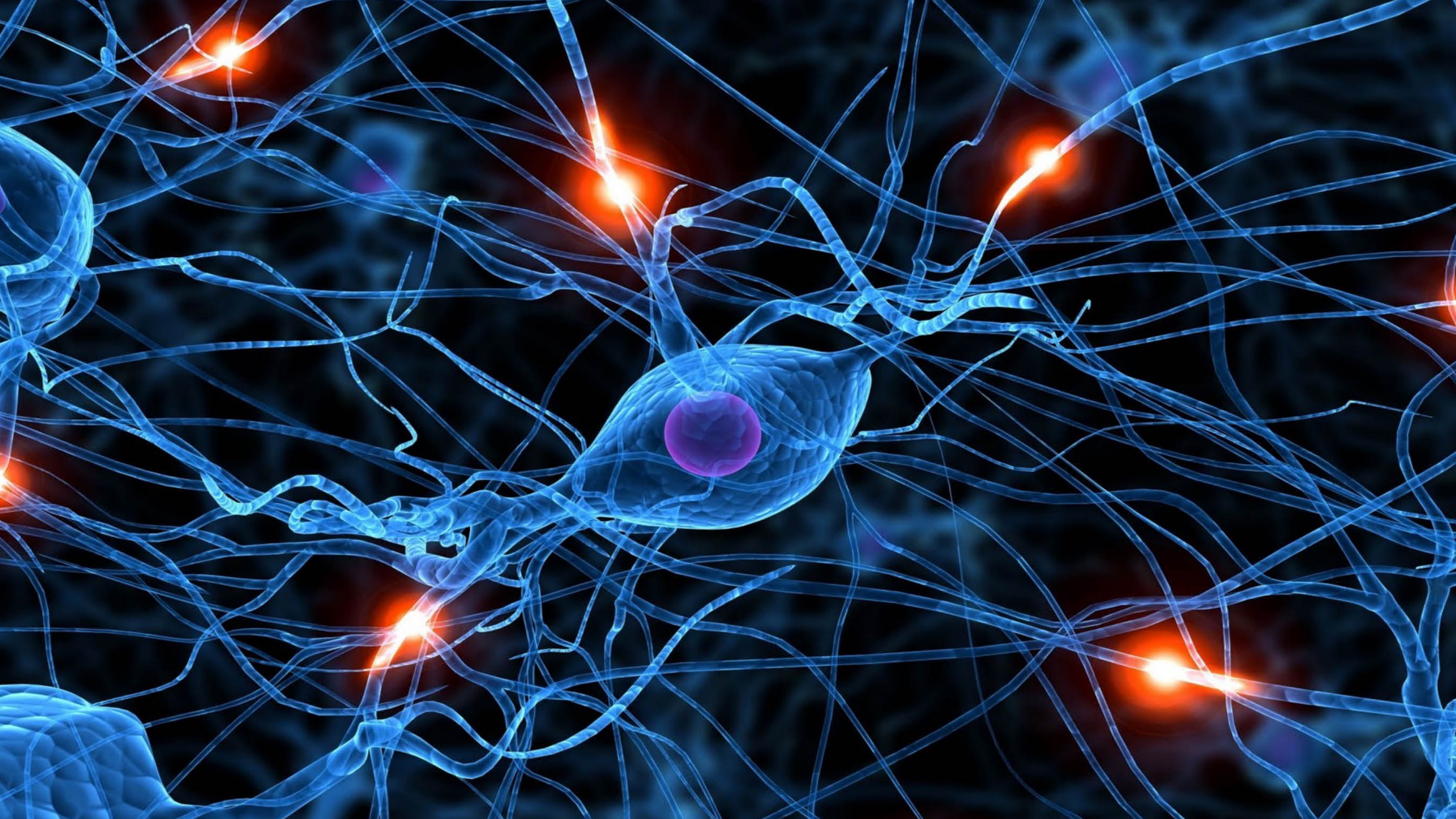
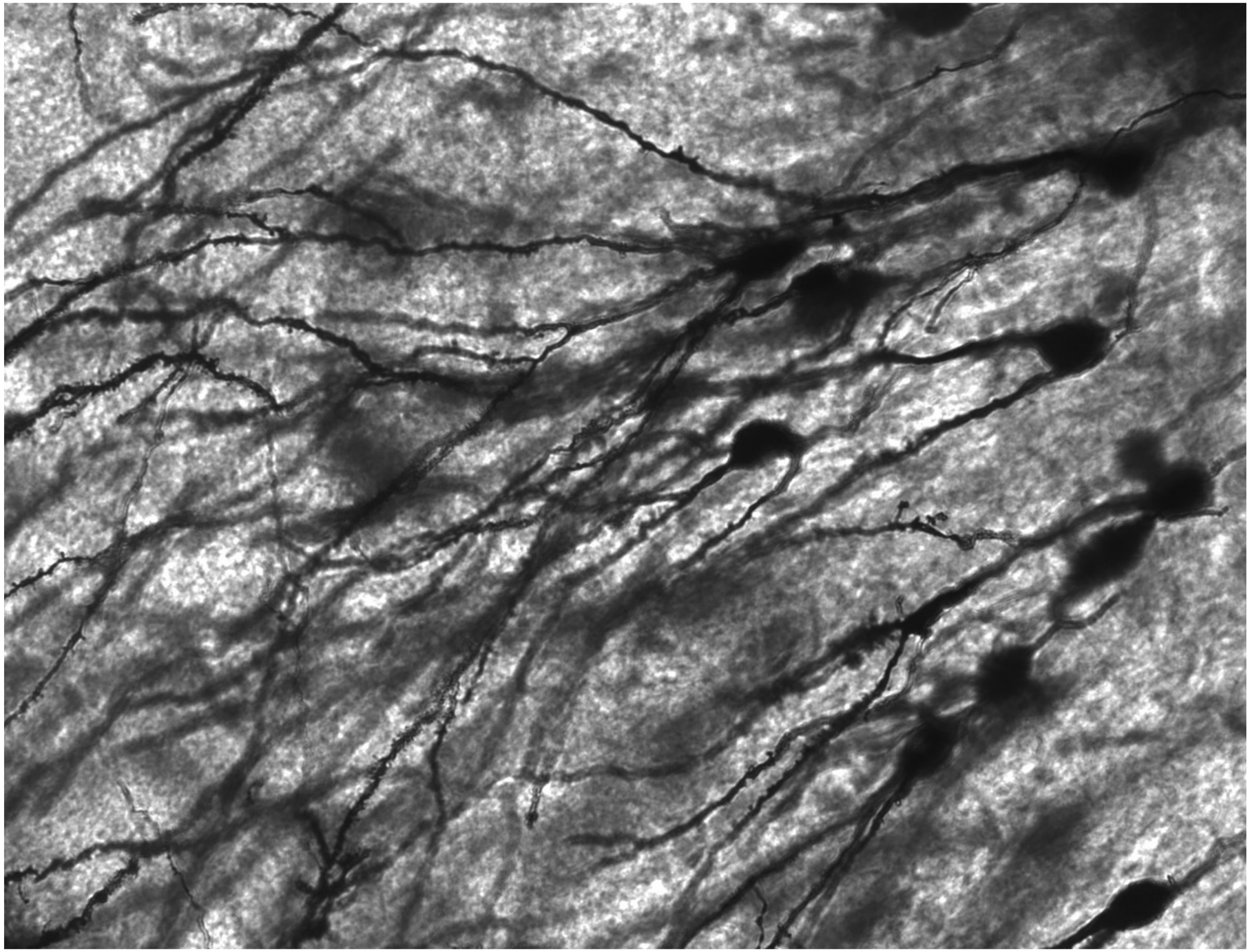
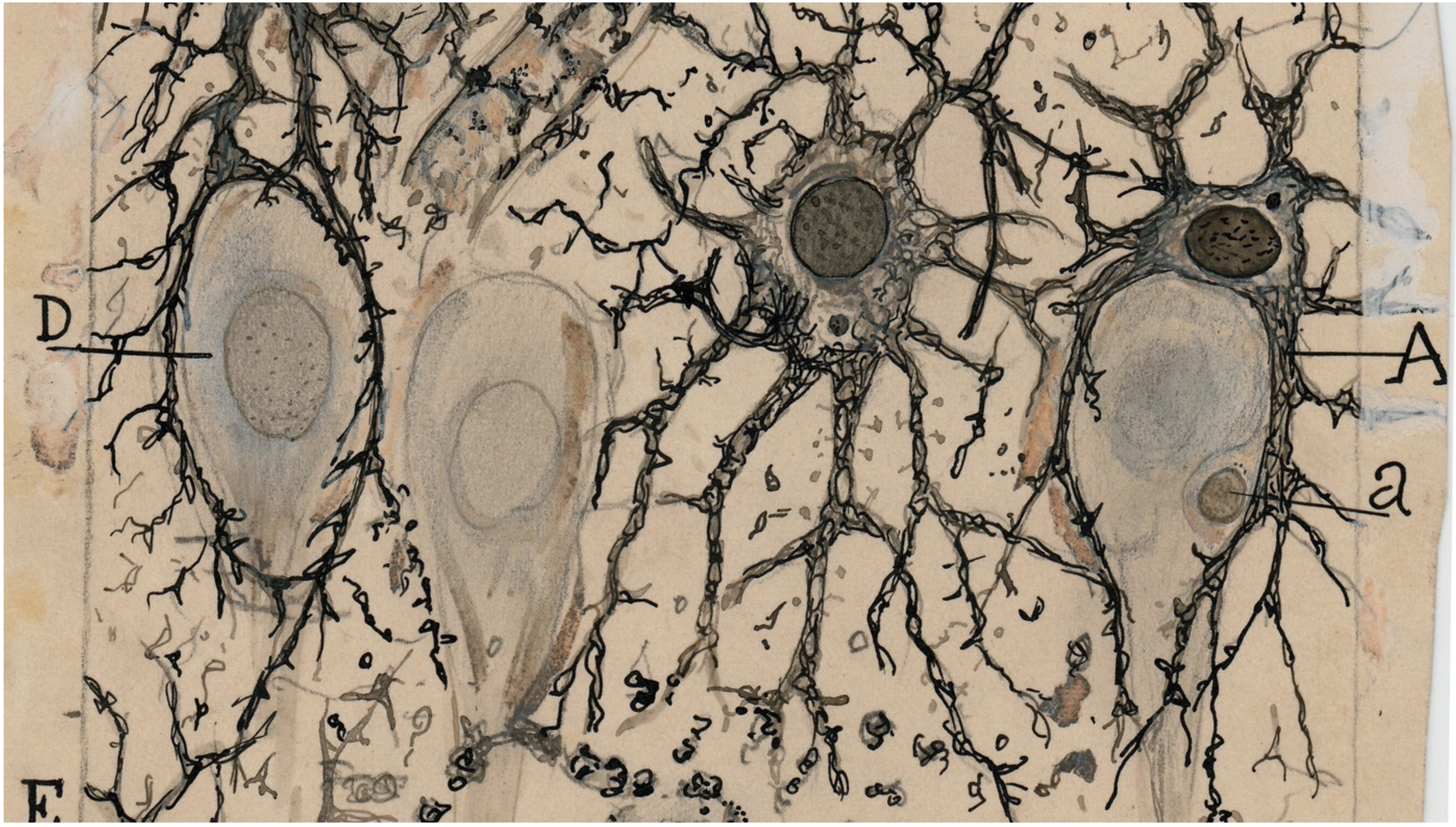
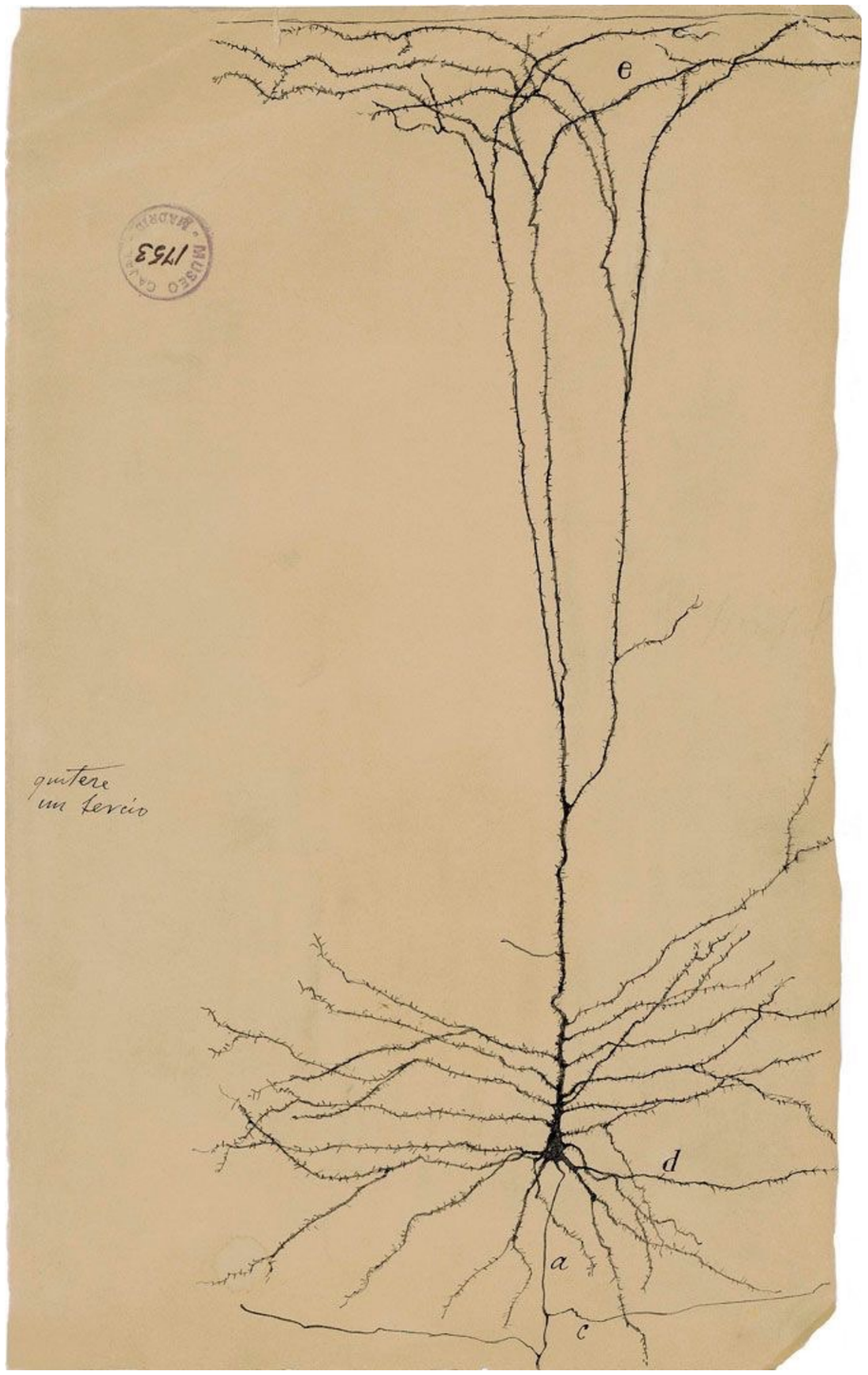


Neurons in a human hippocampus



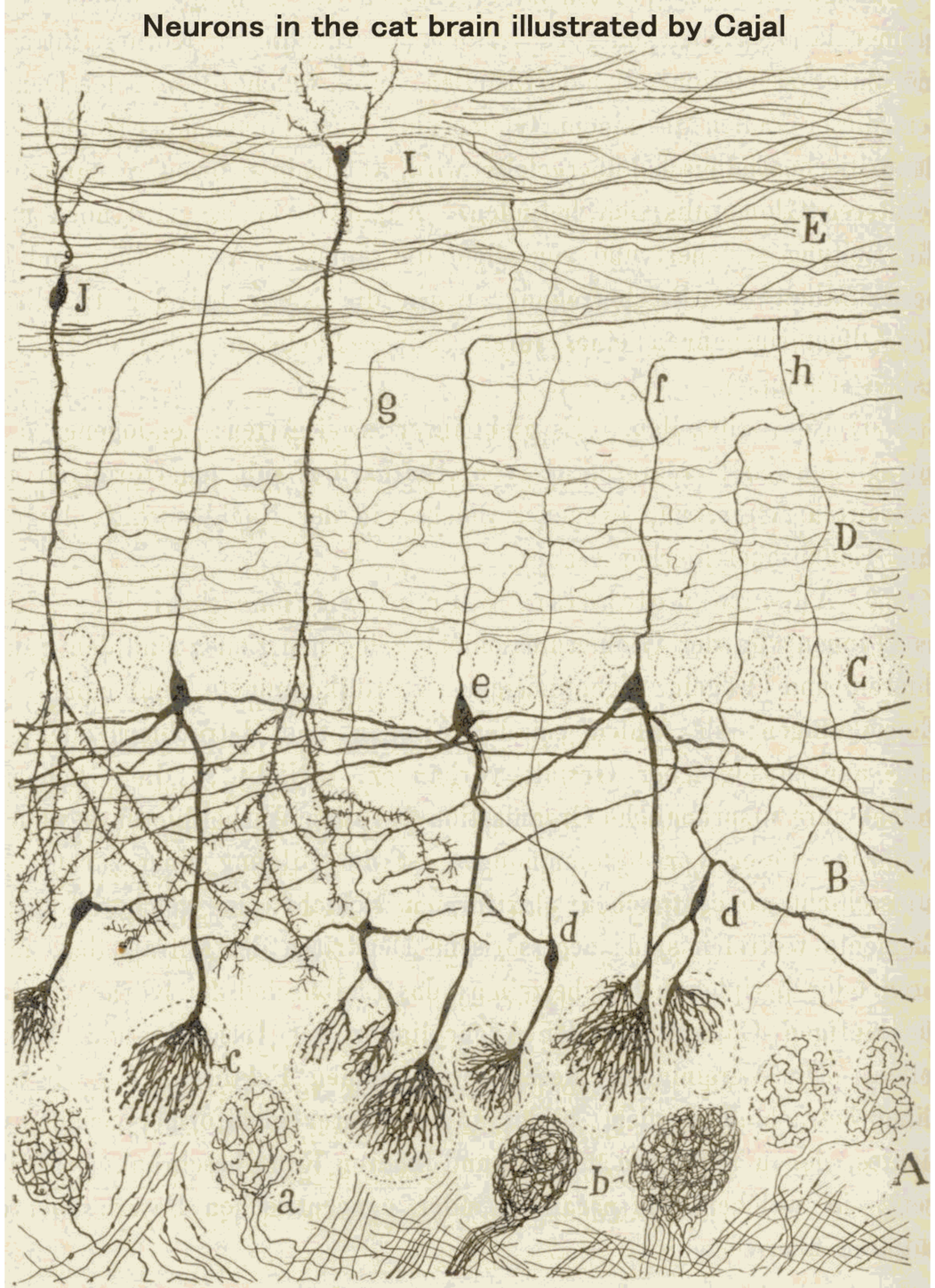
Santiago Ramón y Cajal - Neuroscientist won nobel prize 1906

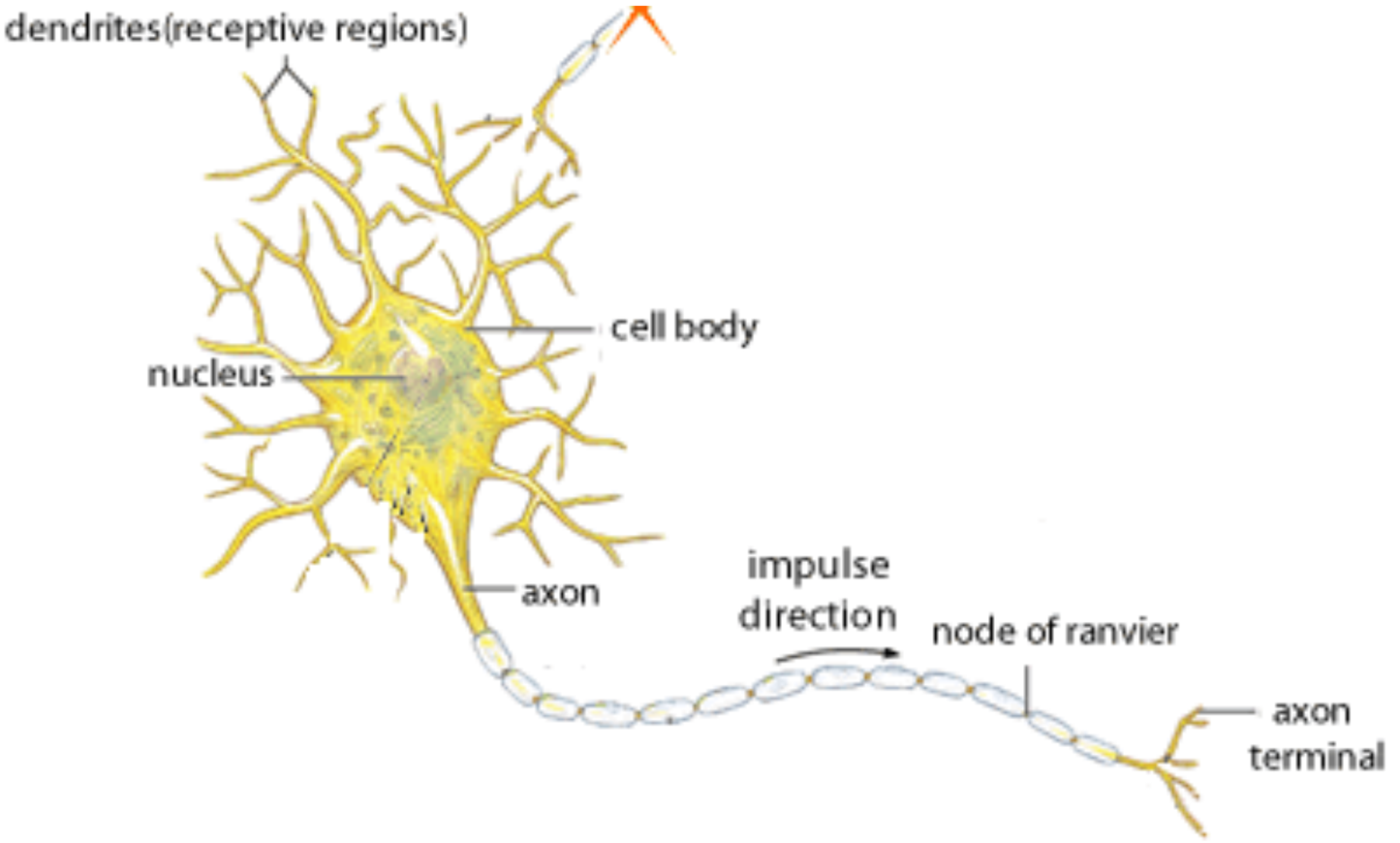

D
A
a
E



quatere
un tercio

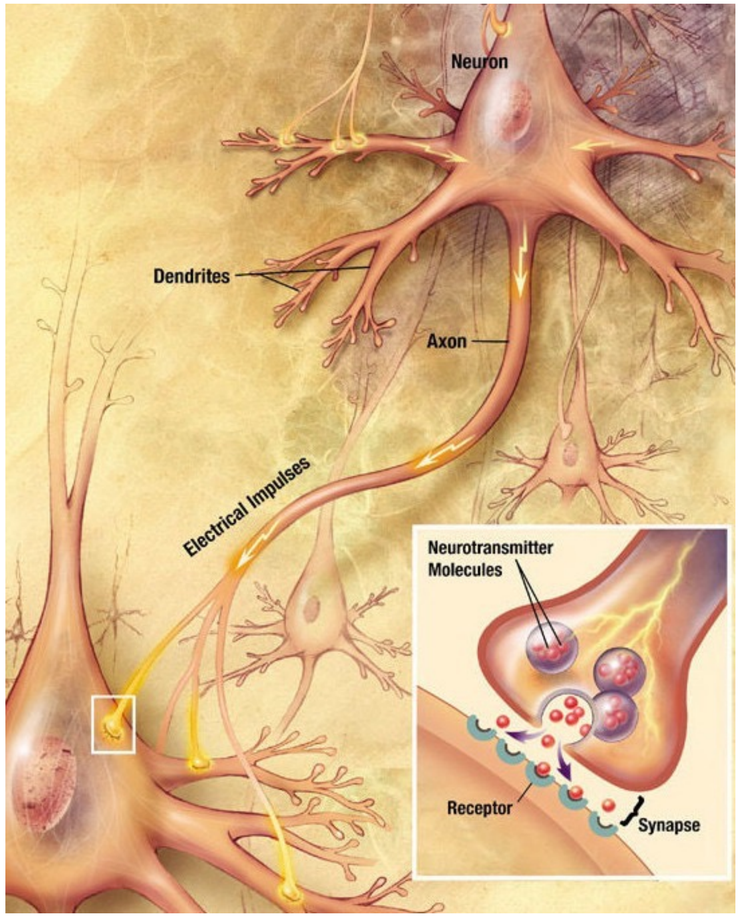
Neurons in the cat brain illustrated by Cajal




dendrites(receptive regions)
cell body
nucleus
axon
impulse
direction
node of ranvier
axon
terminal

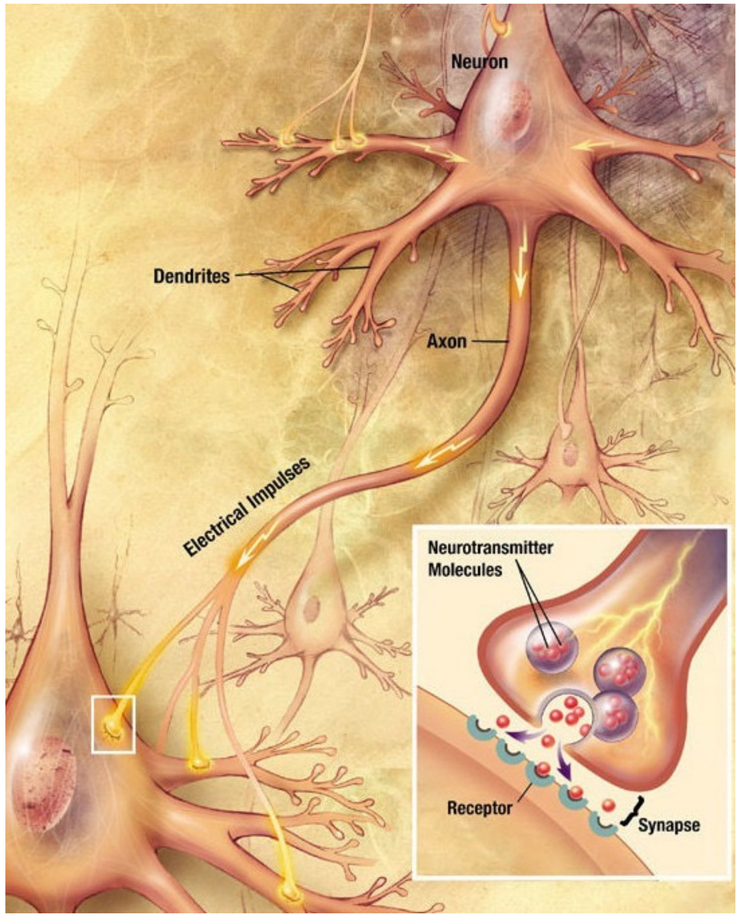
Neuron emits electrical pulses

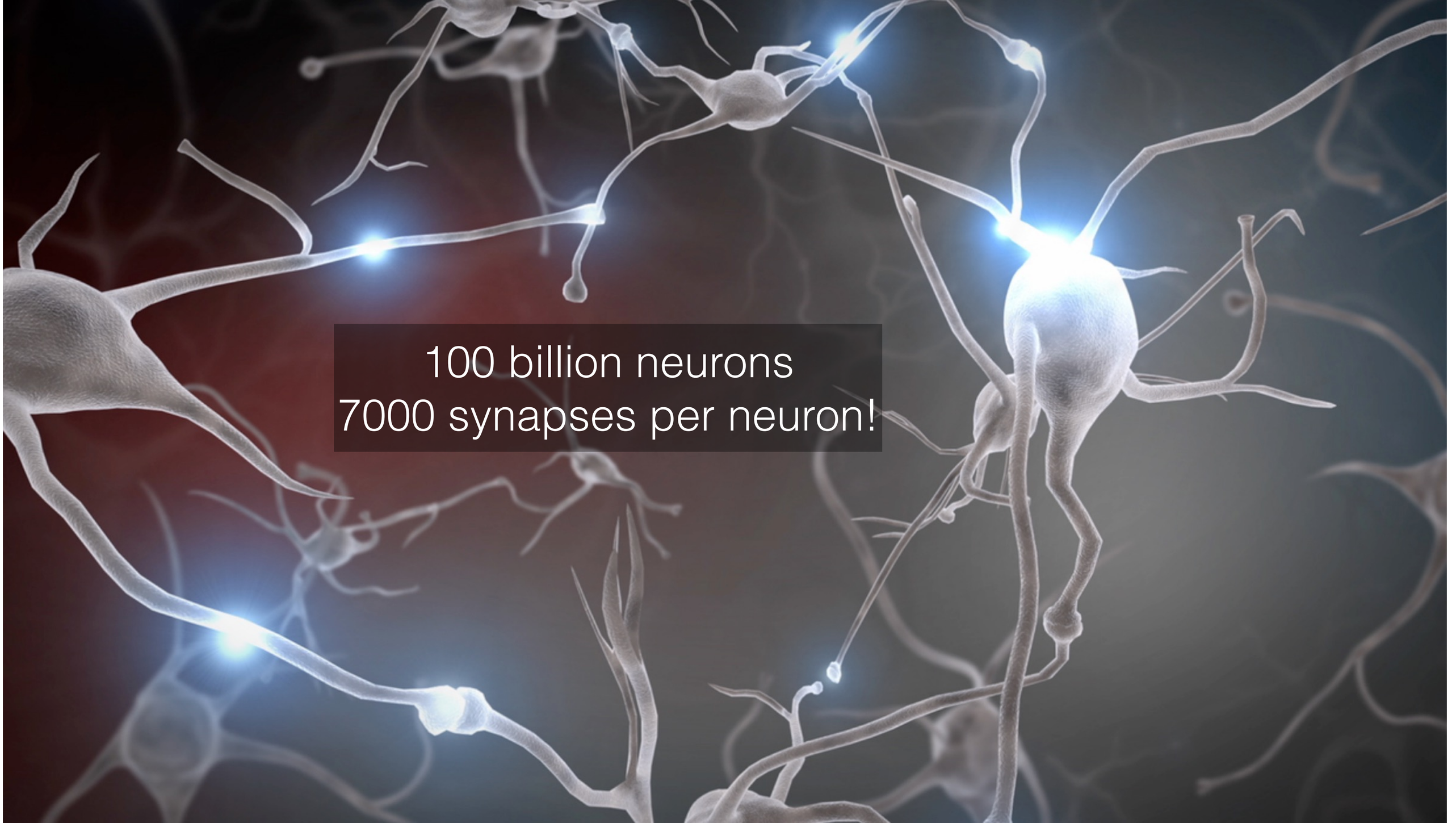
Synapses are the connections between neurons



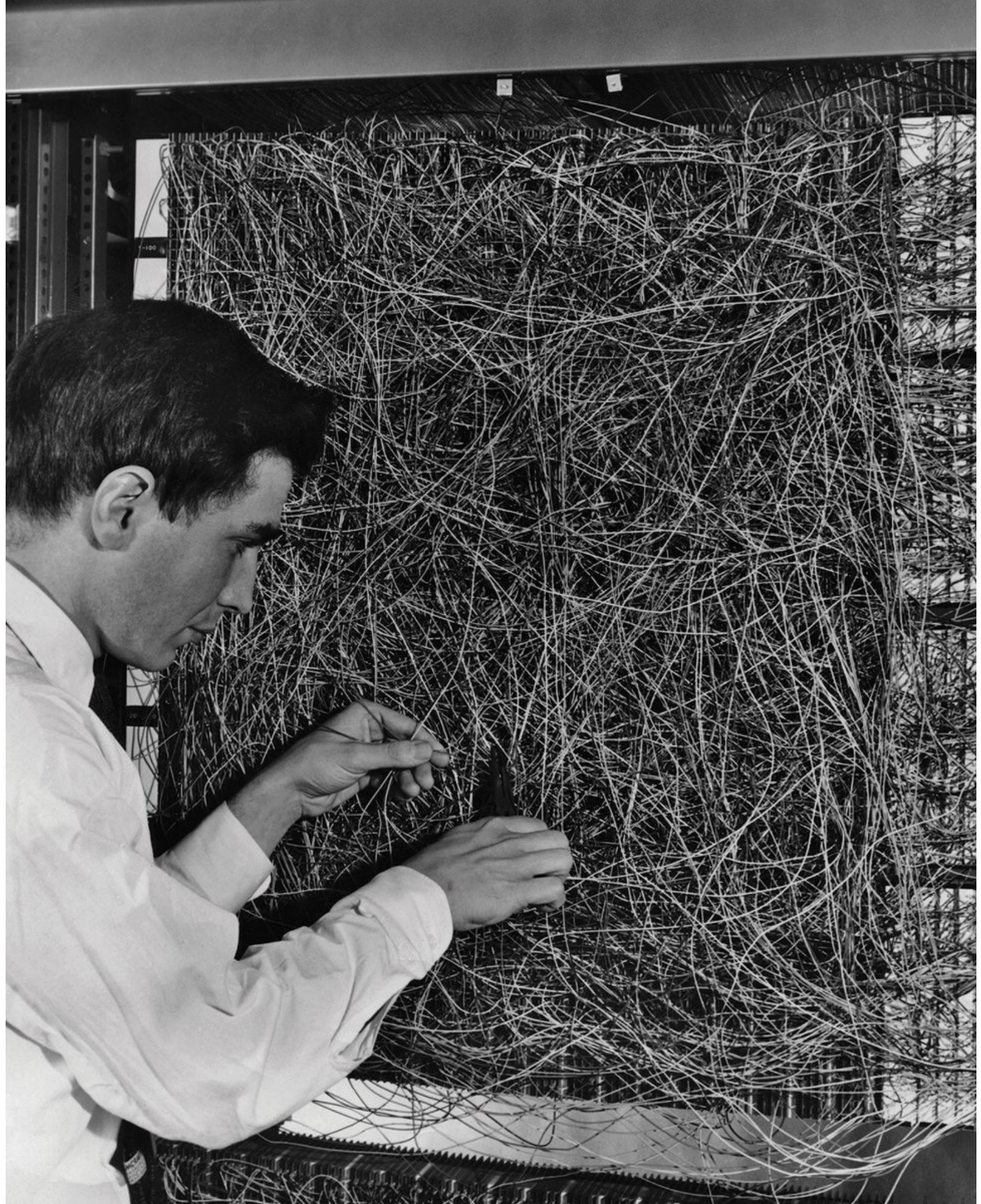
The majority of synapses
are formed in early childhood.

The brain has a high level of plasticity then.



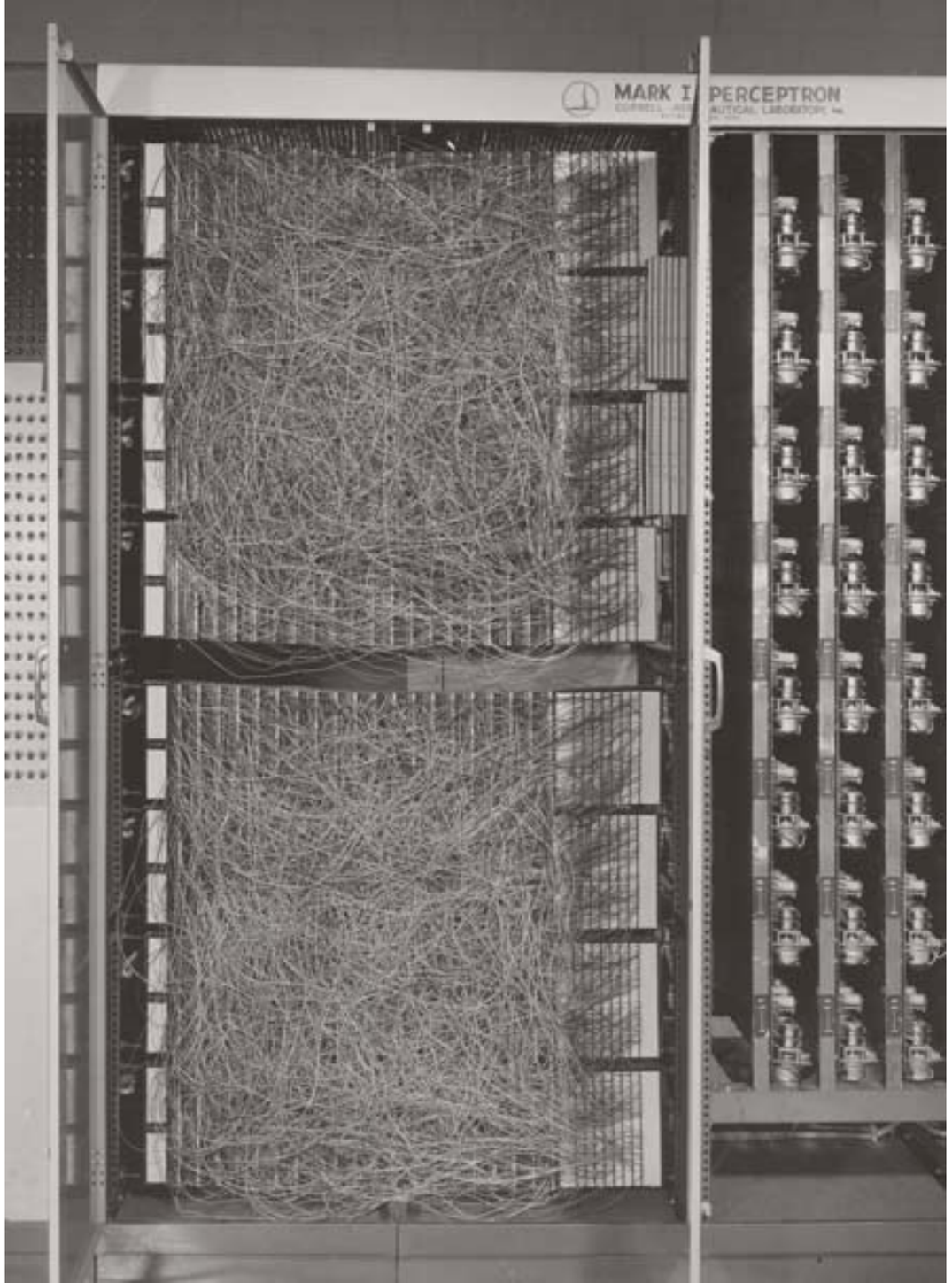

100 billion neurons
7000 synapses per neuron!

Frank Rosenblatt came up with the perceptron algorithm in 1957

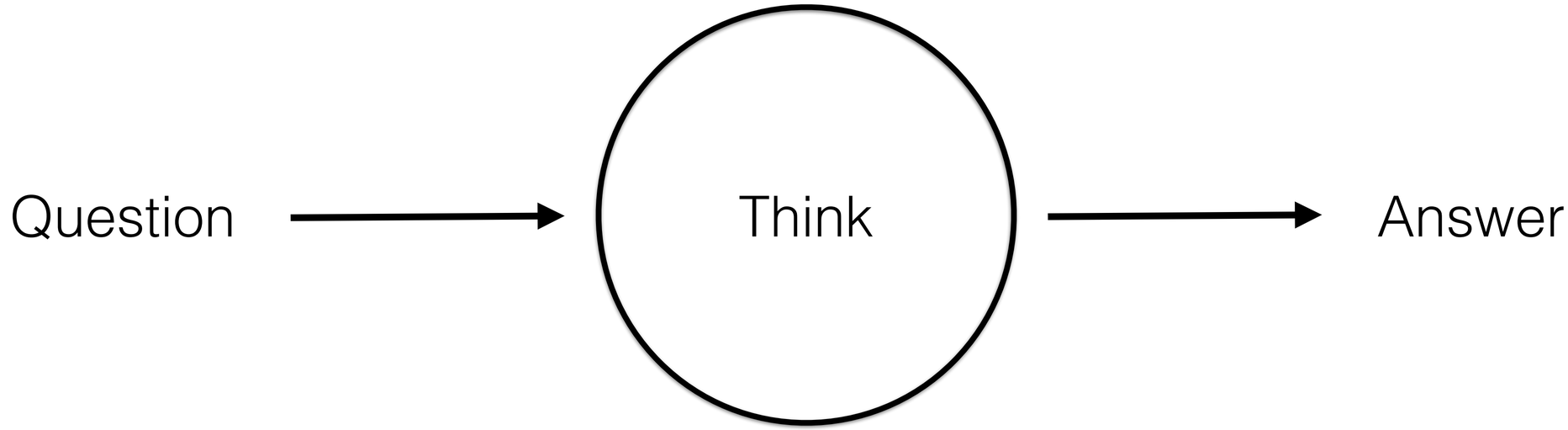


Mark I Perceptron

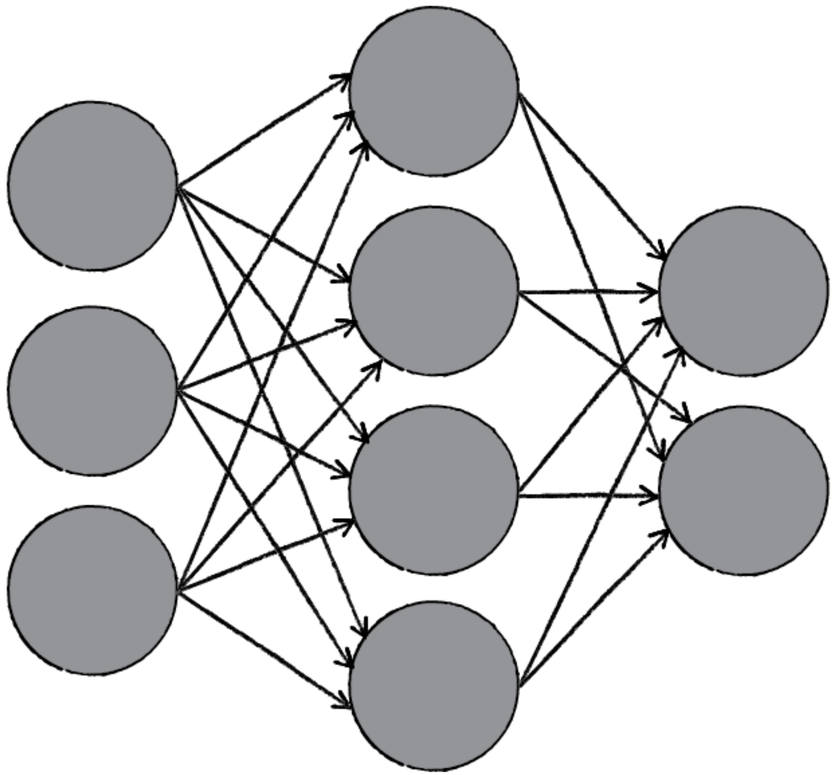
- Designed for image recognition
- 400 photocells
- Weights adjusted by potentiometers
- Motors adjusted the potentiometers

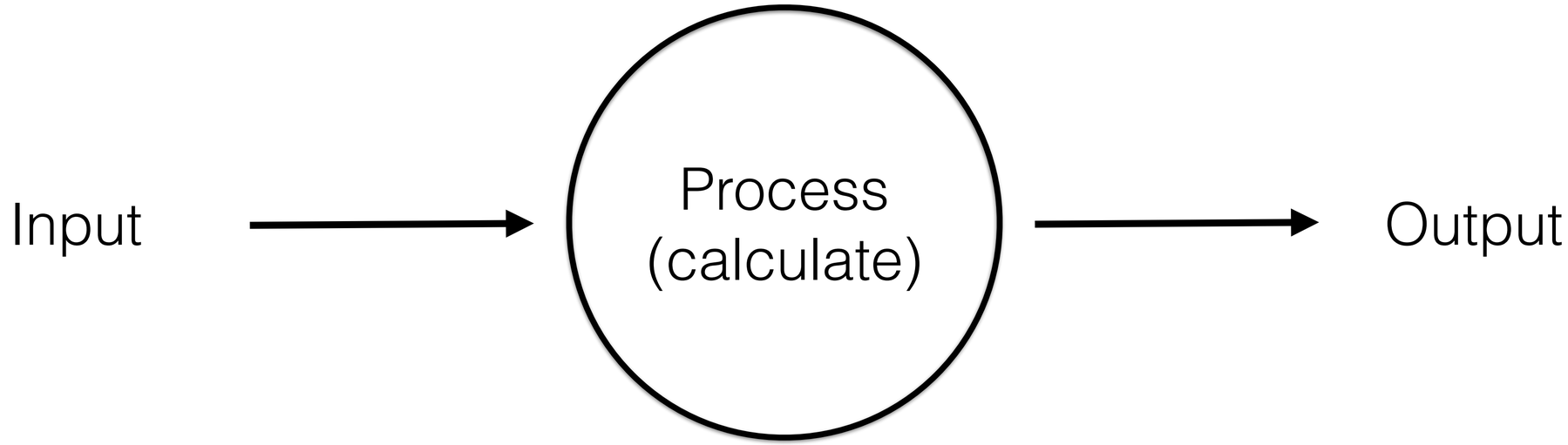


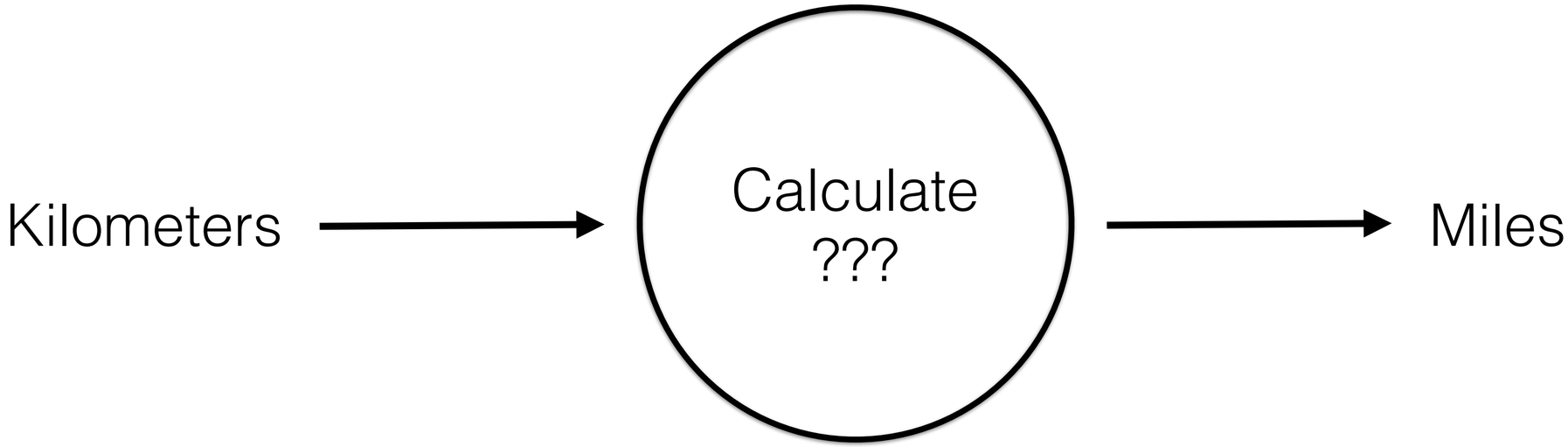
Predicting machine



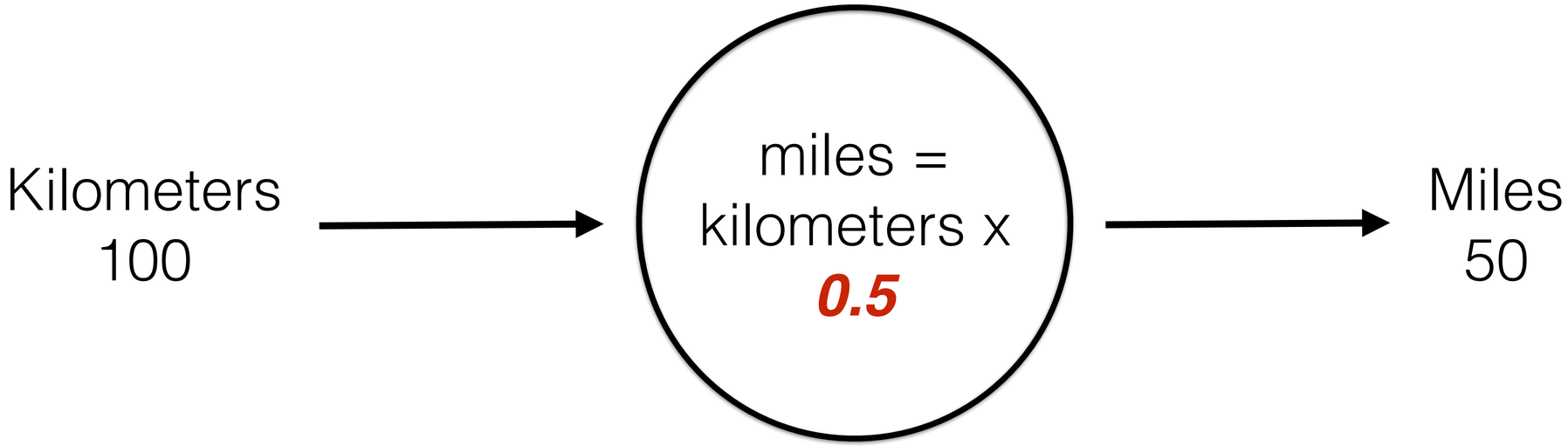
Graph




Input
Process
(calculate)
Output


Kilometers
Calculate
???
Miles

True Examples	Kilometers	Miles
1	0	0
2	100	62.137

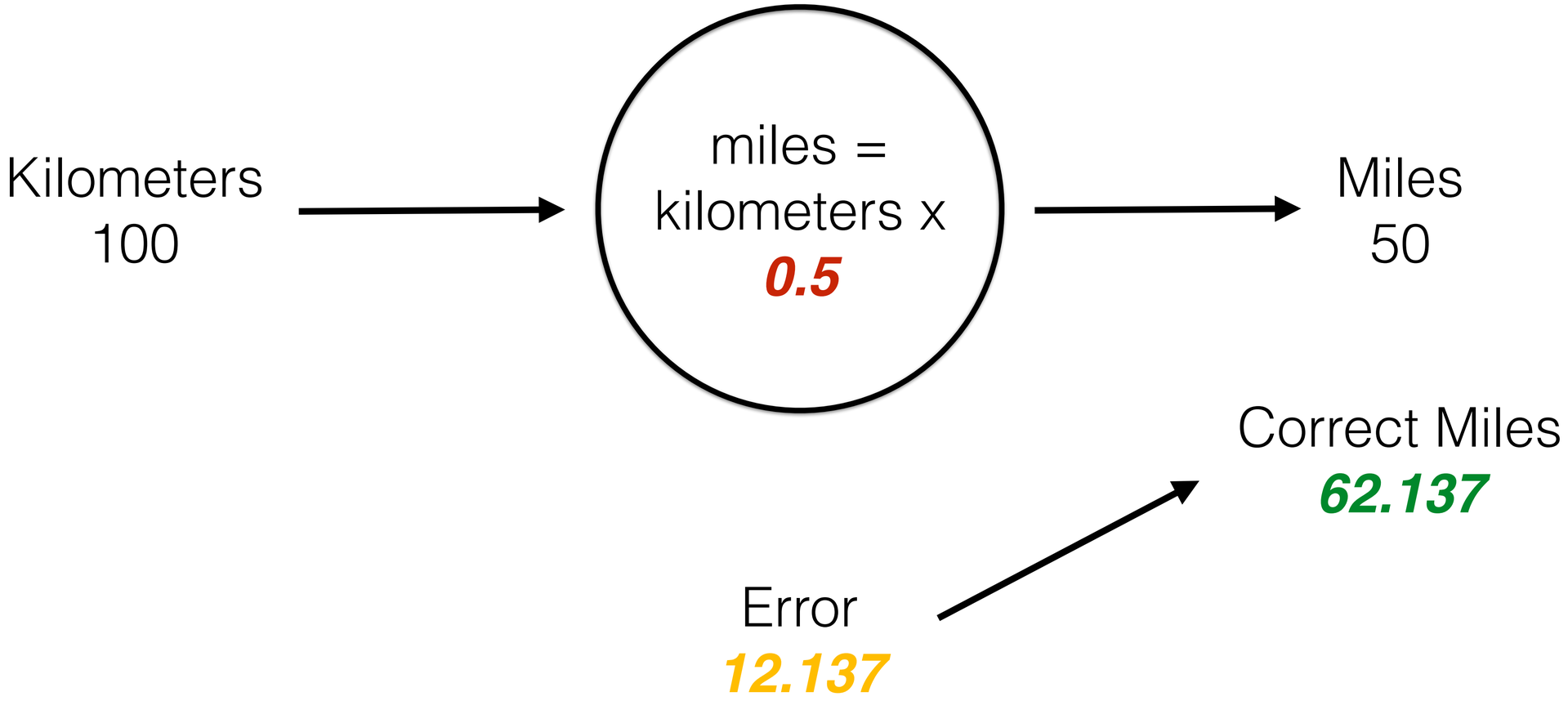


Try some random value to multiply the input by

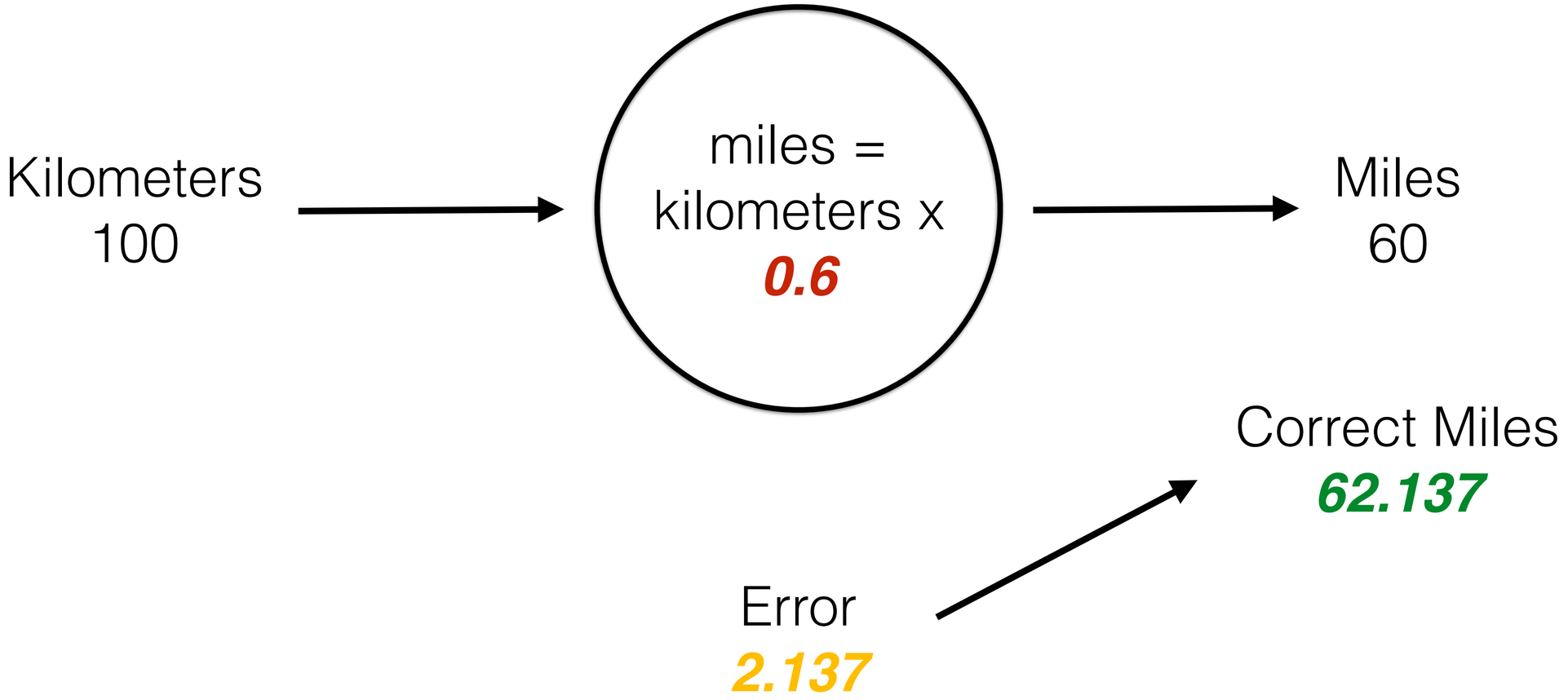
error = desired - calculated

= 62.137 - 50

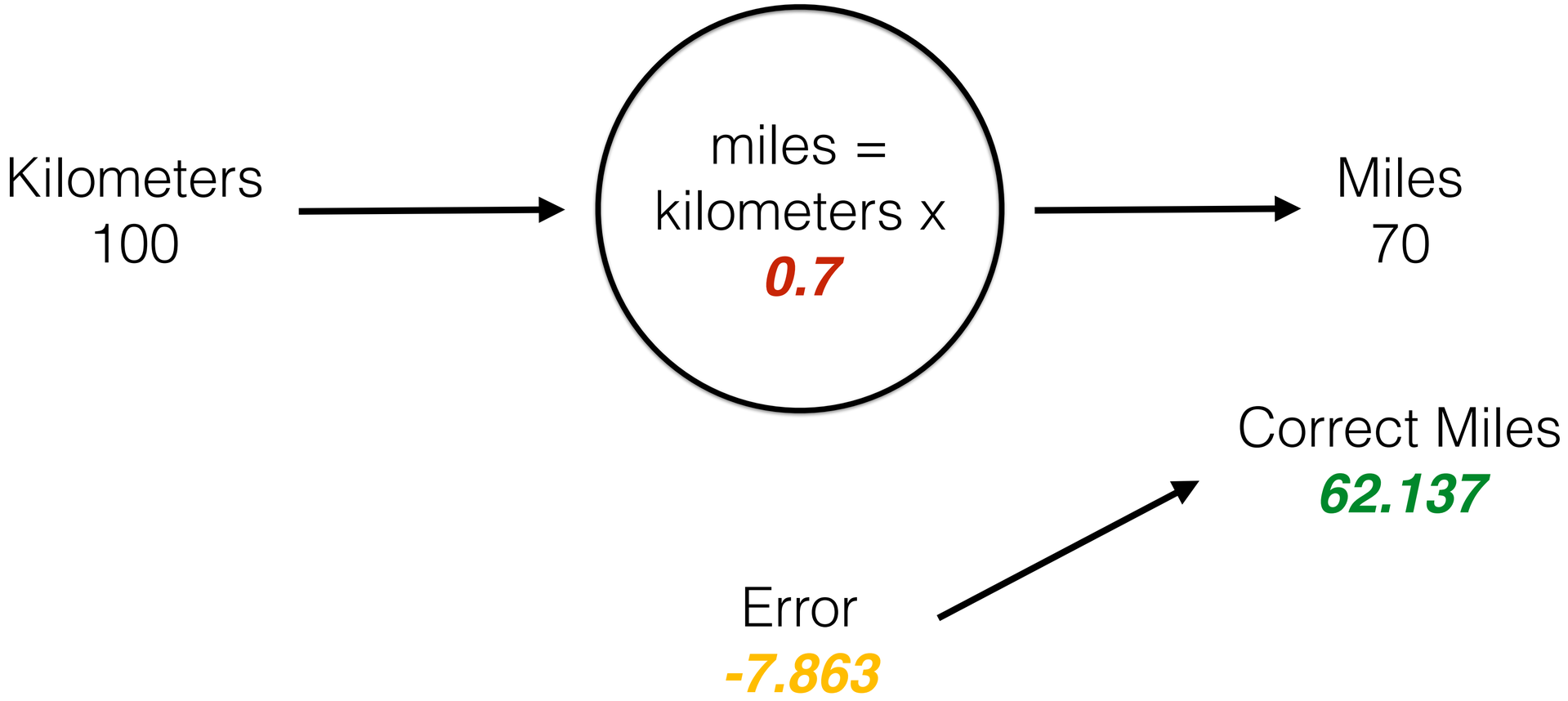
= 12.137


Kilometers
100
miles =
kilometers x
0.5
Miles
50
Correct Miles
62.137
Error
12.137

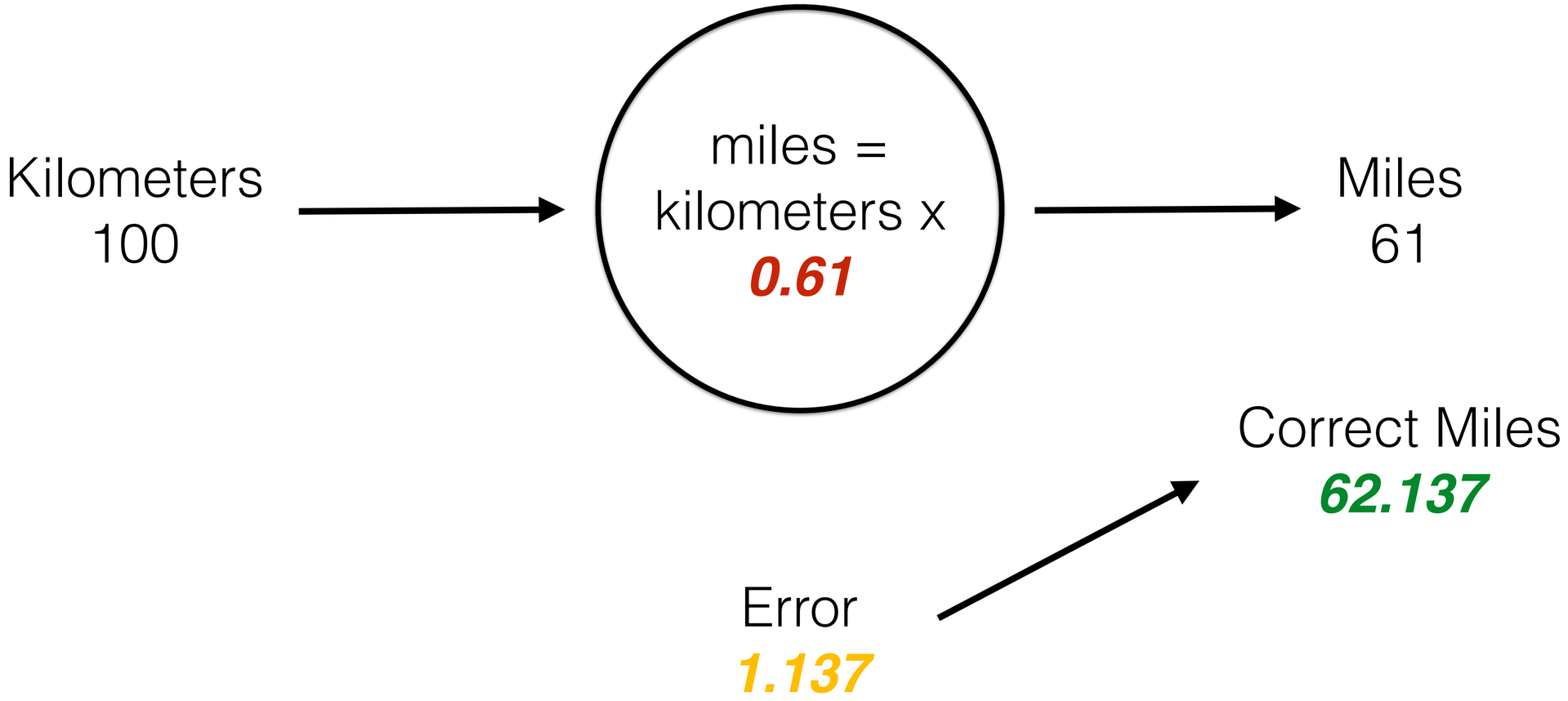
Let’s nudge the value up a little bit



Oops, too far!


Kilometers
100
miles =
kilometers x
0.7
Miles
70
Correct Miles
62.137
Error
-7.863

Getting there…



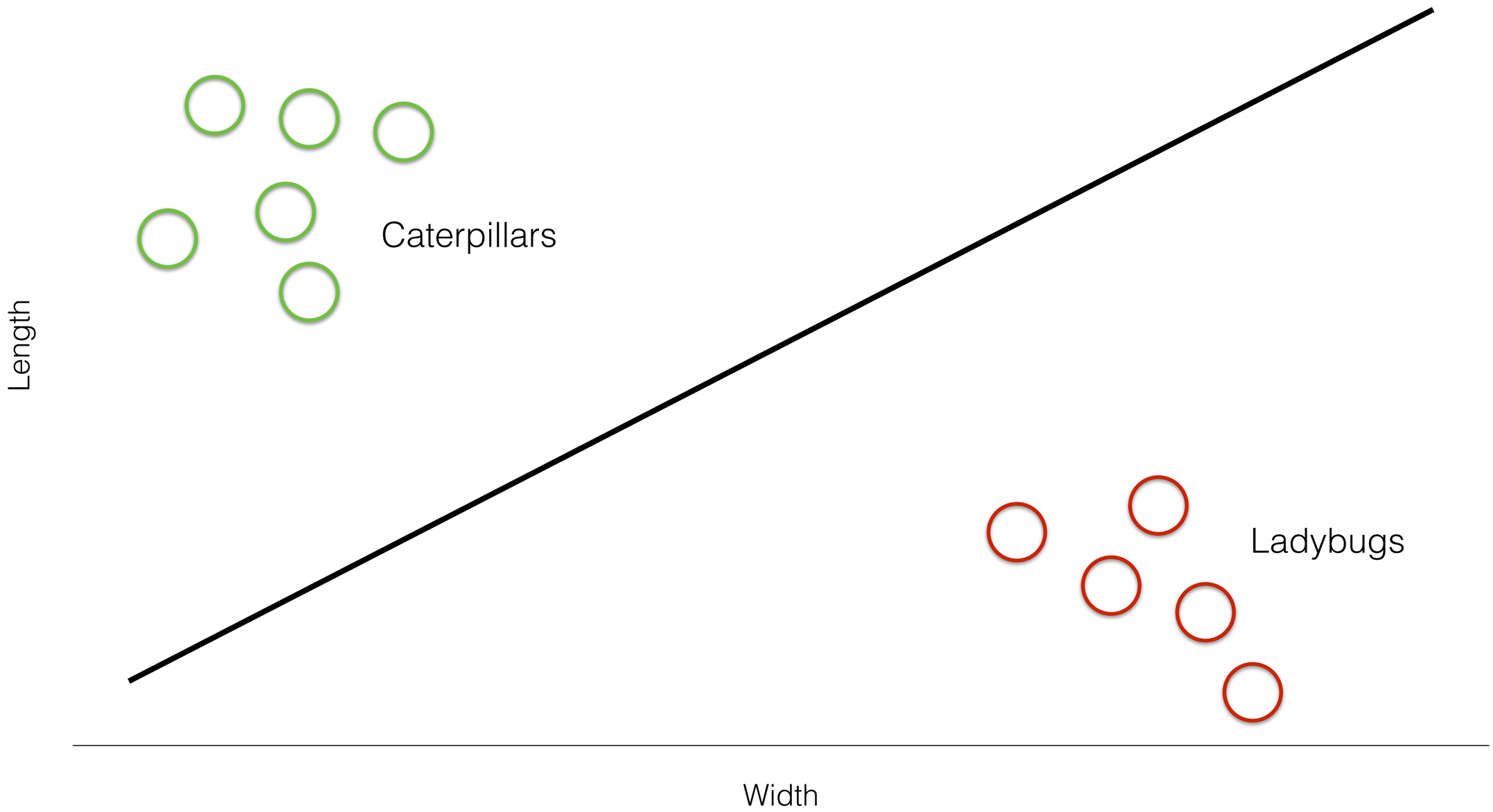
This is how a neural network learns!

In brief

- All computer systems have an input and an output, with a calculation in-between, neural networks are exactly the same.
- If we don’t know how something works, we can try to estimate it with a model that includes parameters we can change
- A good way to refine these models is to adjust the parameters based on how far away the model is compared to a know true example.


Caterpillars
Ladybugs
Length
Width

We want to find some line to divide our two classes



Training Data

Label	Width	Length
caterpillar	0.5	1.7
caterpillar	0.45	2
caterpillar	0.7	2.3
caterpillar	0.2	1.9
caterpillar	0.3	2.4
caterpillar	0.5	2.35
ladybug	2	0.8
ladybug	2.4	0.5
ladybug	2.5	0.2
ladybug	2.3	0.9
ladybug	2.2	0.6

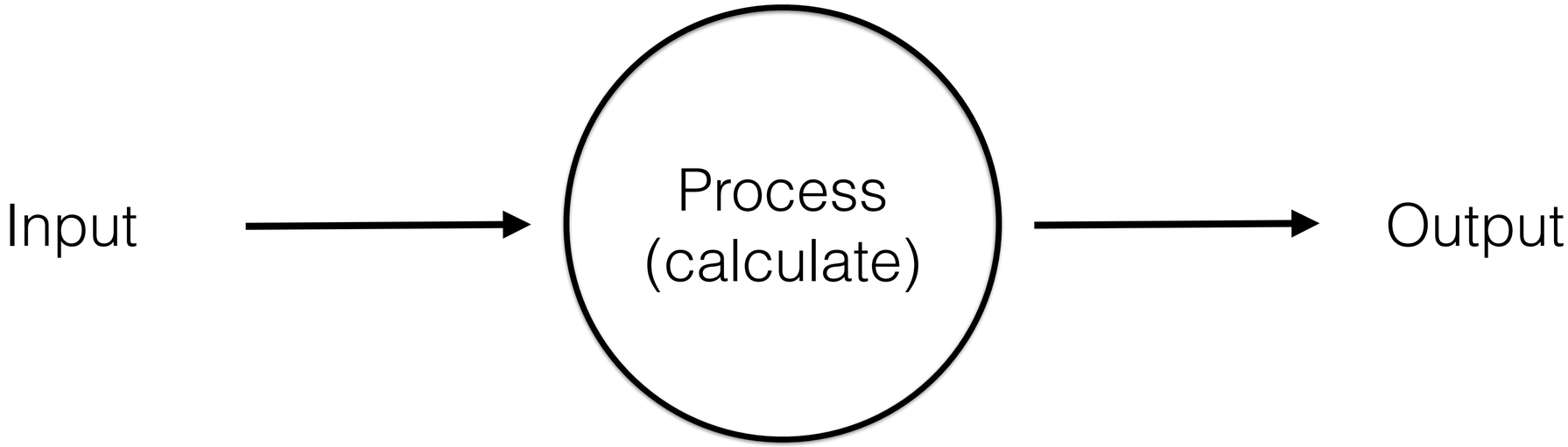
Equation for line:

y = mx + b

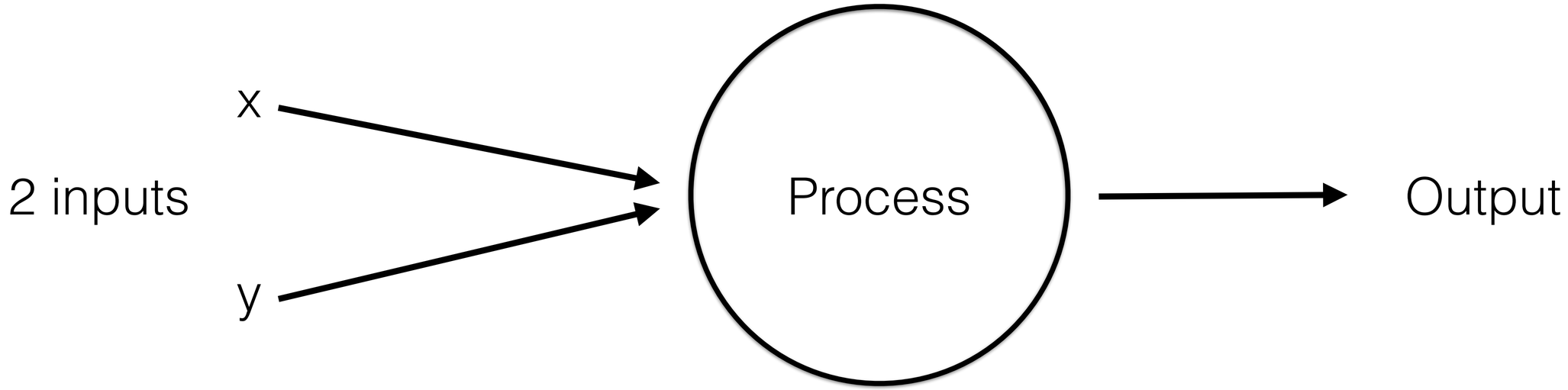
(m is slope & b is y intercept)

We need to figure out m & b

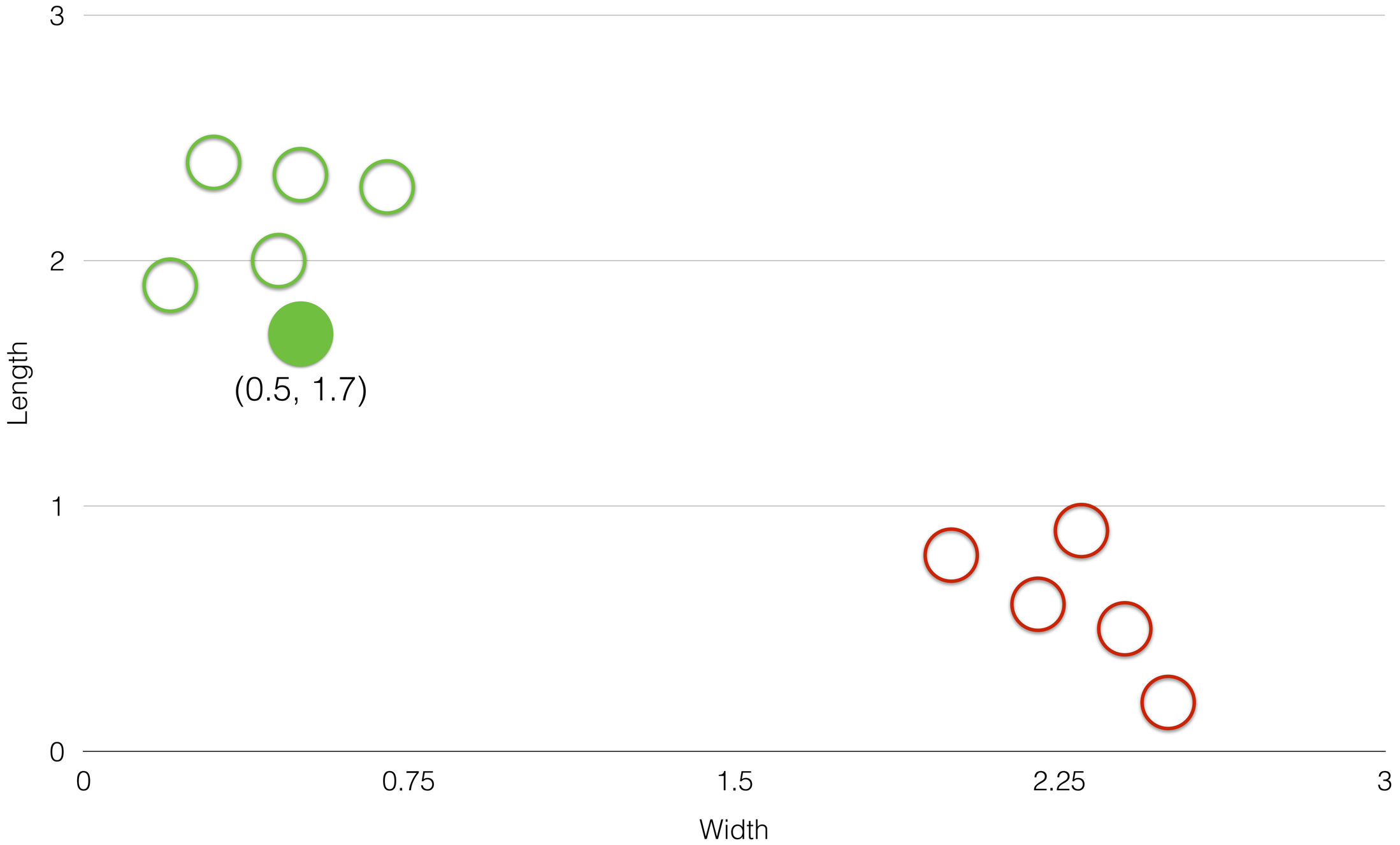
Perceptron



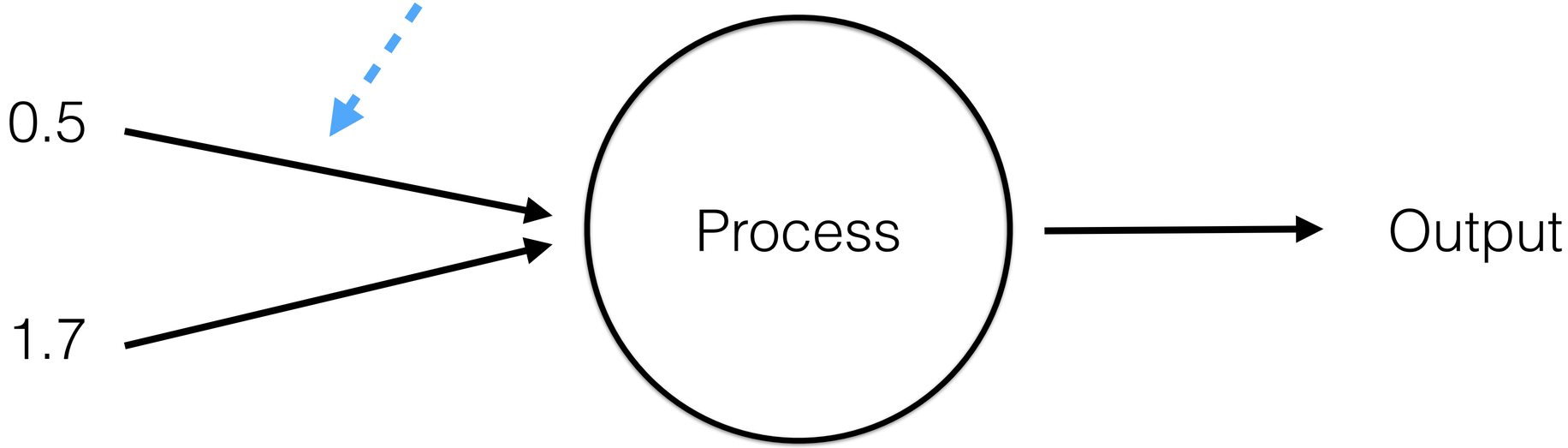
Simplest neural network, with just one neuron



Feed Forward —>


3
2
1
0
Length
(0.5, 1.7)
0
0.75
1.5
2.25
3
Width

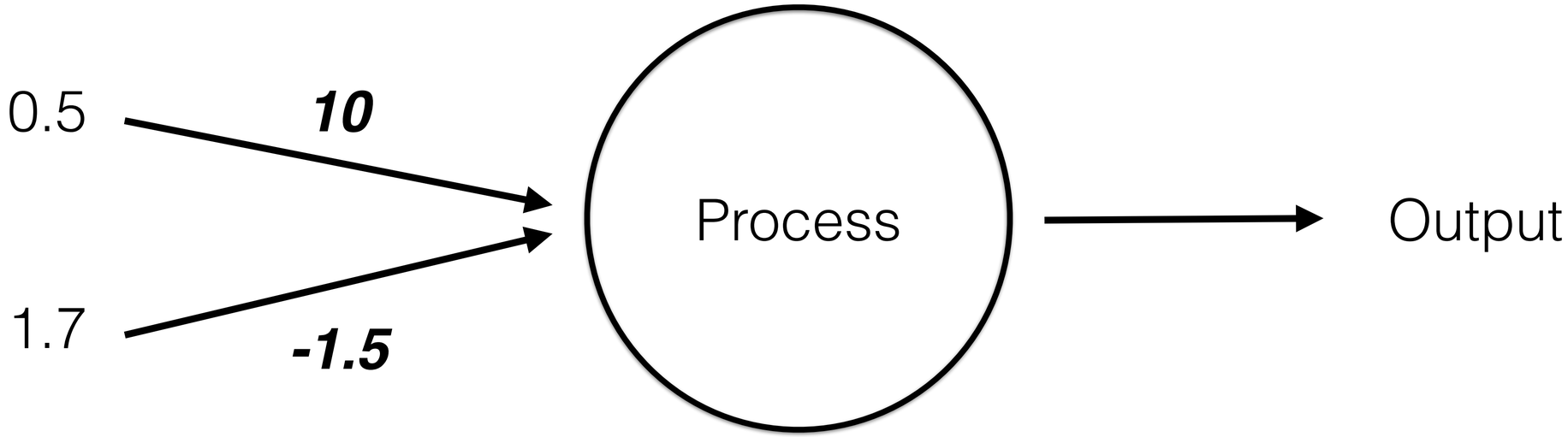
These lines represent synapses and will adjust the strength of the signal going in. In other words they ***weight*** the signal.



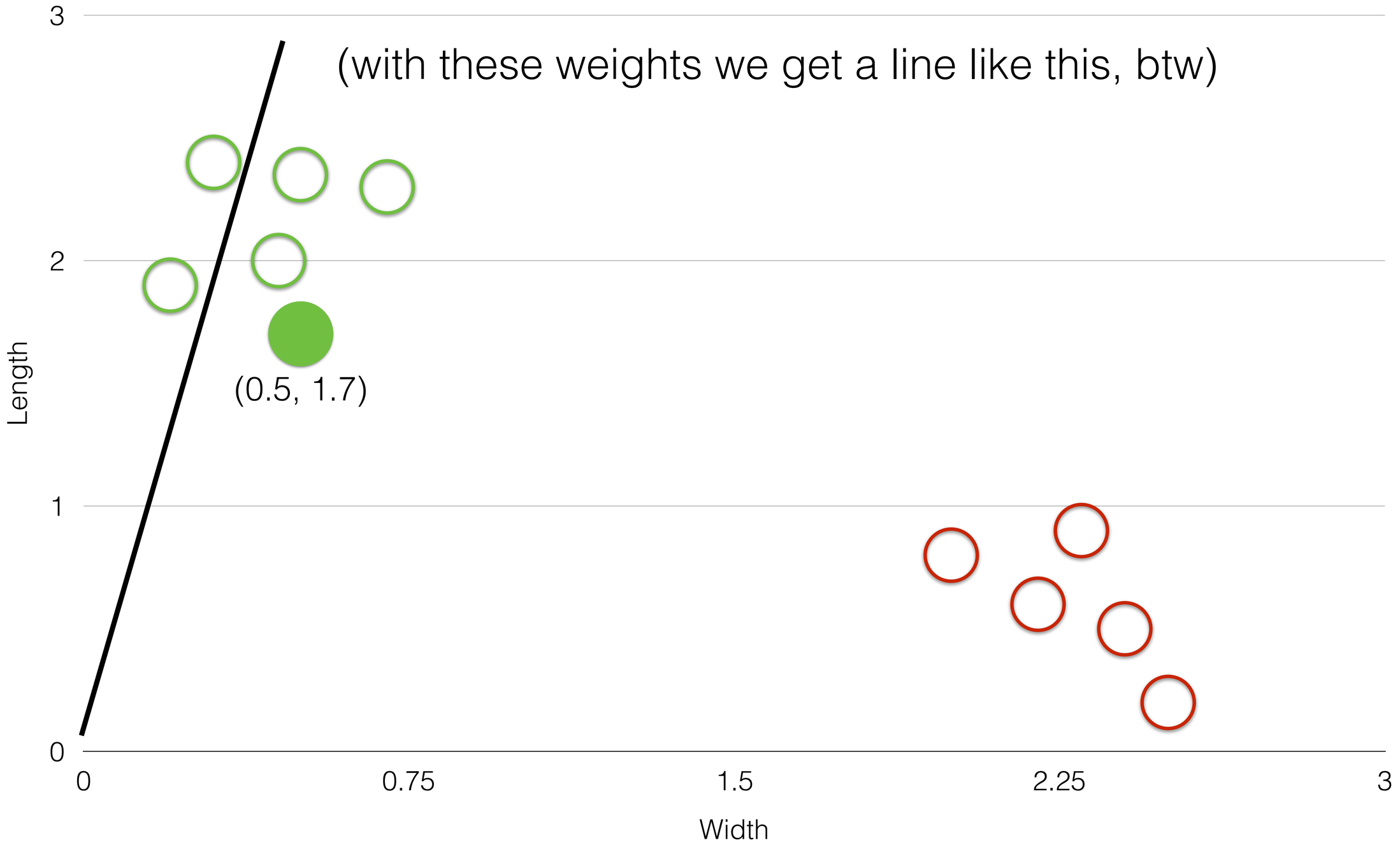
Feed Forward —>

Start with some random weights.

weight 1 = 10
weight 2 = -1.5

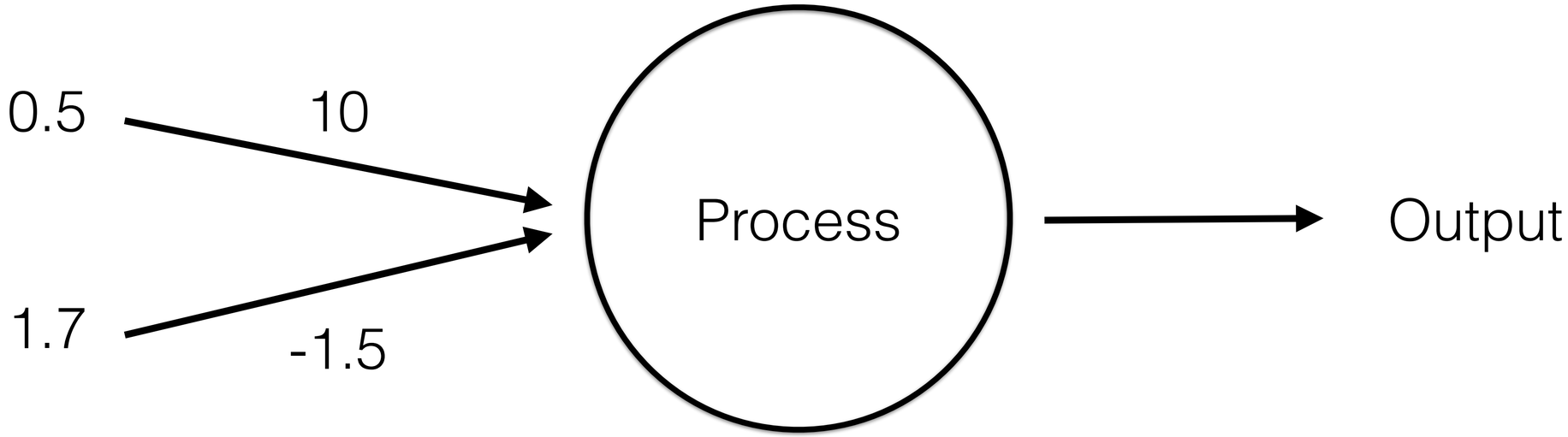


Feed Forward —>

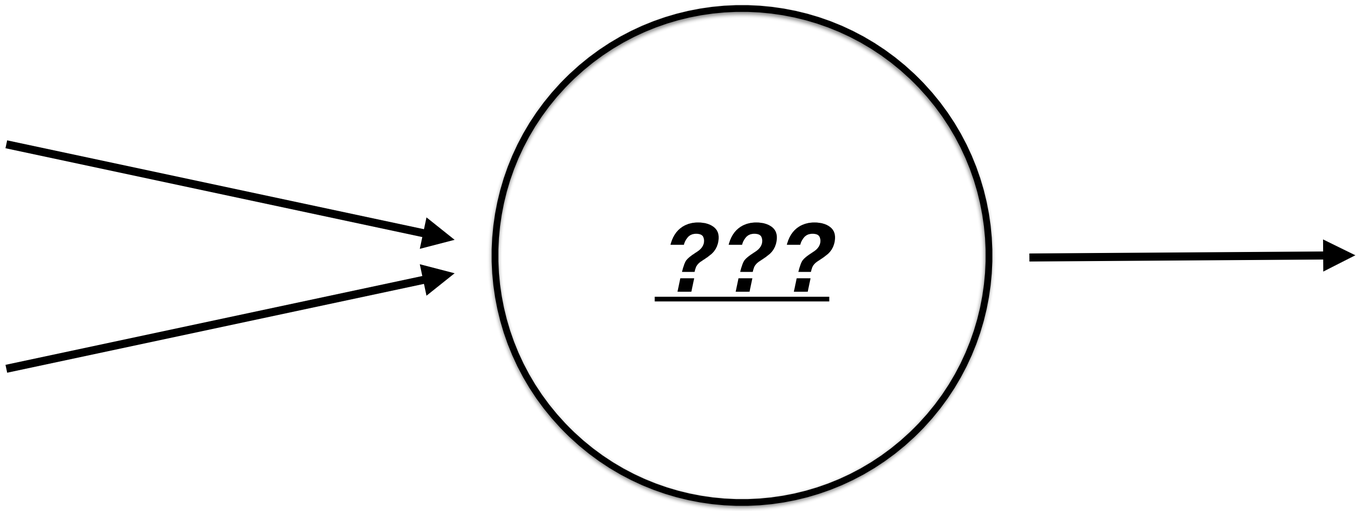

(with these weights we get a line like this, btw)
(0.5, 1.7)
Length
Width
0
1
2
3
0
0.75
1.5
2.25
3

Multiply the signal by the weights, and then sum everything.

$$(0.5 * 10) + (1.7 * -1.5) = 2.45$$

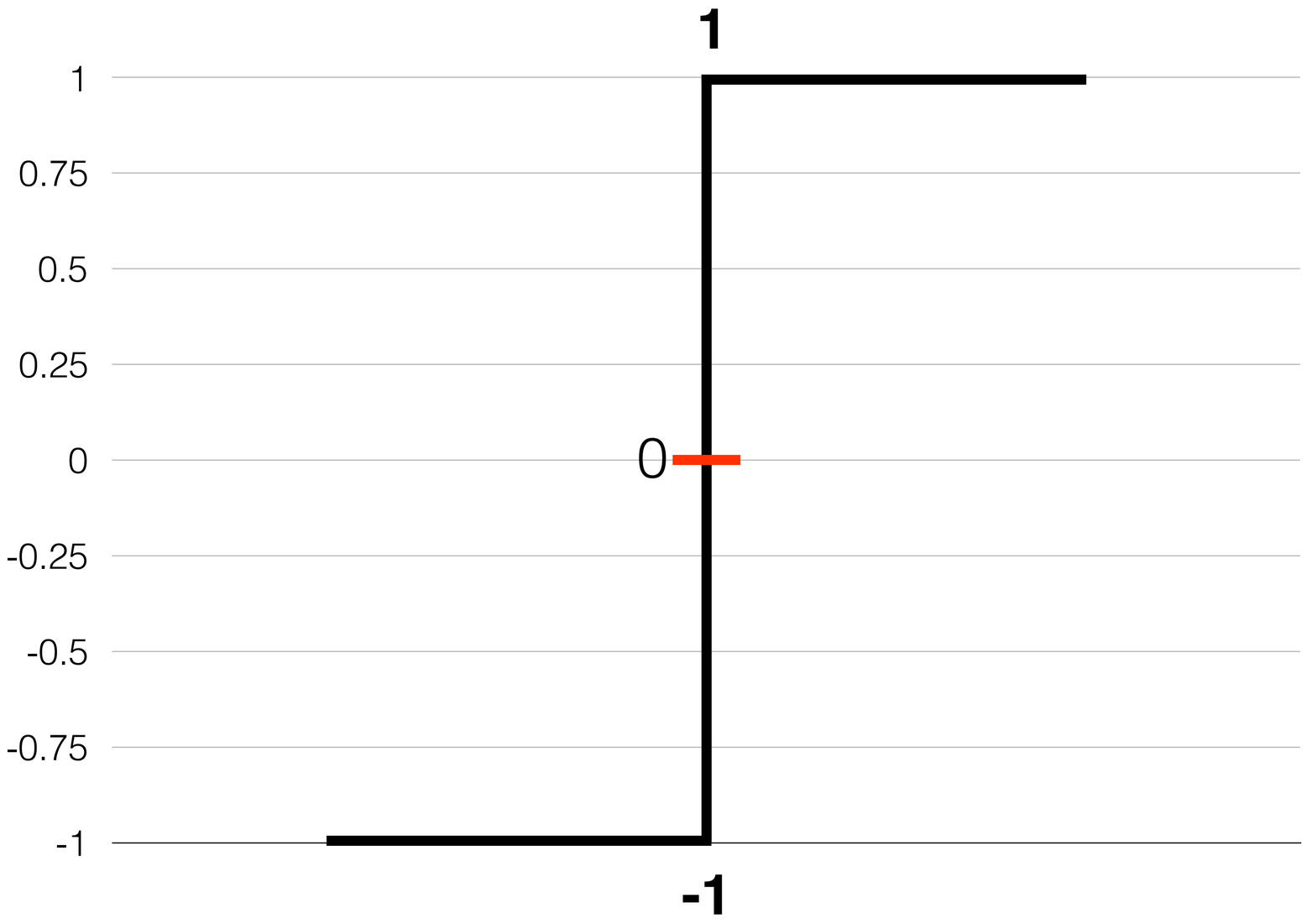


Feed Forward —>



Activation Function

Step (sign) function



There are only two answers, either the point is above or below the line.

If it’s above the correct answer is -1.

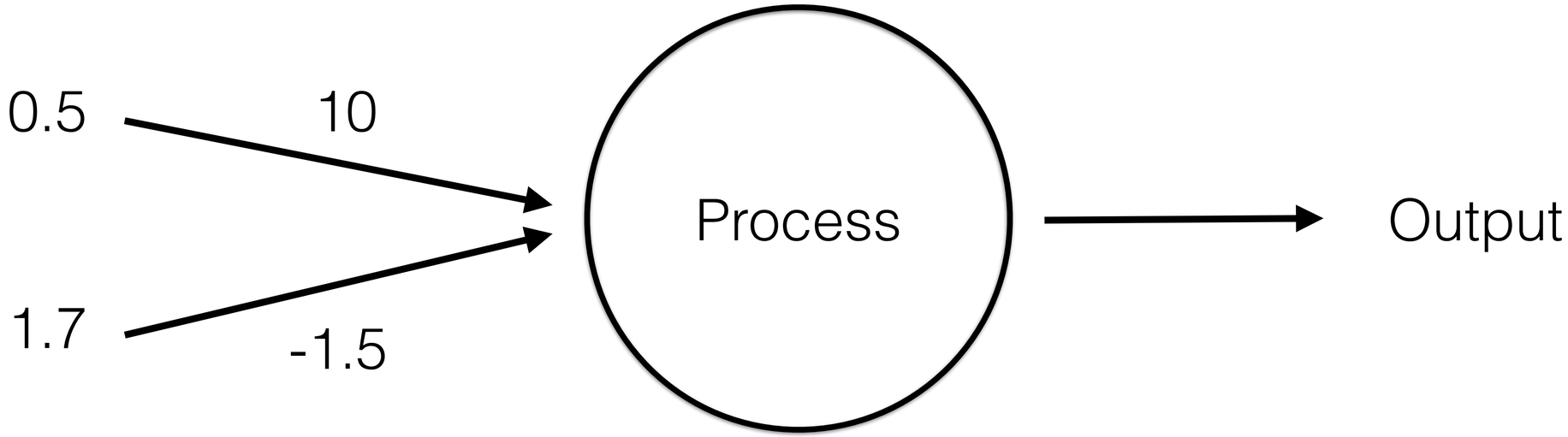
If below, the correct answer is 1.

Training Data

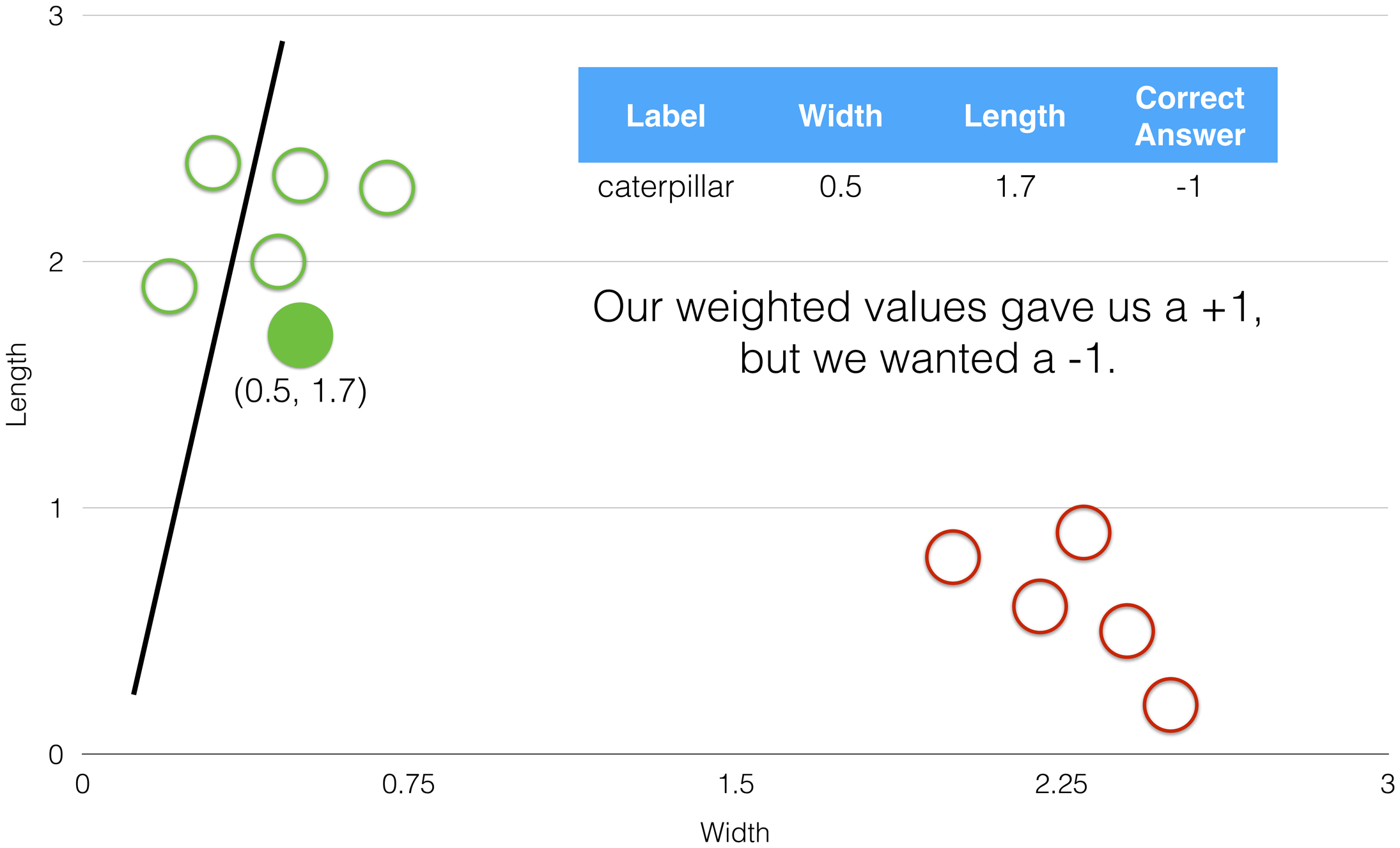
Label	Width	Length	Correct Answer
caterpillar	0.5	1.7	-1
caterpillar	0.45	2	-1
caterpillar	0.7	2.3	-1
caterpillar	0.2	1.9	-1
caterpillar	0.3	2.4	-1
caterpillar	0.5	2.35	-1
ladybug	2	0.8	1
ladybug	2.4	0.5	1
ladybug	2.5	0.2	1
ladybug	2.3	0.9	1
ladybug	2.2	0.6	1

(0.5 * 10) + (1.7 * -1.5) = 2.45

Take the sign of our weighted sum:
sign(2.45) = +1



Activate



Label	Width	Length	Correct Answer
caterpillar	0.5	1.7	-1

Our weighted values gave us a +1, but we wanted a -1.

How do we train our perceptron???

Find the error

error = desired - guess

(*I wanted 10 but I guessed 7, I was 3 off*)

Only four possible results

Desired	Guess	Error
-1	-1	0
-1	+1	-2
+1	-1	2
+1	+1	0

Label	Width	Length	Correct Answer
caterpillar	0.5	1.7	-1

Our weighted values gave us a +1,
but we wanted a -1.

desired - guess = error
-1 - 1 = -2

We need to adjust our weights somehow based on our error.

$$\text{new weight} = \text{weight} + \Delta\text{weight}$$

Δ (delta) means change

$$\text{new weight} = \text{weight} + \Delta\text{weight}$$

$$\Delta\text{weight} = \text{error} * \text{input}$$

$$\text{new weight} = \text{weight} + \Delta\text{weight}$$

$$\Delta\text{weight} = \text{error} * \text{input}$$

$$\textbf{NEW WEIGHT} = \textbf{weight} + \textbf{error} * \textbf{input}$$

NEW WEIGHT = weight + error * input

Example

- Let’s say we had an input value of two, and our weight is 8

NEW WEIGHT = weight + error * input

Example

- Let’s say we had an input value of two, and our weight is 8
- 2 * 8 = 16, but target value was 10

NEW WEIGHT = weight + error * input

Example

- Let’s say we had an input value of two, and our weight is 8
- 2 * 8 = 16, but target value was 10
- Error = desired - guess, so 10 - 16 = -6

NEW WEIGHT = weight + error * input

Example

- Let's say we had an input value of two, and our weight is 8
- 2 * 8 = 16, but target value was 10
- Error = desired - guess, so 10 - 16 = -6
- Plug these values into our equation: 8 (current weight) + -6 (error) * 2 (input)

NEW WEIGHT = weight + error * input

Example

- Let's say we had an input value of two, and our weight is 8
- 2 * 8 = 16, but target value was 10
- Error = desired - guess, so 10 - 16 = -6
- Plug these values into our equation: 8 (current weight) + -6 (error) * 2 (input)
- New weight = -4

NEW WEIGHT = weight + error * input

Example

- Let's say we had an input value of two, and our weight is 8
- 2 * 8 = 16, but target value was 10
- Error = desired - guess, so 10 - 16 = -6
- Plug these values into our equation: 8 (current weight) + -6 (error) * 2 (input)
- New weight = -4

2 * -4 = -8

So far in the opposite direction!!!

Learning Rate

Learning Rate

NEW WEIGHT = weight + error * input * learning rate

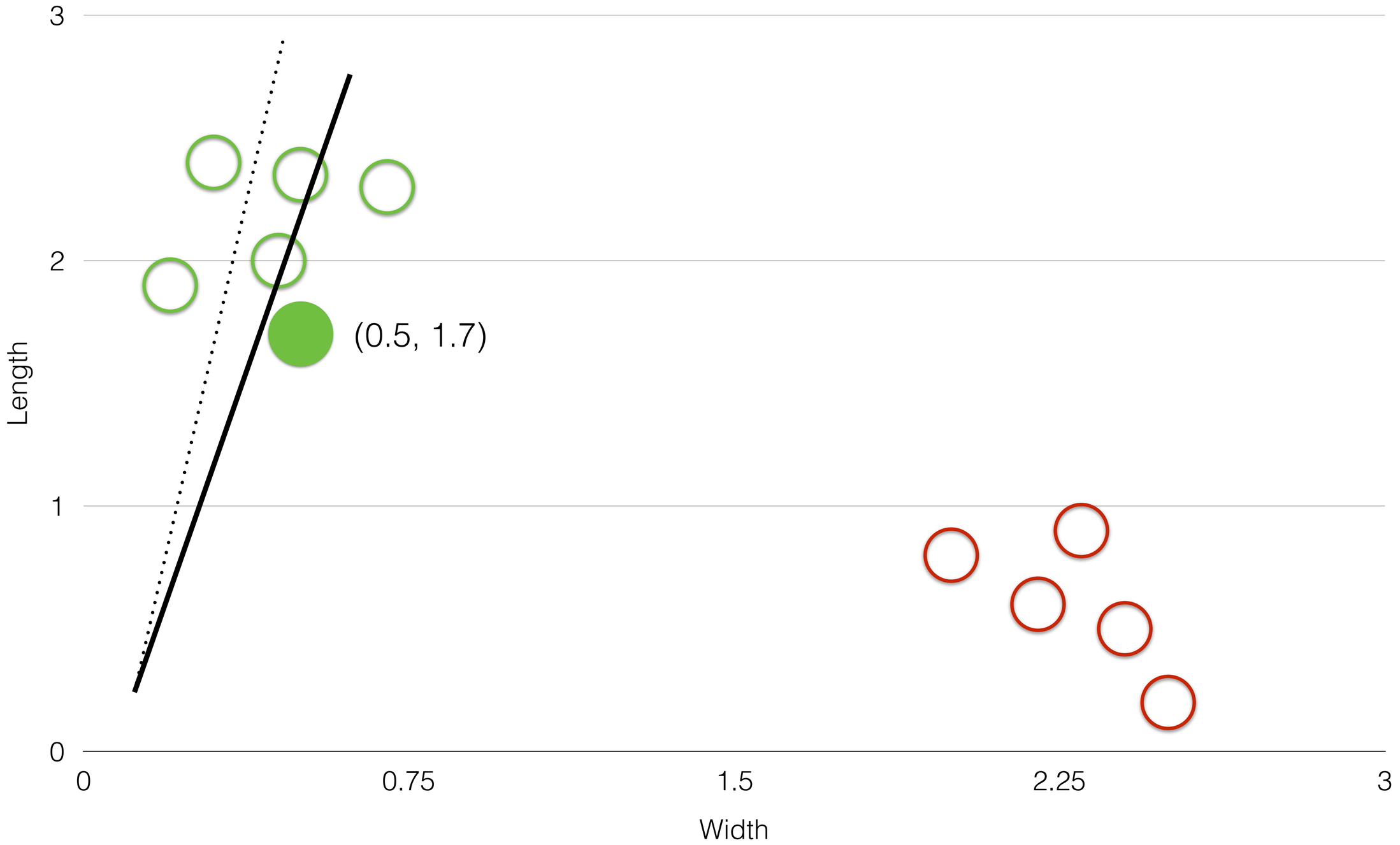
NEW WEIGHT = weight + error * input * learning rate

8 (current weight) + -6 (error) * 2 (input) * 0.1 (learning rate)

new weight = 6.8

2 * 6.8 = 13.6

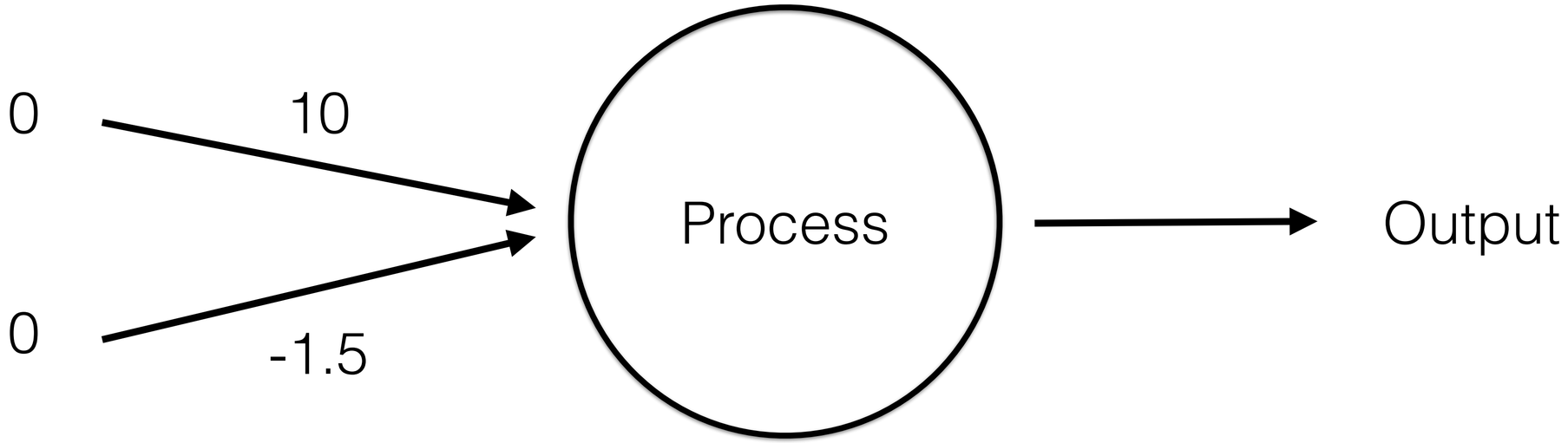
last time was 16, now getting closer to target of 10


3
2
1
0
Length
0
0.75
1.5
2.25
3
Width
(0.5, 1.7)

What if we had an input point of (0,0)?

Multiply the signal by the weights, and then sum everything.

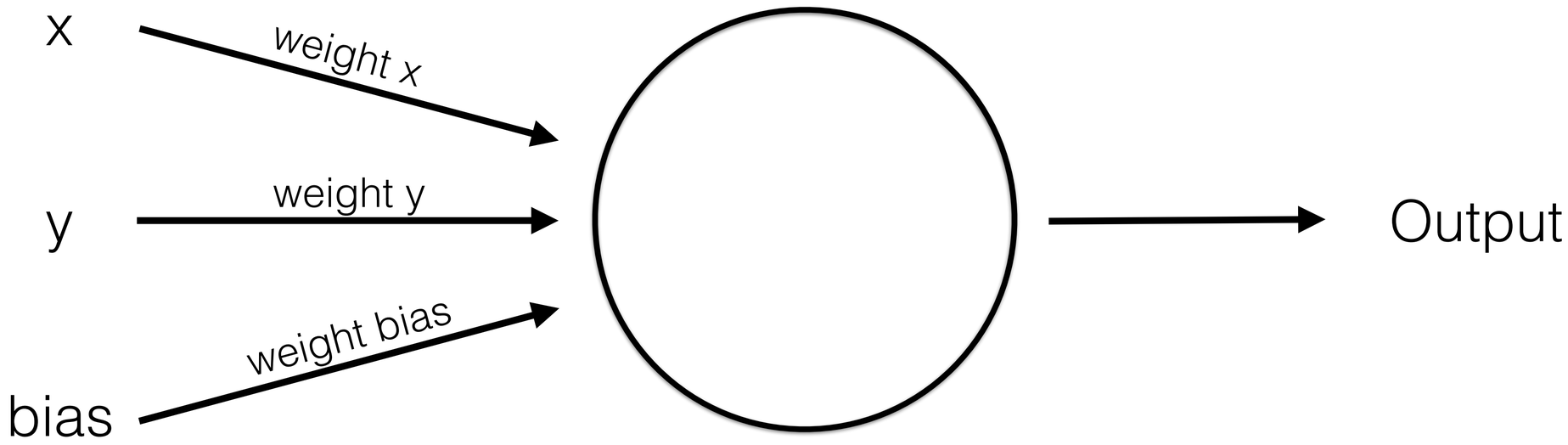
$$(0 * 10) + (0 * -1.5) = 0$$



What if we had an input point of (0,0)?

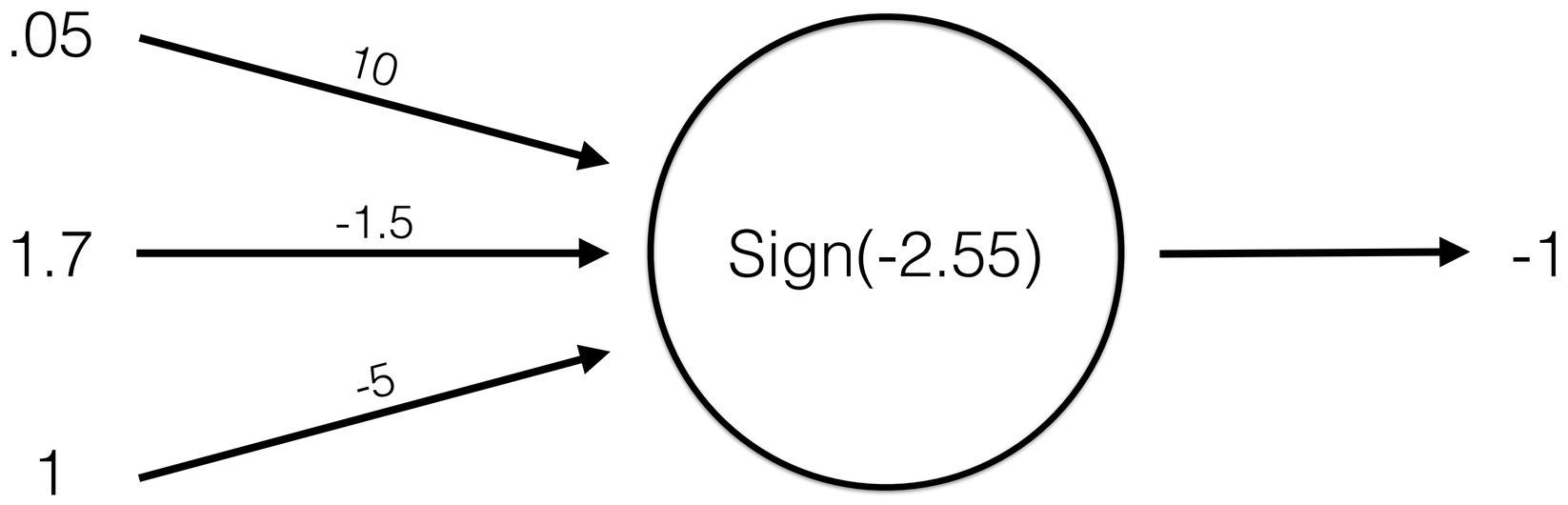
Bias

Bias is 1, it also has a weight



Multiply and sum all inputs, including bias

(0.5 * 10) + (1.7 * -1.5) + (1 * -5)= -2.55



Before we got a +1!!