Introduction to Deep Learning

Franciszek Górski
Gradient Science Club 2023

Plan for Today

- What is Deep Learning?
- Artificial neuron
- Activation function
- Artificial neural network
- Forward propagation
- Backward propagation

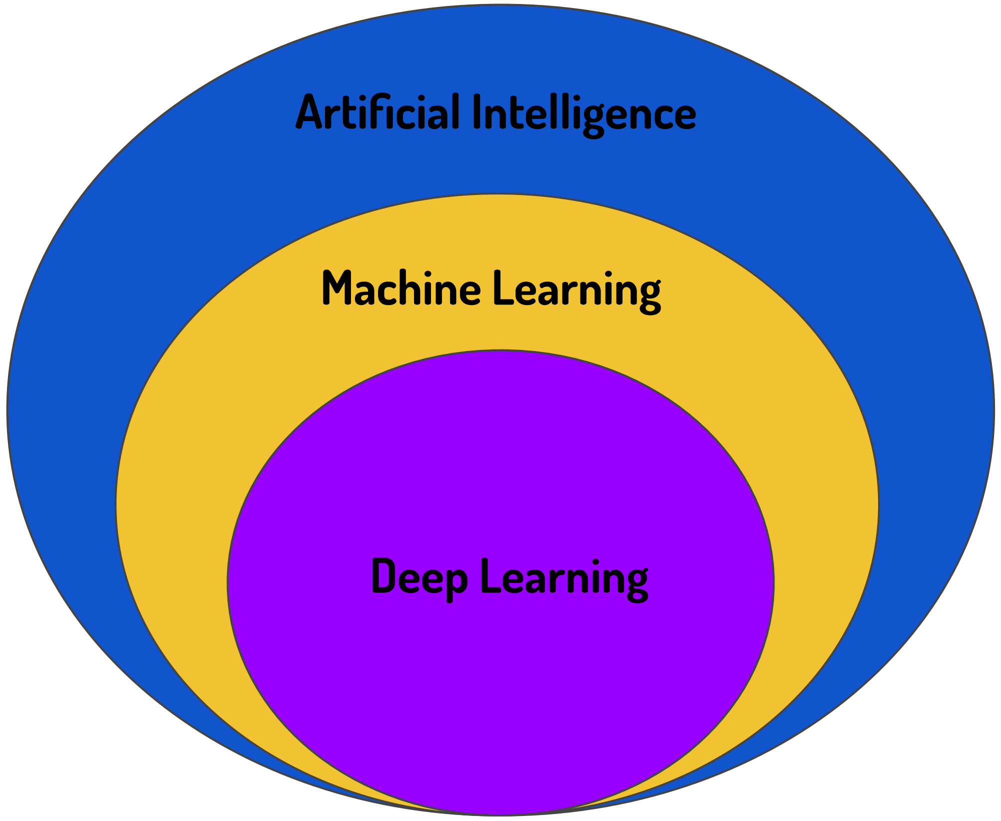
Resources

- Introduction to the mechanisms behind Neural Networks: https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
- Andrew Ng introduction to Deep Learning: https://www.youtube.com/playlist?list=PLkDaE6sCZn6Ec-XTbcX1uRg2_u4xOEky0
- MIT full course on Deep Learning: https://www.youtube.com/playlist?list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI

What is Deep Learning?

Deep learning

- It’s a subdomain of machine learning
- It focus on building and training of **deep** neural networks
- Most of today's breakthroughs in artificial intelligence have come through the development of deep learning



Artificial Neuron

Artificial Neuron

Artificial neuron is an elementary unit of a bigger structure which we called artificial neural network.

It is designed to mimic aspects of its biological counterparts.

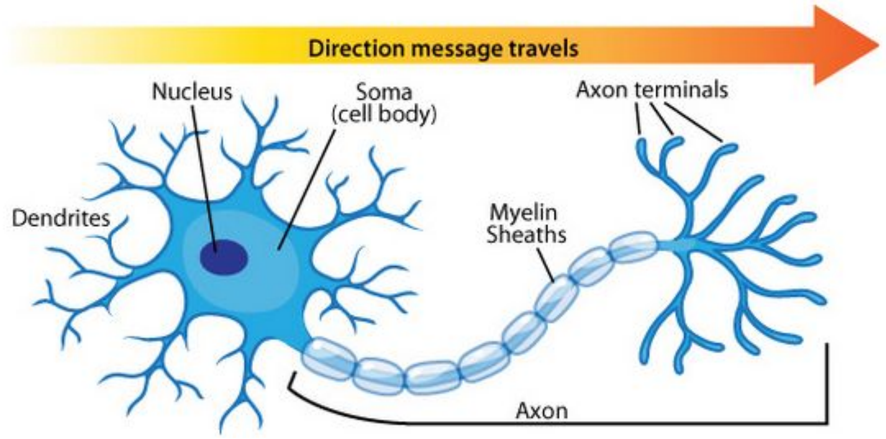
However a significant performance gap exists between biological and artificial neurons.

Biological Neuron

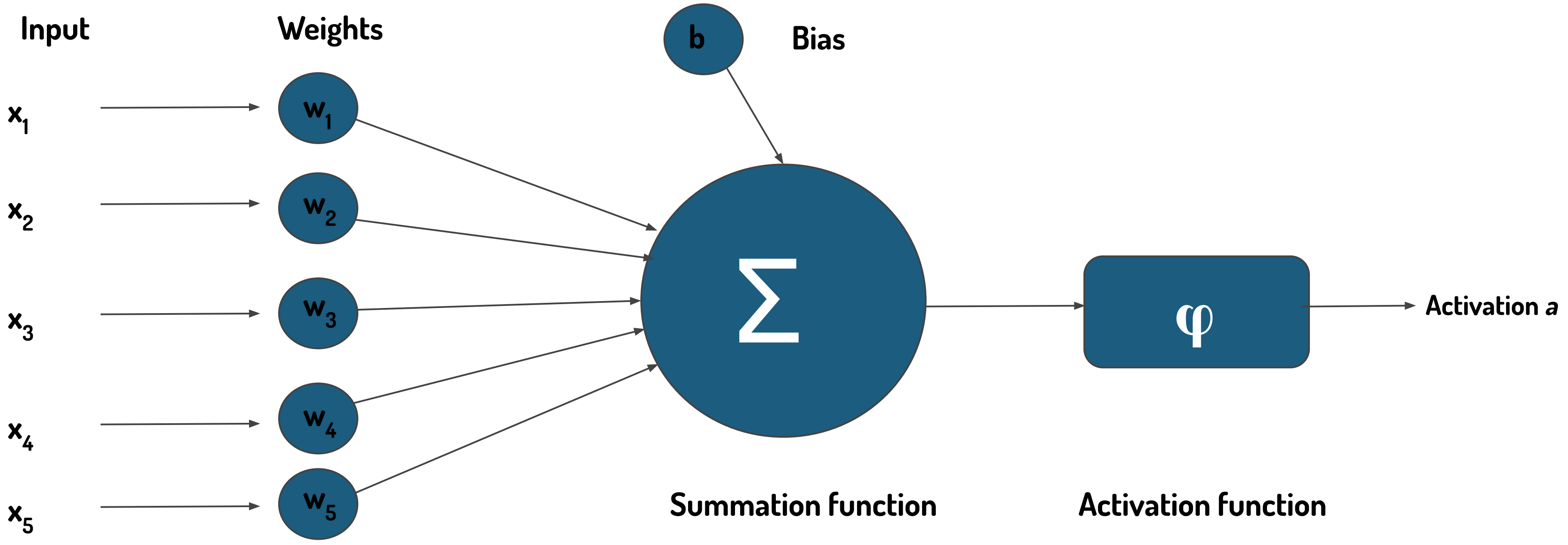
Biological neuron consists of:

- **Dendrites** - that act as the input vector to the neuron and transmit signals from the neighbouring neurons. We can say that each dendrite performs a multiplication of a signal and its weight which could be positive or negative.
- **Soma** - acts as a summation function of positive and negative signals propagated through the dendrites.
- **Axon** - it outputs a summed signals from the soma to the another neurons.

source:
https://askabiologist.asu.edu/neuron-anatomy



Artificial Neuron - Perceptron



Artificial Neuron

However, biological neurons prove ineffective at handling discrete signals. Because of that scientists came to idea of artificial neuron, which is a simpler version of biological neuron. It consists of:

- Input vector $\boldsymbol{x}$ or a which have weights ($\boldsymbol{w}_1$, $\boldsymbol{w}_2$, $\boldsymbol{w}_3$, ...) for each of inputs discrete values ($\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\boldsymbol{x}_3$, ...) or (a_1, a_2, a_3, ...)
- Summation function Σ
- Activation function φ

$$z_{lk} = \sum_{k=1}^{K_{l-1}} w_{l-1\,k} a_{l-1\,k} + b_{l-1\,k} \qquad a_{lk} = \varphi(z_{lk})$$

Activation function

Activation function

Activation function is used for activating an output from a neuron.

It scales the output to the fixed range of values, typically to the range $< 0; 1 >$. This range means activation (value 1) or deactivation (value 0) of a neuron's output.

Activation function

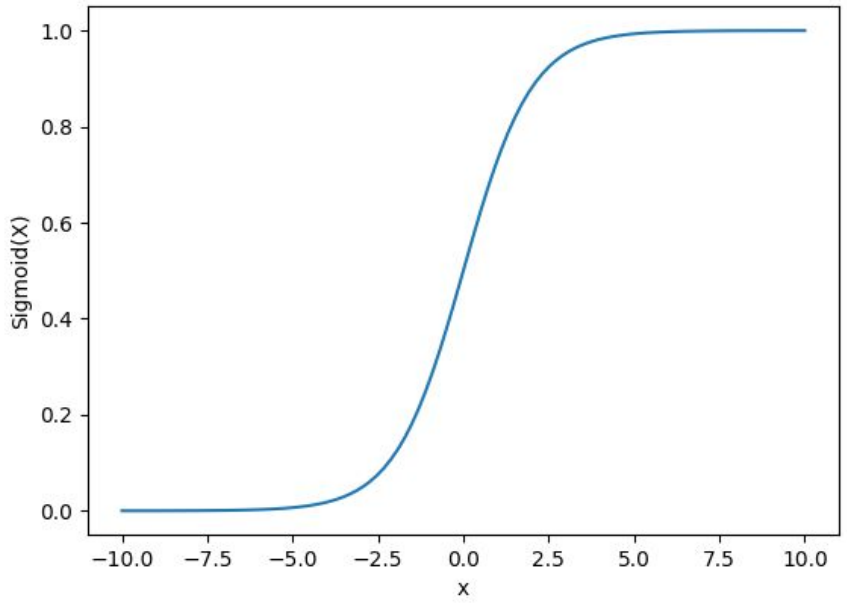
We can distinguish the following activation functions:

- Sigmoid
- Hyperbolic tangent
- ReLu

Activation function - sigmoid

Scale values to the range ⟨0;1⟩

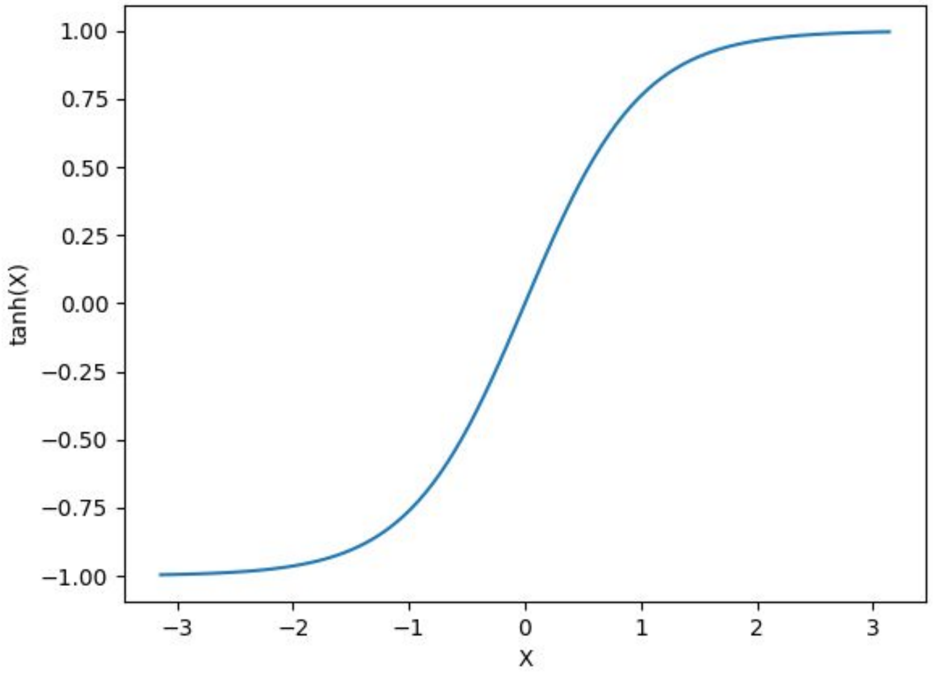
$$Sigmoid(x) = \frac{1}{1 + e^{-x}}$$



Activation function - hyperbolic tangent

Scale values to the range ⟨-1;1⟩

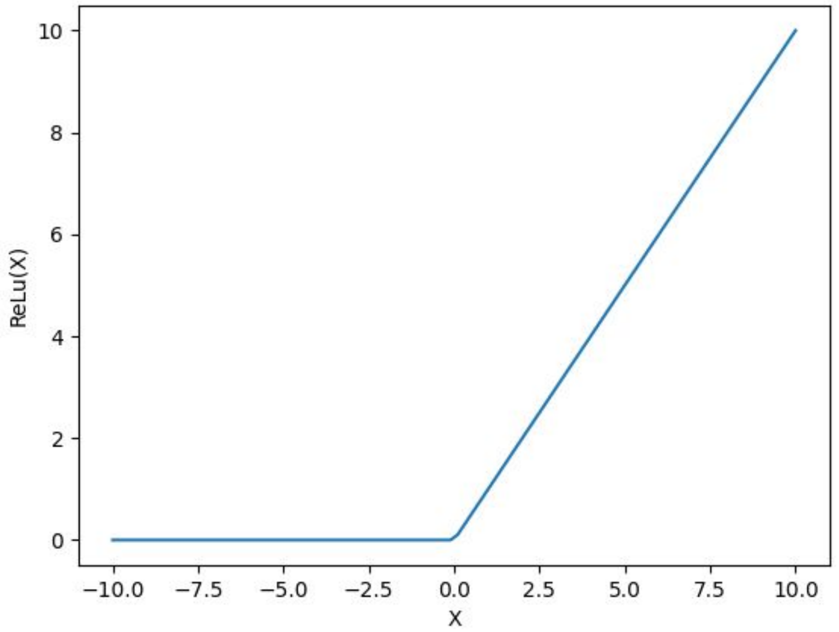
$$Tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$



Activation function - ReLu (Rectified Linear unit)

Scale values to the range ⟨0;x⟩

$$ReLu(x) = max(x, 0)$$



Why do we use activation functions at all?

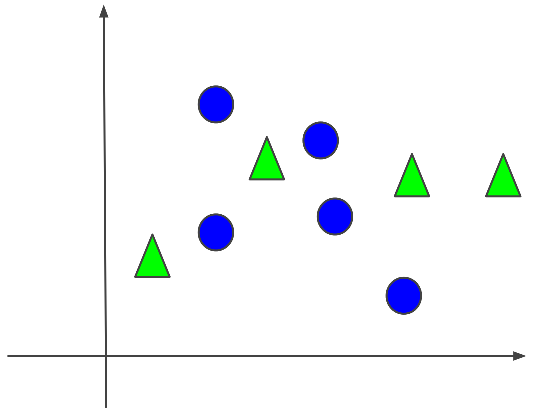
As we can see operations performed by neurons are linear combinations of inputs and weights.

It means that neurons output linear functions.

But are linear functions sufficient to solve complex problems?

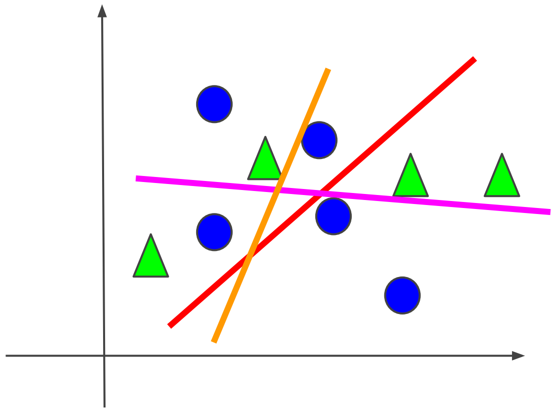
Why we use activation functions?

Let's take this example into consideration and try to solve it with linear function:



Why we use activation functions?

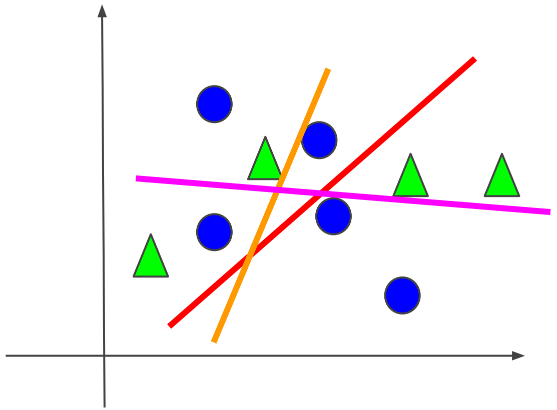
Let's take this example into consideration:



We can try many approaches ...

Why we use activation functions?

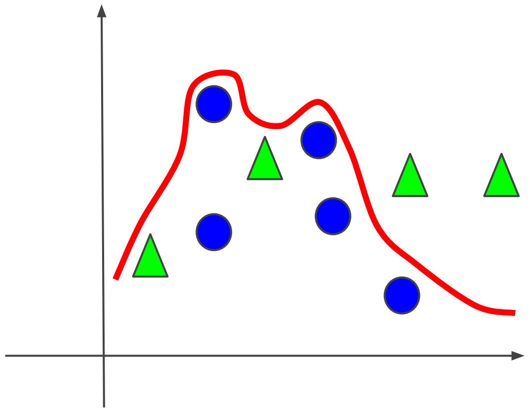
But none of them is satisfying



We can't solve this problem precisely with linear function!

Why we use activation functions?

This could be solved better with non-linear function!



Why we use activation functions?

Activation function introduce non-linearity into our model which helps us solved more precisely a complex problems.

Live is brutal ... and full of complex, nonlinear problems 😏.

Artificial Neural Network

Artificial Neural Network

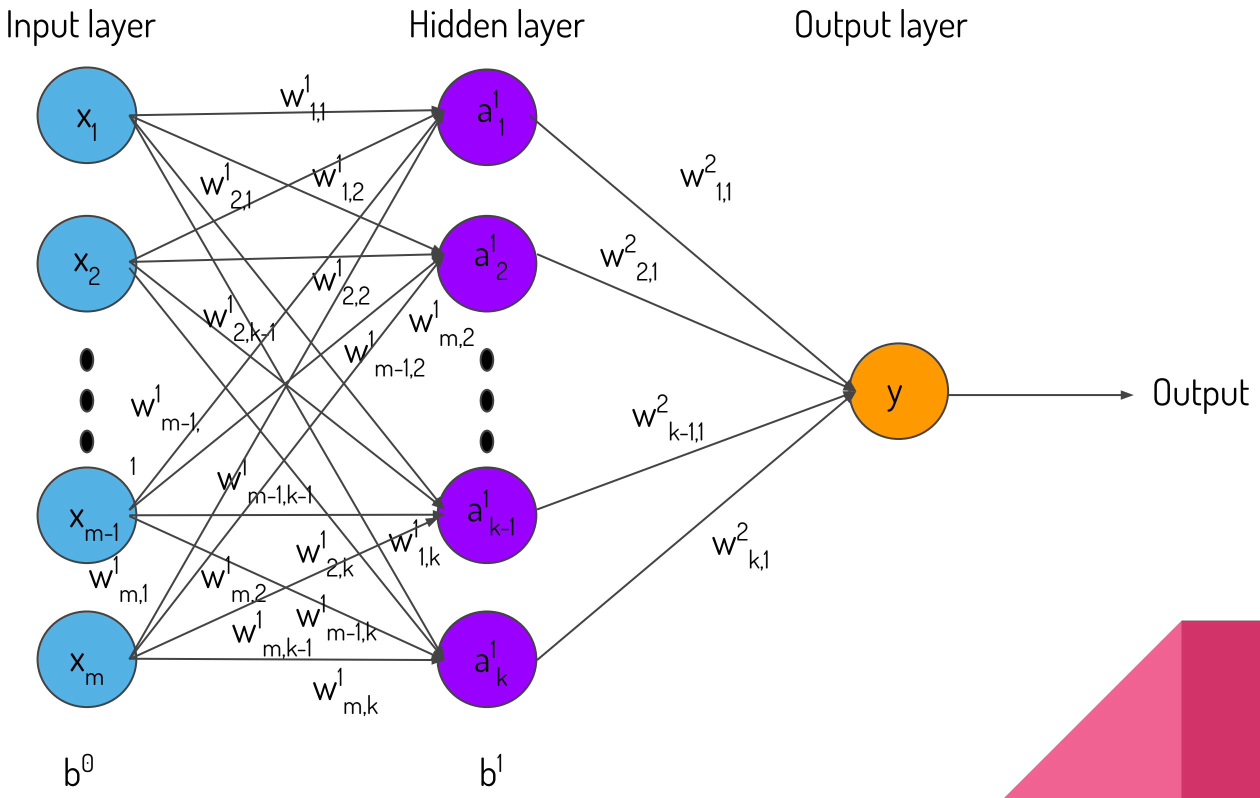
Artificial neural network is a bigger structure. It's build of layers and layers consists of neurons.

In the ANN we can recognize 3 types of layers:

- Input layer
- Hidden layers
- Output layer

A deep neural network is considered one that has **more than one** hidden layer.

Artificial Neural Network - Multilayer Perceptron (MLP)



Artificial Neural Network

Output from a neuron is calculated as follows:

$$z_{lk} = \sum_{k=1}^{K_{l-1}} w_{l-1\,k} a_{l-1\,k} + b_{l-1\,k}$$

where:

l - is a number of layer in a network

k - is the number of a neuron in a layer l

$w_{l-1\,k}$ - are weights between neurons in layer l-1 and k neuron in layer l

$a_{l-1\,k}$ – activations of neurons from layer l-1 (for the first hidden layer we substitute for l = 0 input x)

$b_{l-1\,k}$ - bias value for k neuron in layer l

$$a_{lk} = \varphi(z_{lk})$$

Forward propagation

Forward propagation

- Forward propagation is propagating an input signal through the whole network by calculating an output from each neuron in each layer.
- An input to the layer l is an output from the layer l-1.
- A final result of forward propagation is in fact a prediction of a network.

Forward propagation

$$y'(x) = f^L(W^L f^{L-1}(W^{L-1} f^{L-2}(W^{L-2} \dots f^1(W^1 x))))$$

y'(x) - final result from the whole network

L - number of layers in a network

W^l - weights of the layer l

f^l - activation function at layer l

x - input vector

Backward propagation

Backward propagation (Backpropagation)

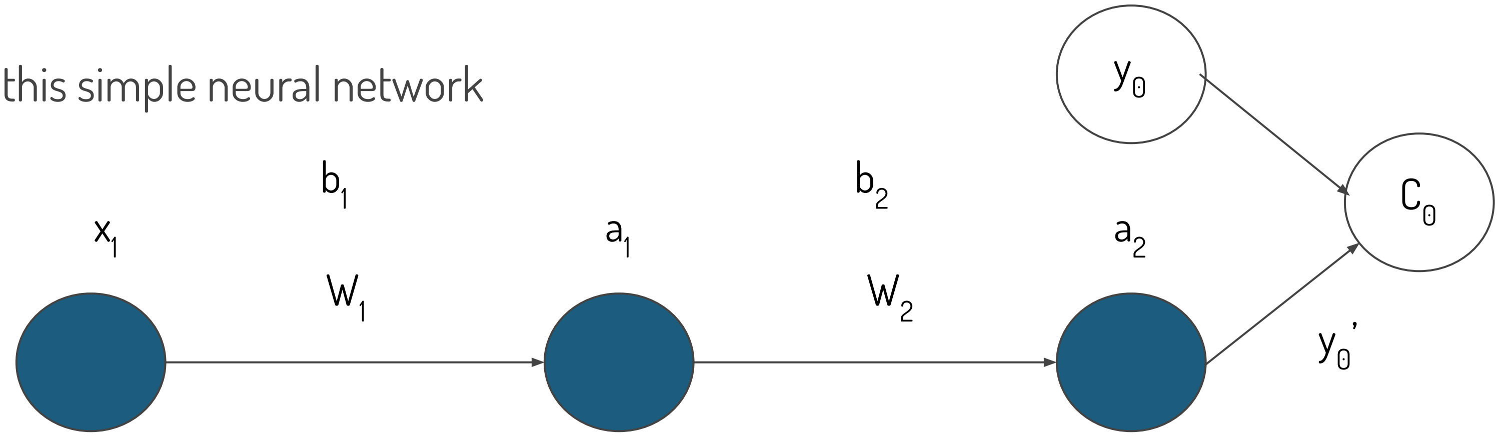
Backward propagation is used for calculating a gradient from all weights in a network.

Backward propagation (Backpropagation)

Backward propagation calculates gradient of a network's cost function.
This gradient is a vector which contains a partial derivatives of the cost function for each weights in the network.

Backward propagation

Consider this simple neural network



We want to calculate the impact of weight W_2 on the cost function C_0 which is a cost function of the first sample x.

Backward propagation

To do it we need to calculate a partial derivative of C_0 with respect to W_2.

$$\frac{\partial C_0}{\partial W_2} = \frac{\partial z_2}{\partial W_2} \frac{\partial a_2}{\partial z_2} \frac{\partial C_0}{\partial a_2}$$

To calculate it we use a ***chain rule*** to calculate the partial derivatives that 'appear' between W_2 and C_0.

Backward propagation

Partial derivative of C_0 with respect to b_2 is as follows:

$$\frac{\partial C_0}{\partial b_2} = \frac{\partial z_2}{\partial b_2} \frac{\partial a_2}{\partial z_2} \frac{\partial C_0}{\partial a_2}$$

And partial derivative of z_2 in respect to b_2 is:

$$\frac{\partial z_2}{\partial b_2} = 1$$

Backward propagation

A partial derivative of C_0 with respect to the weight from the first layer W_1 is:

$$\frac{\partial C_0}{\partial W_1} = \frac{\partial z_1}{\partial W_1} \frac{\partial a_1}{\partial z_1} \frac{\partial z_2}{\partial a_1} \frac{\partial a_2}{\partial z_2} \frac{\partial C_0}{\partial a_2}$$

Backward propagation

But this is only for one training sample. To calculate a gradient of a given weight we need to average over whole dataset.

$$\frac{\partial C}{\partial W^{(L)}} = \frac{1}{n}\sum_{i=1}^{n}\frac{\partial C_i}{\partial W^{(L)}}$$

Backward propagation

When we calculate gradient for all weights and biases we get a vector of gradients:

$$\left[\frac{\partial C}{\partial W_1}, \frac{\partial C}{\partial b_1}, \frac{\partial C}{\partial W_2}, \frac{\partial C}{\partial b_2}\right]$$

Which we use to update these weights and biases

$$W_1 = W_1 - \eta * \frac{\partial C}{\partial W_1}$$

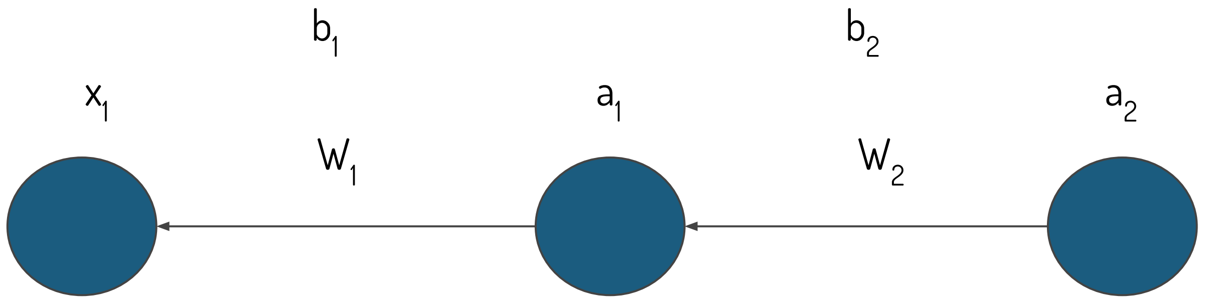
$$b_1 = b_1 - \eta * \frac{\partial C}{\partial b_1}$$

$$W_2 = W_2 - \eta * \frac{\partial C}{\partial W_2}$$

$$b_2 = b_2 - \eta * \frac{\partial C}{\partial b_2}$$

Why we called it Backward propagation?

Because we use an information (signal) obtained from the forward propagation and use it to move back from the last to the first layer of a network.



Bonus
Rapid growth of Deep Learning

Rapid growth of Deep Learning

First implementation of a Perceptron - 1958

Multilayer perceptron uses in many problems - 1980's

Backpropagation algorithm introduction - 1986

Yann LeCun used backpropagation to train neural network for Computer Vision - 1989

Rapid growth of Deep Learning

All methods showed during this course has been invented long time ago.

But we can say that ***Deep Learning explosion*** has begun only a dozen years ago.

Why???

Rapid growth of Deep Learning

Since 1950’ AI survive a lot of hard times that get a name ***AI winters.***

The source of the problems was insufficient performance of AI algorithms.

A breakthrough came a dozen years ago …

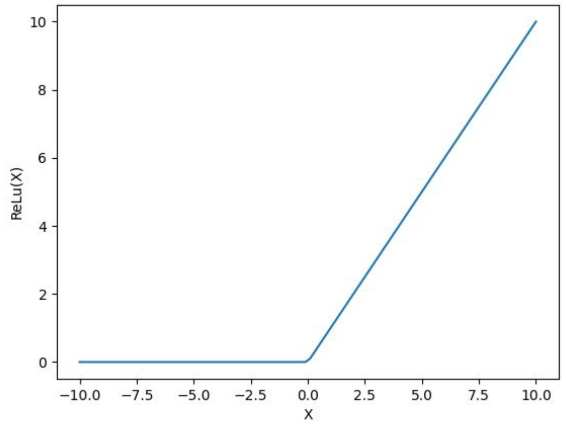
Rapid growth of Deep Learning - reasons

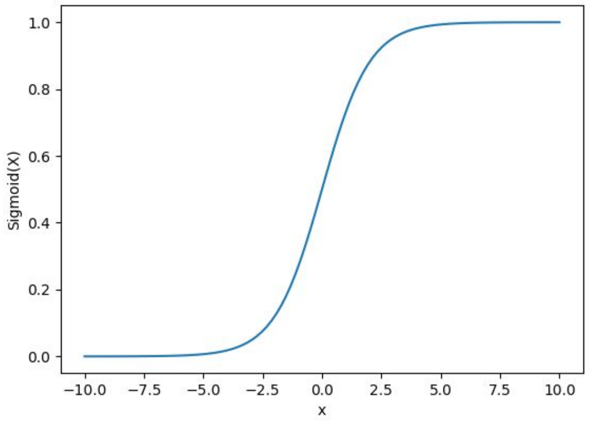
- Development of the infrastructure for collecting large amounts of data - Internet
- Introduction of new, more powerful computational units like GPU, CPU
- Introduction of the ReLU activation function (late 1960’ and 2011)
- Invention of new neural networks architectures like AlexNet (2012), ResNet (2015)
- Introduction of residual connections (2015)
- and many, many more

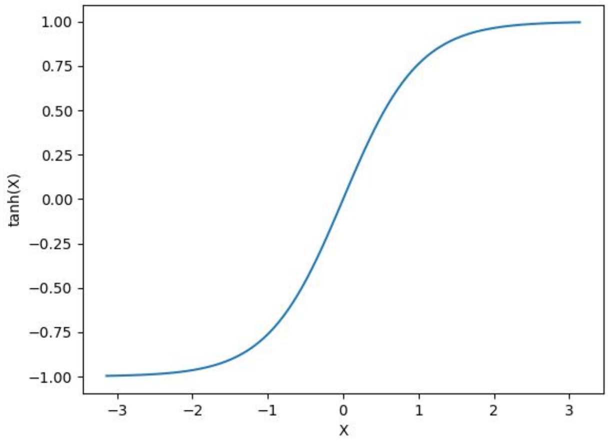
ReLu – why it’s so important?

ReLu - why it's so important?

The problem lies in backpropagation, specifically in calculating a partial derivative in respect to activation of a neuron.





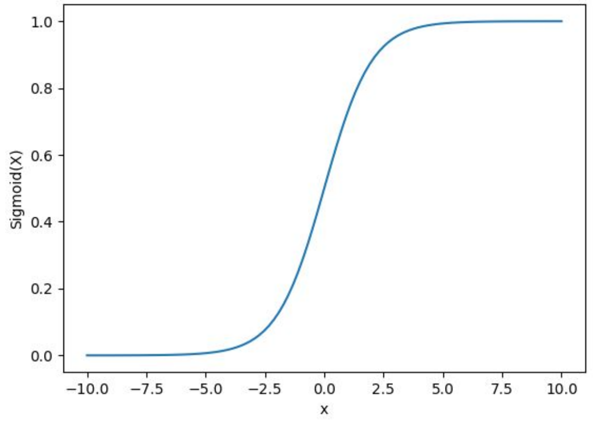


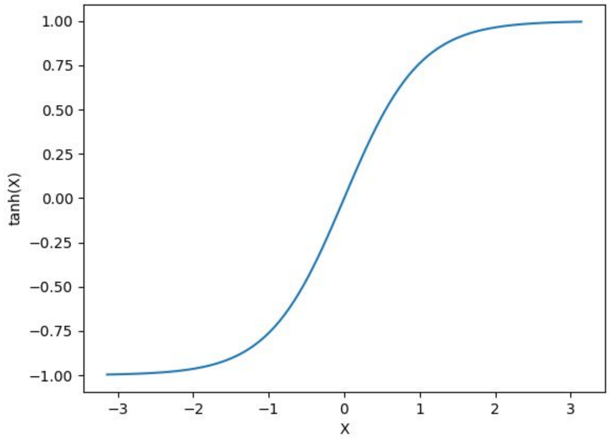
ReLu - why it's so important?

We can see that both functions sigmoid and tanh are moving towards value of 1 or 0/-1.

It means that for further values, derivatives of these functions are getting closer to zero.

And that means vanishing of the gradients.





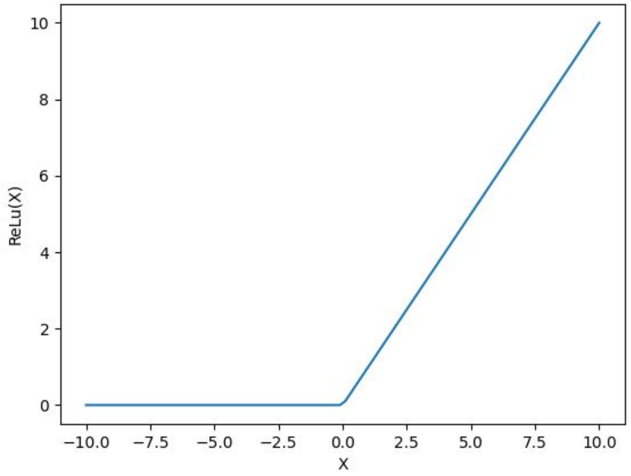
ReLu - why it's so important?

We can see that ReLu function isn't affect of this problem, because for **positive values** of**x it isn't moving towards fixed term**!

Calculating derivative from ReLu we can get two values:

- **0** for x <= 0
- **1** for x > 0

$$ReLu(x) = max(x, 0)$$



Vote for Gradient during the Złote Lwiątka poll!

Questions
&
Discussion

Hands-on

Intro_to_Deep_Learning

All hands-on materials available at
github.com/Gradient-PG/gradient-live-session

Thank you!

This time in NE AUD 2!!!

See you next week on Deep Learning in Computer Vision.