Reconhecimento de Padrões em Imagens
Árvores de Decisão

Dainf - UTFPR

Leyza Baldo Dorini - Rodrigo Minetto

Motivação

Vamos fazer uma simulação: quais seriam as 20 perguntas (com respostas sim/não) que você faria para tentar adivinhar em que pessoa estou pensando?

- Que características/medidas você considerou?
- A ordem das perguntas é relevante?

Além de natural e intuitivo, este processo é particularmente útil para dados nominais, pois não requer métricas. Exemplo: *20-question game* (`http://www.20q.net/`). Tais questões podem ser representadas por uma árvore de decisão.

Árvores de decisão

Árvores de Decisão

Abordagem para classificação de dados nominais (não-métricos), ou seja, de características discretas que não necessariamente tem escala de ordem ou noção de similaridade.

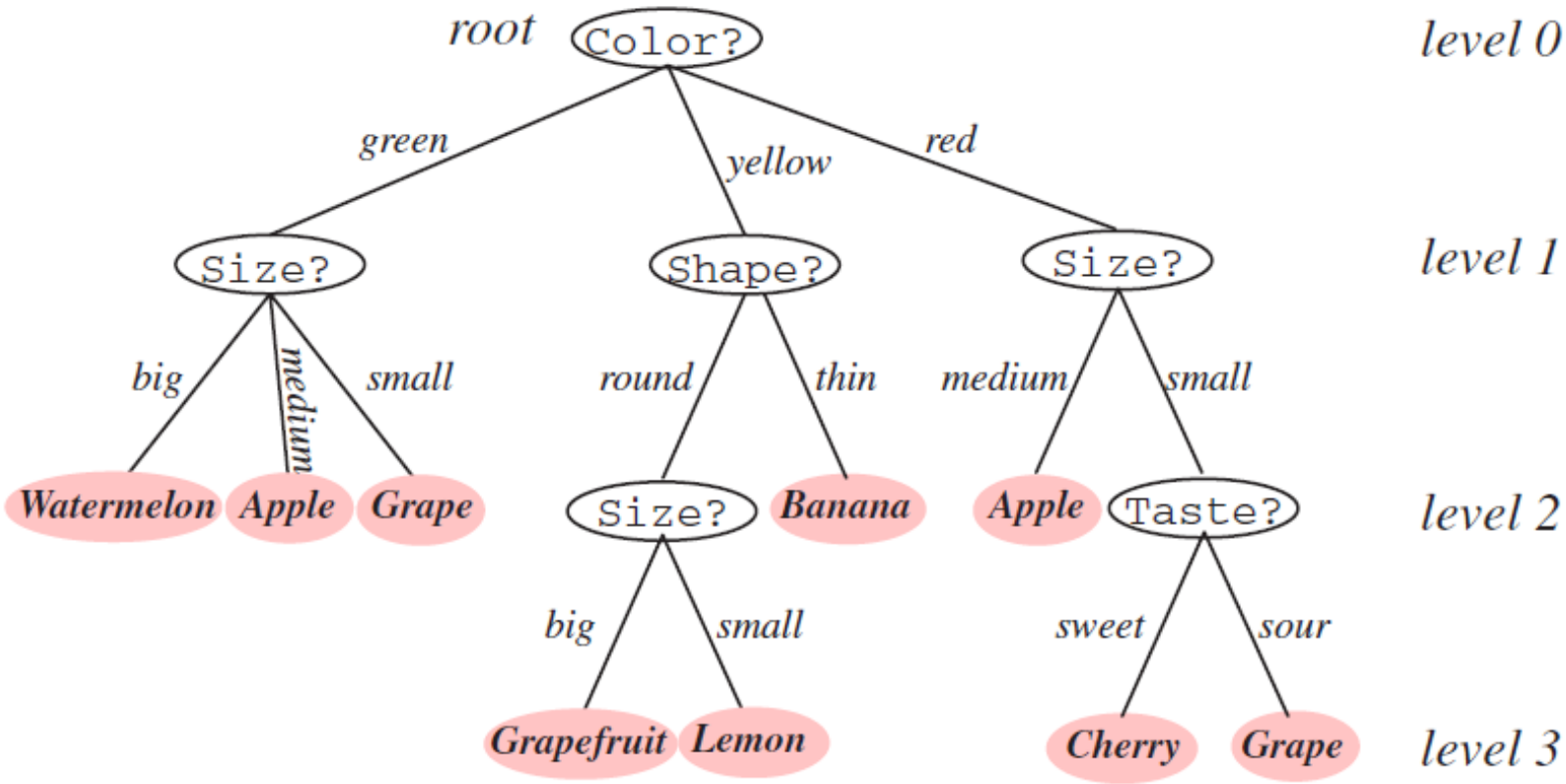
É adequada para problemas com as seguintes características:

- Instâncias são total ou parcialmente representadas por pares `atributo-valor`, com um conjunto fixo de atributos.
- A função alvo tem valores de saída discretos.
- Existência de descrições disjuntivas.
- Quando os dados de treinamento podem conter erros ou valores de atributos faltantes.

Exemplos de aplicações: diagnóstico médico, defeitos de equipamentos, análise de risco de crédito e modelagem de preferências para agendamento de horário.

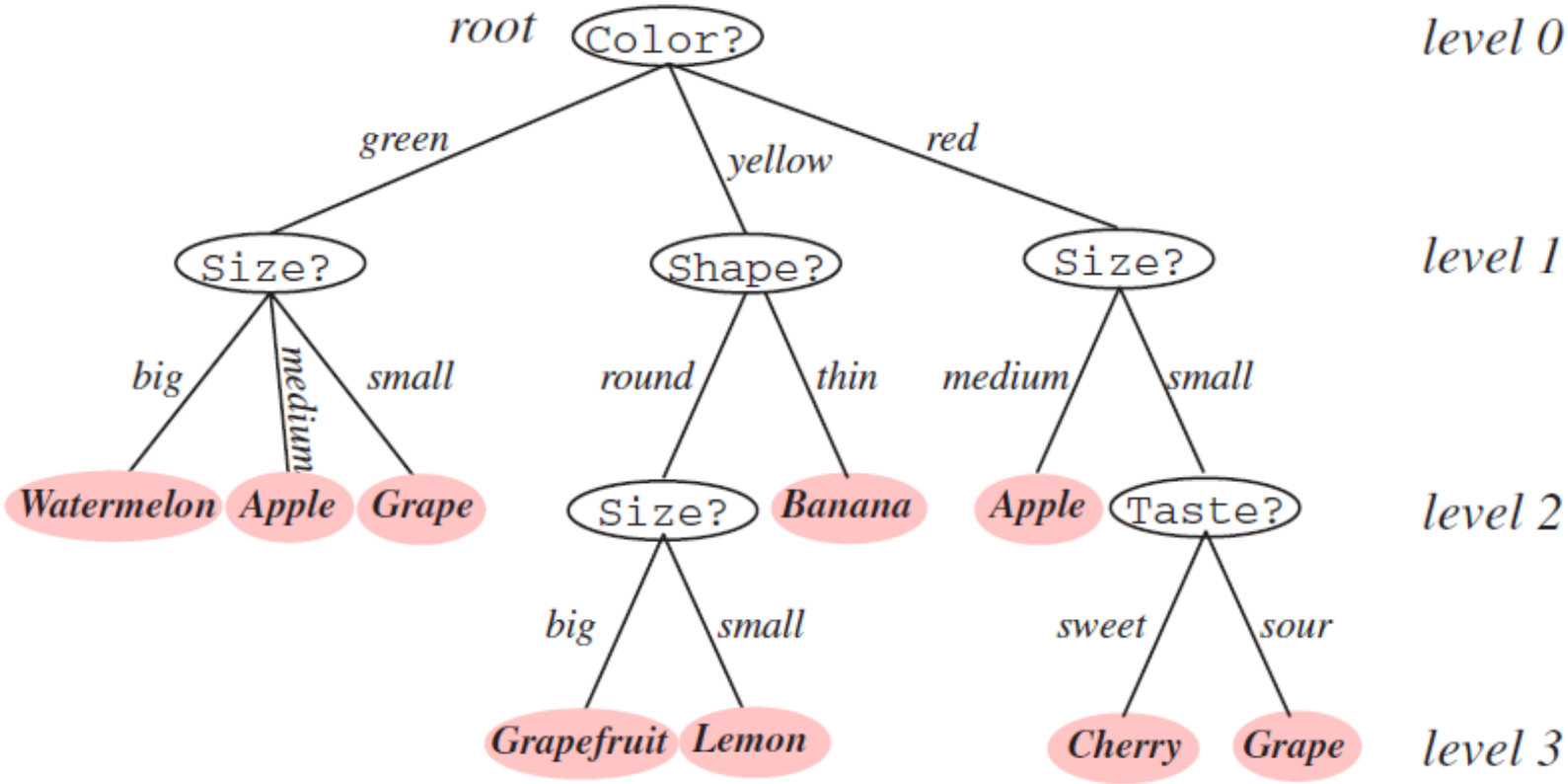
Árvores de decisão

Em uma árvore de decisão (*decision tree, hierarchical classifier, tree classifier*), os nós internos representam o status do problema após uma decisão (baseada nos valores de atributos)



Árvores de decisão

Em uma árvore de decisão (*decision tree, hierarchical classifier, tree classifier*), os nós internos representam o status do problema após uma decisão (baseada nos valores de atributos) e cada nó folha corresponde ao rótulo de uma classe (resultante da regra de classificação associada ao caminho desde o nó raiz).



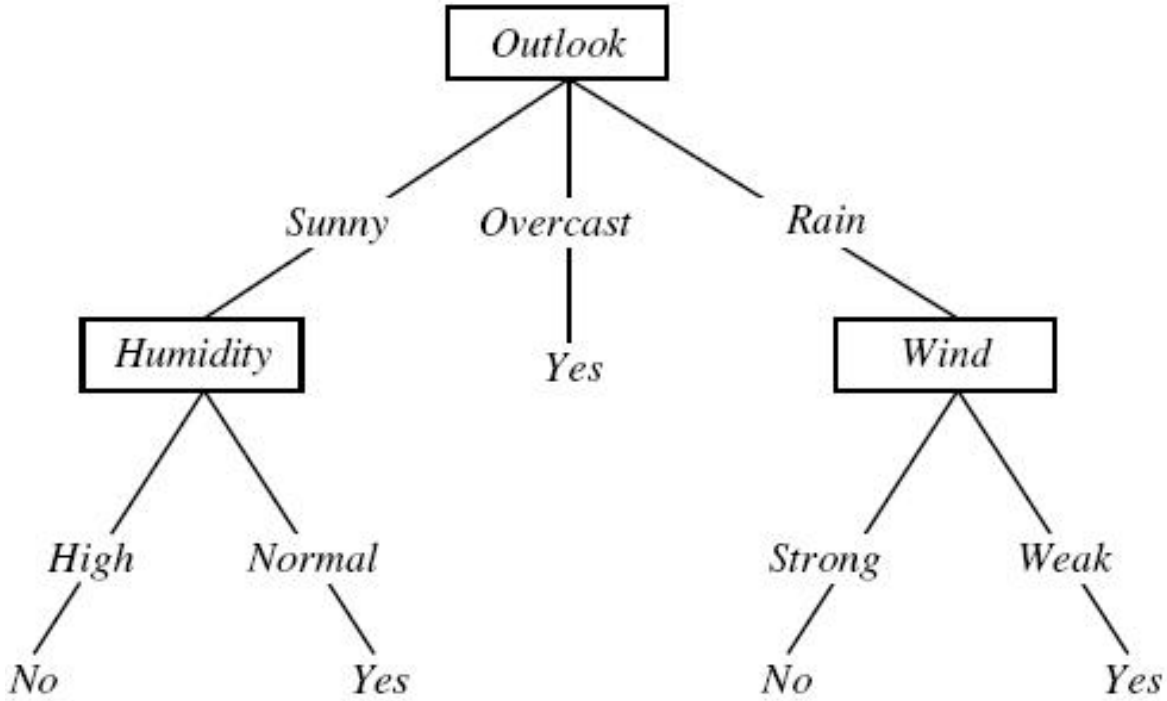
Interpretação da árvore de decisão

As seguintes observações podem ser feitas

- Os rótulos de cada classe são associados aos nós folha.
- O caminho entre a raiz (onde o processo de classificação é iniciado) e a folha define uma **regra de classificação**.
- O processo de classificação requer uma decisão a cada nó interno.
- As características podem ser nominais ou numéricas.
- A árvore pode ou não ser binária.
- Um conjunto de padrões está associado a cada nó (quanto mais próximo da raiz, maior o conjunto).

Interpretação da árvore de decisão

Cada caminho entre a raiz da árvore e uma folha corresponde a uma conjunção de testes de atributos. A própria árvore corresponde a uma disjunção destas conjunções (interpretabilidade).




```
(Outlook = Sunny E Humidity = Normal)
OU (Outlook = Overcast)
OU (Outlook = Rain E Wind = Weak)
```

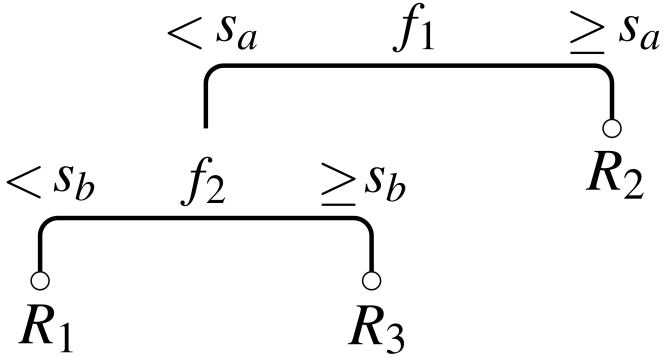
Testes: classificação

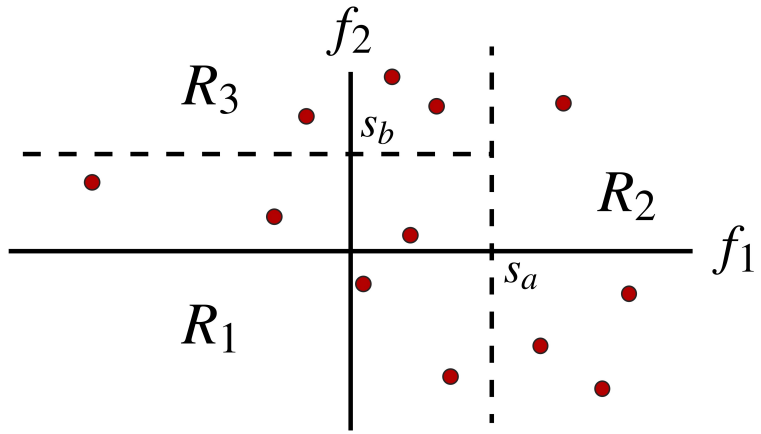
Cada nó interno da árvore de decisão está associado a um teste baseado nos valores de uma ou mais características. Tais testes podem ser classificados em:

1. Teste *axis-parallel*.
2. Teste baseado na combinação linear de características.
3. Teste baseado na combinação não-linear de características.

Testes: classificação

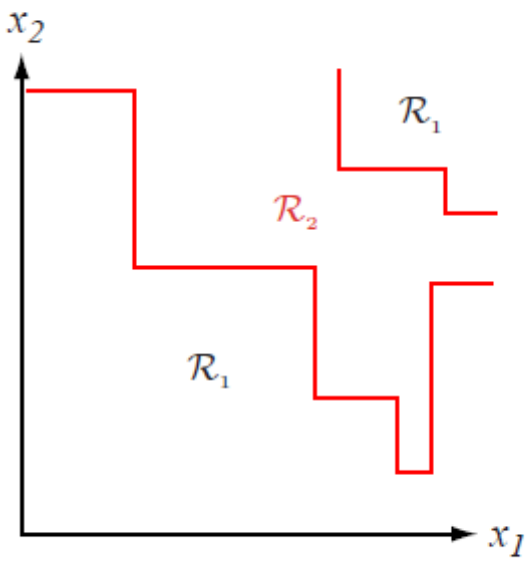
1 Teste *axis-parallel*: da forma $x > a_0$, que que x é a característica de a_0 é um limiar, e envolve essencialmente uma característica. Exemplo: *largura* > 150.

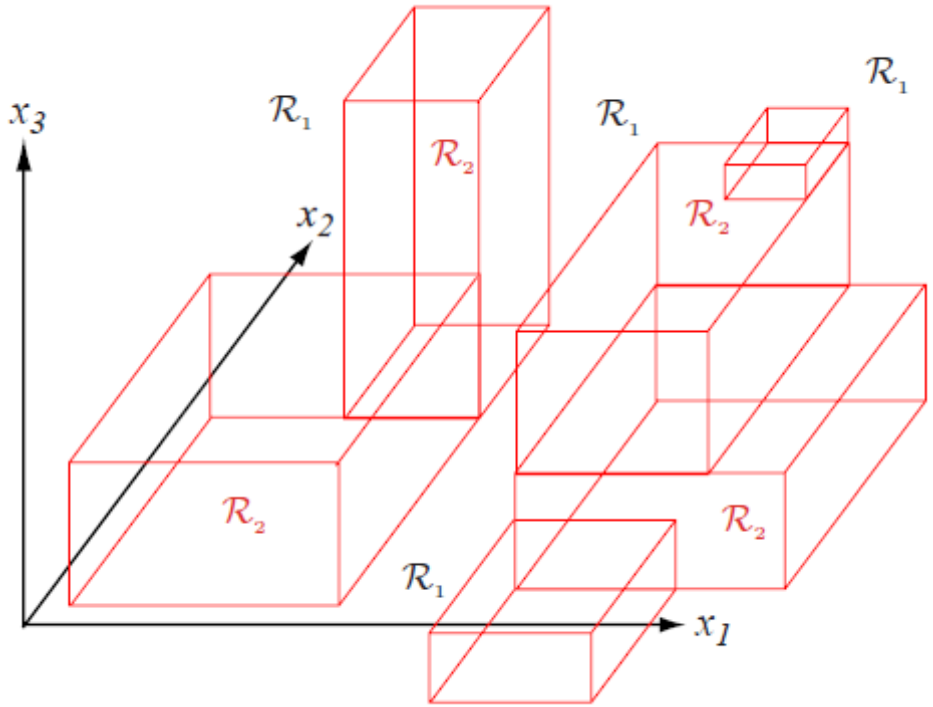




Testes: classificação

Exemplo: regiões de decisão com 2 e 3 categorias.



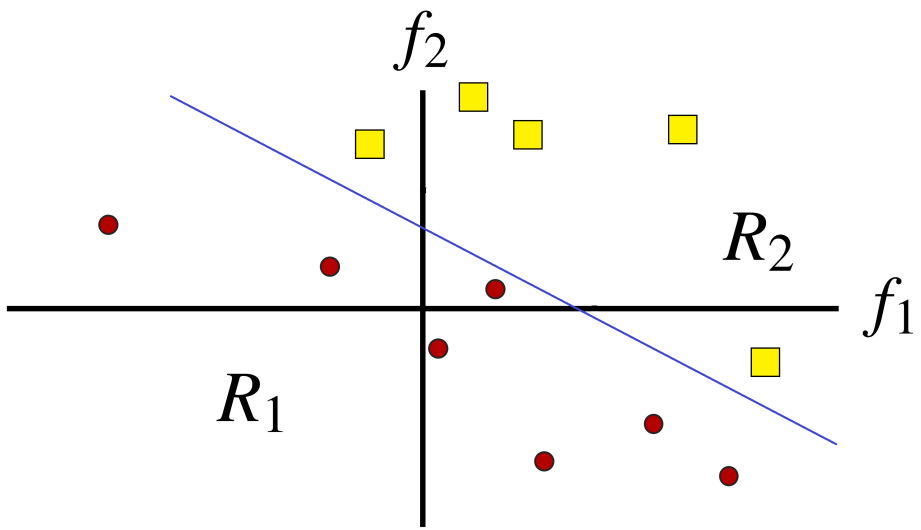


Testes: classificação

2 Teste baseado na combinação linear de características:

$$\sum_{i=1}^{d} a_i x_i > a_0$$

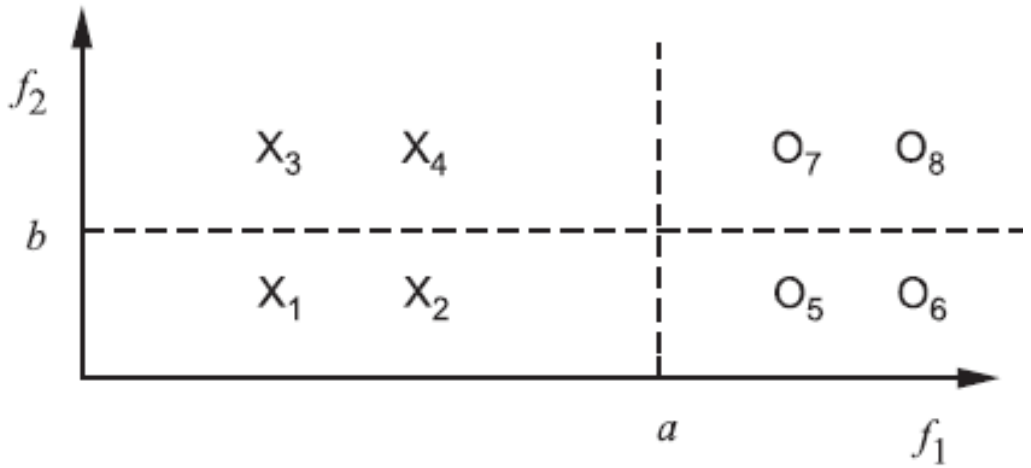
em que x_i é a i-ésima característica e a_i o peso associado (*oblique split*). Exemplo: $0.5 * \mathit{altura} + 0.3 * \mathit{largura} > 50$.



3 Teste baseado na combinação não-linear de características.

Construção de árvores de decisão

- Uma árvore de decisão é induzida a partir de exemplos, ou seja, de um conjunto de padrões rotulados que formam o conjunto de treinamento.
- A cada nó, é preciso escolher um ou mais atributos, a partir dos quais uma decisão é tomada. Quanto mais significativo (discriminativo) o atributo, mais alto deve ser o seu nível na árvore.



- Enquanto uma classificação final não for obtida, o resultado de uma decisão é outra árvore de decisão.

Algoritmos para a construção de árvores de decisão

Os principais algoritmos são:

1. ID3 (*Iterative Dichotomiser 3*): Proposto por Quinlan em 1986. Para cada nó, busca a característica que resulta no maior ganho de informação (entropia de Shannon). Após a construção da árvore até o seu tamanho máximo, uma etapa de poda é tipicamente aplicada, visando melhorar a sua capacidade de generalização na presença de novos dados.
2. C4.5 (1993): é o sucessor do ID3 e foi proposto pelo mesmo autor em 1993. A restrição de que todos as características deveriam ser categóricas foi removida. As árvores treinadas (ou seja, a saída do ID3) são convertidas em conjuntos de regras `if-then`, cujas acurácias são avaliadas para determinar a ordem em que são aplicadas.

Algoritmos para a construção de árvores de decisão

3. CART (*Classification and Regression Trees*): similar ao C4.5, mas suporta variáveis alvo numéricas e não calcula os conjuntos de regras. As árvores binárias são construídas usando a característica e o limiar que resulta no maior ganho de informação em cada nó. Na sequência, iremos discutir esta abordagem, cuja versão otimizada é utilizada pelo `scikit-learn`.

As implementações diferem basicamente nos seguintes fatores: critério para divisão, utilização de modelos de regressão/classificação, técnica pra eliminar/reduzir *overfitting* e capacidade de lidar com dados incompletos.

Construção de árvores - CART

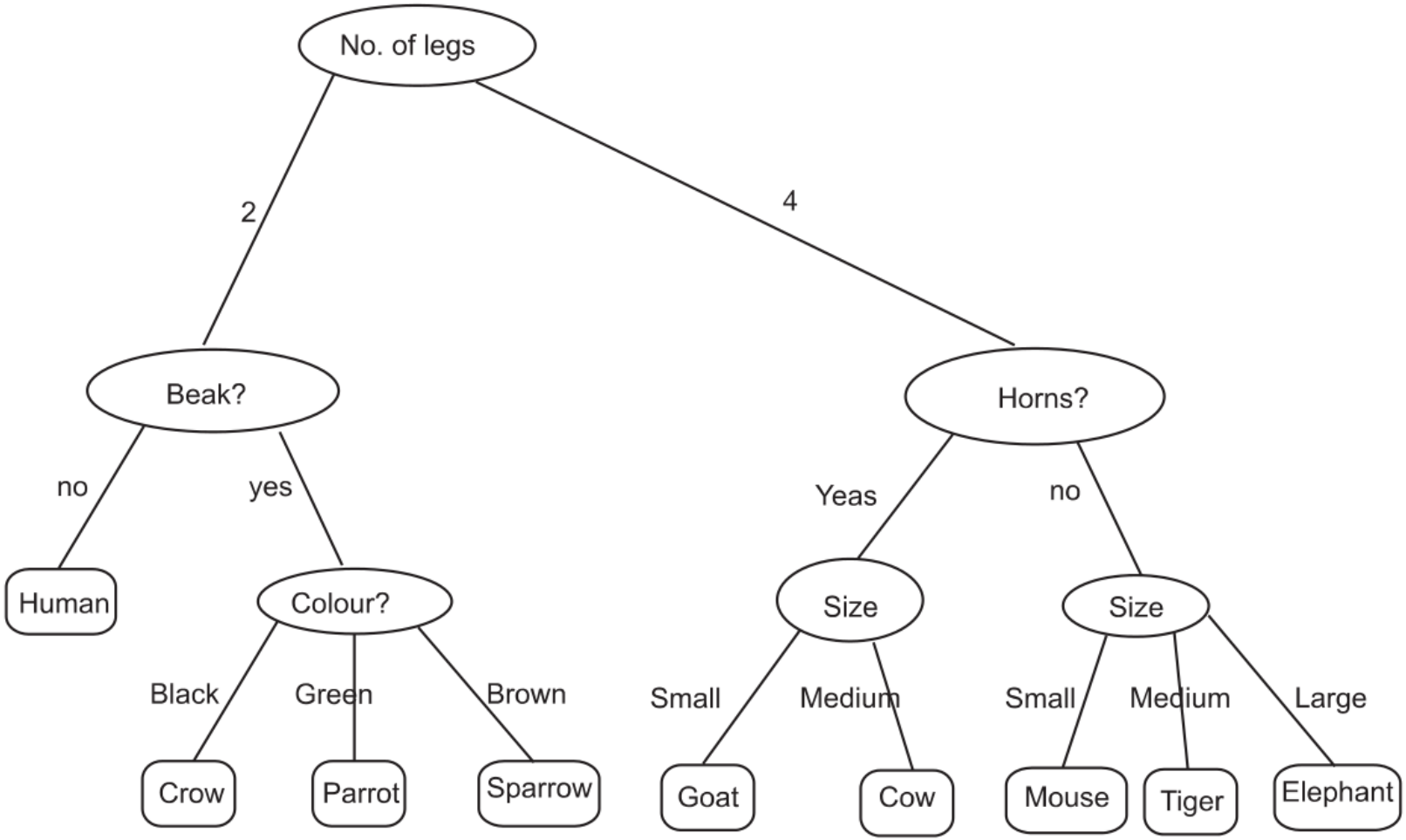
- Considere um conjunto de dados de treinamento já rotulados.
- Dado um conjunto de propriedades que podem ser utilizadas para discriminar os padrões de tal conjunto, o objetivo é organizá-las em uma árvore de decisão que maximize o desempenho da classificação de novos dados.

Por exemplo, considere o conjunto de animais descrito na tabela abaixo.

	Legs	Horns	Size	Colour	Beak	Sound
Cow	4	yes	medium	white	no	moo
Crow	2	no	small	black	yes	caw
Elephant	4	no	large	black	no	trumpet
Goat	4	yes	small	brown	no	bleat
Mouse	4	no	small	black	no	squeak
Parrot	2	no	small	green	yes	squawk
Sparrow	2	no	small	brown	yes	chirp
Tiger	4	no	medium	orange	no	roar

Exemplo

Uma possível árvore de decisão associada é dada por:



Construção de árvores - CART

- Em uma árvore de decisão, o conjunto de dados é progressivamente divido em subconjuntos (estratégia recursiva). Caso um deles tenha apenas elementos de uma mesma categoria, o nó associado é denominado **puro** e não precisa mais ser dividido.
- Contudo, como nós puros raramente ocorrem, é preciso decidir entre:
  - Parar o processo, o que implica em aceitar a taxa de erros associada.
  - Selecionar uma nova propriedade e fazer mais uma subdivisão.

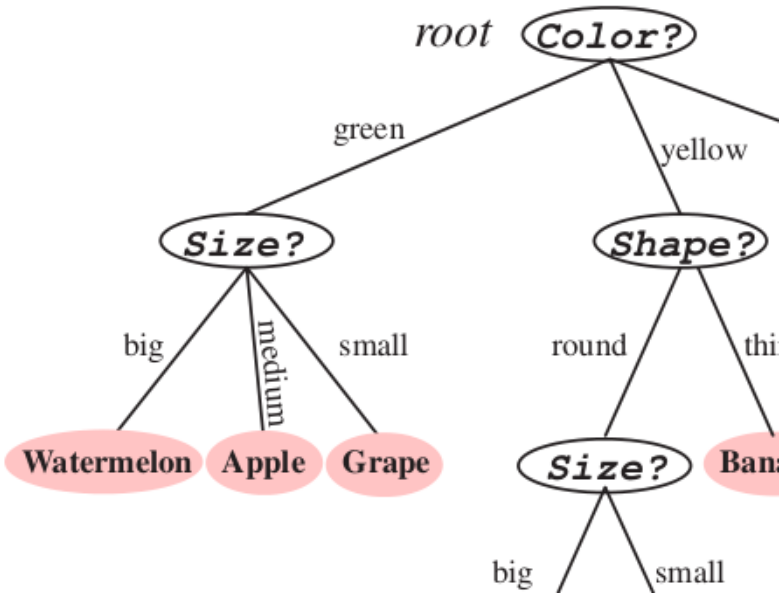
Construção de árvores - CART

CART é um framework geral para gerar árvores de decisão, sendo seis as principais questões que definem a instância gerada:

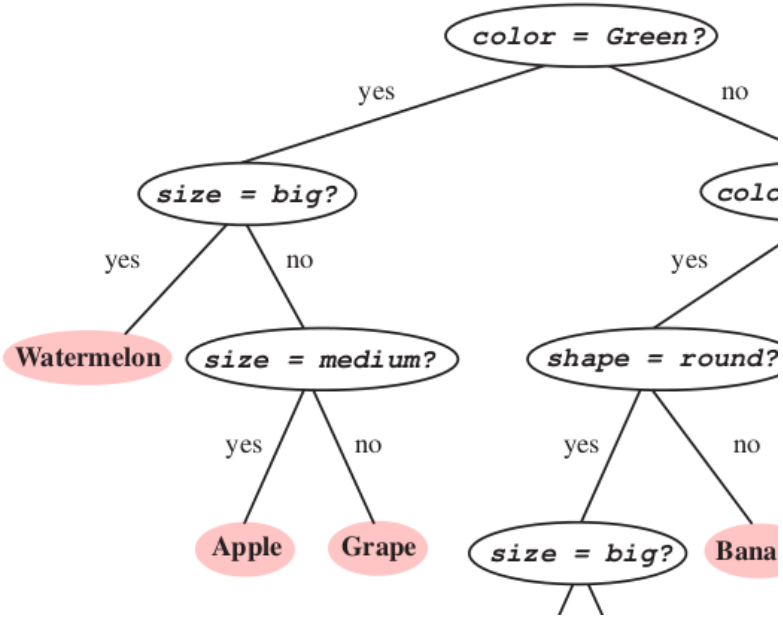
1. As propriedades são binárias ou multivaloradas, ou seja, quantas arestas saem de cada nó?
2. Qual propriedade deve ser testada em um nó?
3. Em que momento a subdivisão de um nó deve parar, ou seja, quando ele se torna folha?
4. Caso a árvore fique muito grande, como realizar a poda?
5. Caso um nó folha não seja puro, qual rótulo deve ser atribuído?
6. Como lidar com dados faltantes?

Número de divisões

O número de divisões de um nó (*branching factor*) está relacionado à Questão 02, podendo ser diferente em cada nó. Cabe observar que divisões por fatores maiores que dois podem ser convertidas em divisões binárias.



(a)



(b)

DHS foca em aprendizagem de árvores binárias.

Seleção de *queries* e impureza do nó

- O princípio consiste em criar uma árvore simples (compacta) e tão precisa quanto possível.
  - Princípio da Navalha de Occam: deve-se escolher o modelo mais simples que descreve os dados, contendo apenas as premissas estritamente necessárias.
- Uma estratégia consiste em escolher as decisões que conduzam a nós "puros".
  - Existem diferentes medidas matemáticas para a impureza, denotada por $i(N)$, de um nó N. Exemplo: entropia.

Medidas de impureza: entropia

A medida mais popular é a entropia:

$$i(N) = -\sum_j P(\omega_j) \log P(\omega_j)$$

Exemplos:

- Considere 10 amostras divididas em 3 classes, sendo 4 na primeira, 5 na segunda e 1 na terceira:

  $$P(\omega_1) = \frac{4}{10}, P(\omega_2) = \frac{5}{10} \text{e} P(\omega_3) = \frac{1}{10}$$

  Portanto, $i(N) = -0.4 \log 0.4 - 0.5 \log 0.5 - 0.1 \log 0.1 = 1.36$
- Quando todos os padrões pertencem a uma mesma classe, temos $i(N) = 0$, dado que $\log(1) = 0$.
- Considere agora que os padrões estão divididos igualmente em dois subconjuntos, ou seja, $P(\omega_j) = 0.5$:

  $$i(N) = -0.5 \log 0.5 - 0.5 \log 0.5 = 1$$

Medidas de impureza: Índice de Gini

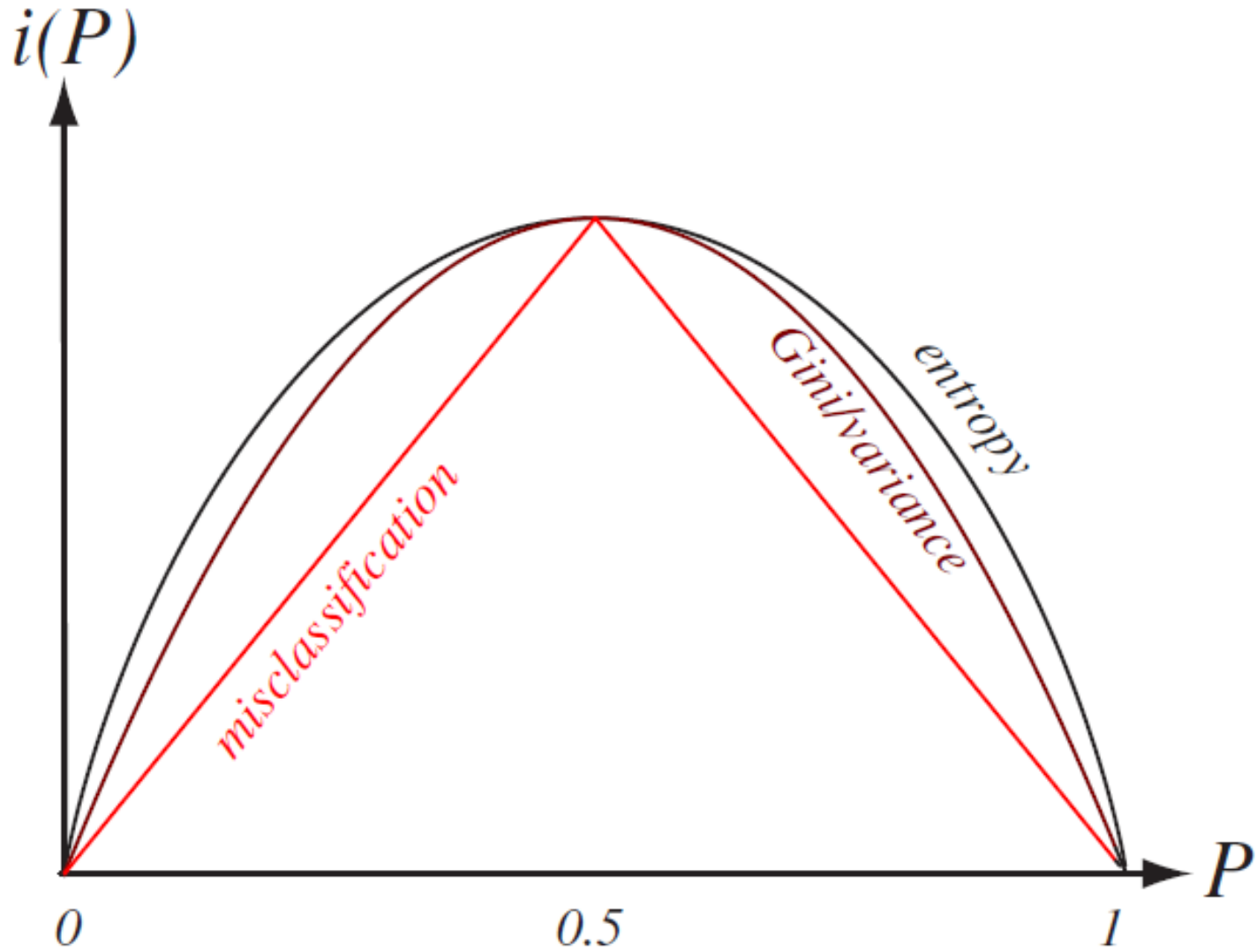
Outra medida corresponde ao *Índice de Gini*:

$$i(N) = \sum_{i \neq j} P(\omega_i) P(\omega_j) = \frac{1}{2} \left[1 - \sum_j P^2(\omega_j) \right]$$

que representa a taxa de erro esperada para o nó N se a categoria for selecionada aleatoriamente (a partir da distribuição de classes presentes no nó N). Exemplos:

- Para o exemplo anterior, temos:

$$i(N) = \frac{1}{2} \left[1 - 0.4^2 - 0.5^2 - 0.1^2 \right] = 0.29$$



Observe que, quando as decisões consideram apenas duas categorias, as medidas de impureza tem o mesmo valor máximo.

Impurity gradient

A questão é: dada uma árvore parcial, o próximo teste T deve ser baseado em qual característica?

- Uma heurística consiste em escolher a característica que mais reduza a impureza, isto é, que maximize o *impurity gradient*:

$$\Delta i(N) = i(N) - P_L i(N_L) - (1 - P_L) i(N_R)$$

  em que N_L e N_R denotam os nós descendentes da esquerda e direita, $i(N_L)$ e $i(N_R)$ são as impurezas associadas e P_L é a fração dos dados que irá para a sub-árvore a esquerda para um dado teste T.

- Caso a medida de entropia seja usada, isso corresponde a escolher a característica com maior **Ganho de Informação** (*Information Gain*).

Observe que a otimização é **local**: assim como a grande maioria dos métodos gulosos, não se pode assegurar que decisões ótimas locais conduzam ao ótimo global!

Exemplo: *impurity gradient*

Considere um exemplo com 100 padrões, sendo 40 pertencentes a C_1, 30 a C_2 e 30 a C_3. Suponha que um dado atributo X os dividiu entre os nós N_L e N_R de acordo com a tabela abaixo.

N_L	N_R	Total	Classe
40	0	40	1
10	20	30	2
10	20	30	3

Ao considerar a entropia como medida de impureza, temos:

- $i(N) = -\frac{40}{100}\log\frac{40}{100} - \frac{30}{100}\log\frac{30}{100} - \frac{30}{100}\log\frac{30}{100} = 1.38$
- $i(N_L) = -\frac{40}{60}\log\frac{40}{60} - \frac{10}{60}\log\frac{10}{60} - \frac{10}{60}\log\frac{10}{60} = 1.25$
- $i(N_R) = -\frac{20}{40}\log\frac{20}{40} - \frac{20}{40}\log\frac{20}{40} = 1$

Portanto:

$$\Delta i(N) = 1.38 - \left(\frac{60}{100} \times 1.25\right) - \left(\frac{40}{100} \times 1\right) = 0.23$$

Information Gain (IG)

O IG (Ganho de Informação) mede a efetividade de um atributo em classificar um conjunto de treinamento. Ele mede a redução da Entropia, ou incerteza, ao selecionar um determinado atributo.

o IG de um atributo A também pode ser escrito como:

$$IG(S, A) = E(S) - \sum_{v \in valores(A)} \frac{|S_v|}{|S|} E(S_v)$$

em que $E(S)$ denota a entropia da coleção de instâncias, S (com exemplos positivos e negativos).

Exemplo: jogar tênis

Considere os seguintes exemplos de treinamento para "jogar tênis".

Dia	Panorama	Temper.	Umidade	Vento	Jogar Tênis
1	Ensolarado	Quente	Alta	Fraco	Não
2	Ensolarado	Quente	Alta	Forte	Não
3	Nublado	Quente	Alta	Fraco	Sim
4	Chuvoso	Intermediária	Alta	Fraco	Sim
5	Chuvoso	Fria	Normal	Fraco	Sim
6	Chuvoso	Fria	Normal	Forte	Não
7	Nublado	Fria	Normal	Forte	Sim
8	Ensolarado	Intermediária	Alta	Fraco	Não
9	Ensolarado	Fria	Normal	Fraco	Sim
10	Chuvoso	Intermediária	Normal	Fraco	Sim
11	Ensolarado	Intermediária	Normal	Forte	Sim
12	Nublado	Intermediária	Alta	Forte	Sim
13	Nublado	Quente	Normal	Fraco	Sim
14	Chuvoso	Intermediária	Alta	Forte	Não

Exemplo: jogar tênis

Qual o IG do atributo vento? Observe que no total são 14 exemplos, $[9+, 5-]$, sendo que 6 dos exemplos positivos e dois dos negativos são definidor por `Vento = Fraco` e os demais por `Vento = Forte`:

- $S = [9+, 5-]$
- $S_{\texttt{Fraco}} = [6+, 2-]$
- $S_{\texttt{Forte}} = [3+, 3-]$

Exemplo: jogar tênis

Sendo

$$IG(S, A) = E(S) - \sum_{v \in valores(A)} \frac{|S_v|}{|S|} E(S_v)$$

temos:

$$E(S) = -\frac{9}{14} \log \frac{9}{14} - \frac{5}{14} \log \frac{5}{14} = 0.94$$

$$E(S_{\texttt{Fraco}}) = -\frac{6}{8} \log \frac{6}{8} - \frac{2}{8} \log \frac{2}{8} = 0.811$$

$$E(S_{\texttt{Forte}}) = -\frac{3}{6} \log \frac{3}{6} - \frac{3}{6} \log \frac{3}{6} = 1$$

$$IG(S, \texttt{vento}) = 0.94 - \frac{8}{14} 0.811 - \frac{6}{14} 1 = 0.048$$

Exemplo: jogar tênis

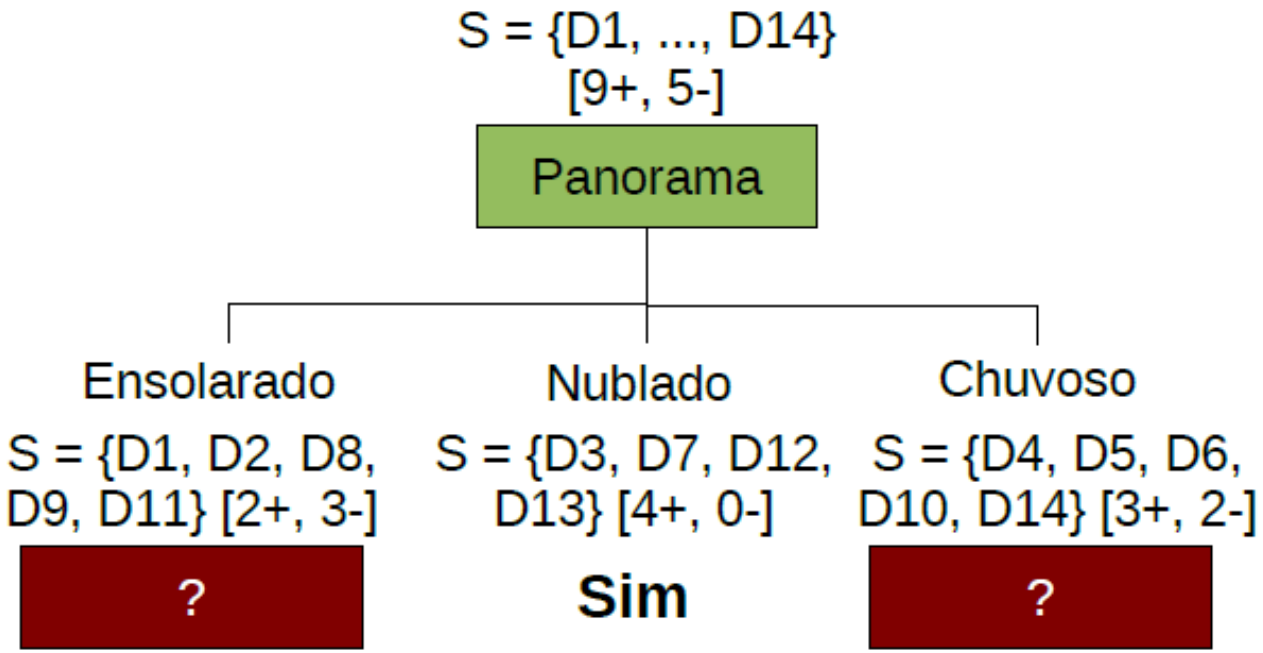
Ao calcular estes valores para os demais atributos, temos que:

- $IG(S, \texttt{panorama}) = 0.246$
- $IG(S, \texttt{umidade}) = 0.151$
- $IG(S, \texttt{vento}) = 0.048$
- $IG(S, \texttt{temperatura}) = 0.029$

Por ser o que mais reduz o nível de incerteza, o atributo com maior IG é selecionado para ser raiz da árvore de decisão. Os nós filhos são criados a partir dos possíveis valores do atributo `panorama`.

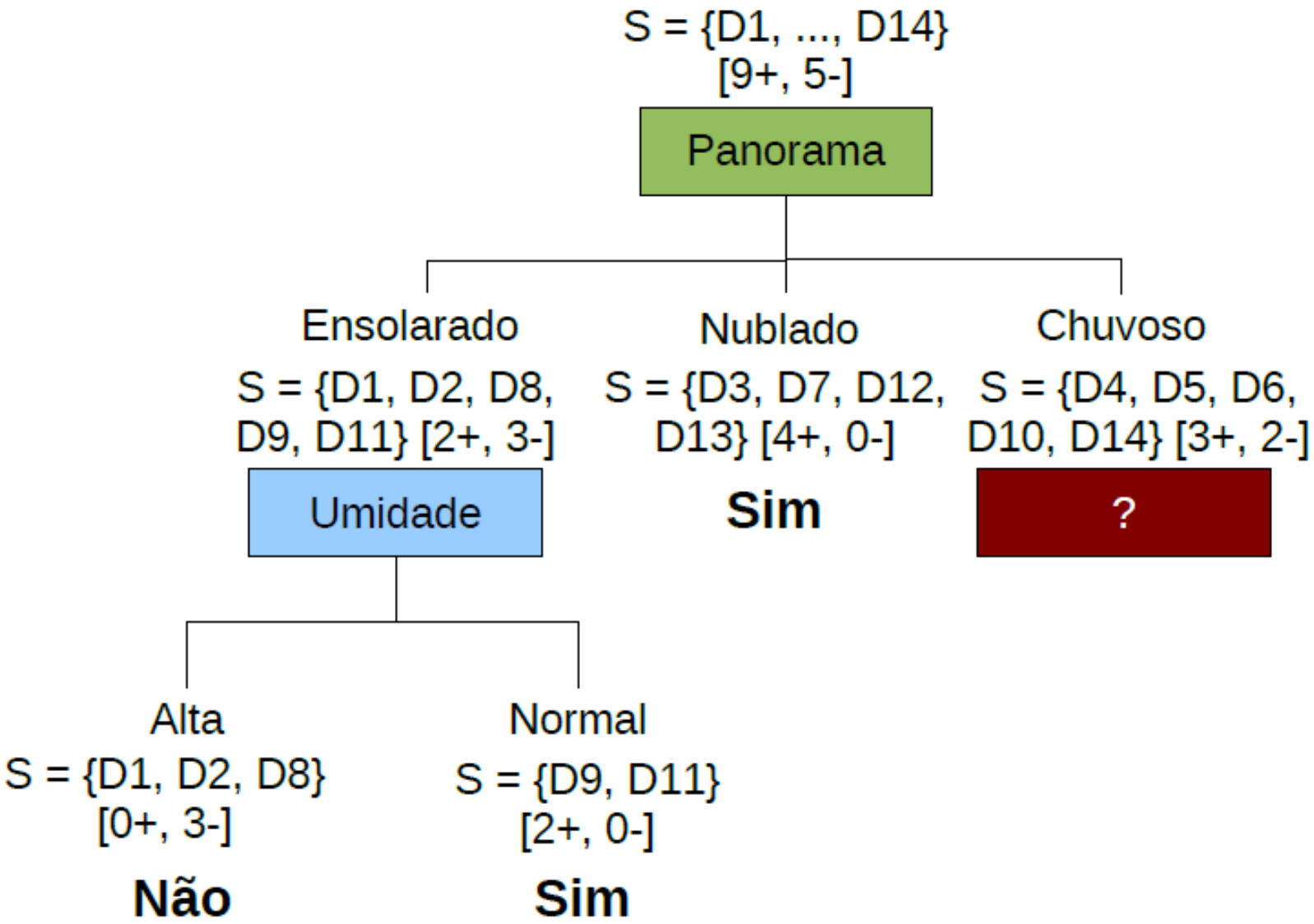
Exemplo: jogar tênis

Devemos proceder da mesma maneira para os demais ramos que surgem a partir da raiz, exceto para aqueles cuja entropia é zero.

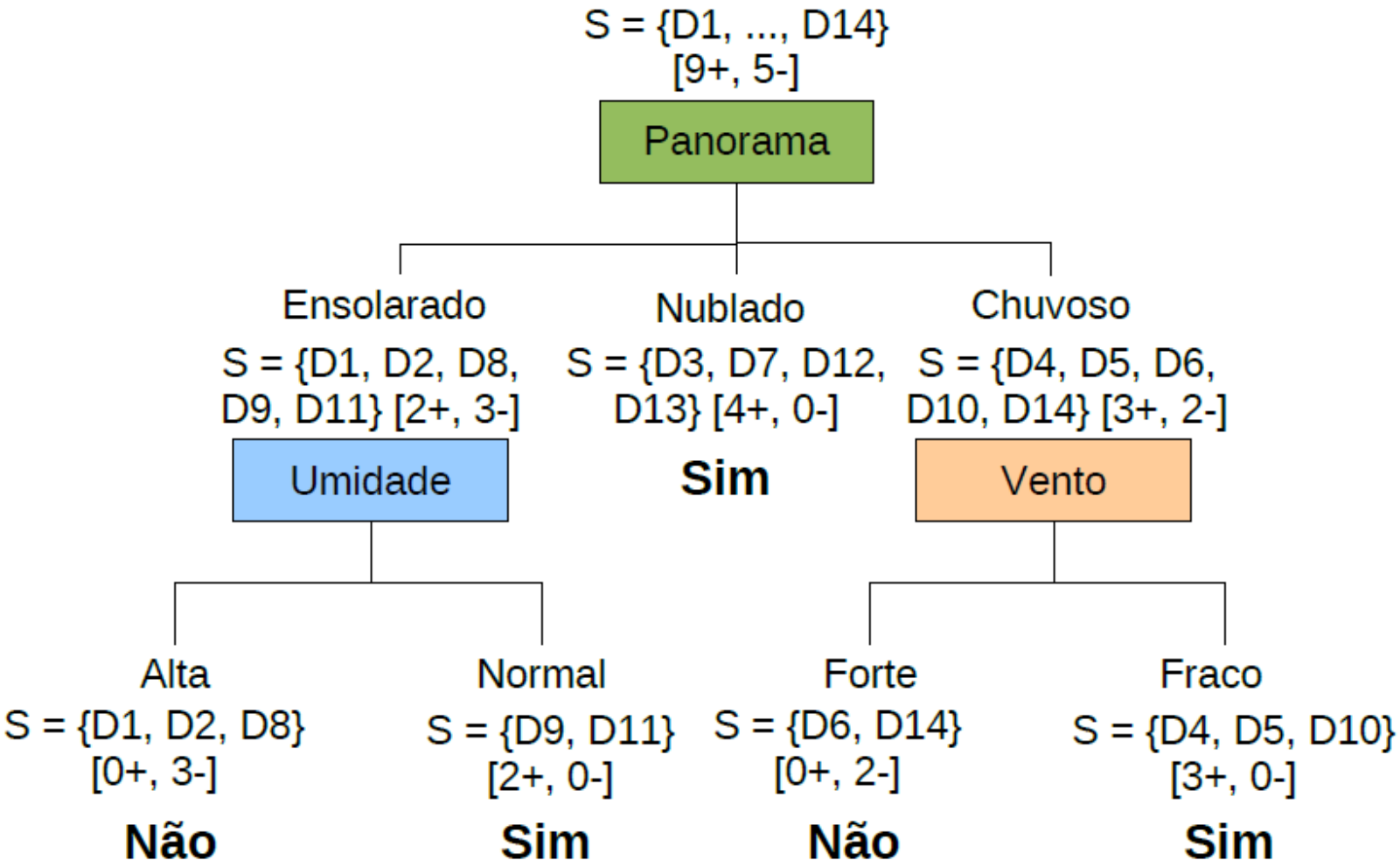


Atributos existentes incorporados acima de determinado nó não entram na avaliação do seu IG. Por exemplo, o atributo `panorama` não será mais avaliado.

Exemplo: jogar tênis



Exemplo: jogar tênis



Qual medida de impureza escolher?

- A função/medida de impureza raramente influencia a precisão do classificador final.
- Devido à sua simplicidade, a entropia é frequentemente escolhida.
- Na prática, o critério de parada e o método de poda são mais importantes!

Quando parar de dividir?

- Se a árvore "crescer" até que cada nó tenha a mínima impureza, tipicamente ocorre o *overfit*. Por outro lado, parar a divisão precocemente faz com que erro de classificação seja maior que o aceitável.
- Os critérios comumente utilizados para decidir quando interromper o processo são[1]
  1. Validação e validação cruzada.
  2. Estabelecer um limiar para o *impurity gradient*.
  3. Minimizar um termo que calcule a complexidade da árvore.
  4. Significância estatística do *impurity gradient*.

[1] Ver Apêndice para maiores informações.

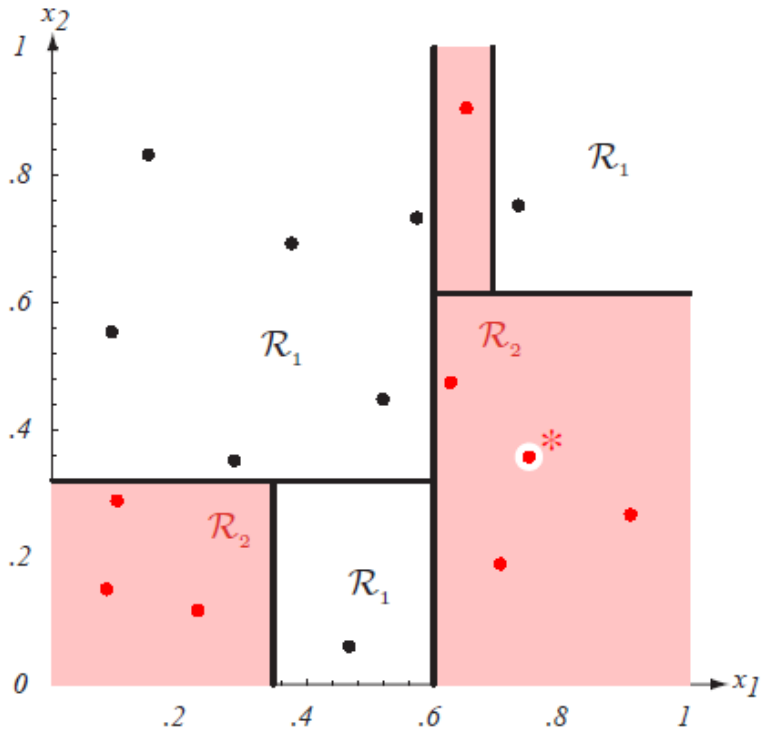
Poda

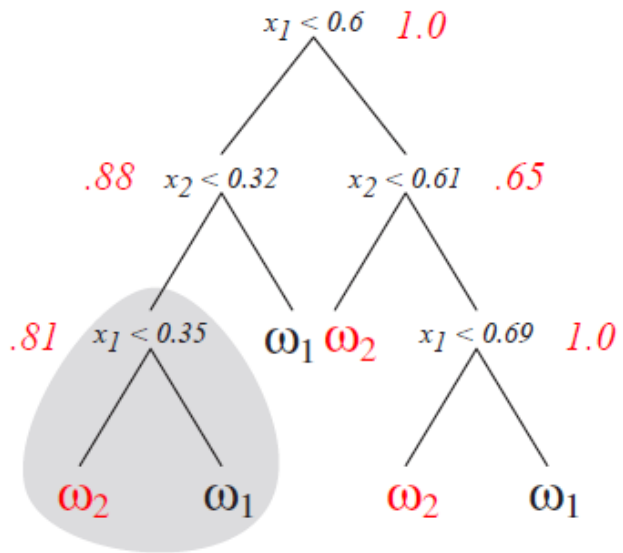
A poda é aplicada após a árvore completa ser construída, ou seja, quando todos nós folha possuem impureza mínima. Os nós folha ligados a um mesmo antecedente são eliminados caso isso implique em um aumento da impureza.

- Benefício 01: evita-se o fenômeno "*horizon effect*".
- Benefício 02: usa todos os dados para treinamento.
- Benefício 03: a árvore não precisa ficar balanceada.
- Desvantagem: possui um custo computacional elevado, podendo ser proibitivo em alguns casos.

Atribuição de rótulos aos nós-folha

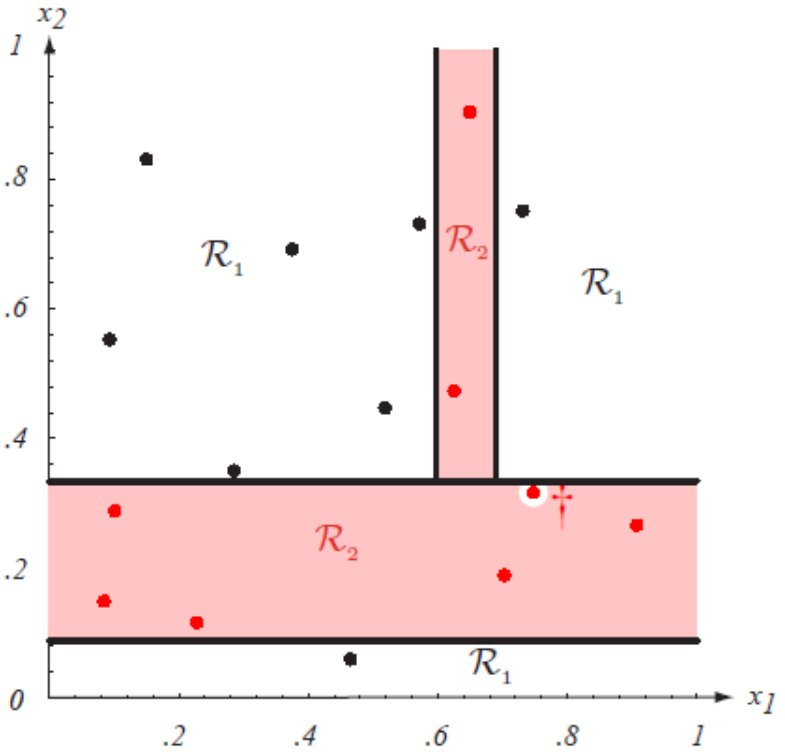
A atribuição de rótulos é um processo fácil: caso o nó seja puro, basta atribuir o rótulo da categoria associada. Caso contrário, basta atribuir o rótulo da classe que possui mais amostras representadas.

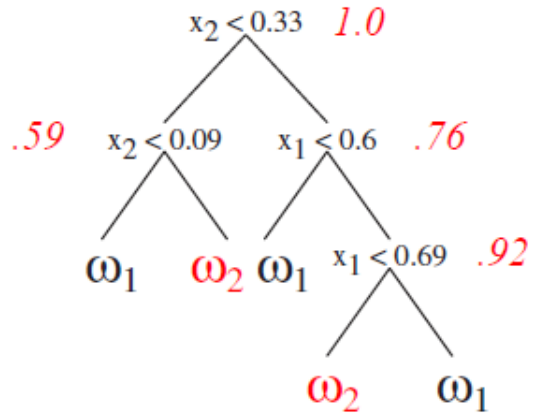




Instabilidade na construção da árvore

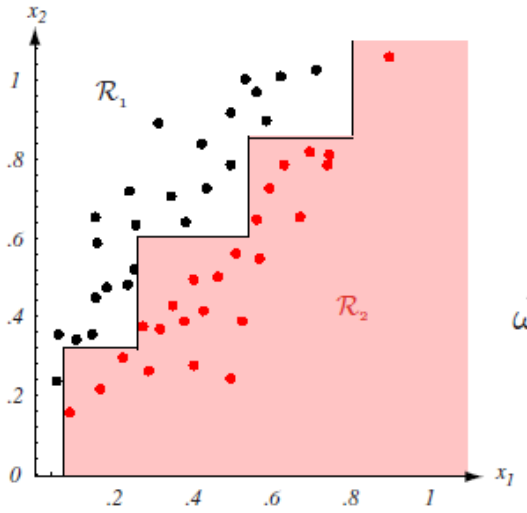
O exemplo abaixo ilustra o quanto a abordagem é sensível aos dados de treinamento. Uma pequena alteração do ponto marcado anteriormente com $*$ para †resultou em uma árvore distinta.

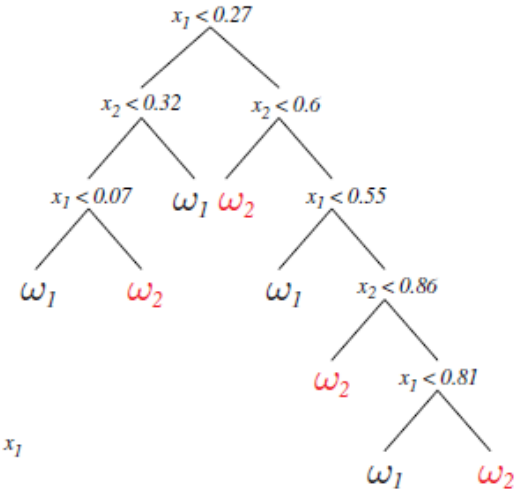


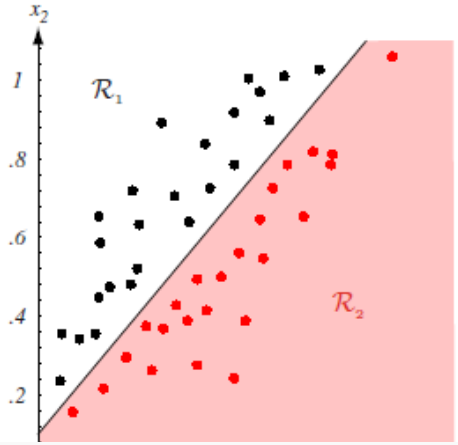


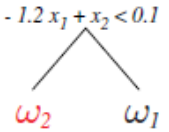
Importância da escolha das características

Assim como na maioria das técnicas de reconhecimento de padrões, a escolha de características adequadas tem influência na precisão, generalização e complexidade da árvore construída. Essa é uma instância do *Ugly Duckling Principle*.

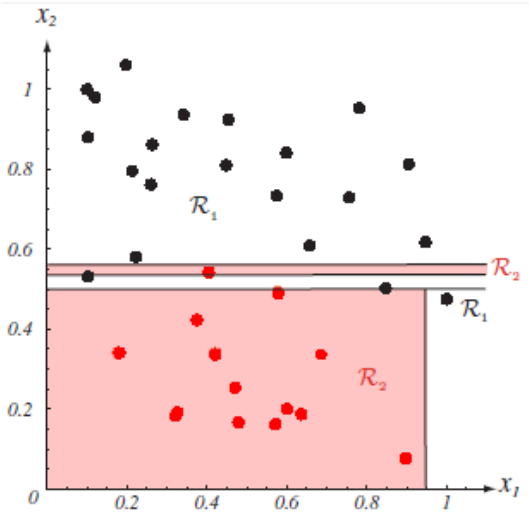


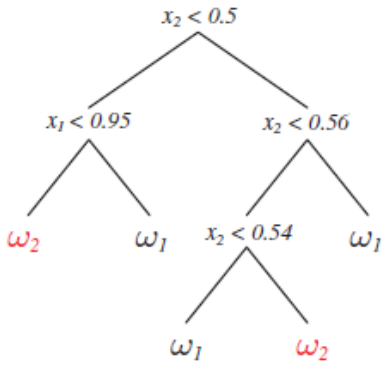


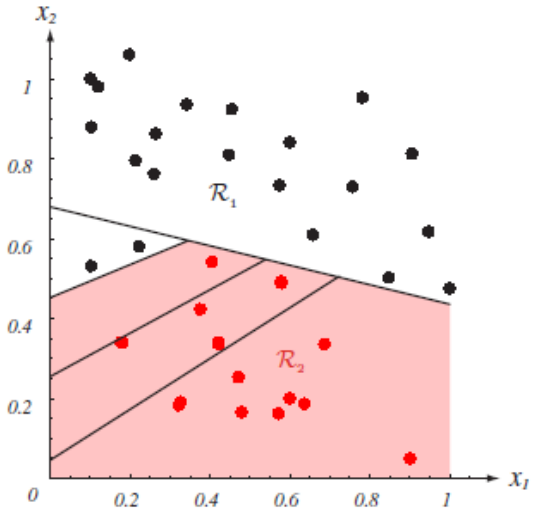


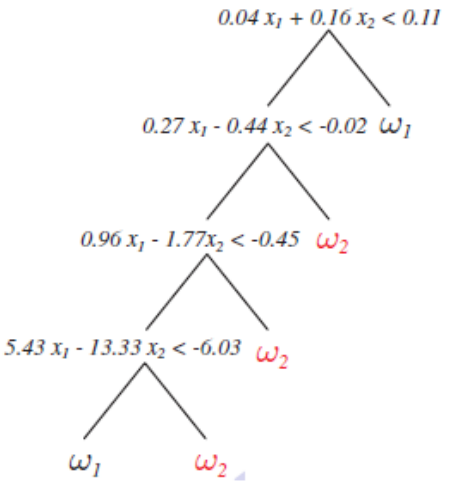


Além disso, a utilização de mais de um atributo em uma decisão pode melhorar a precisão e a capacidade de generalização da árvore.







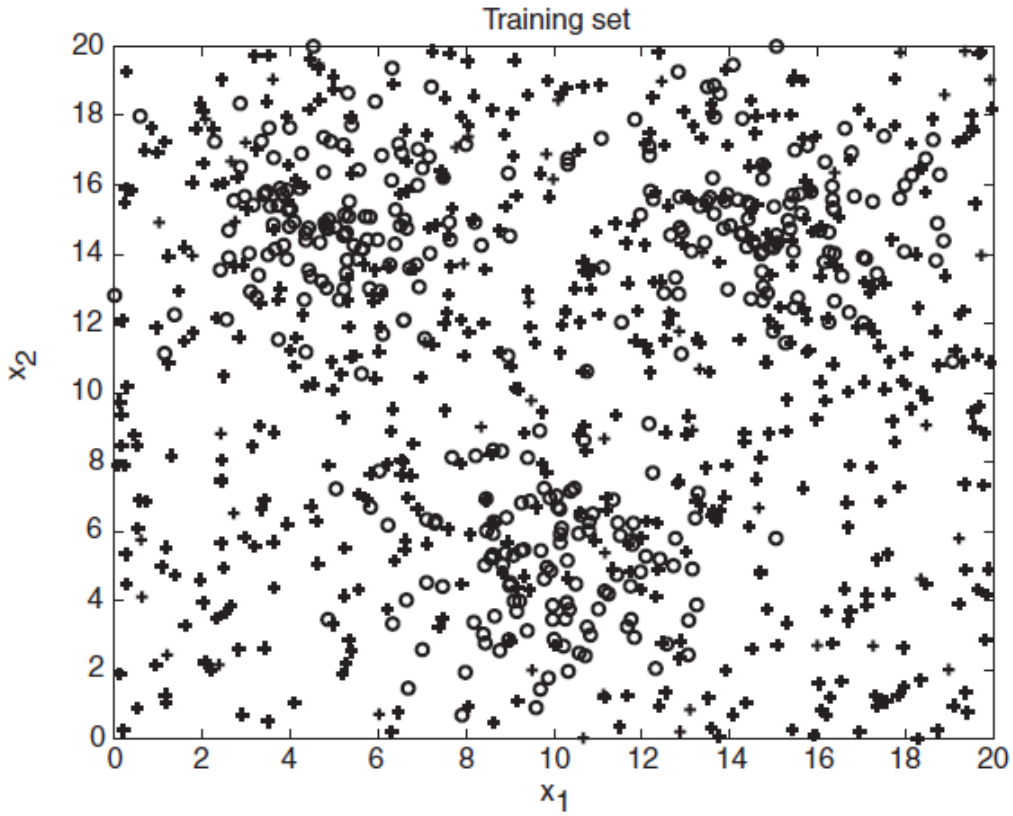


Overfitting

- Os erros de um classificador podem ser classificados em erros de treinamento (ao avaliar os dados utilizados para a construção da árvore) e erros de generalização (na classificação de novos dados).
- Um bom modelo deve ter taxas de erro baixas em ambos os casos.
- Quando o desempenho é bom na base de treinamento, mas o erro de generalização é alto, temos o *overfitting*.

Overfitting

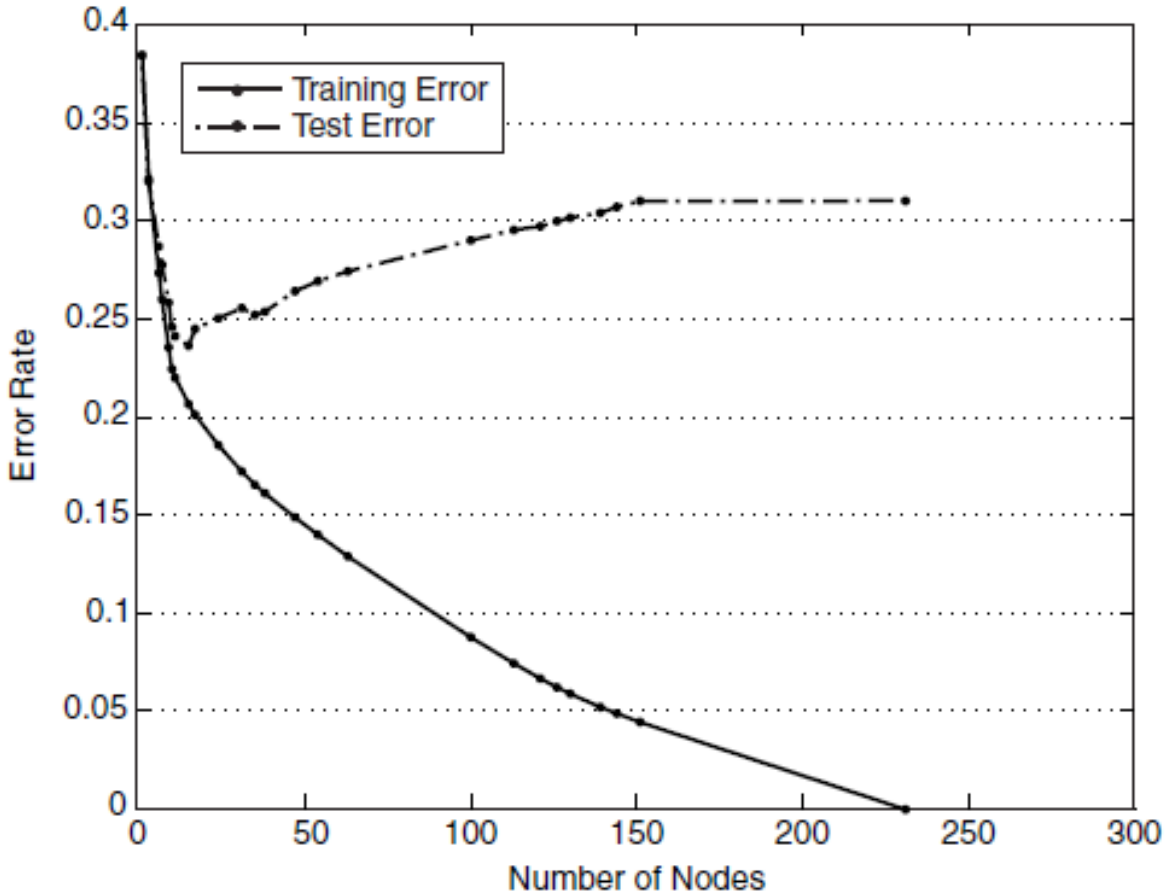
Por exemplo, considere o conjunto de dados ilustrado abaixo:



A classe o foi gerada a partir de uma mistura de três distribuições Gaussianas e tem 1200 pontos. Já a classe + foi gerada a partir de uma distribuição uniforme e tem 1800 pontos.

Overfitting

Para investigar o efeito do *overfitting*, diferentes níveis de poda são aplicados a uma árvore completa (foi utilizado o índice de Gini como medida de impureza).



Observe as taxas de erro: *underfitting*×*overfitting*.

Overfitting

Fatores como ruído e dados faltantes podem tornar o modelo mais suscetível ao *overfitting*. Contudo, a complexidade do modelo tem influência significativa:

- estimativa de resubstituição (*resubstitution estimate*): seleciona o modelo com menor erro para os dados de treinamento;
- incorporar a complexidade do modelo, ou seja, buscar por modelos mais simples (Occam's Razor):
  - Estimativa de erro pessimista
  - Princípio do *Minimum Description Length*
  - Estimativa de limites estatísticos
  - Usar um conjunto de validação

Apêndices

Características

- Por ser uma técnica não-paramétrica, não requer suposições sobre qual a distribuição de probabilidade associada, por exemplo.
- Alguns conjuntos de dados necessitam que as decisões considerem mais de uma característica para particionar corretamente os dados.
- A classificação é eficiente, com um pior caso de $O(h)$, em que h é a altura da árvore.
- As árvores são relativamente fáceis de interpretar.
- São robustas à presença de ruído, em especial de abordagens para evitar *overfitting* são empregadas.

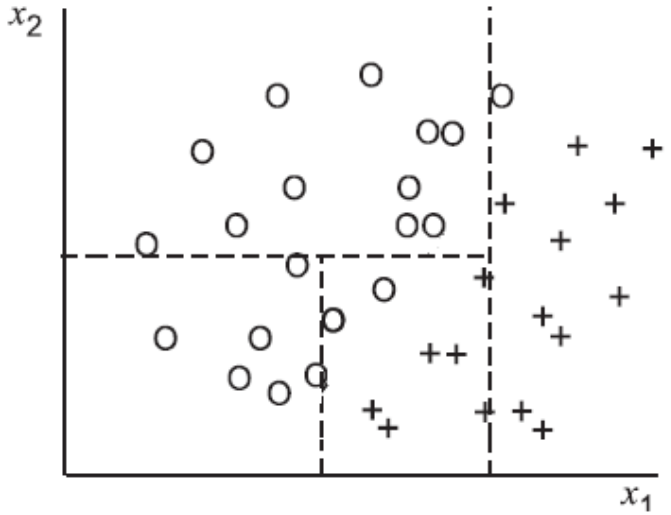
Características

- Embora a presença de atributos redundantes não tenha tanto impacto na precisão, pode conduzir a árvores maiores que o necessário (pré-processamento).
- Como as abordagens para construção são tipicamente top-down e recursivas, em níveis mais avançados os dados podem não ser suficientes para ter significância estatística (*data fragmentation*).
- Uma subárvore pode ser replicada em diversos pontos, tornando a árvore mais complexa que o necessário.
- Também pode ser representada por um conjunto de regras (IF-THEN).
- *White model*, diferentemente de uma rede neural que é muitas vezes referenciada como um *black-box*.

Desvantagens

Dentre as desvantagens das árvores de decisão, estão:

- O tempo de projeto pode ser significativo: por exemplo, o número de divisões oblíquas pode ser exponencial em relação à quantidade de padrões de treinamento. A aprendizagem de uma árvore ótima é um problema NP-completo.
- Árvores de decisão simples (*axis-parallel*) não representam bem regiões que não são retangulares.



Desvantagens

- Alguns conceitos são de difícil aprendizagem em árvores de decisão, gerando árvores extremamente grandes, por exemplo, XOR e paridade.
- Podem criar árvores extremamente complexas que levam ao *overfitting*.
- Utiliza heurísticas (*"'the problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts"*).
- Pode não gerar a melhor árvore.
- Podem ter bias no caso de classe dominante.

Quando parar de dividir?

- Se a árvore "crescer" até que cada nó tenha a mínima impureza, tipicamente ocorre o *overfit*. Por outro lado, parar a divisão precocemente faz com que erro de classificação seja maior que o aceitável.
- Os critérios comumente utilizados para decidir quando interromper o processo são:
  1. Validação e validação cruzada.
  2. Estabelecer um limiar para o *impurity gradient*.
  3. Minimizar um termo que calcule a complexidade da árvore.
  4. Significância estatística do *impurity gradient*.

Caso 01: validação (cruzada)

- Na validação, uma porcentagem do conjunto de treinamento é escolhido como conjunto de validação, e a árvore é treinada com o restante das amostras. A divisão dos nós continua sendo realizada até que o erro de classificação do conjunto de validação seja minimizado (ou, em outras palavras, até que a capacidade de generalização seja maximizada).
- Na validação cruzada, são considerados diferentes subconjuntos escolhidos de forma independente. Exemplo: *leave-one-out*.

Caso 02: limiarização do *impurity gradient*

O processo de divisão é interrompido em um dado nó se a sua impureza for melhor que um determinado limiar, β:

$$\max_s \Delta i(s) \leq \beta$$

- Benefício 1: a árvore é construída (treinada) com base no conjunto de dados completo, o que não ocorre no Caso 01.
- Benefício 2: os nós-folha estão em diferentes níveis da árvore, representando de forma mais adequada as diferentes complexidades de cada atributo/característica.
- Desvantagem: a escolha do limiar (β) não é trivial.
  - Outra possibilidade consiste em delimitar uma quantidade mínima de amostras para que um dado nó seja subdivido. Assim, o tamanho das partições está relacionado à densidade dos dados (similar ao que ocorre no knn).

Casos 03 e 04

Caso 3 Termo de complexidade: definir uma nova função de critério global, a qual considera a relação entre complexidade e precisão:

$$\alpha \times \mathit{size} + \sum_{\mathit{leafnodes}} i(N)$$

em que *size* pode representar o número de arestas ou nós e α é uma constante positiva.

Caso 04 Teste de significância estatística: durante a construção da árvore, estima-se a distribuição de todos os Δi para os nós atuais. Em cada candidato a divisão, estima-se se ele é estatisticamente relevante (ou seja, se ele difere significativamente de uma divisão aleatória) utilizando-se ferramentas tal como o teste de χ^2.