МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания

по дисциплине
«Методы искусственного интеллекта»
для студентов
Специальности 10.05.03 Информационная безопасность
автоматизированных систем
Специализация «Анализ безопасности информационных систем»

Ставрополь
2026

Введение

Дисциплина «Методы искусственного интеллекта» направлена на формирование у студентов теоретических знаний и практических навыков, необходимых для понимания принципов функционирования и разработки интеллектуальных систем. В условиях стремительного развития технологий и цифровой трансформации общества искусственный интеллект (ИИ) становится ключевым элементом в самых разных сферах деятельности — от здравоохранения и образования до промышленности и безопасности.

В рамках изучения дисциплины особое внимание уделяется базовым подходам и алгоритмам, лежащим в основе современных интеллектуальных систем. Это включает машинное обучение, нейронные сети, методы кластеризации, обработку естественного языка, генеративные модели и другие направления. Рассматриваются как теоретические аспекты, так и практические вопросы, связанные с проектированием и реализацией ИИ-моделей с использованием популярных инструментов и фреймворков, таких как TensorFlow, PyTorch и Scikit-learn.

Курс ориентирован на развитие аналитического мышления, умения формализовывать задачи, работать с данными и выбирать адекватные методы их обработки и интерпретации. Особое место занимает практика: каждая тема подкрепляется лабораторными работами, в ходе которых студенты получают опыт построения и анализа интеллектуальных систем на реальных и синтетических данных.

В результате освоения дисциплины обучающиеся смогут не только применять полученные знания при решении прикладных задач, но и разбираться в современной научной литературе и исследованиях в области искусственного интеллекта. Это создаёт основу для их дальнейшего профессионального роста в IT-сфере, наукоёмких отраслях и в проектах, где требуется автоматизация принятия решений и работа с большими объёмами данных.

Лабораторная работа 1.

Введение в интеллектуальные системы: основные понятия и области применения

Цель: изучить основные понятия, связанные с интеллектуальными системами, и рассмотреть области их применения.

Теоретическая часть

Рассмотреть понятие «интеллектуальная система», определить, какими характеристиками она обладает и чем отличается от традиционных

вычислительных систем. Изучить ключевые элементы интеллектуальных систем, включая модули восприятия, анализа, принятия решений и обучения. Особое внимание уделить понятию искусственного интеллекта как ядра интеллектуальных систем. Рассмотреть основные подходы к построению интеллектуальных систем, включая экспертные системы, системы машинного обучения, нейронные сети и гибридные системы. Ознакомиться с примерами применения интеллектуальных систем в различных сферах: здравоохранение, промышленность, транспорт, финансы, образование и информационная безопасность.

Вопросы и задания

Вопросы, выносимые на обсуждение:

1. Что такое интеллектуальная система?
2. Какие отличительные особенности интеллектуальных систем по сравнению с традиционными вычислительными средствами?
3. Какие ключевые компоненты входят в состав интеллектуальной системы?
4. В чем заключается роль модуля принятия решений в интеллектуальных системах?
5. Какие области наиболее активно используют интеллектуальные системы?
6. Что такое экспертная система?
7. Как связаны интеллектуальные системы и искусственный интеллект?
8. Как классифицируются интеллектуальные системы по способу обучения?
9. Какие примеры применения интеллектуальных систем в медицине и промышленности вы можете привести?
10. Каковы перспективы развития интеллектуальных систем?

Практические задания

Задание 1. Составьте таблицу, в которой сравните традиционные вычислительные системы и интеллектуальные системы по следующим критериям: способ обработки информации, способность к адаптации, принятие решений, наличие модуля обучения.

Задание 2. Найдите и кратко опишите три примера использования интеллектуальных систем в различных отраслях (например, интеллектуальные помощники, системы диагностики, системы предиктивной аналитики).

Задание 3. Постройте блок-схему, отражающую структуру интеллектуальной системы, включающую модули ввода данных, обработки, принятия решений и обучения.

Задание 4. Напишите реферат (1–2 страницы) на тему «Интеллектуальные системы в современном мире: возможности и вызовы».

Задание 5. В виде таблицы приведите классификацию интеллектуальных систем по способу функционирования: экспертные системы, обучающиеся системы, адаптивные системы и гибридные модели. Укажите для каждой категории определение и пример применения.

Пример таблицы для Задания 1:

Критерий	**Традиционная система**	**Интеллектуальная система**
Способ обработки информации	Жестко запрограммированный	Гибкий, с элементами обучения
Адаптация	Отсутствует	Возможна при наличии обучения
Принятие решений	По заданным алгоритмам	С учетом знаний и анализа
Наличие модуля обучения	Нет	Да

Лабораторная работа 2

Основы машинного обучения: классификация алгоритмов и их особенности

Цель: изучить базовые принципы машинного обучения, классификацию алгоритмов и определить особенности различных подходов к обучению интеллектуальных систем.

Теоретическая часть

Машинное обучение (Machine Learning, ML) представляет собой один из ключевых разделов искусственного интеллекта, ориентированный на разработку алгоритмов, способных автоматически обучаться на основе данных без явного программирования. В рамках лабораторной работы необходимо ознакомиться с основными типами машинного обучения: обучением с учителем (supervised learning), без учителя (unsupervised learning) и обучением с подкреплением (reinforcement learning).

Обучение с учителем основано на предоставлении алгоритму размеченных данных, где каждая обучающая пара содержит вход и желаемый результат. Наиболее известные алгоритмы этого типа: линейная и логистическая регрессия, метод опорных векторов (SVM), решающие деревья, случайный лес, k-ближайших соседей и нейронные сети.
Обучение без учителя используется для анализа неразмеченных данных и включает методы кластеризации (например, k-средних, иерархическая

кластеризация), а также алгоритмы снижения размерности (PCA, t-SNE). Обучение с подкреплением реализуется через взаимодействие агента с окружающей средой, при этом агент получает вознаграждение или штраф за свои действия, что позволяет ему формировать оптимальную стратегию поведения.

Следует также рассмотреть различия между типами задач машинного обучения (регрессия, классификация, кластеризация, ассоциация), критерии выбора алгоритма, особенности обучения моделей и методы оценки их эффективности (точность, полнота, F-мера, ROC-кривая, перекрёстная проверка).

Вопросы и задания

Вопросы, выносимые на обсуждение:

1. Что такое машинное обучение и какова его роль в интеллектуальных системах?
2. Как классифицируются алгоритмы машинного обучения?
3. В чем заключается отличие между обучением с учителем и без учителя?
4. Приведите примеры алгоритмов для каждого типа обучения.
5. Что такое обучение с подкреплением и в каких задачах оно применяется?
6. Чем классификация отличается от регрессии?
7. Какие задачи решаются с помощью кластеризации?
8. Что такое переобучение (overfitting) и как его можно избежать?
9. Как оценивается качество работы модели машинного обучения?
10. Каковы критерии выбора алгоритма в зависимости от задачи и данных?

Практические задания

Задание 1. Составьте таблицу, отражающую классификацию алгоритмов машинного обучения с указанием типа (обучение с учителем, без учителя, с подкреплением), названия алгоритма, краткого описания, примера применения и типа решаемой задачи.

Задание 2. На основе реальных или гипотетических данных предложите пример задачи классификации и определите, какой алгоритм лучше всего применить, обоснуйте выбор.

Задание 3. Нарисуйте схему, иллюстрирующую три основных типа машинного обучения, с кратким описанием каждого из них и примерами алгоритмов.

Задание 4. Напишите краткий доклад (2–3 страницы) на тему: «Обзор алгоритмов машинного обучения и области их применения».

Задание 5. В виде таблицы представьте основные метрики оценки моделей классификации, включая формулы и пояснение: accuracy, precision, recall, F1-score.

Пример таблицы для Задания 1:

Тип обучения	**Алгоритм**	**Описание**	**Пример применения**	**Тип задачи**
С учителем	Метод опорных векторов	Строит разделяющую гиперплоскость	Распознавание лиц	Классификация
С учителем	Линейная регрессия	Находит линейную зависимость между переменными	Прогноз цен на жилье	Регрессия
Без учителя	K-средних	Разбивает данные на группы по близости	Сегментация клиентов	Кластеризация
С подкреплением	Q-Learning	Агенты учатся через вознаграждение	Управление роботами, игры	Обучение с подкреплением

Пример таблицы для Задания 5:

Метрика	**Формула**	**Назначение**
Accuracy	(TP + TN) / (TP + FP + FN + TN)	Общая доля правильных предсказаний
Precision	TP / (TP + FP)	Доля истинных положительных среди всех положительно предсказанных
Recall	TP / (TP + FN)	Способность модели находить все истинные положительные
F1-Score	2 * (Precision * Recall) / (Precision + Recall)	Баланс между точностью и полнотой

Лабораторная работа 3

Линейная и логистическая регрессия: теория и применение

Цель: изучить теоретические основы линейной и логистической регрессии, а также их практическое применение для решения задач регрессии и классификации.

Теоретическая часть

Линейная и логистическая регрессия являются фундаментальными моделями в машинном обучении и широко применяются при решении задач, связанных с анализом и предсказанием на основе данных.

Линейная регрессия используется для предсказания непрерывных значений. Её цель — найти наилучшую линейную зависимость между независимыми переменными (фичами) и зависимой переменной (таргетом). Основная формула линейной регрессии имеет вид:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n + \varepsilon$$

где y — предсказываемое значение, $x_1...x_n$ — входные признаки, $\beta_0...\beta_n$ — коэффициенты модели, ε — ошибка.

Оценка параметров производится методом наименьших квадратов, минимизирующим сумму квадратов отклонений предсказанных значений от фактических.

Логистическая регрессия применяется для решения задач классификации, в основном бинарной. Она оценивает вероятность принадлежности объекта к определенному классу. Ключевое отличие — использование логистической (сигмоидной) функции активации, которая ограничивает выход модели в диапазоне от 0 до 1:

$$P(y=1|x) = 1 / (1 + e^{-(\beta_0 + \beta_1 x_1 + ... + \beta_n x_n)})$$

Пороговое значение (обычно 0.5) используется для принятия окончательного решения о принадлежности к классу. В качестве функции ошибки применяется логарифмическая функция потерь (log-loss).

Обе модели хорошо интерпретируемы, устойчивы к переобучению при малом числе признаков, но имеют ограничения при наличии сложных или нелинейных зависимостей.

Вопросы и задания

Вопросы, выносимые на обсуждение:

1. Что такое линейная регрессия и в каких задачах она применяется?
2. Какая математическая модель лежит в основе линейной регрессии?
3. Как оцениваются параметры модели линейной регрессии?
4. Что такое логистическая регрессия и в чем её отличие от линейной?
5. Почему в логистической регрессии используется сигмоидная функция?
6. Что представляет собой логарифмическая функция потерь?
7. В каких задачах предпочтительнее использовать логистическую регрессию?

8. Какие метрики используются для оценки качества моделей регрессии и классификации?
9. Как можно визуализировать модель линейной или логистической регрессии?
10. Какие ограничения и допущения характерны для этих моделей?

Практические задания

Задание 1. На основе предложенного датасета (например, зависимости стоимости жилья от площади) постройте модель линейной регрессии и визуализируйте полученную прямую на графике.

Задание 2. Проведите анализ модели: оцените значения коэффициентов, интерпретируйте их, рассчитайте среднеквадратичную ошибку (MSE) и коэффициент детерминации R^2.

Задание 3. На примере датасета с бинарной классификацией (например, предсказание болезни по возрасту и весу) постройте модель логистической регрессии. Выведите вероятности и метки классов.

Задание 4. Постройте ROC-кривую для логистической модели и рассчитайте площадь под кривой (AUC). Интерпретируйте результат.

Задание 5. Сравните на практике линейную и логистическую регрессию: можно ли применять линейную модель для классификации? Какие ошибки могут возникнуть?

Пример таблицы для Задания 2:

Коэффициент	**Значение**	**Интерпретация**
β_0 (свободный член)	2.3	Базовое значение при нулевых входных данных
β_1 (по площади)	0.85	При увеличении площади на 1 кв.м цена растет на 0.85 тыс. руб.

Пример визуализации для Задания 1:
График, где на оси X — площадь, на оси Y — цена, и линия регрессии проходит сквозь облако точек.

Пример интерпретации логистической регрессии:
Если вероятность выхода модели для объекта равна 0.8, то при пороге 0.5 мы относим его к положительному классу. При этом модель уверена в своём решении на 80%.

Лабораторная работа 4

Деревья решений и ансамбли моделей (Random Forest, Gradient Boosting)

Цель: изучить принципы построения деревьев решений и ансамблевых методов на их основе, таких как случайный лес (Random Forest) и градиентный бустинг (Gradient Boosting), а также освоить их применение для задач классификации и регрессии.

Теоретическая часть
Деревья решений — это алгоритмы, которые разбивают пространство признаков на области с однородным ответом, создавая иерархическую структуру решений на основе условий, проверяемых по значениям признаков. Каждый узел дерева соответствует проверке условия, а лист — итоговому прогнозу.

Преимущества деревьев решений включают простоту визуализации и интерпретируемость. Однако одиночные деревья склонны к переобучению, особенно при большой глубине дерева и небольшом объёме данных. Это ограничение устраняется с помощью ансамблевых методов.

Ансамбли моделей объединяют множество слабых моделей в одну сильную. Основные техники:

1. Random Forest (случайный лес):
Создаёт множество деревьев решений, обученных на разных подвыборках исходных данных и случайных подмножествах признаков. Предсказание осуществляется путём голосования (для классификации) или усреднения (для регрессии) по всем деревьям. Такая модель устойчива к переобучению и эффективно работает с высокоразмерными данными.

2. Gradient Boosting (градиентный бустинг):
Модели обучаются последовательно, и каждая новая модель корректирует ошибки предыдущих. Итоговое предсказание формируется как взвешенная сумма всех моделей. Это один из самых точных методов, особенно на структурированных данных, но требует настройки большого числа параметров.

Сравнение моделей:

Параметр	**Decision Tree**	**Random Forest**	**Gradient Boosting**
Интерпретируемость	Высокая	Средняя	Низкая
Устойчивость	Низкая	Высокая	Средняя/высокая
Производительность	Высокая	Средняя	Зависит от реализации

Качество	Среднее	Высокое	Очень высокое

Вопросы и задания

Вопросы, выносимые на обсуждение:

1. Что такое дерево решений и в чем его принцип построения?
2. В каких задачах целесообразно применять деревья решений?
3. Как работает случайный лес и как обеспечивается его устойчивость к переобучению?
4. В чем суть градиентного бустинга и как он строит ансамбль моделей?
5. Какие различия между случайным лесом и градиентным бустингом?
6. Какие параметры влияют на качество деревьев и ансамблей?
7. Что такое переобучение и как с ним борются в ансамблевых методах?
8. Как происходит голосование и усреднение в ансамблевых моделях?
9. Что такое feature importance и как она рассчитывается в Random Forest?
10. Какие популярные библиотеки реализуют данные модели?

Практические задания

Задание 1. Постройте дерево решений для классификации (например, определение вида цветка на основе параметров ириса). Визуализируйте дерево.

Задание 2. Оцените точность дерева на тестовой выборке. Попробуйте изменить глубину дерева и повторите эксперимент. Как это влияет на переобучение?

Задание 3. Постройте модель Random Forest и сравните её точность с одиночным деревом. Постройте график важности признаков.

Задание 4. Постройте модель Gradient Boosting (например, с использованием библиотеки XGBoost или sklearn.ensemble.GradientBoostingClassifier). Проведите сравнение точности с другими моделями.

Задание 5. На основе одного и того же датасета визуализируйте результат работы трёх моделей: дерева решений, случайного леса и градиентного бустинга. Сравните границы решений.

Пример графика важности признаков (feature importance):

Признак	**Важность (Random Forest)**
Длина лепестка	0.45
Ширина лепестка	0.30
Длина чашелистика	0.15
Ширина чашелистика	0.10

Интерпретация:
Из таблицы видно, что длина лепестка вносит наибольший вклад в классификацию, а ширина чашелистика — наименьший.

Вывод:
Использование ансамблевых методов, таких как Random Forest и Gradient Boosting, позволяет значительно повысить точность предсказания моделей машинного обучения и уменьшить риск переобучения за счёт обобщения результатов множества слабых моделей.

Лабораторная работа 5

Методы кластеризации: k-means, DBSCAN, иерархическая кластеризация

Цель: изучить основные методы кластеризации в машинном обучении: алгоритм k-means, иерархическую кластеризацию и метод DBSCAN. Научиться применять эти алгоритмы на практике, сравнивать их особенности и выбирать подходящий метод в зависимости от структуры данных.

Теоретическая часть
Кластеризация — это задача обучения без учителя, целью которой является разбиение множества объектов на подмножества (кластеры) так, чтобы объекты внутри одного кластера были похожи между собой, а объекты из разных кластеров существенно отличались.

1. Метод k-means
Один из самых популярных и простых методов кластеризации. Требует заранее задать количество кластеров *k*. Алгоритм минимизирует внутрикластерное расстояние (обычно евклидово) и работает итеративно:

1. Случайным образом выбираются *k* центров кластеров.
2. Каждая точка назначается ближайшему центру.
3. Центры пересчитываются как среднее всех точек, отнесённых к ним.
4. Шаги 2–3 повторяются до сходимости.

Метод эффективен, но чувствителен к начальному положению центров и форме кластеров (лучше работает с кластерами сферической формы).

2. Иерархическая кластеризация
Не требует заранее заданного числа кластеров. Создаёт древовидную структуру (дендрограмму), объединяя или разъединяя кластеры по определённому критерию (обычно расстояние между точками или центрами кластеров).
Бывает двух видов:

- агломеративная (bottom-up): каждый объект — отдельный кластер, которые постепенно объединяются;
- дивизивная (top-down): все объекты в одном кластере, который затем рекурсивно делится.

Иерархическая кластеризация хорошо визуализирует структуру данных, но плохо масштабируется на большие выборки.

3. DBSCAN (Density-Based Spatial Clustering of Applications with Noise)
Позволяет находить кластеры произвольной формы и автоматически определяет количество кластеров. Основан на плотности:

- точки с достаточным количеством соседей становятся "ядром" кластера;
- к точке присоединяются все "плотно связанные" соседи;
- точки, не входящие ни в один кластер, считаются шумом.

Преимущество DBSCAN — устойчивость к шуму и возможность находить сложные формы кластеров. Однако результаты зависят от выбора параметров ε (радиус окрестности) и MinPts (минимальное число точек).

Сравнительная таблица:

Метод	**Требует k?**	**Устойчив к шуму**	**Кластеры произвольной формы**	**Масштабируемость**
k-means	Да	Нет	Нет	Высокая
Иерархическая	Нет	Нет	Частично	Средняя
DBSCAN	Нет	Да	Да	Средняя/низкая

Вопросы и задания
Вопросы, выносимые на обсуждение:

1. Что такое кластеризация и чем она отличается от классификации?
2. Каков принцип работы алгоритма k-means?
3. Какие недостатки есть у метода k-means?
4. В чём особенности иерархической кластеризации?
5. Что такое дендрограмма и как её использовать?
6. Как работает DBSCAN и в чём его ключевое отличие от других алгоритмов?
7. Как подобрать параметры ε и MinPts в DBSCAN?
8. В каких случаях предпочтительнее использовать иерархическую кластеризацию?
9. Какие метрики качества кластеризации существуют?
10. Что такое шумовые точки в контексте кластеризации?

Практические задания

Задание 1. Загрузите набор данных (например, Iris, Mall Customers или Blobs) и визуализируйте его в двумерном пространстве. Проведите предварительный анализ.

Задание 2. Выполните кластеризацию методом k-means для различных значений *k*. Постройте график «локтя» для определения оптимального числа кластеров.

Задание 3. Постройте дендрограмму для иерархической кластеризации. Проанализируйте, сколько кластеров оптимально выделить.

Задание 4. Примените DBSCAN с различными параметрами. Отобразите кластеры и шумовые точки. Оцените устойчивость алгоритма к выбросам.

Задание 5. Постройте сравнительную таблицу результатов всех трёх алгоритмов по числу кластеров, визуальному качеству и устойчивости к шуму.

Вывод:
Методы кластеризации позволяют группировать данные без предварительной разметки. Выбор метода зависит от формы и структуры данных: для простых задач подойдёт k-means, для визуализации иерархическая кластеризация, а для сложных и зашумлённых — DBSCAN. Комбинация методов и визуальный анализ помогут лучше понять внутреннюю структуру данных.

Лабораторная работа 6

Нейронные сети: архитектура, обучение и основные типы. Создание простой нейросети с использованием фреймворков (TensorFlow/PyTorch)

Цель: изучить архитектуру и принципы обучения нейронных сетей, освоить понятие слоёв и функций активации. Получить практические навыки построения и обучения простой нейронной сети с использованием фреймворка TensorFlow или PyTorch.

Теоретическая часть
Нейронные сети — это класс алгоритмов машинного обучения, вдохновлённый работой биологических нейронов. Основной элемент искусственной нейронной сети — искусственный нейрон, который принимает входные данные, умножает их на веса, суммирует и пропускает через функцию активации. Совокупность таких нейронов формирует слои, а сами слои соединяются в полную архитектуру сети.

Нейросеть может содержать:

- входной слой — получает данные;
- один или несколько скрытых слоёв — обрабатывают информацию;
- выходной слой — формирует ответ системы.

Процесс обучения нейросети представляет собой настройку весов таким образом, чтобы минимизировать функцию потерь. Для этого используется метод обратного распространения ошибки (backpropagation) в сочетании с алгоритмами оптимизации, чаще всего — градиентным спуском.

Существует множество типов нейросетей:

- полносвязные (feedforward) сети — наиболее простые, используются в задачах классификации и регрессии;
- сверточные нейросети (CNN) — хорошо работают с изображениями;
- рекуррентные нейросети (RNN) — применяются в задачах, где важен порядок, например, обработка текста;
- генеративные модели (GAN, autoencoders) — способны создавать новые данные.

В качестве фреймворков для построения нейросетей широко применяются TensorFlow и PyTorch. TensorFlow предоставляет высокий уровень абстракции с использованием Keras, а PyTorch предлагает более гибкую и понятную отладку за счёт динамической модели вычислений.

Вопросы и задания
Вопросы, выносимые на обсуждение:

1. Как устроен искусственный нейрон?
2. Какие функции активации используются в нейросетях и для чего они нужны?
3. Что такое обратное распространение ошибки и какую роль оно играет в обучении нейросети?
4. Чем отличаются полносвязные, сверточные и рекуррентные нейронные сети?
5. В чём особенности архитектуры feedforward-сетей?
6. Какие функции потерь применяются в задачах классификации и регрессии?
7. В чём разница между TensorFlow и PyTorch в контексте построения нейросетей?
8. Что такое эпоха и батч в процессе обучения?
9. Какие существуют способы предотвращения переобучения нейросетей?
10. Как можно оценить качество работы нейросети?

Практические задания

Задание 1. Реализуйте простую полносвязную нейросеть для задачи классификации (например, определение рукописных цифр на наборе данных MNIST). Используйте фреймворк TensorFlow или PyTorch по выбору.

Задание 2. Изучите структуру нейросети: выведите количество параметров, размерность каждого слоя, количество нейронов и используемые функции активации.

Задание 3. Обучите модель и визуализируйте график изменения функции потерь и точности на обучающей и валидационной выборке по эпохам.

Задание 4. Попробуйте изменить структуру сети (добавьте или уберите скрытые слои, измените количество нейронов, функции активации) и проанализируйте влияние этих изменений на точность.

Задание 5. Реализуйте предсказание класса для новых данных, не участвующих в обучении, и оцените точность модели. Отобразите примеры правильно и неправильно классифицированных объектов.

Пример архитектуры сети (TensorFlow, Keras):

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = x_train / 255.0
x_test = x_test / 255.0

y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)

model = Sequential([
    Flatten(input_shape=(28, 28)),
    Dense(128, activation='relu'),
    Dense(64, activation='relu'),
    Dense(10, activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=10, batch_size=64, validation_split=0.2)
```

Вывод:
Понимание архитектуры и принципов работы нейросетей является основой для построения более сложных систем искусственного интеллекта.

Практическая реализация даже простой нейросети помогает усвоить ключевые этапы: от подготовки данных до оценки качества модели. Выбор правильной архитектуры и параметров сети оказывает значительное влияние на точность и устойчивость решения.

Лабораторная работа 7

Сверточные нейронные сети (CNN) для обработки изображений. Разработка CNN для классификации изображений

Цель: изучить архитектуру сверточных нейронных сетей и их применение для обработки изображений. Получить практические навыки построения, обучения и тестирования CNN-модели для задачи классификации изображений с использованием современного фреймворка.

Теоретическая часть

Сверточные нейронные сети (Convolutional Neural Networks, CNN) являются одним из самых эффективных и широко применяемых подходов к анализу изображений, видео и других данных с топологической структурой. В отличие от полносвязных сетей, где каждый нейрон связан со всеми входами, сверточные сети используют локальные связи, что позволяет эффективно извлекать признаки из изображений с минимальным количеством параметров.

Основными структурными элементами CNN являются:

1. Сверточные слои (Convolutional Layers):
Осуществляют свертку входного изображения с набором фильтров (ядер), в результате чего формируются карты признаков (feature maps). Каждое ядро выявляет определённый тип признака — края, углы, текстуру и т.д. Размер окна фильтра, шаг (stride) и паддинг (padding) определяют поведение фильтра на изображении.

2. Нелинейные функции активации:
Наиболее распространённой функцией активации в CNN является ReLU (Rectified Linear Unit), которая позволяет сохранить положительные значения и обнулять отрицательные. Это придаёт модели нелинейность и ускоряет обучение.

3. Слои подвыборки (Pooling Layers):
Применяются для уменьшения размерности карты признаков, снижения вычислительных затрат и повышения устойчивости модели к небольшим смещениям. Наиболее популярный тип — max pooling, который сохраняет максимум значений в окне подвыборки.

4. Полносвязные слои (Dense layers):
На последних этапах CNN подключаются полносвязные слои, которые

обрабатывают извлечённые признаки и формируют финальный выход модели, например, вероятности классов.

5. Softmax-функция:
Обычно используется в выходном слое для задач многоклассовой классификации. Она преобразует выход нейросети в вероятностное распределение по классам.

Обучение CNN аналогично обучению других нейросетей: с помощью метода обратного распространения ошибки и алгоритма оптимизации (например, Adam). При этом градиенты рассчитываются не только для весов полносвязных слоёв, но и для фильтров сверточных слоёв.

CNN обладают высоким качеством распознавания объектов и широко используются в таких задачах, как классификация изображений, детекция объектов, сегментация, распознавание лиц и многие другие.

Вопросы и задания
Вопросы, выносимые на обсуждение:

1. В чём основное отличие архитектуры CNN от полносвязной нейросети?
2. Как работают сверточные слои, и какую роль играют фильтры?
3. Для чего используется операция pooling и как она влияет на обобщающую способность модели?
4. Почему функции активации необходимы в сверточных сетях и какие функции наиболее эффективны?
5. Как влияет глубина сети (количество слоёв) на её способность к извлечению признаков?
6. Какие параметры можно настраивать при создании сверточного слоя?
7. В чём заключается преимущество повторного использования весов в CNN?
8. Как связаны карты признаков и фильтры?
9. Что такое переобучение в контексте CNN и как его можно избежать?
10. В каких практических задачах CNN используются наиболее эффективно?

Практические задания

Задание 1. Постройте сверточную нейронную сеть для распознавания изображений из набора CIFAR-10 (или MNIST, если в первый раз). Используйте минимум два сверточных слоя, слои pooling и один или несколько полносвязных слоёв. Пример архитектуры представлен ниже.

Задание 2. Обучите модель, визуализируйте процесс изменения функции потерь и точности на обучающей и валидационной выборках. Используйте matplotlib.

Задание 3. Внесите изменения в архитектуру: измените количество фильтров, размер ядра, количество слоёв, примените dropout. Сравните результаты и определите оптимальную конфигурацию.

Задание 4. Выведите и визуализируйте первые сверточные фильтры и карты признаков для одного из изображений из тестовой выборки.

Задание 5. Создайте таблицу, отражающую влияние изменения архитектурных параметров на точность модели:

Кол-во слоёв	Dropout	Кол-во фильтров	Точность на тесте
2	нет	32, 64	87.4%
3	0.25	32, 64, 128	89.6%
2	0.5	64, 128	88.1%

Пример реализации (Keras + CIFAR-10):

```
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.utils import to_categorical

(x_train, y_train), (x_test, y_test) = cifar10.load_data()
x_train = x_train / 255.0
x_test = x_test / 255.0

y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)

model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(10, activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=10, batch_size=64, validation_split=0.2)
```

Вывод:
Сверточные нейронные сети демонстрируют выдающуюся эффективность в задачах обработки изображений благодаря способности извлекать и обобщать

визуальные признаки. Практика построения и настройки архитектуры CNN позволяет студентам глубже понять внутренние процессы обучения нейросети и методы повышения точности классификации. Навыки работы с CNN являются фундаментальными для современного ИИ, особенно в таких областях, как компьютерное зрение, медицинская диагностика и автономные системы.

Лабораторная работа 8

Обработка естественного языка (NLP): базовые методы и модели. Практика токенизации, стемминга и работы с Word2Vec

Цель: изучить ключевые методы обработки естественного языка (Natural Language Processing), получить практические навыки работы с текстами: токенизации, стемминга, лемматизации и векторного представления слов на примере модели Word2Vec.

Теоретическая часть

Обработка естественного языка (NLP) — это область искусственного интеллекта, направленная на взаимодействие человека и машины с использованием естественного языка. Основной задачей NLP является преобразование неструктурированного текста в формат, пригодный для анализа и моделирования.

Наиболее распространённые этапы предобработки текста включают:

1. Токенизация — разбиение текста на отдельные элементы (токены): слова, фразы или символы. Это первый и необходимый шаг при работе с текстом, позволяющий перейти от строки к списку лексических единиц.

2. Очистка текста — удаление пунктуации, чисел, знаков препинания, перевод в нижний регистр, удаление стоп-слов (наиболее частотных, но неинформативных слов, таких как «и», «в», «на» и т.п.).

3. Стемминг — процесс приведения слова к его основе (корню) путём обрезания суффиксов и окончаний. Это позволяет уменьшить разнообразие форм слова. Пример: «бегал», «бегающий», «бегают» → «бег».

4. Лемматизация — более интеллектуальный процесс, чем стемминг. Слово приводится к своей начальной (словарной) форме с учётом грамматических правил. Пример: «мышей» → «мышь».

5. Векторизация текста — преобразование текстов в числовой формат. Один из самых популярных методов — модель Word2Vec, которая создаёт эмбеддинги (векторные представления) слов на основе контекста их

употребления. Векторные представления позволяют моделям машинного обучения «понимать» семантическую близость слов.

Word2Vec работает по принципу предсказания слова по контексту (CBOW) или контекста по слову (Skip-gram). Результатом работы являются векторы фиксированной размерности, обладающие интересными свойствами: вектора близких по смыслу слов расположены рядом, а также возможны арифметические операции над векторами (например, vec("король") - vec("мужчина") + vec("женщина") ≈ vec("королева")).

Вопросы и задания
Вопросы, выносимые на обсуждение:

1. В чём заключается задача токенизации и как она реализуется на практике?
2. Каковы различия между стеммингом и лемматизацией?
3. Что такое стоп-слова и зачем они удаляются?
4. Как работают модели Word2Vec? Объясните различие между CBOW и Skip-gram.
5. Какие преимущества даёт представление слов в виде векторов?
6. Как можно оценить семантическую близость слов на основе векторов?
7. В каких прикладных задачах NLP применяются модели Word2Vec?
8. Чем отличаются Bag-of-Words, TF-IDF и Word2Vec?
9. Какие библиотеки Python чаще всего используются для обработки текста?
10. Почему качественная предобработка текста критически важна для результатов машинного обучения?

Практические задания

Задание 1. Реализуйте предобработку текстового корпуса: токенизацию, очистку, удаление стоп-слов, стемминг и/или лемматизацию. Используйте библиотеку NLTK или spaCy.

Задание 2. Постройте частотный словарь токенов и отобразите 10 самых часто встречающихся слов.

Задание 3. Постройте Word2Vec-модель на корпусе новостей или текстов (например, из NLTK или собственного файла). Используйте библиотеку `gensim`.

Задание 4. Визуализируйте векторы 2D (методом снижения размерности t-SNE или PCA) для 30 наиболее частотных слов.

Задание 5. Выполните несколько операций сложения и вычитания векторов слов. Сравните, какие слова находятся ближе всего к полученным векторам.

Пример кода для построения и обучения модели Word2Vec:

```
from nltk.corpus import brown
from gensim.models import Word2Vec
import nltk
nltk.download('brown')

sentences = brown.sents()
model = Word2Vec(sentences, vector_size=100, window=5, min_count=2, sg=1)
model.save("brown_word2vec.model")
```

Пример таблицы — сравнение методов представления текста:

```
similar = model.wv.most_similar('computer', topn=5)
for word, score in similar:
    print(f"{word}: {score:.3f}")
```

Метод	**Описание**	**Учитывает контекст?**	**Размерность**
Bag-of-Words	Счёт количества слов	Нет	По кол-ву слов
TF-IDF	Взвешенный счёт слов	Нет	По кол-ву слов
Word2Vec	Обучение векторов по контексту	Да	Фиксированная (напр., 100)

Вывод:
Базовые методы обработки текста, такие как токенизация, стемминг, лемматизация и векторизация, являются основой любой NLP-системы. Они позволяют преобразовать неструктурированные данные в пригодный для анализа формат. Модель Word2Vec позволяет получить векторные представления слов, обладающие семантическими свойствами, что открывает широкие возможности для анализа текста, построения классификаторов, чат-ботов и рекомендательных систем.

Лабораторная работа 9

Генеративные модели: GAN, VAE и их применение. Генерация изображений или текстов с помощью GAN

Цель: изучить архитектуру и принципы работы генеративных моделей — вариационного автоэнкодера (VAE) и генеративно-состязательной сети

(GAN), а также получить практические навыки по генерации изображений на основе обученной модели GAN.

Теоретическая часть

Генеративные модели — это класс моделей машинного обучения, способных создавать новые данные, схожие по структуре с обучающим набором. В отличие от дискриминативных моделей, которые учатся различать, генеративные учатся создавать. Два наиболее известных представителя этого класса — вариационные автоэнкодеры (VAE) и генеративно-состязательные сети (GAN).

Вариационный автоэнкодер (VAE) представляет собой расширение обычного автоэнкодера, который обучается восстанавливать входные данные через "узкое горлышко" — латентное представление. Основное отличие VAE заключается в том, что в латентном пространстве обучается не один вектор, а распределение (обычно гауссовское). Это позволяет затем легко генерировать новые данные, случайным образом беря точки из этого пространства.

Принцип работы VAE:

1. **Энкодер** получает входное изображение и кодирует его в параметры распределения (среднее и дисперсия).
2. Из распределения берётся сэмпл (вектор z), который затем передаётся в **декодер**.
3. Декодер восстанавливает изображение, приближённое к исходному.
4. Обучение происходит с минимизацией двух составляющих: ошибки восстановления и расстояния между распределением z и нормальным распределением (KL-дивергенция).

Генеративно-состязательные сети (GAN) состоят из двух нейросетей — **генератора** и **дискриминатора**, которые обучаются в состязательном процессе.

Генератор создаёт искусственные данные, стремясь "обмануть" дискриминатор, а дискриминатор, в свою очередь, пытается отличить реальные данные от подделок.

Принцип работы GAN:

1. Генератор получает на вход случайный шум и создает изображение.
2. Дискриминатор получает и реальные, и сгенерированные изображения и выдаёт вероятность того, что данные подлинные.
3. Во время обучения модели поочерёдно обновляют свои веса для улучшения результатов.
   Со временем генератор становится способен создавать изображения, которые трудно отличить от настоящих.

Применения GAN и VAE:

- генерация фотореалистичных изображений;
- улучшение качества изображений (суперразрешение);
- создание DeepFake-видео;
- генерация текста и музыки;
- синтез новых примеров для задач с дефицитом данных.

Вопросы и задания
Вопросы, выносимые на обсуждение:

1. В чём состоит ключевая идея генеративных моделей?
2. Каковы основные архитектурные отличия VAE и GAN?
3. Какие ограничения имеются у VAE и GAN?
4. Объясните, как работают энкодер и декодер в VAE.
5. Какие функции потерь используются при обучении VAE и GAN?
6. Что означает KL-дивергенция в контексте VAE?
7. Почему обучение GAN может быть нестабильным?
8. Какие архитектуры генератора и дискриминатора чаще всего применяются для изображений?
9. Какие подходы позволяют улучшить стабильность обучения GAN?
10. Приведите примеры реальных задач, где используются генеративные модели.

Практические задания

Задание 1. Реализуйте простейшую модель VAE на наборе данных MNIST. Постройте латентное пространство и сгенерируйте изображения из точек этого пространства.

Задание 2. Обучите простую модель GAN на MNIST или Fashion-MNIST. Используйте фреймворк PyTorch или TensorFlow. Визуализируйте полученные изображения на разных эпохах обучения.

Задание 3. Проанализируйте, как изменяются выходы генератора GAN по мере увеличения числа эпох.

Задание 4. Реализуйте функцию потерь дискриминатора и генератора. Сравните использование классических функций и улучшенных вариантов (например, с использованием Wasserstein-GAN).

Задание 5. Сравните качество сгенерированных изображений VAE и GAN. Обсудите визуальные отличия и применимость в разных задачах.

Пример архитектуры простого GAN (PyTorch):

```
class Generator(nn.Module):
    def __init__(self):
        super(Generator, self).__init__()
        self.main = nn.Sequential(
            nn.Linear(100, 256),
            nn.ReLU(True),
            nn.Linear(256, 784),
            nn.Tanh()
        )

    def forward(self, x):
        return self.main(x).view(-1, 1, 28, 28)
```

Пример изображений, сгенерированных GAN (по эпохам):

Эпоха	Примеры изображений
1	Размытые, неразличимые цифры
10	Появляются очертания цифр
30	Большинство цифр различимы
50	Фотореалистичные цифры MNIST

Вывод:

Генеративные модели — один из наиболее перспективных и активно развивающихся разделов искусственного интеллекта. VAE и GAN позволяют создавать новые данные, моделируя их распределение в скрытом пространстве. Несмотря на сложность и нестабильность обучения, GAN зарекомендовали себя как мощный инструмент для генерации изображений, звука и текста. Навыки работы с такими моделями открывают путь к решению задач в креативной индустрии, медиа, медицине и безопасности.