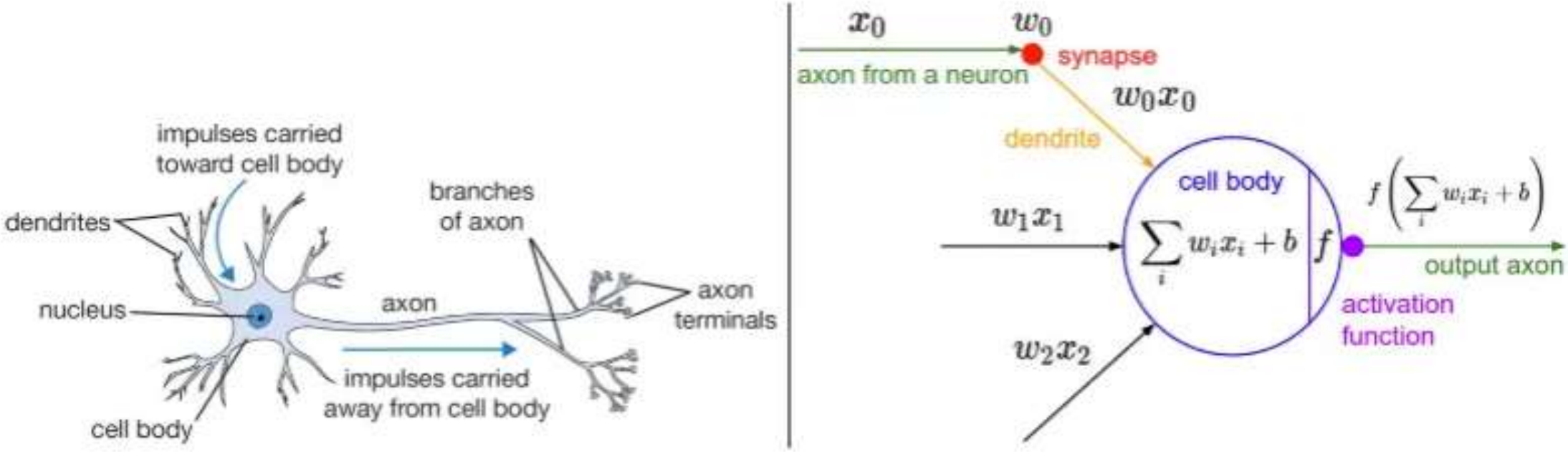
Нейрон и его математическая модель

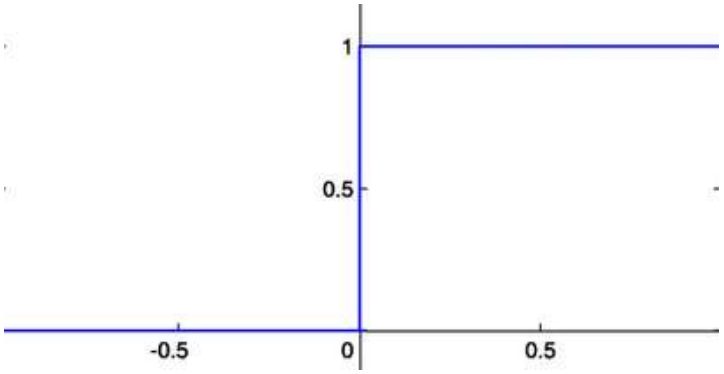


https://srijayjk.medium.com/activation-functions-in-dnn-21729d529364

Функция активации

- Пороговая передаточная функция

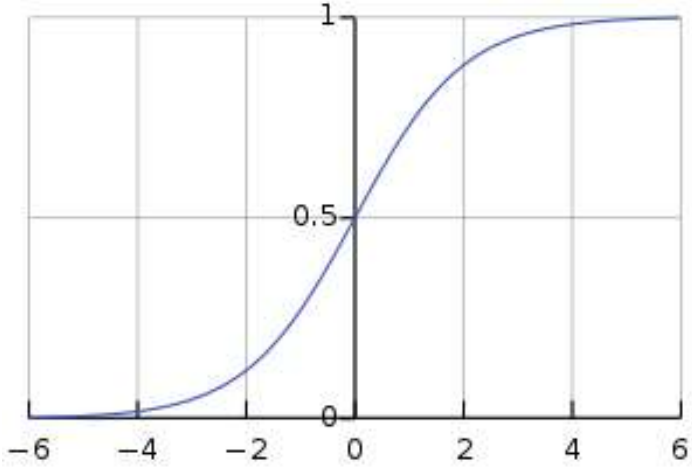
$$f(x) = \begin{cases} 1 & \text{if } x \geq T \\ 0 & \text{else} \end{cases}$$



$$x = \sum w_i x_i + b$$

- Сигмоидальная передаточная функция

$$\sigma(x) = \frac{1}{(1 + \exp(-tx))}$$



https://en.wikipedia.org/wiki/Artificial_neuron

Обучение на массиве 3x1

	Input			Output
Example 1	0	0	1	0
Example 2	1	1	1	1
Example 3	1	0	1	1
Example 4	0	1	1	0

New situation	1	0	0	**?**

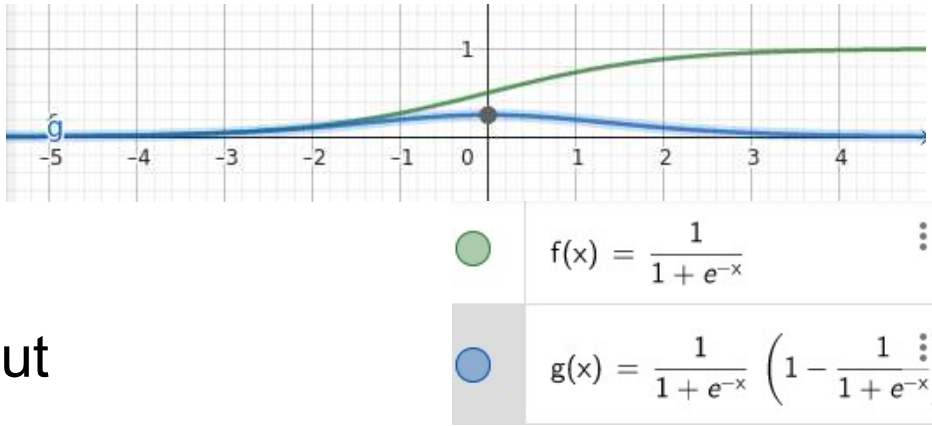
$$\begin{cases} w_1 \cdot 0 + w_2 \cdot 0 + w_3 \cdot 1 = 0 \\ w_1 \cdot 1 + w_2 \cdot 1 + w_3 \cdot 1 = 1 \\ w_1 \cdot 1 + w_2 \cdot 0 + w_3 \cdot 1 = 1 \\ w_1 \cdot 0 + w_2 \cdot 1 + w_3 \cdot 1 = 0 \end{cases}$$

$$w_1 = 1, w_2 = 0, w_3 = 0$$

$$1 \cdot 1 + 0 \cdot 0 + 0 \cdot 0 = 1$$

https://medium.com/technology-invention-and-more/how-to-build-a-simple-neural-network-in-9-lines-of-python-code-cc8f23647ca1

Алгоритм обучения



- 1. Тренировочные наборы input и output
- 2. Случайный выбор весовых коэффициентов w_1, w_2 и w_3
- 3. Расчет суммарного сигнала от входов

$$\sum w_i \cdot x_i = w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3$$

- 4. Расчет выходного значения нейрона

$$f = \frac{1}{1 + e^{-\sum w_i \cdot x_i}}$$

- 5. Расчет поправки к весовым коэффициентам

$$\Delta = error \cdot input \cdot \frac{\partial f}{\partial x} = (output - f) \cdot input \cdot f \cdot (1 - f)$$

- 6. w + \delta –> w, повторить с п. 3.

(neural.py)

```
training_set_inputs:
[[0 0 1]
 [1 1 1]
 [1 0 1]
 [0 1 1]]
```

```
training_set_outputs:
[[0]
 [1]
 [1]
 [0]]
```

```
from numpy import exp, array, random, dot
training_set_inputs = array([[0, 0, 1], [1, 1, 1], [1, 0, 1], [0, 1, 1]])
training_set_outputs = array([[0, 1, 1, 0]]).T
random.seed(1)
synaptic_weights = 2 * random.random((3, 1)) - 1
for iteration in xrange(10000):
    output = 1 / (1 + exp(-(dot(training_set_inputs, synaptic_weights))))
    synaptic_weights += dot(training_set_inputs.T, (training_set_outputs - output) * output * (1 - output))
```

```
synaptic_weights (0-й):
[[-0.16595599],
 [ 0.44064899],
 [-0.99977125]]
```

```
1-й:
[[0.2689864 ]
 [0.3262757 ]
 [0.23762817]
 [0.36375058]]
```

```
2-й:
[[0.29929909]
 [0.44378327]
 [0.32511054]
 [0.41433436]]
```

```
[[0.32550793]
 [0.55437725]
 [0.41995875]
 [0.45332188]]
```

```
[[0.34051508]
 [0.63066567]
 [0.49900324]
 [0.46955296]] ....
```

```
1000-й:
[[0.03178421]
 [0.97414645]
 [0.97906682]
 [0.02576499]]
```

```
print (1 / (1 + exp(-(dot(array([1, 0, 0]), synaptic_weights)))))
```

→ [0.99929937]

Массив 2x1 (neural_single_eq.py)

```
from numpy import exp, array, random, dot
training_set_inputs = array([[0, 1]])
training_set_outputs = array([[0]]).T

iterations = 1000
random.seed(1)
synaptic_weights = 2 * random.random((2, 1)) - 1

for iteration in range(iterations):
    output = 1 / (1 + exp(-(dot(training_set_inputs, synaptic_weights))))
    synaptic_weights += dot(training_set_inputs.T, (training_set_outputs - output) * output * (1 - output))

print( 1 / (1 + exp(-(dot(array([1, 1]), synaptic_weights)))))
print( 1 / (1 + exp(-(dot(array([1, 0]), synaptic_weights)))))
print( 1 / (1 + exp(-(dot(array([0, 0]), synaptic_weights)))))
```

[[0, 1]]

[[0]]

$$w_1 \cdot 0 + w_2 \cdot 1 = 0$$

[[-0.16595599],
 [0.44064899]]

[[0.60841366]]

[[-0.16595599],
 [0.29569657]]

[0.01994744]
[0.45860596]
[0.5]

w1 останется таким, каким был назначен изначально

Массив 10x1 (neural_class_10.py)

```
training_set_inputs = array([[0.5, 1, 0.5, 0, 0, 0, 0, 0, 0, 0], [0, 0.5, 1, 0.5, 0, 0, 0, 0, 0, 0],
                             [0, 0, 0.5, 1, 0.5, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0.5, 1, 0.5, 0, 0],
                             [0, 0, 0, 0, 0, 0, 0.5, 1, 0.5, 0], [0, 0, 0, 0, 0, 0, 0, 0.5, 1, 0.5]])
training_set_outputs = array([[0, 0, 0, 1, 1, 1]]).T
```

```
# [0.5, 1, 0.5, 0, 0, 0, 0, 0, 0, 0]  -> 0
# [0, 0.5, 1, 0.5, 0, 0, 0, 0, 0, 0]  -> 0
# [0, 0, 0.5, 1, 0.5, 0, 0, 0, 0, 0]  -> 0

# [0, 0, 0, 0, 0, 0.5, 1, 0.5, 0, 0]  -> 1
# [0, 0, 0, 0, 0, 0, 0.5, 1, 0.5, 0]  -> 1
# [0, 0, 0, 0, 0, 0, 0, 0.5, 1, 0.5]  -> 1
```

```
Considering new situation  [0 0 0 0 0 0 0 0 0 0] -> ?:
[0.5]
Considering new situation  [0 0 0 0 0 0 0 0 0 1] -> ?:
[0.7899541]
```

Interviewer: What's your biggest strength?

Me: I'm an expert in machine learning.

Interviewer: What's 9 + 10?

Me: Its 3.

Interviewer: Not even close. It's 19.

Me: It's 16.

Interviewer: Wrong. Its still 19.

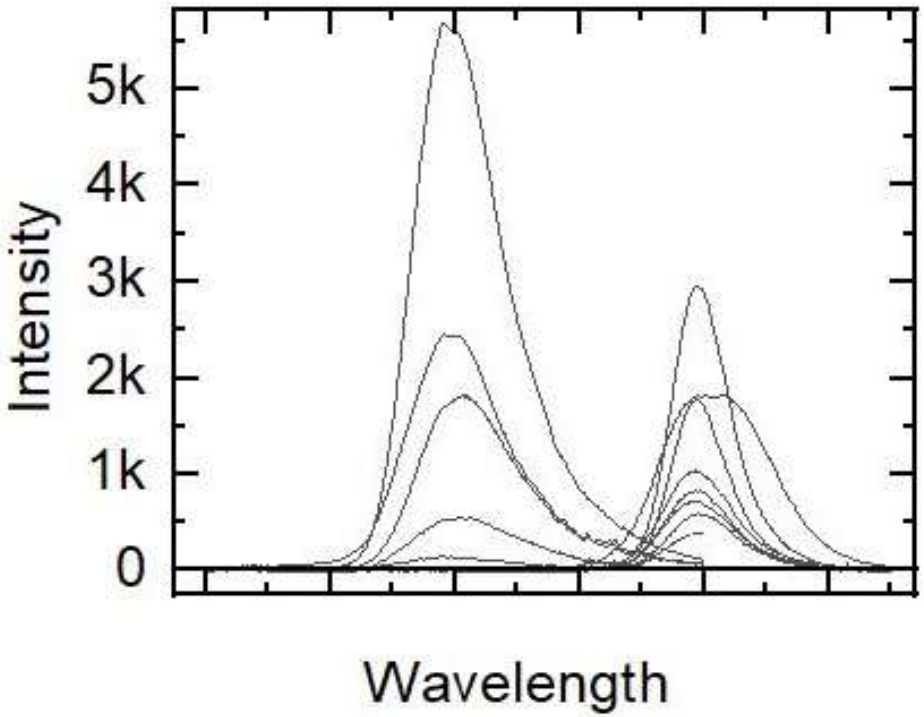
Me: It's 18.

Interviewer: No, it's 19.

Me: it's 19.

Interviewer: You're hired

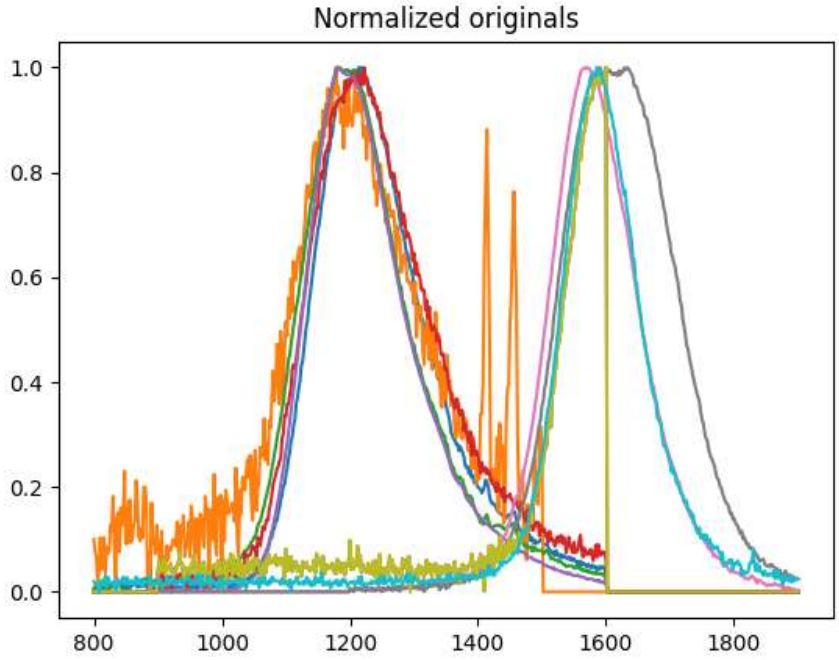
KT PbS1060 + PbS1640 (neuron_pbs.py)



1060:
900.0000
902.0000
...
1600.0000

1060:
800.0000
802.0000
...
1600.0000

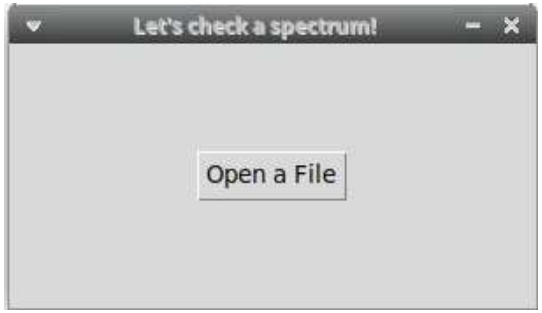
1640:
1200.0000
1202.0000
...
1900.0000

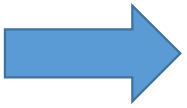


- 0 – PbS1060
- 1 – PbS1640
- Порог - 0,5

training_set_outputs = array([[0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1]]).T

inputs = 551
trainings = 3 #10000





```
if result < 0.5:
    mess = '1060'
else:
    mess = '1640'
```