Введение в нейронные сети

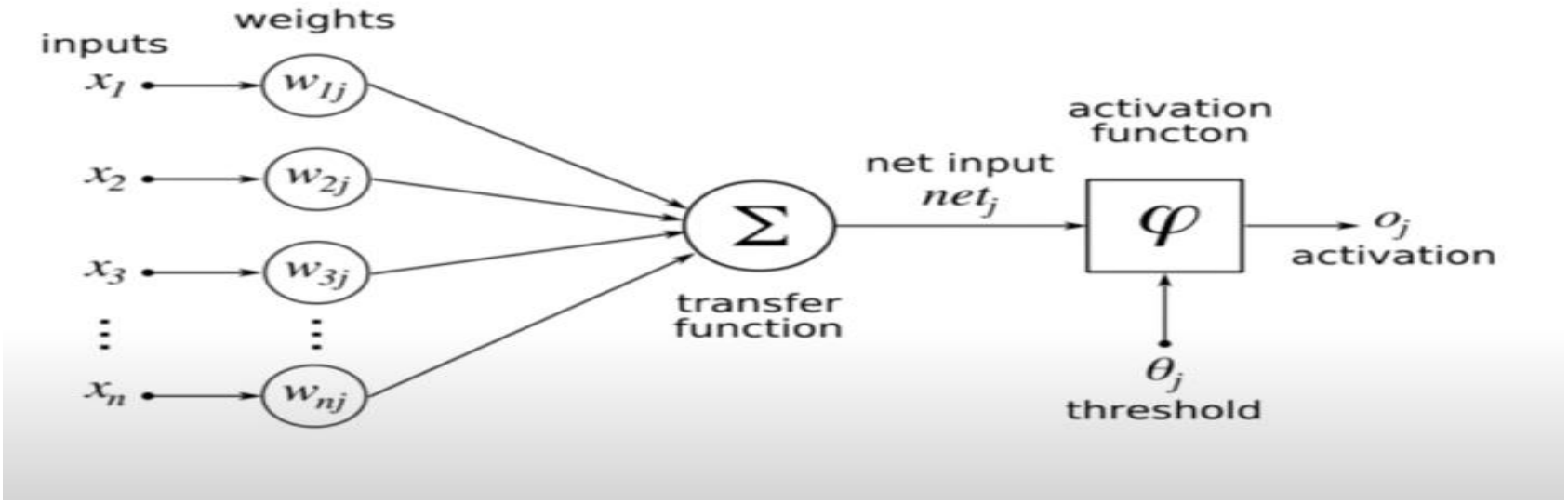
Семинар 15

Введение в нейронные сети

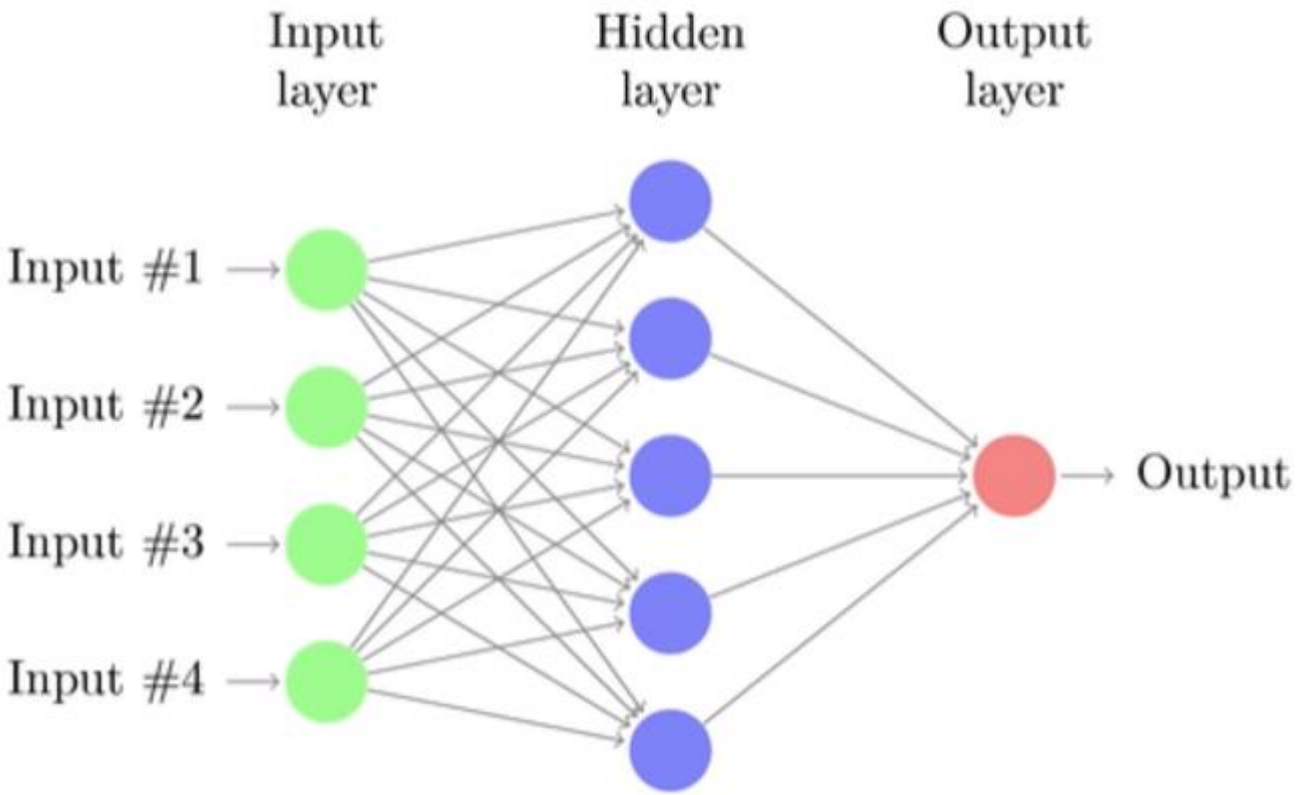
- Нейрон – единица, хранящая информацию, сохраняющая алгоритм в вашей голове.

Основные свойства нейронов:

- Передача сигналов различной мощности (веса).
- Xi – входящие сигналы, у каждого сигнала свой вес wi



Пример простой сети



Зеленый – входящие нейроны
Фиолетовый – передаточные нейроны (недоступные), внутренний (обобщающий) слой
Красный – нейрон, принимающий результат

Искусственные нейросети

- У человека 10^11 нейронов
- У лягушки 10^7 нейронов
- У мухи 10^6 нейронов

Искусственные нейросети

- Что делает искусственный нейрон?

Он считает взвешенную сумму на своих входах и решает, следует это значение исключать или использовать дальше.

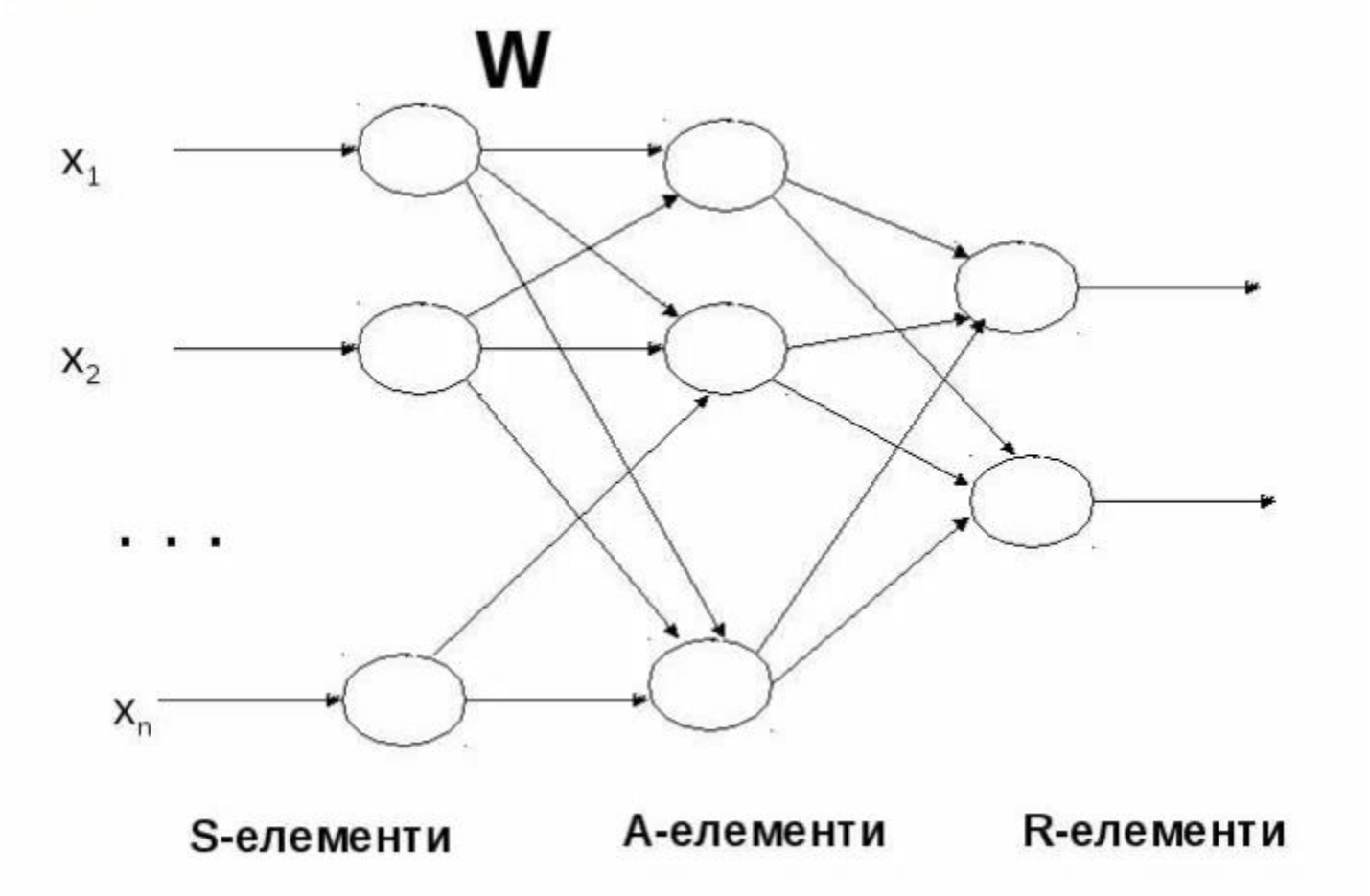
- Как мы решаем, должен ли нейрон быть активирован?

Для этой цели решили добавлять активационную функцию. Она проверяет произведенное нейроном значение *Y* на предмет того, должны ли внешние связи рассматривать этот нейрон как активированный, или его можно игнорировать.

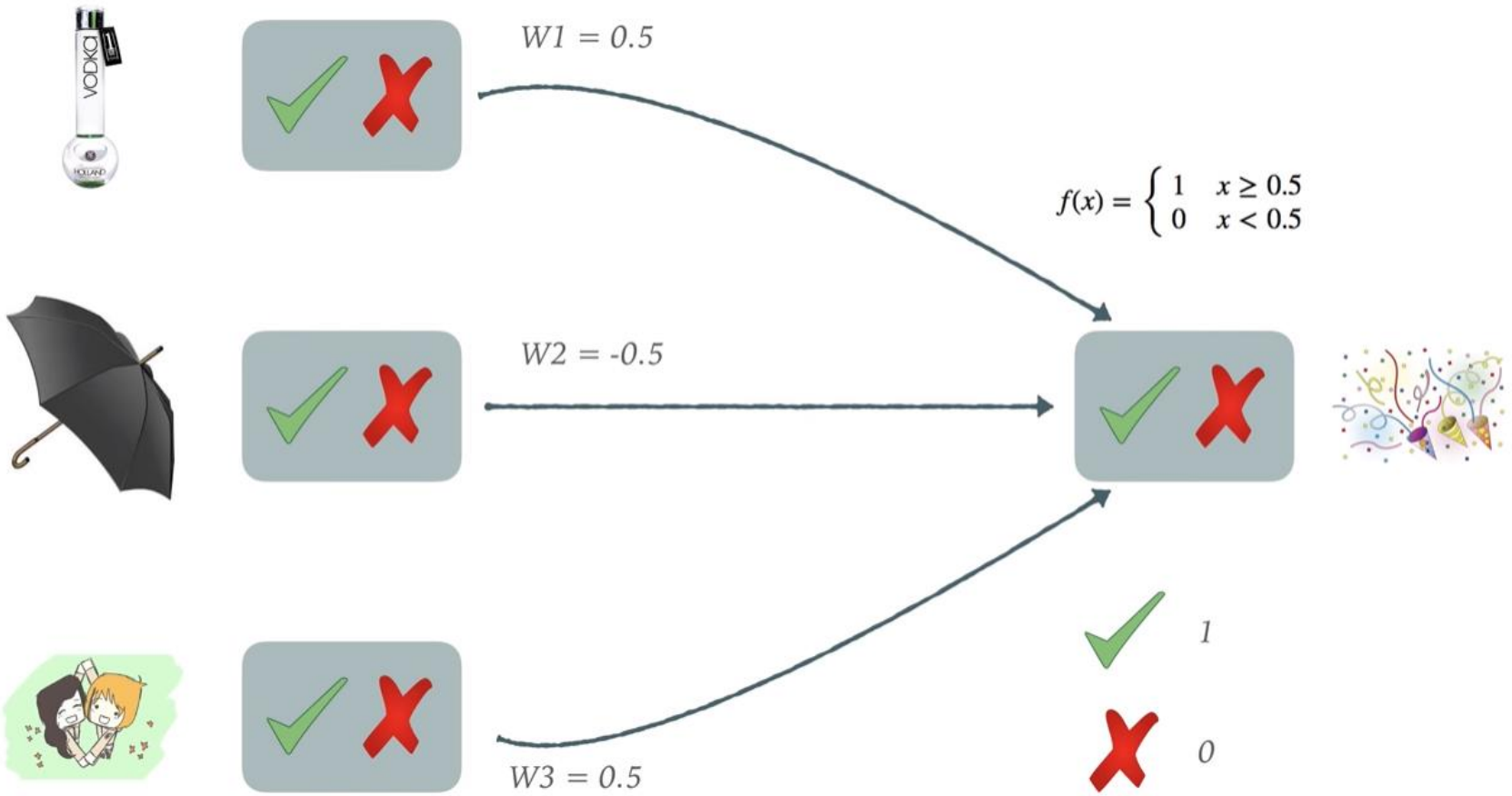
Перцептрон

- Перцептрон – матемаматическая или компьютерная модель восприятия информации мозгом. Перцептрон стал одной из первых моделей нейросетей.
- Перецептрон имеет 3 типа элементов: поступающие от **датчиков** **сигналы** передаются **ассоциативным** элементам, а затем **реагирующим** элементам. Таким образом, перцептроны позволяют создать набор ”ассоциаций” между входными стимулами и необходимой реакцией на выходе. В биологическом плане это соответствует преобразованию, например, зрительной информации в физиологический ответ от двигательных нейронов.

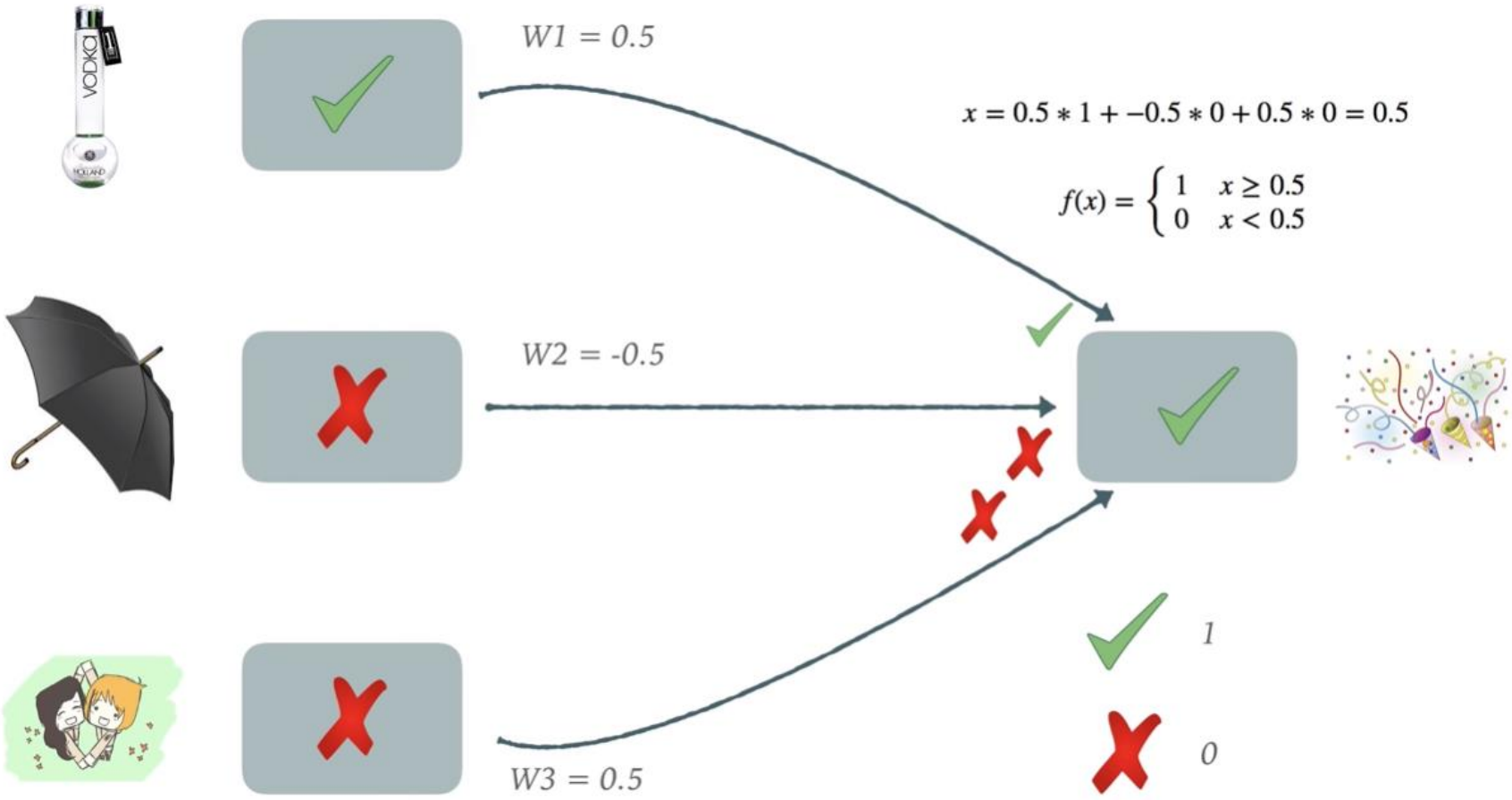
Перцептрон



Принцип работы нейросети



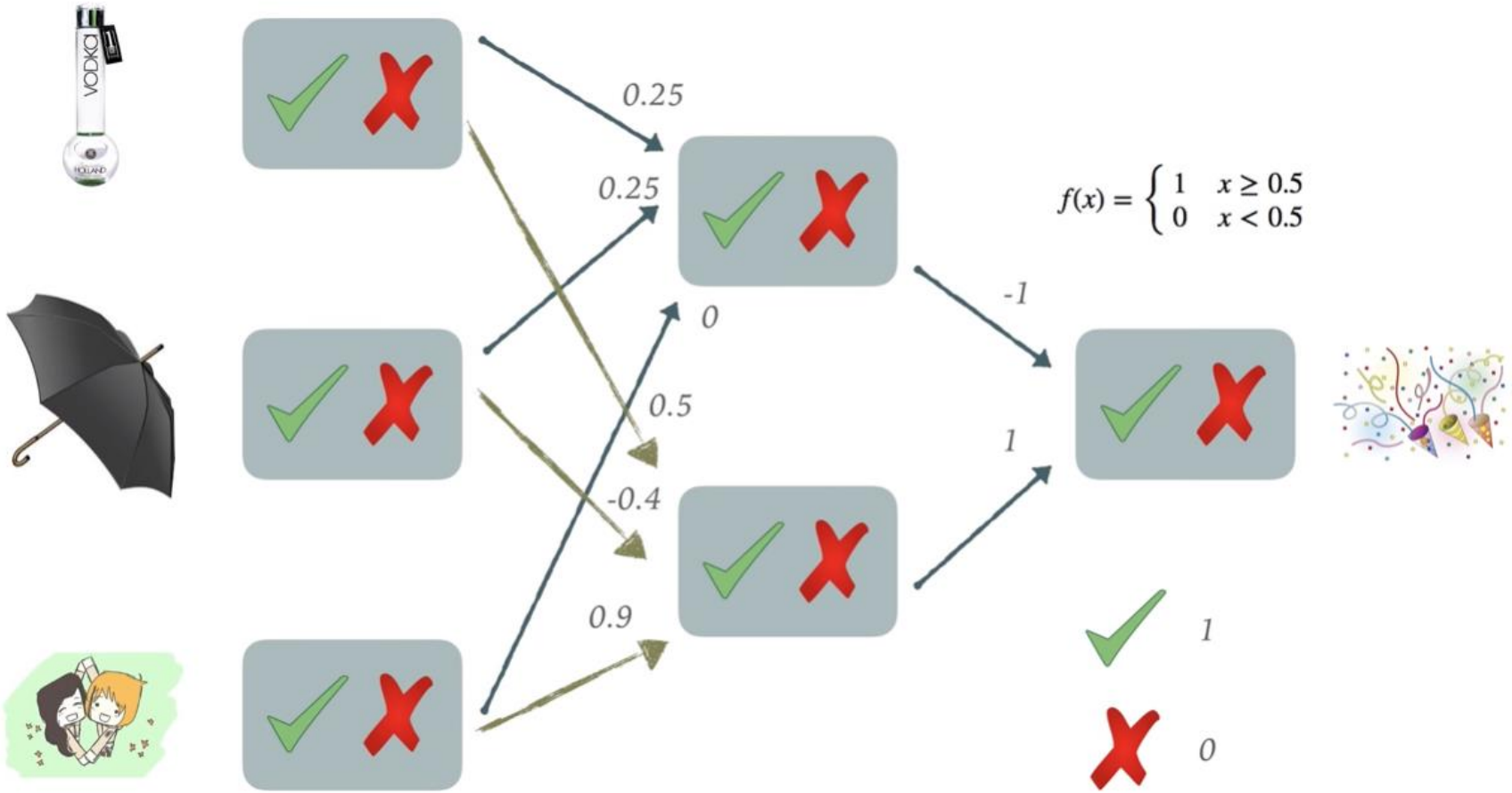
Принцип работы нейросети



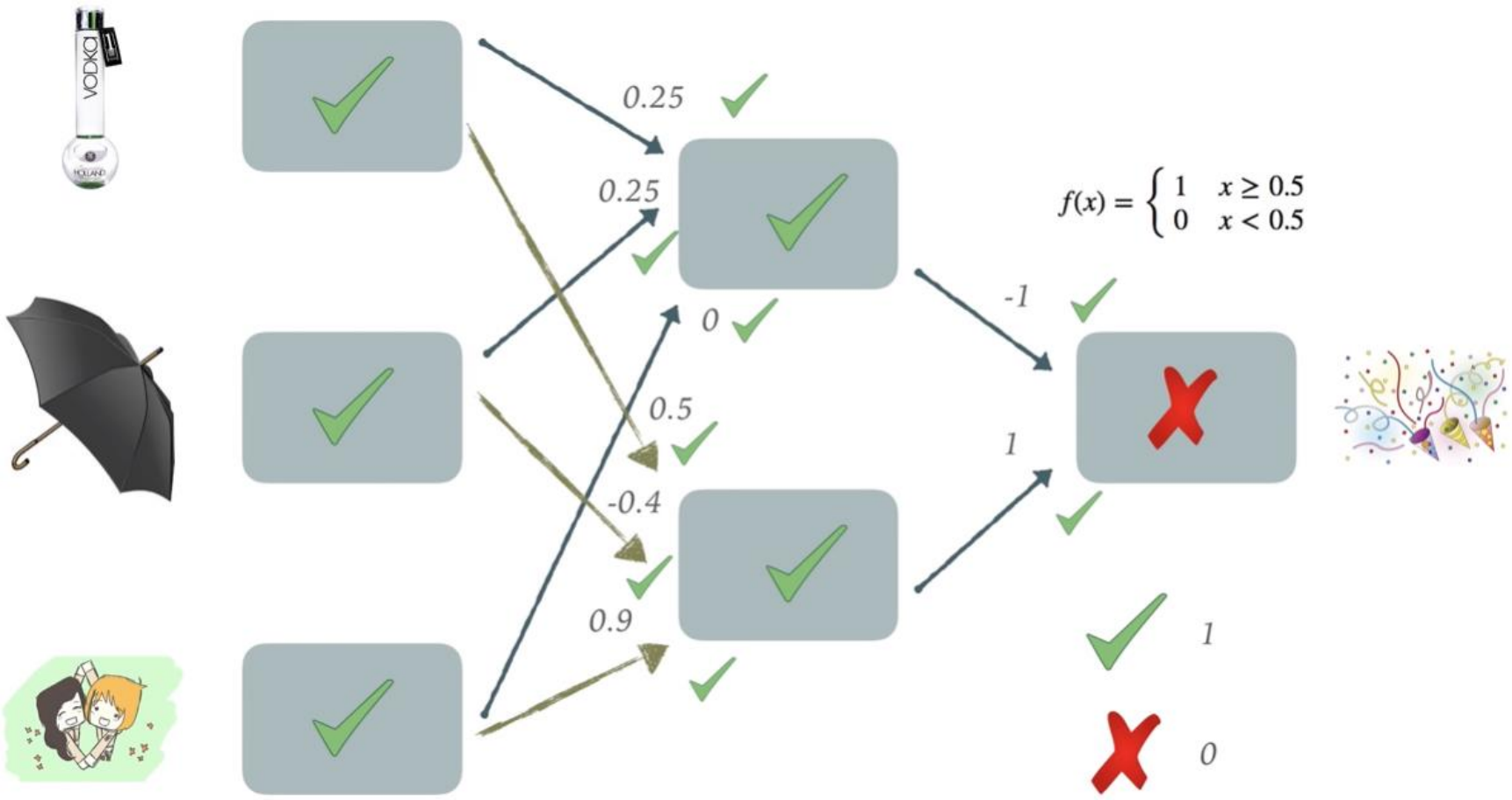
Принцип работы нейросети человека

- Пусть у индивида существует принцип – не идти на вечеринку, если будет дождь и напитки. Но он готов пойти при наличии друга и дождя.
- Сочетания пар приводят к новому показателю. Разные комбинации случаев вносят различный вес.
- Подобрать коэффициенты весов – сложнейшая задача. Процесс нахождения корректных весов – обучение сети.

Пример нейросети с внутренним слоем (обобщение задачи)



Пример нейросети с внутренним слоем (обобщение задачи)

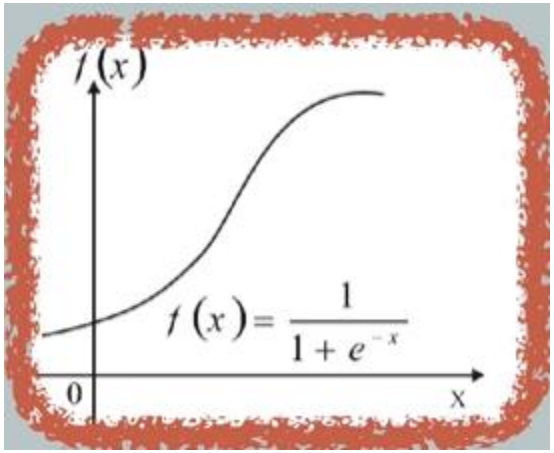


Реальная активационная функция

В сравнении с нейронами нашего мозга, функция активации решает, что должно быть запущено в следующий нейрон. Она принимает выходной сигнал из предыдущей ячейки и преобразует его в некоторую форму, которую можно использовать в качестве входных данных для следующей ячейки.

$$f(x) = \begin{cases} 1 & x \geq 0.5 \\ 0 & x < 0.5 \end{cases}$$

->



Плавный переход из состояния I в состояние j (гладкая, а значит дифференцируема)

Желательные особенности функции активации

- **Проблема исчезающего градиента**: нейронные сети обучаются с использованием процесса Градиентного спуска (Gradient Descent), который состоит из Обратного распространения ошибки (Backpropagation). Последний представляет собой цепное правило для получения изменения весов с целью уменьшения потерь после каждой эпохи.
- **Вычислительные затраты**: функции активации применяются после каждого уровня и должны рассчитываться миллионы раз в глубоких сетях. Следовательно, их вычисление должно быть недорогим в вычислительном отношении.
- **Дифференцируемость**: как уже упоминалось, нейронные сети обучаются с использованием процесса градиентного спуска, поэтому слои в модели должны быть дифференцируемыми. Это необходимое требование для того, чтобы функция работала как уровень функции активации.

Кодировка нейросети

In [3]:
```
import numpy as np

vodka = 0.0
rain = 1.0
friend = 0.0

def activation_function(x):
    if x >= 0.5:
        return 1
    else:
        return 0

def predict(vodka, rain, friend):
    inputs = np.array([vodka, rain, friend])
    weights_input_to_hiden_1 = [0.25, 0.25, 0]
    weights_input_to_hiden_2 = [0.5, -0.4, 0.9]
    weights_input_to_hiden = np.array([weights_input_to_hiden_1, weights_input_to_hiden_2])

    weights_hiden_to_output = np.array([-1,1])

    hiden_input = np.dot(weights_input_to_hiden, inputs)
    print("hiden_input: " + str(hiden_input))

    hiden_output = np.array([activation_function(x) for x in hiden_input])
    print("hiden_output: " + str(hiden_output))

    output = np.dot(weights_hiden_to_output, hiden_output)
    print("output: " + str(output))
    return activation_function(output)>=0.25

print("result:" + str(predict(vodka, rain, friend)))
```

```
hiden_input: [ 0.25 -0.4 ]
hiden_output: [0 0]
output: 0
result:False
```

Пример

https://colab.research.google.com/drive/1u4KxbBZ_IagV-znApkR1R4yOXKEd1hHb?usp=sharing

```
import numpy as np

vodka=1.0
rain=0.0
friend=1.0

def activation_function(x):
 if x>=0.5:
  return 1
 else:
  return 0

def predict(vodka,rain,friend):
 inputs=np.array([vodka,rain,friend])
 weights_input_to_hiden_1=[0.25,0.25,0.0]
 weights_input_to_hiden_2=[0.5,-0.4,0.9]
 weights_input_to_hiden=np.array([weights_input_to_hiden_1,weights_input_to_hiden_2])

 weights_hiden_to_output=np.array([-1,1])

 hiden_input=np.dot(weights_input_to_hiden,inputs)
 print("hiden_input:"+str(hiden_input))

 hiden_output=np.array([activation_function(x) for x in hiden_input])
 print("hiden_output:"+str(hiden_output))

 output=np.dot(weights_hiden_to_output,hiden_output)
 print("output:"+str(output))
 return activation_function(output)==1

print ("result:"+str(predict(vodka,rain,friend)))
```

Задание

- Реализовать моделирование нейронной сети, состоящей из 4 входных нейронов и 3 нейронов скрытого слоя.