

Разбор летучки

Лекция 8

Нейронные сети

Екатерина Тузова

Модель McCulloch-Pitts

$$X = \mathbb{R}^n, Y = \{-1, +1\}$$

Модель McCulloch-Pitts

$$X = \mathbb{R}^n, Y = \{-1, +1\}$$

$$a(x, w) = \sigma\left(\sum_{j=1}^n w_j x^j - w_0\right) = \sigma(\langle \mathbf{w}, \mathbf{x} \rangle)$$

x^j — признаки объекта, $j = 1, \dots, n$

$w_j \in R$ — веса признаков

$\sigma(s)$ — функция активации (например, sign)

Модель McCulloch-Pitts

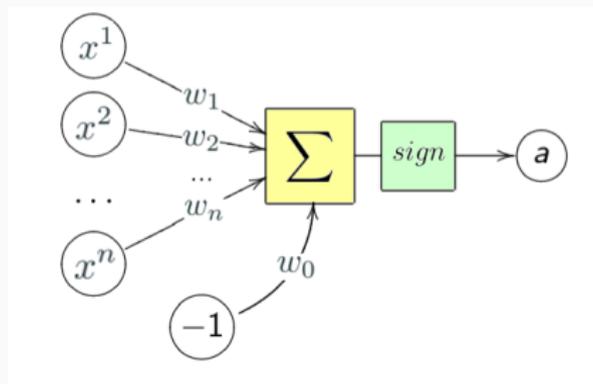
$$X = \mathbb{R}^n, Y = \{-1, +1\}$$

$$a(x, w) = \sigma\left(\sum_{j=1}^n w_j x^j - w_0\right) = \sigma(\langle \mathbf{w}, \mathbf{x} \rangle)$$

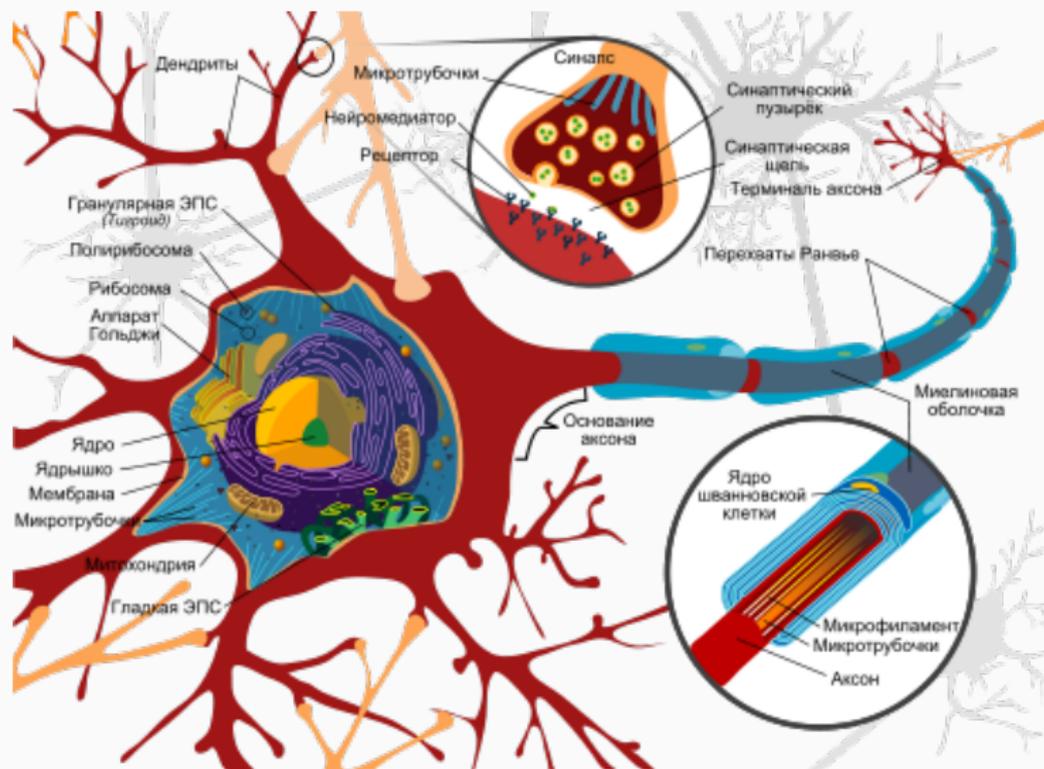
x^j — признаки объекта, $j = 1, \dots, n$

$w_j \in R$ — веса признаков

$\sigma(s)$ — функция активации (например, sign)



Нейрон



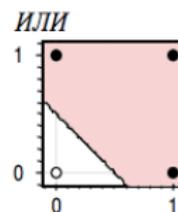
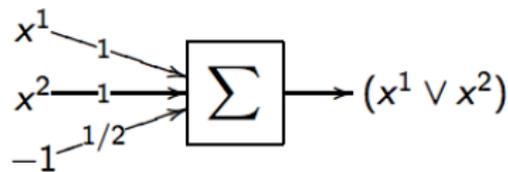
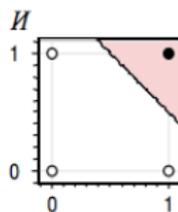
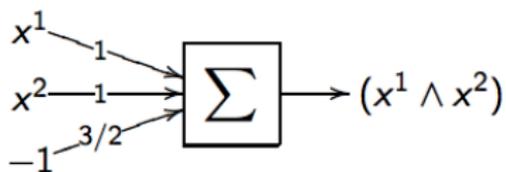
Задача классификации:

$$a(\mathbf{x}, \mathbf{w}) = \text{sign}\langle \mathbf{w}, \mathbf{x} \rangle$$

$$Q(w; X^l) = \sum_{i=1}^l \mathcal{L}(\langle \mathbf{w}, \mathbf{x}_i \rangle y_i) \rightarrow \min_w$$

Какой класс функций можно реализовать нейроном?

Нейронная реализация логических функций



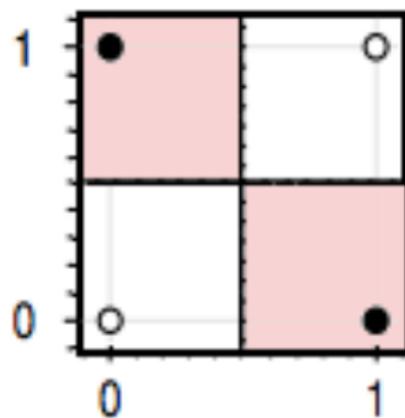
Функции И, ИЛИ от бинарных переменных x_1 и x_2 :

$$x_1 \wedge x_2 = x_1 + x_2 - \frac{3}{2} > 0$$

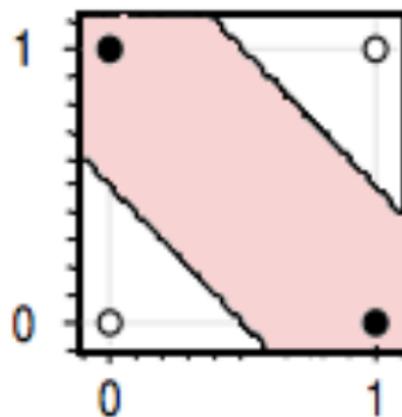
$$x_1 \vee x_2 = x_1 + x_2 - \frac{1}{2} > 0$$

Логическая функция XOR

1-й способ



2-й способ



Логическая функция XOR

Функция $x_1 \oplus x_2 = [x_1 \neq x_2]$ не реализуема одним нейроном.

Логическая функция XOR

Функция $x_1 \oplus x_2 = [x_1 \neq x_2]$ не реализуема одним нейроном.

Два способа реализации:

1. Добавлением нелинейного признака:

$$x_1 \oplus x_2 = [x_1 + x_2 + 2x_1x_2 - \frac{1}{2} > 0]$$

Логическая функция XOR

Функция $x_1 \oplus x_2 = [x_1 \neq x_2]$ не реализуема одним нейроном.

Два способа реализации:

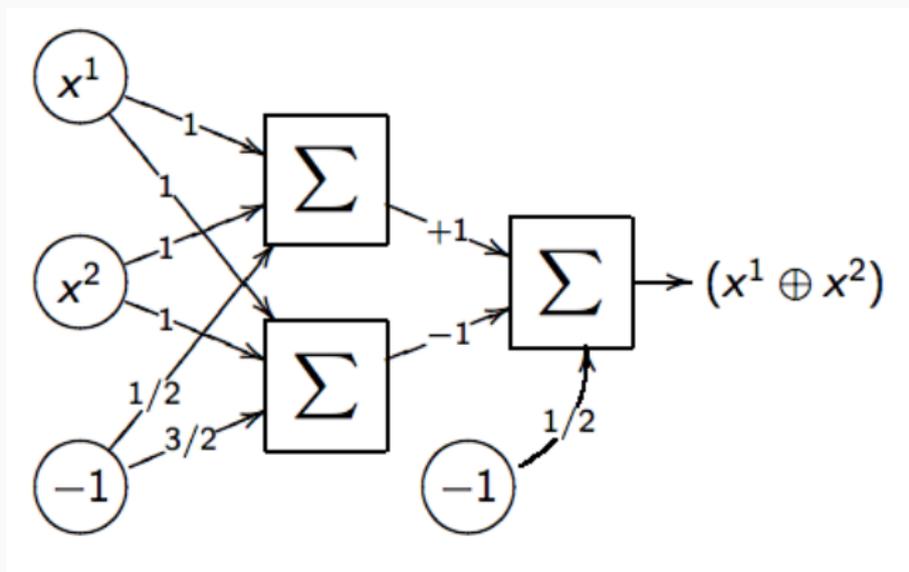
1. Добавлением нелинейного признака:

$$x_1 \oplus x_2 = [x_1 + x_2 + 2x_1x_2 - \frac{1}{2} > 0]$$

2. Сетью (двухслойной суперпозицией) функций И, ИЛИ:

$$x_1 \oplus x_2 = [(x_1 \vee x_2) - (x_1 \wedge x_2) - \frac{1}{2} > 0]$$

Логическая функция XOR



Любую ли функцию можно представить нейросетью?

- Двухслойная сеть в $\{0, 1\}^n$ позволяет реализовать произвольную булеву функцию.

Любую ли функцию можно представить нейросетью?

- Двухслойная сеть в $\{0, 1\}^n$ позволяет реализовать произвольную булеву функцию.
- Двухслойная сеть в \mathbb{R}^n позволяет отделить произвольный выпуклый многогранник.

Любую ли функцию можно представить нейросетью?

- Двухслойная сеть в $\{0, 1\}^n$ позволяет реализовать произвольную булеву функцию.
- Двухслойная сеть в \mathbb{R}^n позволяет отделить произвольный выпуклый многогранник.
- Трёхслойная сеть \mathbb{R}^n позволяет отделить произвольную многогранную область, не обязательно выпуклую.

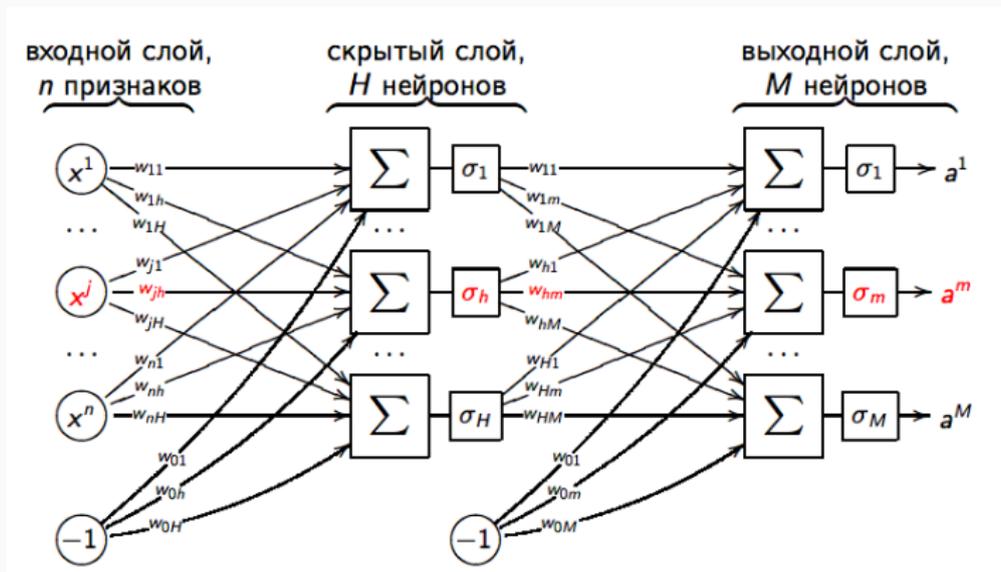
Любую ли функцию можно представить нейросетью?

- Двухслойная сеть в $\{0, 1\}^n$ позволяет реализовать произвольную булеву функцию.
- Двухслойная сеть в \mathbb{R}^n позволяет отделить произвольный выпуклый многогранник.
- Трёхслойная сеть \mathbb{R}^n позволяет отделить произвольную многогранную область, не обязательно выпуклую.
- С помощью линейных операций и одной нелинейной функции активации σ можно приблизить любую непрерывную функцию с любой желаемой точностью

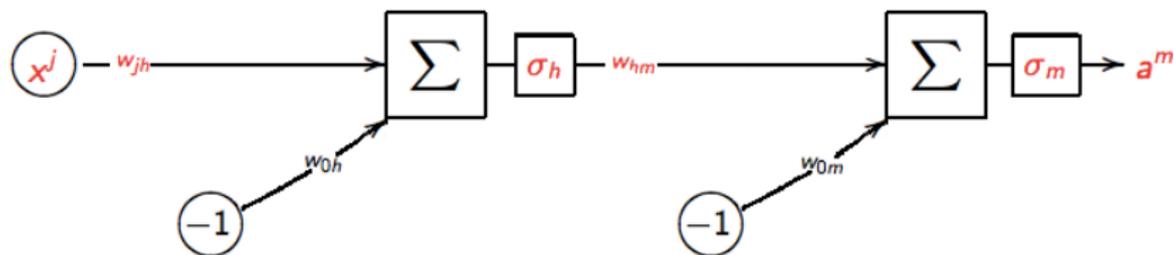
- Двух-трёх слоёв обычно достаточно.
- Можно достраивать нейроны в произвольных местах сети по необходимости.

Многослойная нейронная сеть

Пусть $Y = \mathbb{R}^M$, два слоя в сети.



Многослойная нейронная сеть



Стохастический градиент

$$Q(\mathbf{w}) = \sum_{i=1}^l \mathcal{L}(w, x_i, y_i) \rightarrow \min_w$$

- 1 **function** STOCHASTIC_GRADIENT(X^l, α, η)
- 2 Перемешать данные в X^l
- 3 Инициализировать \mathbf{w}
- 4 $Q(\mathbf{w}) = \sum_{i=1}^l \mathcal{L}(\langle \mathbf{w}, \mathbf{x}_i \rangle y_i)$
- 5 **repeat**[пока Q и/или w не стабилизируются]
- 6 Взять x_i из X^l
- 7 Потеря: $\varepsilon_i = \mathcal{L}(w, x_i, y_i)$
- 8 Градиентный шаг: $w = w - \alpha \nabla \mathcal{L}(w, x_i, y_i)$
- 9 Оценить $Q = (1 - \eta)Q + \eta \varepsilon_i$

Сколько операций
потребуется для вычисления
градиента в точке?

Идея: Сохранять некоторые промежуточные результаты в узлах сети.

Задача дифференцирования суперпозиции функций

Выходные значения $a^m(x_i)$, $m = 1, \dots, M$ на объекте x_i :

$$a^m(x_i) = \sigma_m\left(\sum_{h=0}^H w_{hm} u^h(x_i)\right), \quad u^h(x_i) = \sigma_h\left(\sum_{j=0}^n w_{jh} x_i^j\right)$$

Задача дифференцирования суперпозиции функций

Выходные значения $a^m(x_i)$, $m = 1, \dots, M$ на объекте x_i :

$$a^m(x_i) = \sigma_m\left(\sum_{h=0}^H w_{hm} u^h(x_i)\right), \quad u^h(x_i) = \sigma_h\left(\sum_{j=0}^n w_{jh} x_i^j\right)$$

Возьмем $\mathcal{L}_i(w)$ – средний квадрат ошибки:

$$\mathcal{L}_i(w) = \frac{1}{2} \sum_{m=1}^M (a^m(x_i) - y_i^m)^2$$

Задача дифференцирования суперпозиции функций

Выходные значения $a^m(x_i)$, $m = 1, \dots, M$ на объекте x_i :

$$a^m(x_i) = \sigma_m \left(\sum_{h=0}^H w_{hm} u^h(x_i) \right), \quad u^h(x_i) = \sigma_h \left(\sum_{j=0}^n w_{jh} x_i^j \right)$$

Возьмем $\mathcal{L}_i(w)$ – средний квадрат ошибки:

$$\mathcal{L}_i(w) = \frac{1}{2} \sum_{m=1}^M (a^m(x_i) - y_i^m)^2$$

Промежуточная задача: найти частные производные

$$\frac{\partial \mathcal{L}_i(w)}{\partial a^m} \quad \frac{\partial \mathcal{L}_i(w)}{\partial u^h}$$

Быстрое вычисление градиента

Ошибка на выходном слое:

$$\frac{\partial \mathcal{L}_i(w)}{\partial a^m} = a^m(x_i) - y_i^m = \varepsilon_i^m$$

Быстрое вычисление градиента

Ошибка на выходном слое:

$$\frac{\partial \mathcal{L}_i(w)}{\partial a^m} = a^m(x_i) - y_i^m = \varepsilon_i^m$$

Ошибка на скрытом слое:

$$\frac{\partial \mathcal{L}_i(w)}{\partial u^h} = \sum_{m=1}^M (a^m(x_i) - y_i^m) \sigma'_m w_{hm} = \sum_{m=1}^M \varepsilon_i^m \sigma'_m w_{hm} = \varepsilon_i^h$$

Быстрое вычисление градиента

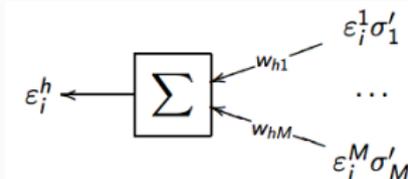
Ошибка на выходном слое:

$$\frac{\partial \mathcal{L}_i(w)}{\partial a^m} = a^m(x_i) - y_i^m = \varepsilon_i^m$$

Ошибка на скрытом слое:

$$\frac{\partial \mathcal{L}_i(w)}{\partial u^h} = \sum_{m=1}^M (a^m(x_i) - y_i^m) \sigma'_m w_{hm} = \sum_{m=1}^M \varepsilon_i^m \sigma'_m w_{hm} = \varepsilon_i^h$$

ε_i^h вычисляется по ε_i^m , если запустить сеть «задом наперёд»:



Теперь, имея частные производные $\mathcal{L}_i(w)$ по a^m и u^h , легко выписать градиент $\mathcal{L}_i(w)$ по весам w .

Быстрое вычисление градиента

Теперь, имея частные производные $\mathcal{L}_i(w)$ по a^m и u^h , легко выписать градиент $\mathcal{L}_i(w)$ по весам w .

Вопрос: как?

$$\frac{\partial \mathcal{L}_i(w)}{\partial w_{hm}} = \frac{\partial \mathcal{L}_i(w)}{\partial a^m} \frac{\partial a^m}{\partial w_{hm}} = \varepsilon_i^m \sigma'_m u^h(x_i),$$

$$m = 1, \dots, M, \quad h = 0, \dots, H$$

$$\frac{\partial \mathcal{L}_i(w)}{\partial w_{hm}} = \frac{\partial \mathcal{L}_i(w)}{\partial a^m} \frac{\partial a^m}{\partial w_{hm}} = \varepsilon_i^m \sigma'_m u^h(x_i),$$

$$m = 1, \dots, M, \quad h = 0, \dots, H$$

$$\frac{\partial \mathcal{L}_i(w)}{\partial w_{jh}} = \frac{\partial \mathcal{L}_i(w)}{\partial u^h} \frac{\partial u^h}{\partial w_{jh}} = \varepsilon_i^h \sigma'_h x_i^j,$$

$$h = 1, \dots, H, \quad j = 0, \dots, n$$

Алгоритм обратного распространения ошибки

1 **function** BACKPROPAGATION(X^l, H, α, η)

2 **repeat**[пока Q не стабилизируются]

3 Взять x_i из X^l

4

$$\left\{ \begin{array}{l} u_i^h = \sigma_h\left(\sum_{j=0}^J w_{jh}x_i^j\right), \quad h = 1, \dots, H \\ a_i^m = \sigma_m\left(\sum_{h=0}^H w_{hm}u_i^h\right), \quad \varepsilon_i^m = a_i^m - y_i^m, \quad m = 1, \dots, M \\ \mathcal{L}_i = \sum_{m=1}^M (\varepsilon_i^m)^2 \end{array} \right.$$

5

$$\left\{ \varepsilon_i^h = \sum_{m=1}^M \varepsilon_i^m \sigma'_m w_{hm}, \quad h = 1, \dots, H \right.$$

6

$$\left\{ \begin{array}{l} w_{hm} = w_{hm} - \alpha \varepsilon_i^m \sigma'_m u_i^h, \quad h = 0, \dots, H, \quad m = 1, \dots, M \\ w_{jh} = w_{jh} - \alpha \varepsilon_i^h \sigma'_h x_i^j, \quad j = 0, \dots, n, \quad h = 1, \dots, H \\ Q = (1 - \eta)Q + \eta \mathcal{L}_i \end{array} \right.$$

- + Эффективность: градиент вычисляется за время, сравнимое со временем вычисления самой сети

- + Эффективность: градиент вычисляется за время, сравнимое со временем вычисления самой сети
- + Легко обобщается на любые σ , \mathcal{L}

- + Эффективность: градиент вычисляется за время, сравнимое со временем вычисления самой сети
- + Легко обобщается на любые σ , \mathcal{L}
- + Возможно динамическое (потокковое) обучение

- + Эффективность: градиент вычисляется за время, сравнимое со временем вычисления самой сети
- + Легко обобщается на любые σ , \mathcal{L}
- + Возможно динамическое (потокковое) обучение
- + На сверхбольших выборках не обязательно брать все x_i

- + Эффективность: градиент вычисляется за время, сравнимое со временем вычисления самой сети
- + Легко обобщается на любые σ , \mathcal{L}
- + Возможно динамическое (потокковое) обучение
- + На сверхбольших выборках не обязательно брать все x_i
- + Возможность распараллеливания

- + Эффективность: градиент вычисляется за время, сравнимое со временем вычисления самой сети
- + Легко обобщается на любые σ , \mathcal{L}
- + Возможно динамическое (потокковое) обучение
- + На сверхбольших выборках не обязательно брать все x_i
- + Возможность распараллеливания
- Возможна медленная сходимость

- + Эффективность: градиент вычисляется за время, сравнимое со временем вычисления самой сети
- + Легко обобщается на любые σ , \mathcal{L}
- + Возможно динамическое (потокковое) обучение
- + На сверхбольших выборках не обязательно брать все x_i
- + Возможность распараллеливания

- Возможна медленная сходимость
- Застревание в локальных минимумах

- + Эффективность: градиент вычисляется за время, сравнимое со временем вычисления самой сети
- + Легко обобщается на любые σ , \mathcal{L}
- + Возможно динамическое (потокковое) обучение
- + На сверхбольших выборках не обязательно брать все x_i
- + Возможность распараллеливания

- Возможна медленная сходимость
- Застревание в локальных минимумах
- Проблема переобучения

- + Эффективность: градиент вычисляется за время, сравнимое со временем вычисления самой сети
- + Легко обобщается на любые σ , \mathcal{L}
- + Возможно динамическое (потокковое) обучение
- + На сверхбольших выборках не обязательно брать все x_i
- + Возможность распараллеливания

- Возможна медленная сходимость
- Застревание в локальных минимумах
- Проблема переобучения
- Подбор комплекса эвристик

Применимы все те же эвристики, что и в обычном SG.

Стандартные эвристики для метода SG

Применимы все те же эвристики, что и в обычном SG.

Напомните.

- Инициализация весов

Стандартные эвристики для метода SG

- Инициализация весов
- Порядок предъявления объектов

Стандартные эвристики для метода SG

- Инициализация весов
- Порядок предъявления объектов
- Оптимизация величины градиентного шага

Стандартные эвристики для метода SG

- Инициализация весов
- Порядок предъявления объектов
- Оптимизация величины градиентного шага
- Регуляризация (сокращение весов)

- Выбор функций активации в каждом нейроне

- Выбор функций активации в каждом нейроне
- Выбор числа слоёв и числа нейронов

- Сигмоида

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

Примеры функций активации

- Сигмоида

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

- Гиперболический тангенс

$$\tanh(x) = 2\sigma(2x) - 1$$

- Сигмоида

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

- Гиперболический тангенс

$$\tanh(x) = 2\sigma(2x) - 1$$

- Rectified Linear Unit

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

Практическое применение

Большинство используемых на сегодняшний день методов придуманы в 80-х.

Почему их снова стали использовать?

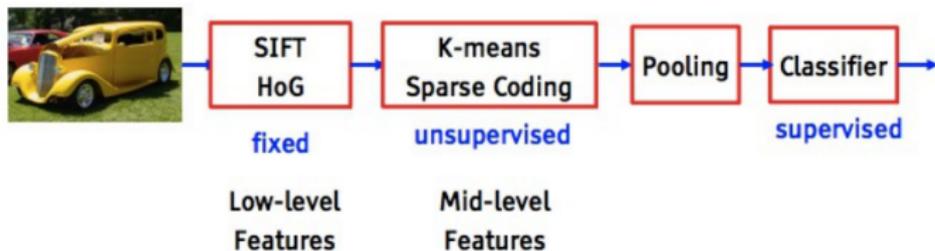
Большинство используемых на сегодняшний день методов придуманы в 80-х.

Почему их снова стали использовать?

- Большие вычислительные возможности (в том числе распределенные вычисления)
- Большие объемы данных
- Новые методы предобучения, новые архитектуры

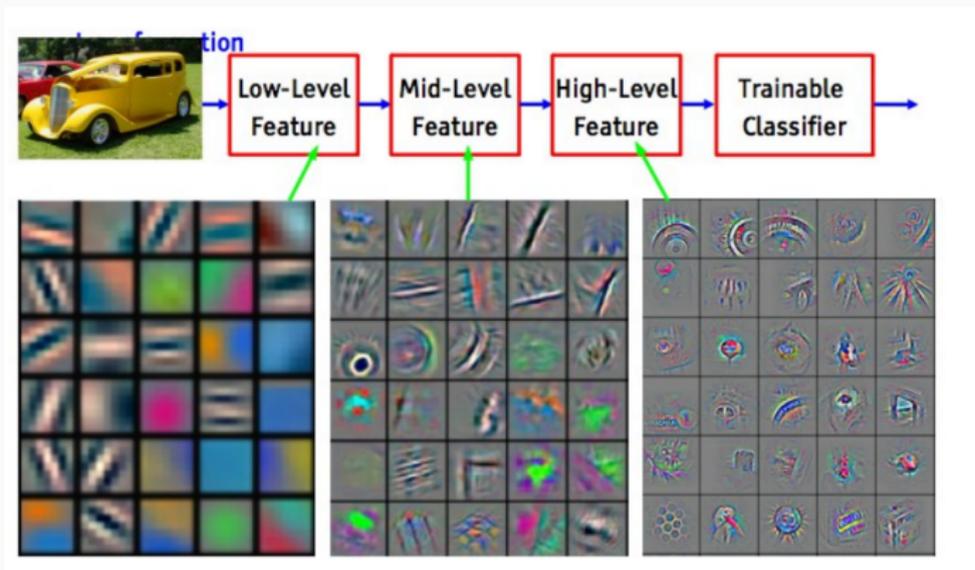
Как было раньше?

Ученые придумывали признаки. На них обучали классификатор.



Deep learning

Извлечение признаков во время обучения.



Архитектура сети выбирается таким образом, чтобы заложить в нее априорные знания из предметной области:

Архитектура сети выбирается таким образом, чтобы заложить в нее априорные знания из предметной области:

1. Пиксель изображения сильнее связан с соседними пикселями (локальная корреляция)

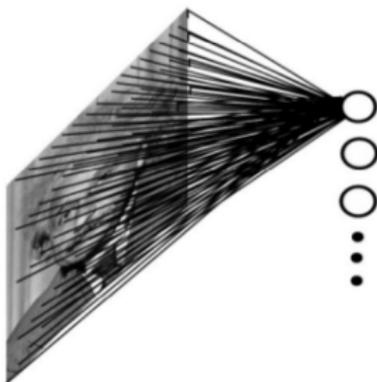
Архитектура сети выбирается таким образом, чтобы заложить в нее априорные знания из предметной области:

1. Пиксель изображения сильнее связан с соседними пикселями (локальная корреляция)
2. Объект может встретиться в любой части изображения (инвариантность к перемещению)

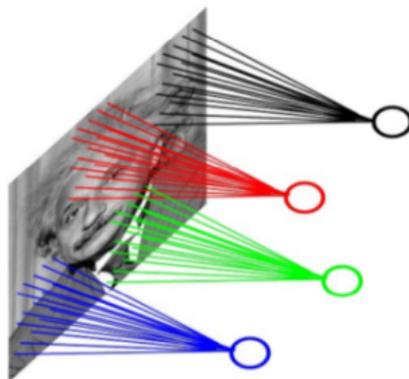
Локально-связанный слой

Закладываем в архитектуру сети априорное знание о том, что соседние пиксели изображения сильнее связаны между собой.

Полносвязный слой

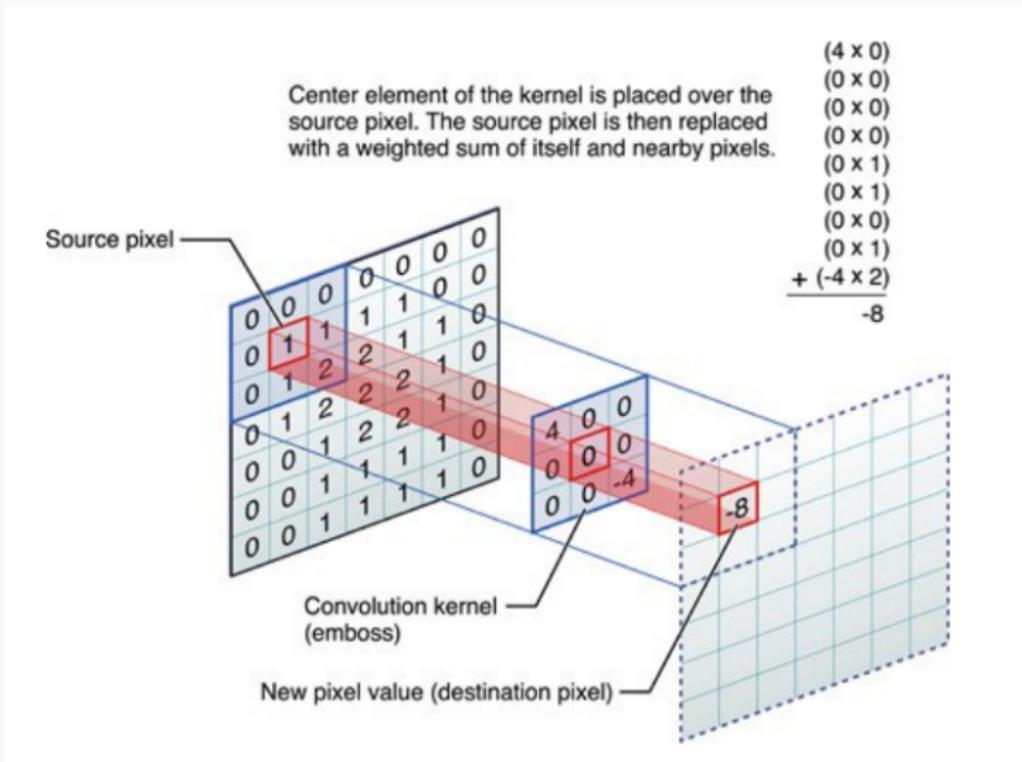


Локально-связанный слой
(смотрит на небольшую
область изображения)

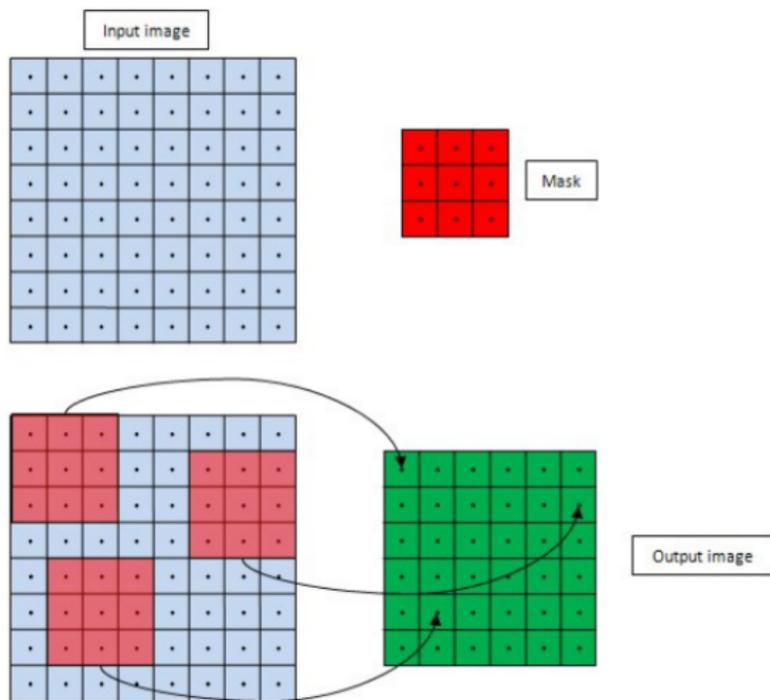


Инвариантность к перемещению

Операция свертки.



Инвариантность к перемещению



Простейшая свертка



Original

0	0	0
0	1	0
0	0	0

?

Простейшая свертка



Original

0	0	0
0	1	0
0	0	0



Filtered
(no change)

Простейшая свертка



Original

0	0	0
0	0	1
0	0	0

?

Простейшая свертка



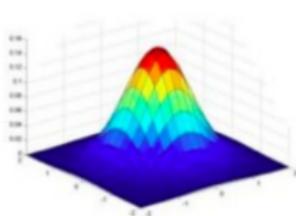
Original

0	0	0
0	0	1
0	0	0



Shifted left
By 1 pixel

$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$



0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, $\sigma = 1$

Original



Noisy



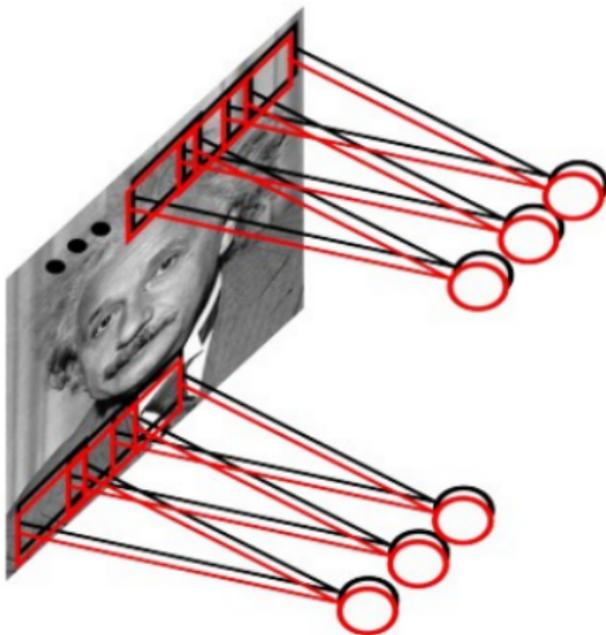
Smoothed



Сверточный слой

Локально-связанный слой с одинаковыми весами в разных частях изображения.

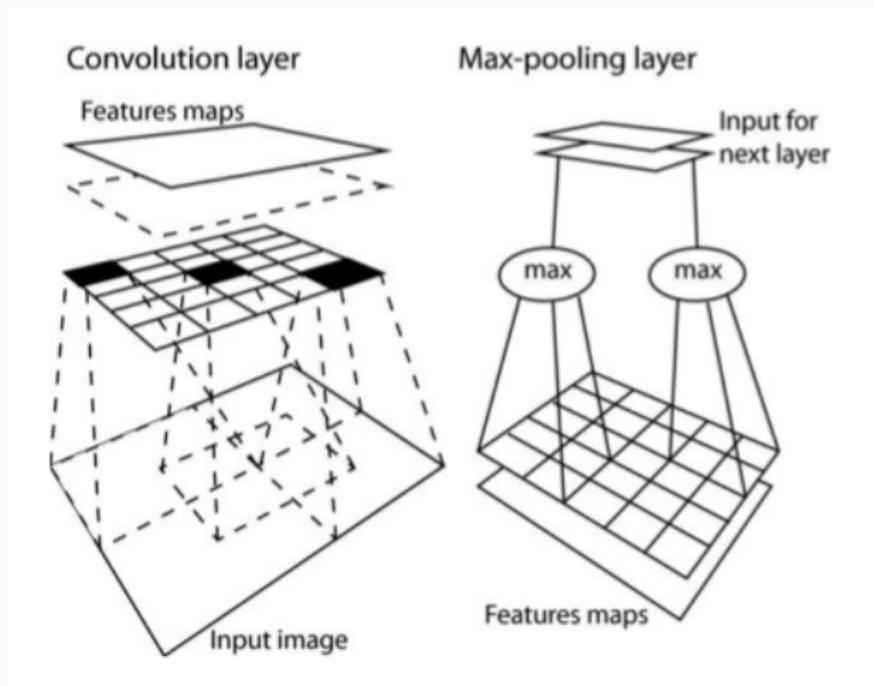
Закладывает в сеть априорное знание о том, что объект может встретиться в любой части изображения.



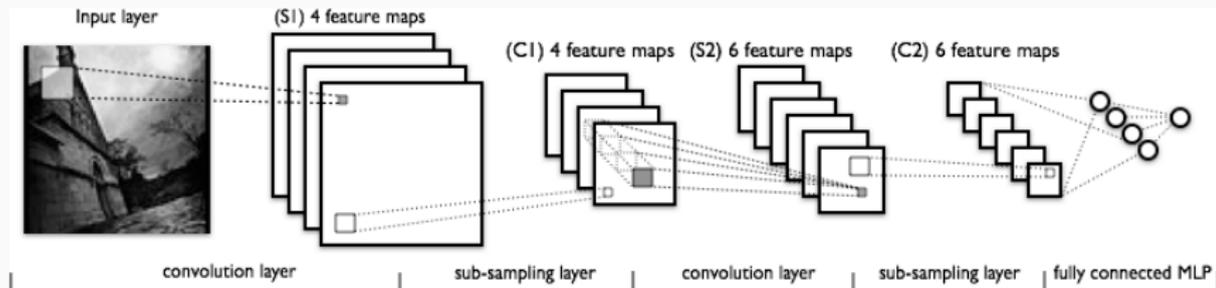
Subsampling слой

Добавляет устойчивости к небольшим деформациям.

Max-subsampling аналогичен сверточному слою, в котором операция «+» заменена на «max»



Сверточная нейросеть



Вопросы?

Что почитать по этой лекции

- Andrej Karpathy lecture notes (cs231n)
- Tom Mitchell "Machine Learning" Chapter 4.5

На следующей лекции

- Задача максимизации зазора - аналог классификатора с регуляризацией
- Двойственная задача
- Что такое опорный вектор
- Регуляризация
- Решение для неразделимых выборок
- Kernel trick