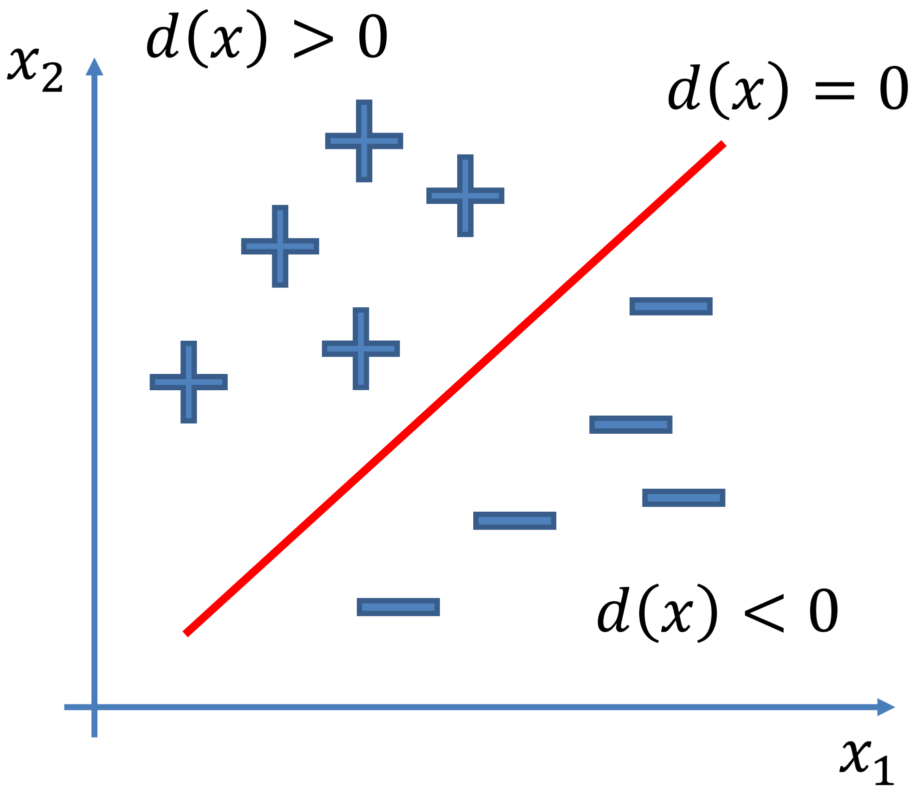
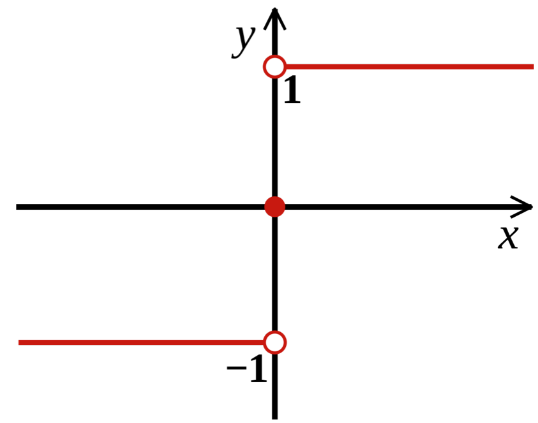
Intro

- In this video we will talk about the simplest neural network – multi-layer perceptron (MLP)

Let's recall linear binary classification

- Features: $x = (x_1, x_2)$
- Target: $y \in \{+1, -1\}$
- Decision function: $d(x) = \boldsymbol{w_0} + \boldsymbol{w_1} x_1 + \boldsymbol{w_2} x_2$
- Algorithm: $a(x) = \mathrm{sign}(d(x))$

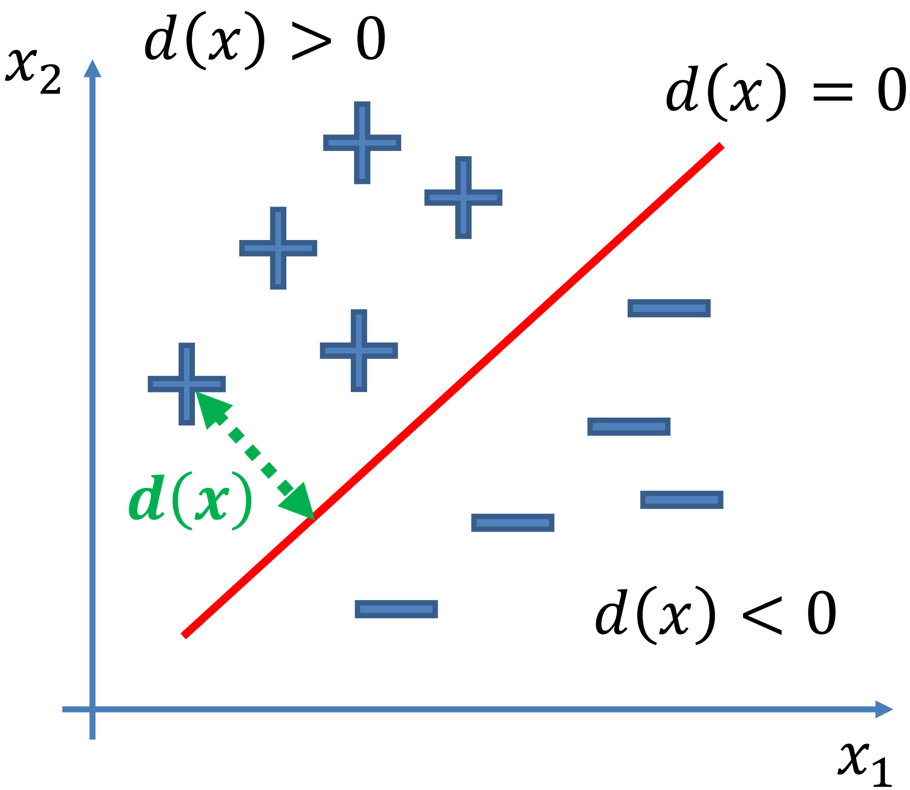


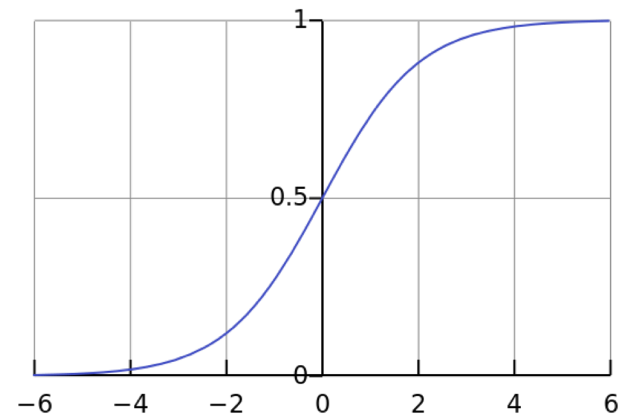


$\mathrm{sign}(x)$

Logistic regression

- Predicts probability of the positive class (+1)
- Decision function: $d(x) = \boldsymbol{w_0} + \boldsymbol{w_1} x_1 + \boldsymbol{w_2} x_2$
- Algorithm: $a(x) = \sigma\big(d(x)\big)$

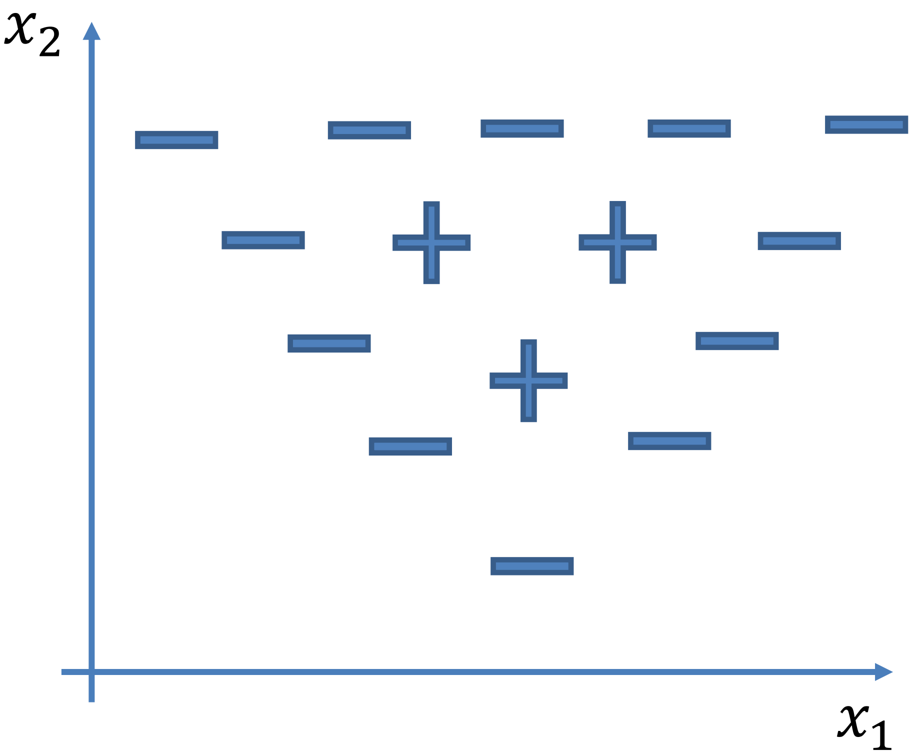




$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

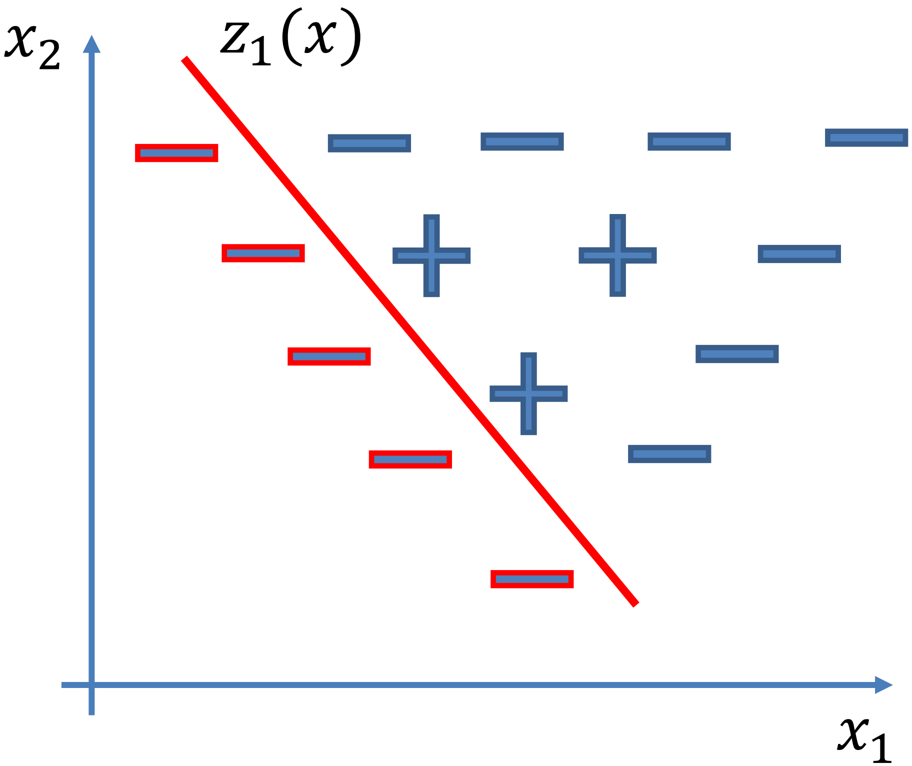
Triangle problem

- Features: $x = (x_1, x_2)$
- Target: $y \in \{+1, -1\}$



Triangle problem: solving a subproblem

- Features: $x = (x_1, x_2)$
- Target: $y \in \{+1, -1\}$

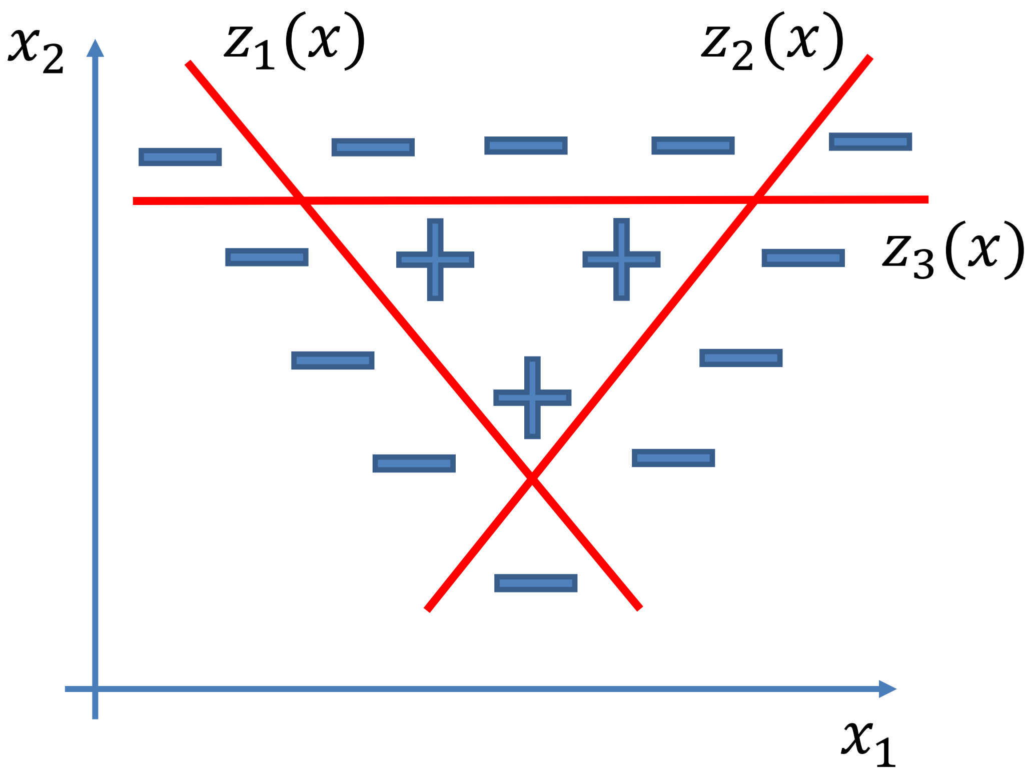


This line helps us to separate minuses on the left

$$z_1 = \sigma(\boldsymbol{w_{0,1}} + \boldsymbol{w_{1,1}} x_1 + \boldsymbol{w_{2,1}} x_2)$$

A logistic regression per a subproblem

- Features: $x = (x_1, x_2)$
- Target: $y \in \{+1, -1\}$

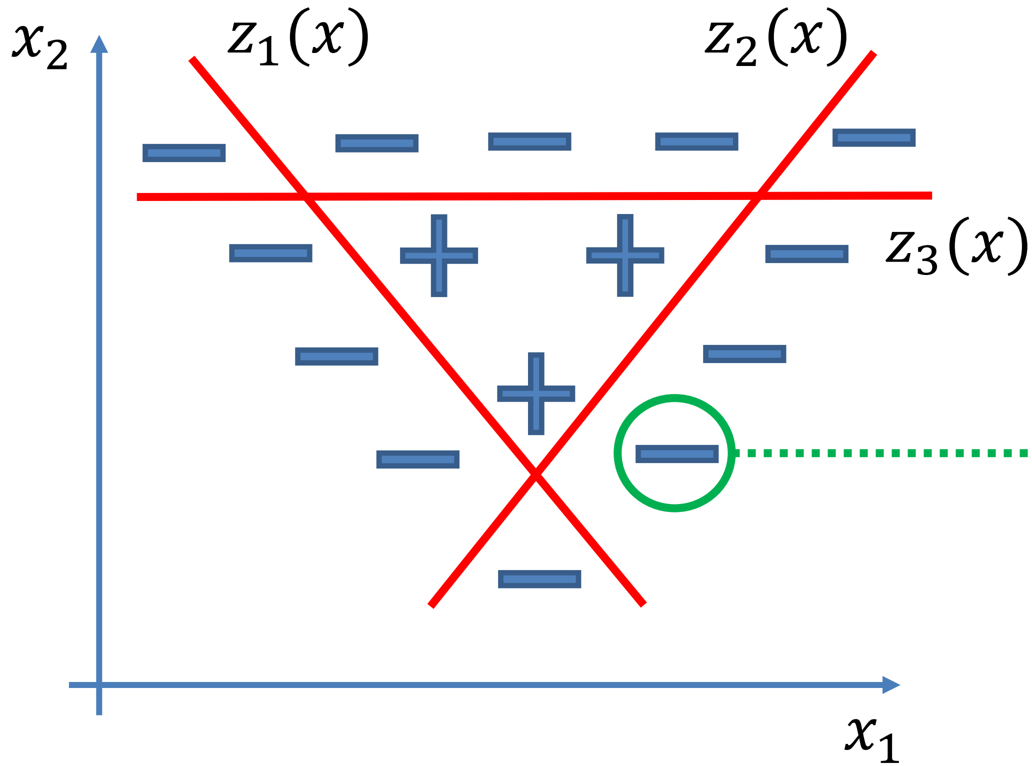


Imagine that we've somehow found these 3 lines

$$z_i = \sigma(\boldsymbol{w}_{\boldsymbol{0},\boldsymbol{i}} + \boldsymbol{w}_{\boldsymbol{1},\boldsymbol{i}} x_1 + \boldsymbol{w}_{\boldsymbol{2},\boldsymbol{i}} x_2)$$

These lines give us new features

- Features: $x = (x_1, x_2)$
- Target: $y \in \{+1, -1\}$



New features:

$z_1(x)$	$z_2(x)$	$z_3(x)$	y
0.6	**0.3**	**0.8**	**-1**
0.7	**0.7**	**0.7**	**+1**

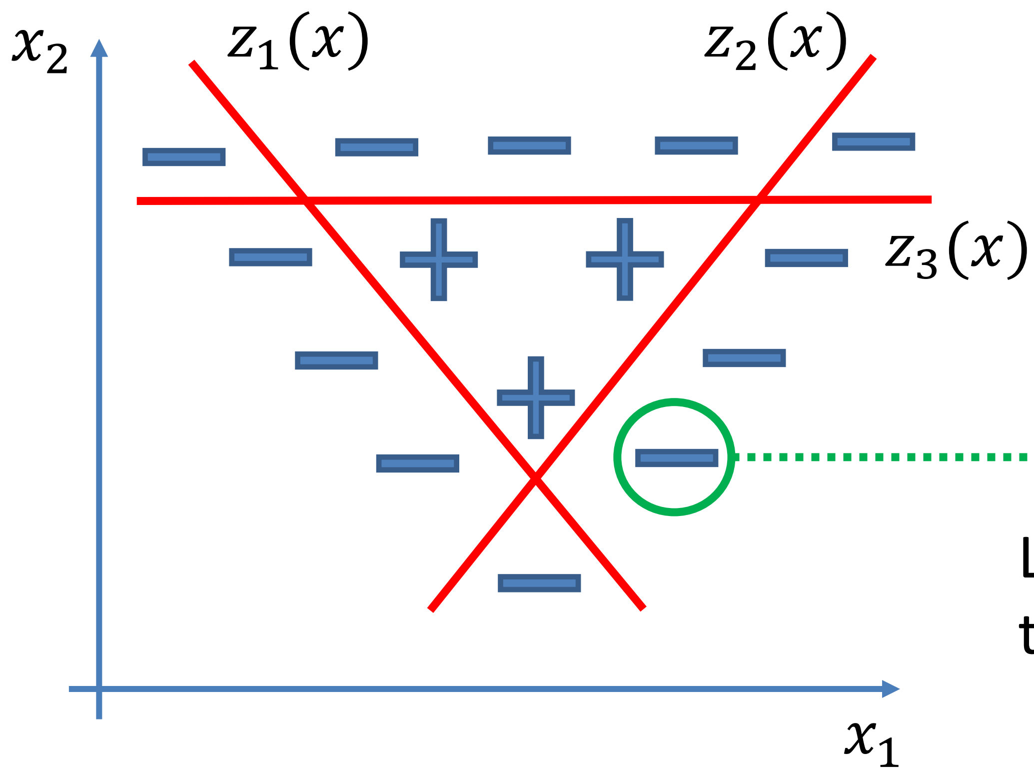
$(x_1, x_2) \rightarrow (z_1, z_2, z_3)$

What to do next?

$$z_i = \sigma(\boldsymbol{w_{0,i}} + \boldsymbol{w_{1,i}} x_1 + \boldsymbol{w_{2,i}} x_2)$$

Final algorithm

- Features: $x = (x_1, x_2)$
- Target: $y \in \{+1, -1\}$



$$z_i = \sigma(\boldsymbol{w_{0,i}} + \boldsymbol{w_{1,i}} x_1 + \boldsymbol{w_{2,i}} x_2)$$

New features:

$z_1(x)$	$z_2(x)$	$z_3(x)$	y
0.6	**0.3**	**0.8**	**-1**
0.7	**0.7**	**0.7**	**+1**

$$(x_1, x_2) \to (z_1, z_2, z_3)$$

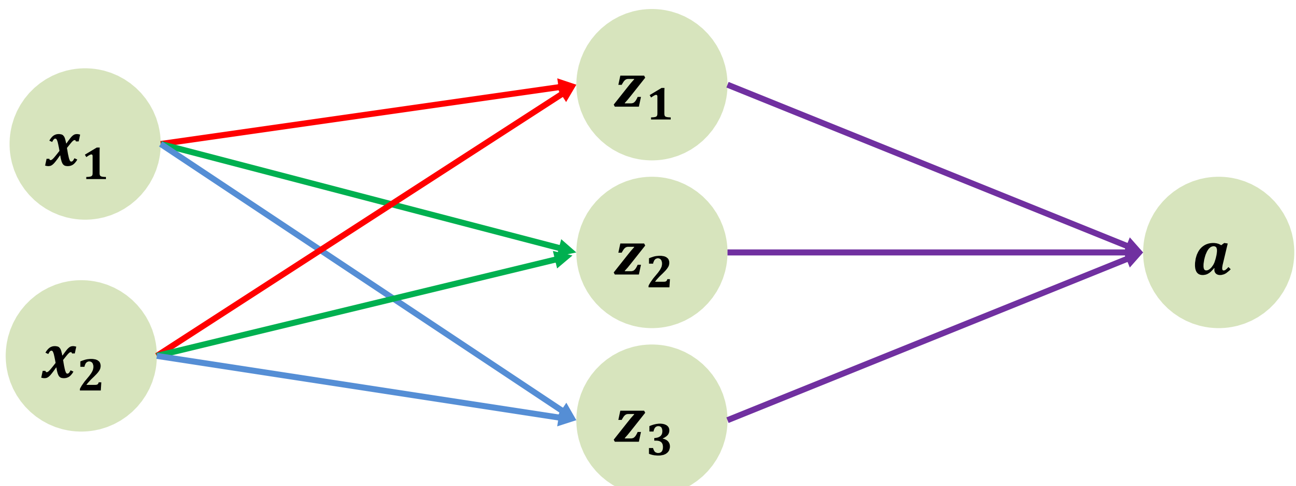
Let's build a linear model on top of these new features:

$$a(x) = \sigma(\boldsymbol{w_0} + \boldsymbol{w_1} z_1(x) + \boldsymbol{w_2} z_2(x) + \boldsymbol{w_3} z_3(x))$$

We still don't know how to find these 4 lines

- But we know how our algorithm will predict once we find them:
  - $z_i = \sigma(\boldsymbol{w_{0,i}} + \boldsymbol{w_{1,i}} x_1 + \boldsymbol{w_{2,i}} x_2)$
  - $a(x) = \sigma(\boldsymbol{w_0} + \boldsymbol{w_1} z_1(x) + \boldsymbol{w_2} z_2(x) + \boldsymbol{w_3} z_3(x))$

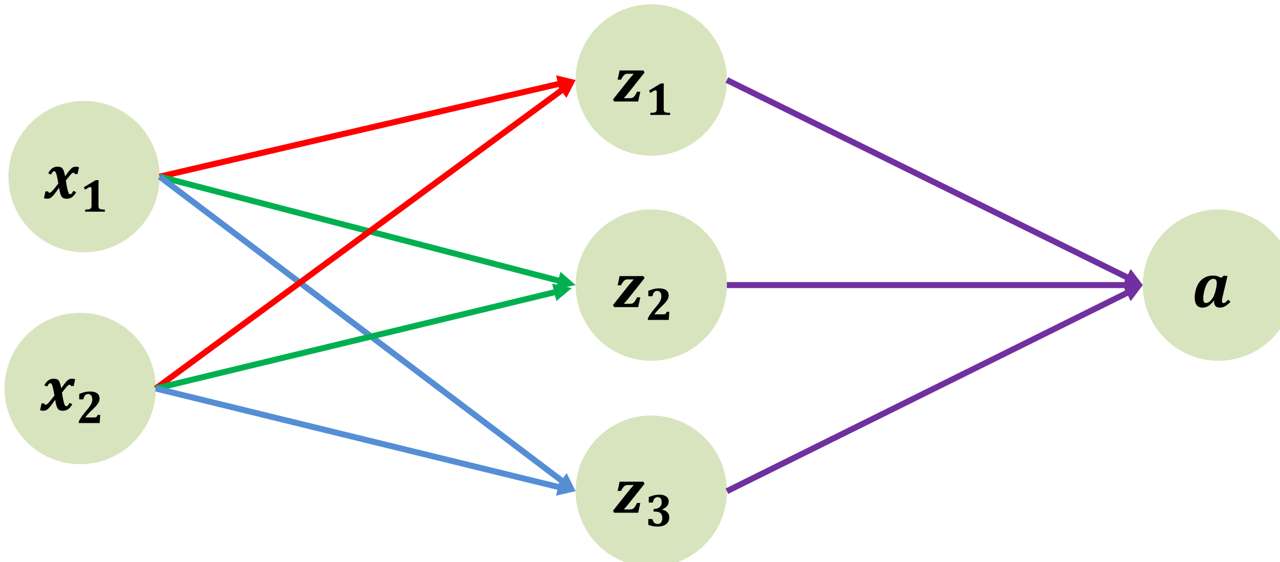
- Let's rewrite our algorithm in terms of a **computation graph**:



Nodes: computed variables ($\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{z_1}, \boldsymbol{z_2}, \boldsymbol{z_3}, \boldsymbol{a}$)
Edges: dependencies (we need $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ to compute $\boldsymbol{z_1}$)

Our computation graph has a name

- Multi-layer perceptron (MLP):



Input layer **Hidden layer** **Output layer**

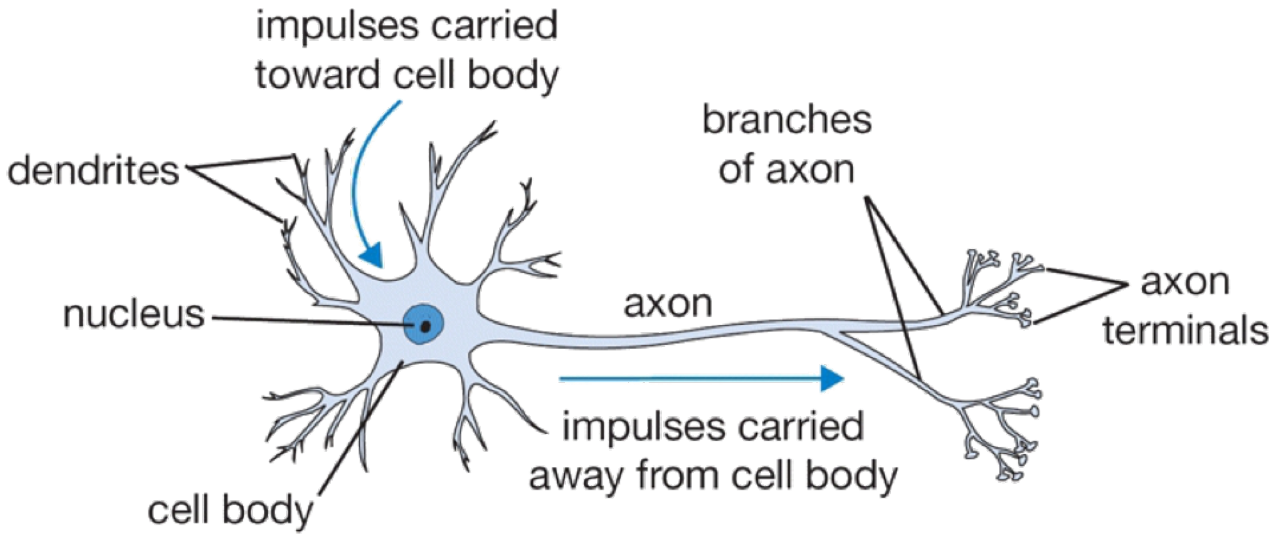
Features

Here each node is a **neuron**:

1. Take a linear combination of inputs
2. Apply **activation** function (e.g. $\sigma(x)$)

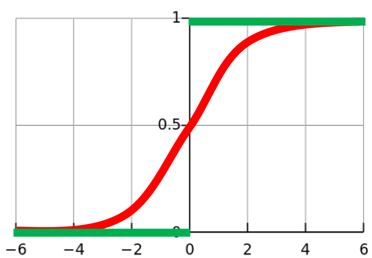
Why is it called a neuron?

- Neuron in a human brain:

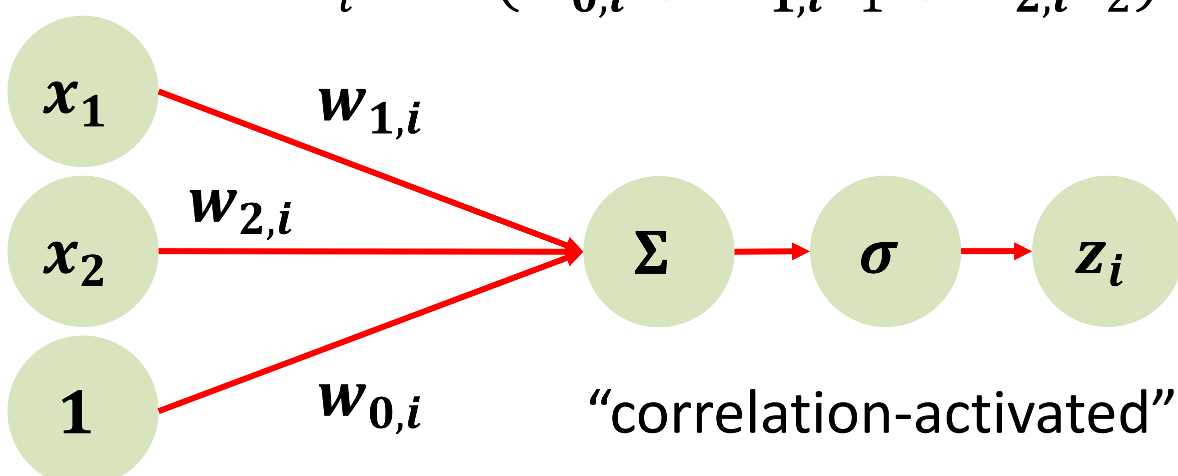


- Artificial neuron:

$$z_i = \sigma(\boldsymbol{w_{0,i}} + \boldsymbol{w_{1,i}} x_1 + \boldsymbol{w_{2,i}} x_2)$$



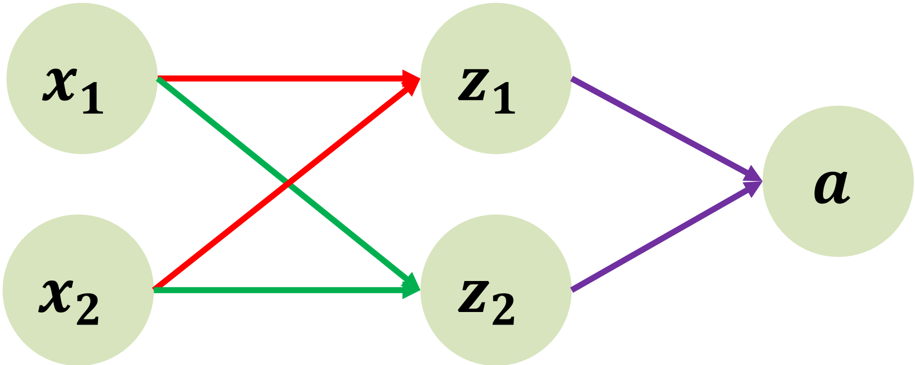
"smooth indicator"



"correlation-activated"

We need a non-linear activation function!

- Let's see what happens if we throw away $\sigma(x)$:



$$z_1 = w_{1,1}x_1 + w_{2,1}x_2$$

$$z_2 = w_{1,2}x_1 + w_{2,2}x_2$$

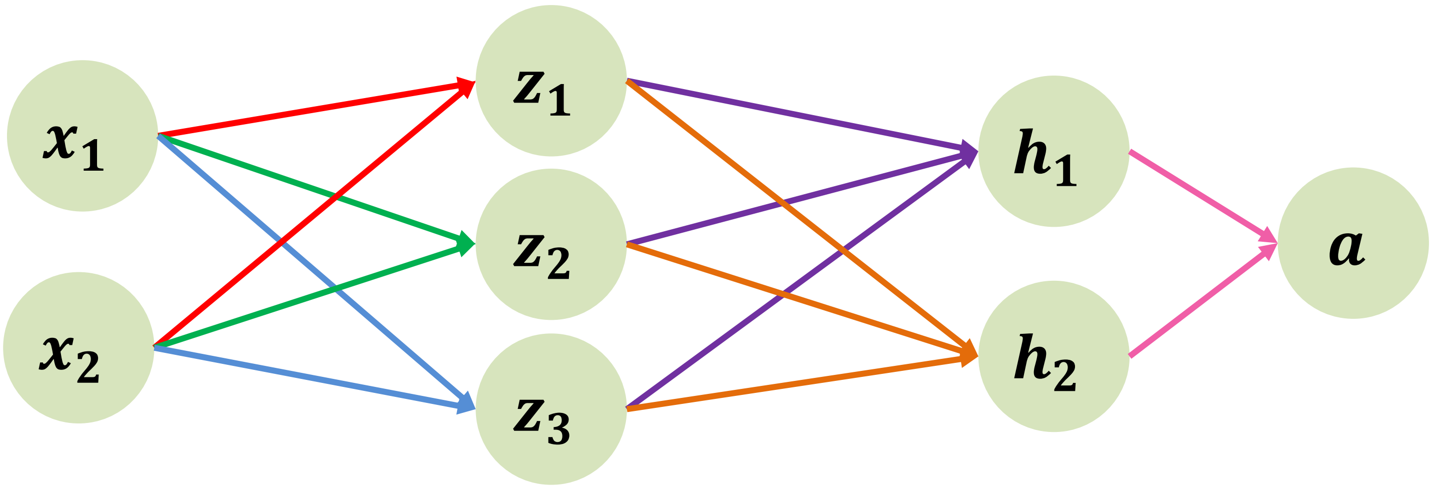
$$a = w_1 z_1 + w_2 z_2$$

- Our algorithm turns into a fancy **linear function**!
  - $a = (w_1 w_{1,1} + w_2 w_{1,2})x_1 + (w_1 w_{2,1} + w_2 w_{2,2})x_2$

MLP overview

- MLP is an example of **artificial neural network**

- MLP can have many hidden layers:



- **Architecture** of an MLP:
  - Number of layers
  - Number of neurons in each layer
  - Activation function

- Hidden layer in MLP:
  - Dense layer
  - Fully-connected layer

How to train your MLP?

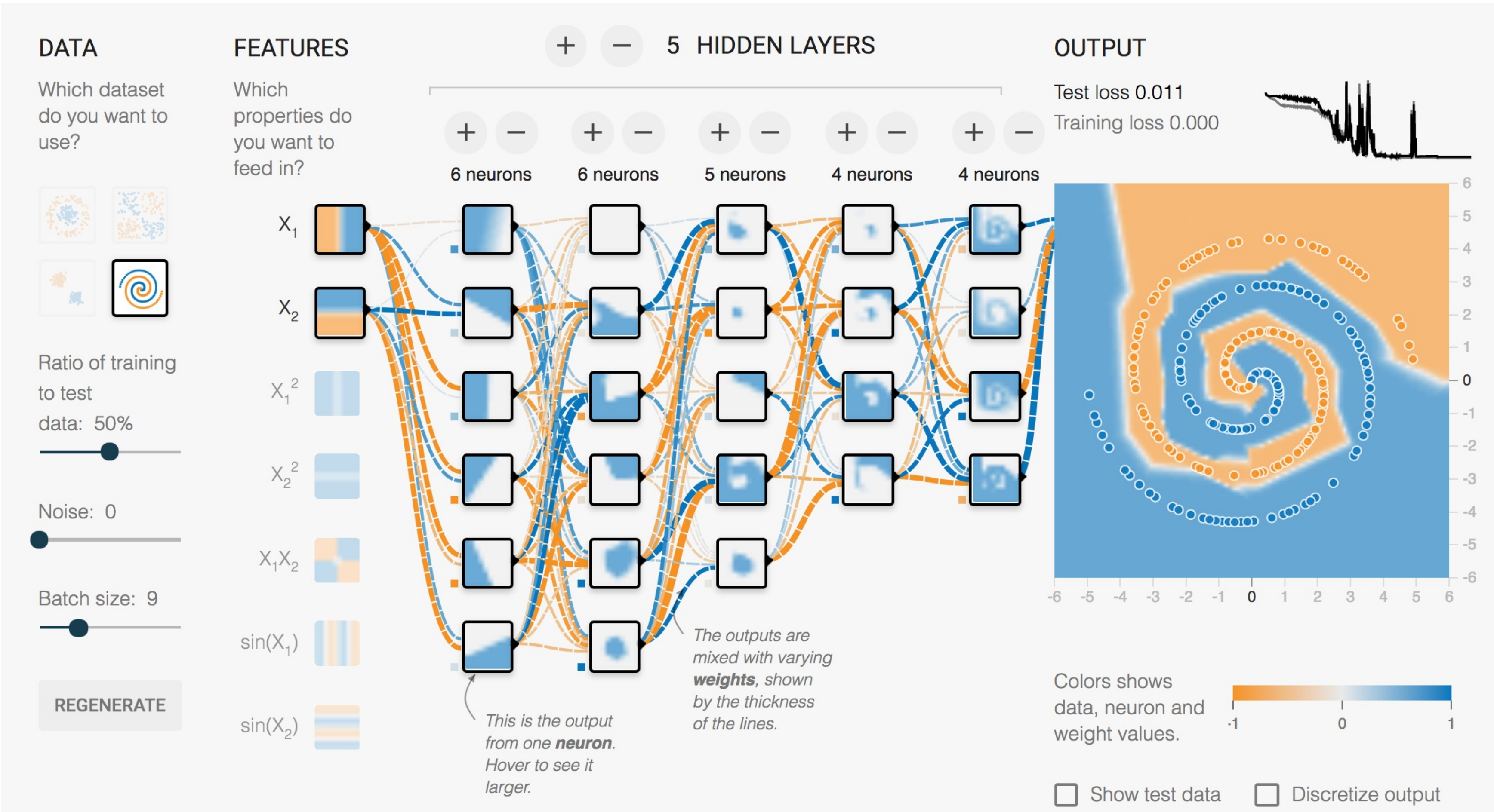
- We know how to train one neuron (e.g. logistic regression): SGD
- Let’s do the same for the whole MLP!

Bob chilling at a local optima

https://hackernoon.com/life-is-gradient-descent-880c60ac1be8

Check out MLP training demo

- http://playground.tensorflow.org (change activation to ReLU)



How to train your MLP?

- Other problems:
  - We can have many hidden layers (hyper-parameter) → we need to calculate gradients **automatically**!
  - We can have many neurons → we need to calculate gradients **fast**!

- In the next video we will solve these problems!