Введение в искусственные нейронные сети

Трифонов Владислав

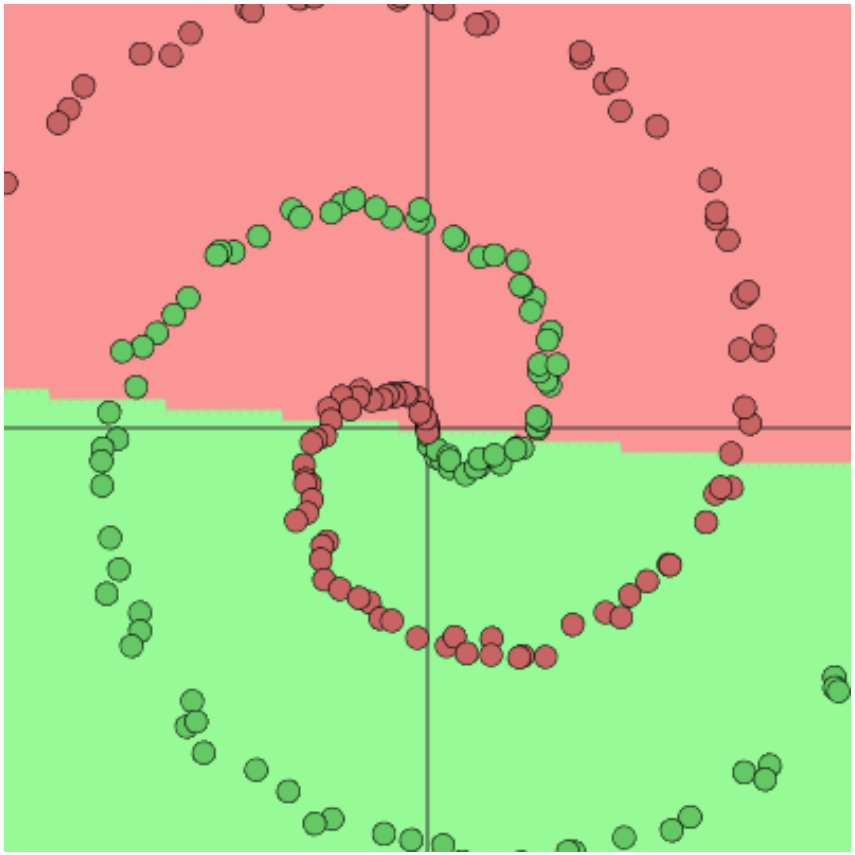
30 сентября 2020 г.

Введение

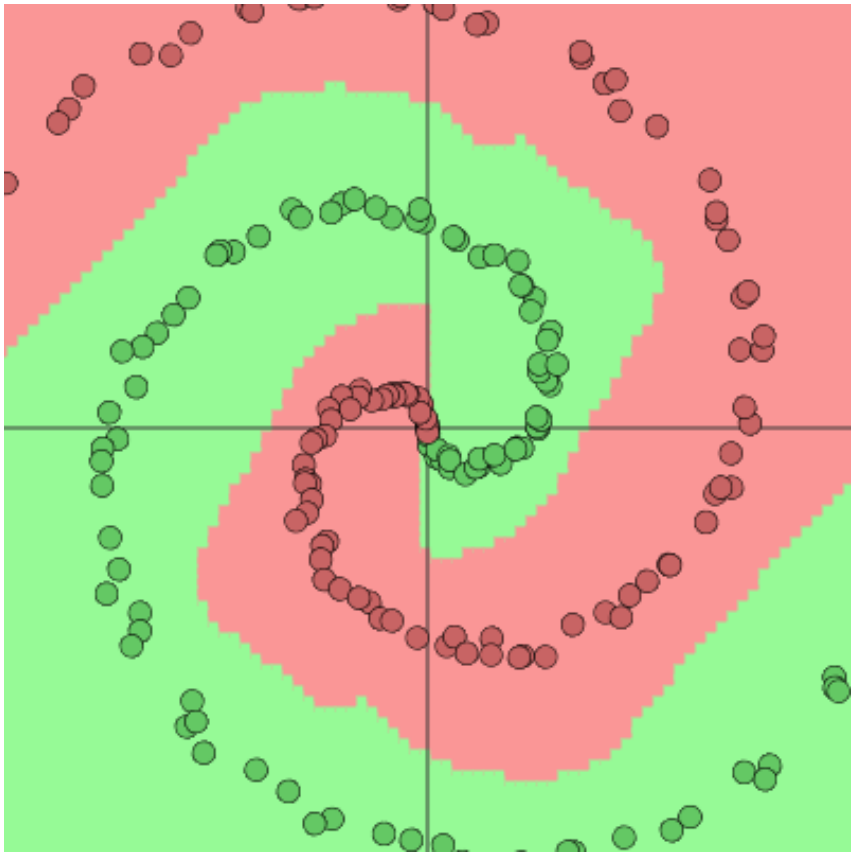
Нейронные сети – мощные алгоритмы машинного обучения, широко используемые при решении задач автоматической обработки текстов, изображений, видео и звуков.

В отличие от традиционных методов ML, они позволяют моделировать сложные зависимости в данных, автоматически строя признаковое пространство.

Введение



Линейный классификатор

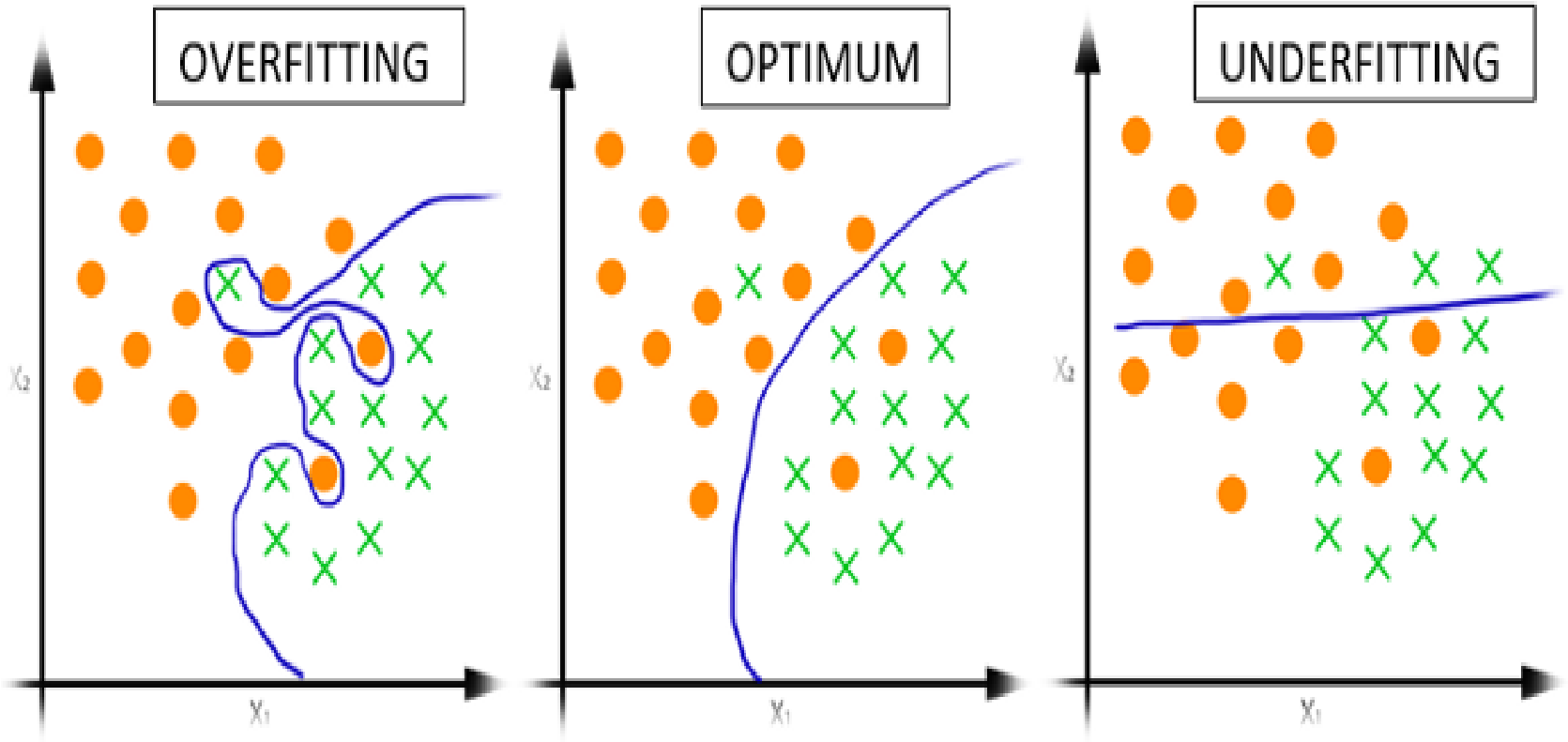


Нейронная сеть с одним скрытым слоем (tanh)

[0] cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

Введение

Чем “выразительнее” модель, тем больше шансов переобучиться!



[0] https://medium.com/@srjoglekar246/overfitting-and-human-behavior-5186df1e7d19

Обучение с учителем

Пусть

X – множество объектов

Y – множество ответов

$\tilde{y} : X \to Y$ – неизвестная зависимость

Дано

$\{x_1, ..., x_n\} \subset X$ – обучающая выборка

$\tilde{y}_i = \tilde{y}(x_i) \in Y$ – ответы на обучающей выборке

Найти

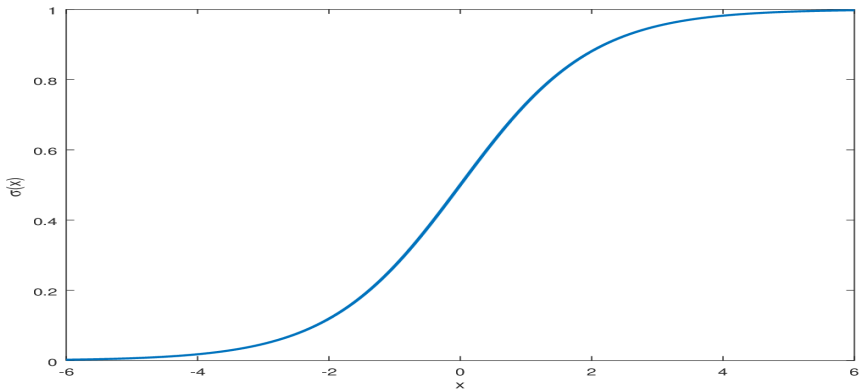
$y : X \to Y$ – алгоритм, решающую функцию, приближающую $\tilde{y}$ на всем множестве X

Логистическая регрессия

Дано

$\{\mathbf{x_1}, ..., \mathbf{x_n}\} \subset \mathbb{R}^d$ – обучающая выборка

$\tilde{y}_i = \tilde{y}(x_i) \in \{0, 1\}$ – ответы на обучающей выборке



Логистическая функция (сигмоида)

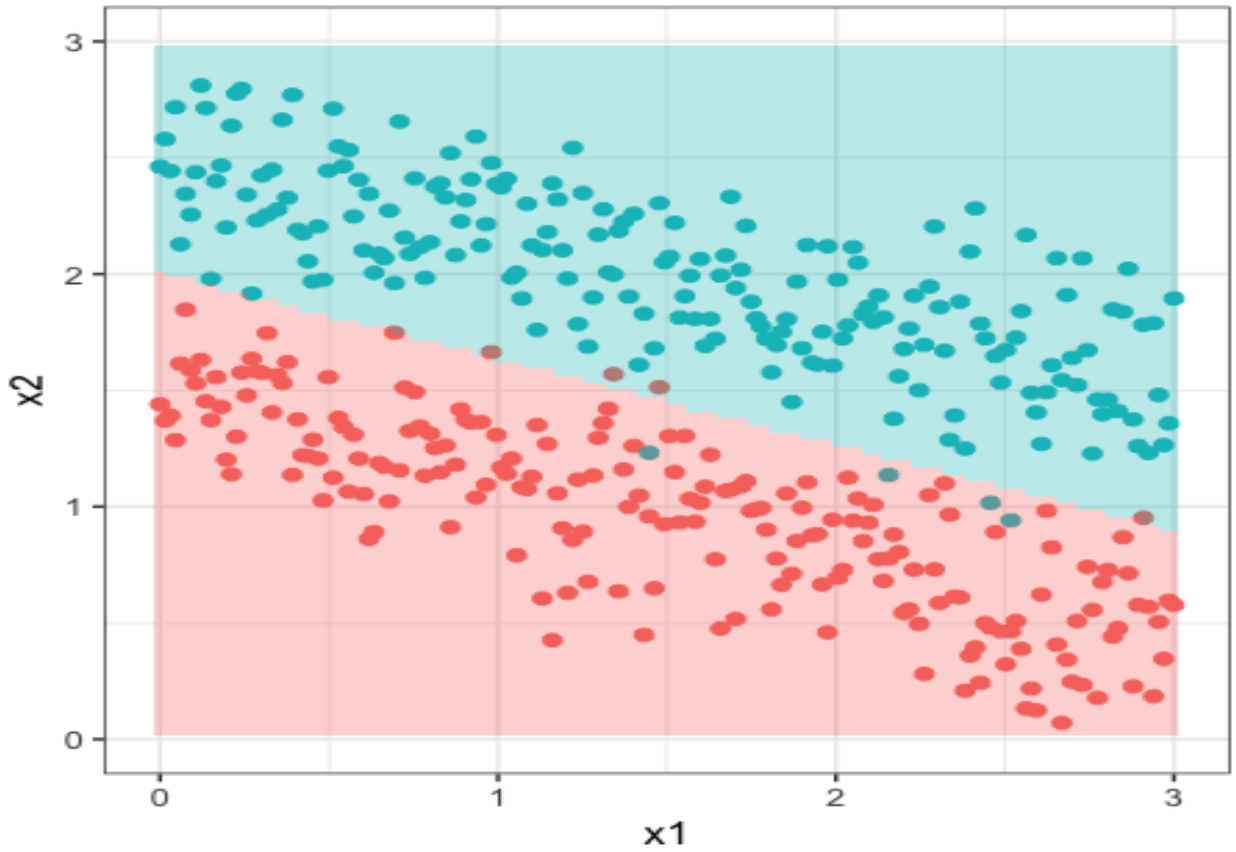
Решение

$$y = \sigma(\mathbf{w}^T \cdot \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T \cdot \mathbf{x})} \in (0, 1)$$

$$\mathbf{w} = \operatorname{argmin}_{\mathbf{w}} L_{\mathbf{w}}$$

$$L_{\mathbf{w}} = -\frac{1}{n} \sum_{i=1}^{n} \tilde{y}_i \log y(\mathbf{x_i}) + (1 - \tilde{y}_i) \log(1 - y(\mathbf{x_i}))$$

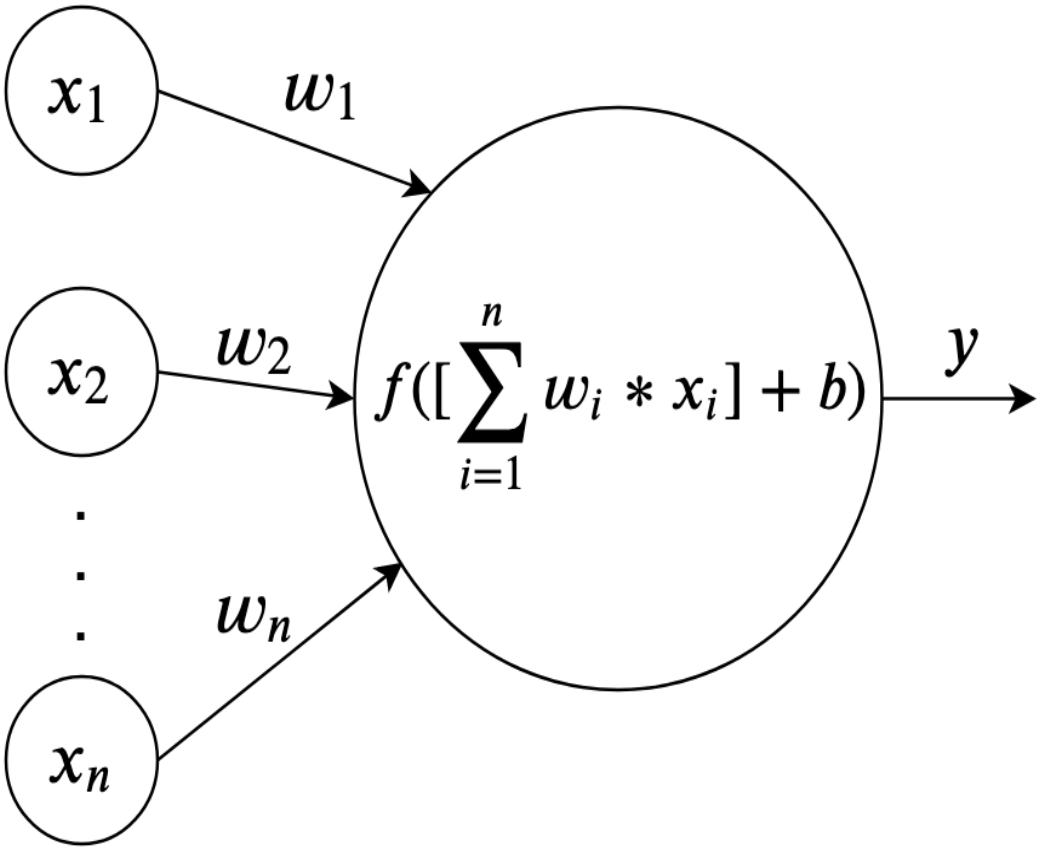
Логистическая регрессия



Решение $y(x) = 0.5$ – разделяющая гиперплоскость

[0] https://towardsdatascience.com/logistic-regression-from-scratch-in-r-b5b122fd8e83

Модель нейрона

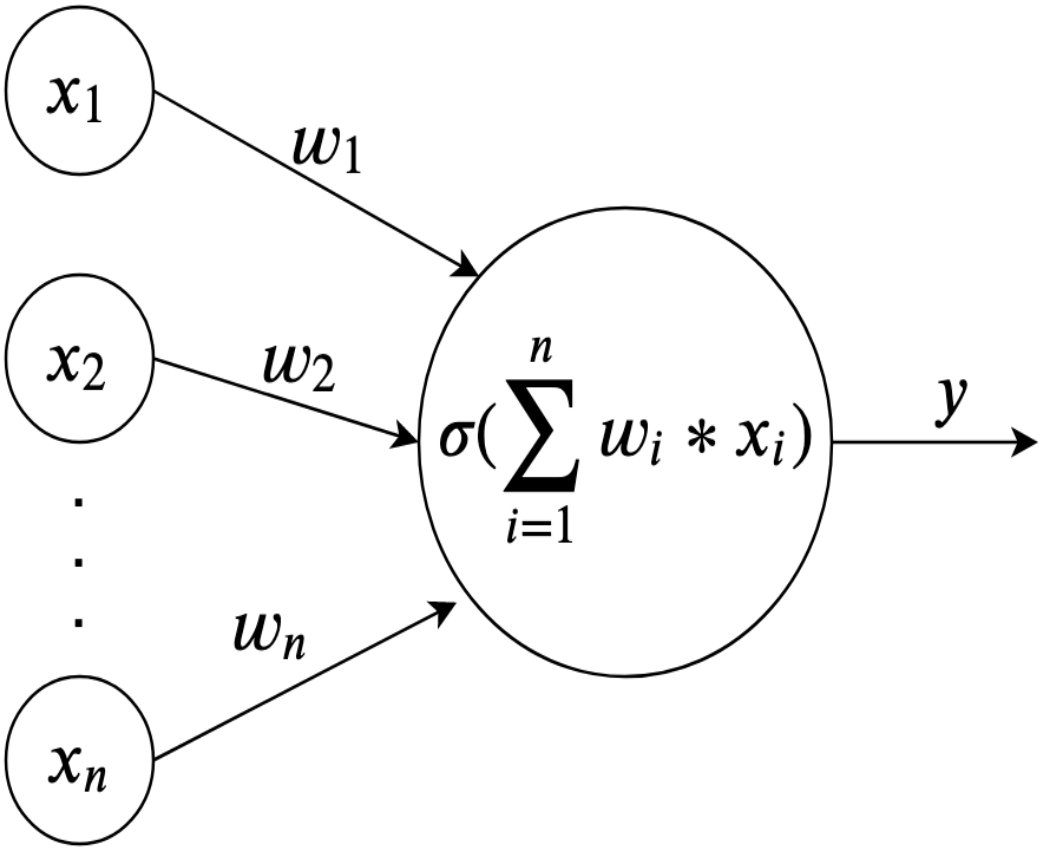


- W. McCulloch, W. Pitts – 1943
- $y = f(\mathbf{w}^T \cdot \mathbf{x} + b)$
- $\mathbf{w} \in \mathbb{R}^n$ – обучаемые веса связей
- $b \in \mathbb{R}$ – смещение (bias)
- f – функция активации (обычно нелинейная)

Модель нейрона

- Нейрон линейно комбинирует входные признаки и с помощью функции активации генерирует число – сигнал, который можно принять за ответ модели или использовать его в качестве нового признака
- Линейные модели (линейная регрессия, линейный классификатор, логистическая регрессия) – простейшие нейронные сети с одним нейроном

Логистическая регрессия как нейрон

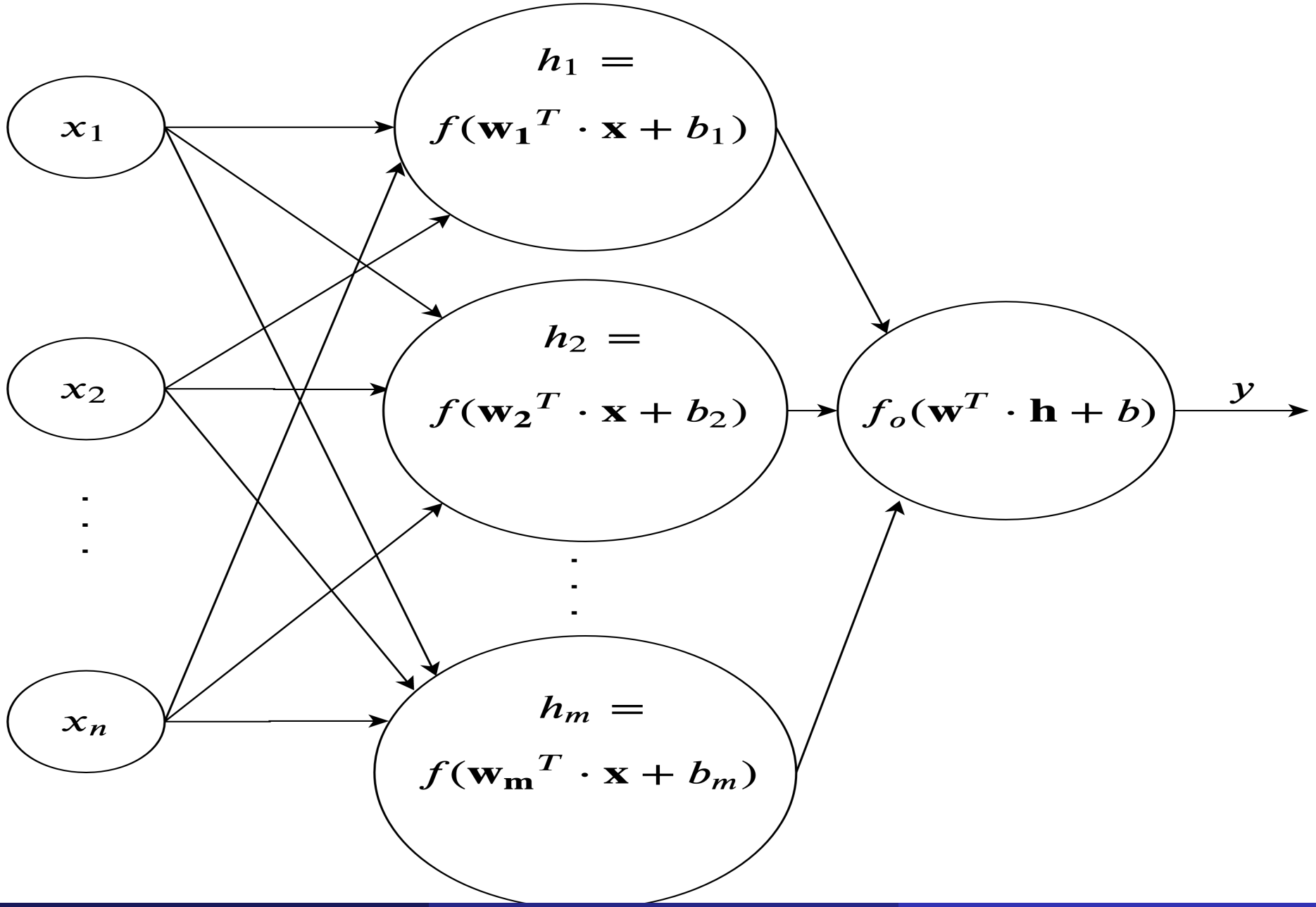


- $y = \sigma(\mathbf{w}^T \cdot \mathbf{x}) = \dfrac{1}{1 + \exp(-\mathbf{w}^T \cdot \mathbf{x})}$
- оптимизируя функцию потерь методом градиентного спуска, находим параметры $\mathbf{w}$

Идея

Перед формированием ответа модели y можно скомбинировать и нелинейно преобразовать исходные признаки $\mathbf{x}$, причем параметры этой трансформации должны быть обучаемыми

Персептрон с одним скрытым слоем



Персептрон

- Скрытый слой можно записать как $\mathbf{h} = f(\mathbf{W}\mathbf{x} + \mathbf{b})$; $\mathbf{W} \in \mathbb{R}^{m \times n}$; $\mathbf{x} \in \mathbb{R}^n$; $\mathbf{b}, \mathbf{h} \in \mathbb{R}^m$
- На практике используется не более 2-3 скрытых слоев. Больше параметров – выше вероятность переобучения

Персептрон

$y = f_o(\mathbf{h})$ – выходной слой. В качестве f_o обычно используется линейная модель:

- $y = \mathbf{w}^T \cdot \mathbf{h} \in (-\infty, +\infty)$ – регрессия
- $y = \sigma(\mathbf{w}^T \cdot \mathbf{h}) = \dfrac{1}{1 + \exp -(\mathbf{w}^T \cdot \mathbf{h})} \in (0, 1)$ – бинарная классификация

- $\mathbf{y} = softmax(\mathbf{W}\mathbf{h}) \in (0, 1)^c, \mathbf{W} \in \mathbb{R}^{c \times m}$ – классификация на $c > 2$ классов

$$softmax_i(\mathbf{x}) = \frac{exp(x_i)}{\sum_{k=1}^{n} exp(x_k)}$$

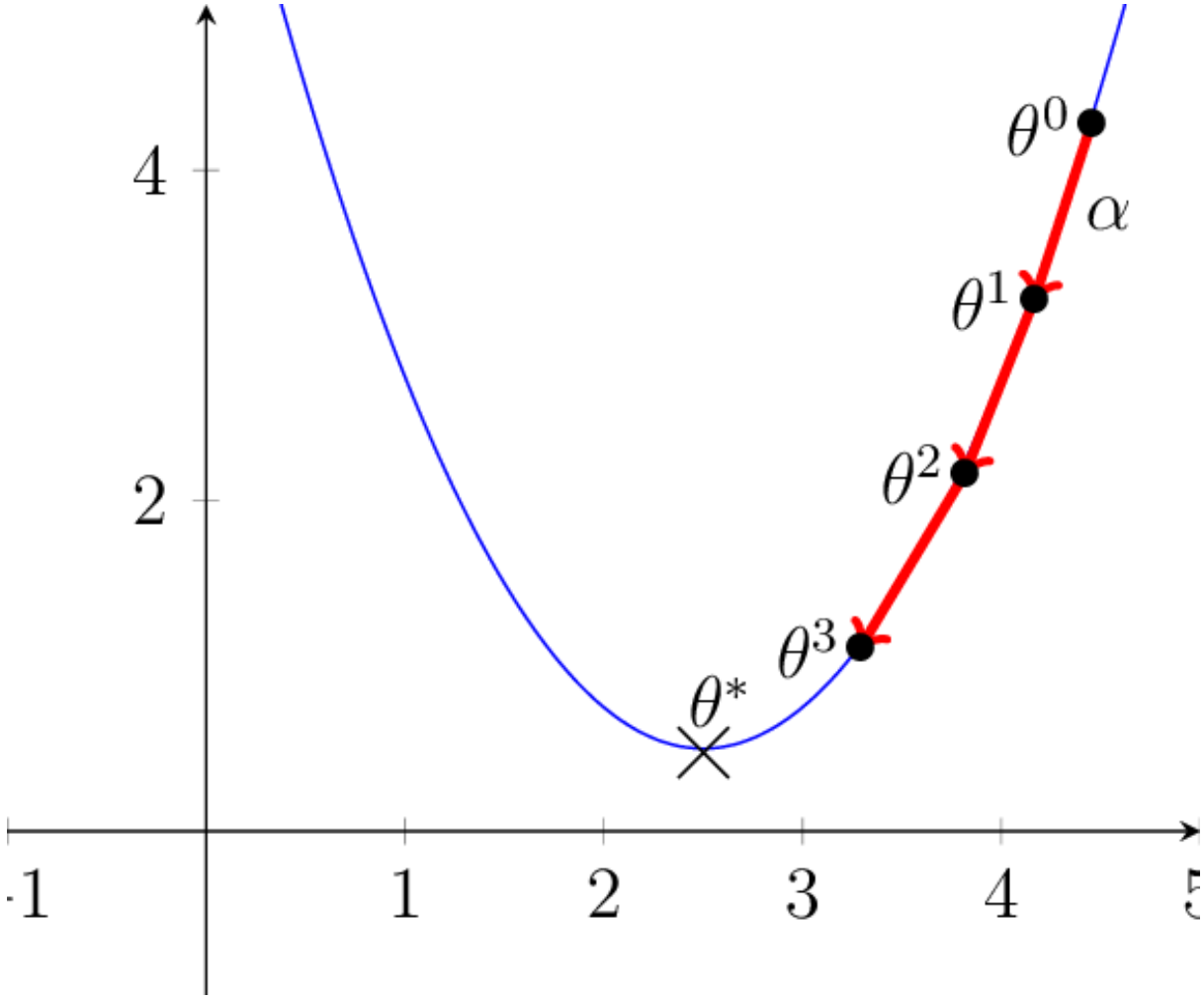
Обучение нейронной сети с учителем

- Пусть $(x_i, \tilde{y}_i), i \in [1, n]$ – обучающая выборка, где x_i – объекты, $\tilde{y}_i$ – правильные ответы
- Пусть $y = y_\theta(x)$ – дифференцируемая нейронная сеть с параметрами θ. Процесс обучения заключается в нахождении оптимальных параметров
- Параметры θ должны быть такими, чтобы нейронная сеть выдавала “близкие” к правильным ответам на элементах обучающей выборки

Обучение нейронной сети с учителем

- Для оценки погрешности алгоритма выбирается функция ошибки $l = l(y', \tilde{y'}), l \in \mathbb{R}$. К примеру, из раннее рассмотренных:
  - MSE (линейная регрессия): $l = (y' - \tilde{y'})^2$
  - cross-entropy (логистическая регрессия): $l = -(\tilde{y'} \log y' + (1 - \tilde{y'}) \log(1 - y'))$
- По всей обучающей выборке строим функционал качества, применяя нейронную сеть и сравнивая ее ответы с правильными:
$$L = L(\theta) = \frac{1}{n} \sum_{i=1}^{n} l(y_\theta(x_i), \tilde{y}_i) = \frac{1}{n} \sum_{i=1}^{n} l_i$$
- Т.к. L – дифференцируемая функция, то можем пытаться попасть в ее локальный минимум, оптимизируя параметры θ с помощью градиентного спуска: $\theta^{i+1} = \theta^i - \alpha \cdot \nabla L(\theta^i)$

Обучение нейронной сети с учителем



Несколько шагов градиентного спуска

SGD

- Вычисление L при больших n – ресурсоемкая задача
- Stochastic gradient descent (SGD) – на каждом шаге градиентного спуска случайно выбираем **один** пример j из обучающей выборки и обновляем параметры: $\theta^{i+1} = \theta^i - \alpha \cdot \nabla l_j(\theta^i)$
- SGD – подвержен “выбросам”, поэтому шаг α для сходимости следует выбирать маленьким, что приводит к большому числу шагов, требуемых для сходимости
- Вычисление ошибки не параллелится по элементам обучающей выборки

miniBatch SGD

- miniBatch SGD – на каждом шаге градиентного спуска случайно выбираем **m** примеров ($1 < m << n$) из обучающей выборки и обновляем параметры: $\theta^{i+1} = \theta^i - \alpha \cdot \nabla[\frac{1}{m} \sum_{j=1}^{m} l_j(\theta^i)]$
- Более устойчив к “выбросам”, поэтому можем выбирать больший шаг α
- Ошибку $\sum_{j=1}^{m} l_j(\theta^i)$ можно считать параллельно

Обучение

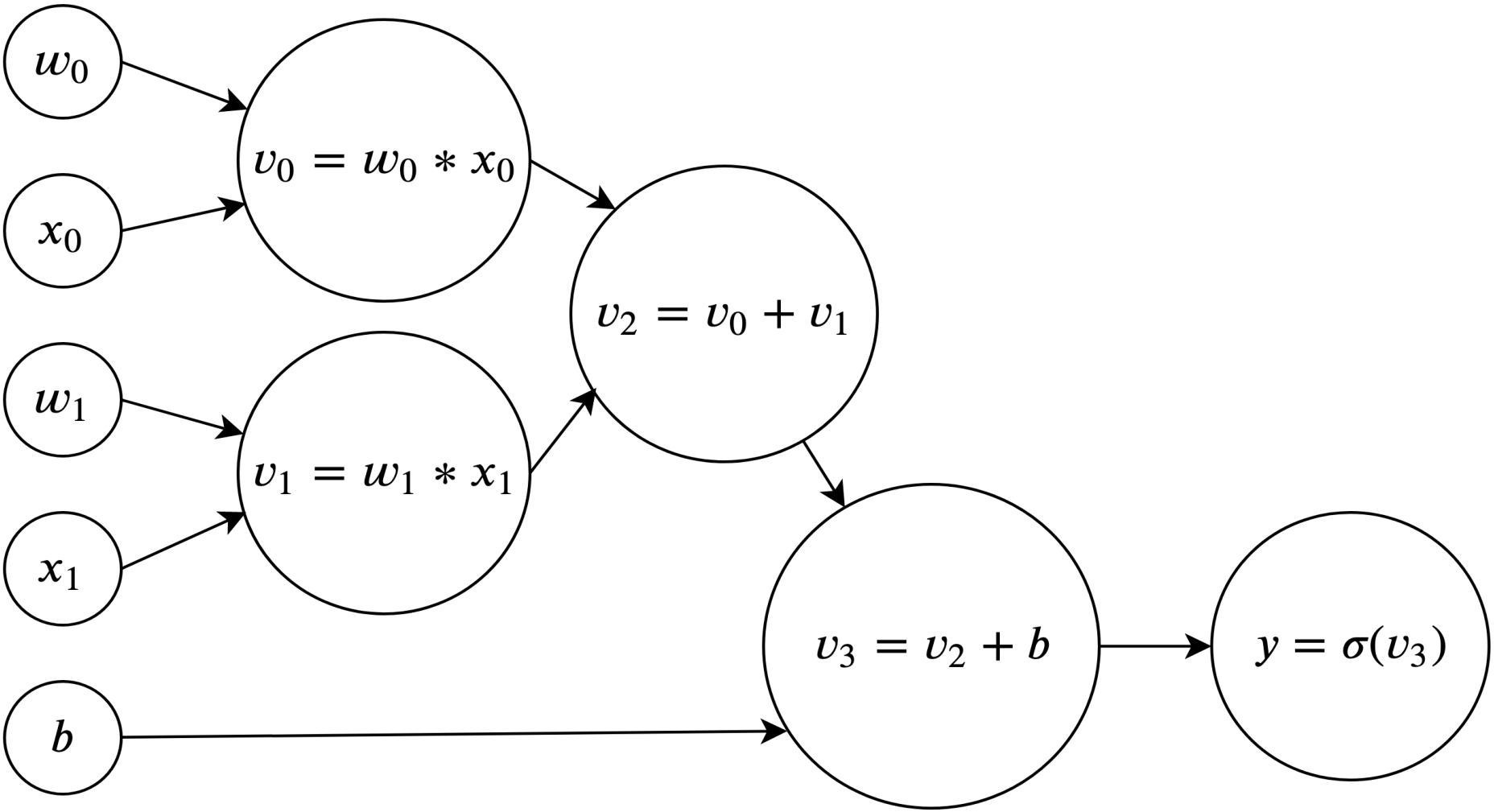
Обучение проводят некоторое заданное число “эпох” – полных итераций по обучающей выборке и останавливаются, когда качество на валидационной выборке перестает улучшаться.
В настоящее время, в основном, используются адаптивные методы градиентного спуска, которые задают значение α на каждом шаге в зависимости от предыдущих значений градиента – Momentum, RMSProp, Adam и т.д.

Вопрос

Как вычислять градиент $\nabla L(\theta^i)$ по каждому параметру θ^i_j глубокой нейронной сети (в общем случае, сложной дифференцируемой функции)?

Граф вычислений

Логистическая регрессия: $y = \frac{1}{1+\exp -(w_0 \cdot x_0 + w_1 \cdot x_1 + b)}$



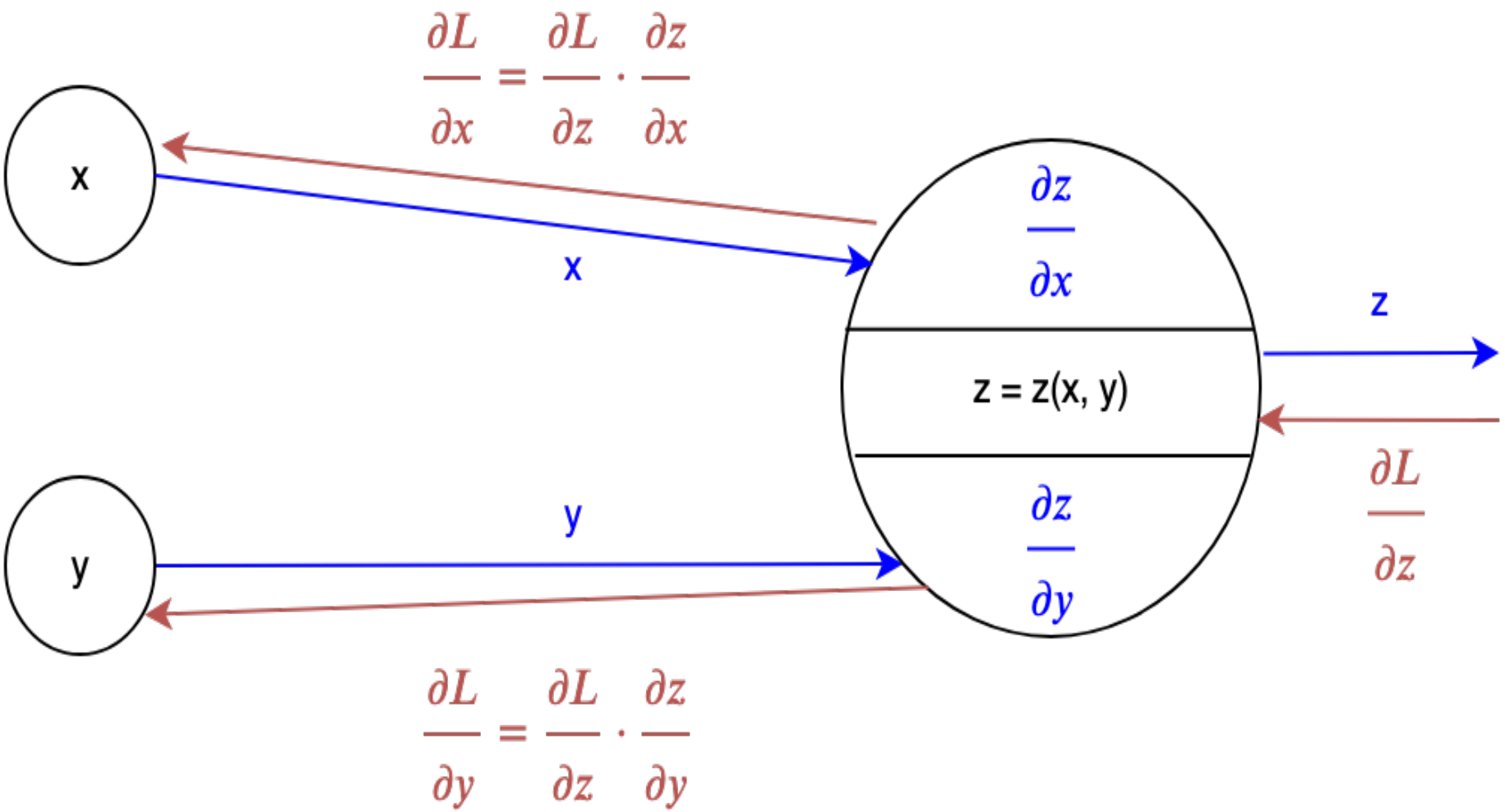
Backpropagation

Backpropagation

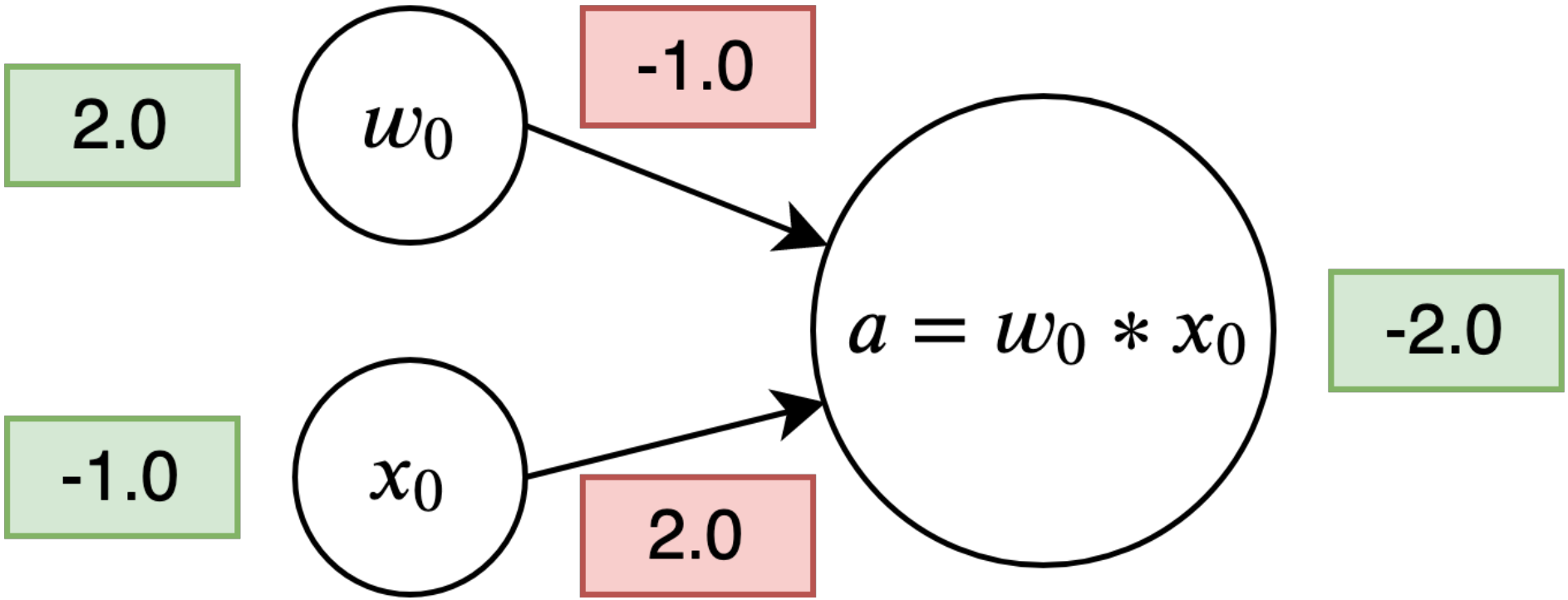
На входе граф вычислений некоторой дифференцируемой функции y.
Алгоритм:

1. Вычисляем значения и частные производные для каждого из узлов (forward pass)
2. Вычисляем частные производные реализуемой функции y по каждому из ее аргументов – узлов графа (backward pass)

Backpropagation



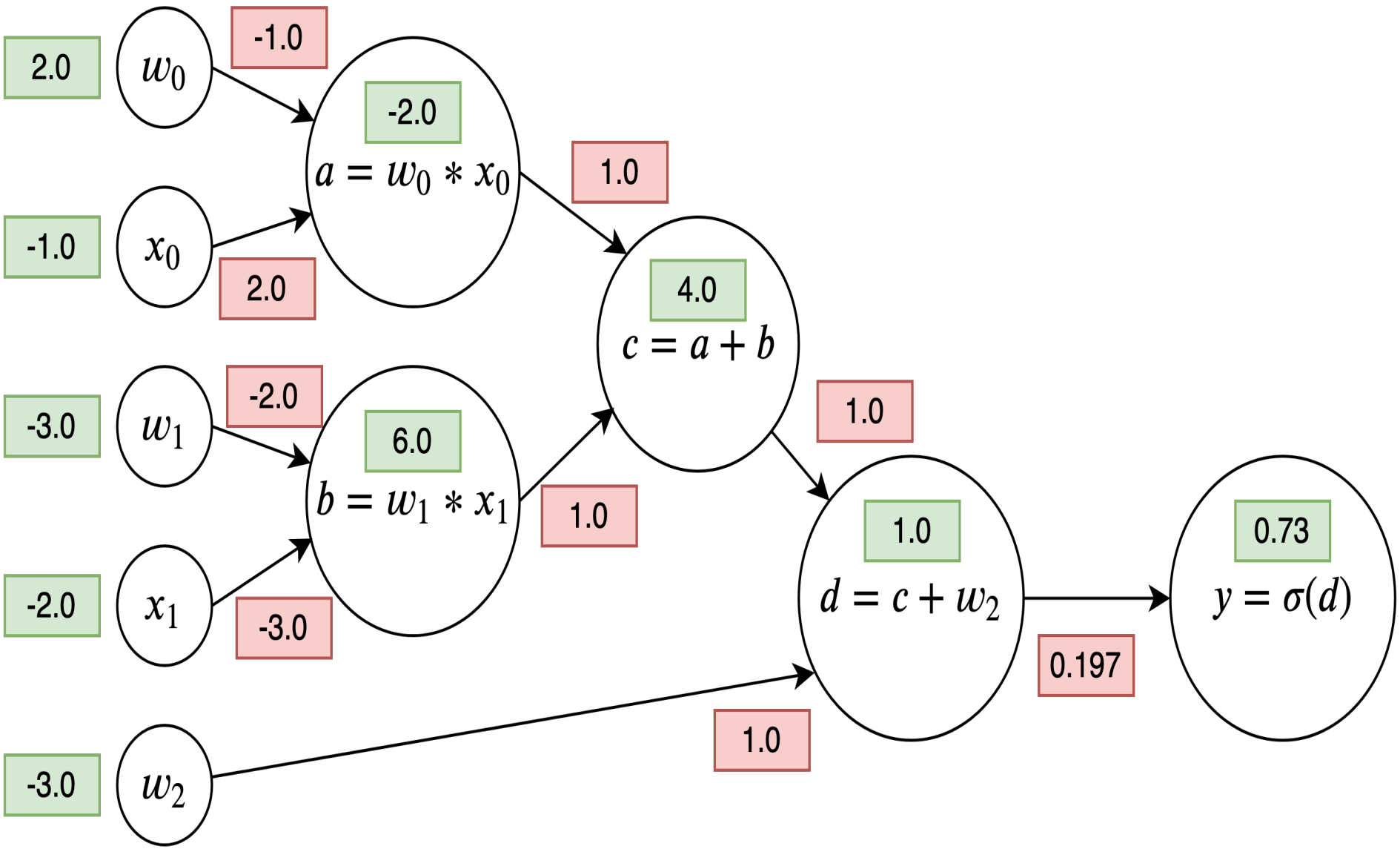
Forward pass



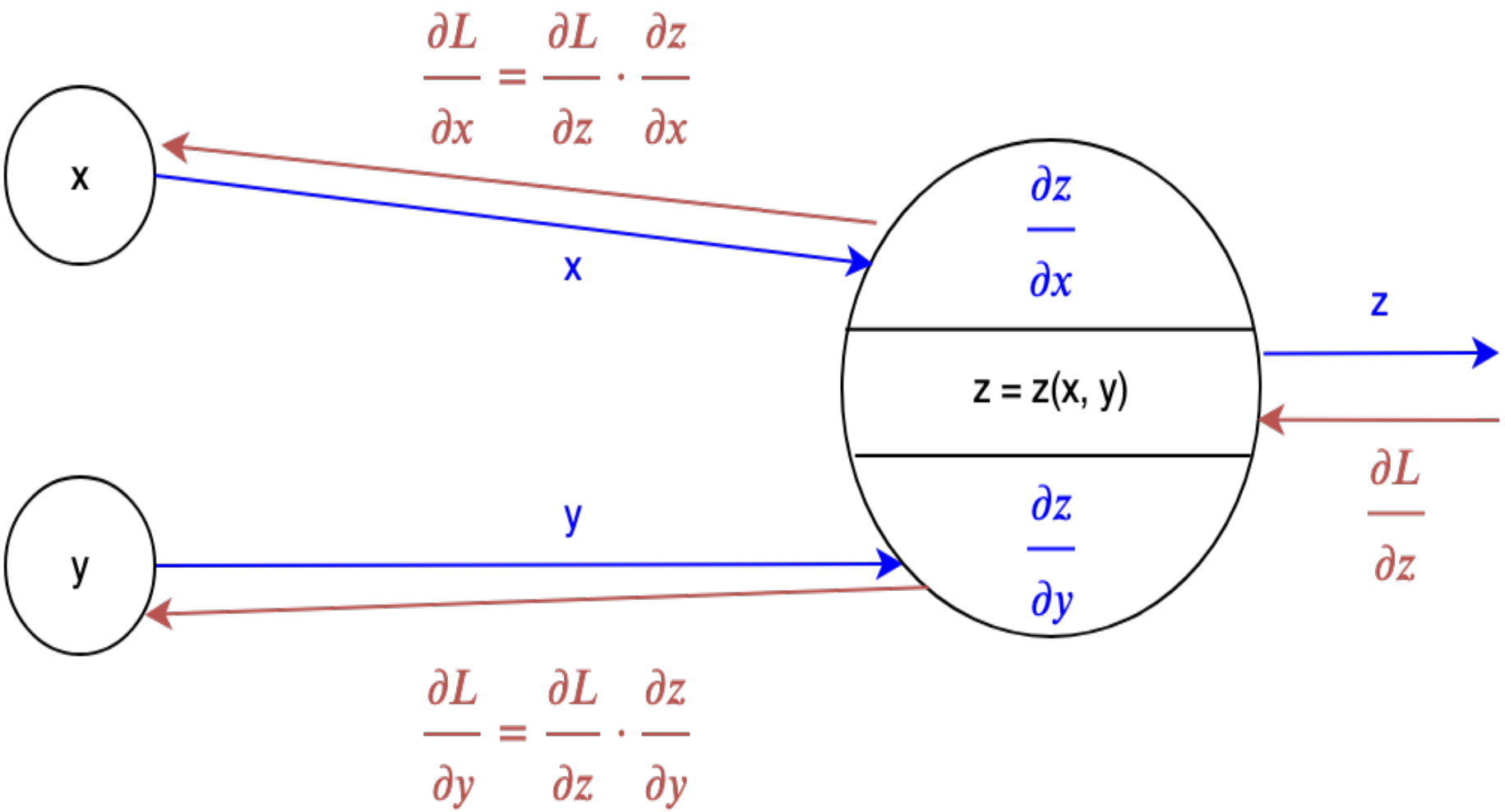
зеленый цвет – значение функции

красный цвет – значение частной производной

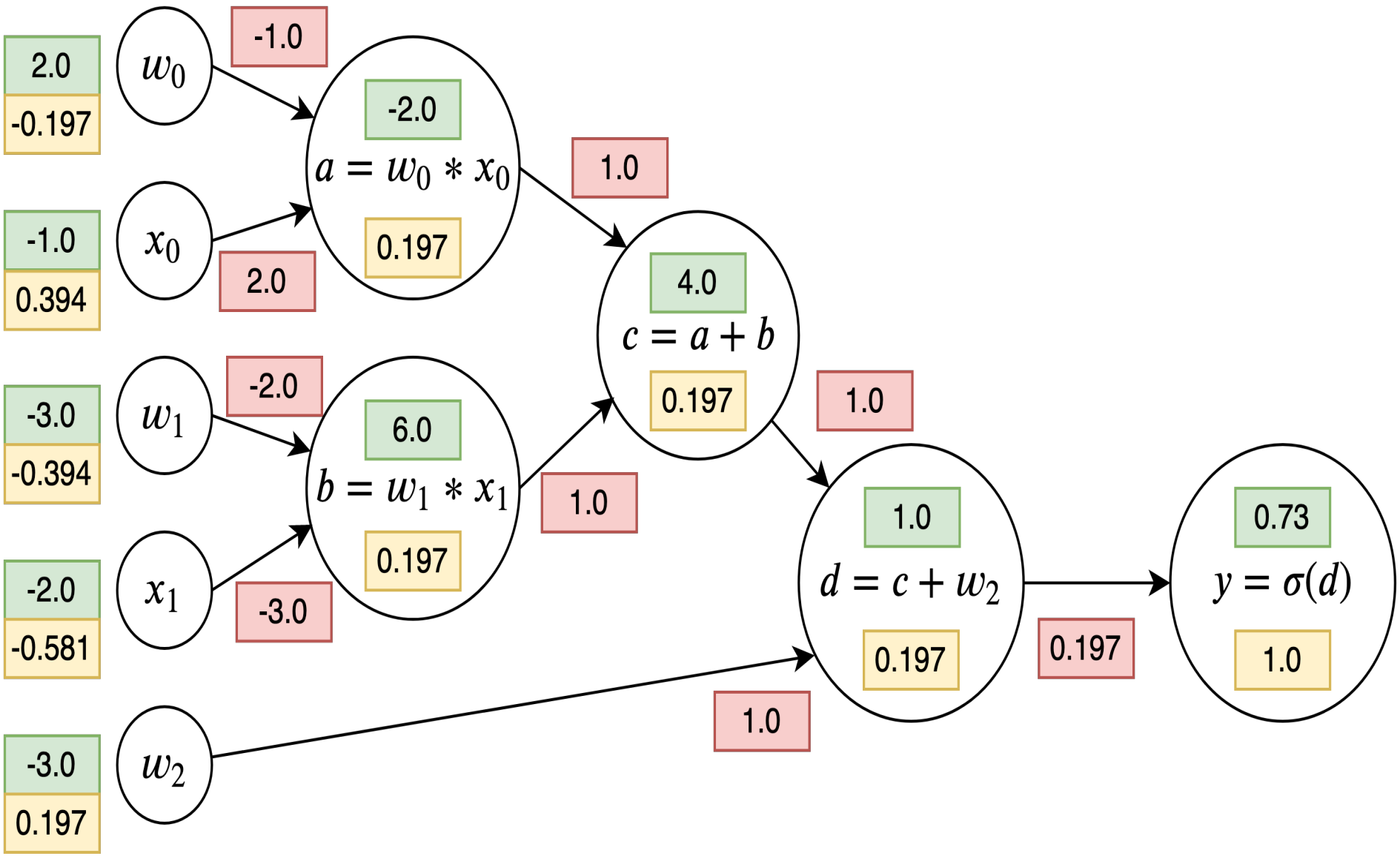
Пример forward pass



Backpropagation



Пример backward pass

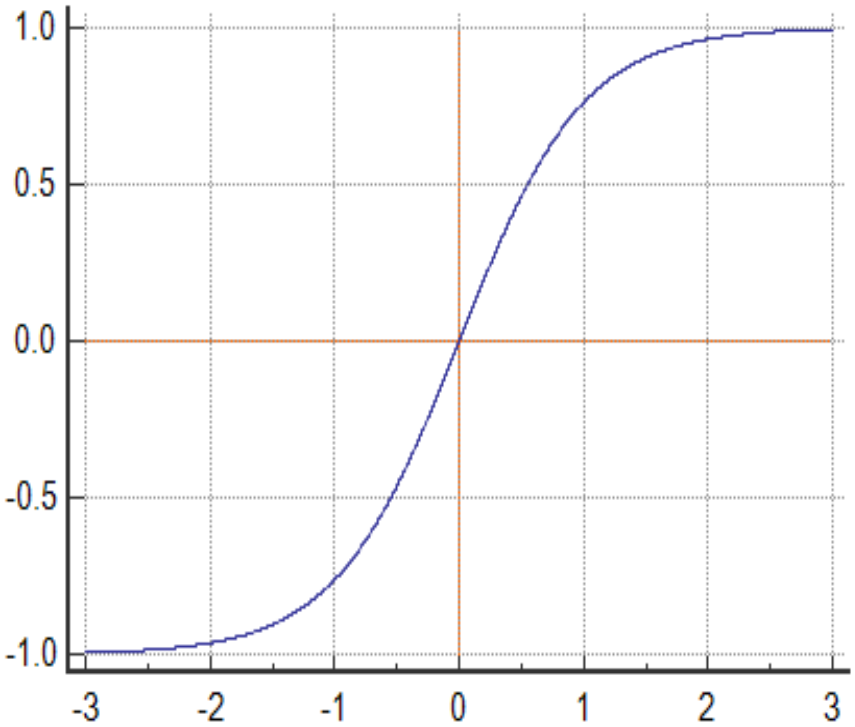


Проблема затухающих градиентов

При backpropagation в глубокой нейронной сети градиент на первых слоях может быть очень маленьким из-за последовательного умножения на $|\frac{\partial z}{\partial y}| << 1$, поэтому эти слои будут обновляться “слабо”.

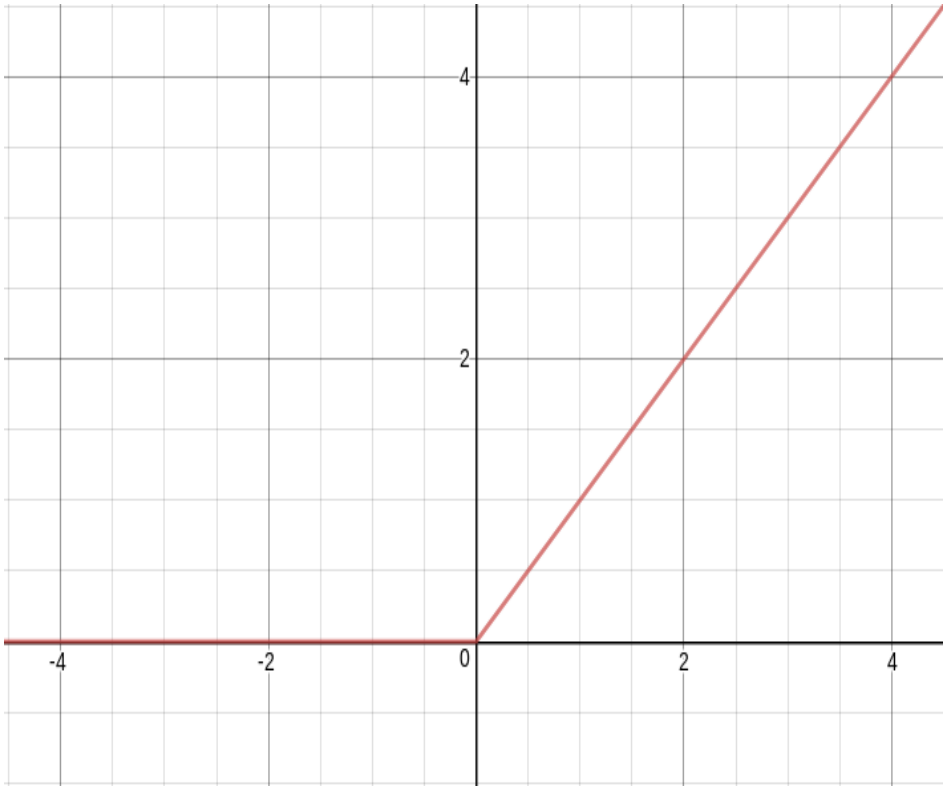
Чтобы с этим бороться, выбирают специальные функции активации и разрабатывают специальные архитектуры, распространяющие градиент на первые слои (skip / residual connections).

Функции активации – tanh



- $y = \tanh(x)$
- $y \in [-1, 1]$
- $\frac{\partial y}{\partial x} \leq 1.0$

Функции активации – ReLU



- $y = ReLU(x) = \max(0, x)$
- $y \in [0, +\infty)$
- $\frac{\partial y}{\partial x} \in \{0, 1\}$
- эффективное вычисление
- Часто используется в скрытых слоях

Начальная инициализация

Перед обучением нейронной сети нужно проинициализировать ее параметры θ.

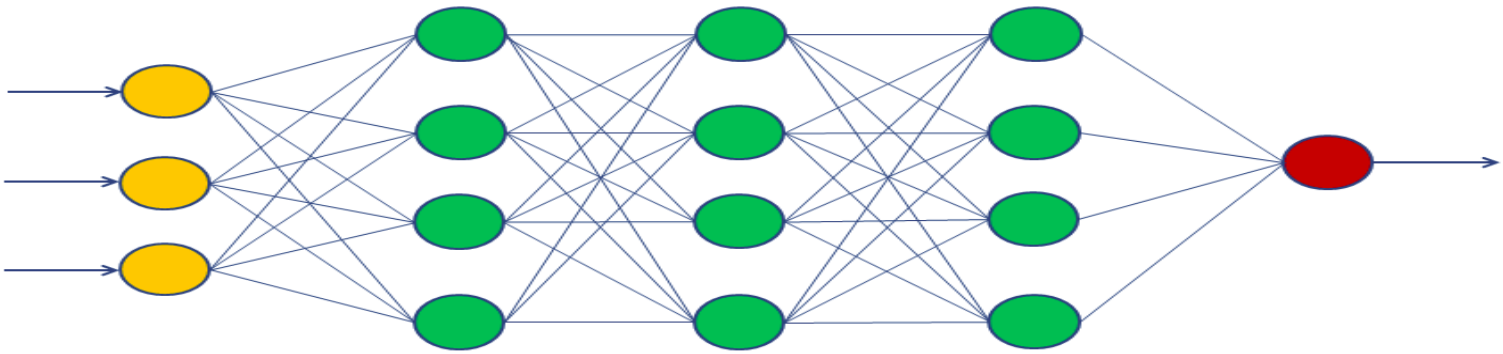
Почему все веса нельзя инициализировать 0? Константой?

- Обычно $\mathbf{W} \sim \mathcal{N}(0, \frac{1}{dim(\mathbf{W})})$, но все зависит от распределения $\mathbf{x}$
- Существуют более продвинутые инициализации, к примеру метод Xavier
- Нужно аккуратно подбирать начальную инициализацию весов, функции активации, размеры и количество слоев на валидационной выборке – проблемы недообучения / переобучения

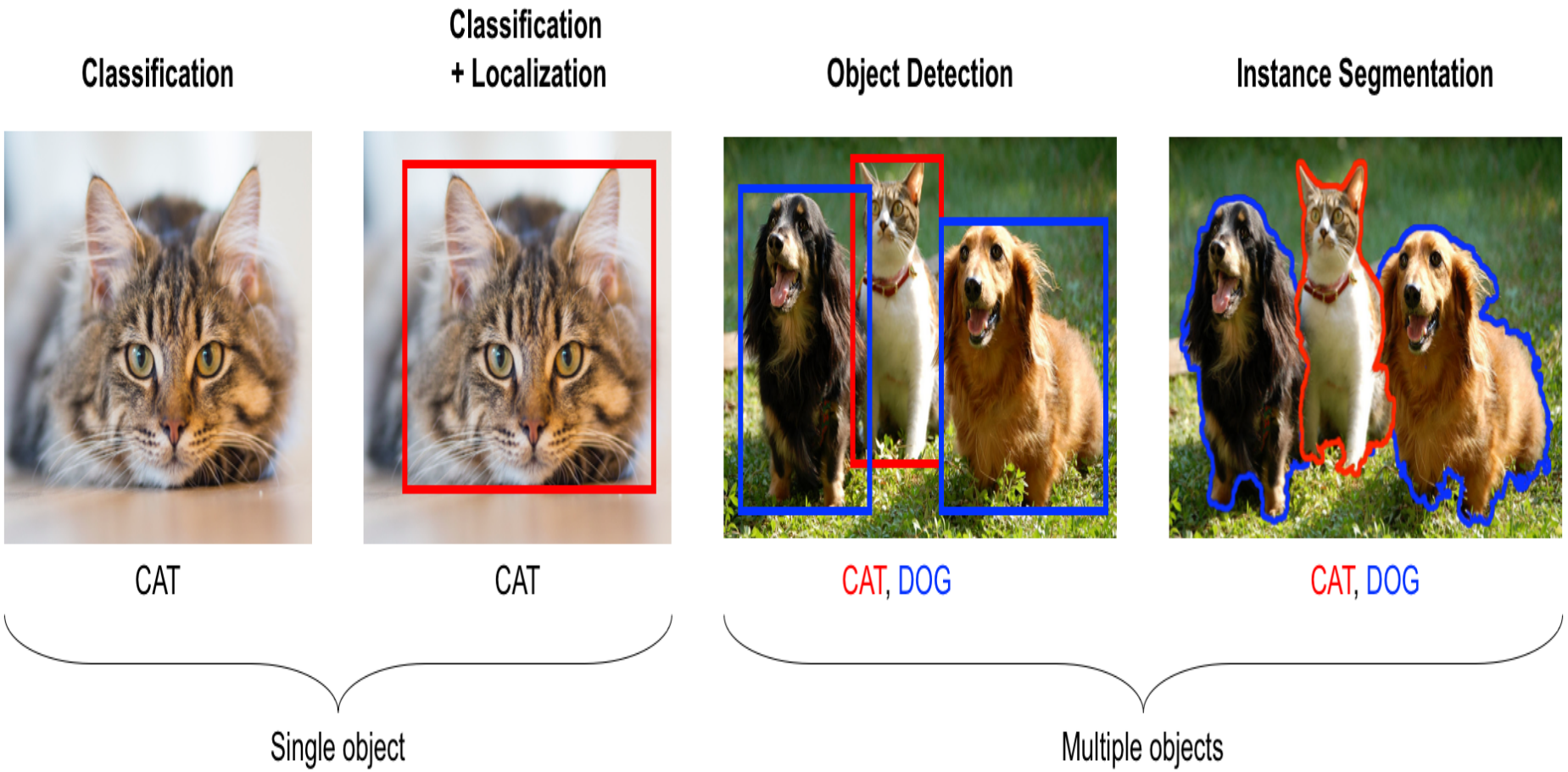
Где используются персептроны

В настоящее время многослойные персептроны редко применяются напрямую для решения задач машинного обучения. Предпочитают использовать более стабильные и простые в обучении методы градиентного бустинга над деревьями, SVM и линейные модели.

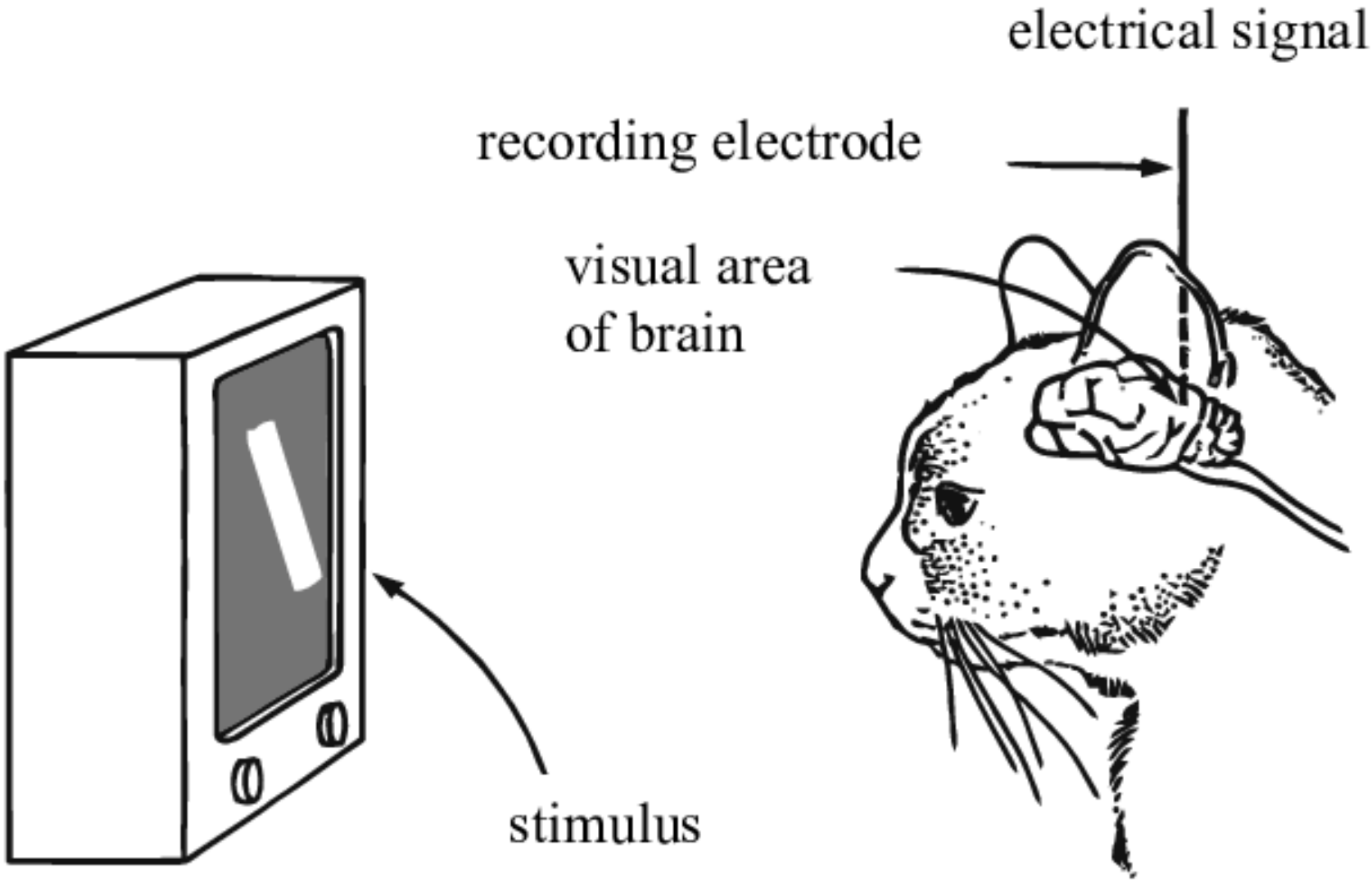
Концепция персептрона лежит в основе более сложных нейронных сетей. Кроме того, его зачастую используют в качестве слоя нелинейного преобразования.



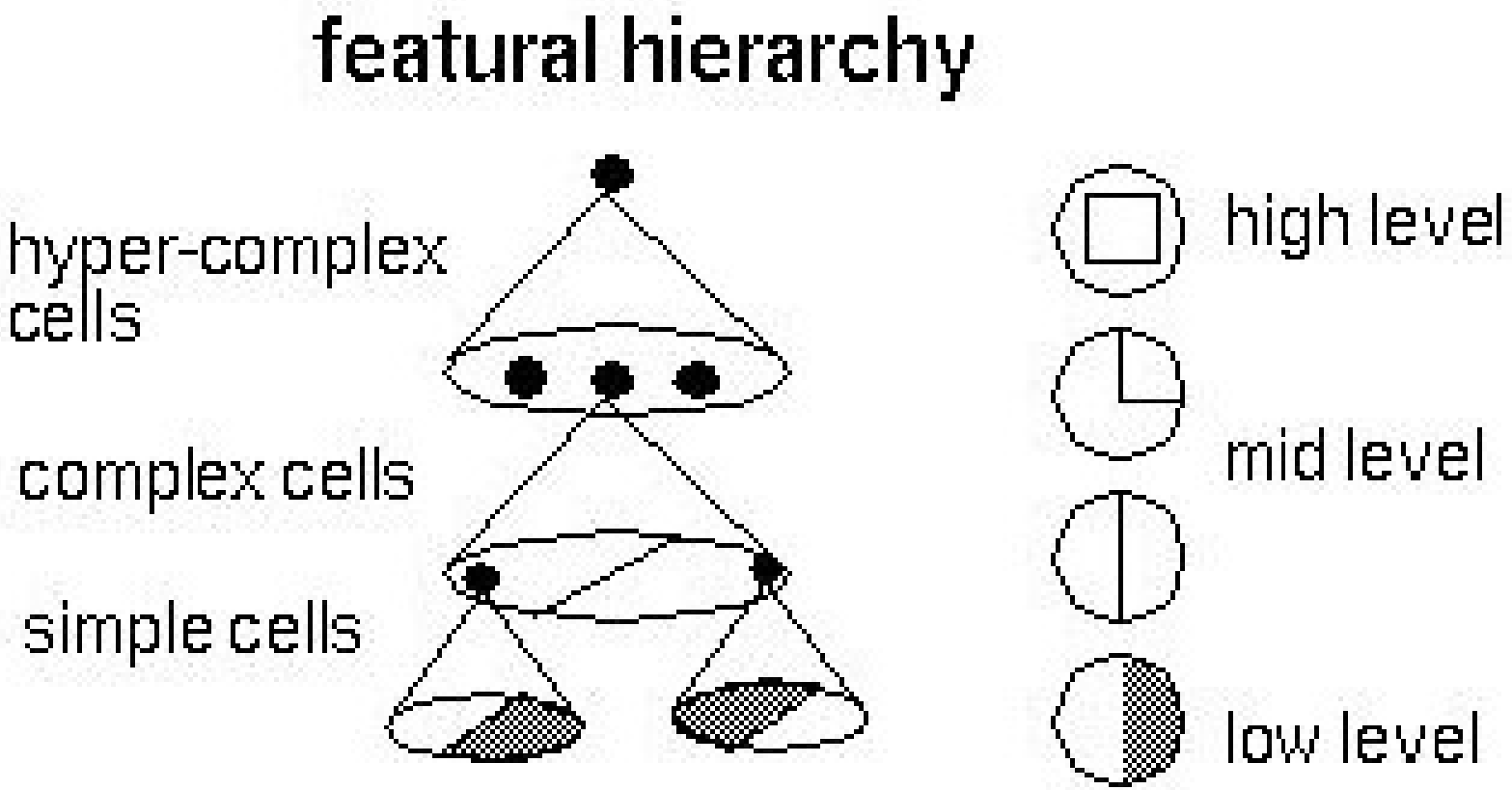
Сверточные нейронные сети



Эксперимент Хьюбела и Визеля (1960е)



Эксперимент Хьюбела и Визеля (1960е)

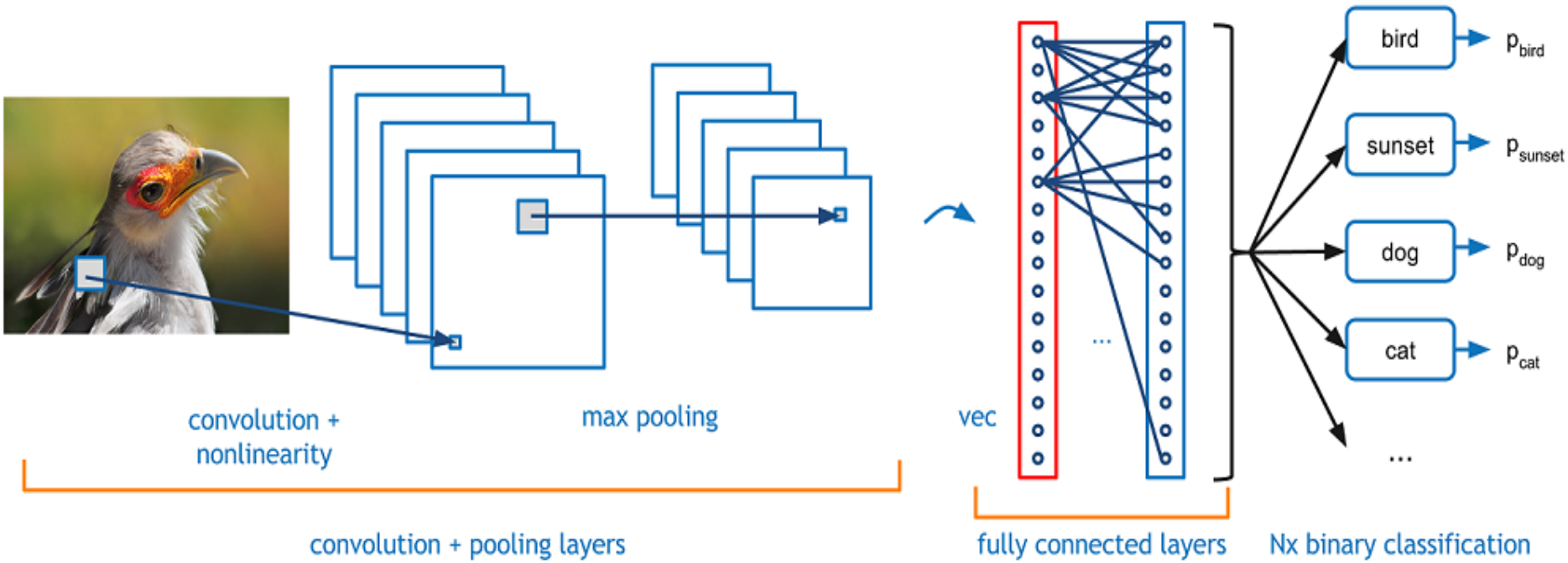


Почему не подходит обычный персептрон

Черно-белое изображение 32x32 => входной вектор признаков размерности 1024.
Возьмем один скрытый слой размерности 1024. Матрица такого преобразования содержит $1024 * 1024 = 1048576$ параметров.

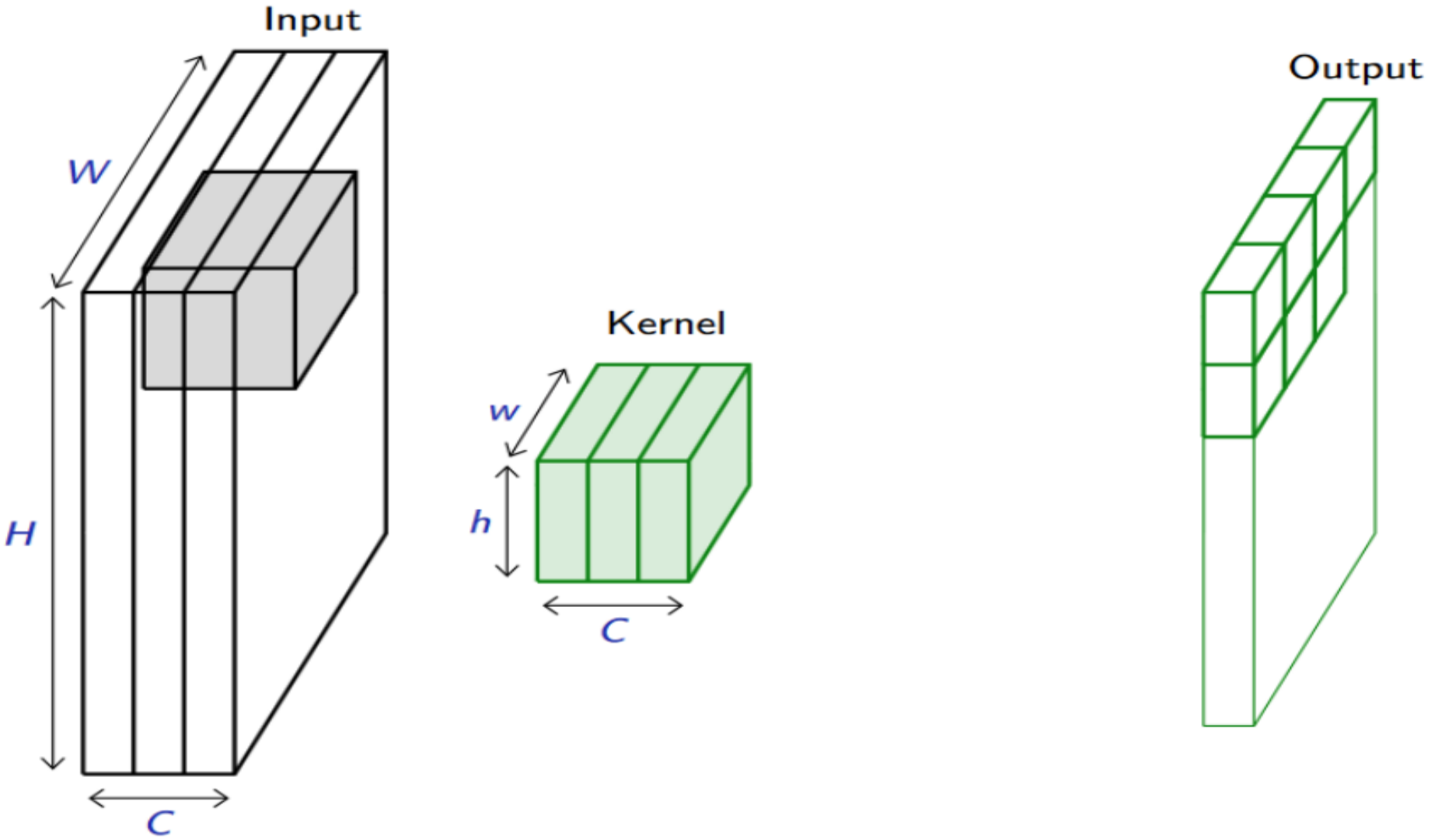
- склонность к переобучению
- не учитывается локальная связь признаков

Сверточная нейронная сеть

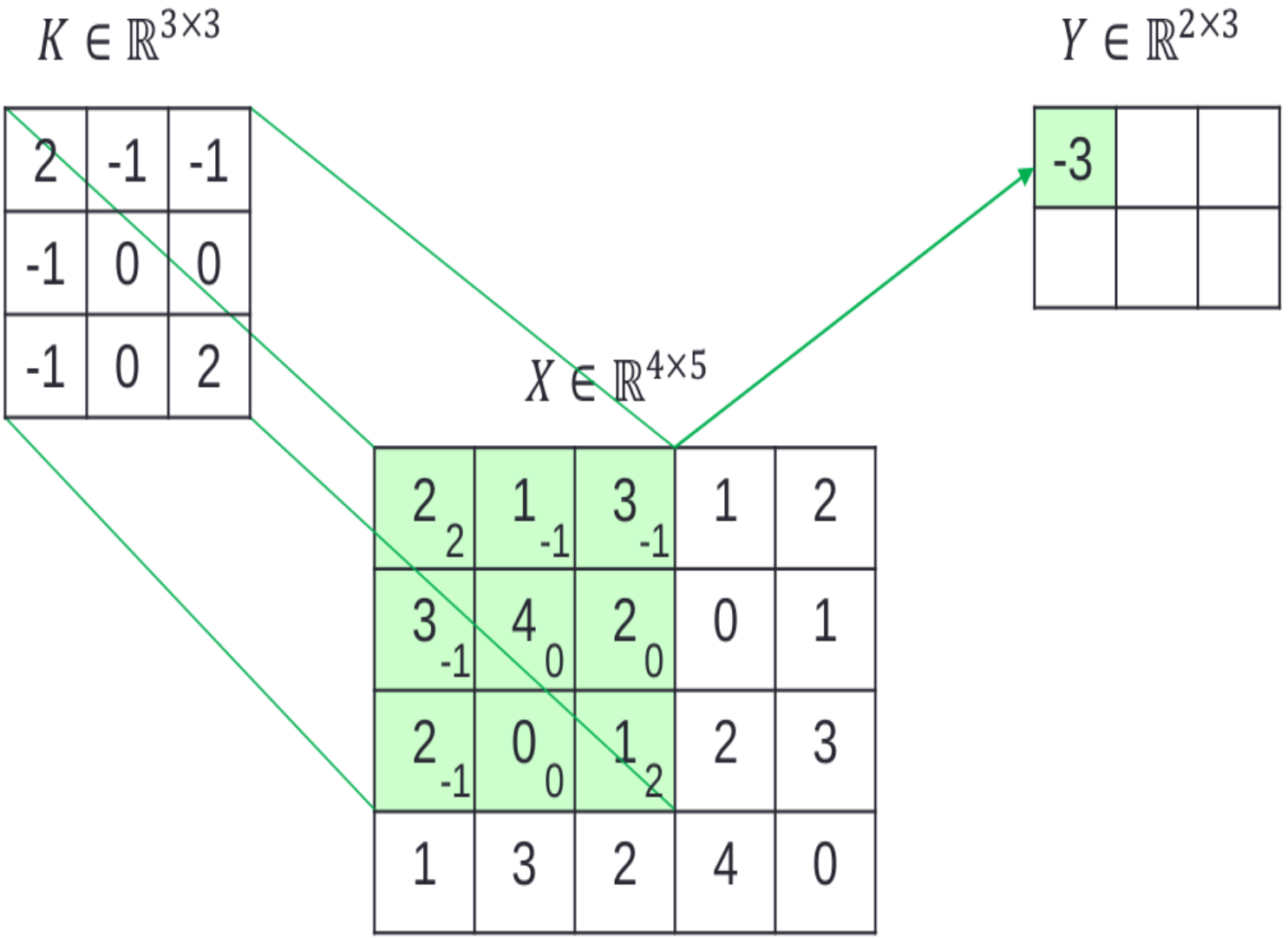


- свертка + активация
- пулинг (снижение размерности)
- полносвязный слой (персептрон)

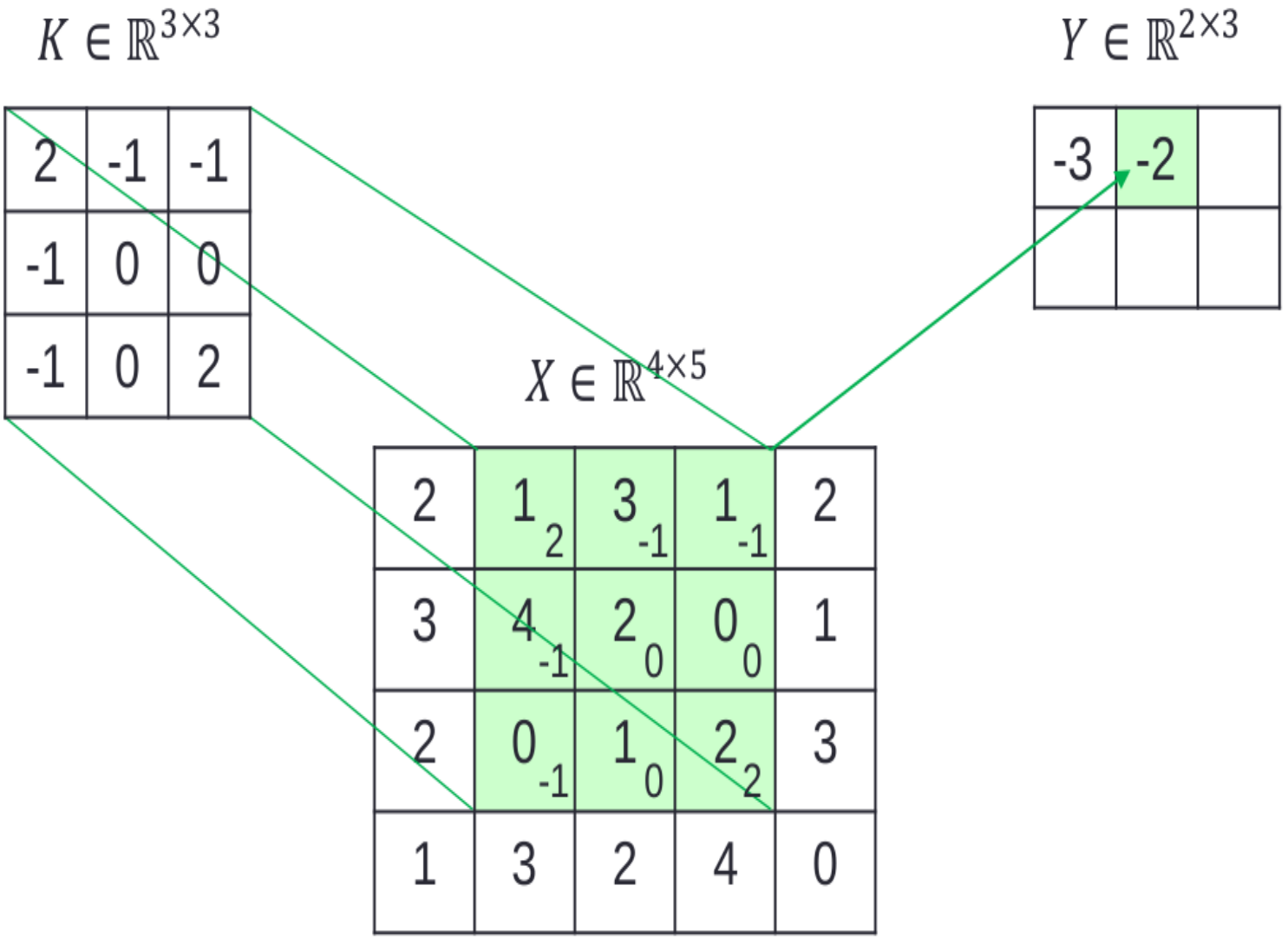
Свертка (convolution)



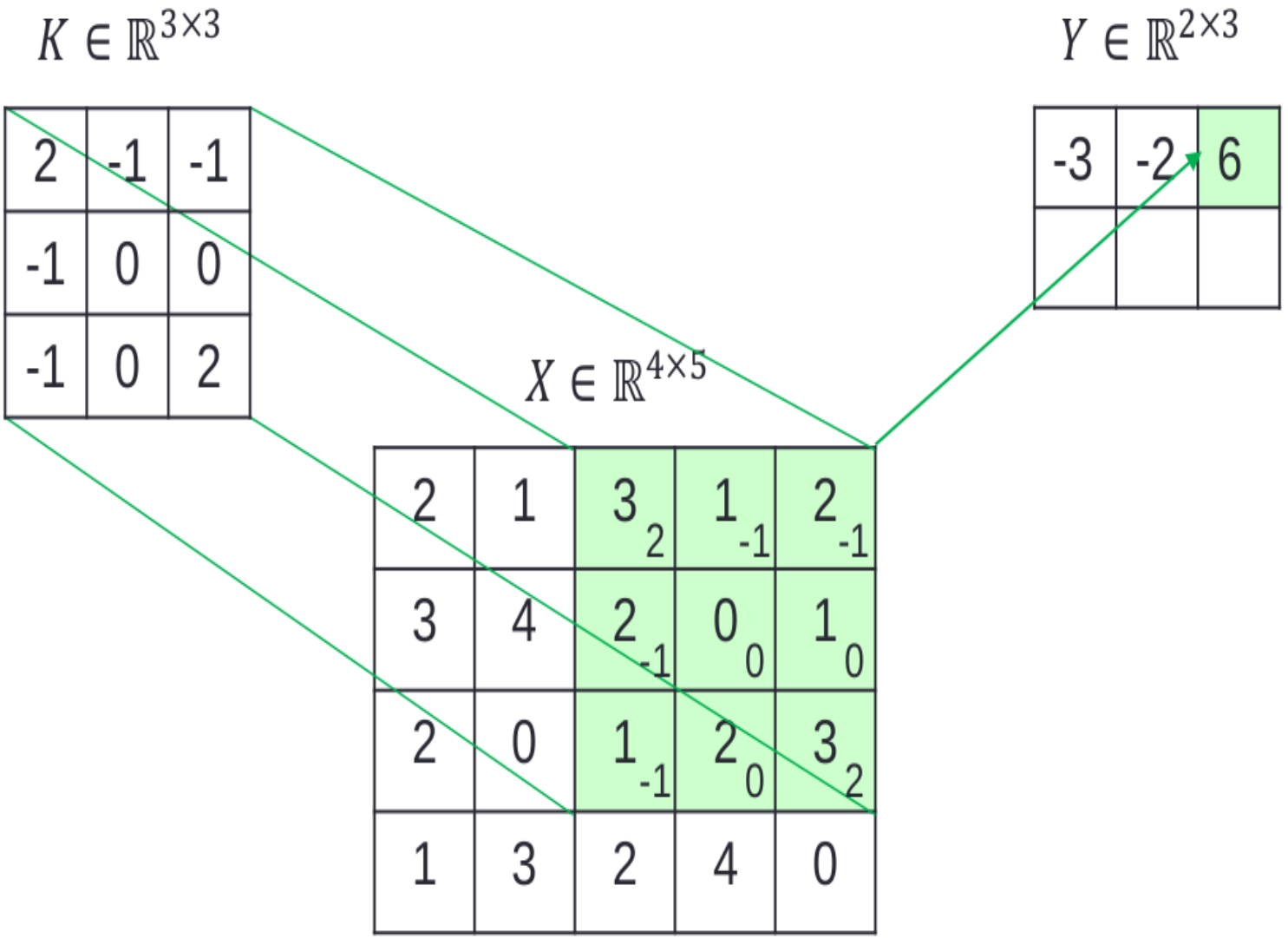
Свертка (convolution)



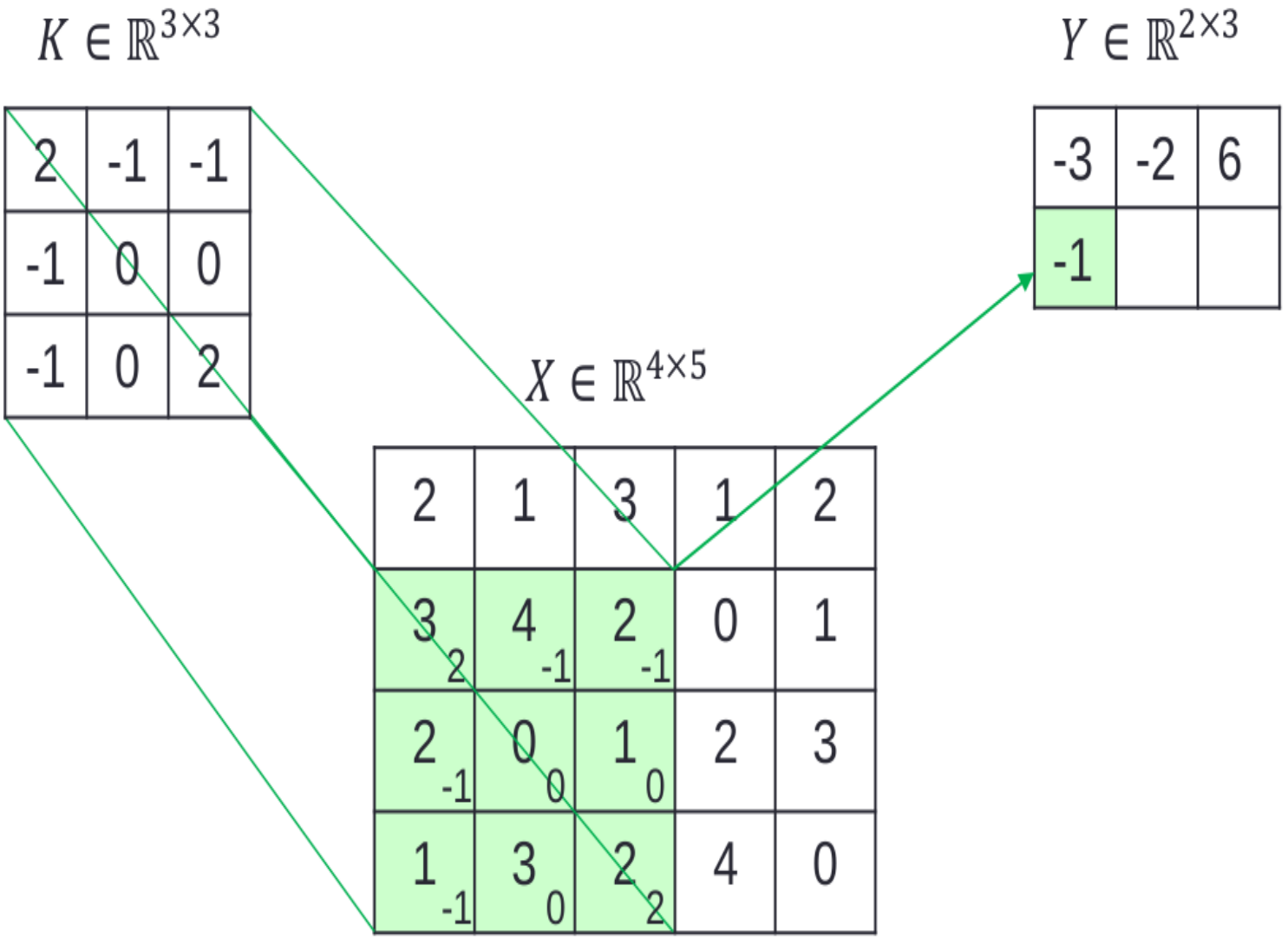
Свертка (convolution)



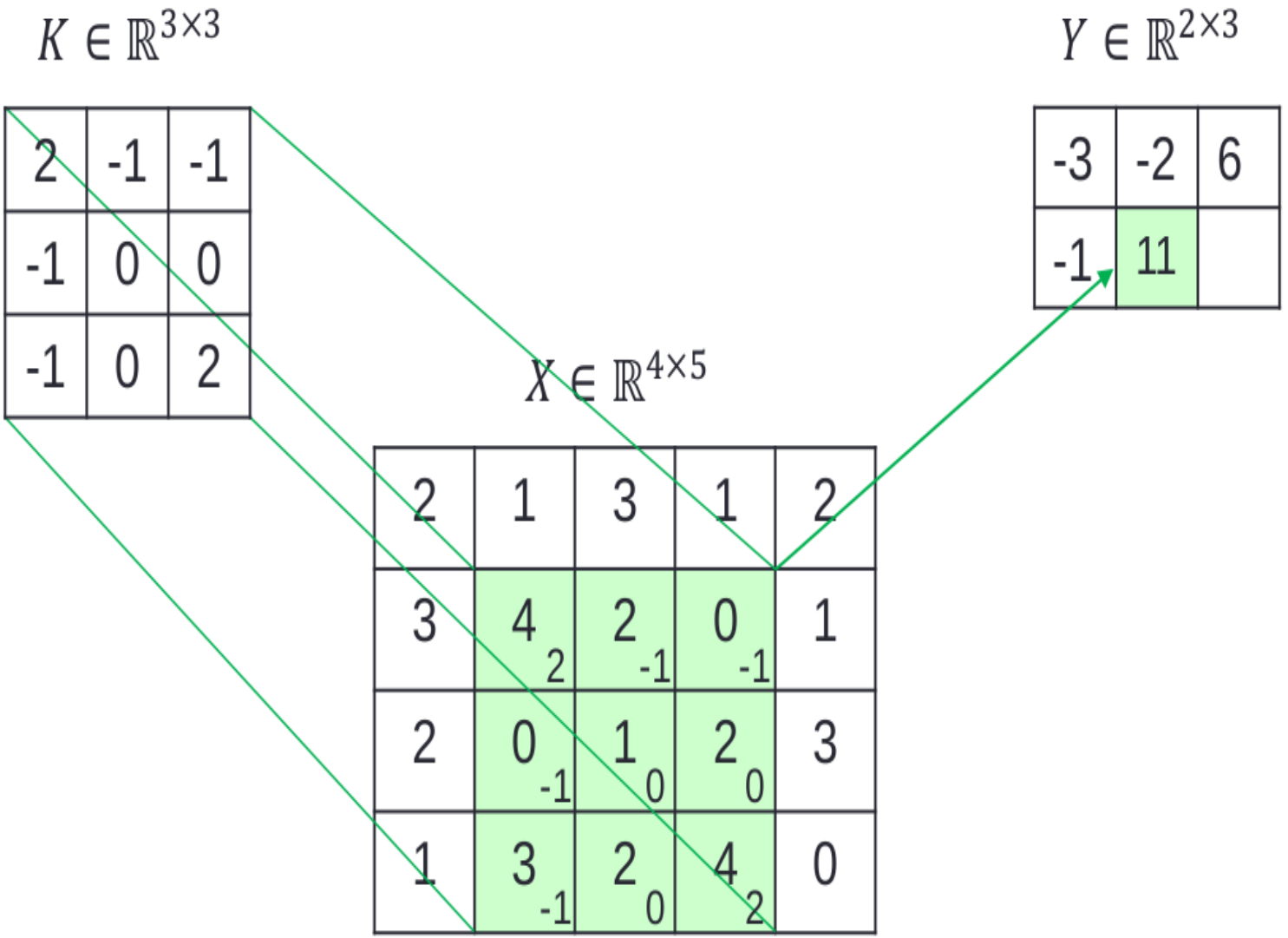
Свертка (convolution)



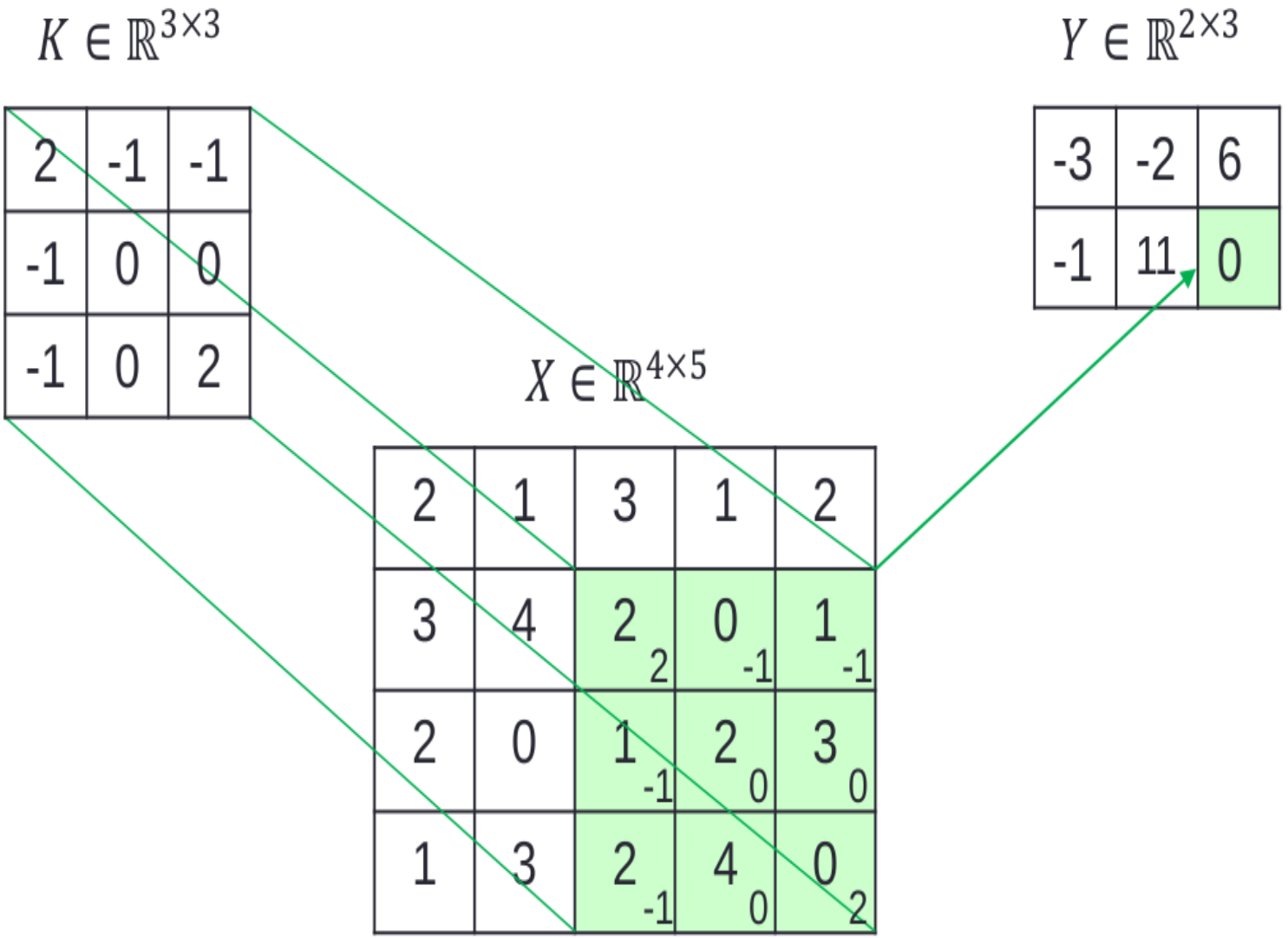
Свертка (convolution)



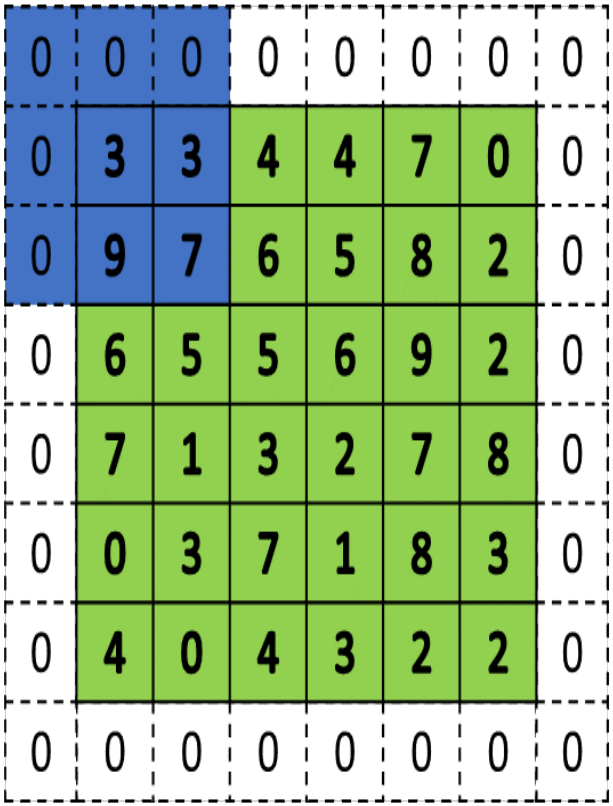
Свертка (convolution)



Свертка (convolution)



Padding (дополнение)



$6 \times 6 \rightarrow 8 \times 8$

$*$

1	0	-1
1	0	-1
1	0	-1

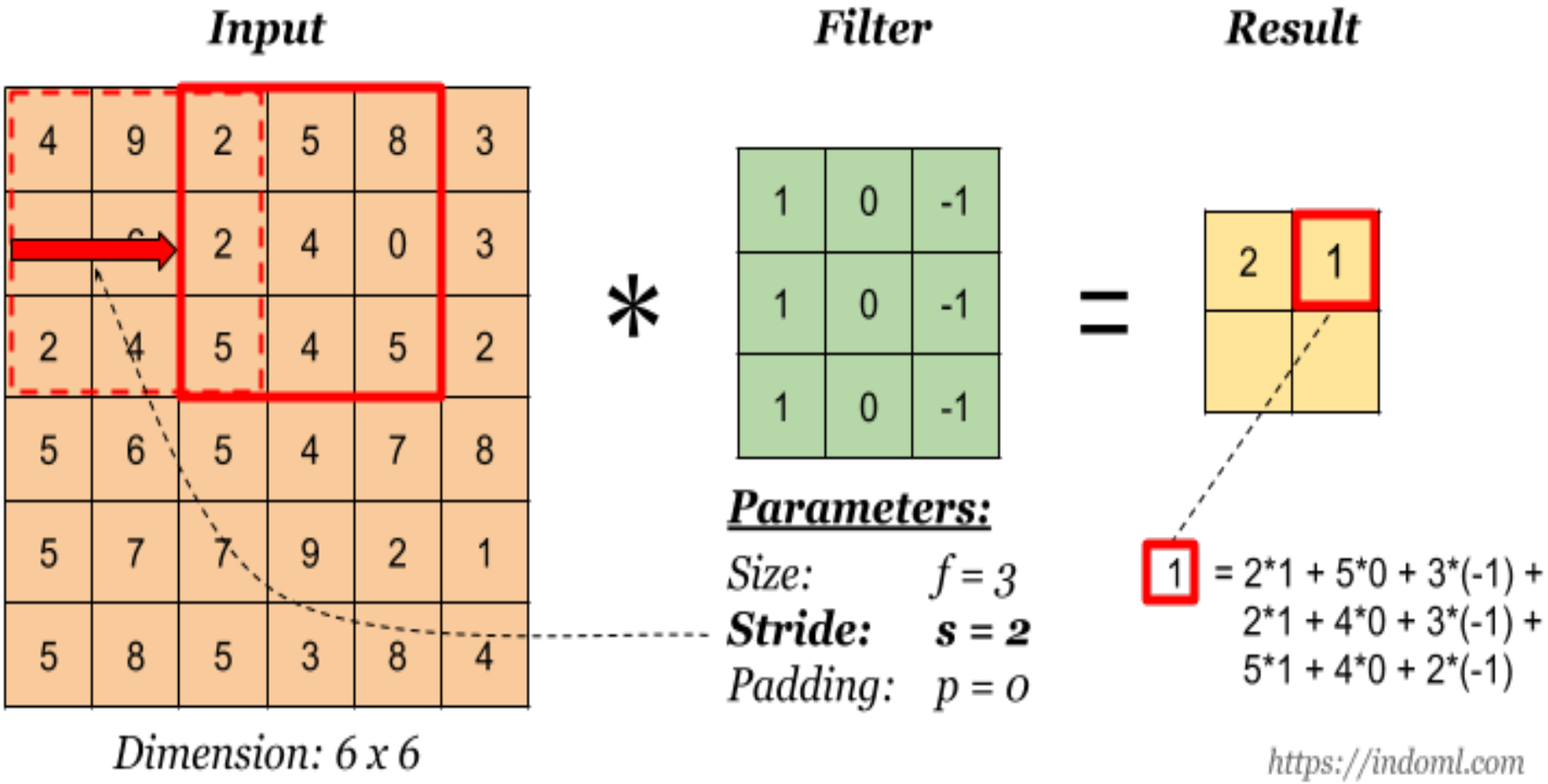
3×3

$=$

-10	-13	1			
-9	3	0			

6×6

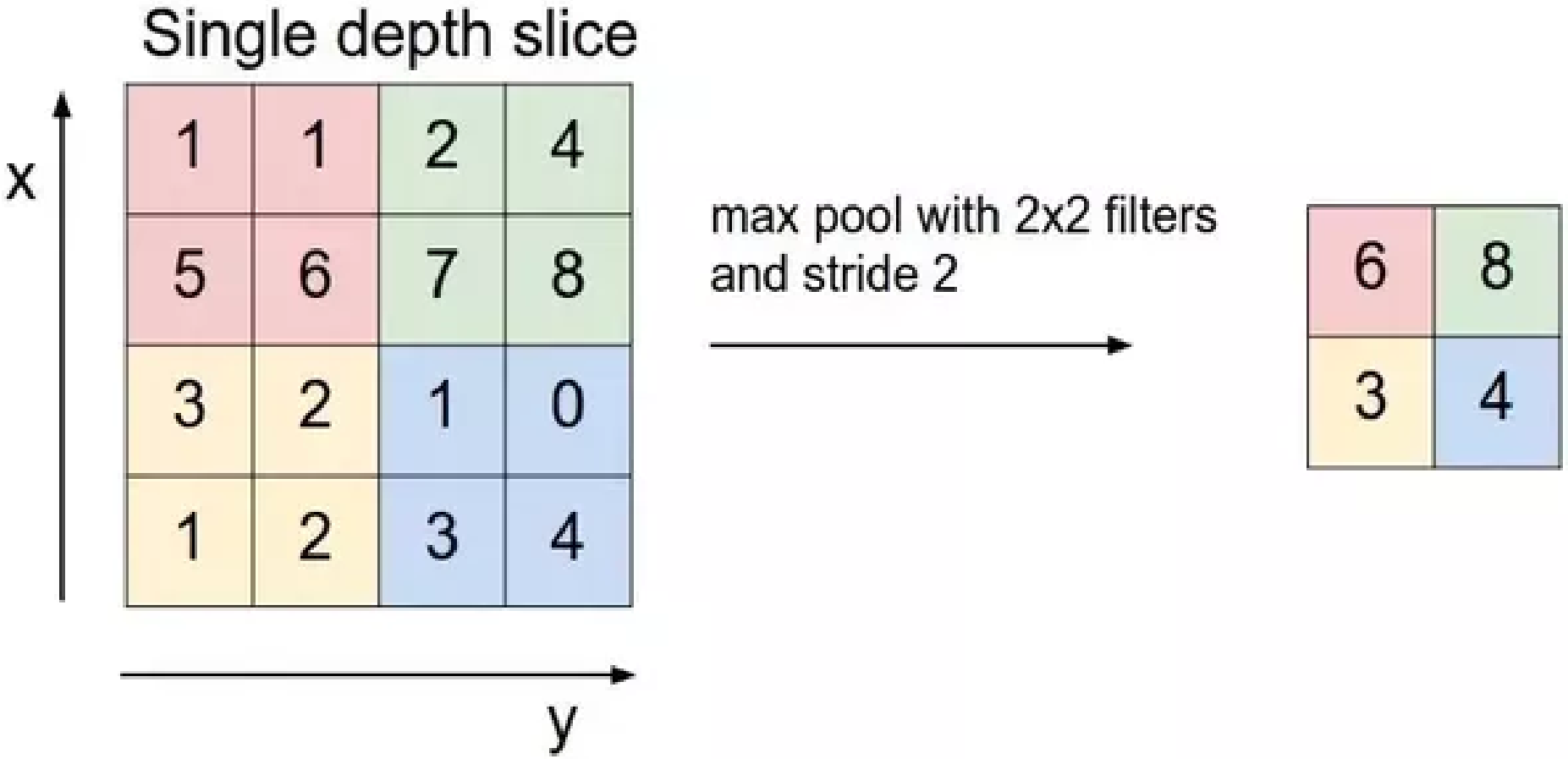
Stride (шаг)


Input
Filter
Result
4 9 2 5 8 3
2 4 0 3
2 4 5 4 5 2
5 6 5 4 7 8
5 7 7 9 2 1
5 8 5 3 8 4
*
1 0 -1
1 0 -1
1 0 -1
=
2 1
Parameters:
Size: f = 3
Stride: s = 2
Padding: p = 0
1 = 2*1 + 5*0 + 3*(-1) +
2*1 + 4*0 + 3*(-1) +
5*1 + 4*0 + 2*(-1)
Dimension: 6 x 6
https://indoml.com

Визуализация признаков

Визуализация сверток в Google LeNet

Pooling


Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

Размерность выхода

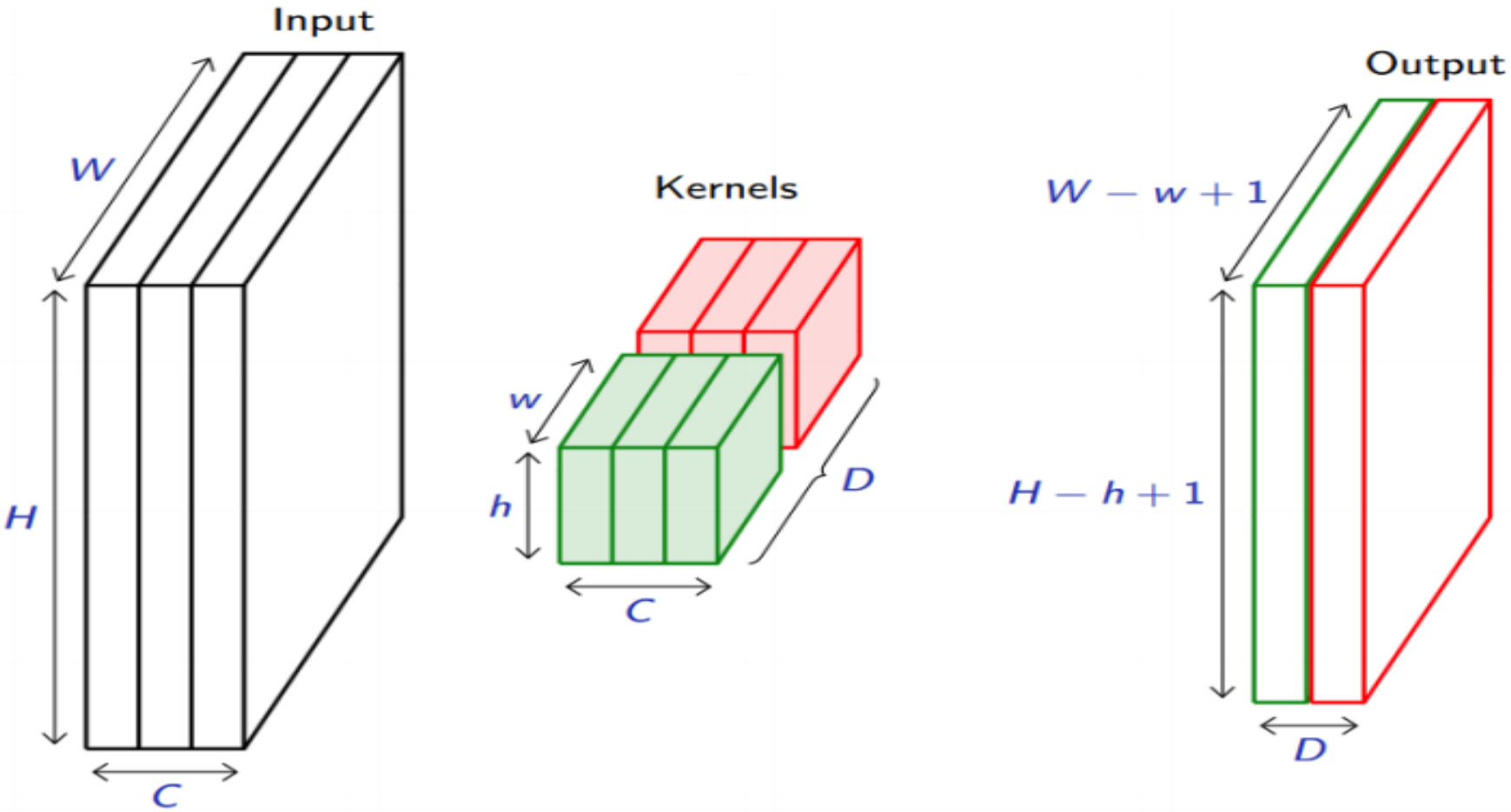
Если к изображению размера NxN применить свертку / пулинг размера FxF с padding p и stride s, то размер выходного изображения равен YxY, где:

$$Y = \lfloor \frac{N + 2 * p - F}{s} \rfloor + 1$$

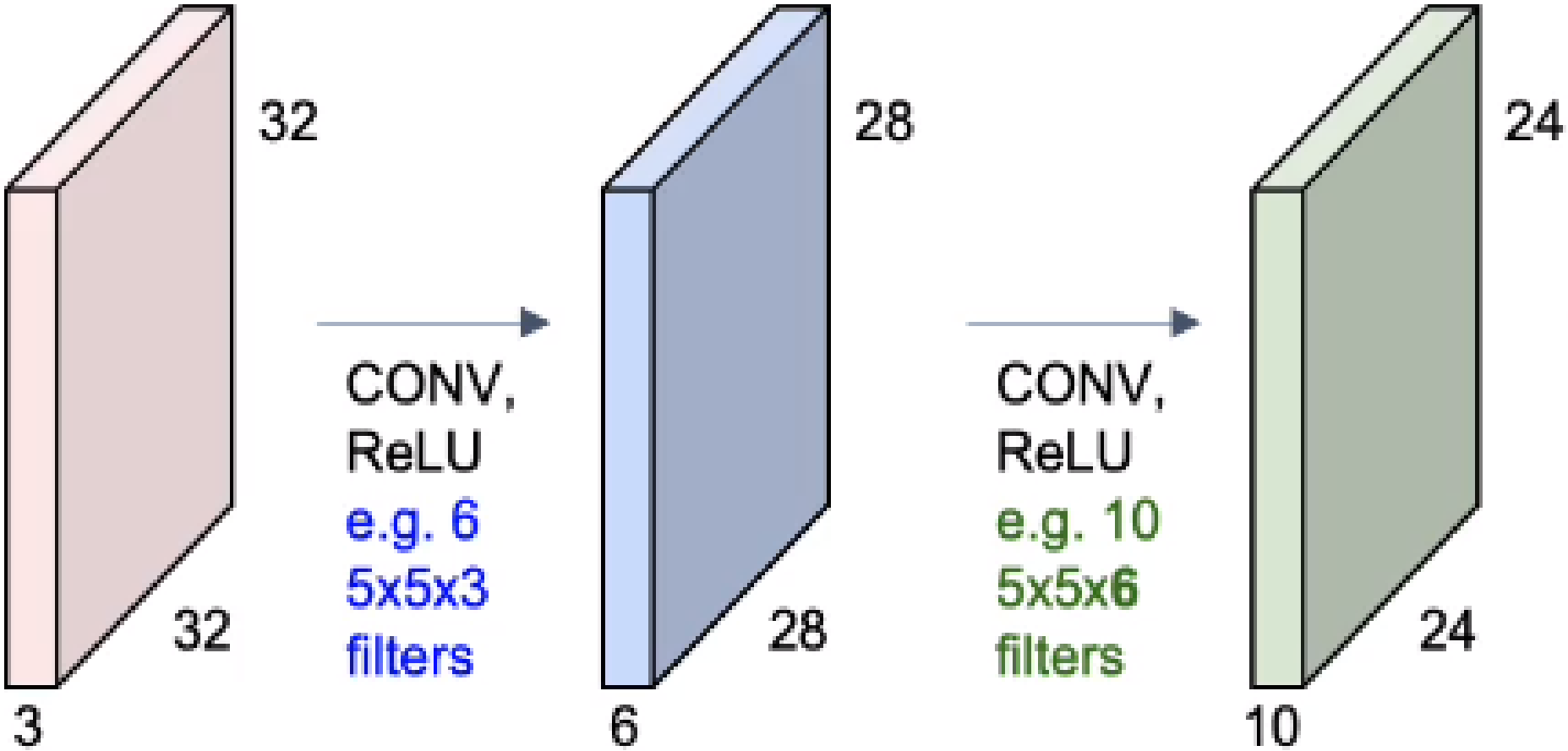
К примеру, если $F = 3, p = 1, s = 1$, то:

$$Y = \lfloor \frac{N + 2 - 3}{1} \rfloor + 1 = N$$

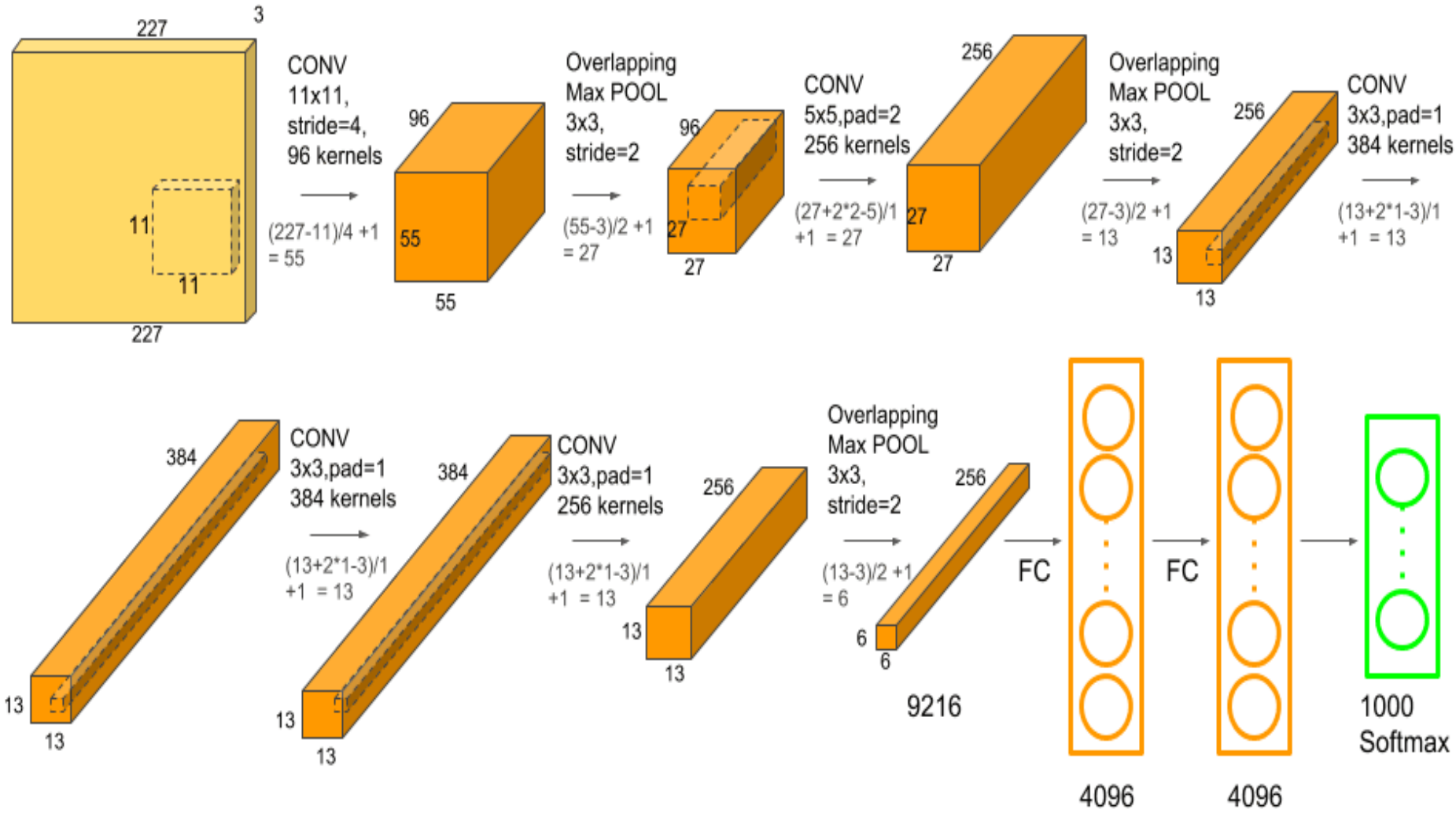
Несколько фильтров



Свертка + активация



Архитектура AlexNet



Что посмотреть, если интересно

- A.Karpathy CS231n Winter 2016 lectures
- C.Manning CS224n Winter 2019 lectures

Вопросы?