Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Томский государственный университет систем управления и радиоэлектроники»

Кафедра компьютерных систем в управлении и проектировании

ИНТЕЛЛЕКТУАЛЬНЫЕ ТЕХНОЛОГИИ И ПРЕДСТАВЛЕНИЕ ЗНАНИЙ

Методические указания к лабораторным работам

Томск 2018

Кочергин М.И, Ганджа Т.В.

Интеллектуальные технологии и представления знаний / Методические указания к выполнению лабораторных работ. – Томск: Томский государственный университет систем управления и радиоэлектроники, 2018. – 59 с.

Методическое пособие для студентов вузов технических направлений посвящено изучению таких разделов современных интеллектуальных технологий как базы знаний, нечёткая логика, нейронные сети, автоматическая обработка текста, интеллектуальный анализ данных. Для каждого из разделов дано по три лабораторные работы, что позволяет регулировать глубину проработки тем. Рассматривается работа в следующих программных продуктах: GNU Prolog, CLIPS, Protégé, Matlab (в том числе Fuzzy Logic Toolbox, Neural Networks Toolbox, Text Analytics Toolbox), Deductor Academic.

ОГЛАВЛЕНИЕ

Лабораторная работа 1. Логическая модель представления знаний

1. Цель работы

Изучение логической модели представления знаний, формирование навыков формализации знаний и создания баз знаний для решения прикладных задач.

2. Указания к выполнению работы

Одним из способов представления знаний является язык математической логики, позволяющий формально описывать понятия предметной области и связи между ними. Логическая модель - это множество предложений, выражающих различные логические свойства именованных отношений. При логическом программировании пользователь описывает предметную область совокупностью предложений в виде логических формул, а ЭВМ, манипулируя этими предложениями, строит необходимый для решения задач вывод.

Терм – это знак (символ) или комбинация знаков (символов), являющаяся наименьшим значимым элементом языка. К термам относятся константы, переменные и функции (структуры).

Предикат – это логическая функция, которая выражает отношение между своими аргументами и принимает значение "истина", если это отношение имеется, или "ложь", если оно отсутствует.

Пример:

- является (ласточка, птица).
- отец (Х, Джон).

Предикаты могут быть связаны логическими связками, образуя атомарные формулы. Наиболее часто в логическом программировании используются связки "И", "ИЛИ", "НЕ" "ЕСЛИ".

Программирование в ПРОЛОГ.

Код (набор фактов и правил необходимо) записывать с помощью блокнота и сохранять с расширением .pl. Имена предикатов и констант записываются с маленькой буквы, переменных – с большой. Константы и предикаты могут содержать как английские символы, так и русские.

Для работы с машиной логического вывода в ПРОЛОГ необходимо подключить файл базы знаний командой «File – Consult». В случае успешного подключения файла компилятор выдаст сообщение «Yes», в случае ошибок, выдаётся ответ «No», а перед ним список ошибок с указанием строки и сути ошибки. В случае успешности подключения можно производить запросы. Они вводятся в строке после символа «| ?–» – приглашения к вводу.

В ПРОЛОГ конъюнкция (логическое И) задаётся символом «,» (запятая), дизъюнкция (логическое ИЛИ) символом «;» (точка с запятой). Отрицания (логического НЕ) в ПРОЛОГе в чистом виде нет, однако можно реализовать его через предикат not за счёт проверки невыводимости выражения-аргумента предиката not (т.е. если нельзя доказать истинность (выводимость) выражения, оно считается ложным). Для этого необходимо реализовать этот предикат следующим образом (вставить следующий код в ваш .pl-файл перед набором правил):

```
not(X):-call(X),!,fail.
not(X).
```

В таком случае предикат not(p(x)) вернёт истину, если его аргумент (предикат p(x)) не является истинным (выводимым из базы знаний), и наоборот, вернёт ложь, если предикат p(x) является истинным, т.е. выводимым.

3. Содержание работы

1. Формализация знаний предметной области
2. Разработка базы знаний

4. Порядок проведения работы

Задание 1. Формализовать знания о предметной области, представленные в тексте (согласно варианту), записав их на языке логики предикатов.

Задание 2. Реализовать базу знаний логического типа на языке ПРОЛОГ (рекомендуется использовать компилятор GNU Prolog).

Задание 3. Сформулировать запросы и провести тестирование разработанной базы знаний.

5. Варианты заданий

1. Консультация при выборе фотоаппаратов
2. Консультация при выборе места отдыха
3. Консультация при выборе спецодежды
4. Консультация при выборе автомобиля
5. Консультация при выборе образовательного учреждения
6. Определение целевой аудитории периодического издания
7. Определение целевой аудитории программного продукта
8. Определение целевой аудитории товара
9. Классификация насекомых
10. Классификация растений
11. Классификация животных
12. Классификация книги

13. Классификация фильмов
14. Классификация компьютерных игр
15. Диагностика заболеваний

6. Контрольные вопросы

– Опишите особенности логической модели представления знаний.
– Основные типы моделей представления знаний.
– Понятие предиката. Понятие высказывания.
– Особенности представления логической модели представления знаний в ПРОЛОГ (переменные, структура, алфавит, операторы и пр.)
– Составьте классифицирующую логическую МПЗ для заданной предметной области согласно варианту (5-10 фактов, 2-3 правила).
– Имеется набор правил (состоящий из предиката «смена(X,Y)» - человек X работает в смену Y). Запишите правило-предикат, позволяющее установить знакомы ли X и Z (они считаются знакомыми если работают в одну смену).

7. Содержание отчета

Отчёт должен содержать: 1) *постановку задачи* (исходные данные), 2) *ход работы* (краткое описание этапов выполнения работы), 3) *результаты работы* (описание, интерпретация и сравнение результатов), 4) *заключение* (выводы, интерпретация полученных результатов), 5) *приложение* (код программы с построчными комментариями).

Лабораторная работа 2. Продукционная модель представления знаний

1. Цель работы

Изучение продукционной модели представления знаний, формирование навыков формализации знаний и создания баз знаний для решения прикладных задач.

2. Указания к выполнению работы

Продукционные модели - это наиболее распространенные на текущий день модели представления знаний, где знания описываются с помощью правил «если-то» (явление → реакция) и представляются а виде:

ЕСЛИ условие (антецедент), **ТО** действие (консеквент)

Под условием понимается некоторое предложение-образец, по которому осуществляется поиск в базе знаний, а под действием – набор действий, выполняемых при успешном исходе поиска. Внутри консеквента могут также генерироваться и добавляться в базу новые факты, которые были получены в результате вычислений или взаимодействия с пользователем.

Общим для систем продукций является то, что они состоят из трех основных элементов: 1. Набора правил, используемых как база знаний (БЗ), который чаще всего называют базой правил. 2. Рабочей памяти, где хранятся предпосылки, касающиеся отдельных задач, а также результаты выводов, получаемых на основе этих предпосылок (динамическая база данных - ДБД). 3. Механизма логического вывода, использующего правила в соответствии с содержимым рабочей памяти.

CLIPS (*C Language Integrated Production System*) начала разрабатываться в космическом центре Джонсона NASA в 1984 году. Сейчас *CLIPS* и документация на этот инструмент свободно распространяется через интернет (http://www.ghg.net/clips/CLIPS.html).

Язык *CLIPS* свободен от недостатков предыдущих инструментальных средств для создания ЭС, основанных на языке LISP. Язык CLIPS получил большое распространение в государственных организациях и учебных заведениях благодаря низкой стоимости, мощности, эффективности и переносимости с платформы на платформу. Например, даже *Web*-ориентированный инструментарий JESS (*Java Expert System Shell*), использующий язык представления знаний CLIPS, приобрел достаточную известность в настоящее время.

Правила в CLIPS состоят из предпосылок и следствия. Предпосылки также называют ЕСЛИ-частью правила, левой частью правила или LHS

правила (*left-hand side of rule*). Следствие называют ТО-частью правила, правой частью правила или RHS правила (*right-hand side of rule*).

Пример правила представлен ниже:

```
(deftemplate data (slot x) (slot y))
(defrule twice
(data (x ? x) (y =(*2 ? x)))
=>)
(assert (data (x2) (y4)); f-0
        (data (x3) (y9))); f-1
```

CLIPS поддерживает следующие процедурные функции , реализующие возможности ветвления, организации циклов в программах и т.,п.:

- If – оператор ветвления;
- While – цикл с предусловием;
- loop-for-count – итеративный цикл;
- prong – объединение действий в одной логической команде;
- prong$ – выполнение набора действий над каждым элементом поля;
- return – прерывание функции, цикла, правила и т.д.;
- break – то же, что и return, но без возвращения параметров;
- switch – оператор множественного ветвления;
- bind – создание и связывание переменных.

3. Содержание работы

1. Формализация знаний предметной области
2. Разработка базы знаний

4. Порядок проведения работы

Задание 1. Формализовать знания о предметной области, представленные в тексте (согласно варианту), записав их в виде продукционных правил.

Задание 2. Реализовать базу знаний продукционного типа в среде CLIPS или Exsys Corvid.

Задание 3. Сформулировать запросы и провести тестирование разработанной базы знаний.

5. Варианты заданий

1. Консультация при выборе фотоаппаратов
2. Консультация при выборе места отдыха
3. Консультация при выборе спецодежды
4. Консультация при выборе автомобиля

5. Консультация при выборе образовательного учреждения
6. Определение целевой аудитории периодического издания
7. Определение целевой аудитории программного продукта
8. Определение целевой аудитории товара
9. Классификация насекомых
10. Классификация растений
11. Классификация животных
12. Классификация книги
13. Классификация фильмов
14. Классификация
15. Диагностика заболеваний

6. Контрольные вопросы

– Опишите особенности представления продукционной модели представления знаний в CLIPS/Exsys Corvid (имена переменных, структура, алфавит, операторы и пр.).

– Опишите особенности продукционной модели представления знаний.

– Формальное (математическое) описание продукционной модели представления знаний.

– Состав и структура типовой системы, основанной на продукционной базы знаний.

– Механизмы вывода в продукционных системах (с примером).

– Составьте консультирующую продукционную МПЗ для заданной ПрО (4-6 правил; по вариантам).

7. Содержание отчета

Отчёт должен содержать: 1) *постановку задачи* (исходные данные), 2) *ход работы* (краткое описание этапов выполнения работы), 3) *результаты работы* (описание, интерпретация и сравнение результатов), 4) *заключение* (выводы, интерпретация полученных результатов), 5) *приложение* (код программы с построчными комментариями).

Лабораторная работа 3. Разработка онтологии предметной области

1. Цель работы

Изучение технологии создания онтологий, формирование навыков формализации знаний, построения концептуальных моделей и создания баз знаний для решения прикладных задач.

2. Указания к выполнению работы

Под онтологической моделью будем понимать концептуализированное представление информации о какой-либо области реальности, представленное в электронном виде.

Том Грубер (Основоположник инженерной онтологии) даёт следующее определение онтологии: «Онтология – эксплицитная спецификация концептуализации». Формально онтология состоит из терминов, организованных в таксономию, их определений и атрибутов, а также связанных с ними аксиом и правил вывода.

Созданию онтологии предметной области предшествует создание концептуальной модели этой области.

Алгоритм построения концептуальной модели

1. Декомпозция. Выделение объектов
2. Идентификация объектов
3. Классификация
4. Описание свойств

Рассмотрим пример построения концептуальной модели (рис. 3.1).

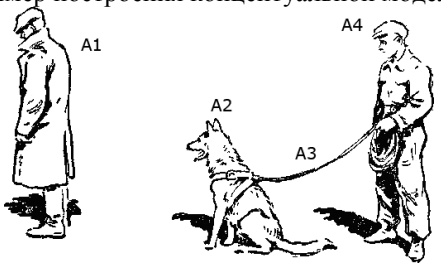


Рис. 3.1 – Пример информационного сообщения

1. Идентифицируем на этом рисунке четыре индивидуальных объекта: человек, собака, поводок, человек.

2. Присвоим им уникальные идентификаторы – соответственно, A1, A2, A3 и A4.

3. Определим набор классов, которые понадобятся для классификации выделенных нами объектов, и объединим их в иерархии (табл. 3.1).

Таблица 3.1 – Иерархия классов

По активности:		По типу:			
Предмет		Человек		Собака	Поводок
Действующее лицо	Статический предмет	Подозреваемый	Кинолог		

4. Зададим логические утверждения для созданных классов:

- Ни одно действующее лицо не является предметом, и наоборот.
- Классы человек, собака, поводок также имеют не пересекающиеся множества значений
- Все люди (в пределах нашей модели) являются либо подозреваемыми, либо кинологами.
- Все подозреваемые являются людьми. Все кинологи являются людьми.
- Все предметы (в пределах нашей модели) являются либо действующими лицами, либо статическими предметами.
- Все действующие лица являются предметами. Все статические предметы являются предметами.

5. Классифицируем наши индивидуальные объекты:

- А1 относится к классам Действующее лицо и Подозреваемый.
- А2 относится к классам Действующее лицо и Собака.
- А3 относится к классам Предмет и Поводок.
- А4 относится к классам Действующее лицо и Кинолог.

6. Определим набор свойств-связей, которые понадобятся для описания взаимодействия объектов (названия свойств-связей выделены курсивом):

- А держит в руке Б.
- А надет на Б.
- А сторожит Б.
- А находится под стражей Б.

7. Опишем характеристики и ограничения для свойств.

- «Держит в руке» относится к объектам класса Человек, а его значениями являются объекты класса Статический предмет.
- «Надет на» относится к объектам класса Действующее лицо, а его значениями являются объекты класса Предмет.
- «Сторожит» и «Находится под стражей» относятся к объектам класса Действующее лицо, и их значениями являются также объекты класса Действующее лицо.

Все перечисленные свойства могут иметь от 0 до любого количества значений.

8. Опишем логические утверждения для свойств:

- Если А сторожит Б, то Б находится под стражей А.

9. Запишем значения свойств для наших объектов:

- А4 держит в руке А3. А3 надет на А2. А2 сторожит А1.

Таким образом, мы получили минимальное по содержанию, но полное по структуре описание сцены, изображенной на рис. 11. В приведенном виде модель позволяет нам сделать только один вывод – о том, что А1 находится под стражей А2. Для придания этой модели практического смысла ее нужно существенно расширить – например, описать те факты, что собака бросится на задержанного, если он попытается бежать, а кинолог даст команду и перестанет держать поводок. Полная с прагматической точки зрения модель должна содержать:

- описание целей (для подозреваемого цель – сбежать, для кинолога – предотвратить побег),
- возможных действий (бежать, напасть), и
- их последствий (обездвижен, ранен).

3. Содержание работы

1. Изучение интерфейса среды Protégé
2. Декомпозиция поставленной задачи и построение её концептуальной модели
3. Создание прототипа онтологии
4. Создание запросов и тестирование онтологии

4. Порядок проведения работы

Задание 1. Ознакомьтесь с интерфейсом Protégé.

Задание 2. Создайте простую онтологию иллюстрирующую отношение Работодатель – Работник.

Задание 3. Сформулируйте запросы на извлечение информации из составленной базы знаний.

Задание 4. Постройте концептуальную модель ситуации, изображенной на картинке согласно варианту. Требования к количеству элементов в модели: классов – не менее 5, экземпляров классов – не менее 5, отношений (свойств объектов) – не менее 5, атрибутов (свойств данных) – не менее 5. Общее количество элементов онтологии (классов, экземпляров, отношений и атрибутов) – 50-80.

Задание 3. Реализуйте прототип онтологии в Protege согласно составленной концептуальной модели.

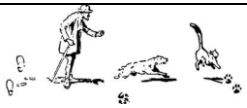
Задание 4. Постройте не менее 5 запросов на извлечение информации из составленной базы знаний.

Задание 5. Проведите интерпретацию результатов работы и составьте отчёт.

5. Варианты заданий

Варианты к заданию представлены в табл. 3.1.

Таблица 3.1. – Варианты для задания

1. 	2. 	3. 
4 	5. 	6. 
7. 		
8. 	9. 	10. 
11. 	12. 	

6. Контрольные вопросы

– Понятие онтологии. Формальная модель онтологии.

– Состояние современной глобальной сети. Семантическая паутина (Semantic web). Рекомендации W3C, связанные с Semantic Web.

– Структура онтологии. Сфера применений онтологии. Общие отношения для различных онтологий:

– Web ontology language (OWL).

– Редакторы онтологий. Protégé

– Общая схема взаимосвязи онтологий. Понятие метаонтологии.

– Составление прототипа онтологии (по вариантам).

7. Содержание отчета

Отчёт должен содержать: 1) *постановку задачи* (исходные данные), 2) *ход работы* (краткое описание этапов выполнения работы), 3) *результаты работы* (описание, интерпретация и сравнение результатов), 4) *заключение* (выводы, интерпретация полученных результатов), 5) *приложение* (код программы с построчными комментариями).

Лабораторная работа 4. Построение нечёткого аппроксиматора

1. Цель работы

Изучение основных функций пакета Fuzzy Logic Toolbox программной среды MatLab, приобретение навыков построения нечеткой аппроксимирующей системы

2. Указания к выполнению работы

Чёткая логика – детерминированная логика, допускающая только две оценки: истина и ложь. Она основана на двузначной булевой алгебре и чётких множествах.

Чёткое множество – совокупность различных элементов, мыслимая как единое целое. $X = \{x\}$, $x \in X$.

Нечёткая логика базируется на понятии нечёткого множества. Нечёткое множество A – математический объект, представляющий собой множество, принадлежность к которому представляет собой не отношение, а функцию; совокупность универсального множества X и характеристической функции $\mu_A(x)$, которая характеризует степень принадлежности элемента x нечёткому множеству.

Функция $\mu_A(x)$ принимает значения в некотором линейно упорядоченном множестве принадлежностей M. Часто в качестве M выбирается отрезок [0, 1].

По аналогии можно вывести понятия нечёткого высказывания и нечётких выводов.

В нечеткой логике вводится понятие лингвистической переменной, значениями которой являются не числа, а слова естественного языка, называемые термами. Например, лингвистическая переменная «скорость» может иметь значения «высокая», «средняя», «очень низкая» и т. д. Фразы, значение которых принимает переменная, в свою очередь, являются именами нечетких переменных. Значения лингвистической переменной (ЛП) определяются через нечеткие множества (НМ), которые, в свою очередь, определены на некотором базовом наборе значений или базовой числовой шкале, имеющей размерность. Каждое значение ЛП определяется как нечеткое множество (например, НМ «низкий рост»).

Назначение и возможности пакета Fuzzy Logic Toolbox

Пакет Fuzzy Logic Toolbox (пакет нечеткой логики) – это совокупность прикладных программ, относящихся к теории размытых или нечетких множеств и позволяющих конструировать так называемые нечеткие экспертные и/или управляющие системы.

Основные возможности пакета:

- построение систем нечеткого вывода (регуляторов, аппроксиматоров зависимостей);
- построение адаптивных нечетких систем (гибридных нейронных сетей);
- интерактивное динамическое моделирование в Simulink. Пакет позволяет работу:
- в режиме графического интерфейса;
- в режиме командной строки;
- с использованием блоков и примеров пакета Simulink.

Графический интерфейс Fuzzy Logic Toolbox

Состав графического интерфейса. В состав программных средств Fuzzy Logic Toolbox входят следующие основные программы, позволяющие работать в режиме графического интерфейса:

- редактор нечеткой системы вывода Fuzzy Inference System Editor (FIS Editor или FIS-редактор) вместе со вспомогательными программами – редактором функций принадлежности (Membership Function Editor), редактором правил (Rule Editor), просмотрщиком правил (Rule Viewer) и осмотрщиком поверхности отклика (Surface Viewer);
- редактор гибридных систем (ANFIS Editor, ANFIS-редактор);
- программа нахождения центров кластеров.

Для запуска редактора нечёткой системы необходимо в командном окне Matlab запустить команду Fuzzy.

3. Содержание работы

1. Построение нечёткого аппроксиматора
2. Исследование влияния вида функции принадлежности на точность аппроксимации

4. Порядок проведения работы

Задание 1. Знакомство с интерфейсом Fuzzy Logic Toolbox

Задание 2. Задание начальных данных

Задание 3. Формирование базы правил нечёткого аппроксиматора

Задание 4. Оценка качества аппроксимации

Задание 5. Исследование влияния вида функции принадлежности на точность аппроксимации и выбор наиболее подходящей

Задание 6. Определение вида приближающей функции. Интерпретация полученных результатов.

5. Варианты заданий

Варианты к заданию представлены в табл. 4.1

Таблица 4.1 – Исходные данные к заданию

№	Задание								
1	*x*	1.20	1.57	1.94	2.31	2.68	3.05	3.42	3.79
	y	2.56	2.06	1.58	1.25	0.91	0.66	0.38	0.21
2	*x*	1.73	2.56	3.39	4.22	5.05	5.89	6.70	7.53
	y	0.63	1.11	1.42	1.96	2.30	2.89	3.29	3.87
3	*x*	-4.38	-3.84	-3.23	-2.76	-2.22	-1.67	-1.13	-0.60
	y	2.25	2.83	3.44	4.51	5.29	6.55	8.01	10.04
4	*x*	1.00	1.64	2.28	2.91	3.56	4.29	4.84	5.48
	y	0.28	0.19	0.15	0.11	0.09	0.08	0.07	0.06
5	*x*	5.89	3.84	6.19	9.22	7.87	6.29	4.43	8.91
	y	79.31	57.43	60.66	90.55	92.12	71.30	70.50	91.25
6	*x*	2.91	2.94	6.35	6.58	3.80	6.43	0.57	5.96
	y	82.16	61.02	44.56	82.52	99.19	70.24	63.23	66.48
7	*x*	1.23	1.79	2.24	2.76	3.20	3.68	4.16	4.64
	y	2.10	2.84	3.21	3.96	4.86	6.06	7.47	9.25
8	*x*	-4.38	-3.84	-3.23	-2.76	-2.22	-1.67	-1.13	-0.60
	y	1.73	2.56	3.39	4.22	5.05	5.89	6.70	7.53
9	*x*	2.56	2.06	1.58	1.25	0.91	0.66	0.38	0.21
	y	0.63	1.11	1.42	1.96	2.30	2.89	3.29	3.87
10	*x*	79.31	57.43	60.66	90.55	92.12	71.30	70.50	91.25
	y	5.89	3.84	6.19	9.22	7.87	6.29	4.43	8.91

6. Контрольные вопросы

– Понятие нечёткого множества и лингвистической переменной. Функция принадлежности нечёткого множества.

– Операции над нечёткими множествами.

– Нечеткие высказывания. Операции над ними.

– Чёткие соответствия и отношения. Нечёткие соответствия и отношения. Их способы задания.

– Нечёткая логика в Matlab (Fuzzy toolbox).

– Расчёт значений функции принадлежности.

– Аппроксимация. Нечёткая аппроксимация

7. Содержание отчета

Отчёт должен содержать: 1) *постановку задачи* (исходные данные), 2) *ход работы* (краткое описание этапов выполнения работы), 3) *результаты работы* (описание, интерпретация и сравнение результатов), 4) *заключение* (выводы, интерпретация полученных результатов), 5) *приложение* (код программы с построчными комментариями).

Лабораторная работа 5. Формирование базы правил нечёткой системы

1. Цель работы

Изучение методики и формирование навыков создания нечётких информационных систем.

2. Указания к выполнению работы

Функции принадлежности

Инструментарий нечеткой логики (ИНЛ) в составе пакета Matlab содержит 11 встроенных типов функций принадлежности (ФП), формируемых на основе кусочно-линейных функций, распределения Гаусса, сигмоидной кривой, квадратических и кубических полиномиальных кривых. К наиболее простым ФП можно отнести треугольную и трапециевидную. Наименование треугольной ФП — trimf (triangle membership function). В параметрическом виде она представляет собой не что иное, как набор трех точек, образующих треугольник.

Описание функции: y = trimf (x, a b, c), где вектор x — базовое множество, на котором определяется ФП. Величины a и c задают основание треугольника, b — его вершину.

Рассмотрим примеры использования различных ФП в системе.

Примеры представляют собой фрагменты программ и комментариев на языке пакета Matlab.

Пример1. Программа использования ФП trimf

```
x=0:0.1:10; % Задается базовое множество
y =trimf (x,[3 6 8]); % Определяется треугольная ФП
plot (x,y); % Выводится график функции
xlabel('trimf (x, P), P = [3 6 8]'); % Подписывается график под осью абсцисс
```

Трапециевидная ФП — trapmf (trapezoid membership function) — отличается от предыдущей функции лишь тем, что имеет верхнее основание.

Описание функции:

y = trapmf (x, [a b c d]), где параметры a n d — нижнее основание трапеции; b и c — верхнее основание трапеции.

Операции с нечеткими множествами

Выделяют три основные логические операции с нечеткими множествами: конъюнкцию, дизъюнкцию и логическое отрицание. В среде Matlab существует возможность определять конъюнктивные и дизъюнктивные операторы с точки зрения минимаксной и вероятностной интерпретаций.

Минимаксная интерпретация является наиболее распространенной при построении нечетких систем. Тем не менее на практике довольно часто используется альтернативная вероятностная интерпретация конъюнктивных и дизъюнктивных операторов.

3. Содержание работы

1. Формализация задачи
2. Построение нечёткой системы

4. Порядок проведения работы

Задание 1. Формализовать условия задачи, дать её математическую постановку задачи: входные/выходные данные, цель работы системы.

Задание 2. Ввести лингвистические переменные, их термы-элементы (нечёткие множества) с допустимыми значениями базовой шкалы.

Задание 3. Выбрать, обосновать и задать функцию принадлежности каждой величины тому или иному терму (нечёткому множеству).

Задание 4. Обосновать и сформировать базу нечетких правил.

Задание 5. Исследовать влияние параметров и вида функции принадлежности на точность результата. Выбрать наиболее подходящую конфигурацию нечёткой системы. Для оценки качества работы нечёткой системы следует использовать справочные таблицы

Задание 5. Провести интерпретацию результаты работы и составить отчёт.

5. Варианты заданий

1. Принятие решения об инвестировании в проект
2. Управление траекторией полёта тела
3. Управление системой расходования жидкости
4. Принятие решения о покупке
5. Оценивание надежности заёмщика
6. Оценка квалификации работника
7. Управление скоростью движения автомобиля
8. Оценка возможности выигрыша
9. Система управления климатом
10. Оценивание степени безопасности условий труда

6. Контрольные вопросы

– Понятие нечёткого множества и лингвистической переменной. Функция принадлежности нечёткого множества.
– Операции над нечёткими множествами.
– Нечеткие высказывания. Операции ними.

– Чёткие соответствия и отношения. Нечёткие соответствия и отношения. Их способы задания.

– Нечёткие логический вывод. Его этапы и особенности.

– Нечёткая логика в Matlab (Fuzzy toolbox).

– Практическое задание: расчёт значений функции принадлежности.

7. Содержание отчета

Отчёт должен содержать: 1) *постановку задачи* (исходные данные), 2) *ход работы* (краткое описание этапов выполнения работы), 3) *результаты работы* (описание, интерпретация и сравнение результатов), 4) *заключение* (выводы, интерпретация полученных результатов), 5) *приложение* (код программы с построчными комментариями).

Лабораторная работа 6. Исследование алгоритма нечёткой классификации

1. Цель работы

Исследование работы алгоритма нечёткой классификации, формирование навыков построения нечётких классификаторов для решения прикладных задач.

2. Указания к выполнению работы

Нечеткая классификация является развитием подхода экспертных систем. Основное отличие и достоинство нечеткой классификации – это возможность формулирования достоверных классификационных заключений исходя из неполных и не вполне достоверных входных посылок.

При сохранении математического аппарата, разработанного для систем четкой логики, в нечетких логических системах решена задача преобразования численной и качественной информации в степень принадлежности значений конкретным нечетким множествам (НМ). НМ описываются посредством функций принадлежности, ставящих в соответствие множеству значений из области определения непрерывной переменной множество значений истинности из интервала [0, 1].

Основные этапы нечеткого логического вывода связаны с процессом формирования классификационных заключений:

1) этап введения нечеткости связан с преобразованием посредством входных функций принадлежности каждого из четких входных значений x_i i=1, …, n – число входных значений в истинность соответствующей посылки μ_{xi} i=1, …, n, для каждого из классификационных правил;

2) этап логического вывода соответствует формированию заключения (нечеткого подмножества) по каждому правилу μ_{Ri} i=1, …, m – количество классификационных правил, исходя из истинности посылок μ_{xi} i=1, …, m;

3) этап композиции нечетких подмножеств позволяет формировать нечеткие подмножества классификационных заключений μ_{Ci} i=1, …, p – число выходов классификатора по правилам μ_{Ri} i=1, …, m, посредством выходных функций принадлежности;

4) этап объединения нечетких подмножеств μ_{Ci} i=1, …, p, и приведение к четкости приводит к формированию выходного четкого значения y.

По аналогии с ЭС нечеткие логические системы основаны на базе знаний квалифицированных специалистов ИБ в виде системы правил If-Then. Однако существенно расширяют область применения ЭС за счет возможности решения задачи классификации исходя из не вполне достоверных данных и качественной информации.

3. Содержание работы

1. Постановка задачи
2. Формирование базы правил нечёткой системы
3. Тестирование работы нечеткой системы

4. Порядок проведения работы

1. Создайте новый проект. Сформируйте систему конъюнктивных продукционных правил, например:

«If (Угроза1) and (Угроза2) and … and (Угроза5)

Then (Механизм защиты1)»

2. Сформируйте систему дизъюнктивных продукционных правил вида:

«If (Угроза1) or (Угроза2) or … or (Угроза5)

Then (Механизм защиты1)»

3. В соответствии с теорией факторов уверенности для каждой из баз знаний задайте экспертные оценки в виде двух функций: меры доверия MB(H,E) и меры недоверия MD (H, E). Функции указывают, соответственно, степень увеличения доверия к гипотезе H, если факт E произошел, и степень увеличения недоверия к гипотезе H, если факт E имел место.

4. Введите рассчитанные значения факторов уверенности в соответствующие строки сформированных конъюнктивных и дизъюнктивных систем продукционных правил. Сохраните каждую из баз знаний в виде структуры нейронной сети для дальнейших исследований.

Пример работы:

Исходя из оговоренного в задании поля угроз (табл.6.1), которые используются в качестве посылок, необходимо сформулировать продукционные правила базы знаний.

Таблица 6.1 – Антецеденты базы знаний

Координаты входного вектора	Посылки для части If продукционных правил – заданный перечень угроз
1	Вирусная атака на ПК
2	Вирусная атака на ЛВС
3	Угрозы целостности информации
4	Угрозы конфиденциальности информации
5	Попытки НСД к информации

В качестве заключений продукционных правил следует использовать заданный перечень используемых механизмов защиты – МЗ (табл. 6.2)

Таблица 6.2 – Консеквенты базы знаний

Координаты входного вектора	Заключения для части Then продукционных правил – заданный перечень МЗ
1	Обнаружение вирусной атаки на ПК
2	Обнаружение вирусной атаки на ЛВС
3	Обнаружение искажения программ и данных
4	Криптографическая защита информации на носителях
5	Криптографическая защита информации в сетях передачи данных
6	Парольная защита
7	Управление полномочиями пользователей
8	Учет работы пользователей

То есть необходимо описать системой продукционных правил соответствие каждого из восьми механизмов защиты пяти типам заданных угроз.

5. Варианты заданий

1. Диагностика состояния здоровья пациента.
2. Определение темперамента человека
3. Поиск неисправности автомобиля
4. Поиска неисправности в компьютере
5. Определение типа геологической породы

6. Контрольные вопросы

– Постановка задачи классификации
– Нечёткая классификация
– Нечеткие отношения
– .База знаний нечёткой системы

7. Содержание отчета

Отчёт должен содержать: 1) *постановку задачи* (исходные данные), 2) *ход работы* (краткое описание этапов выполнения работы), 3) *результаты работы* (описание, интерпретация и сравнение результатов), 4) *заключение* (выводы, интерпретация полученных результатов), 5) *приложение* (код программы с построчными комментариями).

Лабораторная работа 7. Построение нейросетевого аппроксиматора

1. Цель работы

Изучение основ работы с Neural Networks Toolbox в среде Matlab, формирование навыков построения нейросетевых систем их использования для решения задач аппроксимации.

2. Указания к выполнению работы

Нейронная сеть - это набор нейронов, определенным образом связанных между собой.

Нейрон получает входные сигналы (исходные данные либо выходные сигналы других нейронов нейросети) через несколько входных каналов. Каждый входной сигнал проходит через соединение, имеющее определенную интенсивность (или вес); этот вес соответствует синаптической активности биологического нейрона. С каждым нейроном связано определенное пороговое значение. Вычисляется взвешенная сумма входов, из нее вычитается пороговое значение, и в результате получается величина активации нейрона. Сигнал активации преобразуется с помощью функции активации (или передаточной функции). В результате получается выходной сигнал нейрона.

Наиболее распространенные функции активации: пороговая, сигнум (или модифицированная пороговая функция), логистическая, гиперболический тангенс, линейная, линейная ограниченная, радиальнобазисная и др. Как правило, передаточные функции всех нейронов в сети фиксированы, а веса являются параметрами сети и могут изменяться.

Наиболее распространенными являются многослойные сети, в которых нейроны объединены в слои. Слой - это совокупность нейронов, на которые в каждый такт времени параллельно поступает информация от других нейронов сети. После того, как определено число слоев и число элементов в каждом из них, нужно найти значения для весов и порогов сети, которые минимизировали бы ошибку прогноза, выдаваемого сетью. Именно для этого служат алгоритмы обучения. С использованием собранных исторических данных веса и пороговые значения автоматически корректируются с целью минимизации этой ошибки. По сути, этот процесс представляет собой подгонку модели, которая реализуется сетью, к имеющимся обучающим данным. Ошибка для конкретной конфигурации сети определяется путем прогона через сеть всех имеющихся наблюдений и сравнения реально выдаваемых выходных значений с желаемыми (целевыми) значениями. Все такие разности суммируются в функцию ошибок,

значение которой и есть ошибка сети. В качестве функции ошибок чаще всего берется сумма квадратов ошибок

Самый известный вариант алгоритма обучения нейронной сети - так называемый алгоритм обратного распространения [2]. Существуют современные алгоритмы второго порядка, такие, как метод сопряженных градиентов и метод Левенберга-Маркара [3], которые на многих задачах работают существенно быстрее (иногда на порядок). Алгоритм обратного распространения в некоторых случаях имеет определенные преимущества.

Задачи аппроксимации экспериментальных данных можно решать с помощью искусственных нейронных сетей следующих типов: многослойного персептрона, сетей с радиально-базисными функциями, вероятностных сетей, обобщенно-регрессионных сетей.

Пакет Neural Networks Toolbox среды Matlab

Вызов GUI-интерфейса NNTool осуществляется командой nntool из строки командного окна. Основной интерфейс Neural Networks Toolbox (рис. 7.1) имеет следующие кнопки:

- Input Data – последовательность входов;
- Target data – последовательность целей;
- Input Daley States – начальные условия линии задержки входов;
- Networks – список нейронных сетей;
- Output Data – последовательность выходов;
- Error Data – последовательности ошибок сети;
- Layer Delay States – начальные условия линии задержки слоя;
- Help – кнопка вызова окна подсказки;
- New Data… – вызов окна формирования данных;
- New Network… – вызов окна создания новой нейронной сети;
- Import… – вызов окна импорта или загрузки данных;
- Export… – вызов окна экспорта или загрузки данных в файл.

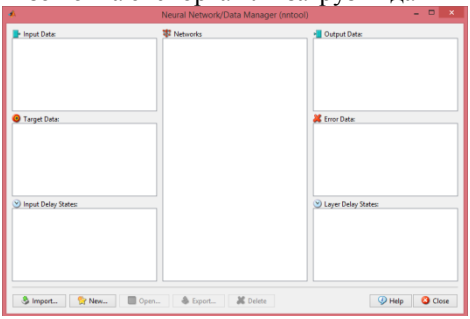


Рис. 7.1 – Основное окно Neural Networks Toolbox

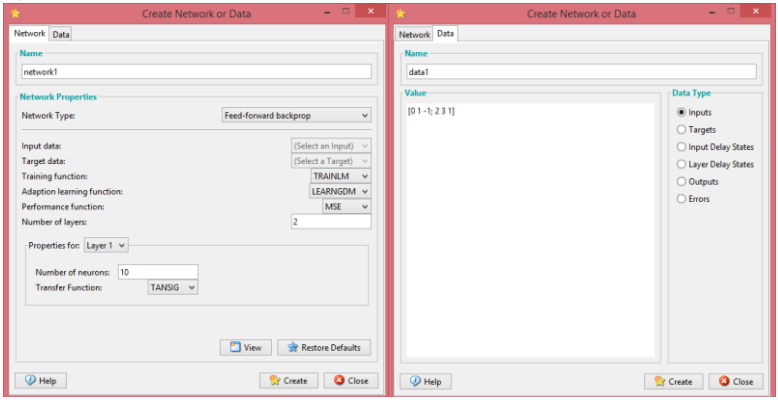
При создании нейронной сети, необходимо выполнить следующие операции:

1) сформировать последовательность входов и целей (кнопка New Data) либо загрузить их из рабочей области системы Matlab или из файла (кнопка Import);

2) создать новую нейронную сеть (кнопка New Network – рис. 7.2а) либо загрузить ее из рабочей области систем Matlab или из файла (кнопка Import);

3) выбрать тип нейронной сети и нажать кнопку Train…, чтобы открыть окно для задания параметров процедуры обучения;

4) открыть окно Network для просмотра, обучения, моделирования и адаптации сети.



а б

Рис. 7.2 – Окна: а) создания новой нейронной сети, б) ввода данных

Окно формирования данных (Create New Data), показанное на рис. 7.2б, содержит две области редактирования текста для записи имени вводимых данных (область Name) и ввода самих данных (область Value), а также 6 кнопок для указания типа вводимых данных:

Inputs (Входы) – последовательность значений входов;

Targets (Цели) – последовательность значений целей;

Input Delay States (состояния линии задержки (ЛЗ) входа) – начальные условия линии задержки на входе;

Layer Delay States (Состояния ЛЗ слоя) – начальные условия линии задержки в слое;

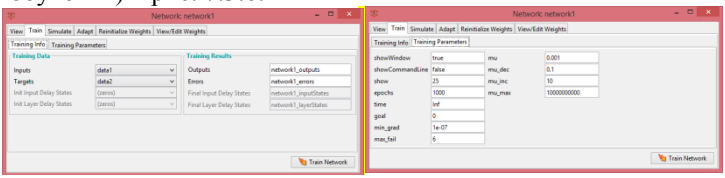
Outputs – последовательность значений выходов сети;

Errors – разность значений целей и выходов.

Обучение нейронной сети в GUI

Обучение нейронной сети производится на вкладке Train.

Панель имеет две закладки: 1) Training info (Информация об обучающих последовательностях) – рисунок 7.3а; 2) Training Parameters (Параметры обучения) – рис. 7.3.б.



а б

Рис. 7.3 – окна с информацией: а) об обучающих последовательностях, б) о параметрах обучения

Результаты обучения можно просмотреть в окне Network/Data Manager, выбрав кнопку Manager.

При необходимости можно экспортировать созданную нейронную сеть в рабочую область системы MatLab (нажав кнопку Export и далее, в открывшемся окне Export from Network/Data Manager – кнопки Select All (Выбрать все) и Export) и получить информацию о весах и смещениях непосредственно в рабочем окне системы, выполнив команду:

3. Содержание работы

1. Знакомство с интерфейсом Neural Networks Toolbox
2. Создание нейросетевого аппроксиматора
3. Исследование влияния типа сети и её параметров на результаты аппроксимации

4. Порядок проведения работы

Задание 1. Ознакомьтесь с интерфейсом Neural Networks Toolbox. Пользуясь справочными материалами, узнайте назначение каждого окна и поля.

Задание 2. Создайте, используя графический интерфейс пользователя, нейронную сеть типа Feed-forward backprop (с прямой передачей сигнала и обратным распространением ошибки) для аппроксимации функции согласно варианту.

Задание 3. Произведите обучение нейронной сети. Оцените результаты аппроксимации. Экспортируйте и сохраните в файл нейронную сеть.

Задание 4. Создайте новую нейронную сеть такого же типа, но с другими параметрами (функция обучения, функция качества обучения, функ-

ция активации). Повторите процедуру из задания 3 для сети с новой конфигурацией.

Задание 5. Выберите нейронную сеть другого вида (согласно варианту), проведите её обучение. Оцените результаты аппроксимации и сравните с предыдущими результатами. Экспортируйте и сохраните в файл нейронную сеть.

5. Варианты заданий

Варианты к заданию 2, 3 представлены в таблице 4.1.

Варианты к заданию 5 представлены в таблице 7.1.

Таблица 7.1 – Типы нейронных сетей

№	Тип сети	Число слоев
1	Perceptron	1
2	Probabilistic	2
3	Radial basis (exact fit)	2
4	Radial basis (fewer neurons)	2
5	Generalized regression	2

6. Контрольные вопросы

- Постановка задачи аппроксимации
- Нейронная сеть. Типы нейронных сетей
- Обучение нейронных сетей
- Работа с нейронными сетями в Matlab

7. Содержание отчета

Отчёт должен содержать: 1) *постановку задачи* (исходные данные), 2) *ход работы* (краткое описание этапов выполнения работы), 3) *результаты работы* (описание, интерпретация и сравнение результатов), 4) *заключение* (выводы, интерпретация полученных результатов), 5) *приложение* (код программы с построчными комментариями).

Лабораторная работа 8. Применение нейросетей для распознавания образов

1. Цель работы

Формирование навыков работы в пакете Neural Networks Toolbox среды Matlab, навыков решения задач распознавания образов с применением нейросетей**.**

2. Указания к выполнению работы

Искусственные нейронные сети представляют собой математическую модель функционирования биологических нейронных сетей – сетей нервных клеток живого организма. Как и в биологической нейронной сети, основным элементом искусственной нейронной сети является нейрон. Соединенные между собой нейроны, образуют слои, количество которых может варьироваться в зависимости от сложности нейронной сети и решаемых ею задач.

Одной из актуальных задач является распознавание визуальных образов. Машины, способные распознавать росписи на бумагах, символы на банковских карточках или чеках, существенно облегчают человеческий труд и ускоряют рабочий процесс, при этом снижают риск ошибки за счет отсутствия человеческого фактора.

Для решения задачи распознавания образов с помощью нейронных сетей необходимо выполнить предобработку исходных данных. Представим код функции ImgRead(‘*путь к файлу’)*, которая будет считывать необходимые признаки символов с графического файла в нужном нам формате.

```
function y = Imgread(x)
%UNTITLED Summary of this function goes here
% Detailed explanation goes here
img=imread(x);
img1=img(:,:,1)';
img2=reshape (img1,[35, 1]);
for i=1:35
 if (img2(i,1)==0)
 img2(i,1)=1;
 end
end
for i=1:35
 if (img2(i,1)==255)
 img2(i,1)=0;
 end
end
y=img2;
end
```

Использование этой функции позволит создать матрицу признаков русского алфавита следующим образом:

```
images1=Imgread('C:\alphabet\А.png');
…
images33=Imgread('C:\alphabet\Я.png');
RA=[images1,images2,images3,images4,images5,images6,images7,images8,...
images9,images10,images11,images12,images13,images14,images15,images16,...
images17,images18,images19,images20,images21,images22,images23,...
images24,images25,images26,images27,images28,images29,images30,images31,...
images32,images33];
```

Таким образом была создана единичная матрица 33×33, которая будет выступать в качестве матрицы целей.

Следующий код позволит записать размерности матрицы и определить некоторые необходимые переменные:

```
%Создаем переменные для создания нейронной сети в NNtool
P=double(RA);
T=eye(33);
[R,Q] = size(P);
[S2,Q] = size(T);
```

Для обучения сети нам понадобятся данные с шумом. Создадим эти данные можно следующим образом:

```
%Создаем переменные для обучения нейросети на зашумленных данных в
%NNtool
P1=P;
T1=T;
for i=1:100
 P1=[P1,P+rand(R,Q)*0.1,P+rand(R,Q)*0.2];
 T1=[T1,T,T];
end
```

Для симуляции сети понадобится переменная, содержащая в себе набор признаков одной буквы, например, буквы И:

```
noisy10 = P(:,10) + randn(35,1)*0.2;
plotchar(noisy10); % Зашумленный символ И
```

Далее необходимо открыть Neural Networks Toolbox, импортировать данные и приступить к обучению нейронной сети.

3. Содержание работы

1. Предобработка данных
2. Обучение нейронной сети
3. Тестирование нейронной сети

4. Порядок проведения работы

Задание 1. Создайте (или скачайте готовый набор) изображения букв русского алфавита размером 7×5 пикселей.

Задание 2. Создайте в Matlab функцию ImgRead для считывания данных из файлов. Создайте матрицу признаков русского алфавита. Объявите необходимые переменные для создания нейронной сети.

Задание 3. Создайте данные (образ одной из букв) с шумом. Выведите на экран исходный образ и его зашумленный вариант.

Задание 4. Импортируйте данные в Neural Networks Toolbox из Matlab. Создайте в нейронную сеть типа Feed-forward backprop. В качестве входных данных используйте матрицу признаков русского алфавита P, в качестве цели – единичную матрицу T.

Задание 5. Произведите обучение нейронной сети. Проверьте правильность её работы на зашумлённом символе. Экспортируйте и сохраните в файл нейронную сеть.

Задание 6. Создайте новую нейронную сеть такого же типа, но с другими параметрами (функция обучения, функция качества обучения, функция активации). Повторите процедуру из задания 5 для сети с новой конфигурацией.

Задание 7. Выберите нейронную сеть другого вида (согласно варианту), проведите её обучение. Оцените результаты распознавания и сравните с предыдущими результатами. Экспортируйте и сохраните в файл нейронную сеть.

5. Варианты заданий

В качестве вариантов к заданию 3-5 можно использовать таблицу букв русского алфавита.

В качестве вариантов к заданию 6 можно использовать данные из табл. 7.1.

6. Контрольные вопросы

- Постановка задачи распознавания образов.
- Постановка задачи классификации.
- .Понятие персептрона.
- Нейронная сеть.
- Нейросетевой классификатор.

7. Содержание отчета

Отчёт должен содержать: 1) *постановку задачи* (исходные данные), 2) *ход работы* (краткое описание этапов выполнения работы), 3) *результаты работы* (описание, интерпретация и сравнение результатов), 4) *заключение* (выводы, интерпретация полученных результатов), 5) *приложение* (код программы с построчными комментариями).

Лабораторная работа 9. Исследование сети Кохонена и алгоритма обучения без учителя

1. Цель работы

Изучение архитектуры самоорганизующихся нейронных слоев Кохонена, а также приобретение навыков построения самоорганизующихся слоев для исследования топологической структуры данных, навыков решения задачи кластеризации данных с применением нейронных сетей.

2. Указания к выполнению работы

Сети Кохонена, или самоорганизующиеся карты (Kohonen maps), предназначены для решения задач распознавания, когда обучающая последовательность образов отсутствует (обучение без учителя). Сеть Кохонена является двухслойной. Она содержит слой входных нейронов и собственно слой Кохонена. Слой Кохонена может быть одномерным, двумерным и трехмерным. В первом случае нейроны расположены в цепочку; во втором – они образуют двумерную сетку (обычно в форме квадрата или прямоугольника), а в третьем – трехмерную систему. Определение весов нейронов слоя Кохонена основано на использовании алгоритмов автоматической классификации (кластеризации или самообучения).

На вход сети подаются последовательно значения векторов $x_l=(x_1, x_2, \ldots x_n)_l$, представляющих отдельные последовательные наборы данных для поиска кластеров, то есть различных классов образов, причем число этих кластеров заранее неизвестно. На стадии обучения (точнее самообучения) сети входной вектор попарно сравнивается со всеми векторами всех нейронов сети Кохонена. Вводится некоторая функция близости d (например, в виде евклидова расстояния). Активный нейрон с номером c слоя Кохонена, для которого значение функции близости $d(x, w_c)$ между входным вектором , характеризующим некоторый образ, к векторам максимально, «победителем». При этом образ, характеризующийся вектором , будет отнесен к классу, который представляется нейроном-«победителем».

Самоорганизующийся слой Кохонена – это однослойная нейронная сеть с конкурирующей передаточной функцией compet, которая анализирует выходные значения нейронов слоя и выдаёт в качестве результата наибольшее из этих значений (значение нейрона-победителя). Функция newc создает слой конкурирующих нейронов (слой Кохонена):

```
net = newc(PR, S, KLR, CLR),
```

где PR – матрица минимальных и максимальных значений для R входных элементов; S – количество нейронов; KLR – уровень обученности Кохонена, по умолчанию 0,01; CLR – рекомендуемый уровень обу-

ченности, по умолчанию 0,001. Соперничающий слой представляет собой слой с функцией вычисления расстояний negdist, сетевой функцией суммирования netsum и функцией акти-вации compet. Слой имеет веса на входах и bias.

Функция newsom создает самоорганизующуюся сеть – карту (SOM):

net = newsom(PR, [D1, D2, ...], TFCN, DFCN, OLR, OSTEPS, TLR, TND),

где PR – матрица минимальных и максимальных значений для R входных элементов; Di – размер i-го слоя; TFCN – функция топологии; DFCN – функция расстояния; OLR – уровень обученности фазы упорядочения, по умолчанию 0,9; OSTEPS – шаги фазы упорядочения, по умолчанию 1000; TLR – уровень обу- ченности фазы настройки, по умолчанию 0,02; TND – расстояние соседства фа- зы настройки, по умолчанию 1.

Применение нейронной сети Кохонена для кластеризации

Сформировать координаты случайных точек и построить их расположение на плоскости можно следующим образом:

```
c = 8; %Число кластеров
n = 6; %Число векторов в кластере
d = .5; %Среднее отклонение от центра кластера
x = [-10 10; -5 5];
[r,q] = size(x);
minv = min(x')';
maxv = max(x')';
v = rand(r,c).*((maxv-minv)*ones(1,c)+minv*ones(1,c));
t = c*n; % число точек
v = [v v v v v v];
v = v+randn(r,t)*d; % координаты точек
P = v;
```

Импорт данных, создание, настройка, обучение нейронной сети и её симуляция осуществляются при помощи графического интерфейса Neural Networks Toolbox.

Следующий алгоритм демонстрирует процедуру обучения самооргани- зующейся нейронной сети Кохонена.

```
plot(P(1,:), P(2,:), '*g');
title('Input vectors');
xlabel('P(1,:)');
ylabel('P(2,:)');
hold on;
nclusters = 6;
a1 = -3;
a2 = 10;
b1 = -3;
b2 = 9;
net = newc([a1 a2; b1 b2], nclusters, 0.1, 0.0005);
wo = net.IW{1};
bo = net.b{1};
net.trainParam.epochs=100;
net.trainParam.show=20;
```

```
net = train(net,P);
w = net.IW{1};
bn = net.b{1};
plot(w(:,1),w(:,2),'kp');
```

3. Содержание работы

1. Внесение исходных данных.

2. Создание сети Кохонена, анализ её структуры, обучение и моделирование.

3. Кластеризация данных.

4. Порядок проведения работы

Задание 1. Создать слой Кохонена для двух векторов входа, проанализировать его структурную схему и параметры вычислительной модели, произвести обучение сети и моделирование, выполнив следующие команды:

```
P = [.1 .8 .1 .9; .2 .9 .1 .8]; % для обучения слоя;
net = newc([01;01],2); % создание слоя;
gensim (net); % структура слоя;
net = train(net,P); % обучение слоя;
w = net.Iw{1,1}; % веса после обучения;
b =net.b{1}; % смещение после обучения;
plot(P(1,:),P(2,:),'+k')
title ('Векторы входа'),xlabel('P(1,:)'), ylabel('P(2,:)')
hold on
plot (w, 'or')
P1= [0.2:0.1:0.7; 0.2:0.1:0.7];
y = sim(net,P1)
yc = vec2ind(Y)
```

Задание 2. Создать одномерную карту Кохонена из 10 нейронов, обучить её на последовательности из 100 двухэлементных векторов единичной длины, распределенных равномерно в пределах от 0 до 90º, построить график распределения векторов по кластерам и выполнить моделирование сети для одного вектора входа, выполнив следующие команды:

```
angels=0 : 0.5 +pi/99 : 0.5*pi;
p=[sin(angels); cos(angels)];
plot(P(1, 1:10:end), P(2, 1:10:end), '*8')
hold on
net=newsom([0 1 ;0 1], [10]);
net.trainparam.epochs=2000;
net.trainparam.show=100;
[net,tr]=train(net,P);
plotsom(net.iw{1,1}, net.layers{1}.distances)
figure(2)
a=sim(net,P) % – моделирование на обучающем
% множестве и построение
% столбцовой диаграммы.
```

```
a=sim(net,[1;0]) % – отнесен к 10-му кластеру
```

Задание 3. Оцифровать и провести кластеризацию данных (табл. 2), используя слой Кохонена и самоорганизующуюся карту SOM согласно варианту.

5. Варианты заданий

Варианты заданий формируются с помощью генератора случайных чисел, посредством разброса 30 точек в плоскости на интервале [-1; 1].

6. Контрольные вопросы

– В чем заключается задача кластеризации?
– Какую структуру имеет нейронная сеть Кохонена?
– Каким алгоритмом обучается нейронная сеть Кохонена?
– Дайте определение самоорганизующейся карты Кохонена.

7. Содержание отчета

Отчёт должен содержать: 1) *постановку задачи* (исходные данные), 2) *ход работы* (краткое описание этапов выполнения работы), 3) *результаты работы* (описание, интерпретация и сравнение результатов), 4) *заключение* (выводы, интерпретация полученных результатов), 5) *приложение* (код программы с построчными комментариями).

Лабораторная работа 10. Создание модели классификации текстов

1. Цель работы

Изучение возможностей Text Analytics Toolbox в Matlab, формирование навыков построения классификаторов слабоструктурированных данных.

2. Указания к выполнению работы

Text Analytics Toolbox в Matlab позволяет создавать простые классификаторы текстов на основе расчета частоты встречаемости слов по модели bag-of-words (мешок слов). Рассмотрим создание простой модели классификации метеорологических сводок на основании их текстового содержимого.

Чтобы воспроизвести результаты этого примера необходимо установить rng в значении 'default'.

Представим код для чтения данных из файла:

```
filename = "weatherReports.csv";
data = readtable(filename,'TextType','string');
head(data)
```

Удалим строки с пустыми сообщениями.

```
idx = strlength(data.event_narrative) == 0;
data(idx,:) = [];
```

Преобразуем метки в столбца event_type таблицы и просмотрим распределение классов в данных с помощью гистограммы (рис. 10.1).

```
data.event_type = categorical(data.event_type);
figure
h = histogram(data.event_type);
xlabel("Class")
ylabel("Frequency")
title("Class Distribution")
```

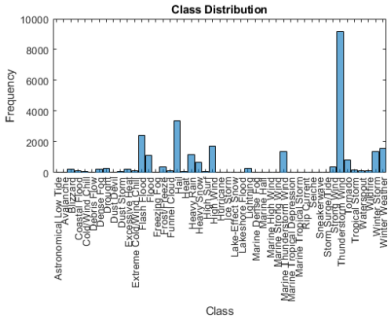


Рис. 10.1 – Гистограмма классов

Получим значения частотной встречаемости классов из гистограммы:

```
classCounts = h.BinCounts;
classNames = h.Categories;
```

Найдём классы, содержащие менее десяти наблюдений, и удалим эти редкие классы из данных.

```
idxLowCounts = classCounts < 10;
infrequentClasses = classNames(idxLowCounts);
idxInfrequent = ismember(data.event_type,infrequentClasses);
data(idxInfrequent,:) = [];
```

Разделим данные в обучающий раздел и тестовый набор. Процент удержания составляет 10%.

```
cvp = cvpartition(data.event_type,'Holdout',0.1);
dataTrain = data(cvp.training,:);
dataTest = data(cvp.test,:);
```

Извлечём текстовые данные и метки из таблиц.

```
textDataTrain = dataTrain.event_narrative;
textDataTest = dataTest.event_narrative;
YTrain = dataTrain.event_type;
YTest = dataTest.event_type;
```

Подготовка текстовых данных для анализа

Используем пример предварительной обработки функции Preprocess WeatherNarratives для подготовки текстовых данных.

```
documents = preprocessWeatherNarratives(textDataTrain);
documents(1:5)
```

Создадим модель bag-of-words из токеновских документов.

```
bag = bagOfWords(documents)
```

Удалим слова из модели суммарного слова, которые не отображаются более двух раз. Удалим все документы, не содержащие слов из модели с сумкой слов, и удалите соответствующие записи в ярлыках.

```
bag = removeInfrequentWords(bag,2);
[bag,idx] = removeEmptyDocuments(bag);
YTrain(idx) = [];
bag
```

Обучение классификатора с учителем

Обучим контролируемую классификационную модель, используя число слов из модели суммирования слов и ярлыков.

Для многоклассовой линейной классификации будем использовать функцию fitcecoc. Укажем свойство Counts модели суммирования слов как предикторы, а метки типа события - как отклик. Зададим линейный тип обучения, что позволит поддерживать разреженный ввод данных.

```
XTrain = bag.Counts;
mdl = fitcecoc(XTrain,YTrain,'Learners','linear')
```

Тестирование классификатора

Сделаем прогноз метки тестовых данных с использованием обученной модели и рассчитаем точность классификации. Точность классификации – это доля меток, которые модель предсказывает правильно.

Предварительно обработаем тестовые данные с использованием тех же шагов предварительной обработки, что и данные обучения. Кодировать полученные тестовые документы необходимо в виде матрицы числовых слов в соответствии с моделью мешков слов.

```
documentsTest = preprocessWeatherNarratives(textDataTest);
XTest = encode(bag,documentsTest);
```

Сделаем прогноз метки тестовых данных с использованием обученной модели и рассчитать точность классификации.

```
YPred = predict(mdl,XTest);
acc = sum(YPred == YTest)/numel(YTest)
```

Классификация новых данных

Классифицируем тип событий новых метеорологических отчетов. Создайте массив строк, содержащий новые отчеты о погоде.

```
str = [ ...
    "A large tree is downed and blocking traffic outside Apple Hill."
    "Damage to many car windshields in parking lot."
    "Lots of water damage to computer equipment inside the office."];
documentsNew = preprocessWeatherNarratives(str);
XNew = encode(bag,documentsNew);
labelsNew = predict(mdl,XNew)
```

Пример предварительной обработки

Функция preprocessWeatherNarratives выполняет следующие шаги:

1. Удаляет пунктуацию, используя erasePunctuation.
2. Преобразует текстовые данные в нижний регистр.
3. Разбивает текст, используя tokenizedDocument.
4. Удаляет список слов остановки (например, «и», «из» и «the»), используя removeWords и stopWords.
5. Удаляет слова с 2 или менее символами (removeShortWords).
6. Удаляет слова с 15 или более символами (removeLongWords).
7. Нормализует слова, используя ствол Портера (normalizeWords).

```
function documents = preprocessWeatherNarratives(textData)
% Erase punctuation.
cleanTextData = erasePunctuation(textData);
% Convert the text data to lowercase.
cleanTextData = lower(cleanTextData);
% Tokenize the text.
documents = tokenizedDocument(cleanTextData);
% Remove a list of stop words.
documents = removeWords(documents,stopWords);
% Remove words with 2 or fewer characters, and words with 15 or greater
% characters.
```

```
documents = removeShortWords(documents,2);
documents = removeLongWords(documents,15);
% Normalize the words using the Porter stemmer.
documents = normalizeWords(documents);
end
```

3. Содержание работы

1. Предобработка данных
2. Обучение классификатора
3. Тестирование классификатора на старых данных
4. Классификация новых данных

4. Порядок проведения работы

Задание 1. Подготовка данных для анализа.

Создайте функцию, которая разделяет и обрабатывает текстовые данные, поэтому она может использоваться для анализа. Функция preprocessWeatherNarratives выполняет следующие шаги в порядке:

1. Сотрите знаки препинания (erasePunctuation).
2. Преобразуйте текстовых данных в нижний регистр.
3. Маркировки текста с помощью tokenizedDocument.
4. Удалите стоп-слова (такие как «и», «из» и «»)
5. Удалите слова с 2 или менее символов
6. Удалите слова с 15 или более символов.
7. Нормализуйте слова (normalizeWords).

Создайте модель «мешок слова» из размеченный документов.

Задание 2. Проведите обучение классификатора с учителем.

Задание 3. Проведите тестирование работы классификатора.

Задание 4. Классифицируйте новые данные.

5. Контрольные вопросы

– Постановка задачи классификации.
– Слабоструктурированные данные.
– Задача классификации документов.
– Text Analytics Toolbox.

6. Содержание отчета

Отчёт должен содержать: 1) *постановку задачи* (исходные данные), 2) *ход работы* (краткое описание этапов выполнения работы), 3) *результаты работы* (описание, интерпретация и сравнение результатов), 4) *заключение* (выводы, интерпретация полученных результатов), 5) *приложение* (код программы с построчными комментариями).

Лабораторная работа 11. Неконтролируемая кластеризация документов

1. Цель работы

Изучение процедуры кластеризации слабоструктурированных данных, формирование навыков решения задачи кластеризации.

2. Указания к выполнению работы

Кластеризация документов — одна из задач информационного поиска. Целью кластеризации документов является автоматическое выявление групп семантически похожих документов среди заданного фиксированного множества документов. Следует отметить, что группы формируются только на основе попарной схожести описаний документов, и никакие характеристики этих групп не задаются заранее, в отличие от классификации документов, где категории задаются заранее.

Пусть имеем 3 документа уже прошедших предобработку: OliverTwist.mat, DonQuixote.mat, Pride&Prejudice.mat.

Каждый из элементов в структурах данных содержит следующие поля: *book* – строка, содержащая имя соответствующей книги; *chap* – целое число, идентифицирующее номер соответствующей главы; *text* – строка, содержащая исходный текст документа (параграфа), представленный таким элементом; *token* – массив ячеек строк, содержащих отдельные токены в документе; *vocab* – массив ячеек строк, содержащих уникальные знаки в документе, то есть словарный запас документа; и *count* – целочисленный массив, содержащий числовые значения частоты для соответствующих терминов словаря.

Теперь загрузим три набора данных и вычислим некоторые основные статистические данные:

```
>> clear;
>> load OliverTwist; data1 = data;
>> load DonQuixote; data2 = data;
>> load Pride&Prejudice; data3 = data;
>> clear data
>> whos
Name Size Bytes Class Attributes
data1 1x840 19714994 struct
data2 1x843 24628740 struct
data3 1x666 17382872 struct
```

где можно увидеть, что набор содержит 840 документов (абзацев) от OliverTwist, 843 документа (абзацев) от DonQuixote и 666 документов (абзацев) от Pride&Prejudice.

Затем вычислим гистограммы размеров документов (по количеству слов) для каждого из трех подмножеств, составляющих набор данных.

```
>> hf = figure(1); % creates a new figure
>> set(hf,'Color',[1 1 1],'Name','Basic description of the Dataset');
```

```
>> % prints subcollection's names and number of documents
>> subplot(2,2,1); axis off
>> for k=1:3
collname = eval(sprintf('data%d(1).book',k));
ndocs = eval(sprintf('length(data%d)',k));
comment = sprintf('%s\nDataset %d (%d documents)\n',collname,k,ndocs);
text(0,1.1-k/3,comment);
        end

>> % plots histogram of document sizes for dataset1 (Oliver Twist)
>> for k=1:length(data1), vals1(k)=sum(data1(k).count); end;
>> [hist1,n1] = hist(vals1,min(vals1):max(vals1));
>> subplot(2,2,2); bar(n1,hist1); axis([61,300,0,25]);
>> text(220,23,'Dataset 1');
        >> xlabel('Document Size (in words)'); ylabel('Frequency');

>> % plots histogram of document sizes for dataset2 (Don Quixote)
>> for k=1:length(data2), vals2(k)=sum(data2(k).count); end;
>> [hist2,n2] = hist(vals2,min(vals2):max(vals2));
>> subplot(2,2,3); bar(n2,hist2); axis([61,300,0,25]);
>> text(220,23,'Dataset 2');
        >> xlabel('Document Size (in words)'); ylabel('Frequency');

>> % plots histogram of document sizes for dataset3 (Pride & Prejudice)
>> for k=1:length(data3), vals3(k)=sum(data3(k).count); end;
>> [hist3,n3] = hist(vals3,min(vals3):max(vals3));
>> subplot(2,2,4); bar(n3,hist3); axis([61,300,0,25]);
>> text(220,23,'Dataset 3');
>> xlabel('Document Size (in words)'); ylabel('Frequency');
```

Как видно из рисунка 11.1, все три подмножества имеют аналогичное распределение длин документов. Однако для случая набора данных 1 (OliverTwist) наблюдается некоторое преобладание более коротких документов, а также относительно большое количество более длинных документов в диапазоне от 200 до 300 слов для случая набора данных 2 (DonQuixote). Также обратите внимание, что набор данных 3 (Pride and Prejudice) является наименьшим из трех подмножеств с примерно на 20% меньше документов, чем два других подмножества.

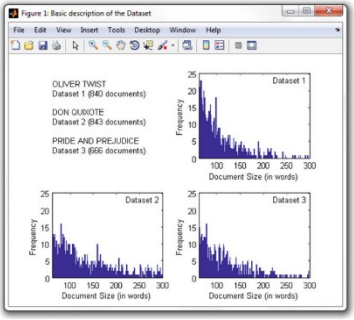


Рис. 11.1 – Гистограмма длины документов

Затем выберем словарь общего сбора данных. Мы делаем это, извлекая словарный запас каждой подкатегории и вычисляя объединение трех словарей:

```
>> % gets the vocabulary of dataset 1 (Oliver Twist)
>> vocabulary1 = data1(1).vocab;
>> for k=2:length(data1)
vocabulary1 = union(vocabulary1,data1(k).vocab);
        end

>> % gets the vocabulary of dataset 2 (Don Quixote)
>> vocabulary2 = data2(1).vocab;
>> for k=2:length(data2)
vocabulary2 = union(vocabulary2,data2(k).vocab);
        end

>> % gets the vocabulary of dataset 3 (Pride & Prejudice)
>> vocabulary3 = data3(1).vocab;
>> for k=2:length(data3)
vocabulary3 = union(vocabulary3,data3(k).vocab);
        end

>> % gets the vocabulary for the overall data collection
>> vocabulary = unique([vocabulary1,vocabulary2,vocabulary3]);
```

После того, как будет добавлен словарь всей коллекции, мы продолжим извлечение тестовых, рабочих и обучающих наборов для экспериментальных целей.

Теперь вычислим некоторые случайные индексы и переменные инициализации для разделения набора данных обучающее, рабочее и тестовое множества, а также рандомизацию каждого из таких подмножеств. Во-первых, мы генерируем случайные индексы для извлечения подмножеств тестов и разработки.

```
>> % generates random indexes for each sub collection
>> ridx1 = randperm(length(data1));
>> ridx2 = randperm(length(data2));
>> ridx3 = randperm(length(data3));
```

Во-вторых, мы генерируем случайные индексы для рандомизации полученных наборов данных обучающей, рабочей и тестовой выборок. Мы будем рассматривать 100 документов из каждого из трех подкатегорий для построения тестового набора, поэтому общий размер полученного тестового набора будет 300 документов. Аналогичным образом, мы будем рассматривать 100 документов из каждой подкатегории для построения набора разработки, поэтому общий размер результирующего набора будет 300 документов. Остальные документы будут использоваться для обучающих.

```
>> % generates random indexes for test, development and train sets
>> randomizetst = randperm(300);
>> randomizedev = randperm(300);
>> randomizetrn = randperm(length(data1)+length(data2)+length(data3)-600);
```

Наконец, мы генерируем набор случайных векторов, которые будут использоваться для целей инициализации в некоторых алгоритмах в следующих разделах:

```
>> % generates a set or random vectors for initialization purposes
>> initmtx = rand(60,length(vocabulary));
>> for k=1:60
initmtx(k,:) = initmtx(k,:)/norm(initmtx(k,:));
        end
```

Теперь мы приступаем к созданию наборов данных тестирования, разработки и обучения с использованием генерируемых случайных индексов:

```
>> % extracts the test, development and train datasets
>> tst = [data1(ridx1(1:100)),data2(ridx2(1:100)),data3(ridx3(1:100))];
>> dev = [data1(ridx1(101:200)),data2(ridx2(101:200)),data3(ridx3(101:200))];
>> trn = [data1(ridx1(201:end)),data2(ridx2(201:end)),data3(ridx3(201:end))];

>> % randomizes the test, development and train datasets
>> tstdata = tst(randomizetst);
>> devdata = dev(randomizedev);
>> trndata = trn(randomizetrn);

>> % eliminates unnecessary variables
>> clear data1 data2 data3 tst dev trn
```

Наконец, мы готовы провести эксперименты по классификации документов с нашим построенным набором данных.

Неконтролируемая кластеризация

Возможность организовать коллекцию объектов в группы или категории является основой для процессов абстрагирования и обобщения человеческого мышления. Такой процесс категоризации основан на наблюдаемых сходствах между наиболее важными атрибутами рассматриваемых объектов и позволяет описать смежную структуру данных. В случае текстовой информации, например, в коллекции документов, основными объектами (документами) являются слова, которые они содержат, и семантические отношения, которые они определяют по объектам в коллекции. В этом смысле проблема организации сбора документов по группам или категориям является проблемой семантической природы.

Неконтролируемая кластеризация (кластеризация) - это процесс автоматической группировки объектов в данной коллекции в соответствии с их сходными особенностями. Хотя ожидается, что элементы внутри каждой группы или кластера будут иметь большое сходство между ними, элементы разных групп или кластеров, как ожидается, будут иметь большие различия между ними. Процесс называется неконтролируемым, поскольку информация о категориях, сосуществующих в коллекции, ранее не была известна или не использовалась во время процесса. Неконтролируемая кластеризация - очень мощная процедура для обнаружения скрытой структуры данных (если таковая имеется).

Алгоритм кластеризации k-средних - это итеративный алгоритм, в котором коллекция n наблюдений разбивается на k кластеров, которые обновляются итеративно, пока они не сходятся в устойчивый раздел. Итерация каждого алгоритма выполняется в два этапа: назначение и обновление. На шаге назначения каждому элементу в коллекции присваивается ближайший кластер с помощью метрики расстояния. Расстояние между кластером и данным элементом вычисляется в представлении векторного пространства, измеряя расстояние между данным представлением элемента и центроидом кластера. На этапе обновления все центроиды кластера обновляются, принимая во внимание новый раздел, сгенерированный на этапе назначения. В кластеризации k-средних центроидные векторы представлены средним вектором всех элементов, принадлежащих соответствующим кластерам. Алгоритм начинается с предопределенного набора центроидов (которые могут быть сгенерированы либо случайным образом, либо с помощью любого другого критерия) и выполняет последовательные итерации шагов назначения и обновления до тех пор, пока не будет больше изменений в разделе в течение последовательных итераций.

Прежде чем применять кластеризацию k-значений для нашего сбора данных, вычислим его векторное пространство, выполнив некоторые процедуры: 1) объединим три набора данных (тестовый, рабочий и обучающий) в единый структурный массив:

```
>> data = [tstdata,devdata,trndata];
```

2) затем мы вычисляем матрицу TF для общего сбора и одновременно создаем вектор ссылочной категории (т.е. числовой массив, содержащий индекс, который указывает, какая книга каждого документа была извлечена из: " 1 " для Oliver Twist, " 2 " для Don Quixote и «3» для Pride and Prejudice. Этот вектор категории будет использоваться только в этом разделе для целей оценки, тогда как в последующих разделах он будет использоваться как для контролируемого обучения, так и для оценки.

```
>> % initializes the tf matrix and category vector
>> ndocs = length(data); nvocs = length(vocabulary);
        >> tfmtx = sparse(nvocs,ndocs); category = zeros(ndocs,1);

>> for k = 1:ndocs % updates the tf matrix and category vector
[void,index] = intersect(vocabulary,data(k).vocab);
tfmtx(index,k) = data(k).count;
switch data(k).book
case 'OLIVER TWIST', category(k) = 1;
case 'DON QUIXOTE', category(k) = 2;
case 'PRIDE AND PREJUDICE', category(k) = 3;
end
        end
```

3) затем мы применяем взвешивание IDF и нормализацию L2:

```
>> % computes the idf vector
>> idfvt = log(ndocs./full(sum(tfmtx>0,2)));
>> tfidf = sparse(nvocs,ndocs);

>> % applies the idf weighting and L2-norm
```

```
>> for k=1:ndocs
tfidf(:,k) = tfmtx(:,k).*idfvt;
tfidf(:,k) = tfidf(:,k)/norm(tfidf(:,k));
        end;
```

4) и снова разделим три набора данных вместе со своей соответствующей категориальной категорией:

```
>> tstmtx = tfidf(:,1:300); tstcat = category(1:300);
>> devmtx = tfidf(:,301:600); devcat = category(301:600);
>> trnmtx = tfidf(:,601:end); trncat = category(601:end);
```

Теперь мы готовы применить алгоритм кластеризации k-средних к рассматриваемой коллекции документов.

```
>> % initializes the parameters for k-means clustering
>> nclusts = 3; % defines the number of clusters to be considered
>> dst = 'cosine'; % defines the distance metric to be used
>> initc = initmtx(1:nclusts,:); % defines the initial set of centroids


>> % applies the k-means clustering algorithm to the test set
>> [idxs,centroids] = kmeans(tstmtx',nclusts,'Distance',dst,'Start',initc);
```

Чтобы оценить эффективность итогового распределения кластеров, мы должны сравнить распределение кластеров, созданное алгоритмом k-средних, с исходными категориями, к которым принадлежат документы. Поскольку три книги (из которых была собрана коллекция документов) различны в тематическом, авторском и литературном стилях, разумно ожидать, что результирующий раздел будет согласован с тремя книгами, используемыми для создания коллекции. Однако перед сопоставлением сгенерированных кластеров с категориями документов нам необходимо идентифицировать соответствия между ними.

Обратите внимание, что алгоритм кластеризации k-mean присваивает каждому документу один из трех кластеров, а именно 1, 2 и 3,

```
>> unique(idxs)'
ans =
        1 2 3
```

и аналогичным образом соответствующие категории документов были определены как 1 (Oliver Twist), 2 (Don Quixote) и 3 (Pride and Prejudice):

```
>> unique(tstcat)'
ans =
        1 2 3
```

Как видно из примера, рассмотрение трех кластеров дает лучший раздел, чем выбор двух или четырех. Это явно согласуется с тем, что наш экспериментальный сбор данных изначально был извлечен из трех разных источников информации.

3. Содержание работы

1. Предобработка данных
2. Обучение кластеризатора

3. Тестирование кластеризатора на старых данных
4. Кластеризация новых данных

4. Порядок проведения работы

Задание 1. Подготовка данных для анализа

Создайте функцию, которая разделяет и обрабатывает текстовые данные, поэтому она может использоваться для анализа. Функция preprocessWeatherNarratives выполняет следующие шаги в порядке:

1. Сотрите знаки препинания (erasePunctuation).
2. Преобразуйте текстовых данных в нижний регистр.
3. Маркировки текста с помощью tokenizedDocument.
4. Удалите стоп-слова (такие как «и», «из» и «»)
5. Удалите слова с 2 или менее символов
6. Удалите слова с 15 или более символов.
7. Нормализуйте слова (normalizeWords).

Создайте модель «мешок слова» из размеченный документов.

Задание 2. Проведите обучение кластеризатора без учителя.

Задание 3. Проведите тестирование работы кластеризатора.

Задание 4. Проведите кластеризацию новых данных.

5. Контрольные вопросы

- Постановка задачи кластеризации.
- Слабоструктурированные данные.
- Задача кластеризации документов.
- Text Analytics Toolbox.
- Кластеризация в Matlab

6. Содержание отчета

Отчёт должен содержать: 1) *постановку задачи* (исходные данные), 2) *ход работы* (краткое описание этапов выполнения работы), 3) *результаты работы* (описание, интерпретация и сравнение результатов), 4) *заключение* (выводы, интерпретация полученных результатов), 5) *приложение* (код программы с построчными комментариями).

Лабораторная работа 12. Информационный поиск

1. Цель работы

Изучение методов информационного поиска, формирование навыков разработки алгоритмов для информационного поиска.

2. Указания к выполнению работы

Информационный поиск (information retrieval) – процесс поиска неструктурированной документальной информации, удовлетворяющей информационные потребности.

Полнотекстовый поиск (Full text searching) – автоматизированный поиск документов, при котором поиск ведётся не по именам документов, а по их содержимому, всему или существенной части.

Для данной работы будем использовать набор данных из предыдущей работы.

Перейдем к иллюстрации очень простого примера бинарного поиска. Для этого рассмотрим главу XVII второй части Don Quixote, которая соответствует категории 2069 в нашем наборе данных: «Wherein is shown the furthest and highest point which the unexampled courage of Don Quixote reached or could reach; together with the happily achieved adventure of the lions».

Предположим, что мы не можем вспомнить ни длинный заголовок, ни номер главы, но мы заинтересованы в поиске всех документов (т. е. абзацев в нашем сборнике данных), которые принадлежат этой главе. Мы должны уметь определять некоторые ключевые слова, относящиеся к главе, а затем строить запрос для проведения поиска. Рассмотрим, например, следующие два ключевых слова: courage и lions, и ищем все документы в коллекции данных, которые содержат оба ключевых слова. С этой базовой стратегией мы получим некоторые из абзацев, относящихся к вышеупомянутой главе.

Мы реализуем эту стратегию поиска следующим образом:

```
>> % defines the keywords to be used for the search
>> kw1 = 'courage'; idx1 = strcmp(vocabulary,kw1);
>> kw2 = 'lions'; idx2 = strcmp(vocabulary,kw2);
>> % conducts the search
>> bsearch_and = find((tfmtx(idx1,:)>0)&(tfmtx(idx2,:)>0));

>> % displays the categories of the retrieved paragraphs
>> chap_id(bsearch_and)
ans =
        2069 2069
```

Стратегия основного поиска, реализованная выше, называется двоичным поиском, поскольку нас не беспокоит частота появления ключевых слов, а только то, появляются они или нет в документах в коллекции. Также обратите внимание, что вхождения ключевых слов объединяются с

помощью логического оператора AND и нацеленного на восстановление тех документов, которые содержат оба ключевых слова. Следовательно, мы назвали переменную, содержащую результат поиска, как bsearch_and.

Обратите внимание, что из нашей стратегии поиска было отобрано в общей сложности 2 документа, оба из которых относятся к рассматриваемой главе (категория 2 069). Это можно автоматически проверить следующим образом:

```
>> sum(chap_id(bsearch_and)==2069)
ans =
     2
```

Естественный вопрос заключается в следующем: насколько хорош этот результат? На первый взгляд, да, поскольку мы получили абзацы из главы, которую искали. Однако к ответу на этот вопрос можно подходить из двух разных, но связанных, перспектив. В первом случае нас будет интересовать процент правильных ответов, которые были достигнуты, т.е. доля правильных из всех полученных документов. Эта мера упоминается в литературе как *precision* (точность), и на самом деле это то, о чем заботятся большинство обычных коммерческих поисковых систем.

Мы уже можем сделать вывод, что точность, полученная в результате нашего поиска, составляет 100%; но в более общем случае это значение можно вычислить следующим образом:

```
>> sum(chap_id(bsearch_and)==2069)/length(bsearch_and)*100
ans =
     100
```

Рассмотрим диаграмму на рисунке 12.1, где как фактическая категория, так и извлеченные подмножества документов представлены для гипотетического поиска документа.

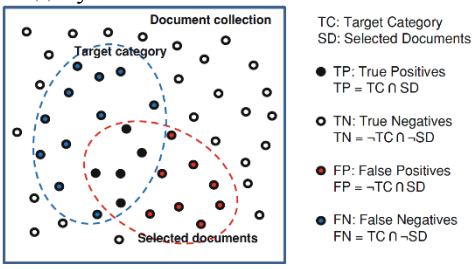


Рис. 12.1 – Целевая категория и выбранные подмножества документов

Как видно из рисунка 12.1, набор целевой категории (класс документов, который мы заинтересованы в поиске), и выбранный набор документов (группа документов, найденная стратегией поиска) разделяют общую коллекцию документов на четыре подмножества, Эти четыре подмножества обычно называются:

• True Positives (TP): это документы, которые были правильно идентифицированы как принадлежащие к целевой категории.

• True Negatives (TN): это документы, которые были правильно идентифицированы как не принадлежащие к целевой категории.

• False Positives (FP): это документы, которые были ошибочно идентифицированы как принадлежащие к целевой категории.

• False Negatives (FN): это документы, которые были ошибочно идентифицированы как не принадлежащие к целевой категории.

Теперь мы можем формально изложить определения точности и напомнить в терминах мощностей подмножеств TP, TN, FP и FN следующим образом:

$$precision = TP / (TP+FP) \times 100\ \%$$

$$recall = TP / (TP+FN) \times 100\ \%,$$

где TP, TN, FP и FN следует понимать как мощности их соответствующих подмножеств.

Другим полезным и часто используемым счетом, который сочетает в себе как *точность* (*precision)*, так и *полноту* (*recall)* в один балл, является F-оценкой, которая определяется с точки зрения гармонического среднего значения *precision* и *recall*:

$$F\text{-}score = 2 \times precision \times recall / (precision + recall)$$

Давайте продолжим экспериментальную работу, вычислив *матрицу ошибок* (*confusion matrix*), точность, полноту и F-оценку для нашего эксперимента по бинарному поиску.

```
>> % generates an indicator vector for the selected documents
>> ndocs = size(tfmtx,2);
>> selecteddocs = zeros(1,ndocs);
>> selecteddocs(bsearch_and) = 1;

>> % generates an indicator vector for the target category
>> chap_idx = 2069;
>> targetcateg = chap_id==chap_idx;

>> % computes the cardinality of the TP, TN, FP, FN subsets
>> truepos = sum((selecteddocs==1)&(targetcateg==1));
>> trueneg = sum((selecteddocs==0)&(targetcateg==0));
>> falsepos = sum((selecteddocs==1)&(targetcateg==0));
>> falseneg = sum((selecteddocs==0)&(targetcateg==1));


>> % constructs and displays the confusion matrix
>> str1 = sprintf(' Category Not-Category');
>> str2 = sprintf(' Selected: %7.2f %7.2f',truepos,falsepos);
>> str3 = sprintf('Not-Selected: %7.2f %7.2f',falseneg,trueneg);
>> disp(sprintf('\n%s\n%s\n%s\n',str1,str2,str3));

Category Not-Category
Selected: 2.00 0.00
Not-Selected: 11.00 2336.00

>> % computes and displays the precision, recall and F-score
>> precision = truepos/(truepos+falsepos)*100;
```

```
>> recall = truepos/(truepos+falseneg)*100;
>> fscore = 2*precision*recall/(precision+recall);
>> str1 = sprintf(' Precision = %6.2f',precision);
>> str2 = sprintf(' Recall = %6.2f',recall);
>> str3 = sprintf(' F-score = %6.2f',fscore);
>> disp(sprintf('\n%s%s%s\n',str1,str2,str3));

    Precision = 100.00 Recall = 15.38 F-score = 26.67
```

Результаты расчёта оценок результата поиска представлены выше.

3. Содержание работы

1. Предобработка данных
2. Информационный поиск
3. Оценка результатов поиска.

4. Порядок проведения работы

Задание 1. Выполните предобработку данных

Задание 2. Проведите информационный поиск слов (по вариантам).

Задание 3. Дайте оценку результатов поиска: рассчитайте TP, TN, FP и FN, *precision*, *recall*, *confusion matrix*, *F-score*.

5. Варианты заданий

Варианты заданий формируются путём выбора преподавателем слов для поиска из исходных текстов с применением генератора случайных чисел.

6. Контрольные вопросы

- Информационный поиск. Постановка задачи.
- Информационный поиск средствами Matlab.
- Ошибки первого рода.
- Ошибки второго рода.
- Меры оценка результатов поиска.

7. Содержание отчета

Отчёт должен содержать: 1) *постановку задачи* (исходные данные), 2) *ход работы* (краткое описание этапов выполнения работы), 3) *результаты работы* (описание, интерпретация и сравнение результатов), 4) *заключение* (выводы, интерпретация полученных результатов), 5) *приложение* (код программы с построчными комментариями).

Лабораторная работа 13. Анализ покупательской корзины (поиск ассоциативных правил в данных)

1. Цель работы

Знакомство со средой Deductor Academic, изучение алгоритмов поиска ассоциативных правил, формирование навыков использования ассоциативных правил для анализа рыночной корзины.

2. Указания к выполнению работы

Аффинитивный анализ (affinity analysis) — один из распространенных методов Data Mining. Его название происходит от английского слова affinity, которое в переводе означает «близость», «сходство». Цель данного метода — исследование взаимной связи между событиями, которые происходят совместно. Разновидностью аффинитивного анализа является анализ рыночной корзины (market basket analysis), цель которого — обнаружить ассоциации между различными событиями, то есть найти правила для количественного описания взаимной связи между двумя или более событиями. Такие правила называются ассоциативными правилами (association rules).

Примерами приложения ассоциативных правил могут быть следующие задачи:

- выявление наборов товаров, которые в супермаркетах часто покупаются вместе или никогда не покупаются вместе;
- определение доли клиентов, положительно относящихся к нововведениям в их обслуживании;
- определение профиля посетителей веб-ресурса;
- определение доли случаев, в которых новое лекарство показывает опасный побочный эффект.

Базовым понятием в теории ассоциативных правил является транзакция — некоторое множество событий, происходящих совместно. Типичная транзакция —приобретение клиентом товара в супермаркете. В подавляющем большинстве случаев клиент покупает не один товар, а набор товаров, который называется рыночной корзиной. При этом возникает вопрос: является ли покупка одного товара в корзине следствием или причиной покупки другого товара, то есть связаны ли данные события? Эту связь и устанавливают ассоциативные правила. Например, может быть обнаружено ассоциативное правило, утверждающее, что клиент, купивший молоко, с вероятностью 75 % купит и хлеб.

Следующее важное понятие — предметный набор. Это непустое множество предметов, появившихся в одной транзакции.

Анализ рыночной корзины — это анализ наборов данных для определения комбинаций товаров, связанных между собой. Иными словами, производится поиск товаров, присутствие которых в транзакции влияет на вероятность наличия других товаров или комбинаций товаров.

Ассоциативное правило состоит из двух наборов предметов, называемых условие (antecedent) и следствие (consequent), записываемых в виде X → Y, что читается следующим образом: «Из X следует Y». Таким образом, ассоциативное правило формулируется в виде: «Если условие, то следствие».

Ассоциативные правила описывают связь между наборами предметов, соответствующими условию и следствию. Эта связь характеризуется двумя показателями — поддержкой (support) и достоверностью (confidence).

Поддержка ассоциативного правила — это число транзакций, которые содержат как условие, так и следствие.

Достоверность ассоциативного правила A → B представляет собой меру точности правила и определяется как отношение количества транзакций, содержащих и условие, и следствие, к количеству транзакций, содержащих только условие.

Если поддержка и достоверность достаточно высоки, можно с большой вероятностью утверждать, что любая будущая транзакция, которая включает условие, будет также содержать и следствие.

Для выполнения работы рекомендуется использовать среду Deductor Academic.

3. Содержание работы

1. Импорт данных в Deductor Academic.
2. Поиск ассоциативных правил.
3. Интерпретация ассоциативных правил и формирование рекомендаций.

4. Порядок проведения работы

Задание 1. Загрузите исходные данные. Запустите поиск ассоциативных правил.

Задание 2. Выполните по вариантам интерпретацию ассоциативных правил. Определите тип правила. Дайте рекомендации по использованию правила.

Задание 3. Постройте дерево правил. Проведите анализ дерева согласно варианту.

Задание 4. Запустите алгоритм apriori на другом диапазоне допустимой достоверности. Проведите интерпретацию новых полученных правил по вариантам.

Задание 5. Проведите анализ ассоциативных правил (по варианту) по схеме «что-если»: ответьте на вопрос, что приобретёт клиент купивший: а) 2 товара из транзакции 1, б) 3 товара из транзакции).

5. Варианты заданий

Для заданий 2-4: номер правила соответствует номеру варианта. Варианты к заданию 5 представлены в таблице 13.1 (числа в таблице соответствуют номеру товара в списке в алфавитном порядке).

Таблица 13.1 – Варианты к заданию 5

Вариант	Транзакция 1		Транзакция 2		
	Товар 1	Товар 2	Товар 1	Товар 2	Товар 3
1	27	26	10	12	6
2	30	25	23	7	27
3	5	13	22	9	11
4	31	22	6	21	18
5	21	6	4	16	6
6	4	24	17	12	20
7	10	2	32	28	9
8	19	10	12	20	22
9	32	2	20	19	23
10	32	4	8	31	25

6. Контрольные вопросы

- Что такое ассоциация?
- В чем заключается основная задача анализа рыночной корзины?
- Предметный набор. Транзакция.
- Поддержка ассоциативного правила
- Достоверность ассоциативного правила. Лифт. Левередж
- Произвести интерпретацию ассоциативного правила.

7. Содержание отчета

Отчёт должен содержать: 1) *постановку задачи* (исходные данные), 2) *ход работы* (краткое описание этапов выполнения работы), 3) *результаты работы* (описание, интерпретация и сравнение результатов), 4) *заключение* (выводы, интерпретация полученных результатов), 5) *приложение* (код программы с построчными комментариями).

Лабораторная работа 14. Анализ эколого-экономических рисков в регионе

1. Цель работы

Формирование навыков использования методов интеллектуального анализа данных для решения практических задач.

2. Указания к выполнению работы

Один из подходов к определению степени негативного воздействия загрязненной среды на человека заключается в оценке показателей риска возникновения заболеваний различных систем организма человека – индексов опасности HI, которые принимают только неотрицательные значения, а в случае превышения 1 сигнализируют о наличии риска. Считается, что чем больше значения индекса опасности, тем выше риск: индекс опасности от 1 до 5 означает низкий уровень риска, от 5 до 10 – средний уровень риска, выше 10 – высокий уровень риска.

Управляющие органам, органы здравоохранения, контроля и планирования желали бы структурировать информацию о наиболее частых сочетаниях видов рисков, а также получить правила, позволяющие по части профиля риска с достаточной точностью оценивать его недостающую часть. Перечисленные задачи можно решить, используя инструментарий ассоциативных правил.

В исходной таблице для лабораторной работы приведены коэффициенты опасности HQ по каждому веществу, а также индексы опасности HI для следующих систем организма: для желудочно-кишечного тракта (ЖКТ), центральной нервной системы (ЦНС), почек (почки), сердечно-сосудистой системы (ССС), крови (кровь), печени (печень), иммунной системы (иммун), репродуктивной системы (репрод), гормональной системы (гормон), кожи (кожа), слизистой (слиз), развития (развитие) и органов дыхания (дыхание). Данные записаны в Excel следующим образом: во всех случаях, когда имелся риск (HI>1), ставилась 1, и 0 в противном случае.

Сформированные таким образом транзакции (118) сохраняются в файл с расширением txt. Следует обратить внимание на кодировку текстового файла – она должна быть ANSI Windows. После запуска Deductor Studio необходимо осуществить ввод данных о транзакциях-рисках в программу. Для этого вызовем «Мастер импорта», кликнув правой клавишей на значке «Сценарии» в левом окне программы или при выделенном значке нажав клавишу F6.

Поиск ассоциативных правил выполняется с помощью «Мастера обработки» (в нем пункт необходимо выбрать пункт «Data Mining» – «Ассоциативные правила»).

Для поиска часто встречающихся множеств установим минимальный порог поддержки в 10%, а максимальный порог 90%, для формирования ассоциативных правил установим минимальную достоверность на уровне 40%, а максимальную – на уровне 90%.

3. Содержание работы

1. Импорт данных в Deductor Academic.
2. Поиск ассоциативных правил.
3. Интерпретация найденных правил, формирование рекомендаций.

4. Порядок проведения работы

Задание 1. Осуществить поиск наиболее часто встречающихся наборов рисков.

Задание 2. Провести поиск ассоциативных правил.

Задание 3. Дать интерпретацию полученным результатам.

Задание 4. Указать направления дальнейшего использования полученных правил.

5. Варианты заданий

Для заданий необходимо выбирать те правила, номер которых в списке соответствует номеру варианта.

6. Контрольные вопросы

- Что такое ассоциация?
- Поддержка ассоциативного правила
- Достоверность ассоциативного правила. Лифт. Левередж
- Задачи, решаемые методами интеллектуального анализа данных
- Методы интеллектуального анализа данных

7. Содержание отчета

Отчёт должен содержать: 1) *постановку задачи* (исходные данные), 2) *ход работы* (краткое описание этапов выполнения работы), 3) *результаты работы* (описание, интерпретация и сравнение результатов), 4) *заключение* (выводы, интерпретация полученных результатов), 5) *приложение* (код программы с построчными комментариями).

Лабораторная работа 15. Применение генетического алгоритма для решения задачи оптимизации

1. Цель работы

Изучение основ работы с генетическими алгоритмами в Matlab, исследование экстремумов функций с помощью генетических алгоритмов.

2. Указания к выполнению работы

Генетические алгоритмы – это метод решения оптимизационных задач, основанный на биологических принципах естественного отбора и эволюции. Генетический алгоритм повторяет определенное количество раз процедуру модификации популяции (набора отдельных решений), добиваясь тем самым получения новых наборов решений (новых популяций). При этом на каждом шаге из популяции выбираются «родительские особи», то есть решения, совместная модификация которых (скрещивание) и приводит к формированию новой особи в следующем поколении. Генетический алгоритм использует три вида правил, на основе которых формируется новое поколение: правила отбора, скрещивания и мутации. Мутация позволяет путем внесения изменений в новое поколение избежать попадания в локальные минимумы оптимизируемой функции.

Генетические алгоритмы оптимизации являются алгоритмами случайно-направленного поиска и применяются в основном там, где сложно или невозможно сформулировать задачу в виде, пригодном для более быстрых алгоритмов локальной оптимизации, либо если стоит задача оптимизации недифференцируемой функции или задача многоэкстремальной глобальной оптимизации.

Типичными применениями ГА являются следующие:

- оптимизация функций;
- оптимизация запросов в базах данных;
- задачи на графах (задача коммивояжера, раскраска и пр.)
- обучение искусственной нейронной сети;
- задачи компоновки;
- составление расписаний;
- игровые стратегии;
- теория приближений и др.

Основными параметрами ГА являются:

- вероятность мутации;
- точность получения результата;
- количество итераций алгоритма или количество поколений;
- размер популяции.

1) Прежде всего, в данном алгоритме для организации начала счета создается произвольное исходное семейство.

2) Далее алгоритм производит некую последовательность новых семейств или поколений. На каждом отдельном шаге алгоритм использует определенные индивидуумы из текущего поколения, для того, что бы создать последующее поколение. При формировании нового поколения в алгоритме проводятся следующие действия:

• Отмечается каждый член текущего семейства посредством вычисления соответствующего значения пригодности;

• Проводится масштабирование полученного ряда значений функции пригодности, что позволяет построить диапазон значений более удобный для последующего использования;

• Выбираются родительские значения на основе значений их пригодности;

• Часть индивидуумов из родительского поколения имеет более меньшие значения функции пригодности и которые в далее выбираются как элитные значения. Эти элитные значения передаются далее уже в последующее поколение;

• Дочерние значения образуются или путем неких случайных изменений отдельного одного родителя - мутация - или путем комбинации векторных компонентов некой пары родителей – кроссовер;

• Замена текущего семейства на дочернее с целью формирования последующего поколения.

3) Останов алгоритма производится тогда, когда выполняется какой-нибудь критерий останова.

3. Содержание работы

1. Поиск минимума функции одной переменной
2. Поиск минимума функции двух переменных

4. Порядок проведения работы

Задание 1. Найти минимум функции одной переменной любым известным способом. Исследовать функцию с помощью генетических алгоритмов. Сравнить полученные результаты. Определить глобальный минимум и значение функции в этой точке.

Провести эксперимент при различном размере начальной популяции: 10; 50; 300; 800. Для каждого из этих значений принять следующие операторы отбора родительских особей:

• Stochastic uniform;

• Uniform;

• Roulette.

Сделать вывод о том, как влияет размер исходной популяции на результаты, а также какое влияние вносят различные операторы отбора родительских особей. Привести графическое решение, найденное с помощью генетических алгоритмов, которое соответствует максимально точному найденному решению. Объяснить каким образом влияют операторы отбора родительских особей на результаты.

Задание 2. Найти минимум функции двух переменных любым известным способом. Провести исследование этой функции с помощью генетических алгоритмов и заполнить табл. 3, изменяя вероятность скрещивания и размер начальной популяции. Выбрать оператор мутации по Гауссу, вероятность мутации принять равным по умолчанию 1,0.

По результатам проведенных экспериментов сделать вывод о том, при каком соотношении вероятности скрещивания к мутации результаты максимально точны. Привести графическое решение, найденное с помощью генетических алгоритмов, которое соответствует максимально точному найденному решению.

Определить относительную погрешность полученных значений функций при каждом значении размера начальной популяции (для наилучшего соотношения вероятности скрещивания к мутации) и построить график зависимости погрешностей от размера начальной популяции. Сделать вывод о влиянии размера начальной популяции на результаты

5. *Варианты заданий*

Варианты к заданию 1 представлены в табл. 15.1.

Таблица 15.1 – Функции одной переменной

Вариант	Целевая функция	Вариант	Целевая функция
1	$f(x)=\sin(x)+\sin(x^2)$	3	$f(x)=-2x-8+3\cdot\sqrt[3]{6\cdot(x+4)^2}$
2	$f(x)=\frac{6\cdot\sqrt[3]{6\cdot(x-3)^2}}{(x-1)^2+8}$	4	$f(x)=\sqrt{3+x^2+3}\cdot\cos(x)$

Варианты к заданию 2 представлены в табл. 15.2.

Таблица 15.2 – Функции двух переменных

Вариант	Целевая функция
1	$y_1=\frac{x_1-x_2}{x_1+x_2}$
2	$y_1=1{,}5\cdot x_1+x_2^{\,3}$
3	$y_1=\sqrt{x_1^2+x_2^2}$
4	$y_1=2{,}3\cdot x_1\cdot x_2-0{,}5\cdot x_1^2+1{,}8\cdot x_2^2.$
5	$y=6\cdot x_1+5\cdot x_1\cdot x_2+x_2^2,$

6. *Контрольные вопросы*

– Каковы механизм передачи наследственной информации?

– Дайте определение генетического алгоритма (ГА).
– Перечислите основные отличительные особенности ГА.
– Перечислите генетические операторы.
– Какие критерии останова используются для ГА?
– Опишите схему классического ГА.
– В чем заключаются особенности совместного использования генетических операторов?
– Сформулируйте фундаментальную теорему ГА.

7. Содержание отчета

Отчёт должен содержать: 1) *постановку задачи* (исходные данные), 2) *ход работы* (краткое описание этапов выполнения работы), 3) *результаты работы* (описание, интерпретация и сравнение результатов), 4) *заключение* (выводы, интерпретация полученных результатов), 5) *приложение* (код программы с построчными комментариями).

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Banchs R. E. Text Mining with MATLAB. – Springer Science+Business Media New York, 2013. – 356 p.
2. Аведьян Э.Д.,. Галушкин А.И, Червяков Н.И., Сахнюк П.А.. Нейросетевые технологии обработки информации: учебное пособие. — Ставрополь: Изд-во «Агрус», 2013. – 292 с.
3. Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интеллектуальных систем. – СПб.: «Питер», 2001. – 382 с.
4. Гладков Л.А., Курейчик В.В., Курейчик В.М. Генетические алгоритмы. 2-е изд., – М.: ФИЗМАТЛИТ, 2006. – 320 с.
5. Горшков С. Введение в онтологическое моделирование [Электронный ресурс]. – ООО «ТриниДата», 2014-2016. – 165 с. – URL: https://trinidata.ru/files/SemanticIntro.pdf (дата обращения: 13.06.2018).
6. Муромцев Д.И. Онтологический инжиниринг знаний в системе Protégé. – СПб: СПб ГУ ИТМО, 2007. – 62 с.
7. Николаев С.В., Баженов Р.И. Распознавание образов с помощью нейронных сетей в среде MatlabR2009b // Nauka-rastudent.ru. – 2015. – No. 13 (13-2015) / [Электронный ресурс] – Режим доступа. – URL: http://nauka-rastudent.ru/13/2355/ (дата обращения 13.06.2018)
8. Потапов А.С. Технологии искусственного интеллекта – СПб: СПбГУ ИТМО, 2010. – 218 с.
9. Сахнюк П.А. Интеллектуальные системы и технологии. — Ставрополь: Агрус, 2012. — 228 с.
10. Седова Е. Н., Раменская А. В., Безбородникова Р.М. Ассоциативные правила в социально-экономических и экологических исследованиях. – Оренбург: ОГУ, 2015. – 170 с.
11. Хабаров С.П. Интеллектуальные информационные системы PROLOG – язык разработки интеллектуальных и экспертных систем. – СПб.: СПбГЛТУ, 2013. – 140 с.