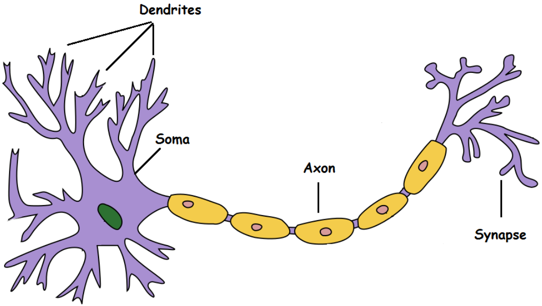
Artificial neurons, neural networks

Tamás Grósz

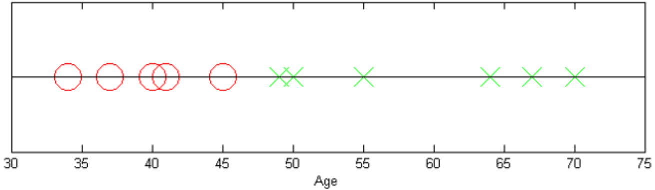
Background

- Artificial neurons are mathematical functions conceived as a model of biological neurons
- They are the basic building blocks of artificial neural networks
- Invented by Rosenblatt in 1957



Geometric explanation

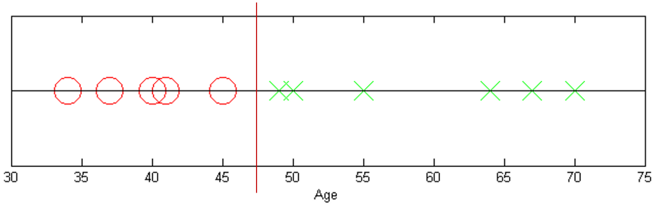
Let's say we have collected 1D data (see below), what would be an optimal decision?



(a) training data

Geometric explanation

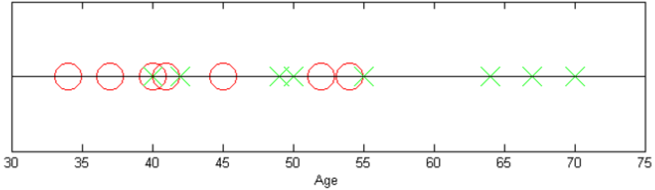
Let's say we have collected 1D data (see below), what would be an optimal decision?



(b) possible decision

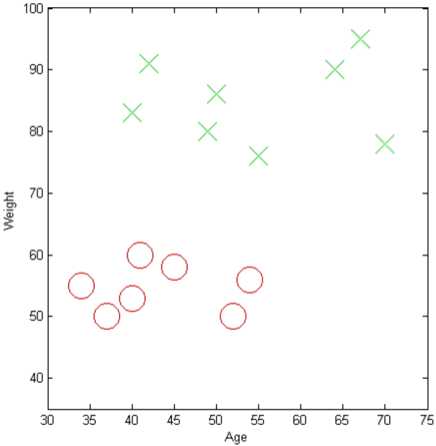
Geometric explanation

Sometimes we cannot separate the data in low dimensions.



(c) training data

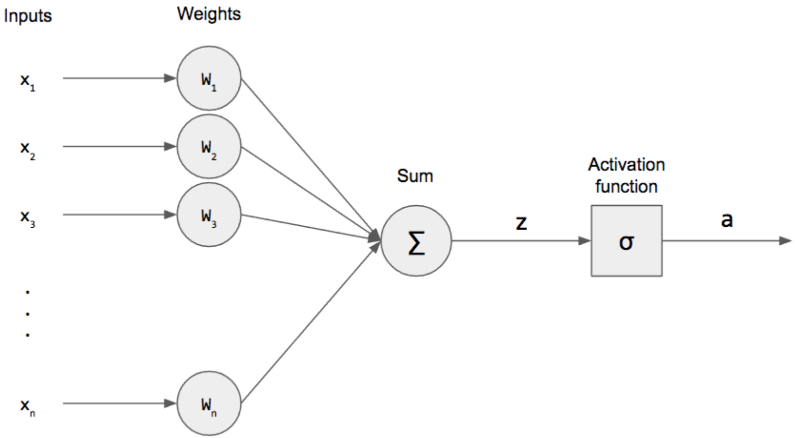
Geometric explanation



(d) Extended feature space

Perceptron model

- The artificial neuron (Perceptron) learns a hyperplane in the feature space



Perceptron

Bias

The bias value allows the shifting of the activation function to the left or the right, which may be critical for successful learning. Usualy it is stored in w_0 and the input vextor (x) is extended with $x_0 = 1$.

Activation functions

- Step function: $Step(x) = \begin{cases} 0, & \text{if } x < 0 \\ 1, & \text{otherwise} \end{cases}$
- Sigmoid: $Sigmoid(x) = \frac{1}{1+e^{-x}}$
- Tangent hyperbolicus: $Tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$

Loss functions

To train the neuron, we need to define a function that measures its loss during training

Mean Squared Errror

$$MSE(w) = \frac{1}{N}\sum_{d=1}^{D}(y_d - o_d)^2$$

$$o_d = Sigmoid(\sum_i w_i x_i)$$

Note: here D=1, because we have only one neuron

Perceptron training algorithm

- Initialy we set each w_i to some samll random value
- Our goal is find new values for w so that the Loss becomes minimal

Perceptron training algorithm

- Initialy we set each w_i to some samll random value
- Our goal is find new values for w so that the Loss becomes minimal
- It is a global optimization problem, if the input is high dimensional we cannot solve it!

Perceptron training algorithm

- Initialy we set each w_i to some samll random value
- Our goal is find new values for w so that the Loss becomes minimal
- It is a global optimization problem, if the input is high dimensional we cannot solve it!
- Solution: look for a local minimum of the Loss function

Perceptron training algorithm

- Initialy we set each w_i to some samll random value
- Our goal is find new values for w so that the Loss becomes minimal
- It is a global optimization problem, if the input is high dimensional we cannot solve it!
- Solution: look for a local minimum of the Loss function
- It can be achieved by a hill climbing optimizer

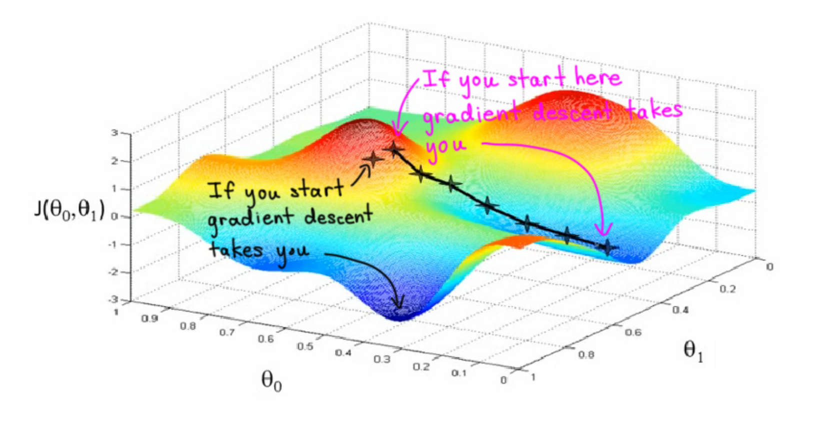
Gradient descent algorithm

It is a first-order iterative optimization algorithm for finding the minimum of a function.
To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient.

Gradient of a neuron

$$\frac{\partial MSE(w)}{\partial w_i} = \underbrace{2 \times (t_d - o_d) \times}_{\text{derivate of MSE}} \underbrace{o_d \times (1 - o_d) \times}_{\text{Sigmoid function}} \underbrace{x_i}_{\text{wx}}$$

Gradient descent algorithm



Perceptron update rule

- $w^t = w^{t-1} + \alpha \frac{\partial MSE(w^{t-1})}{\partial w^{t-1}}$
- α is the learning rate

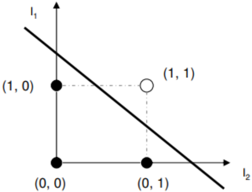
Perceptron update rule

- $w^t = w^{t-1} + \alpha \frac{\partial MSE(w^{t-1})}{\partial w^{t-1}}$
- α is the learning rate
- To calculate the new weights, we can use many examples (batch)
- If the batchsize<N, then the optimization method is called Stochasatic Gradien Descent
- Epoch/iteration: all the available data was show to the neuron

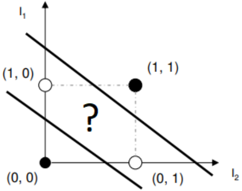
Perceptron representation strength

One perceptron has limited representation strength:

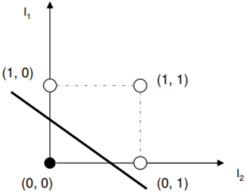
AND		
I_1	I_2	out
0	0	0
0	1	0
1	0	0
1	1	1



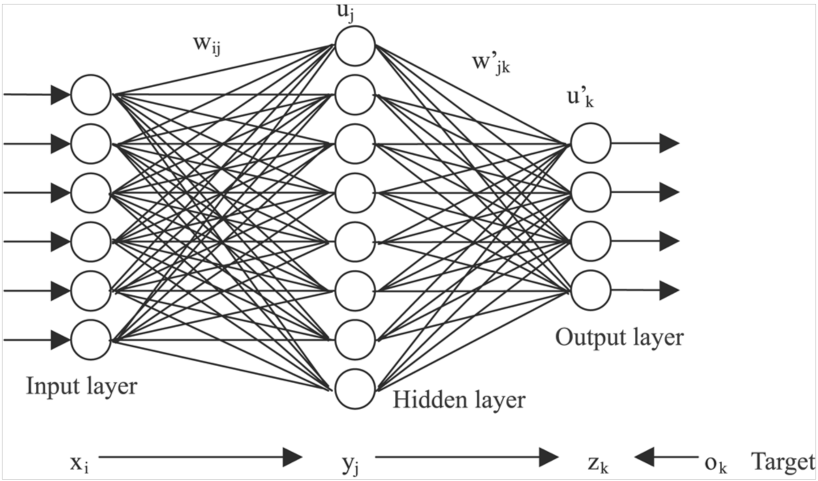
XOR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	0



OR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	1



Artificial Neural Networks



Practice

https://playground.tensorflow.org
Python tutorial: practice_02.ipynb