### **Отчёт по работе “Большие данные: кластеризация и классификация”**

Грицаенко Никита, Жуков Александр, Сергеев Георгий, Митрофанова Алина, Чепулис Михаил, Плаксин Даниил

#### **Задача кластеризации**

**Задание**

Имеется набор данных о растениях Армориканской возвышенности (файл ). Требуется провести кластерный анализ данных методом k-медиан с целью их разбиения на групп со сходными признаками (Рассмотреть ). Сделать выводы.

**Данные**

Описание данных: 136 наблюдений, 31 переменная.

#### **Код программы**

library(cluster)  
library(NbClust)  
  
  
euc.dist <- function(x1, x2)  
 sqrt(sum((x1 - x2) ^ 2))

**Борьба с NA**

Предложим такой способ борьбы с NA (Not Available) - ячейки матрицы, помеченные как NA, заменим на медианное значение соответствующих ячеек в других прецедентах:

removeNA <- function(matrix) {  
 data\_dim = ncol(matrix)  
 data\_size = nrow(matrix)  
 M = matrix  
 for (i in seq(1, data\_dim)) {  
 med <- median(M[, i], na.rm = TRUE)  
 for (j in seq(1, data\_size)) {  
 if (is.na(M[j, i])) {  
 M[j, i] = med  
 }  
 }  
 }  
 return(M)  
}

**Масштабирование**

normalizeMatrix <- function(matrix) {  
 data\_dim = ncol(matrix)  
 M = matrix  
 for (i in seq(1, data\_dim)) {  
 M[, i] = M[, i] / norm(data.matrix(M[, i]), type = "M")  
 }  
 return(M)  
}

**Кластеризация**

Кластеризуем с помощью функции pam. В качестве критерия кластеризации возвращаем среднеквадратическую ошибку разбиения:

clusterize <- function(matrix, num\_of\_clusters) {  
 data\_dim = ncol(matrix)  
 data\_size = nrow(matrix)  
   
 cl <-  
 pam(  
 matrix,  
 k = num\_of\_clusters,  
 metric = "euclidean",  
 stand = T,  
 keep.diss = TRUE  
 )  
   
 centers = cl$medoids  
 error = 0  
 for (i in seq(1, data\_size)) {  
 min\_dist = 1000000  
   
 for (j in seq(1, num\_of\_clusters)) {  
 dist = euc.dist(matrix[i, ], centers[j, ])  
   
 if (dist < min\_dist) {  
 min\_dist = dist  
 }  
 }  
 error = error + min\_dist ^ 2  
 }  
   
 return(sqrt(error))  
}

**Считывание данных, их обработка**

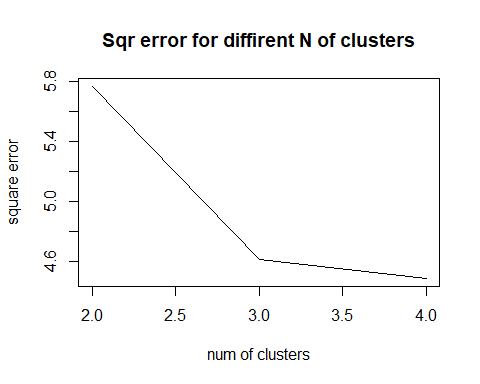
# Read data in matrix M  
path = 'data/plants.dat'  
data.plants <- read.table(  
 path,  
 sep = ';',  
 header = TRUE,  
 na.strings = "NA",  
 stringsAsFactors = T  
)  
  
data.plants$plant.name = NULL  
M = data.matrix(data.plants)  
  
data\_dim = ncol(M)  
data\_size = nrow(M)  
  
# Process data  
M = removeNA(M)  
M = normalizeMatrix(M)

**Выбор компонент для кластеризации**

#Построим матрицу корреляции  
corr\_dm = cor(M, method = "pearson")  
  
#Посмотрим, где корреляция менее всего зависит от других переменных  
corr = 1:data\_dim  
for (i in 1:data\_dim) {  
 corr[i] = sum(abs(corr\_dm[, i]))  
}  
  
variables\_idx = c(5, 7, 30, 1, 6)  
dim = 5  
  
 #Соберём новую матрицу, которая состоит из наименее коррелированных столбцов  
newM <- matrix(0, nrow = data\_size, ncol = dim)  
  
for (i in 1:dim) {  
 newM[, i] = M[, variables\_idx[i]]  
}

**Кластеризация**

# Trying to clustering  
min\_num\_of\_clusters = 2  
max\_num\_of\_clusters = 4  
  
error\_vec = matrix(0, nrow = 1, ncol = 3)  
i = 1  
  
for (k in seq(min\_num\_of\_clusters, max\_num\_of\_clusters)) {  
 error\_vec[i] = clusterize(newM, k)  
 i = i + 1  
}  
  
plot(min\_num\_of\_clusters:max\_num\_of\_clusters,  
 error\_vec,  
 "l",  
 main = "Sqr error for diffirent N of clusters",  
 xlab = "num of clusters",  
 ylab = "square error")



Видим, что по критерию среднеквадратичной ошибки наилучшее число кластеров: 4

**Поиск оптимального числа кластеров с помощью NbClust**

С помощью пакета NbClust можно найти оптимальную схему объединения в кластеры, используя 30 индексов. При этом происходит перебор различных комбинаций числа групп, метрик дистанции и методов кластеризации. Вывод об оптимальном числе классов делается с помощью голосования: берется то число кластеров, за которое “проголосовало” большинство критериев.

# Finding optimal N of clusters  
NbClust(  
 data = newM,  
 diss = NULL,  
 distance = "euclidean",  
 min.nc = min\_num\_of\_clusters,  
 max.nc = max\_num\_of\_clusters,  
 method = "median",  
 index = "all"  
)

\* Among all indices:   
\* 9 proposed 2 as the best number of clusters   
\* 4 proposed 3 as the best number of clusters   
\* 11 proposed 4 as the best number of clusters   
  
 \*\*\*\*\* Conclusion \*\*\*\*\*   
   
\* According to the majority rule, the best number of clusters is 4

***Видим, что наш результат совпал с результатом NbClust.***

#### **Задача классификации**

Имеется множество объектов , конечное множество ответов .

Задана выборка и множество известных ответов , вектор – набор признаков, совокупность упорядоченных пар “объект-ответ” – обучающая выборка.

Ставится задача построить решающее правило , которое приближало бы функцию на всем множестве (построить алгоритм, классифицирующий произвольный объект из исходного множества).

#### **Задание**:

Имеется таблица данных о качестве белых вин (Файл ). Требуется методом деревьев по 90% данных построить классификатор и проверить его на 10% приведенных данных. Сделать выводы.

#### **Решение**:

Описание данных: 4898 наблюдений, 12 переменных. Качество вина оценивается переменной , значения которой от 0 (плохое вино) до 10 (самое лучшее вино). Следующая гистограмма отображает исходные данные.

data.wine <- read.table('data/winequality-white.csv',   
 sep=';',   
 header=TRUE,   
 na.strings="NA",  
 stringsAsFactors=T)  
  
data.wine$quality <- as.factor(data.wine$quality)  
  
data.wine[, -dim(data.wine)[2]] <- scale(data.wine[, -dim(data.wine)[2]])  
  
element\_samples <- summary(data.wine$quality)  
#barplot(element\_samples, col = "peachpuff1")

[**Гистограмма распределения вин по качеству**]

Видно, что больше всего имеется сведений о вине среднего качества, а о вине низшего и высшего сорта известно мало. Вообще, для задачи классификации такое распределение исходных данных является очень плохим. Классификатор, построенный по этим данным, будет плохо работать.

Перемешаем данные, разделим их на две группы – тренировочную (90%) и тестовую (10%). Получим следующие гистограммы для тренировочной и тестовой выборок.

element\_samples <- summary(data.train$quality)#apply(data.train[c(-10)] != 0, 2, sum)  
barplot(element\_samples, col = "peachpuff1")  
abline(h = nrow(data.train), lty = 2, col = 2)  
title(main = "Train set types number")  
  
element\_samples <- summary(data.test$quality)#apply(data.test[c(-10)] != 0, 2, sum)  
barplot(element\_samples, col = "peachpuff1")  
abline(h = nrow(data.test), lty = 2, col = 2)  
title(main = "Test set types number")

[**Гистограммы выборок**]

Построим дерево решений при помощи rpart. Полученное дерево решений представлено на иллюстрации ниже.

tree <- rpart(quality ~., data.train)  
rpart.plot(tree,   
 type=4,  
 extra=101,   
 box.palette="GnBu",  
 branch.lty=3,   
 shadow.col="gray",   
 nn=TRUE  
)  
predict.test <- predict(tree, data.test, type = "class")  
predict.train <- predict(tree, data.train, type = "class")  
  
result.test <- table(data.test$quality, predict.test)  
result.train <- table(data.train$quality, predict.train)  
  
accuracy.test <- sum(diag(result.test)) / sum(result.test)  
accuracy.train <- sum(diag(result.train)) / sum(result.train)

[**Дерево классификатора**]

Полученное дерево охватывает не все категории вин из исходных данных, а только 5, 6 и 7. Из-за этого результат применения к тестовой выборке ожидается неудовлетворительным.

Применим дерево решений сначала к исходным данным, используя predict. Полученную классификацию вин сравниваем с исходной.

* Для тренировочной выборки точность классификации составляет 0.5673469
* Для тестовой выборки точность классификации составляет 0.5383394

Полученный результат следует признать плохим, поскольку дерево решений правильно классифицирует чуть больше 50% данных.