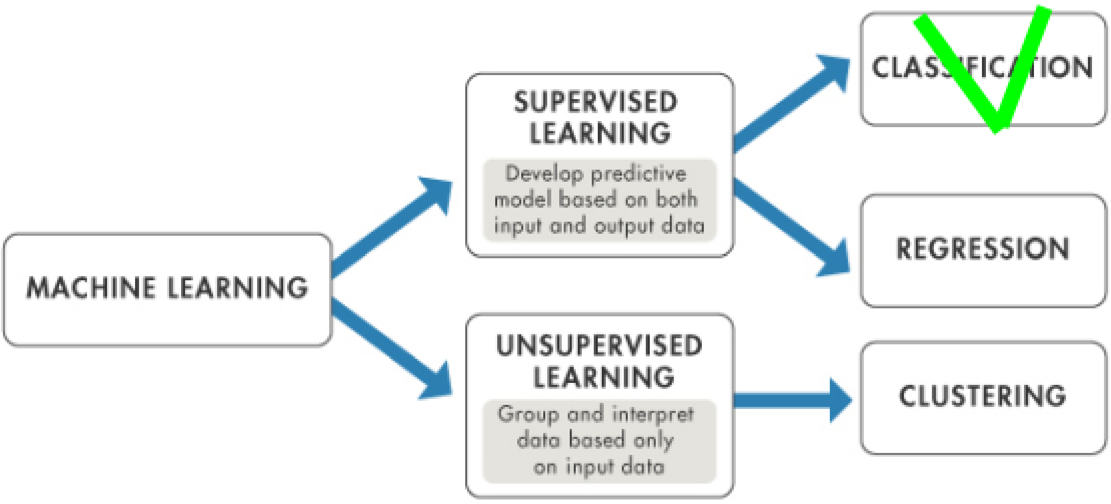
[2021.09] Google Sheet exercise (Supervised Learning) (Perceptron Classification) (AND, OR, XOR)



MACHINE LEARNING

SUPERVISED LEARNING

Develop predictive model based on both input and output data

UNSUPERVISED LEARNING

Group and interpret data based only on input data

CLASSIFICATION

REGRESSION

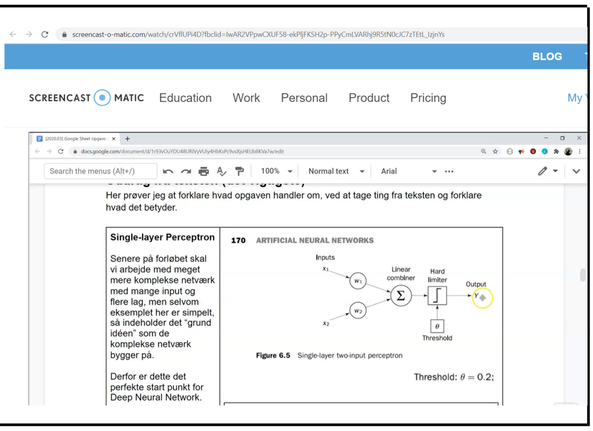
CLUSTERING

This is an important task. This is where the basic understanding of how the computer learns is shown. We are laying the foundation for Deep Neural Network (DNN) understanding here. If you understand what this task is trying to show you, the rest of the course, where we work with simulation and python, will be much easier for you. Therefore, give yourself plenty of time to understand and solve the task. Rather too slow than too fast is important here.

Video

The exercise is explained (11 min.) here.

The topic may be difficult, so be patient. Re-watch if necessary.



screencast-o-matic.com/watch/crVflUPi4D?fbclid=IwAR2VPpwCXUF58-ekPljFKSH2p-PPyCmLVARhj9R5tN0cJC7zTEtL_IzjnYs

docs.google.com/document/d/1r93vOuYDU48URIVyVUly4HbKsPc9vxXjsHEUb8KVa7w/edit

Her prøver jeg at forklare hvad opgaven handler om, ved at tage ting fra teksten og forklare hvad det betyder.

Single-layer Perceptron

Senere på forløbet skal vi arbejde med meget mere komplekse netværk med mange input og flere lag, men selvom eksemplet her er simpelt, så indeholder det "grund idéen" som de komplekse netværk bygger på.

Derfor er dette det perfekte start punkt for Deep Neural Network.

170 ARTIFICIAL NEURAL NETWORKS

Figure 6.5 Single-layer two-input perceptron

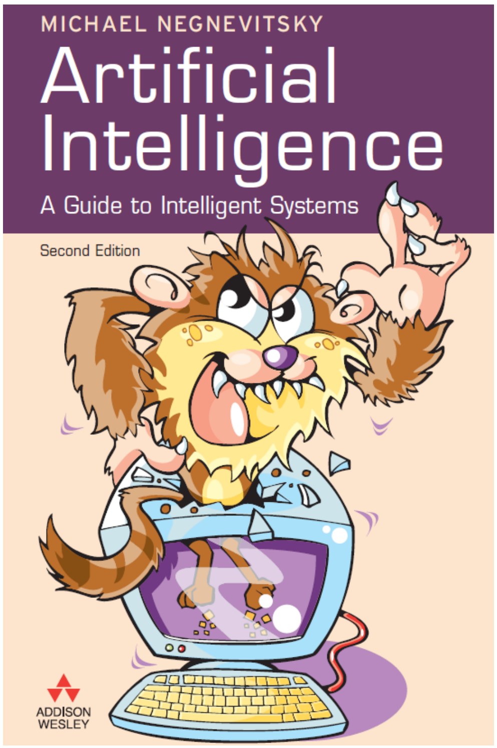
Threshold: $\theta = 0.2$;

Artificial Intelligence: A Guide to Intelligent Systems

The example we are working on in this assignment comes from this book.

The complete text which describes the perceptron example can be read here…

The important things from the text, come further down in this text.



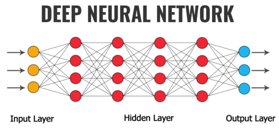
Excerpt from the text

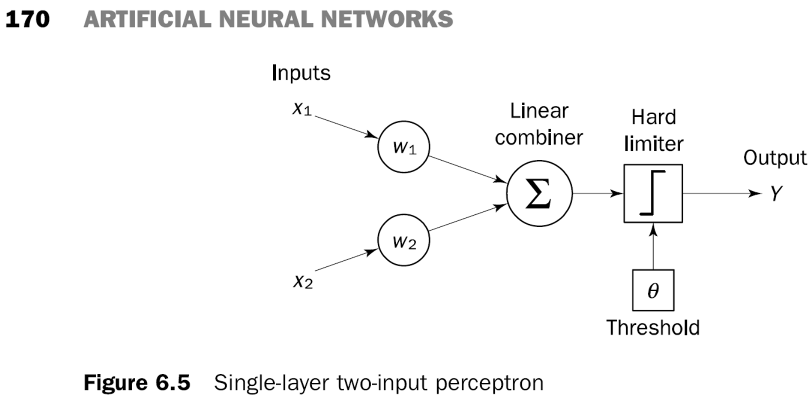
Here I will explain the exercise, based on the text book

Single-layer Perceptron

Later in the course, we will work with much more complex networks with many inputs and several layers, but even though the example here is simple, it contains the “basic idea” on which the complex networks are based.

Therefore, this is the perfect starting point for Deep Neural Network.





170 ARTIFICIAL NEURAL NETWORKS

Figure 6.5 Single-layer two-input perceptron

Threshold: $\theta = 0.2$;

A perceptron can learn the AND operation

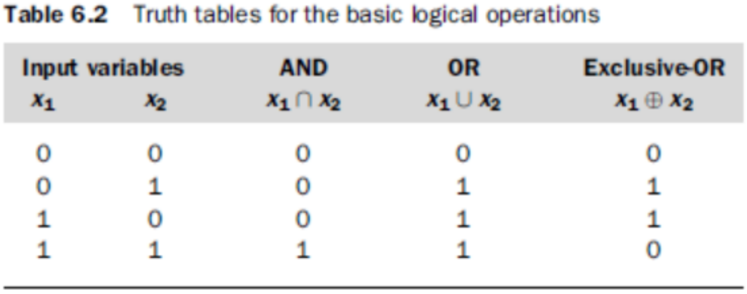
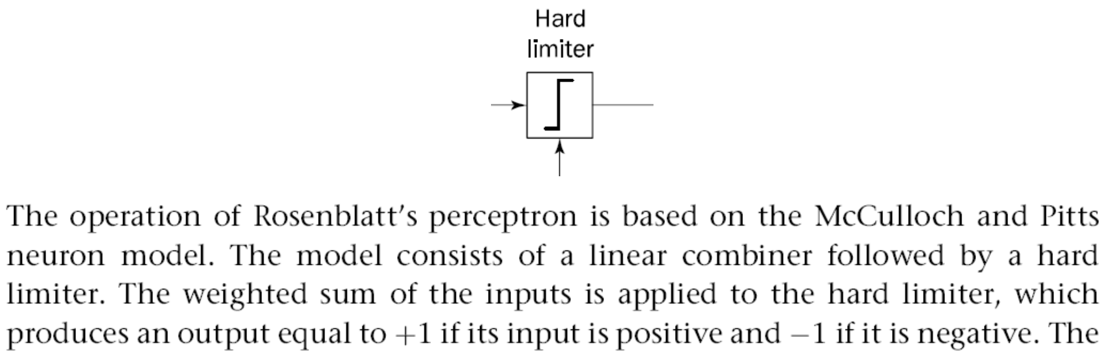


Table 6.2 Truth tables for the basic logical operations

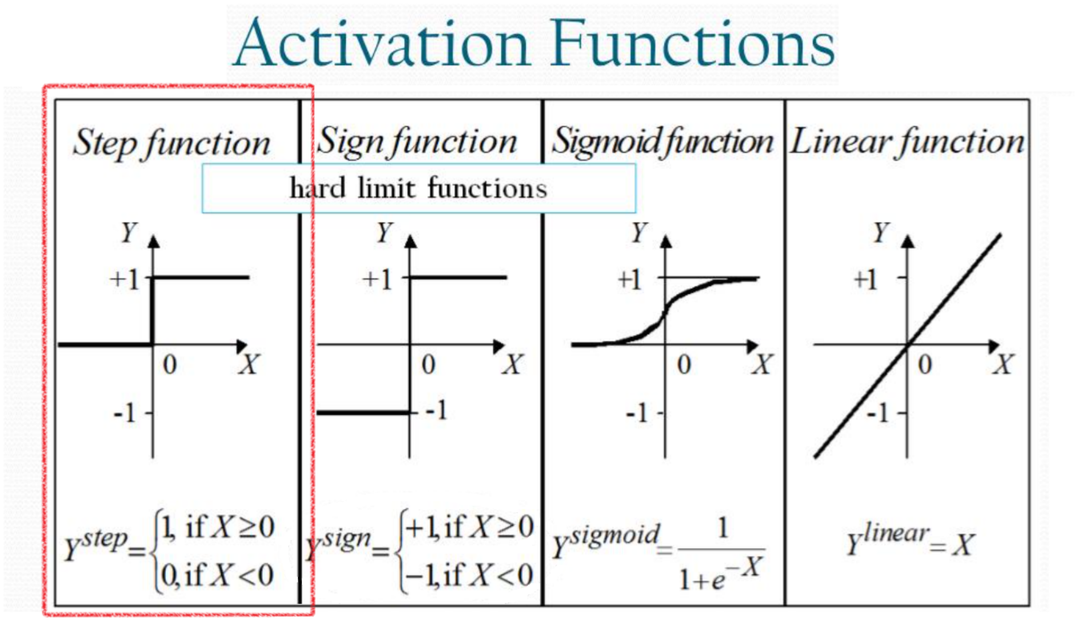
Input variables		AND	OR	Exclusive-OR
x_1	x_2	$x_1 \cap x_2$	$x_1 \cup x_2$	$x_1 \oplus x_2$
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0


Hard
limiter

The operation of Rosenblatt's perceptron is based on the McCulloch and Pitts neuron model. The model consists of a linear combiner followed by a hard limiter. The weighted sum of the inputs is applied to the hard limiter, which produces an output equal to +1 if its input is positive and −1 if it is negative. The

Important: It says that a negative input gives -1. But that's not the one we use for the algorithm itself. It says that we use STEP function. See the overview below. Here you see that the STEP function returns 0 if the input is negative and 1 if it is positive.

```
Step function definition:
if x >= 0
   y = 1
if x < 0
   y = 0
```



Activation Functions

$$Y^{step} = \begin{cases} 1, \text{ if } X \geq 0 \\ 0, \text{ if } X < 0 \end{cases}$$

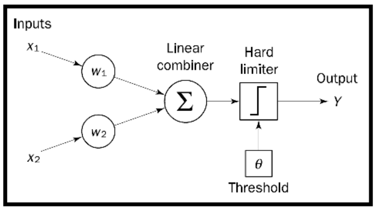
$$Y^{sign} = \begin{cases} +1, \text{ if } X \geq 0 \\ -1, \text{ if } X < 0 \end{cases}$$

$$Y^{sigmoid} = \frac{1}{1+e^{-X}}$$

$$Y^{linear} = X$$

Perceptron Algorithm

Step 1: ***Initialisation***
Set initial weights $w_1, w_2, \ldots, w_n$ and threshold θ to random numbers in the range $[-0.5, 0.5]$.

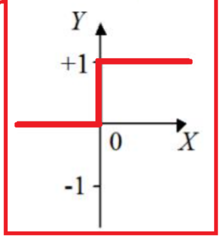


Step 2: ***Activation***
Activate the perceptron by applying inputs $x_1(p), x_2(p), \ldots, x_n(p)$ and desired output $Y_d(p)$. Calculate the actual output at iteration $p = 1$

$$Y(p) = step\left[\sum_{i=1}^{n} x_i(p) w_i(p) - \theta\right], \tag{6.6}$$

where n is the number of the perceptron inputs, and $step$ is a step activation function.

Step function



Step 3: ***Weight training***
Update the weights of the perceptron

$e(p) = Y_d(p) - Y(p)$ | error

$$w_i(p+1) = w_i(p) + \Delta w_i(p), \tag{6.7}$$

where $\Delta w_i(p)$ is the weight correction at iteration p.
The weight correction is computed by the **delta rule**:

$$\boxed{\Delta w_i(p) = \alpha \times x_i(p) \times e(p)} \tag{6.8}$$

learning rate: $\alpha = 0.1$.

Step 4: ***Iteration***
Increase iteration p by one, go back to Step 2 and repeat the process until convergence.

Algorithm idea

For each time (p) do the following:

1. **Find the error**
2. **Adjust the weights according to the error.**

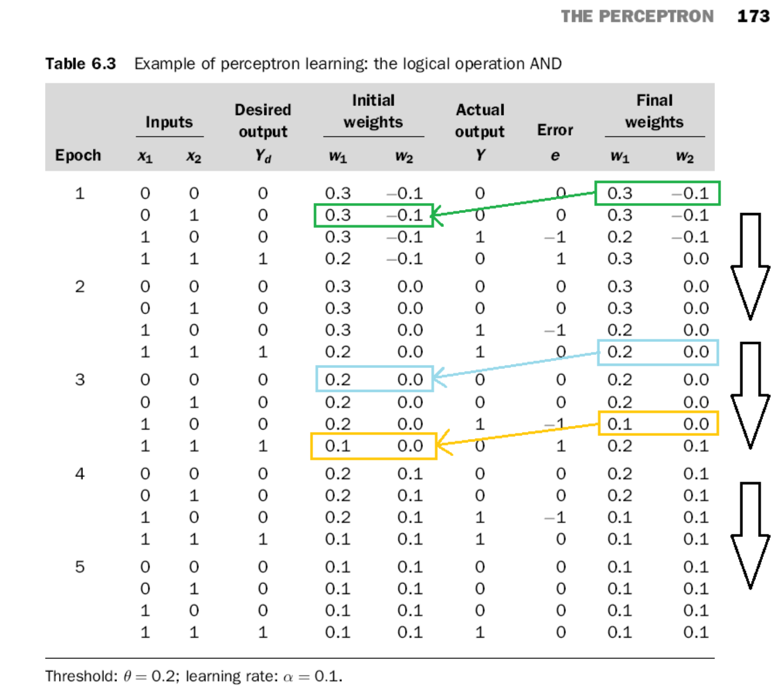
The error is central for learning. No error, no learning.

Exercise

On page 173 in the document, you will see this table (6.3)

1. Create a new Google Sheet.
   a. See this video, on how to work with Google Sheets (formulas and such)
2. You need to create a Perceptron in Google Sheet, that can learn the AND operation (see picture to the right)
3. Here you can download a template that you can use as a starting point.
4. Important: The calculations of w1 and w2 must be rounded, otherwise the solution will not work. It can be done like this:
    = ROUND (I2, 2)
   The number 2 stands for number of digits after a comma.

- When you are done, make a new sheet to model the OR operation.
- … and finally a sheet for the XOR operation. This will not work…



THE PERCEPTRON 173

Table 6.3 Example of perceptron learning: the logical operation AND

	Inputs		Desired output	Initial weights		Actual output	Error	Final weights	
Epoch	x_1	x_2	Y_d	w_1	w_2	Y	e	w_1	w_2
1	0	0	0	0.3	−0.1	0	0	0.3	−0.1
	0	1	0	0.3	−0.1	0	0	0.3	−0.1
	1	0	0	0.3	−0.1	1	−1	0.2	−0.1
	1	1	1	0.2	−0.1	0	1	0.3	0.0
2	0	0	0	0.3	0.0	0	0	0.3	0.0
	0	1	0	0.3	0.0	0	0	0.3	0.0
	1	0	0	0.3	0.0	1	−1	0.2	0.0
	1	1	1	0.2	0.0	1	0	0.2	0.0
3	0	0	0	0.2	0.0	0	0	0.2	0.0
	0	1	0	0.2	0.0	0	0	0.2	0.0
	1	0	0	0.2	0.0	1	−1	0.1	0.0
	1	1	1	0.1	0.0	0	1	0.2	0.1
4	0	0	0	0.2	0.1	0	0	0.2	0.1
	0	1	0	0.2	0.1	0	0	0.2	0.1
	1	0	0	0.2	0.1	1	−1	0.1	0.1
	1	1	1	0.1	0.1	1	0	0.1	0.1
5	0	0	0	0.1	0.1	0	0	0.1	0.1
	0	1	0	0.1	0.1	0	0	0.1	0.1
	1	0	0	0.1	0.1	0	0	0.1	0.1
	1	1	1	0.1	0.1	1	0	0.1	0.1

Threshold: $\theta = 0.2$; learning rate: $\alpha = 0.1$.

NOTE: The weights are calculated for each row, and then transferred to the row below. (see table 6.3)

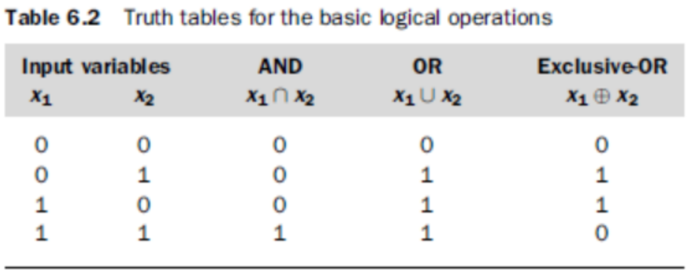


Table 6.2 Truth tables for the basic logical operations

Input variables		AND	OR	Exclusive-OR
x_1	x_2	$x_1 \cap x_2$	$x_1 \cup x_2$	$x_1 \oplus x_2$
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Questions

Reflect on the following questions:

- What is our model?
- What exactly is learning?