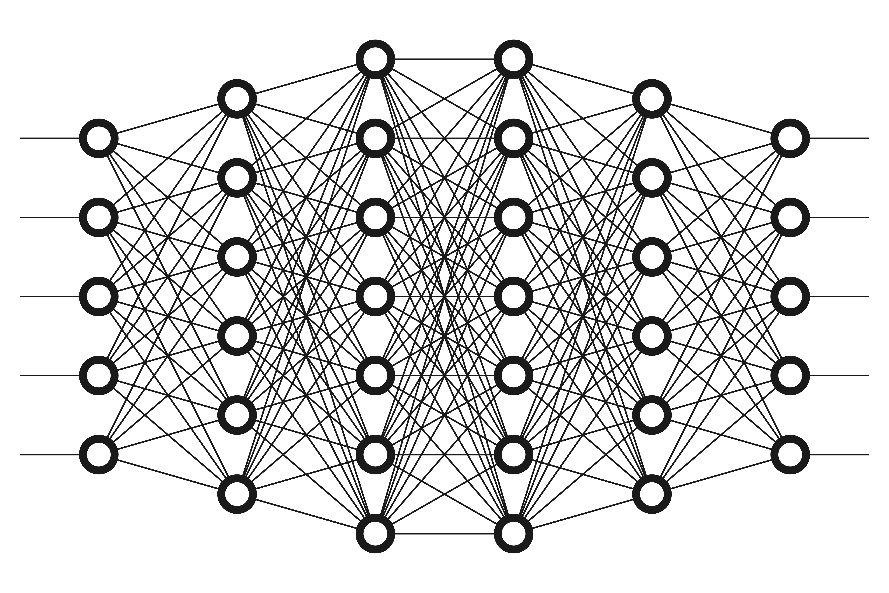
Маркелов Сергей, 1ПИб-02-3оп-22

**Вопросы к экзамену по дисциплине «Искусственные нейронные сети»**

1. Основные понятия нейронных сетей (НС): преимущества НС

**Нейронные сети (НС)** – это математические модели, вдохновленные биологическими нейронными сетями мозга. Они состоят из множества взаимосвязанных искусственных нейронов, которые обрабатывают информацию и обучаются на данных.



Основные понятия:

* Нейрон – базовая единица НС, принимает входные данные, умножает на веса, применяет функцию активации.
* Слой нейронов:
  + Входной слой – получает исходные данные (например, пиксели изображения).
  + Скрытые слои – промежуточные слои между входом и выходом (могут быть десятки или сотни).
  + Выходной слой – возвращает результат (например, класс объекта или число).
* Функция активации – определяет выход нейрона (ReLU, сигмоида, гиперболический тангенс и др.).
* Веса и смещения – параметры, которые настраиваются в процессе обучения.

Преимущества нейронных сетей:

1. Способность к обучению сложным закономерностям – НС могут выявлять нелинейные зависимости в данных, что позволяет их использовать для анализа изображений, текста, звука и др.
2. Адаптивность и универсальность – один и тот же алгоритм может применяться в разных областях: от распознавания лиц до медицинской диагностики.
3. Устойчивость к шумам и искажениям – НС могут работать с зашумленными данными (например, частично повреждённые изображения).
4. Автоматическое извлечение признаков – в отличие от классического машинного обучения, где признаки выбираются вручную, НС сами учатся выделять важные характеристики данных.
5. Высокая точность в сложных задачах - в задачах компьютерного зрения, обработки естественного языка и других сложных областях НС часто превосходят традиционные алгоритмы.
6. Параллельная обработка данных – благодаря архитектуре НС и использованию GPU, обучение и предсказание можно эффективно распараллеливать.
7. Масштабируемость – чем больше данных и мощнее архитектура, тем лучше работает нейросеть (в отличие от некоторых классических алгоритмов, которые упираются в пределы точности).
8. Основные понятия нейронных сетей (НС): модели нейронов

*Определение НС и основные понятия из вопр. 1*

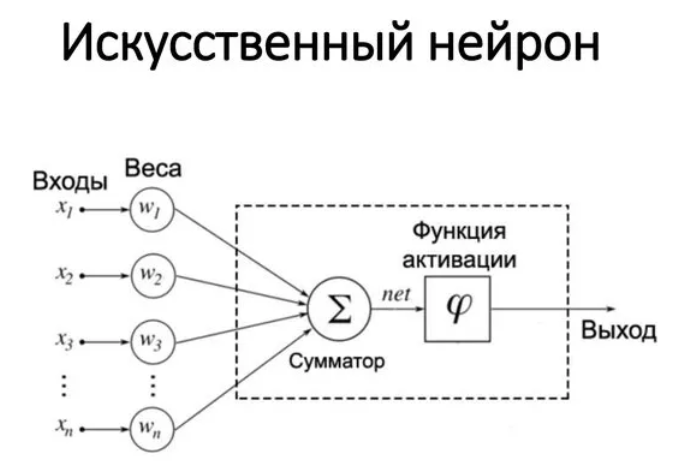
**Нейрон** – это базовая вычислительная единица нейронной сети, вдохновленная биологическим нейроном. Он принимает входные сигналы, обрабатывает их и передает выход дальше.

Простейшая модель нейрона (перцептрон) включает:

* Входы *x1*, *x2*, ..., *xn*​ – признаки данных или выходы других нейронов
* Веса *w1*, *w2*, ..., *wn*​ – коэффициенты важности каждого входа
* Смещение *b* – порог активации
* Функция активации *f* – определяет выход нейрона

Формула выхода нейрона:

где – взвешенная сумма входов, *f* – сумма активации



Разновидности моделей нейронов:

1. Обычный (полносвязный) нейрон:
   * Каждый вход связан с каждым нейроном следующего слоя
   * Используется в классических многослойных перцептронах (MLP)
2. Нейрон в сверточной сети (CNN):
   * Принимает не все входы, а только локальную область (например, часть изображения)
   * Использует ядро свёртки (фильтр), которое «скользит» по данным
   * Позволяет выявлять локальные паттерны (например, границы на изображении)
3. Рекуррентный нейрон (RNN)
   * Имеет память – учитывает предыдущие состояния
   * Используется в последовательностях (текст, временные ряды)
4. Нейрон с механизмом внимания
   * Вместо фиксированных связей динамически вычисляет внимание (какие входы важнее)
   * Используется в современных архитектурах (например, GPT, BERT)
5. Основные понятия нейронных сетей (НС): направленные графы для представления нейронных сетей

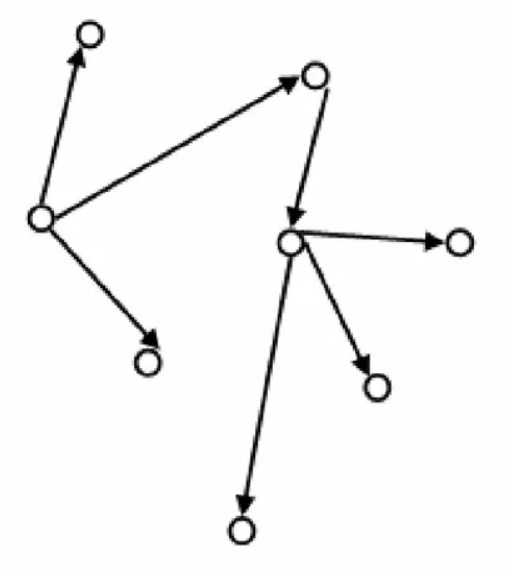
*Определение НС и основные понятия из вопр. 1*

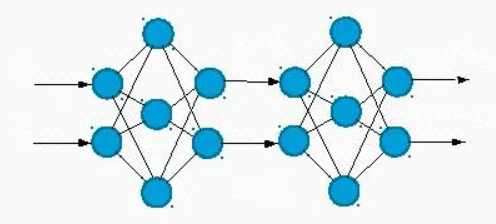
Нейронные сети можно представить в виде **направленных графов**, где:

* Вершины – нейроны или слои
* Рёбра – передача данных с весами

Этот подход позволяет наглядно описывать архитектуру НС и анализировать поток информации.

Типы графов в нейронных сетях:

* Ациклические направленные графы. В этих графах нет циклов, информация движется только вперёд. Подходит для задач классификации и регрессии. Примеры: полносвязные сети (MLP), свёрточные сети (CNN), трансформеры (без рекуррентных связей)
* Рекуррентные графы. Есть циклы, информация передаётся между шагами последовательности. Используется для обработки временных рядов, текста. Примеры: RNN, LSTM, GRU



Преимущества графового представления

* Наглядность – легко визуализировать архитектуру
* Анализ зависимостей – можно отследить поток данных
* Автоматическое дифференцирование – графы используются в фреймворках (TensorFlow, PyTorch) для расчёта градиентов
* Гибкость – можно комбинировать разные типы слоёв

1. Основные понятия нейронных сетей (НС): обратная связь

*Определение НС и основные понятия из вопр. 1*

**Обратная связь** – это механизм передачи информации от последующих слоёв или временных шагов к предыдущим.

Типы обратной связи:

* Обратное распространение ошибки – алгоритм обучения, при котором ошибка распространяется от выходного слоя к входному для корректировки весов.

Формула обновления весов:

где *η* – скорость обучения, – градиент ошибки по весу.

Пример: обучение многослойного перцептрона (MLP).

* Рекуррентные связи (в RNN, LSTM, GRU) – выход нейрона на предыдущем шаге передаётся на вход на следующем шаге.

Формула для RNN:

где *ht​* — скрытое состояние на шаге *t*, *Whh*, *Wxh* – матрицы весов.

Пример: Обработка текстов или временных рядов.

* Обратные связи в архитектурах с памятью (например, LSTM) – специальные механизмы, регулирующие поток информации.

Компоненты LSTM:

* + Forget gate – решает, какую информацию забыть.
  + Input gate – обновляет состояние ячейки.
  + Output gate – определяет выходное значение.
* Обратные связи в сетях с вниманием – механизм, позволяющий сети фокусироваться на важных частях входных данных. Пример: Трансформеры, где веса внимания пересчитываются на каждом шаге.

1. Основные понятия нейронных сетей (НС): архитектура НС

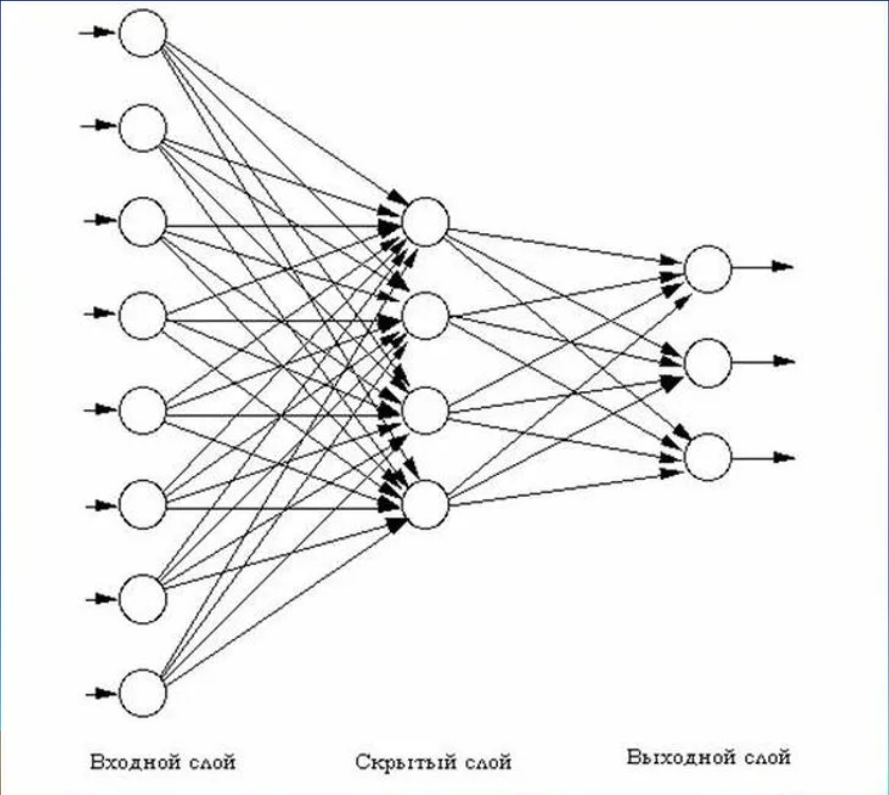
*Определение НС и основные понятия из вопр. 1*

**Архитектура нейронной сети** определяет:

* Структуру слоев и их взаимосвязи
* Типы нейронов и функции активации
* Поток данных (прямой/обратный)
* Специализацию под конкретные задачи

Основные компоненты архитектуры:

* Входной слой – получает исходные данные. Количество нейронов равно количеству признаков
* Скрытые слои – производят преобразование данных, могут быть разных типов (полносвязные, сверточные и др.)
* Выходной слой – формирует конечный результат. Зависит от задачи (классификация, регрессия и т. д.)



Основные типы архитектур:

* Полносвязные сети (MLP) – состоят из последовательности плотных слоев. Применяются для табличных данных и простых задач классификации.
* Сверточные сети (CNN) – содержат сверточные слои для выделения признаков, пулинг-слои для уменьшения размерности и полносвязные слои для классификации. Применяются для обработки изображений и видео.
* Рекуррентные сети (RNN/LSTM/GRU) – имеют циклические связи для запоминания состояния. Используются для работы с временными рядами, текстом и речью.
* Автокодировщики состоят из энкодера для сжатия данных и декодера для восстановления. Применяются для сжатия данных и уменьшения шума.
* Трансформеры используют механизм внимания (self-attention) и позиционное кодирование. Широко применяются в NLP и компьютерном зрении.

1. Основные понятия нейронных сетей (НС): представление знаний

*Определение НС и основные понятия из вопр. 1*

**Представление знаний в нейронных сетях** – это способ кодирования информации, который отражает взаимосвязи в данных, позволяет сети делать выводы и определяет, как входные данные преобразуются в полезные признаки.

Основные подходы к представлению знаний:

* Распределённые представления – информация кодируется паттернами активации множества нейронов. Пример: word2vec в NLP, где слова представлены векторами
* Локальные представления – конкретный нейрон или группа отвечает за определённый признак. Пример: классические алгоритмы машинного обучения
* Гибридные подходы – сочетают преимущества распределённых и локальных представлений. Пример: современные трансформеры в NLP

Уровни представления знаний

* Низкоуровневые признаки – выявляются начальными слоями сети. Пример: края и текстуры в CNN
* Высокоуровневые абстракции – формируются глубокими слоями. Пример: объекты целиком в компьютерном зрении

Методы представления знаний:

* Векторные представления – слова, изображения, пользователи представляются как векторы. Пример: GloVe, BERT-эмбеддинги
* Матричные представления – используются в рекомендательных системах. Пример: разложение матриц
* Графовые представления - знания представляются в виде графов. Пример: графовые нейронные сети (GNN)

1. Нейронные сети и искусственный интеллект

Нейронные сети и искусственный интеллект (ИИ) – это две взаимосвязанные области, которые играют ключевую роль в современной компьютерной науке и технологиях.

**ИИ** – это широкая область, изучающая создание машин и алгоритмов, способных выполнять задачи, требующие человеческого интеллекта:

* Обучение (машинное обучение, глубокое обучение)
* Логика и планирование (например, алгоритмы игры в шахматы)
* Обработка естественного языка (NLP) (чат-боты, переводчики)
* Компьютерное зрение (распознавание изображений, лиц)
* Робототехника (автономные роботы)

**Нейронные сети** – это математические модели, вдохновленные биологическими нейронами мозга. Они используются в глубоком обучении – подразделе машинного обучения.

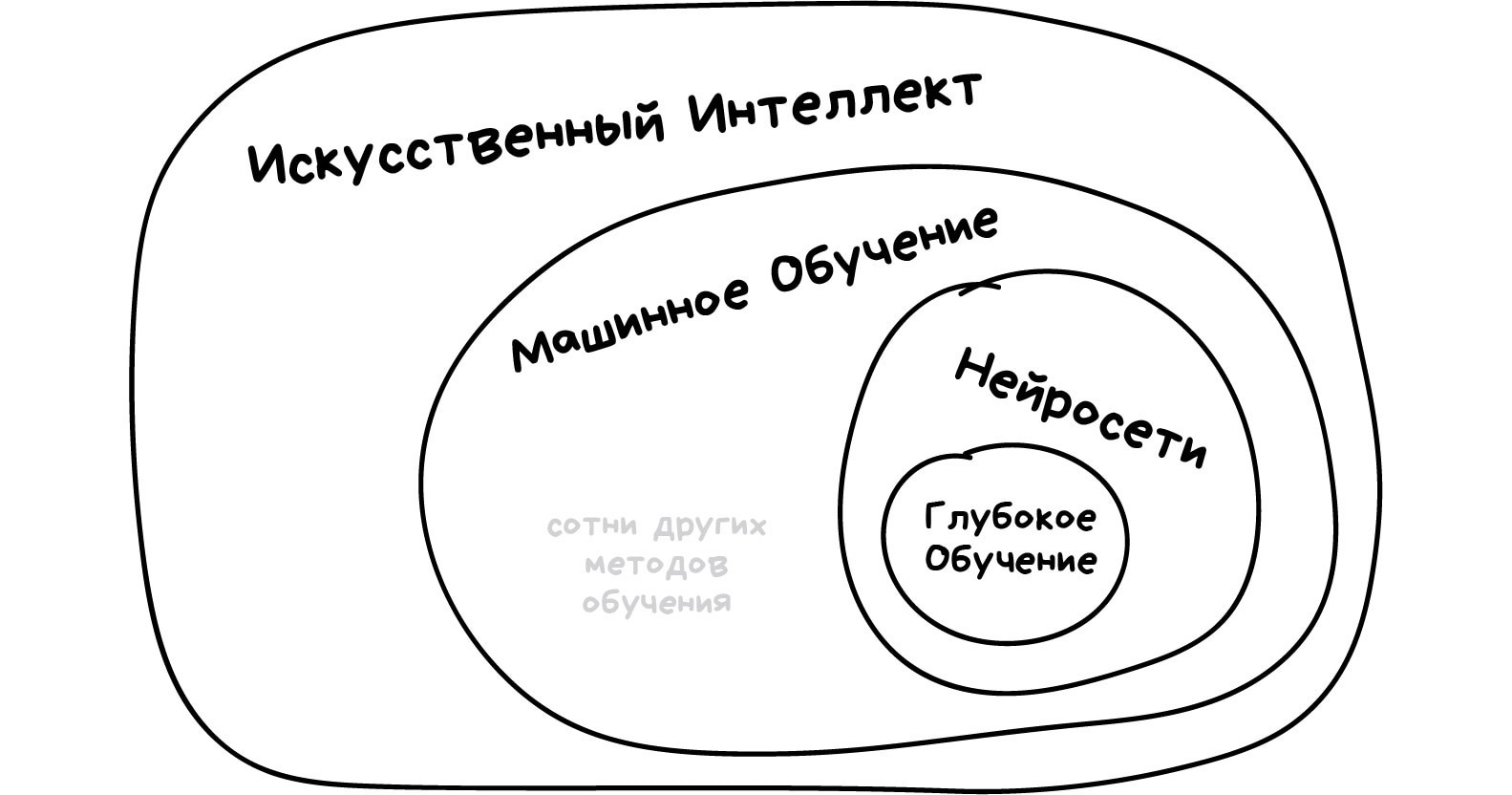
Основные типы нейронных сетей:

* Полносвязные сети – базовые модели для классификации
* Сверточные нейронные сети (CNN) – для обработки изображений и видео
* Рекуррентные нейронные сети (RNN, LSTM, GRU) – для работы с последовательностями (текст, речь, временные ряды)
* Трансформеры – современные архитектуры для NLP (например, GPT, BERT).
* Генеративно-состязательные сети (GAN) – для генерации изображений, музыки и др.

Нейронные сети – это инструмент в арсенале ИИ. Не все ИИ используют нейросети (есть и другие методы: деревья решений, SVM, экспертные системы). Но именно глубокое обучение на нейросетях дало огромный прорыв в последние 10 лет (распознавание речи, генерация текста, беспилотные автомобили).

Примеры применения:

* ChatGPT, Gemini, Claude – языковые модели на трансформерах
* MidJourney, Stable Diffusion – генерация изображений с помощью GAN и диффузионных моделей
* Автопилот Tesla – компьютерное зрение + нейросети



1. Основные подходы к обучению: обучение, основанное на коррекции ошибок

**Обучение нейронных сетей** – это процесс настройки их параметров (весов и смещений) для минимизации ошибки предсказания. Существует несколько ключевых подходов к обучению:

* Обучение, основанное на коррекции ошибок
* Обучение на основе памяти
* Обучение Хебба
* Конкурентное обучение
* Обучение Больцмана
* Обучение с учителем
* Обучение без учителя

**Обучение нейронной сети на основе коррекции ошибок** – это ключевой принцип работы многих алгоритмов машинного обучения. Основная идея заключается в том, чтобы минимизировать разницу между предсказаниями модели и правильными ответами (целевыми значениями).

Основные этапы обучения:

1. Прямое распространение – входные данные проходят через нейросеть, и на выходе получается предсказание
2. Вычисление ошибки – сравнивается предсказание с правильным ответом (например, через MSE или Cross-Entropy)
3. Обратное распространение – ошибка передаётся назад по сети, вычисляются градиенты (производные) по каждому весу
4. Коррекция весов – веса обновляются, чтобы уменьшить ошибку (например, градиентным спуском)

Методы коррекции ошибок

* Градиентный спуск – обновление весов по формуле

где *η* – скорость обучения, *L* – функция потерь

* Стохастический градиентный спуск (SGD) – коррекция весов на каждом отдельном примере
* Алгоритмы адаптивной оптимизации – автоматически подстраивают скорость обучения для каждого параметра

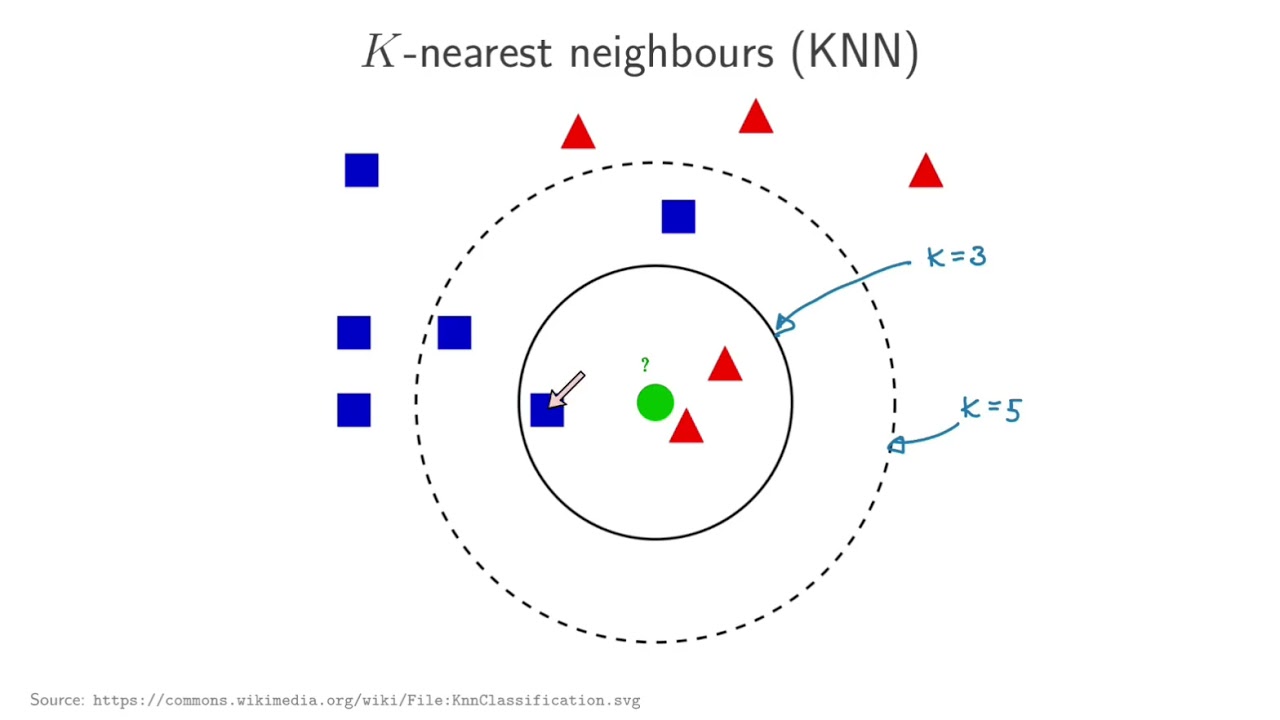
1. Основные подходы к обучению: обучение на основе памяти

*Определение и перечисление основных подходов из вопр. 8*

**Обучение нейросетей на основе памяти** – это подход, при котором нейросеть использует хранимую информацию (память) для улучшения предсказаний. В отличие от классического обучения с обратным распространением ошибки, здесь модель может запоминать важные примеры из обучающих данных, быстро адаптироваться к новым данным без переобучения и использовать внешнюю память.

Основные методы:

* k-NN (k-ближайших соседей) – классификация на основе похожих примеров из обучающей выборки



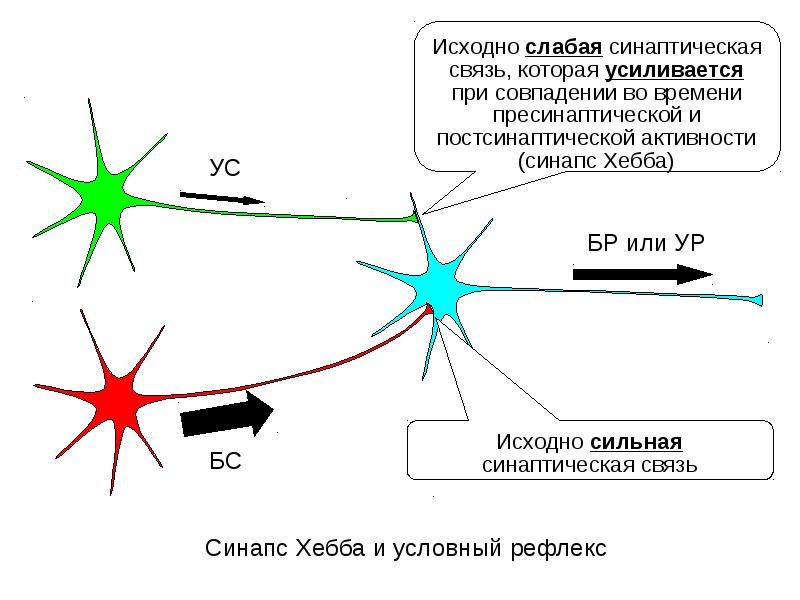
* LSTM и GRU — рекуррентные сети с памятью для последовательностей
* Neural Turing Machine (NTM) – нейросеть + внешняя память (аналог RAM)
* Differentiable Neural Computer (DNC) – улучшенная NTM с более сложным доступом к памяти
* MAML (Model-Agnostic Meta-Learning) – модель учится быстро адаптироваться к новым задачам
* Memory-Augmented Neural Networks (MANN) – нейросети, использующие память для быстрого обучения

1. Основные подходы к обучению: обучение Хебба

*Определение и перечисление основных подходов из вопр. 8*

**Правило Хебба** – один из старейших и биологически правдоподобных алгоритмов обучения нейронных сетей. Основано на принципе: нейроны, которые активируются вместе, связываются вместе. Если 2 нейрона часто активируются одновременно, сила связи (синаптический вес) между ними увеличивается:

где: *wij* – вес между нейроном *i* и *j*, *η* – скорость обучения, *xi​* – входной сигнал, *yj​* – выходной сигнал.



Применение:

* Ассоциативные памяти (например, сеть Хопфилда) – запоминает бинарные образы через матрицу весов. Способна восстанавливать образы по частичному или зашумлённому входу.
* Самоорганизующиеся карты (Kohonen SOM) – применяется для визуализации и кластеризации высокоразмерных данных.
* Гибридные модели (например, Hebbian-Autoencoders) – комбинация с обратным распространением ошибки для улучшения обучения глубоких сетей.

1. Основные подходы к обучению: конкурентное обучение

*Определение и перечисление основных подходов из вопр. 8*

**Конкурентное обучение** – это тип неконтролируемого обучения, при котором нейроны сети соревнуются за право активироваться в ответ на входные данные. В результате только один или несколько нейронов (победителей) обновляют свои веса, что приводит к самоорганизации сети.

Ключевые идеи

* «Победитель получает всё» – только нейрон, наиболее близкий к входному образцу, обновляет свои веса
* Латеральное торможение – активированный нейрон подавляет активность соседей (аналог биологического торможения)
* Самоорганизация – сеть автоматически находит кластеры или прототипы в данных.

Формула для нейрона-победителя j, ближайшего к входу x:

где: *wj* – веса нейрона-победителя, *η* – скорость обучения.

Основные модели:

* Сети Кохонена –проецирование высокоразмерных данных на 2D-сетку с сохранением топологии. Каждый нейрон имеет весовой вектор *wi*​. Среди них находится нейрон-победитель (с наименьшим расстоянием до входа). После этого обновляются веса победителя и его соседей (по Гауссову ядру). Применение: визуализация данных, кластеризация, анализ рынков
* Learning Vector Quantization – классификация на основе прототипов. Каждый класс представлен набором нейронов-прототипов. Нейрон-победитель обновляется. Если класс верный: . Если класс неверный:
* ART-сети – динамическое добавление новых кластеров без катастрофического забывания. Использует порог сходства для создания новых нейронов при несоответствии.

Применение:

* Кластеризация (анализ покупательского поведения, биологические данные)
* Сжатие данных (векторное квантование в JPEG, MP3)
* Биоинформатика (классификация ДНК-последовательностей)
* Робототехника (адаптивное управление)

1. Основные подходы к обучению: обучение Больцмана

*Определение и перечисление основных подходов из вопр. 8*

**Обучение Больцмана** – это стохастический метод обучения для нейронных сетей с симметричными связями, основанный на принципах статистической физики. Используется в машинах Больцмана и ограниченных машинах Больцмана.

Ключевые особенности

* Стохастические нейроны: активация нейронов определяется вероятностно
* Энергетическая функция: состояние сети описывается через энергию (аналогично физическим системам)
* Двухфазное обучение: чередование «положительной» и «отрицательной» фаз

Функция энергии для состояния нейронов :

где *wij*​ – веса, *bi* – смещения

Вероятность состояния:

где *Z* — нормировочная сумма (статсумма), *T* — «температура».

Алгоритм обучения (метод Контрастивной дивергенции, CD-k):

1. Положительная фаза:
   * Фиксируются видимые нейроны на обучающих данных
   * Скрытые нейроны обновляются по вероятности *P(h*∣*v)*
   * Измеряются корреляции *⟨vihj⟩data*
2. Отрицательная фаза:
   * Сеть «свободно» эволюционирует из случайного состояния.
   * После *k* шагов Гиббса измеряются корреляции *⟨vihj⟩model*
3. Обновление весов: *Δwij=η(⟨sisj⟩data−⟨sisj⟩model)*

Применение:

* Ограниченная машина Больцмана (RBM) – связи только между видимым и скрытым слоями (нет внутрислойных). Обучение через CD-k (часто *k = 1*). Используется в предобучении глубоких сетей, уменьшении размерности данных, рекомендательных системах
* Deep Belief Networks (DBN) – каскад из нескольких RBM. Обучение «снизу вверх» (жадный алгоритм)

1. Основные подходы к обучению: обучение с учителем

*Определение и перечисление основных подходов из вопр. 8*

**Обучение с учителем** – это метод машинного обучения, при котором алгоритм обучается на размеченных данных (пар «вход-выход»). Цель – научиться предсказывать правильные выходные значения для новых, ранее не встречавшихся входных данных.

Ключевые компоненты

* Обучающая выборка – множество примеров *{(x1, y1), (x2, y2), ..., (xN, yN)}*, где *xi​* – входной вектор признаков, *yi* – целевое значение (метка)
* Модель – функция *f(x)*, которую нужно настроить так, чтобы *f(xi)≈yi*
* Функция потерь – мера ошибки предсказания (например, MSE, Cross-Entropy)
* Алгоритм оптимизации – метод обновления параметров модели для минимизации ошибки (например, градиентный спуск)

Основные задачи:

* Классификация – отнесение объекта к одному из классов. Примеры: распознавание изображений (кошка/собака), спам-фильтр (спам/не спам). Методы:
  + Логистическая регрессия
  + Решающие деревья
  + SVM (Support Vector Machines)
  + Нейронные сети
* Регрессия – предсказание непрерывного значения. Примеры: прогнозирование цены дома, предсказание температуры. Методы:
  + Линейная регрессия
  + Метод опорных векторов (SVR)
  + Деревья решений

Этапы обучения модели:

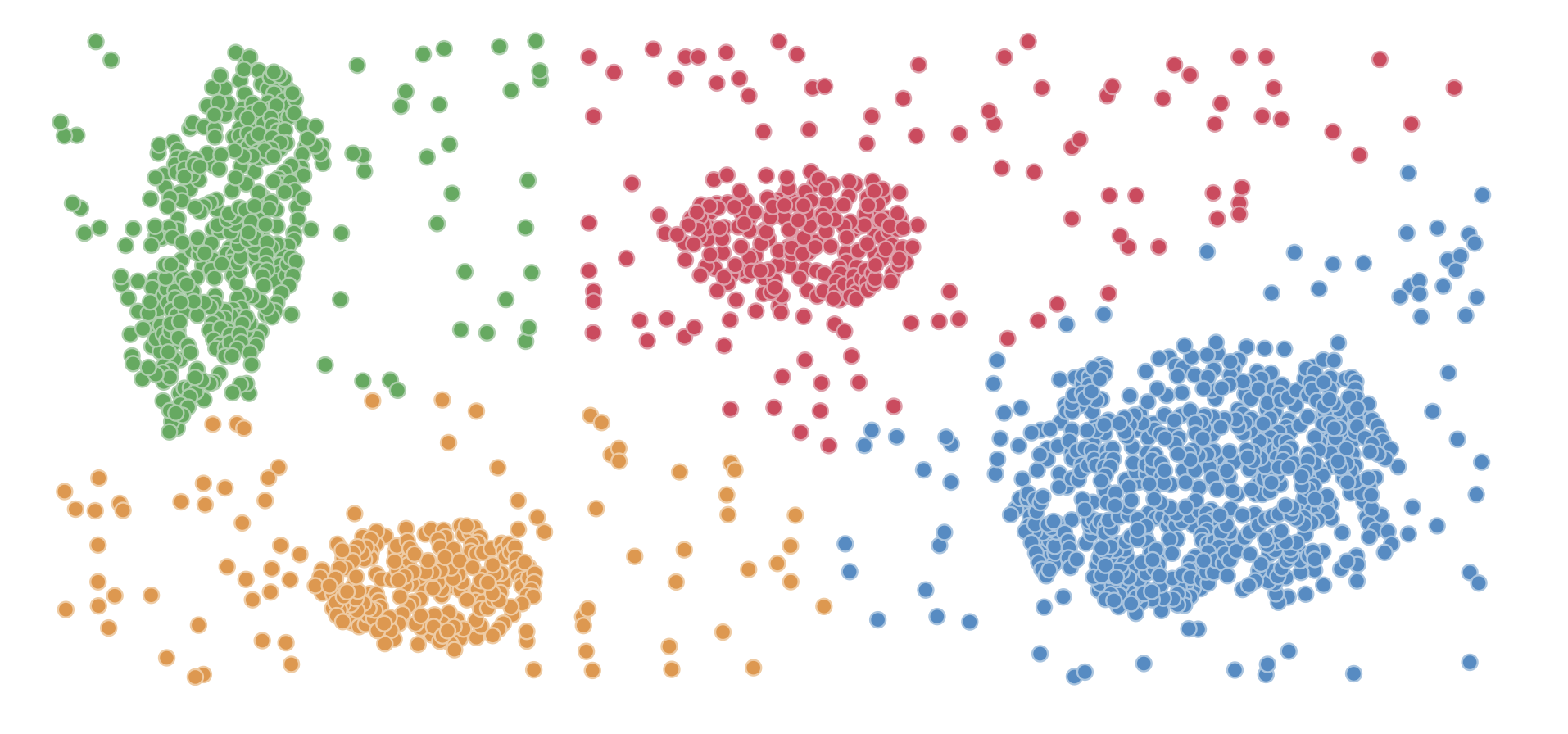
1. Подготовка данных – нормализация/стандартизация признаков, разделение на обучающую и тестовую выборки
2. Выбор модели – линейные методы, если зависимость линейна, или ансамбли, если данные сложные
3. Обучение – минимизация функции потерь на обучающих данных
4. Оценка – проверка качества на тестовой выборке
5. Основные подходы к обучению: обучение без учителя

*Определение и перечисление основных подходов из вопр. 8*

**Обучение без учителя** – это метод машинного обучения, при котором алгоритм анализирует данные без меток и самостоятельно выявляет скрытые структуры, закономерности или аномалии. В отличие от обучения с учителем, здесь нет правильных ответов, а цель – исследовать данные и найти в них полезные паттерны.

Основные задачи:

* Кластеризация – разделение данных на группы (кластеры) так, чтобы объекты внутри одного кластера были похожи, а между разными – отличались. Примеры: сегментация клиентов по поведению, группировка документов по темам. Методы:
  + K-means
  + Иерархическая кластеризация
  + DBSCAN



* Понижение размерности – уменьшение количества признаков с сохранением важной информации. Примеры: визуализация многомерных данных, ускорение работы алгоритмов. Методы:
  + PCA (Principal Component Analysis)
  + t-SNE
  + UMAP
* Поиск аномалий – обнаружение необычных объектов (выбросов). Примеры: обнаружение мошеннических транзакций, диагностика неисправностей оборудования. Методы:
  + Isolation Forest
  + One-Class SVM
* Ассоциативные правила – нахождение зависимостей между событиями или объектами. Примеры: анализ покупок в магазине, рекомендательные системы. Методы:
  + Apriori
  + FP-Growth

1. Основные подходы к обучению: задачи обучения

*Определение и перечисление основных подходов из вопр. 8*

**Обучение нейросетей** – это процесс настройки их параметров (весов и смещений) так, чтобы они могли решать поставленные задачи (классификация, регрессия, генерация и т. д.). Основные задачи обучения нейросетей можно разделить на несколько категорий:

* Классификация – отнесение входных данных к одному из заранее определённых классов. Примеры: распознавание изображений (кошка/собака), спам-фильтрация
* Регрессия – предсказание непрерывного значения. Примеры: прогнозирование цен, температуры
* Кластеризация – группировка данных по схожести без явных меток. Примеры: сегментация клиентов, анализ текстов
* Генерация данных – создание новых данных, похожих на обучающую выборку. Примеры: генерация изображений (GAN), текстов (GPT)
* Рекомендательные системы – предсказание предпочтений пользователя. Примеры: рекомендации фильмов (Netflix), товаров (Amazon)
* Обработка естественного языка (NLP) – анализ и генерация текста. Примеры: перевод, классификация тональности
* Обучение с подкреплением – обучение через взаимодействие со средой и получение наград. Примеры: игры (AlphaGo), управление роботами.

1. Однослойный перцептрон: адаптивная фильтрация

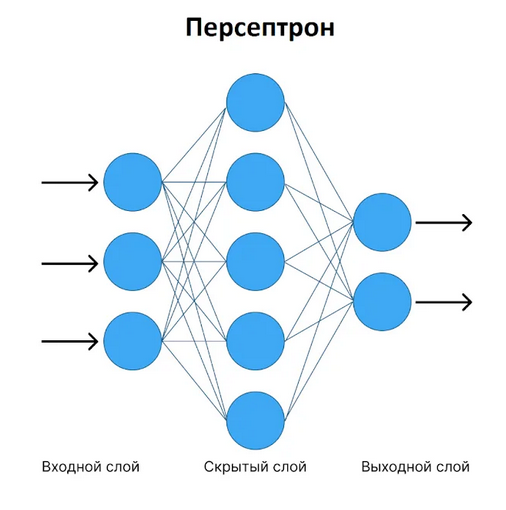
**Однослойный перцептрон** – это простейшая искусственная нейронная сеть, состоящая из одного слоя искусственных нейронов. Он был предложен Фрэнком Розенблаттом в 1957 году и является основой для более сложных архитектур нейронных сетей.

Структура однослойного перцептрона:

* Входной слой – принимает входные данные (признаки). Каждый вход *xi​* умножается на соответствующий вес *wi*. Может включать смещение *b*
* Единственный слой нейронов – вычисляет взвешенную сумму входов и применяет активационную функцию.

Функция активации *f* может быть пороговой (например, ступенчатая функция для бинарной классификации), линейной (для регрессии), сигмоидой или ReLU (в современных вариациях)

* Выходной слой – выдает результат (класс или численное значение)



Адаптивный фильтр – это система с изменяемыми параметрами (весами), которая подстраивается под характеристики входного сигнала. Однослойный перцептрон реализует линейную комбинацию входов:

где *y(n)* – выход фильтра на шаге *n*, *w(n)=[w0, w1,…, wN−1]T* – вектор весов, *x(n)=[x(n), x(n−1), …, x(n−N+1)]T* – вектор входных задержанных сигналов.

Для адаптации весов используется алгоритм наименьших квадратов (LMS):

1. Ошибка фильтрации: *e(n)=d(n)−y(n)*, где *d(n)* – желаемый сигнал (например, очищенный эталонный сигнал)
2. Обновление весов: *w(n+1)=w(n)+μ⋅e(n)⋅x(n)*, где *μ* – шаг обучения (коэффициент, влияющий на скорость и устойчивость сходимости)

Области применения:

* Подавление шумов (например, в аудиосигналах)
* Предсказание сигналов (адаптивное прогнозирование)
* Системы идентификации (оценка параметров канала связи)
* Эхокомпенсация в телекоммуникациях

1. Однослойный перцептрон: методы оптимизации

*Определение и структура перцептрона из вопр. 16*

Оптимизация – это процесс настройки параметров модели (весов и смещений) таким образом, чтобы минимизировать функцию потерь (ошибку), улучшая её способность решать поставленную задачу (классификацию, регрессию и т. д.).

Несмотря на ограниченность, однослойный перцептрон можно оптимизировать разными способами.

Основные методы оптимизации:

1. Градиентный спуск – корректировка весов в направлении, обратном градиенту функции потерь.
2. Метод наименьших квадратов – аналитическое решение для линейной регрессии (можно адаптировать для перцептрона). Формула:
3. Правило Хебба – веса обновляются по правилу: , где *η* – скорость обучения, *y* – целевой выход.
4. Правило перцептрона – обновление весов только при ошибке классификации:
5. Метод опорных векторов (SVM) – можно использовать hinge-loss вместо MSE:
6. Однослойный перцептрон: фильтр на основе метода наименьших квадратов

*Определение и структура перцептрона из вопр. 16*

Однослойный перцептрон, использующий метод наименьших квадратов (МНК) для обучения, по сути, представляет собой линейный классификатор или регрессор, где веса подбираются так, чтобы минимизировать сумму квадратов ошибок между предсказанными и истинными значениями.

МНК применяется для нахождения оптимальных весов *w*, минимизирующих сумму квадратов ошибок между предсказанными и реальными значениями.

Пусть:

*X* – матрица признаков размером *m×(n+1)*, где *m* – число примеров, *n* – число признаков, *+1* – смещение)

*y* – вектор целевых значений (размер *m×1*)

*w* – вектор весов (размер *(n+1)×1*)

Решение МНК:

(если *XTX* не вырождена).

Применение:

* Линейная регрессия: *y=wTx* (линейная активация)
* Классификация: если y≥0, то класс 1, иначе класс 0 (ступенчатая функция)

1. Однослойный перцептрон: алгоритм минимизации среднеквадратической ошибки

*Определение и структура перцептрона из вопр. 16*

Рассмотрим алгоритм обучения однослойного перцептрона с целью минимизации среднеквадратической ошибки (MSE, Mean Squared Error).

Дано:

* Обучающая выборка: X = {x1​, x2​, …, xN​}
* Целевые значения (метки): y = {y1, y2, …, yN}
* Выход перцептрона: yi = f(wTxi + b), где f – функция активации (например, сигмоида или линейная функция)
* В случае линейной регрессии: yi = wTxi + b

Цель: найти веса *w* и смещение *b*, минимизирующие среднеквадратическую ошибку:

Алгоритм минимизации MSE (градиентный спуск):

1. Инициализация весов: *w = 0* или случайные малые значения, *b = 0*
2. Вычисление градиента по весам и смещению: и
3. Обновление весов: ​, , где *η* – скорость обучения
4. Шаги 2-3 повторяются до сходимости (например, пока ошибка не стабилизируется или не достигнуто максимальное число эпох)
5. Однослойный перцептрон: перцептрон Розенблатта

*Определение и структура перцептрона из вопр. 16*

**Перцептрон Розенблатта** – это одна из первых математических моделей искусственного нейрона, предложенная американским психологом и учёным Фрэнком Розенблаттом в 1957 году. Этот алгоритм стал основой для развития искусственных нейронных сетей и машинного обучения.

Структура:

* Состоит из одного слоя искусственных нейронов (в отличие от многослойных сетей)
* Имеет входной слой (принимает сигналы x1, x2, ..., xn​)
* Имеет весовые коэффициенты w1​, w2​, ..., wn​ (настраиваются в процессе обучения)
* Имеет пороговую функцию активации (обычно ступенчатая или сигмоидальная)

Функция активации: выход нейрона вычисляется по формуле:

где *f* – функция активации (например, ступенчатая: *f(z)=1*, если *z ≥ 0*, иначе *0*), *b* – смещение.

Обучение (правило Розенблатта): перцептрон обучается методом коррекции ошибок:

где η – скорость обучения, d – желаемый выход (0 или 1), y – текущий выход перцептрона.

Перцептрон Розенблатта использовался для:

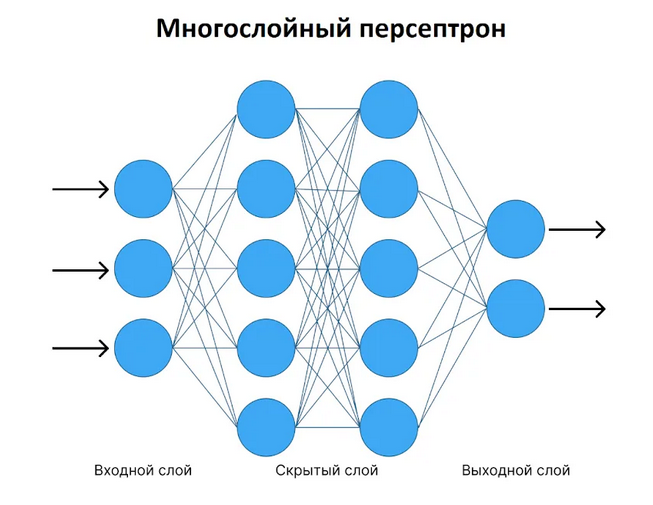
* Классификации изображений (в простых случаях)
* Распознавания паттернов
* Моделирования биологических нейронных процессов

1. Многослойный перцептрон: алгоритм обратного распространения

**Многослойный перцептрон (MLP)** – это одна из искусственная нейронная сеть прямого распространения, состоящая из нескольких слоёв нейронов. В отличие от однослойного перцептрона, многослойный способен решать сложные нелинейные задачи благодаря наличию скрытых слоёв.

Основные компоненты:

1. Входной слой – содержит нейроны, которые принимают входные данные (признаки). Количество нейронов равно размерности входного вектора.
2. Скрытые слои – промежуточные слои между входом и выходом. Каждый нейрон применяет линейное преобразование (веса × входы + смещение) и нелинейную функцию активации (ReLU, sigmoid, tanh и др.). Чем больше скрытых слоёв и нейронов, тем сложнее моделируемые зависимости (но возможен переобучение).
3. Выходной слой – формирует итоговый результат (класс, число и т. д.). Функция активации зависит от задачи. При регрессии линейная или отсутствует. При бинарной классификации – sigmoid. При многоклассовой классификации – softmax.



Функции активации:

1. ReLU (*f(x) = max(0, x)*) — популярна в скрытых слоях
2. Sigmoid (*f(x) = 1 / (1 + e⁻ˣ)*) — для вероятностей
3. Tanh — аналогична sigmoid, но выход в диапазоне *[-1, 1]*
4. Softmax — нормирует выходы в вероятности (для классификации)

**Алгоритм обратного распространения ошибки** – это метод обучения MLP, который минимизирует ошибку сети, корректируя веса связей между нейронами.

1 этап – прямое распространение. На этом этапе входные данные проходят через сеть, и на выходе получается предсказание.

Входной вектор *x = (x1​, x2​, ..., xn​)* подаётся на входной слой. Каждый нейрон скрытого слоя вычисляет взвешенную сумму входов: , где wji – вес связи, bj – смещение. Затем применяется функция активации: *aj​=σ(zj​)*. Аналогично вычисляется выход сети *y=(y1, y2, ..., ym)*.

2 этап – вычисление ошибки

Сравниваем выход сети *y* с истинным значением *t* и вычисляем ошибку, например, через среднеквадратичную ошибку (MSE):

3 этап – обратное распространение ошибки. Цель – вычислить градиенты ошибки по весам и скорректировать их.

Градиент для выходного слоя:

* Ошибка для нейрона выходного слоя – *δk = (yk − tk) ⋅ σ′(zk)*, где *σ′* - производная функции активации
* Корректировка весов выходного слоя – Δwkj = −η ⋅ δk ⋅ aj, где *η* – скорость обучения

Градиент для скрытого слоя:

* Ошибка для нейрона скрытого слоя –
* Корректировка весов скрытого слоя – *Δwji = −η ⋅ δj ⋅ xi*

4 этап – обновление весов

Веса обновляются по правилу: *wnew = wold + Δw*

1. Многослойный перцептрон: обратное распространение ошибки и дифференцирование

*Определение и структура многослойного перцептрона из вопр. 21*

**Обратное распространение ошибки** – это алгоритм обучения, который минимизирует ошибку предсказания сети, корректируя веса с помощью градиентного спуска.

Основные шаги:

1. Прямой проход. Входные данные подаются на сеть. Вычисляются выходы каждого слоя до получения предсказания на выходном слое. Ошибка вычисляется как разница между предсказанием и истинным значением (например, MSE, кросс-энтропия).
2. Обратный проход. Вычисляются градиенты ошибки по весам сети, начиная с выходного слоя и двигаясь назад. Корректируются веса с учётом градиента (используется правило цепочки).

Пусть нейрон в скрытом слое имеет:

* Входы *x1*, *x2*
* Веса *w1*, *w2*
* Смещение *b*
* Функцию активации *σ*

Его выход: *z = σ(w1x1 + w2x2 + b)*

Если ошибка на выходе сети L, то производная по весу w1​:

где: – градиент от следующего слоя, *a = w1​x1​ + w2​x2​ + b* – взвешенная сумма до активации, *σ′(a)* – производная функции активации.

1. Многослойный перцептрон: обучение методом обратного распространения

*Определение и структура многослойного перцептрона из вопр. 21*

**Обучение многослойного перцептрона** – это итеративный процесс настройки весов сети для минимизации функции потерь с использованием алгоритма обратного распространения ошибки.

Обучение состоит из следующих шагов:

1. Инициализация весов – веса *W* и смещения *b* инициализируются случайными малыми значениями (методы Xavier/Glorot, He). Плохая инициализация (например, нулями) приводит к симметрии градиентов и остановке обучения.
2. Прямое распространение – входные данные *X* проходят через сеть: *z(l) = W(l) ⋅ a(l−1 )+ b(l), a(l) = σ(z(l))*, где *σ* – функция активации (ReLU, сигмоида, tanh). На выходном слое вычисляется ошибка L (MSE, кросс-энтропия).
3. Обратное распространение ошибки – градиенты вычисляются от выходного слоя к входному: , *δ(l) = δ(l+1) ⋅ W(l+1)T ⊙ σ′(z(l))*.
4. Обновление весов (градиентный спуск) – веса корректируются в направлении, обратном градиенту: , где *η* – скорость обучения.
5. Многослойный перцептрон: сети свертки

*Определение и структура многослойного перцептрона из вопр. 21*

**Свёрточные нейронные сети (CNN)** – это специальный тип нейронных сетей, разработанный для обработки данных с пространственной структурой, таких как изображения, видео, аудиосигналы и даже тексты. Они особенно эффективны в задачах компьютерного зрения (распознавание объектов, классификация изображений, сегментация), но также применяются в других областях, например, в обработке естественного языка.

CNN можно рассматривать как специализированную разновидность MLP, адаптированную для работы с данными, имеющими пространственную или топологическую структуру (например, изображениями).

Обе модели являются искусственными нейронными сетями и состоят из слоёв нейронов. Обе используют нелинейные функции активации (ReLU, sigmoid, tanh и др.). Обучаются с помощью обратного распространения ошибки и градиентного спуска.

Полносвязные слои, используемые в MLP, часто применяются в CNN на последних этапах для классификации. CNN можно представить как комбинацию специальных слоёв (свёрточных, пулинговых) + классический MLP в конце.

В MLP каждый нейрон связан со всеми входами, что делает его неэффективным для изображений (слишком много параметров). Например, если подать изображение 256x256 пикселей (65 536 входов) в MLP, даже один скрытый слой с 1000 нейронами даст 65 млн параметров – это неэффективно и приводит к переобучению. CNN вводит локальные связи (свёртки) и разделение весов, что уменьшает количество параметров и учитывает пространственные зависимости. Свёртка 3x3 с 32 фильтрами имеет всего 288 параметров (3×3×32 + b), но применяется ко всему изображению.

1. Сети на основе радиальных базисных функций (RBF-сети): сети регуляризации

**RBF-сети** – это тип искусственных нейронных сетей, использующих радиальные базисные функции в качестве активационных функций. Они применяются для задач аппроксимации, классификации и временного прогнозирования.

RBF-сеть состоит из трёх слоёв:

1. Входной слой – передаёт данные в сеть
2. Скрытый слой – содержит нейроны с радиальными базисными функциями (обычно гауссовыми)
3. Выходной слой – линейная комбинация выходов скрытого слоя

Выход сети: , где *x* – входной вектор, *cj​* – центр *j*-й RBF-функции, *ϕj​* – радиальная функция (чаще всего гауссова), *wj​* – веса выходного слоя, *w0*​ – смещение.

Обучение включает 2 этапа:

1. Определение центров и ширины RBF-функций – метод k-средних (кластеризация) и выбор *σj*​ (например, среднее расстояние между центрами)
2. Расчёт весов выходного слоя – решается линейная задача (метод наименьших квадратов, псевдообратная матрица)

**Сети регуляризации** – это подход к построению моделей машинного обучения, в котором добавляется штраф за сложность модели, чтобы избежать переобучения и улучшить обобщающую способность.

RBF-сети тесно связаны с теорией регуляризации Тихонова, где задача обучения формулируется как минимизация:

где *λ* – параметр регуляризации, P – оператор, штрафующий сложность функции (например, оператор Лапласа).

Решение этой задачи может быть представлено в виде разложения по радиальным базисным функциям:

Практическое применение:

* Уменьшение переобучения
* Повышение устойчивости к шуму
* Автоматический подбор гладкости модели

1. Обобщенные сети на основе RBF-сетей

*Определение и структура RBF-сетей из вопр. 25*

Обобщённые сети – это расширения классических RBF-сетей, направленные на улучшение их универсальности, адаптивности и точности при решении задач регрессии, классификации и аппроксимации функций.

1. Гибридные RBF-сети

Объединение RBF-сетей с другими архитектурами нейронных сетей позволяет улучшить их производительность:

* RBF + MLP (многослойный перцептрон) – RBF-слой может использоваться для предварительной обработки данных перед подачей в полносвязную сеть.
* RBF + SVM (метод опорных векторов) – ядро RBF в SVM может быть заменено на адаптивную RBF-сеть для нелинейного разделения данных.

1. Адаптивные RBF-сети

Традиционные RBF-сети используют фиксированные центры и ширины базисных функций. Обобщённые подходы включают:

* Самоорганизующиеся карты (SOM) для выбора центров – вместо случайного выбора или k-means, можно использовать SOM для лучшего распределения центров.
* Обучение с подкреплением – адаптация параметров RBF в процессе обучения.

1. Нечеткие RBF-сети

Интеграция нечёткой логики позволяет улучшить интерпретируемость модели:

* Нечёткие правила на основе RBF – каждый нейрон RBF интерпретируется как нечёткое правило (например, в ANFIS)
* Адаптивные функции принадлежности – параметры RBF настраиваются с помощью нечётких алгоритмов

1. Глубокие RBF-сети

Иерархические структуры на основе RBF позволяют улучшить представление данных:

* Многослойные RBF – несколько RBF-слоёв, где выход одного слоя является входом для следующего
* RBF в свёрточных сетях (CNN) – использование RBF-функций вместо традиционных свёрток

1. Вероятностные RBF-сети

Введение вероятностной интерпретации делает модель более устойчивой:

* Гауссовские процессы – RBF как ковариационная функция в гауссовских процессах
* Байесовские RBF – априорные распределения на параметрах сети

1. Сети на основе радиальных базисных функций (RBF-сети): стратегии обучения

*Определение и структура RBF-сетей из вопр. 25*

**Обучение RBF-сети** обычно включает три ключевых шага:

1. Выбор центров RBF-функций (определение позиций нейронов скрытого слоя)
2. Определение ширины (размаха) RBF-функций (параметр, влияющий на область активации)
3. Оптимизация весов выходного слоя (линейная или нелинейная регрессия)

Выбор центров RBF:

* Случайный выбор из обучающих данных – простой, но не всегда оптимальный метод
* Кластеризация (K-means, GMM) – центры определяются как центроиды кластеров
* Самоорганизующиеся карты Кохонена (SOM) – альтернатива K-means
* Ортогональные методы (OLS) – итеративный выбор наиболее значимых центров

Определение ширины RBF:

* Фиксированная ширина (*σ = const*), например, среднее расстояние между центрами.
* Индивидуальная ширина для каждого центра (на основе расстояний до ближайших соседей).
* Оптимизация через градиентный спуск (если используется обучение с учителем).

Обучение весов выходного слоя:

* Линейный метод (псевдообратная матрица) – быстрое решение методом наименьших квадратов: *W=(ΦTΦ)−1ΦTY*, где Φ – матрица активаций RBF, Y – целевые значения
* Градиентный спуск – используется, если выходной слой нелинейный или при онлайн-обучении
* Регуляризация (Ridge-регрессия) – для борьбы с переобучением: W=(ΦTΦ+λI)−1ΦTY

1. Машина опорных векторов для задачи распознавания образов

**Машина опорных векторов –** это мощный алгоритм машинного обучения, используемый для задач классификации и регрессии. SVM особенно эффективен в случаях малоразмерных данных, данных с высокой размерностью (например, тексты, изображения), нелинейно разделимых классов.

Основные идеи SVM:

1. Максимизация зазора – SVM ищет гиперплоскость (в n-мерном пространстве), которая наилучшим образом разделяет классы, максимизируя расстояние до ближайших точек (опорных векторов)
2. Опорные векторы – это точки, лежащие ближе всего к разделяющей гиперплоскости. Именно они определяют положение гиперплоскости
3. Ядерный трюк – если данные нельзя разделить линейно, SVM использует ядерные функции для отображения в пространство большей размерности, где разделение возможно

**Распознавание образов** – это задача классификации объектов по их признакам. SVM широко применяется в этой области благодаря своей способности эффективно разделять сложные данные, даже в высокоразмерных пространствах.

Если объекты разных классов можно разделить гиперплоскостью, SVM находит оптимальную границу с максимальным зазором. Примеры: распознавание рукописных цифр, классификация объектов по геометрическим признакам.

Если классы нельзя разделить линейно, SVM использует ядерные функции для отображения данных в пространство большей размерности, где разделение возможно. Примеры ядер: Гауссово (для сложных нелинейных границ), полиноминальное (если есть полиномиальные зависимости), сигмоидное (альтернатива нейронным сетям).

1. Машина опорных векторов для задач нелинейной регрессии

*Определение и идеи машины опорных векторов из вопр. 28*

**Support Vector Regression (SVR)** **–** это адаптация SVM для задач регрессии, где вместо разделения классов мы пытаемся найти функцию, которая аппроксимирует данные с минимальной ошибкой, оставаясь в пределах заданной «трубки» (ε-полосы).

Основные идеи SVR:

1. ε-полоса – SVR ищет функцию *f(x)=wTϕ(x)+b*, которая минимизирует ошибку, игнорируя отклонения меньше *ε*. Точки вне полосы штрафуются (аналогично SVM с зазором).
2. Опорные векторы – точки, лежащие на границе или вне ε-полосы, становятся опорными векторами. Они определяют форму регрессионной функции.
3. Ядерный трюк – как и в SVM, SVR использует ядра для работы с нелинейными зависимостями.

Применение SVR в нелинейной регрессии

1. Прогнозирование временных рядов (пример: предсказание цен акций, температуры, спроса)
2. Аппроксимация физических зависимостей (пример: зависимость напряжения от деформации в материалах)
3. Биомедицинские данные (пример: предсказание уровня глюкозы по данным сенсоров)
4. Ассоциативные машины: подходы к построению – методы усреднения по ансамблю

**Ассоциативные машины** **–** это алгоритмы машинного обучения, которые строят предсказания на основе ассоциаций между входными данными и целевой переменной. Они часто используются для задач классификации и регрессии, особенно в случаях, когда данные имеют сложную структуру или высокую размерность.

Основные подходы к построению:

* Усреднение по ансамблю
* Усиление
* Иерархическое смешение мнений экспертов
* И др.

Усреднение по ансамблю – подход, который позволяет улучшить качество предсказаний за счет комбинирования нескольких моделей.

Основные методы:

1. Бэггинг – построение множества моделей на разных подвыборках данных (бутстрап-выборках) и усреднение их предсказаний. Пример: Random Forest – ансамбль решающих деревьев, обученных на разных подвыборках с дополнительным случайным выбором признаков.
2. Бустинг – последовательное обучение моделей, где каждая следующая исправляет ошибки предыдущей. Пример: AdaBoost, Gradient Boosting (XGBoost, LightGBM, CatBoost).
3. Стекинг – комбинирование предсказаний нескольких моделей с помощью «мета-модели» (например, линейной регрессии или нейросети).
4. Блендинг – упрощенная версия стекинга, где предсказания базовых моделей усредняются с весами или объединяются простым способом (например, голосованием).
5. Ассоциативные машины: подходы к построению – методы усиления

*Определение и подходы ассоциативных машин из вопр. 30*

Для повышения эффективности ассоциативных машин применяются различные методы усиления, направленные на ускорение поиска, увеличение объёма хранимых данных, улучшение точности ассоциаций и обеспечение отказоустойчивости. Эти подходы включают как аппаратные оптимизации, так и алгоритмические усовершенствования, а также интеграцию современных технологий, таких как нейросетевые модели и квантовые вычисления.

1. Повышение скорости поиска – использование параллельных вычислений (GPU, FPGA), оптимизация алгоритмов хэширования и индексирования
2. Увеличение емкости памяти – иерархические структуры (комбинация быстрой CAM и медленной, но ёмкой памяти), сжатие данных (использование методов типа Bloom filter для эффективного хранения ассоциаций)
3. Повышение точности ассоциаций – машинное обучение (обучение весовых коэффициентов для улучшения поиска), квантовые ассоциативные памяти (использование квантовых алгоритмов для ускорения поиска)
4. Отказоустойчивость – избыточное хранение (например, в нейросетевых моделях), коррекция ошибок (алгоритмы типа Majority Voting в BAM)
5. Ассоциативные машины: подходы к построению – методы иерархического смешения мнений экспертов

*Определение и подходы ассоциативных машин из вопр. 30*

Ассоциативные машины могут использовать коллективное принятие решений на основе мнений нескольких экспертных систем. Иерархическое смешение (ансамблирование) позволяет комбинировать слабые или специализированные модели в более мощную и устойчивую систему.

Основные подходы к иерархическому смешению мнений:

1. Ансамблирование моделей

* Бэггинг – несколько ассоциативных моделей обучаются на разных подвыборках данных, а их выходы усредняются (например, Random Forest для ассоциативного поиска)
* Бустинг – последовательное обучение моделей с коррекцией ошибок (AdaBoost, Gradient Boosting)
* Стекинг (Stacking) – комбинация предсказаний нескольких моделей через мета-классификатор

1. Многоуровневая иерархия экспертов

* Каскадные системы – решения принимаются поэтапно, где каждый уровень уточняет результат предыдущего
* Гибридные нейросетевые архитектуры (например, Deep Belief Networks + ассоциативная память)

1. Динамическое взвешивание мнений

* Байесовские методы – обновление весов экспертов на основе их точности
* Нейро-нечёткие системы – адаптивное комбинирование решений с учётом неопределённости

1. Структура анализа главных компонентов

**Анализ главных компонентов (PCA)** – это метод снижения размерности данных, который преобразует исходные коррелированные переменные в новые некоррелированные компоненты, упорядоченные по убыванию их дисперсии.

Основные этапы:

1. Стандартизация данных – приведение всех признаков к одному масштабу (обычно z-нормализация): . Необходимо, если признаки измерены в разных единицах.
2. Вычисление ковариационной матрицы – она показывает, как изменяются признаки относительно друг друга: . Если данные стандартизированы, ковариационная матрица совпадает с корреляционной.
3. Нахождение собственных значений и собственных векторов – решается уравнение: *cov(X)⋅v=λ⋅v*. Собственные значения (*λ*) показывают важность компоненты. Собственные векторы (*v*) задают направление новой оси.
4. Сортировка компонент по убыванию дисперсии. Компоненты упорядочиваются по убыванию собственных значений. Первая главная компонента (PC1) объясняет наибольшую дисперсию.
5. Выбор числа компонент:

* Критерий Кайзера: оставляют компоненты с *λ > 1*.
* Доля объяснённой дисперсии (например, 95%).
* График «каменистой осыпи» – визуальный анализ излома.

1. Проецирование данных на новые оси. Формируется матрица проекции W (из k собственных векторов). Новые данные: T = X ⋅ W, где T – матрица счетов, W – матрица нагрузок.

Применение PCA:

* Снижение размерности для визуализации (2D/3D)
* Удаление шума и избыточности данных
* Ускорение работы алгоритмов машинного обучения

1. Анализ главных компонентов: основные представления данных

*Определение анализа главных компонентов из вопр. 33*

**Основные представления данных в PCA** – это способы интерпретации и визуализации данных в контексте метода главных компонент. Они описывают, как многомерные данные преобразуются и структурируются в процессе снижения размерности.

1. Геометрическая интерпретация – данные представляются как точки в многомерном пространстве. PCA ищет такие направления (оси), вдоль которых дисперсия данных максимальна. Первая главная компонента (PC1) соответствует направлению наибольшей дисперсии, вторая (PC2) — ортогональна первой и объясняет следующую по величине дисперсию, и т. д.
2. Статистическая интерпретация – PCA минимизирует ошибку проекции данных на подпространство меньшей размерности. Каждая главная компонента – это линейная комбинация исходных признаков.
3. Алгебраическая интерпретация – PCA основан на сингулярном разложении (SVD) или собственных значениях ковариационной матрицы. Собственные векторы задают направления главных компонент, а собственные значения – их важность (долю объясняемой дисперсии).
4. Оптимизационная интерпретация – PCA можно рассматривать как задачу поиска наилучшего линейного приближения данных меньшей размерности. Альтернативно – как максимизацию дисперсии после проекции.
5. Анализ главных компонентов: сокращение размерности

*Определение анализа главных компонентов из вопр. 33*

**Сокращение размерности** – это процесс уменьшения количества признаков в данных при сохранении максимально полезной информации.

PCA преобразует исходные признаки в новый набор ортогональных (некоррелированных) признаков – главных компонент, которые упорядочены по убыванию их вклада в дисперсию данных. Первая главная компонента (PC1) объясняет наибольшую дисперсию, вторая (PC2) – следующую по величине и так далее.

Для чего нужносокращение размерности:

* Уменьшение вычислительной сложности – меньше признаков – быстрее работа алгоритмов
* Борьба с переобучением – в данных с большим числом признаков модели могут переобучаться
* Визуализация данных – проще отображать данные в 2D или 3D
* Удаление шума и избыточности – PCA помогает выделить наиболее значимые признаки

1. Анализ главных компонентов на основе фильтра Хебба

*Определение анализа главных компонентов из вопр. 33*

Правило Хебба гласит: «Нейроны, которые возбуждаются вместе, связываются вместе».

Математически это можно выразить как:

где: *wi*​ – вес синапса, *η* – скорость обучения, *xi*​ – входной сигнал, *y* – выход нейрона (​).

Если применить это правило к одному нейрону с линейной активацией, его веса будут сходиться к первому главному компоненту данных.

PCA ищет направление с максимальной дисперсией данных. Фильтр Хебба сам находит это направление, потому что он усиливает веса в сторону, где данные больше всего «размазаны». Чем чаще вход *x* даёт большой выход *y*, тем сильнее веса подстраиваются под это направление.

Фильтр Хебба позволяет нейросетевым методам находить главные компоненты данных, что делает PCA не только статистическим, но и нейроинспирированным методом.

1. Карты самоорганизации: конкурентное обучение – процессы конкуренции

**Карты самоорганизации** – это метод искусственных нейронных сетей, используемый для визуализации и кластеризации данных. Они были предложены Тойво Кохоненом в 1980-х годах и относятся к методам неконтролируемого обучения.

**Конкурентное обучение** – это метод обучения нейронных сетей, при котором нейроны соревнуются за право активироваться в ответ на входной сигнал. Победителем становится нейрон, чей вектор весов наиболее близок к входному вектору данных.

Конкуренция в самоорганизующихся картах позволяет выделять ключевые паттерны данных, уменьшая избыточность и формируя эффективные представления.

Процессы конкуренции:

* Выявление победителя – для каждого входного вектора вычисляется расстояние (например, евклидово) между ним и весами всех нейронов. Нейрон с наименьшим расстоянием объявляется победителем.
* Латеральное торможение – победивший нейрон подавляет активность других нейронов, что усиливает конкуренцию. Это позволяет сети формировать разреженные представления данных.

Примеры алгоритмов

* Алгоритм Кохонена (SOM) – использует конкуренцию для выбора BMU, после чего применяет кооперацию и адаптацию.
* Learning Vector Quantization (LVQ) – модификация конкурентного обучения с учителем, где нейроны представляют классы.

1. Карты самоорганизации: конкурентное обучение – процессы кооперации

*Определения карт самоорганизации и конкурентного обучения из вопр. 37*

После этапа конкуренции нейроны не только победитель, но и его соседи адаптируются к входным данным, формируя топологически упорядоченную карту.

Кооперация обеспечивает формирование упорядоченных структур в SOM, сохраняя отношения между данными.

Процессы кооперации:

* Окрестность победителя – адаптируются не только BMU, но и нейроны в его окрестности (например, по гауссову ядру)
* Функция соседства (h) – определяет степень влияния BMU на соседей. Пример: , где *dij*​ – расстояние между нейронами.

Роль кооперации:

* Сохранение топологии – близкие в входном пространстве данные активируют соседние нейроны на карте
* Сглаживание весов – позволяет избежать резких изменений и делает обучение устойчивым

1. Карты самоорганизации: конкурентное обучение – процессы адаптации

*Определения карт самоорганизации и конкурентного обучения из вопр. 37*

**Адаптация** – это процесс обновления весов нейронов-победителей и их соседей для лучшего соответствия входным данным.

Адаптация завершает процесс обучения, обеспечивая точную настройку весов и формирование осмысленных представлений данных.

Процессы адаптации:

* Правило обучения Кохонена: *wi(t+1)=wi(t)+α(t)⋅hij(t)⋅(x−wi(t))*, где: α(t) – скорость обучения (уменьшается со временем), *hij(t)* – функция соседства
* Динамика параметров – скорость обучения *α(t)* и радиус окрестности *σ(t)* уменьшаются со временем для стабилизации карты

Результаты адаптации:

* **Кластеризация** – веса нейронов сходятся к центрам кластеров входных данных
* **Визуализация –** SOM позволяет проецировать многомерные данные на 2D-карту

1. Карты самоорганизации: этапы адаптации – упорядочивание и сходимость

*Определение карт самоорганизации из вопр. 37*

Процесс обучения SOM можно разделить на два основных этапа:

1. Фаза упорядочивания – формирование топологической структуры карты (грубая настройка).

Параметры:

* Начальный размер соседства (*σ*) – большой, охватывает значительную часть карты
* Скорость обучения (*η*) – высокая, чтобы нейроны быстро адаптировались к данным

Процесс:

* Нейроны начинают организовываться в соответствии с распределением входных данных
* Веса нейронов корректируются так, чтобы близкие точки в исходном пространстве оставались близкими на карте

2. Фаза сходимости – точная подстройка весов для минимизации ошибки.

Параметры:

* Размер соседства уменьшается (вплоть до 1 нейрона или даже 0)
* Скорость обучения становится очень низкой

Процесс:

* Карта стабилизируется, нейроны окончательно закрепляются за кластерами данных
* Уточняются позиции нейронов для лучшего соответствия входным данным

Фаза упорядочивания создает общую структуру карты, а фаза сходимости ее уточняет. Правильный баланс между этими этапами критичен для получения качественной карты, которая хорошо отражает топологию исходных данных.

1. Стохастические машины и нейродинамическое программирование: машина Больцмана

**Стохастические машины** – это вычислительные системы, которые используют случайные (стохастические) процессы для принятия решений или оптимизации. Они применяются в задачах, где детерминированные методы оказываются неэффективными, например, генетические алгоритмы, метод Монте-Карло, стохастическая оптимизация.

**Нейродинамическое программирование** – это подход, объединяющий динамическое программирование и нейронные сети для решения сложных задач оптимизации и управления.

Основные идеи:

* Использует нейросети для аппроксимации функций ценности или политик
* Применяется в задачах с большими пространствами состояний, где классическое динамическое программирование неприменимо
* Включает методы обучения с подкреплением (RL), такие как Q-обучение и Deep Q-Networks (DQN)

Машина Больцмана – это тип стохастической рекуррентной нейронной сети, вдохновленный статистической физикой (распределение Больцмана). Она используется для обучения без учителя, моделирования сложных распределений данных и решения задач оптимизации.

Структура и принцип работы

* Состоит из бинарных нейронов (обычно 0/1 или -1/+1)
* Полносвязная архитектура (каждый нейрон связан со всеми остальными)
* Стохастическая активация: нейрон включается с вероятностью, зависящей от взвешенной суммы входов и температуры

Формула активации: , где si​ – состояние нейрона, Ei​ – энергия активации, T – параметр температуры (контролирует уровень стохастичности).

Применение:

* Генеративное моделирование (например, создание изображений)
* Рекомендательные системы (коллаборативная фильтрация)
* Обучение признакам в глубоких сетях
* Оптимизация (аналогично алгоритму имитации отжига)

1. Стохастические машины и нейродинамическое программирование: машина Гельмгольца

*Определения стохастических машин и нейродинамического программирования из вопр. 41*

**Машина Гельмгольца** – это стохастическая генеративная нейронная сеть, которая обучается без учителя, используя принцип вариационного вывода и иерархическое представление данных. Она названа в честь Германа фон Гельмгольца, который изучал восприятие и обучение в биологических системах.

Эта модель сочетает в себе идеи:

* Глубоких генеративных сетей
* Обратного распространения ошибки
* Стохастического обучения (как в машинах Больцмана)

Машина Гельмгольца состоит из 2 взаимодействующих сетей:

1. Модель распознавания – аппроксимирует апостериорное распределение *Q(z∣x)* – определяет, какие скрытые переменные *z* могли породить входные данные x. Аналог энкодера в VAE
2. Генеративная модель – моделирует совместное распределение *P(x,z) = P(x*∣*z)P(z)*. Аналог декодера в VAE

Применение

* Генерация данных (изображения, текст)
* Обучение представлений
* Моделирование восприятия в когнитивной науке

1. Нейродинамическое программирование и обучение с подкреплением

*Определение нейродинамического программирования из вопр. 41*

**Обучение с подкреплением (RL)** – это область машинного обучения, где агент учится принимать решения, максимизируя награду в процессе взаимодействия со средой.

Ключевые компоненты:

* Агент – обучаемая система
* Среда – внешний мир, с которым взаимодействует агент
* Состояние – текущая ситуация
* Действие – выбор агента
* Награда – обратная связь от среды
* Политика – стратегия выбора действий

Методы RL:

* Q-обучение – оценка полезности действий через Q-функцию
* Deep Q-Networks (DQN) – RL + глубокие нейросети
* Policy Gradient – прямое обучение политике
* Actor-Critic – гибрид value-based и policy-based методов

**Нейродинамическое программирование (НДП)** – это подход, объединяющий динамическое программирование (ДП) и аппроксимацию функций (например, нейросетями) для решения сложных задач оптимизации.

Особенности:

* Использует приближенные методы для борьбы с «проклятием размерности»
* Применяется в задачах, где классическое ДП неэффективно
* Часто включает нейросети для аппроксимации функций стоимости или политик

Связь с RL:

* НДП можно рассматривать как теоретическую основу для некоторых методов RL
* Алгоритмы RL (например, DQN, Actor-Critic) используют идеи НДП (аппроксимация функций, Bellman equations)
* НДП чаще применяется в стохастическом управлении, а RL – в более широком контексте обучения агентов

1. Нелинейные динамические системы: многослойные нейросетевые структуры прямого распространения

**Нелинейные динамические системы (НДС)** – это системы, поведение которых описывается нелинейными дифференциальными или разностными уравнениями.

Их ключевые особенности:

* Нелинейность – выход системы не пропорционален входу (например, из-за наличия квадратичных, экспоненциальных или других сложных зависимостей)
* Динамика – состояние системы изменяется во времени, часто с памятью о предыдущих состояниях
* Сложное поведение – возможны хаос, бифуркации, аттракторы

Примеры:

* Модель Лоренца (описание конвекции в атмосфере)
* Биологические нейронные сети
* Экономические и социальные процессы

**Многослойные нейронные сети прямого распространения (MLP)** – классическая архитектура искусственных нейронных сетей, состоящая из:

* Входного слоя (передача данных)
* Скрытых слоёв (нелинейные преобразования)
* Выходного слоя (финальное предсказание)

Связь с НДС:

Каждый слой MLP можно рассматривать как дискретную динамическую систему, где состояние на слое l вычисляется как *hl = σ(Wlhl – 1 + bl)*, где σ – нелинейная функция активации (например, ReLU, sigmoid), *Wl​* — матрица весов, *bl*​ – смещение.

Нелинейность позволяет моделировать сложные зависимости (например, XOR, которую нельзя решить линейным классификатором).

Применение MLP:

* Прогнозирование временных рядов
* Классификация изображений
* Обработка естественного языка
* Кредитный скоринг
* Управление роботами
* Медицинская диагностика
* Генерация данных

1. Нелинейные динамические системы: рекуррентные сети

*Определение нелинейных динамических систем из вопр. 44*

**Рекуррентные нейронные сети (RNN)** – это класс искусственных нейронных сетей, обладающих памятью за счёт цикличных связей.

RNN можно представить как дискретную НДС, где:

* Состояние системы – скрытый вектор *ht​* на шаге *t*
* Динамика задаётся рекуррентным уравнением: *ht=σ(Whht−1+Wxxt+b)*, где *Wh* – матрица рекуррентных весов, *Wx* – матрица входных весов, *σ* – нелинейная функция активации (например, tanh или ReLU).

Свойства RNN как НДС:

* Нелинейность – активации σ вносят нелинейные искажения
* Зависимость от предыдущих состояний – ht​ зависит от ht−1​, что аналогично авторегрессии в НДС
* Возможность хаоса и сложных аттракторов – при определённых параметрах RNN могут демонстрировать хаотическое поведение

Применение:

* Прогнозирование временных рядов (финансы, погода)
* Обработка естественного языка (NLP: языковые модели, машинный перевод)
* Управление роботами (моделирование динамики систем)
* Нейродинамическое программирование (аналог дифференциальных уравнений)