Введение в нейронные сети

Анастасия Миллер

5 марта 2018

Постановка задачи

X – множество *объектов*,
Y – множество *возможных ответов*,
$f : X \to Y$ – функция, значения которой известны лишь на конечном множестве объектов

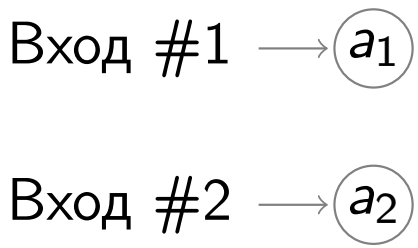
$$\{x_i, y_i | f(x_i) = y_i\}_{i=1}^{n} .$$

Задача машинного обучения: по известному множеству $\{x_i, y_i\}_{i=1}^{n}$ восстановить функцию f.

Как выглядят нейронные сети? Самая простая модель

Как выглядят нейронные сети? Самая простая модель

Как выглядят нейронные сети? Самая простая модель



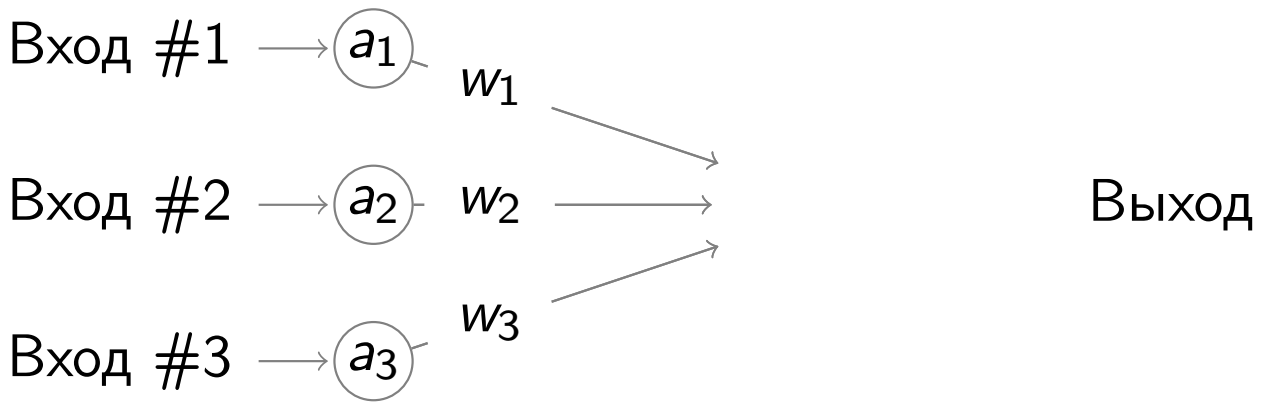
Как выглядят нейронные сети? Самая простая модель

Как выглядят нейронные сети? Самая простая модель

Как выглядят нейронные сети? Самая простая модель

Перцептрон

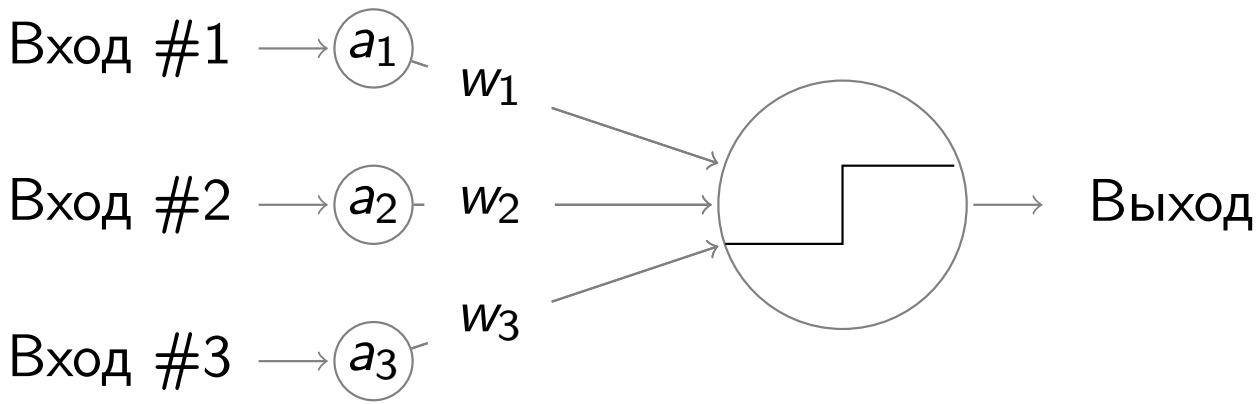
$$\hat{f}(x) = \begin{cases} 1, & \text{если } \sum_{j=1}^{K} w_j \cdot x^{(j)} + b > 0 \\ 0, & \text{если } \sum_{j=1}^{K} w_j \cdot x^{(j)} + b \leq 0 \end{cases}.$$



Как выглядят нейронные сети? Самая простая модель

Перцептрон

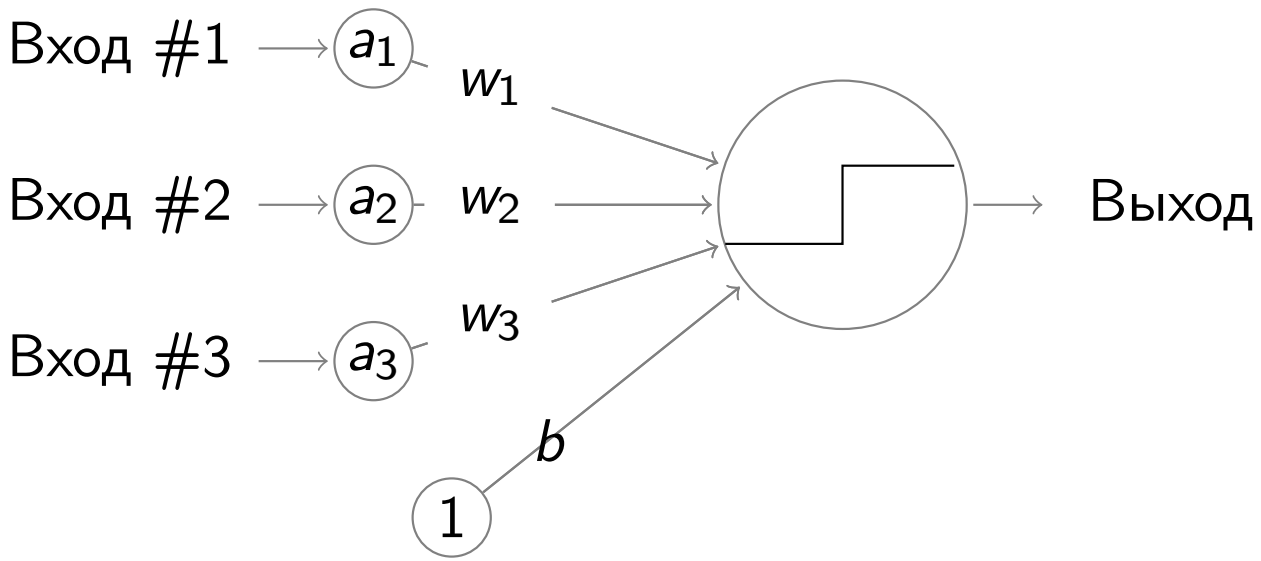
$$\hat{f}(x) = \begin{cases} 1, & \text{если } \sum_{j=1}^{K} w_j \cdot x^{(j)} + b > 0 \\ 0, & \text{если } \sum_{j=1}^{K} w_j \cdot x^{(j)} + b \leq 0 \end{cases}.$$



Как выглядят нейронные сети? Самая простая модель

Перцептрон

$$\hat{f}(x) = \begin{cases} 1, & \text{если } \sum_{j=1}^{K} w_j \cdot x^{(j)} + b > 0 \\ 0, & \text{если } \sum_{j=1}^{K} w_j \cdot x^{(j)} + b \leq 0 \end{cases}.$$



Как выглядят нейронные сети? Самая простая модель

Перцептрон

$$\hat{f}(x) = \begin{cases} 1, & \text{если } \sum_{j=1}^{K} w_j \cdot x^{(j)} + b > 0 \\ 0, & \text{если } \sum_{j=1}^{K} w_j \cdot x^{(j)} + b \leq 0 \end{cases}.$$

Оцениваем качество работы $\hat{f}$:

$$\hat{y}_i = \hat{f}(x_i)$$

Как выглядят нейронные сети? Самая простая модель

Перцептрон

$$\hat{f}(x) = \begin{cases} 1, & \text{если } \sum_{j=1}^{K} w_j \cdot x^{(j)} + b > 0 \\ 0, & \text{если } \sum_{j=1}^{K} w_j \cdot x^{(j)} + b \leq 0 \end{cases}.$$

Оцениваем качество работы $\hat{f}$:

$$\hat{y}_i = \hat{f}(x_i)$$

$$J\left(\hat{f}, \{x_i\}, \{y_i\}\right) =$$

Как выглядят нейронные сети? Самая простая модель

Перцептрон

$$\hat{f}(x) = \begin{cases} 1, & \text{если } \sum_{j=1}^{K} w_j \cdot x^{(j)} + b > 0 \\ 0, & \text{если } \sum_{j=1}^{K} w_j \cdot x^{(j)} + b \leq 0 \end{cases}.$$

Оцениваем качество работы $\hat{f}$:

$$\hat{y}_i = \hat{f}(x_i)$$

$$J\left(\hat{f}, \{x_i\}, \{y_i\}\right) = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \ln \hat{y}_i + (1 - y_i)\ln\left(1 - \hat{y}_i\right)\right)$$

Как выглядят нейронные сети? Самая простая модель

Перцептрон

$$\hat{f}(x) = \begin{cases} 1, & \text{если } \sum_{j=1}^{K} w_j \cdot x^{(j)} + b > 0 \\ 0, & \text{если } \sum_{j=1}^{K} w_j \cdot x^{(j)} + b \leq 0 \end{cases}.$$

Оцениваем качество работы $\hat{f}$:

$$\hat{y}_i = \hat{f}(x_i)$$

$$J\left(\hat{f}, \{x_i\}, \{y_i\}\right) = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \ln \hat{y}_i + (1 - y_i)\ln\left(1 - \hat{y}_i\right)\right)$$

Что делать, если качество плохое?

Как выглядят нейронные сети? Самая простая модель

Перцептрон

$$\hat{f}(x) = \begin{cases} 1, & \text{если } \sum_{j=1}^{K} w_j \cdot x^{(j)} + b > 0 \\ 0, & \text{если } \sum_{j=1}^{K} w_j \cdot x^{(j)} + b \leq 0 \end{cases}.$$

Оцениваем качество работы $\hat{f}$:

$$\hat{y}_i = \hat{f}(x_i)$$

$$J\left(\hat{f}, \{x_i\}, \{y_i\}\right) = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \ln \hat{y}_i + (1 - y_i)\ln\left(1 - \hat{y}_i\right)\right)$$

Что делать, если качество плохое?
Менять $\hat{f}$ так, чтобы качество было лучше!

Градиентный спуск

Задача: найти $\arg\min_x J(x)$, J – дифференцируемая функция.
Решение:

- $\nabla J(x)$ – градиент J в точке x – указывает направление наискорейшего роста;
- в достаточно малой окрестности x $-\nabla J(x)$ указывает направление наискорейшего спуска.

Градиентный спуск

Задача: найти $\arg\min_x J(x)$, J – дифференцируемая функция.
Решение:

- $\nabla J(x)$ – градиент J в точке x – указывает направление наискорейшего роста;
- в достаточно малой окрестности x $-\nabla J(x)$ указывает направление наискорейшего спуска.

Алгоритм: Градиентный спуск

Вход: функция J, размер шага λ, условие остановки ε

Градиентный спуск

Задача: найти $\arg\min_x J(x)$, J – дифференцируемая функция.
Решение:

- $\nabla J(x)$ – градиент J в точке x – указывает направление наискорейшего роста;
- в достаточно малой окрестности x $-\nabla J(x)$ указывает направление наискорейшего спуска.

Алгоритм: Градиентный спуск

Вход: функция J, размер шага λ, условие остановки ε
$\theta^{[0]}$ – точка начала поиска ;
$\theta^{[1]} = \theta^{[0]} - \lambda \cdot \nabla J(\theta^{[0]})$;

Градиентный спуск

Задача: найти $\arg\min_x J(x)$, J – дифференцируемая функция.
Решение:

- $\nabla J(x)$ – градиент J в точке x – указывает направление наискорейшего роста;
- в достаточно малой окрестности x $-\nabla J(x)$ указывает направление наискорейшего спуска.

Алгоритм: Градиентный спуск

Вход: функция J, размер шага λ, условие остановки ε
$\theta^{[0]}$ – точка начала поиска ;
$\theta^{[1]} = \theta^{[0]} - \lambda \cdot \nabla J(\theta^{[0]})$;
$i = 1$;
while $\left|\theta^{[i]} - \theta^{[i-1]}\right| > \varepsilon$ **do**
  $\theta^{[i+1]} = \theta^{[i]} - \lambda \cdot \nabla J(\theta^{[i]})$;
  $i = i + 1$;
Выход: $\theta^{[i]}$

Как выглядят нейронные сети? Самая простая модель

Перцептрон

$$\hat{f}(x) = \begin{cases} 1, & \text{если } \sum_{j=1}^{K} w_j \cdot x^{(j)} + b > 0 \\ 0, & \text{если } \sum_{j=1}^{K} w_j \cdot x^{(j)} + b \leq 0 \end{cases}.$$

Оцениваем качество работы $\hat{f}$:

$$\hat{y}_i = \hat{f}(x_i)$$

$$J\left(\hat{f}, \{x_i\}, \{y_i\}\right) = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \ln \hat{y}_i + (1 - y_i)\ln\left(1 - \hat{y}_i\right)\right)$$

Что делать, если качество плохое?
Менять $\hat{f}$ так, чтобы качество было лучше!

Самая простая модель. Настройка весов.

Перцептрон

$$\hat{f}(x) = \text{greater than } 0 \left(\sum_{j=1}^{K} w_j \cdot x^{(j)} + b \right).$$

Хотим найти $\arg\min_{\hat{f}} J\left(\hat{f}, \{x_i\}, \{y_i\}\right) =$

Самая простая модель. Настройка весов.

Перцептрон

$$\hat{f}(x) = \text{greater than } 0 \left(\sum_{j=1}^{K} w_j \cdot x^{(j)} + b \right).$$

Хотим найти $\arg\min_{\hat{f}} J\left(\hat{f}, \{x_i\}, \{y_i\}\right) =$

$$\arg\min_{\{w_j\}} J\left(\{w_j\}, \{x_i\}, \{y_i\}\right).$$

Самая простая модель. Настройка весов.

Перцептрон

$$\hat{f}(x) = \text{greater than } 0 \left(\sum_{j=1}^{K} w_j \cdot x^{(j)} + b \right).$$

Хотим найти $\arg\min_{\hat{f}} J\left(\hat{f}, \{x_i\}, \{y_i\}\right) =$

$$\arg\min_{\{w_j\}} J\left(\{w_j\}, \{x_i\}, \{y_i\}\right).$$

Самая простая модель. Настройка весов.

Перцептрон

$$\hat{f}(x) = \sigma\left(\sum_{j=1}^{K} w_j \cdot x^{(j)} + b\right).$$

Хотим найти $\arg\min_{\hat{f}} J\left(\hat{f}, \{x_i\}, \{y_i\}\right) =$

$$\arg\min_{\{w_j\}} J\left(\{w_j\}, \{x_i\}, \{y_i\}\right).$$

Самая простая модель. Настройка весов.

Перцептрон

$$\hat{f}(x) = \sigma\left(\sum_{j=1}^{K} w_j \cdot x^{(j)} + b\right).$$

Хотим найти $\arg\min_{\hat{f}} J\left(\hat{f}, \{x_i\}, \{y_i\}\right) =$

$$\arg\min_{\{w_j\}} J\left(\{w_j\}, \{x_i\}, \{y_i\}\right).$$

$$\begin{cases} J(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \tilde{J}(y_i, \hat{y}_i) \\ \tilde{J}(y_i, \hat{y}_i) = y_i \ln \hat{y}_i + (1 - y_i)\ln(1 - \hat{y}_i) \\ \hat{y}_i = \hat{f}(x_i) = \sigma\left(\sum_{j=1}^{K} w_j x_i^{(j)} + b\right) \end{cases}$$

Самая простая модель. Настройка весов.

Перцептрон

$$\hat{f}(x) = \sigma\left(\sum_{j=1}^{K} w_j \cdot x^{(j)} + b\right).$$

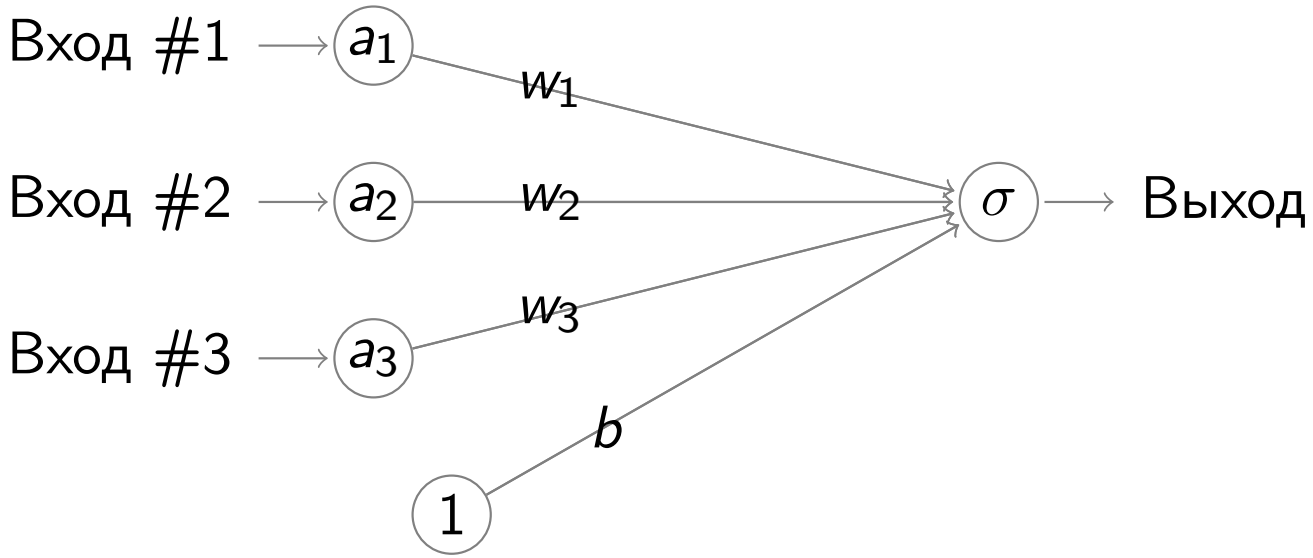
Хотим найти $\arg\min_{\hat{f}} J\left(\hat{f}, \{x_i\}, \{y_i\}\right) =$

$$\arg\min_{\{w_j\}} J\left(\{w_j\}, \{x_i\}, \{y_i\}\right).$$

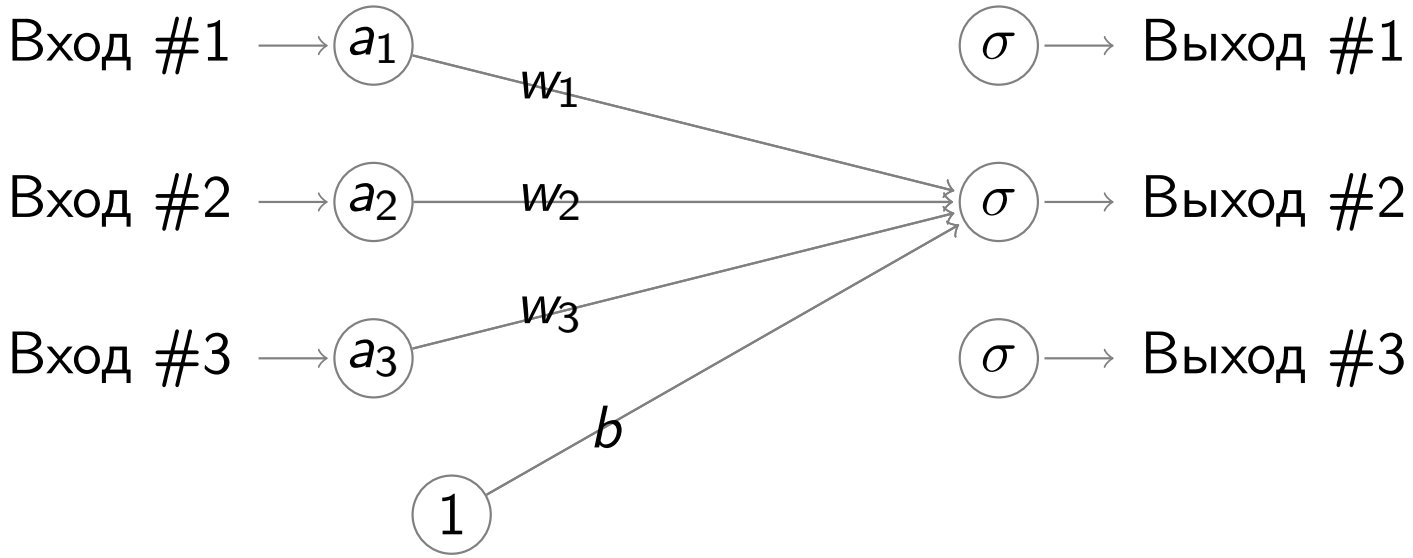
$$\begin{cases} J(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \tilde{J}(y_i, \hat{y}_i) \\ \tilde{J}(y_i, \hat{y}_i) = y_i \ln \hat{y}_i + (1 - y_i)\ln(1 - \hat{y}_i) \\ \hat{y}_i = \hat{f}(x_i) = \sigma\left(\sum_{j=1}^{K} w_j x_i^{(j)} + b\right) \end{cases}$$

$$\nabla J(\theta) = \left(\frac{\partial J}{\partial w_1}, \ldots, \frac{\partial J}{\partial w_K}, \frac{\partial J}{\partial b}\right)^T$$

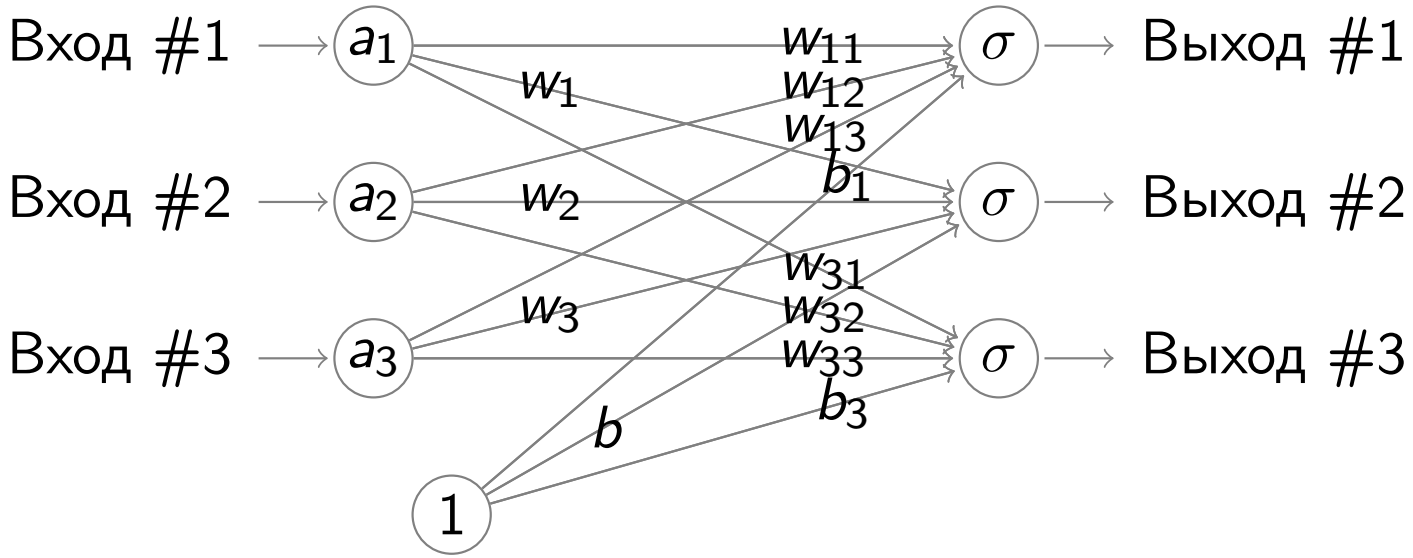
Усложняем модель: многоклассовая классификация



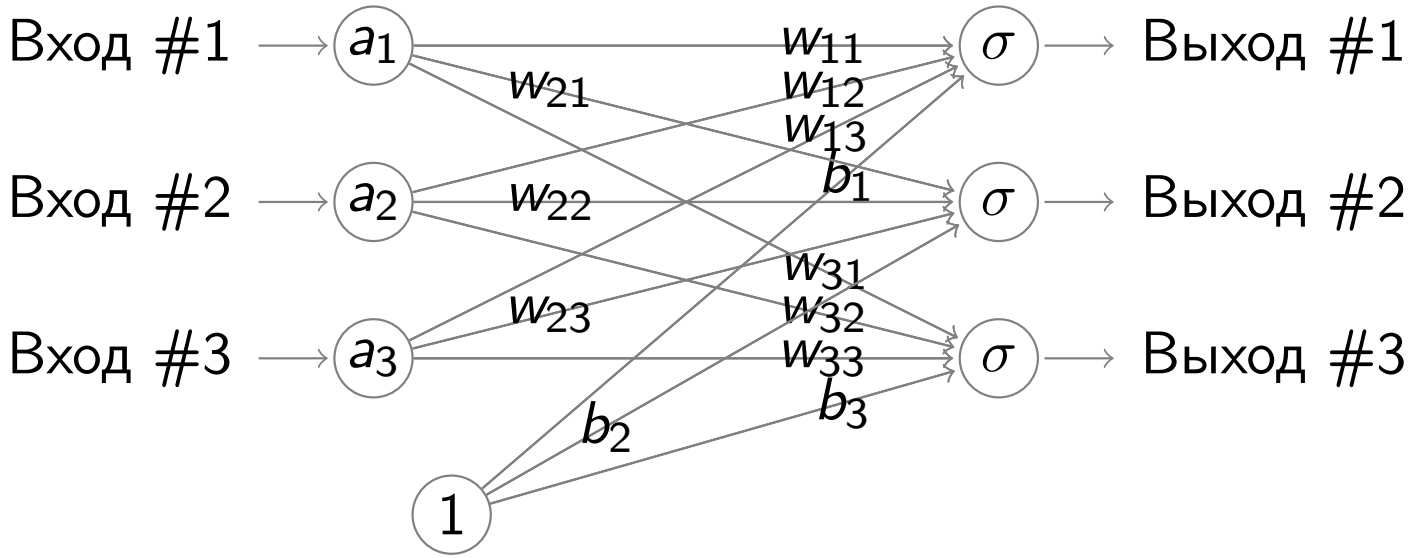
Усложняем модель: многоклассовая классификация



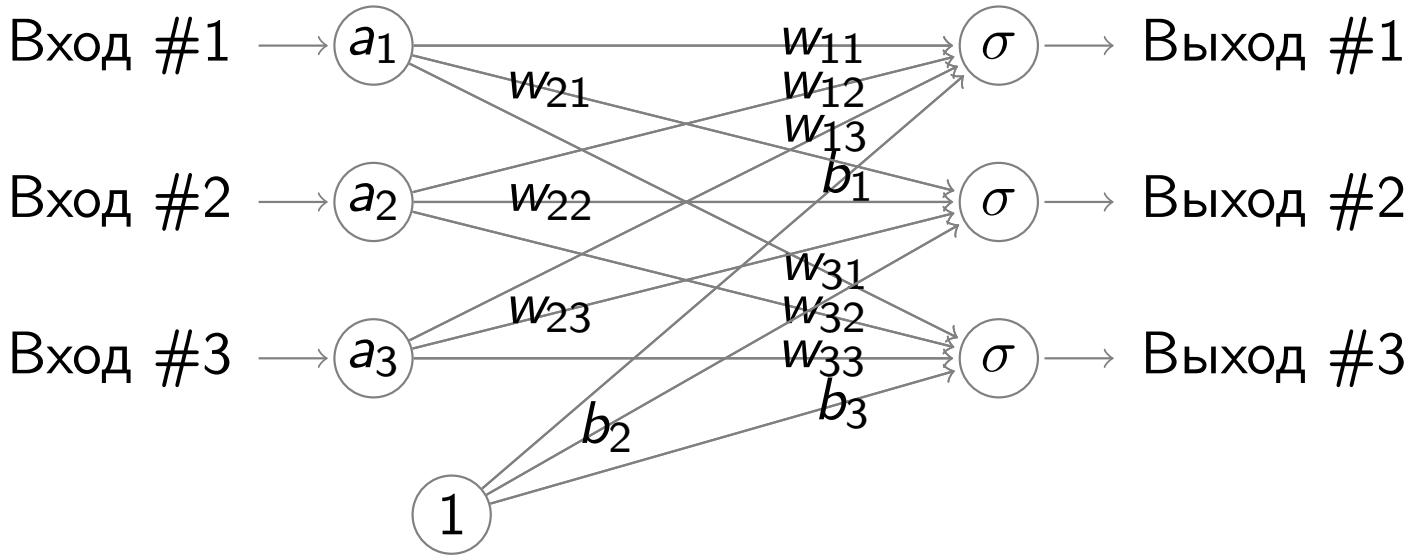
Усложняем модель: многоклассовая классификация



Усложняем модель: многоклассовая классификация



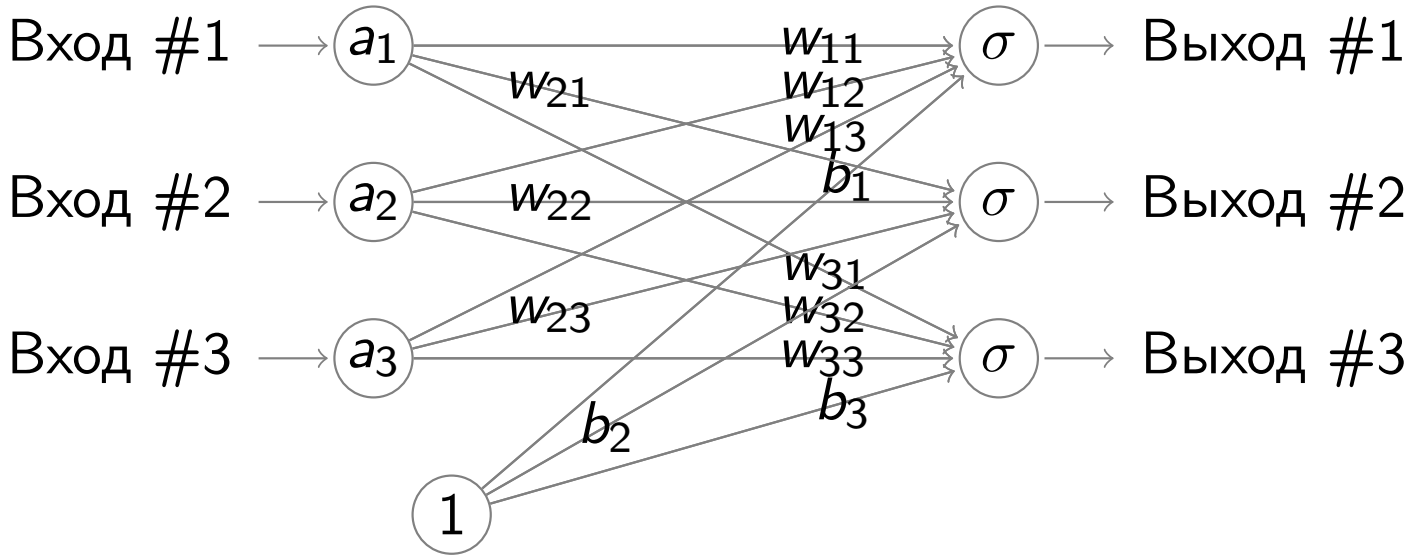
Усложняем модель: многоклассовая классификация



Функция потерь для K-классовой классификации:

$$J\left(\hat{f}, \{x_i\}, \{y_i\}\right) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{K}\left(y_i^{(j)} \ln \hat{y}_i^{(j)} + \left(1 - y_i^{(j)}\right) \ln\left(1 - \hat{y}_i^{(j)}\right)\right)$$

Усложняем модель: многоклассовая классификация



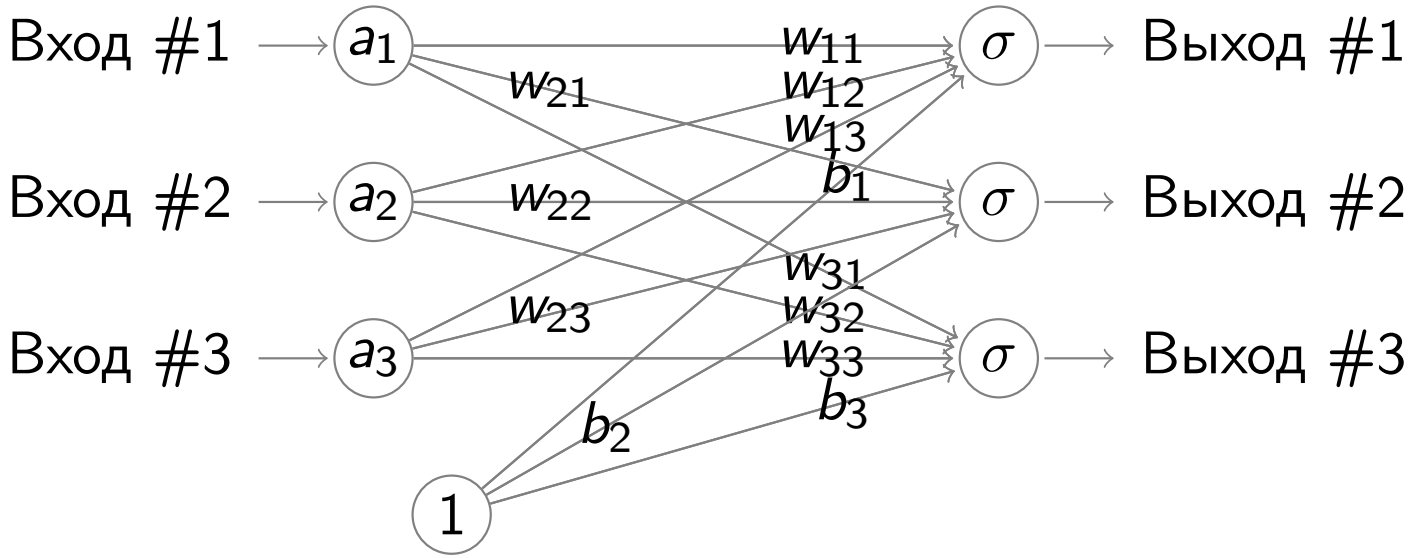
Функция потерь для K-классовой классификации:

$$J\left(\hat{f}, \{x_i\}, \{y_i\}\right) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{K}\left(y_i^{(j)} \ln \hat{y}_i^{(j)} + \left(1 - y_i^{(j)}\right) \ln\left(1 - \hat{y}_i^{(j)}\right)\right)$$

Вопрос

Что нужно изменить в алгоритме настройки весов?

Усложняем модель: многоклассовая классификация



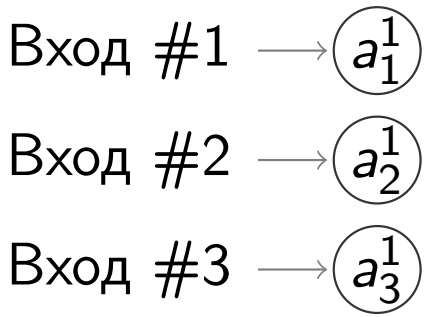
Функция потерь для K-классовой классификации:

$$J\left(\hat{f}, \{x_i\}, \{y_i\}\right) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{K}\left(y_i^{(j)} \ln \hat{y}_i^{(j)} + \left(1 - y_i^{(j)}\right) \ln\left(1 - \hat{y}_i^{(j)}\right)\right)$$

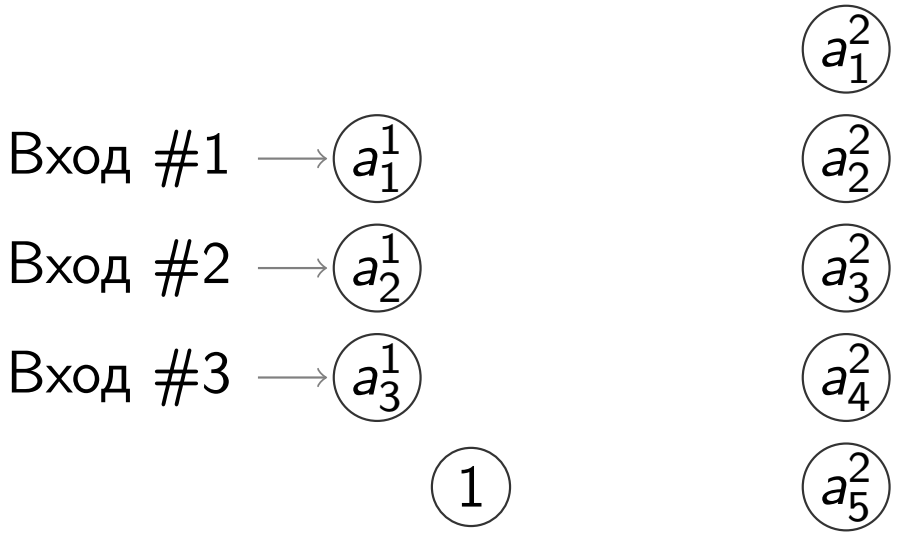
Вопрос

Что нужно изменить в алгоритме настройки весов? Ничего.

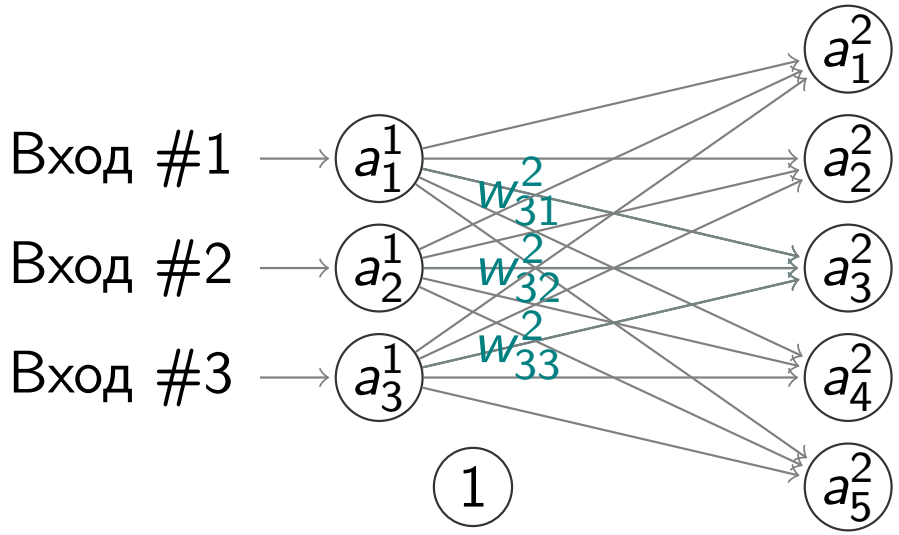
Усложняем модель: больше слоёв



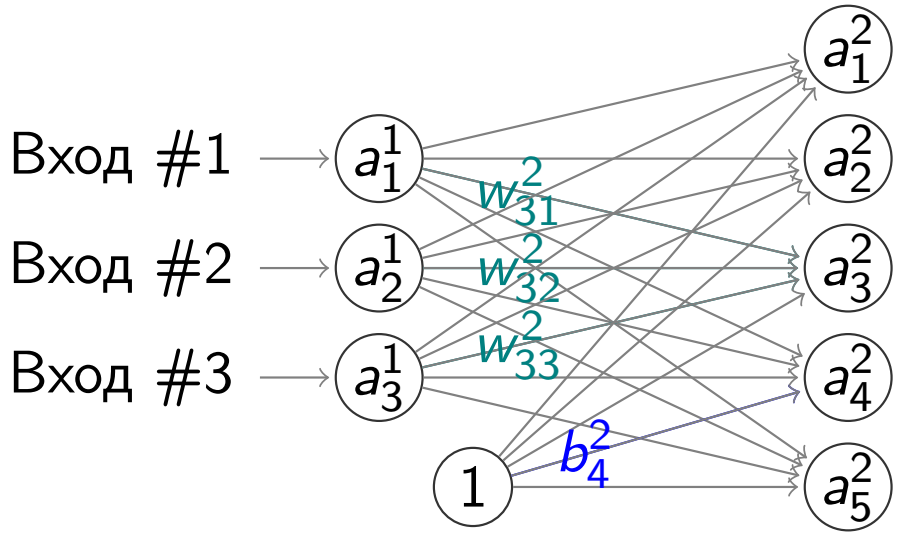
Усложняем модель: больше слоёв



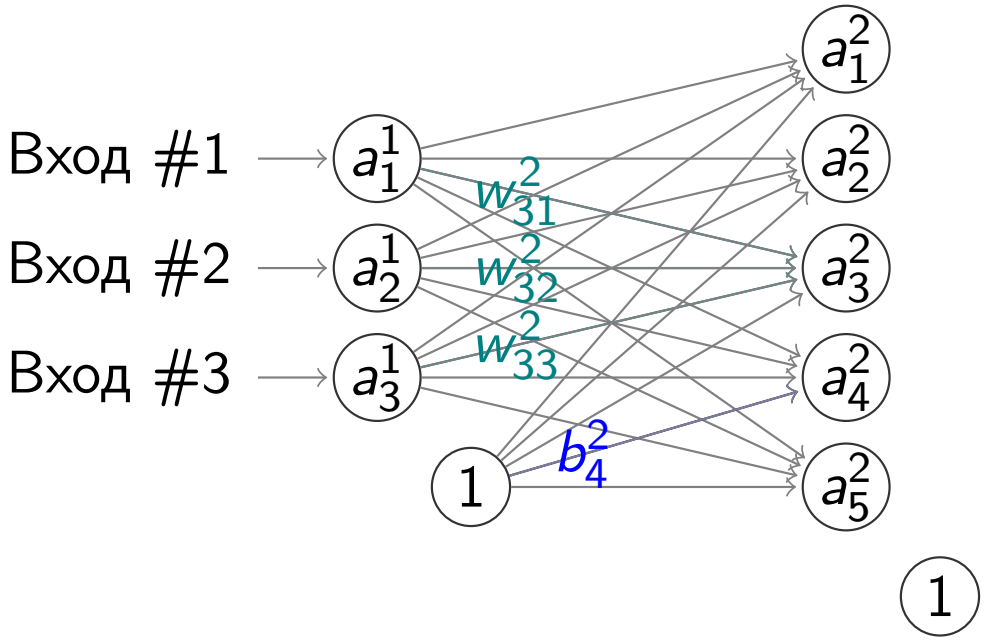
Усложняем модель: больше слоёв

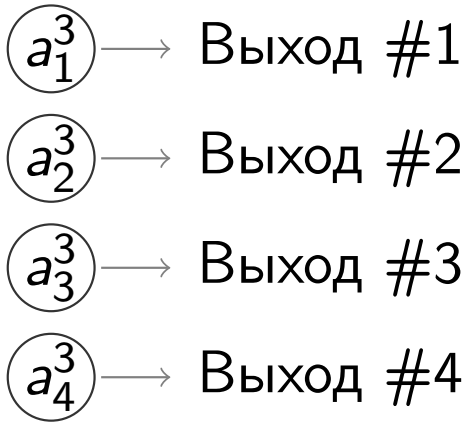


Усложняем модель: больше слоёв

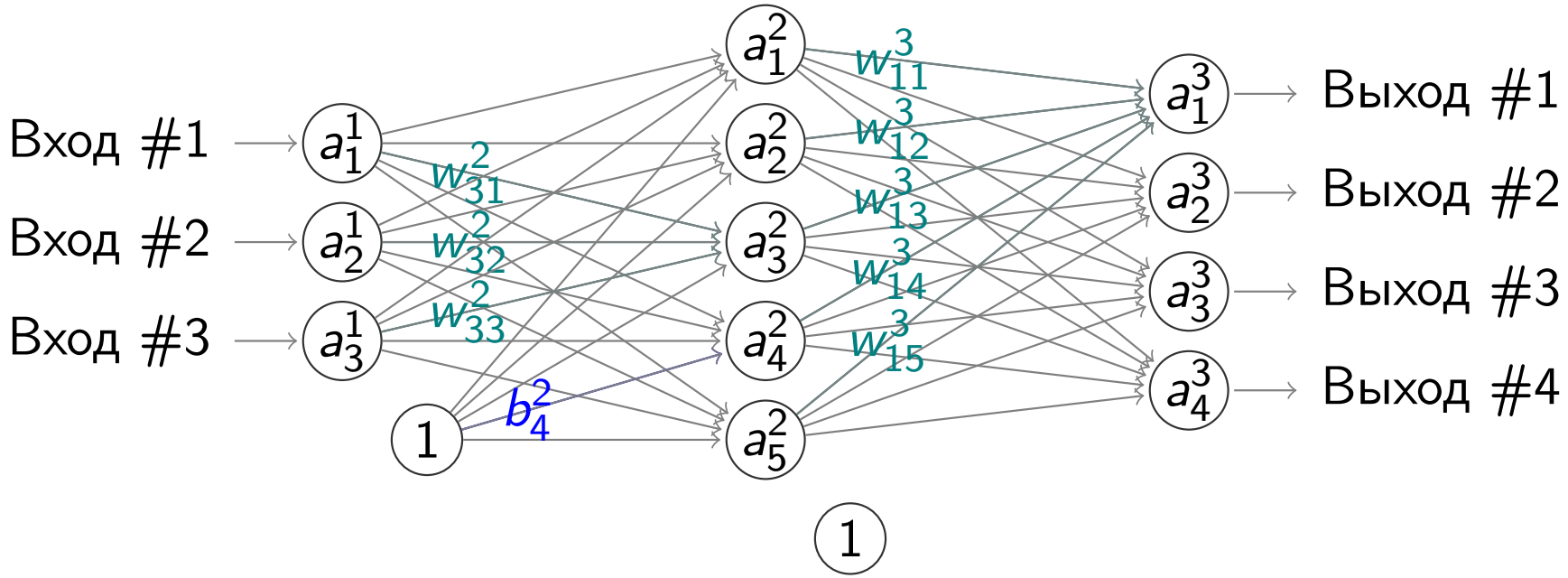


Усложняем модель: больше слоёв

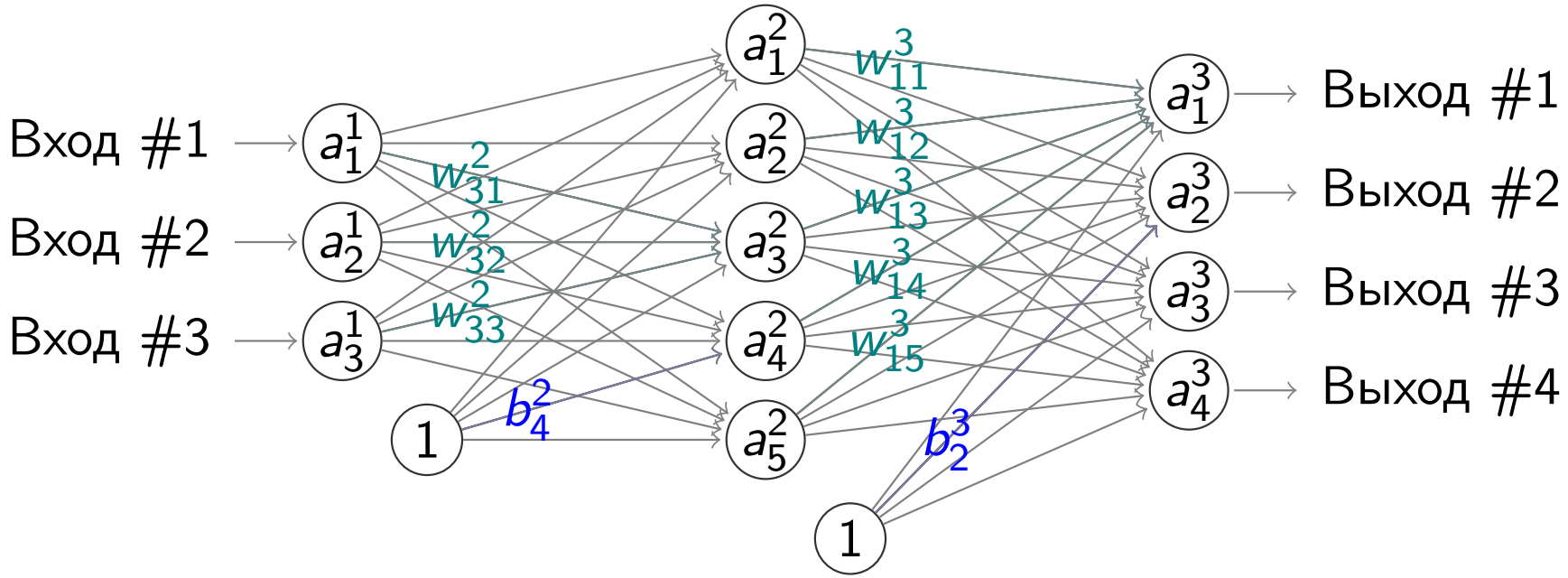




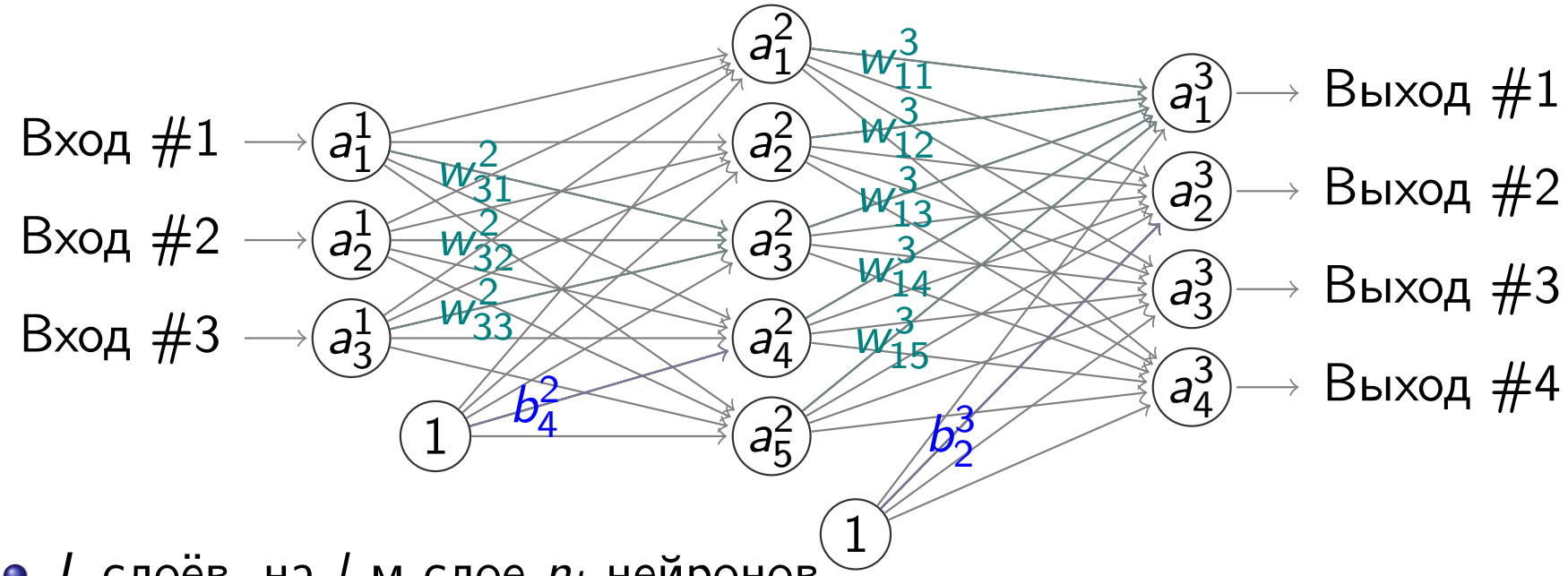
Усложняем модель: больше слоёв



Усложняем модель: больше слоёв



Усложняем модель: больше слоёв



- L слоёв, на l-м слое n_l нейронов,
- w_{jk}^l – вес выхода k-го нейрона $l-1$-го слоя в j-м нейроне l-го слоя,
- $W^l = \left\{ w_{jk}^l \right\}$ – матрица весов между $l-1$ и l-м слоем размера $n_l \times n_{l-1}$,
- b_j^l – смещение, добавляемое ко взвешенной сумме нейронов $l-1$-го слоя в j-м нейроне l-го слоя,
- $s_j^l = \sum_{k=1}^{n_{l-1}} w_{jk}^l a_k^{l-1} + b_j^l, \quad a_j^l = \sigma\left(s_j^l\right)$

Усложняем модель: больше слоёв

- $a_j^l = \sigma\left(\sum_{k=1}^{n_{l-1}} w_{jk}^l a_k^{l-1} + b_j^l\right)$,

Усложняем модель: больше слоёв

- $a_j^l = \sigma\left(\sum_{k=1}^{n_{l-1}} w_{jk}^l a_k^{l-1} + b_j^l\right)$,
- $a^l = \sigma\left(W^l \times a^{l-1} + b^l\right)$

Усложняем модель: больше слоёв

- $a_j^l = \sigma\left(\sum_{k=1}^{n_{l-1}} w_{jk}^l a_k^{l-1} + b_j^l\right)$,
- $a^l = \sigma\left(W^l \times a^{l-1} + b^l\right)$

Усложняем модель: больше слоёв

- $a^l_j = \sigma\left(\sum_{k=1}^{n_{l-1}} w^l_{jk} a^{l-1}_k + b^l_j\right)$,
- $a^l = \sigma\left(W^l \times a^{l-1} + b^l\right)$

Вопрос

Что такое a^1 в терминах исходной задачи? Откуда берутся значения a^1?

Усложняем модель: больше слоёв

- $a_j^l = \sigma\left(\sum_{k=1}^{n_{l-1}} w_{jk}^l a_k^{l-1} + b_j^l\right)$,
- $a^l = \sigma\left(W^l \times a^{l-1} + b^l\right)$

Вопрос

Что такое a^1 в терминах исходной задачи? Откуда берутся значения a^1? $a^1 = x_i$, т.е. значения активации на первом слое сети – это входные данные

Усложняем модель: больше слоёв

- $a_j^l = \sigma\left(\sum_{k=1}^{n_{l-1}} w_{jk}^l a_k^{l-1} + b_j^l\right)$,
- $a^l = \sigma\left(W^l \times a^{l-1} + b^l\right)$

Вопрос

Что такое a^1 в терминах исходной задачи? Откуда берутся значения a^1? $a^1 = x_i$, т.е. значения активации на первом слое сети – это входные данные

Вопрос

Что такое a^L в терминах исходной задачи? Откуда берутся значения a^L?

Усложняем модель: больше слоёв

- $a_j^l = \sigma\left(\sum_{k=1}^{n_{l-1}} w_{jk}^l a_k^{l-1} + b_j^l\right)$,
- $a^l = \sigma\left(W^l \times a^{l-1} + b^l\right)$

Вопрос

Что такое a^1 в терминах исходной задачи? Откуда берутся значения a^1? $a^1 = x_i$, т.е. значения активации на первом слое сети – это входные данные

Вопрос

Что такое a^L в терминах исходной задачи? Откуда берутся значения a^L? $a^L = \hat{y}_i$, т.е. активация последнего слоя сети – это ответ, выдаваемый нейронной сетью.

Усложняем модель: больше слоёв

- $a_j^l = \sigma\left(\sum_{k=1}^{n_{l-1}} w_{jk}^l a_k^{l-1} + b_j^l\right)$,
- $a^l = \sigma\left(W^l \times a^{l-1} + b^l\right)$

Вопрос

Что такое a^1 в терминах исходной задачи? Откуда берутся значения a^1? $a^1 = x_i$, т.е. значения активации на первом слое сети – это входные данные

Вопрос

Что такое a^L в терминах исходной задачи? Откуда берутся значения a^L? $a^L = \hat{y}_i$, т.е. активация последнего слоя сети – это ответ, выдаваемый нейронной сетью.

Вопрос

Что нужно изменить в алгоритме настройки весов?

Усложняем модель: больше слоёв

- $a_j^l = \sigma\left(\sum_{k=1}^{n_{l-1}} w_{jk}^l a_k^{l-1} + b_j^l\right)$,
- $a^l = \sigma\left(W^l \times a^{l-1} + b^l\right)$

Вопрос

Что такое a^1 в терминах исходной задачи? Откуда берутся значения a^1? $a^1 = x_i$, т.е. значения активации на первом слое сети – это входные данные

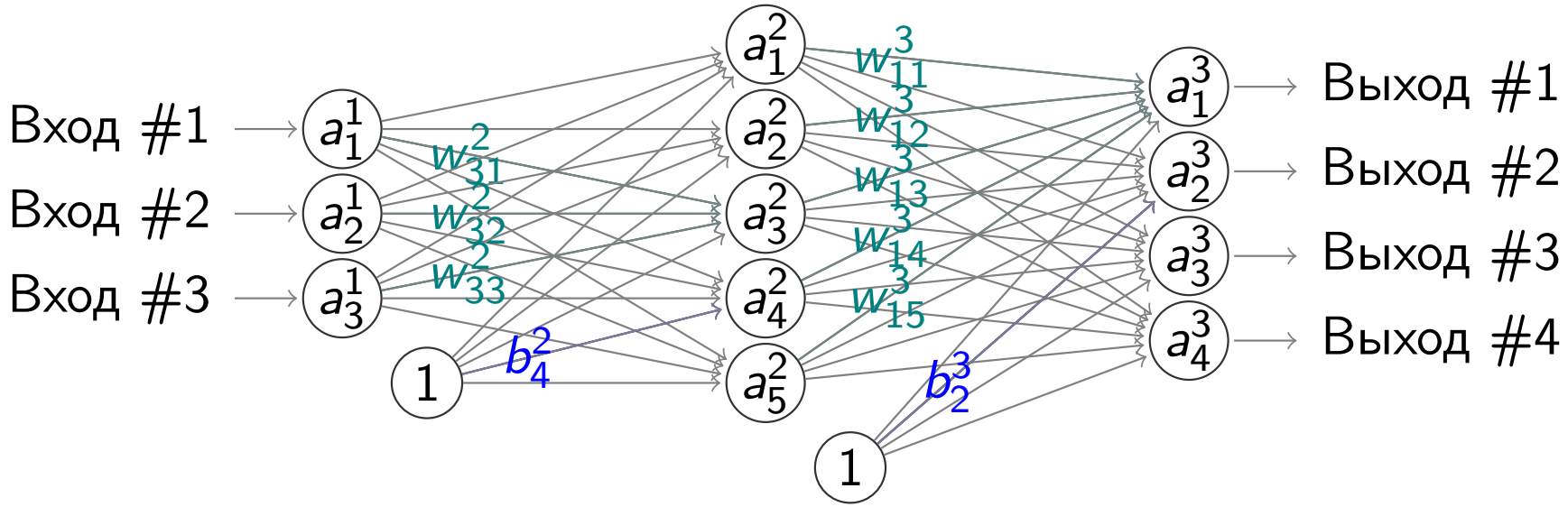
Вопрос

Что такое a^L в терминах исходной задачи? Откуда берутся значения a^L? $a^L = \hat{y}_i$, т.е. активация последнего слоя сети – это ответ, выдаваемый нейронной сетью.

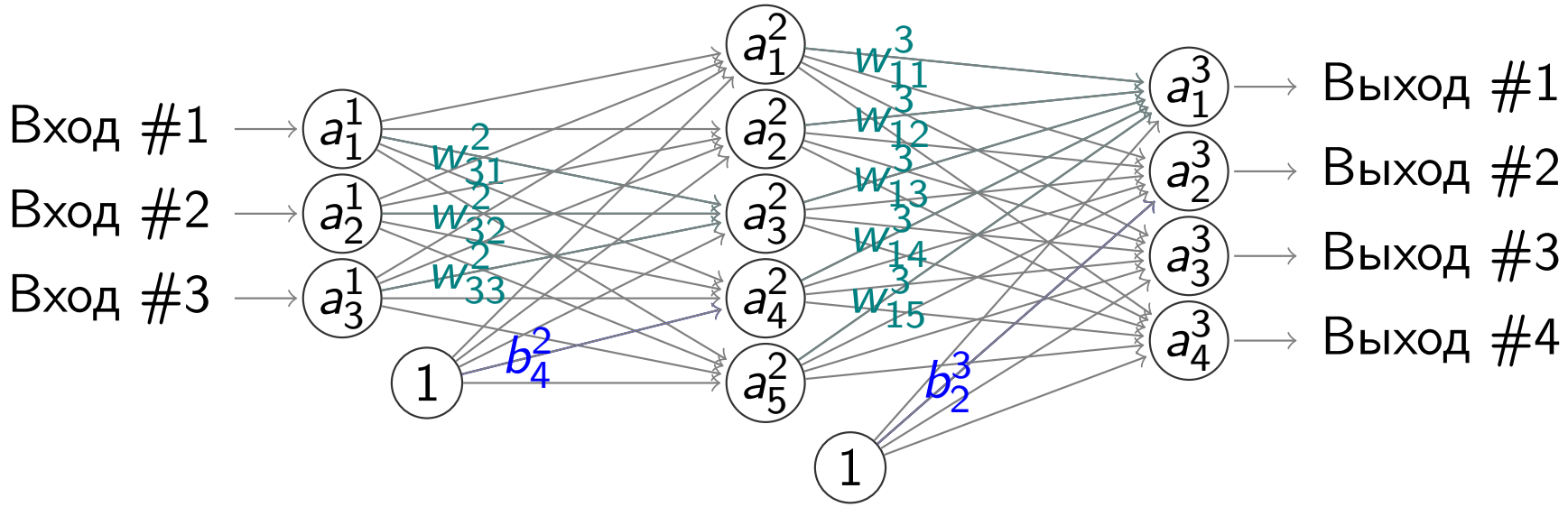
Вопрос

Что нужно изменить в алгоритме настройки весов? Ничего.

Обратное распространение ошибки

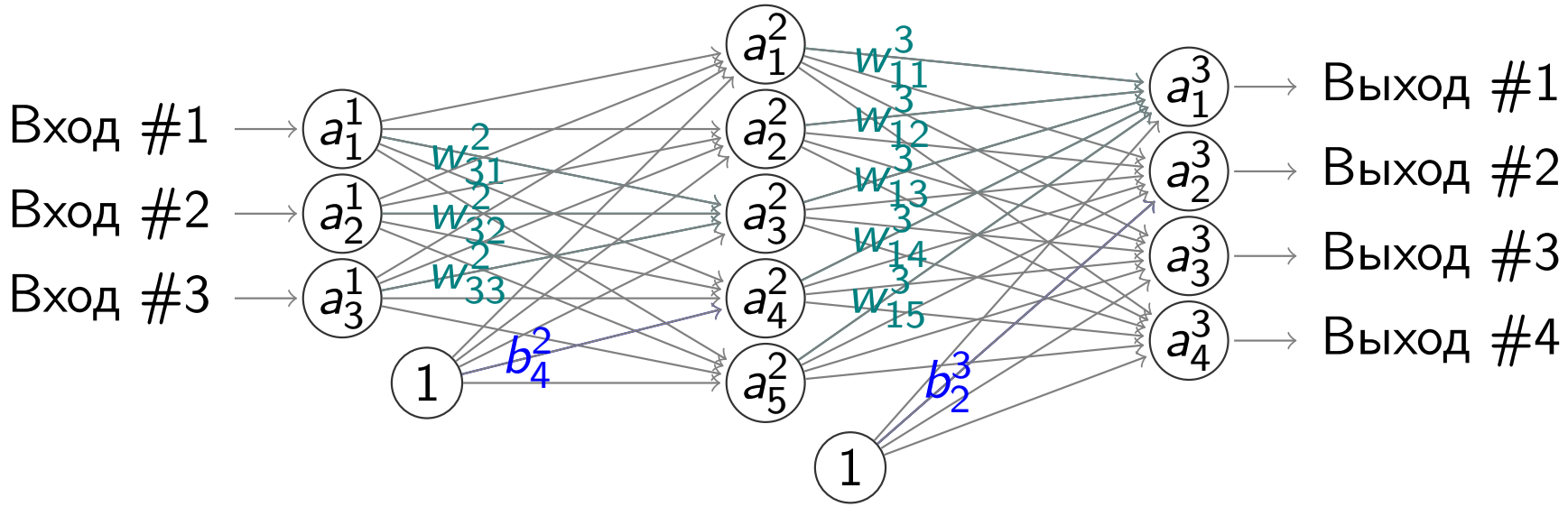


Обратное распространение ошибки



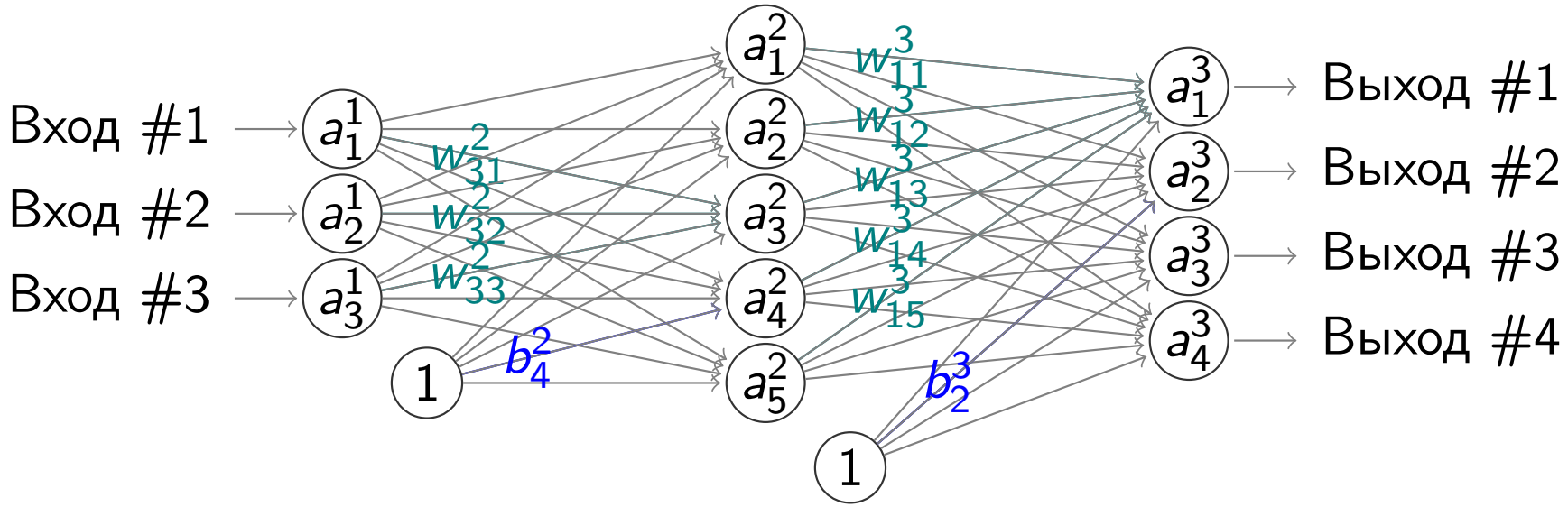
Пусть $x \in \mathbb{R}^{n_1}$ – пример, y – правильный ответ для x, $\hat{y}$ – оценка нейросети для x.

Обратное распространение ошибки



Пусть $x \in \mathbb{R}^{n_1}$ – пример, y – правильный ответ для x, $\hat{y}$ – оценка нейросети для x.

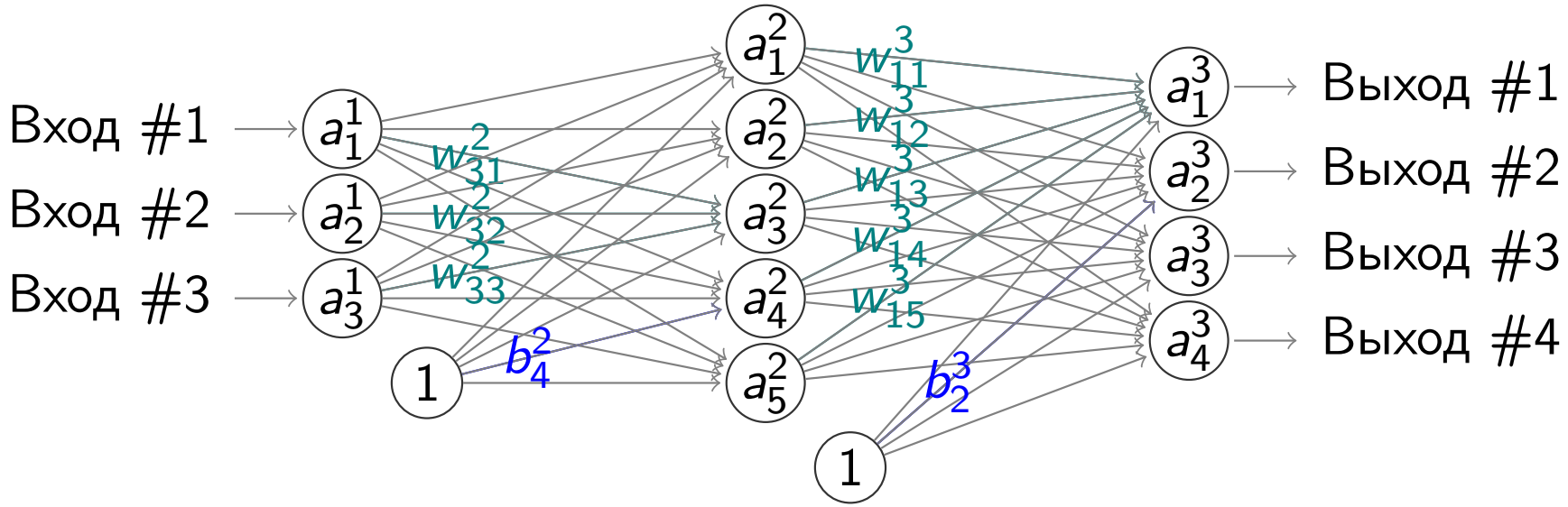
Обратное распространение ошибки



Пусть $x \in \mathbb{R}^{n_1}$ – пример, y – правильный ответ для x, $\hat{y}$ – оценка нейросети для x.

$$\delta_j^L = \frac{\partial J}{\partial s_j^L} = \frac{\partial J}{\partial \hat{y}(j)} \cdot \sigma'\left(s_j^L\right); \quad \delta^L = \nabla_{\hat{y}} J \odot \sigma'\left(s^L\right)$$

Обратное распространение ошибки

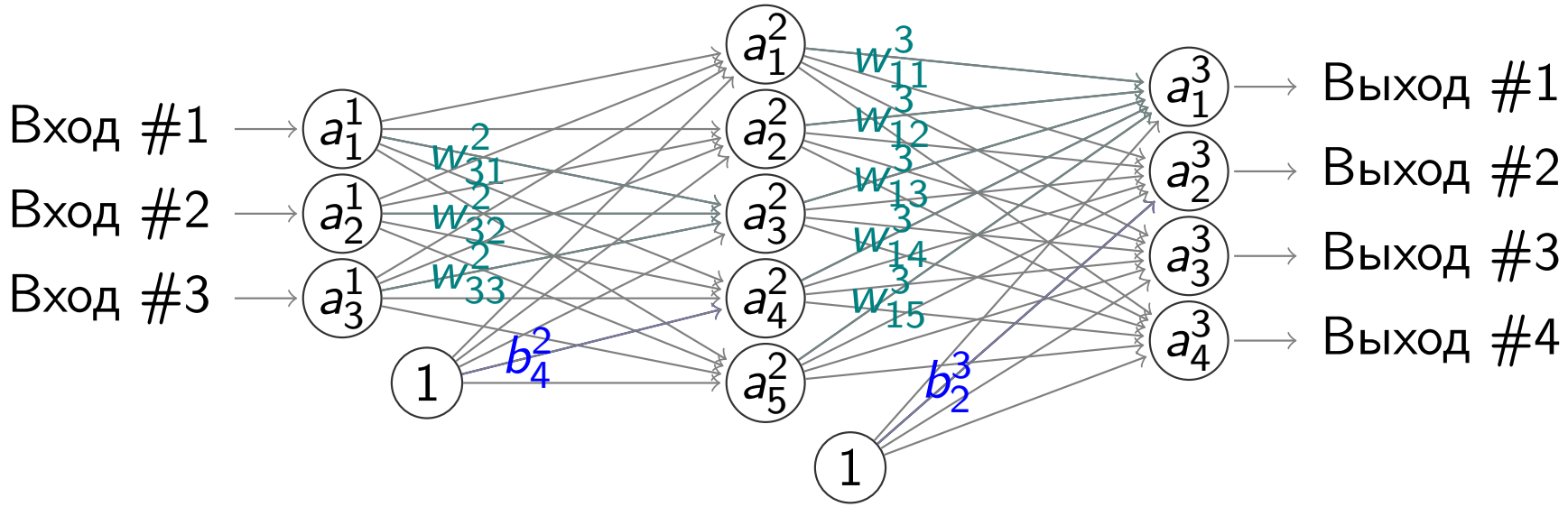


Пусть $x \in \mathbb{R}^{n_1}$ – пример, y – правильный ответ для x, $\hat{y}$ – оценка нейросети для x.

$$\delta_j^l = \frac{\partial J}{\partial s_j^l} = \sum_{k=1}^{n_{l+1}} \frac{\partial J}{\partial s_k^{l+1}} \cdot \frac{\partial s_k^{l+1}}{\partial s_j^l} = \sum_{k=1}^{n_{l+1}} \delta_k^{l+1} \cdot \frac{\partial s_k^{l+1}}{\partial s_j^l} =$$

$$= \sum_{k=1}^{n_{l+1}} \delta_k^{l+1} \cdot \frac{\partial}{\partial s_j^l} \left(\sum_{j_l=1}^{n_l} w_{kj_l}^l \sigma(s_{j_l}^l) + b_k^{l+1} \right) = \sum_{k=1}^{n_{l+1}} \delta_k^{l+1} \cdot w_{kj}^{l+1} \sigma'(s_j^l)$$

Обратное распространение ошибки



Пусть $x \in \mathbb{R}^{n_1}$ – пример, y – правильный ответ для x, $\hat{y}$ – оценка нейросети для x.

$$\delta_j^L = \frac{\partial J}{\partial s_j^L} = \frac{\partial J}{\partial \hat{y}(j)} \cdot \sigma'\left(s_j^L\right); \quad \delta^L = \nabla_{\hat{y}} J \odot \sigma'\left(s^L\right)$$

$$\delta_j^l = \sum_{k=1}^{n_{l+1}} \delta_k^{l+1} \cdot w_{kj}^{l+1} \sigma'(s_j^l), \quad \delta^l = \left((W^{l+1})^{\mathsf{T}} \delta^{l+1}\right) \odot \sigma'(s^l)$$

Обратное распространение ошибки

$$\delta_j^L = \frac{\partial J}{\partial s_j^L} = \frac{\partial J}{\partial \hat{y}(j)} \cdot \sigma'\left(s_j^L\right);$$

Обратное распространение ошибки

$$\delta_j^L = \frac{\partial J}{\partial s_j^L} = \frac{\partial J}{\partial \hat{y}(j)} \cdot \sigma'\left(s_j^L\right); \quad \delta^L = \nabla_{\hat{y}} J \odot \sigma'\left(s^L\right)$$

$$\delta_j^l = \sum_{k=1}^{n_{l+1}} \delta_k^{l+1} \cdot w_{kj}^{l+1} \sigma'(s_j^l); \quad \delta^l = \left((W^{l+1})^{\mathsf{T}} \delta^{l+1}\right) \odot \sigma'(s^l)$$

Обратное распространение ошибки

$$\delta_j^L = \frac{\partial J}{\partial s_j^L} = \frac{\partial J}{\partial \hat{y}(j)} \cdot \sigma'\left(s_j^L\right); \quad \delta^L = \nabla_{\hat{y}} J \odot \sigma'\left(s^L\right)$$

$$\delta_j^l = \sum_{k=1}^{n_{l+1}} \delta_k^{l+1} \cdot w_{kj}^{l+1} \sigma'(s_j^l); \quad \delta^l = \left((W^{l+1})^{\mathsf{T}} \delta^{l+1}\right) \odot \sigma'(s^l)$$

$$\frac{\partial J}{\partial b_j^l} = \delta_j^l, \quad \frac{\partial J}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$$

Обратное распространение ошибки

$$\delta_j^L = \frac{\partial J}{\partial s_j^L} = \frac{\partial J}{\partial \hat{y}(j)} \cdot \sigma'\left(s_j^L\right); \quad \delta^L = \nabla_{\hat{y}} J \odot \sigma'\left(s^L\right)$$

$$\delta_j^l = \sum_{k=1}^{n_{l+1}} \delta_k^{l+1} \cdot w_{kj}^{l+1} \sigma'(s_j^l); \quad \delta^l = \left((W^{l+1})^{\mathsf{T}} \delta^{l+1}\right) \odot \sigma'(s^l)$$

$$\frac{\partial J}{\partial b_j^l} = \delta_j^l, \quad \frac{\partial J}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$$

Ура, мы научились брать производные всех переменных по входным данным.

Обратное распространение ошибки

$$\delta_j^L = \frac{\partial J}{\partial s_j^L} = \frac{\partial J}{\partial \hat{y}(j)} \cdot \sigma'\left(s_j^L\right); \quad \delta^L = \nabla_{\hat{y}} J \odot \sigma'\left(s^L\right)$$

$$\delta_j^l = \sum_{k=1}^{n_{l+1}} \delta_k^{l+1} \cdot w_{kj}^{l+1} \sigma'(s_j^l); \quad \delta^l = \left((W^{l+1})^{\mathsf{T}} \delta^{l+1}\right) \odot \sigma'(s^l)$$

$$\frac{\partial J}{\partial b_j^l} = \delta_j^l, \quad \frac{\partial J}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$$

Ура, мы научились брать производные всех переменных по входным данным. Go learn something!

Ссылки по теме лекции

- Алгоритм обратного распространения ошибки в книге Neural Networks and Deep Learning
- Об автоматическом дифференцировании с примерами из курса CS231n Stanford University
- Зачем нужно понимать back propagation и о двух классических его проблемах от Андрея Карпаты