Нейронные сети: основы

от линейных моделей до backpropagation

Что мы хотим уметь после лекции

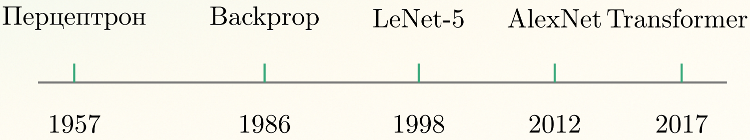
- Объяснять нейросеть как композицию функций (слои + нелинейности).
- Понимать, где ломается линейная модель: XOR и нелинейные границы.
- Знать базовые элементы: нейрон, слой Dense, функция потерь, активации.
- Понимать обучение: градиентный спуск и роль learning rate.
- Уметь сделать backprop на простом примере и читать матричные формулы.

Где применяются нейросети (и почему именно там)

- Картинки: имеют локальную структуру (пиксели рядом связаны) $\Rightarrow$ CNN/ViT.
- Текст: последовательность + контекст $\Rightarrow$ Transformer.
- Звук/речь: сигнал во времени $\Rightarrow$ модели для последовательностей/спектрограмм.
- Таблицы: нейросети тоже работают, но часто сильны бустинги (CatBoost/XGBoost).

Важно понимать: какой тип данных и какая модель подходит.

Короткая история (таймлайн)



- Backprop сделал обучение многослойных сетей практичным.
- AlexNet показал силу: данные + GPU + хорошие приёмы.

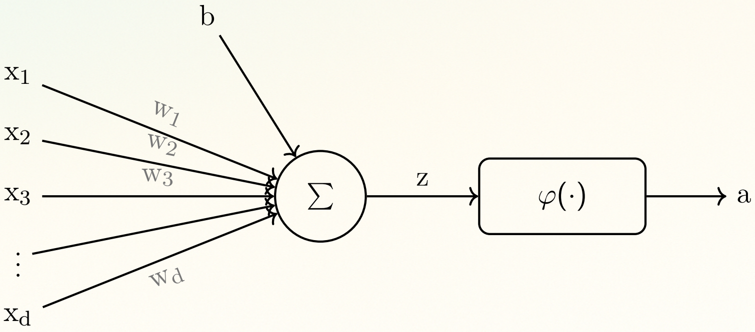
Нейрон (перцептрон): из чего состоит

$$z = w^\top x + b, \qquad a = \varphi(z)$$

- $w^\top x$ — скалярное произведение (насколько x совпадает с направлением w).
- b — сдвиг (bias): двигает границу.
- φ — нелинейность.

Удобная интерпретация: нейрон проверяет “похоже ли x на шаблон w”.

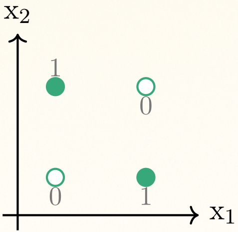
Нейрон (перцептрон): схема



$z = w^\top x + b$ $\qquad$ $a = \varphi(z)$

XOR: пример, который “ломает” линейность

$$\mathrm{XOR}(x_1, x_2) = 1 \iff x_1 \neq x_2$$



Невозможно отделить точки класса 1 от класса 0 одной прямой.

Мини-доказательство: XOR не линейно разделим

Предположим, что существует прямая (гиперплоскость)

$$\mathrm{w}_1\mathrm{x}_1 + \mathrm{w}_2\mathrm{x}_2 + \mathrm{b} = 0$$

которая отделяет 1 от 0.
Для XOR хотим:

$$(0,1) \text{ и } (1,0) \Rightarrow \mathrm{w}_2 + \mathrm{b} > 0,\ \mathrm{w}_1 + \mathrm{b} > 0$$

$$(0,0) \text{ и } (1,1) \Rightarrow \mathrm{b} < 0,\ \mathrm{w}_1 + \mathrm{w}_2 + \mathrm{b} < 0$$

Сложим первые два неравенства:

$$(\mathrm{w}_1 + \mathrm{b}) + (\mathrm{w}_2 + \mathrm{b}) > 0 \Rightarrow \mathrm{w}_1 + \mathrm{w}_2 + 2\mathrm{b} > 0$$

но при $\mathrm{b} < 0$ и $\mathrm{w}_1 + \mathrm{w}_2 + \mathrm{b} < 0$ получаем противоречие для подходящих значений.

Как можно “спасти” XOR без нейросетей

Добавим вручную новый признак:

$$x_3 = x_1 x_2$$

Тогда XOR становится линейно разделим в пространстве (x_1, x_2, x_3).

- Это пример ручной инженерии признаков.
- В реальных задачах придумать хорошие признаки сложно.
- Нейросеть делает это автоматически: скрытые слои играют роль генератора признаков.

Dense-слой: много нейронов сразу (матрицы)

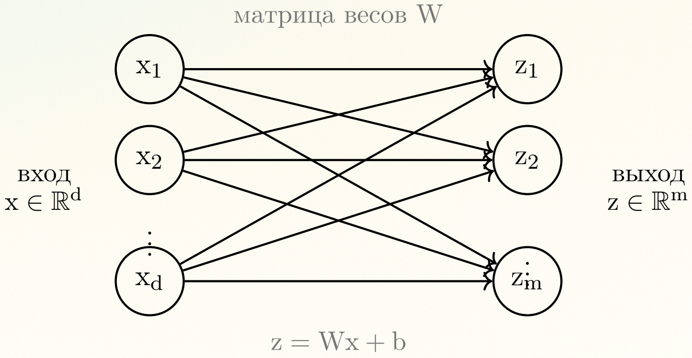
Если вход $\mathrm{x} \in \mathbb{R}^{\mathrm{d}}$, а нейронов m, то:

$$\mathrm{z} = \mathrm{W}\mathrm{x} + \mathrm{b}, \quad \mathrm{W} \in \mathbb{R}^{\mathrm{m}\times\mathrm{d}},\ \mathrm{b} \in \mathbb{R}^{\mathrm{m}}$$

$$\mathrm{a} = \varphi(\mathrm{z}) \quad \text{(поэлементно)}$$

- Строка $\mathrm{W}_{\mathrm{j}:}$ — веса j-го нейрона.

Dense-слой: каждый выходной нейрон видит все входы



- $z_j = \sum_{i=1}^{d} W_{j,i} x_i + b_j$ — формула для j-го выхода.
- "Полносвязный" = связи между всеми входами и всеми выходами.

Что такое “слой”: договоримся о терминах

В этой лекции:

- Линейный (Dense) слой: $z = Wx + b$ — имеет параметры.
- Слой активации: $a = \varphi(z)$ — параметров обычно нет.
- Скрытый слой: всё, что между входом и выходом.

В разных книгах “1-layer network” может означать разное

Сеть без скрытых слоёв: что умеет

Модель:

$$\hat{y} = \psi(Wx + b)$$

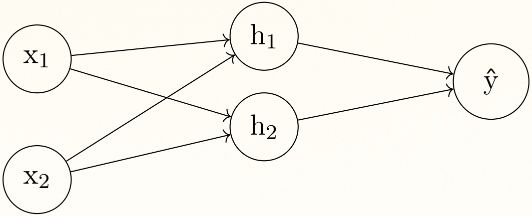
- Умеет строить только линейные границы.
- AND/OR решает, XOR — нет.

Мощность модели ограничена геометрией линейного разделения.

Сеть с 1 скрытым слоем: формула

$$h = \varphi(W_1 x + b_1), \qquad \hat{y} = \psi(W_2 h + b_2)$$

- h — скрытые признаки (сеть их выучила).
- W_1 учится “делать признаки”, W_2 — “собирать ответ”.



Почему без нелинейности глубина не помогает

Если φ линейна, то:

$$\mathrm{W}_2(\mathrm{W}_1\mathrm{x} + \mathrm{b}_1) + \mathrm{b}_2 = (\mathrm{W}_2\mathrm{W}_1)\mathrm{x} + (\mathrm{W}_2\mathrm{b}_1 + \mathrm{b}_2)$$

- Два линейных слоя превращаются в один линейный.
- Поэтому нелинейность — обязательна, иначе сеть “не становится умнее”.

Интуиция: как скрытый слой помогает XOR

Скрытый слой может “нарезать” плоскость на области.

- Каждый нейрон скрытого слоя задаёт “мягкое условие” типа (по одну сторону прямой или по другую).
- Комбинация нескольких условий даёт сложную границу.

Можно взять второй слой выражающий AND и OR, после чего выход модели будет OR - AND > 0

Теорема Цыбенко: что она реально говорит

- Сеть с одним скрытым слоем и подходящей нелинейностью может приблизить любую непрерывную функцию на ограниченной области с любой точностью.

Но!

- нейронов может понадобиться очень много;
- теорема не обещает, что обучение найдёт хорошие веса быстро;
- “аппроксимировать” $\neq$ “хорошо обобщать”.

Типовые loss: регрессия и классификация

Регрессия (MSE):

$$\mathcal{L} = (\hat{y} - y)^2 \quad \text{или} \quad L = \frac{1}{2}(\hat{y} - y)^2$$

Бинарная классификация: если $\hat{p} \in (0, 1)$,

$$\mathcal{L} = -\Big(y \log \hat{p} + (1 - y) \log(1 - \hat{p})\Big)$$

- MSE — просто, но для классификации часто хуже.
- Cross-entropy даёт более "правильные" градиенты для вероятностей.

Градиентный спуск: смысл и формула

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathrm{L}$$

- $\nabla_\theta \mathrm{L}$ показывает направление максимального роста loss.
- Мы идём в противоположную сторону.
- η — learning rate: слишком большой $\Rightarrow$ “скачет”, слишком маленький $\Rightarrow$ “не учится”.

GD vs SGD vs mini-batch

- GD: градиент по всему датасету — точно, но медленно.
- SGD: по одному примеру — быстро, но шумно.
- Mini-batch: компромисс (обычно 32–512 объектов).

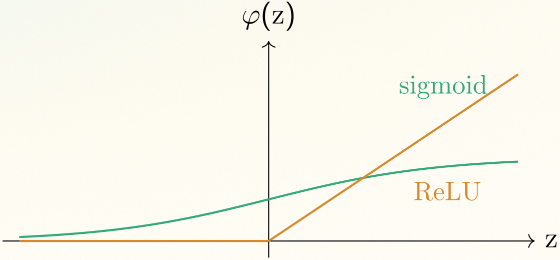
Шум в SGD иногда помогает “выпрыгивать” из плохих локальных ситуаций.

Переобучение (overfitting): как распознать и что делать

- Симптом: качество на train растёт, на val падает.
- Причины: слишком много параметров, мало данных, сильный шум.
- Лечения:
  - уменьшить модель,
  - early stopping,
  - регуляризация (L2/Dropout),
  - больше данных / аугментации.

Даже если loss на train маленький, модель может быть плохой на новых данных.

Графики активаций: sigmoid и ReLU



- Sigmoid насыщается по краям $\Rightarrow$ градиенты маленькие.
- ReLU не насыщается при $z > 0 \Rightarrow$ учиться проще (часто).

Производные активаций (почему бывают проблемы)

Sigmoid:

$$\sigma'(\mathrm{z}) = \sigma(\mathrm{z})(1 - \sigma(\mathrm{z}))$$

- Максимум около 0.25, а на краях стремится к 0.
- В глубоких сетях это может давать затухание градиента.

ReLU:

$$\mathrm{ReLU}'(\mathrm{z}) = \begin{cases} 1, & \mathrm{z} > 0 \\ 0, & \mathrm{z} < 0 \end{cases}$$

- Проблема: если нейрон ушёл в $\mathrm{z} < 0$ надолго, он может “умереть” (dead ReLU).

Практическая памятка по активациям

- Скрытые слои: ReLU — базовый выбор.
- Если “умирают” нейроны: LeakyReLU/ELU (идея: небольшой наклон при $z < 0$).
- Выход:
  - регрессия: часто линейный выход,
  - бинарная классификация: sigmoid,
  - многокласс: softmax.

Backpropagation: что это (одной фразой)

Backpropagation — алгоритм, который считает градиенты

$$\nabla_\theta L$$

для всех параметров сети за один проход (по сути: цепное правило, организованное эффективно).

Цепное правило: мини-разминка

Если

$$\mathrm{u} = \mathrm{g}(\mathrm{v}), \quad \mathrm{v} = \mathrm{h}(\mathrm{x})$$

то

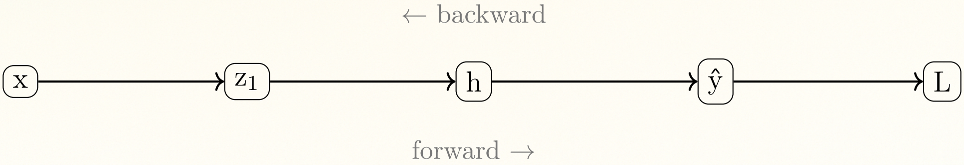
$$\frac{\mathrm{du}}{\mathrm{dx}} = \frac{\mathrm{du}}{\mathrm{dv}} \cdot \frac{\mathrm{dv}}{\mathrm{dx}}$$

- В нейросети таких вложений много.
- Backprop делает это системно: идём от L к параметрам справа налево.

Вычислительный граф: как будем мыслить

Мы будем хранить:

- forward значения в узлах: $x \to z_1 \to h \to \hat{y} \to L$
- градиенты в узлах: $\frac{\partial L}{\partial(\cdot)}$



Пример backprop: функция $f = (x + y)z$

Идея:

- строим вычислительный граф из простых операций;
- делаем forward-pass (значения);
- делаем backward-pass (градиенты) по цепному правилу.

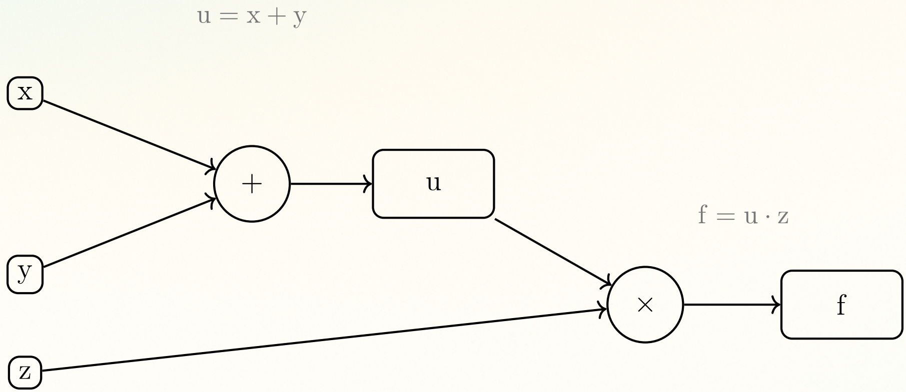
Наша цель: найти

$$\frac{\partial f}{\partial x}, \quad \frac{\partial f}{\partial y}, \quad \frac{\partial f}{\partial z}.$$

Вычислительный граф

Разобьём на промежуточные узлы:

$$u = x + y, \qquad f = u \cdot z$$



Forward-pass (посчитаем значения на примере)

Возьмём числа для наглядности:

$$x = 1, \quad y = 2, \quad z = 3$$

Тогда:

$$u = x + y = 1 + 2 = 3, \qquad f = u \cdot z = 3 \cdot 3 = 9.$$

узел	значение
x	1
y	2
z	3
$u = x + y$	3
$f = uz$	9

Локальные производные (правила для узлов)

Для суммы:

$$u = x + y \Rightarrow \frac{\partial u}{\partial x} = 1, \quad \frac{\partial u}{\partial y} = 1$$

Для произведения:

$$f = u \cdot z \Rightarrow \frac{\partial f}{\partial u} = z, \quad \frac{\partial f}{\partial z} = u$$

Backprop: градиент “сверху” умножаем на локальную производную и передаём назад.

Backward-pass: шаг 1 (от f назад к u и z)

Начинаем с:

$$\frac{\partial \mathrm{f}}{\partial \mathrm{f}} = 1$$

Так как $\mathrm{f} = \mathrm{u} \cdot \mathrm{z}$, то:

$$\frac{\partial \mathrm{f}}{\partial \mathrm{u}} = \mathrm{z}, \qquad \frac{\partial \mathrm{f}}{\partial \mathrm{z}} = \mathrm{u}$$

В числах (из forward: $\mathrm{u} = 3$, $\mathrm{z} = 3$):

$$\frac{\partial \mathrm{f}}{\partial \mathrm{u}} = 3, \qquad \frac{\partial \mathrm{f}}{\partial \mathrm{z}} = 3$$

Backward-pass: шаг 2 (из u назад к x и y)

Так как $u = x + y$, то:

$$\frac{\partial u}{\partial x} = 1, \quad \frac{\partial u}{\partial y} = 1$$

По цепному правилу:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial u} \cdot \frac{\partial u}{\partial x} = z \cdot 1 = z$$

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial u} \cdot \frac{\partial u}{\partial y} = z \cdot 1 = z$$

В числах ($z = 3$):

$$\frac{\partial f}{\partial x} = 3, \qquad \frac{\partial f}{\partial y} = 3$$

Итог: градиенты (символически и численно)

Символически:

$$f = (x + y)z \Rightarrow \frac{\partial f}{\partial x} = z, \quad \frac{\partial f}{\partial y} = z, \quad \frac{\partial f}{\partial z} = x + y$$

Для примера $x = 1, y = 2, z = 3$:

$$\nabla f = (3,\ 3,\ 3)$$

Это “backprop в миниатюре”: граф + локальные производные + цепное правило.

Главная идея backprop: достаточно знать градиенты слоёв и активаций

- Нейросеть — это цепочка блоков:

$$x \to \text{Dense} \to \text{Activation} \to \text{Dense} \to \dots \to L$$

- Backprop работает так:
  - справа налево (от L к входу) передаём градиент по выходу блока;
  - внутри каждого блока считаем градиент по входу и по параметрам.
- Значит, чтобы реализовать backprop для всей сети, достаточно иметь “шпаргалку” производных для двух типов блоков:
  - линейный слой (Dense): $z = Wx + b$
  - активация: $a = \varphi(z)$

Как LEGO: если умеем считать производные у деталей, то соберём градиент для любой конструкции.

Backprop для Dense-слоя

Слой:

$$z = Wx + b, \qquad a = \varphi(z)$$

Если известен $\frac{\partial L}{\partial a}$, то:

$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial a} \odot \varphi'(z)$$

$$\frac{\partial L}{\partial W} = \left(\frac{\partial L}{\partial z}\right) x^\top, \qquad \frac{\partial L}{\partial b} = \frac{\partial L}{\partial z}$$

$$\frac{\partial L}{\partial x} = W^\top \frac{\partial L}{\partial z}$$

$\odot$ — поэлементное умножение. Следите за размерностями!

Практика: TensorFlow Playground (на урок/дом)

- https://playground.tensorflow.org/
- Задания:
  1. XOR: 1 слой vs 2 слоя (что меняется?)
  2. circles/moons: как влияет число нейронов?
  3. tanh vs ReLU: скорость/качество
  4. learning rate: найдите слишком большой и слишком маленький

Итоги

- Нейросеть = Dense + нелинейность (иначе всё сведётся к линейной модели).
- XOR показывает, зачем нужен скрытый слой.
- Обучение = минимизация loss градиентным спуском.
- Backprop = цепное правило, применённое к вычислительному графу.
- Матричные формулы backprop — базовый инструмент для олимпиад и практики.

Великие карьеры, великие достижения рождаются из встречи характера, гения и удачи.

— Наполеон Бонапарт

The End