MAI4CAREU

Master programmes in Artificial Intelligence 4 Careers in Europe

University of Ruse

INTELLIGENT COMPUTER SYSTEMS

Svetlana Stefanova

September, 2022

Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

LECTURE 11

SHALLOW NEURAL NETWORKS

1. Introduction
2. Hidden layer
3. Mathematical modeling
4. Weights definition

Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

CONTENT 1

Definition

Under NN we imagine a structure consisting of many layers, including hidden ones. But there is also a type of NN with only a few hidden layers.

Shallow neural networks consist of only 1 hidden layer. Understanding them provides insight into exactly what is happening in a deep neural network.

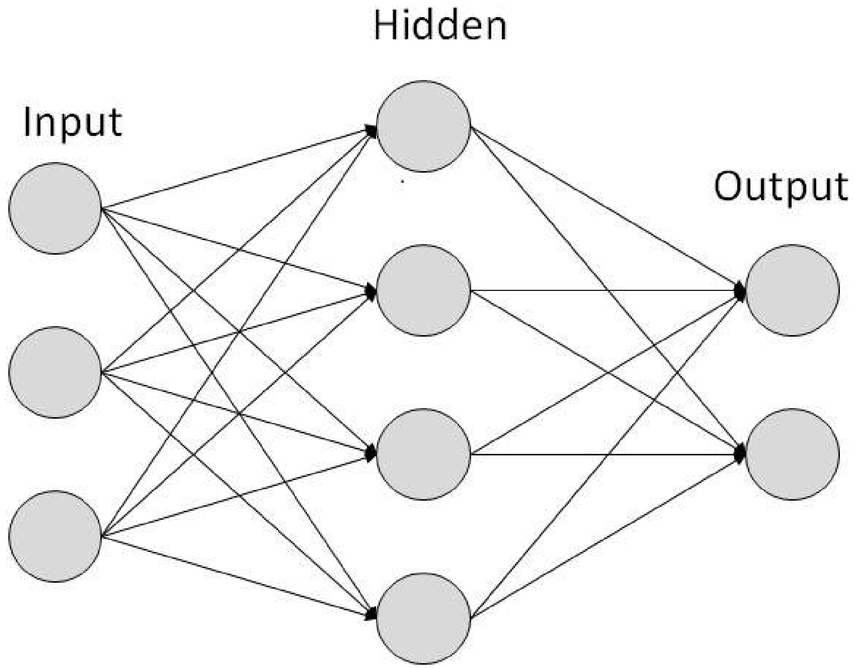
Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

CONTENT 1

Example

Shallow neural network with 1 hidden layer, 1 input layer and 1 output layer.



This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

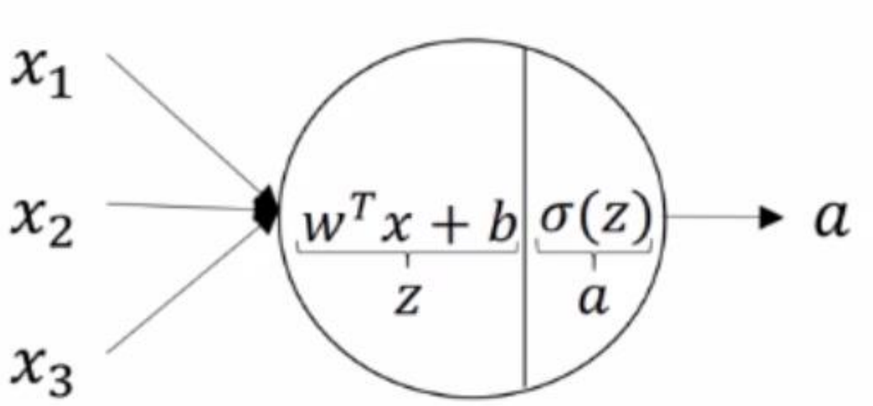
CONTENT 1

Model of a neuron

The neuron is the building block of the NN. Given an input, it provides the output and passes that output as input to the next layer. A neuron can be seen as a combination of 2 parts:

The first part - computes the output **z** by using the inputs and their weights.

The second part - performs the activation of **z** to give the final output **a** of the neuron.



Co-financed by the European Union
Connecting Europe Facility

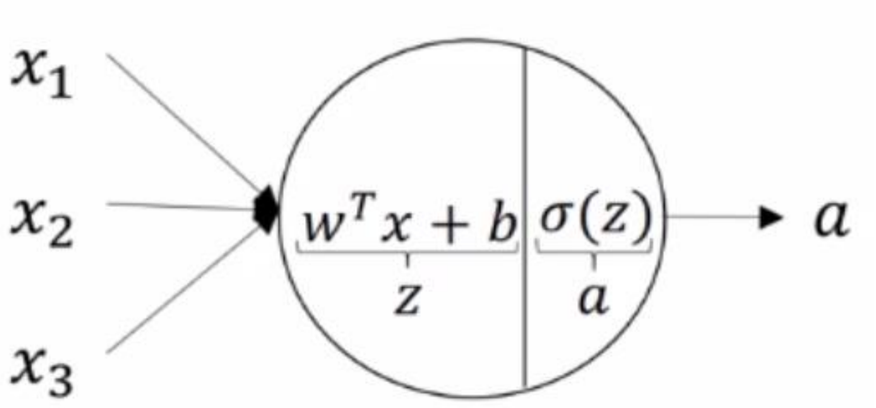
This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Hidden layer

The hidden layer consists of different neurons, each of which performs the 2 consecutive calculations.

Let's have 4 neurons in the hidden layer of our shallow NN computing the following:

$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}, a_1^{[1]} = \sigma\left(z_1^{[1]}\right)$$
$$z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]}, a_2^{[1]} = \sigma\left(z_2^{[1]}\right)$$
$$z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]}, a_3^{[1]} = \sigma\left(z_3^{[1]}\right)$$
$$z_4^{[1]} = w_4^{[1]T} x + b_4^{[1]}, a_4^{[1]} = \sigma\left(z_4^{[1]}\right)$$

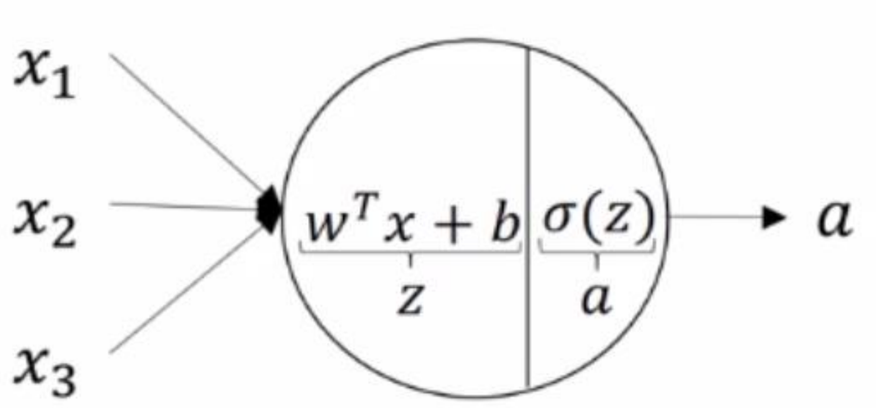


This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Hidden layer

In the above equations,

- the above index *[i]* denotes the layer number, and the index number *j* denotes the neuron number in a particular layer.
- ***X*** is the input vector consisting of 3 features.
- ***W [i] j*** is the weight associated with neuron ***j***, present in layer ***i***.
- ***b [i] j*** is the bias associated with neuron ***j***, present in layer ***i***

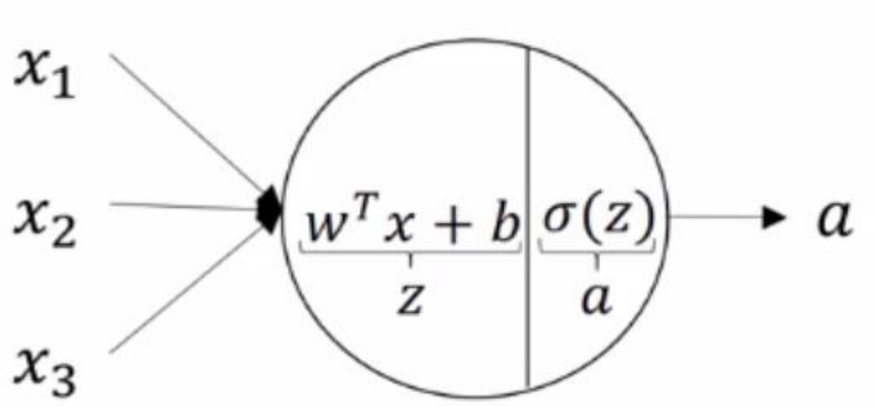


This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Hidden layer

- $Z^{[i]}_j$ is the intermediate output associated with neuron ***j***, present in layer ***i***.
- $a^{[i]}_j$ is the final output associated with neuron ***j***, present in layer ***i***.
- σ is the activation function (e.g. sigmoid). Mathematically it is defined as:

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



This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Hidden layer

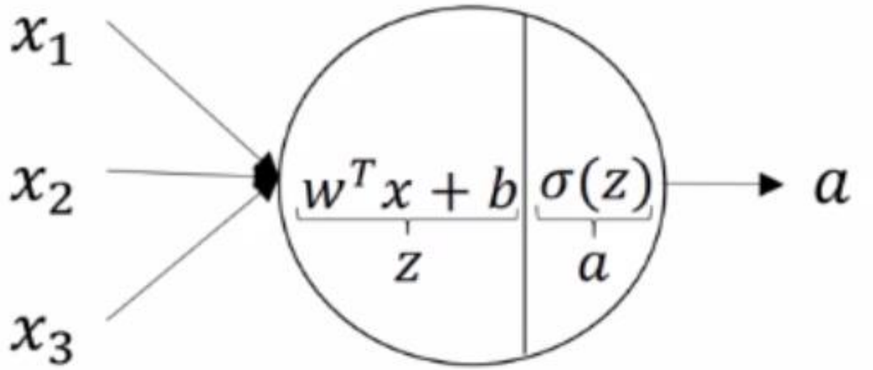
We can vectorize the equations:

The first equation calculates all intermediate outputs ***Z*** in a single matrix multiplication.

The second equation calculates all activations ***A*** in a single matrix multiplication.

$$Z^{[1]} = W^{[1]T} X + b^{[1]}$$

$$A^{[1]} = \sigma\left(Z^{[1]}\right)$$



This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

CONTENT 3

Forward propagation equations

- The 1st equation calculates the intermediate output ***Z*** $^{[1]}$ of the first hidden layer.
- The 2nd equation calculates the final output ***A*** $^{[1]}$ of the first hidden layer.
- The 3rd equation calculates the intermediate output ***Z*** $^{[2]}$ of the output layer.
- The 4th equation calculates the final output ***A*** $^{[2]}$ of the output layer, which is also the final output of the entire neural network.

$$Z^{[1]} = W^{[1]T} X + b^{[1]}$$

$$A^{[1]} = \sigma\left(Z^{[1]}\right)$$

$$Z^{[2]} = W^{[2]T} A^{[1]} + b^{[2]}$$

$$\hat{y} = A^{[2]} = \sigma\left(Z^{[2]}\right)$$

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

CONTENT 3

Activation function

NN is mainly a set of mathematical equations and weights. To make it stable so that it performs well in different scenarios, we use activation functions. They introduce non-linear properties into the NN.

Let's try to understand why activation functions are crucial for any NN using the shallow neural network.

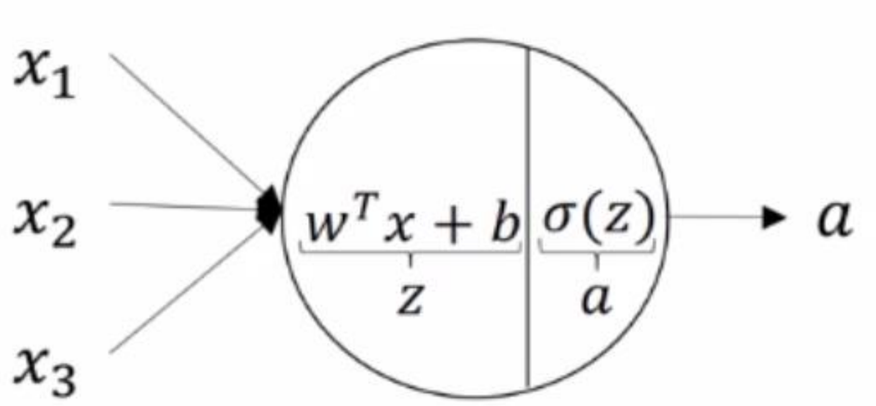
Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

CONTENT 3

Activation function

Without the activation functions, our shallow neural network can be represented:



$$Z^{[1]} = W^{[1]T} X + b^{[1]}$$

$$\hat{y} = Z^{[2]} = W^{[2]T} Z^{[1]} + b^{[2]}$$

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

CONTENT 3

Activation function

If we substitute the value of $Z^{[1]}$ from equation 1 into equation 2, we get the following:

$$Z^{[1]} = W^{[1]T}X + b^{[1]}$$
$$\hat{y} = Z^{[2]} = W^{[2]T}W^{[1]T}X + W^{[2]T}b^{[1]} + b^{[2]}$$
$$\hat{y} = Z^{[2]} = W_{new}X + b_{new}$$

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

CONTENT 3

Activation function

As we can see, the output will become a linear combination of a new weight matrix ***W***, with an input ***X*** and new bias ***b***, which means that the neurons present in the hidden layer and their weights remain irrelevant.

Therefore:

- to introduce non-linearity into the network we need to use the activation functions;
- it is not necessary to use one activation function for all layers.

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

CONTENT 4

Initialization of weights

The weight matrix ***W*** of NN is initialized **randomly**.

Why it cannot be initialized to 0 or some specific value? – let us figure this out using the shallow neural network.

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

CONTENT 4

Initialization of weights

Let

W^1 - the matrix of weights of layer 1 and

W^2 - the matrix of weights of layer 2

are initialized to 0 or some other value.

If the weight matrices are the same - the activation of neurons in the hidden layer will be the same. Also, the derivatives of the activations would be the same. Therefore, the neurons in the hidden layer will modify the weights in a similar way, i.e. it will not matter if there is more than 1 neuron in one hidden layer.

We do not want that - we want each neuron in the hidden layer to be unique, have a different weight, and work as a unique function. So we initialize the weights randomly.

Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

CONTENT 4

Xavier’s method

The best initialization method is Javier's method. Mathematically, it is defined as:

$$W^{[l]} \sim \mathcal{N}\left(\mu = 0, \sigma^2 = \frac{1}{n^{[l-1]}}\right)$$

$$b^{[l]} = 0$$

It states that the weight matrix ***W*** of a particular layer ***l*** is randomly selected from a normal distribution with value ***μ = 0*** and variance ***σ² = the multiplicative inverse of the number of neurons in layer l − 1***.

The bias ***b*** of all layers is initialized with **0**.

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

CONTENT 4

Gradient descent

We already know that the neural network weights are initialized randomly.

To use the neural network for correct predictions, we need to update these weights.

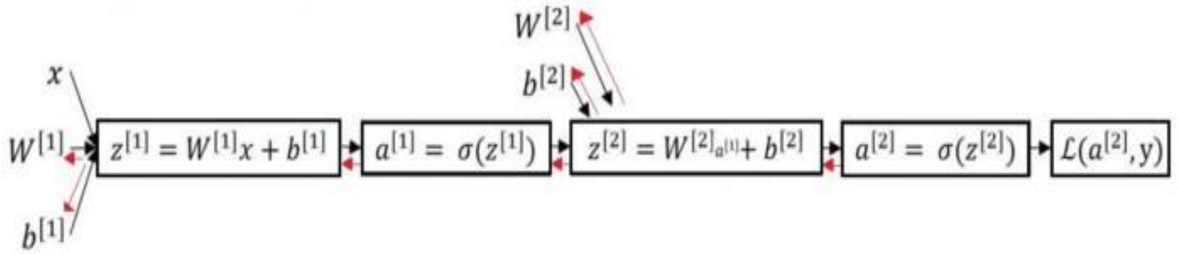
The method by which we update the weights is known as **gradient descent**.

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

CONTENT 4

Gradient descent

- **Forward propagation** (the **black** lines) - used to calculate the output for a given input ***X***.
- **Backward propagation** (the red lines) - used to update the weight matrices $W^{[1]}, W^{[2]}$ and biases $b^{[1]}, b^{[2]}$. It is done by calculating derivatives of the inputs at each step.



This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

CONTENT 4

Result error

The error ***L*** is defined mathematically:

$$L(\hat{y}, y) = -[y \log \hat{y} + (1 - y) \log (1 - \hat{y})]$$

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

CONTENT 4

Backward propagation

Using the error equation ***L*** and a sigmoid activation function of the hidden and output layers, using a chain rule of derivatives, we calculate:

$$dA^{[2]} = \frac{\delta L\left(A^{[2]},Y\right)}{\delta A^{[2]}} = \frac{-Y}{A^{[2]}} + \frac{1-Y}{1-A^{[2]}}$$

$$dZ^{[2]} = \frac{\delta L\left(A^{[2]},y\right)}{\delta Z^{[2]}} = \frac{\delta L\left(A^{[2]},y\right)}{\delta A^{[2]}} * \frac{\delta A^{[2]}}{\delta Z^{[2]}} = A^{[2]} - Y$$

$$dW^{[2]} = \frac{\delta L\left(A^{[2]},y\right)}{\delta W^{[2]}} = \frac{\delta L\left(A^{[2]},y\right)}{\delta Z^{[2]}} * \frac{\delta Z^{[2]}}{\delta W^{[2]}} = dZ^{[2]}A^{[1]T}$$

$$db^{[2]} = \frac{\delta L\left(A^{[2]},y\right)}{\delta b^{[2]}} = \frac{\delta L\left(A^{[2]},y\right)}{\delta Z^{[2]}} * \frac{\delta Z^{[2]}}{\delta b^{[2]}} = dZ^{[2]}$$

$$dA^{[1]} = \frac{\delta L\left(A^{[2]},Y\right)}{\delta A^{[1]}} = \frac{\delta L\left(A^{[2]},Y\right)}{\delta Z^{[2]}} * \frac{\delta Z^{[2]}}{\delta A^{[1]}} = dZ^{[2]}W^{[2]}$$

$$dZ^{[1]} = \frac{\delta L\left(A^{[2]},y\right)}{\delta Z^{[1]}} = \frac{\delta L\left(A^{[2]},y\right)}{\delta A^{[1]}} * \frac{\delta A^{[1]}}{\delta Z^{[1]}} = W^{[2]T}dZ^{[2]} * \sigma'\left(Z^{[1]}\right)$$

$$dW^{[1]} = \frac{\delta L\left(A^{[2]},y\right)}{\delta W^{[1]}} = \frac{\delta L\left(A^{[2]},y\right)}{\delta Z^{[1]}} * \frac{\delta Z^{[1]}}{\delta W^{[1]}} = dZ^{[1]}X^{T}$$

$$db^{[1]} = \frac{\delta L\left(A^{[2]},y\right)}{\delta b^{[1]}} = \frac{\delta L\left(A^{[2]},y\right)}{\delta Z^{[1]}} * \frac{\delta Z^{[1]}}{\delta b^{[1]}} = dZ^{[1]}$$

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

MAI4CAREU

Master programmes in Artificial Intelligence 4 Careers in Europe

Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423