Sigmoid Neuron & Neural networks

Alexandre Bergel
http://bergel.eu
21/09/2020

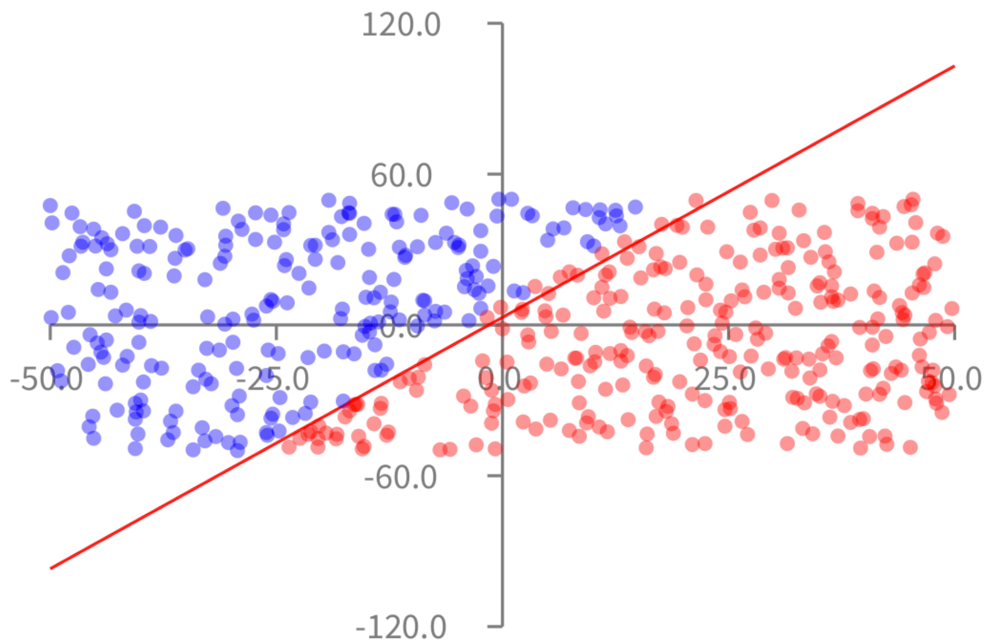
Outline

1. Sigmoid Neuron
2. Neural Network

Outline

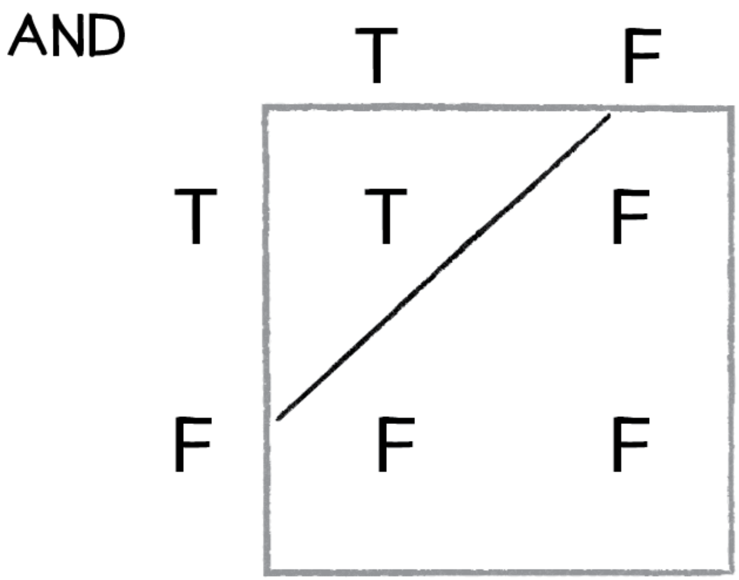
1. **Sigmoid Neuron**
2. Neural Network

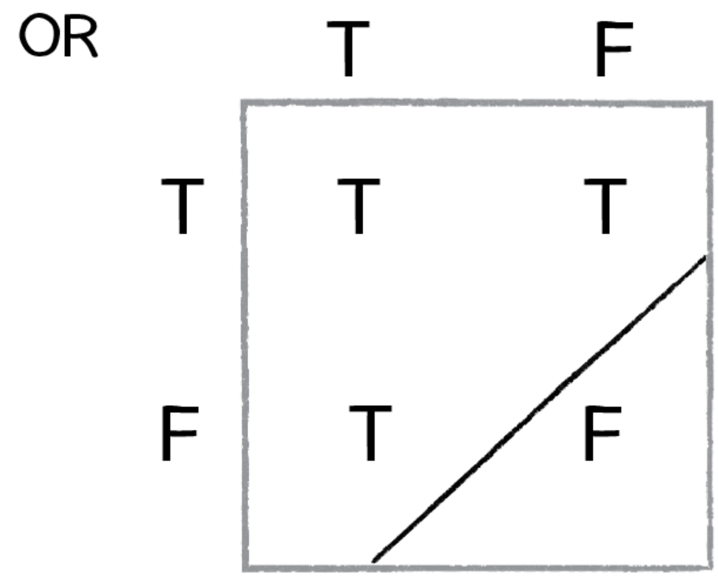
What we have seen so far



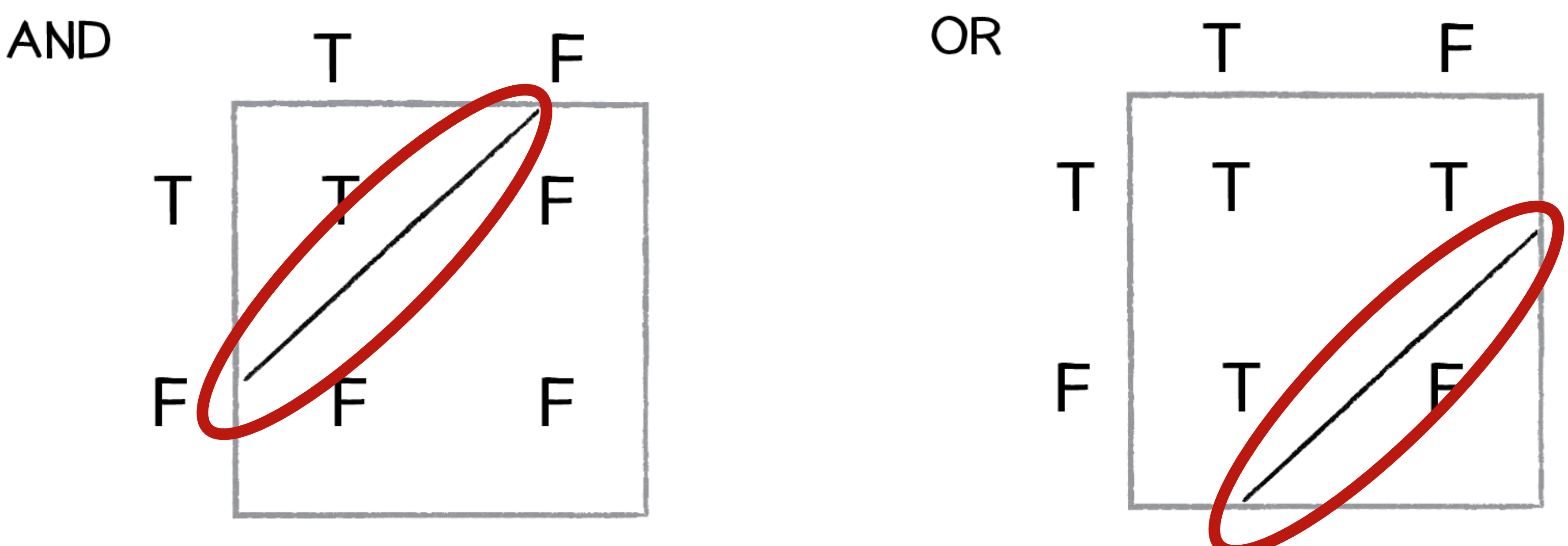
We have seen that the perceptron can (more or less accurately) guess the side on which a point is located

What we have seen so far



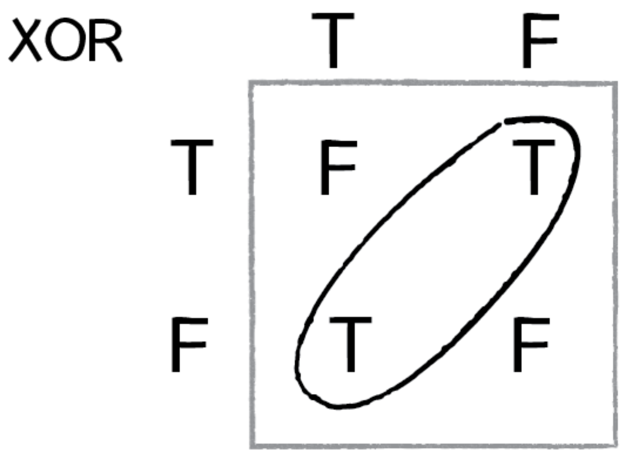


What we have seen so far



This is very similar to the space & point problem.
It is all about having a line as a limit

Limitation of a perceptron



With the XOR operation, you cannot have one unique line that limit the range of true and false

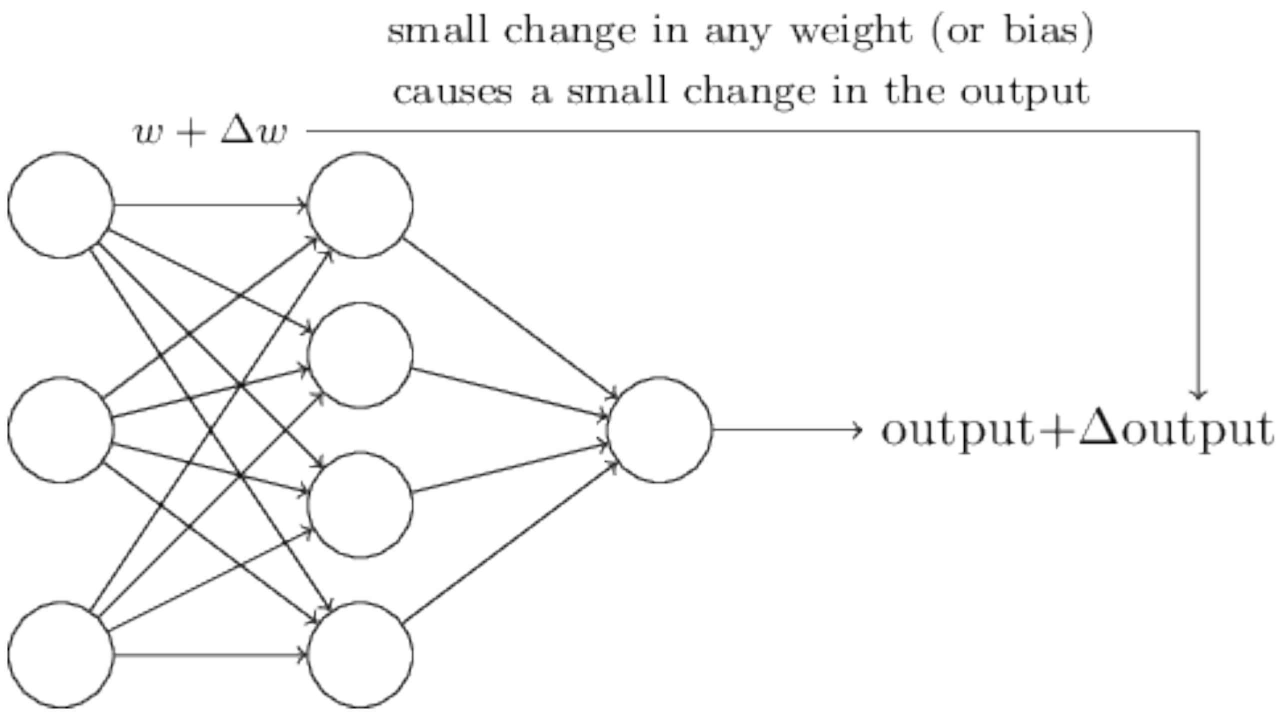
Limitation of a perceptron

A perceptron cannot express the XOR formulation. The behavior exhibited by XOR is too complex for a single perceptron.

We therefore needs to put perceptron together, to form a network

But before looking at a network, we need to improve our perceptron

Making a network learn



Making a network learn

Suppose we are training a network to recognize hand written digit

=> 504192

If the network is mistakenly classifying an image as a “0” when it should a “9”, then we should find a small change in the weights and biases…

… so the network gets a little closer to make the right classification

The network would be learning

What about a network of perceptrons?

The problem, is that a network of *perceptrons cannot learn properly*

Because *a small change in the weights or bias* of any single perceptron can sometimes cause the output of that perceptrons *to completely flip* (0 to 1, or 1 to 0)

That flip may then cause the behavior of the rest of the network to completely change

So, while the “0” can now be classified correctly, the behavior on all the other images is likely to have changed

Sigmoid Neuron

It is difficult using the perceptron to *gradually* modify the weights and biases to get close to the desired behavior

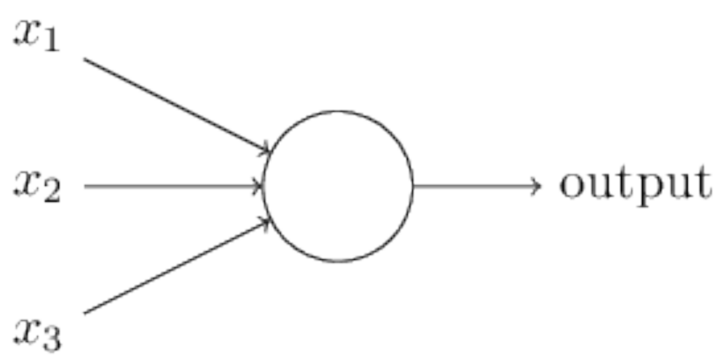
We will introduce a new type of artificial neuron, called a *sigmoid* neuron

A sigmoid neuron is similar to a perceptron, but:

- A small changes in its weight and bias cause only a small change in its output

That is a crucial fact which will allow a network of sigmoid neurons to learn

Sigmoid Neuron



Just like a perceptron, the sigmoid neuron has inputs

Just like a perceptron, the sigmoid neuron has inputs, x1, x2, …, xN

But instead of being 0 and 1, inputs can take any value between 0 and 1

Sigmoid Neuron

A sigmoid neuron has *a weight for each input and an overall bias*

The sigmoid function is $\sigma(z) = \dfrac{1}{1 + e^{-z}}$

The output of the sigmoid neuron is $\sigma(w \,.\, x + b)$

Sigmoid Neuron

To summarize, the output of a sigmoid neuron is

$$\frac{1}{1+e^{-z}} = \frac{1}{1+e^{-\sum_j w_j x_j - b}}$$

Well … that is different from a perceptron

Sigmoid neuron & perceptron

To understand the similarity we have $z = w \,.\, x + b$

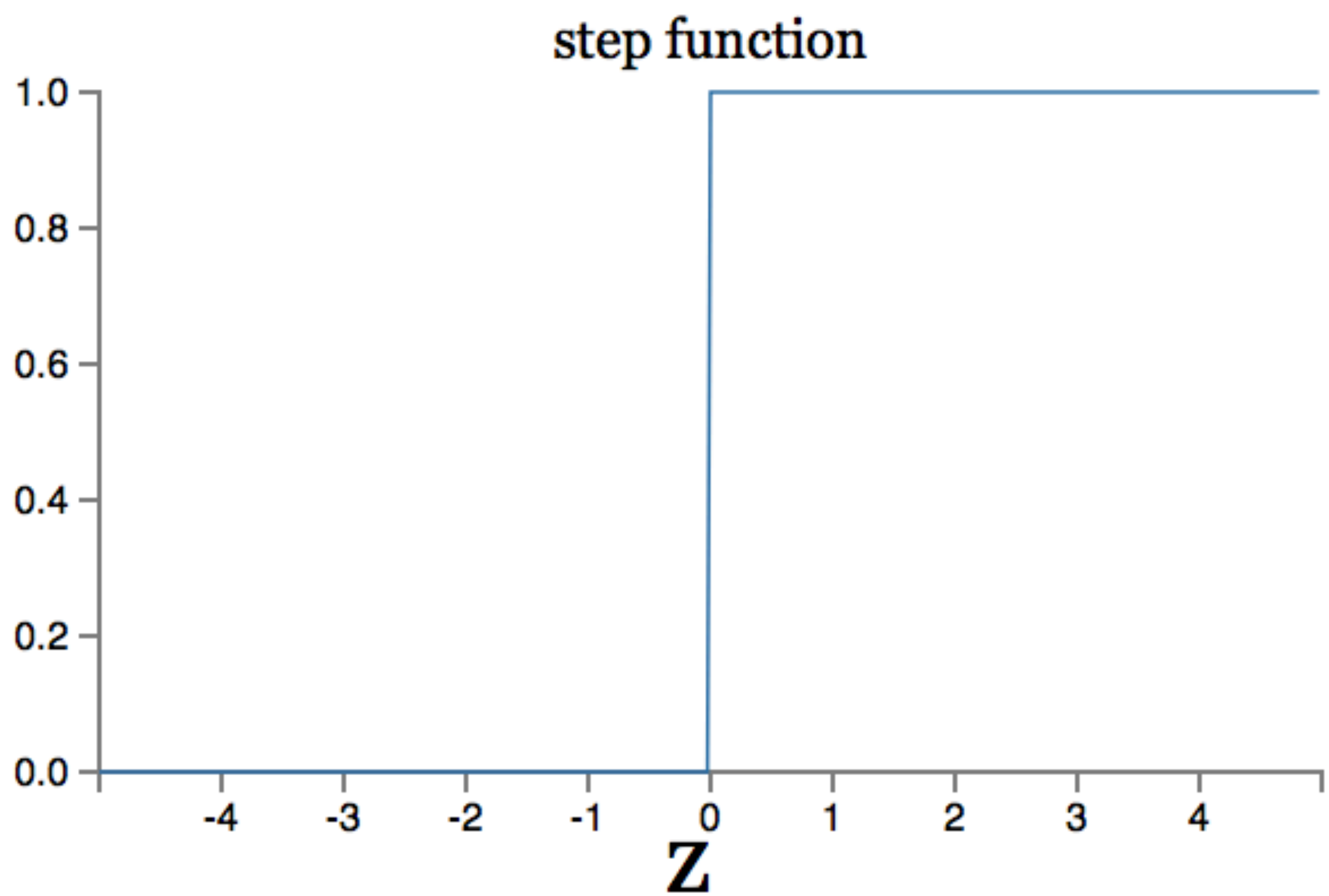
If z is large and > 0, then a sigmoid neuron is like a perceptron

If z is very negative, then the output is close to 0

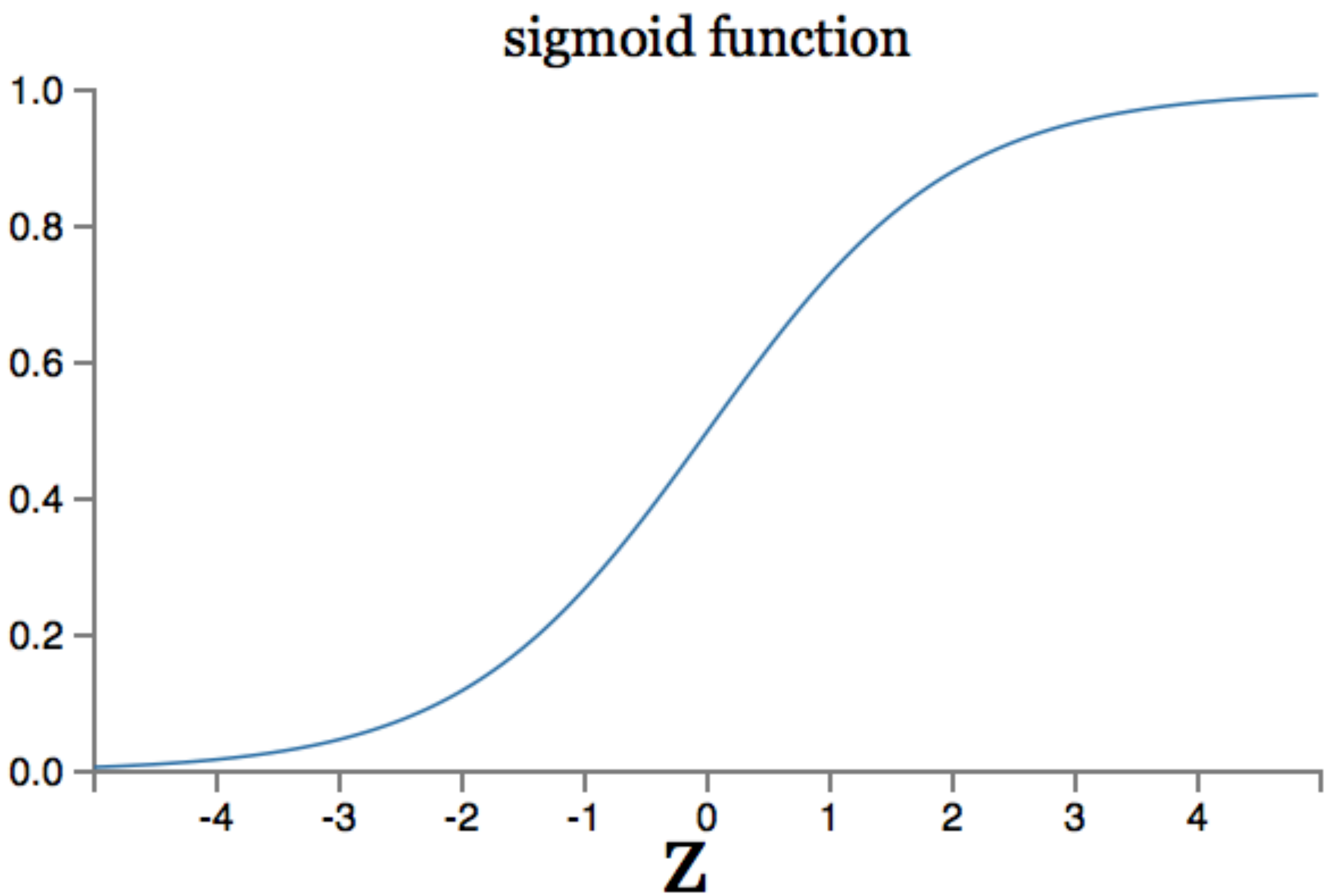
It is only when z is close to 0 that sigmoid neuron is different from perceptron

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

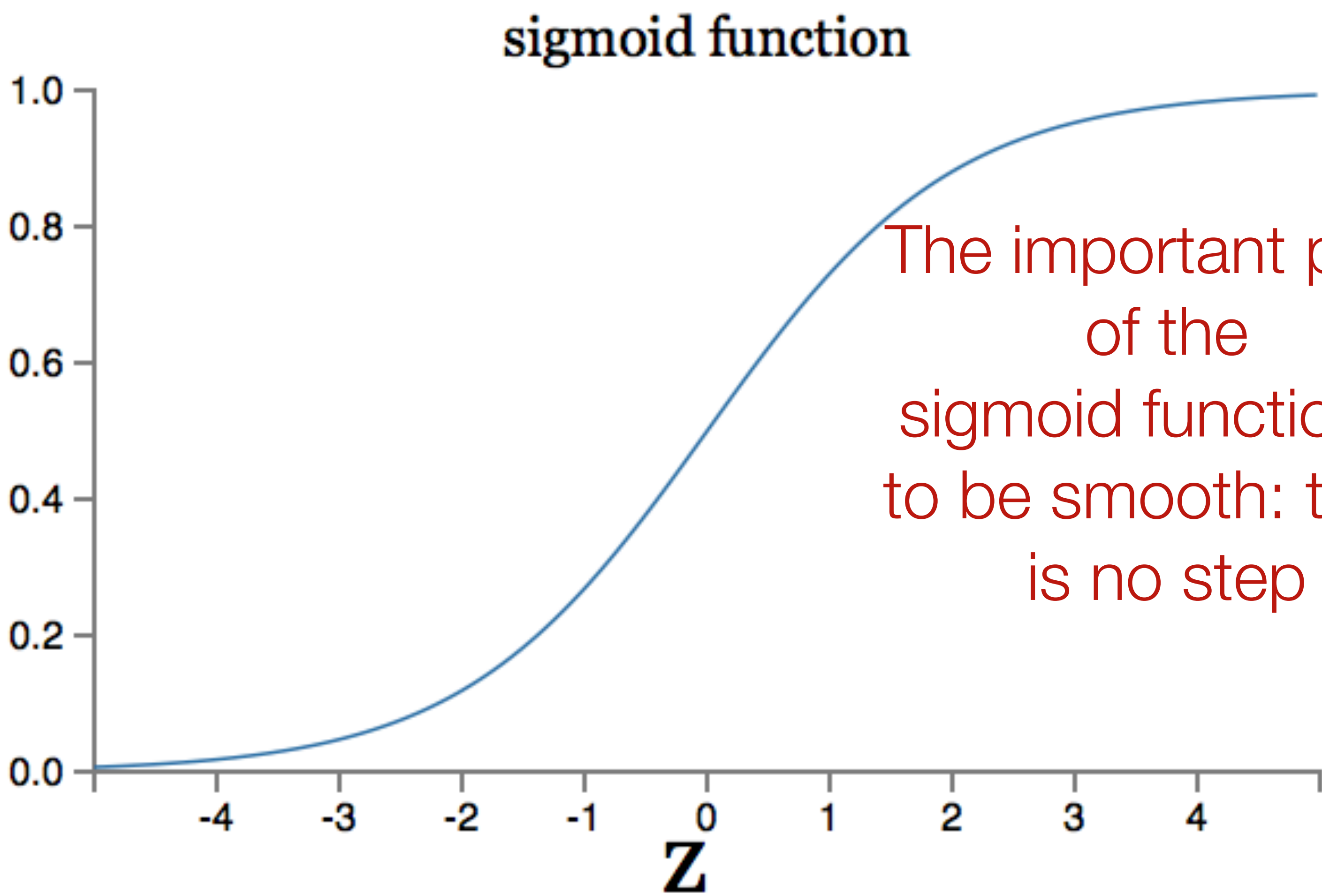
Perceptron



Sigmoid neuron

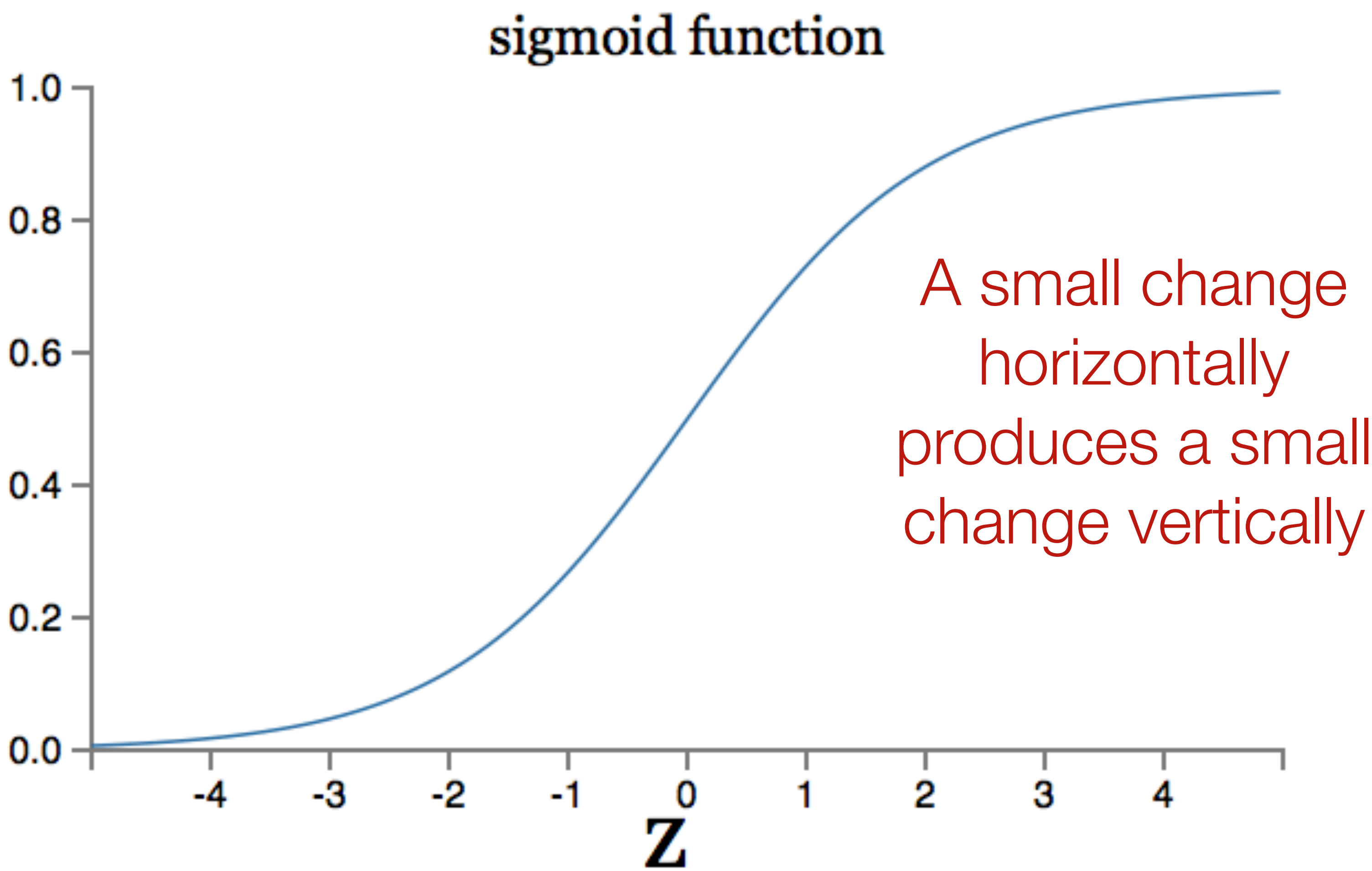


Sigmoid neuron



The important point of the sigmoid function is to be smooth: there is no step

Sigmoid neuron



Activation function

Actually, the exact shape of $\sigma(z)$ does not really matter

σ is called *an activation function* and many other functions are available

One important aspect of an activation function is the partial derivative

σ is commonly used in neural networks

Interpreting the output?

How should we interpret the output of a sigmoid function?

In perceptron, an output is either 0 or 1

With a sigmoid neuron, output could be any number between 0 and 1

How to express things like: “the input image is a 9”

Threshold is here again to the rescue

E.g., If the output > 0.5, …

Exercise

Implement a sigmoid neuron

Can a sigmoid neuron represent AND, OR, NAND logical gates?

Can a sigmoid neuron be used for learning the 2D point location example? What are the difference in terms of learning (i.e., how does the sigmoid neuron behaves with the perceptron learning algorithm)?

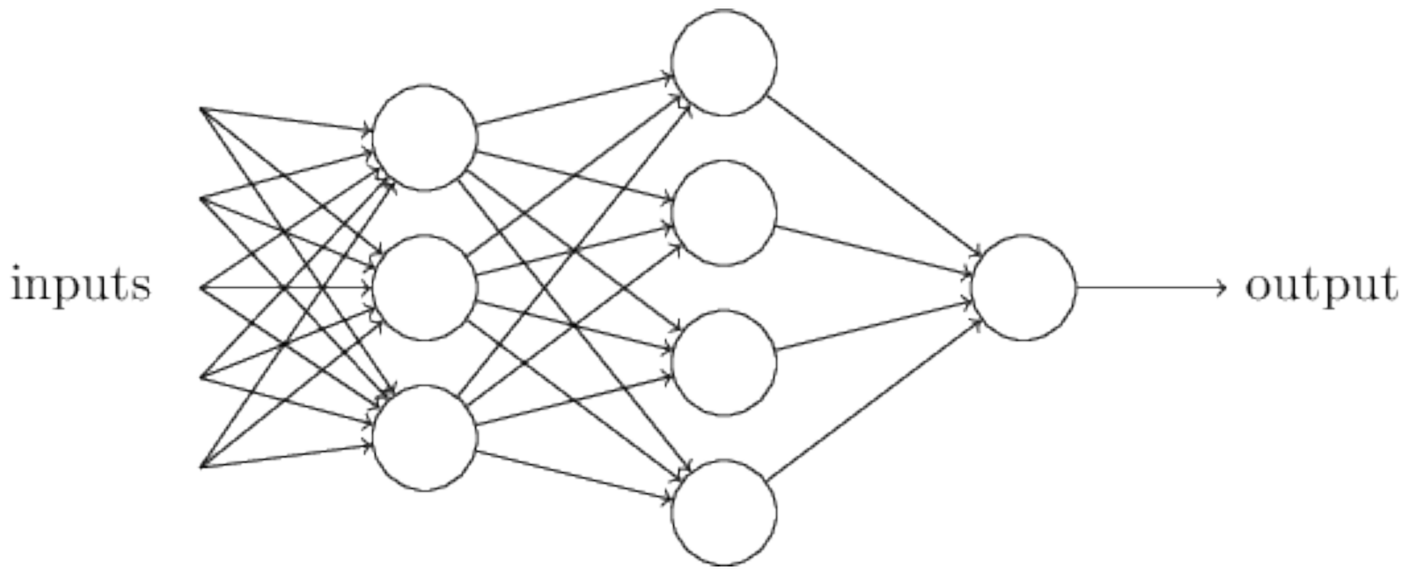
Outline

1. Sigmoid Neuron
2. **Neural Network**

CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

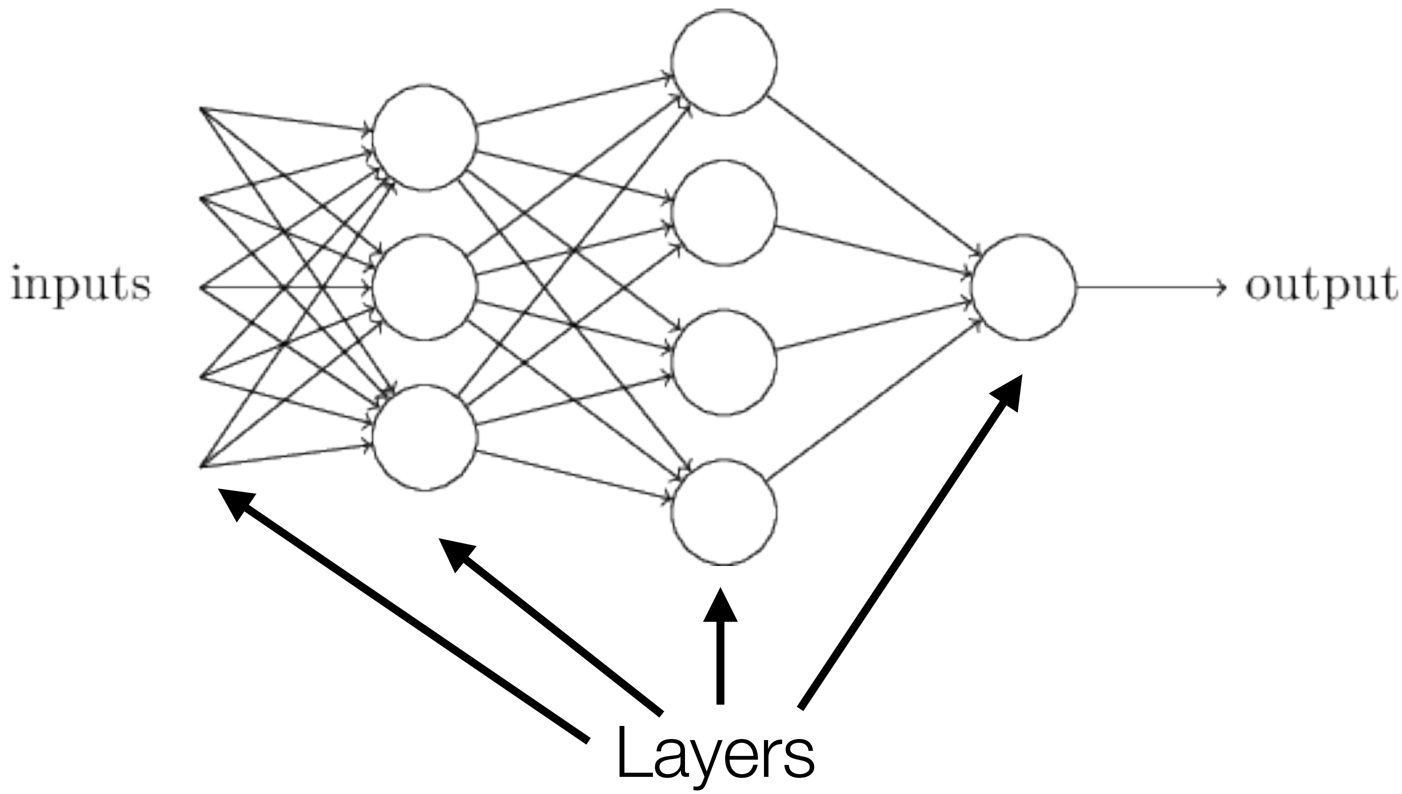
Network of neuron

A network has the following structure



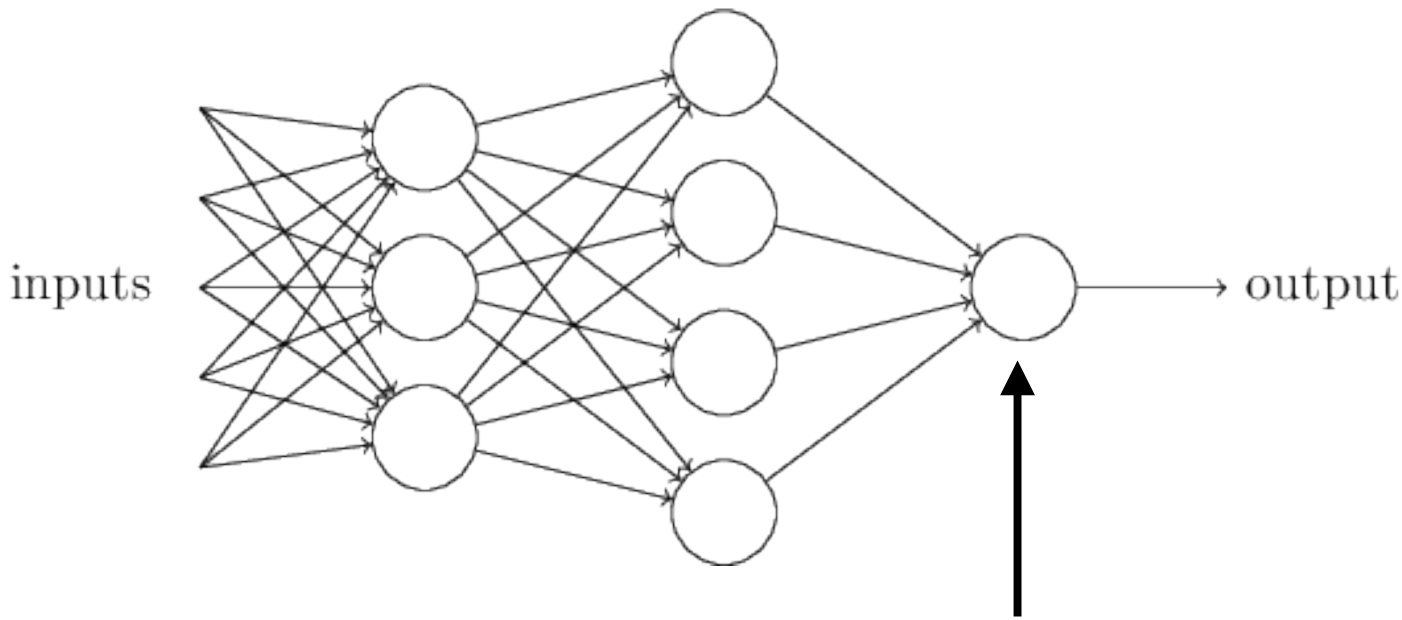
Network of neuron

A network has the following structure



Network of neuron

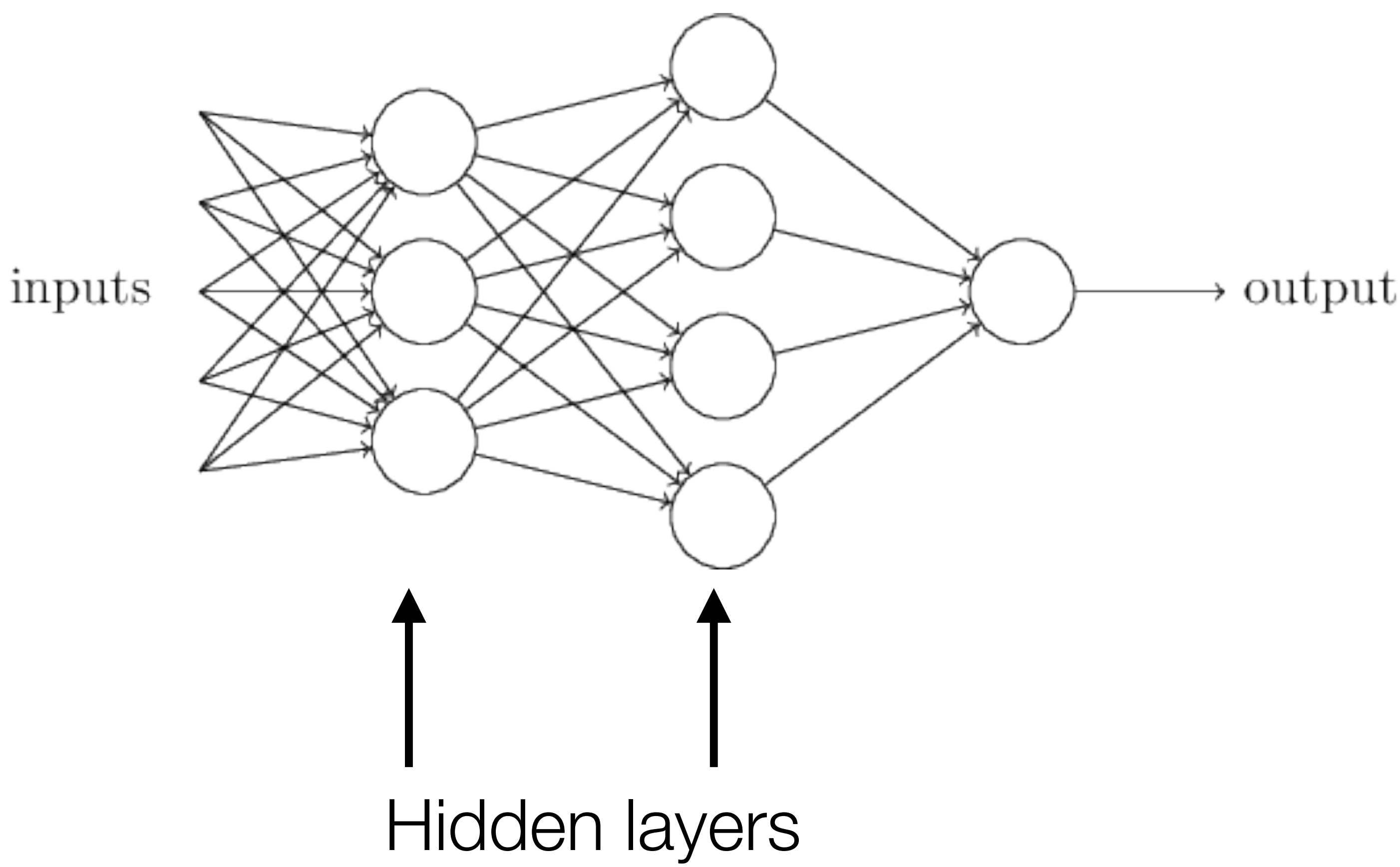
A network has the following structure



Output layer
(This example shows only one output neuron, but we could have many)

Network of neuron

A network has the following structure



Forward feeding network

When you provide inputs, data flows from the input layer to the output layer

When training information flow from the output layer to the input layer

There is no loop

Recurrent neural networks allow to have loop in the architecture. Output of a layer may be used as input in an earlier layer

Feeding

Operation that consists in providing a set of inputs to the network, and obtain a set of outputs

Consider L(n) the layer at position n

Output of L(n-1) is provided to the input of L(n)

The feeding process is often called as *forward feeding*

NeuralNetwork.py

```
# The 4 training examples by columns
X = np.array([[0, 0, 1, 1],
              [0, 1, 0, 1]])

# The outputs of the XOR for every example in X
Y = np.array([[0, 1, 1, 0]])

# No. of training examples
m = X.shape[1]
```

NeuralNetwork.py

```
# The 4 training examples by columns
X = np.array([[0, 0, 1, 1],
              [0, 1, 0, 1]])

# The outputs of the XOR for every example in X
Y = np.array([[0, 1, 1, 0]])

# No. of training examples
m = X.shape[1]
```

Input examples

Expected outputs

We have 4 examples

NeuralNetwork.py

```
# Set the hyperparameters
n_x = 2      #No. of neurons in first layer
n_h = 4      #No. of neurons in hidden layer
n_y = 1      #No. of neurons in output layer

#The number of times the model has to learn the dataset
number_of_iterations = 10000
learning_rate = 0.01

# define a model
trained_parameters = model(X, Y, n_x, n_h, n_y,
number_of_iterations, learning_rate)
```

DCC — CIENCIAS DE LA COMPUTACIÓN, UNIVERSIDAD DE CHILE

NeuralNetwork.py

```
# Set the hyperparameters
n_x = 2      #No. of neurons in first layer
n_h = 4      #No. of neurons in hidden layer
n_y = 1      #No. of neurons in output layer

#The number of times the model has to learn the dataset
number_of_iterations = 10000
learning_rate = 0.01

# define a model
trained_parameters = model(X, Y, n_x, n_h, n_y,
number_of_iterations, learning_rate)
```

The variable trained_parameters is all you need to make a prediction

NeuralNetwork.py

```
# Test 2X1 vector to calculate the XOR of its elements.
# You can try any of those: (0, 0), (0, 1), (1, 0), (1, 1)
X_test = np.array([[0], [1]])
y_predict = predict(X_test, trained_parameters)

# Print the result
print('Neural Network prediction for example ({:d}, {:d}) is 
{:d}'.format(
    X_test[0][0], X_test[1][0], y_predict))
```

Exercise

In u-cursos the complete code to build and train a small neural network is provided

You need:

1 - Run the code

2 - The provided code train a neural network for the XOR behavior. Verify that the network can learn the AND, OR

3 - Verify it can learn the NOT

4 - Produce a chart that indicates the cost vs iterations

License

dcc

CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

www.dcc.uchile.cl

/ DCCUCHILE