МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
по выполнению лабораторных работ
по дисциплине «Искусственный интеллект в профессиональной сфере»
для студентов направления подготовки
09.03.03 Прикладная информатика
Направленность (профиль)
«Прикладная информатика в экономике»

Ставрополь 2026

Содержание

ЛАБОРАТОРНАЯ РАБОТА №1

Тема: Работа с пакетом NetworkX языка программирования Python

Цель работы

Овладение навыками построения, визуализации и анализа графовых структур с использованием библиотеки NetworkX в языке программирования Python. Изучение методов представления состояний и переходов в задачах поиска.

Теоретическое обоснование

Во многих задачах искусственного интеллекта объекты и отношения между ними могут быть представлены в виде графов. Графовая структура используется при решении задач поиска в пространстве состояний, построении маршрутов, моделировании сетевых взаимодействий, анализа связей и др.

Пакет `NetworkX` является одним из наиболее распространённых инструментов для работы с графами в языке Python. Он предоставляет широкие возможности по созданию как ориентированных, так и неориентированных графов, добавлению и удалению узлов и рёбер, хранению атрибутов элементов графа, а также реализации алгоритмов поиска, центральности, кратчайших путей и других методов анализа.

Работа с графами предполагает как программную реализацию, так и визуализацию структуры графа, что особенно важно при интерпретации результатов в профессиональной сфере. В задачах искусственного интеллекта граф может представлять собой пространство состояний, где вершины — это состояния, а рёбра — возможные действия. На практике графовые структуры используются, например, при планировании действий интеллектуального агента, в системах рекомендаций, логистике, биоинформатике, а также при моделировании поведения пользователей и сетевых систем.

Понимание основных операций с графами и умение реализовать их на практике — это фундамент для дальнейшего изучения методов поиска и принятия решений, включая информированные и неинформированные методы.

Аппаратура и материалы

Персональный компьютер с установленной операционной системой Windows, macOS или Linux
Интерпретатор Python версии 3.7 и выше
Среда разработки: Jupyter Notebook, Google Colab или аналогичная
Установленные пакеты: `networkx`, `matplotlib` (для визуализации)
Методические указания по лабораторной работе

Требования по технике безопасности

Работа за компьютером должна выполняться в хорошо освещённом помещении
Рекомендуется соблюдать режим работы с компьютером: не более 2 академических часов без перерыва
Рабочее место должно быть организовано в соответствии с санитарными нормами
Не допускается установка или модификация программного обеспечения без разрешения преподавателя
Следует избегать работы с личными и конфиденциальными данными в учебной среде

Типовые задачи

1. Реализовать неориентированный граф с пятью узлами и несколькими рёбрами. Отобразить граф с подписями
2. Построить ориентированный граф и вычислить кратчайший путь между двумя заданными вершинами с помощью алгоритма Дейкстры
3. Добавить к графу веса рёбер и реализовать обход графа в ширину, выводя последовательность посещённых вершин

Контрольные вопросы

1. Что такое граф в контексте ИИ и какие задачи он может представлять

2. Чем отличаются ориентированные и неориентированные графы
3. Какие типы данных используются для хранения графов в NetworkX
4. Как в NetworkX осуществляется добавление и удаление узлов и рёбер
5. В чём преимущества визуализации графа
6. Какие алгоритмы обхода графа реализованы в NetworkX
7. Что представляет собой пространство состояний и как оно связано с графом
8. Как реализовать взвешенные рёбра и использовать их в алгоритме поиска
9. Какие ограничения есть у стандартных алгоритмов поиска в графе
10. Как можно сохранить и загрузить граф из файла

ЛАБОРАТОРНАЯ РАБОТА №2

Тема: Реализация поиска в ширину с помощью NetworkX

Цель работы

Ознакомление с реализацией алгоритма поиска в ширину (Breadth-First Search, BFS) в библиотеке NetworkX. Приобретение навыков моделирования задач поиска в пространстве состояний с использованием графов.

Теоретическое обоснование

Поиск в ширину является базовым алгоритмом неинформированного поиска, который используется в задачах нахождения кратчайшего пути в невзвешенном графе. Алгоритм исходит из начального узла и последовательно исследует все его соседние вершины, затем переходит к соседям второго уровня, и так далее, пока не достигнет целевой вершины или не исследует весь граф.

BFS гарантирует нахождение кратчайшего пути (по числу рёбер) в неориентированном графе без отрицательных весов. Он широко применяется в системах искусственного интеллекта для поиска в пространстве состояний, генерации стратегий, анализа маршрутов, организации очередей и планирования. Применение поиска в ширину также актуально в таких задачах, как социальный граф, вычисление расстояния между элементами, нахождение компонентов связности и верификация систем.

Библиотека NetworkX предоставляет удобные инструменты для моделирования графов и выполнения BFS. Методы `networkx.bfs_edges()` и `networkx.shortest_path()` позволяют не только реализовать обход графа, но и извлечь путь к целевому узлу.

Понимание логики поиска в ширину и его применение через практическую реализацию является важным шагом в освоении интеллектуальных алгоритмов и их адаптации к профессиональным задачам.

Аппаратура и материалы

Персональный компьютер с ОС Windows, macOS или Linux
Интерпретатор Python версии 3.7 и выше
Среда разработки: Jupyter Notebook, Google Colab или аналогичная
Установленные пакеты: `networkx`, `matplotlib`
Методические указания к лабораторной работе

Требования по технике безопасности

Работа должна выполняться в хорошо проветриваемом и освещённом помещении
Нужно соблюдать безопасную дистанцию от монитора (не менее 50 см)
Рекомендуется выполнять перерыв через каждые 45–60 минут работы
Запрещается производить установку стороннего ПО без одобрения преподавателя
Следует работать только с учебными файлами и публичными данными

Типовые задачи

1. Реализовать граф из 7 узлов и построить визуальное отображение его структуры

2. Найти путь от заданного начального узла до целевого с помощью функции `networkx.shortest_path()`

3. Сформировать очередь обхода и вывести порядок посещения узлов с использованием `networkx.bfs_edges()`

Контрольные вопросы

1. Какова основная идея поиска в ширину
2. В каких случаях BFS эффективнее других алгоритмов
3. Какой тип структуры данных используется для хранения очереди узлов при BFS
4. Чем отличается BFS от DFS по принципу действия
5. Какие задачи из профессиональной области можно моделировать с использованием BFS
6. Как реализуется граф в библиотеке NetworkX
7. Как сохранить порядок обхода узлов при BFS
8. Как осуществляется визуализация графа в NetworkX
9. Как проверить достижимость узла с помощью поиска в ширину
10. Какие ограничения имеет BFS в приложении к большим графам

ЛАБОРАТОРНАЯ РАБОТА №3

Тема: Реализация поиска в глубину с помощью NetworkX

Цель работы

Ознакомление с реализацией алгоритма поиска в глубину (Depth-First Search, DFS) на графах с использованием библиотеки NetworkX. Освоение принципов обхода графа в глубину и его применения в задачах поиска и анализа.

Теоретическое обоснование

Поиск в глубину — один из основных методов неинформированного поиска, который реализует стратегию исследования максимально глубоко вложенных путей в графе до тех пор, пока не будет достигнута цель или тупик. После этого происходит возврат и поиск продолжается по другим ветвям.

DFS не гарантирует нахождение кратчайшего пути, однако он эффективен с точки зрения памяти, так как требует хранения только текущего пути. Алгоритм находит своё применение в задачах обхода дерева решений, генерации комбинаций, проверки связности графов, планирования маршрутов и построения топологической сортировки.

Библиотека NetworkX предоставляет функции для реализации DFS, включая `dfs_edges()`, `dfs_preorder_nodes()` и `dfs_successors()`. Эти инструменты позволяют как визуализировать порядок обхода, так и анализировать структуру связности графа.

Понимание механизма поиска в глубину и умение применять его в профессиональных задачах позволяет студентам решать широкий спектр задач, включая задачи анализа состояния систем, маршрутизации и построения решений на основе дерева вариантов.

Аппаратура и материалы

Персональный компьютер с операционной системой Windows, macOS или Linux
Интерпретатор Python версии 3.7 и выше
Среда разработки: Jupyter Notebook, Google Colab или аналогичная
Установленные пакеты: `networkx`, `matplotlib`
Методические указания к лабораторной работе

Требования по технике безопасности

Работать в хорошо освещённом помещении, соблюдая санитарные нормы
Следить за положением спины и дистанцией до экрана (не менее 50 см)
Использовать только разрешённые преподавателем средства разработки

Соблюдать регламент продолжительности работы за ПК
Не использовать личные данные или сетевые ресурсы в коде без разрешения

Типовые задачи

1. Построить ориентированный граф с минимум 6 вершинами. Визуализировать его структуру
2. Реализовать обход графа в глубину, используя `networkx.dfs_edges()` и вывести порядок обхода
3. Найти достижимые вершины из заданного узла и построить дерево обхода

Контрольные вопросы

1. В чём состоит принцип работы алгоритма поиска в глубину
2. Какие структуры данных используются при реализации DFS
3. В каких ситуациях DFS предпочтительнее BFS
4. Каковы особенности применения DFS к ориентированным и неориентированным графам
5. Какие функции NetworkX используются для реализации DFS
6. Что такое дерево обхода и как его построить
7. Как можно определить, содержит ли граф циклы, используя DFS
8. Какие ограничения имеет алгоритм DFS
9. Можно ли использовать DFS для поиска в бесконечных графах
10. В чём заключается различие между `dfs_edges()` и `dfs_preorder_nodes()`

ЛАБОРАТОРНАЯ РАБОТА №4

Тема: Реализация итеративного поиска с помощью NetworkX

Цель работы

Освоение принципов и реализация алгоритма поиска с итеративным углублением (Iterative Deepening Search, IDS) средствами Python и библиотеки NetworkX. Изучение сочетания преимуществ поиска в глубину и поиска в ширину.

Теоретическое обоснование

Поиск с итеративным углублением — это стратегия поиска в пространстве состояний, сочетающая достоинства поиска в глубину и в ширину. Алгоритм выполняет серию обходов в глубину, ограничивая их максимальной глубиной, которая поэтапно увеличивается. Такой подход позволяет экономить память, как в DFS, и при этом гарантировать нахождение кратчайшего пути, как в BFS.

Алгоритм IDS особенно эффективен при отсутствии информации о размере или структуре пространства состояний. Он применяется в задачах поиска решений в деревьях и графах, планирования действий, нахождения решений в неограниченных по глубине пространствах с конечным решением.

В библиотеке NetworkX отсутствует встроенная реализация IDS, однако благодаря возможностям создания пользовательских обходов и ограничения глубины в рекурсивной реализации DFS возможно моделирование этого алгоритма вручную.

Освоение итеративного поиска позволяет студентам глубже понять стратегические подходы к решению задач в пространстве состояний и научиться адаптировать алгоритмы к специфике профессиональной предметной области.

Аппаратура и материалы

Персональный компьютер с установленной ОС (Windows, macOS, Linux)
Интерпретатор Python версии 3.7 и выше
Среда разработки: Jupyter Notebook, Google Colab
Установленные пакеты: `networkx`, `matplotlib`
Методические указания к лабораторной работе

Требования по технике безопасности

Работать в чистом и хорошо освещённом помещении
Соблюдать регламент продолжительности непрерывной работы (не более 45–60 минут)
Рабочее место должно соответствовать требованиям эргономики
При выполнении работы использовать только разрешённые программные средства
Не подключать сторонние устройства или использовать внешние ресурсы без согласования

Типовые задачи

1. Реализовать ограниченный по глубине поиск в глубину (DFS) в графе с помощью рекурсивной функции
2. Реализовать итеративное углубление на основе ограниченного DFS. Установить максимальную глубину 5
3. Построить граф из 8–10 узлов, применить IDS и вывести путь к целевой вершине, если он найден

Контрольные вопросы

1. В чём заключается принцип работы поиска с итеративным углублением
2. Какие преимущества IDS имеет по сравнению с DFS и BFS
3. Что происходит при достижении заданного предела глубины
4. Какие ограничения есть у IDS
5. Как можно реализовать ограничение глубины при обходе в NetworkX
6. Почему IDS гарантирует нахождение кратчайшего пути
7. Как влияет структура графа на эффективность IDS
8. Можно ли использовать IDS в графах с циклами
9. Как связаны IDS и стратегии поиска с ограничением ресурсов
10. В каких профессиональных задачах может применяться IDS

ЛАБОРАТОРНАЯ РАБОТА №5

Тема: Реализация жадного эвристического поиска с помощью NetworkX

Цель работы

Овладение навыками реализации жадного эвристического поиска на графах с использованием библиотеки NetworkX. Изучение применения эвристических оценок в алгоритмах информированного поиска.

Теоретическое обоснование

Жадный эвристический поиск (Greedy Best-First Search) — это стратегия информированного поиска, которая выбирает для расширения ту вершину, у которой эвристическая оценка минимальна. Эвристическая функция (`h(n)`) приближённо оценивает расстояние от текущей вершины до целевой.

Жадный поиск является более эффективным по сравнению с неинформированными алгоритмами при наличии эвристики, отражающей геометрическую или логическую близость к цели. Однако он не гарантирует нахождение кратчайшего пути, так как учитывает только предполагаемую близость, а не фактическую стоимость пути.

В задачах профессионального применения жадный поиск может использоваться для приблизительного планирования маршрутов, приоритезации действий, поиске решений в системах, где скорость важнее оптимальности.

В библиотеке NetworkX можно реализовать жадный поиск вручную, используя структуры данных `heapq` (очередь с приоритетом) и пользовательскую функцию `h(n)`, заданную в виде словаря с эвристическими оценками для каждой вершины.

Овладение этой стратегией способствует развитию представления о применимости эвристик и формировании инженерного мышления при проектировании ИИ-систем.

Аппаратура и материалы

Персональный компьютер с ОС Windows, macOS или Linux
Интерпретатор Python версии 3.7 и выше

Среда разработки: Jupyter Notebook, Google Colab или аналогичная
Установленные пакеты: `networkx`, `matplotlib`, `heapq` (входит в стандартную библиотеку Python)
Методические указания к лабораторной работе

Требования по технике безопасности

Работа должна выполняться за исправным оборудованием в соответствии с санитарными нормами
Не допускается работа с конфиденциальными данными
Во время выполнения задания соблюдать временной регламент и делать перерывы
Запрещается изменять конфигурации ПО без разрешения преподавателя
Следует использовать только проверенные источники эвристической информации

Типовые задачи

1. Построить взвешенный ориентированный граф из 8–10 узлов. Назначить каждой вершине эвристическую оценку (в виде словаря)
2. Реализовать алгоритм жадного поиска, при котором на каждом шаге выбирается вершина с минимальным `h(n)`
3. Вывести порядок обхода и путь от начальной до целевой вершины. Сравнить с результатом BFS

Контрольные вопросы

1. Что такое жадный эвристический поиск и чем он отличается от неинформированного
2. Как задаётся эвристическая функция и что она отражает
3. В каких случаях жадный поиск может не найти оптимального решения
4. Какие структуры данных используются в реализации жадного поиска
5. Как в NetworkX хранить и использовать эвристические оценки
6. Какие типы задач в профессиональной сфере подходят для жадного поиска
7. В чём потенциальная опасность переоценки или недооценки `h(n)`
8. Почему важно правильно выбирать эвристику при проектировании системы
9. Что происходит при равенстве эвристических оценок у нескольких узлов
10. Можно ли комбинировать жадный поиск с другими стратегиями

ЛАБОРАТОРНАЯ РАБОТА №6

Тема: Реализация алгоритма A* с помощью NetworkX

Цель работы

Освоение принципов работы алгоритма A* и его реализация средствами Python с использованием библиотеки NetworkX. Формирование навыков проектирования эвристических функций и нахождения оптимальных путей в графах.

Теоретическое обоснование

Алгоритм A* (A-star) — один из наиболее эффективных и популярных алгоритмов информированного поиска в пространстве состояний. Он использует две функции: `g(n)` — фактическую стоимость пути от начальной вершины до текущей, и `h(n)` — эвристическую оценку стоимости пути от текущей вершины до цели. Объединённая оценка `f(n) = g(n) + h(n)` позволяет A* находить путь, который является одновременно кратчайшим и наиболее перспективным с точки зрения эвристики.

Алгоритм гарантирует оптимальность найденного пути при условии, что эвристика является допустимой (не переоценивает расстояние до цели). A* используется в робототехнике, логистике, интеллектуальных системах управления, игровых ИИ, навигационных системах и других профессиональных задачах, где необходимо найти путь с наименьшей стоимостью.

Библиотека NetworkX включает встроенную реализацию `networkx.astar_path()` и `networkx.astar_path_length()`, позволяющие использовать пользовательскую

эвристическую функцию и весовые параметры рёбер. Также можно реализовать A* вручную с использованием очередей с приоритетами.

Знание и практическое применение алгоритма A* формирует у студентов способность к построению эффективных решений задач поиска в реальных профессиональных сценариях.

Аппаратура и материалы

Персональный компьютер с ОС Windows, macOS или Linux
Интерпретатор Python версии 3.7 и выше
Среда разработки: Jupyter Notebook, Google Colab или аналогичная
Установленные пакеты: `networkx`, `matplotlib`
Методические указания к лабораторной работе

Требования по технике безопасности

Выполнять работу в хорошо проветриваемом и освещённом помещении
Соблюдать нормативы по продолжительности непрерывной работы за компьютером
Использовать только официальные библиотеки и разрешённые средства разработки
Исключить использование вредоносных скриптов, сетевого доступа без согласования
Своевременно сохранять результаты работы во избежание потери данных

Типовые задачи

1. Построить взвешенный ориентированный граф из 6–8 узлов, визуализировать его
2. Реализовать алгоритм A* с использованием `networkx.astar_path()`, указав эвристическую функцию
3. Сравнить результат A* с результатами жадного поиска и поиска в ширину. Оценить длину маршрута и число посещённых узлов

Контрольные вопросы

1. Какова формула оценки `f(n)` в алгоритме A*
2. Что означает допустимая эвристика и почему она важна
3. В каких случаях A* может работать неоптимально
4. Чем A* отличается от жадного эвристического поиска
5. Как реализовать собственную эвристическую функцию в NetworkX
6. Как задаются веса рёбер в графе NetworkX
7. Как влияет точность эвристики на производительность A*
8. В каких профессиональных задачах используется A*
9. Что произойдёт, если `h(n)` всегда равно нулю
10. Какие способы оптимизации реализации A* вы знаете

ЛАБОРАТОРНАЯ РАБОТА №7

Тема: Допустимая эвристика. Доминирование эвристик. Релаксация эвристик

Цель работы

Изучение понятий допустимости и доминирования эвристических функций, освоение принципов построения эвристик на основе релаксации исходной задачи. Приобретение навыков анализа корректности эвристических оценок в алгоритмах поиска.

Теоретическое обоснование

Эвристическая функция — это приближённая оценка расстояния от текущей вершины до целевой. В алгоритмах информированного поиска, таких как A*, эвристика играет ключевую роль в эффективности и корректности поиска.

Допустимая эвристика (admissible heuristic) — такая эвристика, которая никогда не переоценивает истинную стоимость пути до цели: h(n)≤h∗ (n)h(n) \leq h^*(n). Использование допустимой эвристики гарантирует, что A* найдёт кратчайший путь.

Доминирование эвристик означает, что одна эвристика информативнее другой, если она всегда выдаёт более высокие оценки, не нарушая допустимости. Более доминирующая эвристика может существенно ускорить поиск.

Релаксация эвристик — метод построения эвристических функций путём упрощения (ослабления) исходной задачи. Например, в задаче планирования можно временно игнорировать ограничения или затраты, чтобы получить приближённую, но допустимую оценку.

Практическое владение этими понятиями позволяет анализировать и проектировать эвристики, адаптированные к конкретным задачам, что особенно важно в профессиональной сфере — при разработке навигационных систем, планировщиков и систем поддержки принятия решений.

Аппаратура и материалы

Персональный компьютер с ОС Windows, macOS или Linux
Интерпретатор Python версии 3.7 и выше
Среда разработки: Jupyter Notebook, Google Colab
Установленные пакеты: `networkx`, `matplotlib`
Методические указания к лабораторной работе

Требования по технике безопасности

Работа за компьютером должна вестись в комфортных условиях с соблюдением санитарных норм
Рекомендуется делать перерыв каждые 45–60 минут
Не допускается использование несанкционированных программных средств
Результаты работы необходимо сохранять регулярно
Работать следует только с учебными данными или открытыми источниками

Типовые задачи

1. Построить граф с 8 вершинами и задать две эвристики: допустимую и недопустимую. Проверить корректность пути, найденного A*
2. Сформировать две эвристики, одна из которых доминирует над другой. Сравнить производительность поиска
3. На основе исходной задачи составить релаксированную задачу и вывести эвристику, полученную из её решения

Контрольные вопросы

1. Что означает допустимость эвристики
2. Как проверяется, является ли эвристика допустимой
3. В чём состоит смысл доминирования одной эвристики над другой
4. Какие преимущества даёт использование более информативной эвристики
5. Что такое релаксация задачи в контексте эвристик
6. Какую задачу можно использовать для построения эвристики в задаче маршрутизации
7. Почему важно не переоценивать стоимость в эвристике
8. Можно ли использовать недопустимую эвристику в A*
9. Какова связь между эвристикой и временем работы поиска
10. Приведите примеры релаксированных задач из профессиональной сферы

ЛАБОРАТОРНАЯ РАБОТА №8

Тема: Реализация алгоритма «Подъём на холм»

Цель работы

Изучение принципов локального поиска и реализация алгоритма «Подъём на холм» (Hill Climbing) для оптимизации функций. Формирование представления о применимости локальных методов в задачах ИИ.

Теоретическое обоснование

Локальные методы поиска используются в ситуациях, когда пространство решений велико, и отсутствует возможность полного перебора. Алгоритм «Подъём на холм» относится к жадным стратегиям и заключается в переходе от текущего состояния к соседнему, имеющему более высокую оценку по целевой функции. Процесс повторяется до достижения локального максимума, после чего алгоритм останавливается.

Основной недостаток алгоритма — высокая вероятность застревания в локальных максимумах. Несмотря на это, он эффективен в задачах, где допустимы приближённые решения или ограничено время вычислений.

Применение подъёма на холм возможно в задачах настройки параметров, оптимизации производственных процессов, компоновки ресурсов, маршрутизации, а также при решении профессиональных задач, требующих быстрого поиска допустимого решения.

Реализация алгоритма требует определения представления состояния, функции оценки и способа генерации соседних состояний. При работе с функциями двух переменных удобна визуализация хода поиска на поверхности значений.

Аппаратура и материалы

Персональный компьютер с ОС Windows, macOS или Linux
Интерпретатор Python версии 3.7 и выше
Среда разработки: Jupyter Notebook, Google Colab
Установленные пакеты: `matplotlib`, `numpy`, `random`
Методические указания к лабораторной работе

Требования по технике безопасности

Работа должна выполняться в комфортных и безопасных условиях
Нужно избегать длительного напряжения зрения и соблюдать рабочие паузы
При использовании ноутбуков следует размещать экран на уровне глаз
Разрешено использовать только проверенное программное обеспечение
Не допускается работа с несанкционированным сетевым доступом или внешними скриптами

Типовые задачи

1. Реализовать алгоритм «Подъём на холм» для функции $f(x)=x\cdot \sin(x)$ на интервале [0, 10]
2. Отобразить график функции и траекторию движения алгоритма к локальному максимуму
3. Изучить влияние выбора начальной точки на результат поиска и количество шагов

Контрольные вопросы

1. Как работает алгоритм «Подъём на холм»
2. Что такое локальный максимум и чем он отличается от глобального
3. Какие недостатки имеет алгоритм и как их можно компенсировать
4. В каких ситуациях алгоритм эффективен
5. Как выбрать функцию оценки для профессиональной задачи
6. Как влияет размер шага на скорость и точность поиска
7. Что произойдёт при отсутствии улучшений в соседних состояниях
8. Можно ли использовать случайные начальные точки для повышения качества поиска
9. В чём отличие подъёма на холм от имитации отжига
10. Приведите примеры использования локального поиска в профессиональной сфере

ЛАБОРАТОРНАЯ РАБОТА №9

Тема: Реализация алгоритма имитации отжига

Цель работы

Овладение методами стохастической оптимизации через реализацию алгоритма имитации отжига (Simulated Annealing). Изучение механизмов принятия решений в условиях локальных экстремумов и случайных колебаний.

Теоретическое обоснование

Алгоритм имитации отжига относится к методам локального поиска, в которых допускается случайное принятие менее выгодных решений с определённой вероятностью. Это позволяет избежать застревания в локальных экстремумах. Принцип основан на аналогии с процессом физического отжига металлов, когда нагретый материал постепенно охлаждается, достигая минимальной внутренней энергии.

Ключевыми компонентами алгоритма являются: функция энергии (или целевая функция), начальная температура, механизм охлаждения (temperature schedule) и правило приёма новых состояний. На начальных этапах алгоритм склонен к исследованию пространства, а ближе к концу — к стабилизации и выбору наилучшего найденного решения.

Имитация отжига применяется в задачах, где требуется найти приближённо оптимальное решение: маршрутизация, планирование, распределение ресурсов, настройка параметров моделей. В профессиональной сфере этот подход особенно полезен там, где пространство решений сложно устроено, а точный перебор невозможен.

Аппаратура и материалы

Персональный компьютер с ОС Windows, macOS или Linux
Интерпретатор Python версии 3.7 и выше
Среда разработки: Jupyter Notebook, Google Colab
Установленные пакеты: `numpy`, `matplotlib`, `random`
Методические указания к лабораторной работе

Требования по технике безопасности

Работа должна выполняться за оборудованием, соответствующим нормам эргономики и санитарии
Необходимо соблюдать режим труда и отдыха
Рекомендуется использовать только разрешённые программные средства и учебные данные
Следует избегать длительного непрерывного использования экранных устройств
Промежуточные результаты следует сохранять во избежание потери данных

Типовые задачи

1. Реализовать алгоритм имитации отжига для функции $f(x)=x\cdot \sin(x)$ на интервале [0, 10]
2. Визуализировать траекторию движения по функции и проанализировать, как меняется выбор при изменении температуры
3. Сравнить поведение алгоритма при разных схемах охлаждения: линейной и экспоненциальной

Контрольные вопросы

1. В чём заключается принцип имитации отжига
2. Что представляет собой функция энергии и как она соотносится с целевой функцией
3. Как определяется вероятность приёма ухудшающего состояния
4. Как влияет температура на поведение алгоритма
5. В чём отличие имитации отжига от подъёма на холм
6. Какие схемы охлаждения существуют и как они влияют на результат
7. Что такое глобальный и локальный минимум в контексте оптимизации
8. В каких задачах предпочтителен стохастический подход
9. Как визуально отследить поведение алгоритма

10. Приведите примеры применения имитации отжига в профессиональной деятельности

ЛАБОРАТОРНАЯ РАБОТА №10

Тема: Реализация лучевого поиска

Цель работы

Изучение принципов лучевого (beam) поиска и его реализация в задачах поиска решений. Овладение методами сокращения пространства поиска с помощью эвристических ограничений.

Теоретическое обоснование

Лучевой поиск (Beam Search) — это разновидность информированного поиска, основанная на модификации жадного подхода. В отличие от алгоритма A* или жадного эвристического поиска, лучевой поиск сохраняет на каждом уровне не все возможные пути, а только ограниченное число наиболее перспективных (по эвристике). Это позволяет существенно сократить требования к памяти и времени, особенно в задачах с широким разветвлением.

Основной параметр алгоритма — ширина луча (beam width), определяющая количество альтернатив, сохраняемых на каждом уровне. Слишком малая ширина может привести к потере оптимального решения, тогда как большая ширина делает алгоритм ближе к полному поиску.

Лучевой поиск применяется в системах машинного перевода, генерации текстов, задачах планирования, логистики, распознавании речи и др. В профессиональной сфере он актуален там, где важно быстро найти приемлемое решение среди множества возможных, не гарантируя оптимальности.

В Python и библиотеке NetworkX лучевой поиск можно реализовать вручную, комбинируя структуры данных с пользовательской функцией оценки.

Аппаратура и материалы

Персональный компьютер с ОС Windows, macOS или Linux
Интерпретатор Python версии 3.7 и выше
Среда разработки: Jupyter Notebook, Google Colab
Установленные пакеты: `networkx`, `heapq`, `matplotlib`
Методические указания к лабораторной работе

Требования по технике безопасности

Выполнять работу в организованной среде с соблюдением требований к размещению экрана и освещённости
Избегать чрезмерного напряжения зрения и регулярно делать перерывы
Не использовать неразрешённые ресурсы и внешние библиотеки
Сохранять промежуточные результаты и комментировать код
Работать только с учебными или открытыми данными

Типовые задачи

1. Реализовать граф с 10 вершинами и назначить каждой вершине эвристическую оценку (например, расстояние до цели)
2. Реализовать лучевой поиск с шириной луча 2 и 3. Зафиксировать путь к цели и число просмотренных узлов
3. Сравнить результаты лучевого поиска с результатом A* и жадного поиска на том же графе

Контрольные вопросы

1. В чём суть алгоритма лучевого поиска
2. Что означает ширина луча и как она влияет на результаты
3. Как формируется список кандидатов на каждом уровне
4. В каких случаях лучевой поиск даёт неточное решение

5. Чем лучевой поиск отличается от жадного поиска
6. Как реализовать приоритетную очередь в Python
7. Как отбираются узлы для расширения на следующем уровне
8. Можно ли адаптировать лучевой поиск под задачи генерации текстов
9. Приведите пример применения лучевого поиска в профессиональной области
10. Почему лучевой поиск не гарантирует нахождение оптимального решения

ЛАБОРАТОРНАЯ РАБОТА №11

Тема: Реализация генетического алгоритма

Цель работы

Овладение основами эволюционного поиска путём реализации генетического алгоритма. Формирование навыков моделирования и настройки популяционных методов оптимизации для решения прикладных задач.

Теоретическое обоснование

Генетический алгоритм (ГА) относится к классу стохастических методов оптимизации, вдохновлённых принципами естественного отбора и эволюции. Он оперирует популяцией возможных решений (особей), которые подвергаются операциям селекции, кроссовера (скрещивания) и мутации. Отбор осуществляется на основе значений целевой функции (функции приспособленности).

ГА не требует гладкости или непрерывности функции и может применяться в дискретных и комбинированных пространствах. Алгоритм хорошо справляется с многомодальными функциями и применим в задачах, где традиционные методы неэффективны.

В профессиональной сфере ГА используется для задач расписаний, распределения ресурсов, проектирования систем, логистики, настройки гиперпараметров и др. Реализация ГА требует выбора схемы кодирования, определения параметров (размер популяции, вероятность мутации, количество поколений) и правил отбора.

Аппаратура и материалы

Персональный компьютер с ОС Windows, macOS или Linux
Интерпретатор Python версии 3.7 и выше
Среда разработки: Jupyter Notebook, Google Colab
Установленные пакеты: `numpy`, `matplotlib`, `random`
Методические указания к лабораторной работе

Требования по технике безопасности

Работать в помещении с достаточным освещением, за оборудованием, соответствующим требованиям эргономики
Соблюдать правила рациональной организации труда, избегать длительной непрерывной работы
Использовать только разрешённое ПО и учебные данные
Регулярно сохранять результаты и контролировать корректность работы кода
Не использовать алгоритмы и библиотеки из внешних непроверенных источников

Типовые задачи

1. Реализовать ГА для оптимизации функции $f(x)=x\cdot \sin(x)$ на интервале [0, 10] с вещественным кодированием
2. Отобразить эволюцию значений приспособленности по поколениям
3. Исследовать влияние параметров алгоритма (размер популяции, вероятность мутации) на результат

Контрольные вопросы

1. Какова основная идея генетического алгоритма
2. Что такое особь, популяция и функция приспособленности

3. Как реализуются операции кроссовера и мутации
4. В чём различие между одноточечным и двухточечным кроссовером
5. Какие критерии остановки применяются в ГА
6. Каковы преимущества и ограничения ГА по сравнению с градиентными методами
7. Почему важна диверсификация популяции
8. Как адаптировать ГА к задачам с ограничениями
9. Приведите примеры задач, решаемых с помощью ГА в профессиональной сфере
10. Как можно визуализировать процесс эволюции решений

ЛАБОРАТОРНАЯ РАБОТА №12

Тема: Работа с пакетом `transformers` языка программирования Python

Цель работы

Овладение базовыми навыками работы с библиотекой `transformers` для решения задач обработки естественного языка. Ознакомление с предобученными языковыми моделями и их применением в профессиональных задачах.

Теоретическое обоснование

Библиотека `transformers`, разработанная компанией Hugging Face, предоставляет унифицированный доступ к широкому спектру современных моделей глубокого обучения для обработки естественного языка (NLP). Она включает модели семейства BERT, GPT, RoBERTa, DistilBERT, T5 и другие.

С помощью `transformers` можно решать разнообразные прикладные задачи: классификация текста, извлечение именованных сущностей (NER), ответ на вопрос (QA), суммаризация, генерация текста и перевод. Библиотека обеспечивает удобную загрузку предобученных моделей и токенизаторов, а также возможность настройки и дообучения.

В профессиональной сфере трансформеры используются для анализа документов, автоматической обработки обращений клиентов, интеллектуального поиска, построения чат-ботов и голосовых ассистентов, что требует уверенного владения инструментами `transformers`.

Работа с этой библиотекой формирует компетенции, связанные с применением больших языковых моделей (LLM) в конкретных предметных областях.

Аппаратура и материалы

Персональный компьютер с ОС Windows, macOS или Linux
Интерпретатор Python версии 3.8 и выше
Среда разработки: Jupyter Notebook, Google Colab
Установленные пакеты: `transformers`, `torch`, `numpy`
Доступ в интернет для загрузки моделей Hugging Face
Методические указания к лабораторной работе

Требования по технике безопасности

Работать в проветриваемом помещении с нормированным уровнем освещения
Соблюдать правила работы за компьютером, включая перерывы каждые 45–60 минут
Использовать официальные репозитории и библиотеки
Избегать передачи в модель конфиденциальных данных
Своевременно сохранять файлы и проверять работу кода на малых примерах

Типовые задачи

1. Использовать модель `distilbert-base-uncased` для задачи классификации: распознать позитивный или негативный тон текста
2. Применить модель `bert-base-cased` для извлечения именованных сущностей из текста профессиональной тематики

3. С помощью пайплайна `question-answering` реализовать систему ответа на вопрос по заданному абзацу

Контрольные вопросы

1. Какие задачи можно решать с помощью библиотеки `transformers`
2. Что такое предобученная модель и зачем она используется
3. Какова роль токенизатора в модели
4. Какие функции предоставляет объект `pipeline`
5. Как загрузить модель из репозитория Hugging Face
6. Что такое task-specific pipeline и приведите примеры
7. Какова структура входных и выходных данных для моделей классификации
8. Какие модели используются для извлечения сущностей
9. Как контролировать параметры генерации ответа в задачах QA
10. В каких прикладных задачах можно использовать трансформеры в вашей профессиональной области

ЛАБОРАТОРНАЯ РАБОТА №13

Тема: Запуск больших языковых моделей с помощью Ollama

Цель работы

Изучить возможности запуска и взаимодействия с большими языковыми моделями (LLM) в локальной среде с использованием инструмента Ollama. Получить практический опыт генерации ответов и обработки пользовательских запросов с помощью моделей, таких как LLaMA, Mistral, Gemma и др.

Теоретическое обоснование

Большие языковые модели (LLM) стали основой современных систем искусственного интеллекта, способных к генерации текстов, ведению диалога, переводу, ответу на вопросы, резюмированию и многим другим функциям. Разработка таких моделей требует значительных вычислительных ресурсов, однако запуск их локальных версий становится возможным благодаря инструментам вроде **Ollama.**

Ollama — это фреймворк, позволяющий просто и удобно запускать и использовать LLM на локальном компьютере. Он предоставляет доступ к множеству популярных моделей, поддерживает взаимодействие через командную строку и API, а также совместим с Jupyter и другими Python-средами.

В профессиональной деятельности локальный запуск LLM даёт возможность обеспечить приватность, автономность и настройку моделей под нужды конкретных задач — от автоматического составления отчётов до интеллектуального анализа текстов.

Аппаратура и материалы

Персональный компьютер с ОС Windows, macOS или Linux
Оперативная память не менее 8 ГБ (желательно от 16 ГБ), наличие дискретного GPU приветствуется
Установлен инструмент Ollama (https://ollama.com/)
Доступ в интернет для первой загрузки моделей
Python-среда (например, Jupyter Notebook) при необходимости интеграции через API
Методические указания к лабораторной работе

Требования по технике безопасности

Работать в нормированных условиях, соблюдая требования к продолжительности и организации труда
Следить за использованием дискового пространства и ресурсами ЦП/ГП
Не использовать модели Ollama для обработки персональных или конфиденциальных данных
При запуске и взаимодействии с LLM соблюдать правила ответственного использования
Работать только с официально поддерживаемыми моделями и сборками

Типовые задачи

1. Установить Ollama на локальную машину и загрузить одну из моделей: `llama2`, `mistral`, `gemma`
2. Провести диалог с моделью через терминал, задав не менее 3-х тематических запросов профессиональной направленности
3. Настроить простое приложение (CLI или скрипт), которое получает входной текст и возвращает ответ от модели Ollama через API

Контрольные вопросы

1. Что такое Ollama и в чём его преимущества по сравнению с облачными LLM-сервисами
2. Какие языковые модели можно запускать через Ollama
3. Каковы минимальные системные требования для работы с Ollama
4. Чем отличается локальный запуск LLM от работы через API (например, OpenAI, YandexGPT)
5. Как происходит взаимодействие с моделью Ollama через терминал
6. Что такое prompt и как его правильно формулировать
7. Как можно интегрировать Ollama с Python-кодом
8. Какие ограничения есть у локальных моделей
9. Какие аспекты безопасности следует учитывать при работе с LLM
10. Каковы потенциальные применения Ollama в вашей профессиональной сфере

ЛАБОРАТОРНАЯ РАБОТА №14

Тема: Работа с YandexGPT

Цель работы

Ознакомление с возможностями использования модели YandexGPT для генерации текстов, ответов на вопросы и поддержки профессиональных задач. Освоение взаимодействия с моделью через веб-интерфейс и API.

Теоретическое обоснование

YandexGPT — это большая языковая модель, разработанная Яндексом, предназначенная для обработки естественного языка на русском языке. Она способна генерировать тексты, резюмировать документы, отвечать на вопросы, вести диалог и выполнять задачи интеллектуальной поддержки пользователей.

В отличие от моделей, размещённых на зарубежных платформах, YandexGPT предоставляет преимущества локализации, соответствия российским требованиям безопасности, а также более высокого качества генерации для русского языка. Модель доступна через веб-интерфейс Yandex Cloud и интегрируемый API, что позволяет применять её в профессиональных решениях: чат-ботах, службах поддержки, экспертных системах.

В профессиональной среде YandexGPT может использоваться для автоматической генерации текстов, анализа данных, формирования ответов в диалоговых системах, интеллектуального поиска, а также как инструмент поддержки принятия решений.

Аппаратура и материалы

Персональный компьютер с доступом в интернет
Аккаунт в Yandex (для получения доступа к Yandex Cloud)
Браузер или Python-среда с библиотеками `requests`, `json`
При необходимости — доступ к Yandex Cloud API и токен авторизации
Методические указания к лабораторной работе

Требования по технике безопасности

Не использовать персональные или конфиденциальные данные в запросах
Работать только в авторизованной и контролируемой среде

Соблюдать правила безопасной работы с внешними API
Сохранять промежуточные и итоговые результаты работы
Работать в условиях, соответствующих санитарным и эргономическим требованиям

Типовые задачи

1. Получить доступ к модели YandexGPT через веб-интерфейс или Yandex Cloud API
2. Сформулировать 3–5 профессионально ориентированных запросов (поиск информации, резюме, генерация инструкций)
3. Написать Python-скрипт, взаимодействующий с YandexGPT по API и обрабатывающий входной текст

Контрольные вопросы

1. Что такое YandexGPT и в чём его отличие от других LLM
2. Какие задачи можно решать с помощью этой модели
3. Как получить доступ к API YandexGPT
4. Что такое токен авторизации и как он используется
5. Как формулировать эффективные запросы (prompt)
6. Какие ограничения действуют при работе с YandexGPT
7. Какие существуют способы интеграции модели в приложения
8. Как контролировать длину и формат ответа модели
9. Какие сферы профессиональной деятельности выигрывают от внедрения YandexGPT
10. Чем YandexGPT может быть полезен для автоматизации текстовой аналитики

ЛАБОРАТОРНАЯ РАБОТА №15

Тема: Обучение с подкреплением: реализация табличного Q-Learning на примере задачи GridWorld

Цель работы

Овладение базовыми принципами обучения с подкреплением путём реализации алгоритма Q-Learning в дискретной среде. Формирование навыков построения и обновления таблицы Q, выбора действий, анализа стратегии агента.

Теоретическое обоснование

Обучение с подкреплением (Reinforcement Learning, RL) — это подход в машинном обучении, в котором агент обучается путём взаимодействия со средой и получения вознаграждения за действия. В отличие от обучения с учителем, здесь отсутствуют заранее размеченные данные, и целевая функция формируется на основе опыта.

Алгоритм Q-Learning относится к методам обучения с использованием ценности действий (value-based learning). Он направлен на обучение оптимальной стратегии выбора действий с максимизацией суммарного вознаграждения. Для каждой пары «состояние-действие» Q(s,a)Q(s, a) хранится оценка полезности. Обновление производится по формуле Беллмана:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \cdot [r + \gamma \cdot \max_{a'} Q(s',a') - Q(s,a)]$$

где:

- ss — текущее состояние,
- aa — выбранное действие,
- rr — вознаграждение,
- s′s' — следующее состояние,
- α\alpha — скорость обучения (learning rate),
- γ\gamma — коэффициент дисконтирования (discount factor).

Примером среды для такой задачи может быть GridWorld — двумерное поле, по которому агент перемещается к цели, избегая препятствий. Это простая, но наглядная

модель, иллюстрирующая обучение через опыт. Реализация Q-Learning в GridWorld позволяет продемонстрировать, как агент адаптирует поведение и формирует стратегию.

Аппаратура и материалы

Персональный компьютер с ОС Windows, macOS или Linux
Среда разработки: Jupyter Notebook, Google Colab
Интерпретатор Python 3.7+
Установленные пакеты: `numpy`, `matplotlib`, (по желанию — `gymnasium`, `seaborn`)
Методические указания к лабораторной работе

Требования по технике безопасности

Работа должна выполняться в хорошо освещённом и организованном рабочем месте
Соблюдать правила работы за компьютером: перерывы каждые 45–60 минут
Не использовать сторонние библиотеки и среды без разрешения
При запуске циклов обучения контролировать нагрузку на систему
Не применять обученные модели вне учебных задач без проверки

Типовые задачи

1. Реализовать среду GridWorld размерами 5×5, в которой цель находится в правом нижнем углу, а несколько ячеек являются «ловушками» с отрицательным вознаграждением
2. Реализовать табличный Q-Learning с параметрами: $\varepsilon = 0.1$, $\alpha = 0.5$, $\gamma = 0.9$. Обучить агента на 1000 эпизодах
3. Визуализировать полученную Q-таблицу и построить траекторию движения агента из стартовой точки при следовании оптимальной стратегии

Контрольные вопросы

1. Что представляет собой обучение с подкреплением
2. Какова цель использования таблицы Q в алгоритме
3. Что означают параметры α, γ и ε в Q-Learning
4. Почему используется ε-жадная стратегия
5. В чём разница между обучением с учителем и обучением с подкреплением
6. Почему алгоритм Q-Learning можно считать off-policy
7. Какие ограничения имеет табличный Q-Learning
8. Что произойдёт, если коэффициент обучения α слишком велик или мал
9. Как интерпретировать значения в Q-таблице
10. Где в профессиональной сфере может быть применён Q-Learning

ЛАБОРАТОРНАЯ РАБОТА №16

Тема: Сравнение активного и пассивного обучения с подкреплением

Цель работы

Изучить различия между активным и пассивным обучением с подкреплением, освоить методы оценки политики и построения оптимальной стратегии на примере дискретной среды. Научиться анализировать эффективность поведения агента при фиксированной и обучаемой политике.

Теоретическое обоснование

В обучении с подкреплением агент изучает взаимодействие со средой, стремясь максимизировать суммарное вознаграждение. Различают два подхода:

- **Пассивное обучение** — агент исследует поведение в среде при заданной (фиксированной) политике и стремится оценить её качество, не изменяя стратегию.
- **Активное обучение** — агент одновременно оценивает среду и изменяет стратегию на основе накопленного опыта, обучаясь выбирать наилучшие действия.

Пассивный подход полезен в случаях, когда политика задаётся извне (например, экспертами), и требуется её оценка. Активный подход реализуется в классических

алгоритмах, таких как Q-Learning или SARSA, где стратегия формируется агентом на основе опыта и обратной связи.

В данной работе используется среда **FrozenLake** из пакета `gymnasium`, моделирующая задачу движения по скользкой поверхности с целью достижения цели, избегая ям. Это стохастическая среда, что делает её подходящей для анализа стратегий.

Аппаратура и материалы

Персональный компьютер с ОС Windows, macOS или Linux
Среда разработки: Jupyter Notebook, Google Colab
Язык программирования Python версии 3.8+
Установленные пакеты: `gymnasium`, `numpy`, `matplotlib`
Методические указания к лабораторной работе

Требования по технике безопасности

Соблюдать нормы организации труда и освещения рабочего места
Использовать только одобренные преподавателем библиотеки
Регулярно сохранять промежуточные результаты и проверять корректность работы кода
Избегать чрезмерной нагрузки на вычислительные ресурсы
Не использовать внешние или неофициальные реализации среды

Типовые задачи

1. Запустить среду `FrozenLake-v1` и реализовать пассивную оценку заданной политики, используя алгоритм **Policy Evaluation**
2. Реализовать алгоритм активного Q-Learning и обучить агента в той же среде
3. Сравнить поведение и итоговую стратегию в обоих случаях по количеству успешных эпизодов и среднему вознаграждению

Контрольные вопросы

1. В чём принципиальное отличие активного и пассивного обучения с подкреплением
2. Что означает политика агента и как она формируется
3. В каких случаях целесообразно использовать пассивное обучение
4. Какие задачи можно решать с помощью активного обучения
5. Как оценивается ценность состояния (Value Function)
6. Что такое функция действия (Action-Value Function)
7. Почему среда FrozenLake считается стохастической
8. Какие показатели можно использовать для сравнения стратегий
9. Как влияет ε-жадная стратегия на активное обучение
10. Какие ограничения есть у пассивного подхода

ЛАБОРАТОРНАЯ РАБОТА №17

Тема: Основы обработки изображений с использованием OpenCV

Цель работы

Изучить основные операции обработки изображений с использованием библиотеки OpenCV. Освоить методы чтения, отображения, преобразования и фильтрации изображений как этапа подготовки данных в системах компьютерного зрения.

Теоретическое обоснование

Обработка изображений — фундаментальный этап в системах компьютерного зрения, от качества которого зависит успех задач распознавания, классификации, отслеживания и анализа объектов. Современные подходы предполагают использование специализированных библиотек, таких как **OpenCV** (Open Source Computer Vision Library), которая обеспечивает широкий набор инструментов для захвата, обработки и анализа изображений.

Основные операции включают:

- **Загрузка и отображение изображений** в различных цветовых форматах (BGR, RGB, Grayscale);
- **Изменение размера и ориентации изображений** (масштабирование, поворот);
- **Фильтрация** (размытие, медианный фильтр, гауссов фильтр);
- **Обнаружение границ** с помощью операторов Собеля, Лапласа и Canny;
- **Работа с цветами** (перевод между цветовыми пространствами, маскирование по цвету).

Овладение этими методами позволяет готовить изображения для последующего анализа, что особенно важно в профессиональных задачах автоматизации визуального контроля, диагностики, видеонаблюдения, анализа документов и др.

Аппаратура и материалы

Персональный компьютер с ОС Windows, macOS или Linux
Среда разработки: Jupyter Notebook, Google Colab
Интерпретатор Python версии 3.8 и выше
Установленные пакеты: `opencv-python`, `numpy`, `matplotlib`
Набор изображений в формате JPG/PNG (подготовленный преподавателем или загруженный из открытых источников)
Методические указания к лабораторной работе

Требования по технике безопасности

Работа должна выполняться в эргономичной среде с правильной организацией рабочего места
Соблюдать регламент времени за компьютером (перерывы каждые 45–60 минут)
Избегать обработки персональных и конфиденциальных изображений
Сохранять промежуточные результаты обработки и проверять корректность работы кода
Следовать инструкциям преподавателя при загрузке внешних изображений

Типовые задачи

1. Считать и отобразить изображение в цвете и в градациях серого, сохранить результат
2. Изменить размер изображения, повернуть его на 90 градусов и отразить по горизонтали
3. Применить к изображению три фильтра: размытие по Гауссу, медианный фильтр и оператор Canny для выделения границ

Контрольные вопросы

1. Какие форматы изображений поддерживает OpenCV
2. В чём отличие цветовых пространств BGR и RGB
3. Как изменить размер изображения и зачем это необходимо
4. Что такое фильтрация и какие фильтры применяются в OpenCV
5. Для чего применяется оператор Canny
6. Как отобразить несколько изображений на одном графике
7. Какие типы искажения изображения могут мешать его анализу
8. Чем отличается медианный фильтр от гауссового
9. В каких задачах используется обработка изображений в профессиональной сфере
10. Какие ограничения есть у OpenCV при работе с изображениями большого размера

ЛАБОРАТОРНАЯ РАБОТА №18

Тема: Обнаружение и классификация объектов на изображениях с использованием каскадов Хаара и нейронных сетей

Цель работы

Освоить методы детекции объектов на изображениях с использованием алгоритма каскадов Хаара и предобученных нейросетевых моделей. Получить практические навыки применения инструментов OpenCV и нейронных сетей для задач распознавания объектов.

Теоретическое обоснование

Обнаружение объектов — одна из ключевых задач компьютерного зрения, заключающаяся в локализации и идентификации объектов на изображении. Существуют два основных подхода:

- **Каскады Хаара** — классический метод на основе каскадных классификаторов и признаков Хаара. Эффективен для детекции простых объектов (например, лиц) в условиях ограниченных вычислительных ресурсов.
- **Нейронные сети** — современные методы, основанные на глубоких сверточных архитектурах (например, MobileNet, YOLO, SSD), позволяющие выполнять как локализацию, так и классификацию объектов с высокой точностью.

В библиотеке OpenCV реализован удобный интерфейс для работы с каскадами Хаара и интеграции с нейросетевыми моделями, загружаемыми в формате ONNX или через модуль `dnn`. Эти подходы широко применяются в системах видеонаблюдения, интеллектуального анализа изображений, робототехнике, медицине и промышленности.

Аппаратура и материалы

Персональный компьютер с ОС Windows, macOS или Linux
Среда разработки: Jupyter Notebook, Google Colab
Язык программирования Python версии 3.8 и выше
Установленные пакеты: `opencv-python`, `numpy`, `matplotlib`
Файлы каскадов Хаара (`haarcascade_frontalface_default.xml`)
Предобученные нейросетевые модели (например, MobileNet SSD или YOLO в формате ONNX или `.pb`)
Изображения с объектами (лица, автомобили, знаки и пр.)
Методические указания к лабораторной работе

Требования по технике безопасности

Работать в организованном, проветриваемом помещении
Избегать перегрузки вычислительных ресурсов при обработке видео и больших изображений
Не использовать конфиденциальные или персональные изображения
Соблюдать правила обращения с файлами и сетевыми ресурсами
Контролировать корректность сохранения и отображения результатов

Типовые задачи

1. Использовать встроенный классификатор OpenCV для обнаружения лиц на изображении с помощью каскадов Хаара
2. Загрузить предобученную нейросетевую модель и выполнить детекцию объектов на фотографии с помощью модуля `cv2.dnn`
3. Сравнить качество и скорость срабатывания каскада Хаара и нейросети на одинаковом наборе изображений

Контрольные вопросы

1. Что такое каскад Хаара и как он работает
2. Какие преимущества и недостатки у каскадов Хаара
3. В чём отличие между детекцией и классификацией объектов
4. Как загружается и используется нейросетевая модель в OpenCV
5. Что такое предварительная обработка входных изображений для нейросетей
6. Как реализовать прорисовку ограничивающих прямоугольников (bounding boxes)
7. Каковы требования к входным изображениям в `cv2.dnn`
8. Какие параметры влияют на точность и производительность детекции

9. Как интерпретировать confidence score в результатах нейросети

10. Где в профессиональной деятельности применяются методы детекции объектов

ЗАКЛЮЧЕНИЕ

Лабораторные работы, представленные в рамках дисциплины *«Искусственный интеллект в профессиональной сфере»*, являются неотъемлемой частью учебного процесса и направлены на формирование у студентов практических навыков применения методов искусственного интеллекта к реальным задачам профессиональной деятельности.

В ходе выполнения лабораторных заданий студенты осваивают ключевые алгоритмы и технологии: от неинформированных и эвристических стратегий поиска до локальных методов оптимизации и обучения с подкреплением, от базовой обработки изображений до работы с современными языковыми моделями и нейросетевыми средствами детекции. Особое внимание уделено реализации алгоритмов в средах программирования Python, использовании специализированных библиотек (`NetworkX`, `OpenCV`, `transformers`, `gymnasium`, `Ollama`, `YandexGPT`) и интерпретации полученных результатов в прикладном контексте.

Комплекс лабораторных работ способствует развитию системного мышления, алгоритмической культуры, способности к инженерному моделированию и критическому анализу. Ориентация заданий на профессионально-ориентированные кейсы усиливает междисциплинарную направленность подготовки специалистов и создает основу для дальнейшего освоения продвинутых ИИ-инструментов в исследовательской и прикладной деятельности.

Выполнение лабораторных работ по дисциплине обеспечивает практико-ориентированное усвоение теоретических положений, способствует развитию компетенций ПК-1 и ПК-2 и формирует готовность к использованию интеллектуальных систем в своей профессиональной сфере.

УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Основная литература:

Болотский, А. В.
    Исследование операций и методы оптимизации Электронный ресурс / Болотский А. В., Кочеткова О. А. - Санкт-Петербург : Лань, 2020. - 116 с. - ISBN 978-5-8114-4568-4

Сараев, П. В.
    Методы машинного обучения Электронный ресурс : Методические указания и задания к лабораторным работам по курсу / П. В. Сараев. - Липецк : Липецкий государственный технический университет, ЭБС АСВ, 2017. - 48 с. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 2227-8397

Сопов, Е. А.
    Многокритериальные нейроэволюционные системы в задачах машинного обучения и человеко-машинного взаимодействия Электронный ресурс / Сопов Е. А., Иванов И. А. : монография. - Красноярск : СФУ, 2019. - 160 с. - ISBN 978-5-7638-3969-2

Дополнительная литература:

Методические рекомендации к практическим занятиям по дисциплине "Теория и практика машинного перевода" : Направление подготовки 45.04.02 Лингвистика. Магистерская программа "Перевод и переводоведение".Квалификация выпускника - магистр. Форма обучения - очная. Учебный план 2015 года. Изучается в 3 семестре. - Ставрополь : СКФУ, 2015. - 9 с.

Неделько, В. М.
    Основы статистических методов машинного обучения Электронный ресурс : Учебное пособие / В. М. Неделько. -

Новосибирск : Новосибирский государственный технический университет, 2010. - 72 c. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-5-7782-1385-2

Рапаков, Г. Г.
    Методы и алгоритмы машинного обучения при принятии управленческих решений в региональной системе медицинской профилактики (опыт Вологодской области) Электронный ресурс / Рапаков Г. Г., Касимов Р. А. : монография. - Вологда : ВоГУ, 2014. - 143 c. - ISBN 978-5-87851-548-1

Учебно-методическая литература:

Сараев, П. В.
    Методы машинного обучения Электронный ресурс : Методические указания и задания к лабораторным работам по курсу / П. В. Сараев. - Липецк : Липецкий государственный технический университет, ЭБС АСВ, 2017. - 48 c. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 2227-8397