

# Искусственные нейронные сети

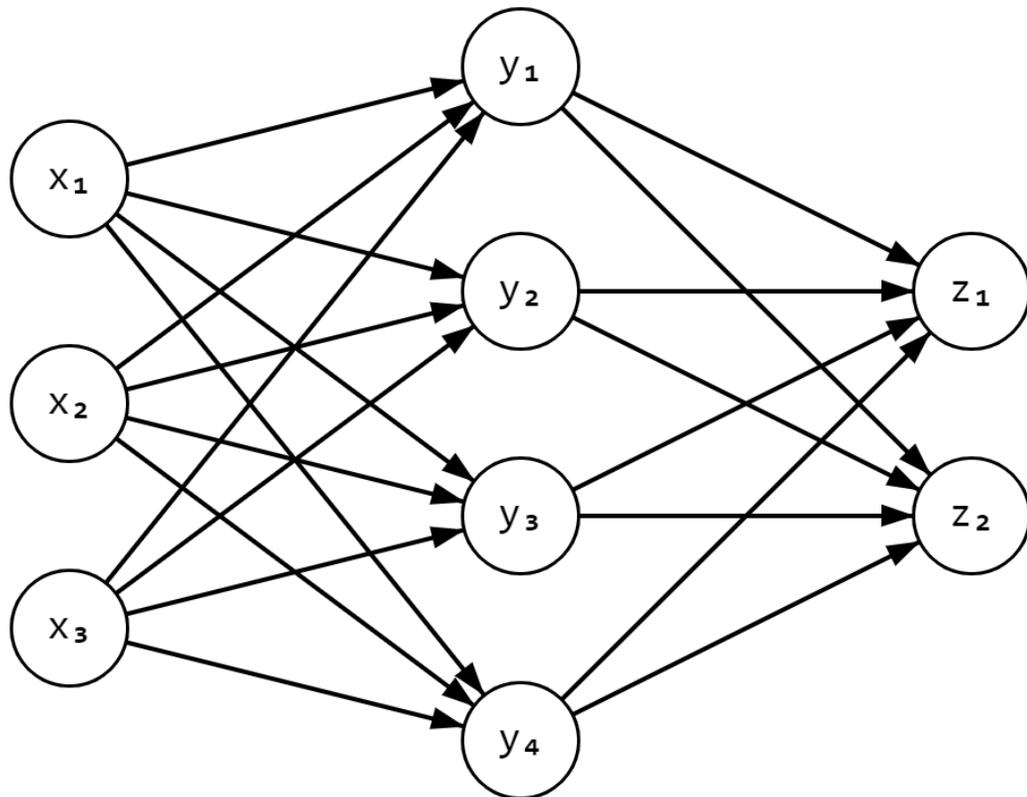
---

Перминов Андрей

29 сентября 2021



**Нейронные сети** – алгоритмы машинного обучения, активно используемые для решения задач автоматической обработки различных данных (текстов, изображений, аудио, видео, ...)



Пусть:

$X$  – множество объектов

$Y$  – множество ответов

$\check{y}: X \rightarrow Y$  – неизвестная зависимость

Дано:

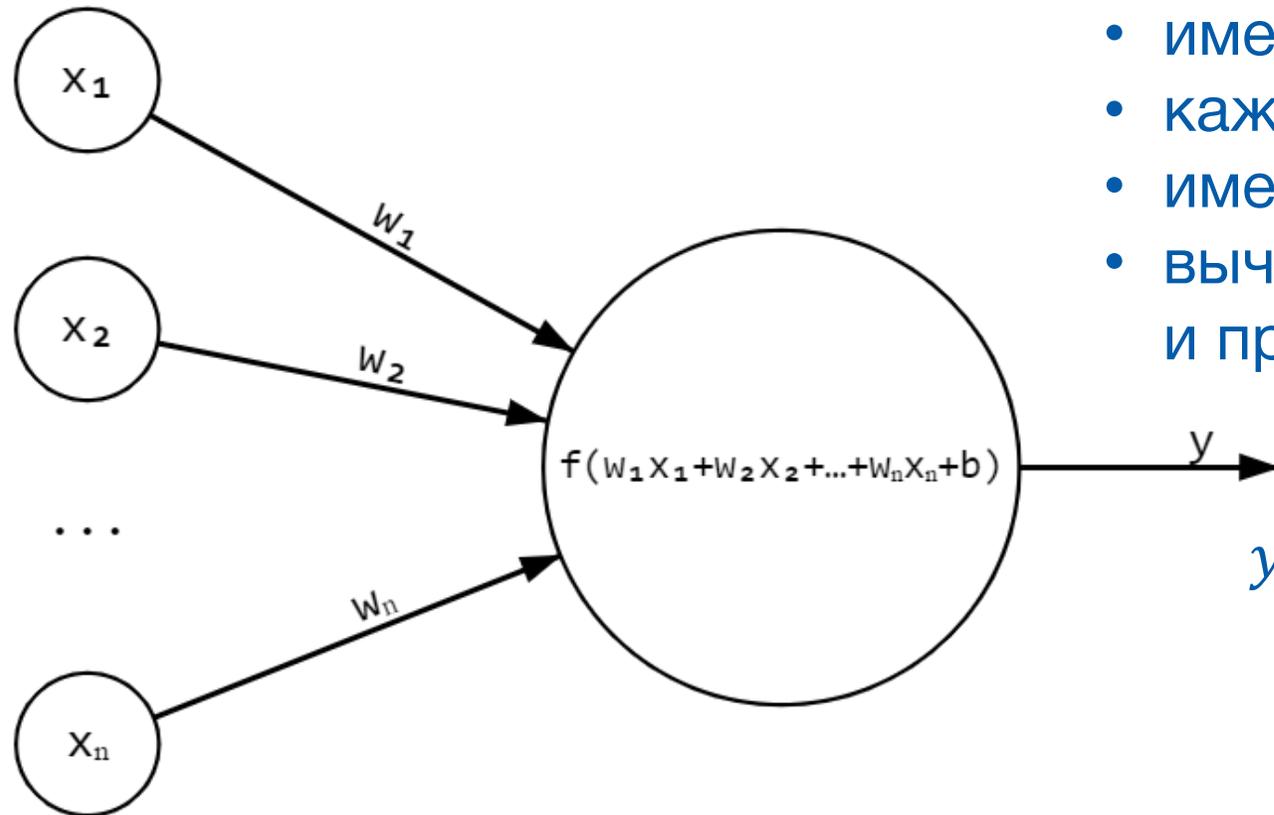
$\{x_1, \dots, x_n\} \subset X$  – обучающая выборка

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

Найти:

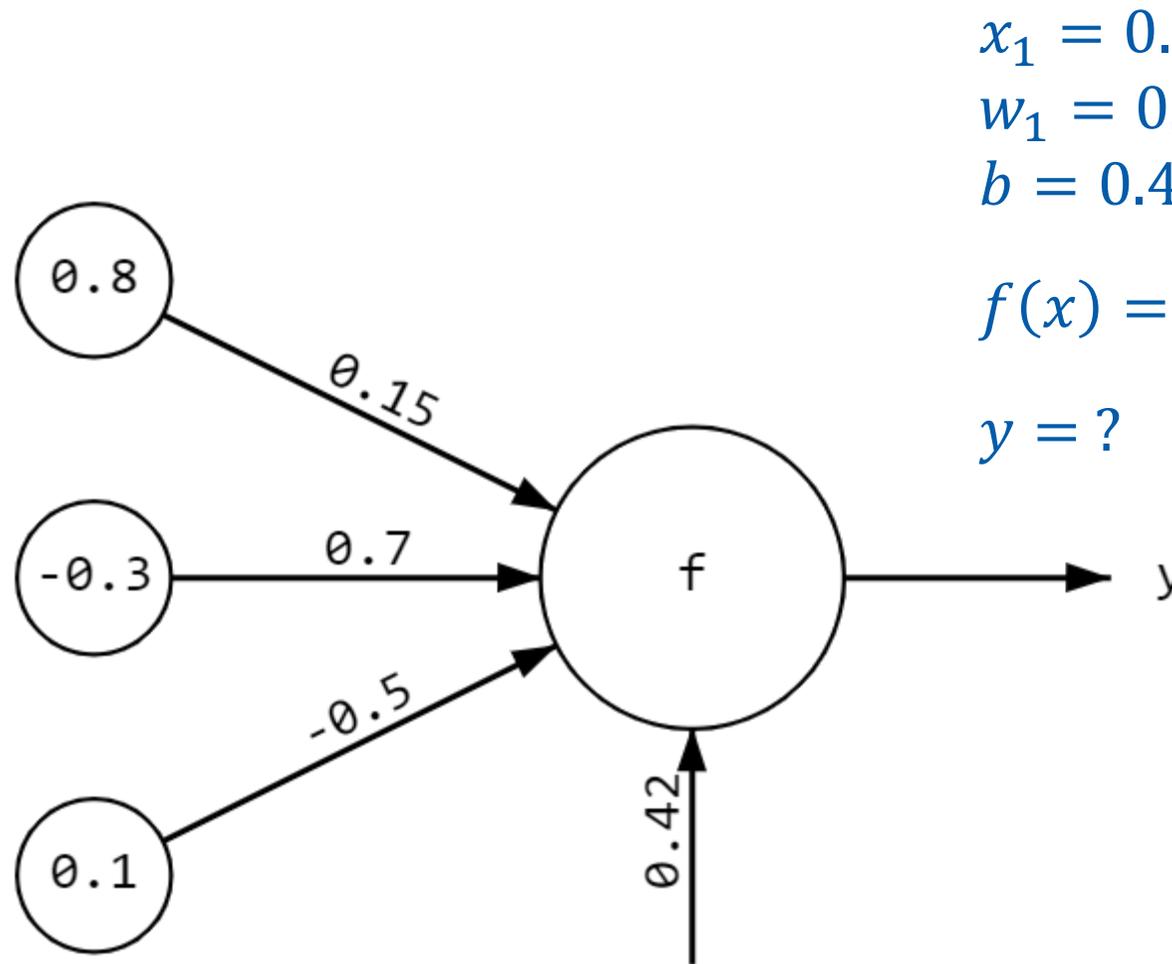
$y: X \rightarrow Y$  – решающую функцию, приближающую  $\check{y}$  на всем множестве  $X$





- имеет  $n$  входов ( $x_1, \dots, x_n$ ) и один выход ( $y$ )
- каждый вход имеет вес ( $w_1, \dots, w_n$ )
- имеется дополнительный вес смещения ( $b$ )
- вычисляет линейную комбинацию входов и применяет к ней функцию активации:

$$y = f\left(\sum_{i=1}^n w_i \cdot x_i + b\right) = f(w^T \cdot x + b)$$



$x_1 = 0.8, x_2 = -0.3, x_3 = 0.1$   
 $w_1 = 0.15, w_2 = 0.7, w_3 = -0.5$   
 $b = 0.42$

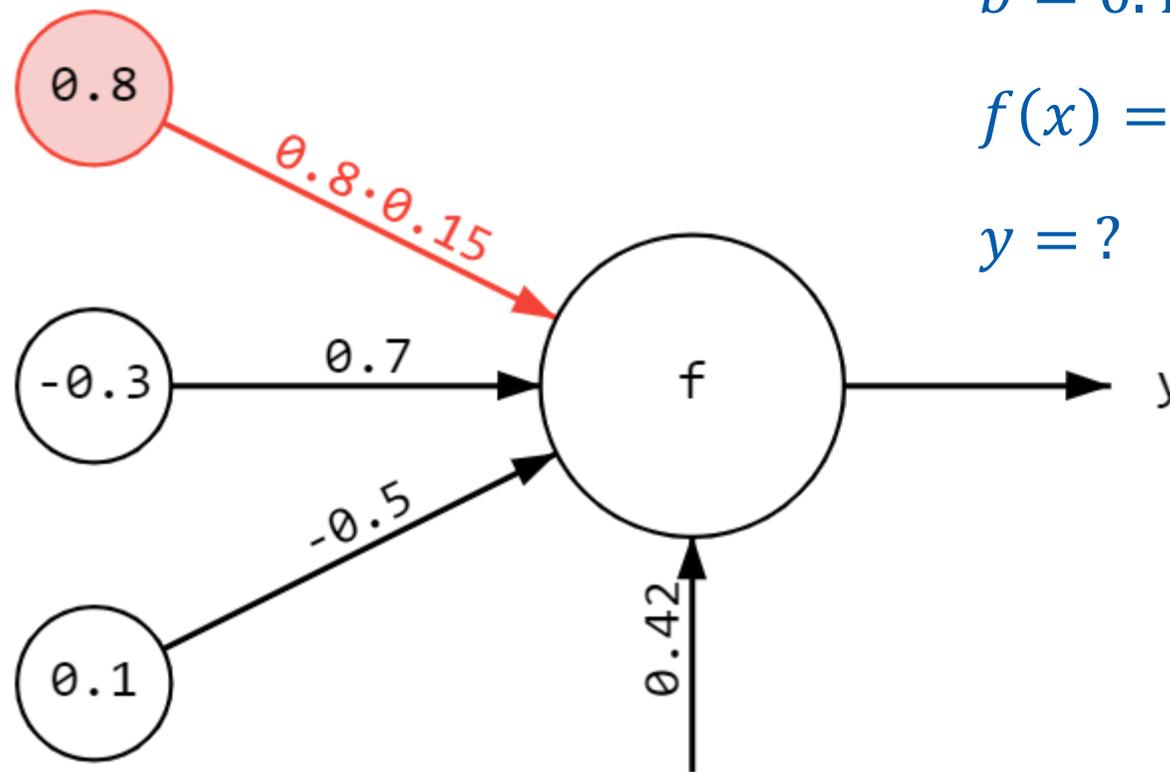
$$f(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

$y = ?$

$$x_1 = 0.8, x_2 = -0.3, x_3 = 0.1$$
$$w_1 = 0.15, w_2 = 0.7, w_3 = -0.5$$
$$b = 0.42$$

$$f(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

$$y = ?$$

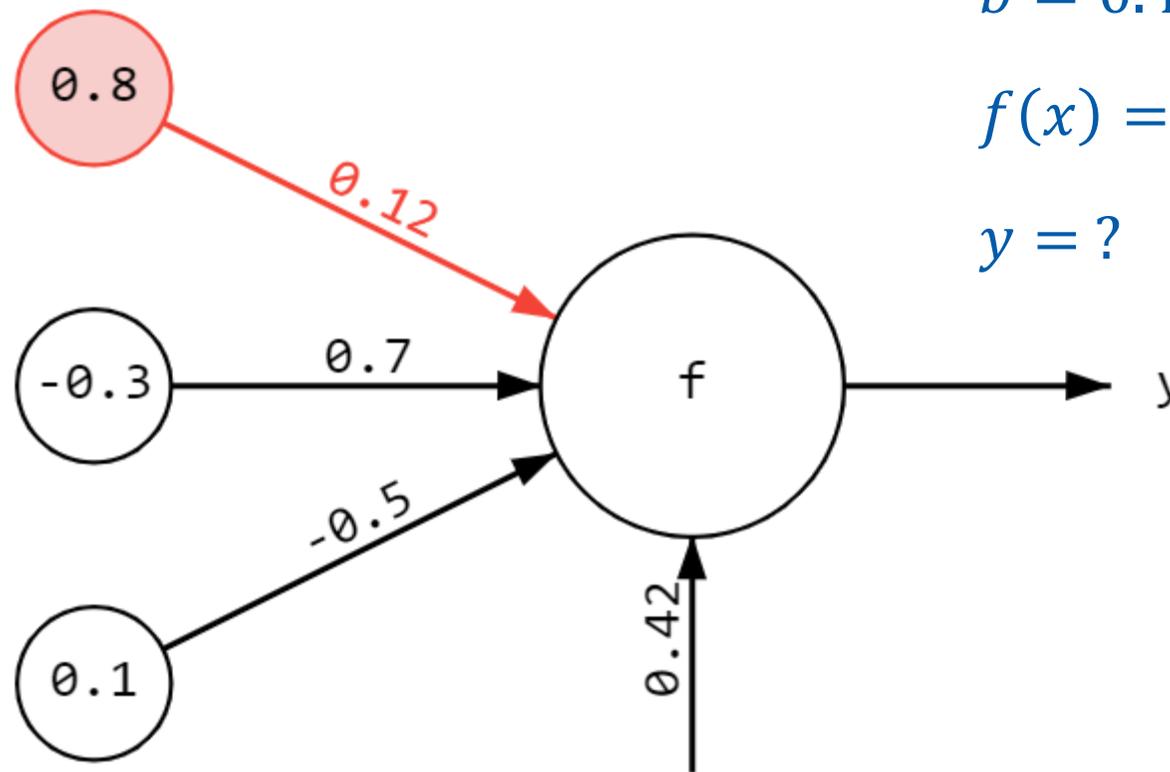


$$y = f(0.8 \cdot 0.15 +$$

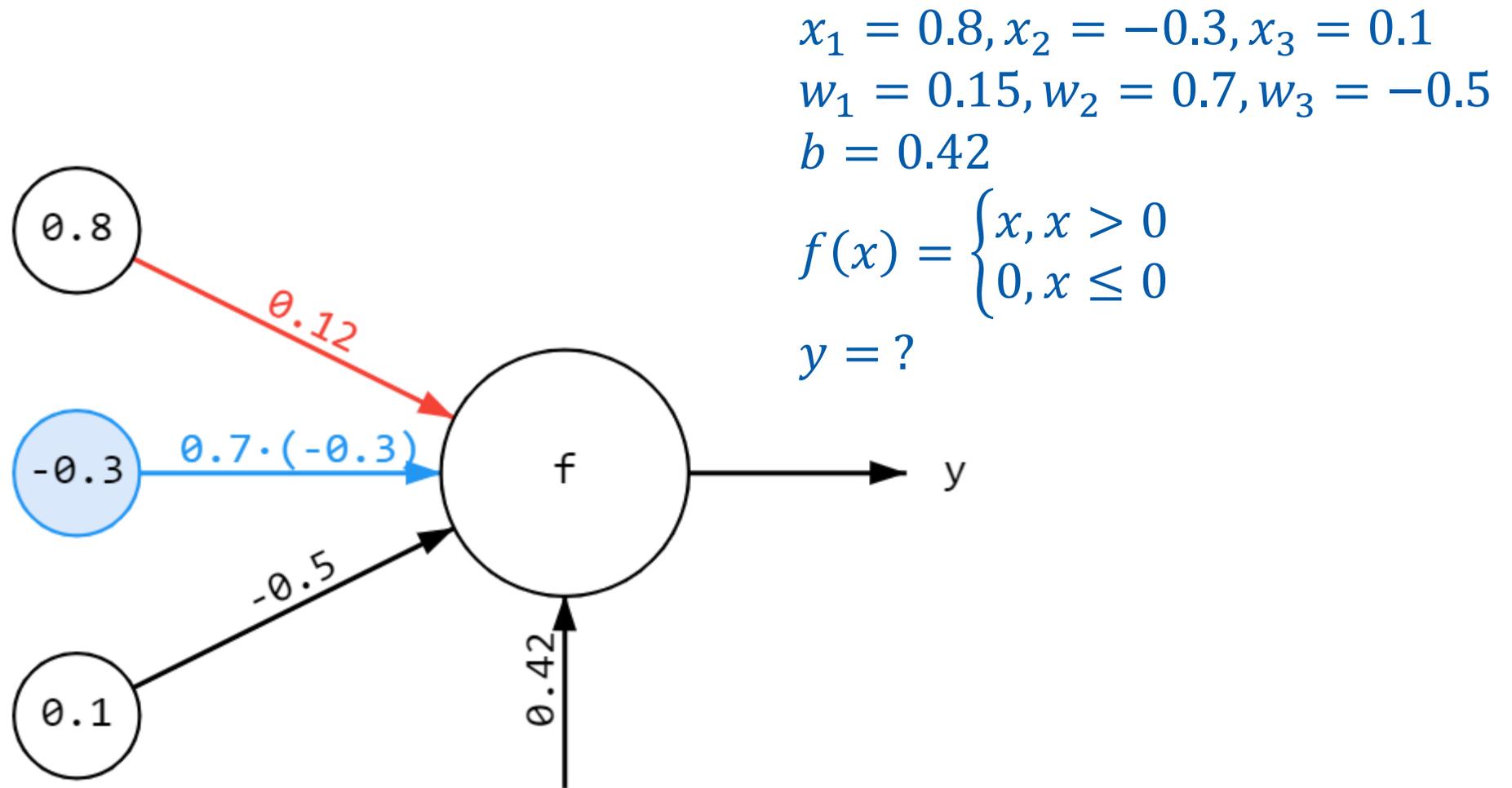
$$x_1 = 0.8, x_2 = -0.3, x_3 = 0.1$$
$$w_1 = 0.15, w_2 = 0.7, w_3 = -0.5$$
$$b = 0.42$$

$$f(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

$$y = ?$$



$$y = f(0.12 +$$



$$x_1 = 0.8, x_2 = -0.3, x_3 = 0.1$$
$$w_1 = 0.15, w_2 = 0.7, w_3 = -0.5$$
$$b = 0.42$$

$$f(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

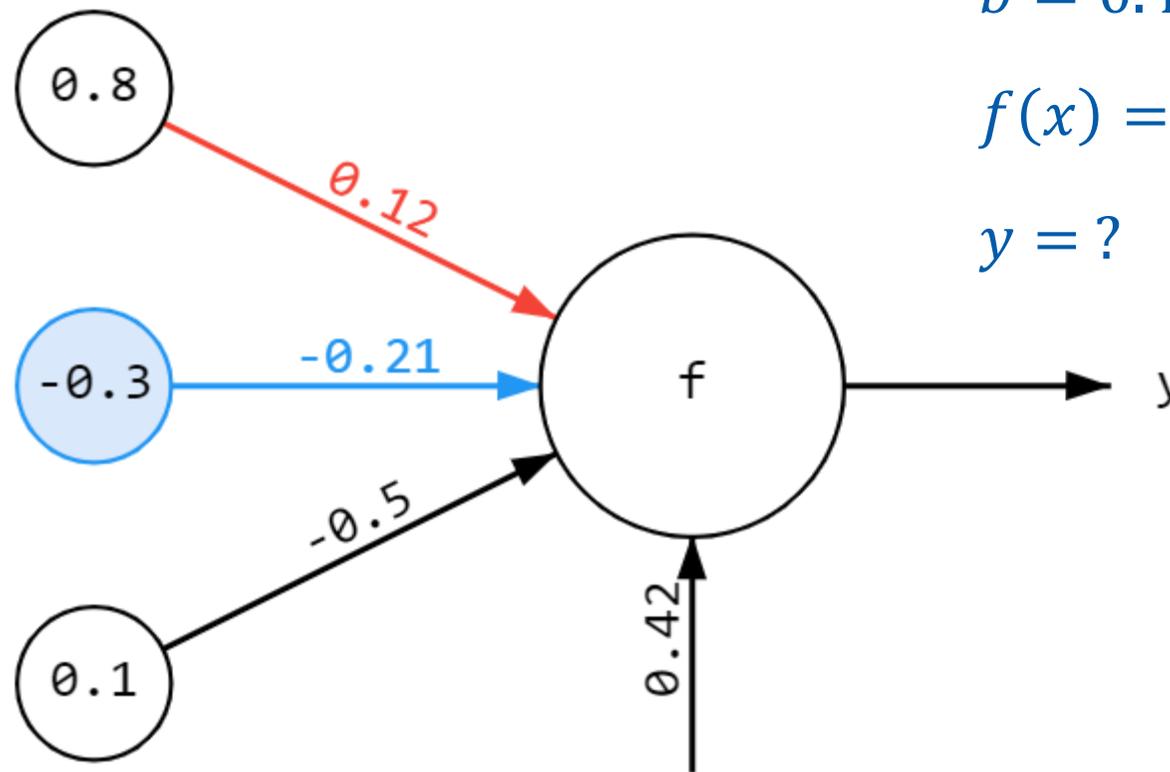
$$y = ?$$

$$y = f(0.12 + 0.7 \cdot (-0.3) +$$

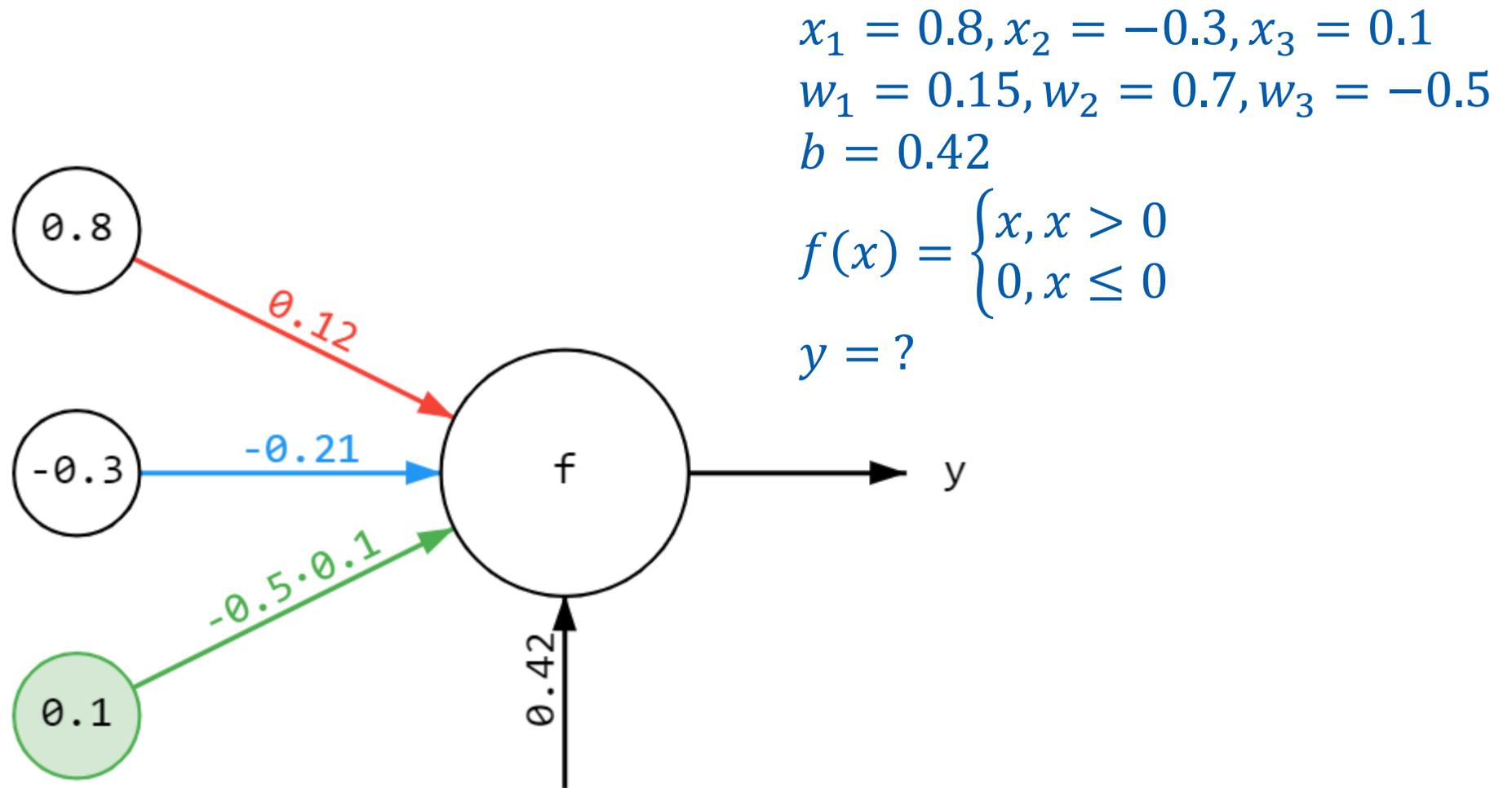
$$x_1 = 0.8, x_2 = -0.3, x_3 = 0.1$$
$$w_1 = 0.15, w_2 = 0.7, w_3 = -0.5$$
$$b = 0.42$$

$$f(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

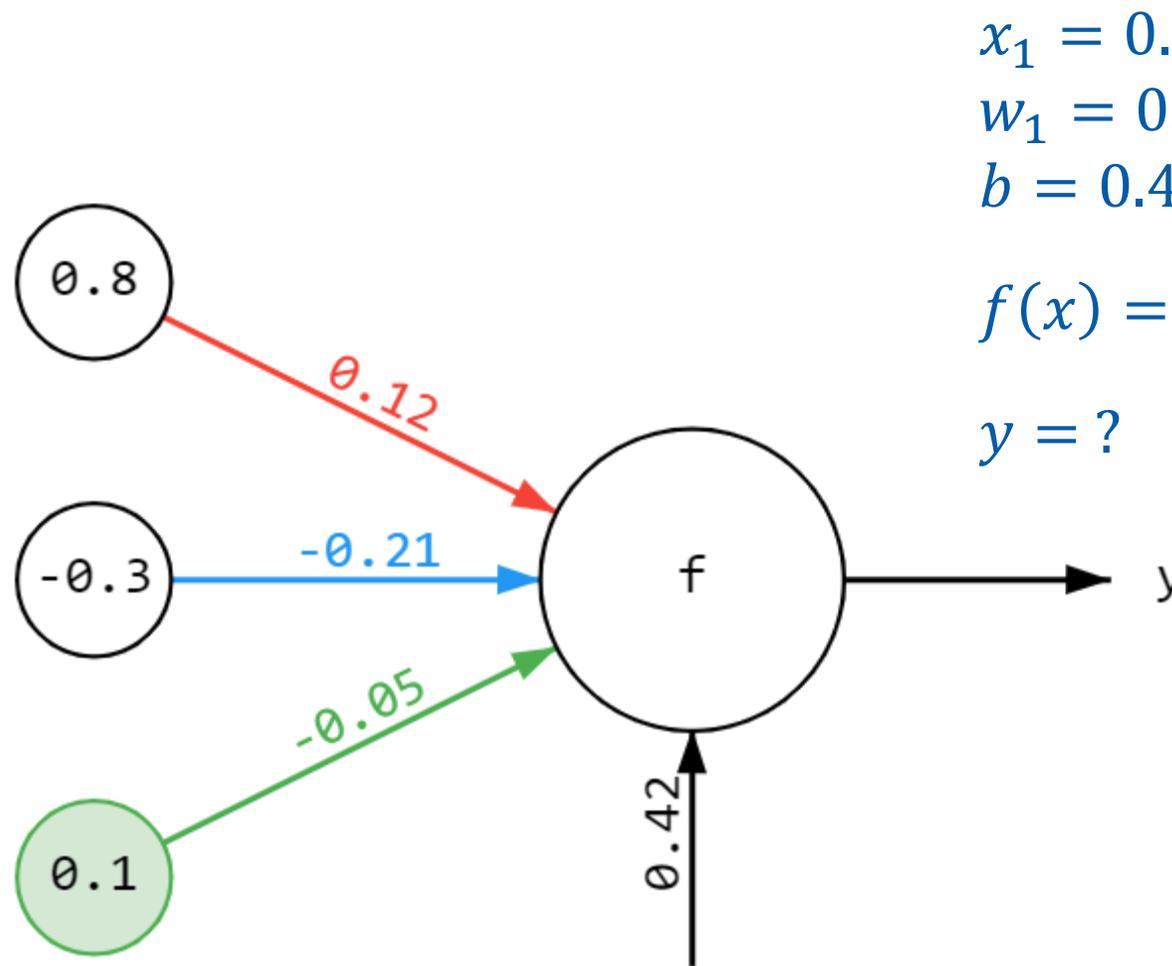
$$y = ?$$



$$y = f(0.12 - 0.21 +$$



$$y = f(0.12 - 0.21 + (-0.5) \cdot 0.1 +$$

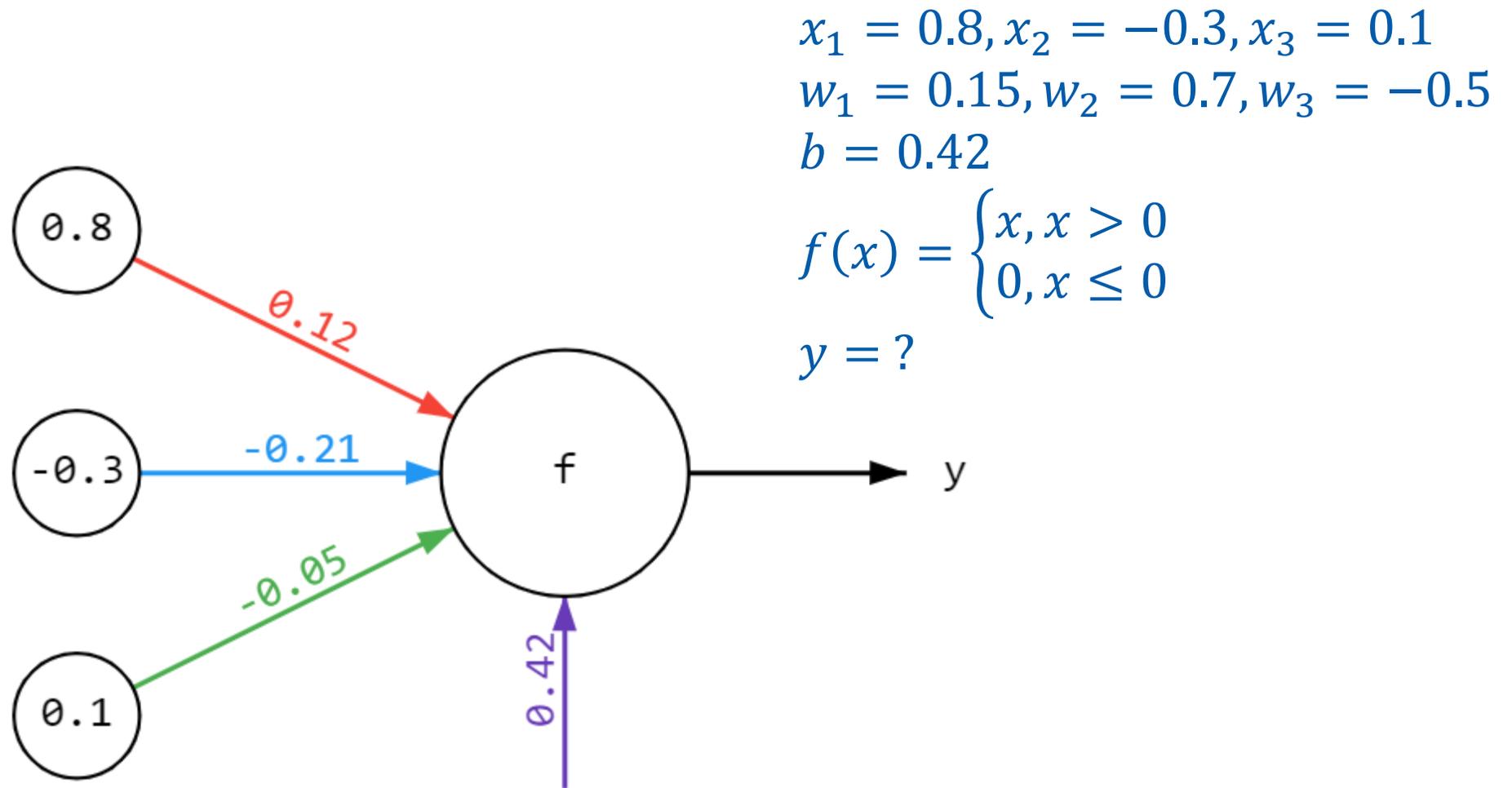


$$x_1 = 0.8, x_2 = -0.3, x_3 = 0.1$$
$$w_1 = 0.15, w_2 = 0.7, w_3 = -0.5$$
$$b = 0.42$$

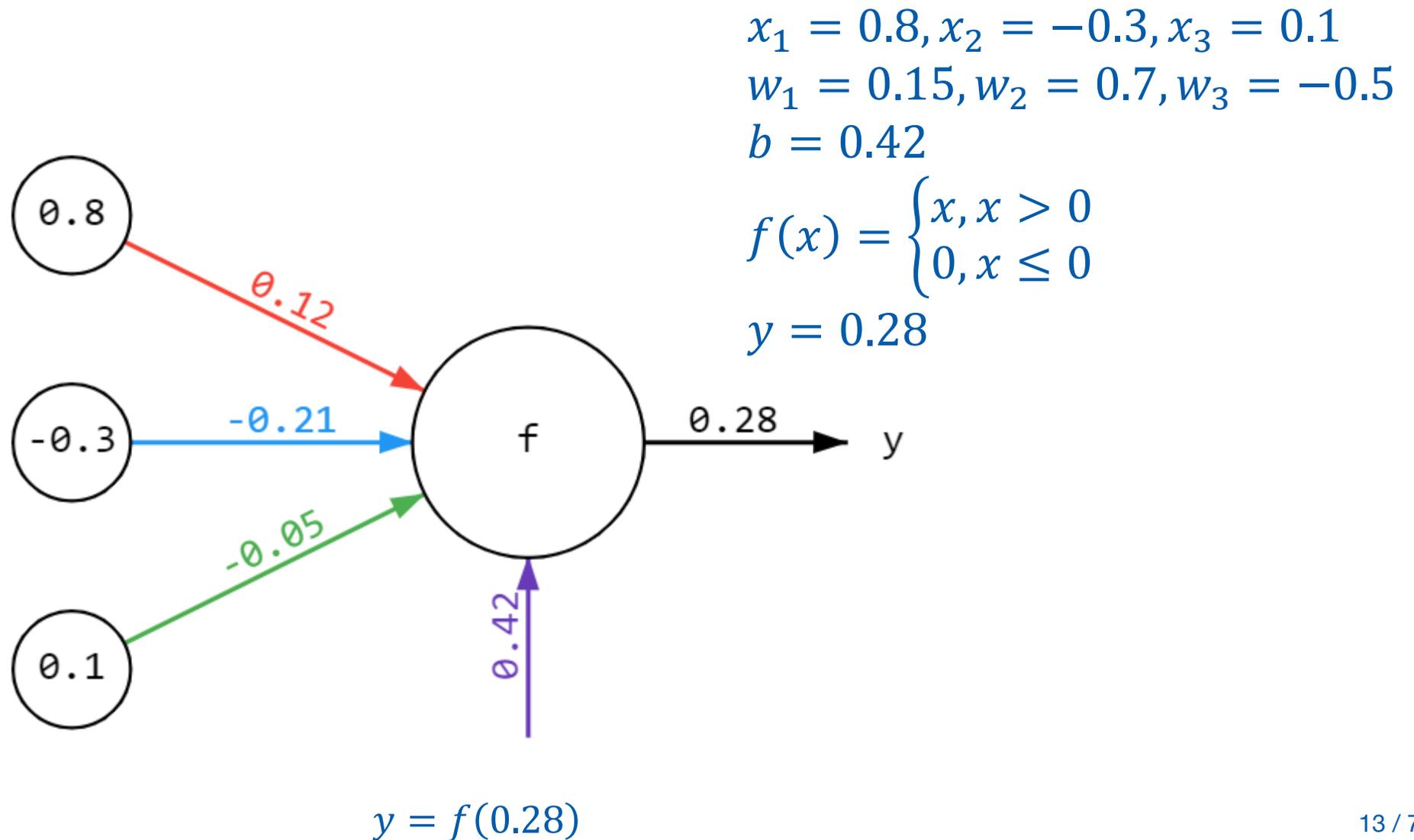
$$f(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

$$y = ?$$

$$y = f(0.12 - 0.21 - 0.05 +$$



$$y = f(0.12 - 0.21 - 0.05 + 0.42)$$

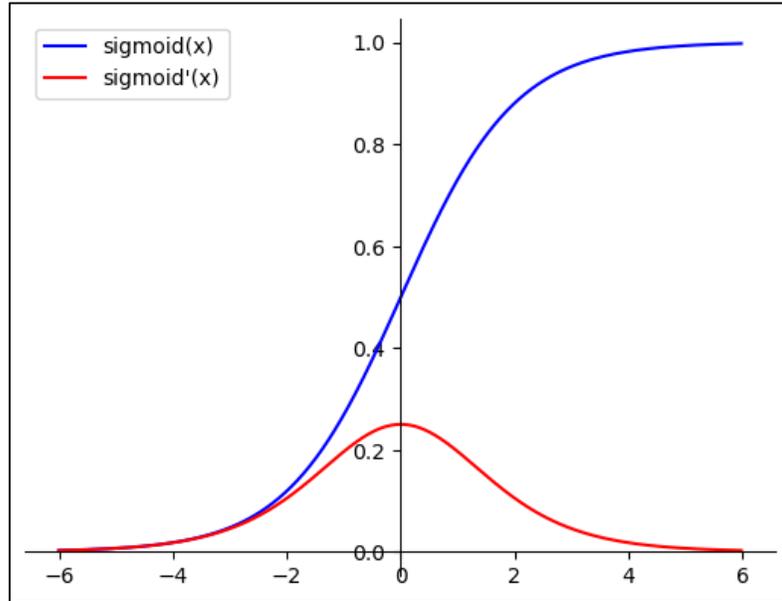


Функция активации – это некоторая дифференцируемая функция (обычно нелинейная), применяемая к линейной комбинации входных значений.

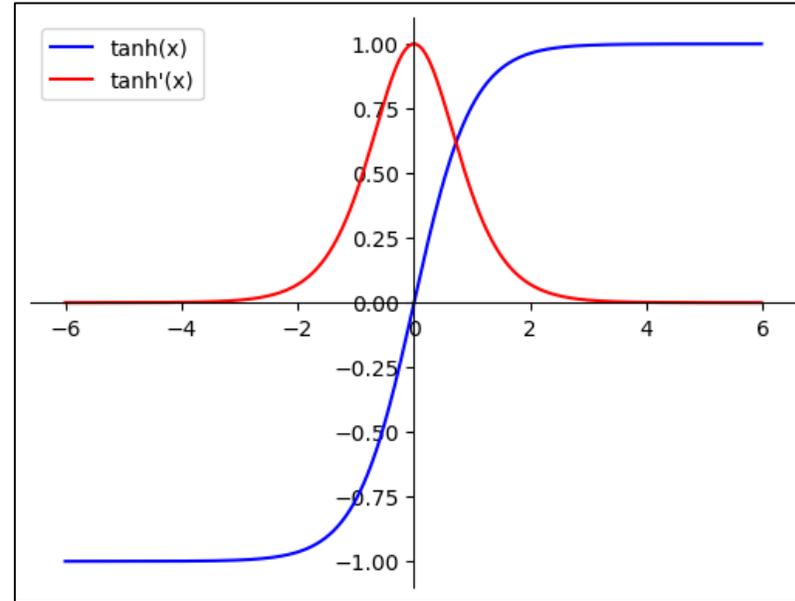
Наиболее распространённые функции активации:

| Название                       | Функция  | Производная   | Область значений |
|--------------------------------|--|---|------------------|
| Сигмоидальная (sigmoid)        | $\frac{1}{1 + e^{-x}}$   | $f(x)(1 - f(x))$  | $(0, 1)$         |
| Гиперболический тангенс (tanh) | $\tanh(x)$   | $1 - f^2(x)$  | $(-1, 1)$        |
| Выпрямитель (ReLU)             | $f(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases}$ | $f'(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases}$ | $[0, +\infty)$   |

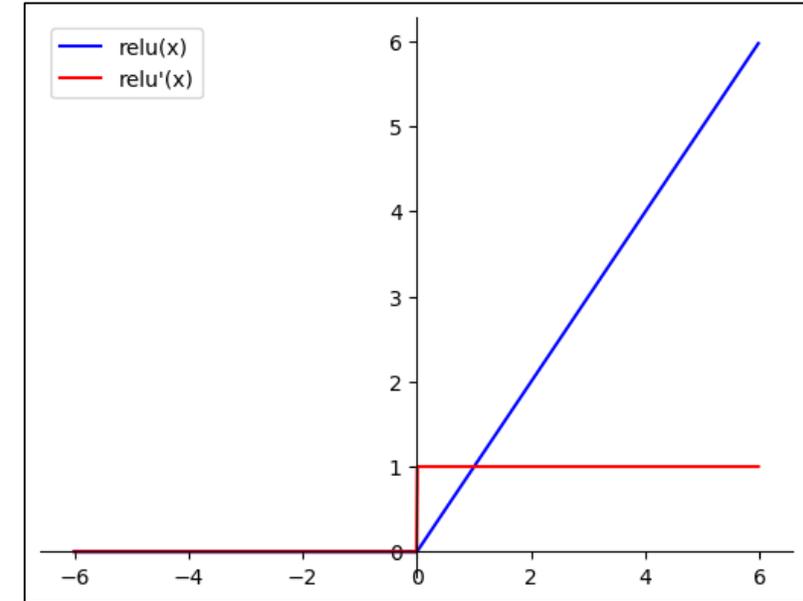
Вопрос: почему функции активации обычно нелинейные? Чем плоха линейная функция?



*sigmoid*



*tanh*



*ReLU*

Дано:

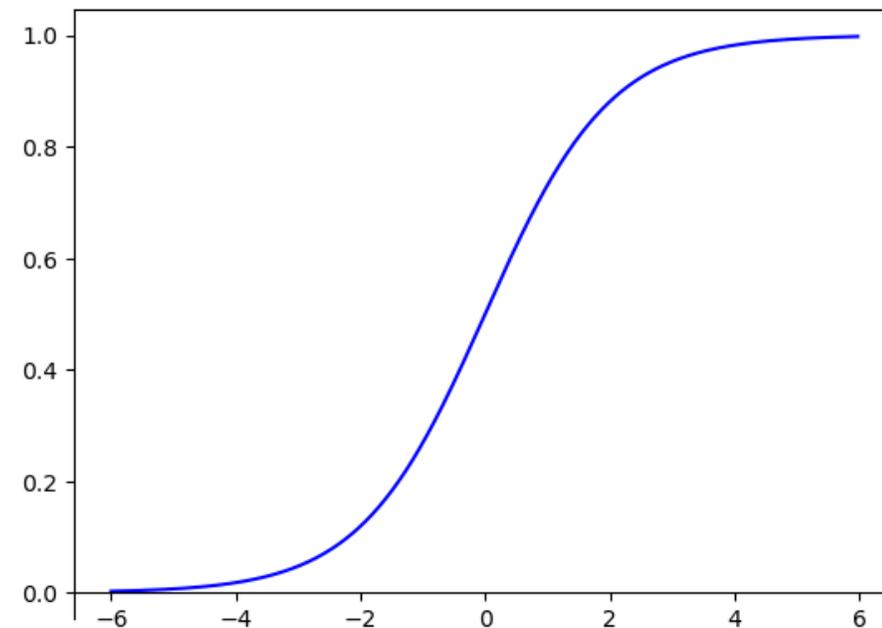
$\{x_1, \dots, x_n\} \subset \mathbb{R}^d$  – обучающая выборка  
 $\tilde{y}_i = \tilde{y}(x_i) \in \{0,1\}$  – ответы на обучающей выборке

Решение:

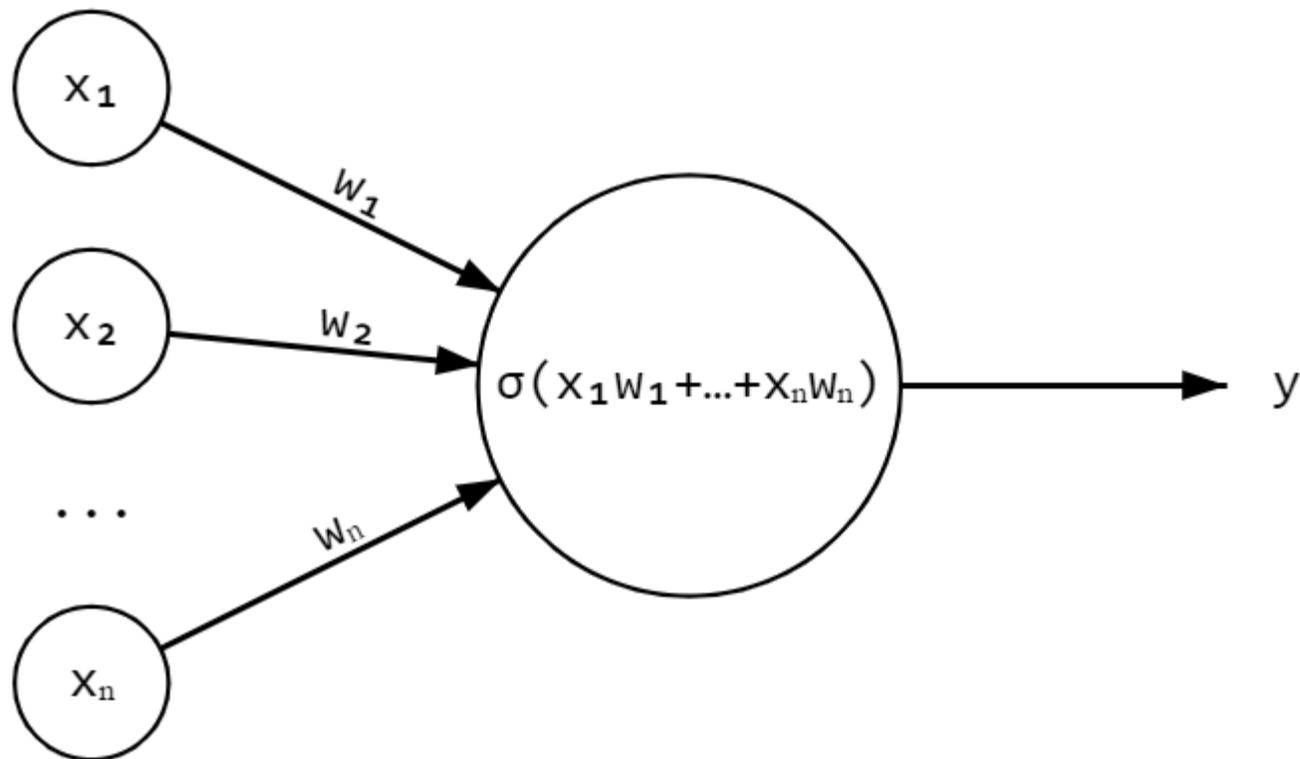
$$y = \sigma(w^T \cdot x) = \frac{1}{1 + e^{-w^T \cdot x}} \in (0, 1)$$

$$w = \operatorname{argmin}_w L(y, \tilde{y})$$

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

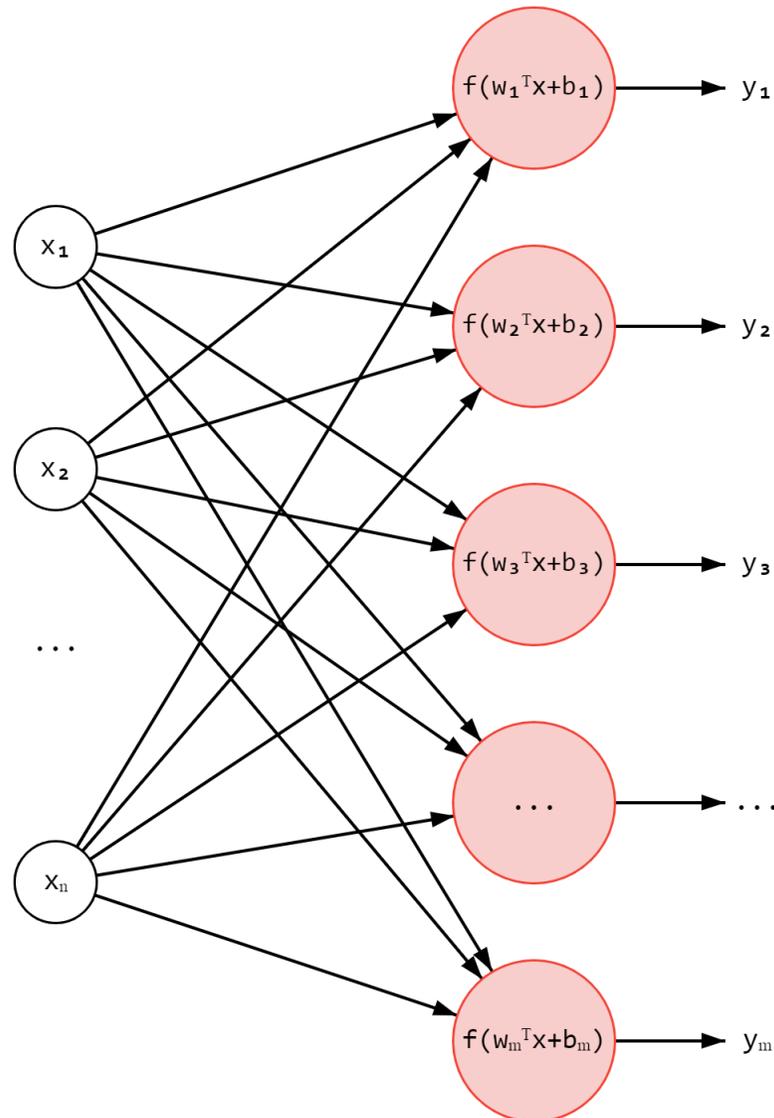


Логистическая функция (сигмоида)

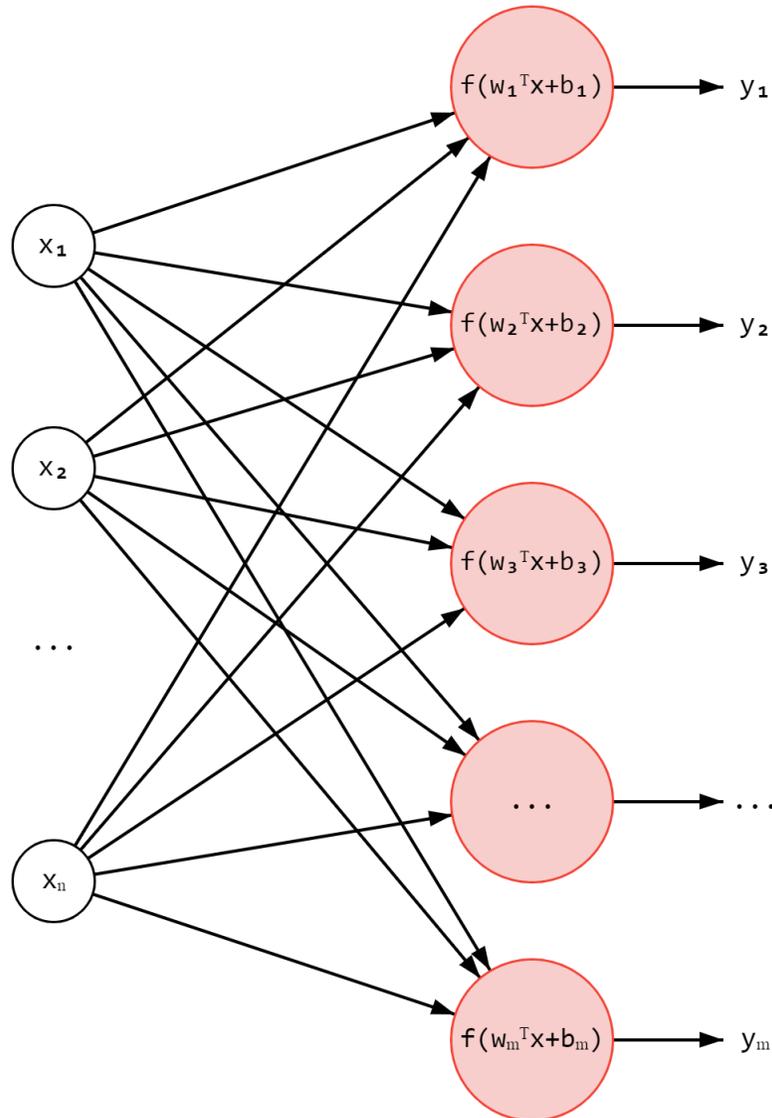


$$y = \sigma(w^T \cdot x) = \frac{1}{1 + e^{-w^T \cdot x}}$$

Оптимизируя функцию потерь методом градиентного спуска, находим параметры  $w_1, \dots, w_n$ .



- состоит из  $m$  независимых нейронов
- функция активации у каждого нейрона обычно общая для всего слоя
- соответствующие входы нейронов соединены между собой – слой содержит  $n$  входов
- имеет  $(n + 1) \cdot m$  весовых коэффициентов
- выходом слоя является вектор, а не число, в отличие от одного нейрона



Удобно представлять слой в виде матриц весов:

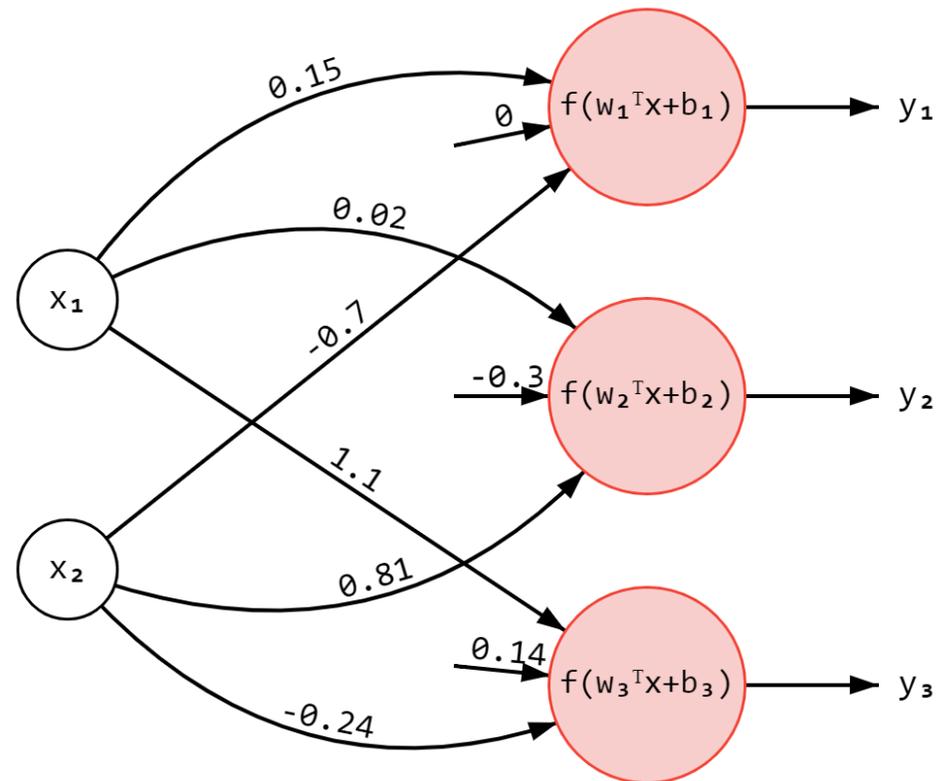
$$W = \begin{bmatrix} W_{11} & W_{12} & \dots & W_{1n} \\ W_{21} & W_{22} & \dots & W_{2n} \\ \dots & \dots & \dots & \dots \\ W_{m1} & W_{m2} & \dots & W_{mn} \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_m \end{bmatrix}$$

Вход слоя – такой же вектор, как для нейрона

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$$

Выходное значение слоя вычисляется аналогично нейрону:  $y = f(Wx + b)$

Пусть слой имеет 2 входа и 3 выхода, функция активации –  $\tanh$ , а весовые коэффициенты и входной сигнал такие:



$$W = \begin{bmatrix} 0.15 & -0.7 \\ 0.02 & 0.81 \\ 1.1 & -0.24 \end{bmatrix} \quad b = \begin{bmatrix} 0 \\ -0.3 \\ 0.14 \end{bmatrix} \quad x = \begin{bmatrix} 0.5 \\ -0.3 \end{bmatrix}$$

$$Wx + b = \begin{bmatrix} 0.15 & -0.7 \\ 0.02 & 0.81 \\ 1.1 & -0.24 \end{bmatrix} \cdot \begin{bmatrix} 0.5 \\ -0.3 \end{bmatrix} + \begin{bmatrix} 0 \\ -0.3 \\ 0.14 \end{bmatrix} = \begin{bmatrix} 0.285 \\ -0.533 \\ 0.762 \end{bmatrix}$$

$$y = f(Wx + b) = \begin{bmatrix} \tanh(0.285) \\ \tanh(-0.533) \\ \tanh(0.762) \end{bmatrix} \approx \begin{bmatrix} 0.277 \\ -0.487 \\ 0.642 \end{bmatrix}$$

Функция  $\text{softmax}(z)$  преобразует вектор  $z$  размерности  $n$  в вектор той же размерности, где каждая координата полученного вектора представлена вещественным числом в интервале  $[0,1]$  и сумма координат равна 1.

$$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_{i=1}^n e^{z_i}}$$

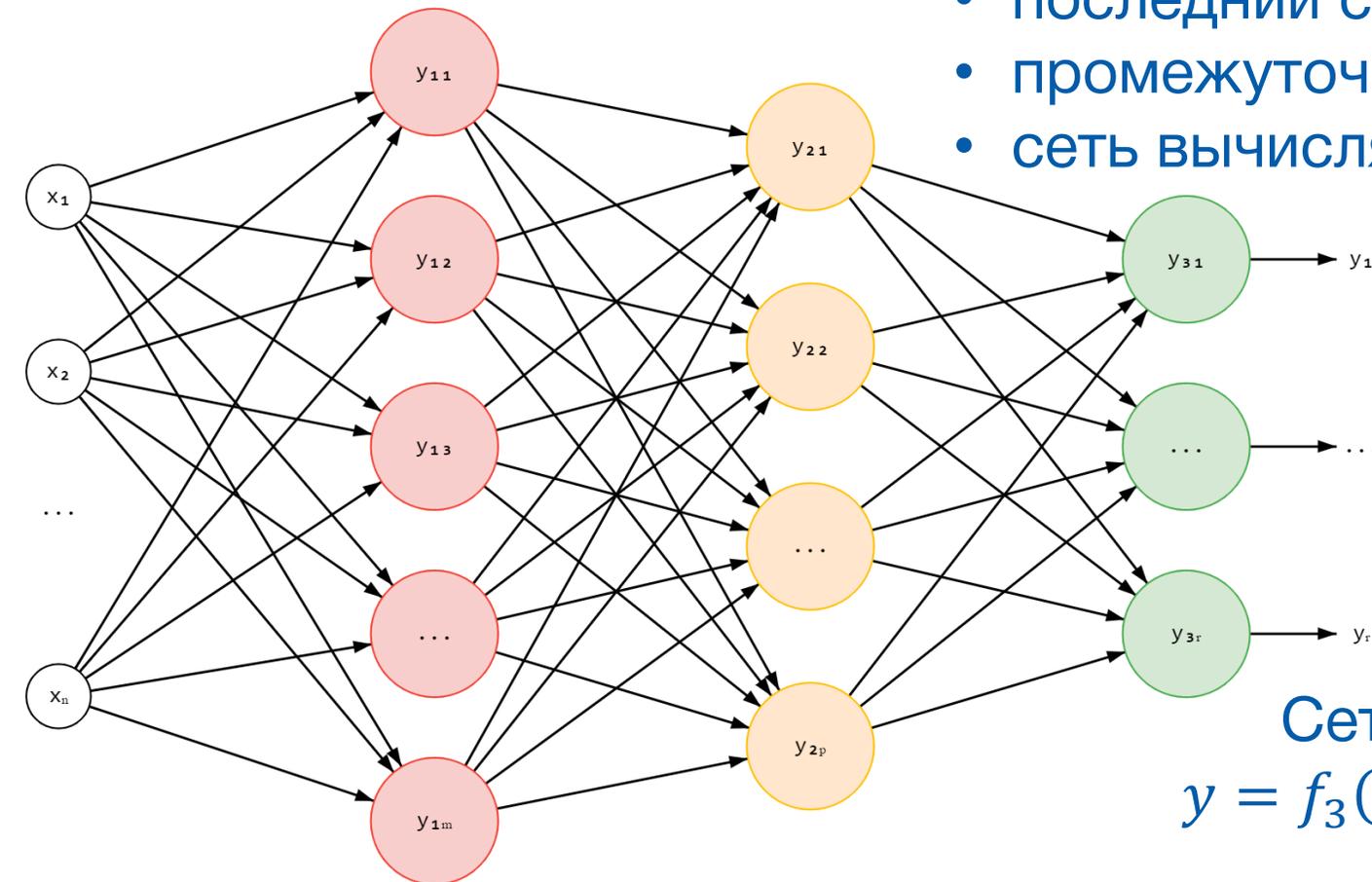
Используется для классификации на 2 и более классов.

Пример:

Пусть  $z = [1, 0, 0.5, -1, 1]$

Тогда  $\text{softmax}(z) \approx [0.322, 0.118, 0.195, 0.043, 0.322]$

- каждый слой – линейный
- выход слоя является входом следующего слоя
- последний слой называют **выходным**
- промежуточные слои называют **скрытыми**
- сеть вычисляет выход следующим образом:



Слой 1:  $y_1 = f_1(W_1x + b_1)$

Слой 2:  $y_2 = f_2(W_2y_1 + b_2)$

...

Слой  $i$ :  $y_i = f_i(W_iy_{i-1} + b_i)$

Выходной слой:  $y = f_n(W_ny_{n-1} + b_n)$

Сеть моделирует следующую функцию:

$$y = f_3(W_3 \cdot f_2(W_2 \cdot f_1(W_1x + b_1) + b_2) + b_3) \dots$$

$(x_i, y_i), i \in 1..n$  – обучающая выборка, где  $x_i$  – объекты,  $y_i$  – правильные ответы  
 $f(x, \theta)$  – дифференцируемая нейронная сеть с параметрами  $\theta$

Процесс обучения заключается в поиске таких параметров  $\theta$ , при которых нейронная сеть будет выдавать «близкие» к правильным ответам на элементах обучающей выборки.

Для оценки «близости» ответов используются различные функции ошибки (потерь)  $L(y, t)$ , где  $y$  – выход сети, а  $t$  – ожидаемое значение, например:

$$MSE: (y - t)^2$$

$$MAE: |y - t|$$

$$cross-entropy: -t \ln y - (1 - t) \cdot \ln(1 - y)$$

Чтобы обучить сеть, применяем её ко всем элементам входной выборки, сравниваем получаемые ответы с правильными и строим функционал качества:

$$L = L(\theta) = \frac{1}{n} \sum_{i=1}^n l(f(x_i, \theta), y_i) = \frac{1}{n} \sum_{i=1}^n l_i$$

Так как  $L(\theta)$  - дифференцируемая функция, то, оптимизируя её параметры с помощью градиентного спуска, можно попытаться попасть в локальный минимум:

$$\theta^{i+1} = \theta^i - \alpha \cdot \nabla L(\theta^i)$$

Самая сложная часть – вычисление градиента  $\nabla L(\theta^i)$

Вычисление  $L$  при больших  $n$  – трудоёмкая задача.

Решение – стохастический градиентный спуск:

- каждую итерацию выбирается случайный элемент из обучающей выборки ( $x_j$ )
- рассчитывается функция потерь  $l_j = l(f(x_j, \theta^i), y_j)$
- обновляются параметры:  $\theta^{i+1} = \theta^i - \alpha \cdot \nabla l_j(\theta^i)$

Недостатки:

- подвержен выбросам, из-за чего необходимо использовать малое значение  $\alpha$ , что приводит к большому числу шагов для сходимости
- невозможно распараллелить вычисление ошибки по элементам обучающей выборки

- каждую итерацию произвольно выбирается  $m$  ( $1 < m \ll n$ ) примеров из обучающей выборки
- вычисляется суммарная функция потерь на выбранных примерах:

$$l_{batch} = \frac{1}{m} \sum_{j=1}^m l(f(x_j, \theta^i), y_j)$$

- обновляются параметры:

$$\theta^{i+1} = \theta^i - \alpha \cdot \nabla \left[ \frac{1}{m} \sum_{j=1}^m l(f(x_j, \theta^i), y_j) \right]$$

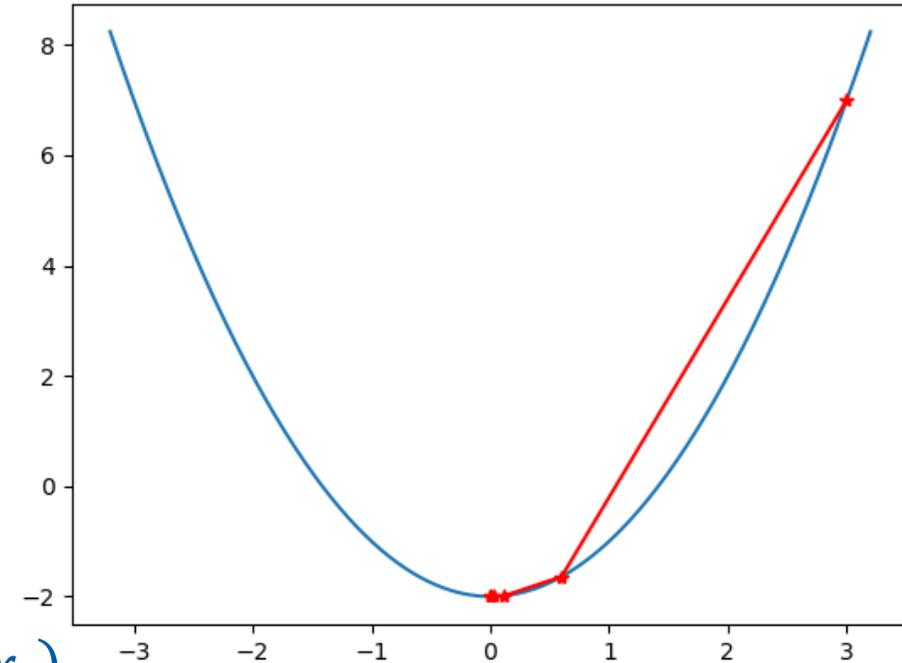
Более устойчив к выбросам, так что можно выбирать больший шаг  $\alpha$   
Ошибку  $l_{batch}$  можно считать параллельно

Найти минимум функции  $f(x) = x^2 - 2$  с помощью градиентного спуска.

В качестве начальной точки возьмём  $x_0 = 3$

Шаг спуска выберем  $\alpha = 0.4$

Градиент функции равен  $\nabla f(x) = 2x$



Запустим итерационный процесс:  $x_{i+1} = x_i - \alpha \cdot \nabla f(x_i)$

$$x_1 = x_0 - \alpha \cdot \nabla f(x_0) = 3 - 0.4 \cdot 2 \cdot 3 = 0.6, f(x_1) = -1.64$$

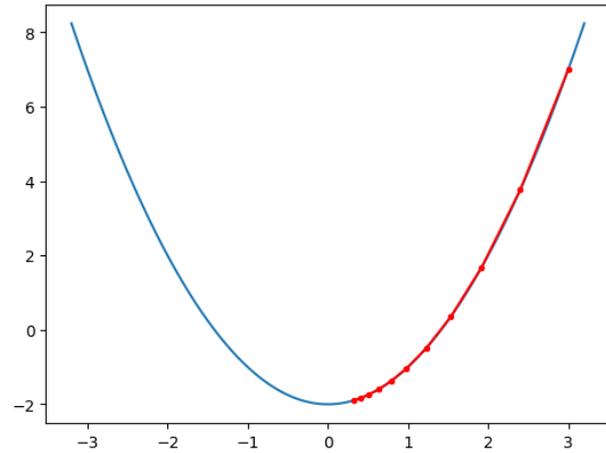
$$x_2 = x_1 - \alpha \cdot \nabla f(x_1) = 0.6 - 0.4 \cdot 2 \cdot 0.6 = 0.12, f(x_2) = -1.9856$$

$$x_3 = x_2 - \alpha \cdot \nabla f(x_2) = 0.12 - 0.4 \cdot 2 \cdot 0.12 = 0.024, f(x_3) = -1.9994$$

$$x_4 = x_3 - \alpha \cdot \nabla f(x_3) = 0.024 - 0.4 \cdot 2 \cdot 0.024 = 0.0048, f(x_4) = -1.9997$$

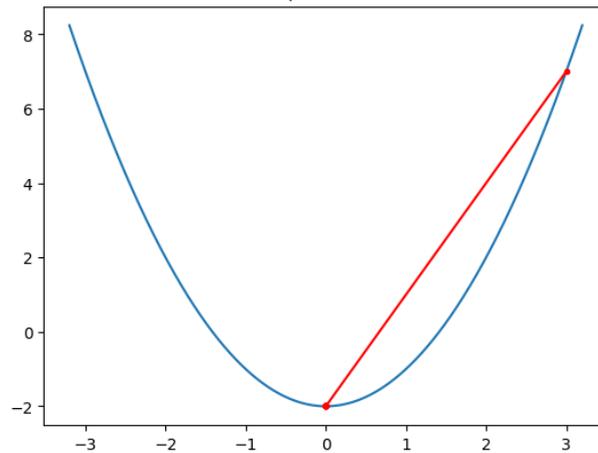
$$x_5 = x_4 - \alpha \cdot \nabla f(x_4) = 0.0048 - 0.4 \cdot 2 \cdot 0.0048 = 0.00096, f(x_4) = -1.9999$$

alpha = 0.1



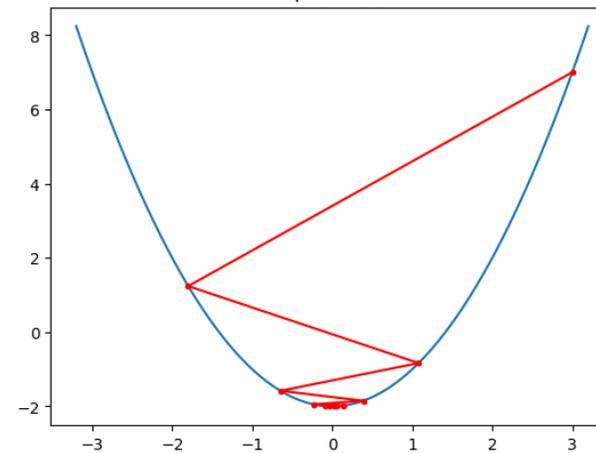
Шаг слишком мал  
Долгая сходимость

alpha = 0.5



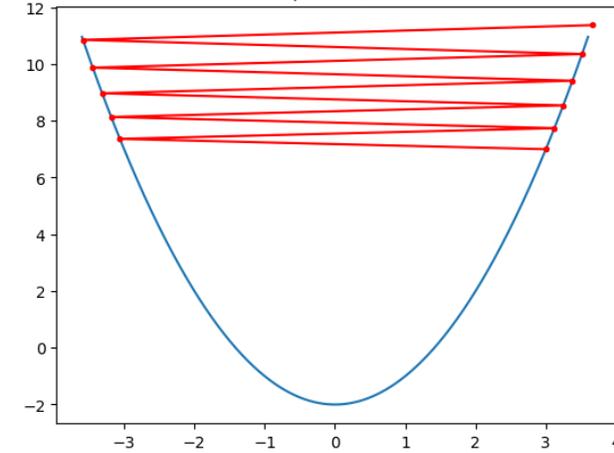
Идеальный шаг  
Моментальная сходимость

alpha = 0.8



Большой шаг  
Плохая сходимость,  
градиент постоянно  
меняет знак

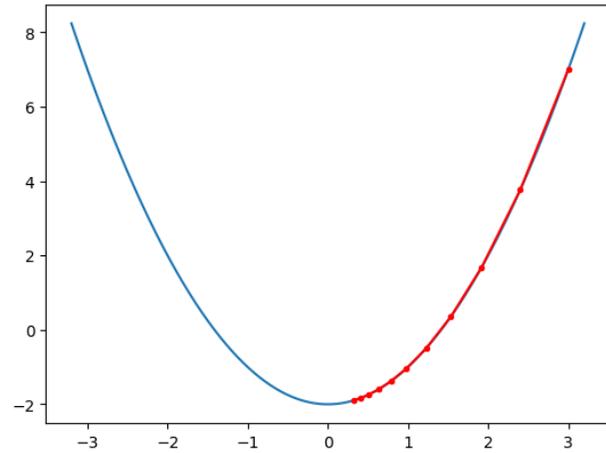
alpha = 1.01



Очень большой шаг  
Расходимость

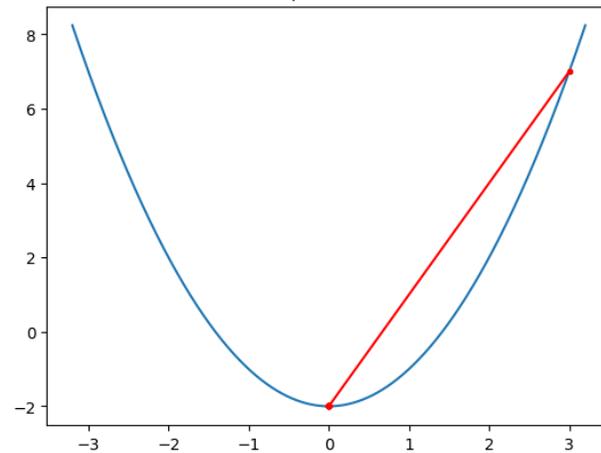
Как выбирать оптимальный шаг?

alpha = 0.1



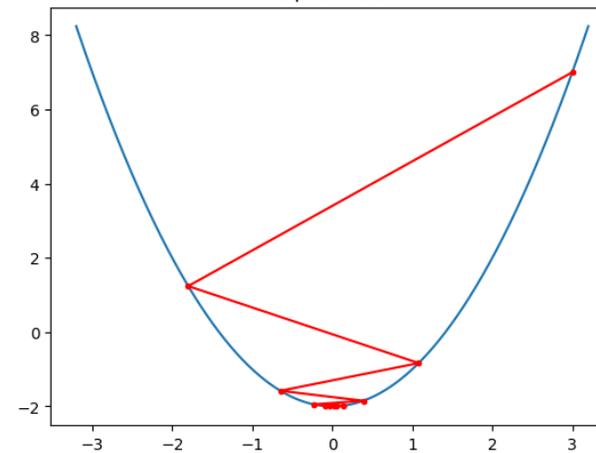
Шаг слишком мал  
Долгая сходимость

alpha = 0.5



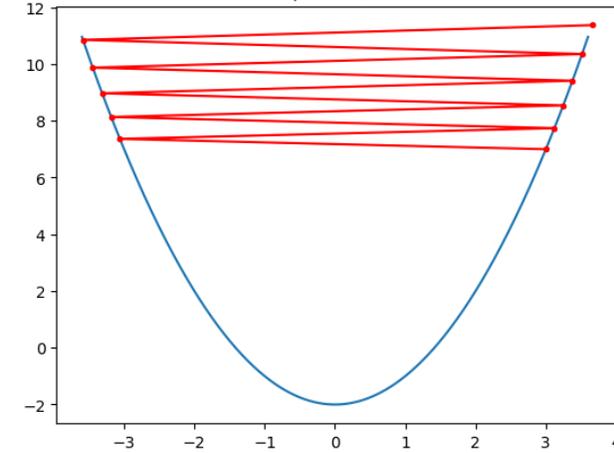
Идеальный шаг  
Моментальная сходимость

alpha = 0.8



Большой шаг  
Плохая сходимость,  
градиент постоянно  
меняет знак

alpha = 1.01



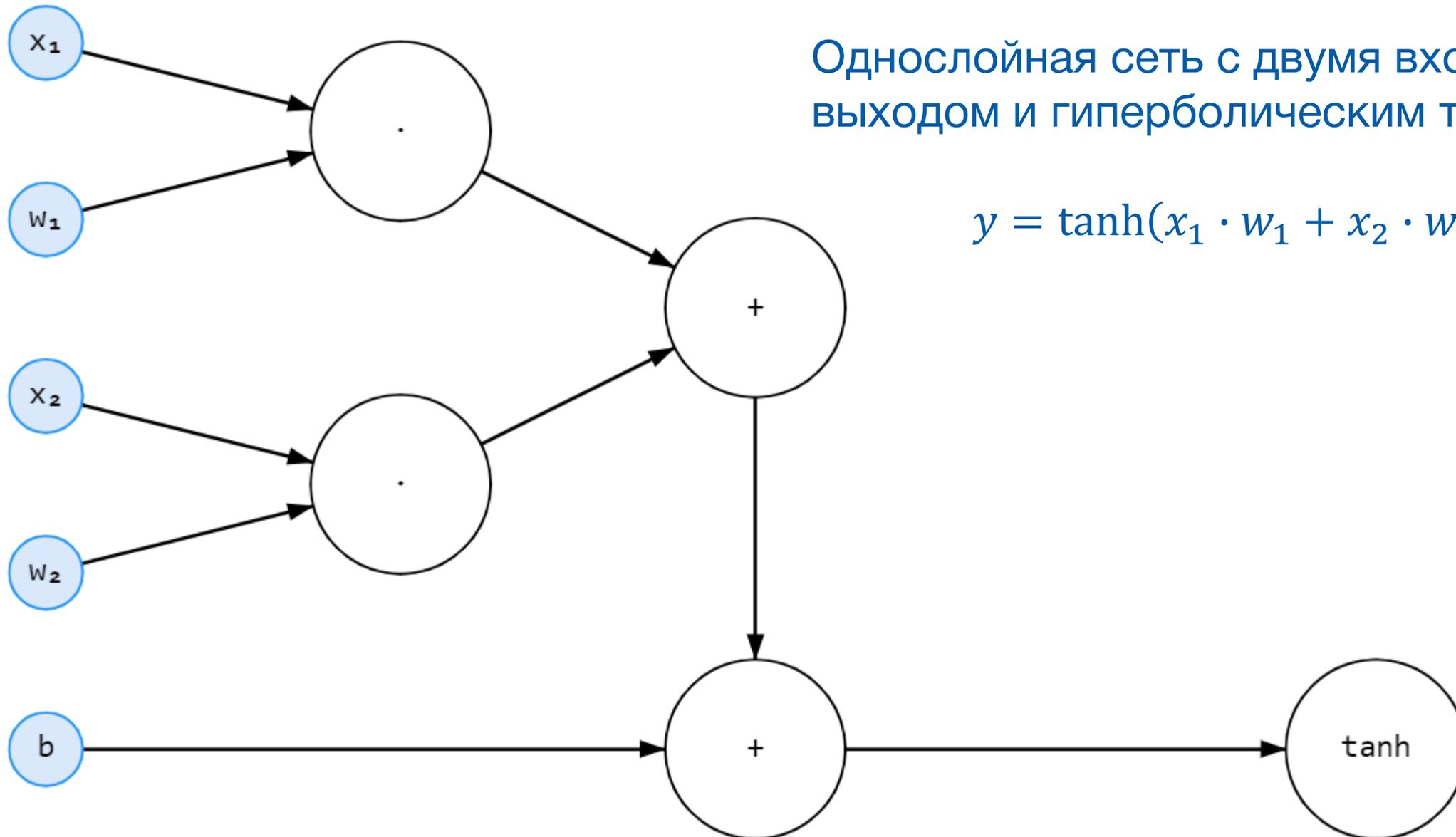
Очень большой шаг  
Расходимость

Как выбирать оптимальный шаг – эксперименты!

Обучение выполняется некоторое количество «эпох» - полных итераций по всей обучающей выборке. Иногда обучение останавливают раньше, если на валидационной выборке качество перестаёт улучшаться (early stopping).

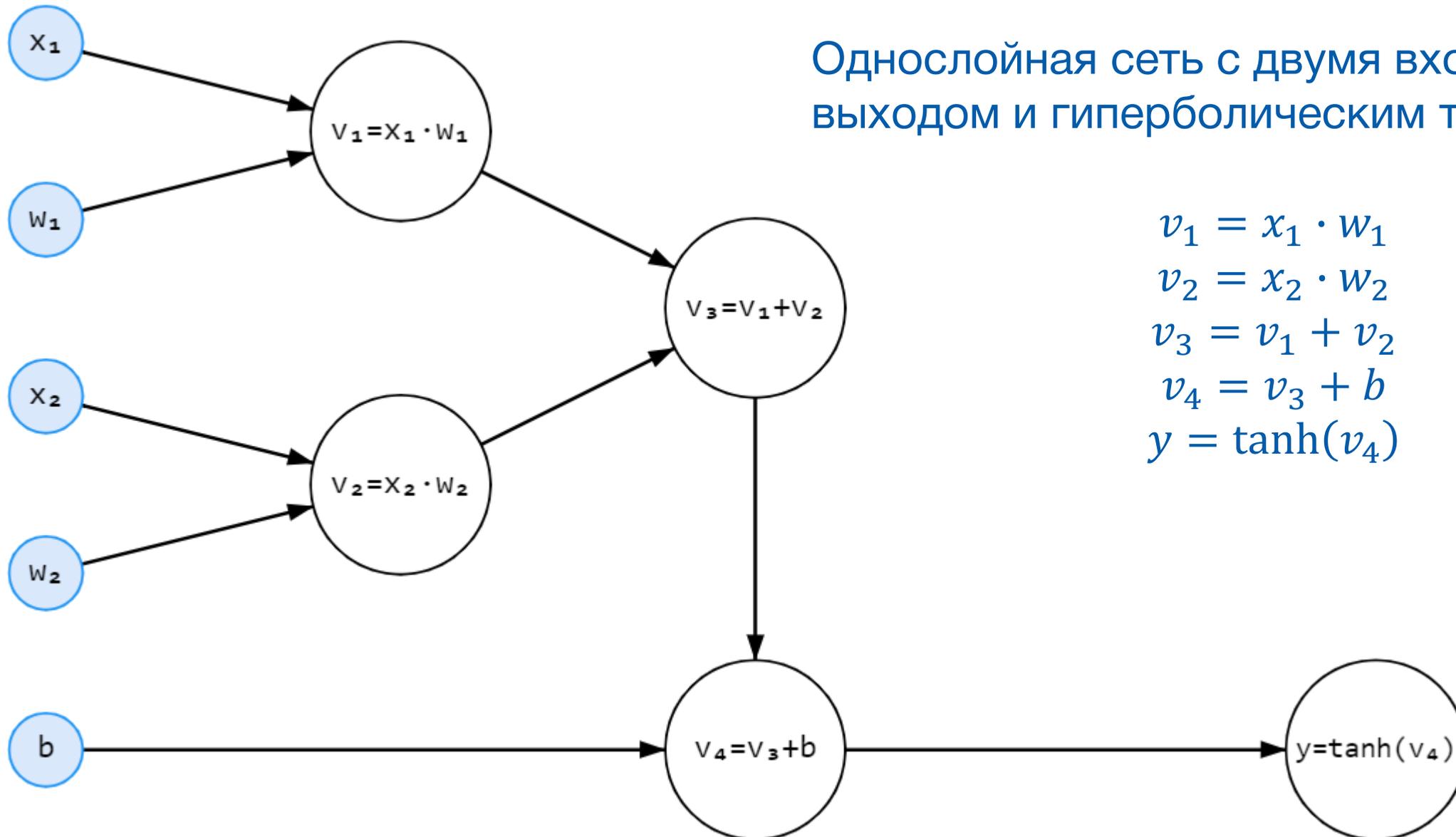
Поскольку выбор шага довольно сложная задача, обычно используются адаптивные методы градиентного спуска, такие как SGD with momentum, Adam, Adagrad, RMSprop и тд. Зачастую эти методы используют информацию о градиенте на прошлых итерациях и подстраивают шаг при каждом обновлении.

Вопрос: как вычислять градиент  $\nabla L(\theta^i)$  по каждому параметру  $\theta_j^i$  глубокой нейронной сети (сложной дифференцируемой функции)?



Однослойная сеть с двумя входами, одним выходом и гиперболическим тангенсом:

$$y = \tanh(x_1 \cdot w_1 + x_2 \cdot w_2 + b)$$



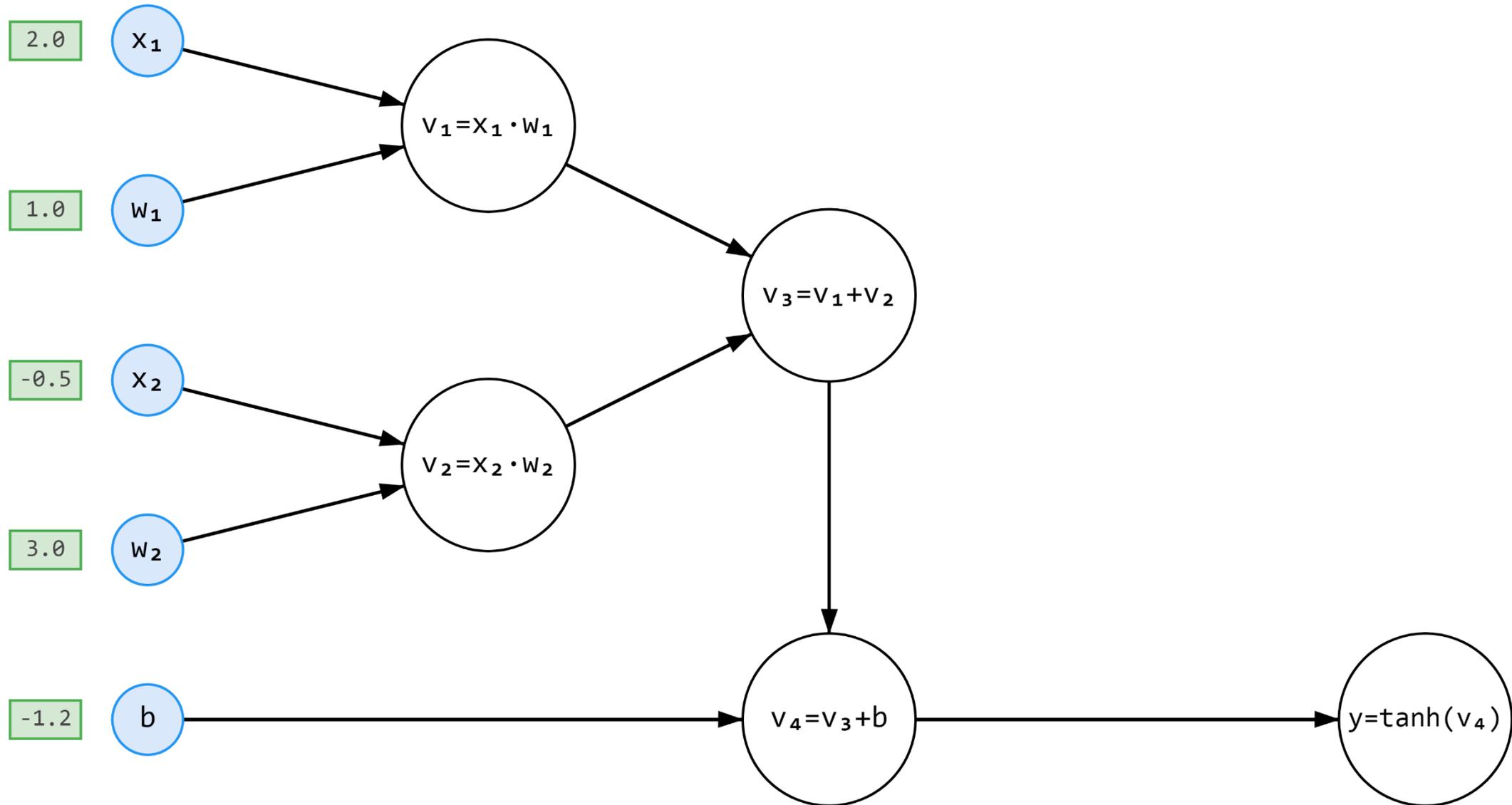
Однослойная сеть с двумя входами, одним выходом и гиперболическим тангенсом:

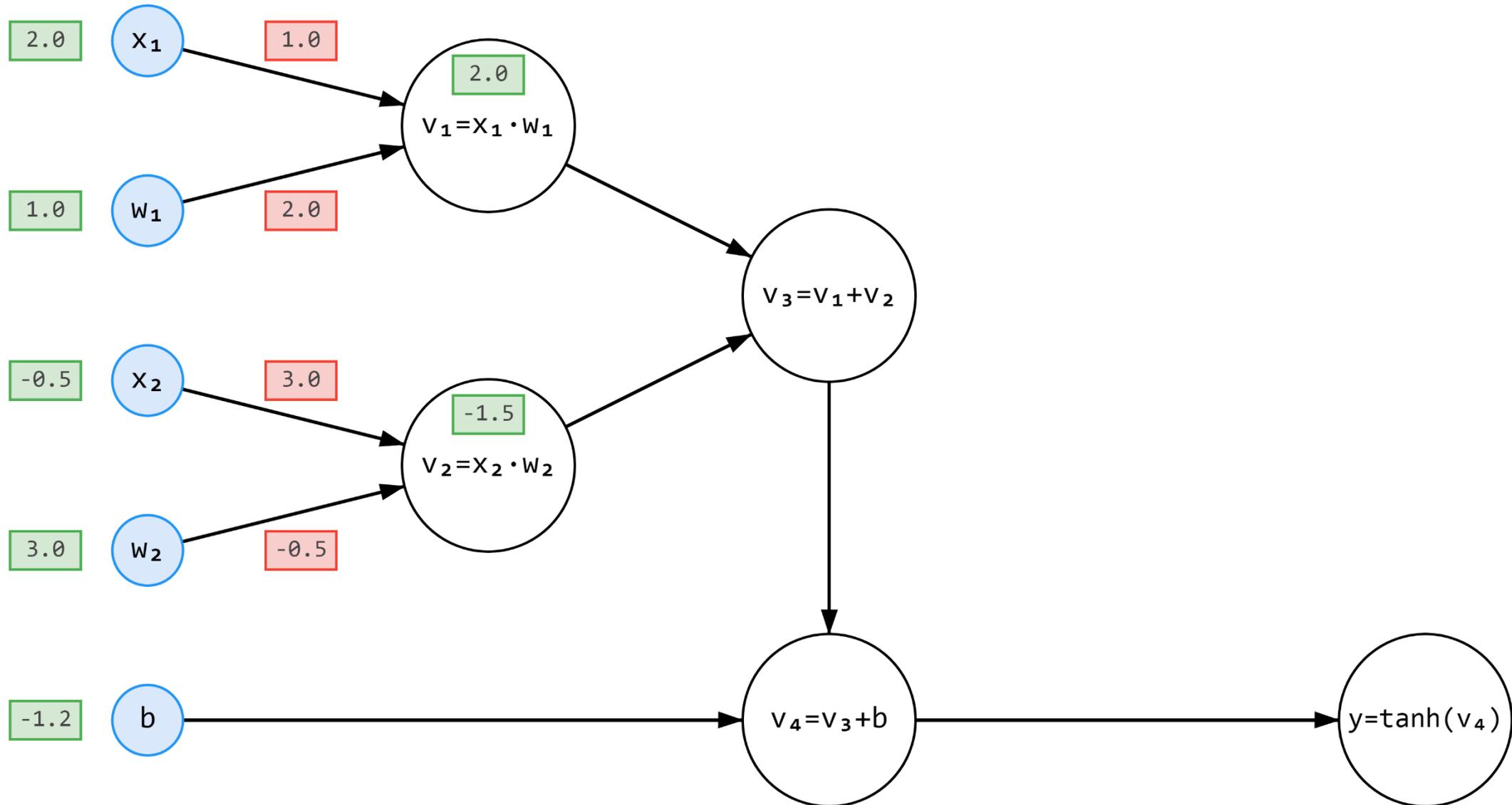
$$\begin{aligned}v_1 &= x_1 \cdot w_1 \\v_2 &= x_2 \cdot w_2 \\v_3 &= v_1 + v_2 \\v_4 &= v_3 + b \\y &= \tanh(v_4)\end{aligned}$$

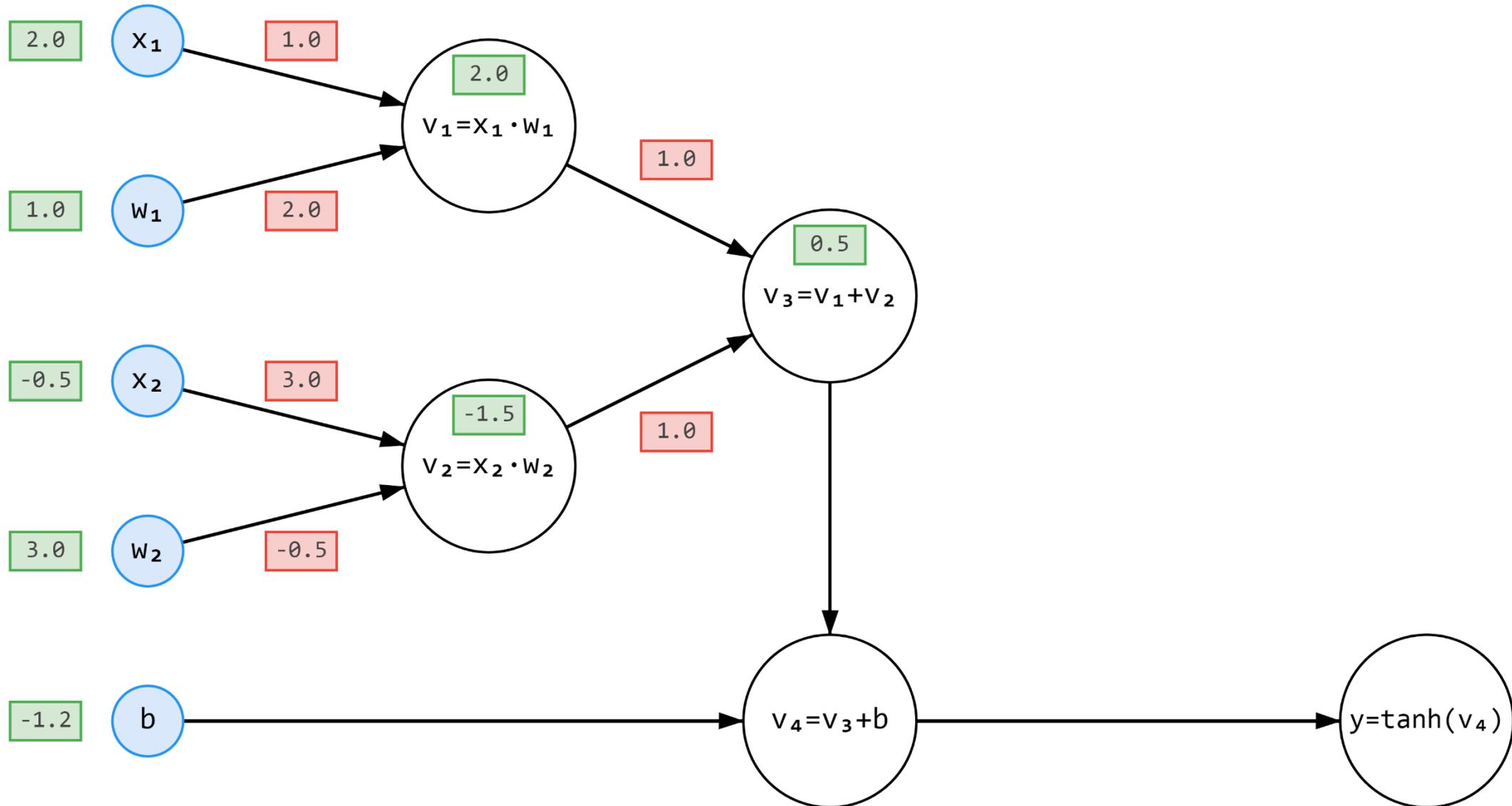
Обратное распространение ошибки (backpropagation) – алгоритм, позволяющий вычислить частные производные на произвольном графе вычислений. Состоит из двух этапов:

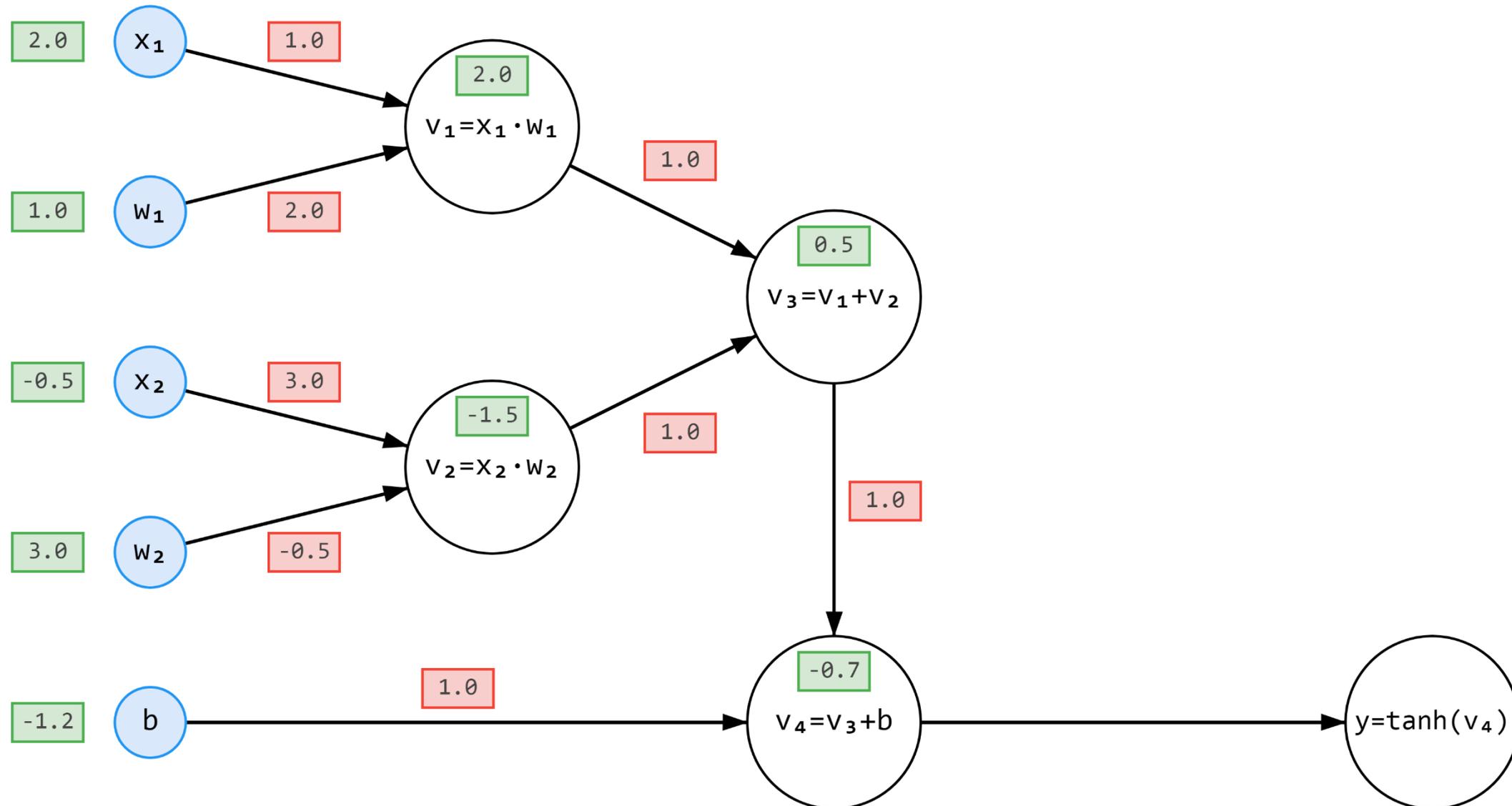
**Прямое распространение (forward pass):** входные сигналы проходят по графу через каждый узел и сохраняют выходные значения и частные производные узлов.

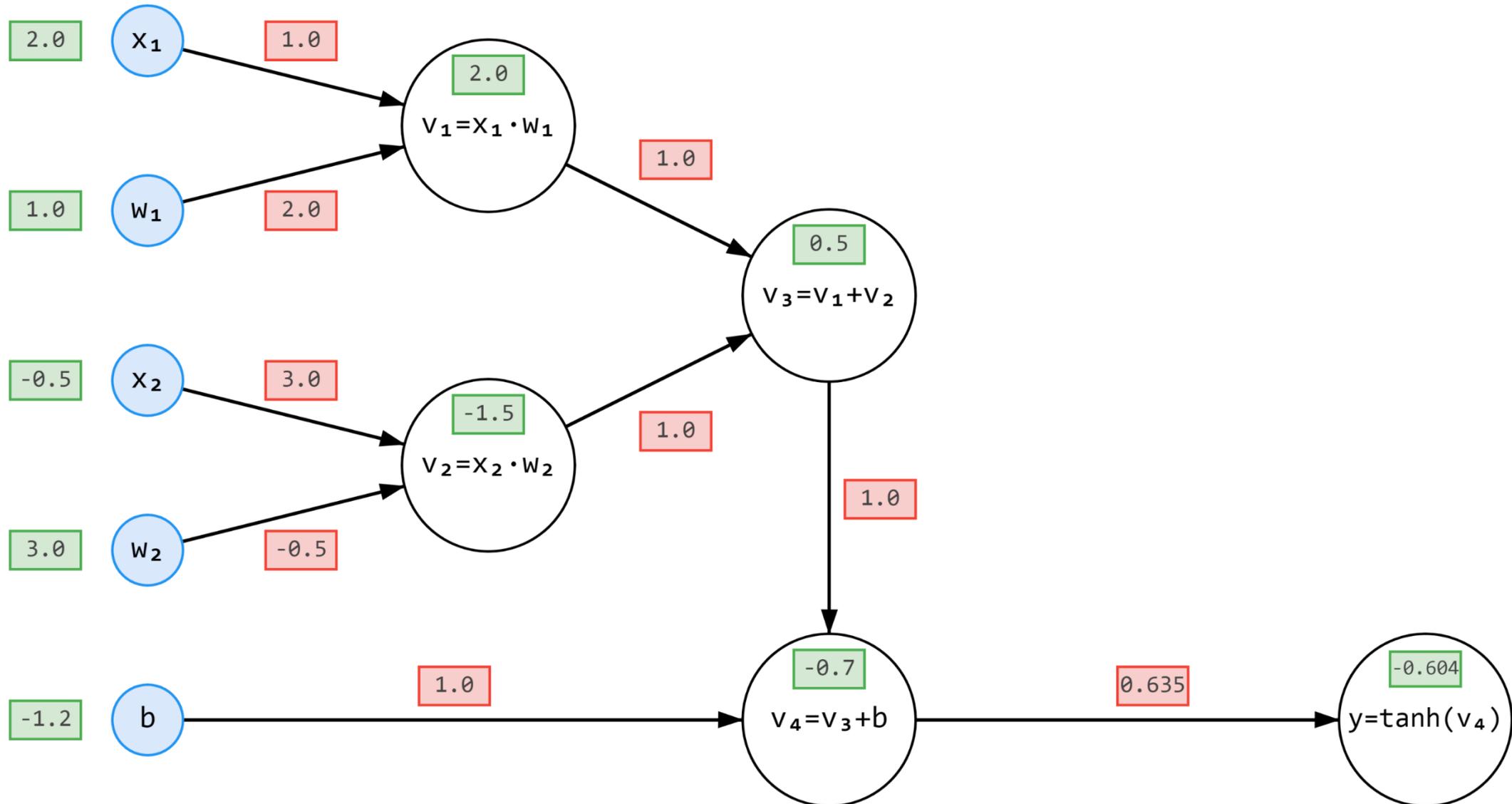
**Обратное распространение (backward pass):** на основе вычисленных значений при прямом проходе вычисляются частные производные реализуемой функции по каждому из её аргументов – входных узлов графа.

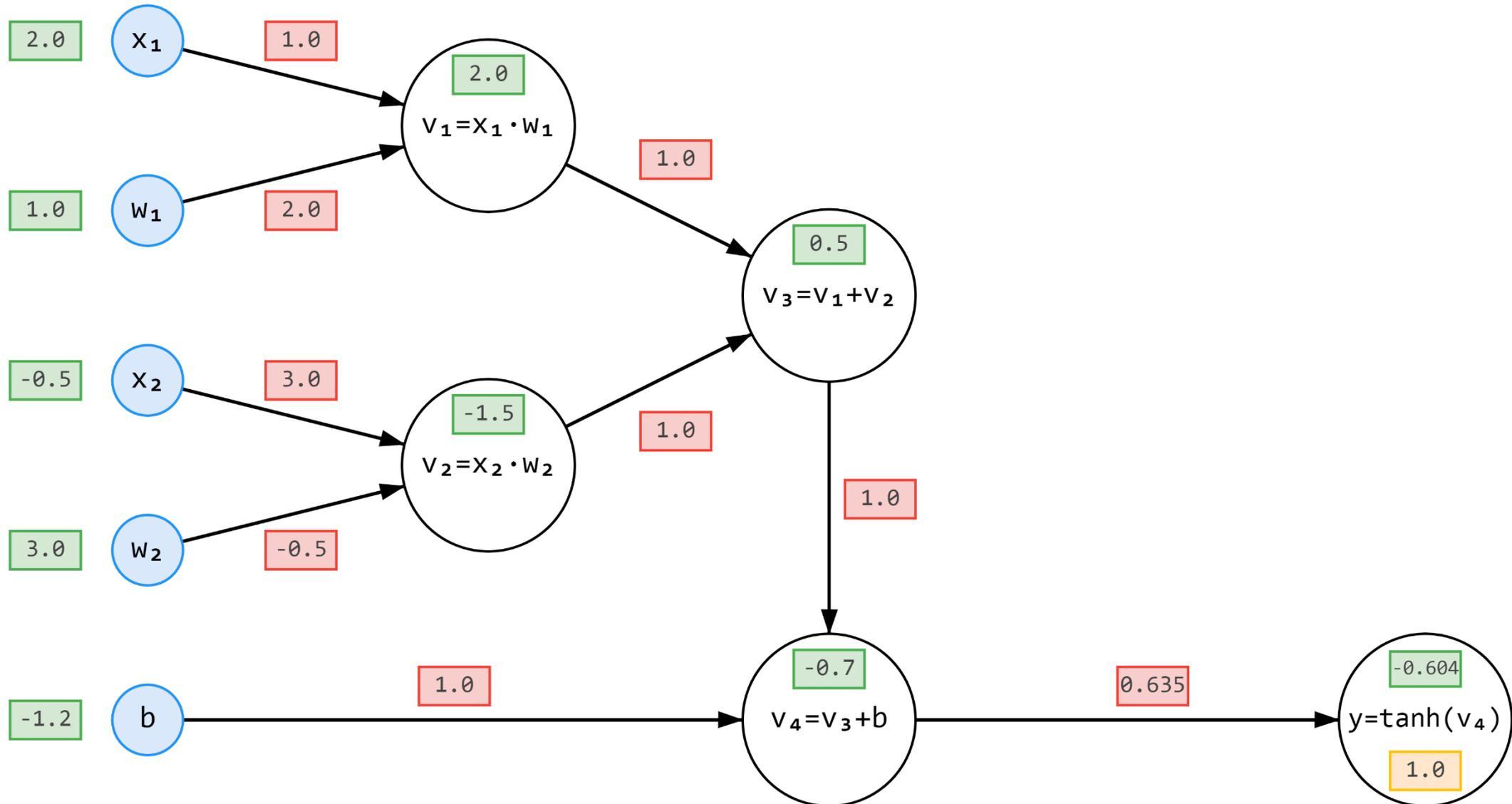


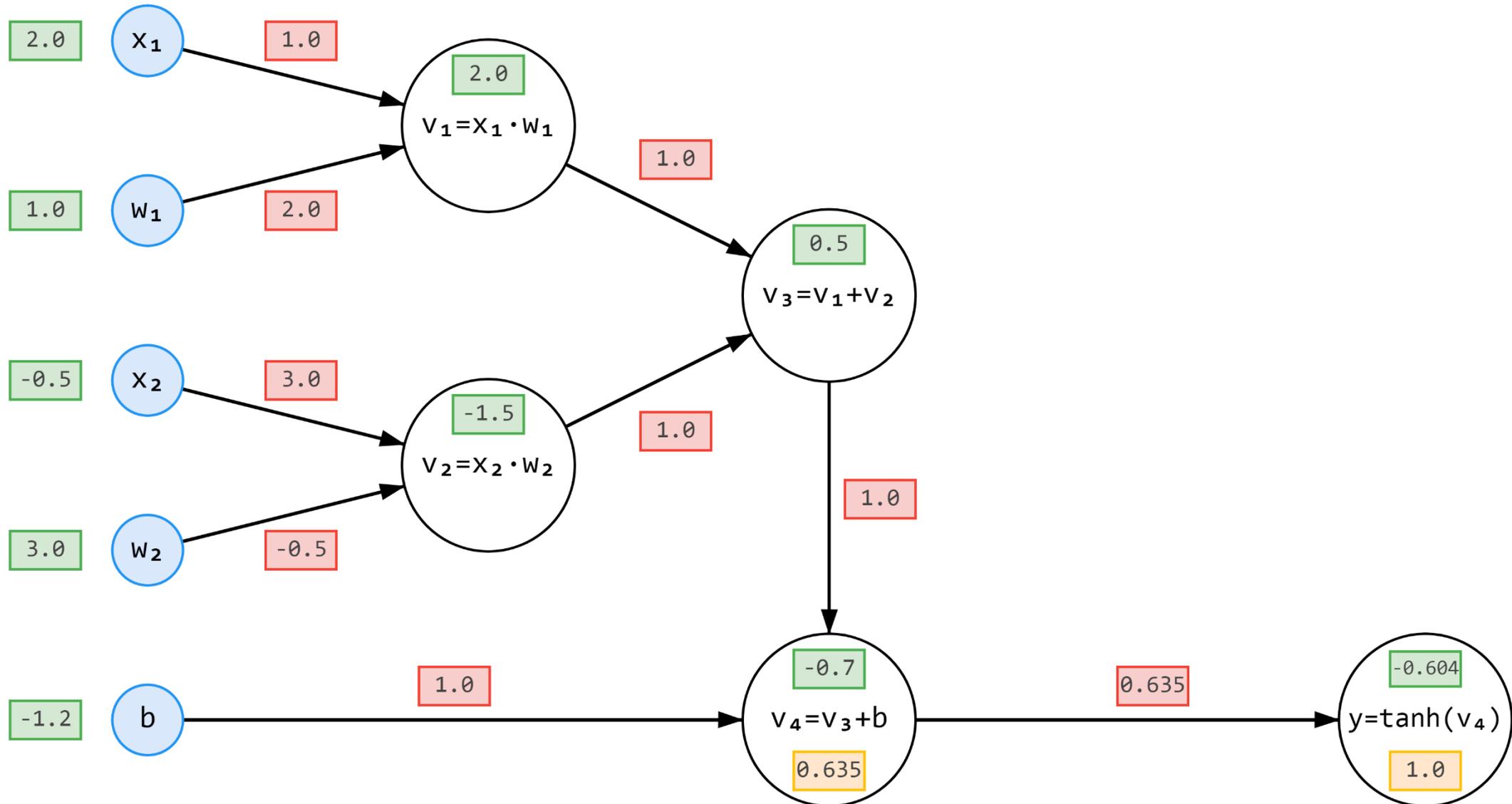


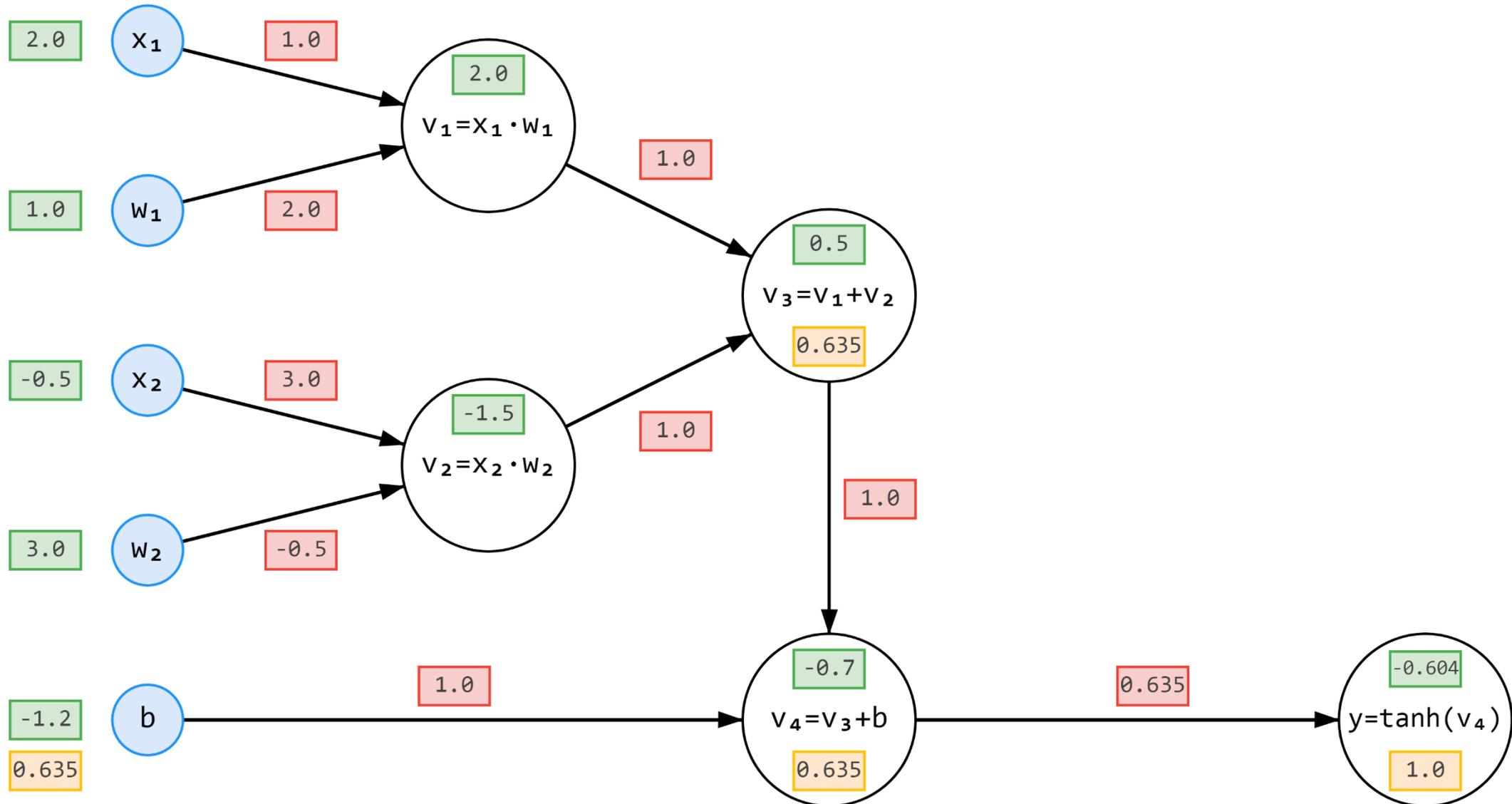


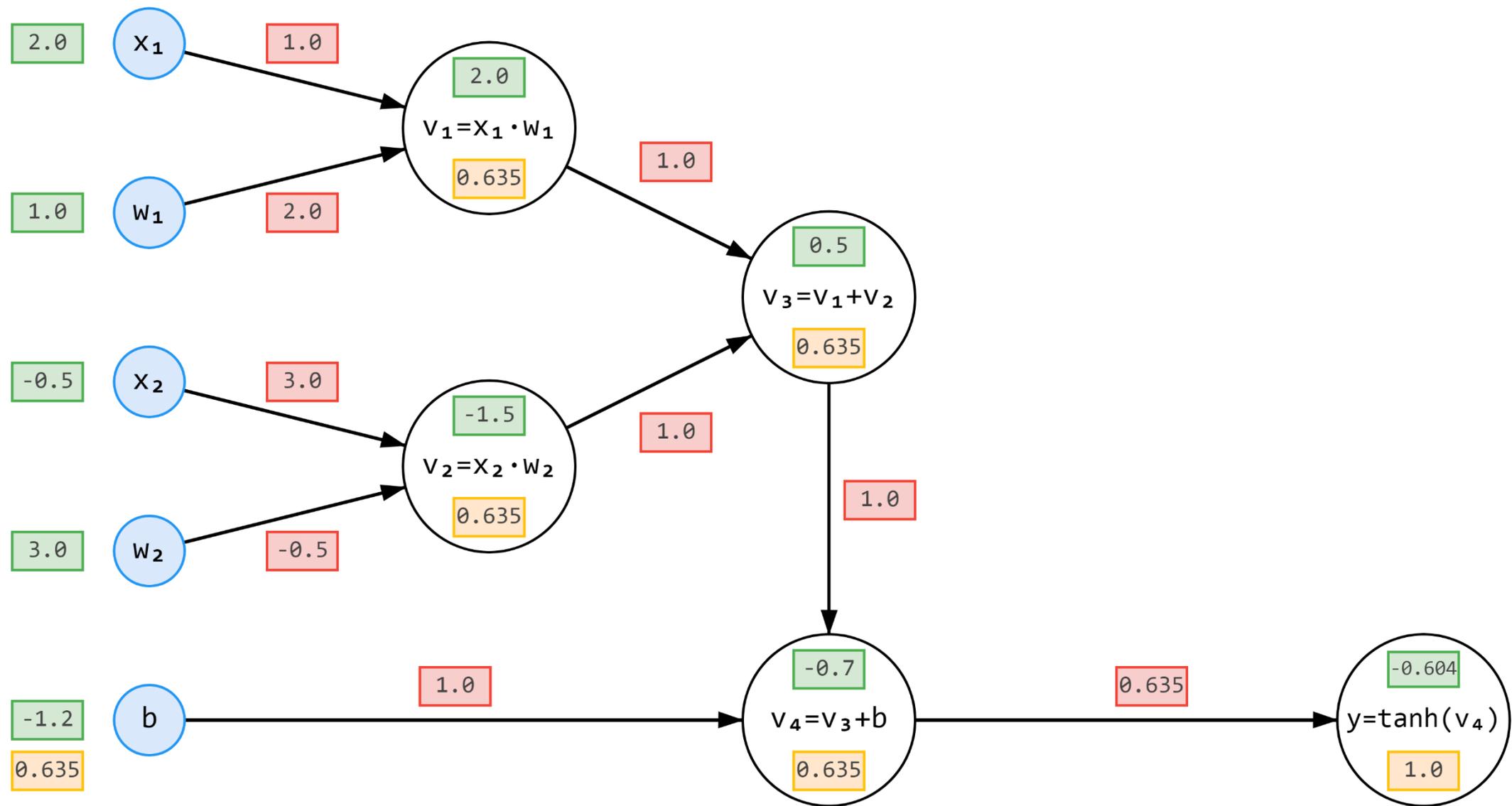


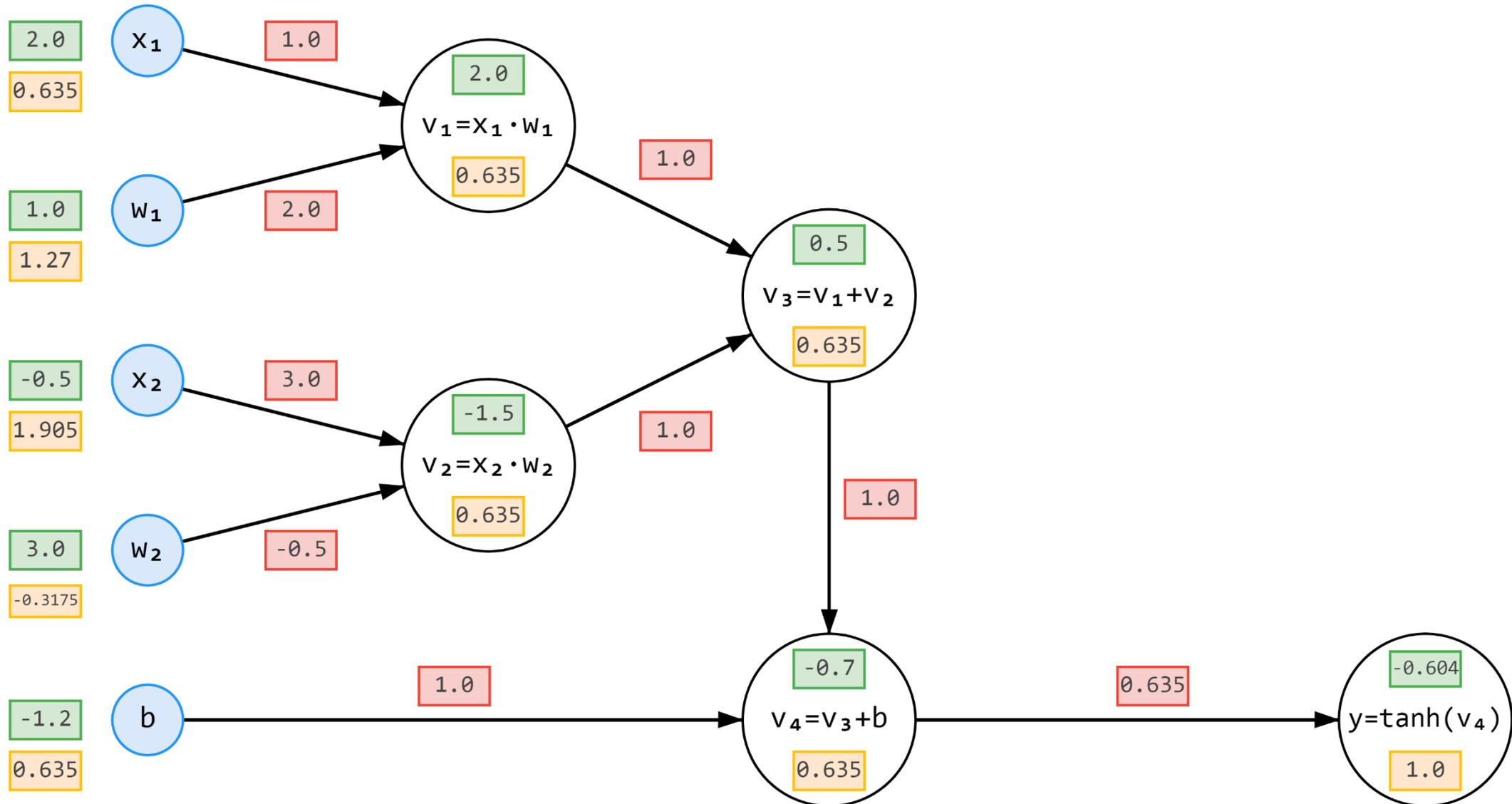












При backpropagation в глубокой нейронной сети градиент на первых слоях может быть очень маленьким из-за последовательного умножения на  $\left| \frac{\partial z}{\partial y} \right| \ll 1$ , поэтому эти слои будут обновляться «слабо».

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

Перед обучением нейронной сети нужно проинициализировать ее параметры  $\theta$ . Обычно используется нормальное распределение с нулевым средним и среднеквадратичным отклонением обратным пропорциональным количеству коэффициентов:  $W \sim N\left(0, \frac{1}{\dim(W)}\right)$ .

Существуют техники, учитывающие функции активации и размеры слоя (He, Xavier).

Нужно аккуратно подбирать начальную инициализацию весов, функции активации, размеры и количество слоев на валидационной выборке – проблемы недообучения / переобучения

Вопрос: почему бы не инициализировать все веса нулём? А константой?

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.

**Classification**



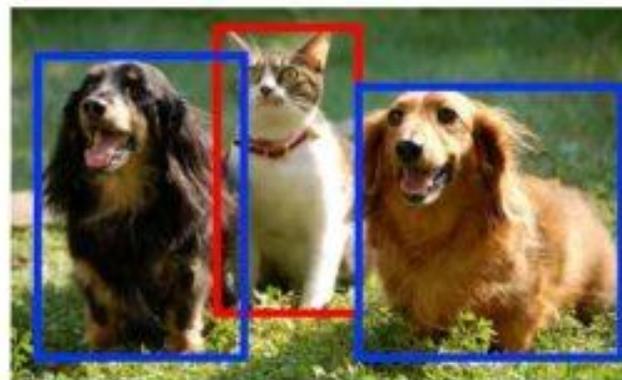
CAT

**Classification  
+ Localization**



CAT

**Object Detection**



CAT, DOG

**Instance Segmentation**



CAT, DOG

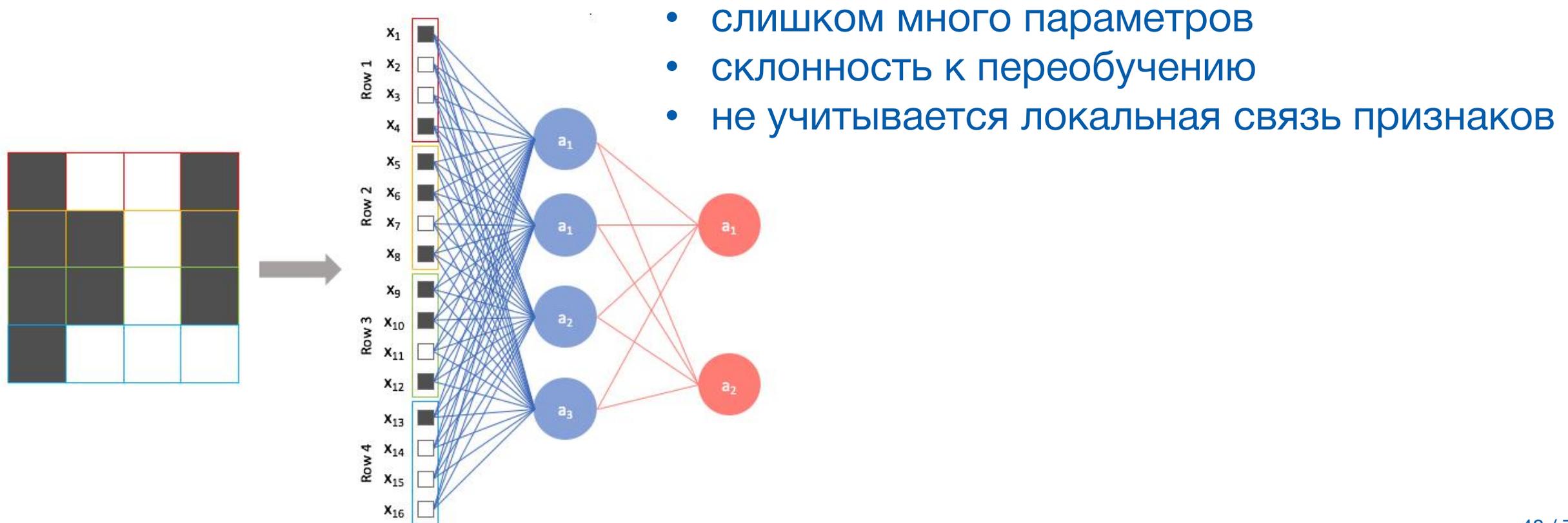
Single object

Multiple objects

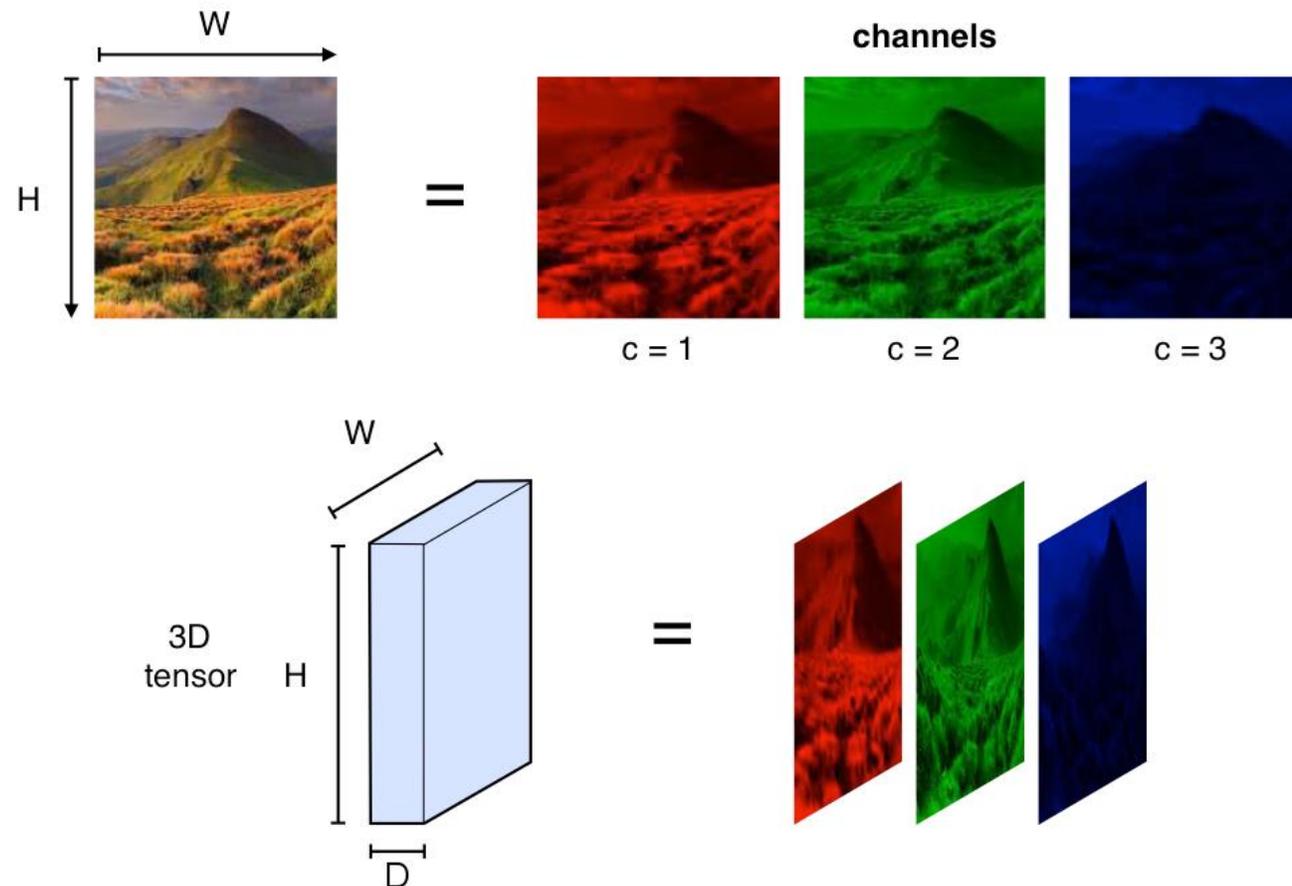
Черно-белое изображение 32x32  $\Rightarrow$  вектор признаков размерности 1024.

Возьмем один скрытый слой размерности 1024.

Матрица такого преобразования содержит  $1024 \times 1024 = 1048576$  параметров.

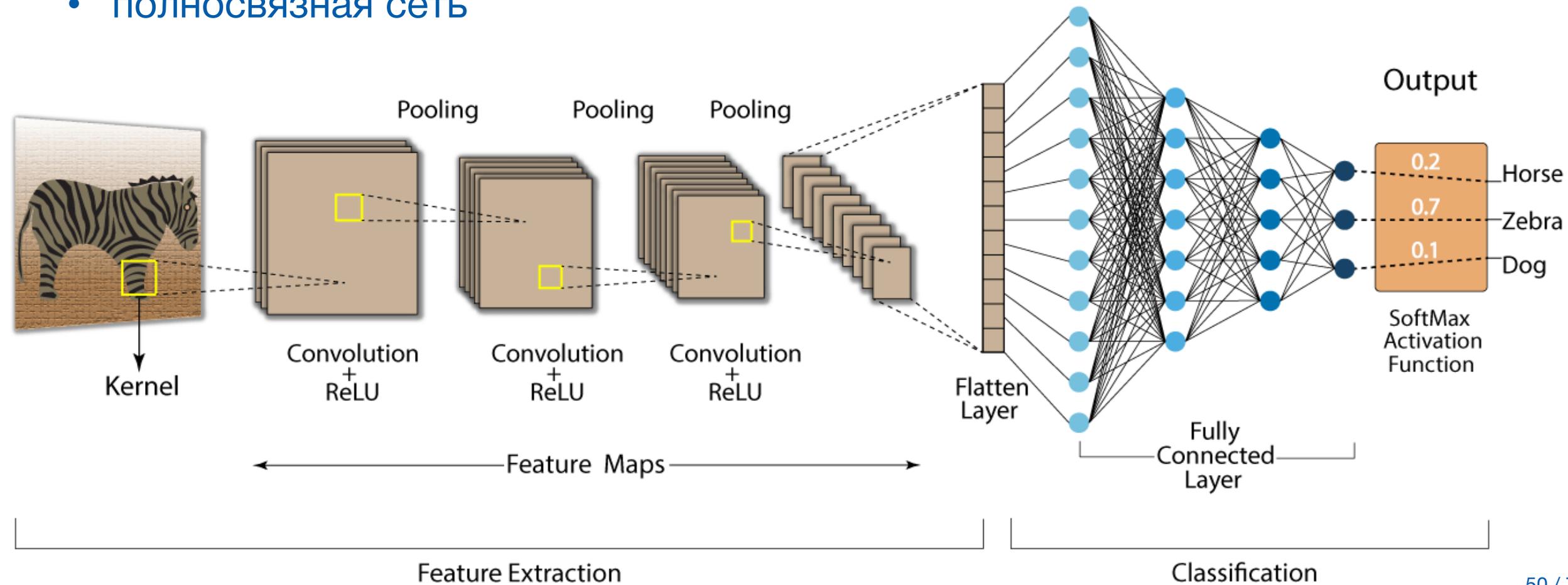


Полносвязные сети работают с векторами. Представлять изображения в виде векторов неразумно, так как теряется пространственная связь между элементами. На смену векторам приходят тензоры.



Свёрточные сети состоят из следующих частей:

- свертка + активация
- пулинг (понижение размерности)
- полносвязная сеть



Свёртка – это функция, показывающая «схожесть» одной функции и отражённой и сдвинутой другой. В двумерном случае свёртку можно описать следующей формулой:

$$\langle input * filter \rangle(i, j) = \sum_{x=0}^{f_w} \sum_{y=0}^{f_h} input(i - x, j - y) \cdot filter(x, y)$$

Фильтр в свёрточном слое – тензор размерности  $[f_w, f_h, d]$ , где  $d$  – «глубина» входного тензора.



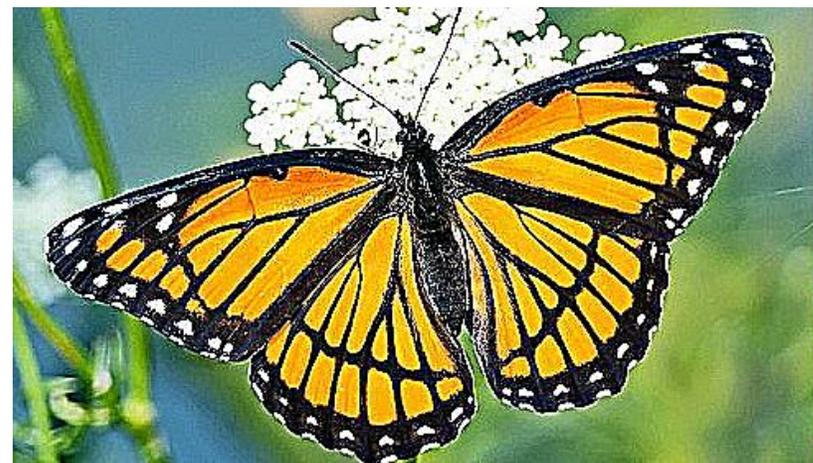
|   |   |   |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

Без изменений



|    |    |    |
|----|----|----|
| -1 | -2 | -1 |
| -2 | 4  | 2  |
| -1 | 2  | -1 |

Выделение границ



|    |    |    |
|----|----|----|
| -1 | -4 | -1 |
| -4 | 21 | -4 |
| -1 | -4 | -1 |

Повышение резкости



|     |     |     |
|-----|-----|-----|
| 1/8 | 1/8 | 1/8 |
| 1/8 | 1/8 | 1/8 |
| 1/8 | 1/8 | 1/8 |

Размытие

$$6 \cdot 3 + 5 \cdot 2 + 3 \cdot (-1) + 4 \cdot (-1) + 6 \cdot 1 + 2 \cdot 4 + 1 \cdot 1 + 3 \cdot 3 + 6 \cdot 2 = 57$$

|                 |                |                 |    |   |   |
|-----------------|----------------|-----------------|----|---|---|
| 6 <sub>3</sub>  | 5 <sub>2</sub> | 3 <sub>-1</sub> | -1 | 6 | 7 |
| 4 <sub>-1</sub> | 6 <sub>1</sub> | 2 <sub>4</sub>  | 4  | 6 | 2 |
| 1 <sub>1</sub>  | 3 <sub>3</sub> | 6 <sub>2</sub>  | 0  | 0 | 1 |
| 5               | 7              | 2               | 1  | 5 | 8 |

Изображение 4x6

×

|    |   |    |
|----|---|----|
| 3  | 2 | -1 |
| -1 | 1 | 4  |
| 1  | 3 | 2  |

Фильтр

=

|    |  |  |  |
|----|--|--|--|
| 57 |  |  |  |
|    |  |  |  |

Результат свёртки: 2x4

$$5 \cdot 3 + 3 \cdot 2 + (-1) \cdot (-1) + 6 \cdot (-1) + 2 \cdot 1 + 4 \cdot 4 + 3 \cdot 1 + 6 \cdot 3 + 0 \cdot 2 = 55$$

|   |                 |                |                  |   |   |
|---|-----------------|----------------|------------------|---|---|
| 6 | 5 <sub>3</sub>  | 3 <sub>2</sub> | -1 <sub>-1</sub> | 6 | 7 |
| 4 | 6 <sub>-1</sub> | 2 <sub>1</sub> | 4 <sub>4</sub>   | 6 | 2 |
| 1 | 3 <sub>1</sub>  | 6 <sub>3</sub> | 0 <sub>2</sub>   | 0 | 1 |
| 5 | 7               | 2              | 1                | 5 | 8 |

Изображение 4x6

×

|    |   |    |
|----|---|----|
| 3  | 2 | -1 |
| -1 | 1 | 4  |
| 1  | 3 | 2  |

Фильтр

=

|    |    |  |  |
|----|----|--|--|
| 57 | 55 |  |  |
|    |    |  |  |

Результат свёртки: 2x4

$$3 \cdot 3 + (-1) \cdot 2 + 6 \cdot (-1) + 2 \cdot (-1) + 4 \cdot 1 + 6 \cdot 4 + 6 \cdot 1 + 0 \cdot 3 + 0 \cdot 2 = 33$$

|   |   |                 |                 |                 |   |
|---|---|-----------------|-----------------|-----------------|---|
| 6 | 5 | 3 <sub>3</sub>  | -1 <sub>2</sub> | 6 <sub>-1</sub> | 7 |
| 4 | 6 | 2 <sub>-1</sub> | 4 <sub>1</sub>  | 6 <sub>4</sub>  | 2 |
| 1 | 3 | 6 <sub>1</sub>  | 0 <sub>3</sub>  | 0 <sub>2</sub>  | 1 |
| 5 | 7 | 2               | 1               | 5               | 8 |

Изображение 4x6

×

|    |   |    |
|----|---|----|
| 3  | 2 | -1 |
| -1 | 1 | 4  |
| 1  | 3 | 2  |

Фильтр

=

|    |    |    |  |
|----|----|----|--|
| 57 | 55 | 33 |  |
|    |    |    |  |

Результат свёртки: 2x4

$$(-1) \cdot 3 + 6 \cdot 2 + 7 \cdot (-1) + 4 \cdot (-1) + 6 \cdot 1 + 2 \cdot 4 + 0 \cdot 1 + 0 \cdot 3 + 1 \cdot 2 = 14$$

|   |   |   |                 |                |                 |
|---|---|---|-----------------|----------------|-----------------|
| 6 | 5 | 3 | -1 <sub>3</sub> | 6 <sub>2</sub> | 7 <sub>-1</sub> |
| 4 | 6 | 2 | 4 <sub>-1</sub> | 6 <sub>1</sub> | 2 <sub>4</sub>  |
| 1 | 3 | 6 | 0 <sub>1</sub>  | 0 <sub>3</sub> | 1 <sub>2</sub>  |
| 5 | 7 | 2 | 1               | 5              | 8               |

Изображение 4x6

×

|    |   |    |
|----|---|----|
| 3  | 2 | -1 |
| -1 | 1 | 4  |
| 1  | 3 | 2  |

Фильтр

=

|    |    |    |    |
|----|----|----|----|
| 57 | 55 | 33 | 14 |
|    |    |    |    |

Результат свёртки: 2x4

$$4 \cdot 3 + 6 \cdot 2 + 2 \cdot (-1) + 1 \cdot (-1) + 3 \cdot 1 + 6 \cdot 4 + 5 \cdot 1 + 7 \cdot 3 + 2 \cdot 2 = 78$$

|   |   |   |    |   |   |
|---|---|---|----|---|---|
| 6 | 5 | 3 | -1 | 6 | 7 |
| 4 | 6 | 2 | 4  | 6 | 2 |
| 1 | 3 | 6 | 0  | 0 | 1 |
| 5 | 7 | 2 | 1  | 5 | 8 |

Изображение 4x6

×

|    |   |    |
|----|---|----|
| 3  | 2 | -1 |
| -1 | 1 | 4  |
| 1  | 3 | 2  |

Фильтр

=

|    |    |    |    |
|----|----|----|----|
| 57 | 55 | 33 | 14 |
| 78 |    |    |    |

Результат свёртки: 2x4

$$6 \cdot 3 + 2 \cdot 2 + 4 \cdot (-1) + 3 \cdot (-1) + 6 \cdot 1 + 0 \cdot 4 + 7 \cdot 1 + 2 \cdot 3 + 1 \cdot 2 = 36$$

|   |                 |                |                 |   |   |
|---|-----------------|----------------|-----------------|---|---|
| 6 | 5               | 3              | -1              | 6 | 7 |
| 4 | 6 <sub>3</sub>  | 2 <sub>2</sub> | 4 <sub>-1</sub> | 6 | 2 |
| 1 | 3 <sub>-1</sub> | 6 <sub>1</sub> | 0 <sub>4</sub>  | 0 | 1 |
| 5 | 7 <sub>1</sub>  | 2 <sub>3</sub> | 1 <sub>2</sub>  | 5 | 8 |

Изображение 4x6

×

|    |   |    |
|----|---|----|
| 3  | 2 | -1 |
| -1 | 1 | 4  |
| 1  | 3 | 2  |

Фильтр

=

|    |    |    |    |
|----|----|----|----|
| 57 | 55 | 33 | 14 |
| 78 | 36 |    |    |

Результат свёртки: 2x4

$$2 \cdot 3 + 4 \cdot 2 + 6 \cdot (-1) + 6 \cdot (-1) + 0 \cdot 1 + 0 \cdot 4 + 2 \cdot 1 + 1 \cdot 3 + 5 \cdot 2 = 17$$

|   |   |                 |                |                 |   |
|---|---|-----------------|----------------|-----------------|---|
| 6 | 5 | 3               | -1             | 6               | 7 |
| 4 | 6 | 2 <sub>3</sub>  | 4 <sub>2</sub> | 6 <sub>-1</sub> | 2 |
| 1 | 3 | 6 <sub>-1</sub> | 0 <sub>1</sub> | 0 <sub>4</sub>  | 1 |
| 5 | 7 | 2 <sub>1</sub>  | 1 <sub>3</sub> | 5 <sub>2</sub>  | 8 |

Изображение 4x6

×

|    |   |    |
|----|---|----|
| 3  | 2 | -1 |
| -1 | 1 | 4  |
| 1  | 3 | 2  |

Фильтр

=

|    |    |    |    |
|----|----|----|----|
| 57 | 55 | 33 | 14 |
| 78 | 36 | 17 |    |

Результат свёртки: 2x4

$$4 \cdot 3 + 6 \cdot 2 + 2 \cdot (-1) + 0 \cdot (-1) + 0 \cdot 1 + 1 \cdot 4 + 1 \cdot 1 + 5 \cdot 3 + 8 \cdot 2 = 58$$

|   |   |   |                 |                |                 |
|---|---|---|-----------------|----------------|-----------------|
| 6 | 5 | 3 | -1              | 6              | 7               |
| 4 | 6 | 2 | 4 <sub>3</sub>  | 6 <sub>2</sub> | 2 <sub>-1</sub> |
| 1 | 3 | 6 | 0 <sub>-1</sub> | 0 <sub>1</sub> | 1 <sub>4</sub>  |
| 5 | 7 | 2 | 1 <sub>1</sub>  | 5 <sub>3</sub> | 8 <sub>2</sub>  |

Изображение 4x6

×

|    |   |    |
|----|---|----|
| 3  | 2 | -1 |
| -1 | 1 | 4  |
| 1  | 3 | 2  |

Фильтр

=

|    |    |    |    |
|----|----|----|----|
| 57 | 55 | 33 | 14 |
| 78 | 36 | 17 | 58 |

Результат свёртки: 2x4

С каждой стороны добавляется ряд из  $P$  нулей (или другой константы):

|                 |                |                 |   |    |   |   |   |
|-----------------|----------------|-----------------|---|----|---|---|---|
| 0 <sub>3</sub>  | 0 <sub>2</sub> | 0 <sub>-1</sub> | 0 | 0  | 0 | 0 | 0 |
| 0 <sub>-1</sub> | 6 <sub>1</sub> | 5 <sub>4</sub>  | 3 | -1 | 6 | 7 | 0 |
| 0 <sub>1</sub>  | 4 <sub>3</sub> | 6 <sub>2</sub>  | 2 | 4  | 6 | 2 | 0 |
| 0               | 1              | 3               | 6 | 0  | 0 | 1 | 0 |
| 0               | 5              | 7               | 2 | 1  | 5 | 8 | 0 |
| 0               | 0              | 0               | 0 | 0  | 0 | 0 | 0 |

Изображение 4x6,  $P = 1 \rightarrow 6x8$

×

|    |   |    |
|----|---|----|
| 3  | 2 | -1 |
| -1 | 1 | 4  |
| 1  | 3 | 2  |

Фильтр

=

|           |    |    |    |    |    |
|-----------|----|----|----|----|----|
| <b>50</b> | 37 | 14 | 46 | 61 | 13 |
| 44        | 57 | 55 | 33 | 14 | 31 |
| 44        | 78 | 36 | 17 | 58 | 52 |
| 32        | 13 | 20 | 37 | 35 | 5  |

Результат свёртки: 4x6

Каждый раз фильтр сдвигается не на  $stride$  элементов:

|                 |                |                 |   |    |   |   |   |
|-----------------|----------------|-----------------|---|----|---|---|---|
| 0 <sub>3</sub>  | 0 <sub>2</sub> | 0 <sub>-1</sub> | 0 | 0  | 0 | 0 | 0 |
| 0 <sub>-1</sub> | 6 <sub>1</sub> | 5 <sub>4</sub>  | 3 | -1 | 6 | 7 | 0 |
| 0 <sub>1</sub>  | 4 <sub>3</sub> | 6 <sub>2</sub>  | 2 | 4  | 6 | 2 | 0 |
| 0               | 1              | 3               | 6 | 0  | 0 | 1 | 0 |
| 0               | 5              | 7               | 2 | 1  | 5 | 8 | 0 |
| 0               | 0              | 0               | 0 | 0  | 0 | 0 | 0 |

Изображение 4x6,  $P = 1$ ,  $S = 2$

×

|    |   |    |
|----|---|----|
| 3  | 2 | -1 |
| -1 | 1 | 4  |
| 1  | 3 | 2  |

Фильтр

=

|    |  |  |
|----|--|--|
| 50 |  |  |
|    |  |  |

Результат свёртки: 2x3

Каждый раз фильтр сдвигается не на  $stride$  элементов:

|   |   |                 |                |                 |   |   |   |
|---|---|-----------------|----------------|-----------------|---|---|---|
| 0 | 0 | 0 <sub>3</sub>  | 0 <sub>2</sub> | 0 <sub>-1</sub> | 0 | 0 | 0 |
| 0 | 6 | 5 <sub>-1</sub> | 3 <sub>1</sub> | -1 <sub>4</sub> | 6 | 7 | 0 |
| 0 | 4 | 6 <sub>1</sub>  | 2 <sub>3</sub> | 4 <sub>2</sub>  | 6 | 2 | 0 |
| 0 | 1 | 3               | 6              | 0               | 0 | 1 | 0 |
| 0 | 5 | 7               | 2              | 1               | 5 | 8 | 0 |
| 0 | 0 | 0               | 0              | 0               | 0 | 0 | 0 |

Изображение 4x6,  $P = 1$ ,  $S = 2$

×

|    |   |    |
|----|---|----|
| 3  | 2 | -1 |
| -1 | 1 | 4  |
| 1  | 3 | 2  |

Фильтр

=

|    |    |  |
|----|----|--|
| 50 | 14 |  |
|    |    |  |

Результат свёртки: 2x3

Каждый раз фильтр сдвигается не на  $stride$  элементов:

|   |   |   |   |                  |                |                 |   |
|---|---|---|---|------------------|----------------|-----------------|---|
| 0 | 0 | 0 | 0 | 0 <sub>3</sub>   | 0 <sub>2</sub> | 0 <sub>-1</sub> | 0 |
| 0 | 6 | 5 | 3 | -1 <sub>-1</sub> | 6 <sub>1</sub> | 7 <sub>4</sub>  | 0 |
| 0 | 4 | 6 | 2 | 4 <sub>1</sub>   | 6 <sub>3</sub> | 2 <sub>2</sub>  | 0 |
| 0 | 1 | 3 | 6 | 0                | 0              | 1               | 0 |
| 0 | 5 | 7 | 2 | 1                | 5              | 8               | 0 |
| 0 | 0 | 0 | 0 | 0                | 0              | 0               | 0 |

Изображение 4x6,  $P = 1$ ,  $S = 2$

×

|    |   |    |
|----|---|----|
| 3  | 2 | -1 |
| -1 | 1 | 4  |
| 1  | 3 | 2  |

Фильтр

=

|    |    |    |
|----|----|----|
| 50 | 14 | 61 |
|    |    |    |

Результат свёртки: 2x3

Каждый раз фильтр сдвигается не на stride элементов:

|   |   |   |   |    |   |   |   |
|---|---|---|---|----|---|---|---|
| 0 | 0 | 0 | 0 | 0  | 0 | 0 | 0 |
| 0 | 6 | 5 | 3 | -1 | 6 | 7 | 0 |
| 0 | 4 | 6 | 2 | 4  | 6 | 2 | 0 |
| 0 | 1 | 3 | 6 | 0  | 0 | 1 | 0 |
| 0 | 5 | 7 | 2 | 1  | 5 | 8 | 0 |
| 0 | 0 | 0 | 0 | 0  | 0 | 0 | 0 |

Изображение 4x6, P = 1, S = 2

×

|    |   |    |
|----|---|----|
| 3  | 2 | -1 |
| -1 | 1 | 4  |
| 1  | 3 | 2  |

Фильтр

=

|    |    |    |
|----|----|----|
| 50 | 14 | 61 |
| 44 |    |    |

Результат свёртки: 2x3

Каждый раз фильтр сдвигается не на  $stride$  элементов:

|   |   |   |   |    |   |   |   |
|---|---|---|---|----|---|---|---|
| 0 | 0 | 0 | 0 | 0  | 0 | 0 | 0 |
| 0 | 6 | 5 | 3 | -1 | 6 | 7 | 0 |
| 0 | 4 | 6 | 2 | 4  | 6 | 2 | 0 |
| 0 | 1 | 3 | 6 | 0  | 0 | 1 | 0 |
| 0 | 5 | 7 | 2 | 1  | 5 | 8 | 0 |
| 0 | 0 | 0 | 0 | 0  | 0 | 0 | 0 |

Изображение  $4 \times 6$ ,  $P = 1$ ,  $S = 2$

$$\begin{array}{|c|c|c|} \hline 3 & 2 & -1 \\ \hline -1 & 1 & 4 \\ \hline 1 & 3 & 2 \\ \hline \end{array}$$

Фильтр

$$\begin{array}{|c|c|c|} \hline 50 & 14 & 61 \\ \hline 44 & 36 & \\ \hline \end{array}$$

Результат свёртки:  $2 \times 3$

Каждый раз фильтр сдвигается не на *stride* элементов:

|   |   |   |   |    |   |   |   |
|---|---|---|---|----|---|---|---|
| 0 | 0 | 0 | 0 | 0  | 0 | 0 | 0 |
| 0 | 6 | 5 | 3 | -1 | 6 | 7 | 0 |
| 0 | 4 | 6 | 2 | 4  | 6 | 2 | 0 |
| 0 | 1 | 3 | 6 | 0  | 0 | 1 | 0 |
| 0 | 5 | 7 | 2 | 1  | 5 | 8 | 0 |
| 0 | 0 | 0 | 0 | 0  | 0 | 0 | 0 |

Изображение 4x6, P = 1, S = 2

×

|    |   |    |
|----|---|----|
| 3  | 2 | -1 |
| -1 | 1 | 4  |
| 1  | 3 | 2  |

Фильтр

=

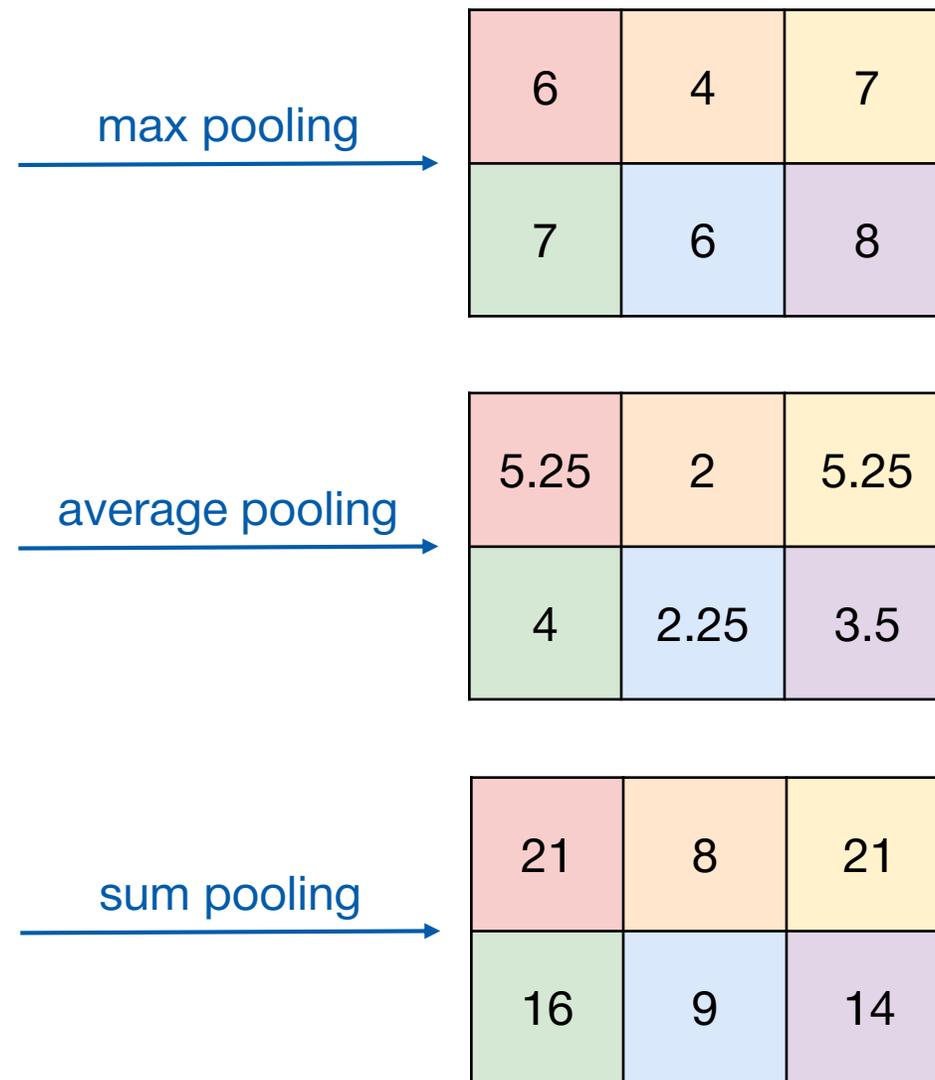
|    |    |    |
|----|----|----|
| 50 | 14 | 61 |
| 44 | 36 | 58 |

Результат свёртки: 2x3

В настоящее время используется редко. Чаще используется слой свёртки с шагом (stride) больше единицы.

|   |   |   |    |   |   |
|---|---|---|----|---|---|
| 6 | 5 | 3 | -1 | 6 | 7 |
| 4 | 6 | 2 | 4  | 6 | 2 |
| 1 | 3 | 6 | 0  | 0 | 1 |
| 5 | 7 | 2 | 1  | 5 | 8 |

Пулинг размерности 2x2 с шагом 2



Если к изображению размера  $W_{in} \times H_{in}$  применить свертку / пулинг размера  $F_w \times F_h$  с дополнением нулями  $P$  и шагом  $S$ , то размер выходного изображения будет равен  $W_{out} \times H_{out}$ , где:

$$W_{out} = \left\lfloor \frac{W_{in} + 2P - F_w}{S} \right\rfloor + 1$$
$$H_{out} = \left\lfloor \frac{H_{in} + 2P - F_h}{S} \right\rfloor + 1$$

Примеры:

$$F = 3 \times 3, P = 1, S = 1:$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2 - 3}{1} \right\rfloor + 1 = W_{in}$$

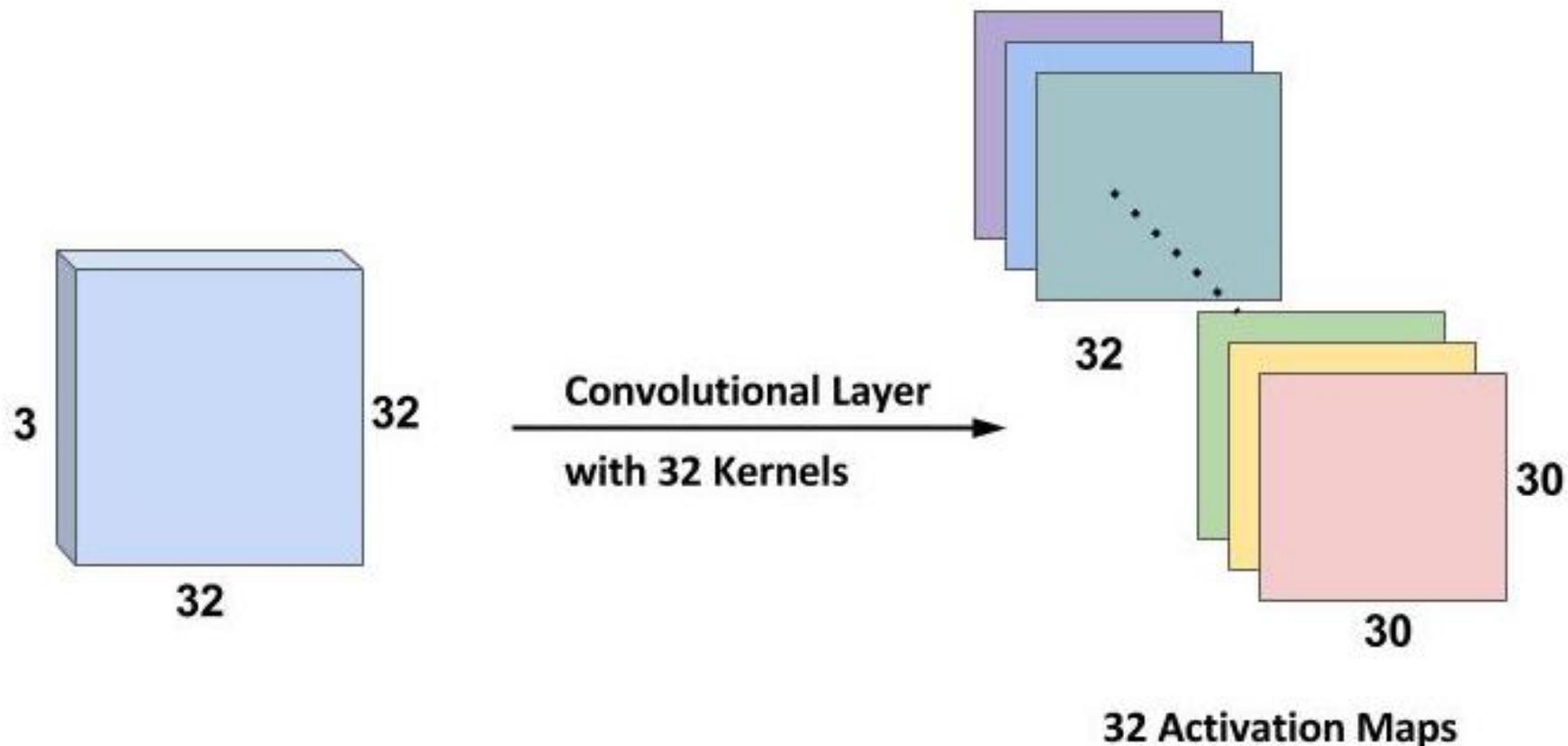
$$H_{out} = \left\lfloor \frac{H_{in} + 2 - 3}{1} \right\rfloor + 1 = H_{in}$$

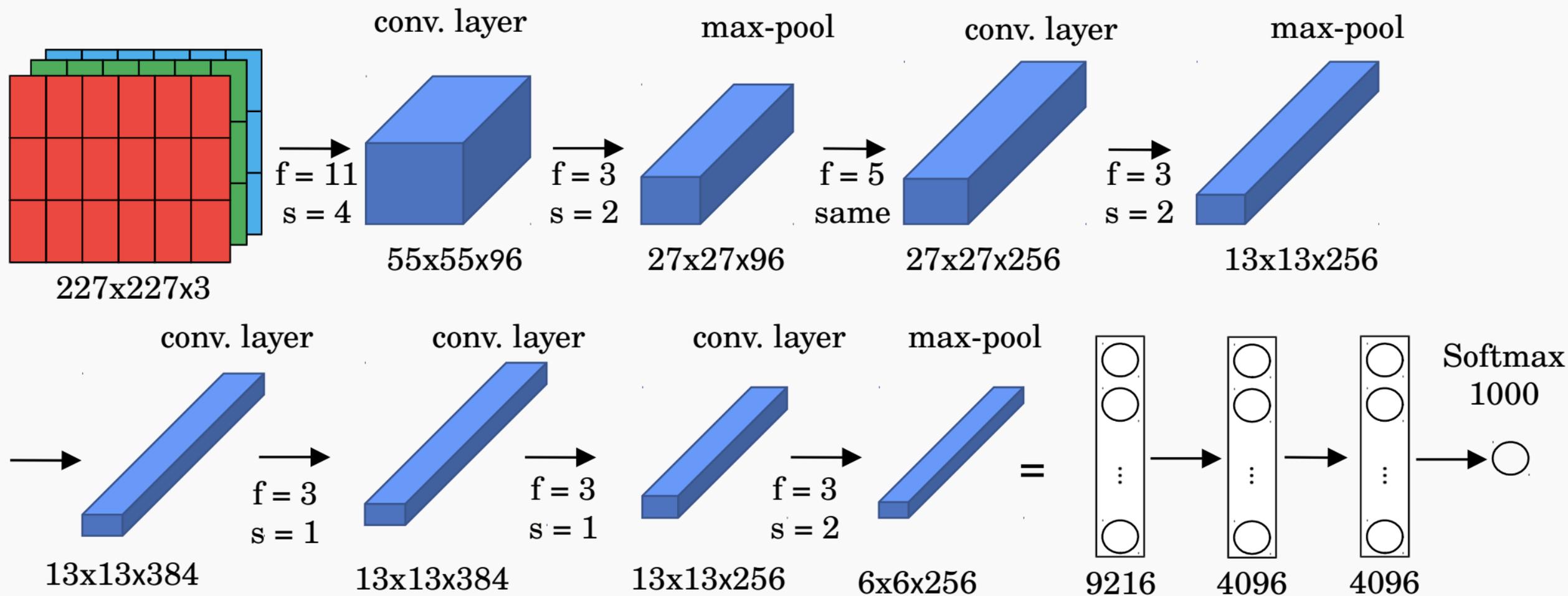
$$F = 5 \times 3, P = 3, S = 2:$$

$$W_{out} = \left\lfloor \frac{W_{in} + 6 - 5}{2} \right\rfloor + 1 = \left\lfloor \frac{W_{in} + 1}{2} \right\rfloor + 1$$

$$H_{out} = \left\lfloor \frac{H_{in} + 6 - 3}{2} \right\rfloor + 1 = \left\lfloor \frac{H_{in} + 3}{2} \right\rfloor + 1$$

В свёрточном слое находится не один, а сразу несколько фильтров. Каждый из них получает из входного изображения плоскую матрицу, а для получения выходного тензора получаемые матрицы конкатенируются:





# Вопросы?