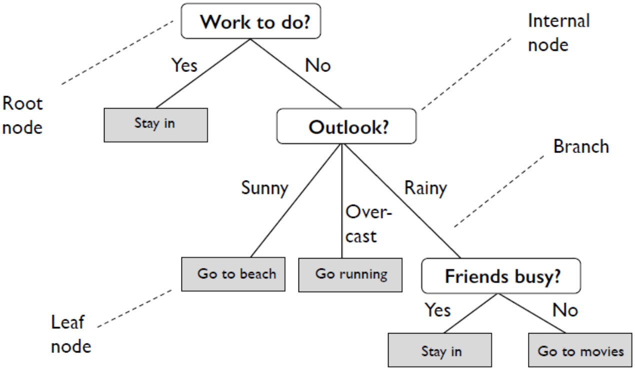
Aula 15 - Árvore de Decisão

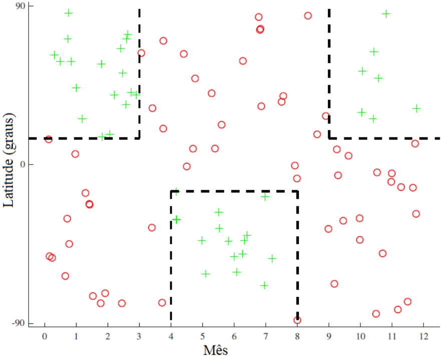
João Florindo

Instituto de Matemática, Estatística e Computação Científica
Universidade Estadual de Campinas - Brasil
florindo@unicamp.br

Outline


Work to do?
Yes
No
Stay in
Outlook?
Sunny
Over-
cast
Rainy
Go to beach
Go running
Friends busy?
Yes
No
Stay in
Go to movies
Root
node
Internal
node
Branch
Leaf
node

Viabilidade de esquiar (verde="SIM", vermelho = "NÃO"):



- Baseado em regras
- Naturalmente não linear
- Algoritmo fácil de explicar, entender e interpretar
- Algoritmo guloso: "dividir para conquistar"
- Relação com aprendizado baseado em regras

- Baseado em regras
- Naturalmente não linear
- Algoritmo fácil de explicar, entender e interpretar
- Algoritmo guloso: "dividir para conquistar"
- Relação com aprendizado baseado em regras

- Baseado em regras
- Naturalmente não linear
- Algoritmo fácil de explicar, entender e interpretar
- Algoritmo guloso: "dividir para conquistar"
- Relação com aprendizado baseado em regras

- Baseado em regras
- Naturalmente não linear
- Algoritmo fácil de explicar, entender e interpretar
- Algoritmo guloso: “dividir para conquistar”
- Relação com aprendizado baseado em regras

- Baseado em regras
- Naturalmente não linear
- Algoritmo fácil de explicar, entender e interpretar
- Algoritmo guloso: “dividir para conquistar”
- Relação com aprendizado baseado em regras

- Particionar o espaço de atributos $\mathcal{X}$ em n regiões R_i:

$$\begin{cases} \mathcal{X} = \cup_{i=0}^{n} R_i, & n \in \mathbb{Z}^+ \\ \text{s.t. } R_i \cup R_j = \emptyset \text{ para } i \neq j. \end{cases}$$

- Algoritmo geral:
  1. Divide $\mathcal{X}$ em duas regiões "filhas" por *threshold* de um único atributo
  2. Escolhe um novo atributo e divide novamente as regiões filhas em duas por *thresholding*.

- Formalmente, temos uma região "pai" R_p, um atributo indexado por j e um *threshold* $t \in \mathbb{R}$. As regiões filhas R_1 e R_2 são obtidas por:

$$R_1 = \{X | X_j < t, X \in R_p\}$$
$$R_2 = \{X | X_j \geq t, X \in R_p\}$$

- Particionar o espaço de atributos $\mathcal{X}$ em n regiões R_i:

$$\begin{cases} \mathcal{X} = \cup_{i=0}^{n} R_i, & n \in \mathbb{Z}^+ \\ \text{s.t. } R_i \cup R_j = \emptyset \text{ para } i \neq j. \end{cases}$$

- Algoritmo geral:
  1. Divide $\mathcal{X}$ em duas regiões "filhas" por *threshold* de um único atributo
  2. Escolhe um novo atributo e divide novamente as regiões filhas em duas por *thresholding*.

- Formalmente, temos uma região "pai" R_p, um atributo indexado por j e um *threshold* $t \in \mathbb{R}$. As regiões filhas R_1 e R_2 são obtidas por:

$$R_1 = \{X | X_j < t, X \in R_p\}$$
$$R_2 = \{X | X_j \geq t, X \in R_p\}$$

- Particionar o espaço de atributos $\mathcal{X}$ em n regiões R_i:

$$\begin{cases} \mathcal{X} = \cup_{i=0}^{n} R_i, & n \in \mathbb{Z}^+ \\ \text{s.t. } R_i \cup R_j = \emptyset \text{ para } i \neq j. \end{cases}$$

- Algoritmo geral:
  1. Divide $\mathcal{X}$ em duas regiões “filhas” por *threshold* de um único atributo
  2. Escolhe um novo atributo e divide novamente as regiões filhas em duas por *thresholding*.

- Formalmente, temos uma região “pai” R_p, um atributo indexado por j e um *threshold* $t \in \mathbb{R}$. As regiões filhas R_1 e R_2 são obtidas por:

$$R_1 = \{X | X_j < t, X \in R_p\}$$
$$R_2 = \{X | X_j \geq t, X \in R_p\}$$

- Particionar o espaço de atributos $\mathcal{X}$ em n regiões R_i:

$$\begin{cases} \mathcal{X} = \cup_{i=0}^{n} R_i, & n \in \mathbb{Z}^+ \\ \text{s.t. } R_i \cup R_j = \emptyset \text{ para } i \neq j. \end{cases}$$

- Algoritmo geral:
  1. Divide $\mathcal{X}$ em duas regiões "filhas" por *threshold* de um único atributo
  2. Escolhe um novo atributo e divide novamente as regiões filhas em duas por *thresholding*.
- Formalmente, temos uma região "pai" R_p, um atributo indexado por j e um *threshold* $t \in \mathbb{R}$. As regiões filhas R_1 e R_2 são obtidas por:

$$R_1 = \{X | X_j < t, X \in R_p\}$$
$$R_2 = \{X | X_j \geq t, X \in R_p\}$$

- Particionar o espaço de atributos $\mathcal{X}$ em n regiões R_i:

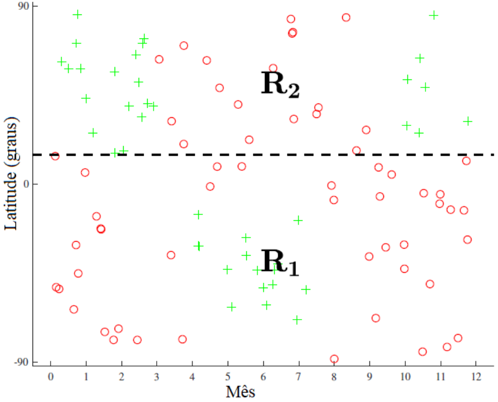
$$\begin{cases} \mathcal{X} = \cup_{i=0}^{n} R_i, & n \in \mathbb{Z}^+ \\ \text{s.t. } R_i \cup R_j = \emptyset \text{ para } i \neq j. \end{cases}$$

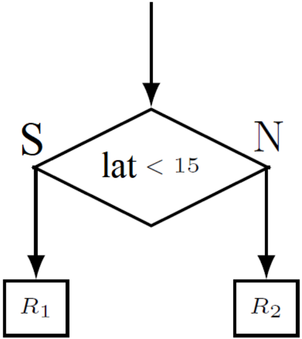
- Algoritmo geral:
  1. Divide $\mathcal{X}$ em duas regiões "filhas" por *threshold* de um único atributo
  2. Escolhe um novo atributo e divide novamente as regiões filhas em duas por *thresholding*.
- Formalmente, temos uma região "pai" R_p, um atributo indexado por j e um *threshold* $t \in \mathbb{R}$. As regiões filhas R_1 e R_2 são obtidas por:

$$R_1 = \{X | X_j < t, X \in R_p\}$$
$$R_2 = \{X | X_j \geq t, X \in R_p\}$$

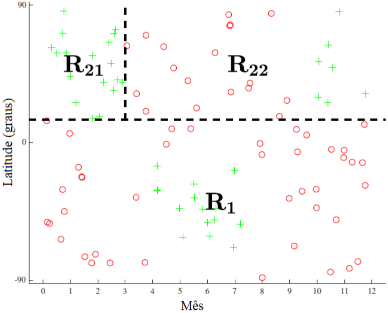
Outline

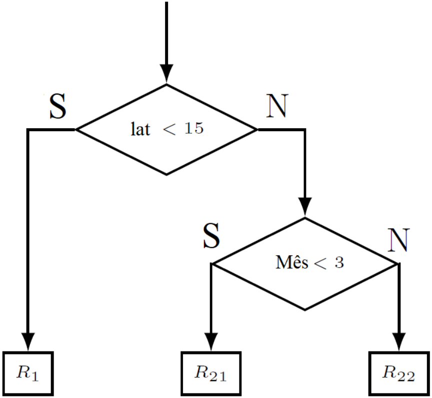
- PASSO 1: Dividimos $\mathcal{X}$ pelo atributo “latitude” com *threshold* 15.



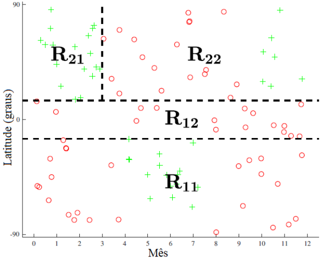


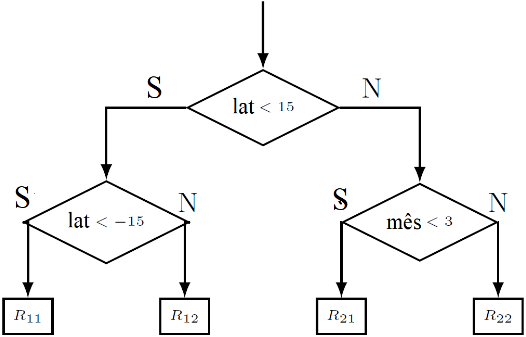
- PASSO 2: Escolhemos uma das regiões (no caso R_2), o atributo "Mês" e *threshold* 3.





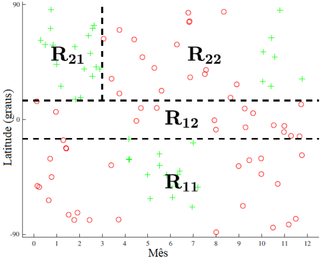
- PASSO 3: Escolhemos um dos nós folha (R_1, R_{21}, R_{22}). No caso, optamos por R_1, com atributo "longitude" e *threshold* -15.

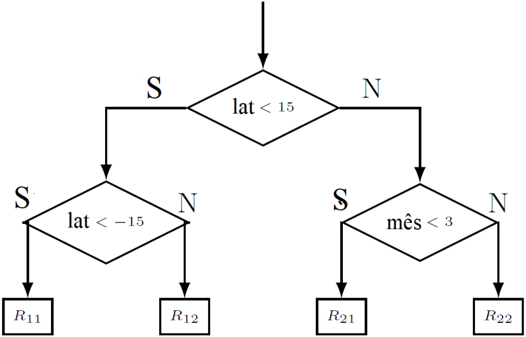




- Continuamos até um critério de parada e prevemos a classe majoritária em cada nó folha.

- PASSO 3: Escolhemos um dos nós folha (R_1, R_{21}, R_{22}). No caso, optamos por R_1, com atributo "longitude" e *threshold* -15.





- Continuamos até um critério de parada e prevemos a classe majoritária em cada nó folha.

Outline

- Como escolher os pontos de divisão?
- Perda para cada região pai R_p que se divide em duas filhas R_1 e R_2:

$$\frac{|R_1|L(R_1) + |R_2|L(R_2)}{|R_1| + |R_2|}.$$

- Objetivo é maximizar a diminuição da perda em cada divisão:

$$L(R_p) - \frac{|R_1|L(R_1) + |R_2|L(R_2)}{|R_1| + |R_2|}.$$

- Na classificação:

$$L_{misclass}(R) = 1 - \underset{c}{\operatorname{argmax}}(\hat{p}_c),$$

em que $\hat{p}_c$ é a proporção de exemplos em R que são da classe c.

- Como escolher os pontos de divisão?
- Perda para cada região pai R_p que se divide em duas filhas R_1 e R_2:

$$\frac{|R_1|L(R_1) + |R_2|L(R_2)}{|R_1| + |R_2|}.$$

- Objetivo é maximizar a diminuição da perda em cada divisão:

$$L(R_p) - \frac{|R_1|L(R_1) + |R_2|L(R_2)}{|R_1| + |R_2|}.$$

- Na classificação:

$$L_{misclass}(R) = 1 - \underset{c}{\text{argmax}}(\hat{p}_c),$$

em que $\hat{p}_c$ é a proporção de exemplos em R que são da classe c.

- Como escolher os pontos de divisão?
- Perda para cada região pai R_p que se divide em duas filhas R_1 e R_2:

$$\frac{|R_1|L(R_1) + |R_2|L(R_2)}{|R_1| + |R_2|}.$$

- Objetivo é maximizar a diminuição da perda em cada divisão:

$$L(R_p) - \frac{|R_1|L(R_1) + |R_2|L(R_2)}{|R_1| + |R_2|}.$$

- Na classificação:

$$L_{misclass}(R) = 1 - \underset{c}{\operatorname{argmax}}(\hat{p}_c),$$

em que $\hat{p}_c$ é a proporção de exemplos em R que são da classe c.

- Como escolher os pontos de divisão?
- Perda para cada região pai R_p que se divide em duas filhas R_1 e R_2:

$$\frac{|R_1|L(R_1) + |R_2|L(R_2)}{|R_1| + |R_2|}.$$

- Objetivo é maximizar a diminuição da perda em cada divisão:

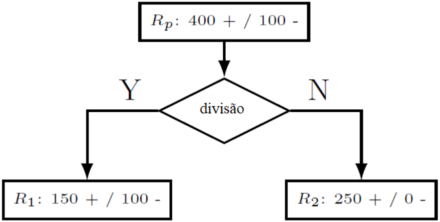
$$L(R_p) - \frac{|R_1|L(R_1) + |R_2|L(R_2)}{|R_1| + |R_2|}.$$

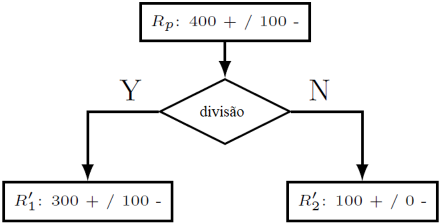
- Na classificação:

$$L_{misclass}(R) = 1 - \underset{c}{\operatorname{argmax}}(\hat{p}_c),$$

em que $\hat{p}_c$ é a proporção de exemplos em R que são da classe c.

Mas temos um problema!





$$L(R_p) - \frac{|R_1|L(R_1) + |R_2|L(R_2)}{|R_1| + |R_2|} = \frac{100}{500} - \left(\frac{250 \cdot \frac{100}{250} + 250 \cdot \frac{0}{250}}{250 + 250} \right) = 0.$$

$$L(R_p) - \frac{|R_1'|L(R_1') + |R_2'|L(R_2')}{|R_1'| + |R_2'|} = \frac{100}{500} - \left(\frac{400 \cdot \frac{100}{400} + 100 \cdot \frac{0}{100}}{400 + 100} \right) = 0.$$

Mas temos um problema!

PROBLEMAS:

1. Ambas as divisões têm a mesma perda.
2. A perda do nó pai $L(R_p)$ não diminuiu.

Melhor opção é a **cross-entropy**:

$$L_{cross} = -\sum_{c} \hat{p}_c \log_2 \hat{p}_c,$$

em que $\hat{p} \log_2 \hat{p} \equiv 0$ se $\hat{p} = 0$.

Mas temos um problema!

PROBLEMAS:

1. Ambas as divisões têm a mesma perda.
2. A perda do nó pai $L(R_p)$ não diminuiu.

Melhor opção é a **cross-entropy**:

$$L_{cross} = -\sum_{c} \hat{p}_c \log_2 \hat{p}_c,$$

em que $\hat{p} \log_2 \hat{p} \equiv 0$ se $\hat{p} = 0$.

Mas temos um problema!

PROBLEMAS:

1. Ambas as divisões têm a mesma perda.
2. A perda do nó pai $L(R_p)$ não diminuiu.

Melhor opção é a **cross-entropy**:

$$L_{cross} = -\sum_{c} \hat{p}_c \log_2 \hat{p}_c,$$

em que $\hat{p} \log_2 \hat{p} \equiv 0$ se $\hat{p} = 0$.

Mas temos um problema!

Neste caso:

$$L(R_p) = -\left(\frac{400}{500}\log_2\left(\frac{400}{500}\right) + \frac{100}{500}\log_2\left(\frac{100}{500}\right)\right) = 0.722$$

$$L(R_1) = -\left(\frac{150}{250}\log_2\left(\frac{150}{250}\right) + \frac{100}{250}\log_2\left(\frac{100}{250}\right)\right) = 0.971$$

$$L(R_2) = -\left(\frac{250}{250}\log_2\left(\frac{250}{250}\right) + 0\right) = 0$$

$$L(R_p) - \frac{|R_1|L(R_1) + |R_2|L(R_2)}{|R_1| + |R_2|} = 0.722 - \left(\frac{250 \cdot 0.971 + 250 \cdot 0}{250 + 250}\right) = 0.236.$$

Do mesmo modo (cheque você mesmo!):

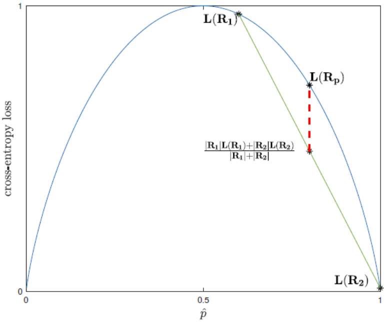
$$L(R_p) - \frac{|R_1'|L(R_1') + |R_2'|L(R_2')}{|R_1'| + |R_2'|} = 0.722 - \left(\frac{400 \cdot 0.811 + 100 \cdot 0}{400 + 100}\right) = 0.07.$$

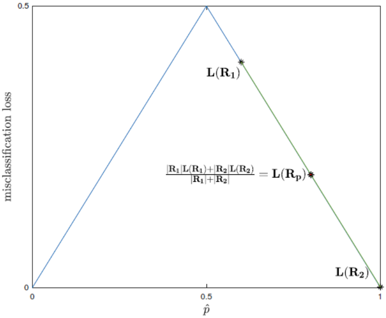
Mas temos um problema!

Para 2 classes, sendo $\hat{p}_i$ a proporção de exemplos positivos em R_i:

$$L_{misclass}(R) = L_{misclass}(\hat{p}) = 1 - \max(\hat{p}, 1 - \hat{p})$$

$$L_{cross}(R) = L_{cross}(\hat{p}) = -\hat{p}\log\hat{p} - (1 - \hat{p})\log(1 - \hat{p}).$$





Outline

- Parecida com a classificação, exceto que a predição final é dada pela média:

$$\hat{y} = \frac{\sum_{i \in R} y_i}{|R|}.$$

- A função de perda neste caso é a **perda quadrática**:

$$L_{squared}(R) = \frac{\sum_{i \in R}(y_i - \hat{y})^2}{|R|}.$$

- Parecida com a classificação, exceto que a predição final é dada pela média:
$$\hat{y} = \frac{\sum_{i \in R} y_i}{|R|}.$$

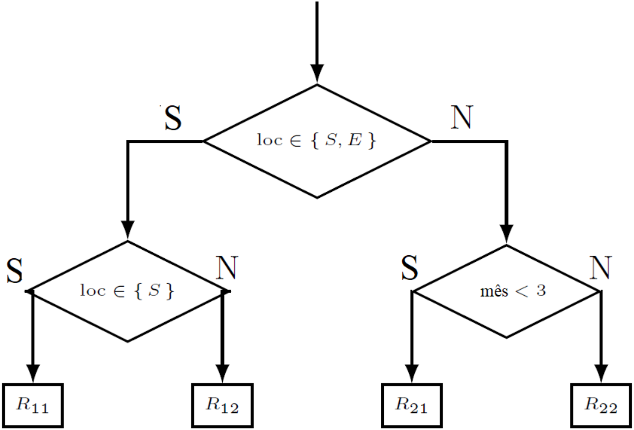
- A função de perda neste caso é a **perda quadrática**:

$$L_{squared}(R) = \frac{\sum_{i \in R}(y_i - \hat{y})^2}{|R|}.$$

Outline

Atributos Categóricos

- EXEMPLO: localização no exemplo do esqui poderia ser hemisfério norte, sul ou equador (loc$\in \{N, S, E\}$).



Atributos Categóricos

ATENÇÃO

Se o atributo assume um número muito grande de categorias, o custo computacional pode ser muito alto e ainda ser propenso a *overfitting*.

Melhor converter para quantitativo.

Regularização

- Tamanho mínimo de folha
- Profundidade máxima
- Número máximo de nós
- Poda

ATENÇÃO

Não se recomenda parar apenas porque a redução na perda após uma divisão é pequena. A árvore pode perder interações de alto nível entre os atributos e assim terminar prematuramente.

Regularização

- Tamanho mínimo de folha
- Profundidade máxima
- Número máximo de nós
- Poda

ATENÇÃO

Não se recomenda parar apenas porque a redução na perda após uma divisão é pequena. A árvore pode perder interações de alto nível entre os atributos e assim terminar prematuramente.

Regularização

- Tamanho mínimo de folha
- Profundidade máxima
- Número máximo de nós
- Poda

ATENÇÃO

Não se recomenda parar apenas porque a redução na perda após uma divisão é pequena. A árvore pode perder interações de alto nível entre os atributos e assim terminar prematuramente.

Regularização

- Tamanho mínimo de folha
- Profundidade máxima
- Número máximo de nós
- Poda

ATENÇÃO

Não se recomenda parar apenas porque a redução na perda após uma divisão é pequena. A árvore pode perder interações de alto nível entre os atributos e assim terminar prematuramente.

Regularização

- Tamanho mínimo de folha
- Profundidade máxima
- Número máximo de nós
- Poda

ATENÇÃO

Não se recomenda parar apenas porque a redução na perda após uma divisão é pequena. A árvore pode perder interações de alto nível entre os atributos e assim terminar prematuramente.

Tempo de Execução

- Seja a classificação binária com n exemplos, f atributos e árvore com profundidade d.
- No teste, o tempo é $\mathcal{O}(d)$. Se a árvore estiver balanceada, $d = \mathcal{O}(\log n)$.
- No treino, o tempo é $\mathcal{O}(nfd)$.

Tempo de Execução

- Seja a classificação binária com n exemplos, f atributos e árvore com profundidade d.
- No teste, o tempo é $\mathcal{O}(d)$. Se a árvore estiver balanceada, $d = \mathcal{O}(\log n)$.
- No treino, o tempo é $\mathcal{O}(nfd)$.

Tempo de Execução

- Seja a classificação binária com n exemplos, f atributos e árvore com profundidade d.
- No teste, o tempo é $\mathcal{O}(d)$. Se a árvore estiver balanceada, $d = \mathcal{O}(\log n)$.
- No treino, o tempo é $\mathcal{O}(nfd)$.

Perda da Estrutura Aditiva

Uma árvore de decisão não captura facilmente uma estrutura aditiva.
Considere uma simples fronteira de decisão $x_1 + x_2$