Полносвязные нейронные сети

Степанов Иван

Московский физико-технический институт

Москва, 26 февраля 2025 г.

План лекции

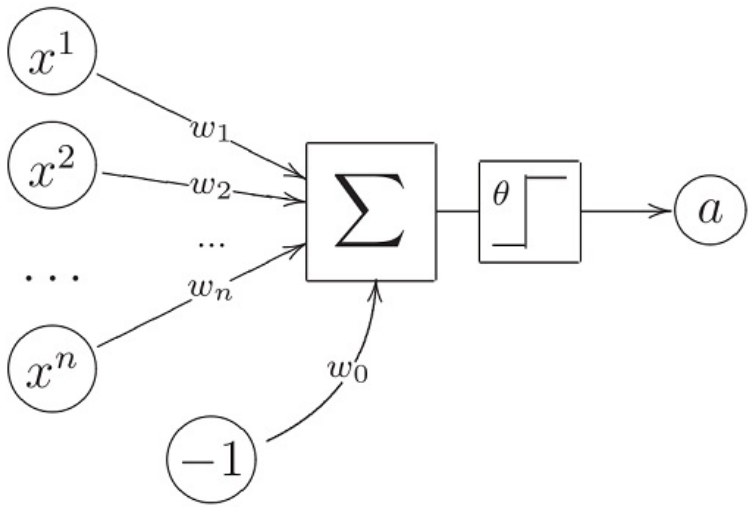
- Нейрон
- Полносвязная нейронная сеть
- Кросс-энтропия
- Функции активации
- Backpropagation

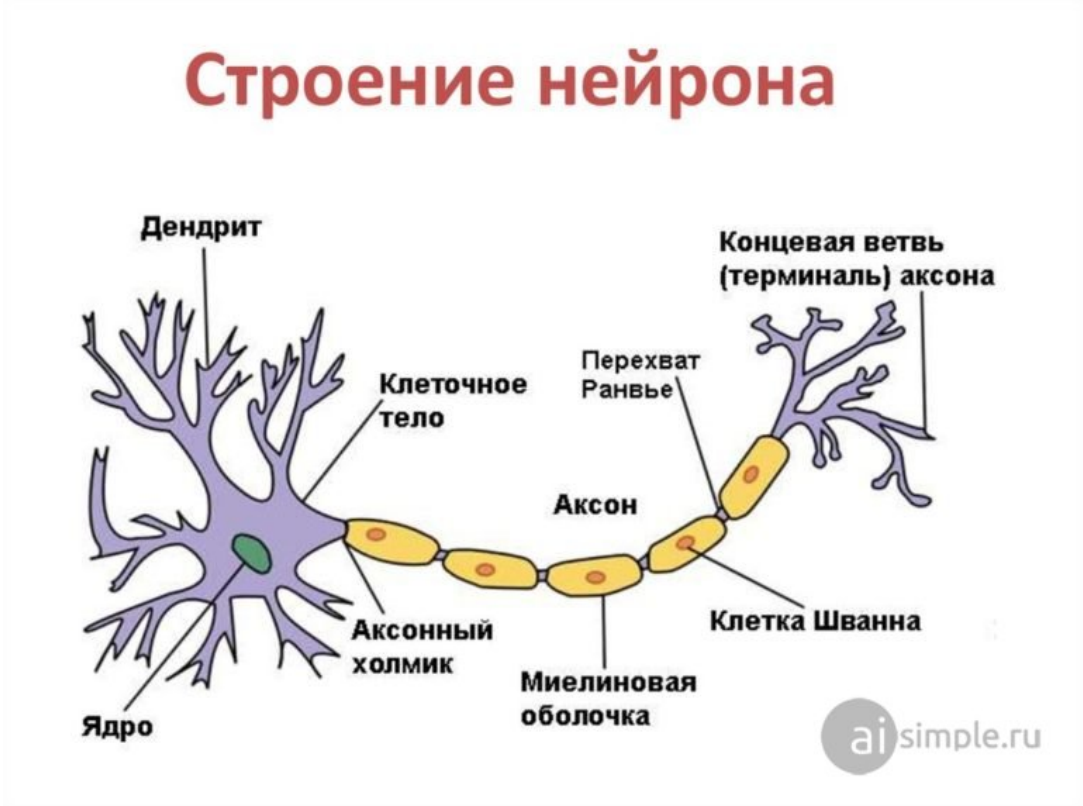
Нейрон (или узел) — это базовый элемент нейронной сети.

$$z = \mathbf{w}^T \mathbf{x} + b$$

$$a = f(z)$$

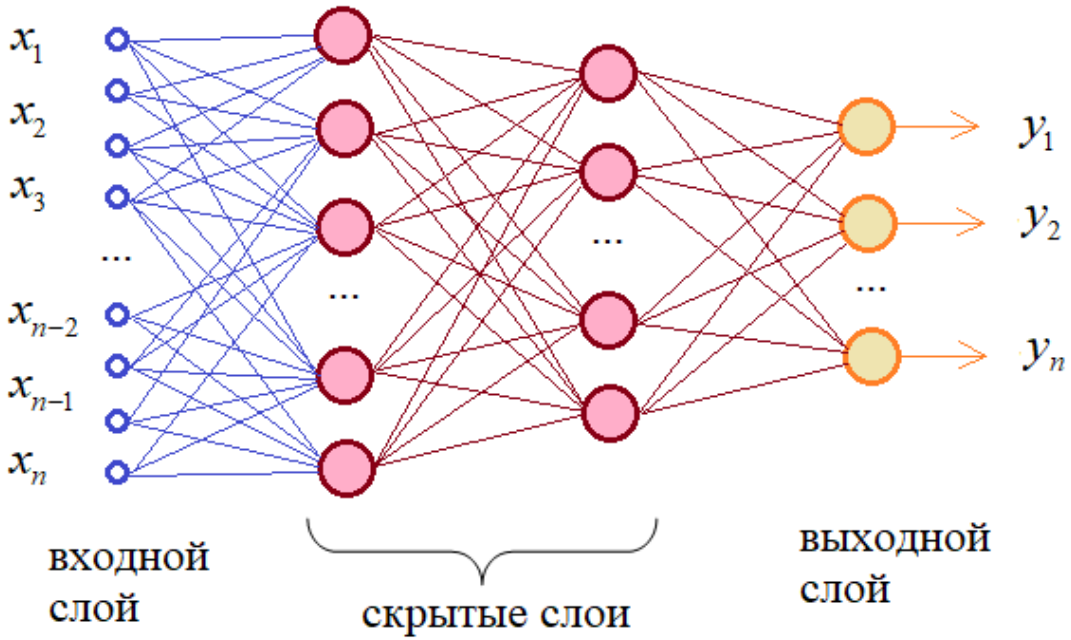
- $\mathbf{x} \in \mathbb{R}^n$ — вектор входных данных размерности n,
- $\mathbf{w} \in \mathbb{R}^n$ — вектор весов,
- $b \in \mathbb{R}$ — смещение (bias),
- $f(\cdot)$ — функция активации (**нелинейная**).





- **Дендриты** : принимают входные сигналы от других нейронов.
- **Ядро** : обрабатывает входные сигналы.
- **Аксон** : передает выходной сигнал другим нейронам через синапсы.
- **Синапсы** : точки контакта между нейронами, где происходит передача информации.

Полносвязная нейронная сеть



- **Входной слой** : принимает данные. Каждый нейрон этого слоя соответствует одному признаку входного вектора.
- **Скрытые слои** : обрабатывают данные через последовательность линейных преобразований и нелинейных функций активации.
- **Выходной слой** : производит окончательный результат (например, классификацию или регрессию).

Зачача1

Почему k линейных преобразований - это линейное преобразование?

Задача2

XOR (исключающее "или") — это классический пример задачи, которая не может быть решена линейной моделью.
Чтобы решить эту задачу, нужно использовать **нелинейную модель**.
Предложите нейронную сеть, решающую данную задачу.

Функция softmax — функция, преобразующая вектор вещественных чисел (логитов) в вектор вероятностей.

Для вектора $\mathbf{z} = [z_1, z_2, \ldots, z_K]$, где K — количество классов, softmax-функция вычисляется как:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \quad i = 1, 2, \ldots, K.$$

- z_i — логит для i-го класса,
- e^{z_i} — экспонента от z_i,
- $\sum_{j=1}^{K} e^{z_j}$ — нормализующий коэффициент, который обеспечивает то, что сумма всех выходов равна 1.

В случае задачи классификации наиболее популярной функцие потерь является **кросс-энтропия**.

$$L = -\sum_{i=1}^{K} y_i \log(\hat{y}_i),$$

- K – количество классов,
- y_i – истинное значение для i-го класса ($y_i = 1$ для правильного класса и $y_i = 0$ для остальных),
- $\hat{y}_i$ – предсказанное значение для i-го класса (выход softmax или другой функции активации).

Предположим, что мы решаем задачу с тремя классами ($K = 3$). Для некоторого объекта (класса $y_2 = 1$) нейронная сеть выдала следующие вероятности: $\hat{y}_1 = 0.1, \hat{y}_2 = 0.8, \hat{y}_3 = 0.1$.

$$L = -[0 \cdot \log(0.1) + 1 \cdot \log(0.8) + 0 \cdot \log(0.1)] = -\log(0.8).$$

$$L = -\log(0.8) \approx -(-0.223) \approx 0.223.$$

Предположим, что мы решаем задачу с тремя классами ($K = 3$). Для другого объекта (класса $y_2 = 1$) нейронная сеть выдала следующие вероятности: $\hat{y}_1 = 0.5, \hat{y}_2 = 0.2, \hat{y}_3 = 0.3$.

$$L = -[0 \cdot \log(0.5) + 1 \cdot \log(0.2) + 0 \cdot \log(0.3)] = -\log(0.2).$$

$$L = -\log(0.2) \approx -(-1.609) \approx 1.609.$$

Популярные функции активации:

- Sigmoid
- Tanh
- ReLU
- Leaky ReLU

Популярные функции активации:

- Sigmoid
- Tanh
- ReLU
- Leaky ReLU

Формула

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

+

- Выход ограничен диапазоном $[0, 1]$, что позволяет интерпретировать результат как вероятность.

-

- При больших положительных или отрицательных значениях входа (x) производная становится очень маленькой ($f'(x) \approx 0$), что замедляет обучение.
- Выход всегда больше 0, что может затруднять обучение глубоких сетей.

Tanh (Гиперболический тангенс)

Формула

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

+

- Выход ограничен диапазоном $[-1, 1]$, что делает её симметричной относительно нуля.
- Более эффективна, чем sigmoid, так как имеет более широкий диапазон значений.

-

- Также страдает от проблемы вырождения градиента ($f'(x) \approx 0$ при больших $|x|$).

Формула

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

+

- Линейная функция для положительных значений, что делает её быстрой в вычислениях.
- Для $x > 0$ производная равна 1, что ускоряет обучение.

-

- Если веса обучаются таким образом, что $x < 0$ для большинства примеров, то нейрон "умирает"(всегда даёт выход 0).
- Не ограничивает значения сверху, что может привести к проблемам с разрывом градиента (exploding gradients).

Формула

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

где α — малое положительное число (например, 0.01).

+

- Устраняет проблему "умирающих нейронов так как для $x < 0$ существует небольшой градиент (α).
- Сохраняет преимущества ReLU.

-

- Параметр α требует настройки.

Математическая формулировка формулировка

Пусть:

- $f : \mathbb{R}^n \to \mathbb{R}$ — произвольная непрерывная функция,
- $K \subset \mathbb{R}^n$ — компактное множество (замкнутое и ограниченное),
- $\sigma : \mathbb{R} \to \mathbb{R}$ — нелинейная функция активации.

Тогда для любого $\epsilon > 0$ существует нейронная сеть с одним скрытым слоем, такая что:

$$\sup_{x \in K} |F(x) - f(x)| < \epsilon,$$

где $F(x)$ — выход нейронной сети, определяемый как:

$$F(x) = \sum_{i=1}^{N} w_i \sigma(\mathbf{v}_i^T x + b_i),$$

- N — количество нейронов в скрытом слое,
- $\mathbf{v}_i \in \mathbb{R}^n$ — весовые векторы,
- $b_i \in \mathbb{R}$ — смещения,
- $w_i \in \mathbb{R}$ — веса выходного слоя.

Веса модели обновляются с помощью градиентного спуска.

Прямое распространение

Для каждого слоя l:

$$z^{(l)} = W^{(l)} f(z^{(l)}) + b^{(l)},$$

- $z^{(l)}$ — линейное преобразование,
- $f(\cdot)$ — функция активации.

Обратное распространение

Градиенты вычисляются рекурсивно, начиная с выходного слоя и двигаясь к входному. Для выходного слоя:

$$\delta^{(L)} = \frac{\partial L}{\partial z^{(L)}} = \nabla_a L \odot f'(z^{(L)}),$$

- $\delta^{(L)}$ – ошибка выходного слоя,
- $\nabla_a L$ – градиент функции потерь относительно активаций выходного слоя,
- $f'(z^{(L)})$ – производная функции активации.

$$\delta^{(l)} = \left(W^{(l+1)T}\delta^{(l+1)}\right) \odot f'(z^{(l)}),$$

- $W^{(l+1)T}$ – транспонированная матрица весов следующего слоя,
- $\delta^{(l+1)}$ – ошибка следующего слоя.

Градиенты по весам и смещениям:

$$\frac{\partial L}{\partial W^{(l)}} = \delta^{(l)}(a^{(l-1)})^T,$$

$$\frac{\partial L}{\partial b^{(l)}} = \delta^{(l)}.$$

Нейронная сеть

Рассмотрим простую нейронную сеть с одним скрытым слоем.
Архитектура:
Входной слой: 2 нейрона (x_1, x_2),
Скрытый слой: 2 нейрона,
Выходной слой: 1 нейрон.

Дано

Пусть:

Веса первого слоя: $W^{(1)} = \begin{bmatrix} 0.5 & -0.3 \\ 0.4 & 0.6 \end{bmatrix}$

Смещения первого слоя: $b^{(1)} = \begin{bmatrix} 0.1 \\ -0.2 \end{bmatrix}$

Веса второго слоя: $W^{(2)} = \begin{bmatrix} 0.8 & -0.7 \end{bmatrix}$

Смещение второго слоя: $b^{(2)} = 0.3$

Функция активации: sigmoid $f(x) = \frac{1}{1+e^{-x}}$.

Вычислим выход скрытого слоя:

$$z^{(1)} = W^{(1)}x + b^{(1)} = \begin{bmatrix} 0.5 & -0.3 \\ 0.4 & 0.6 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 0.1 \\ -0.2 \end{bmatrix} = \begin{bmatrix} 0.5 - 0.6 + 0.1 \\ 0.4 + 1.2 - 0.2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1.4 \end{bmatrix}.$$

$$a^{(1)} = \sigma(z^{(1)}) = \begin{bmatrix} \frac{1}{1+e^{-0}} \\ \frac{1}{1+e^{-1.4}} \end{bmatrix} \approx \begin{bmatrix} 0.5 \\ 0.8 \end{bmatrix}.$$

Вычислим выход выходного слоя:

$$z^{(2)} = W^{(2)}a^{(1)} + b^{(2)} = \begin{bmatrix} 0.8 & -0.7 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.8 \end{bmatrix} + 0.3 = 0.4 - 0.56 + 0.3 = 0.14.$$

$$a^{(2)} = \sigma(z^{(2)}) = \frac{1}{1 + e^{-0.14}} \approx 0.535.$$

Предположим, что истинное значение $y = 1$, а функция потерь – кросс-энтропия:

$$L = -[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})].$$

$$\delta^{(2)} = \nabla_a L \cdot f'(z^{(2)}) = -0.465 \cdot \sigma(z^{(2)})(1 - \sigma(z^{(2)})) \approx -0.465 \cdot 0.535 \cdot (1 - 0.535) \approx -0.114.$$

$$\delta^{(1)} = \left(W^{(2)T} \delta^{(2)}\right) \odot f'(z^{(1)}) = \begin{bmatrix} 0.8 \\ -0.7 \end{bmatrix} (-0.114) \odot \begin{bmatrix} 0.5 \cdot (1 - 0.5) \\ 0.8 \cdot (1 - 0.8) \end{bmatrix}.$$

$$\delta^{(1)} = \begin{bmatrix} -0.091 \\ 0.064 \end{bmatrix}.$$

$$\frac{\partial L}{\partial W^{(2)}} = \delta^{(2)} (a^{(1)})^T = -0.114 \cdot \begin{bmatrix} 0.5 \\ 0.8 \end{bmatrix} = \begin{bmatrix} -0.057 \\ -0.091 \end{bmatrix}.$$

$$\frac{\partial L}{\partial b^{(2)}} = \delta^{(2)} = -0.114.$$

$$\frac{\partial L}{\partial W^{(1)}} = \delta^{(1)} (x)^T = \begin{bmatrix} -0.091 \\ 0.064 \end{bmatrix} \begin{bmatrix} 1 & 2 \end{bmatrix} = \begin{bmatrix} -0.091 & -0.182 \\ 0.064 & 0.128 \end{bmatrix}.$$

$$\frac{\partial L}{\partial b^{(1)}} = \delta^{(1)} = \begin{bmatrix} -0.091 \\ 0.064 \end{bmatrix}.$$

Тонкости обучения

- Инициализация весов
- Регуляризация нейронных сетей
- Методы градиентного спуска

Инициализация 0

Задача 2

Почему инициализация 0 - это очень плохо?

Случайная инициализация

Веса инициализируются случайными значениями из некоторого распределения.

Xavier (Glorot) инициализация

$$W \sim \text{Uniform}\left(-\frac{\sqrt{6}}{\sqrt{n_{\text{in}} + n_{\text{out}}}}, \frac{\sqrt{6}}{\sqrt{n_{\text{in}} + n_{\text{out}}}}\right),$$

- n_{in} — количество входных связей
- n_{out} — количество выходных связей.

Во время обучения Dropout случайным образом обнуляет выходы некоторых нейронов с заданной вероятностью p (где $p \in [0, 1]$). Это заставляет сеть не полагаться на конкретные нейроны и вместо этого учиться распределять важность между всеми нейронами, тем самым снижая риск переобучения.
На этапе тестирования Dropout отключается, и все нейроны используются, но их выходы масштабируются для компенсации эффекта "выключения"во время обучения.

Итоги

- Нейроная сеть - это просто **сложная функция**.
- Теорема Цыбенко говорит лишь о теоретических возможностях нейронных сетей.
- Существуют различные функции активации.
- Существуют различные тонкости обучения нейронных сетей

Литература

- Полезные ссылки
- Полезные ссылки

Вопросы?