Classification Tree (CART)

Версия 16.10.2016

Обзор пакетов, реализующих или использующих другие деревья классификации
https://www.r-bloggers.com/a-brief-tour-of-the-trees-and-forests/
Внимание: обзор написан в 2013 году, немного устарел

Impurity measures (Меры загрязненности)

Пусть в задаче k классов.

Энтропия (entropy)

$$H_1 = -\sum_{j=1} p_j \cdot \log_2 p_j$$

Индекс Джини (Gini index)

$$H_2 = 1 - \sum_{j=1} p_j^{\,2} = \sum_{j=1} p_j \cdot (1 - p_j)$$

Ошибки классификации (classification error)

$$H_3 = 1 - max(p_j)$$

где p_j - вероятность принадлежать классу j. (На практике — доля объектов класса j в узле)

Зададимся некоторым разделением узла на 2 потомка. Тогда увеличение чистоты узлов измеряется как

$$\Delta H = H_{родителя} - \left(\frac{n_{левый}}{n_{родителя}} \cdot H_{левый} + \frac{n_{правый}}{n_{родителя}} \cdot H_{правый}\right)$$

Основные понятия
==========

узел (node)
- родительский (parent node)
- потомок(child node)
- конечный узел (лист дерева) (final node)

пороговое значение

Все узлы пронумерованы.
Номера узлов не идут подряд, а следуют правилу

"узлы-потомки узла номер n имеют номера 2*n и 2*n+1"

При расщеплении, когда условие выполнено, наблюдение попадает в левый узел,
см. summary.

oblique trees

пространство разделяется произвольными гиперплоскостями, снято требование, чтобы гиперплоскость была перпендикулярна одной из осей координат

Пакет R, реализующий процедуру
oblique.tree (раньше был глючным...)

oblivious trees

В пределах одного слоя дерева все расщепления производятся по одной переменной.

Пакет R, реализующий процедуру: отсутствует (?)
Говорят, реализован в Matrixnet...

Чтение данных

==========

```
#  Шаг 0. Прочитаем данные

credit.01 <- read.table("Credit_ru_1.csv", header=T, sep=";")


#  Проверка: импортировали правильно?

head(credit.01)
dim(credit.01)
#  [1] 323   5

names(credit.01)
#  [1] "кредит"   "класс"    "з_плата"  "возраст"  "кр_карта"

class(credit.01[ , 1])
class(credit.01[ , 2])
class(credit.01[ , 3])
class(credit.01[ , 4])
class(credit.01[ , 5])
#  [1] "integer"


#  Теперь надо принимать решение.
#  Переменные должны быть факторами или нет?
#  То есть определяемся со шкалой, в которой измерены переменные.
```

 # Вариант 1. Все переменные измерены в порядковой шкале.

```
library(rpart)


credit.01.res <- rpart(кредит ~ класс + з_плата + возраст + кр_карта,
    data = credit.01, method="class",
    control=rpart.control(minsplit=10, minbucket=5, maxdepth=6) )
```

credit.01.res
: файл с результатами

rpart
: процедура, которая строит дерево классификации

кредит ~ .
: зависимая переменная - "кредит", независимые переменные - все остальные

data = credit.01
: анализируются переменные из таблицы данных credit.01

method="class"
: строится дерево классификации, а не дерево регрессии

model = TRUE
: список с результатами , x = FALSE, y = FALSE, weights = weights.zzz, control=rpart.control(minsplit=10, minbucket=5, maxdepth=6))

Графика: дерево классификации

 # 1 вариант: некрасивое дерево

```
plot(credit.01.res)
text(credit.01.res, use.n=T)
```

 # 2 вариант: красивое дерево: много вариантов

```
# Необходима библиотека rpart.plot

library(rpart.plot)


rpart.plot(credit.01.res)

# Красиво, но неинформативно

rpart.plot(credit.01.res, type=0)
rpart.plot(credit.01.res, type=1)
rpart.plot(credit.01.res, type=2)
rpart.plot(credit.01.res, type=3)
rpart.plot(credit.01.res, type=4)
```

 # Красиво, и информативно

```
rpart.plot(credit.01.res, type=2, extra = 1)


extra.val <- 109

rpart.plot(credit.01.res, type=2, extra = extra.val)
```

Использование дерева

```
predict(credit.01.res, credit.01[ , -1], type="class")

predict(credit.01.res, credit.01[ , -1])[ , 2]


table(credit.01[ , 1], predict(credit.01.res, credit.01[ , -1]),  type="class"))
```

Разберемся с результатами работы команды print(...)

Команды

credit.01.res
и
print(credit.01.res)
выдают один и тот же результат.
Но во втором случае имеются дополнительные возможности
управления выводом. Смотри описание опций.
Например, в команде print можно сократить число знаков после запятой.
print(credit.01.res, digits = 2)

```
n= 323

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 323 155 0 (0.520123839 0.479876161)
   2) з_плата< 1.5 165  22 0 (0.866666667 0.133333333)
     4) возраст< 2.5 158  15 0 (0.905063291 0.094936709) *
     5) возраст>=2.5 7   0 1 (0.000000000 1.000000000) *
   3) з_плата>=1.5 158  25 1 (0.158227848 0.841772152)
```

```
    6) возраст< 1.5 49  24 1 (0.489795918 0.510204082)
     12) класс< 2.5 43  19 0 (0.558139535 0.441860465) *
     13) класс>=2.5 6   0 1 (0.000000000 1.000000000) *
    7) возраст>=1.5 109   1 1 (0.009174312 0.990825688) *
```

Расшифровка информации

node), split, n, loss, yval, (yprob)

node == номер узла. Номера не идут подряд, а следуют правилу
"узлы-потомки узла номер n имеют номера 2*n и 2*n+1"
split == условие, которое должно быть выполнено,
чтобы наблюдение попало в узел при расщепления узла-родителя
n == количество наблюдений из обучающей выборки, попавших в узел
loss == число наблюдений другого класса, попавших в узел (число ошибок)
yval == какой класс приписывается наблюдению из данного узла. Определяется тем, наблюдения какого класса составляют в узле большинство.
(yprob) == Доля объектов из каждого класса в узле. Интерпретируется как вероятность того, что объект, попавший в узел, будет принадлежать классу.
Например, проверим для второго узла, где (yprob) = (0.8667 0.1333)

```
> 22/165
[1] 0.1333333
> (165-22)/165
[1] 0.8666667
```

<u>Разберемся с результатами работы команды summary(...)</u>

summary(credit.01.res)

```
Call:
rpart(formula = кредит ~ ., data = credit.01, method = "class",
   control = rpart.control(minsplit = 10, minbucket = 5, maxdepth = 6))
```

Напоминание о виде выполняемой команды

n= 323

Число наблюдений в наборе данных

```
        CP nsplit  rel error     xerror       xstd
1 0.69677419      0 1.0000000 1.0000000 0.05792787
2 0.04516129      1 0.3032258 0.3032258 0.04088561
3 0.01612903      2 0.2580645 0.2903226 0.04015094
4 0.01000000      4 0.2258065 0.2774194 0.03938948
```

без комментариев

```
Variable importance
 з_плата  возраст    класс кр_карта
      38       32       28        2
```

Веса предикторов, нормализованы так, чтобы в сумме получается 100

на примере первого узла

```
Node number 1: 323 observations,    complexity param=0.6967742
  predicted class=0  expected loss=0.4798762  P(node) =1
    class counts:   168   155
   probabilities: 0.520 0.480
  left son=2 (165 obs) right son=3 (158 obs)
  Primary splits:
      з_плата  < 1.5 to the left,  improve=81.0164500, (0 missing)
      возраст  < 1.5 to the left,  improve=73.3493500, (0 missing)
      класс    < 2.5 to the right, improve=53.7945900, (0 missing)
      кр_карта < 0.5 to the left,  improve= 0.1134861, (0 missing)
  Surrogate splits:
      класс    < 2.5 to the right, agree=0.842, adj=0.677, (0 split)
      возраст  < 1.5 to the left,  agree=0.765, adj=0.519, (0 split)
      кр_карта < 0.5 to the left,  agree=0.542, adj=0.063, (0 split)
```

Node number 1:

номер узла

323 observations

число наблюдений, попавших в узел

complexity param=0.6967742

??? не знаю

predicted class=0

какой класс приписывается наблюдению из данного узла.

Определяется тем, наблюдения какого класса составляют в узле большинство.

expected loss=0.4798762

доля ошибок среди наблюдений, попавших в узел, доля неверно расклассифицированных наблюдений. Вычисляется по вероятностям (см. ниже)

P(node) =1

какая доля наблюдений попала в узел (доля от общего числа наблюдений)

class counts: 168 155

Для каждого класса число наблюдений из этого класса, попавших в узел

probabilities: 0.520 0.480

Для каждого класса доля наблюдений из этого класса, попавших в узел

left son=2 (165 obs) right son=3 (158 obs)

Номер узла-потомка и (в скобках) число наблюдений,
попавших в узлы-потомки после расщепления рассматриваемого узла
Если узел конечный, информация не выводится (потомков нет)

Primary splits:

```
   з_плата  < 1.5 to the left,  improve=81.0164500, (0 missing)
   возраст  < 1.5 to the left,  improve=73.3493500, (0 missing)
   класс    < 2.5 to the right, improve=53.7945900, (0 missing)
   кр_карта < 0.5 to the left,  improve= 0.1134861, (0 missing)
```

Какие варианты расщепления давали наибольшее improvement (в убывающем порядке). Победитель, то расщепление, на котором остановилась процедура, приводится в первой строке.

Surrogate splits:

```
   класс    < 2.5 to the right, agree=0.842, adj=0.677, (0 split)
   возраст  < 1.5 to the left,  agree=0.765, adj=0.519, (0 split)
   кр_карта < 0.5 to the left,  agree=0.542, adj=0.063, (0 split)
```

Если присутствуют пропущенные значения, то код пропущенного значения считается отдельным классом, и для него строятся сурогатные расщепления. Зачем сурогатные расщепления приводятся

Почему приводится при анализе данных без пропусков - не знаю.

 # Вариант 2. Все переменные измерены в номинальной шкале.

```
#  Преобразуем данные

credit.02 <- data.frame(as.factor(credit.01[ ,1]),  as.factor(credit.01[ ,2]),
                        as.factor(credit.01[ ,3]),as.factor(credit.01[ ,4]),as.factor(credit.01[ ,5]))

#  Зададим имена
names(credit.02) <- c("кредит", "класс", "з_плата", "возраст", "кр_карта")

# Зададим уровни факторов

levels(credit.02[,1]) <- c( "Низкий", "Высокий")

levels(credit.02[,2]) <- c( "Management", "Professional", "Clerical", "Skilled Manual", "Unskilled")

levels(credit.02[,3]) <- c( "Еженедельно", "Ежемесячно")
levels(credit.02[,4]) <- c( "Молод (< 25)", "Средний(25-35)", "Пожилой( > 35)")
levels(credit.02[,5]) <- c( "Нет", "Да")

#  кредит
#  0 = "Низкий"
#  1 = "Высокий"

#  класс
#  "Management", "Professional", "Clerical", "Skilled Manual", "Unskilled"

#  з_плата
#  "Еженедельно", "Ежемесячно"

#  возраст
#  "Молод (< 25)", 2 = "Средний(25-35)", "Пожилой( > 35)"

#  кр_карта
#  "Нет", "Да"



library(rpart)

credit.03.res <- rpart(кредит ~ класс + з_плата + возраст + кр_карта, data = credit.02,
                       method="class", control=rpart.control(minsplit=10, minbucket=5, maxdepth=6) )
```