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

Н. Н. Яцков, Е. В. Лисица

АНАЛИЗ БОЛЬШИХ ДАННЫХ

Методические указания
к лабораторным работам
для студентов специальностей
1-98 80 03 «Аппаратное и программно-техническое обеспечение информационной безопасности»,
1-31 80 07 «Радиофизика»,
1-31 80 08 «Физическая электроника»

МИНСК
2019

УДК 604.8 (076.5)
ББК 32.813
Я93

Рекомендовано советом
факультета радиофизики и компьютерных технологий
28 ноября 2019 г., протокол № 3

Р е ц е н з е н т
кандидат технических наук, доцент *В. И. Микулович*

Яцков, Н. Н.

Я93 Анализ больших данных : метод. указания к лабор. работам / Н. Н. Яцков, Е. В. Лисица. – Минск : БГУ, 2019. – 50 с.

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

УДК 604.8 (076.5)
ББК 32.813

ВВЕДЕНИЕ

Данное издание представляет собой руководство к лабораторным работам по курсу «Анализ больших данных». Лабораторные работы выполняются в интегрированной вычислительной среде статистического программирования R. Цель настоящего издания – научить студента применять методы анализа больших данных для решения ряда прикладных задач обработки и анализа многомерных наборов данных на примере использования среды R. Приводятся алгоритмы предварительного анализа, распараллеливания вычислений, методы главных компонент, кластерного анализа, классификации *k*-ближайших соседей и опорных векторов. 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 лабораторным работам.

ОСНОВЫ РАБОТЫ В R.
ЛАБОРАТОРНАЯ РАБОТА 1

Цель работы. Изучение элементов языка, основных функций и среды R. Решение систем линейных уравнений. Создание пользовательских функций и построение их графиков в среде R.

Краткие сведения. Система R предназначена для выполнения статистических расчетов на компьютере. С ее помощью эффективно решаются задачи математической статистики, математического моделирования и анализа данных.

R – это одновременно операционная среда и язык программирования. Язык R прост и легко поддается изучению. Из командной строки можно вызвать отдельную команду или программу (последовательность команд), оформленную в виде r-файла. Для удобства работы со средой R рекомендуется загрузить и установить RStudio.

1. Загрузка системы RStudio и работа в главном окне

Загрузить RStudio: кнопка **Пуск** / **Программы** / **RStudio**. Создать на доступном диске собственную папку: *имя папки набирайте латинскими буквами*. Текущей папкой среды RStudio назначьте свою созданную папку с использованием команды setwd("C:/…") (рис. 1.1).

Рис. 1.1. Окно текущей папки среды RStudio

Вычисления можно производить прямо в командном окне. Для этого в главном окне после символа > наберите какие-либо арифметические действия.

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 r-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 r-файла

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

Для создания r-файла следует зайти в меню **File\ New File\ R Script**. В результате открывается текстовый редактор RStudio, предназначенный для создания и редактирования подобных файлов.

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

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

ВНИМАНИЕ! Имена объектов должны начинаться с буквы или точки (“.”) и состоять из букв, цифр, знаков точки и подчеркивания. R чувствителен к регистру!

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

3. Использование справочной системы

R 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 в качестве аргумента. Например, *sin*: *help(sin)* или *?sin*. Вызов расширенной справки производится из главного меню RStudio: **Help\ R Help** (или нажать клавишу **F1**). Поиск нужной функции осуществляется в строке поиска по некоторому ключевому слову.

Задание 3. Получите справку по следующим операторам и функциям: *t, asin, plot*.

4. Вспомогательные функции

Очистка экрана с помощью клавиш <*CTRL*>+<*L*>. С помощью сочетания клавиш <CTRL>+<L> можно очистить видимое содержимое командного окна системы R. Для этого в главном окне необходимо просто ввести сочетание клавиш <CTRL>+<L> – экран очистится от всех введенных записей.

Очистка рабочего пространства. Рабочее пространство системы R (Workspace / Global Environment) – специально зарезервированная область

оперативной памяти, в которой хранятся значения переменных, вычисляемых в рамках текущего сеанса.

С помощью команды *rm(list=ls( ))* произведите очитку рабочего пространства системы R – 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 <- c(1,2,3,4,5)                    # Создание вектора
M <- matrix(1:15,nrow=3,ncol=5)      # Создание матрицы
```

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 <- matrix(1:15,nrow=3,ncol=5)
S <- 1:5
M <- rbind(M,S)
```

Добавление **столбца** к матрице:

```
M <- matrix(1:15,nrow=3,ncol=5)
S <- 1:3
M <- cbind(M,S)
```

Удаление всего третьего столбца матрицы **M**: `M <- M[,-3]`.
Удаление всей второй строки матрицы **M**: `M <- M[-2,]`.

6. Нахождение максимального и минимального элементов матрицы

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

`apply(C,2,min)` – вектор минимальных элементов, найденных по столбцам,

`apply(C,1,min)` – вектор минимальных элементов, найденных по строкам (аналогично для функции *max*).

Для поиска минимального и максимального элемента по всей матрице применяются функции *min* и *max* соответственно.

Задание 5. Введите в **test_matrix.r** 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.r** 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.r**.

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.

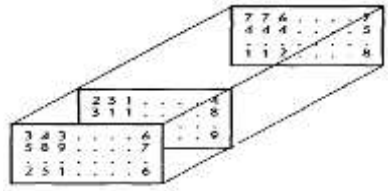


Рис. 1.2. Схематическое представление трехмерного массива

Для объявления, например, трехмерного массива состоящего из нулей следует вызвать функцию *array*:

```
A <- array(0,dim = c(2,3,4))
```

В результате получится трехмерный массив, 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 *dim*:

```
dim(A)
```

Транспонирование матрицы. Транспонирование матрицы осуществляется помощью функции *t*:

```
b <- matrix(1:4,2,2)
s <- t(b)
```

8. Арифметические операции с векторами и матрицами

По умолчанию все действия (+, –, *, /, ^ – возведение в степень) выполняются над элементами векторов или матриц

```
a <- 2:5
b <- 1:4
c <- a + b
```

Для поэлементного перемножения двух векторов или матриц

```
d <- a * b
```

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

```
a <- 2:5
b <- 1:4
c <- a %*% b
```

9. Массивы строковых переменных

Создание строковой переменной производится с использованием одинарных или двойных кавычек

```
Distance <- 'euclidean'
```

Создание вектора строковых переменных производится с помощью функции *c*:

```
Distances <- c('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 <- solve(A,B)
```

Функция *solve* вычисляет решение **x** системы линейных уравнений вида **A*x** = **B**.

Решите уравнение **A*x** = **B**, **A**=[2 1; 3 4] и **B**=[4; 11].

Проверьте найденное решение, подставив его в исходное уравнение.

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

Запустите программу на выполнение (клавиши <Ctrl>+<Shft>+<S>).

10. Работа с графикой

Пример программы, реализующей построение графика функции sin.

```
x <- seq(0,6.28,0.01)   # Создание вектора аргумента 0<x<6.28 с
шагом 0.1
y <- sin(x)    # Вычисление значений функции sin в заданных точках
plot(x,y,xlab = 'Argument x',ylab = 'Function y')  # Построение
графика функции
grid() # Отображение сетки на рисунке
title(main='Function  sin(x)')# Заголовок графика
```

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 r-файла)

```
tiff(filename = "plot.tiff",res = 100)
plot(x,y)
dev.off()
```

где `tiff` указывает формат файла (tiff), `res` = 100 – разрешение файла, `filename` – имя файла с рисунком.

Задание 8. Создайте файл **test_graphic.r**, наберите приведенный код п. 10. Сохраните полученный график в формате tiff с разрешением `res` = 100 (имя файла произвольное).

12. Построение трехмерных графиков

Рассмотрим построение трехмерных графиков функций на примере функции Розенброка. В отдельном r-файле введите код:

```
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 <- seq(Lx,Rx,stepx)
ys <- seq(Ly,Ry,stepy)
```

```
z <- outer(xs, ys, function(x, y) 100*(y - x^2)^2 + (1-x)^2)
persp(xs, ys, z, phi = 30, theta = -30,col = "lightblue")
```

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

13. Создание пользовательских функций

Рассмотрим задачу реализации функции, возвращающей значения двумерной случайной величины ξ=(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. Введите в новый r-файл следующий код функции, а затем сохраните файл:

```
my_func <- function(left, right, N){
# Генерация N значений случайной величины X в области задания X
   runif(n = N, min = left, max = right)
}
```

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

Задание 10. Создайте новый r-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.r**, 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

N <- 1000 # Задание количества сгенерированных точек
source("myfunc.R") # Вызов функции
```

```
X <- my_func(X_left, X_right, N)
Y <- my_func(Y_left, Y_right, N)

plot(X,Y) # Построение графика функции
```

Построение гистограммы с помощью *hist.* В файле **test_my_func.r** добавьте следующий код:

```
# Инициализация параметров для построения 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

# Построение гистограмм для X и Y
hist(X,X_bins)
hist(Y,Y_bins)
```

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

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

Для этого воспользуйтесь встроенной функцией *rnorm*, которая позволяет получать реализацию нормальной случайной величины α.

Код, реализующий созданную функцию (построение гистограммы), сохраните в новом файле **test_my_gauss_gen.r**.

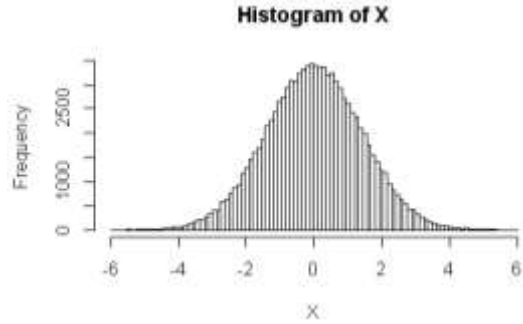


Рис. 1.3. Результат работы функции **my_gauss_gen** с параметрами ***m***=0 и ***D***=2, N=10000, количество каналов гистограммы равно 40

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 <- matrix(0,nrow = n, ncol = m);

# В цикле заполняем матрицу Q новыми значениями
for (k in 1:n){
    for (j in 1:m){
        Q[k,j] <- round(10*runif(1));
    }
}
print(Q)
```

Цикл *while*. Пример:

```
k <- 0
while(k < 20){
  print(k)
  k <- k+1
}
```

Просмотрите результаты выполнения программы в главном окне RStudio.

Задание 13. Создайте файл **test_cycle.r**, в котором сформируйте матрицу **Q** произвольного размера и заполните ее случайными целыми числами. Выведите в главном окне RStudio матрицу **Q** (добавьте в код команду `print(Q)`).

Задание 14. В файле **test_cycle.r** реализуйте цикл, позволяющий вычислять сумму элементов матрицы **Q**. Необходимый код напишите в файле **test_cycle.r**.

15. Работа с текстовыми файлами

Запись данных в файл с помощью функции *write.table*. Пример:

```
x <- seq(from=0,to=5,by=0.1)
y <- 2*x^2 + x - 1
M <- cbind(x,y)
write.table(M,"MyFile.txt")
```

Чтение данных из файла с помощью функции *read.table*. Пример:

```
A = read.table("MyFile.txt")
plot(A[,1],A[,2], xlab='Argument x', ylab='Function y')
title(main='My polynom function')
```

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

Задание 16. Создайте новый r-файл с именем **read_txt**, в котором наберите код считывания из файла с помощью *read.table.* Запустите программу на выполнение. Выведите на экран матрицу данных из файла.

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

Контрольные вопросы

1. Каким образом производятся вычисления в строке интерпретатора команд в командном окне R или RStudio?
2. Как получить информацию о требуемой функции R?
3. Как образом удалить из оперативной памяти переменные, зарезервированные во время рабочей сессии?
4. Каким образом производится очистка рабочего окна?
5. Как создать в командной строке вектор, матрицу, трехмерный массив? Выведите в рабочем окне элемент вектора, матрицы, массива.
6. Как добавить заданный столбец/строку к матрице?
7. Что произойдет в результате применения операции t(**b)**, где **b** – матрица?
8. Как определить размер вектора, матрицы, массива?
9. С помощью какой функции R производится построение двумерного графика функции? Трехмерного графика функции?
10. С помощью каких функций R производится запись/чтение в/из файл(а)?

ПРЕДВАРИТЕЛЬНЫЙ АНАЛИЗ БОЛЬШИХ НАБОРОВ ДАННЫХ. ЛАБОРАТОРНАЯ РАБОТА 2

Цель работы: Практическое освоение базовых элементов предварительного анализа данных – очистка данных и вычисление среднего значения.

КРАТКИЕ СВЕДЕНИЯ

Методы предварительного анализа данных нацелены на очистку данных, получение информации о центральной тенденции и вариации в данных, оценке степени статистической взаимосвязи в исходных данных.

Пропущенные значения. После того как получен набор данных, необходимо оценить качество данных и, если потребуется, выполнить процедуру очистки данных. Под данными низкого качества или грязными данными подразумеваются отсутствующие, неточные или бесполезные данные с точки зрения практического применения. Наиболее распространенный вид грязных данных – пропущенные значения (missing values). Очистка данных направлена на устранение пропущенных значений в данных. Пропущенные значения в экспериментальных данных могут быть обусловлены различными причинами, как правило, экспериментального характера. Пример. $n_1 = (1, NA, 10, NA, 3, 5)$, где NA (от англ. Not Available) – пропущенное значение. Наиболее простой способ устранения влияния пропущенных значений на результаты работы алгоритмов анализа – исключить объекты с пропущенными значениями из дальнейшей обработки.

Выборочное среднее значение – приближение теоретического среднего распределения выборки $x_{1j}, x_{2j}, \ldots, x_{Nj}$ объема N, основанное на вычислении арифметического среднего значения по выборке из него

$$\bar{X}_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij} \cdot \qquad (2.1)$$

Данные. Необходимо выполнить предварительный анализ данных, размещенных в архиве *specdata.zip*. Работа состоит из двух частей, в каждой из которых требуется запрограммировать функцию, позволяющую выполнить заданные расчеты для указанного набора файлов.

В архиве *specdata.zip* размещены 332 файла формата CSV (comma-separated-value), содержащие данные о мониторинге загрязнений воздуха твердыми частицами в различных районах США. Определенный файл содержит данные из одной отдельной лаборатории (далее – источник данных). Идентификационный номер (ID number) источника данных указан в

названии файла. Например, данные из источника 200 расположены в файле 200.csv. В файле размещены четыре переменные:

«Date» (Дата) – дата регистрации наблюдения в формате YYYY-MM-DD (year-month-day).

«sulfate» (Сульфаты) – уровень загрязнения воздуха сульфатами в указанный день (мкг/см^3).

«nitrate» (Нитраты) – уровень загрязнения воздуха нитратами в указанный день (мкг/см^3).

«ID» – идентификационный номер.

Для выполнения заданий требуется разархивировать файл и создать в рабочем каталоге папку *specdata*.

Замечание! Нельзя модифицировать разархивированные файлы. В файлах могут отсутствовать данные (закодированы константой NA).

Задание 1. Вычисление среднего значения. Запрограммируйте функцию pollutantmean, которая должна вычислить среднее значение уровня загрязнения сульфатами или нитратами для заданного количества источников (задается номерами файлов). Аргументы функции pollutantmean: directory – путь и название папки с данными; pollutant – тип загрязнителя ("sulfate" или "nitrate") и id – вектор цифровых значений источников данных. Принимая на входе список аргументов, функция pollutantmean считывает данные о заданном загрязнителе от указанных источников (из папки 'directory'), вычисляет среднее значение по всем наблюдениями, игнорируя пропущенные значения, и возвращает вычисленное среднее значение. Образец прототипа функции:

```
pollutantmean <- function(directory, pollutant, id = 1:332) {
## directory – вектор типа character, указывающий путь к папке,
содержащей  CSV файлы.
## pollutant – вектор типа character, указывающий имя загрязнителя
("sulfate" or "nitrate"), для которого необходимо вычислить среднее
значение.
## id – вектор типа integer, идентификационные номера файлов данных.
## На выходе функции возвращается среднее значение по всем наблюде-
ниям указанных источников, игнорируя пропущенные значения NA.
}
```

Примеры вызова разработанной функции pollutantmean.R.

```
> source("pollutantmean.R")
> pollutantmean("specdata", "sulfate", 1:10)
[1] 4.064
> pollutantmean("specdata", "nitrate", 70:72)
[1] 1.706
> pollutantmean("specdata", "nitrate", 23)
   [1] 1.281
```

Задание 2. Очистка данных от пропусков. Запрограммируйте функцию, которая считывает указанные в вашем варианте файлы и возвращает количество наблюдений, не содержащих пропущенные значения, для каждого из файлов. Функция должна вернуть объект типа Data Frame, в котором первая колонка – имя файла ('id'), вторая колонка ('nobs') – количество наблюдений, не содержащих пропущенные значения в данном файле. Образец прототипа функции:

```
complete <- function(directory, id = 1:332) {
 ## directory – вектор типа character, указывающий путь к папке,
содержащей  CSV файлы.
 ## id – вектор типа integer, идентификационные номера файлов данных.
 ## На выходе функции возвращается объект типа Data Frame:
        ## id nobs
        ## 1  117
        ## 2  1041
        ## ...
        ## где id – номер источника данных,
        ## nobs – количество наблюдений, не содержащих пропущенные
значения в данном файле.
}
```

Пример вызова разработанной функции complete.R.

```
> source("complete.R" ")
> complete("specdata", c(9, 11, 15, 17, 19))
  id nobs
1  9  275
2 11  443
3 15   83
4 17  927
5 19  353
```

Порядок выполнения

1. С помощью *help* изучить функции *dir*, *getwd*, *setwd*, *read.csv*, *length*, *as.character*, *paste*, *mean*, *complete.cases*, *data.frame*.
2. Загрузить данные к **заданию 1** согласно вашему варианту. Выполнить **задание 1** (табл. 2.1).
3. Загрузить данные к **заданию 2** согласно вашему варианту. Выполнить **задание 2**.

Форма отчёта: работающие функции, реализующие **задания 1** и **2**, тексты программ.

Таблица 2.1

Варианты заданий

Вариант	Задание 1. ID, pollutant	Задание 2. ID
1	1:49, "nitrate"	2, 4, 8, 10, 12
2	50:99, "sulfate"	3, 5, 9, 11, 13
3	99:149, "nitrate"	4, 6, 10, 13, 14
4	150:199, "sulfate"	5, 7, 11, 13, 15
5	200:249, "nitrate"	6, 8, 12, 14, 16
6	250:299, "sulfate"	7, 9, 13, 15, 17
7	300:322, "nitrate"	8, 10, 14, 16, 18
8	20:70, "sulfate"	9, 11, 15, 17, 19
9	120:170, "nitrate"	10, 12, 16, 18, 20
10	140:190, "sulfate"	11, 13, 17, 19, 21
11	220:270, "nitrate"	12, 14, 18, 20, 22
12	240:290, "sulfate"	13, 14, 18, 20, 22

Контрольные вопросы

1. С какой целью используются методы предварительного анализа данных?
2. Перечислите базовые элементы предварительного анализа данных.
3. Назовите меры оценки центральной тенденции в данных.
4. Перечислите меры оценки вариации в данных.
5. Что подразумевается под данными низкого качества или грязными данными?
6. Объясните значение термина «пропущенное значение».
7. Для каких целей используется константа NA?
8. Каким образом вычислить среднее значение для набора данных?
9. Как можно создать объект типа Data Frame?
10. Для каких целей используются функции R: *dir*, *getwd*, *setwd*, *read.csv*, *length*, *as.character*, *paste*, *mean*, *complete.cases*, *data.frame*?

РАСПАРАЛЛЕЛИВАНИЕ ВЫЧИСЛЕНИЙ. ЛАБОРАТОРНАЯ РАБОТА 3

Цель работы. Распараллеливание вычислений в ходе предварительного анализа данных на примере расчета линейного коэффициента корреляции Пирсона.

КРАТКИЕ СВЕДЕНИЯ

Распараллеливание вычислений позволяет значительно сократить время выполнения вычислений за счет использования многоядерных компьютерных систем и компьютерных кластеров. Основная идея распараллеливания состоит в разделении вычислений на независимые блоки, одновременное выполнение блоков в параллельном режиме с использованием вычислительных узлов (ядер процессора или компьютерного кластера), и объединении результатов на некотором главном узле.

Алгоритм распараллеливания вычислений:

Шаг 1. Настройка и инициализация вычислительных единиц (ядра процессора или компьютеры кластера).

Шаг 2. Представление решаемой задачи в виде независимых циклов (например, по аналогии с итерациями цикла for).

Шаг 3. Разбиение циклов на число вычислительных единиц и передача их на выполнение во все ведомые устройства.

Шаг 4. Выполнение параллельных вычислений на вычислительных узлах.

Шаг 5. Сбор результатов на главном узле (суммирование, объедение, добавление и т.д.).

Теоретический предел ожидаемого времени параллельных вычислений оценивается как

$$T = t_{CPU} / n, \tag{3.1}$$

где t_{CPU} – время вычислений на одном ядре/компьютере, n – количество ядер/компьютеров.

Научным сообществом разработано большое количество R-пакетов для реализации распараллеленных вычислений, среди которых можно выделить пакеты Rmpi, snow, snowfall, parallel и foreach. Рассмотрим подробнее пакет foreach с прицелом на использование для распараллеливания вычислительных задач на многоядерном процессоре и OS Windows.

Пакет foreach, основан на пакете snow и модели распараллеливания SOCKETS, предназначен для автоматизации и упрощения распараллеливания вычислений и позволяет относительно просто реализовать решаемую задачу в виде независимых циклов. Ключевой функцией пакета является *foreach*. В теле функции производится создание независимых циклов для распараллеливания. Три важных свойства функции:

1. функция формирует собственное пространство памяти и «не видит» загруженные в родительское пространство пакеты и функции;
2. сборка результатов осуществляется с помощью операций типа суммирование, добавление, присоединение (регулируется формальным параметром .combine = '+', c, cbind, rbind);
3. по умолчанию возвращается объект типа list, прочие типы объектов задаются операциями объединения.

Пример распараллеливания задачи вычисления суммы чисел от 1 до 10.

```
> library(foreach)
> library(doSNOW)
> cluster <- makeCluster(2, type = "SOCK")
> registerDoSNOW(cluster)
> sum.par <- foreach(i=1:10, .combine= '+') %dopar% {i}
> print(sum.par)
[1] 55
> stopCluster(cluster)
```

Выборочный коэффициент корреляции Пирсона для двух признаков X_i и X_j, представляет собой отношение их ковариации к произведению среднеквадратических отклонений этих величин:

$$r_{X_i X_j} = \frac{\mathrm{cov}_{X_i X_j}}{\sigma_{X_i} \sigma_{X_j}}, \tag{3.2}$$

или

$$r_{X_i X_j} = \frac{\sum_{l=1}^{N} (x_{li} - \bar{X}_i)(x_{lj} - \bar{X}_j)}{\sqrt{\sum_{l=1}^{N} (x_{li} - \bar{X}_i)^2 \sum_{l=1}^{N} (x_{lj} - \bar{X}_j)^2}}, \tag{3.3}$$

$$\bar{X}_j = \frac{1}{N} \sum_{i=1}^{N} x_{ij} \cdot \tag{3.4}$$

Основные свойства коэффициента корреляции:

1. $-1 \le r_{X_i X_j} \le 1 \cdot$

2. $r_{X_i X_j} = \pm 1$ – для линейно связанных признаков.
3. $r_{X_i X_j} = 0$ – для не связанных линейной связью признаков.
4. Коэффициент корреляции – безразмерная величина.

Данные. Необходимо выполнить предварительный анализ данных, размещенных в архиве *specdata.zip*. Описание файлов архива представлено в лабораторной работе №2. Работа состоит из двух частей, в каждой из которых требуется запрограммировать функцию, позволяющую выполнить заданные расчеты.

Задание 1. Определение линейной статистической связи. Запрограммируйте функцию, считывающую все файлы данных (332 файла) и вычисляющую коэффициент корреляции Пирсона между переменными "sulfate" или "nitrate" в файлах данных, для которых число не пропущенных значений выше, чем заданный порог threshold. Функция должна вернуть вектор коэффициентов корреляции для источников данных, удовлетворяющих пороговому значению. Если не наблюдается источников данных удовлетворяющих требуемому условию, то возвращается вектор типа numeric длины 0. Образец прототипа функции:

```
corr <- function(directory, threshold = 0) {
 ## directory – вектор типа character, указывающий путь к папке,
содержащей  CSV файлы.
 ## threshold – вектор типа numeric, содержащий пороговое значение,
при превышении которого вычисляется коэффициент корреляции Пирсона
между переменными "sulfate" или "nitrate" для каждого из источников
данных
 ## На выходе функции возвращается вектор типа numeric, содержащий
корреляционные коэффициенты для источников, удовлетворяющих порого-
вому условию.
}
```

Пример вызова разработанной функции corr.R.

```
> source("corr.R" ")
> print(as.matrix(corr("specdata",950))
            [,1]
[1,] -0.01895754
[2,]  0.23198440
[3,]  0.04191777
[4,]  0.42479896
[5,]  0.19014198
[6,]  0.26878050
Time difference of 3.1875 secs
```

Задание 2. Распараллеливание вычислений для определения линейной статистической связи на многоядерном процессоре. Распараллельте

задачу *задания 1*, запрограммируйте функцию с использованием программных средств пакета foreach с учетом числа ядер процессора вашего компьютера. Для автоматической оценки числа ядер процессора можно воспользоваться функцией *detectCores* пакета parallel.

Пример вызова разработанной функции corrParForeach.R.

```
> source("corrParForeach.R" ")
> print(as.matrix(corrParForeach("specdata",950))
            [,1]
[1,] -0.01895754
[2,]  0.23198440
[3,]  0.04191777
[4,]  0.42479896
[5,]  0.19014198
[6,]  0.26878050
Time difference of 2.765625 secs
```

Порядок выполнения

1. С помощью функций *install.packages* и *library* установить и подключить пакет foreach, DoSNOW, parallel. Для проверки правильности установки пакета в командной строке RStudio/R набрать help(*foreach*) и нажать Enter.

2. С помощью *help* изучить функции R и пакета foreach: *cor*, *Sys.time*, *detectCores*, *makeCluster*, *foreach*, *stopCluster*.

3. Загрузить данные к **заданию 1** согласно вашему варианту (табл. 3.1). Выполнить **задание 1**.

4. Выполнить **задание 2**.

5. Проверить совпадают ли ответы заданий 1 и 2. Сравнить время вычисления функций в заданиях 1 и 2. Оценить эффективность распараллеливания задачи (в % относительно времени выполнения нераспараллеленного варианта).

Форма отчета: работающие функции, реализующие **задания 1** и **2**, тексты программ.

Таблица 3.1

Варианты заданий

Вариант	**Задание 1. Threshold**
1	1000
2	950
3	970
4	980
5	930
6	920
7	935
8	945
9	800
10	850
11	900
12	830

Контрольные вопросы

1. В чем состоит идея распараллеливания вычислений?
2. Перечислите основные этапы алгоритма распараллеливания вычислений.
3. Чему равен теоретический предел времени распараллеленных вычислений?
4. Перечислите пакеты R для распараллеливания вычислений.
5. Назовите основные свойства функции *foreach* пакета foreach.
6. Объясните основные формальные аргументы функции *foreach*.
7. Как оценить линейную статистическую связь между двумя наборами данных?
8. Для каких целей используются функции: *cor*, *Sys.time*, *detectCores*, *makeCluster*, *foreach*, *stopCluster*?

МЕТОД ГЛАВНЫХ КОМПОНЕНТ ДЛЯ СЖАТИЯ БОЛЬШИХ ДАННЫХ. ЛАБОРАТОРНАЯ РАБОТА 4

Цель работы. Практическое освоение метода главных компонент для решения задач сжатия 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. 4.1), каждый из которых $n_i = (x_{i1}, x_{i2}, \ldots, x_{iK})$, $i = 1, 2, \ldots, K$, характеризуется набором из K – признаков $X_1, X_2, \ldots, X_K$ (измеряемых характеристик объектов).

Таблица 4.1

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 $\boldsymbol{A} = \{a_{ij}\}$, $i = 1, 2, \ldots, K$ и $j = 1, 2, \ldots, K$, учитывающих тесноту связи между исходными признаками и главными компонентами:

$$\boldsymbol{Z} = \boldsymbol{X}\,\boldsymbol{A}, \tag{4.1}$$

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:

$$Z_j = a_{1j} X_1 + a_{2j} X_2 + \ldots + a_{Kj} X_K\,, \tag{4.2}$$

где элементы a_{ij}, $i = 1, 2, \ldots, K$ и $j = 1, 2, \ldots, K$, – называются параметрами загрузки главных компонент.

Задача сводится к определению матрицы $\boldsymbol{A}$ (нахождению неизвестных параметров a_{ij}, где $i = 1, 2, \ldots, K$ и $j = 1, 2, \ldots, K$). Для этого используется аппарат матричной алгебры. Элементы матрицы $\boldsymbol{A}$ рассчитываются на основе корреляционной матрицы $\boldsymbol{R}$ (построенной по входным данным n_i, $i = 1, 2, \ldots, N$), решением систем линейных уравнений:

$$\boldsymbol{R}\,\boldsymbol{A} = \boldsymbol{A}\,\boldsymbol{\Lambda}, \tag{4.3}$$

где матрица

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & \lambda_K \end{pmatrix} \tag{4.4}$$

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}, $i = 1, 2, \ldots, K$ и $j = 1, 2, \ldots, K$. Причем выполняется равенство

$$\lambda_i = \sigma^2(Z_i), \quad i = 1, 2, \ldots, K\,. \tag{4.5}$$

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

$$\sigma^2(Z_1) + \sigma^2(Z_2) + \ldots + \sigma^2(Z_K) = \sigma^2(X_1) + \sigma^2(X_2) + \ldots + \sigma^2(X_K)\,. \tag{4.6}$$

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

$$\alpha_j = \frac{\sigma^2(Z_j)}{\sigma^2(Z_1)+\sigma^2(Z_2)+...+\sigma^2(Z_k)} = \frac{\lambda_j}{\lambda_1+\lambda_2+...+\lambda_k} \qquad (4.7)$$

Относительная доля разброса, приходящаяся на *i* первых компонент

$$\gamma_i = \frac{\sigma^2(Z_1)+\sigma^2(Z_2)+...+\sigma^2(Z_i)}{\sigma^2(Z_1)+\sigma^2(Z_2)+...+\sigma^2(Z_k)} = \frac{\lambda_1+\lambda_2+...+\lambda_i}{\lambda_1+\lambda_2+...+\lambda_k}. \qquad (4.8)$$

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, $i = 1, 2, ..., N$

$$\boldsymbol{X} = \begin{pmatrix} x_{11} & x_{12} & ... & x_{1K} \\ x_{21} & x_{22} & ... & x_{2K} \\ ... & ... & ... & ... \\ x_{N1} & x_{N2} & ... & x_{NK} \end{pmatrix}. \qquad (4.9)$$

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

$$x'_{ij} = \frac{x_{ij} - \overline{X}_j}{\sigma(X_j)}, \qquad (4.10)$$

где $\overline{X}_j$ и $\sigma(X_j)$ – математическое ожидание и среднеквадратическое отклонение по *j*-ой колонке, матрица $\boldsymbol{X'} = \{x'_{ij}\}$, $i = 1, 2, \ldots, N$ и $j = 1, 2, \ldots, K$.

Шаг 3. Построить матрицу ковариации

$$\boldsymbol{Cov} = \boldsymbol{R} = \boldsymbol{(X'^{T}X') / (N-1)} \qquad (4.11)$$

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 (4.3) .

Шаг 5. Рассчитать проекции объектов на главные компоненты

$$\boldsymbol{Z} = \boldsymbol{X'A}. \qquad (4.12)$$

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

$$d = N\sum_{i=1}^{K}\sum_{j=i+1}^{K} r_{ij}^2 \qquad (4.13)$$

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. $d \leq \chi^2$, вычисленное при заданном уровне доверительной вероятности и заданном числе степеней свободы, то применение метода главных компонент нецелесообразно.

Порядок выполнения

1. С помощью *help* изучить функции *dim, as.matrix, colMeans, apply, sd, t, print, sum, qchisq, eigen, %*%, var*.

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

3. Выполнить анализ экспериментальных данных методом главных компонент.

- Загрузить данные согласно вашему варианту (табл. 4.2).
- 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.

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

Таблица 4.2

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? Чему равно математическое ожидание и дисперсия стандартизованной переменной?
3. Какова дисперсия *i*-й главной компоненты? Чему равна сумма выборочных дисперсий проекций объектов на главные компоненты?
4. Какова относительная доля разброса, приходящаяся на *j*-ую главную компоненту? Какова относительная доля разброса, приходящаяся на *i* первых главных компонент?
5. Целесообразно ли использование метода главных компонент, если ковариационная матрица исходных признаков диагональная?
6. Как проверить значимость корреляционной матрицы исходных данных?
7. Какова интерпретация первых двух главных компонент?
9. Какой вид связи между признаками и главными компонентами?
10. Для каких целей используются функции R: *dim, as.matrix, colMeans, apply, sd, t, print, sum, qchisq, eigen, %*%, var*?

ИЕРАРХИЧЕСКИЕ МЕТОДЫ КЛАСТЕРНОГО АНАЛИЗА БОЛЬШИХ ДАННЫХ. ЛАБОРАТОРНАЯ РАБОТА 5

Цель работы. Практическое освоение методов иерархического кластерного анализа больших 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

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

2) стандартизированное Евклидово расстояние

$$d(n_i, n_j) = \left[\sum_{l=1}^{K} \frac{(x_{il} - x_{jl})^2}{\sigma_l^2} \right]^{\frac{1}{2}}, \qquad (5.2)$$

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

3) метрика города (городских кварталов, Хэмминга, Манхэттена)

$$d_H(n_i, n_j) = \sum_{l=1}^{K} \left| x_{il} - x_{jl} \right|; \qquad (5.3)$$

4) расстояние Минковского

$$d_{Mnkw}(n_i, n_j) = \left[\sum_{l=1}^{K} | x_{il} - x_{jl} |^p \right]^{\frac{1}{p}}, \qquad (5.4)$$

где $p = 1, 2, \ldots, \infty$;

5) расстояние Канберра (Canberra)

$$d_{Can}(n_i, n_j) = \sum_{l=1}^{K} \frac{\left| x_{il} - x_{jl} \right|}{\left| x_{il} + x_{jl} \right|}; \qquad (5.5)$$

6) метрика Чебышева

$$d_{\text{ч}}(n_i, n_j) = \max_{1 \le l \le K} \left| x_{il} - x_{jl} \right|. \qquad (5.6)$$

Иерархические методы используют последовательное объединение объектов в кластеры, малые кластеры в большие, которое может быть визуально отображено в виде дерева вложенных кластеров – дендрограммы (рис. 5.1), 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. Иерархические методы делятся на:

- *агломеративные*, характеризуемые последовательным объедением 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);
- *гибридные*, сочетающие положительные возможности механизмов кластеризации как агломеративных, так и дивизимных алгоритмов.

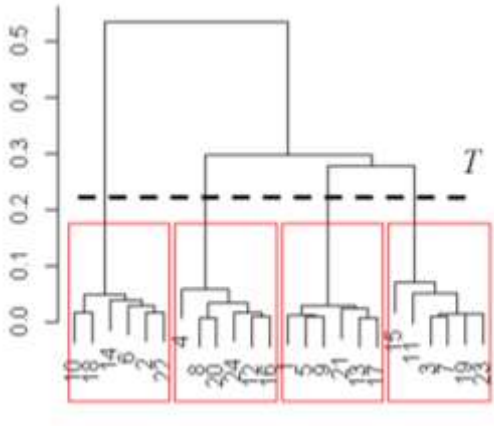


Рис. 5.1. Дендрограмма

Алгоритмы 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)

$$d_{rs} = \min(d_{is}, d_{js}); \qquad (5.7)$$

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

$$d_{rs} = \max(d_{is}, d_{js}); \qquad (5.8)$$

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

$$d_{rs} = \frac{m_i d_{is} + m_j d_{js}}{m_i + m_j}, \qquad (5.9)$$

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

4) центроидный метод

$$d_{rs} = \frac{m_i d_{is} + m_j d_{js}}{m_i + m_j} - \frac{m_i m_j d_{ij}}{\left(m_i + m_j\right)^2}; \qquad (5.10)$$

5) метод медианной связи

$$d_{rs} = (d_{is} + d_{js}) / 2 - d_{ij} / 4 . \qquad (5.11)$$

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. 5.1). 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 R

Этап 1. Вычисление матрицы расстояния между объектами D (функция *dist*).

Этап 2. Связывание или группировка объектов в иерархические деревья (дендрограммы) (функция *hclust*).

Этап 3. Оценка качества кластеризации (функции *cophenetic* и *cor*).

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

Этап 5. Визуализация и анализ значимых кластеров (функции *plot* и *rect.hclust*).

Порядок выполнения

1. С помощью *help* изучить функции *dist, hclust, cophenetic, cutree, rect.hclust, points*.

2. Загрузить данные согласно вашему варианту (табл. 5.1).

3. Вычислить расстояния между объектами. Использовать меры для расчета расстояний согласно варианту (табл. 5.1).

4. Выполнить кластерный анализ исходных данных алгоритмом иерархической кластеризации по методам связывания согласно варианту.

5. Выполнить анализ качества кластеризации с помощью вычисления кофенетического корреляционного коэффициента. Заполнить таблицу для кофенетического корреляционного коэффициента по расстояниям и методам связывания согласно варианту.

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

$$D_{C_k} = \frac{1}{N_{C_k}} \sum_{i=1}^{N_{C_k}} d^2\left(n_i, c\right), \tag{5.12}$$

где N_{C_k} – число объектов в рассматриваемом кластере, геометрические расстояния от элементов до центров кластеров, расстояния между центрами кластеров.

9. Выполнить анализ исходных данных с использованием алгоритма метода главных компонент, реализованного в лабораторной работе №3. Отобразить графически найденные кластеры в пространстве первых двух главных компонент (использовать диаграмму рассеяния в цвете).

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

Таблица 5.1

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, 3, 5	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, 3, 4	b, c, d	data7.txt
8	1, 2, 5	b, c, e	data8.txt
9	1, 3, 5	b, d, e	data9.txt
10	1, 2, 3	c, d, e	data10.txt
11	2, 4, 5	a, b, c	data11.txt
12	2, 3, 6	a, b, d,	data12.txt

* Метрики расстояния: 1 – Евклидово, 2 – стандартизированное Евклидово, 3 – города (method = ‘manhattan’), 4 – Канберра, 5 – Минковского (p = 4), 6 – Чебышева (method = ‘maximum’).

****** Методы связывания: a – ближнего соседа (method = ‘single’), b – дальнего соседа (method = ‘complete’), c – средней связи (method = ‘average’), d – центроидный (method = ‘centroid’), e – медианной связи (method = ‘median’).

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, дивизимные и гибридные алгоритмы кластерного анализа?
6. Перечислите основные способы связывания объектов в кластеры.
7. Что такое кофенетический корреляционный коэффициент?
8. Перечислите этапы иерархического кластерного анализа.
9. Каким образом определить значимое число кластеров?
10. Для каких целей используются функции R: *dist*, *hclust*, *cophenetic*, *cutree*, *rect.hclust*, *points*?

НЕИЕРАРХИЧЕСКИЕ МЕТОДЫ КЛАСТЕРНОГО АНАЛИЗА. МЕТОД *K*-МЕДОИДОВ. ЛАБОРАТОРНАЯ РАБОТА 6

Цель работы. Практическое освоение неиерархического кластерного анализа больших данных на примере метода *k*-медоидов.

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. 6.1). Требуется разбить заданное множество объектов на кластеры.

Таблица 6.1

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 неэффективны, ввиду больших вычислительных затрат и сложности интерпретации дерева кластеров. В таких случаях могут использоваться неиерархические методы кластерного анализа, одним из которых является метод *k*-медоидов. Метод подобен алгоритму *k*-средних, где вместо центров (центроидов) кластеров вычисляются медоиды.

Медоид – объект, у которого среднее расстояние до других объектов в данном кластере наименьшее. Учет медоидов вместо центроидов позволяет существенно снизить влияние выбросов и шума в ходе кластерного анализа, что значительно увеличивает вероятность сходимости алгоритма в область глобального минимума функционала качества кластеризации.

Суть алгоритма *k*-медоидов 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 медоидов. Для каждого объекта рассчитываются расстояния до

медоидов, и данный объект относится к тому кластеру, расстояние до медоида которого оказалось минимальным. После распределения объектов в k кластеров, рассчитывается новые медоиды для сформированных кластеров. В общем случае, в результате применения метода k-медоидов 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

$$d_2(n_i, n_j) = \left[\sum_{l=1}^{K} (x_{il} - x_{jl})^2 \right]^{\frac{1}{2}}; \qquad (6.1)$$

2) расстояние Минковского

$$d_{Mnkw}(n_i, n_j) = \left[\sum_{l=1}^{K} | x_{il} - x_{jl} |^p \right]^{\frac{1}{p}}. \qquad (6.2)$$

Для хранения информации о принадлежности объекта к некоторому кластеру в методе k-медоидов вводится матрица $U = \{u_{ij}\}$, $i = 1, 2, \ldots, N$ 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 медоидов кластеров.

Алгоритм k-медоидов является итерационным. Блок-схема алгоритма неиерархической кластеризации по методу k-медоидов представлена на рис. 6.1.

Алгоритм метода k-медоидов

Шаг 1. Инициализация начальных параметров метода (блок 1, рис. 6.1). Задать: k – количество предполагаемых кластеров, матрицу координат начальных медоидов $M^{(0)} = \{\mu_l^{(0)}\}$, $l = 1, 2, \ldots, k$ (например, случайно 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 медоидов кластеров $M^{(m-1)}$, определенных на предыдущей итерации. Заполнить матрицу $U^{(m)}$, исходя из расположения медоидов $M^{(m-1)}$, 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 новые медоиды кластеров $M^{(m)}$ (блок 4).

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

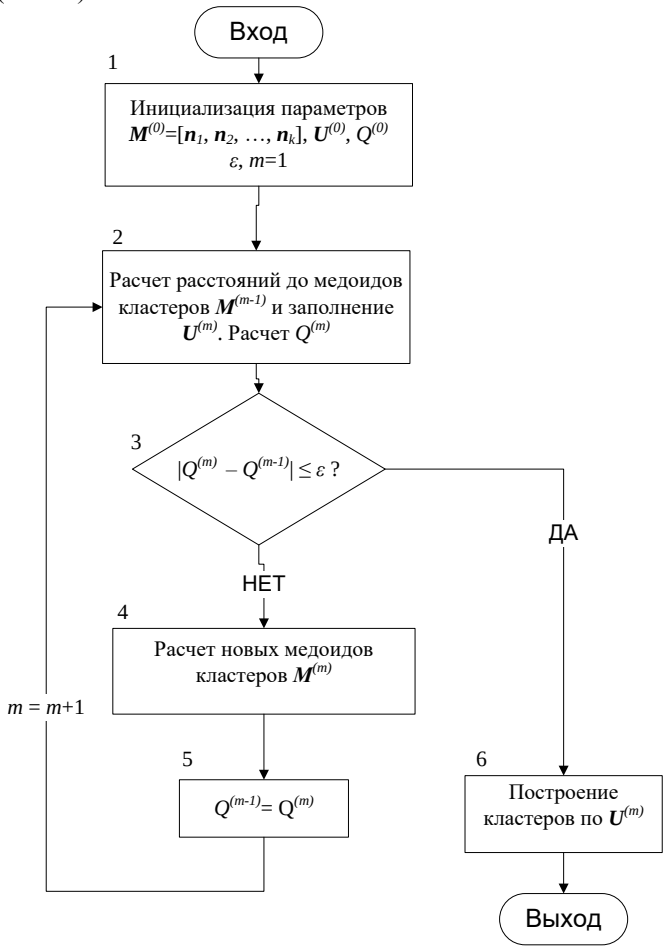


Рис. 6.1. Блок-схема алгоритма неиерархической кластеризации по методу *k*- медоидов

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 медоидов кластеров:

$$Q_1 = \sum_{l=1}^{k} \sum_{i \in S_l} d\left(n_i, \mu_l\right) \to \min, \tag{6.3}$$

l – номер кластера, $l = 1, 2, \ldots, k$, n_i – вектор признаков i-го объекта в l-ом кластере, μ_l – вектор признаков медоида в l-ом кластере, S_l – множество объектов в l-ом кластере.

2. Сумма квадратов расстояний до медоидов кластеров:

$$Q_2 = \sum_{l=1}^{k} \sum_{i \in S_l} d^2\left(n_i, \mu_l\right) \to \min. \tag{6.4}$$

3. Сумма внутрикластерных расстояний между объектами:

$$Q_3 = \sum_{l=1}^{k} \sum_{i,j \in S_l} d\left(n_i, n_j\right) \to \min. \tag{6.5}$$

Порядок выполнения

1. Изучить функции R: *source*, *abs*, *min*, *which.min*, *rbind*, *for*, *while*, *is.null*, *sum*, *list*.

2. Загрузить многомерные данные согласно вашему варианту (табл. 6.2). Выполнить анализ исходных данных с использованием алгоритма метода главных компонент, реализованного в лабораторной работе №3. Построить графическое изображение экспериментальных данных (диаграмму рассеяния для первых двух главных компонент). Визуально оценить число кластеров k по построенному изображению.

3. Разработать алгоритм кластеризации k-медоидов и программно его реализовать в среде R. Для отладки разработанного алгоритма можно воспользоваться эталонным набором данных из файла *test_data.txt* (характеристики данных: $N = 3000$, $K = 2$, $k = 3$).

4. Выполнить кластерный анализ исходных данных методом k-медоидов (параметры метода см. в табл. 6.2).

5. Отобразить графически найденные кластеры в пространстве первых двух главных компонент (использовать диаграмму рассеяния в цвете).

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

Таблица 6.2

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*-медоидов?
4. Укажите основное достоинство алгоритма *k*-медоидов.
5. Перечислите основные этапы неиерархического кластерного анализа по методу *k*-медоидов.
6. Для каких целей в алгоритме *k*-медоидов вводится матрица разбиений *U*?
7. Какие критерии остановки автоматической кластеризации используются на практике?
8. Каким образом оценить число кластеров в алгоритме *k*-медоидов в ходе анализа больших данных?
9. С какой целью в работе используется метод главных компонент?
10. Для каких целей используются функции R: *source*, *abs*, *min*, *which.min*, *rbind*, *for*, *while*, *is.null*, *sum*, *list*?

МЕТОД *K*-БЛИЖАЙШИХ СОСЕДЕЙ. ЛАБОРАТОРНАЯ РАБОТА 7

Цель работы: Практическое освоение метода *k*-ближайших соседей для решения задачи классификации данных.

КРАТКИЕ СВЕДЕНИЯ

Задача классификации сводится к определению класса объекта по его характеристикам. В данной задаче множество классов, к которым может принадлежать объект, заранее известно. Экспертные значения Y известны и дискретны для ограниченного набора объектов $n_1, n_2, \ldots, n_L$, где $L < N$, которые принадлежат m классам (табл. 7.1). Необходимо $(N - L)$ объектов разнести в m классов. Для решения данной задачи используются: метод k-ближайших соседей, байесовская классификация, методы деревьев решений, нейронные сети, метод опорных векторов.

Таблица 7.1

Многомерные данные

	X_1	X_2	…	X_K	Y
n_1	x_{11}	x_{12}	…	x_{1K}	y_1
n_2	x_{21}	x_{22}	…	x_{2K}	y_2
…	…	…	…	…	…
n_L	x_{L1}	x_{L2}	…	x_{LK}	y_L
n_{L+1}	$x_{(L+1)1}$	$x_{(L+1)2}$	…	$x_{(L+1)K}$	
…	…	…	…	…	
n_N	x_{N1}	x_{N2}	…	x_{NK}	

Метод *k*-ближайших соседей (от англ. *k*-nearest neighbour algorithm, *kNN*) – метод автоматической классификации объектов. Основным принципом метода ближайших соседей является то, что объект присваивается тому классу, который является наиболее распространенным среди соседей данного элемента (рис. 7.1). Соседи берутся из множества объектов, классы которых уже известны, и исходя из ключевого для данного метода значения *k*, высчитывается, какой класс наиболее многочислен среди них. Классификация неизвестного объекта n_i из тестируемой выборки производится следующим образом:

1) для объекта n_i вычисляются расстояния до L объектов из обучающей выборки и определяются k ближайших соседей-объектов, наиболее близко расположенных к данному объекту;

2) для объекта n_i определяются классы k ближайших соседей;

3) объект n_i классифицируется в тот класс, в который попало наибольшее число объектов из k-ближайших соседей.

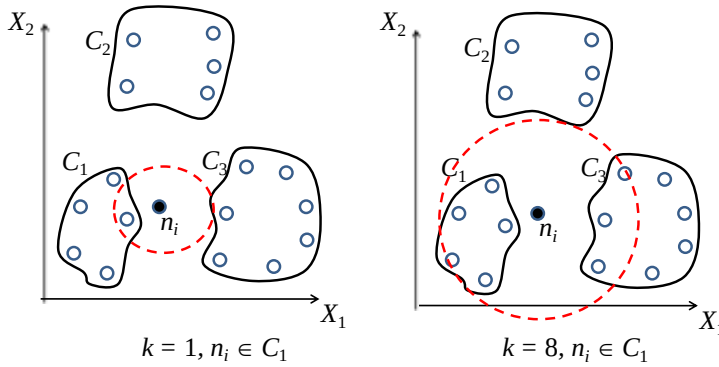


Рис. 7.1. k ближайших соседей к объекту n_i

Каким образом выбрать k ближайших соседей? Если k слишком мал, то метод чувствителен к шуму. Если k слишком велик, то окрестность может включать объекты из других классов, что приводит к ошибке классификации объектов. Один из методов, широко используемых для оценки числа ближайших соседей – метод V-кратного перекрестного контроля.

Алгоритм V-кратного перекрестного контроля

Шаг 1. Установить минимальные и максимальные числа ближайших соседей $k_{\min}$ и $k_{\max}$, $k = k_{\min}$. Разбить обучающие множество объектов $A=\{n_1, n_2, \ldots, n_L\}$ на S_i, $i = 1, 2, \ldots, V$, подмножеств, или сегментов. Предположим, что сегмент S_i является тестируемым множеством, а $A \setminus S_i$ – обучающим множеством.

Шаг 2. Выполнить классификацию по методу k-ближайших соседей для всех подмножеств $A \setminus S_i$, $i = 1, 2, \ldots, V$. Вычислить ошибки классификации Q_i для всех сегментов S_i, $i = 1, 2, \ldots, V$.

Для V классификаций вычислить среднюю ошибку классификации Q_k

$$Q_k = \frac{1}{V}\sum_{i=1}^{V} Q_i. \qquad (7.1)$$

Шаг 3. Если $k < k_{\max}$, то $k = k + 1$ и перейти к *шагу 2*. Иначе определить минимальное Q_k и соответствующие ему k.

Метрика для расчета расстояния между объектами выборки выбирается исходя из условий решаемой задачи. В качестве расстояний между двумя объектами могут использоваться метрики в кластерного анализа.

Достоинства метода *k*-ближайших соседей: простота реализации алгоритма и интерпретируемость решений, полученных на основе прямой обработки прецедентов.

Недостатки: приходится хранить в оперативной памяти компьютера всю обучающую выборку, а также неустойчивость работы алгоритма в случае наличия шумов и выбросов.

Порядок выполнения

1. С помощью функций *install.packages* и *library* установить и подключить пакет rgl. Для проверки правильности установки пакета в командной строке RStudio/R набрать > *help(rgl)* и нажать **Enter**.
2. Изучить функции R: *plot3d* (пакет rgl), *sample*, *names*, *row.names*, *sort*, *table*, *as.integer*.
3. Загрузить данные согласно вашему варианту (табл. 7.2). Построить графическое изображение экспериментальных данных (диаграмму рассеяния). Оценить число классов по построенному изображению.
4. Разработать алгоритм классификации метода *k*-ближайших соседей и программно его реализовать в среде R.
5. Из исходных данных вашего варианта сформировать обучающую и тестируемую выборки (в соотношении 1 к 1). В данных для тестируемой выборки удалить метки классов и сохранить их отдельно для последующего вычисления ошибки классификации.
6. Выполнить классификацию тестируемых данных методом *k*-ближайших соседей (параметры метода см. в табл. 7.2). Вычислить ошибку классификации как отношение числа неправильно классифицированных объектов к общему числу объектов в тестируемой выборке.
7. Отобразить графически найденные классы в пространстве исходных признаков (использовать диаграмму рассеяния в цвете).
8. Определить оптимальное число *k* ближайших соседей с использованием алгоритма *V*-кратного перекрестного контроля (например, для $V = 3$, $k_{min} = 2$ и $k_{max} = 10$).

Форма отчета: работающий программный модуль, реализующий задание, тексты программ. Результаты классификации тестируемых данных методом *k*-ближайших соседей. Выводы по результатам обработки экспериментальных данных.

Таблица 7.2

Варианты заданий

Вариант	**Метрика (расстояние)**	***k*-ближайших соседей**	**Данные**
1	Минковского (P = 4)	5	data1.txt
2	Минковского (P = 4)	6	data2.txt
3	Минковского (P = 4)	7	data3.txt
4	Минковского (P = 4)	5	data4.txt
5	Минковского (P = 4)	6	data5.txt
6	Минковского (P = 4)	7	data6.txt
7	Евклидово	5	data7.txt
8	Евклидово	6	data8.txt
9	Евклидово	7	data9.txt
10	Евклидово	5	data10.txt
11	Евклидово	6	data11.txt
12	Евклидово	7	data12.txt

Контрольные вопросы

1. В чем состоит задача классификации данных?
2. В чем суть метода *k*-ближайших соседей?
3. Перечислите основные этапы классификации данных по методу *k*-ближайших соседей.
4. Как определить качество классификации данных?
5. Каким образом выбрать *k* ближайших соседей?
6. Как вычислить расстояние между объектами в методе *k*-ближайших соседей?
7. В чем состоит идея алгоритма *V*-кратного перекрестного контроля?
8. Перечислите основные этапы алгоритма *V*-кратного перекрестного контроля.
9. Назовите достоинства и недостатки метода *k*-ближайших соседей.
10. Для каких целей используются функции R: *plot3d*, *sample*, *names*, *row.names*, *sort*, *table*, *as.integer*?

МЕТОД ОПОРНЫХ ВЕКТОРОВ. ЛАБОРАТОРНАЯ РАБОТА 8

Цель работы: Решение задачи классификации данных на основе метода (машины) опорных векторов и практическое использование R функций сторонних пакетов.

КРАТКИЕ СВЕДЕНИЯ

Метод опорных векторов (*support vector machines – SVM*) относится к методам обучения с учителем и предназначен для нахождения оптимальных в некотором смысле функций классификации данных (решающих функций). Метод наиболее эффективен при разделении двух классов.

Линейно разделимый случай. Рассмотрим задачу нахождения наилучшего в некотором смысле линейного разделения множества объектов $\{n_1, n_2, \ldots, n_N\}$ на два класса C_1 и C_2. Каждый объект $n_i = (x_{i1}, x_{i2}, \ldots, x_{iK})$, $i = 1, 2, \ldots, K$, характеризуется набором из K признаков. Есть обучающая выборка $A = \{n_1, n_2, \ldots, n_L\}$, где $L < N$, и $Y = \{y_1, y_2, \ldots, y_L\}$ – множество меток классов C_1 и C_2 для объектов обучающей выборки A. Требуется по обучающей выборке A построить линейную функцию $f(n_i)$, которая удовлетворяла бы условию

$$f(n_i) > 0 \text{ для } n_i \in C_1,$$
$$f(n_i) < 0 \text{ для } n_i \in C_2. \quad (8.1)$$

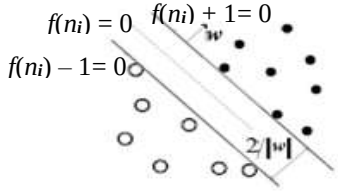
Можно полагать, что метки равны

$$y_i = \begin{cases} 1, x_i \in C_1 \\ -1, x_i \in C_2 \end{cases}, \quad (8.2)$$

тогда поставленную задачу можно переформулировать

$$y_i \cdot f(n_i) > 0. \quad (8.3)$$

Умножая, если нужно функцию f на некоторое положительное число, нетрудно видеть, что система неравенств (8.3) равносильна системе



$$y_i \cdot f(n_i) > 1, \quad \text{для всех } n_i \in A. \quad (8.4)$$

С учетом того, что $f(n_i)$ – линейная функция, то система неравенств примет вид

$$y_i \cdot ((\mathbf{w} \cdot n_i) + b) \geq 1, i = 1, 2, \ldots, L, \quad (8.5)$$

Рис. 8.1. Разделяющая полоса

где $\boldsymbol{w}$ (или $\vec{w}$) – вектор весовых коэффициентов, b – некоторое число. Тогда разделяющей два класса гиперплоскостью будет $(\boldsymbol{w}\cdot n_i) + b = 0$.

Нетрудно видеть, что и все гиперплоскости вида $(\boldsymbol{w}\cdot n_i) + b' = 0$, $b' \in (b-1, b+1)$ также будут разделяющими (см. рис. 8.1). Расстояние между граничными гиперплоскостями $(\boldsymbol{w}\cdot n_i) + b - 1 = 0$ и $(\boldsymbol{w}\cdot n_i) + b + 1 = 0$ равно $2/\| \boldsymbol{w} \|$, где $\| \boldsymbol{w} \| = (w_1^2 + w_2^2 + \ldots + w_K^2)^{1/2}$ – Евклидова норма вектора $\boldsymbol{w}$.

Опорными векторами (не менее двух) называются обучающие вектора, находящиеся на граничных гиперплоскостях.

Для надежного разделения объектов классов C_1 и C_2 необходимо чтобы расстояние между разделяющимися гиперплоскостями было как можно больше, т.е. $\| \boldsymbol{w} \|$ как можно меньше. Тогда задача нахождения классифицирующей функции $f(n_i)$ по параметрам $\boldsymbol{w}$ и b сводится к нахождению минимума квадратичного функционала $(\boldsymbol{w}\cdot\boldsymbol{w})/2$ в выпуклом многограннике, задаваемом системой неравенств (8.5). Данная задача является задачей квадратичной оптимизации, решается математически и равносильна поиску седловой точки лагранжиана

$$\Phi(\vec{w}, b, \Lambda) = \frac{\vec{w}\cdot\vec{w}}{2} - \sum_{i=1}^{L} \lambda_i (y_i(\vec{w}\cdot n_i + b) - 1) \to \min_{\vec{w},b} \max_{\Lambda} \qquad (8.6)$$

по множителям Лагранжа $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_L\}$, $0 \le \lambda_i \le C$, при условии, что $\lambda_i \cdot (y_i\ (\boldsymbol{w}\cdot n_i + b) - 1) = 0$ (что равносильно $\lambda_i = 0$ или $y_i\ (\boldsymbol{w}\cdot n_i + b) - 1 = 0$). Константа C задает соотношение между плоскостью функции $f(n_i)$ и допустимым значением нарушения границы ε. Параметр C выбирается вручную для каждой ситуации отдельно.

Из необходимых условий существования седловой точки следует, что $\boldsymbol{w}$ следует искать в виде

$$\vec{w} = \sum_{i=1}^{L} \lambda_i y_i n_i, \quad \textit{причем} \sum_{i=1}^{L} \lambda_i y_i = 0 \cdot \qquad (8.7)$$

Параметр b равен

$$b = y_s^{-1} - \boldsymbol{w}\cdot n_s \qquad (8.8)$$

где n_s – любой из опорных векторов. С учетом подстановки выражений (8.7) и (8.8) в (8.6) лагранжиан примет вид

$$\Phi(\Lambda) = \sum_{i=1}^{L} \lambda_i - \frac{1}{2} \sum_{i=1}^{L} \sum_{j=1}^{L} \lambda_i \lambda_j y_i y_j (n_i \cdot n_j) \qquad (8.9)$$

Для нахождения множителей Лангража $\{\lambda^*_1, \lambda^*_2, \ldots, \lambda^*_L\}$ решается задача поиска максимума лагранжиана при условии $\sum_{i=1}^{L} \lambda_i y_i = 0$ и $0 \le \lambda_i \le C$ с использованием методов оптимизации.

Таким образом, решением задачи (8.6) является функция вида

$$f(n_i) = \sum_{j=1}^{L} \lambda_j^* y_j (n_j \cdot n_i) + y_s^{-1} - \sum_{j=1}^{L} \lambda_j^* y_j (n_j \cdot n_s), \qquad (8.10)$$

где (n_s, y_s) – один из опорных векторов.

Линейно неразделимый случай. Метод опорных векторов также используется для нелинейного разделения классов. В этом случае необходимо вложить пространство A в пространство H большей размерности с помощью отображения φ: $A \to H$.

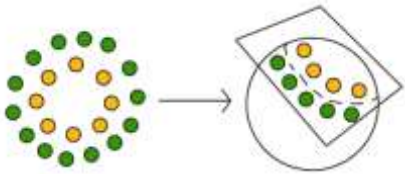


Рис. 8.2. Пример перехода к расширенному пространству

Для этого необходимо скалярное произведение двух векторов $n_j \cdot n_i$ заменить на скалярное произведение двух преобразованных векторов

$$K(n_j \cdot n_i) = \varphi(n_j) \cdot \varphi(n_i). \qquad (8.11)$$

Функция $K(n_j, n_i)$ – называется ядром. Ядро гарантирует разделение L векторов на любые два класса. В данном случае решением задачи (8.6) является функция вида

$$f(n_i) = \sum_{j=1}^{L} \lambda_j^* y_j K(n_j, n_i) + y_s^{-1} - \sum_{j=1}^{L} \lambda_j^* y_j K(n_j, n_s), \qquad (8.12)$$

где (n_s, y_s) – один из опорных векторов. Основные виды функций классификации, применяемые в методе опорных векторов следующие:

1. линейная

$$K(n_j, n_i) = n_j \cdot n_i; \qquad (8.13)$$

2. полиномиальная степени d

$$K(n_j, n_i) = (\gamma (n_j \cdot n_i) + \beta)^d; \qquad (8.14)$$

где γ и β – некоторые константы;

3. радиальная базовая функция

$$K(n_j, n_i) = e^{-\gamma |n_j - n_i|^2}. \qquad (8.15)$$

Стоит отметить, что не существует общего подхода к автоматическому выбору ядра в случае линейной неразделимости классов.

Программная реализация алгоритма в R проектах

Различные варианты алгоритма метода опорных векторов реализованы в R пакетах e1071, kernlab, klaR, svmpath, shogun. Первая и наиболее интуитивная версия алгоритма была реализована в пакете e1071 в виде функции *svm*. Вызов функции:

```
svm(formula,data,scale,type,kernel,degree,gamma,coef0, cost,...)
```

где `formula` – выражение, слева от знака "~" указывается имя колонки данных обучающей выборки, содержащей метки классов, справа – колонки признаков объектов, по которым будет проводиться обучение; `data` – набор данных обучающей выборки (Data Frame), для решения задачи классификации колонка данных, содержащая метки классов, должна иметь тип данных – factor (табл. 1); `scale` – логический вектор, указывает набор признаков для стандартизации; `type` – указывает тип анализа, для решения задачи классификации необходимо использовать "`C-classification`"; `kernel` – указывает вид функции ядра ("`linear`","`radial`", "`polynomial`"); `degree` – степень полинома *d* в формуле (14); `gamma` – параметр γ в формулах (14) и (15); `coef0` – параметр β в формуле (14); `cost` – константа *C* (см. пояснение к формуле (6)).

Функция *svm* возвращает объект класса svm, содержащий: `SV` – координаты опорных векторов; `index` – индексы опорных векторов в исходном наборе данных; `coefs` – произведения оцененных значений множителей Лангранжа на метки классов объектов $\{ y_1 \cdot \lambda^*_1, y_2 \cdot \lambda^*_2, \ldots, y_L \cdot \lambda^*_L \}$; `rho` – отрицательное значение оцененного параметра *b* (см. формулу (8.5)).

Для визуализации результатов функции *svm* (в примере ниже – объект `clt`) используются функции *predict*, *table*, *plot*. Пример.

```
>> print(table(predict(clt), Training_data$V3)) # the confusion ma-
trix
>> plot(clt,Training_data) # the palette scatter plot
```

Порядок выполнения

1. С помощью функций *install.packages* и *library* установить и подключить пакет e1071. Для проверки правильности установки пакета в командной строке RStudio/R набрать *help(svm)* и нажать **Enter**.

2. С помощью команды *help* изучить основные функции пакета e1071 для решения задач классификации: *svm*, *predict*, *plot*, *summary*.

3. Загрузить данные согласно вашему варианту (табл. 8.2). Построить графическое изображение экспериментальных данных (диаграмму рассеяния).

4. С помощью линейного метода опорных векторов разделить исходные данные на два класса. Определить опорные вектора, классифицирующую функцию (параметры $\boldsymbol{w}$ и b), ошибку классификации (матрицу неточностей – confusion matrix). Отобразить графически классифицированные объекты и классифицирующую функцию ($w_1 \cdot x + w_2 \cdot y$) + $b = 0$ (использовать диаграмму рассеяния в цвете. См. пример на рис. 8.3).

5. С использованием нелинейного метода опорных векторов разделить исходные данные на два класса с учетом вида функции классификации согласно вашему варианту. Отобразить графически классифицированные объекты с помощью функции *plot* (см. пример на рис. 8.3).

Форма отчета: работающий программный модуль, реализующий задание, тексты программ. Результаты классификации данных методом опорных векторов. Выводы по результатам обработки экспериментальных данных.

Таблица 8.1

Пример варианта исходных данных для классификации

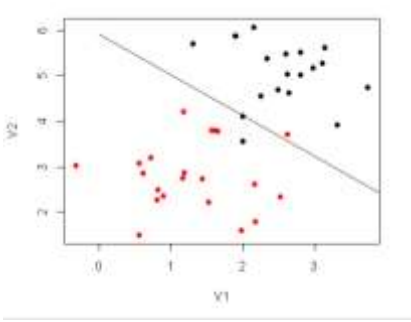
Объект	V1 (Признак 1)	V2 (Признак 2)	V3 (Класс)
1	1	1	-1
2	3	3	1
3	1	3	1

Таблица 8.2

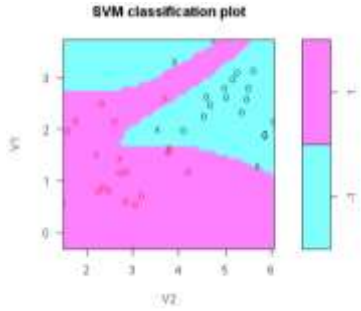
Варианты заданий

Ядра: 1 – $K(u, v\cdot) = (\gamma(u \cdot v) + \beta)^d$, 2 – $K(u,v) = e^{-\gamma|u-v|^2}$. Параметр $C = 10$

Вариант	Ядро	Параметры ядра	Данные
1	1	$\gamma = 1$, $\beta = 1$, $d = 2$	data1.txt
2	2	$\gamma = 1$	data2.txt
3	1	$\gamma = 1$, $\beta = 1$, $d = 3$	data3.txt
4	2	$\gamma = 2$	data4.txt
5	1	$\gamma = 1$, $\beta = 1$, $d = 3$	data5.txt
6	2	$\gamma = 2$	data6.txt
7	1	$\gamma = 1$, $\beta = 1$, $d = 2$	data7.txt
8	2	$\gamma = 4$	data8.txt
9	1	$\gamma = 1$, $\beta = 1$, $d = 4$	data9.txt
10	2	$\gamma = 1$	data10.txt
11	1	$\gamma = 1$, $\beta = 1$, $d = 5$	data11.txt
12	2	$\gamma = 2$	data12.txt



Линейно разделимый случай



Линейно неразделимый случай

Рис. 8.3. Примеры графического отображения классифицированных объектов

Контрольные вопросы

1. Для каких задач обработки экспериментальных данных используется метод опорных векторов?
2. Для решения какой задачи метод опорных векторов наиболее эффективен?
3. В чем состоит суть метода опорных векторов?
4. Что представляют собой опорные векторы?
5. Запишите выражение классифицирующей функции для линейно разделимого случая.
6. Запишите выражение классифицирующей функции для линейно неразделимого случая.
7. В каких R пакетах реализованы алгоритмы метода опорных векторов?
8. Укажите список аргументов функции *svm* пакета e1071.
9. Перечислите основные атрибуты объект класса svm, возвращаемого функцией *svm*.
10. Улучшилось ли разделение исходных данных на два класса в результате применение нелинейной классификации?

СПИСОК ЛИТЕРАТУРЫ

1. 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.

2. Кабаков, Р. И. R в действии. Анализ и визуализация данных в программе R / Р. И. Кабаков ; пер. с англ. – М. : ДМК Пресс, 2014. – 588 с.

3. Анализ больших наборов данных / Ю. Лесковец [и др.]; – Пер. с англ. – М. : ДМК Пресс, 2016. – 498 с.

4. Флах, П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных / П. Флах; – пер. с англ. – М.: ДМК Пресс, 2015. – 400 с.

5. Яцков, Н. Н. Интеллектуальный анализ данных : пособие / Н. Н. Яцков. – Минск : БГУ, 2014. – 151 с.

6. Vapnik, V. N. The nature of statistical leaning theory / V. N. Vapnik. – 2nd ed., – New York : Springer-Verlag, 2000. – 314 p.

7. Suthaharan, S. Machine Learning Models and Algorithms for Big Data Classification / S. Suthaharan. – New York : Springer Science+Business Media, 2016. – 359 p.

СОДЕРЖАНИЕ

Учебное издание

Яцков Николай Николаевич

Лисица Евгения Владимировна

АНАЛИЗ БОЛЬШИХ ДАННЫХ

Методические указания к лабораторным работам для студентов специальностей 1-98 80 03 «Аппаратное и программно-техническое обеспечение информационной безопасности», 1-31 80 07 «Радиофизика», 1-31 80 08 «Физическая электроника»

В авторской редакции

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

Подписано в печать 18.12.2019. Формат 60×84/16. Бумага офсетная.
Печать офсетная. Усл. печ. л. 3,02. Уч.-изд. л. 2,57.
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.