Árvores de Classificação e Regressão

Paulo C. Marques F. e Hedibert F. Lopes

Sexta-feira, 10 de Novembro de 2017

Insper

Continuamos no contexto de aprendizagem supervisionada.

Temos um vetor de p variáveis preditoras $X = (X_1, \ldots, X_p) \in \mathbb{R}^p$.

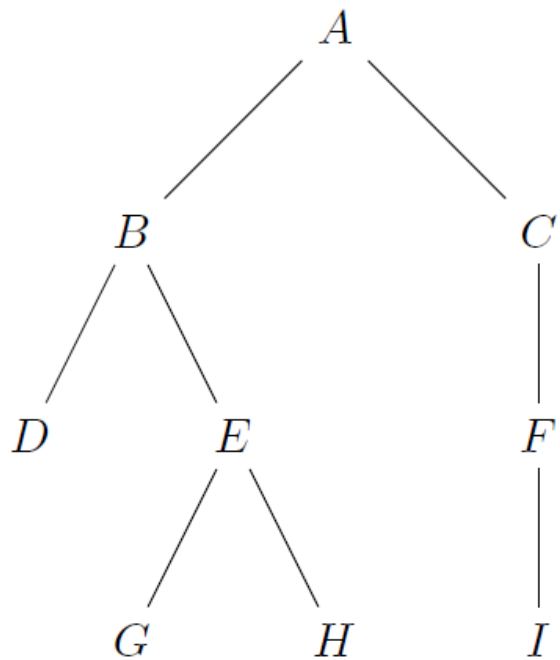
Por exemplo, $X_1 =$ "peso" e $X_2 =$ "maior comprimento" do peixe.

E uma variável resposta $Y \in \{1, \ldots, c\}$.

Por exemplo, "salmão" (0) e "robalo" (1).

Terminologia

Na *árvore binária* abaixo, o nó A é a *raiz*, e os nós D, G, H e I são as *folhas* (ou *nós terminais*).

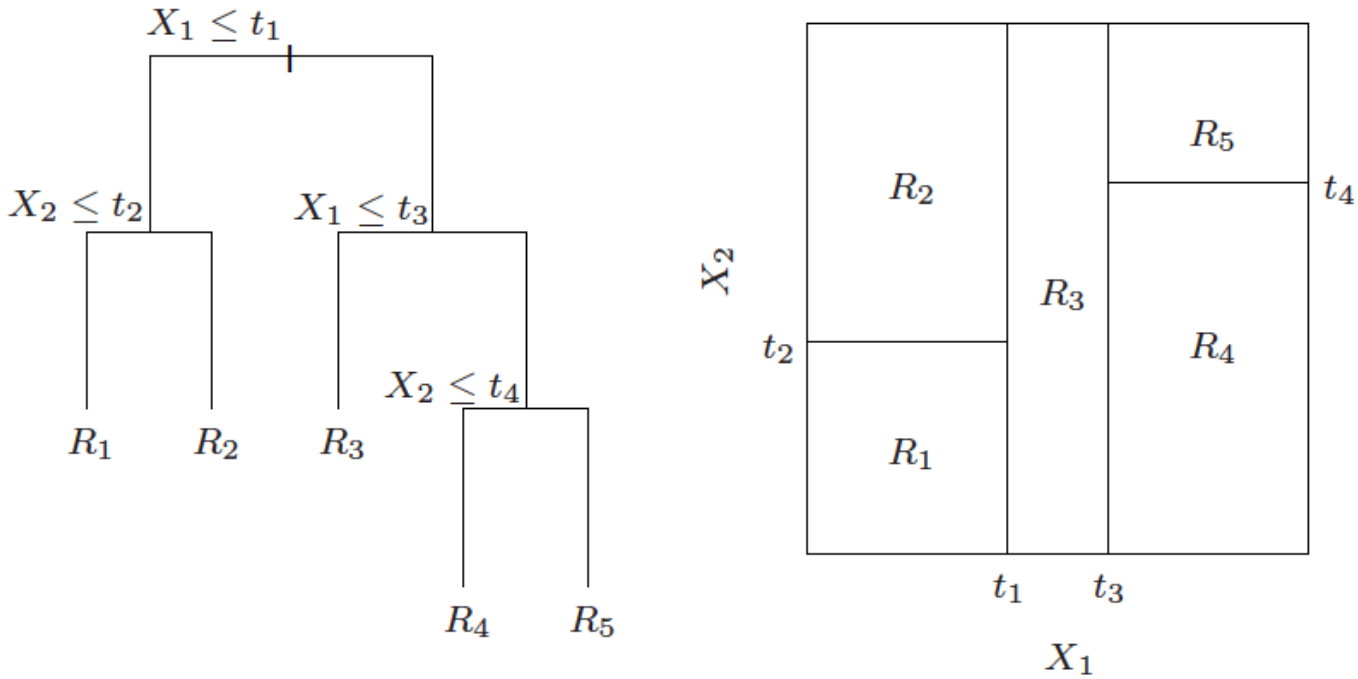


A árvore tem quatro níveis de *altura*. A *profundidade* do *nó interno* D é igual a dois.

Regiões determinadas pela árvore de classificação

Insper

Na figura abaixo temos uma árvore de classificação T e a partição do espaço das preditoras nas regiões correspondentes.



Cada nó não terminal de T define uma divisão ("split") em uma das preditoras. Cada folha de T corresponde a uma região retangular R_j.

Como classificar?

Insper

Suponha que a árvore de classificação T do slide anterior nos foi dada *ex machina*.

Suponha que temos dados de treinamento

$$(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^2 \times \{0, 1\},$$

para $i = 1, \ldots, n$.

Tendo em mãos um novo $x \in \mathbb{R}^2$, inicialmente determinamos a qual região R_j pertence este vetor de preditoras x.

Note que não precisamos examinar todas as regiões: basta descer a árvore a partir da raiz para saber a qual região x pertence.

Isso é uma tremenda vantagem do ponto de vista computacional.

Uma vez determinada a região R_j a qual x pertence, classificamos este exemplar como sendo da classe mais frequente entre os dados de treinamento pertencentes à mesma região R_j (voto da maioria).

Formalizando o classificador

Insper

Dados de treinamento $(x_i, y_i) \in \mathbb{R}^p \times \{1, \ldots, c\}$, para $i = 1, \ldots, n$.

A árvore de classificação T define as regiões retangulares $R_1, \ldots, R_m$ que particionam o espaço das preditoras $\mathbb{R}^p$.

Seja $n_j = \sum_{i=1}^{n} I_{R_j}(x_i)$ o número de preditoras dos dados de treinamento pertencentes à região R_j, para $j = 1, \ldots, m$.

A fração de exemplos da classe k na região R_j é igual a

$$\hat{p}_k(R_j) = \frac{1}{n_j} \sum_{\{i \,:\, x_i \in R_j\}} I_{\{k\}}(y_i),$$

para $k = 1, \ldots, c$.

A classe predita para a região R_j é $c_j = \arg\max_k \hat{p}_k(R_j)$, que é a proporção de exemplos (dados de treinamento) na região R_j que são da classe predominante. O classificador fica escrito como

$$\hat{\varphi}(x) = \sum_{j=1}^{m} c_j \, I_{R_j}(x).$$

Como construir uma árvore de classificação?

Insper

Como escolher cada uma das divisões ("splits")?

Lembrando que em cada divisão precisamos escolher uma das preditoras e o ponto de separação.

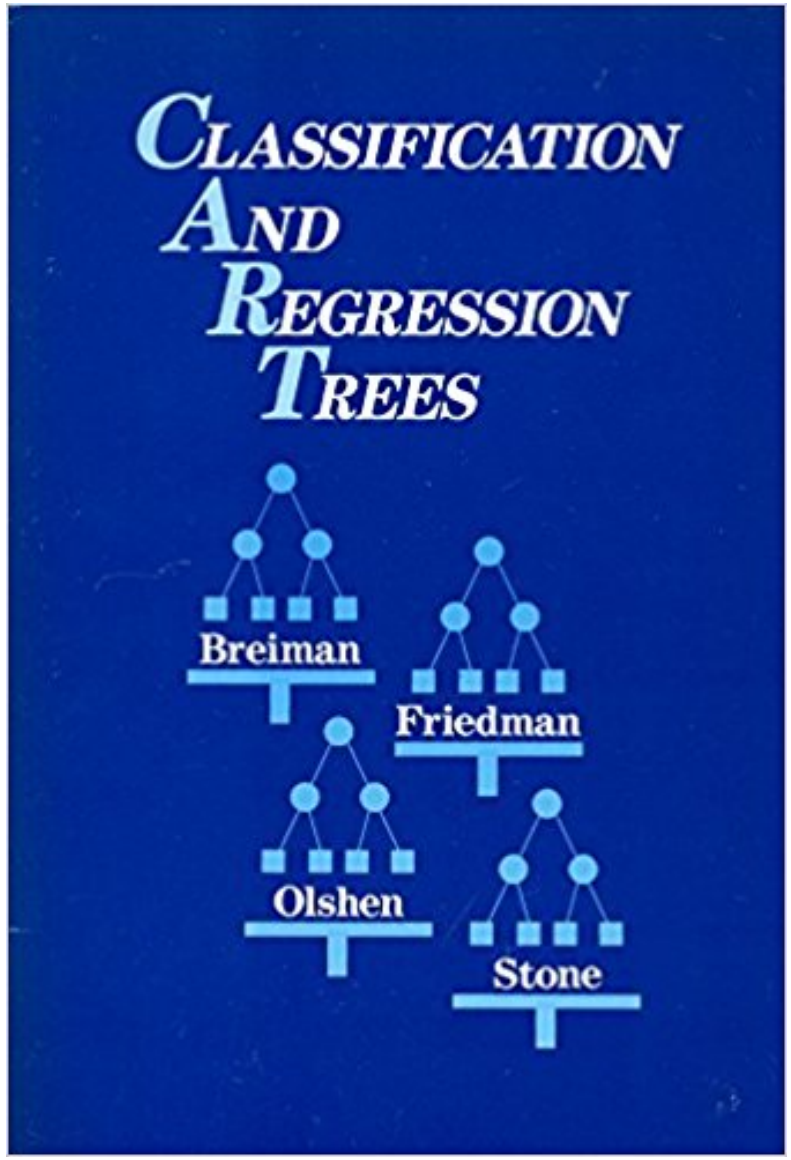
Que altura deve ter a árvore de classificação?

Note que se a árvore for binária e balanceada, então em cada nível temos aproximadamente $2^{\text{nível}-1}$ nós e para cada um deles podemos escolher uma das p preditoras para fazer a divisão.

Qual algoritmo utilizar?

Algoritmo CART (1)

Insper



Algoritmo CART (2)

Insper

CART: Classification and Regression Trees. Breiman el al. (1984).

O algoritmo CART começa na raiz da árvore e efetua uma divisão, criando dois nós no próximo nível da árvore.

Depois disso, descemos para o primeiro nível da árvore e repetimos o procedimento para os dois nós que foram criados.

Continuamos da mesma maneira nos níveis seguintes.

Em cada etapa, escolhemos a divisão que produz a maior queda no erro de classificação.

O algoritmo CART cresce uma árvore alta e poda alguns dos seus ramos no final do processo.

Algoritmo CART (3)

Insper

Formalmente, o algoritmo CART começa na raiz da árvore e define as regiões disjuntas

$$R_1 = \{X \in \mathbb{R}^p : X_j \leq t\} \quad \text{e} \quad R_2 = \{X \in \mathbb{R}^p : X_j > t\}.$$

Utilizando os dados de treinamento, fazemos a divisão escolhendo $\hat{j}$ e $\hat{t}$ tais que

$$(\hat{j}, \hat{t}) = \arg\min_{(j,t)} \left((1 - \hat{p}_{c_1}(R_1)) + (1 - \hat{p}_{c_2}(R_2)) \right),$$

em que

$$c_1 = \arg\max_{k=1,\ldots,c} \hat{p}_k(R_1)$$

é a classe dominante no retângulo R_1 e

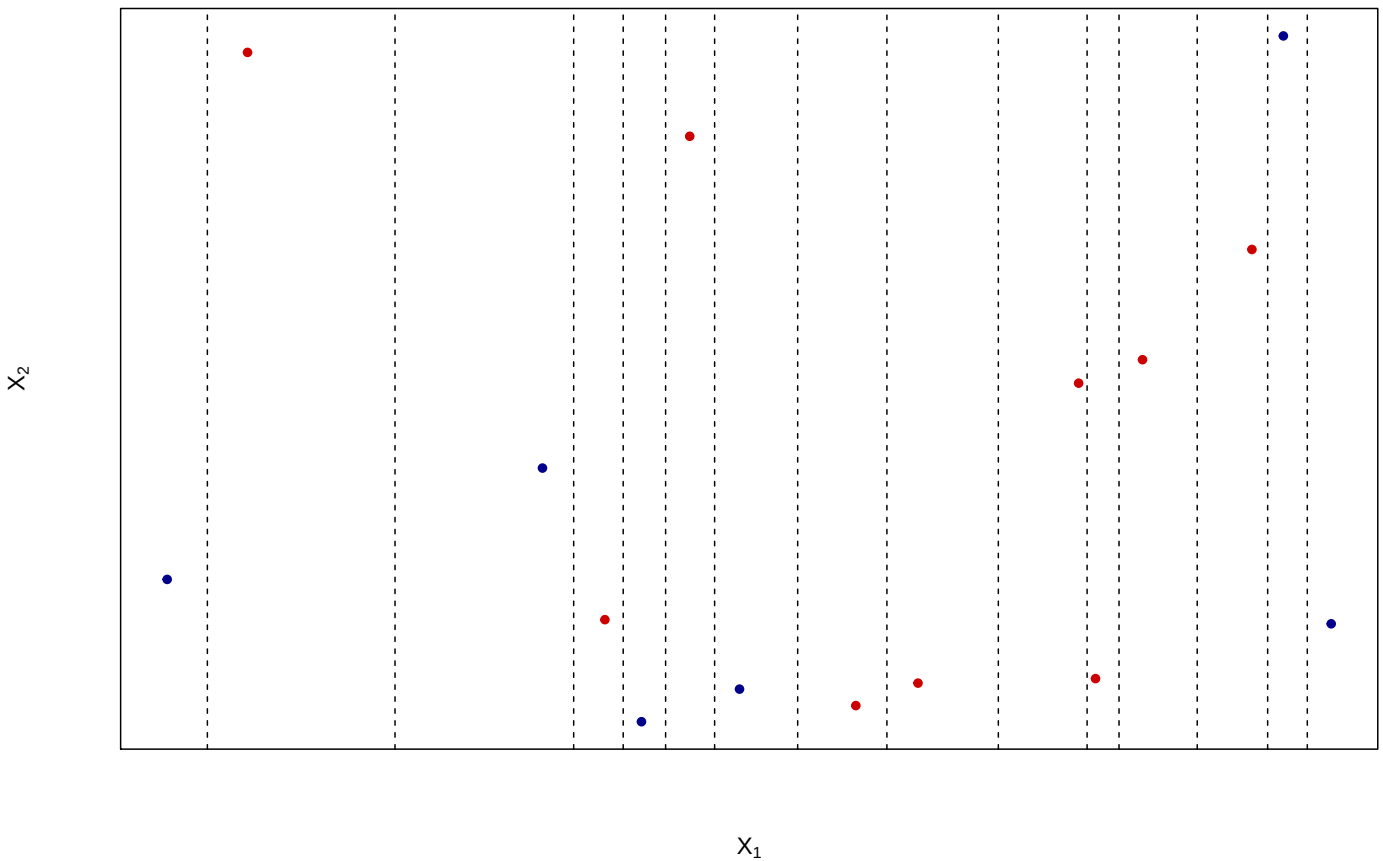
$$c_2 = \arg\max_{k=1,\ldots,c} \hat{p}_k(R_2)$$

é a classe dominante no retângulo R_2.

Algoritmo CART (4)

Insper

Note que para encontrar o ponto t de divisão de uma região retangular R_m precisamos considerar apenas $n_m - 1$ divisões da variável preditora X_j que estivermos examinando.



Algoritmo CART (5)

Insper

Procedemos de maneira análoga para os novos nós criados, até que seja satisfeito algum critério de parada; por exemplo, quando tivermos apenas dados de treinamento de uma certa classe na nova região gerada.

Este procedimento gera uma árvore T_0 que será podada: algumas de suas folhas serão colapsadas aos seus nós pais.

Para uma árvore de classificação T, denote por $|T|$ o número de suas folhas e, para $\alpha \geq 0$, defina

$$C_\alpha(T) = \sum_{j=1}^{|T|}(1 - \hat{p}_{c_j}(R_j)) + \alpha|T|.$$

O algoritmo CART escolhe a árvore T que minimiza $C_\alpha(T)$, escolhendo α por validação cruzada (geralmente em 5 ou 10 lotes).

Note que há uma forma de regularização contida na definição de C_α, uma vez que estamos penalizando árvores com muitas folhas.

Spam ≠ Ham

Insper

≠

Classificação de e-mails em *spam* e *ham* (1)

Insper

Os dados disponíveis em

http://ftp.ics.uci.edu/pub/machine-learning-databases/spambase/

apresentam informações sobre 4601 e-mails, entre os quais 1813 foram marcados como *spam* (e-mails ilegítimos, não solicitados, com conteúdo comercial ou duvidoso).

A partir destes dados de treinamento, queremos construir um classificador de e-mails que separe novas mensagens em duas categorias: *spam* (ilegítimo) ou *ham* (legítimo).

Classificação de e-mails em *spam* e *ham* (2)

Insper

Esta base de dados apresenta para cada e-mail o valor das seguintes variáveis preditoras.

1. Temos 48 variáveis com as porcentagens das ocorrências de palavras específicas, tais como `business`, `address`, `internet`, `free` etc.
2. Temos 6 preditoras com as porcentagens das ocorrências de caracteres específicos, tais como `;`, `$`, `!`.
3. O tamanho médio das sequências ininterruptas de letras maiúsculas (`CAPAVE`). O tamanho da maior sequência ininterrupta de letras maiúsculas (`CAPMAX`). A soma dos comprimentos das sequências ininterruptas de letras maiúsculas (`CAPTOT`).

Dados de treinamento

Insper

O primeiro passo é ler a base de dados de treinamento. A função `read.table()` retorna um objeto do tipo `data.frame`.

```
db <- read.table("./spambase/spambase.data", sep = ",", header = FALSE)

class(db)
```

```
## [1] "data.frame"
```

De posse dos dados, nomeamos adequadamente as colunas do `data.frame`.

```
colnames(db) <- c(
    # 1
    "make", "address", "all", "w_3d", "our", "over", "remove", "internet", "order", "mail",
    "receive", "will", "people", "report", "addresses", "free", "business", "email", "you",
    "credit", "your", "font", "w_000", "money", "hp", "hpl", "george", "w_650", "lab", "labs",
    "telnet", "w_857", "data", "w_415", "w_85", "technology", "w_1999", "parts", "pm", "direct",
    "cs", "meeting", "original", "project", "re", "edu", "table", "conference",
    # 2
    "ch_semi", "ch_lparen", "ch_lbrack", "ch_bang", "ch_dollar", "ch_hash",
    # 3
    "CAPAVE", "CAPMAX", "CAPTOT",
    # Spam = 1, Ham = 0
    "Class"
)
```

Crescendo a árvore de classificação (1)

Insper

Para facilitar a visualização dos resultados, recodificamos as classes dos e-mails de treinamento.

```
db$Class <- as.factor(ifelse(db$Class, "Spam", "Ham"))

table(db$Class)
```

```
##
##  Ham Spam
## 2788 1813
```

Note que forçamos esta coluna do `data.frame` a ser do tipo `factor` utilizando a função de conversão de tipo `as.factor()`. Esta conversão é necessária para utilizarmos a biblioteca `tree`.

A biblioteca `tree` é uma das implementações do algoritmo CART disponíveis no `R`. A função `tree()` desta biblioteca utiliza uma sintaxe similar à da função `lm()`.

```
library(tree)

tree <- tree(Class ~ . , data = db)
```

Crescendo a árvore de classificação (2)

Insper

Usando a função `summary()` obtemos os detalhes da árvore construída.

```
summary(tree)
```

```
##
## Classification tree:
## tree(formula = Class ~ ., data = db)
## Variables actually used in tree construction:
##  [1] "ch_dollar" "remove"    "ch_bang"   "hp"        "CAPMAX"
##  [6] "our"       "CAPAVE"    "free"      "george"    "edu"
## Number of terminal nodes:  13
## Residual mean deviance:  0.4879 = 2238 / 4588
## Misclassification error rate: 0.08259 = 380 / 4601
```

Vemos quais preditoras foram utilizadas para definir as divisões, o número de folhas (13) e o erro de classificação de treinamento (8,3%). Neste sumário, a *residual mean deviance* é definida por

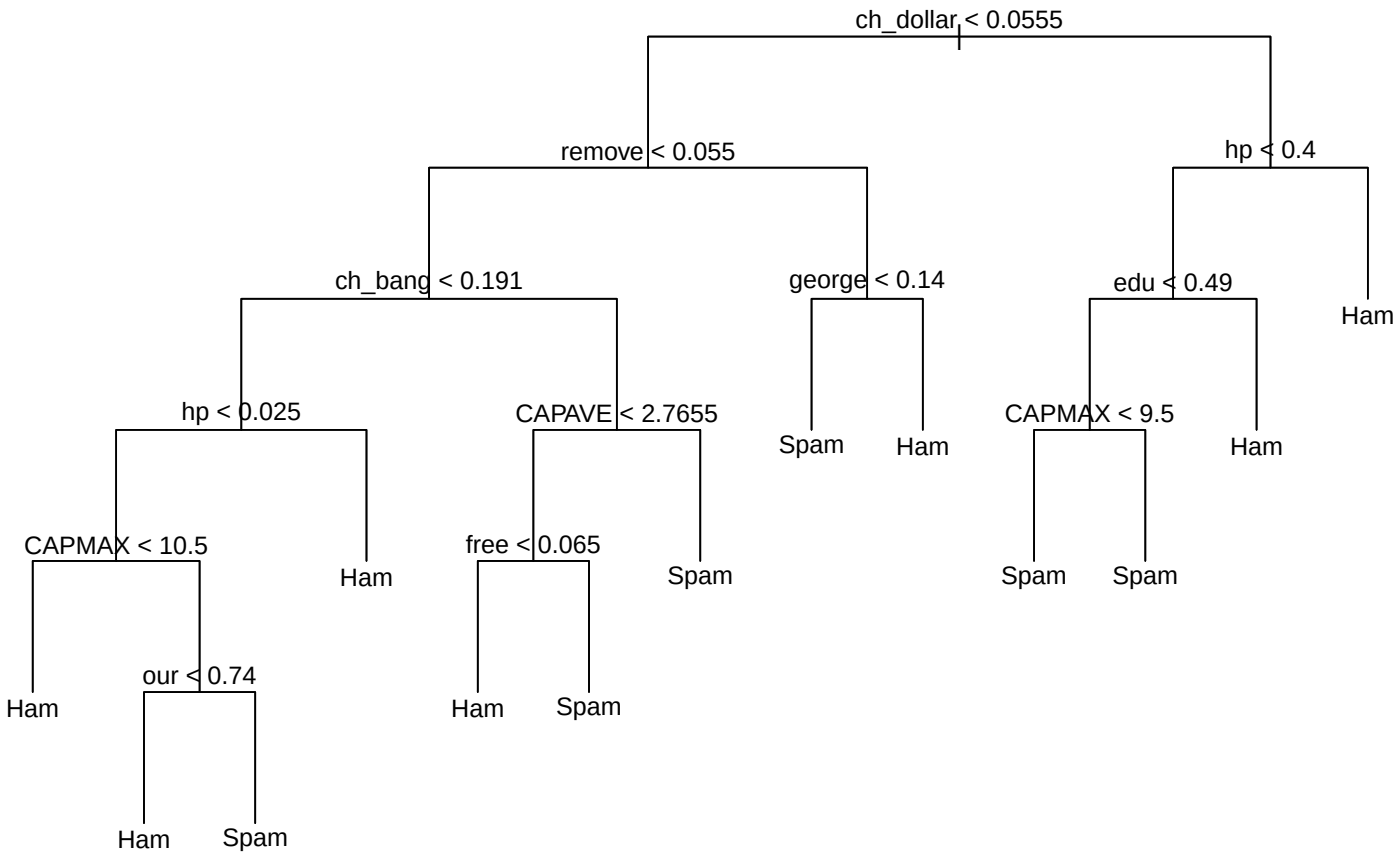
$$\frac{-2\sum_{m,k} n_{mk} \log \hat{p}_{mk}}{n - |T_0|}$$

em que n_{mk} é o número de observações na região correspondente à folha m que pertencem à class k. Um valor pequeno indica um bom ajuste dos dados de treinamento.

Visualizando a árvore de classificação obtida

Insper

```
plot(tree, type = "uniform"); text(tree, cex = 0.95)
```



Performance preditiva (1)

Insper

Para a árvore de classificação obtida podemos estimar o erro de classificação esperado.

Conforme discutido nas aulas anteriores, isso pode ser feito através de uma validação cruzada em k lotes.

Neste exemplo, iremos apenas dividir aleatoriamente a base de dados em um conjunto de treinamento, com 70% dos dados, e um conjunto de teste, com 30% dos dados (essencialmente um procedimento de validação cruzada com um único lote).

```
set.seed(1234)

tr_idx <- sample(1:nrow(db), size = round(0.7*nrow(db)), replace = FALSE)

training <- db[tr_idx,]
test <- db[-tr_idx,]
```

Performance preditiva (2)

Insper

Agora, crescemos uma árvore de classificação a partir dos dados de treinamento.

```
tr_tree <- tree(Class ~ . , data = training)
```

Para prever as respostas dos dados de teste com esta árvore, utilizamos a função `predict()`.

```
pred <- predict(tr_tree, test, type = "class")

conf <- table(pred, test$Class, dnn = c("Predicted", "Actual"))
print(conf)
```

```
##           Actual
## Predicted Ham Spam
##      Ham  764   71
##      Spam  77  468
```

Portanto, 89.3% dos e-mails de teste foram classificados corretamente.

Podando a árvore de classificação (1)

Insper

É interessante verificar se conseguimos melhorar a performance preditiva podando a árvore de classificação original.

A função `cv.tree()` escolhe por validação cruzada em 10 lotes o parâmetro α, definido no slide 12, que determina a complexidade ideal da árvore podada. Nesta função `cv.tree()`, `k` é o nosso α, o número de folhas da árvore é `size`, e `dev` é a estimativa de validação cruzada do número esperado de erros de classificação.

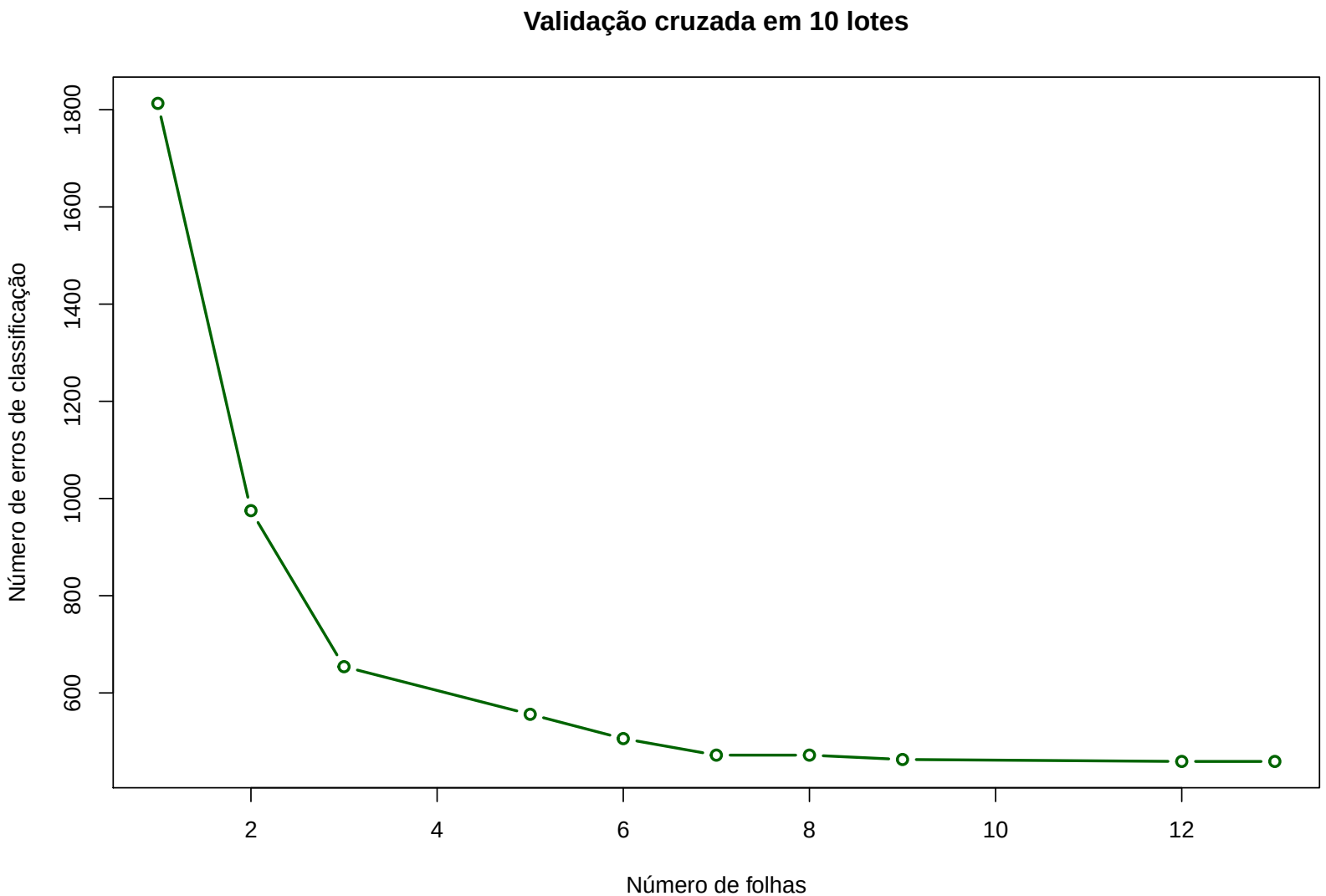
```
set.seed(1234)

cv <- cv.tree(tree, FUN = prune.misclass)
print(cv)
```

```
## $size
##  [1] 13 12  9  8  7  6  5  3  2  1
##
## $dev
##  [1]  459  459  463  472  472  506  556  654  975 1813
##
## $k
##  [1]      -Inf   0.00000  10.33333  13.00000  15.00000  32.00000  56.00000
##  [8]  76.00000 270.00000 864.00000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

Podando a árvore de classificação (2)

Insper

```
plot(cv$size, cv$dev, type = "b", lwd = 2, col = "dark green", xlab = "Número de folhas",
     ylab = "Número de erros de classificação", main = "Validação cruzada em 10 lotes")
```

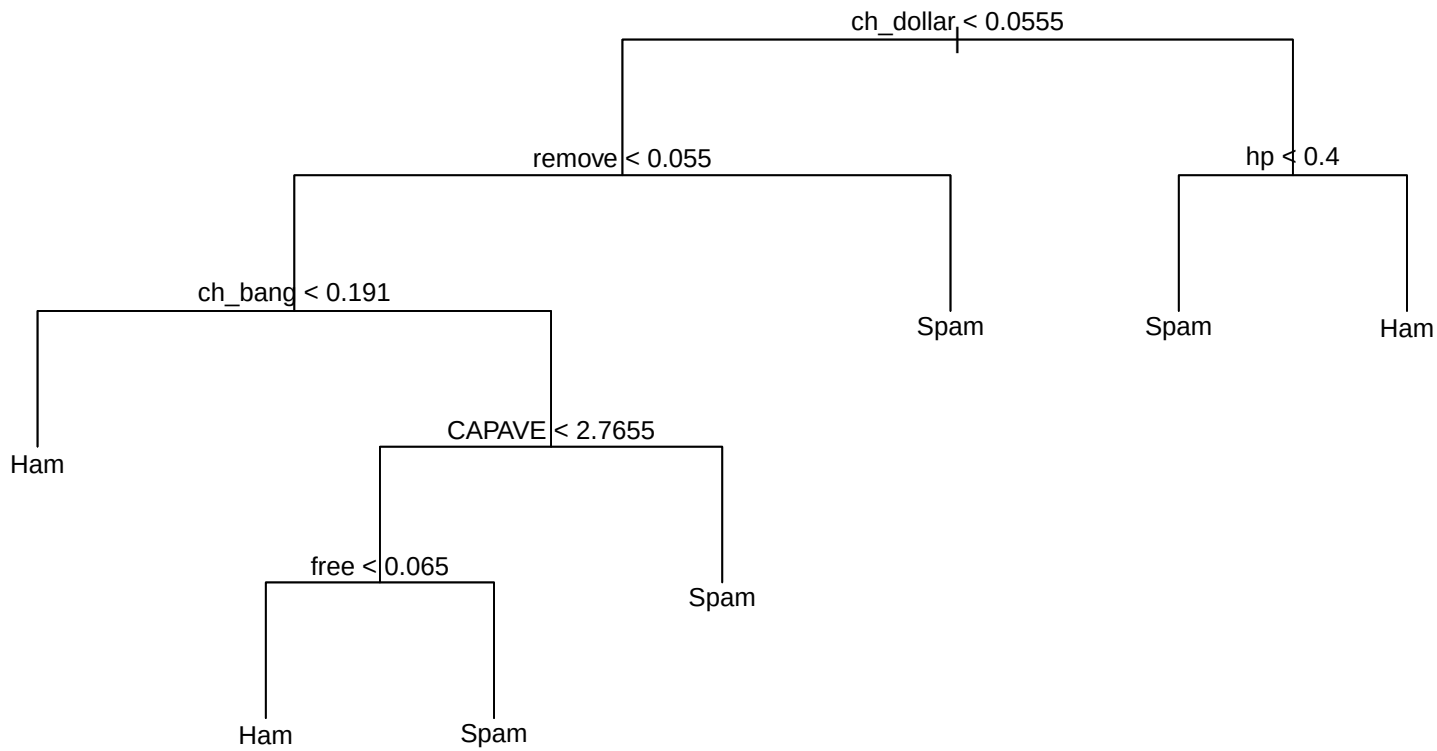


Árvore de classificação podada

Insper

Esta análise sugere que a árvore original seja podada para que tenha apenas 7 folhas. Fazemos isto com a função `prune.misclass()`.

```
pruned <- prune.misclass(tree, best = 7);
plot(pruned, type = "uniform"); text(pruned, cex = 0.95)
```



Árvores de regressão (1)

Insper

No problema de regressão, a variável resposta é quantitativa: $Y \in \mathbb{R}$.

O CART constrói a árvore de regressão de maneira análoga ao caso de classificação.

A principal diferença é que para definir as divisões utilizamos uma perda quadrática ao invés do erro de classificação:

$$(\hat{j}, \hat{t}) = \arg\min_{(j,t)} \left(\sum_{\{i: x_i \in R_1\}} (y_i - \hat{y}_{R_1})^2 + \sum_{\{i: x_i \in R_2\}} (y_i - \hat{y}_{R_2})^2 \right),$$

em que $\hat{y}_{R_1}$ e $\hat{y}_{R_2}$ são as médias das respostas dos dados de treinamento que pertencem às regiões R_1 e R_2, respectivamente.

Árvores de regressão (2)

Insper

Para cada região R_j correspondente a um nó terminal da árvore de regressão obtida, o CART associa uma constante c_j que é a média das respostas dos dados de treinamento que pertencem à região R_j.

$$c_j = \frac{1}{n_j} \sum_{\{i:x_i \in R_j\}}^{m} y_i.$$

A estimativa CART para a função de regressão é simplesmente

$$\hat{\psi}(x) = \sum_{j=1}^{m} c_j \, I_{R_j}(x).$$

Sugestões para o laboratório

Insper

1. Examinar a definição e os valores obtidos da *residual mean deviance* na biblioteca `tree`.
2. Examinar o impacto do uso de outras medidas para a definição das divisões da árvore de classificação no algoritmo CART, tais como o índice de Gini e a entropia cruzada.
3. Implementar exemplos de árvores de regressão.
4. Refazer as análises da aula utilizando a implementação do CART da biblioteca `rpart` do `R`.
5. Comparar o CART com seu algoritmo "concorrente" C5.0, implementado na biblioteca `C50` do `R`.

Créditos

A figura do slide 4 foi extraída do livro

The Elements of Statistical Learning,

dos autores T. Hastie, R. Tibshirani and J. Friedman.

http://web.stanford.edu/~hastie/ElemStatLearn/