ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ АСТРАХАНСКОЙ ОБЛАСТИ
ВЫСШЕГО ОБРАЗОВАНИЯ
«АСТРАХАНСКИЙ ГОСУДАРСТВЕННЫЙ
АРХИТЕКТУРНО-
СТРОИТЕЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра «САПРиМ»

ПРИКЛАДНОЙ ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ (БАЗОВЫЙ УРОВЕНЬ)

Методические указания по выполнению лабораторных работ для студентов
направление подготовки
09.04.02 «Информационные системы и технологии»
Программа «Искусственный интеллект в проектировании и производстве»
Направленность (профиль)
«Искусственный интеллект в проектировании городской среды»_
очной и заочной форм обучения

Астрахань 2021

Составитель у.т.н. доцент _____ /И.Ю. Петрова
(занимаемая должность, учёная степень и учёное звание) (подпись) И. О. Ф.

Рецензент: д.т.н. профессор _____ /Т.В. Хомяченко
(занимаемая должность, учёная степень и учёное звание) (подпись) И. О. Ф.

Методические указания к выполнению лабораторных работ для студентов направление подготовки 09.04.02 «Информационные системы и технологии» направленность (профиль) «Искусственный интеллект в проектировании городской среды» очной и заочной форм обучения рассмотрены и одобрены на заседании кафедры «Систем автоматизированного проектирования и моделирования» ГАОУ АО ВО «АГАСУ»

Протокол № 2 от 22.09.2021 г

Зав.кафедрой /Евдошенко О.И.

Согласовано с УМУ ГАОУ АО ВО «АГАСУ» 24.09.2021

Специалист УМУ _____ / Д.Л. Фурсова
подпись И.О.Ф.

Методические указания к выполнению лабораторных работ для студентов направление подготовки 09.04.02 «Информационные системы и технологии» направленность (профиль) «Искусственный интеллект в проектировании городской среды» утверждены и рекомендованы к публикации на заседании МКН подготовки «Информационные системы и технологии» направленность (профиль) «Искусственный интеллект в проектировании городской среды»

Председатель МКН «Информационные системы и технологии» направленность (профиль) «Искусственный интеллект в проектировании городской среды»

Зав.кафедрой,
доцент, к.т.н. /О.И.Евдошенко

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ.

Цель изучения дисциплины

Целью освоения дисциплины «Прикладной искусственный интеллект (базовый уровень)» является углубление уровня освоения компетенций обучающихся в соответствии с требованиями Федерального государственного образовательного стандарта высшего образования по направлению подготовки 09.04.02 «Информационные системы и технологии».

Задачей искусственного интеллекта как научного направления является воссоздание с помощью компьютера разумных рассуждений и действий. Прикладной искусственный интеллект использует средства анализа данных (такие как искусственный интеллект, машинное обучение, а также модели и алгоритмы глубокого обучения) с целью определить ключевые сведения, закономерности и рекомендации. При реализации дисциплины «Прикладной искусственный интеллект (базовый уровень)» проводятся лекционные и лабораторные занятия, а также отводится время на самостоятельную работу по углубленному рассмотрению отдельных разделов дисциплины. В рамках изучения этой дисциплины предусмотрена одна контрольная работа и завершающий экзамен.

Из всего многообразия научных и технических исследований, определяемых искусственным интеллектом, в учебной дисциплине «Прикладной искусственный интеллект (базовый уровень)» выбраны направления, связанные с проблемами представления знаний и вывода на знаниях, принципами построения систем искусственного интеллекта в виде экспертных систем на основе онтологического инжиниринга.

Перечень планируемых результатов обучения по дисциплине

УК-1ИИП. Способен понимать фундаментальные принципы работы современных систем искусственного интеллекта, разрабатывать правила и стандарты взаимодействия человека и искусственного интеллекта и использовать их в социальной и профессиональной деятельности

УК-1ИИП.1 Использует нормативно-правовую базу, правовые, этические правила, стандарты при решении задач искусственного интеллекта

УК-1ИИП.1 З-1. Знает правовую базу информационного законодательства, правовые нормы и стандарты в области искусственного интеллекта и смежных областей

УК-1ИИП.1 З-2. Знает содержание нормативно-правовых документов в сфере информационных технологий, искусственного интеллекта и информационной безопасности

УК-1ИИП.1 У-1. Умеет применять правовые нормы и стандарты в области искусственного интеллекта при создании систем искусственного интеллекта

УК-1ИИП.1 У-2. Умеет применять этические нормы и стандарты в области искусственного интеллекта при создании систем искусственного интеллекта

УК-1ИИП.1 У-3. Умеет использовать нормативно-правовые документы в сфере информационных технологий, искусственного интеллекта и информационной безопасности при разработке стандартов, норм и правил

УК-1ИИП.2 Применяет современные методы и инструменты для представления результатов научно-исследовательской деятельности

УК-1ИИП.2 З-1. Знает современные методы и инструменты для представления результатов научно-исследовательской деятельности

УК-1ИИП.2 У-1. Умеет применять современные методы и инструменты для представления результатов научно-исследовательской деятельности

ПК-1ИИП. Способен исследовать применение интеллектуальных систем для различных предметных областей

ПК-1ИИП.1Исследует направления применения систем искусственного интеллекта для различных предметных областей

ПК-1ИИП.1 З-1. Знает направления развития систем искусственного интеллекта, методы декомпозиции решаемых задач с использованием искусственного интеллекта

ПК-1ИИП.1 У-1. Умеет осуществлять декомпозицию решаемых задач с использованием искусственного интеллекта

ПК-1ИИП.2 Выбирает комплексы методов и инструментальных средств искусственного интеллекта для решения задач в зависимости от особенностей предметной области

ПК-1ИИП.2 З-1. Знает методы и инструментальные средства систем искусственного интеллекта, критерии их выбора и методы комплексирования в рамках применения интегрированных гибридных интеллектуальных систем различного назначения

ПК-1ИИП.2 У-1. Умеет выбирать и комплексно применять методы и инструментальные средства систем искусственного интеллекта, критерии их выбора

ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ ПРИКЛАДНЫХ ТЕХНОЛОГИЙ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В УМНОМ ГОРОДЕ.

В рамках программы «Цифровая экономика России один из разделов - «Умный город» - посвящен описанию «инновационных городов, в которых можно внедрять комплекс технических решений и организационных мероприятий, направленных на достижение максимально возможного качества управления ресурсами и предоставления услуг, в целях создания благоприятных условий проживания и пребывания, деловой активности нынешнего и будущего поколений». Ключевыми составляющими программы «Умный город» станут искусственный интеллект, технология блокчейна и виртуальная реальность. Согласно программе, к 2025 г. должна быть разработана онтологическая модель деятельности «умного» города, представляющая собой структурированное описание объектов умного города и отношений между ними. Это соответствует основным требованиям международных стандартов, реализующих подход «Умный город»

Городское население в мире неумолимо растет (в России в городах проживают 72% населения), поэтому развитие цифровой экономики происходит в первую очередь в городах. Можно сказать, что города – это столицы цифровой экономики.

Термин «умный город» относится к городу, который разумным образом управляет всеми связанными с ним ресурсами (городской инфраструктурой: транспортом, образованием, здравоохранением, системами ЖКХ, безопасности и т.д.) с целью повышения качества услуг, предоставляемых гражданам, и улучшения качества их жизни.

Модель города - цифровое представление физических и социальных характеристик города.

Одним из перспективных направлений развития концепции «умный город» становится углубление использования искусственного интеллекта с возможностью его постоянного обучения. Это ключевое направление окажет гораздо более сильное влияние на рынок городских данных и способность искусственного интеллекта учитывать их. В дополнение к жестким условиям конкурентного рынка это, несомненно, окажет влияние на развитие экономики не только одного города, но и целой цепочки городов в перспективе всей страны.

Онтологическая модель умного города.

Один из подходов к формализации знаний в модели «Умный город» основан на создании онтологий. **Онтология** – это структурная спецификация некоторой предметной области (например, умного города как сложной социально-экономической системы), ее формализованное представление, которое включает словарь указателей на термины предметной области и логические выражения, которые описывают, как они соотносятся друг с другом.

Онтологии могут быть представлены семантическими сетями с описанием основных сущностей предметной области в виде классов, атрибутов и отношений. Это помогает отделить базы данных (БД) и код разработчиков от знаний, которые крайне важны для адаптации системы. На данный момент концепция Умного города чаще всего опирается на создание единого хранилища данных в городе (единая городская БД) для обеспечения доступа к услугам.

Для реализации подхода «умный город» разрабатываются и применяются открытые стандарты, гарантирующие интероперабельность технологических решений, совместимость и взаимодействие с различными системами (поставщиками) и различными платформами данных. Во всем мире на основе стандартов разрабатываются онтологии городских данных, основанные на едином понимании семантики и связанных таксономиях во всех отраслях городского хозяйства: транспорт, электронное правительство, энергетика и т. д.

Онтологии могут использоваться в качестве посредника между пользователями (жители и гости города, сотрудники городских ЖКХ-структур, администраций и т.д.) и городской информационной системой, они позволяют формализовать договоренности о терминологии в единой городской БД.

Онтологии позволяют на основе общей терминологии связывать информацию, представленную в виде, требуемом для обработки, с информацией, представленной в удобной форме для восприятия человеком.

Система управления знаниями «Умный город»

На рис.1 показана структурная схема управления знаниями «Умный город», которая является главным инструментом для менеджеров знаний в ключевых секторах управления городом

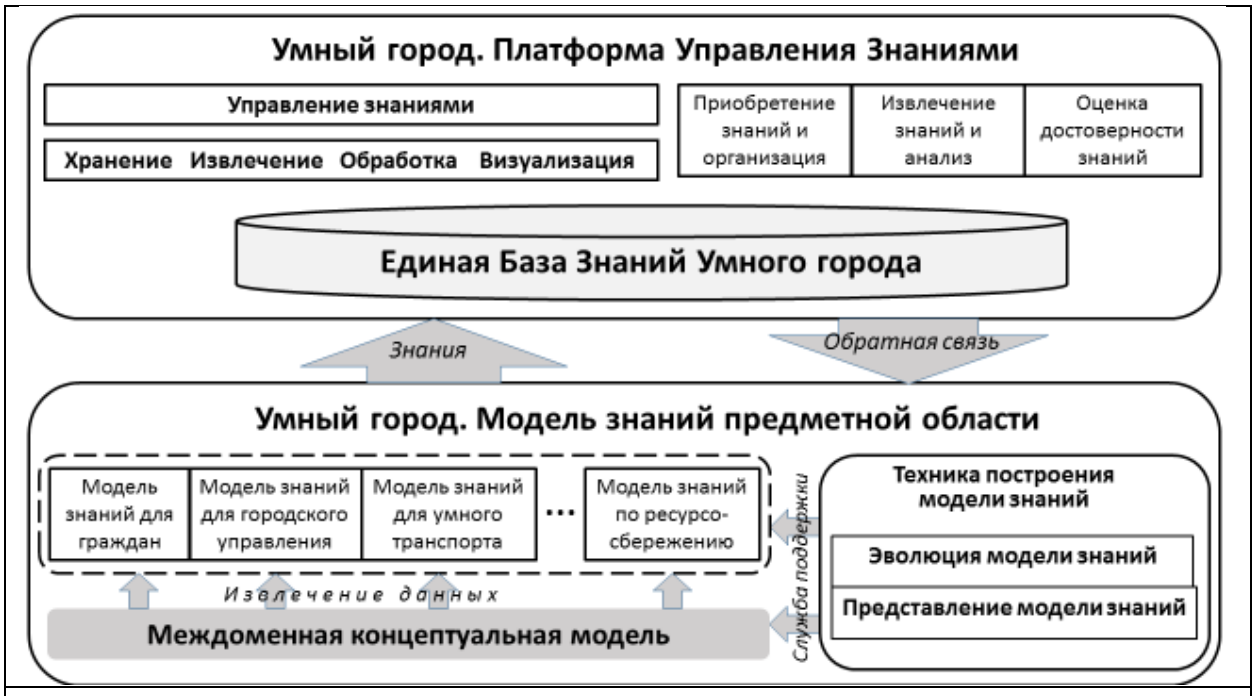


Рис.1. Структурная схема управления знаниями «Умный город»

Используя *семантические технологии*, становится возможным создавать единую базу знаний умного города, извлекать важную информацию из описания различных подсистем, интегрировать разнородные системы между собой. Все эти решения позволят привести к более низкой стоимости жизненного цикла городских систем.

Семантические технологии на основе онтологий обеспечивают аргументацию, используя связи, правила, логику и условия, описанные в онтологии. Семантическое отображение с помощью онтологий - механизм для объединения информации из отраслевых систем (а не их реинжиниринга).

Можно выделить следующие сферы практического применения прикладных технологий искусственного интеллекта в умном городе:

- Умная энергия
- Умный транспорт
- Аналитика массива данных
- Интеллектуальная инфраструктура городского развития
- Интеллектуальная инфраструктура
- Интеллектуальная мобильность данных и технологий
- Интеллектуальные устройства Интернета вещей

Прикладной искусственный интеллект использует средства анализа данных (такие как искусственный интеллект, машинное обучение, а также модели и алгоритмы глубокого обучения) с целью определить ключевые сведения, закономерности и рекомендации в развитии умного города.

Лабораторная работа № 1

Классификация знаний

Специалисты в области информационных технологий по роду своей деятельности обязаны иметь четкое понятие о категориях "информация", "данные", "знание".

Данные - 1) сведения, необходимые для какого-либо вывода, решения, процедуры (например: много данных, цифровые данные); 2) основания для чего-нибудь, качества (например: голосовые данные, иметь все данные для получения премии).

Данные - это совокупность сведений, зафиксированных на определенном носителе в форме, пригодной для постоянного хранения, передачи и обработки. Преобразование и обработка данных позволяет получить информацию.

Информация – (лат. informatio) - 1) сообщение о чем-либо; 2) сведения, являющиеся объектом хранения, переработки и передачи (например, генетическая информация); 3) в математике (кибернетике) - количественная мера устранения неопределенности (энтропия), мера организации системы; в теории информации - раздел кибернетики, изучающий количественные закономерности, связанные со сбором, передачей, преобразованием и вычислением информации.

Информация - это результат преобразования и анализа данных. Отличие информации от данных состоит в том, что данные - это фиксированные сведения о событиях и явлениях, которые хранятся на определенных носителях, а информация появляется в результате обработки данных при решении конкретных задач. Например, в базах данных хранятся различные данные, а по определенному запросу система управления базой данных выдает требуемую информацию.

Знания - 1) постижение действительности сознанием, наука (например: важная область знания, тяга к знанию); 2) совокупность сведений, познаний в какой-либо области (например: область знаний, тяга к знаниям).

Знание (англ. - Knowledge) — **проверенные общественной практикой полезные сведения, которые могут многократно использоваться людьми для решения тех или иных задач.**

Знания – это совокупность сведений (данных или программ), отражающая их знания человека-специалиста (эксперта) в определенной предметной области и предназначенных для хранения в базах знаний. Знания отражают множество возможных ситуаций, связанных с состоянием и конкретной реализацией объектов определенного типа, способы перехода от одного описания объекта к другому. Для знаний характерны внутренняя интерпретируемость, структурированность, связанность и активность- Условно можно записать, что "знания = факты + убеждения + правила".

Человек приобретает *знания*, получая сообщения из различных источников. Знания делятся на декларативные и процедурные.

Декларативные знания - это знания об определенных явлениях, событиях, свойствах объектов, зависимостях. Декларативные знания начинаются со слов: "Я знаю, что..."

Процедурные знания определяют действия для достижения какой-либо цели. Процедурные знания начинаются со слов: "Я знаю, как ..."

Взаимосвязь данных, информации и знаний в процессе принятия решений представлена на рисунке 2.

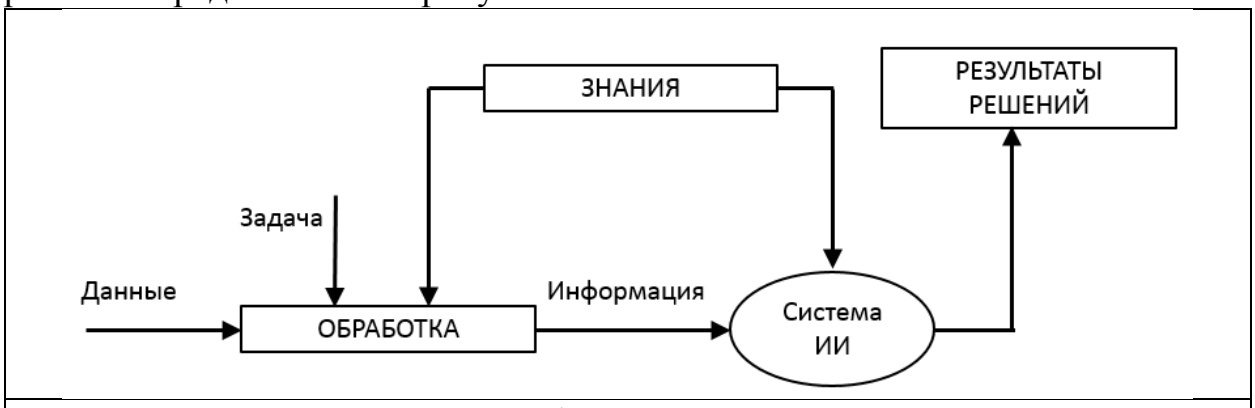


Рис.2. Взаимосвязь данных, информации и знаний в процессе принятия решений

Цель работы: освоить построение модели знаний в виде графа.

Задачи: изучить заданную предметную область и построить модель знаний в виде графа.

Методические указания.

Для построения модели представления знаний в виде графа необходимо выполнить следующие шаги:

1) Определить целевые действия задачи (являющиеся решениями).

2) Определить промежуточные действия или цепочку действий, между начальным состоянием и конечным (между тем, что имеется, и целевым действием).

3) Опередить условия для каждого действия, при котором его целесообразно и возможно выполнить. Определить порядок выполнения действий.

4) Добавить конкретные факты, исходя из поставленной задачи.

5) Преобразовать полученный порядок действий и соответствующие им факты, условия и действия.

6) Для проверки правильности построения записать цепочки, явно проследив связи между ними.

Этот набор шагов предполагает движение при построении модели от результата к начальному состоянию, но возможно и движение от начального состояния к результату (шаги 1 и 2).

7) Присвоить обозначения фактам Ф, правилам П, действиям Д.

8) Построить граф предметной области. (пример рис.1)

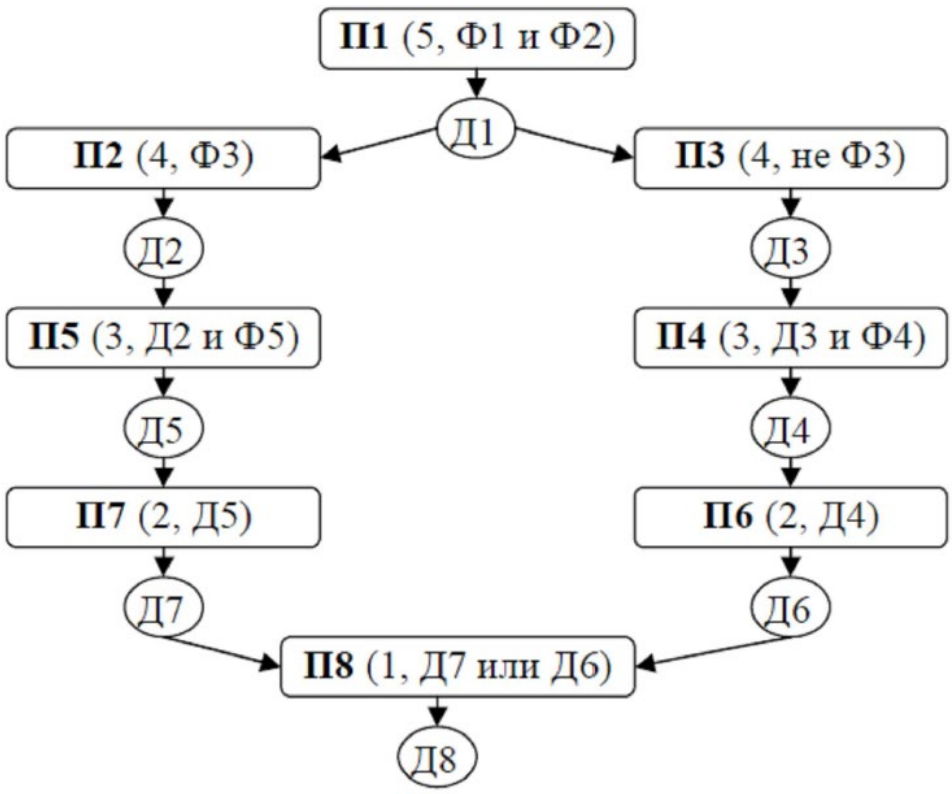


Рис.2. Пример графа модели знаний

Задание 1.

1. Из предложенного списка выберите декларативные и процедурные знания:
   1.1. чтобы определить кислотность раствора, нужно опустить в него лакмусовую бумажку и посмотреть на ее цвет: если цвет красный, то раствор кислотный; если цвет синий, то раствор щелочной; если цвет не изменился, то раствор нейтральный;
   1.2. Вашингтон — столица США;
   1.3. чтобы найти площадь прямоугольника, нужно его ширину умножить на длину;
   1.4. для определения числа решений квадратного уравнения, нужно вычислить его дискриминант: если он меньше 0, то решений нет; если равен 0, то решение одно; если больше 0, то решений — 2;
   1.5. А. С. Пушкин — автор поэмы «Полтава»;
   1.6. Н. Винер — основатель кибернетики;
   1.7. чтобы сократить дробь, нужно найти наибольший общий делитель числителя и знаменателя и разделить на него и числитель, и знаменатель;
   1.8. Юрий Гагарин — первый космонавт.

Ответ оформите в виде таблицы:

Декларативные знания	Процедурные знания

Задание 2

1. Построить модель представления знаний в предметной области «Компьютерные сети» (организация).
2. Построить модель представления знаний в предметной области «Университет» (учебный процесс).
3. Построить модель представления знаний в предметной области «Компьютерная безопасность» (средства и способы ее обеспечения).
4. Построить модель представления знаний в предметной области «Компьютерная безопасность» (угрозы).
5. Построить модель представления знаний в предметной области «Разработка информационных систем» (ведение информационного проекта).
6. Построить модель представления знаний в предметной области «Операционные системы» (функционирование).
7. Построить модель представления знаний в предметной области «Информационные системы» (виды и функционирование).
8. Построить модель представления знаний в предметной области «Предприятие» (структура и функционирование).

Контрольные вопросы

1. Что такое факт?
2. Дайте определение данным
3. Что такое знания?
4. Что такое поле знаний?
5. Кто такой инженер-когнитолог?

Содержание отчета:

- цель работы
- краткие теоретические сведения
- описание предметной области
- граф предметной области
- ответы на вопросы.

Лабораторная работа № 2

Изучение современного редактора онтологий fluent editor. Создание первой онтологии

Цель работы: Освоить работу с онтологическим редактором Fluent Editor, получить навыки построения онтологий.

Задачи:

1. Создать онтологию классификации предметной области (по варианту).
2. Наполнить онтологию экземплярами (не менее 30 понятий).
3. Осуществить несколько запросов.

Теоретическое введение

Онтологический анализ, помимо упорядочивания знаний о предметной области, также способствует повышению качества выполняемых работ и услуг.

Онтологический анализ начинается с составления словаря терминов, который используется при обсуждении и исследовании характеристик объектов и процессов, составляющих рассматриваемую систему, а также создания системы точных определений этих терминов. Документируются основные логические взаимосвязи между соответствующими введенным терминам понятиями. Таким образом, онтология включает в себя совокупность терминов и правила, согласно которым эти термины могут быть скомбинированы для построения достоверных утверждений о состоянии рассматриваемой системы в некоторый момент времени. Можно сказать, что онтология представляет собой некий словарь данных, включающий в себя терминологию и модель поведения системы.

Редакторами или конструкторами онтологий называют инструментальные программные средства, созданные специально для проектирования, редактирования и анализа онтологий. Основная функция любого редактора онтологий состоит в поддержке процесса формализации знаний и представлении онтологии как спецификации.

Практическая разработка онтологии включает:

- определение классов в онтологии (классы – это абстрактные группы, коллекции или наборы объектов);
- расположение классов в таксономическую иерархию (подкласс, надкласс);
- определение экземпляров в классах (экземпляры – это отдельные представители класса сущностей или явлений, то есть конкретные элементы какой-либо категории);
- определение условий взаимосвязи между классами и экземплярами;
- определение отношений и атрибутов. (отношения представляют тип взаимодействия между понятиями предметной области, атрибуты – свойства классов и экземпляров).

После этого можно создать базу знаний, определив отдельные экземпляры этих классов, введя в определенный слот значение и дополнительные ограничения для слота.

Онтологический редактор fluent editor

Fluent Editor – онтологический редактор польской компании Cognitum для редактирования сложных онтологий, при создании которых используется контролируемый язык (Controlled Natural Language – CNL).

CNL (контролируемый язык) – упрощенная версия естественного языка, созданная путем ограничения грамматики, терминологии и речевых оборотов, чтобы снизить или искоренить многозначность и сложность естественного языка. В Fluent Editor контролируемым естественным языком является английский. Естественно-языковое описание является главным отличием Fluent Editor от других онтологических редакторов и позволяет освоить создание онтологий широкой группе пользователей.

В Fluent Editor имеется встроенный механизм рассуждений (Embedded Reasoner), который автоматически формирует XML, RDF и OWL файлы. Существуют несколько «языков» для записи семантических моделей, основными из которых являются RDF и OWL.

RDF (Resource Description Framework) - позволяет записывать простейшие факты об объектах, классах и свойствах. RDFS (RDF Schema) — язык описания словарей для RDF, семантическое расширение RDF. RDFS определяет классы, свойства и другие ресурсы.

OWL (Web Ontology Language) — язык описания онтологий для семантической паутины. Язык OWL позволяет описывать сложные взаимосвязи классов и отношений между ними, присущих веб-документам и приложениям. Фактически это словарь, расширяющий набор терминов, определенных RDFS.

Fluent Editor содержит интеллектуальный редактор (Predictive Editor) – редактор, самостоятельно отслеживающий грамматически и морфологически неправильный текст, активно помогающий пользователю в грамотном написании онтологии.

1. Первое знакомство с редактором

Онтологический редактор Fluent Editor находится в свободном для скачивания доступе в сети Интернет (https://www.cognitum.eu/download/). Скачайте программу и установите на свой компьютер. Затем запустите Fluent Editor (рис.3).

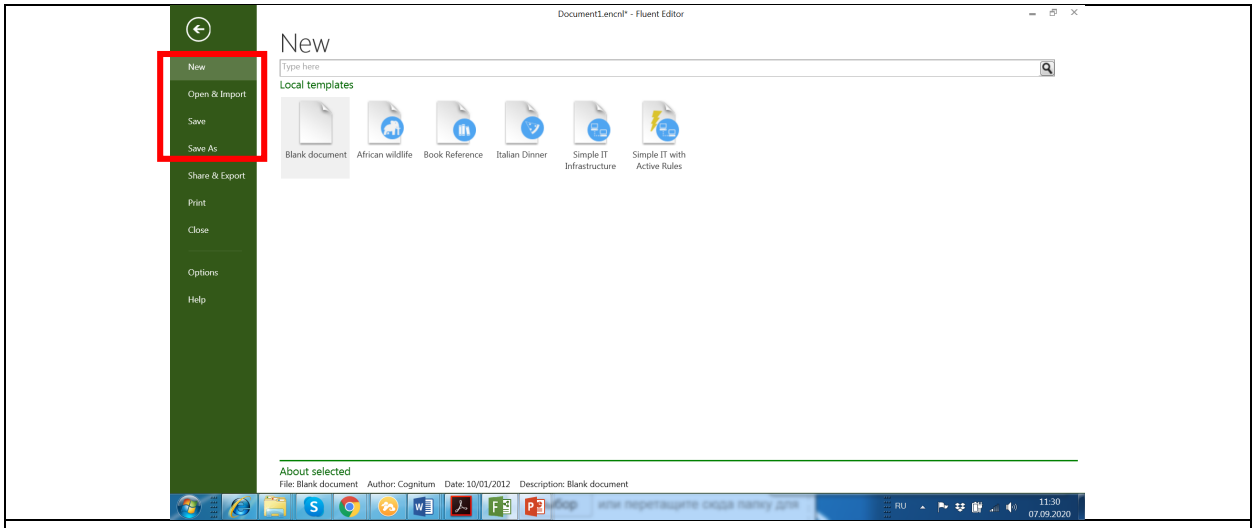


Рис.3. Создание проекта.

Чтобы создать новый проект, выберите «File» на панели быстрого доступа, а затем нажмите кнопку «New». Тогда представиться возможность либо создать новый проект «Blank document», либо открыть уже существующий, который послужит примером структуры проекта. Для создания собственной онтологии выберите «Blank document».

В правой части рабочего окна редактора находится вкладка отображения дерева таксономии «Taxonomy Tree» (рис. 4). Дерево таксономии отображается для каждого файла OWL (Web Ontology Language), а строится и редактируется на основе данных из этого файла.

В центре рабочего окна программы находится поле для создания онтологии – окно редактора CNL. Окно CNL редактора – основная часть Fluent Editor, в которой формируются, просматриваются и редактируются файлы онтологий.

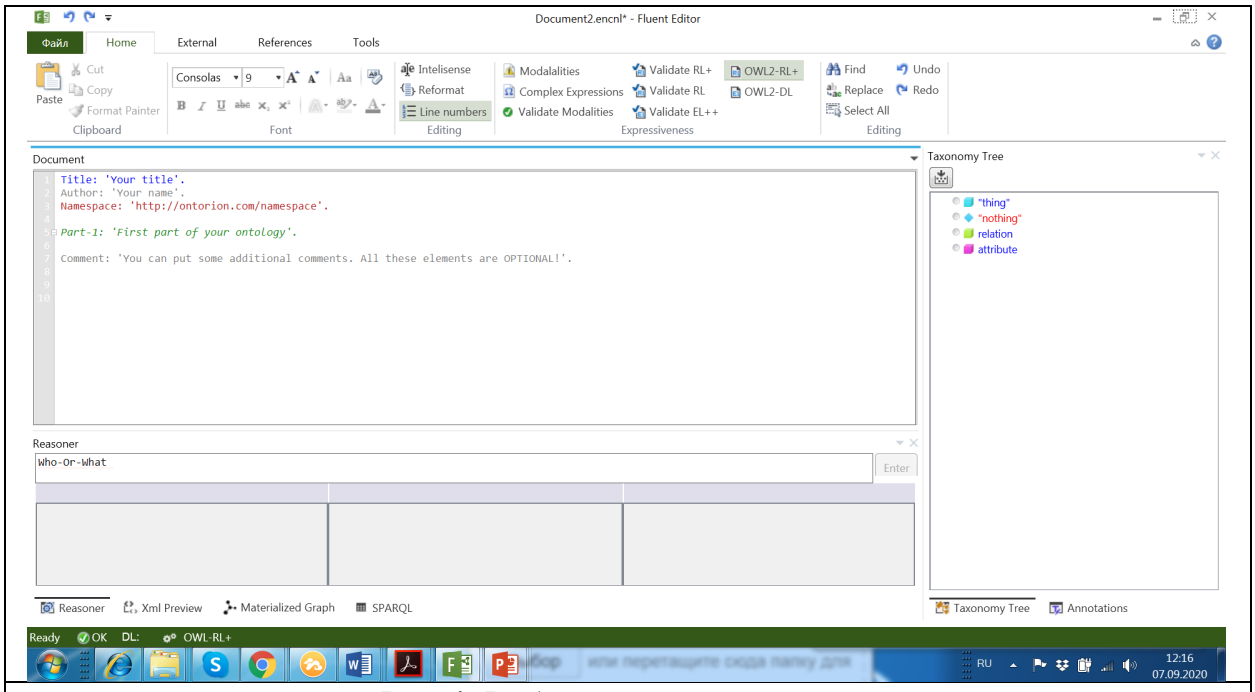


Рис.4. Рабочее окно редактора.

Окно Document содержит в себе несколько полей: Title (Заголовок), Author (Автор), Part-1: ‘First part of your ontology’ (Часть-1: ‘Первая часть вашей онтологии’.), Comment (Комментарий).

После слова «Title» вводится название/заголовок онтологии, в строке «Author» записываются авторы, создавшие онтологию. Название и авторов онтологии необходимо указывать в кавычках, а в конце ставить точку. Например, «Author: 'Петрова И.Ю.'.»

Написание онтологии на языке CNL начинается после слов «Part-1: ‘First part of your ontology’.».

Для сохранения проекта откройте в панели быстрого доступа вкладку «File» -> «Save as» или наберите сочетание клавиш Ctrl + S.

2. Создание классов и подклассов

При вводе текста онтологии в поле Document, редактор Fluent Editor может автоматически оказать помощь в нескольких направлениях:

1) Подсказка «Вставка» дает список последующих слов, которые могут быть написаны после введенной фразы. Можно либо ввести слово вручную, либо выбрать его из выплывающего списка. Ключевые слова редактора помечены синим шрифтом, а пользовательские слова отмечены черным. Чтобы открыть окно подсказки CNL, нажмите кнопку Intelisense или поставьте курсор в Part-1 и воспользуйтесь сочетанием клавиш Ctrl + Spacebar.
2) Синтаксические ошибки будут отмечены красным подчеркиванием. Если ввести фразу, которая неверна в соответствии с грамматикой Fluent Editor, то эта фраза выделится красным подчеркиванием.

Рассмотрим построение онтологии в редакторе Fluent Editor для устройств интернета вещей в умном доме. Каждое устройство умного дома можно описать с помощью онтологий, которые формализуют основные возможности и ограничения работы конкретных устройств. В качестве одного такого устройства рассмотрим робот-пылесос.

Основные функции, которые может выполнять робот-пылесос:

- перемещение на плоскости,
- построение карты помещения,
- анализ светочувствительных датчиков,
- уборка помещения.

В соответствии с этим можно выделить четыре класса верхнего уровня: *Характеристики* (Characteristics) робота-пылесоса, *сенсоры* робота (Sensors), *освещенность помещения* (Lightness) и *операции*, выполняемые роботом (Operation).

Класс *Характеристики*: марка (класс *Brand*), потребляемая мощность (класс *Power*), уровень шума (класс *NoiseLevel*), стоимость (класс *Cost*), вес робота (класс *TotalWeight*)

Класс *Освещенность помещения*: освещенность может быть двух типов: достаточно для работы бесконтактных сенсоров робота-пылесоса (класс *Light*) и недостаточно (класс *Dark*).

Класс *Сенсоры*: датчики освещенности (сенсоры) могут быть контактными (класс ContactSensor) и бесконтактными (класс VideoSensor). Контактные сенсоры функционируют при любой освещенности помещения, в то время как бесконтактные требуют наличия достаточной освещенности в помещении.

Класс *Операции, выполняемые роботом-пылесосом*: всасывание (класс *Cleaning*) и перемещение (класс *Movement*). Для эффективной уборки помещения роботу-пылесосу необходима достаточная его освещенность для построения карты и упорядоченного перемещения по помещению. Если освещенность недостаточна, то робот прекращает работу (класс *Stopped*).

На рис.5 показана иерархическая схема классов для онтологии робота-пылесоса.

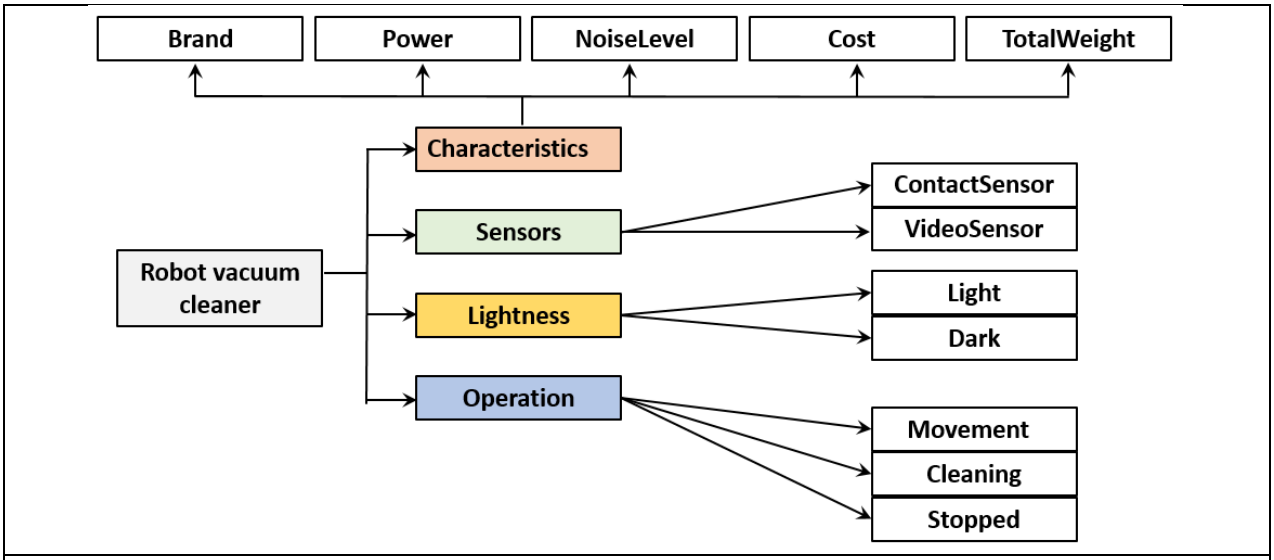


Рис.5 Иерархическая схема классов для онтологии робота-пылесоса.

Перейдем к созданию классов и подклассов в Fluent Editor. Класс создается при задании класса как вещи (от англ. thing) или, когда классу присваивается подкласс или экземпляр.

В первом случае необходимо записать <Every “Robot vacuum cleaner” is a thing.>, во втором – для присвоения классу подкласса необходимо поставить название класса и подкласса в предложении на CNL в качестве дополнения и подлежащего соответственно. Например, присвоим классу «Robot vacuum cleaner» подкласс «Characteristics». Запишем «Every “Characteristics” is an “ Robot vacuum cleaner”.» (Каждая Характеристика – это Робот-пылесос»).

В дереве таксономии (окно справа) появились данные объекты в описанной иерархии (рис. 6). По аналогии присвойте классу «Robot vacuum cleaner» остальные подклассы.

Классы и подклассы в дереве таксономии обозначаются голубыми ромбами, как показано на рис. 6.

3. Создание экземпляров.

Для создания экземпляра, также следует описать его зависимость от класса/подкласса, также как создавали зависимость подкласса от класса. **В отличие от класса, экземпляры описываются всегда с заглавной буквы.**

Создадим экземпляры подкласса «Characteristics». Для этого используем информацию из таблицы 1, в которой показаны характеристики нескольких пылесосов. Например: *Xiaomi is a "Brand"* или *Iboto-Smart is a "Brand"*.

В дереве таксономии экземпляры, присвоенные классам, обозначаются зеленым кругом ●, как показано на рис. 6.

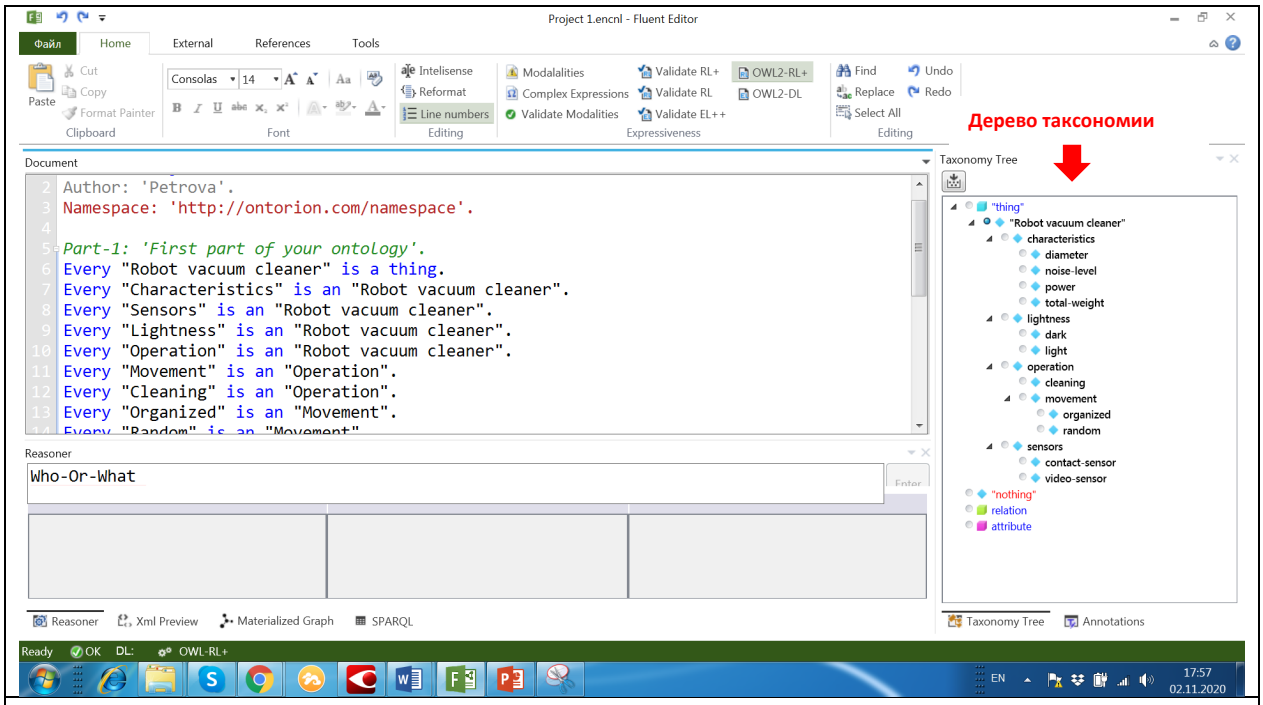


Рис.6. Создание классов

Табл.1

Brand	Power (W)	NoiseLevel (Db)	Cost (₽)	TotalWeight (kg)	Sensor
Xiaomi	40	72	17000	3,6	Contactsensor
Iboto Smart	25	54	20000	2,5	Videosensor
Sweeping Robot	25	N/A	12500	3,4	Contactsensor
Dreame F9	40	72	22000	3,7	Videosensor

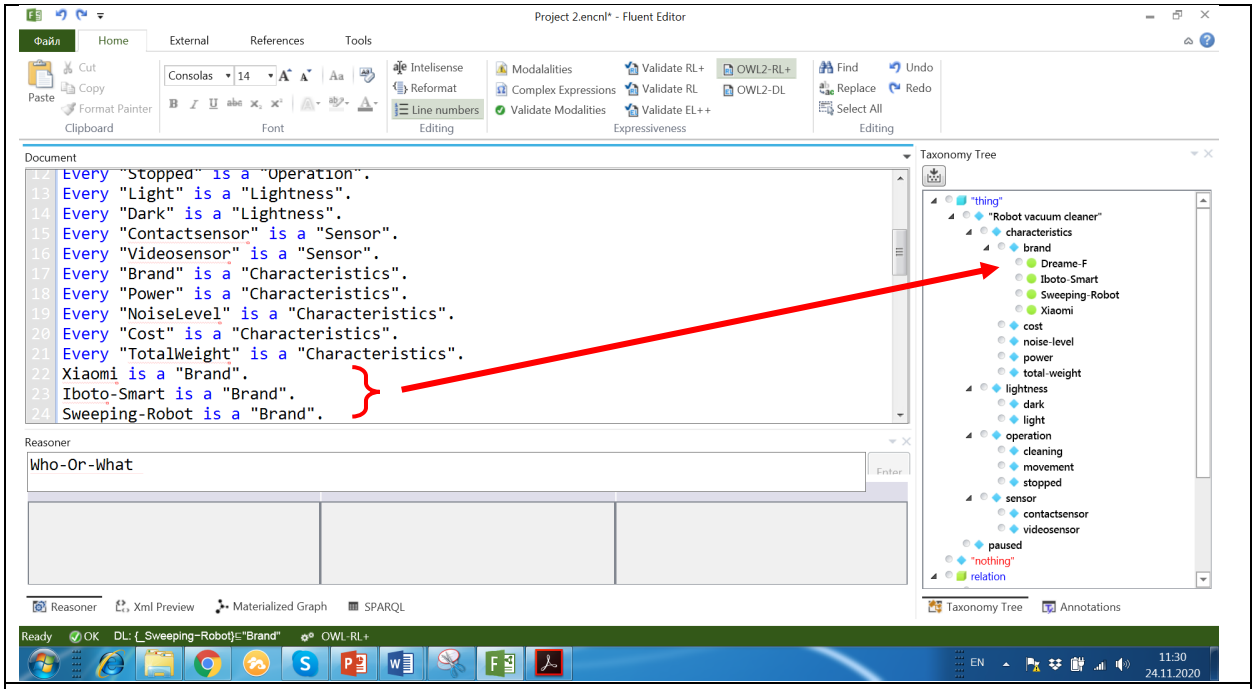
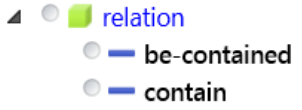


Рис.7. Создание экземпляров классов

4. Создание отношений

В дереве таксономии в пункте «relation», обозначенном зеленым кубом (рис. 7) отражаются все созданные отношения между объектами. Для создания отношений между объектами следует описать их связь на языке CNL, заменяя пробелы между словами в словосочетании на дефис. Например, *Videosensor be-contained Dreame-F.*(Видеосенсор содержится в Dreame-F) или *Iboto-Smart contain a sensor.*(Iboto-Smart содержит сенсор).



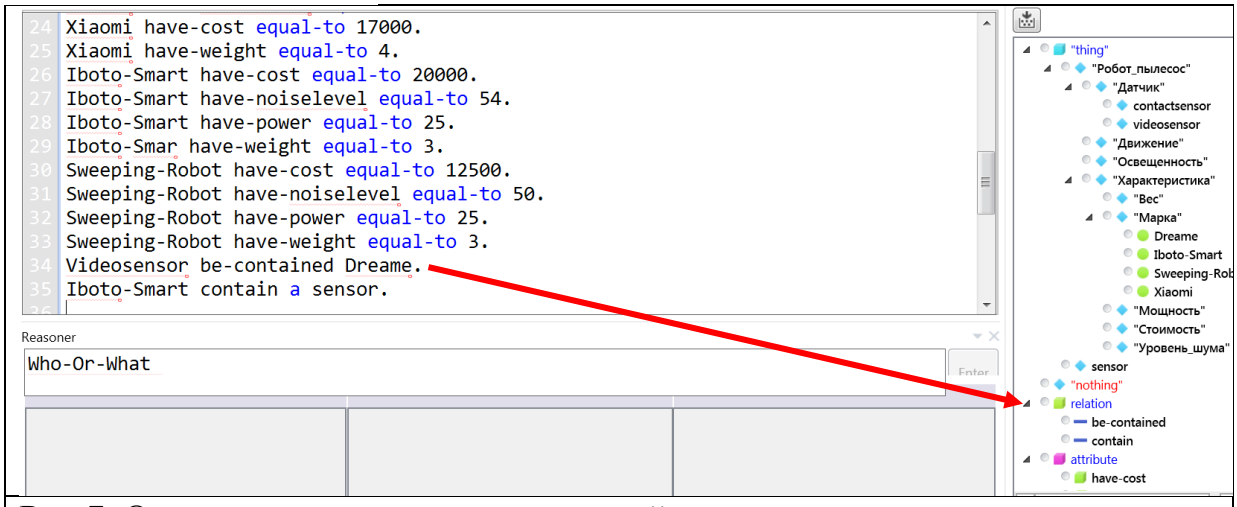


Рис.7. Отражение созданных отношений в дереве таксономии

5. Присвоение свойств

Созданные свойства отражаются в дереве таксономии под пунктом «attribute», обозначенном розовым кубом .

Чтобы присвоить объекту какое-либо свойство следует сделать то же самое, что и при создании отношений. Программа сама определит, что относится к отношениям, а что к свойствам объектов (все свойства объектов начинаются с глагола иметь «have»). При создании свойств, следует учитывать, что свойства, состоящие из нескольких слов, разделяются дефисом и пишутся с маленькой буквы.

Например:

Xiaomi have-power equal-to 40.	Xiaomi имеет_мощность равную 40
Xiaomi have-noiselevel equal-to 72.	Xiaomi имеет_уровень_шума равный 72
Xiaomi have-cost equal-to 17000.	Xiaomi имеет_цену равную 17000

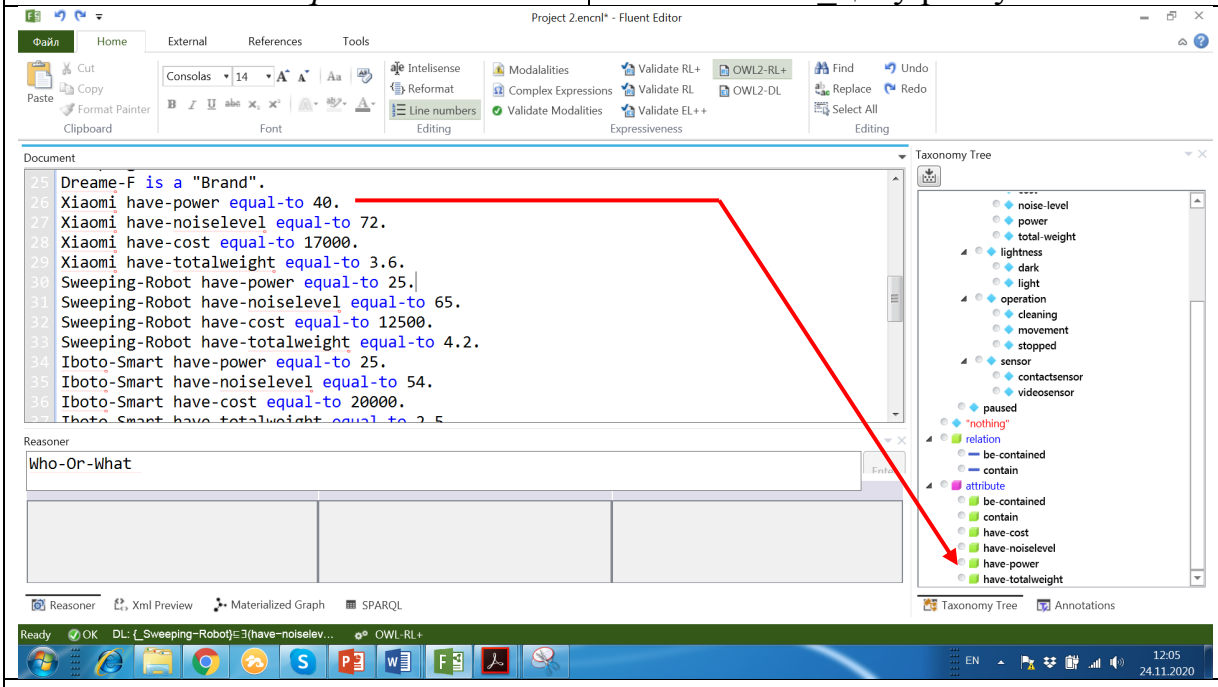


Рис.8. Присвоение свойств пылесосам разных марок (брендов).

6. Проверка грамматики CNL

После написания онтологии, можно переформатировать выделенные строки, нажав на кнопку «Reformat» (), находящуюся во вкладке «Home» на панели быстрого доступа, или Ctrl + U после выбора предложения. Переформатирование выполняет синтаксический анализ и преобразования CNL-предложений, что приводит к исправлению редактором предложений в соответствии с английской грамматикой.

7. CNL-диаграмма

Fluent Editor предоставляет пользователю несколько возможностей представить созданную онтологию в графическом виде. Один из них построение диаграммы CNL, которая отражает зависимости классов и подклассов, экземпляров, а также отношения между ними в виде диаграммы. Для использования данной функции редактора на панели быстрого доступа выберете вкладку «Tools», далее нажмите «CNL Diagram». Рядом с вкладкой «Document» в окне редактора CNL появилась еще одна вкладка «Document Diagram», где сформировалась иерархическая схема описанной онтологии (рис.9.).

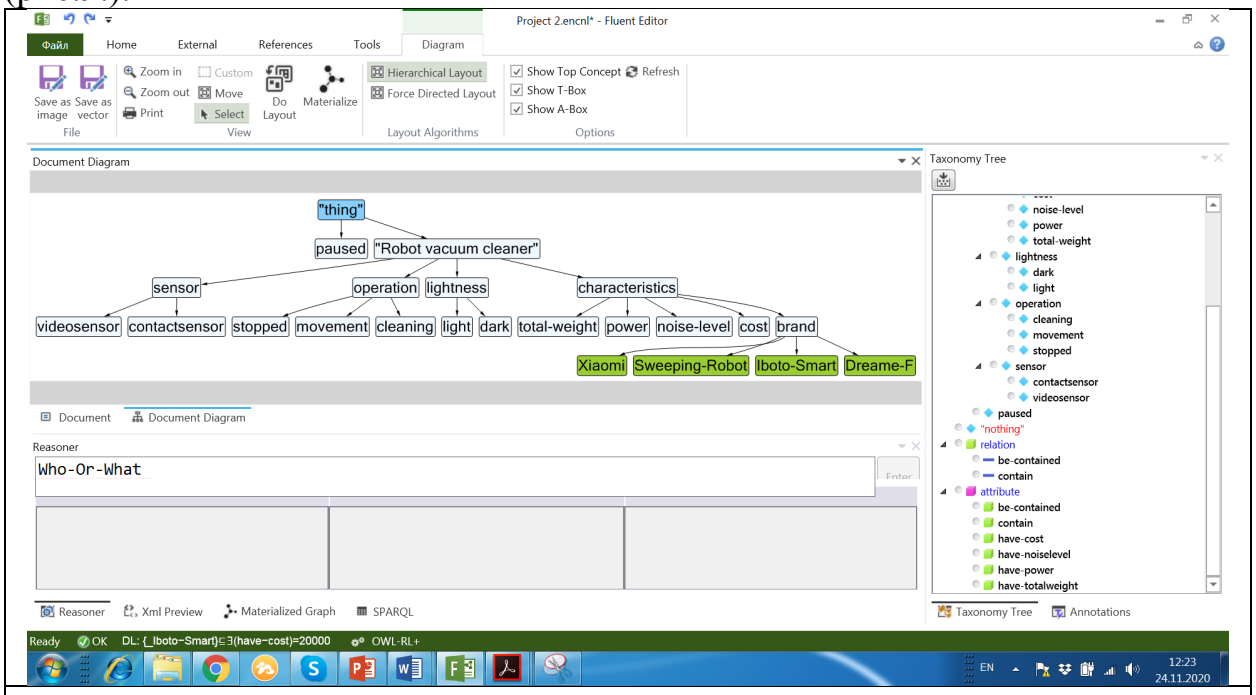


Рис.9. CNL диаграмма.

После открытия вкладки «Tools» и выбора «CNL Diagram», рядом с вкладкой «Tools» образовалась вкладка «Diagram», содержащая в себе инструменты редактирования диаграммы, в ней можно отправить схему на печать, сохранить ее, а также изменять положение объектов по полю с помощью кнопок: Zoom in (увеличение изображения), Zoom out (уменьшение изображения), Do Layout (центрирование изображения на экране), Hierarchical Layout (расположение зависимостей объектов диаграммы от верхнего уровня сверху до нижнего уровня снизу), Force Directed Layout (расположение зависимостей объектов диаграммы от верхнего уровня слева до нижнего уровня справа), Materialize - (обновляет диаграмму, с учетом вновь внесенных изменений в окне «Document», а также добавляет описание отношений к объектам диаграммы). Изменять расположение объектов можно и вручную, передвигая объекты с помощью нажатия и удержания их левой кнопкой мыши. Диаграмму можно сохранять в форматах *.jpg и*.svg, нажав на кнопки «Save as image» и «Save as vector» соответственно.

8. Reasoner

В Fluent Editor встроен механизм, с помощью которого задаются вопросы к онтологии – Reasoner (от англ. «мыслитель»). Интерфейс Reasoner представлен на рис. 10. В верхней строке задаются вопросы, обязательно начинающиеся со слов «Who Or What». Далее следует написать вопрос, учитывая грамматику CNL. Для начинающего пользователя редактор предлагает помощь в написании вопроса к онтологии. Необходимо нажать сочетание клавиш Ctrl+Spacebar, выплывет окно подсказок, т.е. слов, которые вы можете использовать на данном месте, учитывая написанную Вами онтологию и синтаксис CNL. В соответствии с правилами большинства языков после вопросительного предложения следует поставить знак «?». Ответы на вопрос выводятся в трех колонках под строкой, в которой задаются вопросы, в зависимости от того, какие отношения существуют между вопросом и ответом на него. Три колонки отражают соотношение вопроса и ответа как класса с классом или как экземпляра с экземпляром, или как класса с экземпляром.

Рис.10. Окно Reasoner

Например, зададим вопрос к онтологии: «Что является роботом пылесосом»: “Who-Or-What is a "Robot vacuum cleaner” ? Результат запроса показан на рис.11.

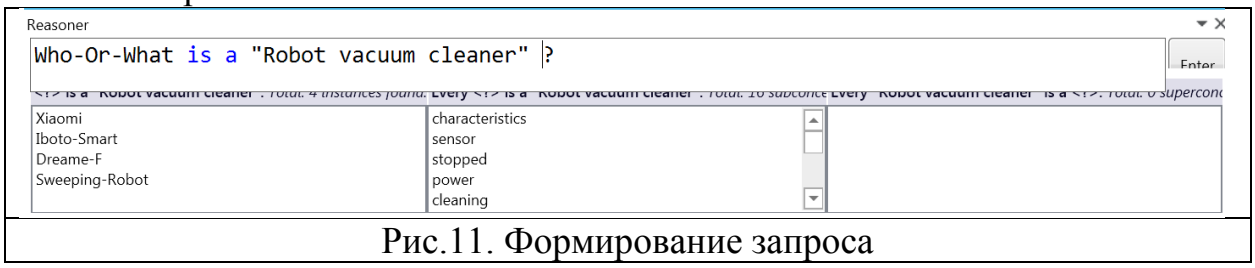


Рис.11. Формирование запроса

Другой пример, зададим вопрос к онтологии «Какой пылесос имеет вес равный 3?», на языке CNL этот вопрос представляется так: «Who Or has-weight equel-to 3?». Для продолжения процесса необходимо либо нажать кнопку в окне Reasoner, либо клавишу «Enter» на клавиатуре. После обработки редактором данных и нахождения решения был получен ответ «Iboto-Smart» и «Sweeping-Robot». Решение соответствует верному (рис. 12).

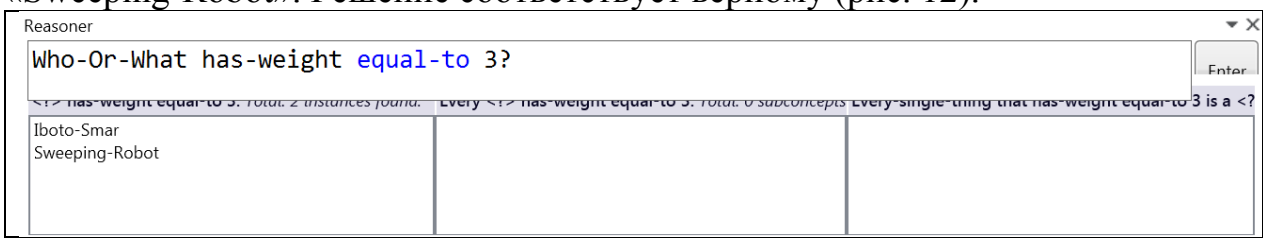


Рис.12. Окно Reasoner с ответом на вопрос «Who Or has-weight equel-to 3?», заданный к написанной онтологии

Обратите внимание на набор логических выражений, встроенных в редактор. В редакторе они всегда выделены синим цветом. Одно из них, использовавшееся ранее для задания величин характеристик робота-пылесоса, выражение «equal-to». При задании вопроса о числовых характеристиках пылесоса (вес, мощность, уровень шума, стоимость), которым соответствуют конкретные числовые значения, также можно использовать логические выражения «lower-than» («менее чем»), «more-than» («более чем»), «lower or equal to» («равен или менее чем») и т.д.

К описываемому примеру применим следующий набор вопросов: «Who Or What have-weight lower-than 4?», «Who-Or-What have-power lower-or-equal-to 40? », «Who Or What have-power greater or equal to 40 and has weight lower than 4?» и т.д.

9. Создание интернет-ссылок

Если в онтологии используются данные, взятые с сети Интернет, необходимо указать ссылки на них. В редакторе Fluent Editor есть специальная функция Reference (от англ. «ссылка») формирующая список использованных источников.

Ссылки создаются в окне Document и прикрепляются только к классам и экземплярам. Для создания Reference необходимо поставить курсор в окно написания онтологии и нажать сочетание клавиш Ctrl+Spacebar, выбрать в выплывающем окне строку или ввести в окне Document слово «References» после чего написать «[id] (‘ссылка на источник’) в конце написания списка всех ссылок обязательно ставится точка «.». Вместо «[id]» следует написать присвоенное классу-экземпляру слово, кратко описывающее ссылку, к примеру, «[xi]», в кавычках (‘’) указывается интернет-адрес – (‘https://mi-shop.com/ru/catalog/smart_devices/vacuumcleaner/’). Необходимо добавить выбранный «id» ко всем упоминаниям класса/экземпляра в онтологии. Цифры в имени «id» не используются, а также пробелы, дефисы и заглавные буквы. Чтобы завершить написание Reference ставится точка. На рис. 13 представлен пример создания ссылки на источник в редакторе Fluent Editor.

```
Xiaomi is a "Марка".
Iboto-Smart is a "Марка".
Sweeping-Robot is a "Марка".
Dreame is a "Марка".
Xiaomi[xi] have-power equal-to 40.
Xiaomi[xi] have-power greater-than 30.
Xiaomi[xi] have-cost equal-to 17000.
Xiaomi[xi] have-noiselevel equal-to 72.
Xiaomi[xi] have-weight equal-to 4.
References:
[xi] ('https://mi-shop.com/ru/catalog/smart_devices/vacuumcleaner/').
```

Рис. 13. Пример создания интернет-ссылки в онтологии

Fluent Editor предоставляет возможность ознакомиться со всеми ссылками в онтологии, используя функцию Reference Explorer (от англ. «проводник ссылок»). Данная функция находится в окне быстрого доступа во вкладке «Tools». При этом справа рядом с вкладкой «Taxonomy Tree» появляется вкладка Reference Explorer, где отражаются ссылки на использованные интернет-источники (рис. 14). Для отображения источников, необходимо нажать на стрелочку слева от слов «Reference Elements».

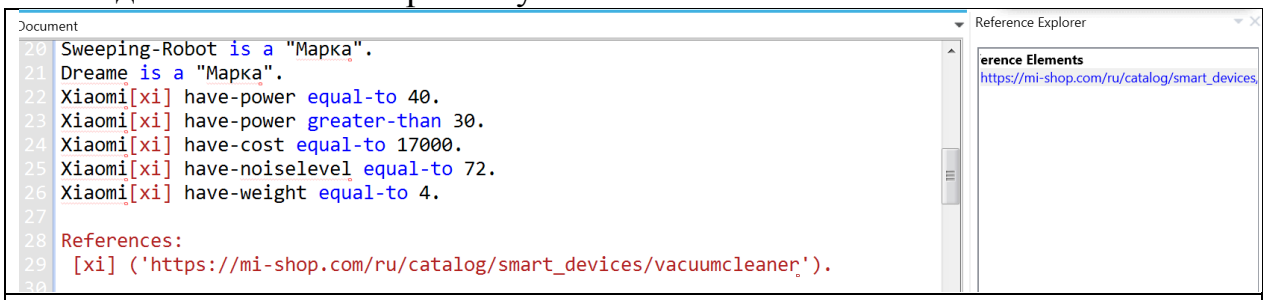


Рис. 14. Окно Reference Explorer

10. Задание

1. Создайте проект и сохраните его под именем «Project 1».
2. Озаглавьте проект. Для этого введите «Project_____» в строку «Title», а в следующей строке «Author» опишите всех авторов, выполняющих лабораторную работу.
3. Ознакомьтесь с таблицей-классификацией, выданной преподавателем.
4. Создайте несколько классов (3-4 класса)
5. Каждому классу присвойте несколько подклассов (3-4 подкласса).
6. При создании зависимостей класс-подкласс в онтологии используйте отношение «is a».
7. Добавьте подклассам экземпляры, выбранные из таблицы предложенной преподавателем.
8. Создайте отношения между экземплярами двух классов.
9. Присвойте значения характеристик каждому экземпляру одного из классов, используя логическое выражение «equal to».
10. Проверьте грамматику написанной онтологии.
11. Постройте CNL диаграмму онтологии, отражающую отношения между классами и экземплярами.
12. Сформулируйте 5 запросов в окне Reasoner, используя не повторяющиеся логические выражения.
13. Создайте Reference, присвоив нескольким экземплярам свой [id]. Ссылки на источники возьмите из сети Интернет.
14. Оформите отчет в соответствии и ответьте на контрольные вопросы.

11. Содержание отчета

Отчет по лабораторной работе №1 должен содержать:

1. Титульный лист.

2. Таблицу-классификацию.
3. Скриншот окна CNL.
4. Скриншот дерева таксономии.
5. Диаграмму CNL.
6. Скриншоты окна Reasoner с ответами на заданные вопросы к онтологии.
7. Скриншот окна Reference Explorer.
8. Вывод по проделанной работе.

12. Контрольные вопросы

1. Что такое онтология?
2. Что является результатом онтологического анализа?
3. Цель создания онтологии?
4. Что является главным отличием Fluent Editorот других редакторов онтологий?
5. Что такое контролируемые языки и для чего они необходимы?
6. Какой встроенный механизм использует Fluent Editor для отслеживания грамматики онтологии на CNL?
7. Каким образом осуществляется создание классов, подклассов, экземпляров в Fluent Editor?
8. Какие существуют правила написания имени класса, подкласса, экземпляра?
9. Для чего используется функция Reasoner?
10. Какие логические выражения используются для создания онтологий в Fluent Editor?
11. Что такое базы данных? Как используются онтологии для создании баз данных?

Лабораторная работа № 3

Продукционная модель представления знаний и ненадежные знания

Цель работы - изучение продукционной модели представления знаний (МПЗ) и механизмов вывода на данной модели, а также методов представления и использования ненадежных знаний.

Краткая теория

Для выполнения данной лабораторной работы необходимо изучить лекционный материал модуля 4.

1. Продукционные модели

Продукционная модель представления знаний - наиболее распространенная МПЗ в системах ИИ.

Основная структура в этой МПЗ - ***правило-продукция*** следующего вида:

"ЕСЛИ" <условия> "ТО" <заключение>

В данном выражении <условия> - это комбинация условий, соединенных операциями И/ИЛИ.

Например:

"ЕСЛИ"((A ИЛИ B) И (C ИЛИ D)) "ТО" (F).

Данное правило можно записать и с помощью логических операций &,∨,→

$$(A \vee B) \& (C \vee D) \rightarrow F$$

где A, B, C, D и F- факты.

Факты могут представляться разными способами, но чаще всего они представляются в виде тройки:

<объект - атрибут (параметр) - значение>.

Иногда факты представляются в виде двойки:

<атрибут - значение>

В этом случае предполагается, что все факты в правилах относятся к одному объекту.

Примеры правил-продукций:

1) правило из предметной области «Прогноз погоды»:

ЕСЛИ <сезон - зима> И (<давление - падает> ИЛИ <ветер-северный>) ТО < прогноз погоды - похолодание >;

2) правило из области ботаники:

ЕСЛИ <тип растения - деревья> И <форма листьев - иглоподобная>

ТО < класс растений - голосеменные >;

3) правило для системы «Служба знакомств»:

ЕСЛИ (<цвет волос - брюнет> ИЛИ <цвет волос - шатен>) И (<рост - высокий>) ИЛИ (<рост - средний>)ТО <внешность - подходит>;

4) правило для системы идентификации химического вещества утечки:

ЕСЛИ < растворимость вещества - нет > И < признак разлива - серебристая пленка >ТО < вещество утечки — нефтепродукт >.

Совокупность всех правил, описывающих некоторую предметную область, составляет ***базу правил (БП).***

Кроме базы правил в системах продукций имеется ***база фактов (БФ)*** или ***рабочая память (РП),*** в которой хранятся текущие факты.

Таким образом, системы продукций характеризуются четким разделением между фактами и правилами (в отличии от логических языков, где и то, и другое - равноправные формулы), что значительно облегчает процедуру вывода.

Сначала в базу фактов (БФ) помещаются исходные факты, описывающие некоторый конкретный объект (или некоторую текущую конкретную ситуацию), для которого (которой) требуется вывести решение. Затем, в процессе вывода база фактов пополняется новыми фактами, выводимыми в результате применения правил.

2. Вывод в продукционной системе

Структура продукционной системы приведена на рисунке 15 .

Рис 15. Структура продукционной системы

В машине вывода можно выделить:

- процедуры «элементарного вывода» (на одном правиле);
- систему управления выводом.

3. Логическое обоснование элементарного вывода

«Элементарный вывод» основывается на трех правилах логического вывода.

1. Правило «Введение конъюнкции»

$$A, B \Longrightarrow A\&B$$

Если два факта А и В одновременно истинны, то их конъюнкция истинна.

2. Правило «Введение дизъюнкции»

$$A \Longrightarrow A \vee B$$

Если факт А - истинен, то дизъюнкция А с любым другим фактом истинна.

3. Правило «Модус поненс»

$$P \rightarrow Q, P \Rightarrow Q$$

Если истинна продукция P → Q и факт P из ее условной части истинен, то и заключение Q - истинно.

Остается добавить, что факт считается истинным, если он содержится в базе фактов, и ложным, если его нет в БФ.

Итак, процедура «элементарного вывода» на одном правиле заключается в проверке «выполнимости» условий правила. Для этого проверяется, имеются ли в БФ факты, содержащиеся в левой части правила.

Для выполнимости совокупности фактов, соединенных через **«&» (И)**, необходимо, чтобы все они содержались в БФ.

Для выполнимости совокупности фактов, соединенных через **«V» (ИЛИ)** достаточно, чтобы хотя бы один из них содержался в БФ.

Если условия в левой части правила «выполнимы», то факт, находящийся в правой части, т.е. «заключение», считается выведенным.

Пример. Имеется правило:

П1: (сезон - зима) & ((давление - падает) v (ветер - северный)) →
(прогноз - похолодание).

И пусть в БФ содержатся следующие факты, описывающие некоторую конкретную ситуацию:

Ф1: (сезон - зима);

Ф2: (ветер - северный).

Сравниваем левую часть П1 (условия) с фактами из БФ.

Совокупность фактов из левой части П1:

(давление - падает) v (ветер - северный)

является выполнимой, т.к. один из них: (ветер - северный) имеется в БФ (Ф2). И вся совокупность фактов в условной части П1 выполнима:

(сезон - зима) & ((давление - падает) v (ветер - северный)).

Следовательно, заключение: (прогноз - похолодание) - выведено.

4. Система управления выводом

Продукции, в отличие от операторов в алгоритме, не «вызывают» непосредственно друг друга. Они совершенно независимы. Связь между правилами осуществляется только через базу фактов. БФ доступна для всех правил через машину вывода. Она выступает в качестве рабочей области, анализируемой и преобразуемой (пополняемой) системой.

Система управления выбирает очередное правило для исполнения. Выбор осуществляется на основе анализа базы фактов в соответствии с некоторой выбранной стратегией вывода. Кроме того, система управления

решает, как поступить, если на некотором шаге вывода возникает конфликтная ситуация -возможен выбор различных вариантов его решения (возможно применение нескольких правил-продукций из конфликтного множества). Вывод заканчивается, когда доказана истинность целевого факта.

5. Механизмы вывода

Для обработки знаний в продукционной модели разработано два алгоритма вывода:

- «Прямой вывод» или «От данных к цели»;
- «Обратный вывод» или «От цели к данным».

5.1. Механизм прямого вывода

Проверяются левые части правил на «выполнимость», т.е. сравниваются факты, содержащиеся в левой части правила, с базой фактов.

Если левая часть некоторого правила содержит факты, имеющиеся в БФ (для фактов, соединенных знаком «V», достаточно, чтобы один из них содержался в базе фактов), то факт, содержащийся в правой части правила, считается выведенным и добавляется к БФ. В противном случае правило отбрасывается и ищется другое «выполнимое» правило, которое пополнит БФ новым фактом.

Процедура повторяется, и теперь уже некоторые правила, которые ранее были отброшены, становятся выполнимыми на расширенной БФ. Таким образом, БФ все время пополняется новыми фактами. Процесс оканчивается, если в БФ поступает искомый факт, являющийся целью вывода, или когда в БП нет «выполнимых» правил.

Система построена так, что выбранное правило из базы правил выполняться будет только один раз. Оно как бы «выгорает».

Пример. Пусть имеется база правил:

П1: $N \& M \rightarrow P$

П2: $C \vee D \rightarrow M$

П3: $(A \& D) \vee C \rightarrow N$

П4: $A \& B \rightarrow D$

И пусть база фактов содержит факты A и B.

Цель вывода - факт P.

Начинаем перебирать правила по очереди и проверять левые части правил на «выполнимость». Правила П1, П2 и П3 отбрасываем, т.к. в левых частях этих правил стоят факты, не содержащиеся в БФ. Левая часть правила П4: A & B - выполнима, т.к. оба факта A и B содержатся в БФ. Следовательно, факт D из правой части П4 добавляется в базу фактов.

Начинаем перебирать сначала. П1 отбрасывается, а правило П2 теперь становится выполнимым, т.к. содержит факты C и D, соединенные «V», и один из них - D содержится в БФ. Факт M из правой части П2 добавляется в БФ.

Правило П3 также выполнимо. В БФ добавляется факт N. Теперь становится выполнимым и П1. В БФ поступает целевой факт P.

Рассмотрим случай, когда на каком-либо шаге прямого вывода появляется сразу несколько правил, которые могут быть «выполнены». Для разрешения конфликтов в продукционных системах используют стратегии «поиск в глубину» («сначала вглубь») и «поиск в ширину» («сначала вширь»).

Стратегия «поиск в глубину» («сначала вглубь»)

Данная стратегия характеризуется активизацией последнего правила, добавленного в конфликтное множество. Именно эта стратегия наиболее часто применяется в экспертных системах продукционного типа, т.к. она соответствует процессу рассуждений эксперта при решении той или иной задачи - он видит, что система стремится свести каждую цель к подцели.

Стратегия «поиск в ширину» («сначала вширь»)

Если активизировать на каждом шаге работы машины вывода первое правило конфликтного множества, то решение будет строиться методом поиска в ширину. В этом случае все альтернативные цепочки рассуждений, имеющиеся в решении, продолжаются параллельно, и ни одна не обгоняет другую.

Пример

Множество правил-продукций

1) A & B → H
2) C & D → I
3) E & F → K
4) G → L
5) O → L
6) H → M
7) I → M
8) K → N
9) L → N
10) M → Goal
11) N → Goal

База фактов: E, F, G

Цель вывода: Goal

Таблица 2.

Прямой вывод. Поиск в глубину

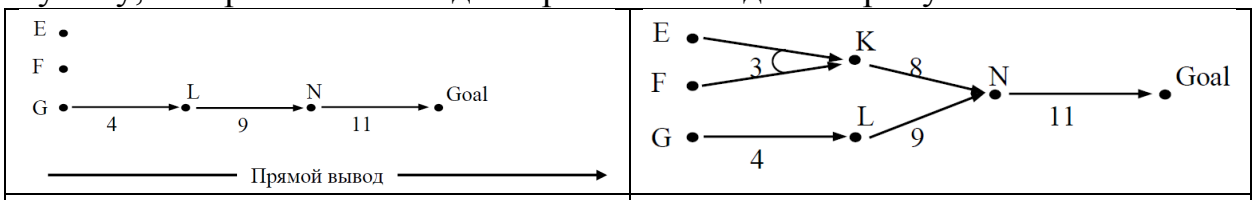
№ шага	База фактов	Конфликтное множество	Активизируемое правило
0	E, F, G	—	—
1	E, F, G	3, 4	4
2	E, F, G, L	3, 9	9
3	E, F, G, L, N	3, 11	11
4	E, F, G, L, N, Goal		Остановка

Таблица 2.

Прямой вывод. Поиск в ширину

№ шага	База фактов	Конфликтное множество	Активизируемое правило
0	E, F, G	—	—
1	E, F, G	3, 4	3
2	E, F,G, K	4, 8	4
3	E, F, G, K, L	8, 9	8
4	E, F, G, K, L, N	9, 11	9
5	E, F, G, K, L, N	11	11
6	E, F, G, K, L, N, Goal		Остановка

На рис. 16а изображен граф решения методом прямого вывода в глубину, а на рис.16б - методом прямого вывода в ширину.



а) прямой вывод в глубину | б) прямой в в ширинуывод

Рис.16. Граф решения

Вершины графа соответствуют высказываниям, а ребра - соответствующим правилам вывода. Процесс вывода начинается с размещения множества исходных фактов E, F, G в базе фактов и заканчивается, когда в базу фактов будет помещена целевой факт Goal. Направление вывода обозначено рядом с графиком и соответствует движению от стартовых к целевому факту, т.е. от исходных данных (высказывания E, F, G).

5.2. Механизм обратного вывода

При обратном выводе начинают с целевого факта, истинность которого требуется доказать.

Проверяются правые части правил на наличие в них целевого факта. Если находится такое правило, то осуществляется проверка, «выполнима» ли левая часть этого правила на базе фактов. Если левая часть правила «выполнима», т.е. содержит факты, имеющиеся в БФ, то цель считается подтвержденной (доказанной).

Если же в левой части правила имеются факты, не содержащиеся в БФ (и не соединенные знаком «v» с фактами из БФ), то они становятся новыми целями. Теперь пытаются доказать эти новые, промежуточные цели так же, как и исходную цель. Если они подтверждаются, то тем самым подтверждается и исходная цель.

При доказательстве какой-либо промежуточной цели могут появиться новые промежуточные цели и так далее.

Например, пусть исходной целью вывода является факт M и в БФ имеется правило:

P & Q & R → M

Если в базе фактов имеются факты P, Q, R, то цель M считается доказанной. Если же в БФ содержатся, например, только факты P и Q, то факт R становится новой целью. В базе фактов отыскивается правило, содержащее в правой части факт R.

Например:

(N V S) & T → R

и все действия повторяются.

Если же некоторый факт, выдвинутый в качестве цели, не подтверждается на базе фактов и не выводится (не содержится в правых частях правил), то цель отвергается.

По аналогии с алгоритмом прямого вывода для разрешения конфликта используются две разновидности стратегии «обратного вывода»:

«поиск в глубину» («сначала вглубь»)

«поиск в ширину» («сначала вширь»).

Стратегия «поиск в глубину» («сначала вглубь»)

Например, при доказательстве факта M рассматривается правило:

P V Q → M

и факты P и Q не содержатся в БФ.

При стратегии «сначала вглубь» в качестве новой цели сначала берется один из них (первый), а второй факт оставляется на потом (запоминается).

В данном случае факт P является новой целью. Далее доказывается выдвинутая цель до тех пор, пока не подтвердится или не будет отвергнута.

Затем возвращаются ко второму факту - Q. Если цель P не была подтверждена, то теперь уже факт Q выдвигается в качестве новой цели. Если P была подтверждена (=И), то нет смысла доказывать Q, т.к. P V Q = И.

Заметим, что если бы факты P и Q в правиле были соединены через «&»: P & Q → M, то обе цели P и Q должны подтвердиться.

Стратегия «поиск в ширину» («сначала вширь»)

Когда при выводе на каком-либо шаге появляется сразу несколько фактов, не содержащихся в БФ, которые могут быть выдвинуты в качестве новых целей, то при стратегии "сначала вширь" все эти факты становятся новыми целями одновременно. И затем одновременно осуществляется доказательство всех промежуточных целей.

6. Дерево вывода

Процесс вывода целевого факта механизмом обратной волны можно изображать в виде дерева вывода. Например, правило A & B → D представляется в виде дерева (рис.17):
Дуга между ветвями означает, что они соединены через И (&).

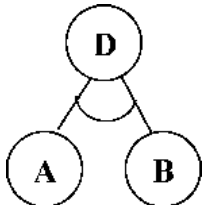


Рис.17.Дерево вывода

Если два правила содержат в правой части один и тот же факт, например:

П1: A & B → D

П2: C V F → D,

то данная ситуация соответствует наличию комбинированной связи (связь КОМБ) и это представляется так (рис.18):

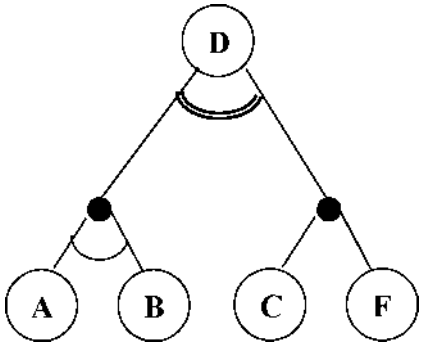


Рис.18. Связь КОМБ

Отсутствие дуги между ветвями означает, что они соединены через ИЛИ (V).

Двойная дуга между ветвями означает, что присутствует связь КОМБ.

Поясним смысл введенного понятия - комбинированная связь.

Если связь комбинированная, то на основании двух и более доказательств независимо подтверждается или опровергается (в случае противоречивых доказательств) одна и та же цель. Другими словами, комбинированная связь возникает тогда, когда в продукционной системе имеется несколько правил, приводящих к одному и тому же заключению.

Рассмотрим на примере, как строится "дерево вывода" (дерево И/ИЛИ/КОМБ).

Пример.

База правил:

П1: S & P → M

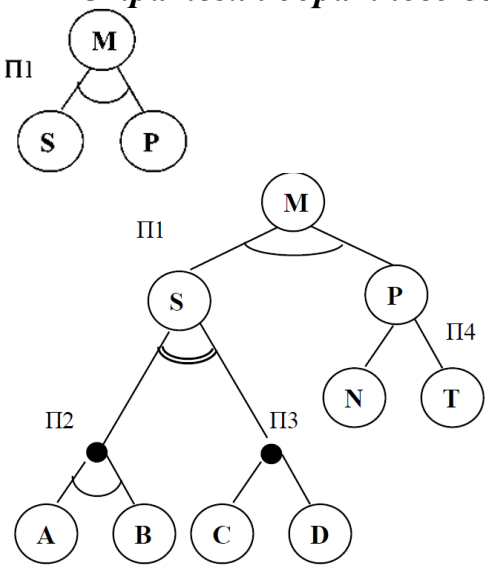
П2: A & B → S

П3: C v D → S

П4: N v T → P

База фактов: C, T Цель вывода: M

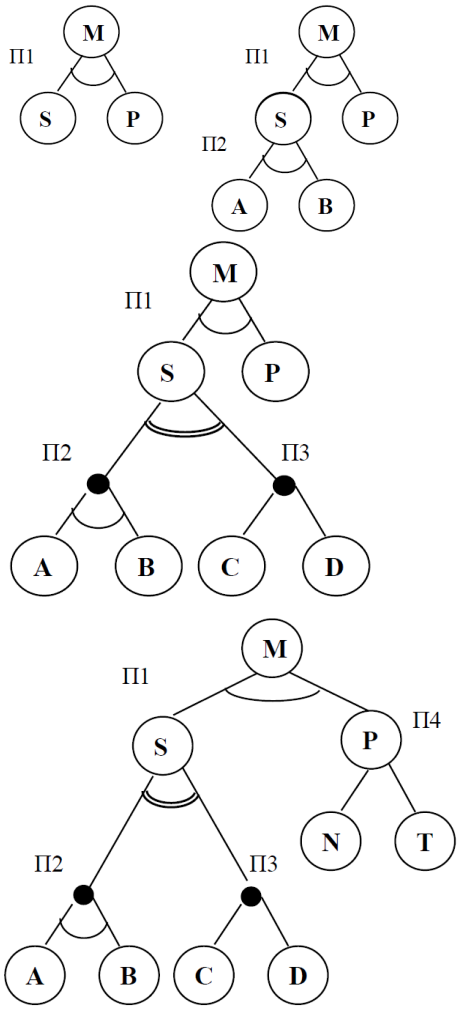
Стратегия обратного вывода «сначала вширь»



Корнем дерева является целевой факт M. Имеется одно правило с этим фактом в правой части - П1. В его левой части два факта - S и P. Включаем в список целей одновременно оба факта.

Ищем правила, содержащие факты S и P в правых частях. Это П2, П3 и П4. Рассматриваем условные части этих правил, т.е. факты: A, B, C, D, N и T. Обнаружив в БФ факт C, подтверждаем цель S через П3. И, обнаружив факт T, подтверждаем цель P через П4. Тем самым подтверждается и исходная цель M.

Стратегия обратного вывода «сначала вглубь»



Корнем дерева является целевой факт M. Ищем правило с этим фактом в правой части. Находим П1. В левой части правила два факта: S и P. Оба не содержатся в БФ. Выдвигаем первый из них (S) в качестве новой цели. Ищем правила с S в правой части. Таких правила два - П2 и П3. Берем одно (первое) - П2. В левой части правила два факта: A и B. Оба не содержатся в БФ и не выводятся. - значит эта ветвь - тупиковая.

Пытаемся опять доказать цель S. Берем второе правило - П3. В левой части правила два факта: C и D. Т.к. они соединены через «V» и факт C содержится в БФ, то цель S подтверждается через правило П3.

Возвращаемся к факту P и теперь его выдвигаем в качестве новой цели. Ищем правило с P в правой части. Это П4, содержащее в левой части факты N и T, соединенные «V». Т.к. факт T содержится в БФ, то цель P подтверждается через П4. Поскольку обе цели S и P подтверждены, тем самым и исходная цель - M считается подтвержденной (доказанной).

7. Ненадежные знания

В задачах, которые решают интеллектуальные системы, иногда приходится применять ненадежные знания и факты, представить которые двумя значениями - истина или ложь (1 или 0) трудно. Существуют знания, достоверность которых, скажем 0.7. Такую ненадежность в современной физике и технике представляют вероятностью, подчиняющейся законам Байеса (для удобства назовем ее байесовской вероятностью), но в инженерии знаний было бы нелогично иметь дело со степенью надежности, приписанной знаниям изначально, как с байесовской вероятностью (нелогично незнания представлять байесовской информацией). Для представления ненадежных знаний используют следующие методы:

- метод использования коэффициентов уверенности (CF) или метод MYCIN;
- субъективный байесовский метод;

- метод выводов на основе теории Демпстера-Шафера;
- нечеткую и вероятностную логику.

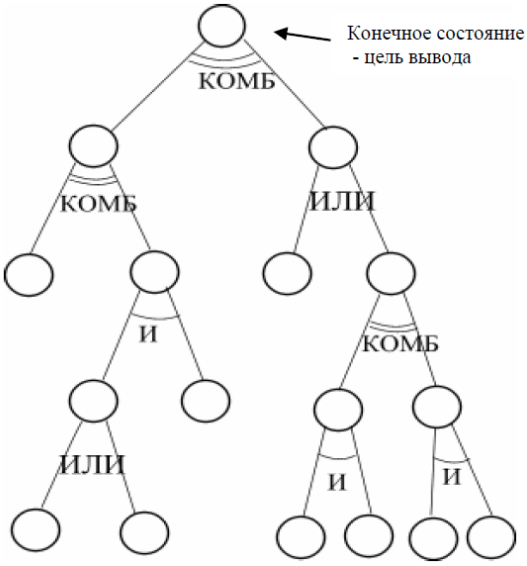


Рис. 19 – Описание дерева И/ИЛИ/КОМБ для задачи с ненадежными данными

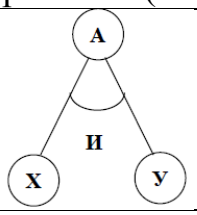
Одним из первых был разработан метод использования ***коэффициентов уверенности*** для системы MYCIN (известная экспертная система по идентификации микроорганизмов в крови). Этот метод не имеет теоретического подкрепления, но стал примером обработки ненадежных знаний. Данный метод представления ненадежных знаний и будем изучать в настоящей работе.

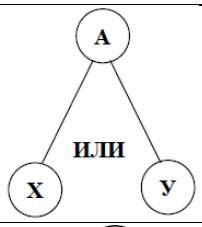
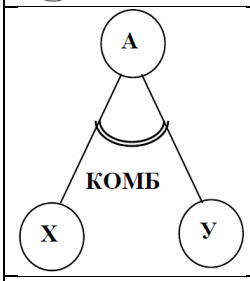
Для решения сложных задач используют метод разбиения их на несколько подзадач. Каждая подзадача в свою очередь разбивается на простые подзадачи, поэтому задача в целом описывается иерархически. Знания, которые по условиям подзадач определяют условия задач высшего уровня, накапливаются фрагментарно. При разбиении на подзадачи возможно соединение И, ИЛИ и КОМБ (комбинированная связь). На рис.19 показано описание задачи в виде дерева И, ИЛИ, КОМБ.

Поскольку эвристические правила ЕСЛИ-ТО основываются исключительно на человеческом опыте, с полной определенностью никогда нельзя сказать, что они верны. Пользователь интеллектуальной системы также не может быть полностью уверен, что значения, которые он приписывает переменным, абсолютно корректны. Рассмотрим представление ненадежных знаний, которые могут иметь степень надежности (коэффициент уверенности), равную не только 1, но и значения между истиной и ложью.

Будем использовать следующую запись правил-продукций, включающих соединения И, ИЛИ и комбинированную связь (рис.20).

Здесь X,Y - условия, предпосылки выполнения правила (левая часть правила); A - заключение правила (правая часть правила, цель); И, ИЛИ, КОМБ - виды связей; C_1, C_2, C_{31}, C_{32}, - это коэффициенты уверенности, приписанные правилам (знаниям).

	Правило: ЕСЛИ X и Y, то A с уверенностью C_1 $(X \,\&\, Y) \rightarrow A(CF_{id} = N_1)$

	Правило: ЕСЛИ X или Y, то A с уверенностью C_2 $(X \vee Y) \rightarrow A(CF_{id} = N_2)$
	Правило 1: Если X, то A с уверенностью C_{31} $(X \rightarrow A(CF_{id} = N_{31})$ Правило 2: Если Y, то A с уверенностью C_{32} $(Y \rightarrow A(CF_{id} = N_{31})$
Рис. 21. Описание правил с помощью связей И, ИЛИ, КОМБ	

Пример 1

Рассмотрим случай определения (диагностирования), простужен ли больной. Пусть предпосылка 1 - кашель у больного - надежна со степенью 0,6 (CF=0,6), а предпосылка 2 - температура 39-40 - надежна со степенью только 0,5 (CF=0,5).

Простудное состояние при наблюдении только одной из предпосылок простуды можно подтвердить с надежностью 0.6 или 0.5. Но если рассмотреть обе предпосылки, то естественно, нужно считать, что простуда куда более достоверна. Если выполнить операцию КОМБ, то должны получить, скажем, надежность 0.8.

И наоборот, пусть предпосылка 1 - наличие кашля - снова имеет надежность 0.6, но температура как предпосылка 2 нормальная, тогда при операции комбинированной связи надежность диагноза простуды уменьшится и будет равна, например, 0.45.

Если выбрать метод вывода как для связи И, ИЛИ, так и для связи КОМБ, то степени надежности можно распространить и на иерархическую сеть выводов. В итоге можно получить степень надежности конечной цели, а также указать ее при окончательном ответе.

Допустим, что для нашей задачи уже определены степени надежности фактов X и Y как результатов предыдущих выводов или наблюдений, и сделаем вывод или вычислим степень надежности факта A, используя правила из базы знаний и метод MYCIN.

Метод MYCIN

В методе MYCIN имеют дело с ненадежностью, представленной ***коэффициентом уверенности CF***. Этот коэффициент принимает значения в отрезке [-1, 1] (1 - заведомо истинно, -1 - заведомо ложно).

При выводе, прежде всего, получают коэффициент уверенности условной части правила - CF_{id} следующим образом:

1. Если в предпосылке только один факт X, то C_1, C_2, C_{31}, C_{32}
$$CF_{id} = \text{CF[X]}$$
2. При связи И
$$CF_{id} = \text{CF[X И Y]} = \min\{\text{CF[X]}, \text{CF[Y]}\}$$

3. При связи ИЛИ

$$CF_{id} = \text{CF[X ИЛИ Y]} = \max\{\text{CF[X]}, \text{CF[Y]}\}$$

Далее определяют значение коэффициента уверенности заключения правила, т.е. факта А. Для этого вычисляют произведение коэффициента уверенности правила и полученного коэффициента уверенности предпосылки правила:

$$\text{CF}_{\text{заключения}}[\text{A}, (\text{X}, \text{У})] = \text{CF}_{\text{правила}} * \text{CF}_{\text{id}}$$

При связи КОМБ отдельно получают коэффициенты уверенности заключения каждого правила, т.е. CF[A, X] и CF[A, Y] по выше описанным формулам, а результирующий коэффициент уверенности заключения А определяют следующим образом:

$$CF\Big(A,(X,Y)\Big) = \begin{cases} 1, \quad \text{если} \quad CF(A,X) = 1 \ \text{или} \quad CF(A,Y) = 1; \\ CF(A,X) + CF(A,Y) - CF(A,X)\cdot CF(A,Y), \quad \text{если} \quad CF(A,X) > 0 \ \text{и} \ CF(A,Y) > 0; \\ CF(A,X) + CF(A,Y), \quad \text{если} \quad CF(A,X)\cdot CF(A,Y) \le 0, \\ \qquad\qquad \text{или} \quad CF(A,X) \ne \pm 1 \ \text{и} \ CF(A,Y) \ne \pm 1; \\ CF(A,X) + CF(A,Y) + CF(A,X)\cdot CF(A,Y), \quad \text{если} \quad CF(A,X) \le 0 \ \text{и} \ CF(A,Y) < 0; \\ -1, \ \text{если} \ CF(A,X) = -1 \quad \text{или} \quad CF(A,Y) = -1. \end{cases}$$

Коэффициент уверенности, полученный из трех и более независимых доказательств, можно вывести, последовательно используя указанные выше формулы.

Приведем примеры расчета коэффициента уверенности.

Пример 2

Правило:

ЕСЛИ Процентные ставки = падают и налоги уменьшаются, ТО уровень цен на бирже = РАСТЕТ

верно не всегда, поэтому можно приписать ему значение некоторого коэффициента уверенности.

Пусть приведенное правило имеет CF, равный 0.8, и нельзя утверждать, что процентные ставки падают, т.е. первому условию правила назначен CF, равной 0,5. Кроме того, допустим, что налоги колеблются (то увеличиваются, то уменьшаются), поэтому предположить уменьшение налогов можно, только если CF равен 0,7. Тогда правило можно записать так:

ЕСЛИ Процентные ставки = падают (CF = 0.5) И

Налоги уменьшаются (CF = 0.7),

ТО Уровень цен на бирже = РАСТЕТ $\left(\text{CF}_{\text{правила}}=0.8\right)$.

Коэффициент уверенности, что уровень цен на бирже будет расти может быть подсчитан следующим образом: выбирается минимальный CF для условной части правила, разделенных логическим оператором И, и умножается на CF для всего правила:

$$CF_{заключения} = (minimum(0.5; 0.7)) * 0.8 = 0.5 * 0.8 = 0.4$$

Следовательно, при CF = 0,4 можно сказать, что уровень цен на бирже будет падать.

Если есть еще одно правило с тем же логическим выводом о росте уровня цен на бирже, но другим наборам условий, то CF для этого вывода необходимо провести по формулам MYCIN.

Пример 3

Рассмотрим два правила с одним и тем же логическим выводом С:
ЕСЛИ A (CF = 0,3) И B (CF = 0,6) ТО C ($CF_{правила}$ = 0,5)
ЕСЛИ D (CF = 0,4) И E (CF = 0,7) ТО C ($CF_{правила}$ = 0,9).
Для первого правила:
$CF_{предпосылки}$ = (minimum(0,3; 0,3)) * 0,5 = 0,3 * 0,5 = 0,15
Для второго правила:
$CF_{предпосылки}$ = (minimum(0,4; 0,7)) * 0,9 = 0,4 * 0,9 = 0,36
Тогда $CF_{заключения}$ = 0,15 + 0,36 – 0,15 * 0,36 = 0.46

Пример 4

Возьмем пример с использованием логического оператора ИЛИ:
ЕСЛИ A(CF = 0,3) И B(CF = 0,6) ИЛИ D(CF = 0,5)
ТО C($CF_{правила}$ = 0,4).
В этом примере CF для логического вывода С считается так:
$CF_{заключения}$ = Maximum(minimum(0,3; 0,6); 0,5) * 0,4 =
= maximum(0,3; 0,5) * 0,4 = 0,5*0,45=0.2

Задание

Реализовать продукционную систему со следующей структурой:

- база правил: область памяти, которая содержит базу знаний – совокупность знаний, представленных в форме правил вида «ЕСЛИ-ТО»;
- глобальная база данных – область памяти, содержащая факты, которые описывают вводимые данные и состояния системы;
- интерпретатор правил (механизм логического вывода) – компонент системы, который формирует заключения, используя базу правил и базу данных.

Для синтаксического представления продукций в модели использовать язык исчисления предикатов первого порядка, то есть основными формализмами представления продукций должны быть:
а) терм, устанавливающий соответствие знаковых символов описываемому объекту;
б) предикат для описания отношения сущностей в виде реляционной формулы, содержащей в себе термы.
Термы должны быть двух видов: терм-переменная и терм-константа.

В записи условной части должна быть предусмотрена возможность наличия логических связок «И» и «ИЛИ».

Требования к системе, реализующей продукционную модель представления знаний:

а) наличие механизма заполнения базы правил и глобальной базы данных, а также отображение результата логического вывода (интерфейс с пользователем);

б) механизм логического вывода выбрать в зависимости от номера варианта в *таблице 1*:

- нечетные номера – прямой вывод;
- четные номера – обратный вывод;

в) представление системы продукций графом «И/ИЛИ»;

г) предусмотреть механизм разрешения конфликта на этапе вывода;

д) обнаружение ошибочных правил в случае, когда либо доказательство заключения закончилось неуспехом, либо получено неверное заключение;

е) протоколирование поиска на графе: описать процесс вывода в системе, используя частичный граф; в случае неуспешного вывода указать невыполнимое условие.

Таблица 1 – Варианты предметных областей для разработки экспертной системы

№№ вар	Предметная область	№№ вар	Предметная область
1	Выбор профессии	6	Оценка стоимости разработки WEB-страниц
2	Неисправности ПЭВМ	7	Оценка стоимости разработки системы освещения в умном доме
3	Компьютерные вирусы	8	Оценка стоимости системы видеонаблюдения в умном доме
4	Оценка технического состояния жилого здания	9	Конфигурация системы управления в умном доме
5	Датчики для системы управления умным домом	10	Управление микроклиматом в жилом помещении

Содержание отчёта по работе

В отчете должны быть представлены:

- конфигурация системы продукций;
- описание способа организации поиска на графе;
- описание конфликтного набора и алгоритм разрешения конфликта при логическом выводе;
- тестовый набор правил – база правил и база данных;
- протокол тестового вычисления: обход по графу.

Лабораторная работа № 4

Разработка базы знаний экспертной системы на основе байесовской стратегии логического вывода

Цель работы: познакомиться с оболочкой ЭС «Малая экспертная система, v.2.0» и разработать базу знаний для этой экспертной системы.

Задачи работы:

1. Изучить описание программы, используя встроенную систему помощи и законспектировать основные положения.
2. Изучить на приведенном в задании примере последовательность действий при работе с ЭС.
3. Выписать данные своего варианта задания. Использую редактор баз знаний, создать базу знаний.
4. Запустить программу, загрузить и испытать работоспособность созданной экспертной системы для двух вариантов ответа: коэффициента уверенности и вероятности истинности свидетельства. Устранить обнаруженные ошибки.

4.1. Изучение принципа работы экспертной оболочки Малая экспертная система, v.2.0 (Mini Expert System).

Программа Mini Expert System представляет собой простую экспертную систему, использующую байесовскую систему логического вывода. Она предназначена для проведения консультации с пользователем в какой-либо прикладной области (на которую настроена загруженная база знаний) с целью определения вероятностей возможных исходов и использует для этого оценку правдоподобности некоторых предпосылок, получаемую от пользователя.

Запускающим файлом программы является MiniES.exe. После запуска появляется диалоговое окно (рис.22).

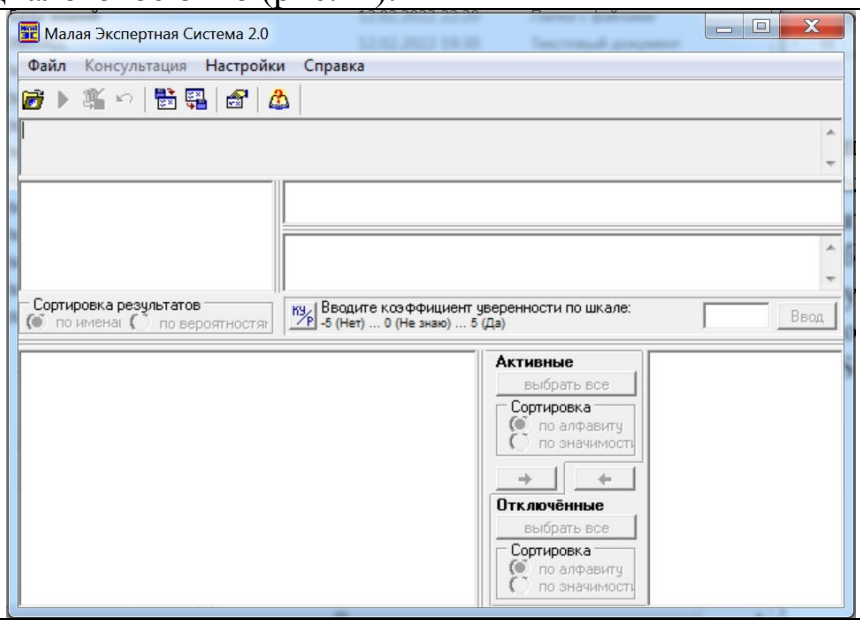
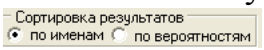


Рис.22. Диалоговое окно программы

Для начала работы с готовой базой знаний необходимо загрузить из файла базу знаний, содержащую информацию о той прикладной области, в которой хотим получить консультацию. Это можно сделать, нажав кнопку «Загрузить базу знаний из файла» , либо с помощью одноимённого пункта меню «Файл» (клавиши <F2>). Например, из папки «Базы знаний» загрузите файл Mushrooms.mkb, который является демонстрационной базой знаний о грибах. При этом в верхней левой части окна появляется список исходов (элементов) базы знаний (в данном случае, лисичка, опенок и т.д.) с указанием текущих значений вероятностей. Ширину этой области можно увеличить или уменьшить, передвинув разделитель, отделяющий её от области обработанных свидетельств.

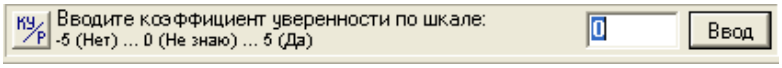
Для более удобного представления список элементов можно упорядочить либо по названиям исходов (в алфавитном порядке), либо по значениям текущих вероятностей. Это делается с помощью радиокнопок 

Ниже списка элементов перечислены вопросы, заложенные экспертом, с помощью которых ЭС определит, о каком исходе идет речь. После загрузки базы, можно начать работу, нажав кнопку «Начать консультацию с экспертной системой» (клавиша <F3> или пункт меню «Консультация / Начать консультацию»). Целью консультации является определение вероятностей возможных исходов. После начала консультации в правой части окна (область запросов) появляется первый запрос системы (название свидетельства, степень истинности которого система желает узнать).

В данной версии на запрос системы можно использовать два варианта ответа. Во-первых, можно задать по некоторой шкале коэффициент уверенности (например, от –5, что может означать «точно нет», до +5 – «точно да»). Во-вторых, пользователь может ввести вероятность истинности свидетельства (число от нуля до единицы). В обоих случаях он может выбирать любые промежуточные значения. Переключение между вариантами ответа осуществляется с помощью кнопки, расположенной слева от приглашения на ввод ответа (либо клавишей <F8>).

Варианты различаются следующим образом: значение коэффициента уверенности (КУ) выбирается практически интуитивно, в то время как вероятность истинности может быть получена из опытов или вычислена математически. В случае выбора коэффициента уверенности, имеется возможность ответить «Не знаю» (ввод числа, соответствующего середине шкалы, например, ноль, если шкала от –5 до +5). Такой ответ никак не повлияет на результат консультации. Во втором случае, при вводе вероятности этой возможности нет, т.к. значение вероятности истинности свидетельства, соответствующее ответу «Не знаю» (т.е. неизменным вероятностям исходов), для каждого исхода своё. Это очень важное различие между двумя способами ответа.

С помощью кнопки КУ/Р нужно выбрать сначала работу с коэффициентом уверенности. При этом справа от кнопки появится текст, показанный ниже.



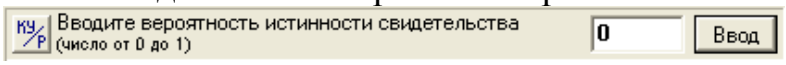
В окно необходимо ввести какое-либо значение коэффициента уверенности в диапазоне от -5 до +5). Диапазон изменения КУ может быть изменен в окне настройки предпочтений.

Вводите +5, если твёрдо уверены в истинности, и –5, если уверены в ложности свидетельства. Если затрудняетесь ответить, вводите 0, и этот вопрос никак не повлияет на результат консультации. Можно также вводить любые промежуточные значения. Например, 4.5, если «почти» уверены в положительном ответе.

Влияние ответа на результаты консультации тем значительнее, чем точнее был ответ (т.е. чем ближе значение, введённое пользователем, к одной из границ диапазона изменения КУ). Оценка с помощью КУ является адекватным отражением уверенности пользователя в истинности свидетельства, и любые промежуточные значения важны для получения правильных результатов.

После нажатия клавиши <Enter> (или кнопки «Ввод» справа от окна ввода ответа) обработанное свидетельство помещается в список, расположенный выше области запроса, и выделяется серым цветом. А слева в списке исходов (элементов) изменяются их вероятности. Отменяя ответы, введите ряд коэффициентов уверенности. Чтобы отменить выбранный ответ, нужно выделить любые свидетельства в верхнем списке и нажать кнопку «Отменить выделенные ответы» (пункт меню «Консультация | Отменить выделенные ответы» или сочетание клавиш <Ctrl + Z>).

С помощью кнопки КУ/Р можно переключиться на введение вероятности истинности свидетельства. При этом справа изменится подсказка.



Вероятность истинности свидетельства может быть получена из таблиц, по результатам статистических исследований или вычислена математически. Пользователь может просто строить предположения о её значении (в этом случае, возможно, более оправданным будет использование коэффициента уверенности). Вероятности исходов могут быть выражены в процентах.

Введите ряд значений вероятности события, каждый раз отменяя предыдущее значение. Получая от пользователя ответы, система корректирует вероятности возможных исходов, которые отражаются в левой части верхней половины окна.

Если часть вопросов решено не использовать при проведении консультации, то их выделяют в левой части окна, и нажав кнопку →, перемещают в правую часть окна. Кнопкой ← их можно вернуть обратно.

Консультацию можно прекратить, нажав кнопку «Возврат к исходным вероятностям результатов» ⊘, на которую заменяется кнопка «Начать консультацию», либо выбрав соответствующий пункт меню «Консультация» (клавиша <F3>). При этом происходит возврат к начальным значениям вероятностей исходов.

Имеется возможность следить за вероятностью конкретного исхода, если выделить его в списке – теперь он всегда будет виден в окне (при этом результаты должны быть упорядочены в алфавитном порядке). Если же результаты упорядочены по вероятностям, то можно выбрать нижнюю строку списка, чтобы в поле зрения всегда был наименее вероятный исход.

После того, как будет обработано последнее свидетельство, система подведёт итог (выдаст число обработанных свидетельств), а вероятности исходов в списке результатов примут окончательные значения. Теперь можно сделать вывод о возможности наступления интересующего исхода или просто прочесть название наиболее вероятного из возможных исходов.

Весь ход консультации можно сохранить в протоколе в виде текстового файла. Для этого нужно щелкнуть кнопку «Сохранить протокол консультации в файле»  . Сохраните протокол консультации, проведенной на готовой базе знаний.

Загрузите остальные файлы баз знаний, содержащиеся в папке «Базы знаний». Проведите консультации с ними. Сохраните результаты консультаций в протоколах.

4.2. Разработка базы знаний с готовыми вопросами.

Создадим базу знаний, используя вопросы о грибах:

- Гриб пластинчатый?
- Гриб полностью жёлтый?
- У гриба прочная толстая ножка?
- Гриб полностью ярко-коричневый?
- Шляпка красная?
- Есть ли венчик на ножке?

Для создания базы знаний необходимо открыть стандартную программу операционной системы Windows «Блокнот» и оформить там исходную информацию в виде текстового файла с расширением .mkb со следующей структурой:

Первая секция включает описание базы знаний, имя автора, комментарий и т.д. (можно в несколько строк). Данная секция заканчивается после первой пустой строки.

Вторая секция содержит формулировку вопросов:

Вопрос № 0 (любой текст, заканчивающийся переносом строки)
Вопрос № 1
Вопрос № 2
…….
Вопрос № N

После последнего вопроса следует одна пустая строка, и вторая секция заканчивается. В последней секции перечисляются исходы и соответствующие им элементы матриц вероятностей. Каждый исход задаётся в отдельной строке.

Исход № 0, P [, i, Py, Pn]

Исход № 1, P [, i, Py, Pn]
.....

Перечисление заканчивается с концом файла.

В результате должен получиться следующий текст (он выделен курсивом):

Пример базы знаний о грибах.
Автор Иванов Иван
Признаки:
Гриб пластинчатый?
Гриб полностью жёлтый?
У гриба прочная толстая ножка?
Гриб полностью ярко-коричневый?
Шляпка красная?
Есть ли венчик на ножке?
Лисичка,0.05, 1,1,0.5, 2,0.9,0.1, 3,0.03,0.3, 4,0,0.1, 5,0,0.25, 6,0,0.35
Свинух,0.05, 1,1,0.5, 2,0.005,0.2, 4,1,0.01, 5,0,0.25, 6,0,0.35
Подберёзовик,0.05, 1,0,0.5, 2,0,0.2, 3,0.25,0.3, 4,0.1,0.1, 5,0,0.25, 6,0,0.35
Подосиновик,0.05, 1,0,0.5, 2,0,0.2, 3,0.98,0.15, 4,0,0.1, 5,0.8,0.22, 6,0,0.35
Белый Гриб,0.05, 1,0,0.5, 2,0,0.2, 3,1,0.15, 4,0.001,0.1, 5,0.05,0.5, 6,0,0.35
Опёнок,0.05, 1,1,0.5, 2,0.7,0.15, 3,0,0.3, 4,0.15,0.1, 5,0,0.25
Мухомор Красный,0.05, 1,1,0.5, 2,0,0.2, 3,0.12,0.3, 4,0,0.1, 5,1,0.2, 6,1,0.3
Сыроежка,0.1, 1,1,0.5, 2,0.02,0.2, 3,0.1,0.3, 4,0,0.1, 5,0.3,0.2, 6,0,0.35
Маслёнок,0.05, 1,0,0.5, 2,0.25,0.2, 3,0.02,0.3, 4,0.05,0.1, 5,0.001,0.25, 6,0.7,0.3

После набора текста в заданном формате файл сохраняется с расширением .mkb. Если необходимая структура была набрана правильно, это файл будут «виден» в экспертной оболочке Mini Expert System, где с ним можно работать.

Проанализируем информацию, записанную в третьей секции. В начале каждой строки перечисляются возможный исход, вероятность которого меняется в соответствии с ответами на заданные вопросы. Исходом является текст, включающий любые символы, кроме запятых и точек.

После запятой указываются элементы матрицы вероятностей.

Первая цифра представляет собой априорную вероятность данного исхода Pi. Например, первая цифра после слова Лисичка – 0.05. Это означает, априорная вероятность данного исхода P = 0,05, то есть любой наугад взятый гриб является лисичкой с вероятностью 5%.

После этого через запятую идёт ряд повторяющихся полей из трёх элементов. Первый элемент i – номер соответствующего вопроса. Следующие два элемента Pyij и Pnij – соответственно, вероятности получения ответа «Да» на этот вопрос, если возможный исход верен и неверен. Эти данные указываются для каждого вопроса, связанного с данным исходом.

Например, первому вопросу (i = 1) соответствует запись «1,1,0.5», где апостериорные вероятности Pyi1 = 1 и Pni1 = 0,5. Первая цифра (Pyi1 = 1) означает, что если гриб является лисичкой, то этот признак обязательно должен

присутствовать и 100% пользователей ответят «Да» на этот вопрос. Соответствующий признак (пластинчатый гриб) может иметь место, если гриб не является лисичкой, поэтому половина пользователей ответят «Да». Ответ «Нет» предполагает, что рассматриваемый гриб не лисичка.

Примечание: P < = 0.00001 считается равной нулю, а P > = 0.99999 – единице, поэтому не следует указывать такие значения – исход с подобной априорной вероятностью обрабатываться не будет. Не обязательно задавать значения вероятностей для всех вопросов. Те вопросы, которые не относятся к данному исходу, могут не обрабатываться.

4.3. Расчет апостериорных вероятностей по теореме Байеса.

При создании новой экспертной системы знания о рассматриваемой области формулируются в виде двух наборов: Q = {qj} – набор вопросов (симптомов, свидетельств) и V = {vi} – набор вариантов исхода (вариантов решения), а также двух матриц вероятностей:

$P_y = \{py_{ij}\}$ и $P_n = \{pn_{ij}\}$ размером m*n, где py_{ij} – вероятность получения положительного ответа на j-й вопрос, если i-й исход верен; pn_{ij} – вероятность получения отрицательного ответа на j-й вопрос, если i-й исход верен; n и m – количества вопросов и исходов, соответственно. Кроме того, каждому исходу ставится в соответствие априорная вероятность данного исхода P_i, то есть вероятность исхода в случае отсутствия дополнительной информации.

При этом вначале априорная и апостериорные вероятности задаются эмпирически экспертом, создающим экспертную систему, исходя из его знаний предмета.

Однако если при заданных вероятностях экспертная система дает недостаточно корректные результаты, то априорные вероятности меняются экспертом, а апостериорные – рассчитываются по теореме Байеса по алгоритму, описанному ниже.

В процессе работы ЭС решатель, пользуясь данными наборами и матрицами, а также теоремой Байеса, определяет апостериорную вероятность каждого исхода, то есть вероятность, скорректированную в соответствии с ответом пользователя на каждый вопрос:

при положительном ответе $P_2 = \frac{Py_{ij} \cdot P_i}{Py_{ij} \cdot P_i + Pn_{ij} \cdot (1 - P_i)}$.

при отрицательном ответе $P_2 = \frac{(1 - Py_{ij}) \cdot P_i}{(1 - Py_{ij}) \cdot P_i + (1 - Pn_{ij}) \cdot (1 - P_i)}$.

при ответе «не знаю» апостериорная вероятность равна априорной.

Рассмотрим использование этих формул для расчета апостериорных вероятностей, соответствующих первому вопросу (i = 1) с записью «1,1,0.5»,

При положительном ответе «Да» (+5) на первый вопрос апостериорная вероятность для рассматриваемого примера составит:

$$P_2 = \frac{Py_{ij} \cdot P_i}{Py_{ij} \cdot P_i + Pn_{ij} \cdot (1 - P_i)} = \frac{1 \cdot 0{,}05}{1 \cdot 0{,}05 + 0{,}5 \cdot (1 - 0{,}05)} = 0{,}095238$$

При отрицательном ответе «Нет» (– 5) на первый вопрос апостериорная вероятность для рассматриваемого примера составит:

$$P_2 = \frac{(1 - Py_{ij}) \cdot P_i}{(1 - Py_{ij}) \cdot P_i + (1 - Pn_{ij}) \cdot (1 - P_i)} = \frac{(1 - 1) \cdot 0{,}05}{(1 - 1) \cdot 0{,}05 + (1 - 0{,}5) \cdot (1 - 1)} = 0 \cdot$$

При ответе «Не знаю» (0) апостериорная вероятность исхода равна априорной: $P_2 = P_i$

При промежуточном ответе h (от – 5 до 0 и от 0 до +5) апостериорная вероятность рассчитывается с учетом степени уверенности принадлежности признака и рассчитывается линейной интерполяцией от значений утвердительных ответов «Да», «Нет», «Не знаю».

При отрицательном ответе (– 5;0): $P_2 = P_i + (P_i - P_2(E)) \cdot \frac{h}{5}$

Например, при ответе $h = -3$: $P_2 = 0{,}05 + (0{,}05 - 0) \cdot \frac{-3}{5} = 0{,}02$

При положительном ответе (0;+5): $P_2 = P_i + (P_2(E) - P_i) \cdot \frac{h}{5}$

Например, при ответе h = +3: $P_2 = 0{,}05 + (0{,}095238 - 0{,}05) \cdot \frac{3}{5} = 0{,}075$

При разработке базы знаний нужно соблюдать следующие правила:

1. вероятность P = 0 указывать нельзя, т.к. теряется смысл опроса по данному исходу.

2. Недопустимо указывать, что P_y и P_n равны друг другу, т.е. нельзя записывать, например, 1, 0.4,0.4 так как это означает, что данное свидетельство не влияет на вероятность исхода, т.е. бессмысленно его упоминать.

3. Допустимым для P_y и P_n считается использование 3-4 знаков после запятой.

4. Вероятность P_y или P_n должна содержать десятичную точку, а не запятую (т.е. 0.01, а не 0,01), так как запятая служит для разделения групп друг от друга (дополнительно группы разделяются пробелами для удобства чтения).

5. В БЗ запись правила для каждого исхода должна располагаться на одной строке, перенос на другую строку считается концом исхода.

6. Допускается отвечать не на все вопросы, сохраняя их нумерацию.

7. Порядок вопросов, на которые составляются ответы, может быть произвольным.

8. Если сумма априорных вероятностей P(H) всех исходов равна 1, то в базе знаний приведены все возможные исходы.

9. Если сумма всех P(H) меньше 1, т.е не все возможные исходы известны эксперту (например, нельзя перечислить все характеристики грибов, которые могут полностью характеризовать каждый вид гриба), то базу знаний следует создавать по другому принципу. В этом случае априорные вероятности исходов P(H) находятся путём статистических исследований, а их сумма будет меньше единицы (невыполнение этого условия не принципиально важно, просто результаты станут менее надёжными). Значения P_y и P_n также берутся из статистики (или указываются примерные значения, кажущиеся правдоподобными эксперту), т.к. вычислить их невозможно.

10. При большом количестве вопросов, не следует указывать их все в каждом правиле. Во-первых, это лишняя работа, а во-вторых, среди свидетельств могут оказаться не влияющие на вероятность данного исхода. Например, вопрос можно ли грибы жарить важен при оценке вероятности выбора кулинарного рецепта, но бесполезен при распознавании вида гриба.

11. Чем ближе к 1 будет значение P_y и ближе к 0 – значение P_n, тем с большей вероятностью будет распознаваться соответствующий исход.

12. Если при составлении БЗ допущены ошибки, то при её запуске появится сообщение об этом с указанием места ошибки - номера строки и номера позиции в строке.

4.4. Задание на лабораторную работу

Требуется разработать базу знаний классифицирующей экспертной системы на основании предложенных вариантов:

№№ варианта	Задание
1	Тип используемого монитора
2	Тип внешнего запоминающего устройства
3	Тип электронного гаджета
4	Тип компьютерной игры
5	Тип информационной системы
6	Тип датчика для определения присутствия человека в помещении
7	Тип датчика сигнализирующего о пожаре
8	Тип печатающего устройства
9	Тип устройства ввода для ПЭВМ
10	Вид подсистемы в умном доме

Вы можете выбрать вариант самостоятельно из любой предметной области.

Оформление отчета

Отчет должен содержать

- титульный лист,
- цели и задачи работы;
- краткие теоретические сведения;
- результаты выполнения разделов работы, включая оконные формы;
- выводы.
-

Контрольные вопросы

1. Назначение ЭС .
2. Порядок работы с ЭС .
3. Формат базы знаний ЭС.
4. Метод логического вывода .
5. Понятие априорной и апостериорной вероятностей.

6. Понятие коэффициента уверенности.
7. Достоинства и недостатки данной системы логического вывода.

Лабораторная работа № 5

Алгоритм горной кластеризации

Цель работы

Разработка программы для реализации алгоритма горной кластеризации.

Краткая теория.

Кластеризация - это объединение объектов в группы (кластеры) на основе схожести признаков для объектов одной группы и отличий между группами. Большинство алгоритмов кластеризации не опираются на традиционные для статистических методов допущения; они могут использоваться в условиях почти полного отсутствия информации о законах распределения данных.

Пусть имеется множество произвольных объектов $\mathbf{x_i}$. Задача кластеризации состоит в разбиении объектов из множества $\mathbf{x_i}$ на несколько подмножеств (кластеров), в которых объекты более схожи между собой, чем с объектами из других кластеров. В метрическом пространстве "схожесть" обычно определяют через расстояние. Расстояние может рассчитываться как между исходными объектами $\mathbf{x_i}$, так и от этих объектов к объектам - прототипам кластеров. Обычно координаты прототипов заранее неизвестны - они находятся одновременно с разбиением данных на кластеры.

Существует множество методов кластеризации, которые можно классифицировать на четкие и нечеткие. Четкие методы кластеризации разбивают исходное множество объектов на несколько непересекающихся подмножеств. При этом любой объект из $\mathbf{x_i}$ принадлежит только одному кластеру. Нечеткие методы кластеризации позволяют одному и тому же объекту принадлежать одновременно нескольким (или даже всем) кластерам, но с различной степенью. Нечеткая кластеризация во многих ситуациях более "естественна", чем четкая, например, для объектов, расположенных на границе кластеров.

Методы кластеризации также классифицируются по тому, определено ли количество кластеров заранее или нет. В последнем случае количество кластеров определяется в ходе выполнения алгоритма на основе распределения исходных данных.

В той лабораторной работе рассматривается так называемый алгоритм горной кластеризации, который не требует задания количества кластеров. Метод предложен Р. Ягером и Д. Филевым в 1993 г.

Пусть $\mathbf{D}(\mathbf{x_i}, \mathbf{x_j})$ функция, определяющая расстояние от объекта $\mathbf{x_i}$, до объекта $\mathbf{x_j}$, исходного множества объектов.

На первом шаге горной кластеризации определяют точки, которые могут быть центрами кластеров. Если о центрах кластеров заранее ничего не известно, то в качестве потенциальных центров удобно принять сами объекты. Обозначим множество потенциальных центров кластеров $\mathbf{Z_h}$. Если центрами кластеров считаются сами объекты, то тогда $\mathbf{Z_h} = \mathbf{x_k}$.

На втором шаге для каждой такой точки рассчитывается значение

потенциала, показывающего возможность формирования кластера в ее окрестности. Чем плотнее расположены объекты в окрестности потенциального центра кластера, тем выше значение его потенциала. После этого итерационно выбираются центры кластеров среди точек с максимальными потенциалами. Потенциал центров кластеров рассчитывается по следующей формуле:

$$P_i(Z_h) = \sum_{i-1}^{M} \exp\left(-\alpha D(Z_h, x_k)\right)$$

где M- количество объектов x_i, exp - функция экспоненты, α - положительная константа, характеризующая масштаб расстояний между объектами. В простейших случаях удобно полагать, что α равно единице, делённой на среднее расстояние между объектами.

В случае, когда объекты кластеризации заданы двумя признаками (точки на плоскости), графическое изображение распределения потенциала будет представлять собой поверхность, напоминающую горный рельеф. Отсюда и название - горный метод кластеризации.

На третьем шаге алгоритма в качестве центров кластеров выбирают координаты "горных" вершин. Для этого центром первого кластера V_i назначают точку с наибольшим потенциалом. Обычно, наивысшая вершина окружена несколькими достаточно высокими пиками. Поэтому назначение центром следующего кластера точки с максимальным потенциалом среди оставшихся вершин привело бы к выделению большого числа близко расположенных центров кластеров. Чтобы выбрать следующий центр кластера необходимо вначале исключить влияние только что найденного кластера. Для этого значения потенциала для оставшихся возможных центров кластеров пересчитывается следующим образом: от текущих значений потенциала вычитают вклад центра только что найденного кластера. Перерасчет потенциала происходит по формуле:

$$P_2(Z_h) - P_1(V_1) \cdot \exp(-\beta \cdot D) \cdot (Z_h, V_1))$$

где β - положительная константа, характеризующая масштаб размера одного кластера (чем значение β меньше, тем больше размер кластера). В простейших случаях ее можно принять равной α.

Центр второго кластера V_2 определяется как точка с максимальным значением потенциала $P_2(Zh)$. Затем снова пересчитывается значение потенциалов:

$$P_3(Z_h) - P_2(Z_h) - P_2(V_2) \cdot \exp(-\beta \cdot D) \cdot (Z_h, V_2))$$

Итерационная процедура пересчета потенциалов и выделения центров кластеров продолжается до тех пор, пока максимальное значение потенциала превышает некоторый порог. В простейших случаях этот порог можно считать равным половине значения максимального потенциала $P_1(V_1)$.

Точка $\mathbf{i_x}$ считается принадлежащим кластеру V_k , если расстояние до него $D(x_i, V_k)$ меньше, чем расстояние до других кластеров. В этом случае все точки будут принадлежать каким-нибудь кластерам. Если же принять, что $\boldsymbol{x_i}$ считается принадлежащим кластеру V_k, если расстояние до него $D(x_i, V_k)$ меньше, чем

некоторое пороговое значение, то могут остаться точки, не принадлежащие ни одному из найденных кластеров. Ниже приводятся рисунки, иллюстрирующие основные идеи метода горной кластеризации.

Задание к лабораторной работе

1. Разработать программу, реализующую следующие функции:
2. Генерация случайных точек на плоскости вокруг трёх центров кластеризации (как на рисунке 23)
3. Определение потенциалов точек на первом шаге алгоритма кластеризации (как на рисунке 24)
4. Упорядочивание по возрастанию потенциалов точек, (как на рисунке 23)
5. Определение центра первого кластера, сравнение его с исходной точкой.
6. Реализация остальных шагов алгоритма, определение центров 2 и 3 кластера с визуализацией процесса

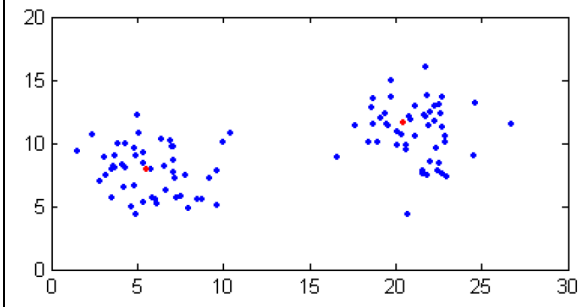


Рисунок 23. Точки на плоскости (синие) и найденные центры кластеров (красные)

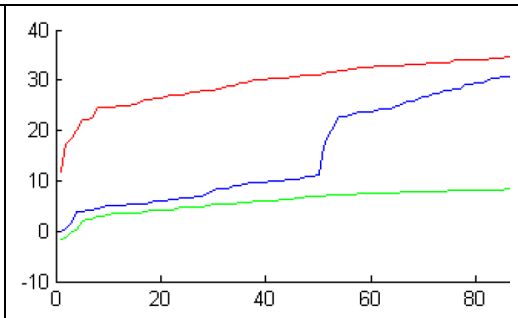


Рисунок 24. Упорядоченные значения потенциалов P1 (красная линия), P2 (синяя), P3 (зеленая).

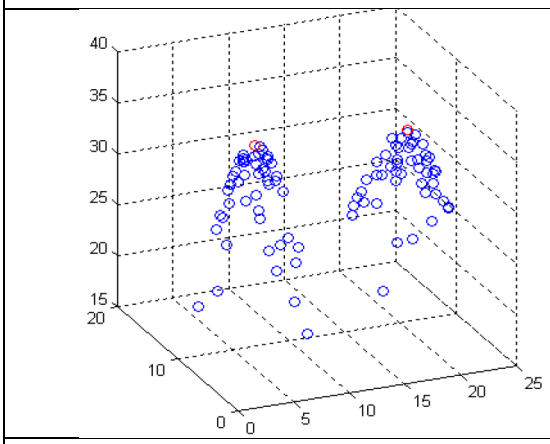


Рисунок 25. Значения потенциалов P1 точек (синим цветом) и значения потенциалов P1 центров кластеров (красным)

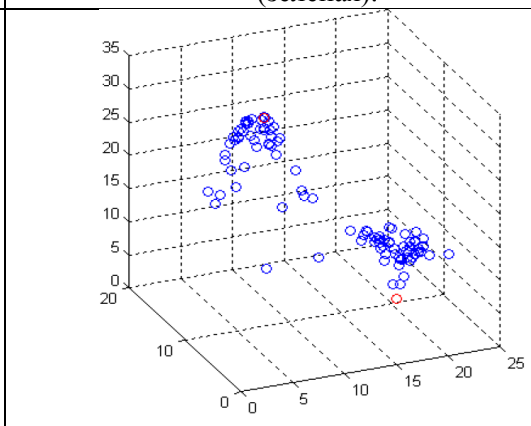


Рисунок 26. Значения потенциалов P2 точек (синим цветом) и значения потенциалов P2 центров кластеров (красным)

Лабораторная работа № 6

Алгоритм кластеризации для объектов с количественными признаками.

Цель работы:

Разработать программу для реализации алгоритма c-средних, разбивающего числовые данные на заданное число кластеров.

Краткая теория.

В этой лабораторной рассматривается кластеризация только для объектов с количественными признаками. Исходной информацией для кластеризации является матрица наблюдений:

$$X = \begin{bmatrix} x_{11} & x_{12} \cdots & x_{1n} \\ x_{21} & x_{22} \cdots & x_{2n} \\ x_{M1} & x_{M2} \cdots & x_{Mn} \end{bmatrix}$$

каждая строчка которой представляет собой значения n признаков одного из M объектов кластеризации.

Задача кластеризации состоит в разбиении объектов из X на несколько подмножеств (кластеров), в которых объекты более схожи между собой, чем с объектами из других кластеров. В метрическом пространстве "схожесть" обычно определяют через расстояние. Расстояние может рассчитываться как между исходными объектами (строчками матрицы X), так и от этих объектов к прототипу кластеров. Обычно координаты прототипов заранее неизвестны - они находятся одновременно с разбиением данных на кластеры.

Существует множество методов кластеризации, которые можно классифицировать на четкие и нечеткие. Четкие методы кластеризации разбивают исходное множество объектов X на несколько непересекающихся подмножеств. При этом любой объект из X принадлежит только одному кластеру. Нечеткие методы кластеризации позволяют одному и тому же объекту принадлежать одновременно нескольким (или даже всем) кластерам, но с различной степенью. Нечеткая кластеризация во многих ситуациях более "естественна", чем четкая, например, для объектов, расположенных на границе кластеров.

Методы кластеризации также классифицируются по тому, определено ли количество кластеров заранее или нет. В последнем случае количество кластеров определяется в ходе выполнения алгоритма на основе распределения исходных данных. В этой лабораторной рассмотрим алгоритм c-средних, разбивающий данные на наперед заданное число кластеров.

Нечеткие кластера опишем следующей матрицей нечеткого разбиения:

$$F = f_{ki}, f_{ki} \subset [0,1], \qquad k = \overline{1,M}, \qquad i = \overline{1,c}$$

где M - число объектов, c - количество кластеров, k-я строчка матрицы $F = f_{ki}$ содержит степени принадлежности объекта $(x_{k1}, x_{k2}, \cdots x_{kn})$ кластерам $A_1, A_2 \cdots A_i$. При том имеют место 2 условия

$\sum_{i=1,c} f_{ki} = 1$ для всех k

$0 < \sum_{k=1,M} f_{ki} < M$ для всех I (то есть не может быть пустых кластеров и все объекты не могут принадлежать одному кластеру).

Нечеткое разбиение позволяет просто решить проблему объектов, расположенных на границе двух кластеров - им назначают степени принадлежностей равные 0,5. Недостаток нечеткого разбиения проявляется при работе с объектами, удаленными от центров всех кластеров. Удаленные объекты имеют мало общего с любым из кластеров, поэтому интуитивно хочется назначить для них малые степени принадлежности. Однако, по условию (12,7) сумма их степеней принадлежностей такая же, как и для объектов, близких к центрам кластеров, т.е. равна единице. Для устранения этого недостатка можно использовать возможностное разбиение, которое требует, только чтобы произвольный объект из X принадлежал хотя бы одному кластеру с некоторой степенью, то есть

$$0 < \sum_{i1,c} f_{ki} < 1$$

Для оценки качества нечеткого разбиения используется такой критерий разброса:

$$\sum_{i=1,c} \sum_{k=1,M} (f_{ki})^m \, \|Y_i - X_k\|^2$$

Где $V_i = \frac{\sum_{k=1M}((f_{ki})^m \cdot X_k}{\sum_{k=1,M}(f_{ki})^m}$ – центры нечетких кластеров, $m = [1, \infty)$ – экспоненциальный вес определяющий нечеткость, размазанность кластеров.

Предложено множество алгоритмов нечеткой кластеризации, основанных на минимизации критерия разброса. Нахождение матрицы нечеткого разбиения F с минимальным значение критерия разброса представляет собой задачу нелинейной оптимизации, которая может быть решена разными методами. Наиболее известный и часто применяемый метод решения этой задачи алгоритм нечетких с-средних, в основу которого положен метод неопределенных множителей Лагранжа. Он позволяет найти локальный оптимум, поэтому выполнение алгоритма из различных начальных точек может привести к разным результатам

Этапы алгоритма:

1. Установить параметры алгоритма: c - количество кластеров; m - экспоненциальный вес; ε - параметр останова алгоритма.
2. Случайным образом сгенерировать матрицу нечеткого разбиения F.читать
3. Рассчитать центры кластеров $V_i = \frac{\sum_{k=1M}((f_{ki})^m \cdot X_k}{\sum_{k=1,M}(f_{ki})^m}, i = \overline{1,c}$
4. Рассчитать расстояния между объектами X_k и центрами кластеров:
$$D_{ki} = \sqrt{\|X_k - V_i\|^2}\,, k = \overline{1,M}\,, i = \overline{1,c}$$
5. Пересчитать элементы матрицы нечеткого разбиения
$$f_{ki} = \frac{1}{\left(D_{ik}^2 \cdot \sum_{j=1,c} \frac{1}{D_{ik}^2}\right)^{1/(m-1)}}, k = \overline{1,M}\,, i = \overline{1,c}$$

6. Рассчитать новые центры кластеров $V_i^* = \frac{\sum_{k=1M}((f_{ki})^m \cdot X_k}{\sum_{k=1,M}(f_{ki})^m}, i = \overline{1,c}$
7. Проверить условие $\|V_i^* - V_i\| < \varepsilon$ для всех $i = \overline{1,c}$ (то есть расстояние между старыми и новыми значениями центров кластеров меньше порогового значения ε. Если условие истинно – то конец, найденные значения V_i^* и есть искомые центры кластеров, если ложное – переход на шаг 4

В приведенном алгоритме самым важным параметром является количество кластеров c. Его выбирают исходя из априорной информации о данных.

Вторым параметром алгоритма кластеризации является экспоненциальный вес (m). Чем больше m, тем конечная матрица нечеткого разбиения F становится более "размазанной", и при m"→∞ она примет вид $f_{ki} = {}^{1}/_{c}$, что является очень плохим решением, т. к. все объекты принадлежат ко всем кластерам с одной и той же степенью. Кроме того, экспоненциальный вес позволяет при формировании координат центров кластеров усилить влияние объектов с большими значениями степеней принадлежности и уменьшить влияние объектов с малыми значениями степеней принадлежности. На сегодня не существует теоретически обоснованного правила выбора значения экспоненциального веса. Обычно устанавливают m = 2.

Задание к лабораторной работе

Разработать программу, реализующую следующие функции:

1. Генерация случайных точек на плоскости вокруг трёх центров кластеризации (как на рисунке 1)
2. Определение центров кластеров и степени принадлежности точек к кластерам алгоритмом c-средних (понадобится 5-7 итераций)

Лабораторная работа № 7.

Распознавание образов методом потенциальных точек.

Цель работы:

Разработать программу для реализации распознавания образов методом потенциальных точек.

Краткие теоретические сведения.

Во многих областях техники используются различные автоматы и устройства, более или менее удачно решающие задачу распознавания образов. Это, например, автомат для сортировки почтовых конвертов по индексу, зенитная ракета, захватывающая горячее сопло самолетного двигателя, но игнорирующая солнце, различные системы анализа спутниковых снимков, голосовой вызов мобильника и многое другое.

Алгоритм распознавания образов методом потенциальных функций относится к числу так называемых алгоритмов с обучением. Доказано, что он имеет глубокую аналогию с популярной в настоящее время нейросетевой технологией распознавания образов. Рассмотрим его в применении к задаче распознавания графических изображений.

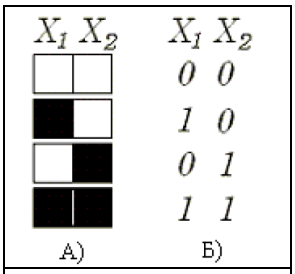


Рис.27.Множество объектов

Для простоты мы будем рассматривать только черно-белые изображения. Пусть рисунок состоит всего из двух пикселей. Тогда множество всех объектов, которое можно будет изобразить (универсальное множество), состоит из четырех объектов: (0,0), (0,1), (1,0), (1,1), где 1 — черный пиксель, 0 — белый .

Все объекты универсального множества можно разместить в вершинах единичного квадрата. Таким образом, множеству фигур, изображенных на двухпиксельном поле, может быть сопоставлено множество точек в двумерном пространстве. Ребру этого квадрата будет соответствовать переход от одного изображения к другому. Для перехода от (1,1) к (0,0) нужно будет пройти два ребра, для перехода от (0,1) к (0,0) — одно. Отметим, что число ребер в нашем переходе — это количество несовпадающих пикселей двух изображений.

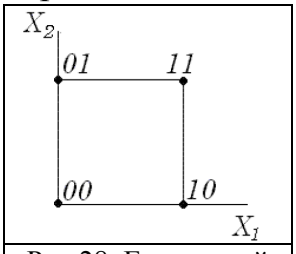


Рис.28. Единичный квадрат

Вывод: расстояние от одного рисунка до другого равно числу несовпадающих пикселей в них. Это расстояние называется расстоянием по Хэммингу.

Теперь представим себе, что у нас рисунок состоит из трех пикселей. Коды изображений тогда будут состоять из трех значений, универсальное множество — из восьми элементов, которые мы разместим в вершинах единичного куба. Но принципиально ничего не изменится, и расстояние по Хэммингу вычисляется так же. Если в задаче используется рисунок 50x70 = 3500 пикселей, то в этом случае код любого изображения состоит из 3500

значений, универсальное множество — из 23500 элементов, которые мы будем размещать в вершинах единичного 3500-мерного куба. Основная идея заключается в том, что в этом многомерном кубе изображения, соответствующие какому-то определенному образу, лежат недалеко друг от друга. Эта идея получила название "Гипотеза о компактности образов".

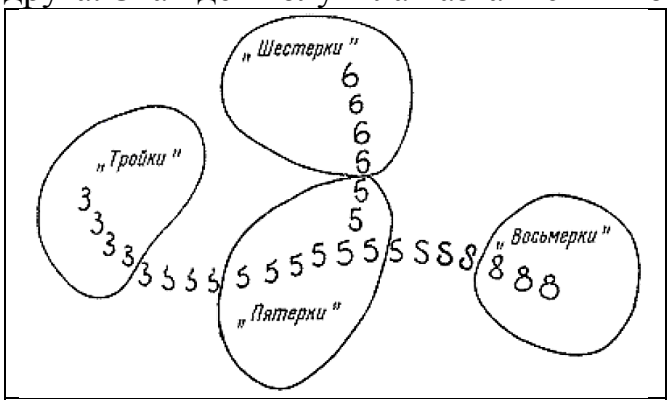


Рис.29. Разбиение множества на части

Теперь можно сформулировать задачу: нужно универсальное множество разбить на "куски", компактные множества, каждому из которых соответствует образ.

Реализация алгоритма несложная. Программе в процессе обучения сообщаются изображения (точки многомерного куба) и указания, к какому образу каждое изображение относится. При распознавании программа просто смотрит, в какую из известных компактных областей попало входное изображение. Для этого используется некоторая характеристика, которая показывает удаленность одного рисунка (точки в вершине многомерного куба) до группы таких же изображений. В качестве меры удаленности рисунка от группы рисунков используется потенциал.

Известно, что электрический заряд создает вокруг себя поле, одной из характеристик которого является потенциал. В любой точке он может быть вычислен по формуле:

$$P = \alpha \cdot \frac{q}{R^2}$$

где α — некоторый постоянный коэффициент, q.— величина заряда, R — расстояние от данной точки до заряда. Если электрическое поле образовано двумя или более зарядами, то потенциал в данной точке равен сумме потенциалов каждого заряда. Аналогия очевидна — каждый рисунок, на котором программа обучалась, создает в пространстве универсального множества потенциал. После обучения программе дают распознать какой- либо рисунок (точку в вершине многомерного куба), программа вычисляет потенциал, создаваемый в этой точке всеми объектами образа "а", образа "б"... на которых программу учили и распознаваемый рисунок относится к образу, который создал наибольший потенциал.

Таким образом, алгоритм представляет собой следующую последовательность операций.

1. Формирование множества распознаваемых образов (например, изображения цифр от 0 до 9) и обучающей выборки для каждого из образов (например, по 10 изображений каждой цифры, нарисованных различными шрифтами).
2. Формирование распознаваемого изображения (например, изображения какой -либо цифры, нарисованной от руки);

3. Вычисление расстояний от распознаваемого изображения до каждого из эталонных изображений обучающей выборки (расстояний по Хэммингу, то есть количества несовпадающих пикселей). Это даст матрицу Rij, где i - номер образа, j - номер обучающего изображения для i-го образа.
4. Вычисление потенциалов, создаваемых обучающими изображениями, в точке распознаваемого изображения по формуле $\varphi(R) = \frac{1000000}{1+R^2}$.
5. Суммирование потенциалов, создаваемых каждым из образов в точке распознаваемого изображения и выбор образа, создающего наибольший потенциал. Это и будет означать, что распознаваемое изображение принадлежит выбранному образу (например, что это - цифра 2).

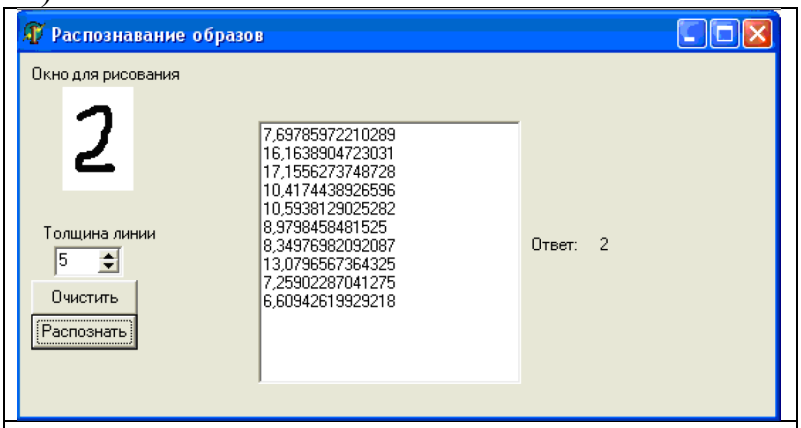


Рис.30. Распознавание цифры 2

Задание

Разработать программу, реализующую следующие функции:

1. Определить три образа: круг, треугольник и квадрат
2. Определить обучающую выборку для каждого из образов, состоящую из 5 отличающихся друг от друга изображений соответствующей геометрической фигуры
3. Задать распознаваемое изображение
4. Определить расстояние по Хэммингу от распознаваемого изображения до изображений обучающей выборки
5. Определить потенциал, создаваемый каждым из образов в распознаваемой точке и выбрать образ с наибольшим потенциалом.

Лабораторная работа № 8

8.1. Выявление показателей, влияющих на валовую прибыль предприятия

Цель работы: приобретение навыков решения экономических задач в нейросетевом логическом базисе.

Теоретическое обоснование

Для рассмотрения результатов разработки и функционирования систем нечёткой логики будем использовать графические средства пакета Fuzzy Logic Toolbox. Эти же средства используются и при разработке систем нечёткого вывода как графический объектно-ориентированный язык автоматического программирования.

В состав этих средств входят:

- редактор систем нечёткого вывода FIS Editor (FIS);
- редактор функций принадлежности систем нечёткого вывода Membership Function Editor (MFE);
- редактор правил систем нечёткого вывода Rule Editor;
- программа просмотра правил системы нечёткого вывода Rule Viewer;
- программа просмотра поверхности нечёткого вывода Sur-face Viewer.

Для описания нечётких высказываний используются нечёткие лингвистические переменные (ЛП).

ЛП — это именованная переменная, которая принимает свои значения из множества лингвистических термов, т.е. символьных величин.

Для нечёткой ЛП терм-множество задаётся как нечёткое множество.

Этот процесс называется фаззификацией. Фаззификация является одной из проблемных задач описания нечёткого вывода и отражает индивидуальные эмпирические знания автора. Нечёткие высказывания в условной части нечёткой продукции могут быть составными, соединёнными связками “И” и/или “ИЛИ”. Эти связки при исчислении высказываний реализуются логическими или арифметическими операциями пересечения или объединения, соответственно.

При получении результата по каждому правилу необходимо дать оценку степени его истинности. Эта оценка зависит от степени истинности высказываний условной части правила, степени истинности отношения, положенного в основу правила, между исходными утверждениями (посылкой) и заключением, т.е. степени истинности импликации, и степени истинности высказывания относительно значения из терм-множества возможных результатов, приведенного в правиле.

Получение оценки степени истинности заключения, полученного по правилу, называют активизацией. В случае необходимости получения чёткого количественного значения результата оно может быть получено на основании функции принадлежности терм-множества результата различными способами по алгоритмам, названым по именам их авторов (Мамдани, Сугено, Цукамото

и т.д.), что определяет тип системы нечёткого вывода. Эта операция называется дефаззификацией.

8.2. Редактор функций принадлежности (MFE)

Редактор функций принадлежности в графическом режиме обеспечивает задание и изменение функции принадлежности любых термов ЛП СНВ.

Для фаззификации лингвистической переменной СНВ следует выделить ее изображение –именованный прямоугольник в левой верхней части окна редактора (см. рис. 31).

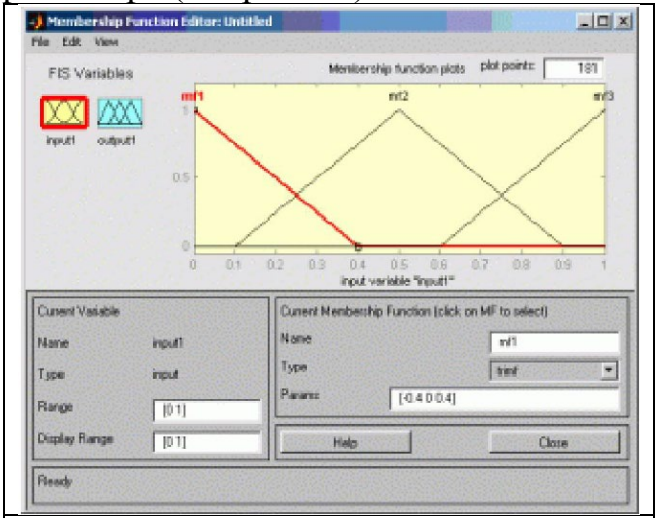


Рис. 31. Окно редактора Membership Function Editor

В окне редактора выводятся графики функций принадлежности для всех значений выделенной ЛП (по умолчанию для трёх значений).

Для описания функции принадлежности каждого значения ЛП используются три поля: Name, Type и Params. Описываемая функция выделяется щелчком по её графику. В поле Name устанавливается значение ЛП. В поле Type, выбором элемента меню, устанавливается имя нужной функции принадлежности (одной из 11-ти встроенных). В поле ввода Params указываются необходимые параметры функции принадлежности, которые определяют положение ее модальных значений на числовой шкале, диапазон изменения которой указывается в полях ввода Range и Display range.

Эти операции выполняются над всеми значениями из терм-множеств лингвистических переменных СНВ.

Добавление нового значения ЛП со встроенной функцией принадлежности производится по команде основного меню Edit > Add MF.

Удаление ненужного значения ЛП производится нажатием клавиши Delete, после выделения графика функции принадлежности этого значения.

8.3. Методика и порядок выполнения работы

Требуется на основе экспертных данных выявить факторы, наиболее сильно влияющие на ежемесячную прибыль предприятия (табл. 3). Для начала проведем предварительный анализ задачи, выделив наименее важные показатели. Это нужно потому, что количество наборов параметров близко к количеству самих параметров, и в данном представлении задача может не иметь видимого решения. В результате ряд следующих показателей может быть исключен из дальнейшего рассмотрения:

- объем реализации по линии бюджета – поскольку данные являются

неполными, содержат пропуски;

- затраты – поскольку мы уже рассматриваем их по частям: на материалы и зарплату;
- объем реализованной продукции, рентабельность, так как они связаны жесткой аналитической зависимостью с другими показателями;
- численность – практически постоянная величина;
- цена единицы продукции – никак не связана с прибылью и другими показателями.

В результате оставлены факторы:

- затраты на материалы,
- объем заработной платы,
- производительность,
- курс доллара.

Структура отчета по лабораторной работе:

1. Название лабораторной работы.
2. Цели лабораторной работы.
3. Ответы на контрольные вопросы.
4. Формулировку задания.
5. Анализ результатов применения нейронной сети.

Таблица 3.

Показатели, характеризующие деятельность предприятия

Фактор	Ед. изм.	1	2	3	4	5	6	7	8	9	10	11	12
Объем реализации (без НДС)	тыс.руб.	354	310	277	302	327	211	263	168	278	292	305	326
в том числе бюджет	тыс.руб.		20		37		27			30	18	10	19
Затраты, в том числе	тыс.руб.	255	281	319	251	215	203	208	172	323	262	239	475
материалы	тыс.руб.	53	58	44	63	38	32	41	33	45	50	39	58
заработная плата	тыс.руб.	122	126	126	104	112	74	102	76	123	117	107	218
Численность	чел.	59	62	62	63	62	62	62	63	63	62	62	62
Производитель-ность	руб./чел.	6003	5002	4474	4798	5273	3410	4237	2673	4406	4711	4833	5253
Цена ед. продукции	руб.	0,08	0,08	0,08	0,06	0,07	0,06	0,08	0,08	0,12	0,10	0,15	0,15
рентабельность	%	38,9	10,4		20,4	52,0	4,1	26,3			11,5	27,4	
Курс $	руб.	6,0	6,1	6,1	6,1	6,2	6,2	6,2	7,9	16,1	16,0	17,9	20,1
Прибыль валовая	тыс.руб	99	29	-42	51	112	8	55	-4	-45	30	66	-149

Вопросы для защиты работы

1. Каким образом производится предварительная обработка результатов для предоставления нейросети?

2. Как определяется структура нейронной сети для решения данной задачи?

3. Что такое синапсы, для чего они служат?

4. Что такое шаг обучения?

5. В каких случаях целесообразно использовать нейронные сети?

6. Приведите примеры задач экономического содержания, для решения которых целесообразно использовать нейропакеты.

Лабораторная работа № 9

Реализация нейронных сетей в пакете Matlab. Графический интерфейс Toolbox NNTOOL

Цель работы: приобретение практических навыков применения нейронных сетей при решении плохо формализованных задач с использованием пакета Matlab.

Теоретическое обоснование

Графический интерфейс пользователя Toolbox NNTOOL для работы с нейронными сетями пакета Matlab позволяет выбирать структуры НС из обширного перечня и предоставляет множество алгоритмов обучения для каждого типа сети.

В лабораторной работе рассмотрены следующие вопросы, относящиеся к работе с NNTool:

- назначение графических управляющих элементов;
- подготовка данных;
- создание нейронной сети;
- обучение сети;
- прогон сети.

Методика и порядок выполнения работы

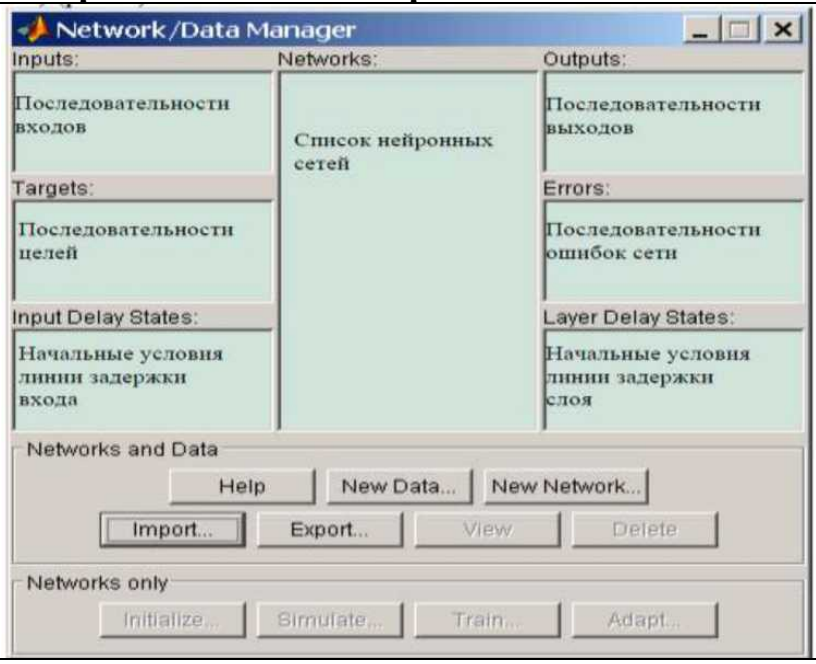


Рисунок 32. Главное окно NNTool

Чтобы запустить NNTool, необходимо выполнить одноимённую команду в командном окне MATLAB:

>> nntool

После этого появится главное окно NNTool, именуемое **Окно управления сетями и данными** (Network/Data Manager) (рис. 28).

Панель **Сети и данные** (Networks and Data) имеет функциональные клавиши со следующими назначениями:

- Помощь (Help) - краткое описание управляющих элементов данного окна.
- Новые данные (New Data...) - вызов окна, позволяющего создавать новые наборы данных.
- Новая сеть (New Network.) - вызов окна создания новой сети.

- Импорт (Import.) - импорт данных из рабочего пространства MATLAB в пространство переменных NNTool.
- Экспорт (Export...) - экспорт данных из пространства переменных NNTool в рабочее пространство MATLAB.
- Вид (View) - графическое отображение архитектуры выбранной сети.
- Удалить (Delete) - удаление выбранного объекта.

На панели **Только сети** (Networks only) расположены клавиши для работы исключительно с сетями. При выборе указателем мыши объекта любого другого типа, эти кнопки становятся неактивными.

При работе с NNTool важно помнить, что клавиши View, Delete, Initialize, Simulate, Train и Adapt (изображены на рис. 4.1 как неактивные) действуют применительно к тому объекту, который отмечен в данный момент выделением. Если такого объекта нет, либо над выделенным объектом невозможно произвести указанное действие, соответствующая клавиша неактивна.

Рассмотрим создание нейронной сети с помощью NNTool на примере.

Пусть требуется создать нейронную сеть, выполняющую логическую функцию **И**.

Создание сети

Выберем сеть, состоящую из одного персептрона с двумя входами. В процессе обучения сети на её входы подаются входные данные и производится сопоставление значения, полученного на выходе, с целевым (желаемым). На основании результата сравнения (отклонения полученного значения от желаемого) вычисляются величины изменения весов и смещения, уменьшающие это отклонение.

Итак, перед созданием сети необходимо заготовить набор обучающих и целевых данных. Составим таблицу истинности для логической функции **И**, где Pi и P_2 - входы, а A - желаемый выход (табл. 4.).

Таблица 4.

Таблица истинности логической функции И

P1	P2	A
0	0	0
0	1	0
1	0	0
1	1	1

Чтобы задать матрицу, состоящую из четырёх векторов-строк, как входную, воспользуемся кнопкой **New Data**. В появившемся окне следует

произвести изменения (рис. 29), и нажать клавишу **Создать** (Create).

После нажатия на **Create** в разделе **Targets** появится вектор target1. Данные в поле **Значение** (Value) могут быть представлены любым понятным MATLAB выражением. К примеру, предыдущее определение вектора целей можно эквивалентно заменить строкой вида bitand([0 0 1 1], [0 1 0 1])..30 Показано на рис.33-35.

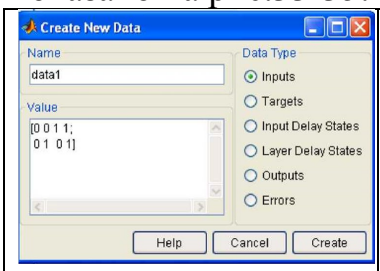


Рис.33. Задание входных векторов

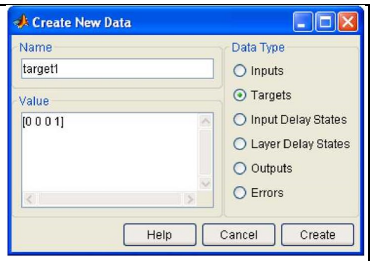


Рис.34. Задание целевого вектора

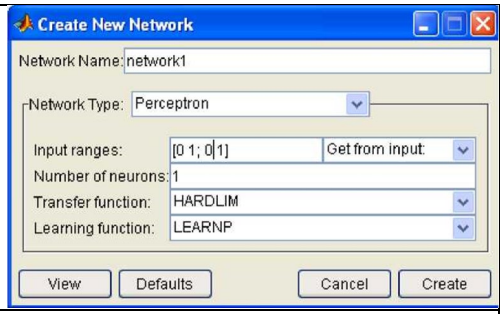


Рис.35. Окно Создание нейронной сети

Теперь следует приступить к созданию нейронной сети. Выбираем кнопку New Network и заполняем форму (рис. 31).

- Имя сети (Network Name) - это имя объекта создаваемой сети.
- Тип сети (Network Type) - определяет тип сети и в контексте выбранного типа представляет для ввода различные параметры в части окна, расположенной ниже этого пункта. Таким образом, для разных типов сетей окно изменяет своё содержание. Для удобства список НС повторен в нижеследующей таблице. Интерфейс NNTool позволяет создавать нейронные сети только с одним или двумя слоями.
- Входные диапазоны (Input ranges) - матрица с числом строк, равным числу входов сети. Каждая строка представляет собой вектор с двумя элементами: первый - минимальное значение сигнала, которое будет подано на соответствующий вход сети при обучении, второй - максимальное. Для упрощения ввода этих значений предусмотрен выпадающий список **Получить из входа** (Get from input), позволяющий автоматически сформировать необходимые данные, указав имя входной переменной.
- Количество нейронов (Number of neurons) - число нейронов в слое.
- Передаточная функция (Transfer function) - в этом пункте выбирается передаточная функция (функция активации) нейронов.
- Функция обучения (Learning function) - функция, отвечающая за обновление весов и смещений сети в процессе обучения.

С помощью клавиши Вид (View) можно посмотреть архитектуру создаваемой сети (рис.36). Так, мы имеем возможность удостовериться, все ли действия были произведены верно. На рисунке изображена персептронная сеть с выходным блоком, реализующим передаточную функцию с жёстким ограничением. Количество нейронов в слое равно одному, что символически отображается размерностью вектора-столбца на выходе слоя и указывается числом непосредственно под блоком передаточной функции. Рассматриваемая сеть имеет два входа, так как размерность входного вектора-столбца равна

двум.

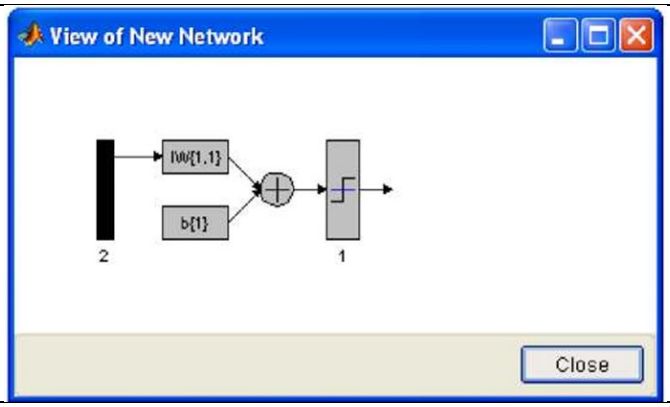


Рис. 36. Предварительный просмотр создаваемой сети

Итак, структура сети соответствует нашему заданию. Теперь можно закрыть окно предварительного просмотра, нажав клавишу **Закрыть** (Close), и подтвердить намерение создать сеть, нажав **Создать** (Create) в окне создания сети. В результате проделанных операций в разделе **Сети** (Networks) главного окна NNTool появится объект с именем network1.

Обучение

Наша цель - построить нейронную сеть, которая выполняет функцию логического **И**. Очевидно, нельзя рассчитывать на то, что сразу после этапа создания сети последняя будет обеспечивать правильный результат (правильное соотношение вход/выход). Для достижения цели сеть необходимо должным образом обучить, то есть подобрать подходящие значения параметров. В MATLAB реализовано большинство известных алгоритмов обучения нейронных сетей, среди которых представлено два для персептронных сетей рассматриваемого вида. Создавая сеть, мы указали LEARNP в качестве функции, реализующей алгоритм обучения.

Вернёмся в главное окно NNTool. На данном этапе интерес представляет нижняя панель **Только сети** (Networks only). Нажатие любой из клавиш на этой панели вызовет окно, на множестве вкладок которого представлены параметры сети, необходимые для её обучения и прогона, а также отражающие текущее состояние сети.

Таблица 5.

Типы нейронных сетей NNTool

№ типа	Тип сети	Название сети	Число слоев	Обучаемые параметры
1	Competitive	Конкурирующая сеть	1	IW{1, 1},
2	Cascade-forward backprop	Каскадная сеть с прямым распространением сигнала и обратным распространением ошибки	2	IW{1, 1}, b{1} LW{2, 1} IW{2, 1}, b{2}
3	Elman backprop	Сеть Элмана с обратным распространением ошибки	2	IW{1, 1}, b{1} LW{2, 1}, b{2} LW{2, 1},
4	Feed-forward backprop	Сеть с прямым распространением сигнала и обратным распространением	2	IW{1, 1}, b{1} LW{2, 1}, b{2}

5	Time delay backprop	Сеть с запаздыванием и обратным распространением ошибки	2	IW{1, 1}, b{1}, LW{2, l},b{2}
6	Generalized regression	Обобщенная регрессионная сеть	2	IW{1, 1}, b{1}, LW{2, l}
7	Hopfield	Сеть Хопфилда	1	LW{l, l},b{l}
8	Linear layer	Линейный слой (создание)	1	IW{1, 1}, b{1}
9	Linear layer (train)	Линейный слой (обучение)	1	IW{1, 1}, b{1}
10	LVQ	Сеть для классификации входных векторов	2	IW{1, 1}, LW{2, 1}
11	Perceptron	Персептрон	1	IW{1, 1), b{l}
12	Probabalistic	Вероятностная сеть	2	IW{1, 1}, b{l}, LW{2, 1}
13	Radial basis (exact fit)	Радиальная базисная сеть с нулевой ошибкой	2	IW{ 1, 1}, b{ 1}, LW{2, 1}
14	Radial basis (fewer neurons)	Радиальная базисная сеть с минимальным числом нейронов	2	IW{1, 1}, b{ 1}, LW{2, 1}, b{2}
15	Self-organizing	Самоорганизующаяся карта Кохонена	1	IW{1, 1}

Отметив указателем мыши объект сети networkl, вызовем окно управления сетью нажатием кнопки **Train**. Перед нами возникнет вкладка **Train** окна свойств сети, содержащая, в свою очередь, ещё одну панель вкладок (рис. 33). Их главное назначение - управление процессом обучения. На вкладке **Информация обучения** (Training info) требуется указать набор обучающих данных в поле **Входы** (Inputs) и набор целевых данных в поле **Цели** (Targets). Поля **Выходы** (Outputs) и **Ошибки** (Errors) NNTool заполняет автоматически. При этом результаты обучения, к которым относятся выходы и ошибки, будут сохраняться в переменных с указанными именами.

Завершить процесс обучения можно, руководствуясь разными критериями. Возможны ситуации, когда предпочтительно остановить обучение, полагая достаточным некоторый интервал времени. С другой стороны, объективным критерием является уровень ошибки.

На вкладке **Параметры обучения** (Training parame-ters) для нашей сети (рис. 34) можно установить следующие поля:

- Количество эпох (epochs) - определяет число эпох (интервал времени), по прошествии которых обучение будет прекращено. Эпохой называют однократное представление всех обучающих входных данных на входы сети.
- Достижение цели или попадание (goal) - здесь задаётся абсолютная величина функции ошибки, при которой цель будет считаться достигнутой.
- Период обновления (show) - период обновления графика кривой обучения, выраженный числом эпох.
- Время обучения (time) - по истечении указанного здесь временного интервала, выраженного в секундах, обучение прекращается.

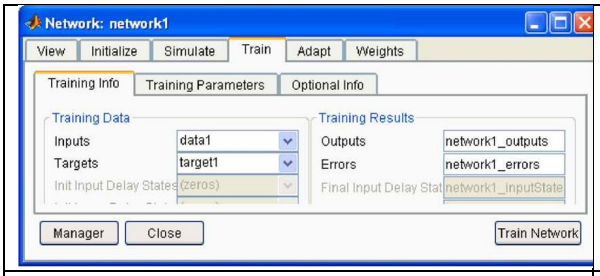


Рис.37. Окно параметров сети, открытое на вкладке **Обучение** (Train)

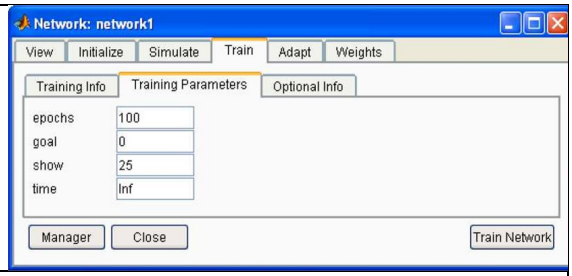


Рис.38. Вкладка параметров обучения

Принимая во внимание тот факт, что для задач с линейно отделимыми множествами (а наша задача относится к этому классу) всегда существует точное решение, установим порог достижения цели, равный нулю. Значения остальных параметров оставим по умолчанию. Заметим только, что поле времени обучения содержит запись Inf, которая определяет бесконечный интервал времени (от английского Infinite - бесконечный).

Следующая вкладка **Необязательная информация** (Optional Info) показана на рис.39.

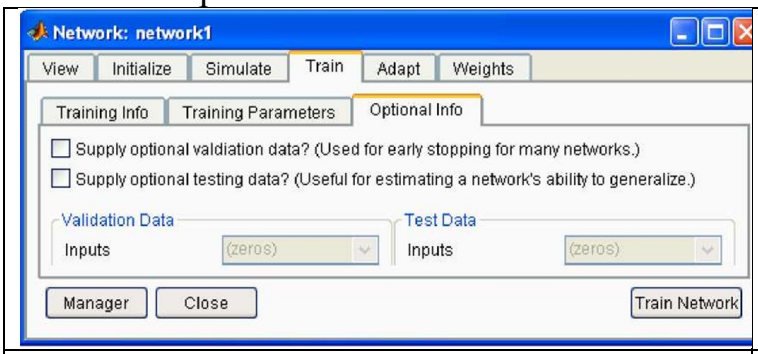


Рис. 39 - Вкладка необязательной информации

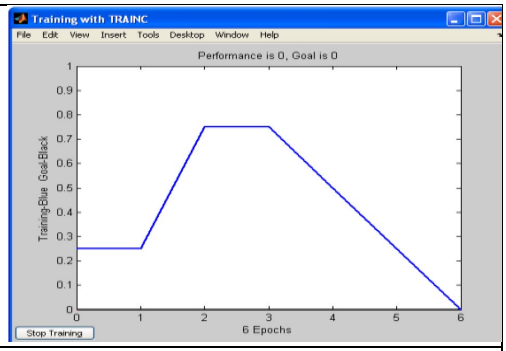


Рис.40. Кривая обучения

Рассмотрим вкладку обучения (Train). Чтобы начать обучение, нужно нажать кнопку **Обучить сеть** (Train Network). После этого, если в текущий момент сеть не удовлетворяет ни одному из условий, указанных в разделе параметров обучения (Training Parameters), появится окно, иллюстрирующее динамику целевой функции - кривую обучения. В нашем случае график может выглядеть так, как показано на рис. 40.

Кнопкой **Остановить обучение** (Stop Training) можно прекратить этот процесс. Из рисунка видно, что обучение было остановлено, когда функция цели достигла установленной величины (goal = 0).

Следует отметить, что для персептронов, имеющих функцию активации с жёстким ограничением, ошибка рассчитывается как разница между целью и полученным выходом. Итак, алгоритм обучения нашёл точное решение задачи. В методических целях убедимся в правильности решения задачи путём прогона обученной сети. Для этого необходимо открыть вкладку **Прогон** (Simulate) и выбрать в выпадающем списке **Входы** (Inputs) заготовленные данные. В данной задаче естественно использовать тот же набор данных, что и при обучении datal. При желании можно установить флажок **Задать цели** (Supply Targets). Тогда в результате прогона дополнительно будут рассчитаны значения ошибки. Нажатие кнопки **Прогон сети** (Simulate Network) запишет

результаты прогона в переменную, имя которой указано в поле **Выходы** (Outputs). Теперь можно вернуться в основное окно NNTool и, выделив мышью выходную переменную network1, нажать кнопку **Просмотр** (View). Содержимое окна просмотра совпадает со значением вектора целей - сеть работает правильно.

Следует заметить, что сеть создаётся инициализированной, то есть значения весов и смещений задаются определённым образом. Перед каждым следующим опытом обучения обычно начальные условия обновляются, для чего на вкладке **Инициализация** (Initialize) предусмотрена функция инициализации. Так, если требуется провести несколько независимых опытов обучения, инициализация весов и смещений перед каждым из них осуществляется нажатием кнопки **Инициализировать веса** (Initialize Weights).

Вернёмся к вкладке **Необязательная информация** (Optional Info) (рис. 35). Чтобы понять, какой цели служат представленные здесь параметры, необходимо обсудить два понятия: переобучение и обобщение.

При выборе нейронной сети для решения конкретной задачи трудно предсказать её порядок. Если выбрать неоправданно большой порядок, сеть может оказаться слишком гибкой и может представить простую зависимость сложным образом. Это явление называется переобучением. В случае сети с недостаточным количеством нейронов, напротив, необходимый уровень ошибки никогда не будет достигнут. Здесь налицо чрезмерное обобщение. Для предупреждения переобучения применяется следующая техника. Данные делятся на два множества: обучающее (Training Data) и контрольное (Validation Data). Контрольное множество в обучении не используется. В начале работы ошибки сети на обучающем и контрольном множествах будут одинаковыми. По мере того, как сеть обучается, ошибка обучения убывает, и, пока обучение уменьшает действительную функцию ошибки, ошибка на контрольном множестве также будет убывать. Если же контрольная ошибка перестала убывать или даже стала расти, это указывает на то, что обучение следует закончить. Остановка на этом этапе называется ранней остановкой (Early stopping). Таким образом, необходимо провести серию экспериментов с различными сетями, прежде чем будет получена подходящая. При этом чтобы не быть введённым в заблуждение локальными минимумами функции ошибки, следует несколько раз обучать каждую сеть. Если в результате последовательных шагов обучения и контроля ошибка остаётся недопустимо большой, целесообразно изменить модель нейронной сети (например, усложнить сеть, увеличив число нейронов, или использовать сеть другого вида). В такой ситуации рекомендуется применять ещё одно множество - тестовое множество наблюдений (Test Data), которое представляет собой независимую выборку из входных данных. Итоговая модель тестируется на этом множестве, что даёт дополнительную возможность убедиться в достоверности полученных результатов. Очевидно, чтобы сыграть свою роль, тестовое множество должно быть использовано только один раз. Если его использовать для корректировки сети, оно фактически превратится в

контрольное множество. Установка верхнего флажка (рис. 35) позволит задать контрольное множество и соответствующий вектор целей (возможно, тот же, что при обучении)

Установка нижнего флажка позволяет задать тестовое множество и вектор целей для него.

Обучение сети можно проводить в разных режимах. В связи с этим, в NNTool предусмотрено две вкладки, представляющие обучающие функции: рассмотренная ранее вкладка **Train** и **Адаптация** (Adapt). Adapt вмещает вкладку информация адаптации (Adaption Info), на которой содержатся поля, схожие по своему назначению с полями вкладки **Training Info** и выполняющие те же функции и вкладку параметры адаптации (Adaption Parameters). Последняя содержит единственное поле **Проходы** (Passes). Значение, указанное в этом поле, определяет, сколько раз все входные векторы будут представлены сети в процессе обучения. Параметры вкладок Train и **Adapt** в MATLAB используются функциями train и adapt, соответственно.

Структура отчета по лабораторной работе:

1. Название лабораторной работы.
2. Цели лабораторной работы.
3. Ответы на контрольные вопросы.

Вопросы для защиты работы

1. В какой форме принимает и выдает данные пакет Matlab?
2. По каким критериям завершается обучение в пакете Matlab?
3. Как использовать обученную в Matlab сеть?
4. Как готовятся тренировочные и тестовые данные для пакета Matlab?
5. Как происходит процесс обучения сети и ее использования в Matlab?
6. Какие функции активации нейронов реализованы в данном пакете?
7. Какие методы упрощения сети реализованы в Matlab?

Лабораторная работа № 10.

Нейронные сети в системах искусственного интеллекта.
«Аппроксимация функций нейронной сетью»

Цель работы: В среде Matlab необходимо построить и обучить нейронную сеть для аппроксимации таблично заданной функции.

Ход работы.

Создаем таблицу экспери-ментальных данных:

y_i – задано вариантом;

x_i – вычисляется по следую-щей формуле:

$$x_i = a + h \cdot i, \qquad i = 0,1,\cdots,10, \qquad h = \frac{b-a}{10} \text{ на отрезке } [a,b]$$

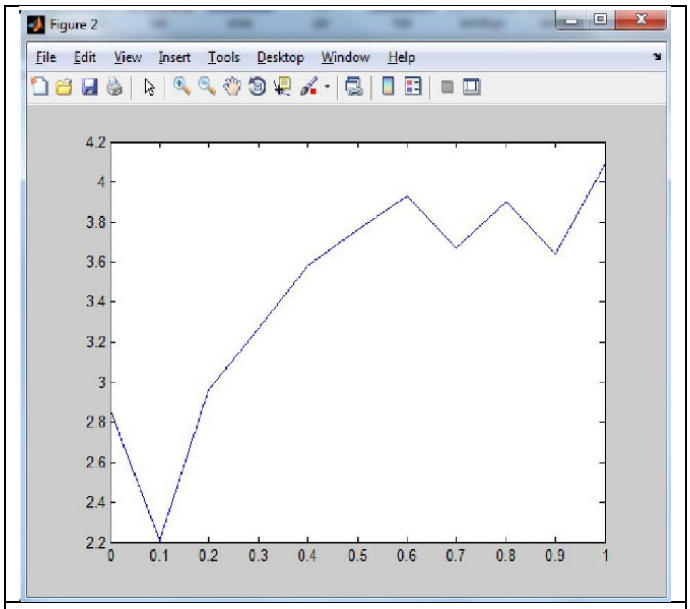


Рис.41. График исходной функции

x_i						
$f(x_i)$						

1. Создание и обучение нейронной сети:

```
x=[00.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1];
y=[2.86 2.21 2.96 3.27 3.58 3.76 3.93 3.67 3.90 3.64 4.09];
net=newff([0 3],[10,1],{'tansig','purelin'},'trainbfg');
net.trainParam.epochs=300;
net.trainParam.show=50;
net.trainParam.goal=1.37e-2;
[net,tr]=train(net,x,y);
an=sim(net,x);
plot(x,y,'+r',x,an,'-g'); hold on;
xx=[0.1850.86];
v=sim(net,xx)
plot(xx,v,'ob','MarkerSize',5,'LineWidth',2)
```

В процессе обучения сети получился график зависимости характеристики точности обучения сети от количества эпох (циклов), и

вычисление среднеквадратичной ошибки сети составляет 0,013305 за 37 циклов (рис.42).

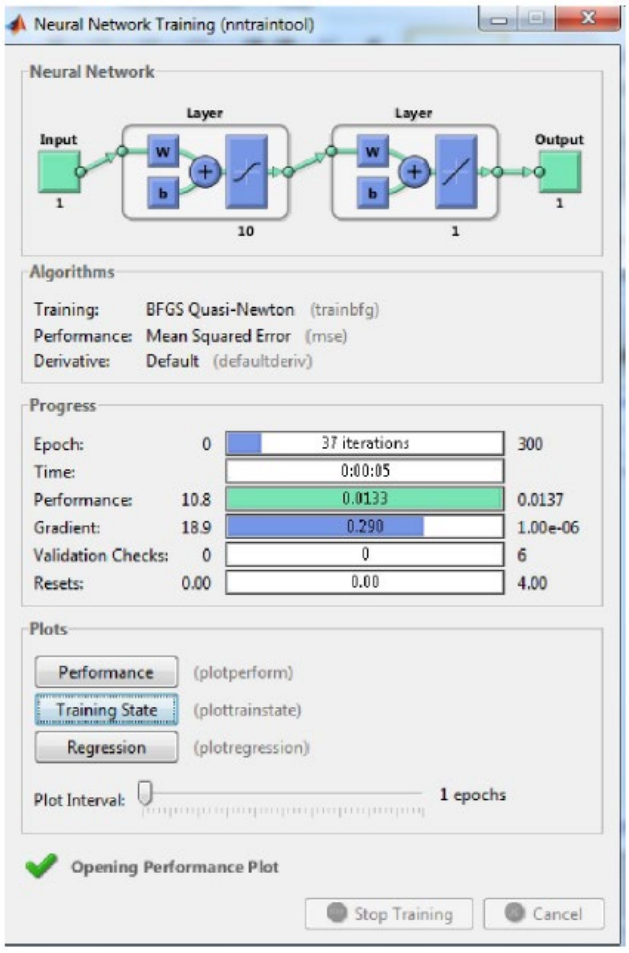


Рис.42. Обучение сети

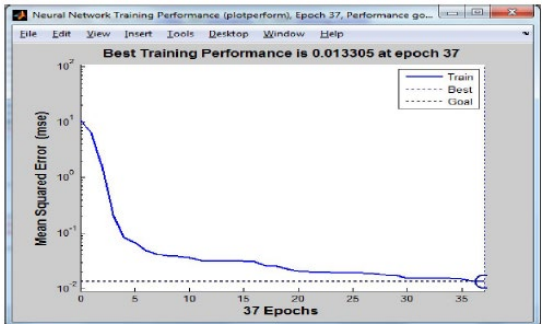


Рис.43. Характеристика точности обучения в зависимости от количества эпох обучения

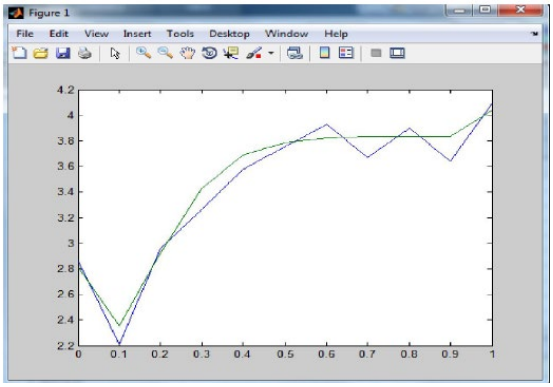


Рис.44. Сравнение графиков исходной функции и аппроксимации

Аппроксимируем входящий набор точек методом наименьших квадратов (МНК). Результаты показаны на рис.43 и 44.

В результате установлено, что по полученным результатам аппроксимации заданного набора значений функции, нейронная сеть намного лучше аппроксимирует исходное значениия функции, чем МНК (рис.45-46).

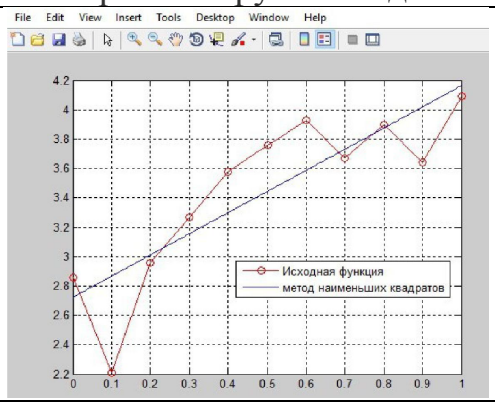


Рис.45. Аппроксимация входящего набора точек методом МНК

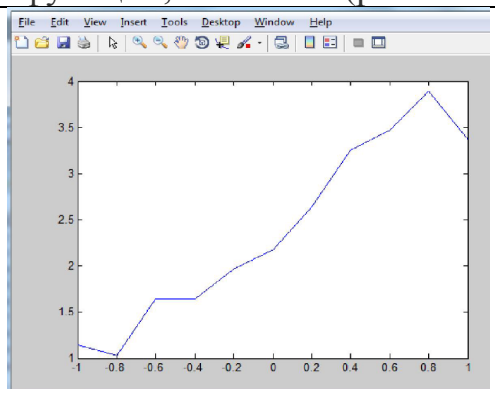


Рис.46. График исходной функции

Контрольные вопросы

1. Каким образом знаний хранятся в нейронной сети?
2. Какой метод обучения используется в нейронной сети?

3. Что такое синапс?
4. Как выполняется обучение в нейронной сети?
5. В чем назначение аксона?

Содержание отчета

1. - цель работы
2. - краткие теоретические сведения
3. - описание предметной области
4. - структура нейронной сети
5. - листинг программы
6. - ответы на вопросы.

Лабораторная работа №11

Нейронные сети для распознавания образов

Цель работы: реализация алгоритмов распознавания образов с помощью нейронных сетей на языке программирования и изучение их свойств.

Последовательность проведения лабораторной работы:

1. Изучить устройство и работу однослойной, двухслойной нейронных сетей, сети встречного распространения.
2. Выбрать исходный алфавит классов (цифры, буквы русского алфавита, буквы английского алфавита, буквы греческого алфавита, знаки препинания и арифметические символы, радиоэлементы, геометрические фигуры, узоры, дорожные знаки, топологические элементы, картографические обозначения) по согласованию с преподавателем.
3. Составить программу, реализующую однослойную, двухслойную нейронные сети и сеть встречного распространения для задачи распознавания. Обучить нейронные сети.
4. Программное обеспечение должно позволять просматривать эталоны (классы) изображений, а также распознаваемые изображения; записывать эталоны изображений в библиотеку на диск, записывать входной образ на диск; позволять редактировать входной образ, сохранять веса нейронов при обучении сети.
5. Испытать программное обеспечение для различных входных данных.
6. Произвести оценку качества распознавания для различных случаев, систематизировав полученные результаты в таблицы. Построить графики выявленных зависимостей, сделать выводы.
7. Результаты работы оформить в виде отчета в текстовом редакторе.

Рекомендации по созданию программного обеспечения

После изучения теоретического материала, и выбора исходных изображений, системы признаков, необходимо определить количество нейронов в слоях. Максимальное количество нейронов первого слоя может совпадать с количеством признаков, хотя теоретически избыточность допускается, но не является рациональным подходом. Желательно, при создании программного обеспечения использовать объектно-ориентированный язык, чтобы была возможность динамически варьировать количество нейронов в слоях - порождать соответствующее количество экземпляров класса и количество слоев.

Рекомендации по исследованиям

При исследовании нейронных сетей необходимо проделать следующие эксперименты по выявлению качества распознавания с помощью созданного программного обеспечения.

Эксперимент №1. Исследование влияния отклонения изображения от эталона (в разных точках изображения) на качество распознавания. В качестве исходного распознаваемого образа берется один из эталонов и подправляется

таким образом, чтобы отсутствовал один, два и т. д. пикселей в изображении эталона. Эксперимент проводится для нескольких случаев (отсутствие пикселей в разных участках изображения) для всех эталонов. При этом сравниваются различные нейронные сети.

Эксперимент №2. Исследование влияния отклонений в виде шума одного, двух, трех и т. д. пикселей в изображении на качество распознавания. В качестве исходного распознаваемого образа берется один из эталонов и добавляется один или несколько пикселей шума. Эксперимент проводится для различного расположения шума и для различных эталонов. В ходе эксперимента также сравниваются различные нейронные сети.

Эксперимент №3. Исследование влияния наличия шума и отклонений в изображении в виде одного, двух, трех и т д. пикселей в изображении на качество распознавания. В качестве исходного распознаваемого образа берется один из эталонов. В изображение распознаваемого образа вносится шум в виде нескольких пикселей и удаляется несколько пикселей в изображении символа. Эксперимент повторяется для различного расположения шума и отклонений, и для разных эталонов на различных типах нейронных сетей. Данный эксперимент является комбинацией первых двух.

Эксперимент №4. Исследование влияния наличия черной строки или столбца в изображении (как помеха в образе) на качество распознавания. В качестве исходного распознаваемого образа берется один из эталонов. В изображение вносится черная строка или столбец. Эксперимент повторяется для различного положения строки или столбца в изображении и для различных эталонов. В ходе эксперимента сравниваются различные нейронные сети.

Эксперимент №5. Исследование влияния наличия белой строки или столбца в изображении (как помеха в образе) на качество распознавания. В качестве исходного распознаваемого образа берется один из эталонов. В изображение вносится белая строка или столбец. Эксперимент повторяется для различного положения строки или столбца в изображении и для различных эталонов на различных нейронных сетях.

Эксперимент №6. Исследование влияния количества нейронов в слоях на качество распознавания. В качестве исходного распознаваемого образа берется один из эталонов. Количество нейронов в слое варьируется от двух до равного числу признаков (для двухслойных сетей слои варьируются последовательно). Эксперимент повторяется на различных нейронных сетях.

Эксперимент №7. Исследование влияния количества нейронов в слоях и количества эталонов на скорость обучения сети. В ходе проведения эксперимента количество нейронов в слое варьируется от двух до равного числу признаков (для двухслойных сетей слои варьируются последовательно), также варьируется количество эталонов (в пределах выбранных по согласованию с преподавателем). В каждом случае фиксируется число итераций и время обучения. Эксперимент повторяется на различных нейронных сетях.

Эксперимент №8. Исследование влияния начертания входных образов на качество распознавания. В качестве исходного распознаваемого образа берется один из эталонов. Эталон модифицируется (делается жирное или наклонное, или подчеркнутое начертание). Эксперимент повторяется на различных нейронных сетях для различных видоизменений.

Примечание. При использовании цвета в изображении эксперименты №4 и №5 следует проводить для цвета фона и цвета образа. В эксперименте №4 вместо черной строки или столбца берется столбец или строка цвета образа. В эксперименте №5 вместо белой строки или столбца берется строка или столбец фонового цвета.

Рекомендации по оформлению отчета

Результаты экспериментов следует оформить в виде таблиц и графиков с пояснениями и подписями. На графиках следует показать зависимость ошибки распознавания для различных нейронных сетей от количества и вида помех, вносимых в изображение, а также от варьируемых параметров.

В качестве таблицы значений можно использовать таблицу следующего вида

Таблица

Численные значения экспериментов

Изображение входного образа	Значение СКО	Нейронная сеть		
		Однослойная	Двухслойная	Кохонена-Гроссберга

Заполняйте две копии этой таблицы - одну для эталонов, другую для предъявляемых изображений, записывая в колонки значения выхода нейронной сети.

В отчете также следует привести данные об обучении нейронной сети: количество итераций, первоначальные и конечные значения весов.

Контрольные вопросы к лабораторной работе

1. Каковы основные понятия теории распознавания?
2. Дайте определение класса образов.
3. Что такое алфавит классов?
4. Дайте определение объекта класса образов.
5. Дайте определение признака класса образов.
6. Какие типы признаков вы знаете? Приведите примеры.
7. Что такое нейронная сеть?
8. Что такое синапс?
9. Что такое аксон?
10. Что определяет уровень активации нейрона?
11. Дайте определение активационной функции.
12. Какие типы активационных функций Вам известны?
13. Что такое персептрон?
14. В чем преимущество сигмоидальной функции?

15. В чем заключается проблема функции «исключающее или»?
16. В чем заключается цель обучения нейронной сети?
17. Что такое обучающая пара?
18. Что такое обучающее множество?
19. Расскажите алгоритм обучения персептрона.
20. Что такое дельта-правило?
21. Перечислите шаги процедуры обратного распространения.
22. Какие действия выполняются при проходе вперед?
23. Какие действия выполняются при обратном проходе?
24. Какие недостатки есть у процедуры обратного распространения?
25. Опишите устройство сети встречного распространения.
26. Как устроен и работает слой Кохонена?
27. Как устроен и работает слой Гроссберга?
28. В чем заключается проблема выбора начальных значений весовых векторов?
29. Как решают проблему выбора начальных значений весовых векторов?

Методические указания рекомендованы на заседании МКН подготовки 09.04.02 «Информационные системы и технологии» профиль «Искусственный интеллект в проектировании городской среды» к размещению на образовательном портале ГАОУ АО ВО «АГАСУ» (http://moodle.aucu.ru)