РОСЖЕЛДОР

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Ростовский государственный университет путей сообщения»
(ФГБОУ ВО РГУПС)

А.В. Суханов, З.В. Лященко

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ И СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Учебно-методическое пособие
для лабораторных работ

Ростов-на-Дону
2017

УДК 681.5(07) + 06

Рецензент – доктор технических наук, профессор М.А. Бутакова

Суханов, А.В.

Интеллектуальные системы и технологии и системы искусственного интеллекта: учебно-методическое пособие для лабораторных работ / А.В. Суханов, З.В. Лященко; ФГБОУ ВО РГУПС. – Ростов н/Д, 2017. – 38 с.

Настоящее пособие знакомит студентов с основными понятиями и структурой машины Тьюринга, интерфейсом и работой программы-тренажера «Машина Тьюринга», программной реализацией алгоритма согласно заданию с помощью программы-тренажера, основными понятиями теории искусственных нейронных сетей, алгоритмом обучения простого перцептрона на языке программирования, а также поможет в самостоятельном обучении перцептрона.

Предназначено для студентов и магистрантов направлений «Информатика и вычислительная техника», «Информационные системы и технологии» и «Механотроника и робототехника», а также для студентов, аспирантов и магистрантов всех специальностей, изучающих дисциплины «Интеллектуальные системы и технологии», «Системы искусственного интеллекта» и смежные дисциплины.

Одобрено к изданию кафедрой «Вычислительная техника и автоматизированные системы управления».

ОГЛАВЛЕНИЕ

Лабораторная работа № 1
Представление алгоритмов на машине Тьюринга

Цель

Получить навыки реализации алгоритмов на машине Тьюринга с помощью программы-тренажера «Машина Тьюринга»

Задачи

1 Ознакомиться с основными понятиями и структурой машины Тьюринга.
2 Изучить интерфейс и работы программы-тренажера «Машина Тьюринга».
3 Запрограммировать алгоритм согласно заданию с помощью программы-тренажера.
4 Оформить индивидуальный отчет.

Содержание индивидуального отчета

1 Краткий конспект основных теоретических положений работы.
2 Функциональная диаграмма одного из алгоритмов, описанных в примере.
3 Функциональная диаграмма программируемого алгоритма.
4 Результаты выполнения работы в виде снимка экрана с программой-тренажером «Машина Тьюринга».
5 Выводы и ответы на контрольные вопросы.

Теоретические сведения

В 1937 году английский математик Алан Тьюринг опубликовал работу, в которой уточнил понятие алгоритма, обратившись к воображаемой вычислительной машине. Машина Тьюринга – один из способов записи алгоритма в функциональной таблицы или функциональной диаграммы. Правилом выполнения алгоритма является описание устройства конкретной машины Тьюринга. Машина Тьюринга, так же как конечный автомат, является дискретным устройством преобразования информации. Приведем ее точное определение, а затем интерпретацию работы.

По своей сути машина Тьюринга является математическим аппаратом для решения задачи построения алгоритма. Название «машина» произошло только потому, что по описанию и составным частям она похожа на вычислительную машину. Принципиальным отличием машины Тьюринга является то, что ее запоминающее устройство представляет собой бесконечную ленту: у реальных вычислительных машин запоминающее устройство может быть сколь угодно большим, но обязательно конечным. Машину Тьюринга нельзя реализовать именно из-за бесконечности её ленты. В этом смысле она мощнее любой вычислительной машины.

Машина Тьюринга состоит из бесконечной в обе стороны счетной ленты (возможны машины Тьюринга с несколькими бесконечными лентами), разделенной на ячейки, и автомата, реализованного в виде головки чтения/записи (каретки). Автомат может находиться в одном из множества конечных состояний $\{Q_0, Q_2, \ldots, Q_n\}$ (это множество также называют алфавитом

состояний). Состояние Q_0 называется пассивным. Считается, что если машина попала в это состояние, то она закончила свою работу. Состояние Q_1 называется начальным. Находясь в этом состоянии, машина начинает свою работу.

Автомат может перемещаться влево и вправо по ленте, читать и записывать символы конечного алфавита $\{a_0, a_2, \ldots, a_k\}$ (или алфавита сигналов) в ячейки. Выделяется особый «пустой» символ a_0, который в начальный момент времени заполняет все ячейки ленты, кроме тех, в которых записаны входные данные (входное слово). Входное слово размещается на ленте по одной букве в расположенных подряд ячейках. Слева и справа от входного слова находятся только пустые ячейки.

Автомат работает согласно правилам перехода p_{ij}, которые представляют алгоритм, реализуемый данной машиной Тьюринга. Каждое правило перехода предписывает машине в зависимости от текущего состояния Q_i и наблюдаемого в текущей клетке символа a_j записать в эту клетку новый символ a_{j1}, перейти в новое состояние Q_{i1} и переместиться на одну клетку влево или вправо по ленте (или остаться на текущей ячейке).

Конкретная машина Тьюринга задаётся перечислением элементов множества букв алфавита $\{a_j\}$, множества состояний $\{Q_i\}$ и набором правил, по которым работает машина. Они имеют вид: $\{p_{ij}=Q_i a_j \rightarrow Q_{i1} a_{j1} d_{ij}\}$ (если автомат находится в состоянии Q_i, а в обозреваемой ячейке записана буква a_j, то автомат переходит в состояние Q_{i1}, в ячейку вместо a_j записывается a_{j1}, каретка делает движение d_{ij}, которое имеет три варианта: на ячейку влево (L), на ячейку вправо (R), остаться на месте (N)). Стоит отметить, что в конкретный момент времени автомат «видит» только одну ячейку, не зная, что происходит в других. Для каждой возможной конфигурации имеется максимум одно правило. Кроме того, необходимо указать конечное и начальное состояния, начальную конфигурацию на ленте и расположение каретки машины.

Структурная схема машины Тьюринга представлена на рис. 1.1, где

B – внешняя память машины, интерпретируемая как лента, не ограниченная в обе стороны и разделенная на ячейки;

M – внутренняя память машины, определяющая состояния, в которых находится машина в любой момент времени;

Z – логический блок, который формирует символ, записываемый на ленту, управляет переходами между состояниями внутренней памяти, а также перемещает управление влево-вправо.

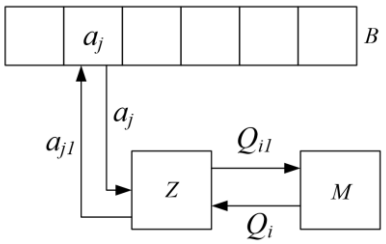

Рис. 1.1. Структурная схема машины Тьюринга

Запись алгоритма для машины Тьюринга определяется в виде функциональной таблицы (табл. 1.1).

Таблица 1.1 – Запись алгоритма в виде функциональной таблицы машины Тьюринга

Значения алфавита сигналов a_j	Значения алфавита состояний Q_i	
	Q_1	Q_2
a_0	Q_2a_0R	Q_1a_1R
a_1	Q_1a_2L	Q_2a_2R
a_2	Q_2a_1L	Q_0a_0N

Тройка символов Q_2a_0R из Таблица , записанная на пересечении столбца Q_1 и строки a_0 означает: если машина находится в состоянии Q_1 и с ленты считан символ a_0, то машина перейдет в состояние Q_2, а на ленту будет записан символ a_0, после чего автомат сдвинется на ячейку вправо. Функциональная таблица может быть записана в упрощенном виде путем исключения не обновляющейся в правиле информации (Таблица 1.2).

Таблица 1.2 – Упрощенная функциональная таблица машины Тьюринга

Значения алфавита сигналов a_j	Значения алфавита состояний Q_i	
	Q_1	Q_2
a_0	Q_2R	Q_1a_1R
a_1	a_2L	a_2R
a_2	Q_2a_1L	Q_0a_0N

Кроме функциональной таблицы информация может быть представлена в виде функциональной диаграммы, которая по своей сути является направленным графом (рис. 1.2).

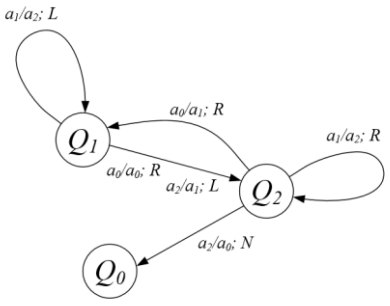

Рис. 1.2. Функциональная диаграмма машины Тьюринга, представленной в табл. 1.2

Для лучшего понимания машины Тьюринга разработан программный продукт-тренажер, фактически реализующий абстрактную машину Тьюринга

на условно[1] бесконечной ленте. Программа-тренажер «Машина Тьюринга» работает следующим образом:

В верхней части программы находится поле редактора, в которое можно ввести условие задачи в свободной форме.

Лента перемещается влево и вправо с помощью кнопок, расположенных слева и справа от нее. Двойным щелчком по ячейке ленты (или щелчком правой кнопкой мыши) можно изменить ее содержимое.

С помощью меню *Лента* можно запомнить состояние ленты во внутреннем буфере и восстановить ленту из буфера.

В поле *Алфавит* задаются символы выбранного алфавита сигналов. Пустой символ добавляется к алфавиту автоматически.

В таблице в нижней части окна набирается программа. В первом столбце записаны символы алфавита, он заполняется автоматически. В первой строке перечисляются все возможные состояния. Добавить и удалить столбцы таблицы (состояния) можно с помощью кнопок, расположенных над таблицей.

При вводе команды в ячейку таблицы сначала нужно ввести новый символ, затем направление перехода («>»– вправо, «<» – влево, «.» – оставаться на месте) и номер состояния. Если символ пропущен, по умолчанию он не изменяется. Если пропущен номер состояния, по умолчанию состояние автомата не изменяется.

Справа в поле *Комментарий* можно вводить в произвольной форме комментарии к решению. Чаще всего там объясняют, что означает каждое состояние машины Тьюринга.

Программа может выполняться непрерывно (F9) или по шагам (F8). Команда, которая сейчас будет выполняться, подсвечивается зеленым фоном. Скорость выполнения регулируется с помощью меню *Скорость*.

Задачи для машины Тьюринга можно сохранять в файлах. Сохраняется условие задачи, алфавит, программа, комментарии и начальное состояние ленты. При загрузке задачи из файла и сохранении в файле состояние ленты автоматически записывается в буфер.

Пример 1

Представить числа от 1 до 5 в унарной форме (в форме «счетных» палочек, например, 3⟶|||) на машине Тьюринга.

Функциональная таблица в таком случае будет представляться в виде Таблица 1.3.3, а реализация на тренажере «Машина Тьюринга» – на рис. 1.3.

В начальный момент времени t_0 автомат находится над символом «5». Согласно функциональной таблице автомату подается команда записать «4» и переместиться вправо, оставшись в прежнем состоянии. В момент времени t_1 автомат находится уже на пустом символе, где записывает символ «=», переходит в состояние Q_2 и перемещается вправо. В момент t_2 автомат поставит символ «|», перейдет в состояние Q_3 и переместится влево.

[1] Условно бесконечной лента названа потому, что длины ее достаточно для воплощения алгоритмов, представленных в лабораторной работе.

Таблица 1.3 – Функциональная таблица для машины Тьюринга, реализующей преобразования числа от 1 до 5 в унарную запись

Значения алфавита сигналов a_j	Значения алфавита состояний Q_i			
	Q_1	Q_2	Q_3	Q_4
_ (пустой символ, пробел)	$Q_2=R$	$Q_3\|L$	Q_1R	
0	Q_4_R		L	
1	$0R$		L	
2	$1R$		L	
3	$2R$		L	
4	$3R$		L	
5	$4R$		L	
=	$Q_2=R$		L	$\_N$
|		$Q_2\|R$	L	

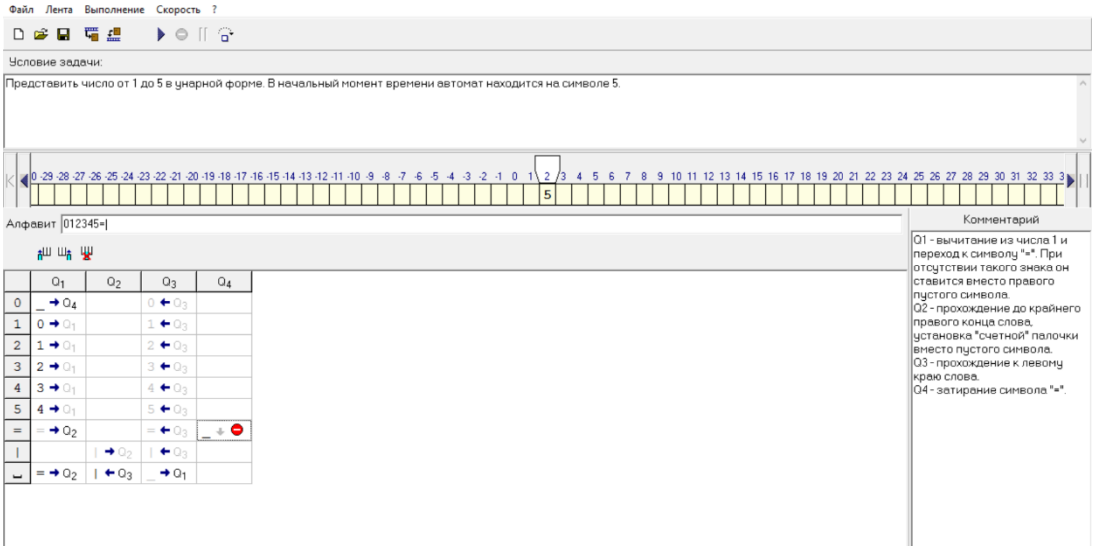

Рис. 1.3. Окно выполнения первого примера на программе-тренажере «Машина Тьюринга» в начальном состоянии.

В моменты t_3-t_4 автомат будет перемещаться влево, не меняя символы в ячейках. В момент t_5 автомат переместится вправо и перейдет в состояние Q_1, после чего алгоритм повторится. Он будет повторяться до тех пор, пока в ячейке с десятеричным числом не окажется символ «0». Он заменится на пустой символ и автомат перейдет в состояние Q_4, в котором автомат затирает символ «=» и переходит в терминальное состояние Q_0.

Пример 2. Представить умножение унарных чисел на машине Тьюринга.

Для решения данного примера представим функциональную таблицу в виде табл. 1.4 и реализацию на тренажере «Машина Тьюринга» на рис. 1.4.

Таблица 1.4 – Функциональная таблица для машины Тьюринга, реализующей умножение двух унарных чисел

Значения алфавита сигналов a_j	Значения алфавита состояний Q_i								
	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9
|	R	xQ_3L	L	xQ_5R	R	R	L	L	
*	Q_2R		Q_4L		R		Q_8L		Q_0N
x		R	L		R		L	$Q_4\|L$	$\|L$
=		Q_9L			Q_6R		L		
_				Q_1R		$Q_7\|L$			

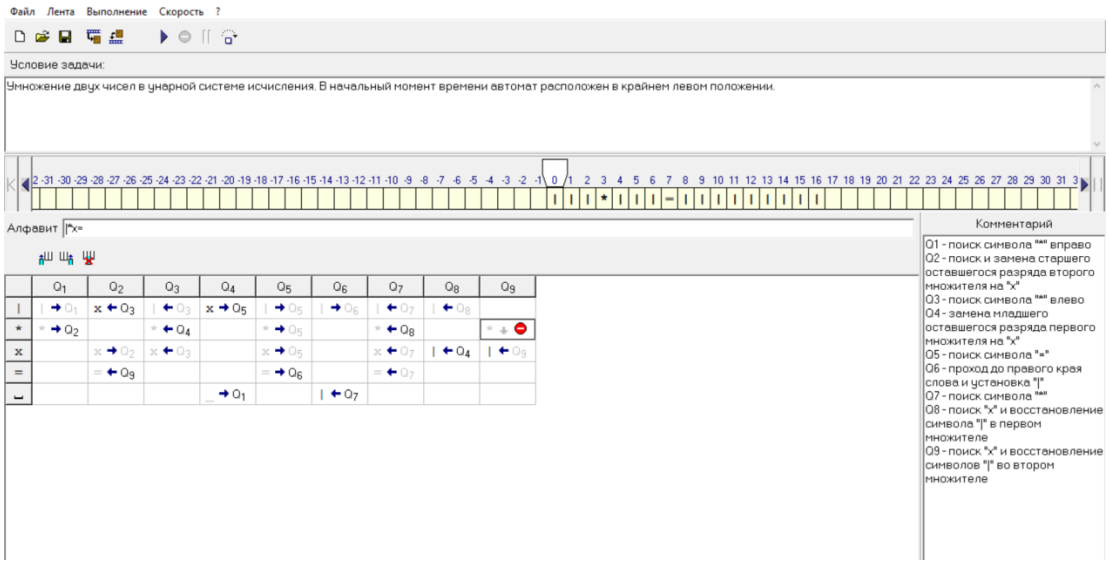

Рис. 1.4. Окно выполнения второго примера на программе-тренажере «Машина Тьюринга» в терминальном состоянии.

Проанализировав табл. 1.4 и рис. 1.4 станет ясно, что алгоритм умножения унарных чисел заключается в переносе палочек первого множителя за знак «=» столько раз, сколько палочек во втором множителя. При этом в качестве «дополнительной» памяти используется символ «x», указывающий, какие палочки уже перенесены за знак «=». Для символа «x» введены дополнительные правила, позволяющие вывести автомат в терминальное состояние.

Лабораторное задание

Вариант 1. Представить сложение десятеричных целых чисел. Автомат в первый момент времени находится справа от выражения (алфавит 0123456789_+).

Вариант 2. Представить вычитание десятеричных целых чисел при условии, что уменьшаемое всегда больше вычитаемого. Автомат в первый момент времени находится справа от выражения (алфавит 0123456789_-).

Вариант 3. Представить алгоритм сложения и вычитания чисел в двоичной системе исчисления. Автомат в первый момент времени находится в крайнем правом положении слова (алфавит 01_+-).

Контрольные вопросы

1 Что такое машина Тьюринга и чем она отличается от конечного автомата?
2 Для чего используется машина Тьюринга
3 Что такое внутренняя память машины Тьюринга?
4 Что такое внешняя память машины Тьюринга?
5 Какие бывают способы представления алгоритма на машине Тьюринга?
6 Почему машина Тьюринга является теоретической и не может быть полностью реализована?

Лабораторная работа № 2
Распознавание образов с помощью нейронных сетей. Простой перцептрон Маккалока – Питтса

Цель

Получить навыки моделирования простого перцептрона.

Задачи

1 Изучить основные понятия теории искусственных нейронных сетей.
2 Записать алгоритм обучения простого перцептрона на удобном языке программирования.
3 Обучить перцептрон на исходных данных согласно заданию.
4 Проанализировать с помощью обученного перцептрона тестовые данные согласно заданию
5 Оформить индивидуальный отчет.

Содержание индивидуального отчета

1 Краткий конспект основных теоретических положений работы.
2 Алгоритм обучения нейрона Маккалока – Питтса.
3 Листинг программы.
4 Результаты классификации тестовых данных.
5 Выводы и ответы на контрольные вопросы.

Теоретические сведения.

Термин «Искусственная нейронная сеть» или «Перцептрон» впервые упоминался в 1943 году в фундаментальной статье Уррена Маккалока и Уолтера Питтса по формализации исчисления идей и нервной активности. Фактически нейронная сеть – попытка машинного воспроизведения нервной системы человека, состоящей из нервных клеток – нейронов (рис. 2.1). Нейронная сеть основана на таких способностях человеческого мозга как способность обучаться и исправлять ошибки.

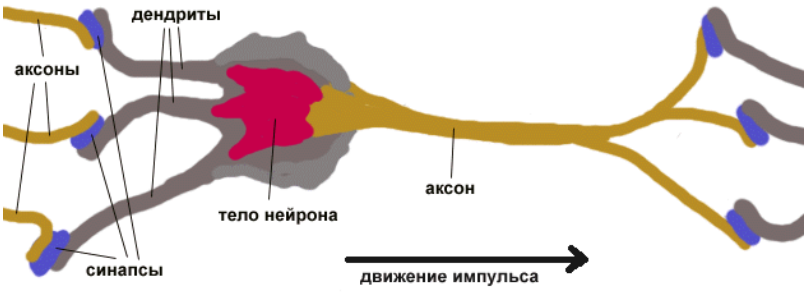

Рис. 2.1. Строение биологического нейрона

Подавляющее большинство существующих нейронных сетей основывается именно на той модели, что была предложена Маккалоком и Питтсом. Согласно модели Маккалока – Питтса нейрон представляет собой пороговый элемент (рис. 2.2).

Такой нейрон содержит:

– Вектор входных сигналов $x=(x_0; x_1,\dots x_n)$, где x_0 – единичный сигнал, подаваемый на пороговый вход;

– Вектор весов $w=(w_0, w_1, \ldots, w_n)$, где w_0 – пороговое значение нейрона;

– Сумматор, преобразующий входной сигнал в линейную комбинацию $u = \sum_{i=0}^{n} w_i x_i$ ($u = wx^T$ в матричной форме);

– Функция активации $f(u)$, которая в большинстве случаев принимает линейную, ступенчатую или сигмоидальную форму (в настоящей работе внимание уделяется ступенчатой функции типа $f(u) = \begin{cases} 1, & \text{если } u \geq 0, \\ 0, & \text{если } u < 0. \end{cases}$);

– Выходной сигнал $y=f(u)$.

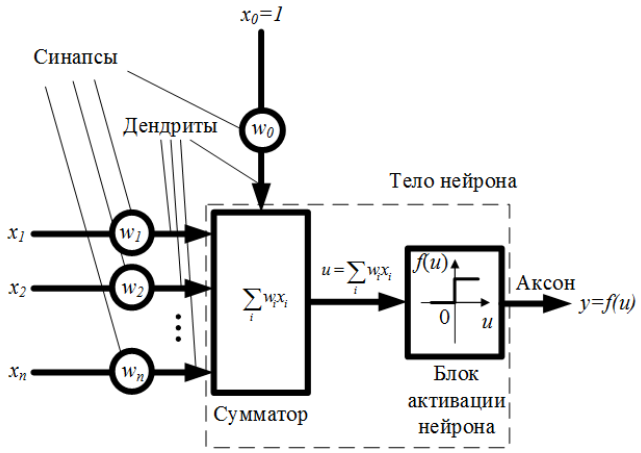

Рис. 2.2. Схема нейрона Маккалока – Питтса

Обучение такого перцептрона производится по следующему алгоритму:

1. Задание начального вектора весов $w(t=0)$ по *методу разделения центров масс*:

$$w = \frac{\sum x^{(1)} - \sum x^{(2)}}{m},$$

где $x^{(1)}$ – примеры (векторы сигналов) первого класса ($y'=1$) в обучающей выборке (здесь и далее под y' понимается известное значение класса, т.е. так называемый целевой выходной сигнал),

$x^{(2)}$ – примеры второго класса ($y'=0$) в обучающей выборке,

m – количество примеров в обучающей выборке.

Начальный вектор весов также может быть задан случайно. При этом значения весов принято брать в пределах [-0.1; 0.1].

2. $t=t+1$.

3 Для каждого вектора x из обучающей выборки:

3.1 Вычисление выходного сигнала y.

3.2 Вычисление нового вектора весов

$$w=w+\alpha(y'-y)x,$$

где α – коэффициент обучения, $\alpha \in [0;1]$.

4 Вычисление ошибки классификации обучающего множества (ошибки обучения):

$$e = \left| \frac{\sum (y' - y)}{m} \right|, \quad (1)$$

5 Если $e \leq e_{min}$ (e_{min} – заданное значение), то остановка обучения.

6 Если $t \geq t_{max}$ (t_{max} – заданное значение), то остановка обучения. В противном случае переход к п.2.

Результатом обучения является вектор весов.

Классификация тестового примера производится путем вычисления для него выходного сигнала с использованием полученного при обучении вектора весов.

Проверка эффективности классификации тестовых данных производится на основе оценки ошибки классификации тестового множества (ошибки тестирования) по формуле (1). Полученные значения ошибки обучения и тестирования позволяют сказать об эффективности модели перцептрона. В этом случае возможно несколько вариантов:

а) Высокая ошибка обучения – данные нелинейно разделимы и необходимо увеличить количество нейронов и/или слоев.

б) Низкая ошибка обучения и высокая ошибка тестирования – недостаточно обучающих данных (нерепрезентативная выборка). В этом случае говорят о «переобучении» нейронной сети.

в) Низкие ошибки обучения и тестирования – данные линейно разделимы и использование перцептрона Маккалока-Питтса является актуальным для решения задачи их классификации

Пример.

Имеется набор точек в двумерном пространстве (рис. 2.3), принадлежащих двум классам и описываемых в виде ($x_0=1; x_1;x_2;y'$):

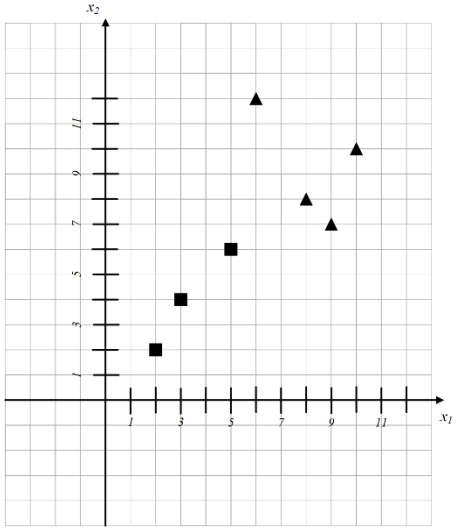

Рис. 2.3. Исходные данные

(1;9;7;1); (1;8;8;1); (1;10;10;1); (1;6;12;1); (1;5;6;0); (1;3;4;0); (1;2;2;0)

Обучим нейронную сеть с параметрами $\alpha=0.1$, $e_{min}=0$, $t_{max}=100$ на данной выборке:

1:

$$w = \frac{\begin{bmatrix}1\\9\\7\end{bmatrix} + \begin{bmatrix}1\\8\\8\end{bmatrix} + \begin{bmatrix}1\\10\\10\end{bmatrix} + \begin{bmatrix}1\\6\\12\end{bmatrix} - \begin{bmatrix}1\\5\\6\end{bmatrix} - \begin{bmatrix}1\\3\\4\end{bmatrix} - \begin{bmatrix}1\\2\\2\end{bmatrix}}{7} = \begin{bmatrix}0.14\\3.29\\3.57\end{bmatrix}$$

2:t=1

3. Для примера (1;9;7;1)

$$u_1 = \begin{matrix}0.14 & 3.29 & 3.57\end{matrix} \cdot \begin{bmatrix}1\\9\\7\end{bmatrix} = 0.14 \cdot 1 + 3.29 \cdot 9 + 3.57 \cdot 7 = 54,74$$

3.1

$$y_1(u_1) = 1$$

$$w = \begin{bmatrix}0.14\\3.29\\3.57\end{bmatrix} + 0.1 \cdot (1-1) \cdot \begin{bmatrix}1\\9\\7\end{bmatrix} = \begin{bmatrix}0.14\\3.29\\3.57\end{bmatrix}$$

3.2

3. Для примера (1;8;8;1)

$$u_2 = \begin{matrix}0.14 & 3.29 & 3.57\end{matrix} \cdot \begin{bmatrix}1\\8\\8\end{bmatrix} = 55,02$$

3.1

$$y_2(u_2) = 1$$

$$w = \begin{bmatrix}0.14\\3.29\\3.57\end{bmatrix} + 0.1 \cdot (1-1) \cdot \begin{bmatrix}1\\8\\8\end{bmatrix} = \begin{bmatrix}0.14\\3.29\\3.57\end{bmatrix}$$

3.2

…

3. Для примера (1;5;6;0)

$$u_5 = \begin{matrix}0.14 & 3.29 & 3.57\end{matrix} \cdot \begin{bmatrix}1\\5\\6\end{bmatrix} = 38,01$$

3.1

$$y_5(u_5) = 1$$

3.2 $$w = \begin{bmatrix}0.14\\3.29\\3.57\end{bmatrix} + 0.1 \cdot (0-1) \cdot \begin{bmatrix}1\\5\\6\end{bmatrix} = \begin{bmatrix}0.04\\2.79\\2.97\end{bmatrix}$$

$$e = \frac{\left|(1-1) + (1-1) + (1-1) + (1-1) + (0-1) + (0-1) + (0-1)\right|}{7} = 0.43$$

4

5 $e>e_{min}$

6 $t<t_{max}$

2. t=2

3. Для примера (1;9;7;1)

3.1 $u_1 = \begin{bmatrix} 0.04 & 2.79 & 2.97 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 9 \\ 7 \end{bmatrix} = 45,94$

$y_1(u_1) = 1$

…

$$w = \begin{bmatrix} -1.56 \\ 0.19 \\ 0.07 \end{bmatrix}$$

В итоге, при t =15 у нас получится и $e==e_{min}$, что положит конец обучению (рис. 2.4).

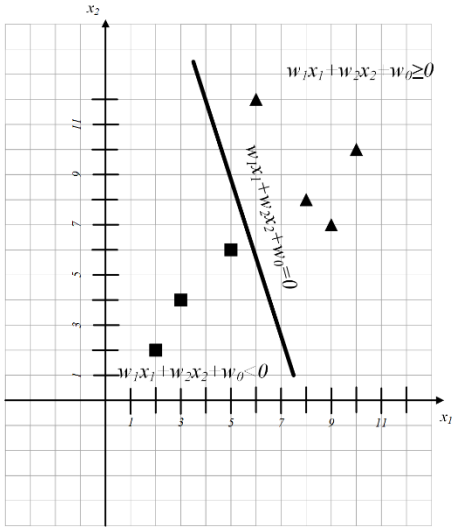

Рис. 2.4. Наглядное представление линейного разделения исходных данных

Допустим, необходимо определить класс точки с координатами (3;13) (рис. 2.5).

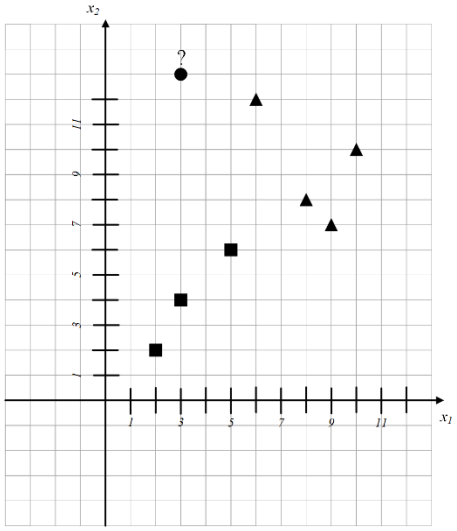

Рис. 2.5. Тестовый пример

Пропустив текущую точку через обученный ранее нейрон, получим следующее:

$$u = \begin{bmatrix} -1.56 & 0.19 & 0.07 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 3 \\ 13 \end{bmatrix} = -0.08$$

$$y(u) = 0$$

Таким образом, класс представленной точке – 0 (рис. 2.6).

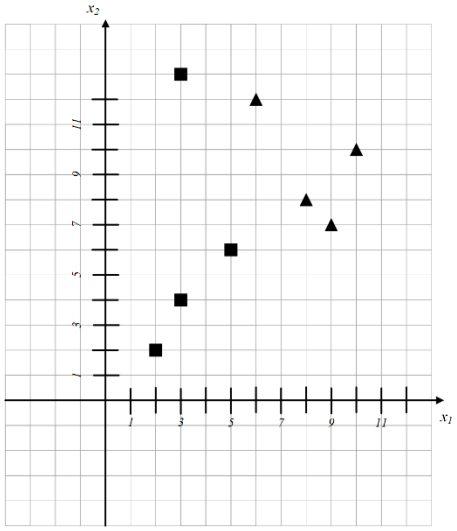

Рис. 2.6. Результат классификации

Если бы класс тестируемой точки был заранее известен, то можно было бы проверить эффективность перцептрона. Стоит отметить, что для более адекватного результата здесь следует использовать как можно большую и разнообразную тестовую выборку.

Лабораторное задание

Вариант 1. Научить нейрон отличать оттенки синего цвета от оттенков зеленого. Заполнить неизвестные оттенки путем классификации на основе полученного нейрона. Все данные указаны в табл. 2.1.

Таблица 2.1 – Данные для варианта 1

Тон	**RGB код**
1	**2**
Зеленый	173, 255, 47
Зеленый	127, 255, 0
Зеленый	124, 252, 0
Зеленый	0, 255, 0
Зеленый	50, 205, 50
Зеленый	152, 251, 152
Зеленый	144, 238, 144
Зеленый	0, 250, 154

Окончание табл. 2.1

1	2
Зеленый	0, 255, 127
Зеленый	60, 179, 113
Зеленый	46, 139, 87
Зеленый	128, 128, 0
Зеленый	85, 107, 47
Зеленый	102, 205, 170
Зеленый	143, 188, 143
Зеленый	32, 178, 170
Синий	240,248,255
Синий	138,43,226
Синий	95,158,160
Синий	152,245,255
Синий	142,229,238
Синий	122,197,205
Синий	83,134,139
Тон	RGB код
Синий	100,149,237
Синий	0,0,139
Синий	0,139,139
Синий	72,61,139
Синий	0,206,209
Синий	0,191,255
Синий	0,191,255
Синий	0,178,238
?	154, 205, 50
?	30,144,255
?	25,25,112
?	34, 139, 34
?	28,134,238
?	0, 128, 0
?	0, 100, 0
?	0,104,139
?	0,154,205
?	107, 142, 35

Вариант 2. Научить нейрон отличать типы растений рода касатиков. Заполнить неизвестные типы путем классификации на основе полученного нейрона. Все данные указаны в табл. 2.2.

Таблица 2.2 – Данные для варианта 2

Тип	Длина чашелистика, см	Ширина чашелистика, см	Длина лепестка, см	Ширина лепестка, см
Щетинистый	5.1	3.5	1.4	0.2
Щетинистый	4.9	3.0	1.4	0.2
Щетинистый	5.0	3.5	1.6	0.6
Щетинистый	5.4	3.9	1.7	0.4
Щетинистый	4.4	2.9	1.4	0.2
Щетинистый	4.8	3.4	1.6	0.2
Щетинистый	5.8	4.0	1.2	0.2
Щетинистый	5.4	3.4	1.7	0.2
Щетинистый	4.6	3.4	1.4	0.3
Щетинистый	5.0	3.3	1.4	0.2
Щетинистый	5.3	3.7	1.5	0.2
Щетинистый	4.4	3.2	1.3	0.2
Щетинистый	4.5	2.3	1.3	0.3
Щетинистый	4.8	3.1	1.6	0.2
Щетинистый	5.7	3.8	1.7	0.3
Разноцветный	7.0	3.2	4.7	1.4
Разноцветный	6.4	3.2	4.5	1.5
Разноцветный	4.9	2.4	3.3	1.0
Разноцветный	5.0	2.0	3.5	1.0
Разноцветный	6.1	2.8	4.7	1.2
Разноцветный	5.1	2.5	3.0	1.1
Разноцветный	5.0	2.3	3.3	1.0
Разноцветный	5.5	2.6	4.4	1.2
Разноцветный	5.5	2.4	3.8	1.1
Разноцветный	6.9	3.1	4.9	1.5
Разноцветный	6.6	2.9	4.6	1.3
Разноцветный	5.2	2.7	3.9	1.4
Разноцветный	6.2	2.2	4.5	1.5
Разноцветный	6.1	2.8	4.0	1.3
Разноцветный	6.2	2.9	4.3	1.3
?	4.3	3.0	1.1	0.1
?	6.5	2.8	4.6	1.5
?	5.5	2.3	4.0	1.3
?	5.7	4.4	1.5	0.4
?	5.4	3.7	1.5	0.2
?	5.7	2.8	4.5	1.3
?	6.3	3.3	4.7	1.6
?	5.9	3.0	4.2	1.5
?	5.1	3.8	1.6	0.2
?	5.5	4.2	1.4	0.2

Вариант 3. Попробовать обучить нейрон реализовывать логические операции «И», «ИЛИ», «ИСКЛЮЧАЮЩЕЕ ИЛИ». Описать полученные ошибки обучения.

Контрольные вопросы

1 Что такое перцептрон? Основная идея создания искусственных нейронных сетей
2 Опишите основные составляющие модели нейрона Маккалока – Питтса.
3 Что является результатом обучения искусственного перцептрона?
4 Что называют ошибкой классификации перцептрона? Каким образом ошибка классификации позволяет определить эффективность обучения искусственной нейронной сети?

Лабораторная работа № 3
Кластерный анализ

Цель

Разработать программу, позволяющую произвести кластеризацию множества объектов. Расстояние между примерами и метод кластеризации выбрать в соответствии с вариантом.

Задачи

1 Изучить основные понятия теории кластерного анализа.
2 Записать алгоритм кластеризации (алгоритм выбрать согласно заданию) на удобном языке программирования.
3 Проанализировать результаты кластеризации.
4 Оформить индивидуальный отчет.

Содержание индивидуального отчета

1 Краткий конспект основных теоретических положений работы.
2 Листинг программы.
3 Результаты кластеризации тестовых данных (количество кластеров и количество примеров в каждом кластере).
4 Выводы и ответы на контрольные вопросы.

Теоретические сведения

Предыдущая лабораторная работа была посвящена искусственным нейронным сетям, которые в первую очередь предназначены для автоматической классификации данных на основе методов искусственного интеллекта. При классификации подразумевается, что множество объектов, которое используется в качестве эталона (например, обучающая выборка для перцептрона), заранее разбито на классы. Однако при решении реальных задач априорное разбиение не всегда представляется возможным. При неизвестной заранее принадлежности объектов исследуемого множества используют автоматическую процедуру разбиения на классы, называемую кластеризацией, результаты которой – подмножества объектов одного класса – называют кластерами.

Первым и основным шагом кластеризации является выбор метрики, т.е. способа определения расстояния d между примерами, представляемыми точками в n-мерном пространстве (или n-мерными векторами). Наиболее часто используется Евклидово расстояние:

$$d = \sqrt{\sum_{i=1}^{n} x_i - y_i^{\,2}},$$

где x_i и y_i – i-ые признаки объектов x и y, соответственно.

Для удобства при вычислении Евклидова расстояния квадратный корень опускают (на результат вычислений это не влияет, так как получаемый результат измеряется в абсолютной, а не в относительной мере).

Одним из основных недостатков Евклидова расстояния является зависимость от дисперсий отдельных признаков. При наличии отдельных

выбросов по признакам эффективность такой метрики резко падает, так как теряются компоненты по оставшимся признакам. По сравнению с Евклидовым в этом плане оказывается эффективнее расстояние городских кварталов (или Манхэттенское расстояние), которое вычисляется по формуле:

$$d = \sum_i \left| x_i - y_i \right| .$$

Для установления одинаковой зависимости по всем признакам также используют нормирование (приведение к одному диапазону). Нормирование в основном производится в диапазоне [0;1] и в основном принимает вид:

$$x_i = \frac{x_i - x_{i\min}}{x_{i\max} - x_{i\min}}$$

С той же целью, но без использования нормирования применяют расстояние Канберра:

$$d = \sum_i \left| \frac{x_i - y_i}{x_i + y_i} \right|$$

При разработке алгоритма кластеризации важным также является определение степени близости (или расстояния) между исследуемым объектом и кластером. Для этого достаточно выбрать представление кластера в виде одного элемента, чтобы потом в качестве степени близости использовать стандартные меры расстояния. В большинстве случаев используется центр кластера (элемент со средними значениями признаков).

Основой кластеризации является разбиение множества образов на подмножества близких между собой образов. Это разбиение может быть различным в зависимости от метода кластеризации:

1 Пороговый метод. Произвольно выбирается пример из множества объектов, который считается центром первого кластера. Далее вычисляется расстояние между этим кластером и следующим объектом. Если расстояние больше заранее установленного порога, то формируется следующий кластер. В противном случае объект добавляется в первый кластер. Для последующих объектов сравнение идет со всеми существующими на момент сравнения кластерами. Если расстояние до всех кластеров больше порогового, то создается новый кластер. Если расстояние до нескольких кластеров меньше порогового, то относим объект к ближайшему кластеру. Следует отметить, что центры кластеров при добавлении нового объекта пересчитываются.

2 Метод кластеризации слиянием. Каждый пример изначально считается кластером. Составляется таблица расстояний между всеми кластерами, в которой выбираются два кластера с наименьшим расстоянием. Эти кластеры сливаются, в результате чего таблица уменьшается. Слияние производится до тех пор, пока не будет достигнута заданная длина таблицы или расстояние между кластерами не станет выше пороговой величины.

3 Метод кластеризации по *k*-средним. На первом этапе выбирается произвольно k центров кластеров (k задано), не обязательно совпадающих с центрами примеров. Далее каждый пример относится к тому кластеру, расстояние до которого минимально. После отнесения всех примеров центры пересчитываются, и операция отнесения повторяется вновь и продолжается до тех пор, пока центры кластеров не стабилизируются.

Пример

Имеется набор точек в двумерном пространстве (Рис. 3.1), принадлежащих двум классам и описываемых в виде (x_1, x_2)

{10, 20}; {0, 1}; {5, 7}; {7, 1}; {0, 12}; {-5, -4}; {20, 10}; {0, 15}; {-1, 9}; {-5, -3}; {18, 0}; {16, 19}.

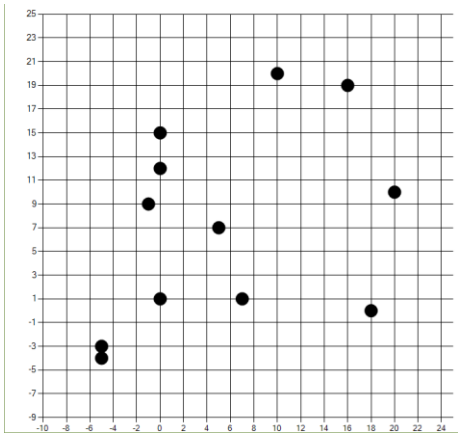

Рис. 3.1. Исходный набор точек

Необходимо разделить данные точки на кластеры. Представим пример реализации порогового метода

1. Нормализуем данные (рис. 3.2)

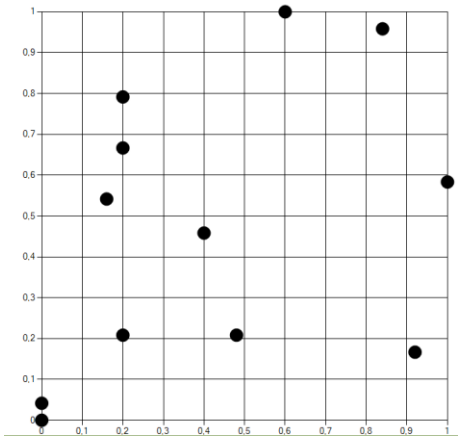

Рис. 3.2. Нормализованный набор точек

2. Произвольно выбираем первый центр кластера и задаем порог 0.3 (рис. 3.3)

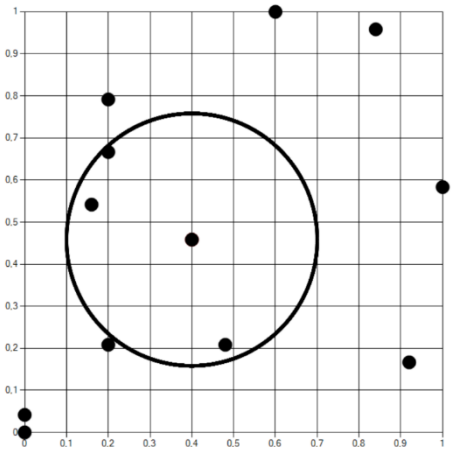

Рис. 3.3. Первый случайно выбранный кластер

3. Для следующего элемента (рис. 3.4) расстояние больше порогового, поэтому создаем новый кластер.

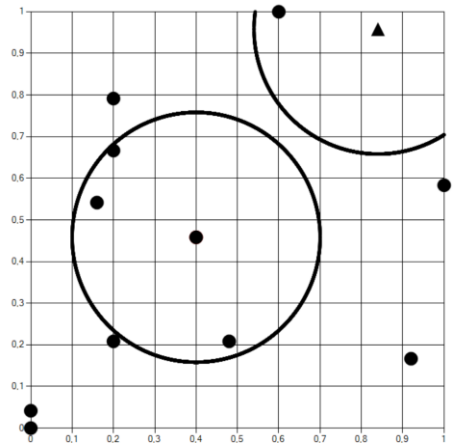

Рис. 3.4. Создание нового кластера

После прохождения всех точек будет получена результирующая картина, представляющая разделение представленных точек на кластеры (рис. 3.5).

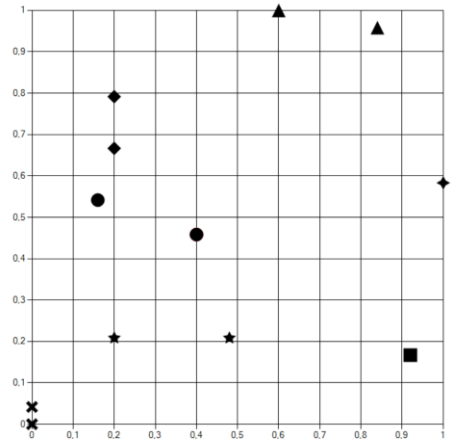

Рис. 3.5. Результат кластеризации данных пороговым методом (пороговое расстояние – 0.3; окружности, описывающие кластеры, не нарисованы для наглядности)

Очевидно, что результат порогового метода является сильно зависимым от порогового значения, а результирующие кластеры ограничиваются только окружностями, вследствие чего он не является оптимальным из трех представленных. Результат разбиения по методу слияния и по k-средним данных из примера представлен на рис. 3.6 и рис. 4.7, соответственно.

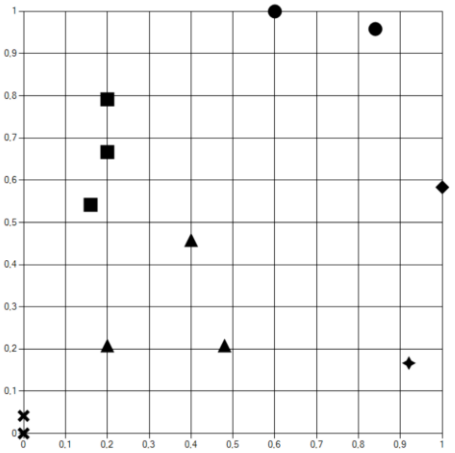

Рис. 3.6. Кластеризация методом слияния
(заданная длина таблицы – 4, пороговое расстояние – 0.4)

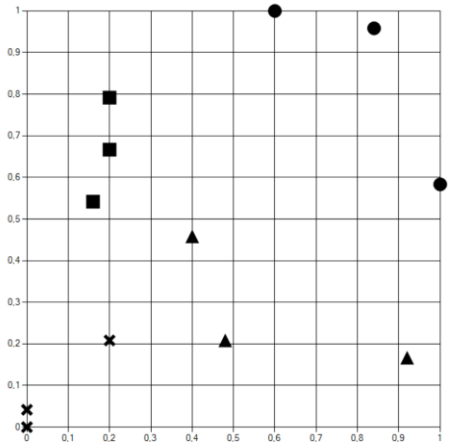

Рис. 3.7. Кластеризация методом k-средних (k=4)

На практике чаще используют метод кластеризации по k-средних, так как он позволяет задать жесткое значение количества кластеров без использования таких нестабильных параметров, как пороговое значение.

Лабораторное задание

Выполнить кластеризацию согласно табл. 3.1.

Таблица 3.1 – Задание для выполнения лабораторной работы № 3

№ варианта	Выборка	Мера расстояния	Метод кластеризации
1	2	3	4
1	s1	Евклидова	Пороговый
2	s2	Евклидова	Пороговый
3	a1	Евклидова	Пороговый

Окончание табл. 3.1

1	2	3	4
4	s1	Манхэттенская	Пороговый
5	s2	Манхэттенская	Пороговый
6	a1	Манхэттенская	Пороговый
7	s1	Канберра	Пороговый
8	s2	Канберра	Пороговый
9	a1	Канберра	Пороговый
10	s1	Евклидова	Слияния
11	s2	Евклидова	Слияния
12	a1	Евклидова	Слияния
13	s1	Манхэттенская	Слияния
14	s2	Манхэттенская	Слияния
15	a1	Манхэттенская	Слияния
16	s1	Канберра	Слияния
17	s2	Канберра	Слияния
18	a1	Канберра	Слияния
19	s1	Евклидова	*k*-средних
20	s2	Евклидова	*k*-средних
21	a1	Евклидова	*k*-средних
22	s1	Манхэттенская	*k*-средних
23	s2	Манхэттенская	*k*-средних
24	a1	Манхэттенская	*k*-средних
25	s1	Канберра	*k*-средних
26	s2	Канберра	*k*-средних
27	a1	Канберра	*k*-средних

Контрольные вопросы

1 Для чего предназначена кластеризация?
2 Основные меры определения расстояния при кластеризации. Достоинства и недостатки.
3 Что такое нормализация данных и для чего она необходима?
4 Основные методы кластеризации. Достоинства и недостатки.

Лабораторная работа № 4
Интеллектуальный анализ данных с помощью ПО Weka.
Предобработка данных

Цель

Изучить интерфейс ПО Weka. Ознакомиться с функционалом модуля Explorer, вкладки Preprocess.

Задачи

1 Изучить интерфейс ПО Weka.
2 Произвести предобработку данных с помощью модуля Explorer, вкладки Preprocess.
3 Сохранить и проанализировать результаты предобработки данных.
4 Оформить индивидуальный отчет.

Содержание индивидуального отчета

1 Краткий конспект основных теоретических положений работы.
2 Скриншоты основных действий в ПО Weka и их описание.
3 Выводы и ответы на контрольные вопросы.

Теоретические сведения

WEKA (Waikato Environment for Knowledge Analysis) — свободное открытое программное обеспечение (ПО) для анализа данных написана на языке Java в университете Вайкато (Новая Зеландия), распространяется под лицензией GNU General Public License version 21 (GPLv2) и предоставляет пользователю возможность предобработки данных, решения задач классификации, регрессии, кластеризации и поиска ассоциативных правил, а также визуализации данных и результатов анализа. Программа очень проста в освоении, пожалуй, имеет самый интуитивный интерфейс среди всех программ такого типа, бесплатна и может быть дополнена новыми алгоритмами, средствами предобработки и визуализации данных.

Для начала работы с WEKA достаточно запустить файл RunWeka.bat. Появится окно Weka GUI Chooser (рис. 4.1), в котором будет предложено выбрать один из четырёх модулей программы. Рассмотрим подробнее модуль Explorer.

Рис. 4.1. Основное окно программы WEKA

Модуль Explorer (Исследователь) – это основной модуль программы (рис. 4.2). Он имеет несколько вкладок.

Вкладка предобработки Preprocess позволяет импортировать данные из базы и применять к ним алгоритмы фильтрации, например, переводить количественные признаки в дискретные, удалять объекты и признаки по заданному критерию.

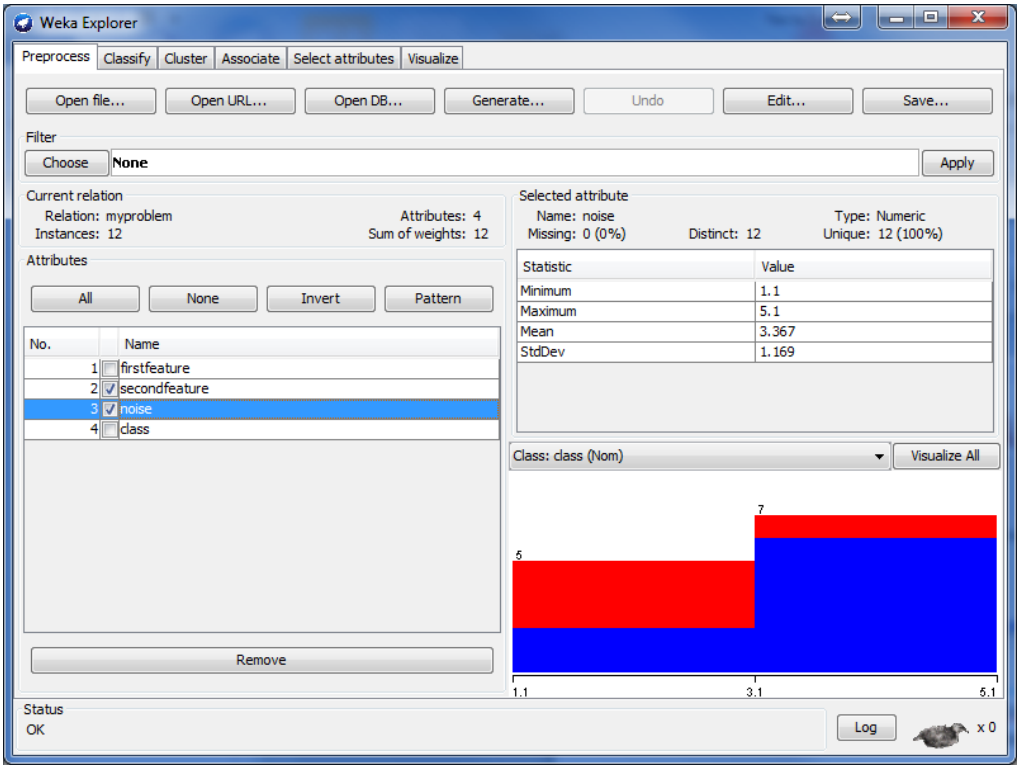

Рис. 4.2. Вкладка Preprocess модуля Explorer (выделены два признака: второй и третий, для выбранного третьего признака построена гистограмма значений)

Вкладка классификации Classify позволяет применять алгоритмы классификации и регрессии (в Weka они не различаются и называются classifiers) к выборке данных, оценивать предсказательную способность алгоритмов, визуализировать ошибочные предсказания, ROC-кривые, и сам алгоритм, если это возможно (в частности, решающие деревья).

Вкладка поиска ассоциативных правил Associate решает задачу выявления всех значимых взаимосвязей между признаками.

Вкладка кластеризации Cluster даёт доступ к алгоритму k-средних, EM-алгоритму и другим.

Вкладка отбора признаков Select attributes даёт доступ к методам отбора признаков.

Вкладка визуализации Visualize строит матрицу графиков разброса признаков (scatter plot matrix), позволяет выбирать и увеличивать графики, и т. д.

Сначала доступна только вкладка Preprocess, поскольку для выполнения остальных задач нужны данные. Отметим, что последовательность вкладок не

всегда соответствует этапам решения задачи. Например, после загрузки данных в память ЭВМ можно перейти к визуализации признаков. В настоящей лабораторной работе рассмотрим возможности работы на вкладке Preprocess.

Вкладка Preprocess является первой точкой анализа данных с помощью Weka. В данном окне пользователь может произвести следующие операции:

1. Загрузка данных (кнопки Open file, Open URL, Open DB). При нажатии на кнопку Open file, открываем необходимый файл (arff, CSV, C4.5, бинарный, libsvm и др.). Основной формат – arff. В каталоге %WEKAPATH%\data можно посмотреть примеры arff-файлов (с программой поставляются данные нескольких задач). При нажатии на кнопки Open URL и Open DB можно загрузить файл из интернета или выгрузить информацию из базы данных, соответственно.

Файл, используемый в качестве данных в Weka, содержит следующую информацию:

- название задачи (секция @relation).
- описание признаков: их названия и типы (каждое описание после ключевого слова @attribute). Основными типами являются числовой (numeric) и номинальный (в фигурных скобках через запятую перечисляются всевозможные значения признака)[1]
- значения признаков (секция @data).

2. Генерация модельных данных (кнопка Generate). При нажатии на кнопку Generate и выборе соответствующих параметров можно автоматически создать данные для последующей обработки.

3. Вспомогательные операции над данными (кнопки Undo, Edit и Save). Данные операции позволяют отменить предыдущее действие (кнопка Undo), редактировать данные (кнопка Edit), а также сохранить данные (кнопка Save).

4. Использование различных фильтров данных (панель Filter). С помощью панели Filter можно выбрать и применить различные методы преобразования данных для последующей обработки.

5. Просмотр признаков текущего объекта, их статистических данных и гистограммы значений, а также выполнение различных операций над ними (панели Attributes и Selected attribute и кнопки «All», None, Invert и др.). С помощью панели Attributes можно выбрать признак (см. рис. 4.2), при этом на панели Selected attribute отображается гистограмма значений признака. Также предусмотрен одновременный просмотр гистограмм всех признаков (кнопка Visualize all) (рис. 4.3). Слева от Visualize all находится компонент выбора целевого признака, т.е. какой признак считать классом при визуализации (выбор NoClass соответствует тому, что все объекты приписываются одному классу и для каждого признака показывается гистограмма распределения всех его значений). В верхней части панели Selected attribute отображается простая

[1] Кроме номинальных и числовых признаков, могут быть также признаки типа string, date, relational и др.

статистика: максимальное, минимальное и среднее значения признака, выборочная дисперсия, число пропусков, тип признака, число уникальных значений. Нажатие на кнопку Remove приводит к удалению отмеченных признаков.

Лабораторное задание

Привести файл с выборкой из табл. 4.1 к формату, распознаваемому в ПО Weka. Выполнить предобработку данных с помощью ПО Weka согласно табл. 4.1. Сохранить полученную выборку.

Таблица 4.1 – Задание для выполнения лабораторной работы № 4

№ вар	Название выборки	Метод предобработки	Примечание
1	2	3	4
1	Iris	Normalize	Интервал от 0 до 1
2	Winequality-white	Randomize	
3	Winequality-red	Normalize	Интервал от -1 до 1
4	Glass	Randomize	
5	Movement_libras	Normalize	Интервал от 0 до 1
6	Spambase	Randomize	
7	Pokerhand	Normalize	Интервал от -1 до 1
8	Segmentation	Randomize	
9	Letter-recognition	Normalize	Интервал от 0 до 1
10	Data_banknote_authentication	Randomize	
11	Agaricus-lepiota	NominalToBinary (Unsupervised)	Признак 1
12	Semeion	NumericToBinary	
13	Flag	NominalToBinary (Unsupervised)	Признак класса
14	Adult	NominalToBinary (Unsupervised)	Признак класса
15	Iris	Discretize	Признак 2, 2 интервала
16	Winequality-white	Normalize	Интервал от 0 до 1
17	Winequality-red	Discretize	Признак 1, 3 интервала
18	Glass	AddExpression	RI^2
19	Movement_libras	Discretize	Признак 4, 4 интервала

Окончание табл. 4.1

1	2	3	4
20	Spambase	NominalToBinary (Unsupervised)	Признак класса
21	Pokerhand	Discretize	Признак 3, 4 интервала
22	Segmentation	RemoveUseless	
23	Letter-recognition	Discretize	Признак 5, 5 интервалов
24	Data_banknote_authentication	NominalToBinary (Unsupervised)	Признак класса
25	Agaricus-lepiota	RemoveUseless	
26	Semeion	AddID	Название: no.
27	Flag	NumericToNominal	Признак класса
28	Adult	Normalize	Интервал от 0 до 1

Контрольные вопросы

1 Основные возможности ПО WEKA 3.7.10.
2 Какой этап является первым и необходимым при интеллектуальном анализе данных? Какие действия он может предполагать?
3 Какие форматы файлов поддерживает ПО Weka?

Лабораторная работа № 5
Интеллектуальный анализ данных с помощью ПО Weka.
Классификация данных

Цель

Изучить функционал модуля Explorer ПО Weka, вкладка Classify.

Задачи

1. Изучить интерфейс вкладки Classify ПО Weka.
2. Произвести классификацию данных с помощью заданного вариантом метода.
3. Сохранить и проанализировать результаты классификации данных.
4. Оформить индивидуальный отчет.

Содержание индивидуального отчета

1. Краткий конспект основных теоретических положений работы.
2. Скриншоты основных действий в ПО Weka и их описание.
3. Выводы и ответы на контрольные вопросы.

Теоретические сведения

Вкладка классификации «Classify» (рис. 5.1) позволяет применять алгоритмы классификации и регрессии (в Weka они не различаются и называются classifiers) к выборке данных, оценивать предсказательную способность алгоритмов, визуализировать ошибочные предсказания, ROC-кривые, и сам алгоритм, если это возможно (в частности, решающие деревья).

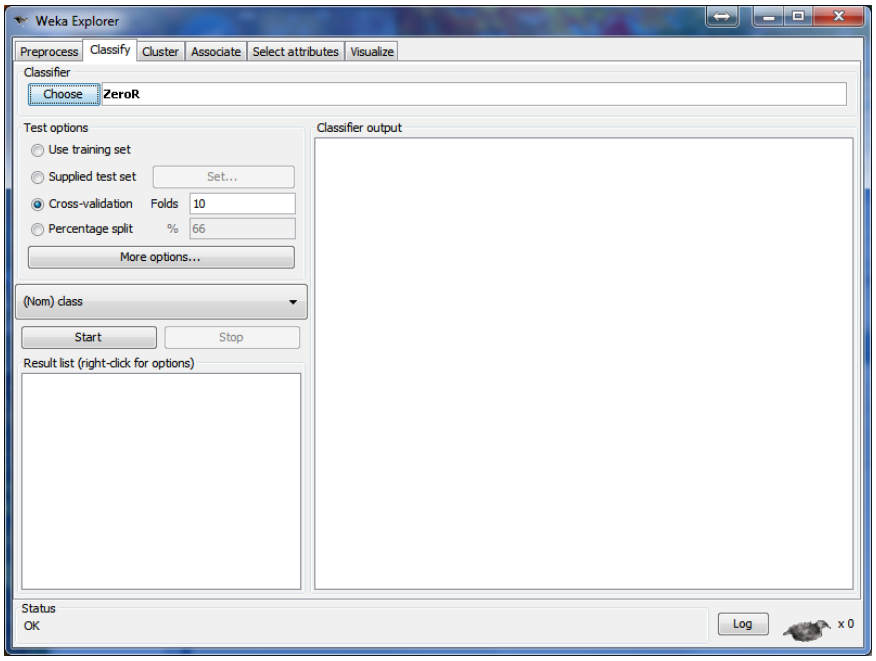

Рис. 5.1. Вид вкладки классификации

Для начала работы по классификации необходимо выбрать классификатор. Для этого отведена отдельная панель классификатора Classifier. При нажатии на кнопку Choose открывается окно выбора классификаторов (рис. 5.2).

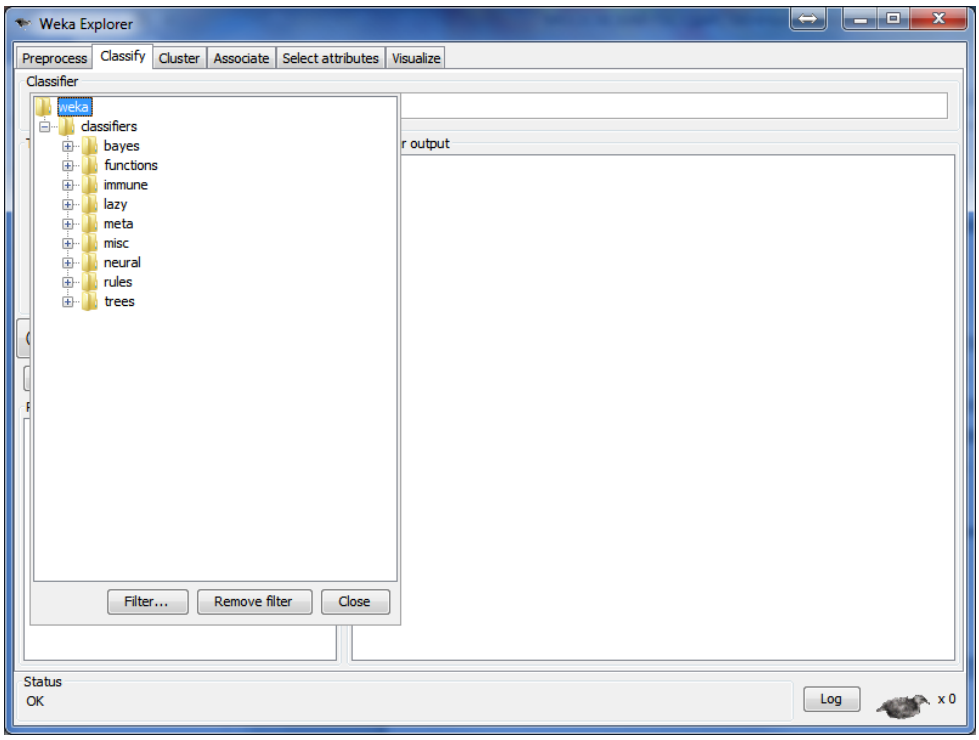

Рис. 5.2. Окно выбора классификаторов

Благодаря панели Test options можно определить, на чем будет тестироваться классификатор: на обучающей выборке (use training set), на тестовой выборке из отдельного файла (supplied test set), по блокам (cross-validation) или с помощью разделения исходной выборки на обучающую и тестовую (percentage split). При выборе некоторых опций придётся указать параметры тестирования. Например, при выборе cross-validation надо указать, на сколько блоков (фолдов) разбивать выборку. Очень полезная кнопка More options, которая позволяет выбрать вид отчёта об обучении классификатора.

Ниже панели Test options находится компонент для выбора целевого признака (что будет считаться классом в задаче).

Нажатие на кнопку Start запускает обучение классификатора. На панели Classifier output отображается отчёт об обучении. На начальном этапе здесь отображается информация о задаче и алгоритме классификации, а по завершению классификации иллюстрируются ее результаты (если включена опция More options/Output predictions). В конце описания представляется различная статистика работы классификатора, включая процент верных ответов (рис. 5.3). Обратим внимание только на следующую информацию, которая традиционна для подобных систем.

```
=== Confusion Matrix ===

 a b   <-- classified as
 7 1 | a = A
 1 3 | b = B
```

Рис. 5.3. Статистика результатов классификации

Выводится матрица размера C x C, где C - число классов, ij -й элемент матрицы равен числу объектов из i-го класса, которые были отнесены к j-му. Число верно классифицированных объектов равно сумме элементов, стоящих на главной диагонали.

На левой нижней панели Result List выводится перечень всех запусков классификаций данных. Новый элемент в этом списке появляется после завершения настройки и тестирования классификатора (вызванных нажатием кнопки Start). В списке указывается время окончания классификации и название классификатора. Если по элементу списка щёлкнуть правой кнопкой мыши, то появится дополнительное меню, с помощью которого Вы сможете посмотреть отчёт в отдельном окне (View in separate window), загрузить и сохранить модель, т.е. тип и параметры классификатора (Load model, Save model), визуализировать ошибки классификатора (Visualize classifier errors, см. также работу с вкладкой Visualize), посмотреть различные кривые отступов и ошибок.

Лабораторное задание

Привести файл с выборкой из табл. 5.1 к формату, распознаваемому в ПО Weka. Выполнить предобработку данных, если необходимо. Выполнить классификацию согласно табл. 5.1.

Таблица 5.1 – Задание для выполнения лабораторной работы № 5.

№ вар.	Название выборки	Метод классификации	Обучающая / тестовая выборка, %	Примечание
1	2	3	4	5
1	Iris	MultilayerPerceptron	90/10	
2	Winequality-white	IBk	90/10	$k = 1$
3	Winequality-red	MultilayerPerceptron	90/10	
4	Glass	IBk	90/10	$k = 5$
5	Movement_libras	MultilayerPerceptron	90/10	
6	Spambase	MultilayerPerceptron	90/10	
7	Pokerhand	IBk	90/10	
8	Segmentation	IBk	90/10	
9	Letter-recognition	MultilayerPerceptron	90/10	
10	Data_banknote_authentication	MultilayerPerceptron	90/10	

Окончание табл. 5.1

1	2	3	4	5
11	Agaricus-lepiota	NaiveBayes	90/10	
12	Semeion	IBk	90/10	*k* = 10, класс – цифра «8»
13	Flag	NaiveBayes	90/10	
14	Adult	NaiveBayes	90/10	
15	Iris	IBk	80/20	*k* = 10
16	Winequality-white	MultilayerPerceptron	80/20	
17	Winequality-red	IBk	80/20	*k* = 1
18	Glass	MultilayerPerceptron	80/20	
19	Movement_libras	IBk	80/20	
20	Spambase	IBk	80/20	
21	Pokerhand	MultilayerPerceptron	80/20	
22	Segmentation	MultilayerPerceptron	80/20	
23	Letter-recognition	NaiveBayes	80/20	
24	Data_banknote_authentication	NaiveBayes	80/20	
25	Agaricus-lepiota	IBk	80/20	*k*= 3
26	Semeion	IBk	80/20	*k* = 10, класс – цифра «5»
27	Flag	IBk	80/20	*k*= 10
28	Adult	IBk	80/20	*k*= 5

Контрольные вопросы

1 Основные аспекты метода классификации, использованного в ходе лабораторной работы.
2 Способы тестирования эффективности классификаторов в ПО Weka.
3 Что отражает отчет о классификации в ПО Weka?

Лабораторная работа № 6
Интеллектуальный анализ данных с помощью ПО Weka.
Кластеризация данных

Цель

Изучить функционал модуля Explorer ПО Weka, вкладка Cluster.

Задачи

1 Изучить интерфейс вкладки Cluster ПО Weka.
2 Произвести кластеризацию данных с помощью заданного вариантом метода.
3 Сохранить и проанализировать результаты классификации данных.
4 Оформить индивидуальный отчет.

Содержание индивидуального отчета

1 Краткий конспект основных теоретических положений работы.
2 Скриншоты основных действий в ПО Weka и их описание.
3 Выводы и ответы на контрольные вопросы.

Теоретические сведения

Вкладка «Cluster» в ПО «Weka» (рис. 6.1) позволяет осуществлять разбиение данных на группы, используя различные методы: *k*-средних, EM, дальнего соседа и др.

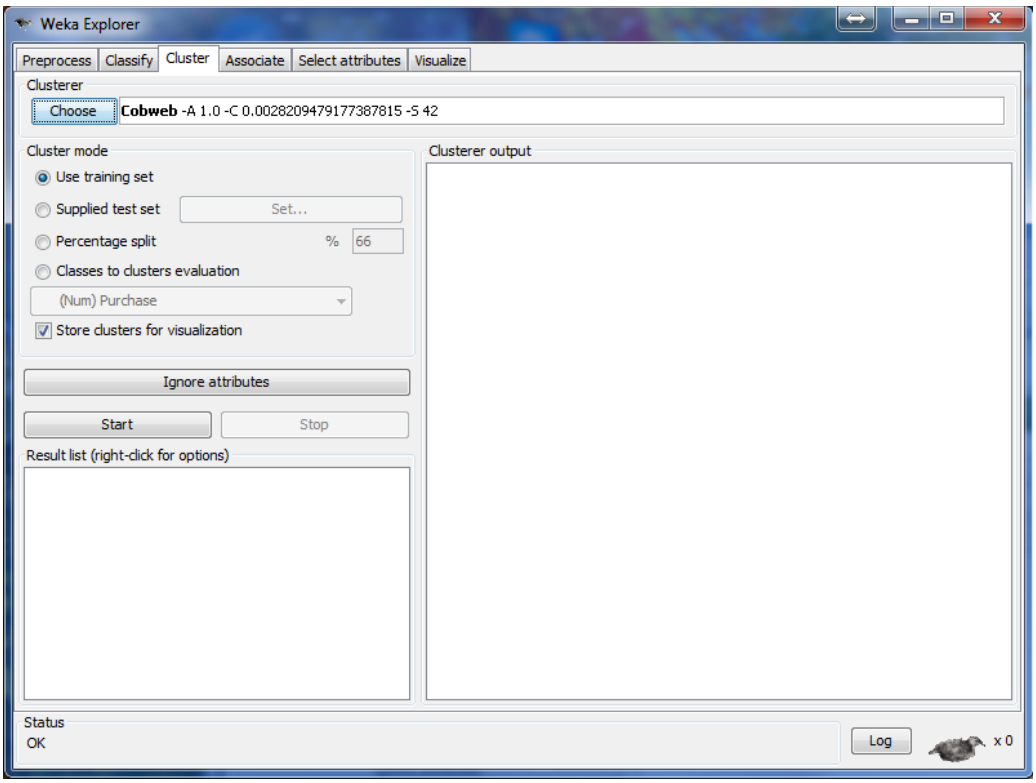

Рис. 6.1. Вкладка «Cluster»

Интерфейс вкладки Cluster похож на интерфейс вкладки Classify (Classifier – Clusterer, Test options – Cluster mode и др.), а также имеет свои специфические возможности:

– Использование классов для оценки эффективности кластеризации (Classes to clusters evaluation);

– Сохранение кластеров для их последующей визуализации (Store clusters for visualization);

– Выбор игнорируемых признаков при кластеризации.

После выполнения кластеризации на панели Clusterer output выводится статистика результатов выполнения кластеризации. В конце статистики всегда приводятся результирующие множества, на которые будет разделена выборка (рис. 6.2).

Рис. 6.2. Статистика результата применения кластеризации

Для наглядной проверки эффективности кластеризации в ПО Weka есть возможность визуализации полученных кластеров. Для этого необходимо кликнуть правой кнопкой мыши на названии используемого метода на панели Result list и выбрать пункт Visualize cluster assignments.

Лабораторное задание

Привести файл с выборкой из табл. 6.1 к формату, распознаваемому в ПО Weka. Выполнить предобработку данных, если необходимо. Выполнить кластеризацию методом *k*-среднего (KMeans) согласно табл. 6.1.

Таблица 6.1 – Задание для выполнения лабораторной работы № 6

№ вар.	Название выборки	Расстояние
1	2	3
1	Iris	Евклидово
2	Winequality-white	Евклидово
3	Winequality-red	Евклидово
4	Glass	Евклидово
5	Movement_libras	Евклидово
6	Spambase	Евклидово
7	Pokerhand	Евклидово
8	Segmentation	Евклидово
9	Letter-recognition	Евклидово
10	Data_banknote_authentication	Евклидово

Окончание табл. 6.1

1	2	3
11	Agaricus-lepiota	Евклидово
12	Semeion	Евклидово
13	Flag	Евклидово
14	Adult	Евклидово
15	Iris	Манхэттенское
16	Winequality-white	Манхэттенское
17	Winequality-red	Манхэттенское
18	Glass	Манхэттенское
19	Movement_libras	Манхэттенское
20	Spambase	Манхэттенское
21	Pokerhand	Манхэттенское
22	Segmentation	Манхэттенское
23	Letter-recognition	Манхэттенское
24	Data_banknote_authentication	Манхэттенское
25	Agaricus-lepiota	Манхэттенское
26	Semeion	Манхэттенское
27	Flag	Манхэттенское
28	Adult	Манхэттенское

Контрольные вопросы

1 Основные аспекты метода *k*-средних.
2 Каковы возможности вкладки Cluster ПО Weka.
3 Что отражает отчет о кластеризации в ПО Weka и для чего нужна визуализация результатов?

Учебное издание

Суханов Андрей Валерьевич

Лященко Зоя Владимировна

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ
И СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Печатается в авторской редакции
Технический редактор Т.И. Исаева

Подписано в печать 05.10.17. Формат 60×84/16.
Бумага газетная. Ризография. Усл. печ. л. 2,2.
Тираж экз. Изд. № 90401. Заказ .

Редакционно-издательский центр ФГБОУ ВО РГУПС.
Адрес университета: 344038, г. Ростов н/Д, пл. Ростовского Стрелкового Полка Народного Ополчения, д. 2.