Программа
«*Методы искусственного интеллекта в CAD*»

Входной ассемент

Лекция 2.

Обязательный блок

2.1. Методы обработки больших данных

2.2. Методы машинного обучения

Что такое большие данные (Big Data)?

Большие данные — массивы информации, которые:

- превышают возможности традиционных СУБД;
- характеризуются **3V**: Volume (объём), Velocity (скорость поступления), Variety (разнообразие форматов).

Примеры источников:

- логи серверов и приложений;
- данные соцсетей и мессенджеров;
- IoT-устройства (датчики, камеры);
- транзакции e-commerce.

Технология Apache Hadoop MapReduce

Hadoop MapReduce — фреймворк для распределённых вычислений на кластерах.

Архитектура Hadoop:

- HDFS (Hadoop Distributed File System)
- MapReduce - модель распределенных вычислений
- YARN - управление ресурсами

Принцип работы:

1. **Map** — разбиение задачи на подзадачи и обработка на узлах кластера.
2. **Shuffle & Sort** — перераспределение данных по ключам.
3. **Reduce** — агрегация результатов.

Технология Apache Hadoop MapReduce

Принцип работы MapReduce:

Map Phase: (k1, v1) → list(k2, v2)

Shuffle: группировка по ключам

Reduce Phase: (k2, list(v2)) → list(v3)

Плюсы:

- масштабируемость (тысячи узлов);
- отказоустойчивость (репликация данных);
- обработка терабайт/петабайт.

Минус: высокая задержка (не для real-time).

Распределённые хранилища: концепция

Распределённое хранилище — система, где данные хранятся на множестве узлов с координацией через сеть.

Ключевые особенности:

- горизонтальное масштабирование (добавление узлов);
- репликация данных (отказоустойчивость);
- партиционирование (шардирование);
- согласованность vs доступность (теорема CAP).

Альтернатива традиционным реляционным СУБД для Big Data.

Распределенные хранилища и NoSQL

Ограничения реляционных СУБД:

- Сложность горизонтального масштабирования
- Жесткая схема данных
- Ограничения производительности при больших объемах

Преимущества NoSQL:

- Горизонтальная масштабируемость
- Гибкая схема данных
- Высокая доступность
- Поддержка распределенных архитектур

NoSQL-хранилища: классификация

NoSQL — нереляционные базы данных для гибкого хранения и высокой производительности.

Типы:

- **Ключ-значение** (Redis) — быстрые операции по ключу.
- **Документные** (MongoDB, CouchDB) — хранение JSON-подобных документов.
- **Колоночные** (Apache Cassandra) — оптимизация для аналитики.
- **Графовые** (Neo4j) — связи между сущностями.
- **Временные ряды** (InfluxDB) — данные с таймштампами.

Выбор зависит от сценария использования.

Примеры NoSQL-решений

1. **MongoDB** — документная СУБД, гибкий schema, индексация.
2. **Neo4j** — графовая СУБД, запросы на Cypher, анализ связей.
3. **Redis** — кэш и брокер сообщений, операции за микросекунды.
4. **InfluxDB** — временные ряды, агрегация по интервалам.
5. **Apache Cassandra** — колоночная, высокая доступность, линейное масштабирование.
6. **CouchDB** — документная, репликация между узлами.
7. **Elasticsearch** — поисковая СУБД, аналитика логов, полнотекстовый поиск.

Примечание: PostgreSQL иногда используют как «гибридную» СУБД с JSON-поддержкой.

Визуализация больших данных: Grafana

Grafana — платформа для мониторинга и визуализации данных.

Возможности:

- дашборды с графиками, таблицами, индикаторами;
- поддержка источников: Prometheus, InfluxDB, Elasticsearch;
- алерты (оповещения) по метрикам;
- совместная работа и экспорт.

Сценарии:

- мониторинг нагрузки кластера;
- анализ трендов продаж;
- визуализация логов приложений.

Плюс: открытый код и плагины.

Облачные вычисления: Apache Spark

Apache Spark — фреймворк для быстрой обработки данных в памяти.

Особенности:
- скорость (в 100× быстрее Hadoop MapReduce);
- поддержка SQL, Streaming, MLlib, GraphX;
- работа с RDD (Resilient Distributed Datasets);
- интеграция с Hadoop, Kafka, S3.

Сценарии:
- ETL-процессы;
- машинное обучение на больших выборках;
- анализ потоков данных (streaming).

Экосистема: Spark SQL, Spark Streaming, MLlib.

Практическое применение и тенденции

Типичный стек технологий:

Сбор данных → Kafka/Flink
Хранение → Hadoop/HDFS + NoSQL
Обработка → Spark/Flink
Анализ → MLlib/TensorFlow
Визуализация → Grafana/Kibana

Современные тенденции:

- Serverless вычисления
- Real-time обработка данных
- AutoML и MLOps
- Гибридные облачные решения
- Edge computing

Ключевые навыки специалиста:

- Понимание распределенных систем
- Знание SQL и NoSQL технологий
- Опыт работы с облачными платформами
- Навыки программирования (Python, Scala, Java)

Сравнение ключевых технологий

Технология	Назначение	Плюсы	Минусы
Hadoop MapReduce	Распределённые вычисления	Масштабируемость, отказоустойчивость	Высокая задержка
Spark	Быстрая обработка в памяти	Скорость, ML-инструменты	Требует RAM
MongoDB	Документное хранилище	Гибкость, индексация	Ограниченная транзакционность
Cassandra	Колонное хранилище	Высокая доступность	Сложность настройки
Grafana	Визуализация	Дашборды, алерты	Требует источников данных

Основы анализа данных на Python

Ключевые библиотеки:

- Pandas - обработка табличных данных
- NumPy - математические вычисления
- Matplotlib/Seaborn - визуализация
- Scikit-learn - машинное обучение

Основные операции:

- Очистка и предобработка данных
- Статистический анализ
- Визуализация распределений
- Корреляционный анализ

Типичный workflow анализа:

```
import pandas as pd
import matplotlib.pyplot as plt

# Загрузка и предобработка
data = pd.read_csv('dataset.csv')
data = data.dropna()

# Анализ и визуализация
data.describe()
data['column'].hist()
plt.show()
```

Факторный, дискриминантный и кластерный анализ

Факторный анализ:

- Сокращение размерности данных
- Выявление скрытых переменных (факторов)
- Применение: психометрия, социология

Дискриминантный анализ:

- Классификация объектов по группам
- Построение дискриминантных функций
- Применение: кредитный скоринг, биометрия

Кластерный анализ:

- Группировка объектов по схожести
- Методы: K-means, иерархическая кластеризация
- Применение: сегментация клиентов, анализ рынка

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3)
clusters = kmeans.fit_predict(data)
```

Архитектура системы обработки больших данных

Типичная структура (слои):

1. **Сбор данных**
   - Источники: IoT, лог-файлы, API, базы данных.
   - Инструменты: Apache Kafka, Flume.

2. **Хранение**
   - Распределённые файловые системы: HDFS.
   - NoSQL-СУБД: Cassandra, MongoDB.
   - Объектное хранилище: Amazon S3.

3. **Обработка**
   - Batch: Apache Hadoop, Spark.
   - Streaming: Spark Streaming, Flink.

4. **Анализ и визуализация**
   - SQL-движки: Hive, Presto.
   - Дашборды: Grafana, Tableau.

5. **Управление и безопасность**
   - Метаданные: Apache Atlas.
   - Контроль доступа: Kerberos, Ranger.

Ключевой принцип:

масштабируемость и отказоустойчивость.

Архитектура системы обработки больших данных

Компоненты Lambda-архитектуры:

Batch Layer (Пакетная обработка)

↓

Speed Layer (Стриминг)

↓

Serving Layer (Обслуживание)

Современные подходы:

- **Kappa-архитектура** - только стриминг
- **Data Lake** - централизованное хранение
- **Data Mesh** - децентрализованная архитектура

Технологический стек:

- Сбор данных: Kafka, Flume
- Обработка: Spark, Flink
- Хранение: Hadoop HDFS, S3
- Оркестрация: Airflow, Kubernetes

Data Mining и Machine Learning

Data Mining

- Цель: обнаружение закономерностей в больших массивах данных.
- Методы: ассоциация, кластеризация, последовательные шаблоны.
- Инструменты: Weka, RapidMiner.

Machine Learning

- Цель: построение предсказательных моделей.
- Типы:
  - Обучение с учителем (классификация, регрессия).
  - Обучение без учителя (кластеризация, снижение размерности).
  - Обучение с подкреплением.
- Алгоритмы: деревья решений, SVM, нейронные сети.

Связь:

Data Mining часто использует ML-методы для анализа.

Примеры задач:

- прогнозирование оттока клиентов;
- детекция мошенничества;
- рекомендательные системы.

Data Mining и Machine Learning

Data Mining процессы:

- CRISP-DM: Business Understanding → Data Understanding → Data Preparation → Modeling → Evaluation → Deployment
- KDD: Selection → Preprocessing → Transformation → Data Mining → Interpretation

Машинное обучение:

```
# Обучение модели
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(X_train, y_train)

# Предсказание
predictions = model.predict(X_test)
```

Основные категории:

- **Supervised Learning**

(с учителем)

- **Unsupervised Learning**

(без учителя)

- **Reinforcement Learning**

(с подкреплением)

Механизм фрагментарного хранения данных

Фрагментарное хранение (шардирование) — разбиение данных на части (шарды) и распределение по узлам.

Типы шардирования:

- **Горизонтальное** — строки таблицы распределяются по серверам.
- **Вертикальное** — столбцы таблицы разделяются по сервисам.
- **Функциональное** — данные группируются по функционалу приложения.

Ключевые механизмы:

- **Ключ шардирования** (shard key) — поле для распределения записей.
- **Маршрутизация** — определение узла по ключу.
- **Репликация** — копирование шардов для отказоустойчивости.
- **Перешардирование** — перераспределение при росте данных.

Плюсы:

- масштабируемость;
- параллельная обработка;
- локализация отказов.

Минусы:

- сложность запросов между шардами;
- балансировка нагрузки.

Преимущества фрагментации:

- Повышение производительности запросов
- Распределение нагрузки
- Локализация данных
- Упрощение управления

Технологии реализации:

- **Sharding** в MongoDB, Cassandra
- **Partitioning** в PostgreSQL, MySQL
- **Region Servers** в Hbase

```
-- Пример создания партиций в PostgreSQL
CREATE TABLE sales (
   id SERIAL,
   sale_date DATE,
   amount DECIMAL
) PARTITION BY RANGE (sale_date);
```

Введение в машинное обучение

Машинное обучение (ML) — набор методов, позволяющих компьютерам находить закономерности в данных и делать предсказания без явного программирования.

Основные типы задач:

- **Регрессия** — предсказание числового значения (цена дома, спрос).
- **Классификация** — отнесение к классу (спам/не спам, диагноз).
- **Кластеризация** — группировка похожих объектов без меток.
- **Снижение размерности** — сжатие данных с сохранением структуры.

Ключевые этапы:

- Сбор и предобработка данных.
- Выбор модели.
- Обучение на тренировочной выборке.
- Оценка на тестовой выборке.
- Внедрение и мониторинг.

Инструменты: Python, scikit-learn, TensorFlow/PyTorch.

```
# Пример разделения данных
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

Базовые алгоритмы ML

1. **Линейная регрессия**
   - Модель: $y=w_0+w_1x_1+\ldots+w_nx_n$.
   - Цель: минимизировать MSE (среднеквадратичную ошибку).
   - Применение: прогнозирование непрерывных величин.

2. **Логистическая регрессия**
   - Для классификации (2+ класса).
   - Функция активации: сигмоида $\sigma(z)=1/(1+e^{-z})$.
   - Выход: вероятность принадлежности к классу.

3. **Регуляризация** (L1/L2)
   - Предотвращает переобучение.
   - Добавляет штраф к функции потерь: $\lambda\sum w_i^2$ (L2) или $\lambda\sum|w_i|$ (L1).

4. **Деревья решений**
   - Иерархическая структура «если-то».
   - Интерпретируемы, работают с нелинейностями.

5. **Метод k-ближайших соседей (k-NN)**
   - Классификация по соседним точкам.
   - Параметр: число соседей k.

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(penalty='l2', C=1.0)
```

Деревья решений, k-NN и кластеризация

Деревья решений:

- Иерархическое разбиение данных
- Критерии: энтропия, Gini impurity
- Преимущества: интерпретируемость

Метод k ближайших соседей (k-NN):

- Классификация по majority vote
- Регрессия по среднему значению
- Важность метрики расстояния

Кластеризация (K-means):

- Группировка похожих объектов
- Выбор числа кластеров (elbow method)
- Методы оценки: silhouette score

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3)
clusters = kmeans.fit_predict(X)
```

Кластеризация — объединение объектов в группы по схожести.

- **k-means**:
  - задаём число кластеров *k*;
  - итеративно перемещаем центры кластеров.
- **Иерархическая кластеризация**:
  - строит дерево объединений (дендрограмму).
- **DBSCAN**:
  - находит кластеры произвольной формы, устойчив к шуму.

Снижение размерности — уменьшение числа признаков.

- **PCA (метод главных компонент)**:
  - находит оси максимальной дисперсии;
  - сохраняет большую часть информации в меньшем числе измерений.
- **t-SNE**:
  - визуализирует высокоразмерные данные в 2D/3D.
- **Autoencoders** (нейросети):
  - кодируют данные в компактное представление.

Нейронные сети: основы

Нейрон — базовый элемент:

$$y=f(w_1x_1+\ldots+w_nx_n+b),$$

где f — функция активации (ReLU, сигмоида, tanh).

Архитектура:

- **Входной слой** — принимает данные.
- **Скрытые слои** — извлекают признаки.
- **Выходной слой** — выдаёт предсказание.

Метод обратного распространения ошибки (Backpropagation)

1. Прямой проход: вычисление выхода сети.
2. Обратный проход:
   - вычисление градиентов ошибки по весам;
   - обновление весов через градиентный спуск.
3. Повторение для множества примеров.

Оптимизаторы: SGD, Adam, RMSprop.

Векторизация данных:

- Преобразование в числовой формат
- One-hot encoding для категорий
- Нормализация и стандартизация

```
import tensorflow as tf
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

Предобработка данных и переобучение

Предобработка данных:

- Обработка пропусков (mean, median, mode)
- Кодирование категориальных признаков
- Масштабирование: MinMax, StandardScaler
- Балансировка классов

Переобучение (overfitting):

- Модель запоминает данные, а не учится
- Признаки: низкая ошибка на train, высокая на test
- Методы борьбы: регуляризация, dropout, early stopping

Кривые обучения:

- Анализ сходимости модели
- Выявление underfitting/overfitting
- Определение достаточного объема данных

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Метрики оценки моделей

Для регрессии:

- MSE (Mean Squared Error)
- RMSE (Root Mean Squared Error)
- MAE (Mean Absolute Error)
- R^2 (коэффициент детерминации)

Для классификации:

- Accuracy (точность)
- Precision и Recall
- F1-score (гармоническое среднее)
- ROC-AUC (площадь под кривой)

Матрица ошибок:

	Predicted 0	Predicted 1
Actual 0	TN (True Negative)	FP (False Positive)
Actual 1	FN (False Negative)	TP (True Positive)

```
from sklearn.metrics import classification_report
print(classification_report(y_true, y_pred))
```

Проектирование архитектуры нейросетей

Ключевые решения:

- Число слоёв и нейронов.
- Функции активации (ReLU популярна для скрытых слоёв).
- Регуляризация (Dropout, BatchNorm).
- Размер батча и скорость обучения (learning rate).

Типичные архитектуры:

- **Полносвязные сети** (MLP) — для табличных данных.
- **Свёрточные сети (CNN)** — для изображений.
- **Рекуррентные сети (RNN/LSTM)** — для последовательностей (текст, время).
- **Трансформеры** — для текста и не только.

Векторизация данных:

- Текст → embedding (Word2Vec, BERT).
- Изображения → пиксели/признаки.
- Категориальные признаки → one-hot или embedding.

Отладка моделей: предобработка и оценка

Предобработка данных:

- Очистка: удаление пропусков, выбросов.
- Нормализация/стандартизация: $(x-\mu)/\sigma$.
- Кодирование категориальных признаков.
- Разбиение на train/val/test.

Обучающие кривые:

- Строим ошибку на train и val-выборках по эпохам.
- **Переобучение:** train-ошибка ↓, val-ошибка ↑.
- **Недообучение:** обе ошибки высокие.

Метрики оценки:

- **Регрессия:** MSE, MAE, $R2$.
- **Классификация:**
  - Accuracy, Precision, Recall;
  - F1-score, ROC-AUC.
- **Кластеризация:** Silhouette score, inertia.

Способы борьбы с переобучением:

- Увеличение данных (augmentation).
- Регуляризация (L1/L2, Dropout).
- Ранняя остановка (early stopping).
- Уменьшение сложности модели.

Best Practices и заключение

Процесс разработки ML-модели:

- Понимание бизнес-задачи
- Сбор и разведка данных
- Предобработка и feature engineering
- Выбор и обучение модели
- Валидация и тонкая настройка
- Деплой и мониторинг

Ключевые принципы:

- Начинайте с простых моделей
- Всегда используйте кросс-валидацию
- Анализируйте ошибки модели
- Документируйте эксперименты

Дальнейшее развитие:

- Глубокое обучение (CNN, RNN)
- Обработка естественного языка
- Компьютерное зрение
- Reinforcement Learning