

Основы машинного обучения

Поляк Марк Дмитриевич

2025

Введение в искусственные нейронные сети

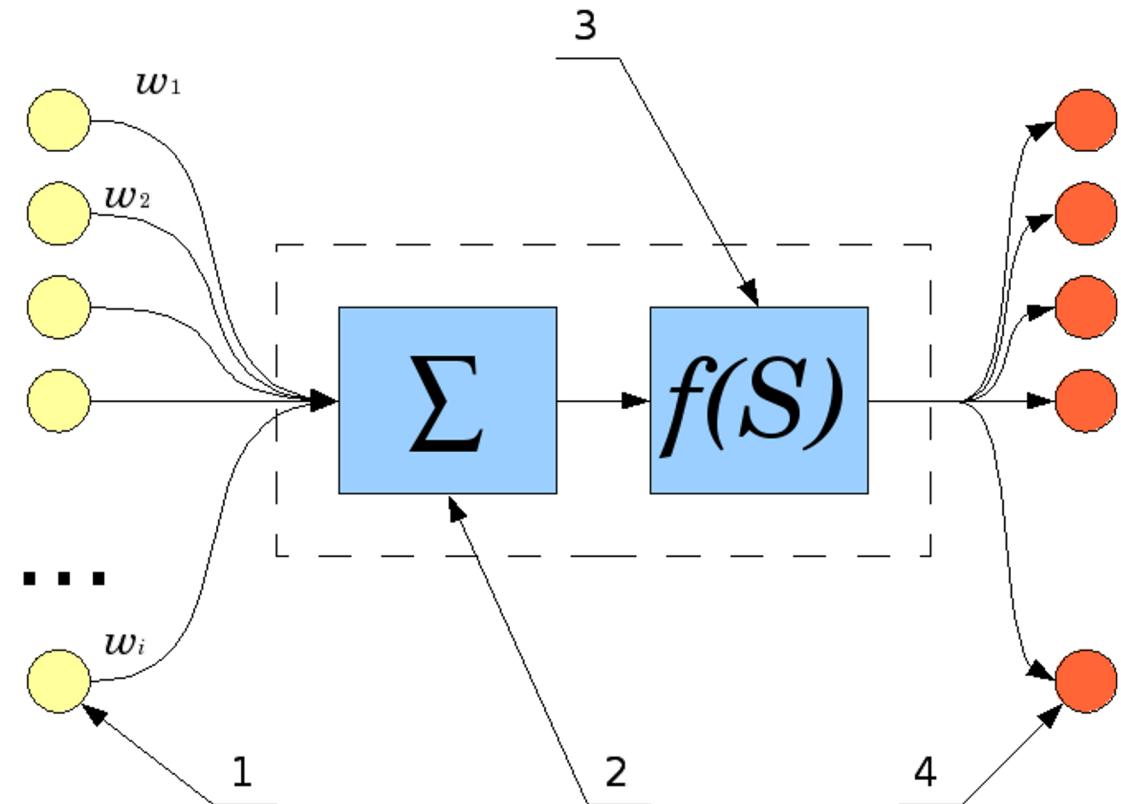
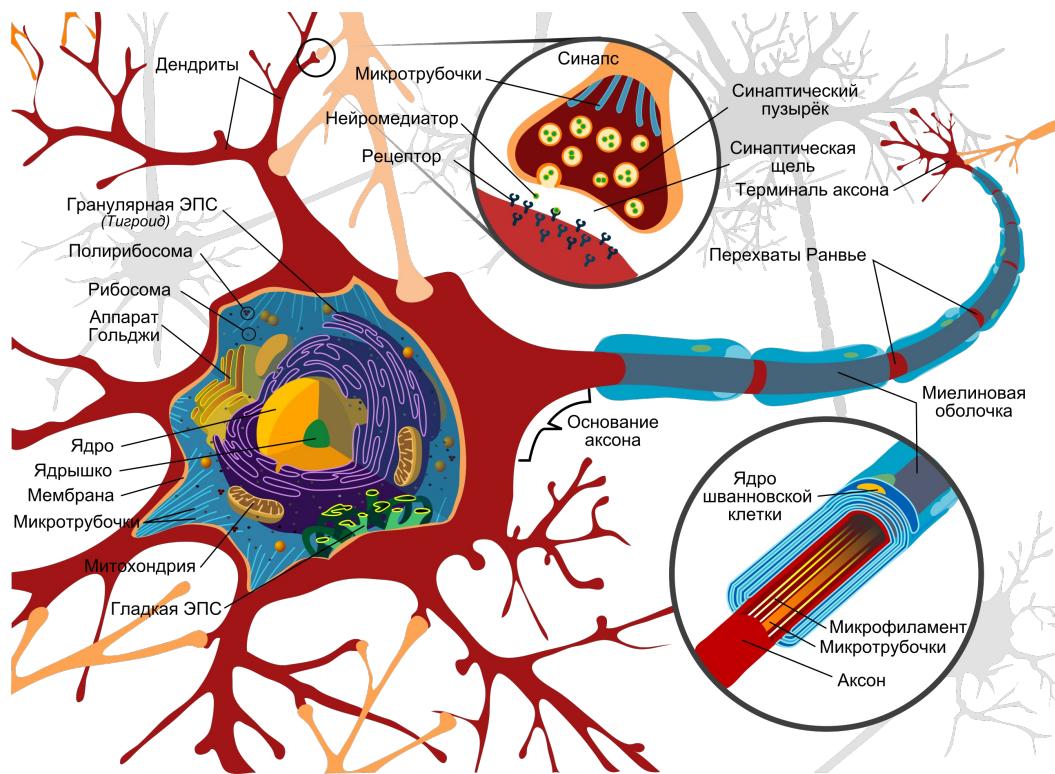
Лекция 7

Историческая справка

Ключевые этапы развития

- **1943:** Модель МакКаллока-Питтса — первый математический нейрон
- **1958:** Перцептрон Розенблатта — первая обучаемая модель
- **1969:** Книга “Перцептроны” Минского и Паперта — доказательство ограничений
- **1986:** Алгоритм обратного распространения ошибки (Румельхарт, Хинтон, Уильямс)
- **2006-2012:** Прорыв в глубоком обучении (предобучение, GPU, большие данные)
- **2012:** AlexNet — революция в компьютерном зрении
- **2017-н.в.:** Трансформеры и крупные языковые модели

Биологические основы



Биологический и искусственный нейрон

- Дендриты → входы (1)
- Синапсы → веса

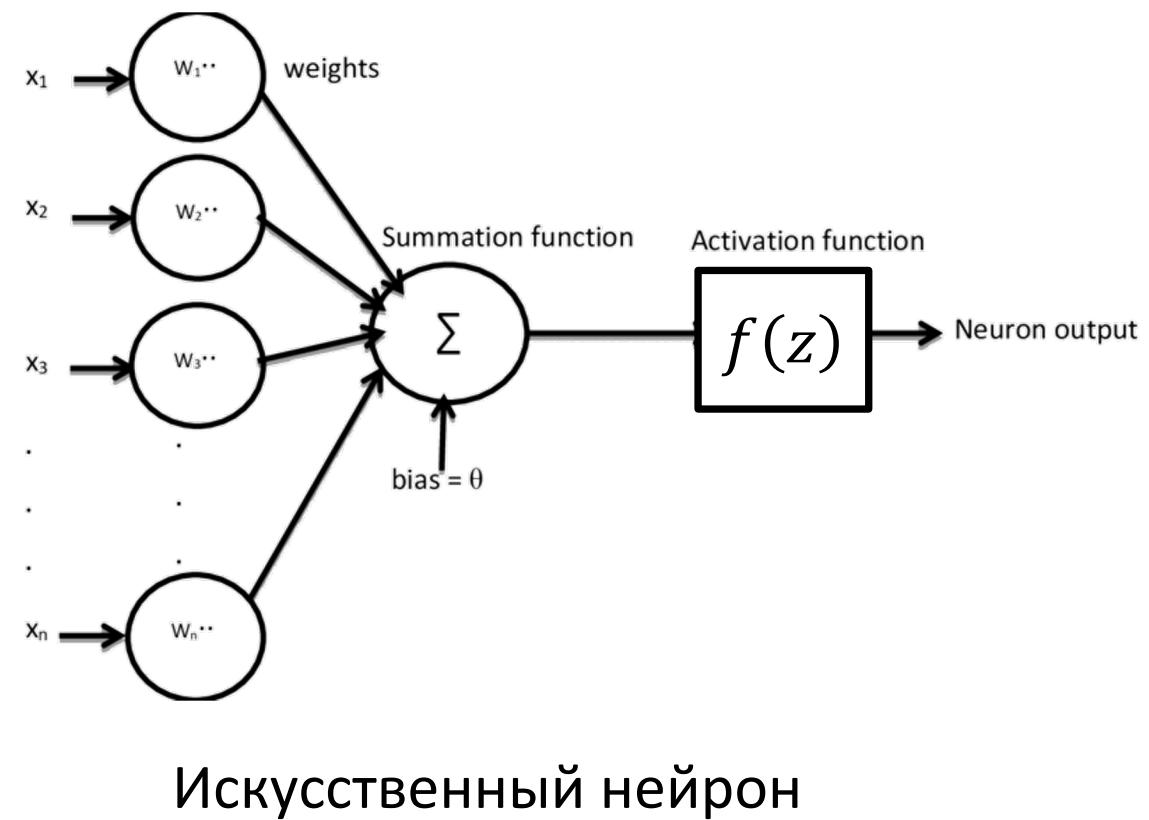
- Сома/ядро → сумматор (2)
- Аксон → выход (4)

Базовая структура нейрона

Математическая модель нейрона

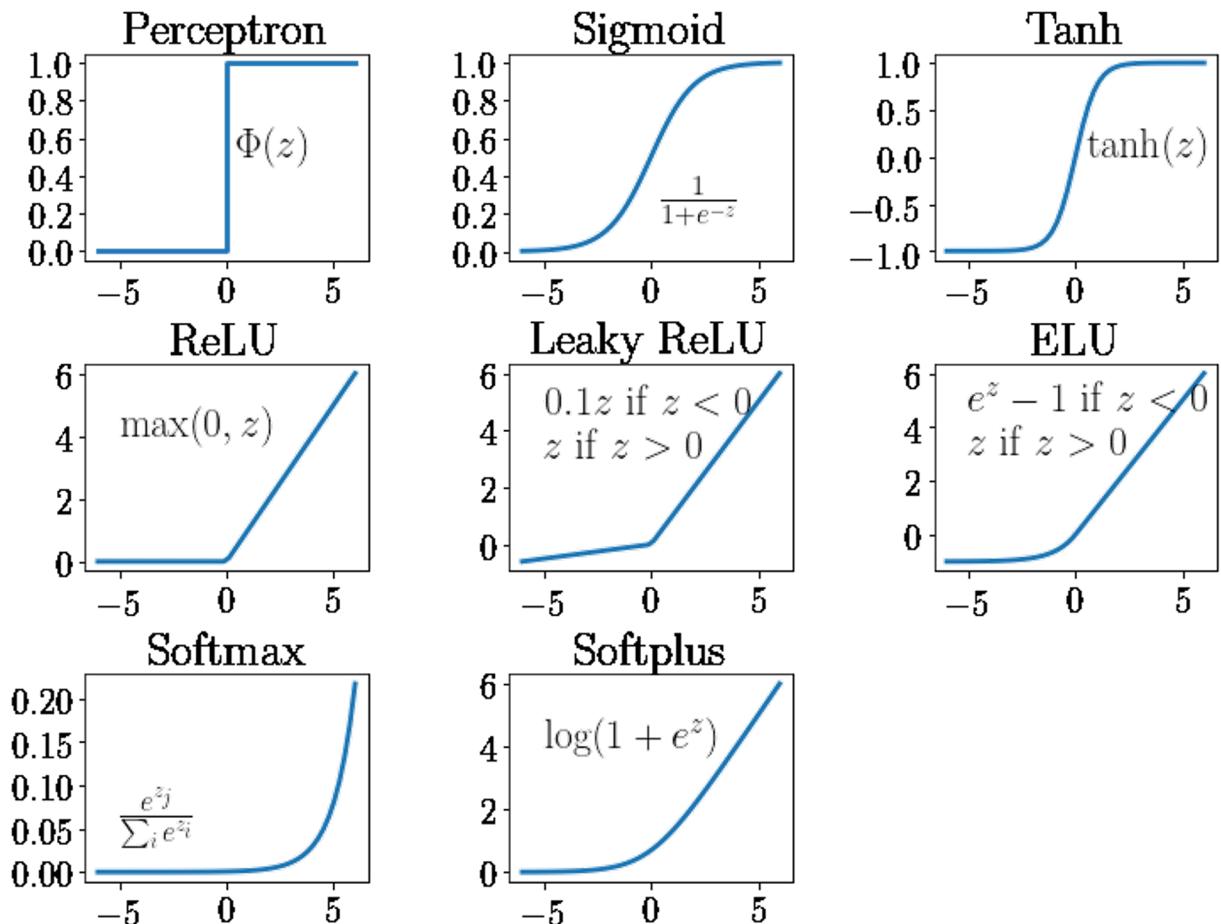
$$y = f \left(\sum_{i=1}^n w_i x_i + b \right)$$

- x_i — входные сигналы
- w_i — весовые коэффициенты
- b — смещение (bias)
- f — функция активации



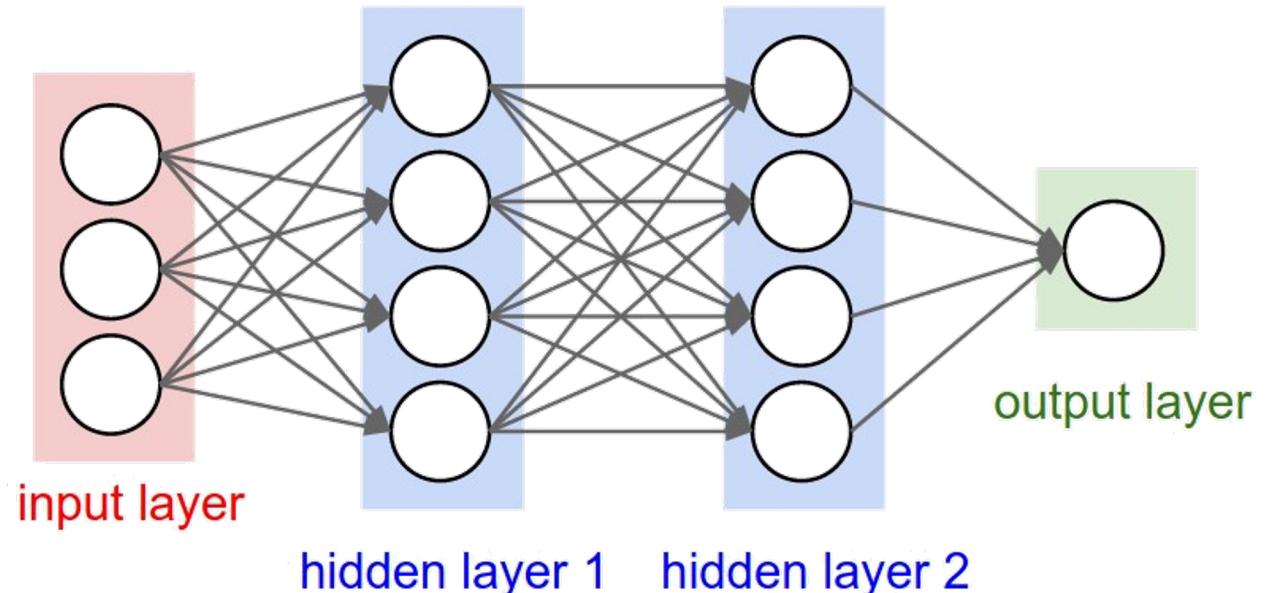
ФУНКЦИИ АКТИВАЦИИ

- **Сигмоида:** $\sigma(z) = \frac{1}{1+e^{-z}}$
- **Гиперболический тангенс:**
$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$
- **ReLU:** $\text{ReLU}(z) = \max(0, z)$
- **Leaky ReLU:** $\text{LeakyReLU}(z) = \max(\alpha z, z)$, α — малое число
- **Softmax:** $\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$



Структура слоев нейронной сети

- **Входной слой:** получает данные
- **Скрытые слои:** выполняют нелинейные преобразования
- **Выходной слой:** формирует результат



Прямое распространение (Forward Propagation)

1. Поступление входных данных \mathbf{X}
2. Для каждого слоя l :
 - Вычисление взвешенной суммы: $\mathbf{Z}^{(l)} = \mathbf{W}^{(l)} \cdot \mathbf{A}^{(l-1)} + \mathbf{b}^{(l)}$
 - Применение функции активации: $\mathbf{A}^{(l)} = f(\mathbf{Z}^{(l)})$
3. Выход последнего слоя — результат работы сети

Обучение нейронных сетей

Функции потерь

- Регрессия:

- Среднеквадратичная ошибка (MSE):

$$\frac{1}{n} \sum_{i=1}^n (y_{\text{true}} - y_{\text{pred}})^2$$

- Классификация:

- Бинарная кросс-энтропия:

$$-\frac{1}{n} \sum_{i=1}^n [y_{\text{true}} \log(y_{\text{pred}}) + (1 - y_{\text{true}}) \log(1 - y_{\text{pred}})]$$

- Категориальная кросс-энтропия:

$$-\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m y_{\text{true},ij} \log(y_{\text{pred},ij})$$

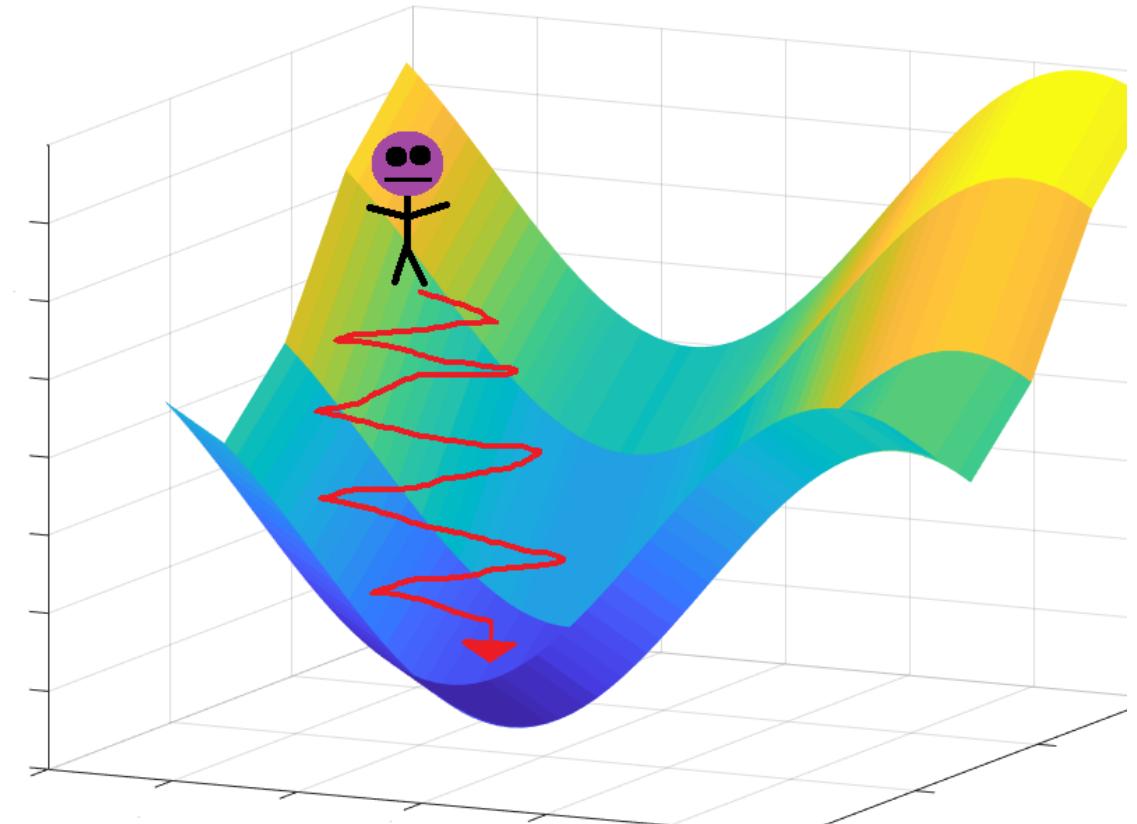
Градиентный спуск

Итеративное обновление
параметров:

$$W_{\text{new}} = W_{\text{old}} - \eta \nabla L(W)$$

Виды:

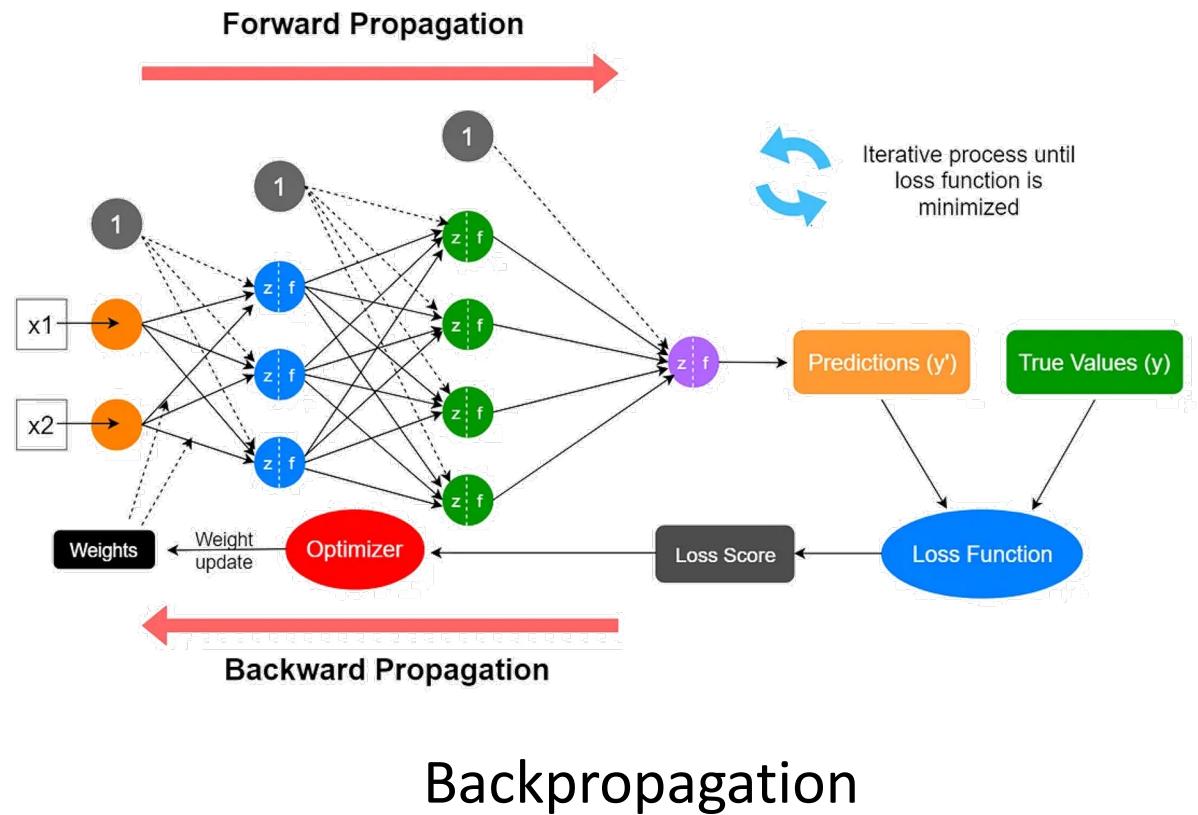
- **Пакетный** (весь набор данных)
- **Стохастический** (один пример)
- **Мини-пакетный** (подмножество)



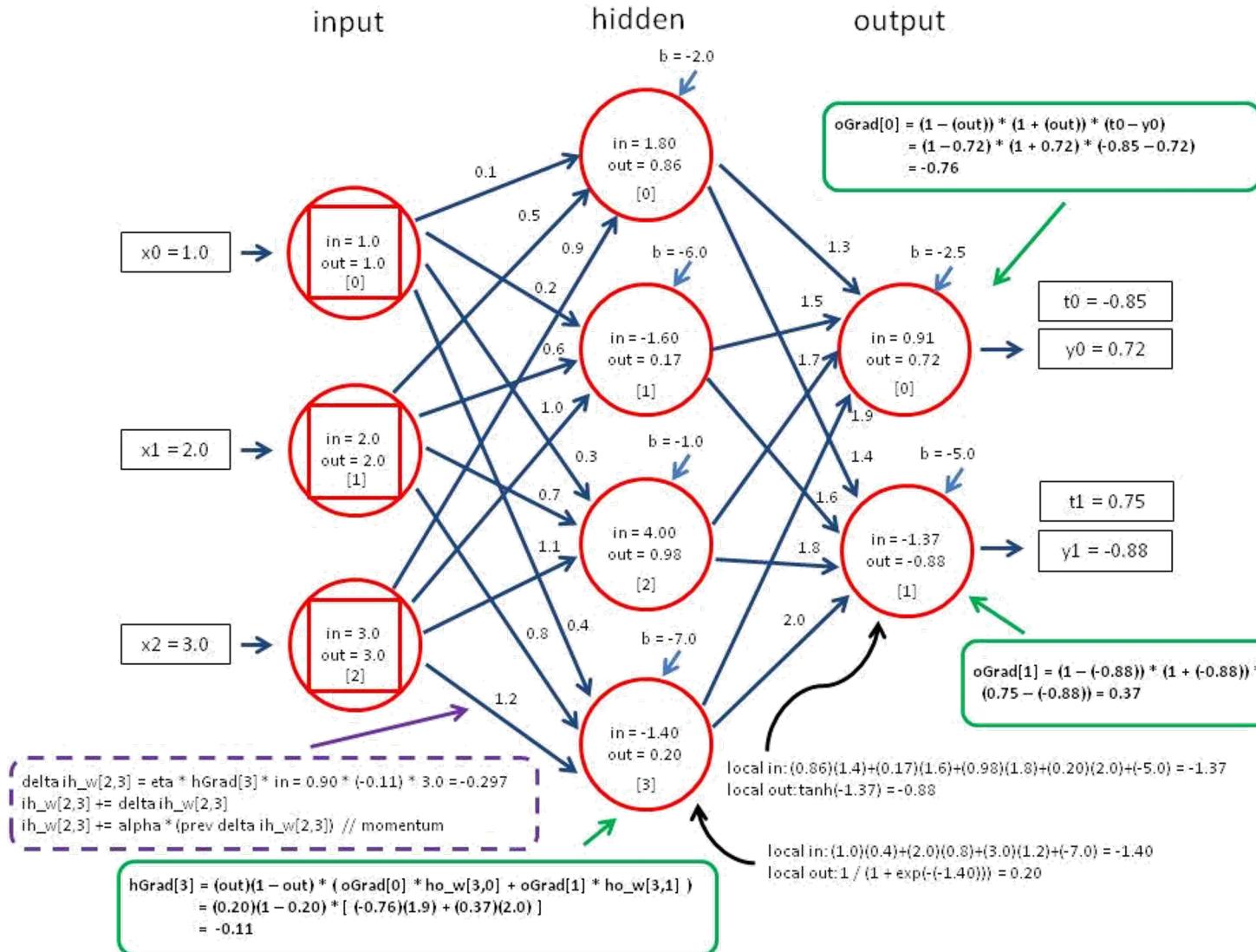
Градиентный спуск

Алгоритм обратного распространения ошибки

1. Прямой проход
(вычисление выхода)
2. Расчет ошибки на выходном слое
3. Распространение ошибки назад через сеть
4. Вычисление градиентов и обновление весов



Алгоритм обратного распространения ошибки



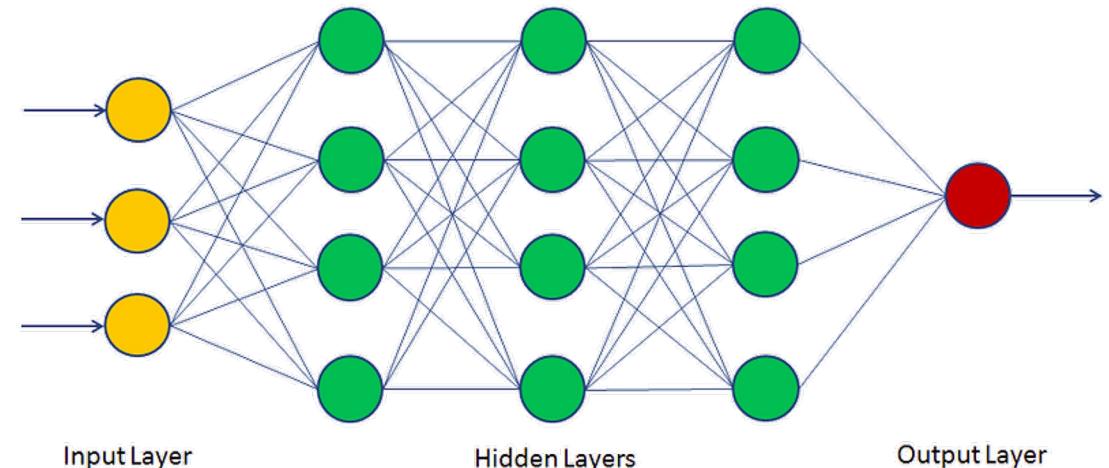
Оптимизаторы для нейронных сетей

- **SGD:** $W_t = W_{t-1} - \eta \nabla L(W)$
- **Momentum:**
 - $v_t = \gamma v_{t-1} + \eta \nabla L(W)$
 - $W_t = W_{t-1} - v_t$
- **RMSprop:**
 - $s_t = \beta s_{t-1} + (1 - \beta)(\nabla L(W))^2$
 - $W_t = W_{t-1} - \eta \frac{\nabla L(W)}{\sqrt{s_t + \epsilon}}$
- **Adam:** сочетает момент и адаптивную скорость обучения

Типы архитектур нейронных сетей

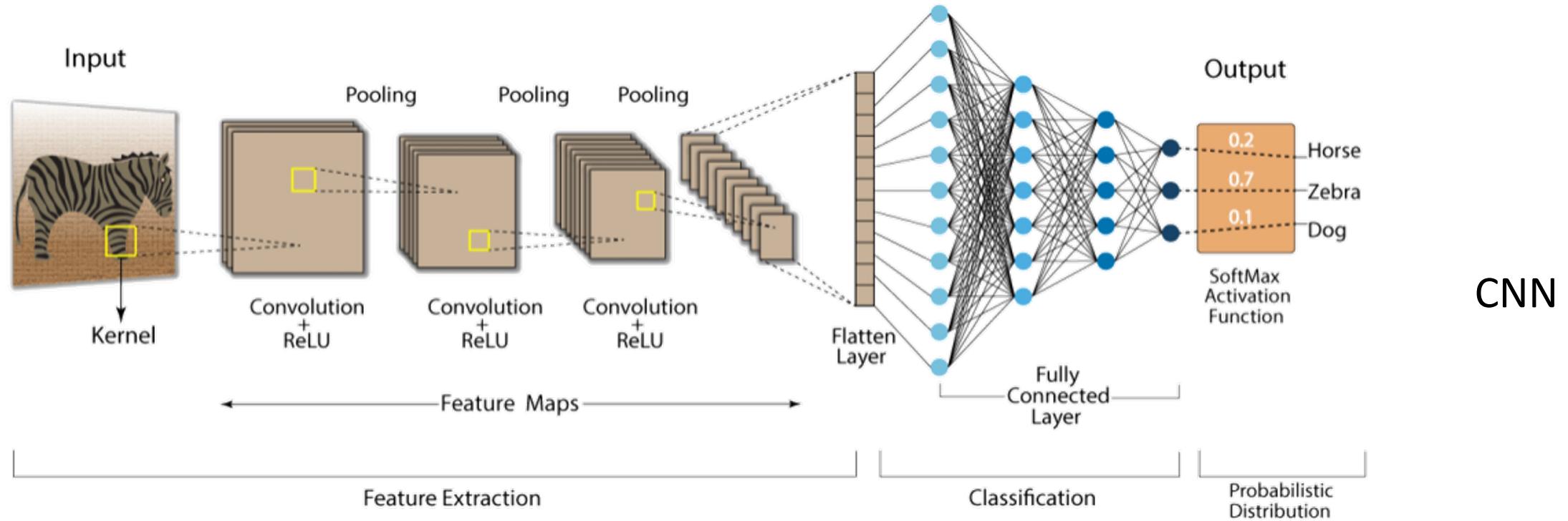
Полносвязные сети (MLP)

- Каждый нейрон связан со всеми нейронами предыдущего слоя
- Простая архитектура для табличных данных
- Не учитывает пространственную или временную структуру



multilayer perceptron (MLP)

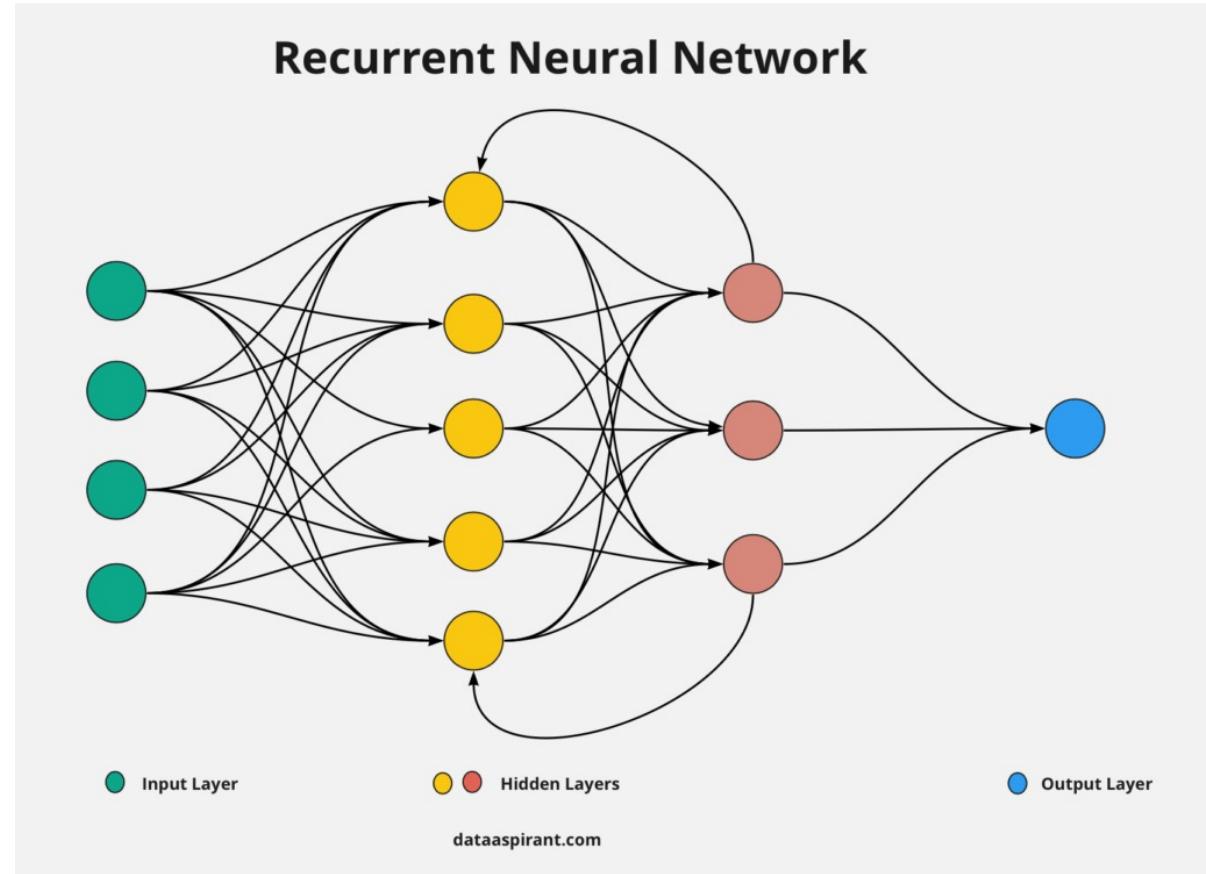
Сверточные нейронные сети (CNN)



- **Сверточные слои:** извлечение признаков с помощью фильтров
- **Пулинговые слои:** снижение размерности
- Локальные рецептивные поля и разделение параметров
- Идеальны для изображений и данных с сеточной структурой

Рекуррентные нейронные сети (RNN)

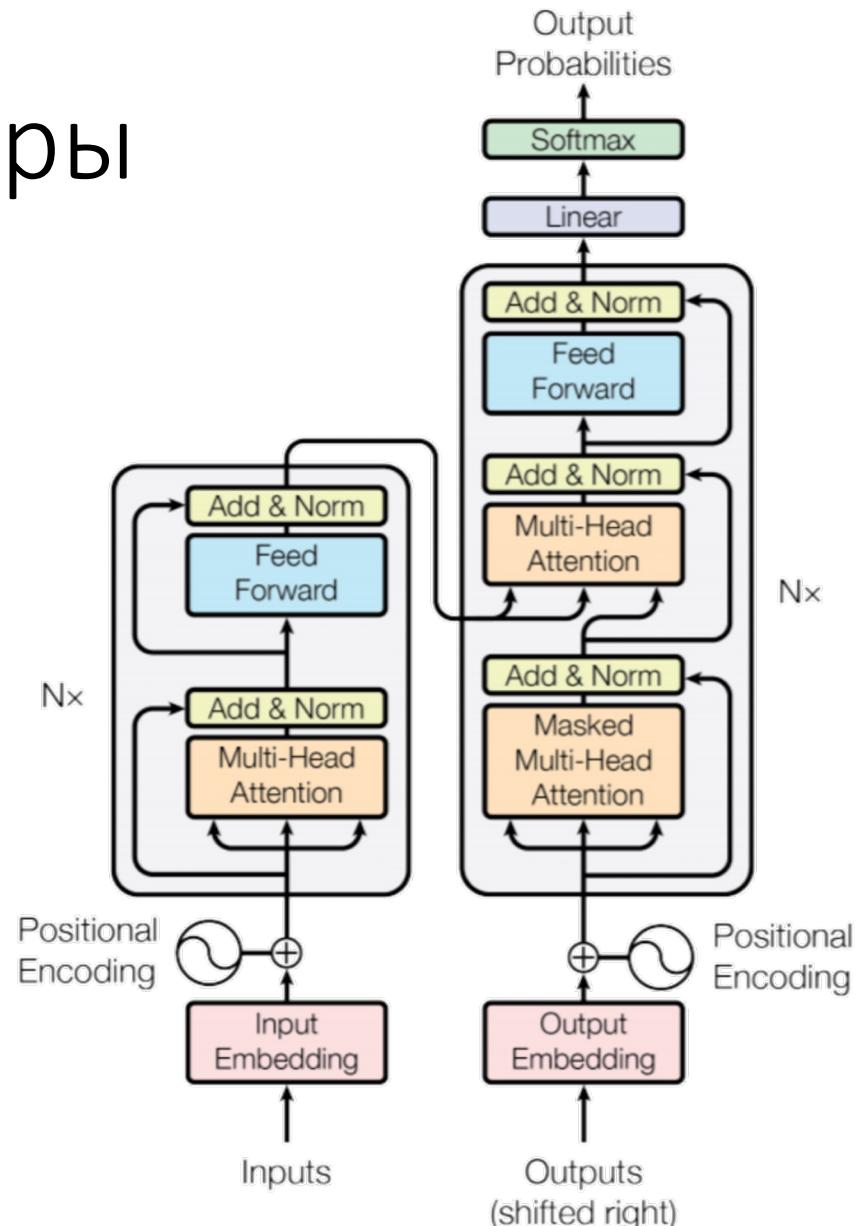
- Содержат циклические связи
- Сохраняют информацию о предыдущих входах
- Подходят для последовательных данных (текст, временные ряды)
- Улучшенные версии: LSTM, GRU



RNN

Трансформеры

- Основаны на механизме внимания (attention)
- Параллельная обработка всей последовательности
- Лучше улавливают дальние зависимости
- Превосходят RNN в обработке текста и других задачах



Transformer

Практические аспекты

Подготовка данных

- **Разделение данных:**

- Тренировочная выборка (60-80%)
- Валидационная выборка (10-20%)
- Тестовая выборка (10-20%)

- **Предобработка:**

- Нормализация/стандартизация числовых признаков
- Кодирование категориальных признаков
- Аугментация данных

Демонстрация простой нейронной сети

Задача: Классификация ирисов

```
import torch
import torch.nn as nn

class IrisNet(nn.Module):
    def __init__(self):
        super(IrisNet, self).__init__()
        self.layer1 = nn.Linear(4, 10)      # 4 входных признака -> 10 нейронов
        self.layer2 = nn.Linear(10, 8)      # 10 -> 8 нейронов
        self.layer3 = nn.Linear(8, 3)       # 8 -> 3 класса
        self.relu = nn.ReLU()
        self.dropout = nn.Dropout(0.2)

    def forward(self, x):
        x = self.relu(self.layer1(x))
        x = self.dropout(x)
        x = self.relu(self.layer2(x))
        x = self.layer3(x)
    return x
```

Преимущества и недостатки нейронных сетей

Преимущества:

- Моделирование сложных нелинейных зависимостей
- Автоматическое выделение признаков
- Универсальность
- Масштабируемость

Недостатки:

- Требовательность к данным и вычислениям
- Проблема “черного ящика”
- Сложность проектирования и настройки
- Склонность к переобучению

Области применения

- **Компьютерное зрение:** классификация и сегментация изображений
- **Обработка естественного языка:** машинный перевод, генерация текста
- **Обработка аудио:** распознавание и синтез речи
- **Временные ряды:** прогнозирование и аномалии
- **Рекомендательные системы**
- **Игры и обучение с подкреплением**
- **Наука и исследования:** биоинформатика, физика, астрономия