Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Уфимский государственный нефтяной технический университет» в г. Салавате
Кафедра «Электрооборудование и автоматика промышленных предприятий»

Учебно-методическое пособие по дисциплине «Проблемы искусственного интеллекта, распознавание образов»

Учебно-методическое пособие к практическим работам, самостоятельной работе обучающихся

Салават 2021

Учебно-методические указания предназначены для студентов направления подготовки 13.04.02 «Электроэнергетика и электротехника», магистерская программа: «Интеллектуальные средства и системы управления, защиты и диагностики электроэнергетических комплексов».

Указания содержат цели работы, теоретические сведения, практические задания, рекомендации для самостоятельной работы студентов.

Составитель: Д. Г. Чурагулов, старший преподаватель кафедры ЭАПП

Рецензенты: Р.Г. Вильданов, доктор техн. наук, профессор кафедры ЭАПП

А.С. Хисматуллин, канд. физ.- мат. наук, доцент кафедры ЭАПП

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Дисциплина «Проблемы искусственного интеллекта, распознавание образов» относится к обязательной части.

Дисциплины, для которых освоение данной дисциплины необходимо как предшествующее «Методология проектирования в электроэнергетике и электротехнике»; «Ознакомительная практика»; «Теоретические и экспериментальные методы научных исследований». Дисциплина является предшествующей для подготовки и написания диссертации.

Учебно-методические указания предназначены для студентов направления подготовки 13.04.02 «Электроэнергетика и электротехника», магистерская программа: «Интеллектуальные средства и системы управления, защиты и диагностики электроэнергетических комплексов».

Указания содержат цели работы, теоретические сведения, практические задания, рекомендации для самостоятельной работы студентов.

Может быть рекомендовано студентам занимающихся программированием, математическим моделированием и численными методами, а также может служить справочным материалом при выполнении курсовых и выпускных квалификационных работ, связанных с расчетами на компьютере.

1 Общие сведения о языке программирования Python

Python – высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным – всё является объектами. Необычной особенностью языка является выделение блоков кода пробельными отступами. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации. Сам же язык известен как интерпретируемый и используется в том числе для написания скриптов. Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нём программ по сравнению с аналогичным кодом, написанным на компилируемых языках, таких как C# или C++.

Python является мультипарадигмальным языком программирования, поддерживающим императивное, процедурное, структурное, объектно-ориентированное программирование, метапрограммирование и функциональное программирование. Задачи обобщённого программирования решаются за счёт динамической типизации]. Аспектно-ориентированное программирование частично поддерживается через декораторы, более полноценная поддержка обеспечивается дополнительными фреймворками. Такие методики как контрактное и логическое программирование можно реализовать с помощью библиотек или расширений. Основные архитектурные черты – динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений с глобальной блокировкой интерпретатора (GIL), высокоуровневые структуры данных. Поддерживается разбиение программ на модули, которые, в свою очередь, могут объединяться в пакеты.

Эталонной реализацией Python является интерпретатор CPython, поддерживающий большинство активно используемых платформ и являющийся стандартом де-факто языка]. Он распространяется под свободной лицензией Python

Software Foundation License, позволяющей использовать его без ограничений в любых приложениях, включая проприетарные. CPython компилирует исходные тексты в высокоуровневый байт-код, который исполняется в стековой виртуальной машине. К другим трём основным реализациям языка относятся Jython (для JVM), IronPython (для CLR/.NET) и PyPy. PyPy написан на подмножестве языка Python (RPython) и разрабатывался как альтернатива CPython с целью повышения скорости исполнения программ, в том числе за счёт использования JIT-компиляции.

Стандартная библиотека включает большой набор полезных переносимых функций, начиная от функционала для работы с текстом и заканчивая средствами для написания сетевых приложений. Дополнительные возможности, такие как математическое моделирование, работа с оборудованием, написание веб-приложений или разработка игр, могут реализовываться посредством обширного количества сторонних библиотек, а также интеграцией библиотек, написанных на C# или C++, при этом и сам интерпретатор Python может интегрироваться в проекты, написанные на этих языках.

Python – это язык с динамическим контролем типа, в котором имена во время выполнения программы могут представлять значения различных типов. И действительно, имена, используемые в программе, – это только метки для различных величин и объектов. Оператор присваивания просто создает связь между именем и значением. В этом состоит одно из отличий данного языка, например, от C, в котором имена представлены объектами с постоянным размером и размещением в памяти, где находятся результаты.

Все данные в Python представлены объектами. Имена являются лишь ссылками на эти объекты и не несут нагрузки по декларации типа. Значения встроенных типов имеют специальную поддержку в синтаксисе языка: можно записать литерал строки, числа, списка, кортежа, словаря (и их разновидностей). Синтаксическую же поддержку операций над встроенными типами можно легко сделать доступной и для объектов определяемых пользователями классов.

Следует также отметить, что объекты могут быть неизменчивыми и изменчивыми. Например, строки в Python являются неизменчивыми, поэтому

операции над строками создают новые строки.

Карта встроенных типов (с именами функций для приведения к нужному типу и именами классов для наследования от этих типов):

1. Специальные типы: None, NotImplemented и Ellipsis;

2. Числа:

1) целые:

- обычное целое int;

- целое произвольной точности long;

- логическое bool;

2) число с плавающей точкой float;

3) комплексное число complex.

3. Последовательности:

1) неизменчивые:

- строка str;

- Unicode-строка Unicod;

- кортеж tuple;

2) изменчивые:

- список list;

- отображения;

- словарь dict.

4. Объекты, которые можно вызвать:

1) функции (пользовательские и встроенные);

2) функции-генераторы;

3) методы (пользовательские и встроенные);

4) классы (новые и «классические»);

5) экземпляры классов (если имеют метод _call_).

5. Модули.

6. Файлы file.

7. Вспомогательные типы buffer, slice.

2 Описание программного продукта Matlab

MATLAB – это высокоуровневый язык и интерактивная среда для программирования, численных расчетов и визуализации результатов. С помощью MATLAB можно анализировать данные, разрабатывать алгоритмы, создавать модели и приложения.

Язык, инструментарий и встроенные математические функции позволяют вам исследовать различные подходы и получать решение быстрее, чем с использованием электронных таблиц или традиционных языков программирования, таких как C/C++ или Java.

MATLAB широко используется в таких областях, как:

- обработка сигналов и связь;
- обработка изображений и видео;
- системы управления;
- автоматизация тестирования и измерений;
- финансовый инжиниринг;
- вычислительная биология и т.п.

Более миллиона инженеров и ученых по всем миру используют MATLAB в качестве языка технических вычислений.

MATLAB по сравнению с традиционными языками программирования (C/C++, Java, Pascal, FORTRAN) позволяет на порядок сократить время решения типовых задач и значительно упрощает разработку новых алгоритмов.

MATLAB представляет собой основу всего семейства продуктов MathWorks и является главным инструментом для решения широкого спектра научных и прикладных задач, в таких областях как: моделирование объектов и разработка систем управления, проектирование коммуникационных систем, обработка сигналов и изображений, измерение сигналов и тестирование, финансовое моделирование, вычислительная биология и др.

Ядро MATLAB позволяет максимально просто работать с матрицами реальных, комплексных и аналитических типов данных и со структурами данных и

таблицами поиска.

MATLAB содержит встроенные функции линейной алгебры (LAPACK, BLAS), быстрого преобразования Фурье (FFTW), функции для работы с полиномами, функции базовой статистики и численного решения дифференциальных уравнений; расширенные математические библиотеки для Intel MKL.

Все встроенные функции ядра MATLAB разработаны и оптимизированы специалистами и работают быстрее или так же, как их эквивалент на C/C++.

Simulink - это графическая среда имитационного моделирования, позволяющая при помощи блок-диаграмм в виде направленных графов, строить динамические модели, включая дискретные, непрерывные и гибридные, нелинейные и разрывные системы.

Интерактивная среда Simulink, позволяет использовать уже готовые библиотеки блоков для моделирования электросиловых, механических и гидравлических систем, а также применять развитый модельно-ориентированный подход при разработке систем управления, средств цифровой связи и устройств реального времени.

Дополнительные пакеты расширения Simulink позволяют решать весь спектр задач от разработки концепции модели до тестирования, проверки, генерации кода и аппаратной реализации. Simulink интегрирован в среду MATLAB, что позволят использовать встроенные математические алгоритмы, мощные средства обработки данных и научную графику.

Simulink Library Browser (средство просмотра Библиотеки Simulink) содержит в себе библиотеку блоков наиболее часто используемых для моделирования систем.

В эту библиотеку входят:

- блоки непрерывной и дискретной динамики, такие как Integrator (Интегратор) и Unit Delay (Звено Задержки);

- алгоритмические блоки, такие как Sum (Сумматор), Product (Произведение), Lookup Table (Справочная Таблица);

- структурные блоки, такие как Mux (Мультиплексор), Switch

(Переключатель), Bus Selector (Селектор Шины).

Можно выполнять симуляцию динамических свойств системы и просматривать результаты, как только симуляция началась.

Чтобы гарантировать заданную скорость симуляции и точность, Simulink предоставляет ODE решатели с фиксированным и переменным шагом, графический отладчик и подпрограмму оценки времени выполнения отдельных функций модели.

Решатели - это числовые алгоритмы интегрирования, которые вычисляют динамику системы в течение определенного промежутка времени, используя информацию, содержащуюся в модели.

Simulink предоставляет решатели для симуляции широкого диапазона типов систем, включая системы непрерывного времени (аналоговые), дискретного времени (цифровые), гибридные (смешанный сигнал) и системы с различными периодами дискретизации любого размера.

При помощи решателей в Simulink можно выполнять симуляцию жёстких систем и систем с разрывами. Можно задавать опции симуляции, включая тип и свойства решателя, время начала и конца симуляции и выполнять загрузку или сохранение данных симуляции. Можно также настраивать оптимизационную и диагностическую информацию. Вместе с моделью можно сохранять разные опциональные комбинации.

Ключевые особенности:

- интерактивная графическая среда для построения блок-диаграмм;

- расширяемая библиотека готовых блоков;

- удобные средства построение многоуровневых иерархических многокомпонентных моделей;

- средство навигации и настройки параметров сложных моделей - Model Explorer;

- средства интеграции готовых C/C++, FORTRAN,ADA и MATLAB-алгоритмов в модель, взаимодействие с внешними программами для моделирования;

- современные средства решения дифференциальных уравнений для

непрерывных, дискретных, линейных и нелинейных объектов (в т.ч. с гистерезисом и разрывами);

- имитационное моделирование нестационарных систем с помощью решателей с переменным и постоянным шагом или методом управляемого из MATLAB пакетного моделирования;

- удобная интерактивная визуализация выходных сигналов, средства настройки и задания входных воздействий;

- средства отладки и анализа моделей;

- полная интеграция с MATLAB, включая численные методы, визуализацию, анализ данных и графические интерфейсы;

Fuzzy Logic Toolbox – это пакет расширения MATLAB, содержащий инструменты для проектирования систем нечеткой логики.

Пакет позволят создавать экспертные системы на основе нечеткой логики, проводить кластеризацию нечеткими алгоритмами, а также проектировать нечеткие нейросети.

Пакет включает графический интерфейс для интерактивного пошагового проектирования нечетких систем, функции командной строки для разработки программ, а также специальные блоки для построения систем нечеткой логики в Simulink.

Все функции пакета написаны на открытом языке MATLAB, что позволяет контролировать исполнение алгоритмов, изменять исходный код, а также создавать свои собственные функции и процедуры.

Первая категория программных инструментов пакета Fuzzy Logic Toolbox содержит функции, которые могут быть вызваны непосредственно путем набора имени функции в командном окне (command line) или из собственных пользовательских приложений. Большинство из этих функций представляют собой матлабовские функции в виде m-файлов. В этом случае пользователь может посмотреть запрограммированные в этих функциях алгоритмы а также редактировать и корректировать эти файлы.

Программа Simulink является приложением к пакету MATLAB. При моделировании с использованием Simulink реализуется принцип визуального программирования, в соответствии с которым, пользователь на экране из библиотеки стандартных блоков создает модель устройства и осуществляет расчеты. При этом, в отличие от классических способов моделирования, пользователю не нужно досконально изучать язык программирования и численные методы математики, а достаточно общих знаний требующихся при работе на компьютере и, естественно, знаний той предметной области в которой он работает.

Simulink является достаточно самостоятельным инструментом MATLAB и при работе с ним совсем не требуется знать сам MATLAB и остальные его приложения. С другой стороны доступ к функциям MATLAB и другим его инструментам остается открытым и их можно использовать в Simulink. Часть входящих в состав пакетов имеет инструменты, встраиваемые в Simulink (например, LTI-Viewer приложения Control System Toolbox – пакета для разработки систем управления). Имеются также дополнительные библиотеки блоков для разных областей применения (например, Power System Blockset – моделирование электротехнических устройств, Digital Signal Processing Blockset – набор блоков для разработки цифровых устройств и т.д).

При работе с Simulink пользователь имеет возможность модернизировать библиотечные блоки, создавать свои собственные, а также составлять новые библиотеки блоков.

При моделировании пользователь может выбирать метод решения дифференциальных уравнений, а также способ изменения модельного времени (с фиксированным или переменным шагом). В ходе моделирования имеется возможность следить за процессами, происходящими в системе. Для этого используются специальные устройства наблюдения, входящие в состав библиотеки Simulink. Результаты моделирования могут быть представлены в виде графиков или таблиц.

3 Microsoft Visual Studio 2017 объектно-ориентированный язык программирования C#

Microsoft Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

IntelliSense – это общий термин для набора очень популярных функций, отображающих сведения о типах в коде непосредственно в редакторе и в некоторых случаях автоматически создающих небольшие отрывки кода. По сути, IntelliSense представляет собой базовую документацию, встроенную в редактор, что избавляет от необходимости поиска информации о типах в отдельном окне справки.Функции

IntelliSense зависят от языка. Дополнительные сведения см. в руководствах по IntelliSense для C# , IntelliSense для Visual C++, IntelliSense для JavaScript и IntelliSense для Visual Basic.

Обобщенный визуальный обзор Visual Studio приведен на рисунке 3.1. На нем показана среда Visual Studio с открытым проектом и несколькими окнами основных инструментов, которые вам, вероятнее всего, придется использовать.

Обозреватель решений позволяет просматривать файлы кода, а также перемещаться по ним и управлять ими. Обозреватель решений позволяет упорядочить код путем объединения файлов в решения и проекты.

В окне Редактор, в котором вы, вероятнее всего, будете проводить большую часть времени, приводится исходный код. В нем можно редактировать этот код и разрабатывать пользовательский интерфейс.

В окно Вывод Visual Studio отправляет уведомления, такие как сообщения об отладке и ошибках, предупреждения компилятора, сообщения о состоянии публикаций и многие другие. Каждый источник сообщений имеет собственную вкладку.

Team Explorer (VSTS) позволяет отслеживать рабочие элементы и использовать код совместно с другими пользователями с помощью технологий управления версиями, таких как Git и система управления версиями Team Foundation (TFVC).

Cloud Explorer позволяет просматривать ресурсы Azure, например виртуальные машины, таблицы, базы данных SQL и т. д., и управлять ими. Если для выполнения конкретной операции требуется портал Azure, Cloud Explorer предоставит ссылки для перехода в нужное место на портале Azure.

Одним из компонентов Microsoft Visual Studio является Visual C#. Visual C# - это реализация языка C# корпорацией Майкрософт. Поддержка Visual C# в Visual Studio обеспечивается с помощью полнофункционального редактора кода, компилятора, шаблонов проектов, конструкторов, мастеров кода, мощного и удобного отладчика и многих других средств. Библиотека классов .NET Framework предоставляет доступ ко многим службам операционной системы и другим

полезным, правильным классам, что существенно ускоряет цикл разработки.

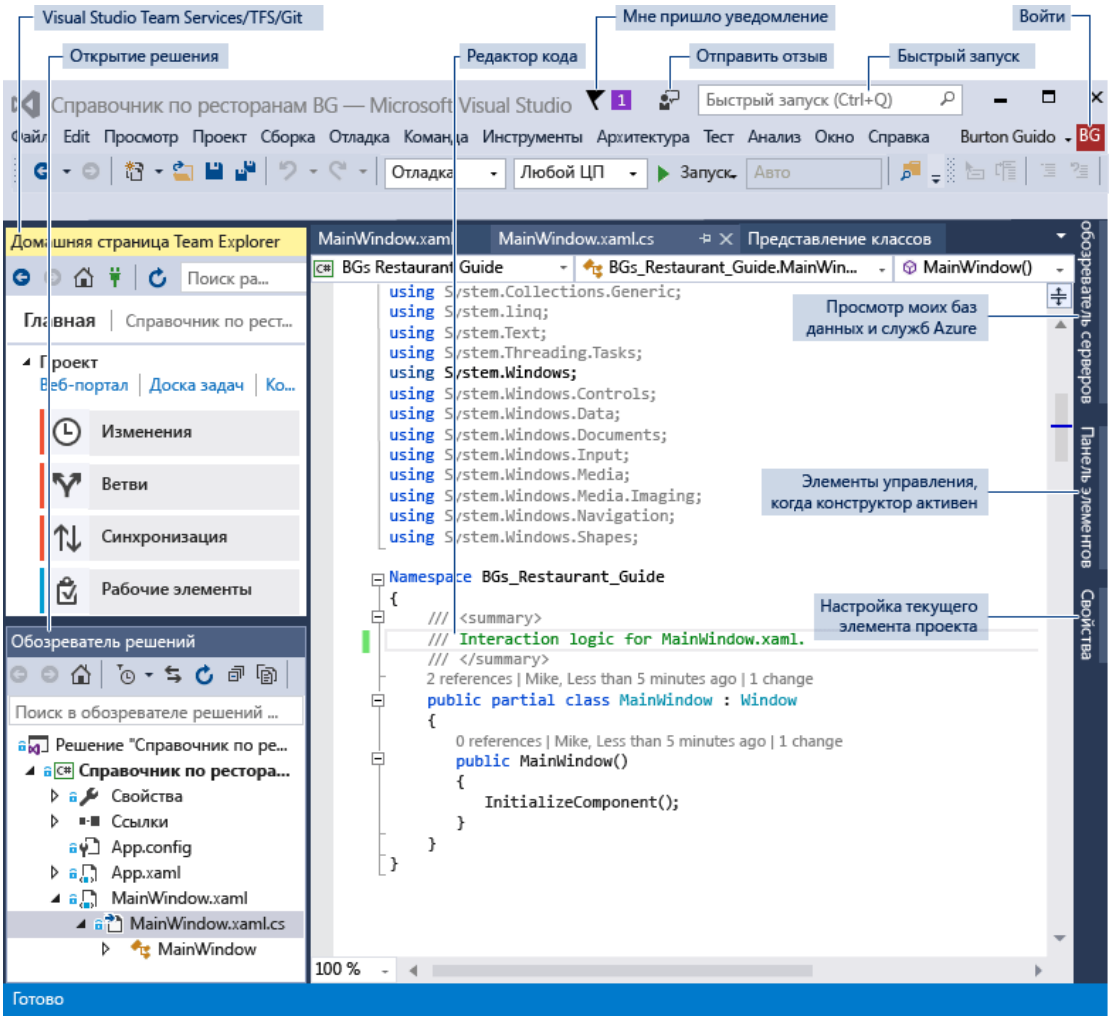

Рисунок 3.1 – Обобщенный визуальный обзор Visual Studio

C# (произносится “Си-шарп”) – это язык программирования, предназначенный для разработки самых разнообразных приложений, предназначенных для выполнения в среде .NET Framework. Язык C# прост, типобезопасен и объектно-ориентирован. Благодаря множеству нововведений C# обеспечивает возможность быстрой разработки приложений, но при этом сохраняет выразительность и элегантность, присущую языкам C [24].

Интегрированная среда разработки Visual C# представляет собой набор средств разработки, предоставляемых через единый пользовательский интерфейс. Некоторые средства используются совместно с другими языками Visual Studio, в то время как другие, например, компилятор C#, свойственны только Visual C#.

4 Практическая работа №1. Основы программирования на Python

Цель работы: изучить основы программирования на языке Python.

Базовые операции в среде программирования Python.

1. Запуск оболочки программ и инструкции языка Python.
2. Встроенные типы данных.
3. Выражения.
4. Функции.
5. Встроенные функции.
6. Классы.
7. Исключения.
8. Функции преобразования типов и классы.
9. Числовые и строковые функции.
10. Функции обработки данных.
11. Функции определения свойств.
12. Функции для доступа к внутренним структурам.
13. Функции компиляции и исполнения.
14. Функции ввода-вывода.
15. Ввод и вывод файлов.
16. Стандартные файлы ввода/вывода данных, и вывода ошибок.
17. Функции для работы с атрибутами.
18. Модули.
21. Модули стандартной библиотеки.
20. Функции как параметры и результат.

Исходные данные задает преподаватель, который указывает условия задачи.

Примеры задач:

Задача 1. На входе функция to_set() получает строку или список чисел. Преобразуйте их в множество. На выходе должно получиться множество и его мощность.

Задача 2. Имеется список с произвольными данными. Поставлена задача

преобразовать его в множество. Если какие-то элементы нельзя хешировать, то пропускаем их. Функция list_to_set() выводит на печать получившееся множество.

Задача 3. На основании 3 исходных множеств (передаются в качестве аргументов функции diff()) требуется написать функцию, которая будет возвращать либо симметричную разность, либо просто разность (если дополнительный аргумент функции symmetric имеет значение False) приведенных объектов в порядке: 1-ое множество, 2-ое множество, 3-е множество.

Задача 4. Напишите функцию superset(), которая принимает 2 множества. Результат работы функции: вывод в консоль одного из сообщений в зависимости от ситуации:

1 - «Супермножество не обнаружено»;

2 – «Объект {X} является чистым супермножеством»;

3 – «Множества равны».

Задача 5. Предоставлен список натуральных чисел. Требуется сформировать из них множество. Если какое-либо число повторяется, то преобразовать его в строку по образцу: например, если число 4 повторяется 3 раза, то в множестве будет следующая запись: само число 4, строка «44» (второе повторение, т.е. число дублируется в строке), строка «444» (третье повторение, т.е. строка множится на 3). Реализуйте вывод множества через функцию set_gen().

Пример содержания отчета:

1 Исходные данные.

2. Решение задачи (код программы)

Выводы.

5 Практическая работа №2. Моделирование нейронных сетей в Matlab

Цель работы: Освоение технологии решения задач с использованием нейронных сетей в пакетах расширения Neural Networks Toolbox и Simulink.

Знакомство со средствами и методами MATLAB и пакета Simulink для моделирования и исследования нейронных сетей. Применение нейронных сетей для аппроксимации функций.

Примеры заданий.

Задание №1.

Создать нейронную сеть, реализующую функциональную зависимость $Y = \frac{a}{b + x^c}$, между входом (x) и выходом (Y). Значения коэффициентов в этом случае заданы. Последовательность решения должна включать все 5 этапов.

1-этап: «Подготовка данных для обучения сети»;

2 этап: «Создание сети»;

3 этап: «Обучение сети»;

4 этап: «Тестирование сети»;

5 этап: «Моделирование сети».

Задание №2.

Сформируйте модель НС, рассмотренную в Задании №1 в Simulink, используя функцию gensim. Обратите внимание на структуру НС в S –модели.

Задание №3.

Выполнить генерацию исходных данных в виде точек на плоскости, разделённой прямой линией X2 = 0.4+1.5·X1. Точки, попавшие по одну сторону линии, будут относиться к классу 1, по другую – к классу 2. Создать нейронную сеть, способную отнести любую новую точку к соответствующему классу.

6 Практическая работа №3. Моделирование нечеткой системы средствами fuzzy logic toolbox системы Matlab

Цель работы: изучение основных функций пакета Fuzzy Logic Toolbox программной среды MATLAB, а также приобретение навыков построения нечеткой аппроксимирующей системы.

Порядок выполнения работы.

Командой (функцией) Fuzzy из режима командной строки запус- кается основная интерфейсная программа пакета Fuzzy Logic – редак- тор нечеткой системы вывода (Fuzzy Inference System Editor, FIS Edi- tor, FIS-редактор).

Вид открывающегося при этом окна приведен на рисунке 6.1.

Главное меню редактора содержит позиции:

File – работа с файлами моделей (их создание, сохранение, считывание и печать);

Edit – операции редактирования (добавление и исключение входных и выходных переменных);

View – переход к дополнительному инструментарию.

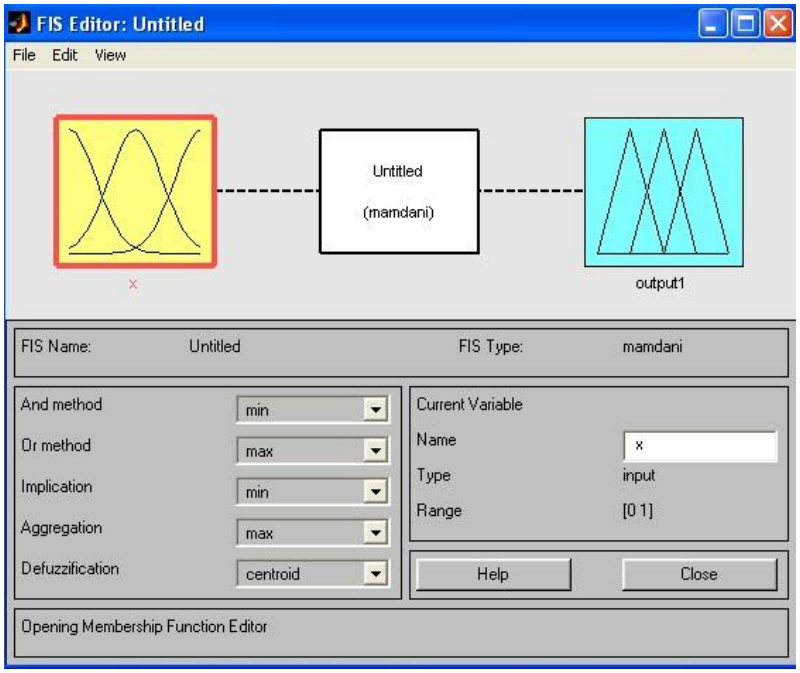

Рисунок 6.1 – Вид окна Fis Editor

Попробуем сконструировать нечеткую систему, отображающую зависимость между переменными x и y, заданную с помощью табл. 1 (легко видеть, что представленные в таблице данные отражают зависимость $y = x^2$).

Таблица 1 – Исходные данные

x	-1	0.6	0	0.4	1
y	1	0.36	0	0.16	1

Требуемые действия отобразим следующими пунктами.

1. В позиции меню File выбираем опцию New Sugeno FIS (новая система типа Sugeno), при этом в блоке, отображаемом белым квадратом, в верхней части окна редактора появится надпись Untitled2 (Sugeno).

2. Щелкнем левой кнопкой мыши по блоку, озаглавленному input l (вход 1). Затем в правой части редактора в поле, озаглавленном Name (Имя), вместо input l введем обозначение нашего аргумента, т.е. x. Обратим внимание, что если теперь сделать где-нибудь (вне блоков редактора) однократный щелчок мыши, то имя отмеченного блока изменится на x; то же достигается нажатием после ввода клавиши Enter.

3. Дважды щелкнем по этому блоку. Перед нами откроется окно редактора функций принадлежности – Membership Function Editor (см. рис. 6.2). Войдем в позицию меню Edit данного редактора и вы- берем в нем опцию Add MFs (Add Membership Funcions – Добавить функций принадлежности). При этом появится диалоговое окно (рис. 6.3), позволяющее задать тип (MF type) и количество (Number of MFs) функций принадлежности (в данном случае все относится к входному сигналу, т. е. к переменной x). Выберем гауссовы функции принадлежности (gaussmf), а их количество зададим равным пяти – по числу значений аргумента в таблице 1. Подтвердим ввод информации нажатием кнопки ОК, после чего произойдет возврат к окну редактора функций принадлежности.

4. В поле Range (Диапазон) установим диапазон изменения x от -1 до +1, т.е. диапазон, соответствующий таблица 1. Щелкнем затем левой кнопкой мыши где-нибудь в поле редактора (или нажмем клавишу ввода Enter). Обратим внимание, что после этого произойдет соответствующее изменение диапазона в поле Display Range

(Диапазон дисплея).

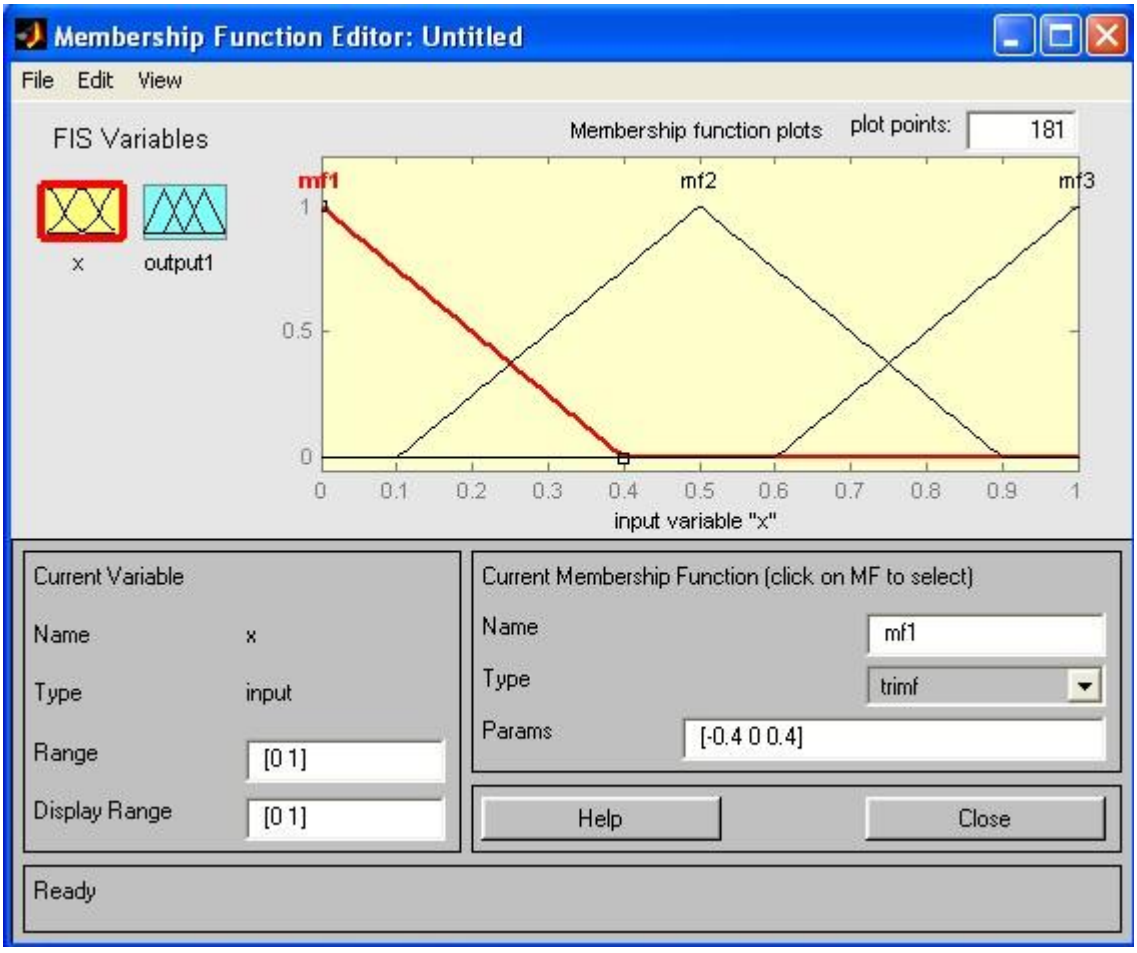

Рисунок 6.2 – Окно редактора функций принадлежности

5. Обратимся к графикам заданных нами функций принадлежности, изображенным в верхней части окна редактора функций принадлежности. Заметим, что для успешного решения поставленной задачи необходимо, чтобы ординаты максимумов этих функций совпадали с заданными значениями аргумента x. Для левой, центральной и правой функций такое условие выполнено, но две другие необходимо «подвинуть» вдоль оси абсцисс. «Передвижка» делается весьма просто: подводим курсор к нужной кривой и щелкаем левой кнопкой мыши. Кривая выбирается, окрашиваясь в красный цвет, после чего с помощью курсора ее и можно подвинуть в нужную сторону (более точную установку можно провести, из- меняя числовые значения в поле Params (Параметры) – в данном случае каждой функции принадлежности соответствуют два параметра, при этом первый определяет размах

кривой, а второй – положение ее центра). Для выбранной кривой, кроме этого, в поле Name можно изменять имя (завершая ввод каждого имени нажатием клавиши Enter).

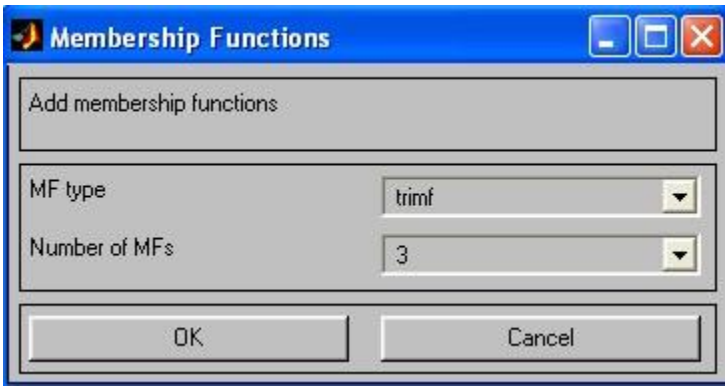

Рисунок 6.3 – Диалоговое окно задания типа и количества функций принадлежности

Проделаем требуемые перемещения кривых и зададим всем пяти кривым новые имена, например:

- самой левой – bn,
- следующей – n,
- центральной – z,
- следующей за ней справа – p,
- самой левой – bp.

Нажмем кнопку Close и выйдем из редактора функций принадлежности, возвратившись при этом в окно редактора нечеткой системы (FIS Editor).

6. Сделаем однократный щелчок левой кнопкой мыши по голубому квадрату (блоку), озаглавленному output l (выход 1). В окошке Name заменим имя output l на y (как в пункте 2).

7. Дважды щелкнем по отмеченному блоку и перейдем к программе – редактору функций принадлежности. В позиции меню Edit выберем опцию Add MFs. Появляющееся диалоговое окно вида рис. 3 позволяет задать теперь в качестве функций принадлежности только линейные (li-near) или постоянные (constant) – в зависимости от того, какой алгоритм Sugeno (1-го или 0-го порядка) мы выбираем. Если в вашем компьютере установлена версия, в которой нет данных функций принадлежностей, то можно оставить по умолчанию – trimf. Это, конечно, повлияет

на результат, поэтому можно поэкспериментировать, изменяя тип функций принадлежности.

В рассматриваемой задаче необходимо выбрать постоянные функции принадлежности с общим числом 4 (по числу различных значений y). Подтвердим введенные данные нажатием кнопки ОК, после чего произойдет возврат в окно редактора функций принадлежности.

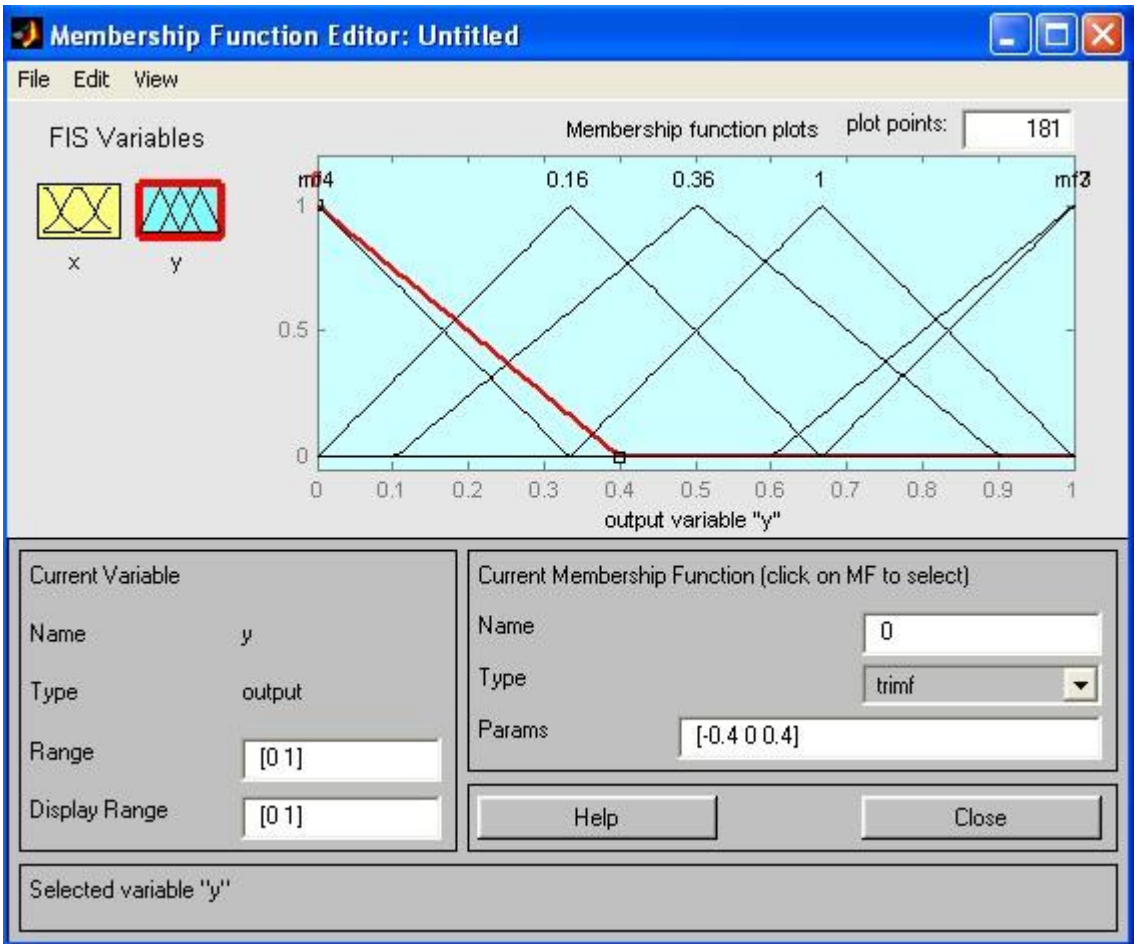

Рисунок 6.4 – Параметры функций принадлежности переменной *y*

8. Обратим внимание, что здесь диапазон (Range) изменения, устанавливаемый по умолчанию – [0, 1], менять не нужно. Изменим лишь имена функций принадлежности (их графики при использовании алгоритма Sugeno для выходных переменных не приводятся), на- пример, задав их как соответствующие числовые значения y, т.е. 0, 0.16, 0.36, 1; одновременно эти же числовые значение введем в поле Params (рис. 6.4). Затем закроем окно нажатием кнопки Close и

вернемся в окно FIS-редактора.

9. Дважды щелкнем левой кнопкой мыши по среднему (белому) блоку, при этом раскроется окно еще одной программы – редактора правил (Rule Editor). Введем соответствующие правила. При вводе каждого правила необходимо обозначить соответствие между каждой функцией принадлежности аргумента x и числовым значением y. Кривая, обозначенная нами bn, соответствует x = -1, т.е. y = 1. Выберем, поэтому в левом поле (с заголовком x is bn), а в правом 1 и нажмем кнопку Add rule (Добавить правило). Введенное правило появится в окне правил и будет представлять собой запись: If (x is bn) then (y is 1)

Аналогично поступим для всех других значений x, в результате чего сформируется набор из 5 правил (см. рис. 6.5). Закроем окно редактора правил и возвратимся в окно FIS-редактора. Построение системы закончено и можно начать эксперименты по ее исследованию. Заметим, что большинство опций выбиралось нами по умолчанию.

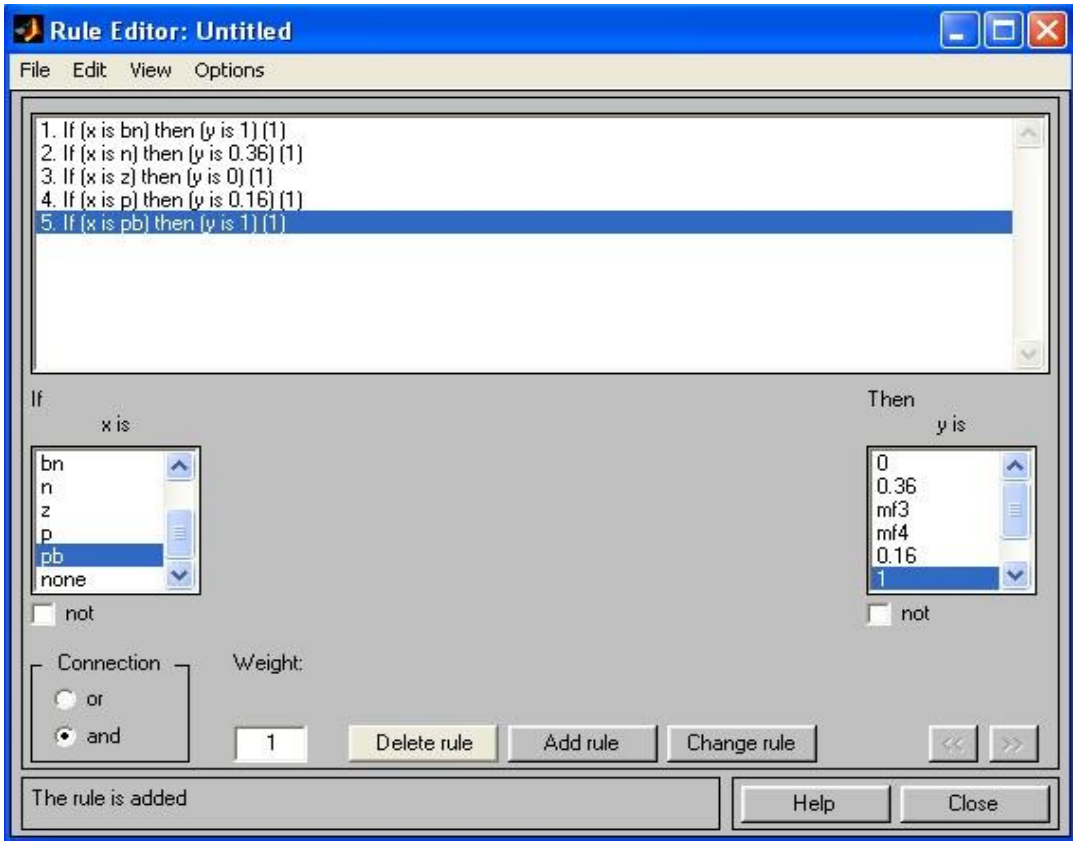

Рисунок 6.5 – Окно редактора правил

10. Предварительно сохраним на диске (используя пункты меню File/Save to disk as...) созданную систему под каким-либо именем, например, Proba.

11. Выберем позицию меню View. Как видно из выпадающего при этом подменю, с помощью пунктов Edit membership functions и Edit rules можно совершить переход к двум выше рассмотренным программам – редакторам функций принадлежности и правил (то же можно сделать и нажатием клавиш Ctrl+2 или Ctrl+3). Но сейчас нас будут интересовать два других пункта – View rules (Просмотр правил) и View surface (Просмотр поверхности). Выберем пункт View rules, при этом откроется окно (см. рис. 6.6) еще одной программы – просмотра правил (Rule Viewer).

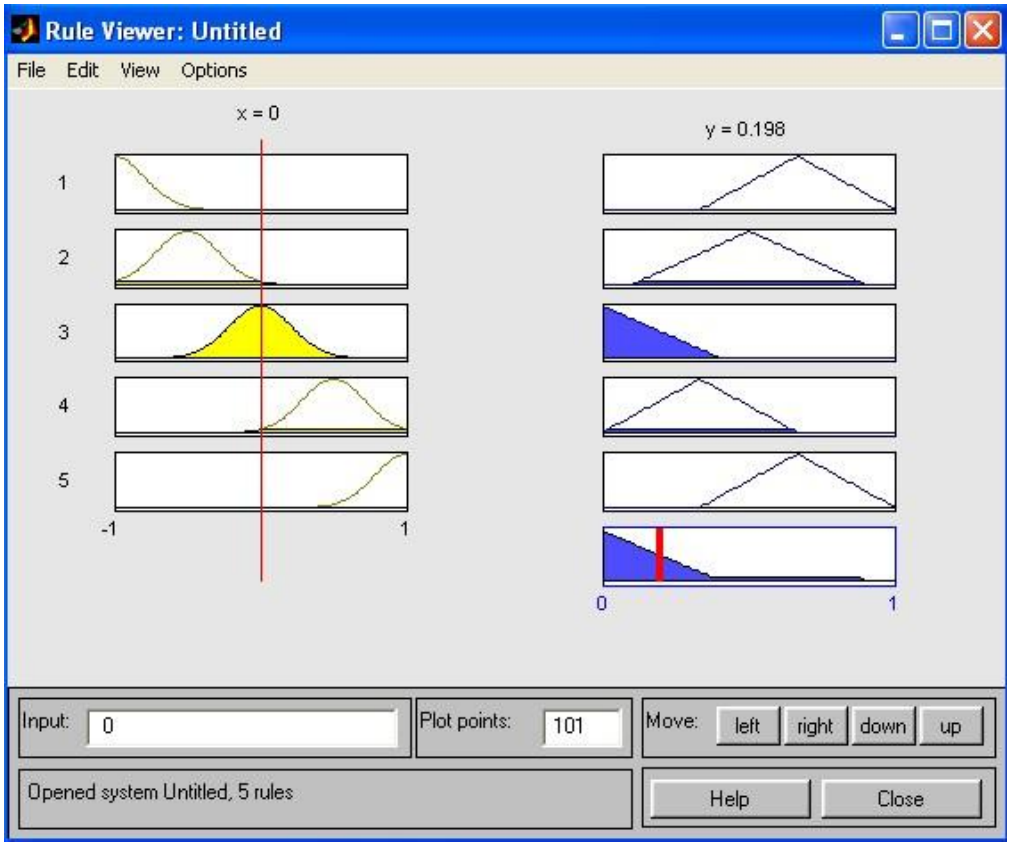

Рисунок 6.6 – Окно просмотра правил

12. В правой части окна в графической форме представлены функции принадлежности аргумента ж, в левой – переменной выхода у с пояснением механизма принятия решения. Красная вертикальная черта, пересекающая графики в правой части окна, которую можно перемещать с помощью курсора, позволяет

изменять значения переменной входа (это же можно делать, задавая числовые значения в поле Input (Вход)), при этом соответственно изменяются значения y в правой верхней части окна. Зададим, например, x – 0.5 в поле Input и нажмем затем клавишу ввода (Enter). Значение y сразу изменится и станет равным 0.202. Таким образом, с помощью построенной модели и окна просмотра правил можно решать задачу интерполяции, т.е. задачу, решение которой и требовалось найти. Изменение аргумента путем перемещения красной вертикальной линии очень наглядно демонстрирует, как система определяет значения выхода.

13. Закроем окно просмотра правил и выбором пункта меню View/View surface перейдем к окну просмотра поверхности отклика (выхода), в нашем случае – к просмотру кривой y(x) (см. рис. 6.7). Видно, что смоделированное системой по таблице данных (табл. 2) отображение не очень-то напоминает функцию x^2. Ну что ж, ничего удивительного в этом нет: число экспериментальных точек невелико, да и параметры функций принадлежности (для x) выбраны, скорее всего, неоптимальным образом.

Ниже мы рассмотрим возможность улучшения качества подобной модели.

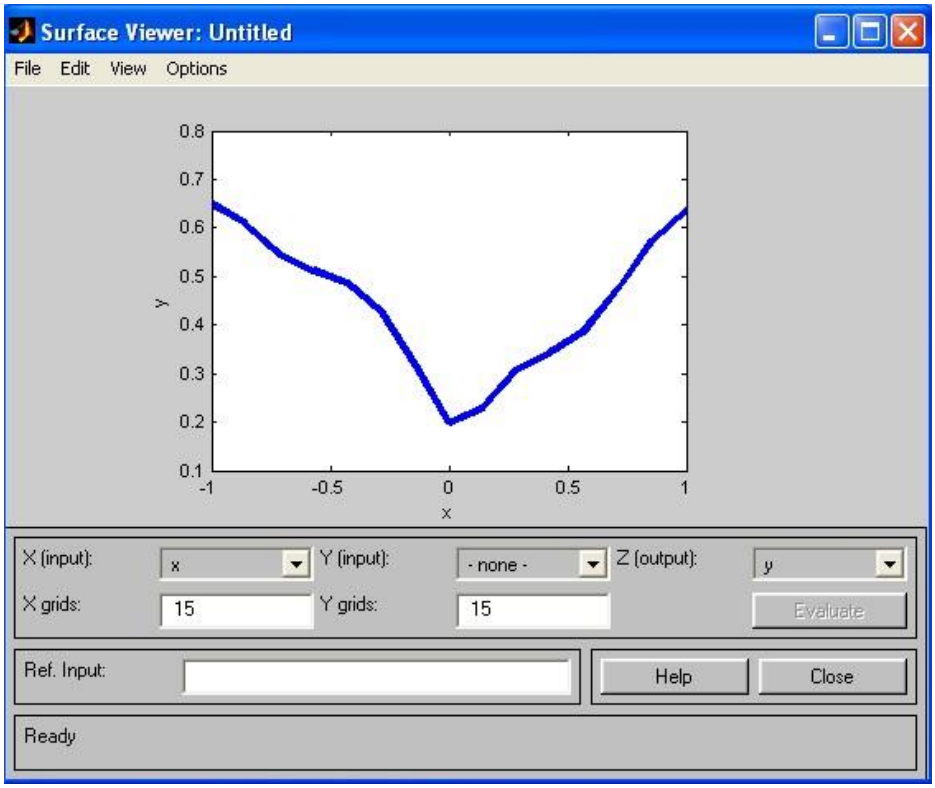

Рисунок 6.7 – Окно просмотра поверхности отклика

В заключение рассмотрения примера отметим, что с помощью вышеуказанных программ - редакторов на любом этапе проектирования нечеткой модели в нее можно внести необходимые коррективы, вплоть до задания какой-либо особенной пользовательской функции принадлежности. Из опций, устанавливаемых в FIS-редакторе по умолчанию при использовании алгоритма Sugeno, можно отметить:

- логический вывод организуется с помощью операции умножения (prod);
- композиция – с помощью операции логической суммы (вероятностного ИЛИ, probor);
- приведение к четкости – дискретным вариантом центроидного метода (взвешенным средним, wtaver). Используя соответствующие поля в левой нижней части окна FIS-редактора, данные опции можно, при желании, изменить.

Задание для студентов.

Сконструируйте нечеткую систему, отображающую зависимость между переменными x и y, заданную с помощью таблице 2. По результатам работы определить тип кривой.

Таблица 2 – Варианты заданий

Варианты	Значение аргумента и функции					
1	x	-1	-0.5	0	0.2	1
	y	1	0.25	0	0.4	1
2	x	-1	-0.6	0.2	0.4	1
	y	-1	-1.67	5	2.5	1
3	x	-1	-0.5	0	0.3	1
	y	-1	-0.13	0	0.27	1
4	x	-1	-0.6	0	0.3	1
	y	0	0.8	1	0.95	0
5	x	-1	-0.5	0	0.2	1
	y	1	-0.125	0	0.008	1
6	x	-1	-0.6	0.2	0.4	1
	y	0	-0.64	-0.96	-0.84	0
7	x	-1	-0.5	0	0.3	1
	y	-3	-2	-1	-0.4	1

8	x	-1	-0.6	0	0.3	1
	y	0.5	0.09	0	0.0225	0.5
9	x	-1	-0.5	0	0.2	1
	y	0.5	0.03125	0	0.0008	0.5
10	x	-1	-0.6	0.2	0.4	1
	y	-1	2.78	25	6.25	1
11	x	-1	-0.5	0	0.3	1
	y	-1	1.8	3	3.6	5
12	x	-1	-0.6	0	0.3	1
	y	-1	-0.216	0	0.027	1
13	x	-1	-0.5	0	0.2	1
	y	-2	-1.5	-1	-0.8	0
14	x	-1	-0.6	0.2	0.4	1
	y	-2	-1.2	0.4	2.5	1
15	x	-1	-0.5	0	0.3	1
	y	0	0.25	0.5	0.65	1
16	x	-1	-0.5	0	0.3	1
	y	-1	-1.75	-1	-0.31	0

Содержание отчета.

1 Цель работы.

2 Задание.

3 Краткое описание действий.

4 Графики по всем пунктам программы.

5 Выводы по работе.

7 Практическая работа №4. Машинное обучение в языке программирования Python

Простое объяснение того, как работают нейронные сети, а также показаны способы их реализации в Python.

Машинное обучение – технология, которая строит саму себя. Это новое явление в нашем мире. Во второй половине XX века машинное обучение развилось в подобласть искусственного интеллекта, которая охватывала разработку самообучающихся алгоритмов. Эти алгоритмы разрабатываются с целью обработки данных и выполнения различных прогнозов более простыми способами чем построение модели в ручном режиме.

Полезнее рассматривать машинное обучение как средство созданий моделей данных. Создается математическая модель для исследования данных с последующим постепенным улучшение качества прогнозных моделей и принятие решений, управляемых данными.

Задачи «обучения» начинаются с появлением у этих моделей настраиваемых параметров, которые можно приспособить для отражения наблюдаемых данных, таким образом, программа обучается на данных.

Наиболее успешные алгоритмы машинного обучения являются те, которые автоматизируют процессы принятия решений путем обобщения известных примеров. В этих методах, пользователь предоставляет алгоритму пары объект-ответ, а алгоритм находит способ получения ответа по объекту. В частности, алгоритм способен выдать ответ для объекта, которого он никогда не видел раньше, без какой-либо помощи.

Каждый год в мире появляются сотни новых алгоритмов с обучением, но все они основаны на небольшом наборе фундаментальных идей.

Существует три типа машинного обучения: обучение с учителем (контролируемое), обучение без учителя (неконтролируемое, или спонтанное) и обучение с подкреплением.

Машинное обучение с учителем.

Основная задача обучения с учителем (supervised learning) состоит в том, чтобы на маркированных тренировочных данных извлечь модель, которая позволяет делать прогнозы о ранее не встречавшихся или будущих данных. Здесь термин «с учителем» относится к подмножеству образцов, в которых нужные выходные сигналы (метки) уже известны. Оно разделяется далее на задачи классификации и задачи регрессии.

Задача классификации – это подкатегория методов машинного обучения с учителем, суть которой заключается в идентификации категориальных меток классов для новых экземпляров на основе предыдущих наблюдений1•

Метка класса представляет собой дискретное, неупорядоченное значение, которое может пониматься как принадлежность группе экземпляров. Извлеченная алгоритмом обучения с учителем прогнозная модель может присваивать новому, немаркированному экземпляру любую метку класса, которая была опреде-лена в тренировочном наборе данных.

Задача регрессии – это предсказание значений непрерывной целевой переменной (регрессионный анализ). Имеется несколько предикторных (объясняющих) переменных и непрерывная (результирующая) переменная отклика, и алгоритм пытается найти между этими переменными связь, которая позволит предсказывать результат.

Машинное обучение без учителя.

Обучение без учителя (unsupervised learning) включает моделирование признаков набора данных без каких-либо меток. Эти модели включают такие за дачи, как кластеризация (clustering) и понижение размерности (dimensionality reduction).

Кластеризация – это метод разведочного анализа данных, который позволяет организовать груду информации в содержательные подгруппы (кластеры), не имея никаких предварительных сведений о принадлежности группе. Каждый кластер, который может появиться во время анализа, обозначает группу объектов, которые обладают определенной степенью подобия и одновременно больше отличаются от объектов в других кластерах, поэтому кластеризацию также иногда называют

«классификацией без учителя».

Еще одна подобласть обучения без учителя представлена методом снижения размерности. При этом методе метки или другая информация определяются исходя из структуры самого набора данных. Смысл задачи понижения размерности состоит в том, чтобы отбросить нерелевантные или избыточные признаки. Задача разбивается на две группы, отбрасывание признаков, выделение признаков.

Кроме того, существуют так называемые методы частичного обучения (semi-supervised learning), располагающиеся примерно посередине между машинным обучением с учителем и машинным обучением без учителя. Методы частичного обучения бывают полезны в случае наличия лишь неполных меток.

Для выполнения задач программирования машинного обучения следует обращаться к библиотеке scikit-learn, которая на сегодня является одной из самых популярных и доступных библиотек машинного обучения с открытым исходным кодом.

Библиотека NumPy – это один из основных пакетов для научных вычислений в Python. Он содержит функциональные возможности для работы с многомерными массивами, высокоуровневыми математическими функциями (операции линейной алгебры, преобразование Фурье, генератор псевдослучайных чисел). В scikit-learn массив NumPy – это основная структура данных. scikit-learn принимает данные в виде массивов NumPy.

Pandas – более новый пакет, надстройка над библиотекой NumPy, обеспечивающий реализацию класса DataFrame. Объекты DataFrame – многомерные массивы с метками для строк и столбцов, а также зачастую с неоднородным типом данных и/или пропущенными данными.

Библиотека scikit-learn это проект с открытым исходным кодом, это означает, что его можно свободно использовать и распространять, и любой человек может легко получить исходный код. Проект scikit-learn постоянно развивается и совершенствуется. Он содержит ряд современных алгоритмов машинного обучения, документацию по каждому алгоритму.

Лучше всего представлять используемые в библиотеке Scikit-Learn данные в

виде таблиц. Простейшая таблица – двумерная сетка данных, в которой строки представляют отдельные элементы набора данных, а столбцы некоторые атрибуты, связанные с каждым из этих элементов. Примером может служить рассматриваемый в данной работе набор данных Iris проанализированный Рональдом Фишером в 1936 году.

Библиотека SciPy – это набор функций для научных вычислений в Python. Она содержит процедуры линейной алгебры, математическую оптимизацию функций, обработку сигналов, специальные математические функции и статистические функции. Scikit-learn использует набор функций SciPy для реализации своих алгоритмов.

Matplotlib – это основная библиотека для построения научных графиков в Python. Она включает функции для создания высококачественных визуализаций типа линейных диаграмм, гистограмм, диаграмм разброса.

Pandas – библиотека Python для обработки и анализа данных. Она построена на основе структуры данных. DataFrame библиотеки pandas представляет собой таблицу, похожую на электронную таблицу Excel. В отличие от NumPy, который требует, чтобы все записи в массиве были одного и того же типа, в pandas каждый столбец может иметь отдельный тип (например, целые числа, даты, числа с плавающей точкой и строки).

Anaconda дистрибутив Python, предназначенный для крупномасштабной обработки данных, прогнозной аналитики и научных вычислений. Anaconda уже включает NumPy, SciPy, matplotlib, pandas, IPython, Jupyter Notebook и scikit-learn.

Необходимо изучить библиотеки scikit-learn машинного обучения и сделать обзор.

Исходные данные задает преподаватель, который указывает условия задачи.

Содержание отчета.

1 Цель работы.

2 Задание.

3 Краткое описание действий.

4 Выводы по работе.

8 Практическая работа №5. Однослойный персептрон на Python

Реализация однослойного персептрона. Персептрон используется для реализации задачи классификации, а однослойный персептрон может реализовывать только линейную классификацию.

Реализация персептрона на Python.

Работу с алгоритмами машинного обучения начнем реализации персептрона Розенблатта. Применяя объектно-ориентированный подход, определим интерфейс персептрона как класс языка Python.

```
In [5]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        from matplotlib.colors import ListedColormap


        class Perceptron(object):
            #Класификатор на основе персептона
            |
            def __init__(self, eta=0.01, n_iter=10):
                self.eta = eta
                self.n_iter = n_iter

         #Выполняет подгонку модели под тренеровочные данные
            def fit(self, X, y):

                self.w_ = np.zeros(1 + X.shape[1])
                self.errors_ = []

                # повторить n_iter раз
                for _ in range(self.n_iter):
                    errors = 0
                    for xi, target in zip(X, y):
                        update = self.eta * (target - self.predict(xi))
                        self.w_[1:] += update * xi
                        self.w_[0] += update
                        errors += int(update != 0.0)
                    self.errors_.append(errors)
                return self

            def net_input(self, X):
                #Расчитать чистый вход
                return np.dot(X, self.w_[1:]) + self.w_[0]

            def predict(self, X):
               #Вернуть метку класса после единичного скачка
                return np.where(self.net_input(X) >= 0.0, 1, -1)
```

Рисунок 8.1 – Реализация персептрона

Новые объекты персептрона смогут обучаться на данных при помощи метода

fit (выполнить подгонку модели) и делать прогнозы при помощи метода predict (распознать). Задается тем обучения eta и число эпох n_iter (проходов), по тренировочному набору. При помощи метода для выполнения подгонки fit инициализируются веса в атрибуте self.w_ нулевым вектором R^{m+t}, где m число признаков в наборе данных, в нулевой позиции которого учитываются веса (порог).

После инициализации весов метод fit в цикле просматривает отдельные образцы тренировочного набора и обновляет веса согласно правилу обучения персептрона. Метки классов распознаются методом predict, который также вызывается в методе fit. Метод fit может применяться и для распознавания меток классов новых данных уже после подгонки модели. Кроме того, в списке self.errors_ собираем число ошибочных распознаваний классов в каждой эпохе, для анализа алгоритма.

Тренировка персептронной модели.

На наборе данных Ирисы Фишера реализуем классифицирующий алгоритм. Этот набор данных содержит данные измерений 150 цветков ириса трех видов: ирис щетинистый (lris setosa), ирис виргинский (Iris virginica) и ирис разноцветный (Iris versicolor) по четырем характеристикам (признакам): длина чашелистика, ширина чашелистика, длина лепестка, ширина лепестка. Согласно рисунку 3 можно увидеть зависимость пар данных.

Реализация модели будет строится на описанного в библиотеке scikit-learn персептона, аналогичного описанному ранее.

Выясним значение математической зависимости между признаками. Наглядно ее можно показать с помощью тепловой карты зависимости признаков. Данный метод выдает Значения коэффициента корреляции. :Согласно рисунку 3 наиболее высокая зависимость между переменными «ширина лепестка» и «длина лепестка» 0.96.

Следовательно, матрицу признаков X составят третий и четвертый столбцы, а вектору y метки классов, которые соответствуют видам цветков (0, 1, 2).

Для оценки алгоритма произвольным образом с помощью метода train_test_split из модуля model_selection библиотеки scikit-learn разделим массивы X

и у на тестовые данные в размере 30% от общего объема (45 образцов) и тренировочные данные в размере 70% (105 образцов).

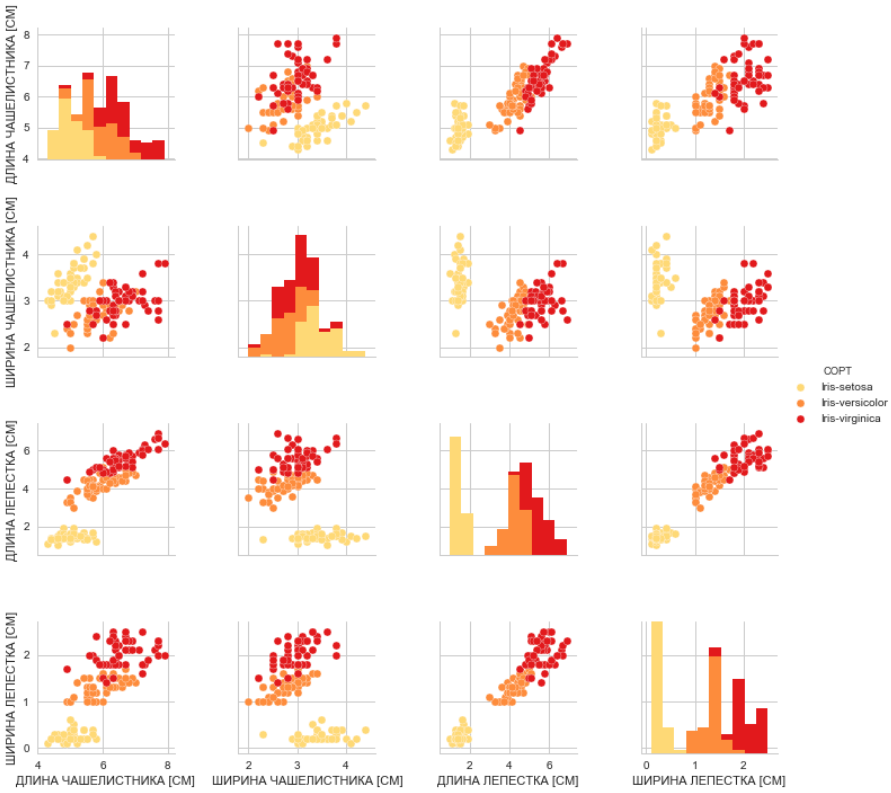

Рисунок 8.2 – Зависимость признаков

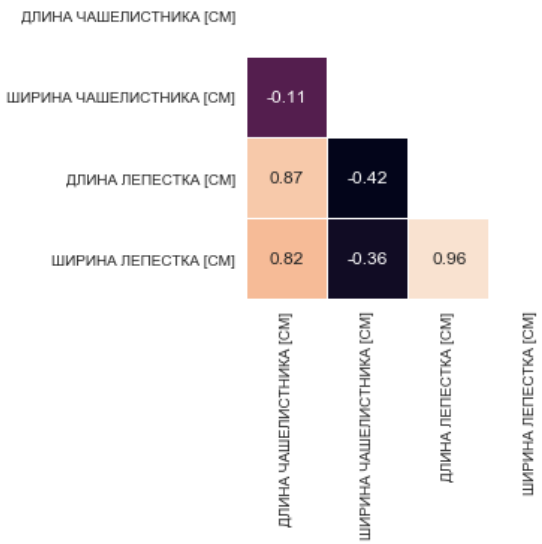

Рисунок 8.3 – Тепловая карта зависимости признаков

Многие алгоритмы машинного обучения и оптимизации в целях улучшения качества также требуют выполнения масштабирования признаков. Выполним стандартизацию дспомощью StandardScaler из модуля preprocessing библиотеки scikit-learn. Этот метод вычисляет параметры μ (эмпирическое среднее) и S (стандартное отклонение) для каждой размерности признаков из тренировочных данных.

Подобрав темп обучения так, что бы модель была достаточно натренированной. После проведения тренеровки видно, что персептрон ошибочно классифицирует 4 из 45 образцов цветков. Ошибка классификации на тестовом наборе данных составляет 0.089.

В библиотеке scikit-learn также реализовано большое разнообразие различных метрик оценки качества работы, которые доступны благодаря модулю metrics. Возможно вычислить оценку верности классификации с использованием персептрона на тестовом наборе.

```
In [45]:
# персептон из библиотеке оналогичен реализованному
ppn = Perceptron(max_iter=40, eta0=0.1, random_state=0)
ppn.fit(X_train_std, y_train)
print('Y набор', y_test.shape)

y_pred = ppn.predict(X_test_std)
print('Число ошибочно классифицируемых образцов: %d' % (y_test != y_pred).sum())
print('Верность: %.2f' % accuracy_score(y_test, y_pred))
```

```
Y набор (45,)
Число ошибочно классифицируемых образцов: 4
Верность: 0.91
```

Рисунок 8.4 – Тренировки модели

Применяя алгоритм к тестовым данным видно, что полностью разделить три класса цветков линейной границей решения не получается, что видно согласно рисунку 8.5.

Минусом алгоритма обучения персептрона является то, что он никогда не сходится на наборах, данных, которые не полностью линейно разделимы.

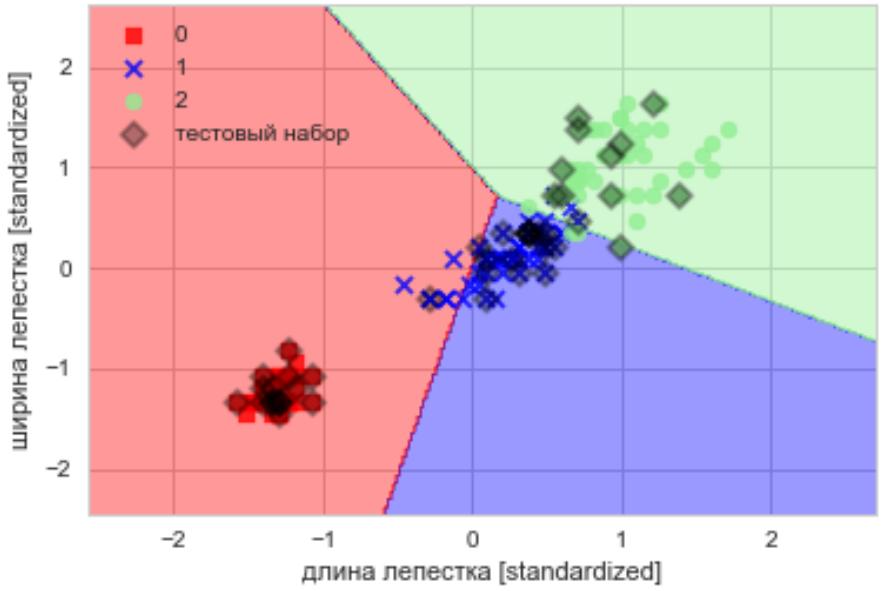

Рисунок 8.5 – График областей решений

Исходные данные задает преподаватель, который указывает условия задачи.

Содержание отчета.

1 Цель работы.

2 Задание.

3 Краткое описание действий.

4 Выводы по работе.

9 Практическая работа №6. Алгоритм случайные леса и К-средних

Изучение алгоритма случайные леса и К-средних в среде программирования Python.

Алгоритм обучения случайные леса (random forests). Он обладает хорошая классификационная способность и масштабируемость. Алгоритм метода:

1) Случайным образом отобрать из тренировочного набора данных n образцов с возвратом;

2) Вырастить дерево решений из выборки. В каждом узле:

а) случайным образом отобрать d признаков без возврата

б) расщепить узел, используя признак, который обеспечивает наилучшее расщепление согласно целевой функции.

в) повторить шаги 1 и 2 k число раз;

г) для назначения метки класса агрегировать прогноз из каждого дерева на основе большинства голосов.

Реализацию классификатора на основе случайного леса RandomForestClassifier в библиотеке scikit-learn, размер выборки выбирается так, чтобы он был эквивалентным числу образцов в исходном тренировочном наборе, что обычно обеспечивает хороший компромисс между смещением и дисперсией. В библиотеке scikit-learn уже существует реализация, которую можно использовать.

При количестве деревьев равное 10, точность алгоритма достаточно велика.

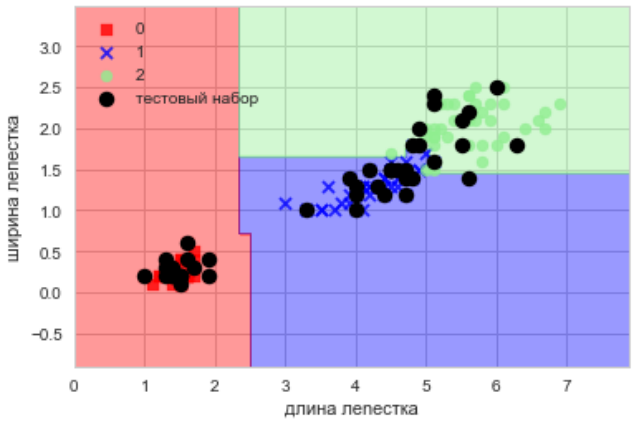

Рисунок 9.1 – График областей решения методом random forests из 10 деревьев

Если увеличить количество деревьев, то точность алгоритма значительно увеличится.

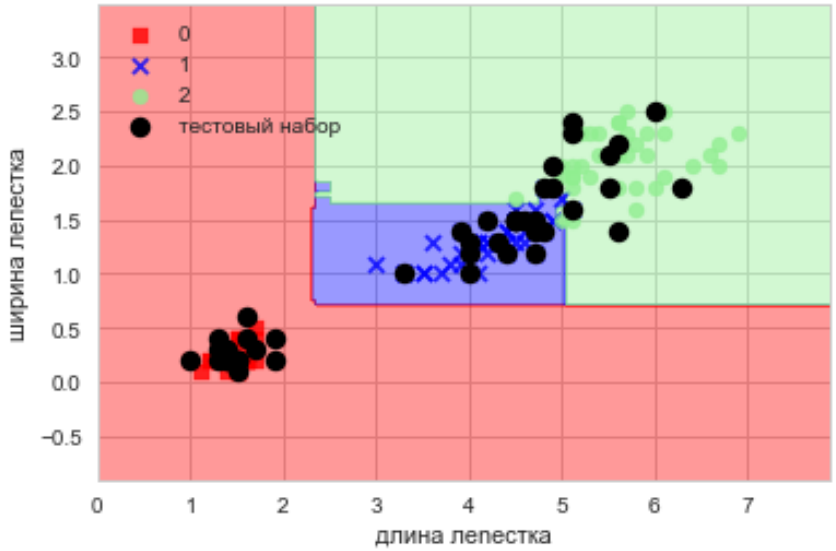

Рисунок 9.2 – График областей решения методом random forests из 50 деревьев

Алгоритм k ближайших соседей это классификатор на основе k ближайших соседей (k-nearest neighbor classifier, KNN), который особенно интересен тем, что он существенно отличается от других алгоритмов обучения.

Алгоритм может быть резюмирован следующими шагами:

1) выбрать число k и метрику расстояния;

2) найти k ближайших соседей образца, который мы хотим классифицировать;

3) присвоить метку класса мажоритарным голосованием.

В тренировочном наборе данных k образцов находятся те, которые являются самыми близкими к классифицируемой точке. Метка класса новой точки данных затем определяется мажоритарным голосованием среди ее k ближайших соседей.

Основное преимущество такого подхода с запоминанием состоит в том, что классификатор немедленно адаптируется по мере сбора новых тренировочных данных. Минусом является вычислительная сложность классифицирования новых образцов. Кроме того, нет возможсти отбросить тренировочные образцы, поскольку нет тренировочного шага.

Правильный выбор числа k крайне важен для нахождения хорошего равновесия между переобучением и недообучением.

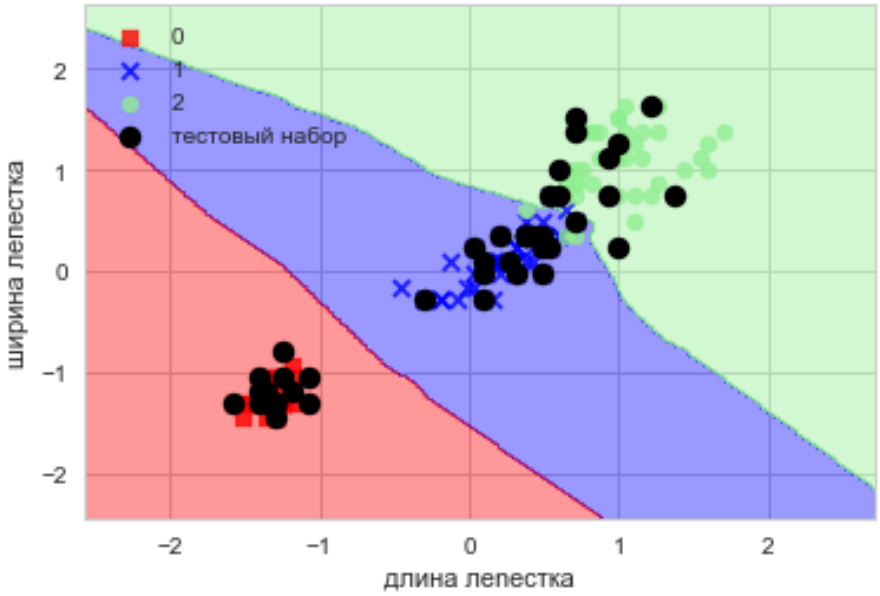

Рисунок 9.3 – График областей решения методом k ближайших соседей

Исходные данные задает преподаватель, который указывает условия задачи.

Содержание отчета.

1 Цель работы.

2 Задание.

3 Краткое описание действий.

4 Выводы по работе.

10 Методические рекомендации по выполнению самостоятельной работы обучающихся

Рекомендации по ведению конспекта лекций

Для ведения конспекта лекций желательно использовать тетрадь большого формата. Поскольку для преподавания дисциплины используется презентация, которая в электронном виде предоставляется студентам, графическую часть лекции целесообразно располагать в соответствующих местах текста (например, вклеивать в конспект). Если какая-то лекция пропускается, лучше сразу переписать ее или, по крайней мере, оставить для нее чистые листы. Хороший конспект может в дальнейшем помочь при изучении других предметов, а также в процессе выполнения магистерской диссертации и в практической деятельности после окончания университета. В тетради следует отвести большие поля (примерно 1/3 ширины листа). Их используют при изучении материала лекции, которую дополняют сведениями из рекомендованной литературы, решением задач и ответами на вопросы, которые преподаватель задает во время лекции. Не оставляйте без внимания непонятные места, найдите ответы сами в литературе или проконсультируйтесь с преподавателем и дополните свой конспект. Перед каждой лекцией необходимо проработать предыдущую, это будет помогать освоить новый материал и закрепить прошлые знания. Очень полезно рассказывать материал заинтересованному слушателю (товарищу по учебе), который не только внимательно выслушает, но и задаст массу вопросов. Такой подход к изучению предмета трудно переоценить, он позволяет сформировать устойчивые знания и развить логическое мышление.

Особенностью практически всех магистерских курсов является небольшое количество лекций, но много практических занятий. Поэтому на них не только более подробно изучается лекционный материал, но и осваивается новый. Соответствующие записи целесообразно продолжать делать в лекционной тетради.

Рекомендации по выполнению и защите реферата

Реферат не имеет жесткой структуры и может включать различные разделы.

После получения темы реферата необходимо отобрать необходимую информацию и проанализировать ее.

Оформление реферата должно полностью соответствовать установленным в УГНТУ требованиям, поэтому перед оформлением результатов своей работы надо внимательно ознакомиться с соответствующей методичкой, которая имеется в библиотеке университета. Требуемый объем реферата – 15-20 страниц.

Поскольку реферат защищается публично на практических занятиях, обязательно надо подготовить небольшую (10-15 слайдов) презентацию, которая впоследствии должна быть распечатана и оформлена как приложение к реферату.

Сделанные в ходе обсуждения реферата замечания по его содержанию должны быть исправлены, после чего реферат сдается лектору.

Дисциплина «Проблемы искусственного интеллекта, распознавание образов»

Лекция 1. Теоретические основы интеллектуальных систем.

Искусственный интеллект, история развития искусственного интеллекта. Основные понятия и современные направления искусственного интеллекта. Формальные языки и формальные системы. Понятие интеллектуальных информационных систем. Стадии разработки экспертных систем. Идентификация проблемы. Концептуализация, как стадия экспертной системы. Экспертные системы. Формализация. Классификация интеллектуальных систем. Классификация по масштабу, по сфере применения. Реализация экспертных систем. Классификация интеллектуальных систем. Классификация по способу организации.

Лекция 2. Методы представления знаний и моделирования рассуждений.

Методы представления знаний. Системы правил для представления знаний. Семантические сети. Системы фреймов. Примеры использования методов представления знаний. Понятие о дедукции, абдукции, индукции, рассуждениях по

аналогии и на основе прецедентов, рассуждениях на основе аргументации. Метод резолюций. Индукция и абдукция. Автоматизация рассуждений на основе аргументации. Рассуждения на основе прецедентов. Рассуждения о пространстве и времени.

Лекция 3. Методы машинного обучения и приобретения знаний интеллектуальными системами.

Проблемы приобретения знаний. Обучение по примерам. Приобретение знаний на основе автоматического анализа текстов. Интерактивные методы приобретения знаний.

Лекция 4. Инструментальные средства и технологические процессы построения интеллектуальных систем.

Архитектура баз знаний интеллектуальных систем. Архитектура машины вывода. Интерфейсы пользователя и приобретения знаний и их архитектуры. Архитектурные особенности интегрированных интеллектуальных систем: интерфейсы с базами данных, пакетами прикладных программ и интеллектуальными системами. Технологии прямого приобретения знаний интеллектуальными системами. Технологии поддержки баз знаний. Технологии проектирования интеллектуальных систем.

Лекция 5. Систематический обзор существующих методов распознавания образов в различных системах.

Классификация на основе байесовской теории решений. Байесовская дискриминантная функция. Принятие решение по максимуму правдоподобия. Ошибки классификации. Оптимальная дискриминантная функция для нормально распределенных образов. Обучение для статистических дискриминантных функций. Непараметрическое оценивание.

Линейный и нелинейный классификаторы. Линейная дискриминантная функция. Алгоритм однослойного перцептрона. Схема Кеслера. Построение оптимальной разделяющей поверхности. Алгоритм Гаусса-Зейделя. Нелинейный классификатор. Многослойный перцептрон.

Методы контекстно-зависимой классификации. Постановка задачи.

Байесовский классификатор. Модель Марковской цепи. Алгоритм Витерби. Скрытые Марковские модели.

Методы селекции признаков. Постановка задачи селекции признаков. Общность классификатора. Предобработка векторов признаков. Селекция на основе проверки статистических гипотез. Векторная селекция признаков. Мера отделимости классов. Оптимальная селекция признаков. Оптимальная селекция на основе нейронной сети.

Лекция 6. Методы генерации признаков. Генерация признаков на основе линейных преобразований. Преобразование Карунена-Лоева. Дискретное преобразование Фурье. Преобразования Адамара и Хаара. Генерация признаков на основе нелинейных преобразований. Признаки, основанные на статистиках первого и второго порядка. Признаки формы и размера. Признаки Фурье. Цепной код.

Методы распознавания образов на основе нейронных сетей. Нейросетевое распознавание образов. Сеть Хопфилда. Сеть Хэмминга. Классификатор Гроссберга. Сети на основе радиально-базисных функций. Обучение без учителя в нейросетевом распознавании образов. Самоорганизующаяся сеть Кохонена. Нейроэволюционное распознавание образов.

Методы распознавания образов на основе кластерного анализа. Цели кластеризации. Расстояния между образами, Меры расстояния между кластерами. Функционалы качества кластеризации. Алгоритмы кластеризации. Статистическая кластеризация на основе ЕМ-алгоритма. Алгоритм К - средних. Иерархическая кластеризация. Определение числа кластеров. Достоверность кластеризации. Многомерное шкалирование. Карта сходства. Диаграмма Шепарда.

В результате освоения дисциплины обучающийся должен:

- знать методы анализа для решения проблемных ситуаций, правовую базу информационного законодательства, правовые нормы и стандарты в области искусственного интеллекта и смежных областей, математические, естественно-научные и технические методы для решения основных, нестандартных задач создания и применения искусственного интеллекта, современные информационно-коммуникационные и интеллектуальные компьютерные технологии,

инструментальные среды, программно-технические платформы для решения профессиональных задач, фундаментальные научные принципы и методы исследований, логические методы и приемы научного исследования; методологические принципы современной науки, направления, концепции, источники знания и приемы работы с ними; основные особенности научного метода познания; программно-целевые методы решения научных проблем; основы моделирования управленческих решений.

- уметь эффективно использовать на практике теоретические компоненты науки: понятия, суждения, умозаключения, законы, применять правовые нормы и стандарты в области искусственного интеллекта при создании систем искусственного интеллекта, адаптировать существующие математические, естественно-научные и социально-экономические методы для решения основных, нестандартных задач создания и приме-нения искусственного интеллекта, применять современные информационно-коммуникационные и интеллектуальные компьютерные технологии ,инструментальные среды, программно-технические плат-формы для решения профессиональных задач, применять при решении задач профессиональной деятельности критерии эффективности функционирования информационного общества и цифровой экономики; структуру интеллектуального капитала, методы оценки эффективности.

- владеть навыками практического применения программных средств и методов построения интеллектуальных систем, навыками разработки стандартов, правил в сфере искусственного интеллекта и смежных областях, навыками решения основных, нестандартных задач создания и применения искусственного интеллекта, в том числе в новой или незнакомой среде и в междисциплинарном контексте, с применением математических, естественно-научных, социально-экономических, общеинженерных знаний и знаний в области искусственного интеллекта, навыками анализа современных методов и средств информатики и искусственного интеллекта для решения задач профессиональной деятельности, навыками методологического обоснования научного исследования, создания и применения библиотек искусственного интеллекта.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Леоненков А.В. Нечеткое моделирование в среде MATLAB и fuzzyTECH. – СПб.: БХВ-Петербург, 2005. – 736с.: ил.
2. Батыршин И.З., Недосекин А.О., Стецко А.А., Тарасов В.Б., Язенин А.В., Ярушкина Н.Г. Нечеткие гибридные системы. Теория и практика / Под редакцией Н.Г. Ярушкиной. – М.: Физматлит, 2007. – 208с.
3. Ярушкина Н.Г. Основы теории нечетких и гибридных систем. – М.: Финансы и статистика, 2004. – 320с.: ил.
4. Остроух, А. В. Системы искусственного интеллекта : монография / А. В. Остроух, Н. Е. Суркова. – 2-е изд., стер. – Санкт-Петербург : Лань, 2021. – 228 с.
5. Гаврилова, Т. А. Инженерия знаний. Модели и методы : учебник для вузов / Т. А. Гаврилова, Д. В. Кудрявцев, Д. И. Муромцев. –4-е изд., стер. – Санкт-Петербург : Лань, 2021. – 324 с.
6. Сергеев, Н. Е. Системы искусственного интеллекта : учебное пособие / Н. Е. Сергеев. – Ростов-на-Дону : ЮФУ, [б. г.]. – Часть 1 – 2016. – 118 с.
7. Рашка С., Python и машинное обучение / Рашка С., пер. с англ. А. В. Логунова. М.: ДМК Пресс, 2017 ¬ 418 с.
8. Вандер Плас Дж., Python для сложных задач: наука о данных и машинное обучение. / Вандер Плас Дж. СПб.: Питер, 2018 – 576 с.
9. Коэльо Л., Построение снстем машишюго обучения на языке Pytho/ Коэльо Луне Перо, Вилл Ричарт. М.: ДМК Пресс, 2016 – 302 с.
10. Домингос П., Верховный алгоритм: как машинное обучение изменит наш мир / Педро Домингос ; пер. с англ. В. Горохова. М. : Манн, Иванов и Фербер, 2016 – 336 с.
11. А.Мюллер, Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными / А.Мюллер, С.Гвидо. СПб.: ООО “Альфа книга” , 2017 – 480 с.
12. Дэви Силен, Основы Data Science и Big Data. Python и наука о данных / Дэви Силен, Арно Мейсман. СПб.: Питер, 2017 – 336с