Session 1. Deep learning: Neurons and networks.

THIS DOCUMENT WAS BASED ON:

- http://neuralnetworksanddeeplearning.com/ BY MICHAEL NIELSEN
- https://www.youtube.com/watch?v=aircAruvnKk BY 3BLUE1BROWN

1) Introduction

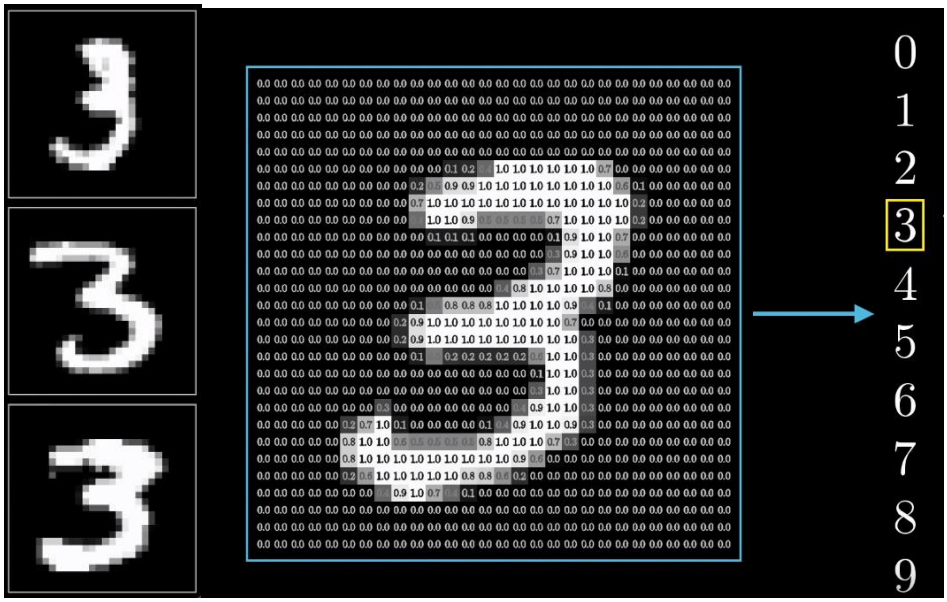
Aim

The aim of this document is to:

- Introduce the concept of neurons and neural networks, its working principle, composition and structure.
- Introduce the *learning* concept and understand the different parameters that play a role in the overall algorithm.
- Showcase the previous concepts with a practical example for handwritten numbers.

Motivation

Humans can automatically tell part one number from the other if a visual representation of it is given:

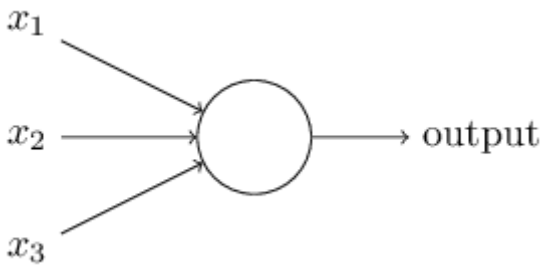


How would you program a computer to detect which number is represented in the image?

2) Artificial neurons: perceptrons and sigmoid neurons

Neural networks are composed by artificial neurons. The first model introduced to represent these neurons were the **perceptron**, back in 1950s-1960s. Nowadays more complex neurons are used, but perceptrons serve as a good first step to properly understand the way sigmoid neurons are defined.

A perceptron takes several binary inputs and gives a binary output: either 0 or 1.



The relationship between the inputs and the output is defined in terms of some weights and a threshold:

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

Where x_i is either 1 or 0.

Example:

Suppose the weekend is coming up, and you've heard that there's going to be a cheese festival in your city. You like cheese, and are trying to decide whether or not to go to the festival. You might make your decision by weighing up three factors:

1) *Is the weather good? (x_1)*
2) *Does your boyfriend or girlfriend want to accompany you? (x_2)*
3) *Is the festival near public transit? (You don't own a car). (x_3)*

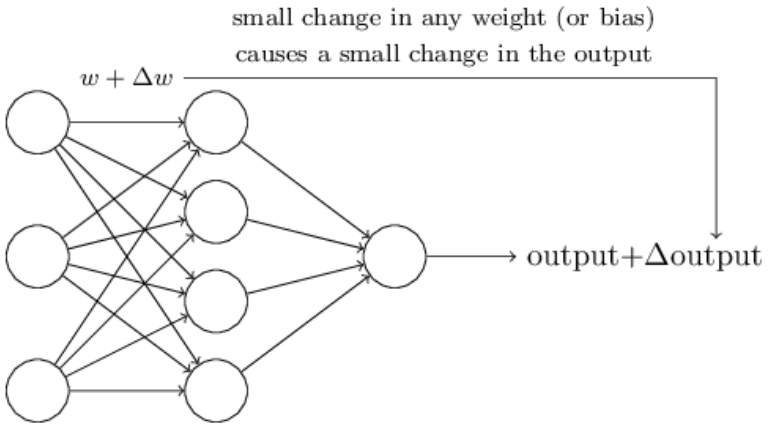
Depending on the importance given to each question, different weights could be assigned for each input. For instance, you could be so interested in going to the festival that you don't care if your girlfriend or boyfriend is coming with you, so $w_2 = 0$.

The threshold value determines the minimum value required for the neuron to activate. This is typically replaced by a **bias**, by moving it to the other side of the equation. The concept of the bias represents how easy it is to output a 1 for the neuron.

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

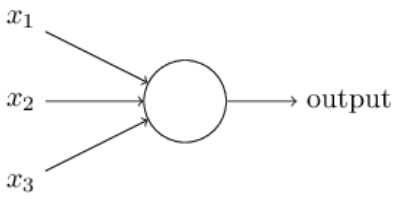
To see how learning might work, suppose we make a small change in some weight (or bias) in the network. What we'd like is for this small change in weight to cause only a small corresponding change in the output from the network. As we'll see in a moment, this property will make learning possible.

This is not what happens with perceptrons.

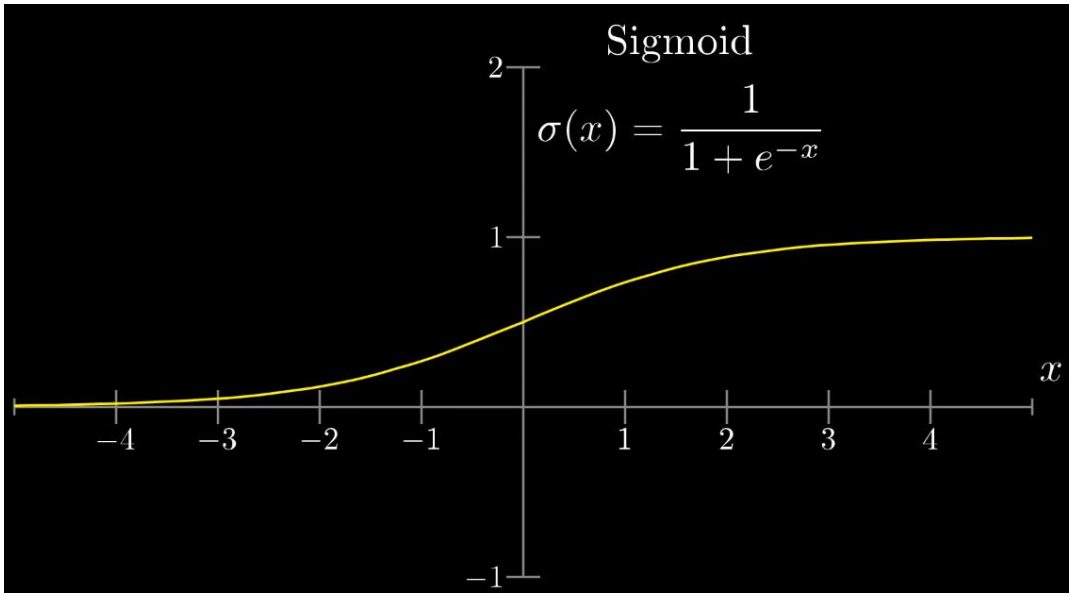


In fact, a small change in the weights or bias of any single perceptron in the network can sometimes cause the output of that perceptron to completely flip, say from 0 to 1. **This is where sigmoid neurons are introduced.**

Sigmoid neurons are similar to perceptrons, but modified so that small changes in their weights and bias cause only a small change in their output. The sigmoid neurons can take input values such as $x_i \in [0,\ 1]$



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

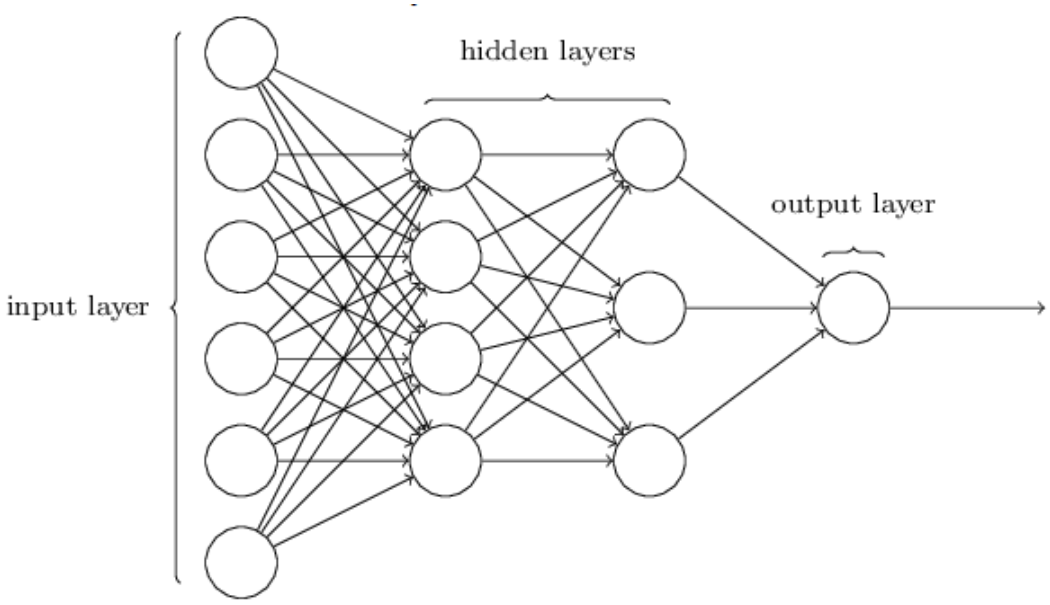


Explicitly, the output of a sigmoid neuron with inputs $x1$, $x2$,…, weights $w1, w2$,…, and bias b is

$$\sigma(z) = \frac{1}{1 + \exp(-\sum_j w_j x_j - b)}$$

3) Neural networks

- Structure:

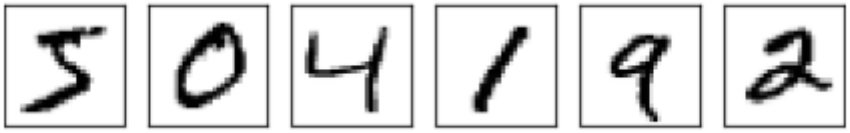


- The design of the **input** and **output** layers is often **straightforward.** For instance, for the handwritten number case, the input is the number of pixels of the image and the output is the total digits considered.
- The design of the **hidden layers** is not easily determined. Researchers have developed many design heuristics for the hidden layers, which may depend on several factors, such as the time required to train the network.
- The considered networks will always be **feedforward,** meaning that information is only transferred one way, forward. However, other networks design exist that contain

feedback loops, although they are not as popular since they are currently not as powerful as feedforward networks.

4) Training

Suppose we try to train a neural network to recognize handwritten digits:



To do so we need:

- Data to learn from -> MNIST data set for this example
- Cost function (mean squared error) that determines how well our network is recognizing each data input.
    - w -> Set of weights
    - b -> Set of bias
    - n -> number of training inputs
    - x -> Training input
    - y -> Desired input
    - a -> Output of the neural network

$$C(w,b) = \frac{1}{2n}\sum \|y(x) - a(x,w,b)\|^2$$

Of course, when a approximates $y(x)$, C will tend to 0. This

In order to update the weights and biases of the neural network, a **descent gradient** approach is used.

$$w_k \to w'_k = w_k - \eta \frac{\partial C}{\partial w_k}$$
$$b_l \to b'_l = b_l - \eta \frac{\partial C}{\partial b_l}.$$

Where the partial derivatives of the cost function with respect to the weights and biases is computed using **the backpropagation algorithm**, which will be covered in the next session. Here, η is called the **learning rate**.

Since the cost function has been defined in terms of an average over the number of training inputs, which has the form $C = \frac{1}{n}\sum_x C_x$, the gradient of the cost function requires to also compute the mean of the partial gradient for every sample , $\nabla C = \frac{1}{n}\sum_x \nabla C_x$

For this reason, an **stochastic gradient** is used, which works by picking only a set of random samples to compute the gradient, and use it as an approximation of the total one. If the random selected samples are denoted by $[X_1, X_2, \dots, X_m]$, we aim to get an approximation such that:

$$\frac{\sum_{j=1}^{m} \nabla C_{X_j}}{m} \approx \frac{\sum_x \nabla C_x}{n} = \nabla C,$$

So the final expression for the gradient is:

$$\nabla C \approx \frac{1}{m} \sum_{j=1}^{m} \nabla C_{X_j},$$

And applied to the case of the weights and biases:

$$w_k \to w_k' = w_k - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial w_k}$$

$$b_l \to b_l' = b_l - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial b_l},$$

Where only some selected samples X_m are used to train the new weights and biases.

5) Python code!

6) Interpreting results

Effect of η

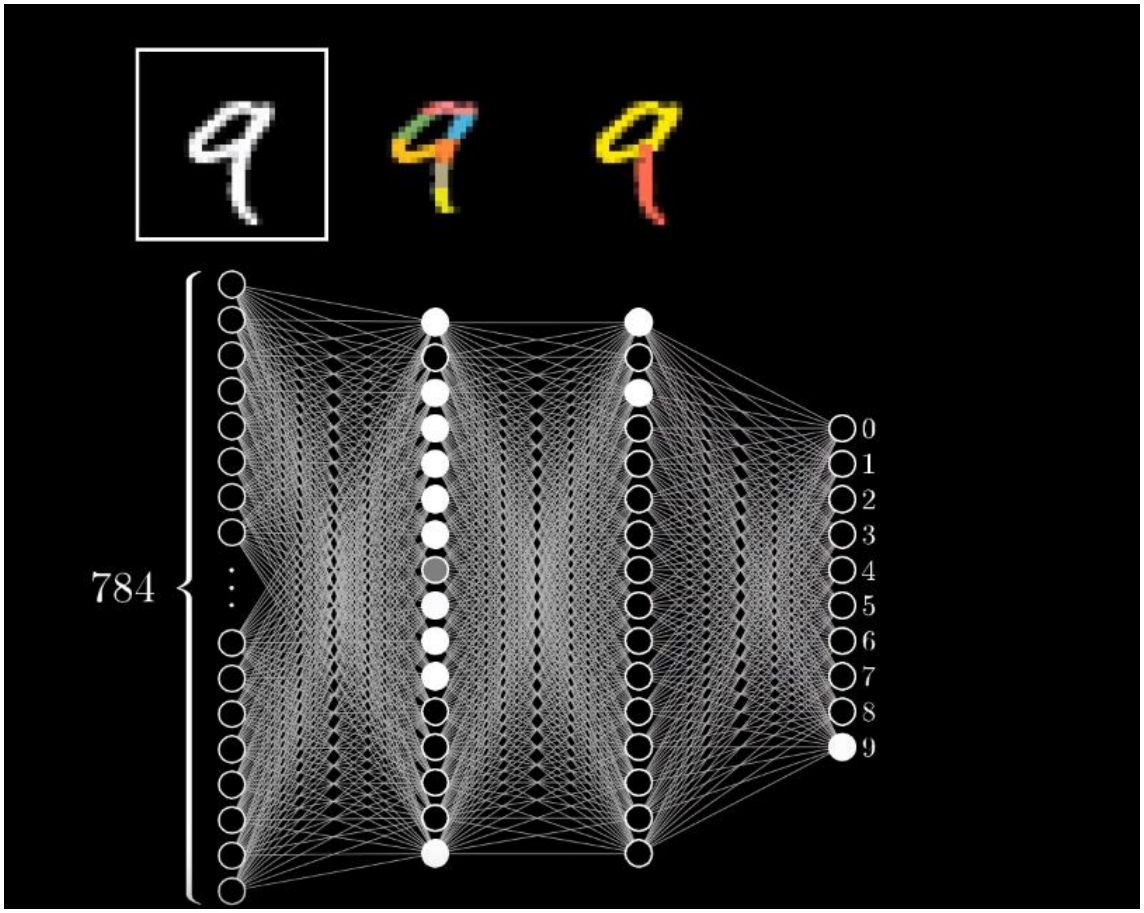
$\eta = 0.001$

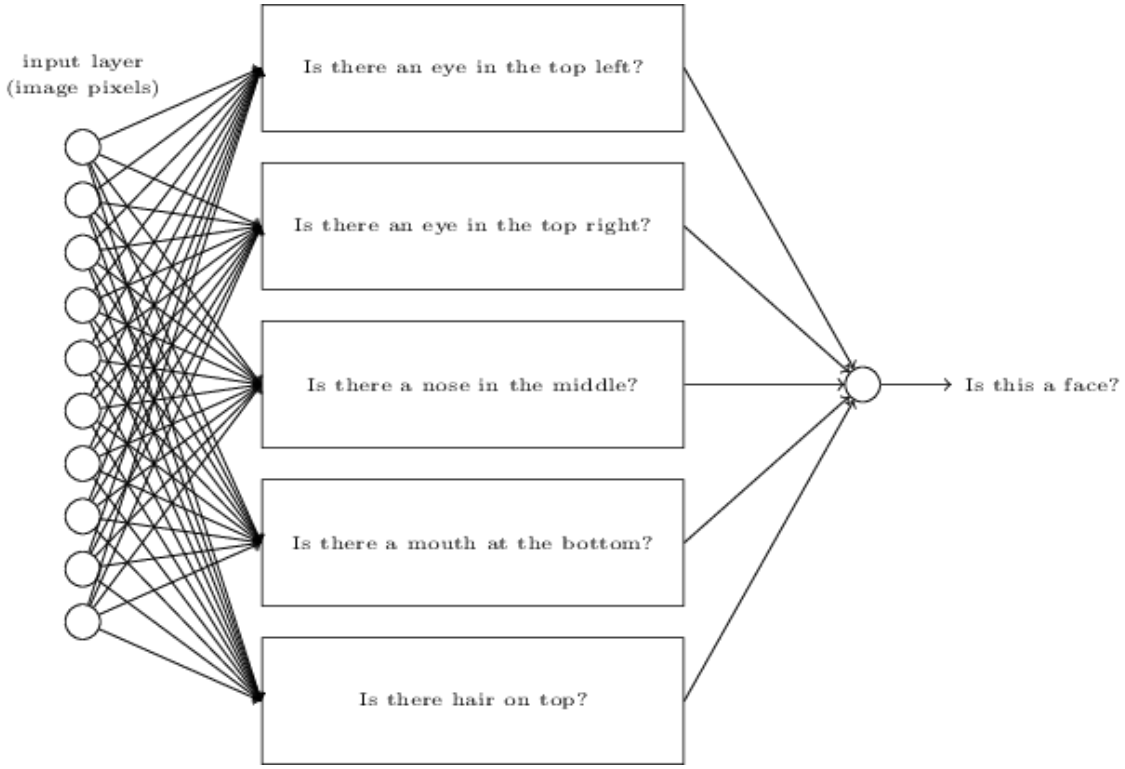
```
Epoch 0: 1139 / 10000
Epoch 1: 1136 / 10000
Epoch 2: 1135 / 10000
...
Epoch 27: 2101 / 10000
Epoch 28: 2123 / 10000
Epoch 29: 2142 / 10000
```

$\eta = 100$

```
Epoch 0: 1009 / 10000
Epoch 1: 1009 / 10000
Epoch 2: 1009 / 10000
...
Epoch 27: 982 / 10000
Epoch 28: 982 / 10000
Epoch 29: 982 / 10000
```

Role of single neurons?





Questions for next session

- Where does the SGD formula come from?
- Different types of neurons?
- Analytical formula of a (whole neural network function)?
- Does the cost function always go down?
- Is the problem convex? Does it depend on initial values?
- Output result interpretation (probabilities)
- Is there an upper or lower limit in the number of neurons?
- Machine learning = neural networks?