**Лабораторная работа №2**

**“Метод поиска ближайшего соседа”**

по дисциплине “Статическое моделирование случайных процессов и систем”

|  |  |
| --- | --- |
| Выполнил  студент гр. 33504/4 | Череповский Д.К. |
| Руководитель | Селин И.А. |

**Оглавление**

[**Задание** 3](#_Toc508974615)

[**Ход работы:** 3](#_Toc508974616)

[**Код программы:** 6](#_Toc508974617)

# **Задание**

1. Исследуйте, как объем обучающей выборки и количество тестовых данных, влияет на точность классификации или на вероятность ошибочной классификации в примере крестики-нолики и спам.

2. Постройте классификатор для обучающего множества **Glass**, данные которого характеризуются 10-ю признаками:

1. Id number: 1 to 214; 2. RI: показатель преломления; 3. Na: сода (процент содержания в соответствующем оксиде); 4. Mg; 5. Al; 6. Si; 7. K; 8. Ca; 9. Ba; 10. Fe.

Классы характеризуют тип стекла:

(1) окна зданий, плавильная обработка

(2) окна зданий, не плавильная обработка

(3) автомобильные окна, плавильная обработка

(4) автомобильные окна, не плавильная обработка (нет в базе)

(5) контейнеры

(6) посуда

(7) фары

Посмотрите заголовки признаков и классов. Перед построением классификатора необходимо также удалить первый признак Id number, который не несет никакой информационной нагрузки. Это выполняется командой **glass <- glass[,-1]**.

Постройте графики зависимости ошибки классификации от значения k и от типа ядра.

Исследуйте, как тип метрики расстояния (параметр **distance**) влияет на точность классификации.

Определите, к какому типу стекла относится экземпляр с характеристиками

RI =1.516 Na =11.7 Mg =1.01 Al =1.19 Si =72.59 K=0.43 Ca =11.44 Ba =0.02 Fe =0.1

Определите, какой из признаков оказывает наименьшее влияние на определение класса путем последовательного исключения каждого признака.

3. Для построения классификатора используйте заранее сгенерированные обучающие и тестовые выборки, хранящиеся в файлах svmdata4.txt, svmdata4test.txt. Найдите оптимальное значение k, обеспечивающее наименьшую ошибку классификации. Посмотрите, как выглядят данные на графике, используя функцию

**plot(mydata.train$X1, mydata.train$X2, pch=21, bg=c("red","blue") [unclass(mydata.train$Colors)], main="My train data")**

# **Ход работы:**

1. Было исследовано как объём обучающей выборки и количество тестовых данных влияют на вероятность ошибочной классификации:

Крестики-нолики

A\_predicted negative positive

negative 207 96

positive 89 471

A\_predicted negative positive

negative 217 42

positive 41 467

A\_predicted negative positive

negative 199 8

positive 29 435

A\_predicted negative positive

negative 164 6

positive 31 374

A\_predicted negative positive

negative 146 1

positive 11 321

A\_predicted negative positive

negative 113 2

positive 11 258

A\_predicted negative positive

negative 82 1

positive 5 200

A\_predicted negative positive

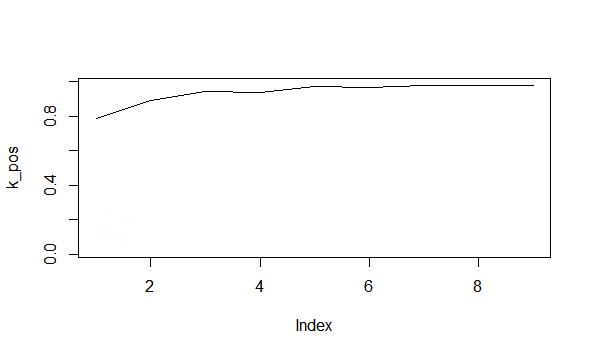
negative 59 1

positive 3 129

A\_predicted negative positive

negative 29 0

positive 2 65



Спам

nonspam spam

nonspam 6 0

spam 7 7

nonspam spam

nonspam 174 14

spam 130 202

nonspam spam

nonspam 361 15

spam 256 388

nonspam spam

nonspam 491 35

spam 420 574

nonspam spam

nonspam 648 40

spam 566 766

nonspam spam

nonspam 864 53

spam 671 932

nonspam spam

nonspam 999 67

spam 806 1148

nonspam spam

nonspam 1078 75

spam 1059 1308

nonspam spam

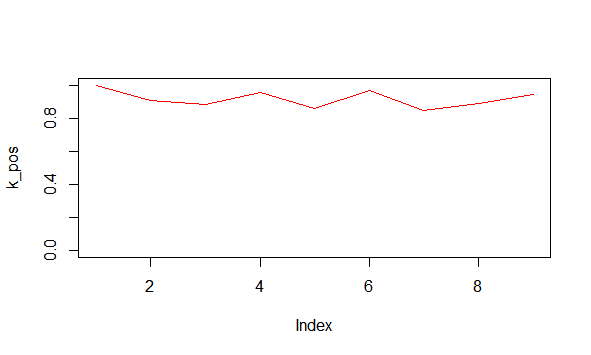
nonspam 1412 84

spam 1022 1502

nonspam spam

nonspam 1049 87

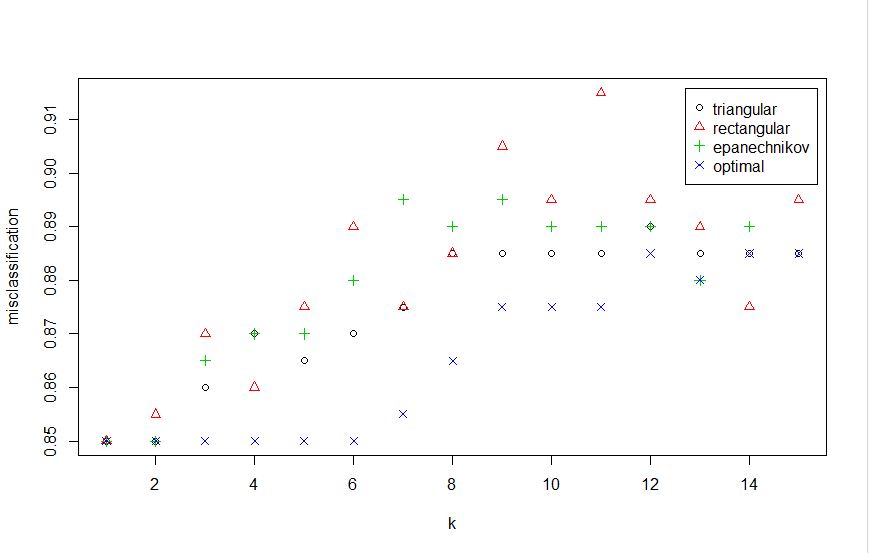
spam 1693 1691



(крестики нолики) При увеличении объема обучающей выборки точность классификации увеличивается.

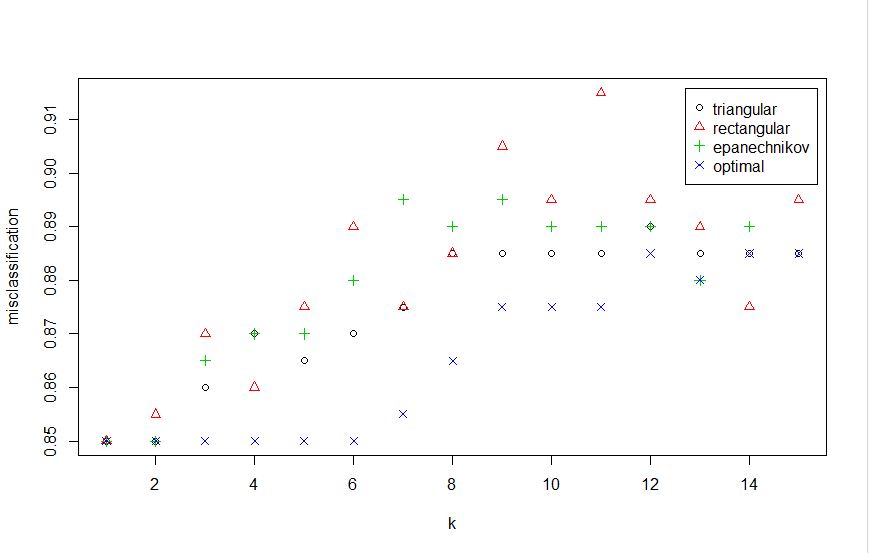
(спам) При увеличении объема обучающей выборки точность классификации меняется без явно выраженной зависимости.

1. Был сгенерирован фрейм (glass), с 10 признаками и 7 классами;
2. Построены графики зависимости ошибки классификации от значения k и от типа ядра;

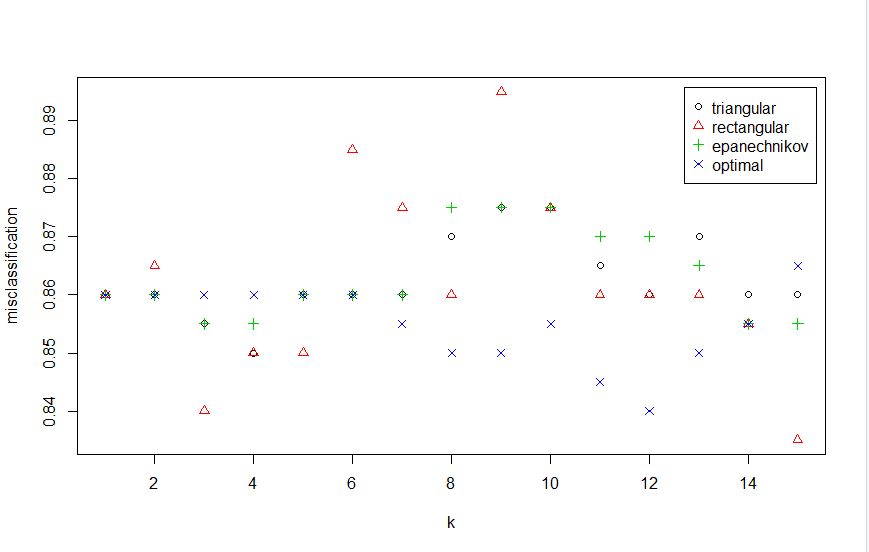


1. Произведен анализ, как метрики расстояния влияют на точность классификации:

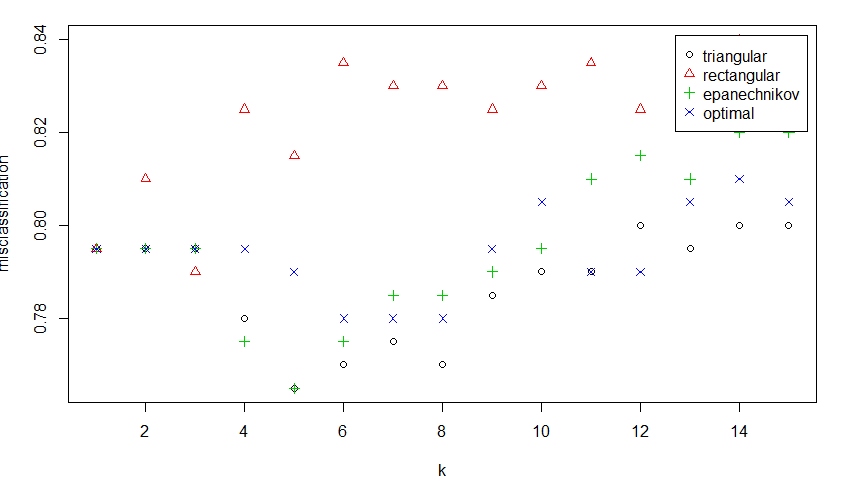
Distance = 1:



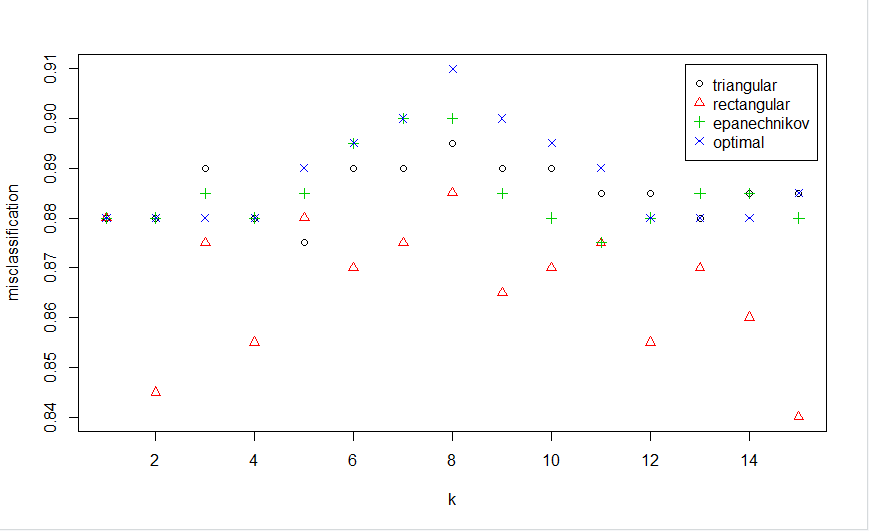
Distance = 2:



Distance = 3:

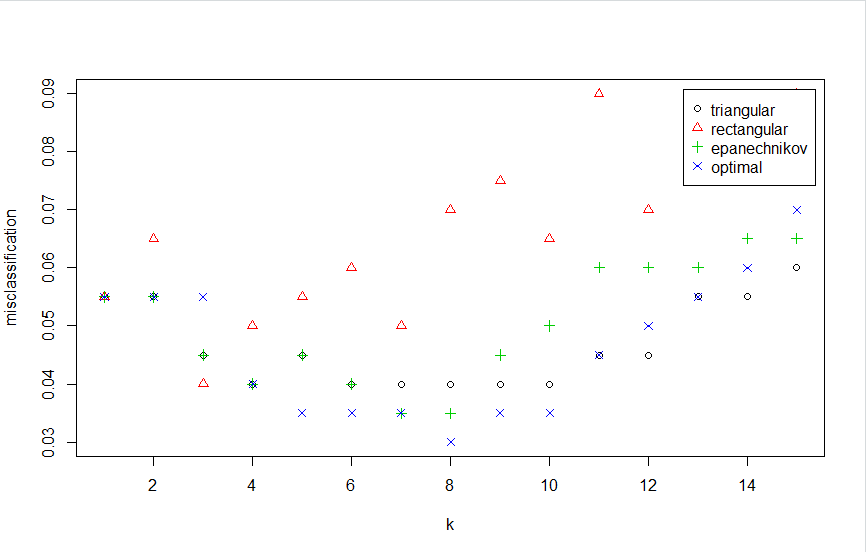


Distance = 4:



Из графиков видно, что метрики классификации влияют на точность классификации.

1. Экземпляр с характеристиками RI =1.516 Na =11.7 Mg =1.01 Al =1.19 Si =72.59 K=0.43 Ca =11.44 Ba =0.02 Fe =0.1 относиться к (5) типу стекла, контейнеры.
2. Признак Ca, оказывает наименьшее влияние на определение класса, при определении признака, последовательно обнуляем признаки и сверяем результат с изначальным.
3. Для построения классификатора используем svmdata4.txt и svmdata4test.txt и подбираем оптимальное значение k.

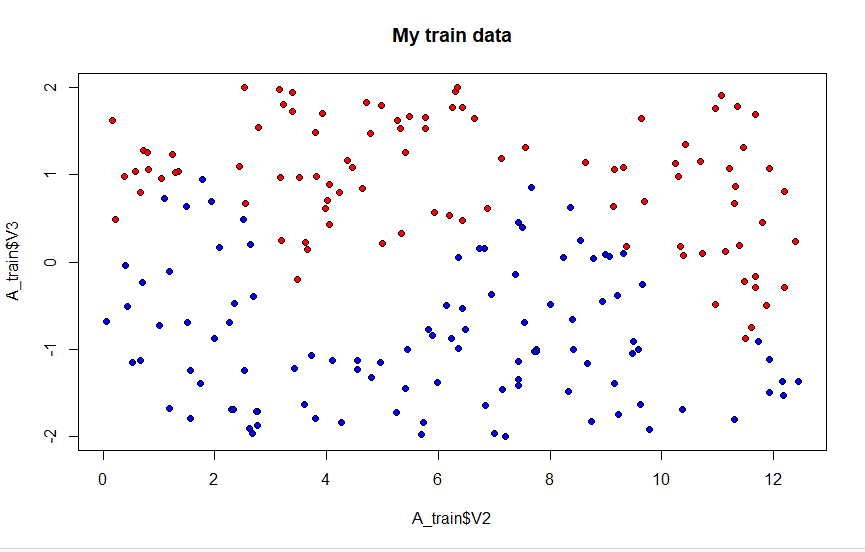


Из графика видно, что оптимальное k = 8

Используя:

plot(A\_train$V2, A\_train$V3, pch=21, bg=c("red","blue") [unclass(A\_train$V4)], main="My train data")

Получаем:



A\_train$V3 -> X1;

A\_train$V2 -> X2;

# **Код программы:**

##исследуем, как объём обучающей выборки и количество тестовых данных влияют на вероятность ошибочной классификации

library(kknn)

A\_raw <- read.table("C:/Users/gdk17/OneDrive/Рабочий стол/универ/machine\_learning/Lab2\_Knn/Tic\_tac\_toe.txt", sep = ",", stringsAsFactors = TRUE)

n <- dim(A\_raw)[1]

set.seed(12345)

A\_rand <- A\_raw[ order(runif(n)), ]

for(i in seq(0.1, 0.9, by = 0.1))

{

nt <- as.integer(n\*i)

A\_train <- A\_rand[1:nt, ]

A\_test <- A\_rand[(nt+1):n, ]

A\_classifier <- train.kknn(V10 ~ ., A\_train, kmax = 15,kernel = c("triangular", "rectangular", "epanechnikov", "optimal"), distance = 1)

A\_predicted <- predict(A\_classifier, A\_test)

t <- table(A\_predicted, A\_test$V10)

print(t)

}

library(kknn)

library(kernlab)

data(spam)

#spam[0:1, ]

size <- length(spam)

k\_pos <- integer(9)

k\_neg <- integer(9)

k\_pos\_test <- integer(9)

k\_neg\_test <- integer(9)

for (i in 1:9) {

idx <- sample(1:dim(spam)[1], size \* i / 10)

spamtrain <- spam[-idx, ]

spamtest <- spam[idx, ]

kknn\_predicted <- kknn(type ~ ., spamtrain, spamtest)

kknn\_res\_table <- table(kknn\_predicted$fitted.values, spamtest$type)

pos <- kknn\_res\_table['nonspam', 'nonspam'] + kknn\_res\_table['spam', 'spam']

neg <- kknn\_res\_table['nonspam', 'spam'] + kknn\_res\_table['spam', 'nonspam']

k\_pos[i] <- pos / length(kknn\_predicted$fitted.values)

k\_neg[i] <- neg / length(kknn\_predicted$fitted.values)

}

png(file = 'k\_spam.jpg')

plot(k\_pos, type = 'l', col="red", ylim=range(c(k\_pos, k\_neg)))

lines(k\_neg, col="green", ylim=range(c(k\_pos, k\_neg)))

dev.off()

## создаем фрейм

id <- seq(1,214)

RI <- runif(214, 0, 3)

Na <- runif(214, 0, 20)

Mg <- runif(214, 0, 3)

Al <- runif(214, 0, 3)

Si <- runif(214, 0, 100)

K <- runif(214, 0, 2)

Ca <- runif(214, 0, 20)

Ba <- runif(214, 0, 1)

Fr <- runif(214, 0, 2)

class <- factor(c(rep((1), 30), rep((2), 30), rep((3),30), rep((4),30), rep((5),30), rep((6),30), rep((7),34)))

glass = data.frame(id, RI, Na, Mg, Al, Si, K, Ca, Ba, Fr, class)

glass <- glass[,-1]

##построение графиков ошибок от K(ядра), анализ oшибок при варьировании distace

library(kknn)

glass.learn <- glass[1:200,]

glass.valid <- glass[-c(1:200),]

fit.kknn <- kknn(class ~ ., glass.learn, glass.valid)

fit.train1 <- train.kknn(class ~ ., glass.learn, kmax = 15, kernel = c("triangular", "rectangular", "epanechnikov", "optimal"), distance = 1)

fit.train2 <- train.kknn(class ~ ., glass.learn, kmax = 15, kernel = c("triangular", "rectangular", "epanechnikov", "optimal"), distance = 4)

#plot(fit.train1)

plot(fit.train2)

##запуск knn с одним ядром, построение диограммы

m <- dim(glass)[1]

val <- sample(1:m, size = round(m/3), replace = FALSE,prob = rep(1/m, m))

glass.learn <- glass[-val,]

glass.valid <- glass[val,]

glass.kknn <- kknn(class~., glass.learn, glass.valid, distance = 1, kernel = "triangular")

summary(glass.kknn)

fit <- fitted(glass.kknn)

table(glass.valid$class, fit)

pcol <- as.character(as.numeric(glass.valid$class))

pairs(glass.valid[1:10], pch = pcol, col = c("green3", "red")

[(glass.valid$class != fit)+1])

##получение класса по заданным признакам

RI <- c(1.51)

Na <- c(11.7)

Mg <- c(1.01)

Al <- c(1.19)

Si <- c(72.59)

K <- c(0.43)

Ca <- c(0.0)

Ba <- c(0.02)

Fr <- c(0.10)

class <- c(1)

example <- data.frame(RI, Na, Mg, Al, Si, K, Ca, Ba, Fr, class)

example.kknn <- kknn(class~., glass.learn, example, distance = 1, kernel = "triangular")

summary(example.kknn)

fit <- fitted(example.kknn)

table(example$class, fit)

pcol <- as.character(as.numeric(example$class))

pairs(glass.valid[1:10], pch = pcol, col = c("green3", "red")

[(glass.valid$class != fit)+1])

library(kknn) #подбираем оптимальное значение k для svmdata4.txt

A\_train <- read.table("C:/Users/gdk17/OneDrive/Рабочий стол/универ/machine\_learning/Lab2\_Knn/svmdata4.txt")

A\_test <- read.table("C:/Users/gdk17/OneDrive/Рабочий стол/универ/machine\_learning/Lab2\_Knn/svmdata4test.txt")

A\_train <- A\_train[,-1]

A\_test <- A\_test[,-1]

A.learn <- A\_train

A.valid <- A\_test

fit.kknn <- kknn(V4 ~ ., A.learn, A.valid)

fit.train2 <- train.kknn(V4 ~ ., A.learn, kmax = 15, kernel = c("triangular", "rectangular", "epanechnikov", "optimal"), distance = 1)

plot(fit.train2)

plot(A\_train$V2, A\_train$V3, pch=21, bg=c("red","blue") [unclass(A\_train$V4)], main="My train data")