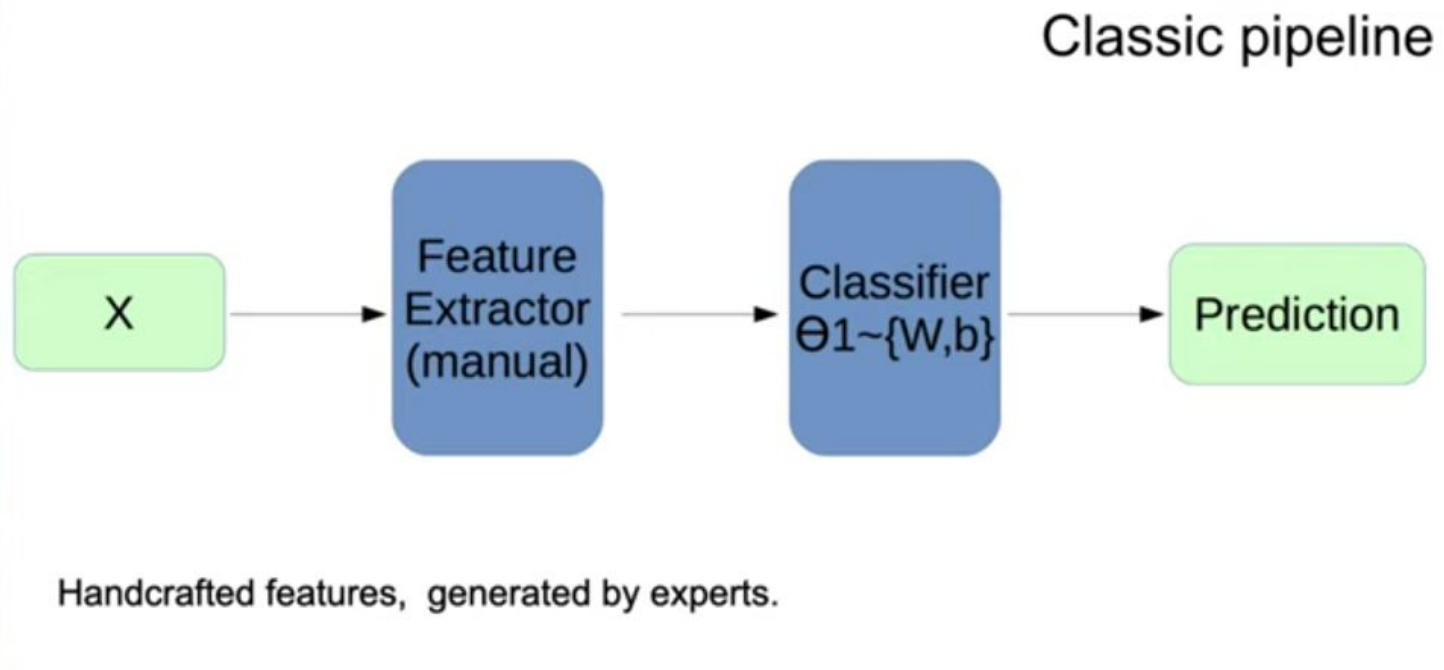
Нейросети, перцептрон

Что было до?

Что было до?


Classic pipeline
X
Feature Extractor (manual)
Classifier Θ1~{W,b}
Prediction
Handcrafted features, generated by experts.

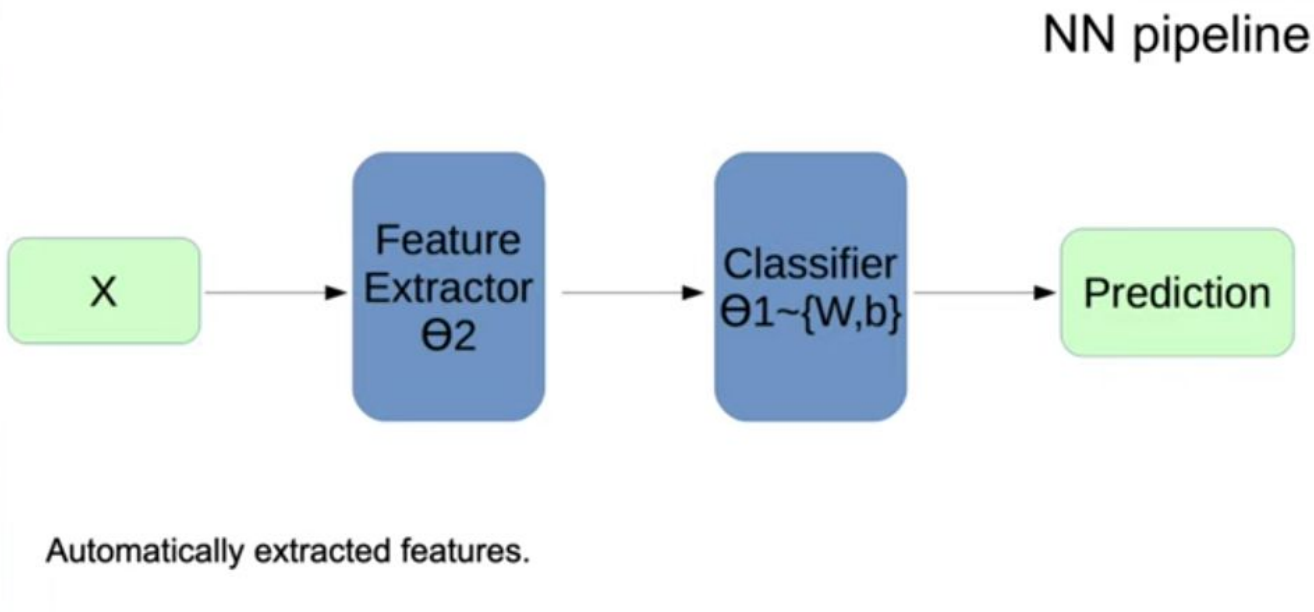
Чего мы хотим?

Чего мы хотим?

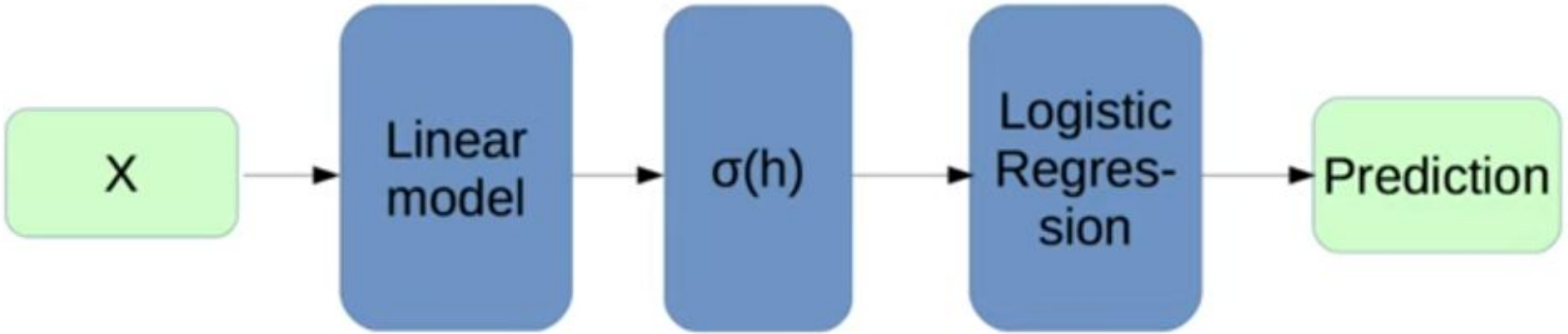
Чтобы “правильные” признаки можно было “выучить”

Чего мы хотим?

Чтобы “правильные” признаки можно было “выучить”

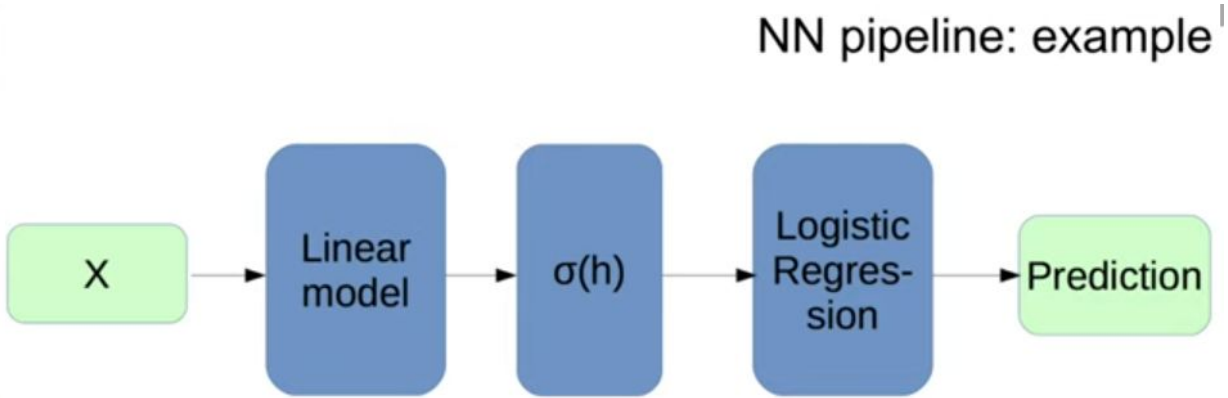


NN pipeline: example

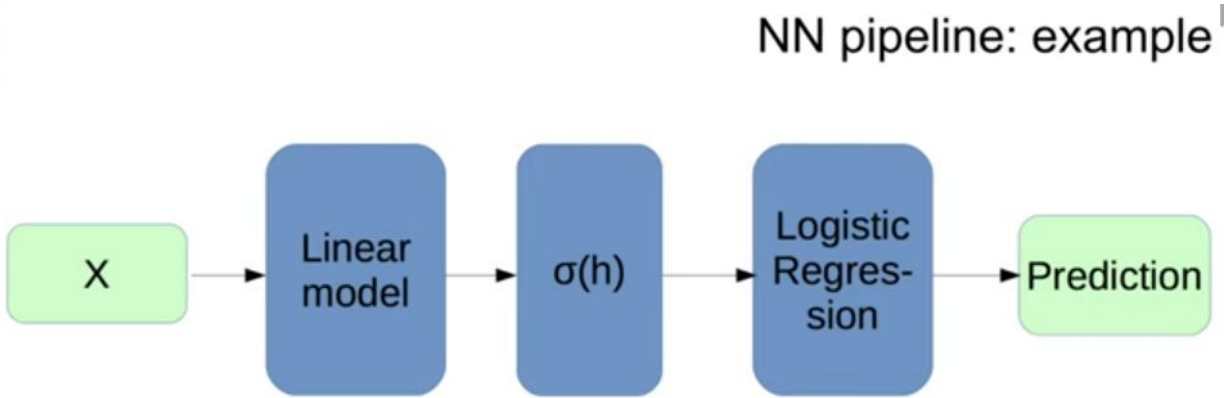


Простой пример

1. Линейная модель
2. Сигмоида
3. Логистическая регрессия



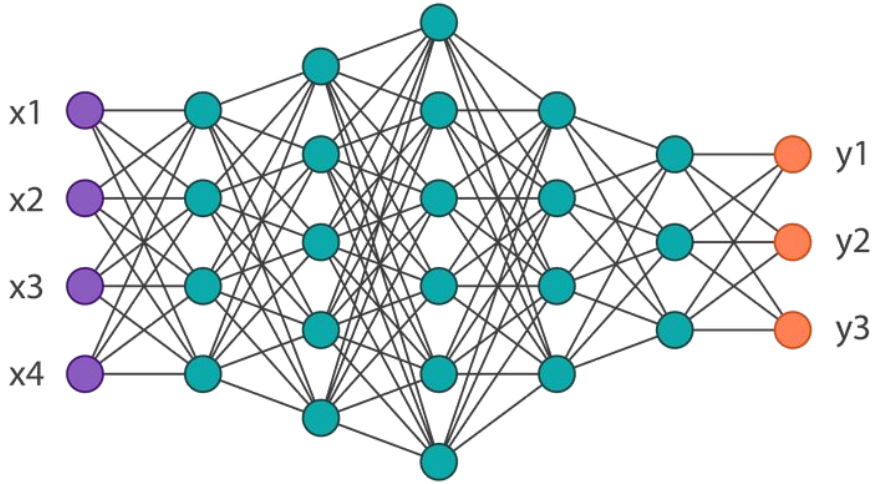
Функция активации (сигмоида в нашем случае) нужна, т.к. иначе можно было бы просто “схлопнуть” нейросеть в одну логистическую регрессию, т.к. несколько последовательно примененных линейных преобразований все еще являются линейным преобразованием, а вот сигмоида - нет.



Нейросеть - последовательность преобразований, преобразующих объект из исходного пр-ва в пр-во целевое

Нейросеть состоит из слоев, например - линейные слои (Wx+b), нелинейные (активация), входной, выходной

Функция активации - функция, применяемая к выходу слоя



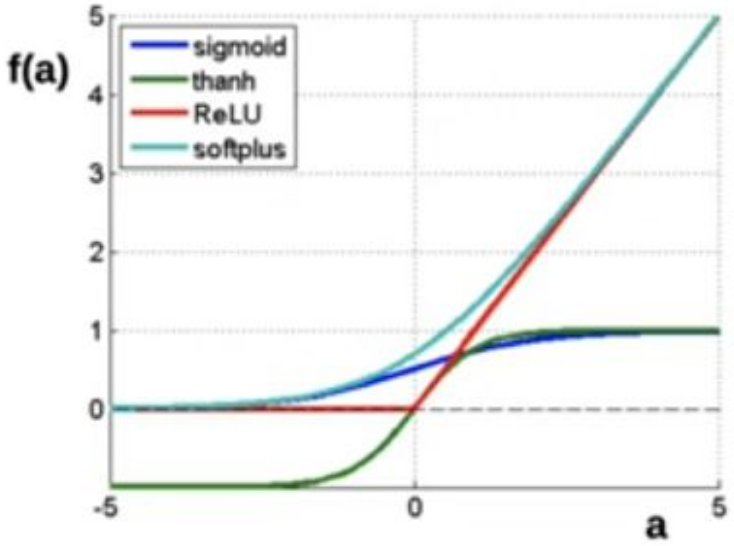
Activation functions: nonlinearities

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

$$f(a) = \tanh(a)$$

$$f(a) = \max(0, a)$$

$$f(a) = \log(1 + e^a)$$



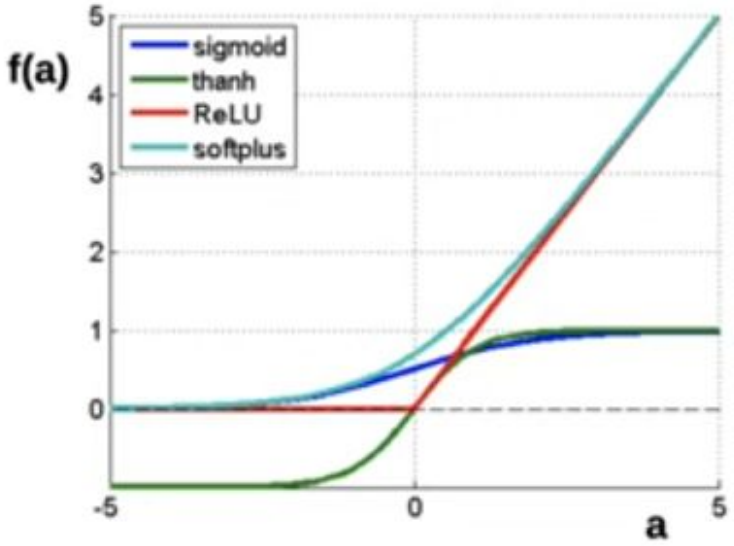
Activation functions: nonlinearities

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

$$f(a) = \tanh(a)$$

$$f(a) = \max(0, a)$$

$$f(a) = \log(1 + e^a)$$



Требование к функции активации - она должна быть дифференцируема

Обратное распространение ошибки

Обратное распространение ошибки

Прежде чем обсудить backpropagation, повторим изученное

Допустим, у нас есть какой-то набор чисел x0.

Мы применяем к нему какую-то функцию f0, которая переводит x0 в x1. При этом размерности x0 и x1 могут отличаться.

Потом мы применяем какую-то функцию f1, которая делает x1 -> x2, и так далее.

В конце концов мы получаем xN, к которому применяем линейную или логистическую регрессию (обозначим ее m) и получаем наш ответ - y

Допустим, у нас есть какой-то набор чисел x0.

Мы применяем к нему какую-то функцию f1, которая переводит x0 в x1. При этом размерности x0 и x1 могут отличаться.

Потом мы применяем какую-то функцию f2, которая делает x1 -> x2, и так далее.

В конце концов мы получаем xN, к которому применяем линейную или логистическую регрессию (обозначим ее m) и получаем наш ответ - y

Получившийся y равен:

$y = m(xN) = m(fN(xN-1)) = \ldots = m(fN(....f2(x1)...)) = m(fN(...f2(f1(x0))$

Допустим, у нас есть какой-то набор чисел x0.

Мы применяем к нему какую-то функцию f1, которая переводит x0 в x1. При этом размерности x0 и x1 могут отличаться.

Потом мы применяем какую-то функцию f2, которая делает x1 -> x2, и так далее.

В конце концов мы получаем xN, к которому применяем линейную или логистическую регрессию (обозначим ее m) и получаем наш ответ - y

Получившийся y равен:

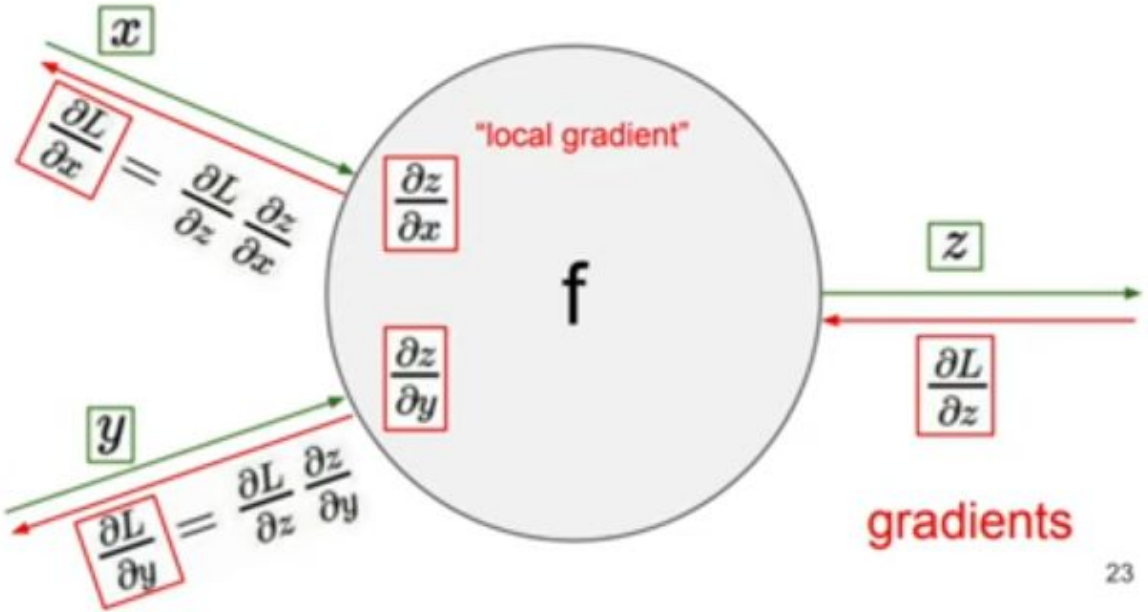
$y = m(xN) = m(fN(xN-1)) = \ldots = m(fN(....f2(x1)...)) = m(fN(...f2(f1(x0))$

Мы получили сложную функцию. И для ее дифференцируемости мы требуем дифференцируемость всех функций f_i

Backpropagation and chain rule

Chain rule is just simple math: $\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$

Backprop is just way to use it in NN training.

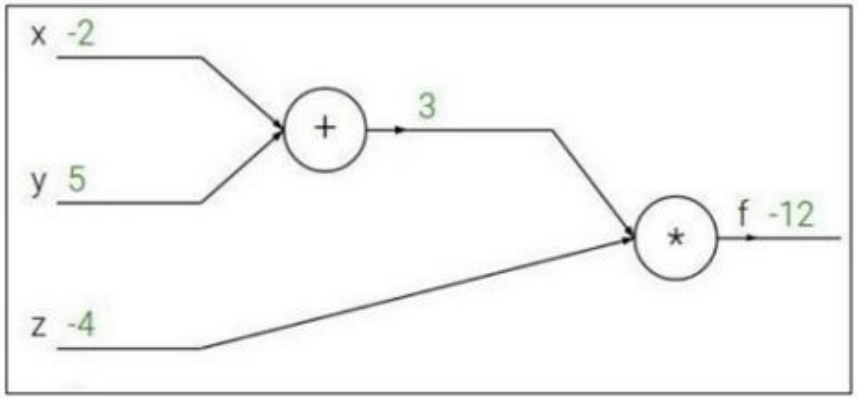


source: http://cs231n.github.io

Пример

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

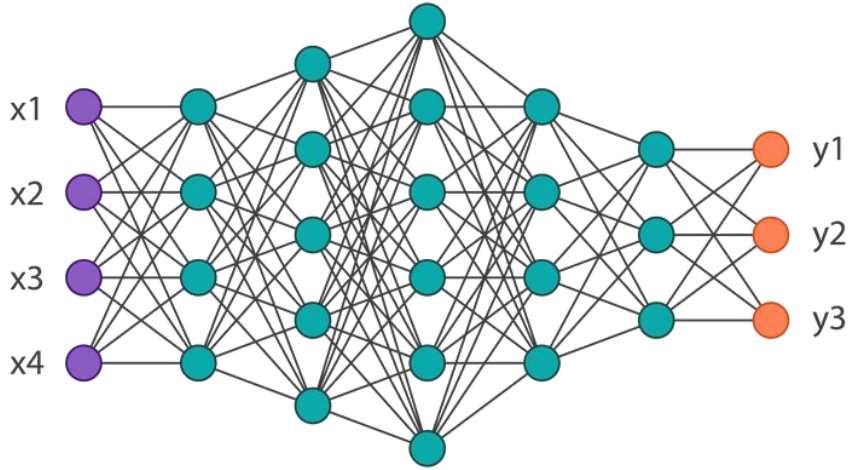


Обучаем нейросетьё

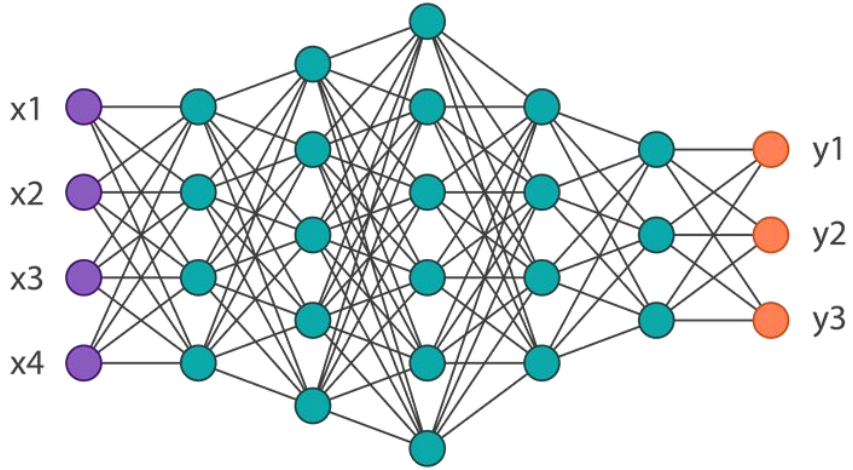
Обучаем нейросеть

Еще раз вспомним, как мы представляли нейросеть:

Обучаем нейросеть

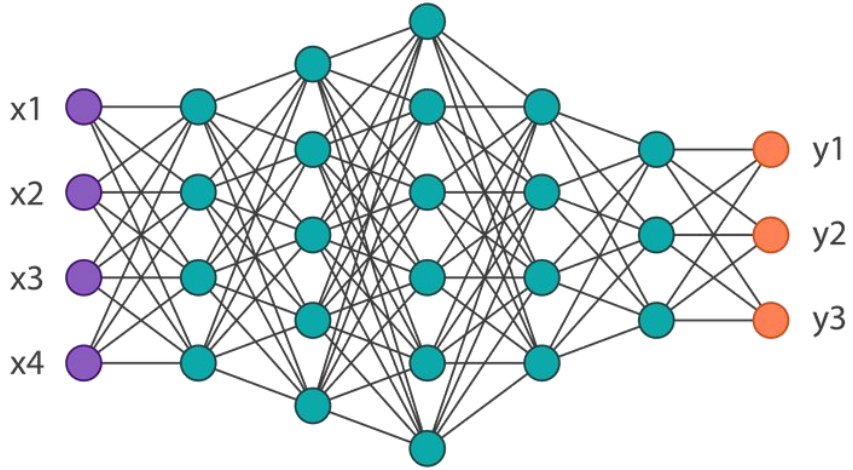


Обучаем нейросеть



$$g(x) := f^L(W^L f^{L-1}(W^{L-1} \cdots f^1(W^1 x) \cdots))$$

Обучаем нейросеть

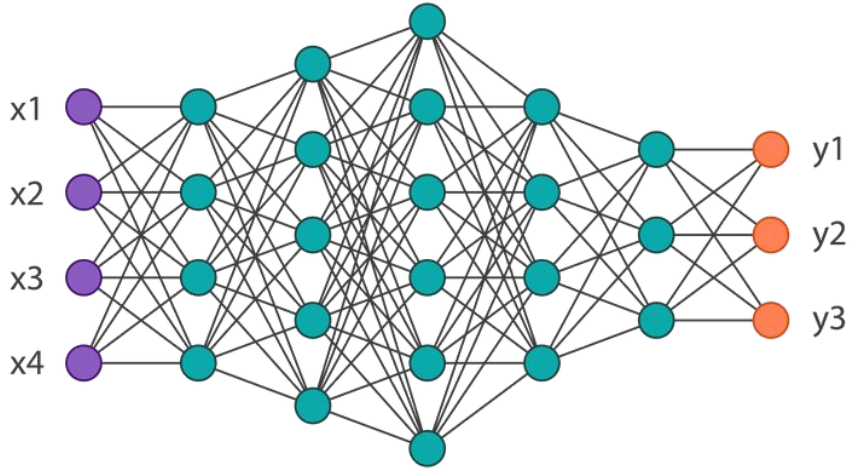


$$g(x) := f^L(W^L f^{L-1}(W^{L-1} \cdots f^1(W^1 x) \cdots))$$

W - веса слоя
f - функция активации

Очередной слой получает на свой вход выход предыдущего слоя, т.е.
f(W*x), где x - вход предыдущего слоя.

Обучаем нейросеть

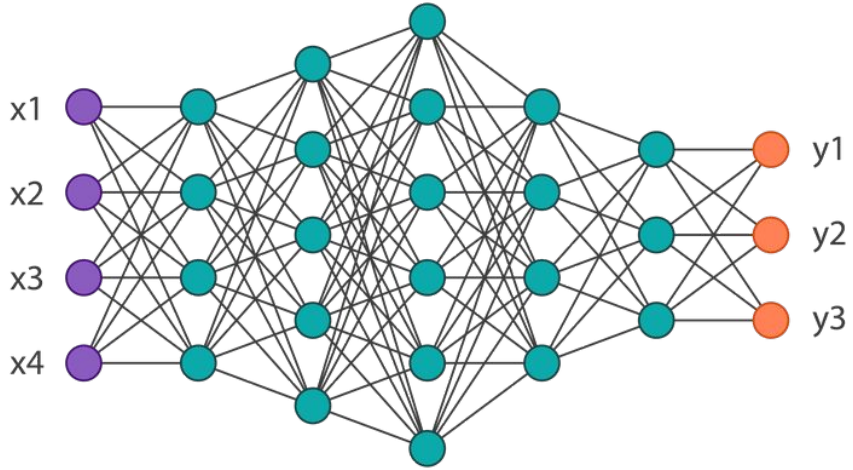


$$g(x) := f^L(W^L f^{L-1}(W^{L-1} \cdots f^1(W^1 x) \cdots))$$

Допустим, на какой-то итерации мы пропустили наши данные через нейросеть и получили ответ.
Посчитаем loss - функцию потерь от верного ответа и нашего предсказания, функцию, на которую будет оптимизироваться нейросеть.

$$C(y, f^L(W^L f^{L-1}(W^{L-1} \cdots f^2(W^2 f^1(W^1 x)) \cdots)))$$

Обучаем нейросеть



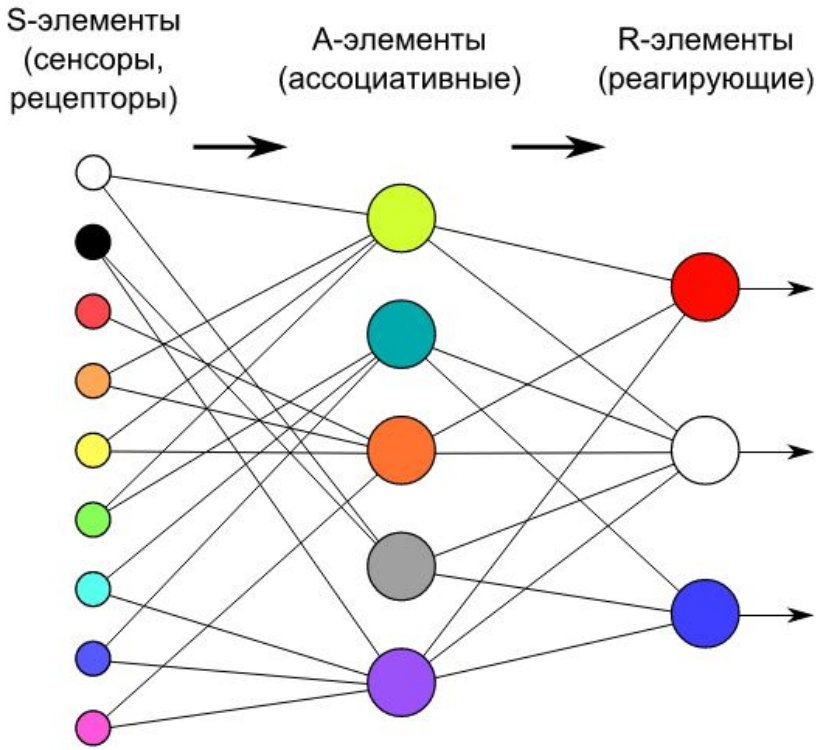
$$g(x) := f^L(W^L f^{L-1}(W^{L-1} \cdots f^1(W^1 x) \cdots))$$

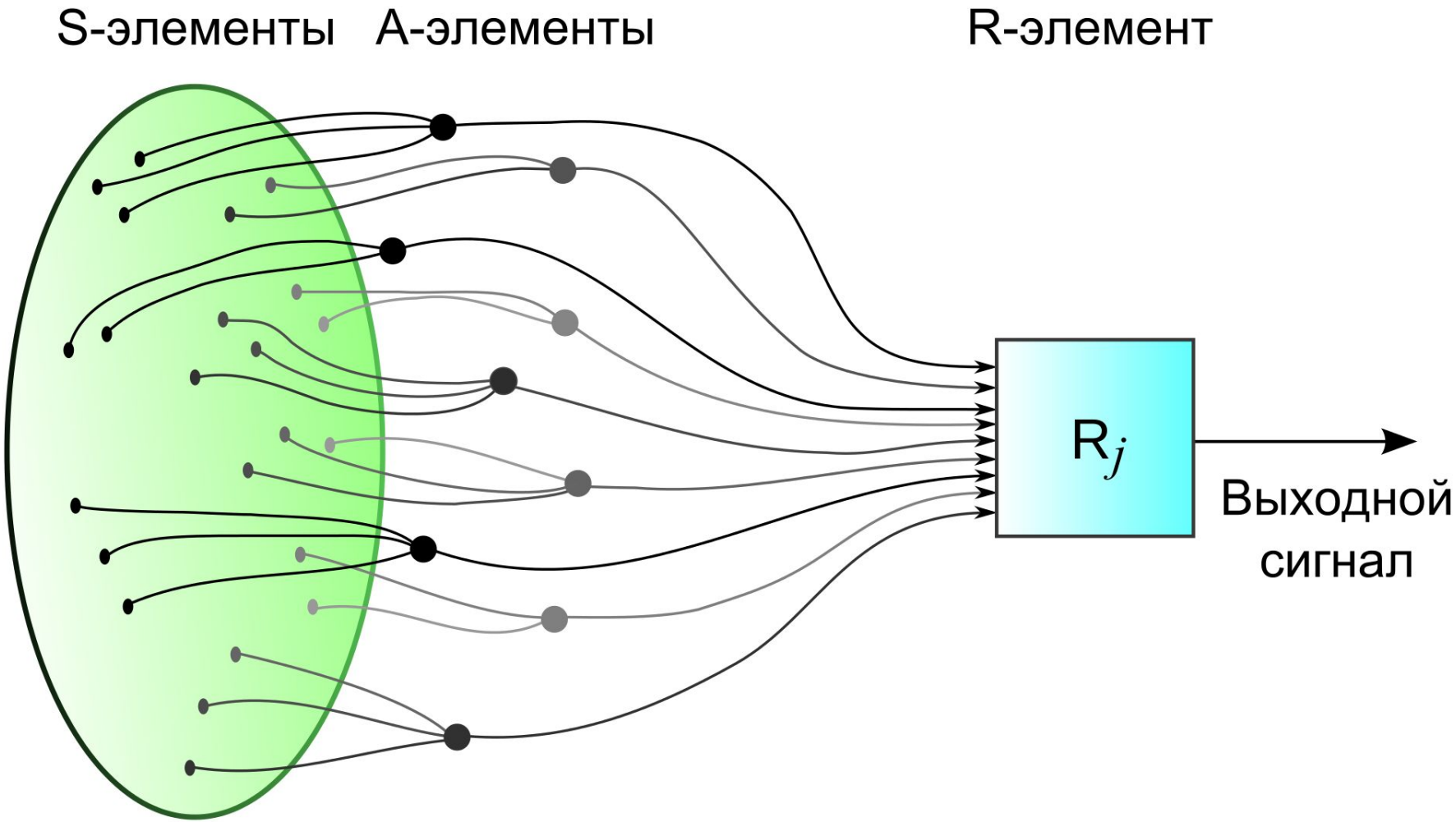
Допустим, на какой-то итерации мы пропустили наши данные через нейросеть и получили ответ.
Посчитаем loss - функцию потерь от верного ответа и нашего предсказания, функцию, на которую будет оптимизироваться нейросеть.

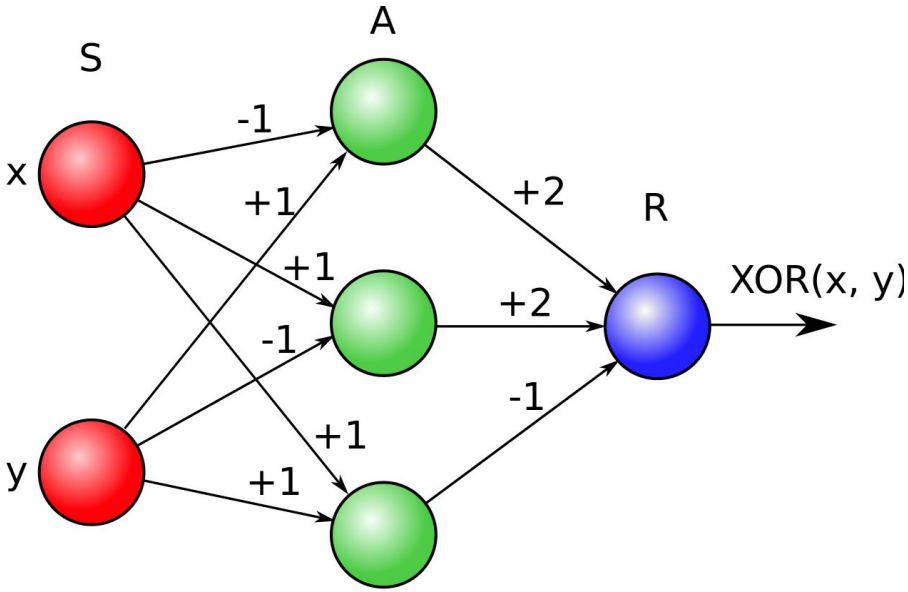
$$C(y, f^L(W^L f^{L-1}(W^{L-1} \cdots f^2(W^2 f^1(W^1 x)) \cdots)))$$

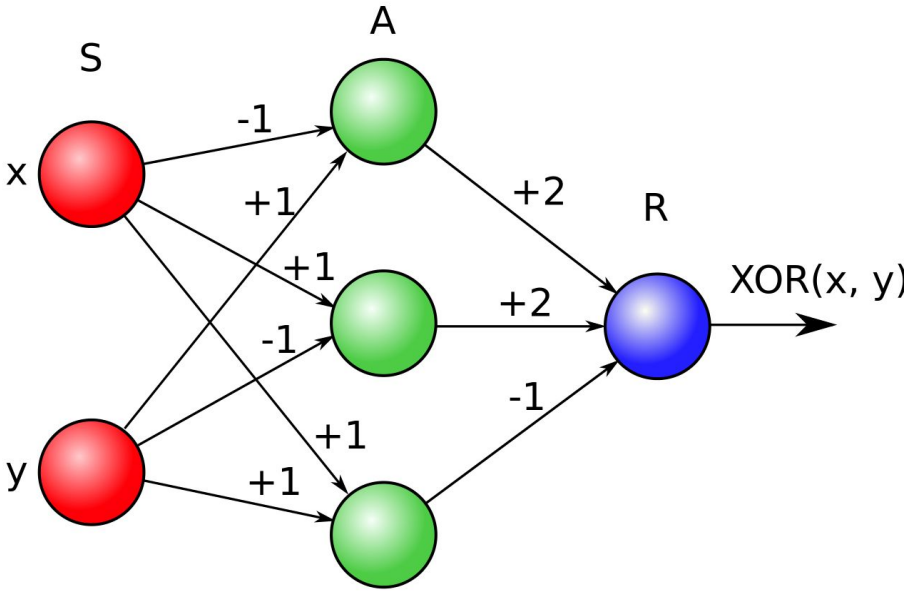
По методу обратного распространения ошибки считаем производную для каждого веса и обновляем его

Перцептрон (англ. *Perceptron*) — простейший вид нейронных сетей. В основе лежит математическая модель восприятия информации мозгом, состоящая из сенсоров, ассоциативных и реагирующих элементов.




S-элементы
A-элементы
R-элемент
R_j
Выходной
сигнал


S
A
x
y
-1
+1
+1
-1
+1
+1
+2
+2
-1
R
XOR(x, y)



x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0