Основы программирования для исследования наноструктур

Однослойный перцептрон и распознавание спектра

Парфенов П.С.

Университет ИТМО / НОЦ Физика наноструктур

Какое значение выхода системы?

	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	**?**

https://www.agiratech.com/how-to-build-a-simple-neural-network-in-9-lines-of-python-code

А если решить систему уравнений?

	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	**?**

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

А если решить систему уравнений?

	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	**?**

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

“Решение”* системы уравнений:

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

Выходное значение:

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

*а если система не имеет решения?

- Решение системы уравнений матричный способом

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \end{cases} \quad \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \bullet \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \quad A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}, \; X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \; B = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

$$A \bullet X = B$$

$$A^{-1} \bullet A \bullet X = A^{-1} \bullet B$$

$$X = A^{-1} \bullet B$$

https://function-x.ru/systems_matrix_method.html
http://mathprofi.ru/kak_naiti_obratnuyu_matricu.html

Алгоритм поиска решения

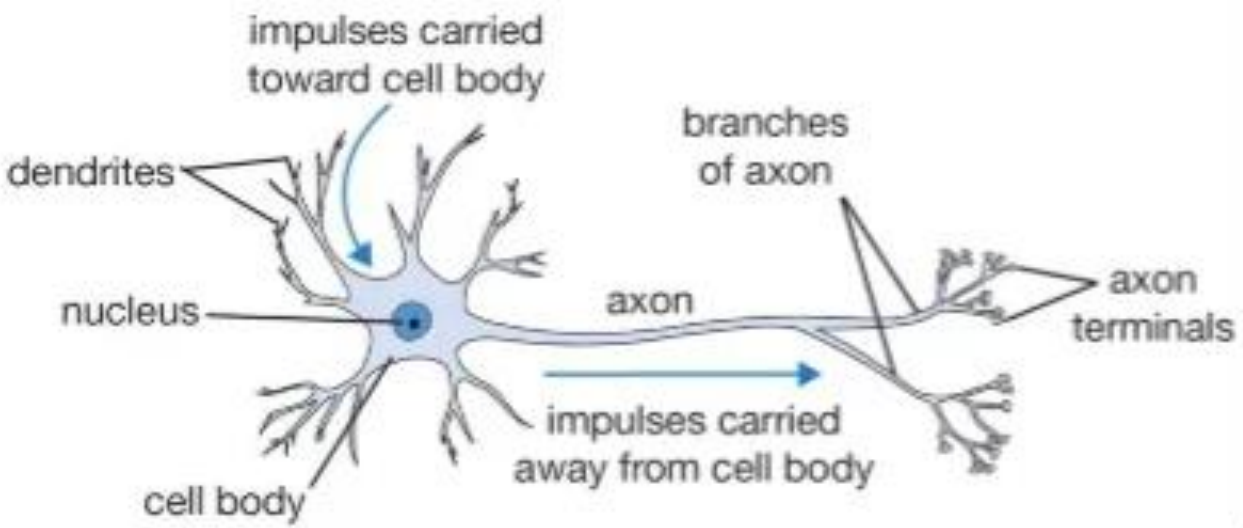
1. Тренировочные наборы input [[0, 0, 1], [1, 1, 1], [1, 0, 1], [0, 1, 1]] и output [[0], [1], [1], [0]]
2. Начальный набор весовых коэффициентов w_1, w_2 и w_3 (пусть w=[0,0,0])
3. Расчет выходных значений системы уравнений input × w
4. Расчет ошибки error = input × w − output
5. Расчет поправки к весовым коэффициентам Delta = $input^T$ × error
6. Прибавление поправки к весовым коэффициентам w += Delta
7. Повторить пп. 3-6 нужное количество раз (пока ошибка не станет меньше требуемого).
8. Вычислить выходное значение по полученным весовым коэффициентам

```python
import numpy as np
training_set_inputs = np.array([[0, 0, 1], [1, 1, 1], [1, 0, 1], [0, 1, 1]])
training_set_outputs = np.array([[0], [1], [1], [0]])
# или training_set_outputs = np.array([[0, 1, 1, 0]]).T
synaptic_weights = np.array([[0.], [0.], [0.]])
l_rate = 0.1
trainings = 100

for iteration in range(trainings):
    output = np.dot(training_set_inputs, synaptic_weights)
    error = training_set_outputs - output
    synaptic_weights += np.dot(training_set_inputs.T, l_rate*error)

print(f'Итоговый результат {np.dot(np.array([1, 0, 0]), synaptic_weights)}')
print(f'Итоговая ошибка = {np.sum(np.multiply(error,error)):.3f}')
```

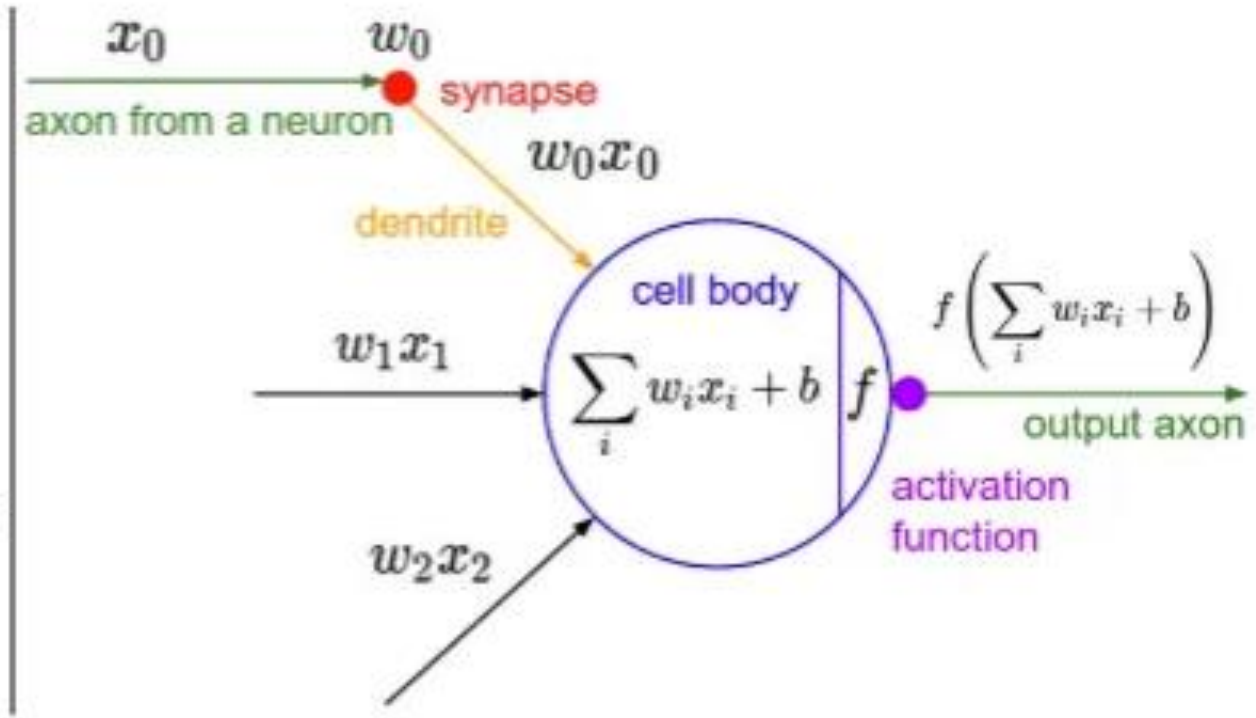
Итоговый результат [0.9995378]
Итоговая ошибка = 0.000

Нейрон и его математическая модель

- Нейрон



- однослойный перцептрон

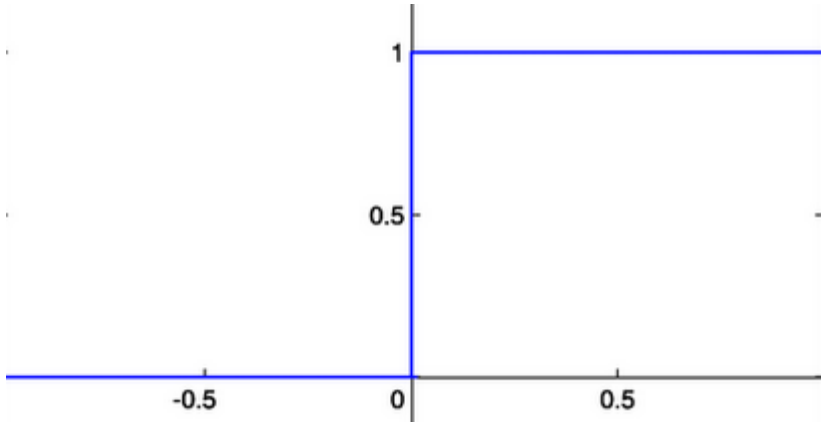


https://srijayjk.medium.com/activation-functions-in-dnn-21729d529364

Функция активации

- Пороговая передаточная функция

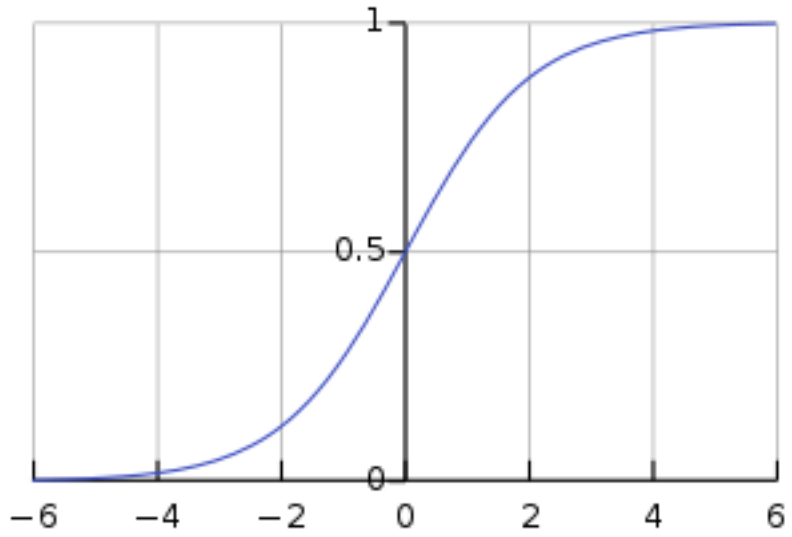
$$f(x) = \begin{cases} 1 & \text{if } x \geq T \\ 0 & \text{else} \end{cases}$$



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

- 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	**?**

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

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

https://www.agiratech.com/how-to-build-a-simple-neural-network-in-9-lines-of-python-code

Алгоритм обучения

- 1. Тренировочные наборы input и output
- 2. Случайный выбор весовых коэффициентов w_1, w_2 и w_3
- 3. Расчет выходных значений f(input × w)

$$f = 1/\left(1 + e^{-\sum w_i \cdot x_i}\right)$$

- 4. Расчет ошибки error = (f − output)
- 5. Расчет поправки к весовым коэффициентам

$$\Delta = input^T \times \left(error \cdot \frac{\partial f}{\partial x}\right) = input^T \times (error \cdot f \cdot (1 - f))$$

- 6. w = w + Δ, повторить с п. 3.

```
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
synaptic_weights =  [[0], [0], [0]]
for iteration in range(1000):
    output = 1 / (1 + exp(-(dot(training_set_inputs, synaptic_weights))))
    synaptic_weights += dot(training_set_inputs.T, (training_set_outputs - output) * output * (1 - output))
result = 1 / (1 + exp(-(dot(array([1, 0, 0]), synaptic_weights))))
```

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

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

```
synaptic_weights (0-й):
[[0],
 [0],
 [0]]
```

```
1-й:
[[0.5]
 [0.5]
 [0.5]
 [0.5]]
```

```
2-й:
[[0.5      ]
 [0.5621765]
 [0.5621765]
 [0.5      ]]
```

```
3-й:
[[0.4913825 ]
 [0.60200263]
 [0.60612512]
 [0.48707534]]
```

```
10-й:
[[0.38433627]
 [0.70154679]
 [0.7389527 ]
 [0.34140463]] ....
```

```
1000-й:
[[0.031916  ]
 [0.9740661 ]
 [0.97911535]
 [0.02573281]]
```

Результат: [0.99929804]

Случайные весовые коэффициенты

```
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 range(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))

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

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

Массив 10x1

```
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]