# Системы искусственного интеллекта

«02.03.03 - Математическое обеспечение и администрирование информационных систем направленность разработка и администрирование информационных систем»

http//vikchas.ru

https://www.famous-scientists.ru/3653/

## Лекция 9 «Нейронные сети»

Часовских Виктор Петрович
д-р техн. наук, профессор кафедры ШИиКМ

ФГБОУ ВО «Уральский государственный экономический университет»

Екатеринбург 2022

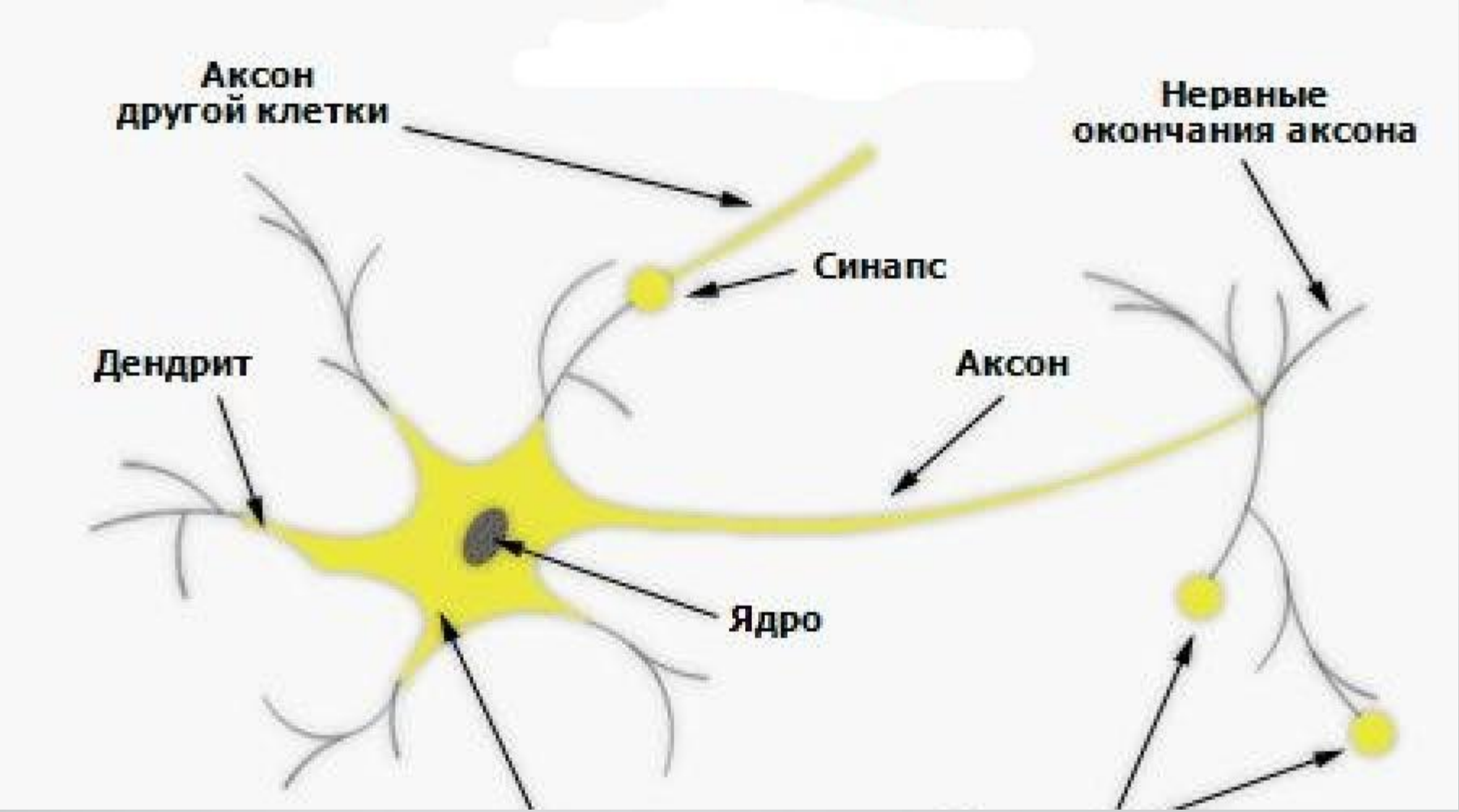
# Искусственный нейрон как основа нейронных сетей

Наш мозг представляет собой сложнейшую биологическую нейронную сеть, которая принимает информацию от органов чувств (глаза, уши, нос), каким-то образом ее обрабатывает (узнает лица, распознает речь, ощущает запахи и т. д.).

На основе полученных данных дает команды различным исполнительным органам (пожать руку знакомому человеку, пойти к доске по просьбе учителя, покинуть помещение при появлении дыма).

Все эти действия выполняет наш мозг, у которого есть биологическая нейронная сеть, состоящая из совокупности нейронов. В головном мозге человека общая нейронная сеть состоит примерно из 90 млрд нейронов, которые соединены друг с другом миллиардными связями. Это, пожалуй, самый сложный объект, известный нам, людям. Основой этой сети является нейрон. Упрощенное строение биологического нейрона показано на следующим рисунке:

# Упрощенная схема нейрона



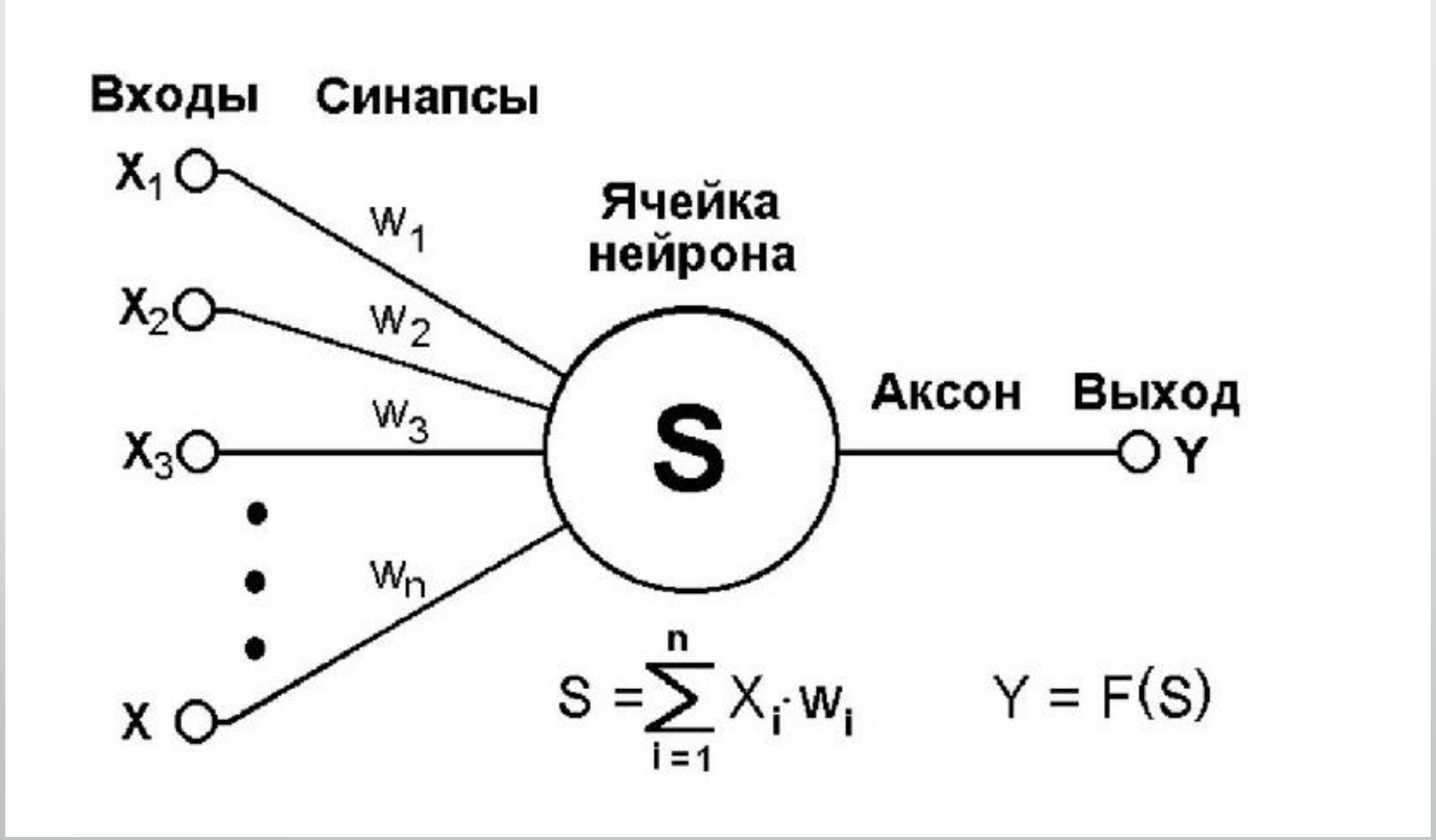

**Дендрит** -получает информацию. **Синапс** -место контакта между двумя нейронами и передачи импульса. **Аксон** - импульсы идут от тела клетки к другому нейрону.

Биологический нейрон - это нервная клетка, которая состоит из **тела**, **дендритов** и **аксона**.

Тело соединено со множеством коротких и толстых отростков, которые называются дендритами. Это входные отростки - они принимают **импульсы** с тех нейронов, которые находятся в сети раньше, чем этот нейрон. Имеется один очень длинный и тонкий отросток, который называется аксоном.

Это выходной отросток, по которому нейрон **передает** электрохимический импульс следующим нейронам в сети. Через множество дендритов нейрон получает входные сигналы, в теле нейрона они обрабатываются, а через единственный аксон выходной импульс от нейрона передается дальше. Окончание аксона имеет разветвления, через которые выходной сигнал может быть передан нескольким следующим нейронам. Реальный биологический нейрон является достаточно сложной системой. Во многом это объясняется тем, что нейрон, помимо обработки сигнала ( основное его назначение), вынужден еще выполнять множество других функций, поддерживающих его жизнь. Более того, сам механизм передачи сигнала от нейрона к нейрону тоже очень сложный с биологической и химической точек зрения.

Достаточно упрощенная схема **искусственного нейрона** представлена на следующим рисунке:



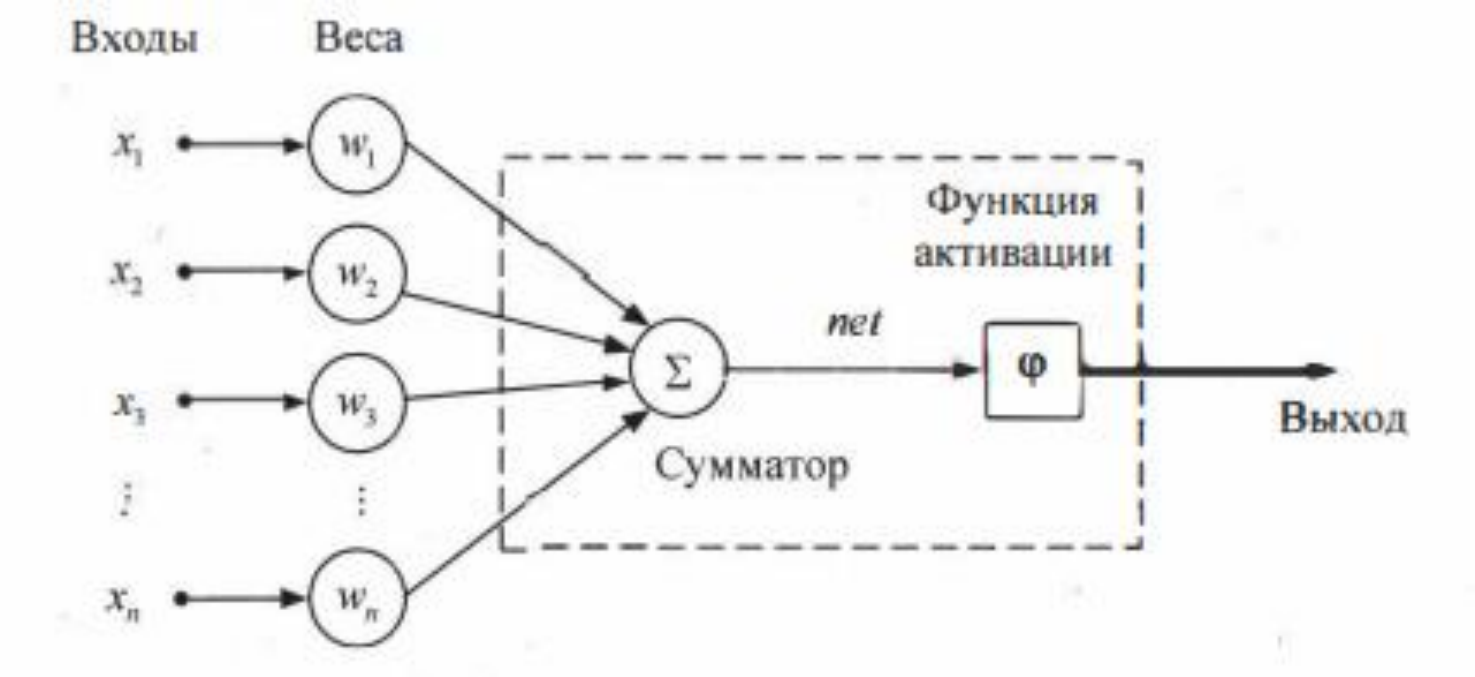
Здесь тело нейрона представлено в виде "черного ящика", который принимает входные данные $(x_1, x_2, x_3 \ldots x_n)$, выполняет с ними некоторые операции, а затем выводит результат - y.

Как нейроны взаимодействуют между собой в нейронной сети? Дело в том, что аксон каждого нейрона на своем конце имеет большое количество так называемых синапсов.

Синапс - это место соединения выходного аксона одного нейрона с входными дендритами другого нейрона. Через синапс передается нервный импульс. А сам нервный импульс формируется в теле нейрона, и фактически тело нейрона выступает сумматором, который принимает все входные импульсы, взвешивает их, передает в некоторую активирующую функцию и принимает решение, запускать импульс дальше по своему аксону или нет.

Решение принимается очень просто - **если суммарные входные импульсы превышают некий заданный порог, то выходной импульс запускается.**

Для реализации искусственного нейрона нужно построить его упрощенную **математическую модель**, в которой реализованы его базовые функции без учета функций его жизнеобеспечения




Данная модель довольно простая. На вход математического нейрона поступает некоторое количество входных параметров $x_1, x_2, x_3 \ldots x_n$, т. е. дендритов, каждый входной параметр имеет свой вес- $w_1, w_2, w_3 \ldots w_n$.

В теле искусственного нейрона имеется **сумматор**, где каждый входной сигнал умножается на некоторый действительный весовой коэффициент и формируется итоговая сумма.

Полученное значение передается в функцию активации.

На выходе осуществляется проверка значения функции активации.

Если ее значение выше некоторого порога, то на выход из тела нейрона передается значение единица. В этом случае нейрон активируется и в аксон передается соответствующий сигнал.

Если же итоговое значение в функции активации ниже порога, то на выход из тела нейрона подается значение, равное нулю, и нейрон считается не активированным.

Рассмотрим, упрощенно, как работает эта модель. У нейрона есть входы, через которые он принимает сигнал -X.

Для входных сигналов вводится понятие весов - W, на которые умножаются эти сигналы. В теле искусственного нейрона поступившие на входы сигналы умножаются на их веса. Сигнал первого входа $x_1$ умножается на соответствующий этому входу вес $w_1$, $x_2$ на $w_2$ , и так до последнего n-го входа. Затем в сумматоре эти произведения суммируются.

В итоге мы получаем сумму произведений значений входных сигналов на их веса S:

$$\mathbf{S = x_1 w_1 + x_2 w_2 + x_3 w_3 + \cdots . + x_n w_n}$$

Роль сумматора в данном случае очевидна: он преобразует все входные сигналы (которых может быть много) в одно число - **взвешенную сумму**, которая характеризует поступивший на нейрон сигнал в целом.

Итак, на вход искусственного нейрона было подано множество входных сигналов X с их весами W, а в сумматоре мы получили единственное число - **S**.

# Программа сумматора в языке Python

```
# Модуль Neuron
import numpy as np    # установить numpy
# Создание класса нейрон
class Neuron:
    def __init__(self, w):
        self.w = w
    def y(self, x):  # Сумматор
        s = np.dot(self.w, x)  # Суммируем входы
        return s  # функция активации
Xi = np.array([2, 3])  # Задание значений входам
Wi = np.array([1, 1])  # Веса входных сенсоров
n = Neuron(Wi)  # Создание объекта из класса Neuron
print('S1= ', n.y(Xi))  # Обращение к нейрону
Xi = np.array([5, 6])  # Веса входных сенсоров
print('S2= ', n.y(Xi))  # Обращение к нейрону
Xi = np.array([1, 0, 0, 1])  # Задание значений
входам
Wi = np.array([5, 4, 3, 1])  # Веса входных сенсоров
n = Neuron(Wi)  # Создание объекта из класса Neuron
print('S= ', n.y(Xi))  # Обращение к нейрону
```

**Результаты выполнения:**

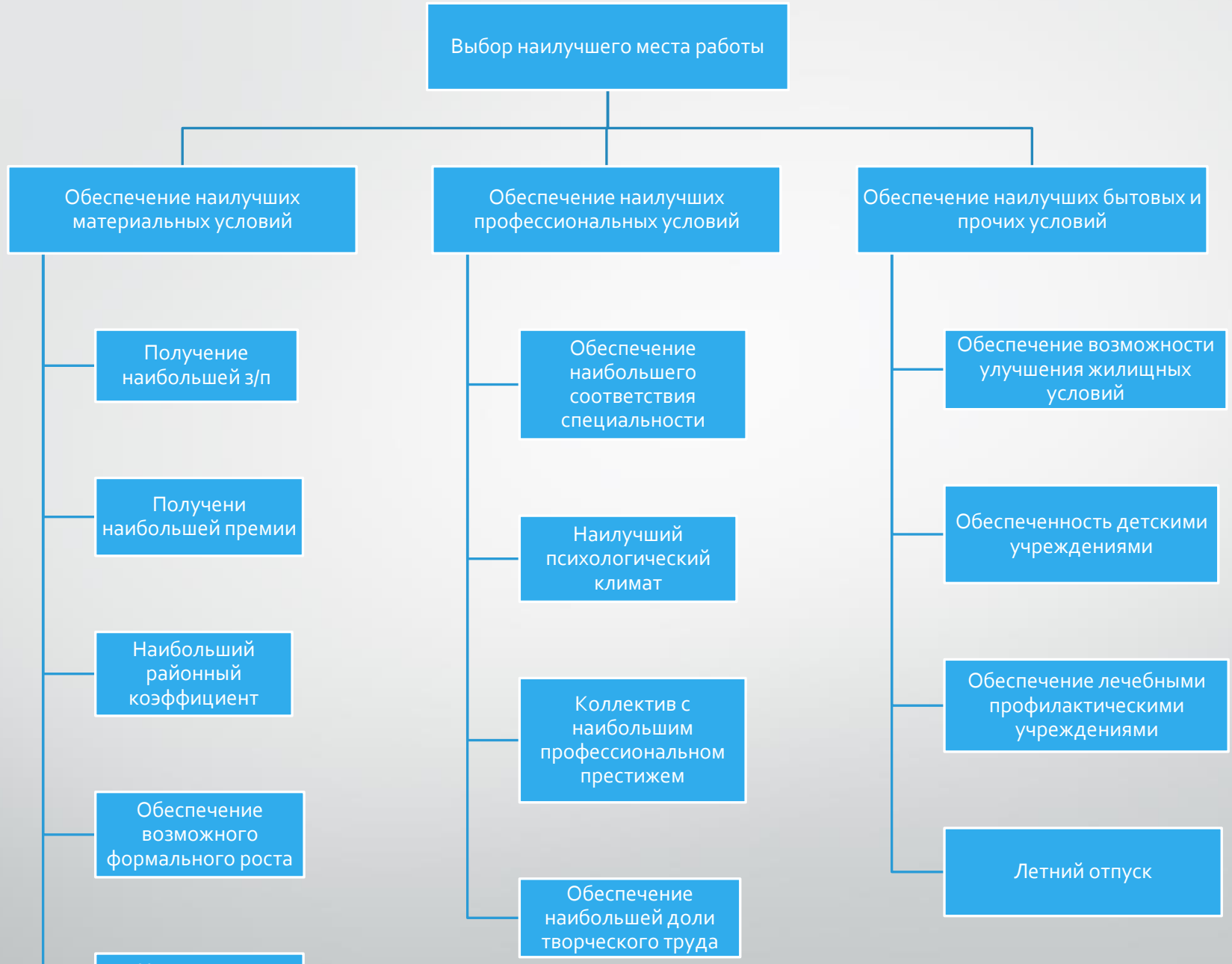
**S1 = 5**

**S2 = 11**

**S = 6**

Теперь перейдем к следующему компоненту искусственного нейрона - функции активации.

Для понимания принципа работы данного компонента рассмотрим простой пример. Допустим, у нас есть один искусственный нейрон, роль которого – выбрать наилучшее место работы. Это типичная задача, в которой нужно проанализировать сочетание множества факторов и на основе этого анализа принять итоговое решение. Для простоты рассмотрим всего три фактора на первом уровне, влияющих на выбор места работы, и три локальных множества факторов на втором уровне. Рассмотрим нейрон для первой составляющей первого уровня - обеспечение наилучших материальных условий. На входы в нейрон мы подадим исходные данные в соответствие с деревом целей:


Выбор наилучшего места работы
Обеспечение наилучших материальных условий
Обеспечение наилучших профессиональных условий
Обеспечение наилучших бытовых и прочих условий
Получение наибольшей з/п
Получени наибольшей премии
Наибольший районный коэффициент
Обеспечение возможного формального роста
Наименьшие транспортные расходы
Обеспечение наибольшего соответствия специальности
Наилучший психологический климат
Коллектив с наибольшим профессиональном престижем
Обеспечение наибольшей доли творческого труда
Обеспечение возможности улучшения жилищных условий
Обеспеченность детскими учреждениями
Обеспечение лечебными профилактическими учреждениями
Летний отпуск

$x_1$ – Получение наибольшей з/п
$x_2$ –Получени наибольшей премии
$x_3$ – Наибольший районный коэффициент
$x_4$ –Обеспечение возможного формального роста
$x_5$ –Наименьшие транспортные расходы

Влияние этих факторов является понятным на обеспечение наилучших материальных условий.

Для упрощения нашей модели присвоим всем этим параметрам логические значения в виде цифр 0 или 1.

Например, если з/п меньше наших желаний, то 0, если больше , то значение 1.

У нашего нейрона есть пять входов, должно быть и пять весовых коэффициентов.

В нашем примере весовые коэффициенты можно представить как показатели · важности каждого входа, влияющие на общее решение нейрона. Чем больше значение коэффициента, тем больше важность входного параметра. Веса входов распределим следующим образом:

W1 =5;  W2 =4; W3 =1; W4 = 2; W5 = 3.

По заданным весовым коэффициентам нетрудно заметить, что очень важную роль играет з/п.

Пусть на входы нашего нейрона мы подаем следующие сигналы 2-х организаций:

| | |
|---|---|
| $x_{11}$ −1 | $x_{21}$ −1 |
| $x_{12}$ −0 | $x_{22}$ −1 |
| $x_{13}$ −0 | $x_{23}$ −0 |
| $x_{14}$ −1 | $x_{24}$ −0 |
| $x_{15}$ −0 | $x_{25}$ −1 |

При поступлении этой информации в сумматор он выдаст следующие итоговые суммы:

Первая организация $S1 = x_{11}w_1 + x_{12}w_2 + x_{13}w_3 + x_{14}w_4 + x_{15}w_5$

Вторая организация $S2 = x_{21}w_1 + x_{22}w_2 + x_{23}w_3 + x_{24}w_4 + x_{25}w_5$

Наибольшее значение итоговой суммы определяет организацию (1-ю или 2-ю) с наилучшим вариантом для составляющей первого уровня дерева целей «Обеспечение наилучших материальных условий».

Проверим это с помощью нашей программы-сумматора, представленной ранее.

Для этого изменим значения соответствующих параметров в программном коде сумматора.

```
Xi = np.array([1, 0, 0, 1, 0])  # Задание значений входам 1-ой организации
Wi = np.array([5, 4, 1, 2, 3])  # Веса входных сенсоров
n = Neuron(Wi)                  # Создание объекта из класса Neuron
print('S1= ', n.y(Xi))           # Обращение к нейрону
```

**Получим S1 = 7**

```
Xi = np.array([1, 1, 0, 0, 1])  # Задание значений входам 2 – ой организации
Wi = np.array([5, 4, 1, 2, 3])  # Веса входных сенсоров
n = Neuron(Wi)                  # Создание объекта из класса Neuron
print('S2= ', n.y(Xi))          # Обращение к нейрону
```

**Получим S2 = 12**

Очевидно, нам нужно как-то преобразовать взвешенные суммы s1 и s2 в итоговое решение. Эту задачу и решает функция активации.

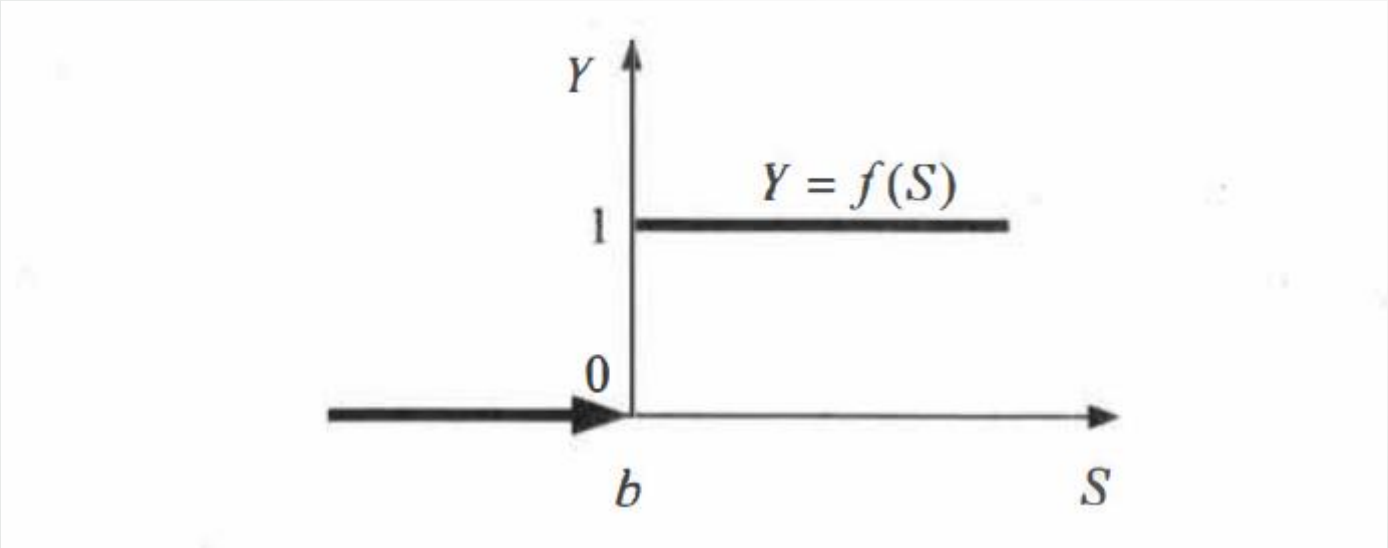
Функция **активации** (activation function)- это такая функция, которая в качестве входного параметра получает взвешенную сумму S как аргумент, а на выходе формирует значение выходного сигнала из нейрона (y). В общем виде эту функцию можно описать выражением

$$y= f(S).$$

Рассмотрим некоторые математические функции, которые могут быть использованы в качестве функции активации.

# Функция единичного скачка

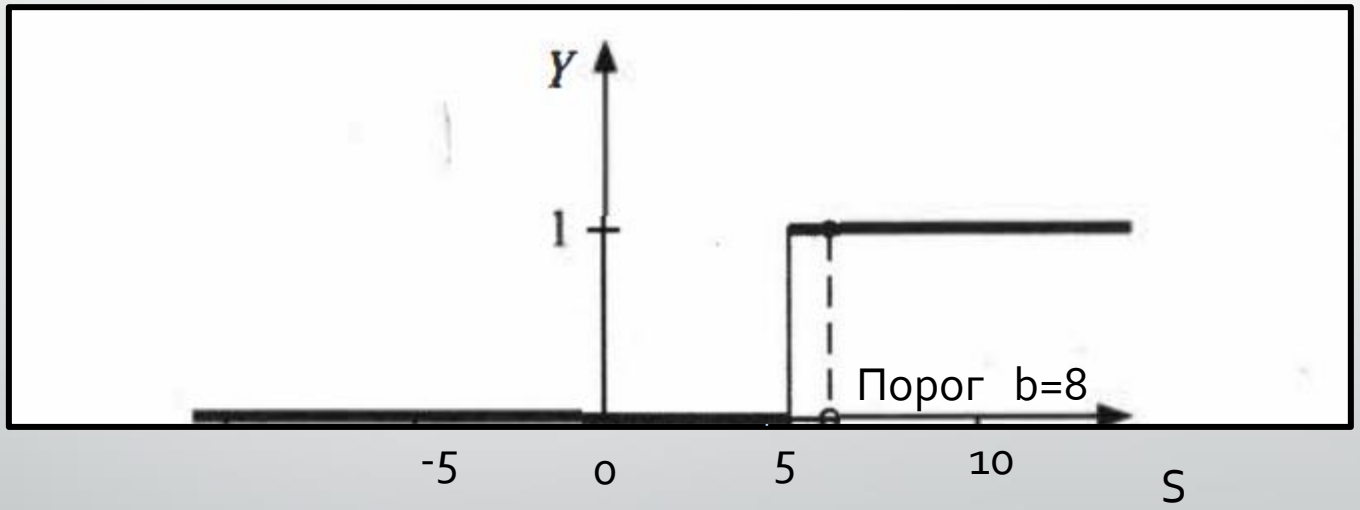
Общий вид функции единичного скачка (или функции Хевисайда) следующий




На горизонтальной оси данной функции расположены величины взвешенной суммы S. На вертикальной оси - значения выходного сигнала У. Это самый простой вид функции активации. При использовании такой функции выход из нейрона У может получать только два значения - 0 или 1.

Какое значение получит выходной сигнал У на выходе из функции активации, зависит от величины **порогового значения** b.

Если взвешенная сумма S будет больше порога b, то выход нейрона получит значение единица (У = 1). Если сумма S окажется меньше или равное пороговому значения b, то выход из нейрона получит значение ноль (У = 0).




В нашем примере взвешенная сумма 12, больше 8. Значит, пороговая функция на выходе выдаст значение для s2 Y= 1, а для s1 Y = 0/

```
# Модуль onestep
import numpy as np
def onestep(x):
    b = 8     # пороговое значение b
    if x >= b:
        return 1
    else:
        return 0
# Создание класса нейрон
class Neuron:
    def __init__(self, w):
        self.w = w
    def y(self, x):  # Сумматор
        s = np.dot(self.w, x)   # Суммируем входы
        return onestep(s)  # функция активации

Xi = np.array([1, 1, 0, 0, 1])  # Задание значений входам 2 - ой организации
Wi = np.array([5, 4, 1, 2, 3])  # Веса входных сенсоров
n = Neuron(Wi)  # Создание объекта из класса Neuron
print('Y= ', n.y(Xi))  # Обращение к нейрону
```

Y=**1** для 2-ой организации. Y= 0 для 1-ой организации Xi = np.array([1, 0, 0, 1, 0])

**Принимаем решение для 2 – ой организации**.

# Сигмоидальная функция активации

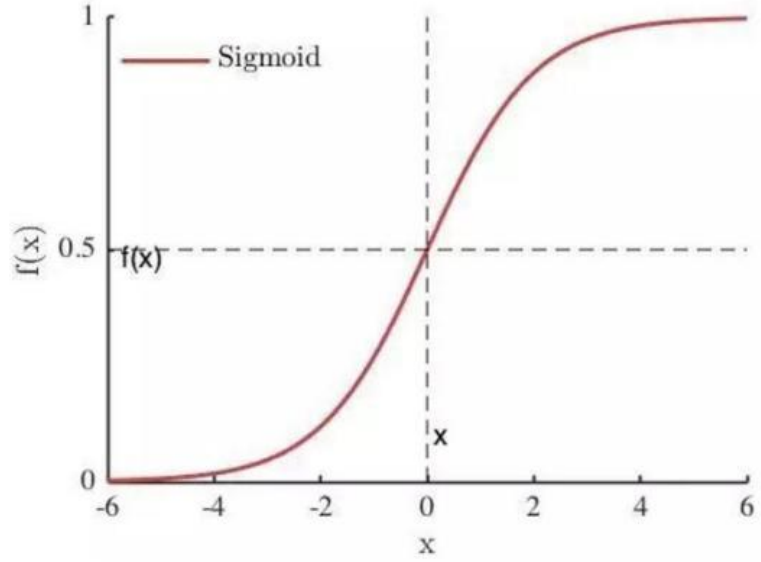
На самом деле существует целое семейство сигмоидальных - **логических** функций, и некоторые из них применяют в качестве функции активации в искусственных нейронах.

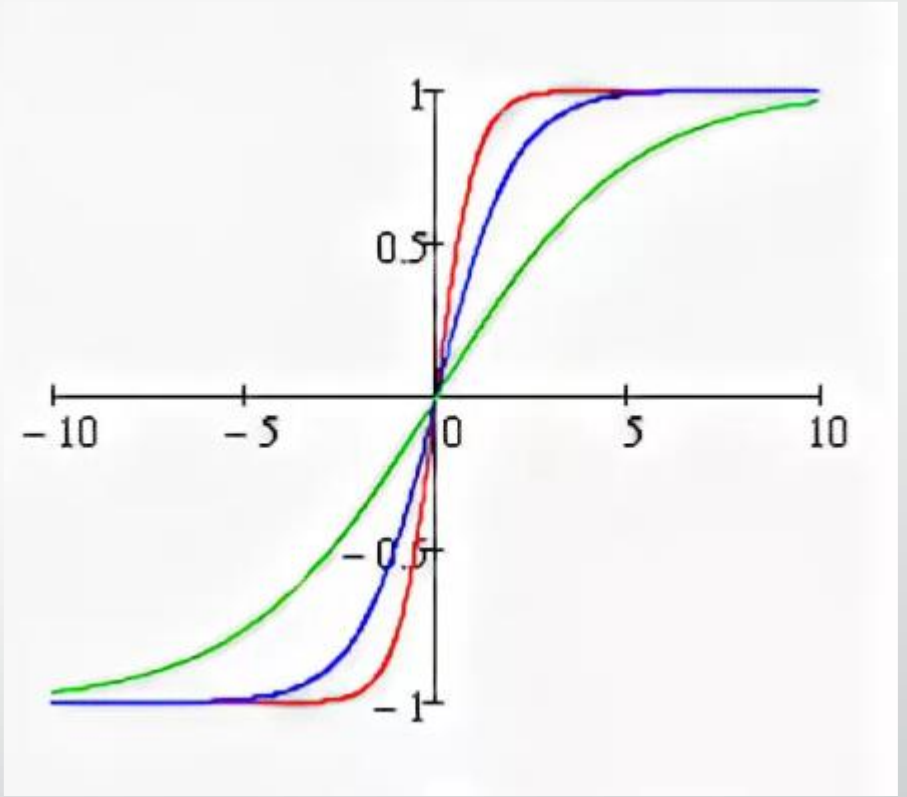
Все эти функции обладают полезными свойствами, ради которых их и применяют в нейронных сетях. Эти свойства станут очевидными после того, как мы более детально ознакомится с графиком этих функций.

Итак, наиболее часто используемая в нейронных сетях функция-это сигмоида, или логистическая. функция  График этой функции выглядит достаточно просто, а внешний вид имеет некоторое подобие английской буквы S. А вот так она записывается аналитически:

$$Y = \frac{1}{1+\exp(-aS)}$$

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




a = 1 a = ½ a=2

**a** это некое число, которое характеризует степень крутизны функции.

При использовании логистической функции мы получаем вероятностный итоговый результат в виде числа между 0 и 1.

Причем чем больше взвешенная сумма S, тем ближе выход У будет к 1 (но никогда не будет точно ей равен). И наоборот, чем меньше взвешенная сумма S, тем выход нейрона У будет ближе к 0.

Логистическая функция имеет следующие свойства:

- она является "сжимающей" функцией, т. е. вне зависимости от аргумента (взвешенной суммы S) выходной сигнал У всегда будет в пределах от 0 до 1;
- она более гибкая, чем функция единичного скачка - ее результатом может быть не только 0 и 1, но и любое число между ними;
- во всех точках она имеет производную, и эта производная может быть выражена через эту же функцию

Чем ближе значение У к единице, тем больше вероятность того, что нужно принимать положительное решение.

И наоборот, чем ближе значение У к нулю, тем большая вероятность принятия отрицательного решения.

Например, в нашем примере при значении выхода из нейрона У= 0,8, скорее всего, все-таки стоит отдать предпочтение этому варианту. Если же значение У= 0, 2, то это означает, что вам почти наверняка нужно отказаться от такого варианта.

Для того чтобы однозначно определить итоговое решение, **нужно задать величину порогового значения**.

Например, если задать пороговое значение b = 0,6, то при рассчитанной величине У > 0,6 всегда принимаем вариант, при У< 0,6 отказываемся от варианта.

# Используем сигмоиду в качестве функции активации в программе

```
# Модуль sigmoid
import numpy as np
# функция активации: f(x) = 1 / (1 + e^(-x))
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
# Создание класса нейрон
class Neuron:
    def __init__(self, w):
        self.w = w
    def y(self, x):  # Сумматор
        s = np.dot(self.w, x)   # Суммируем входы
        return sigmoid(s)  # функция активации
Xi = np.array([1, 1, 0, 0, 1])  # Задание значений входам 2 – ой организации
Wi = np.array([5, 4, 1, 2, 3])  # Веса входных сенсоров
n = Neuron(Wi)  # Создание объекта из класса Neuron
print('Y= ', n.y(Xi))  # Обращение к нейрону
```

```
Y=  0.9999938558253978
```