БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ РАДИОФИЗИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ
Кафедра системного анализа и компьютерного моделирования

Н. Н. Яцков, Е. В. Лисица

ИНТЕЛЛЕКТУАЛЬНЫЙ АНАЛИЗ ДАННЫХ

Методические указания
к лабораторным работам
для студентов специальностей
1-31 04 02 «Радиофизика»,
1-31 04 03 «Физическая электроника»,
1-98 01 01 «Компьютерная безопасность»

МИНСК
2019

УДК 604.8 (076.5)
ББК 32.813
Я93

Рекомендовано советом
факультета радиофизики и компьютерных технологий
26 декабря 2018 г., протокол № 5

Рецензент
кандидат технических наук, доцент *В. И. Микулович*

Яцков, Н. Н.
Я93 Интеллектуальный анализ данных : метод. указания к лабораторным работам / Н. Н. Яцков, Е. В. Лисица. – Минск : БГУ, 2019. – 51 с.

Методические указания предназначены для проведения лабораторного практикума по курсу «Интеллектуальный анализ данных» для студентов факультета радиофизики и компьютерных технологий. Содержится учебный материал по методам обработки и анализа многомерных наборов данных.

УДК 604.8 (076.5)
ББК 32.813

ВВЕДЕНИЕ

Данное издание представляет собой руководство к лабораторному практикуму по курсу «Интеллектуальный анализ данных». Лабораторные работы выполняются в интегрированной вычислительной среде MATLAB. Цель практикума – научить студента применять методы интеллектуального анализа данных для решения ряда прикладных задач обработки и анализа многомерных наборов данных на примере использования среды MATLAB. Тематика практикума охватывает алгоритмы нелинейного метода наименьших квадратов, метода главных компонент, методов стохастического поиска экстремумов функции, а также применение нейронных сетей для кластеризации и классификации данных. Приведенные справочные теоретические сведения содержат достаточный объем информации, исключающий необходимость прибегать к другим источникам литературы в ходе выполнения практических работ.

Прилагается компакт-диск с файлами заданий и приложений к лабораторному практикуму.

ЛАБОРАТОРНАЯ РАБОТА 1

ОСНОВЫ РАБОТЫ В MATLAB

Цель работы. Изучение элементов языка, основных функций и среды MATLAB. Решение систем линейных уравнений. Создание пользовательских функций и построение их графиков в среде MATLAB. Изучение генератора базовой случайной величины.

Краткие сведения. Система MATLAB (MATrix LABoratoty) предназначена для выполнения научных и инженерных расчетов на компьютере. С ее помощью эффективно решаются задачи вычислительной математики, линейной алгебры, математической статистики и математического моделирования (символьная математика, уравнения в частных производных, численное интегрирование, аппроксимация, решение оптимизационных задач). В состав MATLAB входит несколько десятков специализированных пакетов (Toolbox), предназначенных для обработки данных в различных областях науки и техники (системы управления, нечеткие системы, нейронные сети, цифровая обработка сигналов и изображений, анализ временных рядов).

MATLAB – это одновременно и операционная среда и язык программирования. Язык интерпретатора команд MATLAB прост и легко поддается изучению. Из командной строки можно вызвать отдельную команду или программу (последовательность команд), оформленную в виде m-файла.

1. Загрузка системы MATLAB и работа в главном окне

Загрузить MATLAB: кнопка **Пуск** / **Программы** / **MATLAB**. Создать на доступном диске собственную папку: *имя папки набирайте латинскими буквами*. Текущей папкой среды MATLAB назначьте свою созданную папку (рис. 1.1).

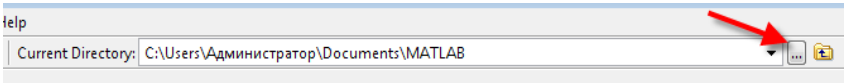

Рис. 1.1. Окно текущей папки среды MATLAB

Вычисления можно производить прямо в командном окне. Для этого в главном окне после символа >> наберите любые арифметические действия. Результат будет сохранен в автоматически созданной переменной *ans*.

При выполнении большого количества вычислений (например, при реализации некоторого алгоритма) работать в подобном режиме не всегда удобно. В таких случаях следует оформить вычисления в виде m-файлов.

Задание 1. Выполните произвольные вычисления в командном окне, попробуйте использовать различные арифметические операции и задайте с помощью круглых скобок приоритет их выполнения.

2. Создание m-файла

М-файлы, как правило, содержат последовательность команд. Фактически в m-файле содержится код пользовательской программы, который удобно редактировать и дополнять.

Для создания m-файла следует зайти в меню **File\ New\ M-file**. В результате открывается текстовый редактор MATLAB, предназначенный для создания и редактирования подобных файлов.

Теперь следует сохранить m-файл: в меню редактора **File\ Save as…**\, в диалоговом окне задать **имя файла\ OK**.

В созданном m-файле можно запрограммировать любые арифметические вычисления. Они будут сохранены в этом файле, и их можно выполнить на другом компьютере, на котором установлен MATLAB.

Полезно включать в m-файл функцию *path*, которая задаёт путь поиска функций и файлов, к которым происходит обращение.

Например, в текущей папке содержится папка Functions, в которой находится файл *myfunc.m*. Чтобы вызвать функцию *myfunc* необходимо указать путь для ее поиска: path('Functions/',path)

ВНИМАНИЕ! Имя файла не должно начинаться с цифры, содержать пробелы. Для задания имени файла следует использовать только латинские буквы.

Задание 2. Создайте m-файл с произвольным именем. Сохраните его в своей рабочей папке.

3. Использование справочной системы

MATLAB содержит большое число встроенных математических функций. Для того, чтобы пользоваться встроенной функцией необходимо знать, какие параметры и в какой последовательности следует в неё передавать. Справку о встроенной функции можно получить набрав в командном окне команду *help* и далее имя используемой функции (например, *sin*): *help sin*. Вызов расширенной справки производится из главного меню MATLAB: **Help\ Product help** (или нажать клавишу **F1**). Поиск нужной функции осуществляется в строке поиска по некоторому ключевому слову.

Задание 3. Получите справку по следующим операциям и функциям: ^, *asin, inv, plot.*

4. Вспомогательные функции

Очистка экрана с помощью *clc*. С помощью команды *clc* можно очистить видимое содержимое командного окна системы MATLAB. Для этого в главном окне необходимо просто набрать *clc* и нажать клавишу **Ввод (Enter)** – экран очистится от всех введенных записей.

Очистка рабочего пространства. Рабочее пространство системы MATLAB (Workspace) – специально зарезервированная область оперативной памяти, в которой хранятся значения переменных, вычисляемых в рамках текущего сеанса.

С помощью команды *clear all* произведите очитку рабочего пространства системы MATLAB – сотрите из оперативной памяти все вычисленные значения переменных.

Задание 4. Произведите очистку экрана и рабочего пространства.

5. Работа с векторами и матрицами

Объявление векторов и матриц. Объявление векторов и матриц производится следующим образом:

```
V=[1 2 3 4 5];                    % Создание вектора
M=[1 2 3 4 5; 6 7 8 9 10; 11 12 13 14 15]; % Создание матрицы
```

Если после объявления вектора (или матрицы) поставить точку с запятой (;), то результат **V** (или **M**) будет вычислен, но значения переменных выводится в командное окно не будут.

Доступ к отдельным элементам вектора и матрицы. Доступ к отдельному элементу вектора: `V(3)` – третий элемент вектора **V**.
Доступ к отдельному элементу матрицы: `M(2,3)` – элемент матрицы из второй строки и третьего столбца матрицы **M**.
Доступ ко всему третьему столбцу матрицы **M**: `M(:,3)`.
Доступ ко всей второй строке матрицы **M**: M(2,:).

Добавление и удаление столбцов матрицы. Добавление **строки** к матрице:

```
M=[1 2 3 4 5; 1 2 3 4 5; 1 2 3 4 5];
S=[1 2 3 4 5];
M=[M;S];
```

Добавление **столбца** к матрице (используется запятая вместо точки с запятой в последнем операторе):

```
M=[1 2 3 4 5; 1 2 3 4 5; 1 2 3 4 5];
S=[1; 2; 3];
M=[M,S];
```

Удаление всего третьего столбца матрицы **M**: `M(:,3)=[ ]`.
Удаление всей второй строки матрицы **M**: `M(2,:)=[ ]`.

6. Нахождение максимального и минимального элементов матрицы

Пусть дана матрица **C**. С помощью функций *min*, *max* можно находить минимальные и максимальные элементы по строкам, по столбцам и по всей матрице:

min(C,[ ],1) – вектор минимальных элементов, найденных по столбцу,

min(C,[ ],2) – вектор минимальных элементов, найденных по строке (аналогично для функции *max*).

Для поиска минимального и максимального элемента по всей матрице соответствующая функция (*min* или *max*) применяется два раза (*min*(*min*(…))).

Задание 5. Введите в **test_matrix.m** описанные выше процедуры объявления вектора и матрицы, процедуры доступа к элементам вектора и матрицы, поиск минимального и максимального элемента в матрице. Просмотрите результат в командном окне.

Добавьте в **test_matrix.m** код, добавляющий и удаляющий вектор к матрице.

Задание 6. Объявите матрицу **C** размерности 5 строк на 5 столбцов с произвольными элементами. С помощью функций *min*, *max* найдите минимальные и максимальные элементы по строкам, по столбцам и по всей матрице.

Добавьте соответствующий код в файл **test_matrix.m**.

7. Многомерные массивы

Многомерные массивы – массивы с размерностью больше двух (рис. 1.2). Для доступа к элементу такого массива необходимо задавать три и более индекса.

Для объявления, например, трехмерного массива состоящего из нулей следует вызвать функцию *zeros*():

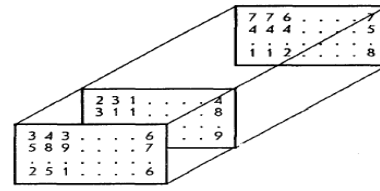

Рис. 1.2. Схематическое представление трехмерного массива

```
A = zeros(2,3,4);
```

В результате получится трехмерный массив (пространственная матрица), состоящий из двумерных матриц.

Доступ к элементу осуществляется путем указания индексов для трехмерного массива: `A(номер строки, номер столбца, номер матрицы): A(1,2,1).`

Размер массива. Размер вектора, матрицы или многомерного массива можно узнать, вызвав функцию *size*():

```
size(A);
```

Транспонирование вектора. Транспонирование вектора и матрицы осуществляется помощью операции ':

```
b=[1 2 3 4];
s=b';
```

8. Арифметические операции с векторами и матрицами

По умолчанию все действия (+, –, *, /, ^ - возведение в степень) выполняются над матрицами по правилам линейной алгебры.

```
a=[2 3 4 5];
b=[1 2 3 4];
c=a+b;
```

Для поэлементного перемножения двух векторов необходимо использовать оператор точка:

```
d=a.*b;
```

В результате перемножения вектора на вектор по правилам линейной алгебры получаем число (вектор b при этом надо транспонировать):

```
a=[2 3 4 5];

b=[1 2 3 4];
c=a*(b');
```

Следует помнить, что для выполнения действий над элементами матриц, перед знаками *, / , ^ необходимо дополнительно поставить точку.

9. Массивы строковых переменных

Создание строковой переменной производится с использованием одинарных кавычек

```
Distance = 'euclidean'
```

Создание массива строковых переменных производится с помощью массива ячеек (cell array)

```
Distances = {'euclidean','cityblock','minkowski'}
```

Задание 7. Получить решение **x** заданной системы линейных уравнений вида **A* x** = **B**, где **A** – квадратная матрица $n \times n$, **B** – вектор размерности n. Решение данной системы уравнений можно получить, набрав строку кода

```
x =inv(A)*B,
```

причем предварительно следует задать значения матрицы **A** и вектора **B**, функция *inv*() – вычисляет обратную матрицу для матрицы A.

Для случая **A**=[2 1; 3 4] и **B**=[4; 11] решите уравнение **A* x** = **B**. Обратите внимание, что вектор **B** – состоит из одного столбца (используется разделитель столбцов матрицы точка с запятой (;)).

Проверьте найденное решение, подставив его в исходное уравнение.

Добавьте в **test_matrix.m** соответствующий код.

10. Работа с графикой

Пример программы, реализующей построение графика функции s*in*.

```
x=0:0.1:6.28;    % Объявление вектора аргумента 0<x<6.28 с шагом 0.1
y=sin(x);        % Вычисление значений функции в заданных точках
plot(x,y)        % Построение графика функции
grid on;         % Отображение сетки
title ('Function  sin(x)')  % Заголовок графика
xlabel('Argument x')        % Подпись по оси x
ylabel('Function y')        % Подпись по оси y
```

11. Сохранение фигуры с графиком в заданном графическом формате (tiff)

Для сохранения текущего окна с графиком в заданном графическом формате необходимо, не закрывая это окно, набрать в командном окне следующий код (его можно также вставлять в код m-файла)

```
print -dtiff -r300 Graphic
```

где *–dtiff* указывает формат файла (tiff), *-r300* – разрешение файла (300 dpi), *Graphic* – имя файла с рисунком (более подробно о функции *print* смотрите в справке MATLAB).

Задание 8. Создайте файл **test_graphic.m**, в котором наберите приведенный код п. 10. Сохраните полученный график в формате tiff с разрешением 200 dpi (имя файла произвольное).

12. Построение трехмерных графиков

Рассмотрим построение трехмерных графиков функций на примере функции Розенброка. Создайте файл **vrosenbrock.m** :

```
function z=vrosenbrock(x,y)
z=100*(y - x.^2).^2 + (1-x).^2;
```

В отдельном m-файле введите код, вызывающий функцию Розенброка:

```
Lx=-5; % Левая граница для x
Rx=5;  % Правая граница для x
stepx=0.05; % Шаг по оси x

Ly=-5; % Левая граница для y
Ry=5; % Правая граница для y
stepy=0.05; % Шаг по оси y

% Создание сетки координат
xs=Lx:stepx:Rx;
ys=Ly:stepy:Ry;

% Вычисление данных для трехмерного графика
[X,Y] = meshgrid(xs,ys);
Z = vrosenbrock(X,Y);

% Построение трехмерного графика график
 surfc(xs,ys,Z)
```

Задание 9. Создайте файл **test_3Dgraphic.m**, в котором наберите приведенный выше код п. 12.

13. Создание пользовательских функций

Рассмотрим задачу реализации функции, возвращающей значения двумерной случайной величины $\xi=(X,Y)$, равномерно распределённой на плоскости с заданными границами (границы области изменения должны задаваться в программе).

Требуется построить:

а) график, отражающий значения реализации случайной величины ξ на плоскости в виде точек;

б) гистограммы распределения компонент двумерной случайной величины.

Создание функции. ВНИМАНИЕ! Имя файла, содержащего объявление функции, должно совпадать с именем функции.

Введите в новый m-файл следующий код функции (имя файла совпадает с именем функции), а затем сохраните файл:

```
function [X,Y]=my_func(X_left, X_right, Y_left, Y_right, N)

% Генерация N значений случайной величины X в области задания X
X=X_left + rand(N,1)*(X_right - X_left);

% Генерация N значений случайной величины Y в области задания Y
Y=Y_left + rand(N,1)*(Y_right - Y_left);
```

Функция *rand*() – генератор значений случайной величины, равномерно распределённой на интервале [0;1]. Вызов *rand*(*N*,1) – возвращает вектор из одного столбца и *N* строк. Результат функции *my_func* возвращается в векторах *X* и *Y*.

Задание 10. Создайте новый m-файл и сохраните его под именем **my_func**. В файле поместите размещенное выше описание функции **my_func.**

Вызов собственной функции. Создайте файл **test_my_func.m**, в котором введите следующий код:

```
% Задание диапазона изменения X
 X_left=-2;
 X_right=2;

% Задание диапазона изменения Y
 Y_left=-3;
 Y_right=3;

% Задание количества сгенерированных точек
 N=1000;
% Вызов функции
[X,Y]=my_func(X_left, X_right, Y_left, Y_right, N);

% Построение графика функции
plot(X,Y,'*')
```

Вычисление гистограммы с помощью *histc*. В файле **test_my_func.m** добавьте следующий код:

```
% Инициализация гистограммы


 BinNumber=20;
 k=0:BinNumber;

% Вычисление границ карманов на оси X

 X_bins=X_left + k*(X_right - X_left)/BinNumber;
% Вычисление границ карманов на оси Y
```

```
 Y_bins=Y_left + k*(Y_right - Y_left)/BinNumber;
% Вычисление гистограммы для X

  N_X = histc(X,X_bins);
% Вычисление гистограммы для Y

  N_Y = histc(Y,Y_bins);
```

Отображение гистограммы с помощью функции *bar*. В файле **test_my_func.m** добавьте следующий код:

```
figure;
bar(X_bins, N_X);

figure;
bar(Y_bins, N_Y);
```

Задание 11. Создайте файл **test_my_func.m**, в котором наберите код, демонстрирующий работу функции **my_func**. Запустите программу и просмотрите результат ее работы.

Создайте заголовки гистограмм и подписи по осям с помощью функций *title*, *xlabel*, *ylabel*, добавив их после функции *bar*.

Задание 12. Создайте функцию **my_gauss_gen**, которая генерирует заданное количество случайных величин ***N***, имеющих гауссовское распределение с заданным математическим ожиданием ***m*** и дисперсией ***D***. Постройте гистограмму (рис. 1.3).

Для этого воспользуйтесь встроенной функцией *randn*(), которая позволяет получать реализацию случайной величины α с гауссовским распределением с параметрами ***m***=0 и ***D***=1. Реализация β требуемой случайной величины может быть получена по правилу:

$$\beta = m + \alpha\sqrt{D}.$$

Код, реализующий созданную функцию (построение гистограммы), сохраняете в новом файле **test_my_gauss_gen.m**.

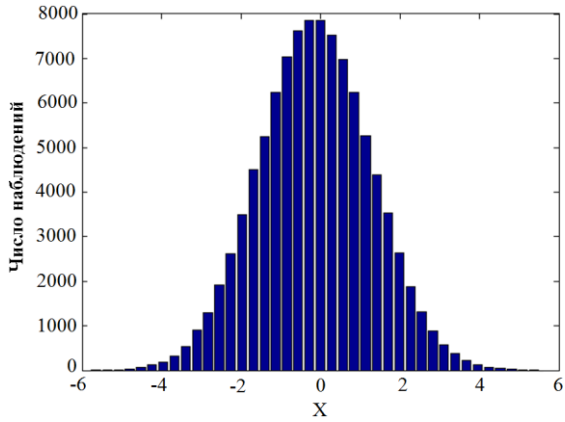

Рис. 1.3. Результат работы функции **my_gauss_gen** с параметрами ***m***=0 и ***D***=2, N=100000, количество каналов гистограммы равно 40

14. Циклы

Формирование матрицы с помощью цикла *for* . Пример:

```
n=5;  % Количество строк
m=10; % Количество столбцов
% Формируем нулевую матрицу Q размером n x m
Q = zeros(n,m);
% В цикле заполняем матрицу Q новыми значениями
for k = 1:n
    for j = 1:m
        Q(k,j) = round(10*rand);
    end
end
```

Цикл *while*. Пример:

```
k=0;
while k < 20
  disp(['Итерация ',num2str(k)]);
  k=k+1;
end
```

Запустите программу на выполнение (клавиша **F5**). Просмотрите результаты выполнения программы в главном окне MATLAB.

Задание 13. Создайте файл **test_cycle.m**, в котором сформируйте матрицу **Q** произвольного размера и заполните ее случайными целыми числами. Выведите в главном окне MATLAB (Command Window) матрицу **Q** (добавьте в код команду *disp*(**Q**)).

Задание 14. В файле **test_cycle.m** реализуйте цикл, позволяющий вычислять сумму элементов матрицы **Q**. Весь необходимый код напишите в файле **test_cycle.m**.

15. Работа с текстовыми файлами

Запись в файл с помощью функции *fprintf*. Пример:

```
x = 0 : 0.1 : 5;
y = 2*x.^2 + x - 1;
M = [x; y];
% Формирование заголовка данных
 str='Значения функции y = 2*x^2 + x - 1';
% Открытие файла с идентификатором fid для записи
 fid = fopen('MyFile.txt','wt');
% Запись заголовок данных
 fprintf(fid,'%s\n',str);
% Запись данных (матрица M) в файл в формате действительных чисел
 fprintf(fid,'%6.2f  %12.8f\n',M);
% Закрытие файла с идентификатором fid
 fclose(fid);
```

Чтение данных из файла с помощью функции *fscanf*. Пример:

```
% Открытие файла для чтения
 fid = fopen('MyFile.txt','r');
% Чтение первой строки файла
 S = fgetl(fid);
% Чтение данных в матрицу A, имеющую две строки
 A = fscanf(fid, '%g %g', [2 inf]);
% Транспонирование матрицы с целью представления данных в виде двух
колонок
 A = A';
% Закрытие файла с идентификатором fid
 fclose(fid);
% Графическое отображение данных
 plot(A(:,1),A(:,2),'-*');
  title ('Function y = 2*x^2 + x - 1')  % Заголовок графика
 xlabel('Argument x')                  % Подпись по оси x
 ylabel('Function y')                  % Подпись по оси y
```

Задание 15. Создайте новый m-файл с именем **write_txt**, в котором наберите код записи данных с помощью функции *fprintf*. Запустите код на выполнение (клавиша **F5**). Откройте созданный текстовый файл и убедитесь, что запись прошла корректно (в файле есть данные).

Задание 16. Создайте новый m-файл с именем **read_txt**, в котором наберите код считывания из файла с помощью *fscanf*. Запустите программу на выполнение (клавиша **F5**). Выведите на экран матрицу данных из файла.

Форма отчёта: работающие программные модули, реализующие задания, листинг программ.

Контрольные вопросы

1. Каким образом производятся вычисления в командном окне MATLAB?
2. Как получить информацию о требуемой функции MATLAB?
3. Можно ли в имени m-файла использовать буквы кириллицы, устанавливать пробелы, начинать имя файла с цифры?
4. Как образом удалить из оперативной памяти переменные, зарезервированные во время рабочей сессии?
5. С помощью какой команды MATLAB производится очистка рабочего окна?
6. Как объявить вектор, матрицу? Выведите в рабочем окне элемент вектора, матрицы.
7. Как добавить заданный столбец/строку к матрице?
8. Что произойдет в результате применения операции **b'**, где **b** – заданный вектор.
9. Как определить размер матрицы?
10. С помощью какой функции MATLAB производится построение двумерного графика функции? Трехмерного графика функции?
11. Может ли имя функции отличаться от имени файла, в котором находится ее описание?
12. С помощью каких функций MATLAB производится запись/чтение в/из файл(а)?

ЛАБОРАТОРНАЯ РАБОТА 2
НЕЛИНЕЙНЫЙ МЕТОД НАИМЕНЬШИХ КВАДРАТОВ

Цель работы: практическое освоение анализа экспериментальных данных с помощью нелинейного метода наименьших квадратов.

КРАТКИЕ СВЕДЕНИЯ

В экспериментальной практике часто измеряют некоторую характеристику, которую можно выразить зависимостью $y(x)$. При этом получают набор точек (x_k, y_k), где $k = 1,2\ldots N$, N – количество точек в наборе данных.

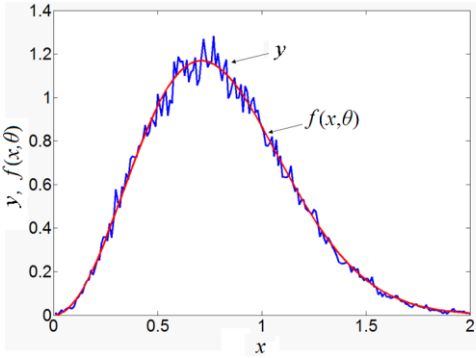

Рис 2.1. Вид измеренной характеристики набора данных $y(x)$ и аппроксимирующей ее функции $f(x, \boldsymbol{\theta})$

Основной целью анализа данных $y(x)$ является подбор некоторой функции или модели $f(x, \boldsymbol{\theta})$, где $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_p)$ – вектор параметров модели, значения которой наилучшим образом совпадают со значениями измеренной величины y_k. Функцию $f(x, \boldsymbol{\theta})$ называют (не)линейной регрессионной моделью (рис. 2.1).

В большинстве случаев подбор включает следующие шаги: 1) выбор вида функции $f(x, \boldsymbol{\theta})$; 2) инициализация вектора параметров $\boldsymbol{\theta}$ начальными приближениями; 3) изменение значений параметров $\boldsymbol{\theta}$ до тех пор, пока не будет достигнуто наилучшее совпадение между $f(x, \boldsymbol{\theta})$ и $y(x)$.

Блок-схема процедуры анализа экспериментальных данных представлена на рис. 2.2. В блоке 1 задаются начальные приближения для параметров функции $f(x, \boldsymbol{\theta})$. Степень совпадения данных $y(x)$ и теоретической модели $f(x, \boldsymbol{\theta})$ оценивается путем их сравнения по заданному критерию (блок 4). Подбор параметров функции $f(x, \boldsymbol{\theta})$ производится с помощью процедуры оптимизации (блок 5). Для найденных оценок параметров $\overline{\boldsymbol{\theta}}$ строятся доверительные интервалы (блок 8). Визуальная оценка качества аппроксимации

(блок 7) заключается в отображении на одном графике значений экспериментальных и теоретических данных (пример, рис. 2.1.), построении графика взвешенных остатков и их автокорреляционной функции (пример, рис 2.3).

Рис 2.2. Блок-схема процедуры анализа экспериментальных данных

Для задания начальных значений параметров **θ** используются специальные алгоритмы, учитывающие свойства функции $f(x, \boldsymbol{\theta})$, либо начальные значения **θ** задаются исходя из условий задачи.

Метод аппроксимации экспериментальных данных, в котором в качестве критерия согласия выступает минимум суммы квадратов отклонений значений измеренной характеристики от значений аппроксимирующей функции, известен как ***нелинейный метод наименьших квадратов*** (НМНК).

В НМНК предполагается[1], что значения измеренной характеристики y_k подчиняются нормальному распределению с математическим ожиданием y_k и среднеквадратическим отклонением σ_k. В этом случае, в качестве минимизируемой целевой функции выбирают выражение

$$\chi^2(\boldsymbol{\theta}) = \sum_{k=1}^{N} \left(\frac{f(x_k,\boldsymbol{\theta}) - y_k}{\sigma_k}\right)^2. \qquad (2.1)$$

Выражение (2.1) выступает в качестве критерия согласия: его минимальное значение соответствует наилучшему совпадению экспериментальных данных и теоретической модели. Часто величину χ^2 нормируют на число степеней свободы ν (приведенное χ^2_ν):

$$\chi^2_\nu = \frac{\chi^2}{\nu}, \qquad (2.2)$$

где $\upsilon = N\text{-}p\text{-}1$ – число степеней свободы, p – число оцениваемых параметров (количество элементов в векторе $\boldsymbol{\theta}$).

При согласии теоретической и исходной характеристик $\chi^2_\nu \to 1$.

В большинстве случаев значения σ_k, которые требуются для расчёта по формуле (2.1), неизвестны и для их определения следует использовать либо различные приближения, полученные для конкретного вида характеристики y, либо проводить серию измерений с их последующей статистической обработкой. Например, для гистограмм (если y_k – частота события) обычно выбирают $\sigma_k = \sqrt{y_k}$.

Для поиска минимума целевой функции (2.1) применяются различные методы оптимизации: метод Хука – Дживса, метод Пауэла, метод Розенброка, метод Марквардта – Левенберга, метод Гаусса – Ньютона, метод наискорейшего спуска.

Для установления точности найденных оценок параметров рассчитываются *доверительные интервалы* – интервал значений оцениваемого параметра, в котором с определённой вероятностью β находится истинное значение параметра. В случае, когда оценка параметра $\bar{\theta}$ имеет нормальное распределение, для построения оценки β-процентного доверительного интервала можно воспользоваться выражением:

$$\bar{\theta} + t_{\alpha/2,\nu}\sqrt{C_{jj}} \le \theta \le \bar{\theta} + t_{1-\alpha/2,\nu}\sqrt{C_{jj}}, \qquad (2.3)$$

где $t_{\alpha/2,\nu}$ – квантиль порядка $\alpha/2$ распределения Стьюдента[2] с $\nu = N\text{-}1$ степенями свободы, соответствующий доверительной вероятности $\beta = (1\text{-}\alpha)$, значения которой обычно задаются в диапазоне 0.95,…,0.99, C_{jj} – диагональный элемент ковариационной матрицы (является оценкой погрешности параметра $\bar{\theta}$).

---

[1] Предположение обусловлено тем, что на измеряемую величину влияет множество случайных факторов и степень их влияния примерно одинакова.

[2] Например, при $\beta = 0.95$ ($\alpha = 0.05$) и $\nu = 20$ квантиль распределения Стьюдента $t_{\alpha/2,\nu} \approx -2{,}09$.

Для оценки качества проведённого анализа часто используют дополнительные критерии, к которым относится график взвешенных остатков, позволяющий визуально оценивать степень совпадения измеряемых данных и теоретической модели. В случае, когда точки характеристики имеют нормальное распределение, взвешенные остатки можно рассчитать по формуле

$$R_k = \frac{y_k - f(x_k,\boldsymbol{\theta})}{\sigma_k},\ k = 1, 2, \ldots, N, \qquad (2.4)$$

где y_k, $f(x_k,\boldsymbol{\theta})$ – значения построенной и теоретической характеристик соответственно, σ_k – среднеквадратическое отклонение в оценке k-й точки характеристики. Если экспериментальная и теоретическая характеристики согласуются, то взвешенные остатки распределены нормально возле нулевого значения.

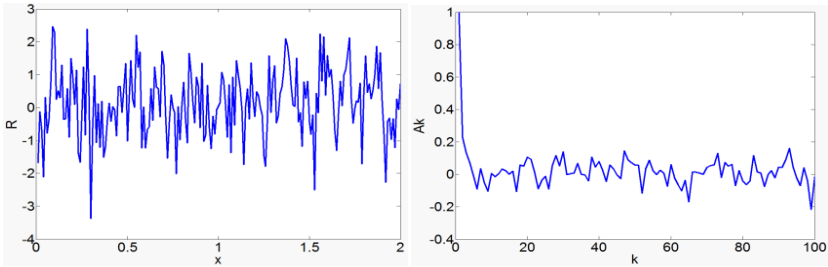

Рис. 2.3. Графики взвешенных остатков (слева) и автокорреляционной функции взвешенных остатков (справа) в случае согласия измеряемых данных и теоретической модели

Если между экспериментальной и теоретической характеристиками имеются незначительные систематические отклонения, неразличимые по графику взвешенных остатков, то для их нахождения применяется *автокорреляционная функция взвешенных остатков*, определяемая как

$$A_k = \frac{\frac{1}{N-k+1}\sum_{i=1}^{N-k+1} R_i R_{i+k-1}}{\frac{1}{N}\sum_{i=1}^{N} R_i^2}. \qquad (2.5)$$

Обычно выбирают $1 \le k \le N/2$. Величина $A_1 = 1$, означает, что каждое значение взвешенных остатков полностью коррелирует с самим собой. Если отклонения между экспериментальными и теоретическими данными случайны, то при хорошем совпадении измеряемых данных и теоретической модели автокорреляционная функция взвешенных остатков осциллирует с малой амплитудой около нуля.

Порядок выполнения

1. Изучите функции MATLAB: *fopen, load, path, fscanf, nlinfit, nlparci, tinv, diag*.

2. Реализуйте загрузку файла с данными с помощью встроенных функций *fopen, path, fscanf (load)*. Файл данных (data*.txt) содержит набор экспериментальных точек и оценок ошибок (x_k, y_k, σ_k), где k = 1, 2,…, N. Постройте график экспериментальных данных $y(x)$. График должен иметь заголовок и подписи осей.

3. Согласно вашему варианту запрограммируйте три функции $f(x, \boldsymbol{\theta})$.

4. Реализуйте процедуру НМНК анализа данных с помощью трех заданных функций. Начальные приближения для параметров функций выбираются произвольно. Для процедуры оптимизации воспользуйтесь функцией Matlab *nlinfit*.

5. Оцените качество анализа данных с помощью нормированного критерия χ_v^2, графика взвешенных остатков и их автокорреляционной функции. Графики должны иметь заголовки и подписи осей. Укажите функцию, которая аппроксимирует набор данных $y(x)$ наилучшим образом. Отобразите на одном графике измеренную характеристику $y(x)$ и аппроксимирующую функцию. График должен иметь заголовок и подписи осей.

6. Постройте 68% доверительные интервалы для оценок параметров. Доверительные интервалы вычисляются по формуле (2.3) и с помощью функции *nlparci*. Сравните полученные результаты. Используйте функции *tinv, diag*.

Форма отчёта: работающий программный модуль, реализующий задание, листинг программ. Выводы по результатам обработки экспериментальных данных.

Таблица 2.1

Варианты заданий

Вариант	**Закон распределения**	**Данные***	**Вариант**	**Закон распределения**	**Данные***
1	3, 5, 6	data1.txt	7	4, 7, 8	data7.txt
2	1, 3, 9	data2.txt	8	2, 4, 6	data8.txt
3	1, 3, 8	data3.txt	9	1, 6, 8	data9.txt
4	3, 4, 7	data4.txt	10	6, 7, 9	data10.txt
5	4, 6, 9	data5.txt	11	2, 6, 8	data11.txt
6	2, 4, 8	data6.txt	12	1, 8, 9	data12.txt

* Файл данных состоит из трех колонок, в которых, соответственно, записаны значения x_k, y_k, σ_k.

Таблица 2.2

Законы плотности распределения случайных величин

N п/п	**Закон распределения**	**Область задания и параметры**	**Плотность распределения *f(x)***
1	Нормальный	$x \in (-\infty;\infty)$, $m \in R$, $\sigma > 0$	$\frac{1}{\sigma\sqrt{2\pi}} exp\left[-\frac{(x-m)^2}{2\sigma^2}\right]$
2	Бета	$x \in (0;1)$, a>0, b>0	$\frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1}$, $B(a,b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$
3	Релея	$x \in (0;\infty)$, $\sigma \in R$	$\frac{x}{\sigma^2} exp\left[-\frac{x^2}{2\sigma^2}\right]$
4	Вейбулла	$x \in (0;\infty)$, a>0, b>0	$abx^{b-1}exp(-ax^b)$
5	Лапласа	$x \in (-\infty;\infty)$, $b \in R$, $a > 0$	$\frac{a}{2} exp[-a\|x-b\|]$
6	Показательный	$x \in (0;\infty)$, $\lambda > 0$	$\lambda\exp(-\lambda x)$
7	Гамма	$x \in (0;\infty)$, a>-1, b > 0	$\frac{x^a \exp(-x/b)}{b^{a+1}\Gamma(a+1)}$
8	χ^2	$x \in (0;\infty)$, $n > 0$	$\frac{1}{2^{\frac{n}{2}}\Gamma\left(\frac{n}{2}\right)} x^{n/2-1} exp(-x/2)$
9	Максвелла	$x \in (0;\infty)$, $a \in R$	$\sqrt{2/\pi}a^3x^2exp(-a^2x^2/2)$

Контрольные вопросы

1. Для каких задач обработки экспериментальных данных используется нелинейный метод наименьших квадратов?
2. В чем состоит суть метода наименьших квадратов?
3. Запишите вид выражения для целевой функции в НМНК.
4. Чему равны веса в выражении для целевой функции $\chi_v^2(\boldsymbol{\theta})$?
5. Чему равно число степеней свободы *v* в выражении для нормированного χ_v^2?
6. Каким образом вычислить доверительный интервал для оцененного параметра $\bar{\theta}$?
7. Для каких целей используются функции MATLAB: *fopen*, *load*, *path*, *fscanf*, *nlinfit*, *nlparci*, *tinv*, *diag*?

ЛАБОРАТОРНАЯ РАБОТА 3
МЕТОД ГЛАВНЫХ КОМПОНЕНТ

Цель работы. Практическое освоение метода главных компонент для решения задач снижения размерности и визуализации многомерных данных.

КРАТКИЕ СВЕДЕНИЯ

Во многих экспериментах по обработке многомерных данных приходится сталкиваться с задачами связанными с наглядным представлением данных (визуализация данных), снижением размерности данных без существенной потери информативности данных или (сжатием данных), стремлением к лаконизму исследуемых данных (упрощение данных). Для решения подобных задач используются методы, в которых снижение размерности пространства происходит одновременно с его преобразованием. Это – метод главных компонент, факторный анализ, канонический анализ. Характерной особенностью данных методов является то, что происходит выбор и оценка значимости не отдельных переменных, а информативных по совокупности групп переменных. В данной работе рассматривается метод главных компонент.

В общем виде задача снижения размерности признакового пространства может быть сформулирована следующим образом. Пусть многомерные данные представлены N - числом исследуемых объектов $n_1, n_2, \ldots, n_N$ (табл. 3.1), каждый из которых $n_i = (x_{i1}, x_{i2}, \ldots, x_{iK})$, $i = 1,\ldots, K$, характеризуется набором из K – признаков $X_1,\ldots, X_K$ (измеряемых характеристик объектов).

Таблица 3.1

Набор многомерных данных

	X_1	X_2	…	X_K
n_1	x_{11}	x_{12}	…	x_{1K}
n_2	x_{21}	x_{22}	…	x_{2K}
…	…	…	…	…
n_N	x_{N1}	x_{N2}		x_{NK}

Необходимо определить такое линейное преобразование, задаваемое матрицей $\boldsymbol{A}$, в результате действия которого исходные данные выражаются набором (матрицей) главных компонент $\boldsymbol{Z} = (Z_1, Z_2, \ldots, Z_K)$, где первые M – главных компонент ($M << K$), обеспечивают требуемую долю дисперсии γ групп признаков (как правило, $\gamma \geq 0.95$).

Матрица главных компонент строится на основе матрицы весовых коэффициентов $\boldsymbol{A} = \{a_{ij}\}$, $i=1,\ldots, K$ и $j=1,\ldots, K$, учитывающих тесноту связи между исходными признаками и главными компонентами:

$$\boldsymbol{Z} = \boldsymbol{X}\,\boldsymbol{A}, \tag{3.1}$$

где $\boldsymbol{Z} = (Z_1, Z_2, \ldots, Z_K)$ – матрица всех K главных компонент и $\boldsymbol{X} = (X_1, X_2, \ldots, X_K)$ – матрица исходных признаков. В развернутом виде j-ая главная компонента матрицы $\boldsymbol{Z}$ выражается через векторы признаков следующим образом:

$$Z_j = a_{1j} X_1 + a_{2j} X_2 + \ldots + a_{Kj} X_K\,, \tag{3.2}$$

где элементы a_{ij}, $i=1,\ldots, K$ и $j=1,\ldots, K$, – называются параметрами загрузки главных компонент.

Задача сводится к определению матрицы $\boldsymbol{A}$ (нахождению неизвестных параметров a_{ij}, где $i=1,\ldots, K$ и $j=1,\ldots, K$). Для этого используется аппарат матричной алгебры. Элементы матрицы $\boldsymbol{A}$ рассчитываются на основе корреляционной матрицы $\boldsymbol{R}$ (построенной по входным данным n_i, $i=1,\ldots,N$), решением систем линейных уравнений:

$$\boldsymbol{R}\,\boldsymbol{A} = \boldsymbol{A}\,\boldsymbol{\Lambda}, \tag{3.3}$$

где матрица

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & \lambda_K \end{pmatrix} \tag{3.4}$$

содержит на главной диагонали собственные значения корреляционной матрицы $\boldsymbol{R}$, такие, что $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_K$, а координаты собственных векторов являются искомыми параметрами a_{ij} $i=1,\ldots,K$ и $j=1,\ldots,K$. Причем выполняется равенство

$$\lambda_i = \sigma^2(Z_i), \quad i = 1,\ldots,K\,. \tag{3.5}$$

Сумма выборочных дисперсий исходных признаков равна сумме выборочных дисперсий проекций объектов на главные компоненты

$$\sigma^2(Z_1) + \sigma^2(Z_2) + \ldots + \sigma^2(Z_K) = \sigma^2(X_1) + \sigma^2(X_2) + \ldots + \sigma^2(X_K)\,. \tag{3.6}$$

Несмотря на то, что вместо K признаков получается такое же количество главных компонент, вклад большей части главных компонент в объясняемую дисперсию оказывается небольшим. Исключают из рассмотрения те главные компоненты, вклад которых мал. При помощи M первых (наиболее весомых) главных компонент можно объяснить основную часть суммарной дисперсии в данных.

Относительная доля разброса, приходящаяся на *j*-ую главную компоненту

$$\alpha_j = \frac{\sigma^2(Z_j)}{\sigma^2(Z_1)+\sigma^2(Z_2)+...+\sigma^2(Z_K)} = \frac{\lambda_j}{\lambda_1+\lambda_2+...+\lambda_K}. \quad (3.7)$$

Относительная доля разброса, приходящаяся на *i* первых компонент

$$\gamma_i = \frac{\sigma^2(Z_1)+\sigma^2(Z_2)+...+\sigma^2(Z_i)}{\sigma^2(Z_1)+\sigma^2(Z_2)+...+\sigma^2(Z_K)} = \frac{\lambda_1+\lambda_2+...+\lambda_i}{\lambda_1+\lambda_2+...+\lambda_K}. \quad (3.8)$$

Таким образом, метод главных компонент позволяет описать большой набор признаков *K* небольшим числом главных компонент *M*, *M* << *K*, при этом различия между объектами зависят от доли изменчивости, связанной с данной главной компонентой. Связи между признаками и главными компонентами – линейные.

На основе выявленных наиболее весомых главных компонент, обычно проводится компонентный анализ для объяснения индивидуальных значений данных главных компонент для рассматриваемых объектов, ранжирования объектов по весу этих значений, выбора наиболее оптимальных объектов.

Алгоритм метода главных компонент

Шаг 1. Сформировать матрицу входных данных на основе объектов n_i, *i=1,...,N*

$$\boldsymbol{X} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1K} \\ x_{21} & x_{22} & \dots & x_{2K} \\ \dots & \dots & \dots & \dots \\ x_{N1} & x_{N2} & \dots & x_{NK} \end{pmatrix}. \quad (3.9)$$

Шаг 2. Для устранения неоднородности в исходных данных выполнить нормировку (стандартизацию) данных по колонкам

$$x'_{ij} = \frac{x_{ij} - \overline{X}_j}{\sigma(X_j)}, \quad (3.10)$$

где $\overline{X}_j$ и $\sigma(X_j)$ – математическое ожидание и среднеквадратическое отклонение по *j*-ой колонке, матрица $\boldsymbol{X'} = \{x'_{ij}\}$, *i=1,..., N* и *j=1,..., K*.

Шаг 3. Построить матрицу ковариации

$$\boldsymbol{Cov = R = (X'^T X') / (N-1)} \quad (3.11)$$

Ввиду произведенной нормализации данных матрица ковариаций будет корреляционной матрицей исходных данных $\boldsymbol{R}$ порядка $K \times K$.

Шаг 4. Вычислить матрицы $\boldsymbol{A}$ и $\boldsymbol{\Lambda}$ путем решения систем линейных уравнений (3.3) .

Шаг 5. Рассчитать проекции объектов на главные компоненты

$$\boldsymbol{Z} = \boldsymbol{X'A}. \qquad (3.12)$$

Замечание. В заключение отметим, прежде чем начинать анализ главных компонент целесообразно проверить значимо ли отличается от единичной матрицы корреляционная матрица исходных нормированных данных. В предположении, что исходные данные подчиняются многомерному нормальному распределению, можно воспользоваться статистикой

$$d = N\sum_{i=1}^{K}\sum_{j=i+1}^{K} r_{ij}^2 \qquad (3.13)$$

где r_{ij} – наддиагональные элементы корреляционной матрицы R. Статистика d подчиняется χ^2-распределению с $K(K\text{-}1)/2$ степенями свободы. Если корреляционная матрица исходных данных не отличается от единичной матрицы, т. е. $d \le \chi^2$, вычисленное при заданном уровне доверительной вероятности и заданном числе степеней свободы, то применение метода главных компонент нецелесообразно.

Порядок выполнения

1. С помощью команды >> *help* изучить функции *mean*, *var*, *chi2inv*, *eig*, *flipud*, *fliplr*, *scatter*.

2. Разработать алгоритм метода главных компонент и программно его реализовать в среде MATLAB.

3. Выполнить анализ экспериментальных данных методом главных компонент.

- Загрузить данные согласно вашему варианту. Отобразите данные на экране монитора в виде таблицы (наподобие табл. 3.1).
- Нормировать (стандартизировать) исходные экспериментальные данные. Построить корреляционную матрицу.
- Удостоверится, что корреляционная матрица значимо отличается от единичной матрицы.
- Рассчитать проекции объектов на главные компоненты.

4. Произвести анализ результатов работы метода главных компонент.

- Проверить равенство сумм выборочных дисперсий исходных признаков и выборочных дисперсий проекций объектов на главные компоненты.
- Определить относительную долю разброса, приходящуюся на главные компоненты. Построить матрицу ковариации для проекций объектов на главные компоненты.

- На основе первых $M = 2$ главных компонент построить диаграмму рассеяния. Дать содержательную интерпретацию первых двух главных компонент.

Форма отчета: Работающий программный модуль, реализующий задание, тексты программ. Результаты метода главных компонент. Выводы по результатам обработки экспериментальных данных.

Таблица 3.2

Варианты заданий

Вариант	Данные	Вариант	Данные
1	data1.txt	7	data7.txt
2	data2.txt	8	data8.txt
3	data3.txt	9	data9.txt
4	data4.txt	10	data10.txt
5	data5.txt	11	data11.txt
6	data6.txt	12	data12.txt

Контрольные вопросы

1. Для каких задач обработки экспериментальных данных используется метод главных компонент?
2. В чем состоит суть метода главных компонент?
3. Чему равно математическое ожидание и дисперсия стандартизованной переменной?
4. Какова дисперсия *i*-й главной компоненты?
5. Чему равна сумма выборочных дисперсий проекций объектов на главные компоненты?
6. Какова относительная доля разброса, приходящаяся на *j*-ую главную компоненту?
7. Какова относительная доля разброса, приходящаяся на *i* первых главных компонент?
8. Целесообразно ли использование метода главных компонент, если ковариационная матрица исходных признаков диагональная?
9. Как проверить значимость корреляционной матрицы исходных данных?
10. Какова интерпретация первых двух главных компонент?
11. Какой вид связи между признаками и главными компонентами?
12. Для каких целей используются функции MATLAB: *chi2inv*, *eig*, *flipud*, *fliplr*, *scatter*?

ЛАБОРАТОРНАЯ РАБОТА 4

ИЕРАРХИЧЕСКИЕ МЕТОДЫ КЛАСТЕРНОГО АНАЛИЗА

Цель работы. Практическое освоение методов иерархического кластерного анализа данных.

КРАТКИЕ СВЕДЕНИЯ

Решение задачи разбиения множества элементов без обучения называется кластерным анализом. Кластерный анализ не требует априорной информации о данных и позволяет разделить множество исследуемых объектов на группы похожих объектов – кластеры. Это дает возможность резко сокращать большие объемы данных, делать их компактными и наглядными.

Задачу кластеризации можно сформулировать следующим образом.

Имеется некоторое конечное множество объектов произвольной природы $C = \{n_1, n_2, \ldots, n_N\}$, каждый из которых $n_i = (x_{i1}, x_{i2}, \ldots, x_{iK})$, $i = 1, 2, \ldots, N$, характеризуется набором из K – признаков (измеряемых характеристик объектов). Необходимо класстеризовать эти объекты, т.е. разбить их множество на заданное или произвольное количество групп (кластеров или классов) таким образом, чтобы в каждую группу оказались включенными объекты, близкие между собой в том или ином смысле. Априорная информация о классификации объектов при этом отсутствует. Таким образом, необходимо разбить множество векторов C на k попарно непересекающихся классов $C_1, C_2, \ldots, C_k$ так, чтобы $\bigcup_{i=1}^{k} C_i = C$, где $1 < k < N$.

Для нахождения определенного решения данной задачи необходимо задать:

- способ сравнения объектов между собой (меру сходства);
- способ кластеризации;
- разбиение данных по кластерам (установление числа кластеров).

Для сравнения двух объектов n_i и n_j могут быть использованы следующие меры:

1. Евклидово расстояние

$$d_2(n_i, n_j) = \left[\sum_{l=1}^{K} (x_{il} - x_{jl})^2 \right]^{\frac{1}{2}}; \qquad (4.1)$$

2. стандартизированное Евклидово расстояние

$$d(n_i, n_j) = \left[\sum_{l=1}^{K} \frac{(x_{il} - x_{jl})^2}{\sigma_l^2} \right]^{\frac{1}{2}}, \qquad (4.2)$$

где σ^2_l – дисперсия по l-му признаку;

3. метрика города

$$d_H(n_i, n_j) = \sum_{l=1}^{K} \left| x_{il} - x_{jl} \right|; \qquad (4.3)$$

4. расстояние Минковского

$$d_{Mnkw}(n_i, n_j) = \left[\sum_{l=1}^{K} | x_{il} - x_{jl} |^p \right]^{\frac{1}{p}}, \qquad (4.4)$$

где $p = 1, 2, \ldots, \infty$;

5. расстояние Махалонобиса

$$d_{Mah}(n_i, n_j) = (n_i - n_j) C^{-1} (n_i - n_j)^T, \qquad (4.5)$$

где C^{-1} – ковариационная матрица входных данных;

6. метрика Чебышева

$$d_{\text{ч}}(n_i, n_j) = \max_{1 \le l \le K} \left| x_{il} - x_{jl} \right|. \qquad (4.6)$$

По способам кластеризации методы кластерного анализа можно разделить на две большие группы: иерархические и неиерархические методы.

Рассматриваемые здесь иерархические методы используют последовательное объединение объектов в кластеры, малые кластеры в большие, которое может быть визуально отображено в виде дерева вложенных кластеров – дендрограммы (рис. 4.1), при этом число кластеров скрыто или условно. Как правило, на графе дендраграммы вдоль оси x располагают номера объектов, а вдоль оси y – значения меры сходства.

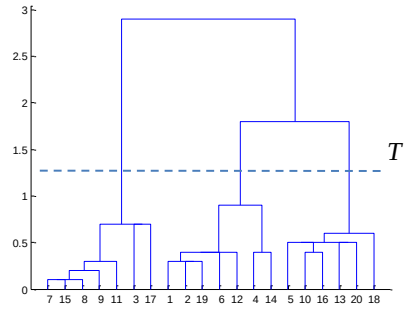

Рис. 4.1. Дендрограмма

Иерархические методы делятся на:

- *агломеративные*, характеризуемые последовательным объединением исходных объектов и соответствующим уменьшением числа кластеров (построение кластеров снизу вверх);
- *дивизимные* (делимые), в которых число кластеров возрастает начиная с одного, в результате чего образуется последовательность расщепляющихся групп (построение кластеров сверху вниз).

Методы иерархического кластерного анализа различаются по способу или методу связывания объектов в кластеры. Если кластеры *i* и *j* объединяются в кластер *r* и требуется рассчитать расстояние до кластера *s*, то наиболее часто используются следующие методы связывания объектов/кластеров *r* и *s*:

1. метод ближнего соседа (расстояние между ближайшими соседями-ближайшими объектами кластеров)

$$d_{rs} = \min(d_{is}, d_{js}); \tag{4.7}$$

2. метод дальнего соседа (расстояние между самыми дальними соседями)

$$d_{rs} = \max(d_{is}, d_{js}); \tag{4.8}$$

3. метод средней связи (среднее расстояние между всеми объектами пары кластеров с учетом расстояний внутри кластеров)

$$d_{rs} = \frac{m_i d_{is} + m_j d_{js}}{m_i + m_j}, \tag{4.9}$$

где m_i и m_j – число объектов в *i*-ом и *j*-ом кластерах соответственно;

4. центроидный метод

$$d_{rs} = \frac{m_i d_{is} + m_j d_{js}}{m_i + m_j} - \frac{m_i m_j d_{ij}}{\left(m_i + m_j\right)^2}; \tag{4.10}$$

5. метод медианной связи

$$d_{rs} = (d_{is} + d_{js}) / 2 - d_{ij} / 4 . \tag{4.11}$$

Таким образом, в большой кластер объединяются те малые кластеры, расстояние между которыми минимально.

Оценка различия. Используется для определения наиболее оптимального выбора метрического расстояния и метода связывания объектов. Два наугад выбранных объекта на иерархическом дереве связаны между собой так называемым кофенетическим расстоянием, величина которого определяется расстоянием между двумя кластерами, в которых находятся данные объекты. Мерой линейной связи между кофенетическими и метрическими расстояниями является кофенетический корреляционный коэффициент κ.

Построение иерархического дерева считается успешным, если кофенетический корреляционный коэффициент близок к 1. Для наиболее успешной кластеризации строится дендрограмма иерархического дерева.

Выделение значимых кластеров. Для выделения значимых кластеров можно задать некоторое пороговое значение *T* меры расстояний сходства (горизонтальная перпендикулярная ось *T* на дендрограмме рис. 4.1). Число значимых кластеров определяется количеством пересечений линии порога *T* и связей иерархического дерева. Причем каждая из отсекаемых линией порога ветвей дерева будет формировать отдельный кластер. На практике часто выбирают пороговое значение *T* на основе визуального анализа плотности ветвей построенной дендрограммы.

Альтернативный способ выделения значимых кластеров – метод задания фиксированного числа кластеров. Порогового значение меры сходства *T* устанавливается в корне иерархического дерева. Затем значение порога *T* постепенно снижается до тех пор, пока не будет установлено число пересечений линии порога *T* и связей иерархического дерева равное заданному количеству кластеров.

Основные этапы иерархического кластерного анализа в MATLAB

Этап 1. Вычисление матрицы расстояния между объектами *D* (функция *pdist, squareform*).

Этап 2. Связывание или группировка объектов в бинарные иерархические деревья (дендрограммы) (функции *linkage, dendrogram*).

Этап 3. Оценка качества кластеризации (функции *cophenet*).

Этап 4. Выделение значимых кластеров (функция *cluster*).

Этап 5. Визуализация и анализ значимых кластеров (функция *gscatter*).

Порядок выполнения

1. С помощью команды >> *help* изучить функции *pdist, squareform, linkage, dendrogram, cophenet, cluster, gscatter*.

2. Загрузить данные согласно вашему варианту. Построить графическое изображение экспериментальных данных.

3. Вычислить расстояния между объектами. Использовать меры для расчета расстояний согласно варианту (табл. 4.2).

4. Выполнить кластерный анализ исходных данных методом иерархической кластеризации по методам связывания согласно варианту (табл. 4.2).

5. Выполнить анализ качества кластеризации с помощью вычисления кофенетического корреляционного коэффициента. Заполнить таблицу для кофенетического корреляционного коэффициента (табл. 4.1).

Таблица 4.1

Обозначения кофенетических коэффициентов

Метод связывания	Метрика		
	1	**2**	**3**
1	κ_{11}	κ_{12}	κ_{13}
2	κ_{21}	κ_{22}	κ_{23}
3	κ_{31}	κ_{32}	κ_{33}

6. Определить наиболее и наименее эффективные способы иерархической кластеризации для анализа исходного набора данных (максимальные и минимальные коэффициенты и соответствующие им способы кластеризации). Для наиболее эффективного способа иерархической кластеризации построить дендрограмму результатов кластерного анализа.

7. Определить количество достоверных кластеров. Для выделения значимых кластеров использовать пороговое значение, рассчитанное по метрике расстояний или методом задания фиксированного числа кластеров.

8. Рассчитать центры и внутрикластерную дисперсию полученных кластеров, геометрические расстояния от элементов до центров кластеров, расстояния между центрами кластеров. Отобразить графически найденные кластеры и их центры (использовать диаграмму рассеяния в цвете).

Форма отчета: Работающий программный модуль, реализующий задание, тексты программ. Результаты иерархической кластеризации. Выводы по результатам обработки экспериментальных данных.

Таблица 4.2

Варианты заданий

Вариант	Метрики (расстояния)	Методы связывания	Данные
1	1, 2, 3	a, b, c	data1.txt
2	1, 3, 4	a, b, d,	data2.txt
3	1, 4, 5	a, b, e	data3.txt
4	1,5, 6	a, c, e	data4.txt
5	1, 2, 4	a, c, d	data5.txt
6	1, 3, 5	a, d, e	data6.txt
7	1, 4, 6	b, c, d	data7.txt
8	1, 2, 5	b, c, e	data8.txt
9	1, 3, 6	b, d, e	data9.txt
10	1, 2, 6	c, d, e	data10.txt
11	2, 4, 6	a, b, c	data11.txt
12	2, 3,6	a, b, d,	data12.txt

* Метрики расстояния: 1 – Евклидово, 2 – стандартизированное Евклидово, 3 – города, 4 – Махаланобиса, 5 – Минковского (p = 4), 6 –Чебышева.
****** Методы связывания: a – ближнего соседа, b – дальнего соседа, c – средней связи, d – центроидный, e – медианной связи.

Таблица 4.3
Вид исходных данных для кластеризации

	X_1	X_2
n_1	x_{11}	x_{12}
n_2	x_{21}	x_{22}
…	…	…
n_N	x_{N1}	x_{N2}

Контрольные вопросы

1. В чем заключается задача кластерного анализа?
2. Для каких задач обработки экспериментальных данных используются методы иерархического кластерного анализа?
3. Перечислите основные меры сравнения объектов между собой.
4. Что такое дендрограмма?
5. Что представляют собой иерархические агломеративные методы кластерного анализа?
6. Что представляют собой иерархические дивизимные методы кластерного анализа?
7. Перечислите основные способы связывания объектов в кластеры.
8. Что такое кофенетический корреляционный коэффициент?
9. В чем заключаются основные этапы иерархического кластерного анализа?
10. Каким образом определить значимое число кластеров?
11. Для каких целей используются функции MATLAB: *pdist, squareform, linkage, dendrogram, cophenet, cluster, gscatter*?

ЛАБОРАТОРНАЯ РАБОТА 5

НЕИЕРАРХИЧЕСКИЕ МЕТОДЫ КЛАСТЕРНОГО АНАЛИЗА

Цель работы. Практическое освоение неиерархического кластерного анализа многомерных данных на примере метода *k*-средних.

КРАТКИЕ СВЕДЕНИЯ

Задачей кластерного анализа является организация наблюдаемых данных в наглядные структуры – кластеры. Исходные данные могут быть представлены в виде матрицы X, строки которой соответствуют объектам (наблюдениям), а столбцы их признакам (см. табл. 5.1). Требуется разбить заданное множество объектов на кластеры.

Таблица 5.1

Многомерные данные

	X_1	X_2	…	X_K
n_1	x_{11}	x_{12}	…	x_{1K}
n_2	x_{21}	x_{22}	…	x_{2K}
…	…	…	…	…
n_N	x_{N1}	x_{N2}		x_{NK}

При большом количестве объектов ($N > 200$) иерархические методы кластерного анализа непригодны, ввиду больших вычислительных затрат и сложности интерпретации дерева кластеров. В таких случаях могут использоваться неиерархические методы кластерного анализа, одним из которых является метод *k*-средних. Суть этого метода заключается в следующем. Предположим, уже имеются гипотезы относительно числа кластеров – т.е. заранее определено количество кластеров k, на которые необходимо разбить имеющиеся объекты. Среди множества объектов выбираются k объектов в качестве начальных центров кластеров. Для каждого объекта рассчитываются расстояния до центров кластеров, и данный объект относится к тому кластеру, расстояние до которого оказалось минимальным. После чего, для этого кластера (в котором изменилось количество объектов) рассчитывается новое положение центра кластера (как среднее по каждому признаку X_i) по всем включенным в кластер объектам. В общем случае, в результате применения метода *k*-средних исходное множество объектов разделяется ровно на k различных кластеров, расположенных на возможно больших расстояниях друг от друга.

Расстояние между объектами n_i и n_j может быть вычислено, например, по следующим формулам:

1. Евклидово расстояние

$$d_2(n_i, n_j) = \left[\sum_{l=1}^{K} (x_{il} - x_{jl})^2 \right]^{\frac{1}{2}}; \quad (5.1)$$

2. расстояние Минковского

$$d_{Mnkw}(n_i, n_j) = \left[\sum_{l=1}^{K} | x_{il} - x_{jl} |^p \right]^{\frac{1}{p}}. \quad (5.2)$$

Для хранения информации о принадлежности объекта к некоторому кластеру в методе *k*-средних вводится матрица $U = \{u_{ij}\}$, $i = 1,\ldots, N$ и $j = 1, 2$. В первом столбце матрицы U содержатся индексы кластеров, к которым относятся объекты данных, во втором столбце – расстояния от объектов до соответственных центров кластеров.

Алгоритм *k*-средних является итерационным. Блок-схема алгоритма неиерархической кластеризации по методу *k*-средних представлена на рис. 5.1.

Алгоритм метода k-средних

Шаг 1. Инициализация начальных параметров метода (блок 1, рис. 5.1).

Задать: *k* - количество предполагаемых кластеров, матрицу координат центров кластеров $C^{(0)} = \{c_l^{(0)}\}$, $l = 1,2,\ldots, k$ (например, выбрать *k* объектов из файла исходных данных), матрицу $U^{(0)}$, начальное значение функционала качества кластеризации $Q^{(0)}$ (некоторое большое число) и точность его вычисления ε (для остановки алгоритма). Установить номер итерации $m = 1$.

Шаг 2. Рассчитать расстояния от объектов $n_1, n_2, \ldots, n_N$ до центров кластеров $C^{(m-1)}$, определенных на предыдущей итерации. Заполнить матрицу $U^{(m)}$, исходя из расположения центров $C^{(m-1)}$ вычисленных на предыдущей итерации (блок 2). Рассчитать значение функционала качества кластеризации $Q^{(m)}$ (с учетом $C^{(m-1)}$).

Шаг 3. Проверить условие остановки алгоритма $| Q^{(m)} - Q^{(m-1)} | \le \varepsilon$ (блок 3). При этом оценивается, привело ли новое объединение объектов в кластеры к существенному улучшению качества кластеризации. Если условие выполняется, то завершить процесс кластеризации (блок 6). Иначе перейти к *шагу 4*.

Шаг 4. Рассчитать новое положение центров кластеров $C^{(m)}$ как среднее арифметическое по координатам объектов, входящих в соответствующие кластеры (блок 4).

Шаг 5. Установить $Q^{(m-1)} = Q^{(m)}$ и перейти к *шагу 2* (новой итерации) с $m = m + 1$ (блок 5).

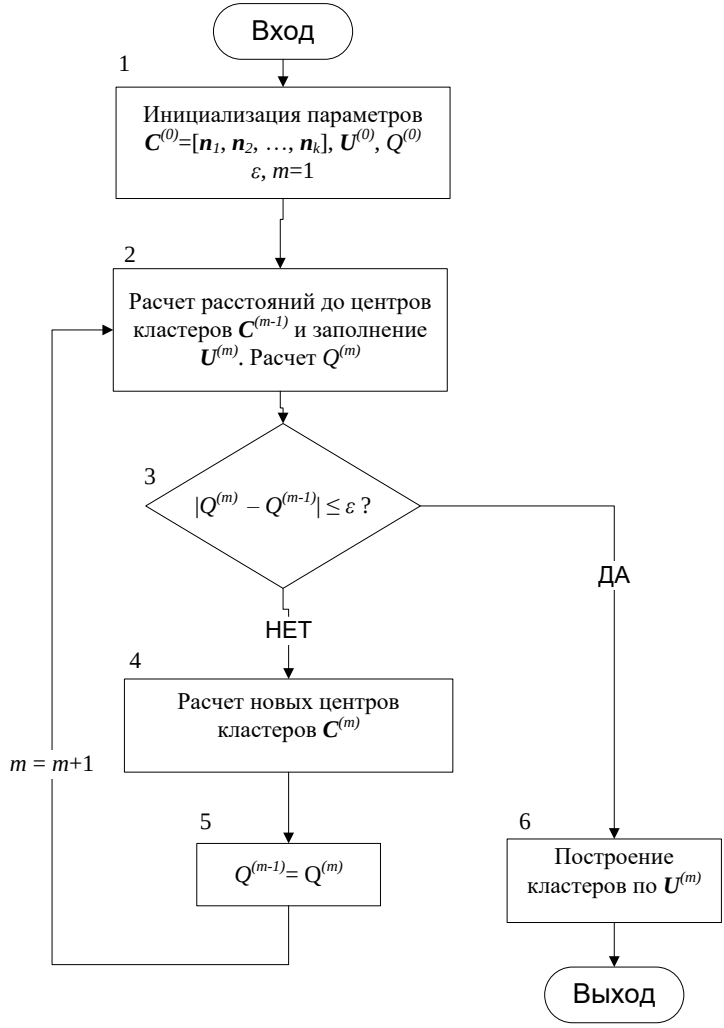

Рис. 5.1. Блок-схема алгоритма неиерархической кластеризации по методу *k*-средних

Автоматическая группировка объектов в кластеры прекращается при выполнении одного из критериев остановки автоматического группирования. На практике используются следующие виды критериев качества автоматического группирования (функционалы качества кластеризации):

1. Сумма расстояний до центров кластеров:

$$Q_1 = \sum_{l=1}^{k} \sum_{i \in S_l} d(n_i, c_l) \to \min , \tag{5.3}$$

l – номер кластера, l= 1, 2, …, k, n_i – вектор признаков i-го объекта в l-ом кластере, S_l – множество объектов в l-ом кластере.

2. Сумма квадратов расстояний до центров кластеров:

$$Q_2 = \sum_{l=1}^{k} \sum_{i \in S_l} d^2(n_i, c_l) \to \min , \tag{5.4}$$

l – номер кластера, l = 1, 2, …, k, n_i – вектор признаков i-го объекта в l-ом кластере, S_l – множество объектов в l-ом кластере.

3. Сумма внутрикластерных расстояний между объектами:

$$Q_3 = \sum_{l=1}^{k} \sum_{i,j \in S_l} d(n_i, n_j) \to \min . \tag{5.5}$$

Основным недостатком алгоритма k-средних является то, что осуществляется локальная, а не глобальная минимизация функционала Q.

Порядок выполнения

1. Изучить функции MATLAB: *scatter*, *gscatter*, *min*, *pdist*, *std.*
2. Загрузить данные согласно вашему варианту (табл. 5.2). Построить графическое изображение экспериментальных данных (диаграмму рассеяния). Визуально оценить число кластеров k по построенному изображению.
3. Разработать алгоритм кластеризации k-средних и программно его реализовать в среде MATLAB.
4. Выполнить кластерный анализ исходных данных методом k-средних (параметры метода см. в табл. 5.2). Определить наиболее оптимальное число кластеров k.
5. Рассчитать центры полученных кластеров. Отобразить графически найденные кластеры (использовать диаграмму рассеяния в цвете).

Форма отчета: Работающий программный модуль, реализующий задание, тексты программ. Результаты неиерархической кластеризации алгоритмом k-средних. Выводы по результатам обработки экспериментальных данных.

Таблица 5.2

Варианты заданий

Вариант	**Метрика (расстояние)**	**Функционал качества кластеризации**	**Данные**
1	Евклидово	Q_1	data1.txt
2	Евклидово	Q_2	data2.txt
3	Евклидово	Q_3	data3.txt
4	Евклидово	Q_1	data4.txt
5	Евклидово	Q_2	data5.txt
6	Евклидово	Q_3	data6.txt
7	Минковского (P = 4)	Q_1	data7.txt
8	Минковского (P = 4)	Q_2	data8.txt
9	Минковского (P = 4)	Q_3	data9.txt
10	Минковского (P = 4)	Q_1	data10.txt
11	Минковского (P = 4)	Q_2	data11.txt
12	Минковского (P = 4)	Q_3	data12.txt

Контрольные вопросы

1. В чем заключается задача неиерархического кластерного анализа?
2. Для каких задач обработки экспериментальных данных используются методы неиерархического кластерного анализа?
3. В чем суть алгоритма *k*-средних?
4. Перечислите основные этапы неиерархического кластерного анализа по методу *k*-средних.
5. Для каких целей в алгоритме *k*-средних вводится матрица разбиений *U*?
6. Какие критерии остановки автоматической кластеризации используются на практике?
7. Каким образом оценить число кластеров в алгоритме *k*-средних?
8. Для каких целей используются функции MATLAB: *scatter*, *gscatter*, *min*, *pdist*, *std*?

ЛАБОРАТОРНАЯ РАБОТА 6
НЕЙРОННЫЕ СЕТИ. СЛОЙ КОХОНЕНА

Цель работы: изучение алгоритма нейронной сети слой Кохонена для решения задачи кластерного анализа.

КРАТКИЕ СВЕДЕНИЯ

Нейронные сети. Нейронные сети представляют собой широкий круг алгоритмов машинного обучения и используются для решения задач классификации, кластеризации, сглаживания функций, прогнозирования, оптимизации, управления и адресации памяти.

Основой нейронной сети является искусственный нейрон – математическая модель нейрона мозга живых организмов. Искусственный нейрон состоит из синапсов (умножителей), ядра и аксона. Нейрон воспринимает входные сигналы через синапсы. Каждый синапс имеет вес, который определяет, насколько соответствующий вход нейрона влияет на его состояние. Ядро нейрона состоит из сумматора и нелинейного преобразователя. Сумматор выполняет сложение сигналов, поступающих по синапсам от других нейронов, и внешних входных сигналов. Нелинейный преобразователь реализует нелинейную функцию одного аргумента – функцию активации (передаточную функцию). Выходной сигнал формируется на аксоне. Через аксон отдельный нейрон может связываться с другими нейронами.

Для формирования искусственной нейронной сети отдельные нейроны объединяются в сети с определенной топологией. Функции активации нейронов выбираются и фиксируются на этапе проектирования сети, веса сети являются её параметрами и могут изменять свои значения. Работа нейронной заключается в преобразовании независимых переменных объекта $n_i = (x_{i1}, x_{i2}, \ldots, x_{iK})$ в выходную зависимую переменную y_i.

Общий принцип разработки нейронной сети состоит в создании модели нейронов сети и поиска неизвестных весов синапсов нейронов. Неизвестные веса определяются таким образом, чтобы минимизировать ошибку обучения – некоторую функцию разности между выходными и желаемыми значениями зависимой переменной.

Слой Кохонена (слой «соревнующихся» нейронов). Нейронные сети Кохонена представляют собой класс нейронных сетей, основным элементом которых является слой Кохонена. Алгоритм функционирования самообучающегося слоя Кохонена является одним из вариантов кластеризации многомерных данных. По аналогии с алгоритмом кластеризации *k*-средних, нейроны слоя Кохонена представляют собой узлы или центры кластеров.

Пусть выполняется анализ некоторого набора данных $n_1, n_2, \ldots, n_N$. Метки классов $y_1, y_2, \ldots, y_N$ объектов не известны. Требуется разбить набор данных на m классов, выделив m эталонных образцов (центров кластеров или классов).

Нейроны слоя Кохонена представляют собой узлы или центры кластеров, синаптические веса которых $W_1, W_2, \ldots, W_m$ должны быть определены. Нейронная сеть анализирует выходные значения нейронов слоя и в качестве результата выдает номер наиболее близко расположенного нейрона сети (нейрона-«победителя») к классифицируемому объекту (рис. 6.1). Номер активного нейрона-«победителя» определяет ту группу (кластер), к которой наиболее близок входной вектор-объект. В результате победивший нейрон, вероятно, выиграет конкуренцию и в том случае, когда будет представлен новый входной вектор, близкий к предыдущему, и его победа менее вероятна, когда будет представлен вектор, существенно отличающийся от предыдущего. Когда на вход сети поступает всё большее и большее число векторов, нейрон, являющийся ближайшим, снова корректирует свой весовой вектор W_j. В конечном счёте, если в слое имеется достаточное количество нейронов, то каждая группа близких векторов окажется связанной с одним из нейронов слоя. В этом и заключается свойство самоорганизации слоя Кохонена.

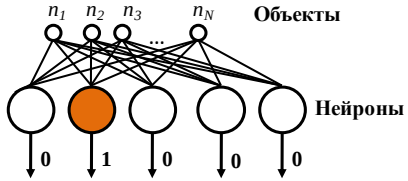

Рис. 6.1. Слой Кохонена. Положительный сигнал (единица) формируется только у одного нейрона – «нейрона-победителя», расположенного наиболее близко к входному объекту n_i. *K*-входов на каждый из нейронов представлены в виде сплошных линий.

В общем случае задача сводится к поиску минимума некоторой функции ошибок, например – суммы квадратов расстояний до центров кластеров или нейронов слоя:

$$F = \sum_{l=1}^{m} \sum_{i \in C_l} d^2\left(n_i, W_l\right) \to \min, \qquad (6.1)$$

l – номер кластера/нейрона, $l = 1, 2, \ldots, m$, n_i – вектор признаков i-го объекта в l-ом кластере, C_l – множество объектов в l-ом кластере.

Минимизацию функции F относительно ядра W_j можно осуществить методом стохастического градиентного спуска по итерационной формуле

$$W_j^{(k)} = W_j^{(k-1)} + h \cdot (n_k - W_j^{(k-1)}), \quad (6.2)$$

где h – параметр скорости обучения.

Если вектор n_k относится к кластеру C_j, то нейрон этого кластера W_j смещается в сторону объекта n_k на расстояние $h \cdot (n_k - W_j^{(k-1)})$.

Алгоритм нейронной сети слой Кохонена

1. Задать размер слоя m. Инициализировать начальные значения нейронов $W_1^{(0)}$, $W_2^{(0)}$, …, $W_m^{(0)}$. Установить номер итерации $k = 1$ и максимальное число итераций k_max.

2. Слою сети случайным образом предъявляется объект n_k. Вычислить расстояния d_{kj} между вектором n_k и всеми нейронами слоя W_j, $j = 1, 2, \ldots, m$, например, Евклидово расстояние

$$d_{kj} = \left[\sum_{t=1}^{K} (x_{kt} - w_{jt})^2 \right]^{\frac{1}{2}}. \quad (6.3)$$

3. Определить нейрон, для которого расстояние d_{kj} минимально. Выполнить коррекцию его весового вектора на величину $h \cdot (n_k - W_j^{(k-1)})$.

4. Проверить условие остановки. Если весовые факторы нейронов не изменяются или превышено максимальное количество итераций k_max, то алгоритм завершает работу.

Иначе $k = k + 1$ и перейти к п. 2.

Порядок выполнения

1. Загрузить в память компьютера исходные данные вашего варианта (табл. 6.1) и отобразить их на экране (*load, fscanf, scatter, plot*). Построить графическое изображение экспериментальных данных (диаграмму рассеяния). Визуально оценить число кластеров m по построенному изображению.

2. Разработать и программно реализовать алгоритм слоя Кохонена, задав количество нейронов равным предполагаемому количеству кластеров.

3. Выполнить кластерный анализ исходных данных с использованием разработанного алгоритма. Определить наиболее оптимальное число нейронов m. Исследовать влияние параметра обучения h на точность кластерного анализа.

4. Классифицировать исходные объекты в кластеры, исходя из принципа: объект относится к тому кластеру, расстояние до нейрона которого оказалось минимальным.

5. Отобразить графически найденные кластеры и их нейроны (*gscatter*).

Форма отчета: Работающий программный модуль, реализующий задания, тексты программ. Результаты кластеризации алгоритмом нейронная сеть слой Кохонена.

Таблица 6.1

Варианты заданий

Вариант	**Данные**
1	data1.txt
2	data2.txt
3	data3.txt
4	data4.txt
5	data5.txt
6	data6.txt
7	data7.txt
8	data8.txt
9	data9.txt
10	data10.txt
11	data11.txt
12	data12.txt

Контрольные вопросы

1. Для каких задач обработки экспериментальных данных используются нейронные сети?
2. Что такое искусственный нейрон?
3. Объясните понятия: синапсы, ядро, аксон нейрона.
4. В чем суть алгоритма слоя Кохонена?
5. Опишите этапы алгоритма слоя Кохонена.
6. Каким образом выбрать размер слоя Кохонена?
7. Какие критерии остановки алгоритма можно использовать?
8. Влияет ли выбор начальных приближений для параметров нейронов на точность кластеризации данных?
9. Влияет ли выбор параметра обучения h на точность кластерного анализа?

ЛАБОРАТОРНАЯ РАБОТА 7

НЕЙРОННЫЕ СЕТИ. КАРТЫ КОХОНЕНА

Цель работы: изучение самоорганизующихся нейронных сетей на основе карт Кохонена для решения задач кластеризации и визуализации многомерных данных.

КРАТКИЕ СВЕДЕНИЯ

Самоорганизующиеся карты Кохонена. Дальнейшим развитием слоя Кохонена стали самоорганизующиеся карты Кохонена (пер с англ. Self-Organizing Maps). Самоорганизующаяся карта Кохонена – это однослойная нейронная сеть без смещения с конкурирующей функцией **compet**, имеющая определенную топологию размещения нейронов в *K*-мерном пространстве признаков. В отличие от слоя Кохонена карта Кохонена после обучения поддерживает такое топологическое свойство, когда близким входным векторам соответствуют близко расположенные активные нейроны. Как правило, карта Кохонена представляет собой двумерную сетку с прямоугольными или шестиугольными ячейками, в узлах которой располагаются нейроны (рис. 7.1).

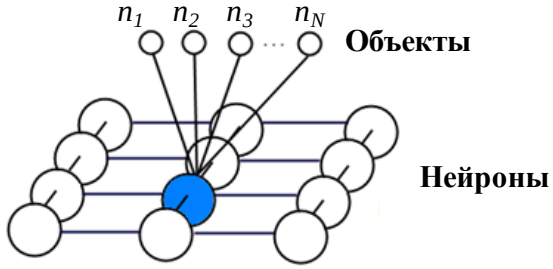

Рис. 7.1. Фрагмент карты Кохонена с прямоугольной конфигурацией. Нейроны сети соединёны с соседними нейронами. При обучении подстраиваются веса не только «нейрона-победителя», но и его соседей. *K*-входов на каждый из нейронов представлены в виде сплошных линий.

В ходе создания карты Кохонена в начале задаётся её конфигурация (прямоугольная или шестиугольная), количество нейронов и размер карты. Затем проводится инициализация весов нейронов, для чего применяются специальные алгоритмы (в простейшем случае, веса инициализируются малыми случайными значениями). Удачно выбранный способ инициализации может существенно ускорить процесс обучения сети. Особенностью карт

Кохонена является обучение без учителя – результат обучения зависит от структуры исходных данных. Один из способов обучения карты Кохонена заключается в том, что на каждом шаге обучения из исходного набора данных случайно выбирается один из векторов, а затем производится поиск наиболее похожего на него вектора коэффициентов нейронов. При этом выбирается «нейрон-победитель», вектор которого наиболее близок к вектору входов. После того, как он найден, производится корректировка весов нейронной сети. При этом, в отличие от слоя «соревнующихся» нейронов, осуществляется коррекция весового вектора не только «нейрона-победителя», **но и весовых векторов остальных активных нейронов**, хотя и в меньшей степени – в зависимости от удаления от «нейрона-победителя». Корректировки весов нейронов ΔW_q, расположенных в окрестности нейрона победителя W_j, выполняется следующим образом (с учетом вектора входа $n_i = (x_{i1}, x_{i2}, \ldots, x_{iK})$):

$$\Delta W_q = A\, h_q\, (n_i - W_q), \qquad (7.1)$$

где h_q – параметр скорости обучения, $A = 1$ для $q = j$, $A = 0.5$ для $q \neq j$ и $d_{qj} < R_q$, где d_{qj} – расстояние между нейронами q и j, R_q – радиус окрестности нейрона победителя, $A = 0$ для $q \neq j$ и $d_{qj} > R_q$.

В результате, векторы весов «нейрона-победителя» и его ближайших соседей перемещаются ближе к входному вектору. Форма и величина окрестности R вокруг «нейрона-победителя» в процессе обучения изменяются. Начинают с очень большой области R (может включать все нейроны слоя Кохонена). Задача нахождения неизвестных векторов весов нейронов решается итерационным методом (функция *train*).

Порядок выполнения

1. Создать нейронную сеть (например, 2×2) на основе карт Кохонена (*newsom*).
2. Обучить сеть на основе карт Кохонена (*net.trainParam.epochs =100, train*) используя выборку вашего варианта (табл. 7.1). Файл данных *Learning_data*.txt*.
3. Классифицировать исходные объекты в кластеры с использованием разработанной нейронной сети (*sim, vec2ind*).
4. Загрузить файл данных *PCA_data*.txt* согласно вашему варианту (табл. 7.1). Файл содержит исходные данные в координатах первых двух главных компонент Z_1 и Z_2. Отобразить графически исходные данные на диаграмме рассеяния с учетом классификации объектов нейронной сетью (*gscatter, axis*).

5. Сгруппировать объекты (файла данных *Learning_data*.txt*) в кластеры, рассчитать средние значения по каждому признаку в кластерах (*mean*).

6. Для каждого кластера построить график средних значений признаков объектов (характеристических векторов) попавших в кластер (*subplot, plot, subtitle*).

7. Отобразить проекции объектов на оси Z_1 и Z_2, включённых в каждый кластер, в командном окне Matlab.

Форма отчета: Работающий программный модуль, реализующий задания.

Таблица 7.1

Варианты заданий

Вариант	**Данные**	
1	Learning_data1.txt	PCA_data1.txt
2	Learning_data2.txt	PCA_data2.txt
3	Learning_data3.txt	PCA_data3.txt
4	Learning_data4.txt	PCA_data4.txt
5	Learning_data5.txt	PCA_data5.txt
6	Learning_data6.txt	PCA_data6.txt
7	Learning_data7.txt	PCA_data7.txt
8	Learning_data8.txt	PCA_data8.txt
9	Learning_data9.txt	PCA_data9.txt
10	Learning_data10.txt	PCA_data10.txt
11	Learning_data11.txt	PCA_data11.txt
12	Learning_data12.txt	PCA_data12.txt

Контрольные вопросы

1. Для каких задач обработки экспериментальных данных используются нейронные сети карты Кохонена?
2. Опишите принцип работы карты Кохонена.
3. Чем характеризуется состояние *j*-го нейрона в карте Кохонена?
4. Каким образом выбрать размер слоя и карты Кохонена?
5. Интерпретируйте полученную карту Кохонена?
6. Для каких целей используются функции MATLAB: *newsom, train, sim, vec2ind, subplot*?

ЛАБОРАТОРНАЯ РАБОТА 8

СТОХАСТИЧЕСКИЕ МЕТОДЫ ПОИСКА

Цель работы: практическое освоение алгоритмов программирования стохастических методов поиска глобального экстремума (минимума) многомодальной целевой функции нескольких переменных.

КРАТКИЕ СВЕДЕНИЯ

Стохастические методы поиска позволяют находить глобальные экстремумы многомодальных функций. Как правило, данная задача характерна для анализа "грязных" экспериментальных данных с высоким уровнем экспериментального шума. Среди множества локальных экстремумов (минимумов) целевой функции необходимо определить истинный, т.е. глобальный минимум, действительно соответствующий физическим параметрам исследуемых процессов. В данной работе рассматриваются два метода стохастического поиска: простой стохастический поиск и метод имитации отжига (от англ. «simulated annealing»).

Простой стохастический поиск. Процедура простого стохастического поиска заключается в вычислении целевой функции в N точках случайно подобранных комбинаций значений переменных. Среди N вычисленных величин функции находят минимальное значение, что соответствует глобальному минимуму функции. Точность нахождения глобального минимума повышается при $N \to \infty$. Выбор случайных точек может производиться различными способами. В простейшем случае N точек выбираются случайным образом в соответствии с равномерным законом распределения.

Пусть $F(\boldsymbol{X})$ – многоэкстремальная функция нескольких переменных, где $\boldsymbol{X} = (x_1, x_2, \ldots, x_M)^T$ – вектор переменных, при ограничениях $a_k < x_k < b_k$, $k = 1, 2, \ldots, M$.

Алгоритм простого стохастического поиска

Шаг 1. Задать N – число случайных точек. Вычислить случайные точки $\boldsymbol{X}_i$, где $x_{ki} = a_k + (b_k - a_k)\xi_{ki}$, где ξ_{ki} – реализация равномерно распределенной случайной величины на интервале (0;1), i=1, 2, …, N, k = 1,2, …, M.

Шаг 2. Вычислить N значений функции $F(\boldsymbol{X}_i)$, где i =1, 2, …, N.

Шаг 3. Определить минимальное значение $F(\boldsymbol{X}_i)$:
$F(\boldsymbol{X}_{min}) = \min_i F(\boldsymbol{X}_i)$. Вектор $\boldsymbol{X}_{min}$ соответствует глобальному минимуму функции F.

Метод имитации отжига. Метод имитации отжига первоначально применялся для моделирования процесса формирования кристаллических решёток. Название метода происходит от физического процесса отжига, применяемого для получения «идеальной» кристаллической решётки. В «идеальной» кристаллической структуре все атомы находятся в её узлах, имея минимальную потенциальную энергию, дефектов нет. Для достижения такого состояния вещество «отжигают» – нагревают до высокой температуры T, а затем медленно охлаждают, пошагово уменьшая температуру. При этом атомы совершают незначительные перемещения, вероятность которых уменьшается с понижением температуры. На каждом шаге системе атомов дается некоторое время, для того, чтобы большинство из них оказалось в состоянии с минимальной при данной температуре энергией. При понижении T до 0, атомы занимают состояние с некоторой потенциальной энергией, приближающейся к минимальной.

В ходе моделирования отжига в результате перемещения атома, вычисляется его потенциальная энергия. Если энергия уменьшается – атом следует переместить в новую позицию. Иначе, рассчитывается вероятность $P = exp(-\Delta E/KT)$, где K = const, и генерируется ξ, реализация равномерно распределённой случайной величины на интервале (0;1). Если $\xi < P$, то переход атома в новое состояние также принимается.

В методе имитации отжига, поиск минимума целевой функции $F(\boldsymbol{X})$ осуществляется следующим образом. Задается начальное состояние, характеризуемое вектором параметров $\boldsymbol{X_0}$. Затем, по принципу случайного блуждания генерируется новое состояние $\boldsymbol{X}'$. Если значение целевой функции в новом состоянии лучше прежнего ($\Delta E<0$), то поиск продолжается из нового состояния ($\boldsymbol{X} = \boldsymbol{X}'$). Иначе, случайным образом разыгрывается по вероятности $P = exp(-\Delta E/T)$ принятие решения $\boldsymbol{X}'$ из запрещенной области ($\Delta E>0$). В методе, для контроля движения в запрещенную область, используется параметр T – температура. При уменьшении температуры, движение в запрещенную область происходит менее интенсивно. Точность нахождения глобального минимума зависит от скорости понижения температуры (увеличивается при более медленном понижении температуры).

На практике реализации метода имитации отжига определяются выбором закона понижения температуры, способом генерации положения пробной точки, выражением для вероятности принятия пробной точки.

Алгоритм метода имитации отжига

Шаг 1. Задать максимальную и минимальную температуры (например, $T = 50$; $T_0 = 0.001$), скорость понижения температуры v (например, v = 0.99 (0 <v <1)).

*Шаг 2.*Выбрать начальные приближения $\boldsymbol{X_0}$, текущую точку $\boldsymbol{X}$ положить $\boldsymbol{X = X_0}$ и номер текущей итерации метода $l = 0$.

Шаг 3. $l = l + 1$. Генерация новых приближений $\boldsymbol{X}'$ в направлении минимума функции F:

$$x'_k = x_k + z_k \cdot T, \tag{8.1}$$

где z_k – реализация нормальной стандартизированной случайной величины N(0;1), $k = 1, 2, \ldots, M$.

Для ускорения процедуры сходимости алгоритма рекомендуется использовать приближения

$$x'_k = x_k + z_k \cdot T \cdot ((1 + l/T)^{(2 \cdot \xi_k - 1)} - 1), \tag{8.2}$$

где ξ_k – реализация равномерно распределенной случайной величины на интервале (0;1), $k = 1,2, \ldots, M$.

Для точки $\boldsymbol{X}'$ проверить выполнение граничных условий: $a_k < x'_k < b_k$, $k = 1, 2, \ldots, M$. В случае их нарушения повторить *шаг 3*.

Шаг 4. Вычислить $F(\boldsymbol{X})$, $F(\boldsymbol{X}')$. Если $\Delta E = F(\boldsymbol{X}') - F(\boldsymbol{X}) < 0$, то $\boldsymbol{X} = \boldsymbol{X}'$ и перейти к *шагу 6*.

Иначе, вычислить вероятность перехода в новую точку $P = \exp(-\Delta E / T)$.

*Шаг 5.*Если ($\xi < P$), где ξ – реализация равномерно распределенной случайной величины на интервале (0;1), то $\boldsymbol{X}=\boldsymbol{X}'$, иначе $T = v*T$.

*Шаг 6.*Если $T < T_0$, то завершить поиск, иначе перейти к *шагу 3*.

Порядок выполнения

1. С помощью команды >> *help* изучить функции *meshgrid*, *mesh*, *surface*, *min* (см. лабораторную работу №1).

2. Построить трехмерный график заданной функции согласно вашему варианту.

3. Реализовать алгоритм простого стохастического поиска глобального минимума целевой функции согласно вашему варианту.

4. Реализовать алгоритм метода имитации отжига.

5. Сравнить точки глобального минимума и значения функции в минимуме, определенные реализованными методами стохастического поиска.

Форма отчета: Работающий программный модуль, реализующий задание, тексты программ. Результаты значения глобальных минимумов, найденных с помощью простого стохастического поиска и метода имитации отжига. Выводы по результатам работы методов.

Таблица 8.1

Функции для программной реализации

№	Функция	Ограничения	Глобальные минимум(ы) (*x*; *y*)	Min(*F*)
1	$F(x,y)=837.9657-x\sin(\sqrt{\lvert x\rvert})-$ $-y\sin(\sqrt{\lvert y\rvert})$	-500 ≤ *x* ≤500 -500 ≤ y ≤500	(420.98, 420.98)	0.0
2	$F(x,y)=(4-2.1x^2+\frac{x^4}{3})x^2+xy+$ $+4(y^2-1)y^2$	-3 ≤ x ≤ 3, -2 ≤ y ≤ 2	(-0.0898; 0.713), (0.0898; -0.713)	-1.03
3	$F(x,y)=20+\left[x^2-10\cos(2\pi x)\right]+$ $+\left[y^2-10\cos(2\pi y)\right]$	-4 ≤ x ≤ 4 -4 ≤ y ≤ 4	(0; 0)	0
4	$F(x,y)=\frac{x^2+y^2}{A}-B(\cos(x)+\cos(y))$, *A*=2, *B*=12	-8 ≤ *x* ≤ 8 -8 ≤ y ≤ 8	(0; 0)	-24
5	$F(x,y)=\frac{x^2+y^2}{50}-\cos(x)\cos\left(\frac{y}{\sqrt{2}}\right)+1$	-10 ≤ *x* ≤ 10 -10 ≤ y ≤ 10	(0; 0)	0
6	$F(x,y)=-20\exp\left[-\sqrt{0.2*(x^2+y^2)}\right]-$ $\exp\left[0.5*(\cos(2\pi x)+\cos(2\pi y))\right]+$ $+20+\exp(1)$	-30 ≤ *x* ≤ 30 -30 ≤ y ≤ 30	(0; 0)	0
7	$F(x,y)=\frac{x^2+y^2}{A}-B(\cos(x)+\cos(y))$, где *A*=2, *B*=6	-8 ≤ *x* ≤ 8 -8 ≤ y ≤ 8	(0; 0)	-12
8	$F(x,y)=-\frac{1+\cos(12\sqrt{x^2+y^2})}{0.5(x^2+y^2)+2}$	-5 ≤ *x* ≤ 5 -5 ≤ y ≤ 5	(0;0)	-1
9	$F(x,y)=\frac{x^2+y^2}{30}-\cos(x)\cos\left(\frac{y}{\sqrt{2}}\right)$	-10 ≤ *x* ≤ 10 -10 ≤ y ≤ 10	(0; 0)	-1
10	$F(x,y)=-x\sin(\sqrt{\lvert x\rvert})-y\sin(\sqrt{\lvert y\rvert})$	-500 ≤ *x* ≤500 -500 ≤ y ≤500	(420.98, 420.98)	-837.96

11	$F(x,y)=\frac{x^2+y^2}{A}-B(\cos(x)+\cos(y))$, где A = 2, B=9	-8 ≤ x ≤ 8 -8 ≤ y ≤ 8	(0; 0)	-18
12	$F(x,y)=-\frac{1+\cos(18\sqrt{x^2+y^2})}{(x^2+y^2)+1}$	-5 ≤ x ≤ 5 -5 ≤ y ≤ 5	(0;0)	-2

Контрольные вопросы

1. Перечислите основные этапы простого стохастического поиска.
2. Перечислите основные этапы метода имитации отжига.
3. Какая особенность метода отжига позволяет находить именно глобальный минимум, а не локальный?
4. С какой целью в методе отжига вводится параметр температура? Зависит ли от температуры величина отклонения пробной точки от текущего положения минимума?
5. Объясните термин *запрещенная область* в методе отжига.

СПИСОК ЛИТЕРАТУРЫ

1. Прикладная статистика: Классификация и снижение размерности: Справ. изд. / С. А. Айвазян [и др.]; под ред. С. А. Айвазяна. – М. : Финансы и статистика, 1989. – 607 с.

2. Технологии анализа данных: Data Mining, Visual Mining, Text Mining, OLAP / А. А. Барсегян [и др.]. – 2-е изд., перераб. и доп. СПб: БХВ-Санкт-Петербург, 2007. – 384 с.

3. *Жиглявский, А. А.* Методы поиска глобального экстремума. / А. А. Жиглявский, А. Г. Жилинскас. – М. : Наука, 1991. – 250 с.

4. *Медведев, В. С.* Нейронные сети. MATLAB 6 / В. С. Медведев, В. Г. Потемкин; под общ. ред. В. Г. Потемкина. – М. : ДИАЛОГ-МИФИ, 2002. – 496 с.

5. *Лагутин, Б. М.* Наглядная математическая статистика: учеб. пособие / Б. М. Лагутин. – М. : БИНОМ. Лаборатория знаний, 2007. – 472 с.

6. *Kirkpatrick, S.* Optimization by Simulated Annealing / S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi //Science. – 1983. – Vol. 220. – P. 671-680.

СОДЕРЖАНИЕ

Учебное издание

Яцков Николай Николаевич

Лисица Евгения Владимировна

ИНТЕЛЛЕКТУАЛЬНЫЙ АНАЛИЗ ДАННЫХ

**Методические указания к лабораторным работам
для студентов специальностей
1-31 04 02 «Радиофизика»,
1-31 04 03 «Физическая электроника»,
1-98 01 01 «Компьютерная безопасность»**

В авторской редакции

Ответственные за выпуск *Н. Н. Яцков, Е. В. Лисица*

Подписано в печать 04.02.2019. Формат 60×84/16. Бумага офсетная.
Печать офсетная. Усл. печ. л. 3,02. Уч.-изд. л. 2,5.
Тираж 50 экз. Заказ

Белорусский государственный университет.
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 1/270 от 03.04.2014.
Пр. Независимости, 4, 220030, Минск.

Отпечатано с оригинал-макета заказчика
на копировально-множительной технике
факультета радиофизики и компьютерных технологий
Белорусского государственного университета.
Ул. Курчатова, 5, 220064, Минск.