ALGORITMOS DE INDUÇÃO DE ÁRVORES
COMO CALCULAR O GANHO?

Cristiane Neri Nobre

Árvore de Decisão – Como gerar?

Até o momento, já aprendemos a avaliar a importância do atributo de **forma visual**.

Mas, como avaliar **matematicamente** o **ganho** de cada atributo?

Árvore de Decisão – Como gerar?

Na maioria dos indutores de árvores de decisão,
as funções de divisão discreta são univariadas, isto é, um nó interno é dividido de acordo com o valor de um único atributo.

Consequentemente, o indutor procura o melhor atributo sobre o qual realizar a divisão.

Árvore de Decisão – Como gerar?

Entropia é o cálculo do ganho de informação baseado em uma medida utilizada na teoria da informação.

Caracteriza a (im)pureza dos dados: em um conjunto de dados, é uma medida da falta de homogeneidade dos dados de entrada em relação a sua classificação.

Dado um conjunto de entrada (S) que pode ter *c* classes distintas, a entropia de *S* será dada por

$$Entropia(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

Árvore de Decisão – Como gerar?

Assim, a entropia é uma medida da **aleatoridade** de uma variável

A física usa o termo entropia para descrever a quantidade de desordem associada a um sistema. Na teoria da informação, este termo tem uma significado semelhante, -- ele mede o grau de desordem de um conjunto de dados.

Escolha de testes de atributos

• O esquema usado na aprendizagem de árvores de decisão para selecionar atributos é projetado para **minimizar** a **profundidade** da árvore final.

• A ideia é escolher o **atributo** que **vá o mais longe** possível na tentativa de fornecer uma classificação exata dos exemplos.

• Um atributo perfeito divide os exemplos em conjuntos que são todos positivos ou todos negativos.

- *Clientes* – bastante bom
- *Tipo* – realmente inútil

Escolha de testes de atributos

• Assim, tudo o que precisamos é de uma medida formal de “bastante bom” e “realmente inútil”

• A função ESCOLHER-ATRIBUTO deverá ter seu valor **máximo** quando o **atributo for perfeito**, e seu valor **mínimo** quando o atributo for **absolutamente inútil**.

• Uma medida apropriada é a quantidade esperada de informações fornecidas pelo atributo, que é calculada através de uma expressão matemática

Escolha de testes de atributos

• Para se entender a **noção de informações**, pode-se pensar como a resposta a uma pergunta. Assim, a quantidade de informações contidas na resposta depende do conhecimento anterior do indivíduo.

• Quanto menos se sabe, mais informações são fornecidas.

• A **teoria da informação** mede o conteúdo de informação em bits.

Escolha de testes de atributos

• Um **bit de informação** é suficiente para responder a uma pergunta do tipo sim/não sobre a qual não se tem nenhuma ideia. Por exemplo se lançarmos uma moeda imparcial qual a quantidade de informação necessária?

Em geral, se cada resposta possível v_i têm probabilidade $P(v_i)$, então o conteúdo de informação "I" da resposta real é dado por:

$$I(P(V_1),\ldots,P(v_n)) = \sum_{i=1}^{n} -P(v_i)\log_2 P(v_i)$$

Resumindo:

$$\text{Entropia}(S) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

Escolha de testes de atributos

Ou seja, a equação da entropia, usada em algoritmos de árvore de decisão como o ID3, C4.5 e seus sucessores, mede o nível de **impureza** ou **incerteza** em um conjunto de dados. A entropia é derivada da teoria da informação e é utilizada para determinar a qualidade de uma divisão (split) dos dados em uma árvore de decisão.

$$\text{Entropia}(S) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

Onde:

- *S* é o conjunto de dados.
- *n* é o número de classes possíveis (ou categorias) no conjunto.
- p_i é a proporção de elementos em *S* pertencentes à classe *i*

Explicação dos termos da equação

$$\text{Entropia}(S) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

- P_i: Refere-se à probabilidade de um dado elemento pertencer à classe *i*. Ou seja, é a razão entre a quantidade de elementos da classe i e o total de elementos no conjunto S.

- $Log_2(p_i)$: Refere-se ao logaritmo de base 2 da probabilidade p_i. A escolha da base 2 está relacionada com o fato de a entropia ser medida em "bits" na teoria da informação.
  Importante: Se desejar que o ganho e a entropia esteja entre 0 e 1, a base do log precisa ser igual ao número de classes.

- Sinal negativo (-): Garante que a entropia seja um valor positivo, pois os logaritmos de probabilidades são números negativos (quando $p_i < 1$)

Interpretação da entropia

- **Entropia alta**: Quando a entropia é alta, significa que as classes estão bem misturadas, ou seja, há uma incerteza maior. Isso indica que o conjunto de dados é "impuro" em termos de classificação.

- **Entropia baixa**: Quando a entropia é baixa, o conjunto de dados é mais homogêneo. Se todos os exemplos em S pertencem à mesma classe, a entropia será zero, representando total pureza e ausência de incerteza.

Referências para entropia

A equação da entropia, conforme usada em árvores de decisão e outros contextos de teoria da informação, foi originalmente proposta por **Claude Shannon** em 1948 em seu trabalho fundamental **"A Mathematical Theory of Communication"**. Este artigo lançou as bases para a teoria da informação e é a principal referência para a equação da entropia.

1. **Shannon, C. E. (1948)**. *A Mathematical Theory of Communication*. The Bell System Technical Journal, 27, 379–423.

2. **Quinlan, J. R. (1986)**. *Induction of Decision Trees*. Machine Learning, 1(1), 81–106.

Escolha de testes de atributos

- No caso do lançamento de uma moeda imparcial, temos:

$$I(\frac{1}{2},\frac{1}{2}) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}$$

$$I(\frac{1}{2},\frac{1}{2}) = -\frac{1}{2}(-1) - \frac{1}{2}(-1)$$

$$I(\frac{1}{2},\frac{1}{2}) = \frac{1}{2} + \frac{1}{2} = 1\,\text{bit}$$

Ou seja**, um bit de informação** é suficiente para responder a uma pergunta **sim/não** sobre a qual não se tem nenhuma ideia, como o lançamento de uma moeda imparcial.

Escolha de testes de atributos

- E se a moeda for adulterada (viciada) para dar 99% de cara?

$$I(\frac{1}{100},\frac{99}{100}) = -\frac{1}{100}\log_2\frac{1}{100} - \frac{99}{100}\log_2\frac{99}{100}$$

$$I(\frac{1}{100},\frac{99}{100}) = -\frac{1}{100}(\text{-}0{,}0145) - \frac{99}{100}(\text{-}6{,}64386)$$

$$I(\frac{1}{100},\frac{99}{100}) = 0{,}014355 + 0{,}066439 = 0{,}080793$$

Escolha de testes de atributos

- Ou seja, como a **probabilidade de caras tente a 1**, a **informação da resposta** tende a 0.

- Para a aprendizagem em árvores de decisão, a pergunta que precisa ser respondida é: para um dado exemplo, qual é a **classificação correta**?

Escolha de testes de atributos

• Assim, uma estimativa das probabilidades das respostas possíveis antes de quaisquer atributos serem testados é dada pelas proporções de exemplos positivos e negativos no conjunto de treinamento.

Vamos supor que o conjunto de treinamento contenha *p* exemplos positivos e *n* exemplos negativos. Então uma estimativa das informações contidas em uma resposta correta é:

$$I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Escolha de testes de atributos

• Um teste em um único atributo A normalmente nos fornecerá algumas informações.

• Podemos medir exatamente quantas informações ainda precisaremos depois do teste do atributo.

• Qualquer atributo A divide o conjunto de treinamento E em subconjuntos E_1, …, E_v de acordo com seus valores para A, onde A pode ter *v* valores distintos.

Escolha de testes de atributos

• Cada subconjunto E_i tem P_i exemplos positivos e n_i exemplos negativos

• Assim, se seguirmos ao longo dessa ramificação, precisaremos de

$$I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

bits de informação para responder à pergunta.

Escolha de testes de atributos

• Um exemplo escolhido ao acaso a partir do conjunto de treinamento tem o i-ésimo valor para o atributo com probabilidade $(p_i+n_i)/(p+n)$ e assim, em média, depois de testar o atributo A, precisaremos de:

$$\text{Entropia(atributo)} = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} I\left(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right)$$

bits de informação para classificar o exemplo.

Escolha de testes de atributos

• O **ganho de informação** a partir do teste de atributo é a diferença entre o requisito de informação original e o novo requisito:

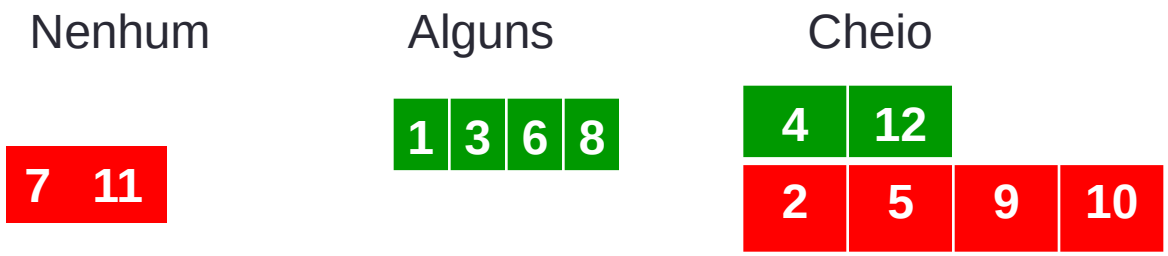
$$ganho(atributo) = Entropia(classe) - Entropia(atributo)$$

• A heurística usada na função ESCOLHER-ATRIBUTO é simplesmente escolher o atributo com o **maior ganho**.

Escolha de testes de atributos

Voltando aos atributos considerados no problema citado, qual o ganho do atributo **cliente**?

Escolha de testes de atributos



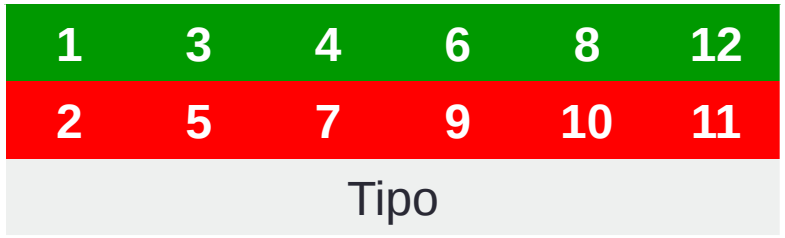
$$ganho(clientes) = 1 - \left[\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I\left(\frac{2}{6}, \frac{4}{6}\right) \right] \approx 0{,}541$$

Importante: Nesta equação, o valor 1 corresponde à entropia da classe: E(1/2;1/2), pois a base possui 6 exemplos com classe '**Sim**' e 6 exemplos da classe '**Não**'

Escolha de testes de atributos

E do atributo **Tipo**?

Escolha de testes de atributos



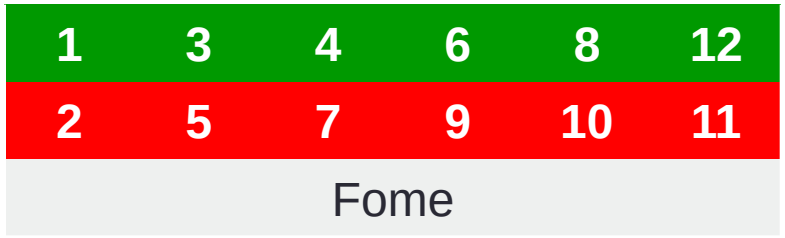
Francês	Italiano	Tailandês	Hambúrger
1	6	4 8	3 12
5	10	2 11	7 9

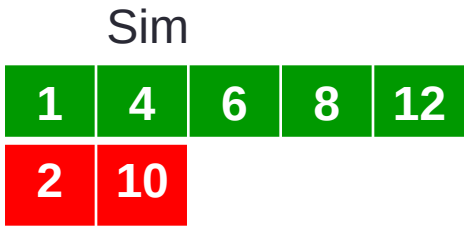
$$ganho(tipo) = 1 - \left[\frac{2}{12} I(\frac{1}{2},\frac{1}{2}) + \frac{2}{12} I(\frac{1}{2},\frac{1}{2}) + \frac{4}{12} I\left(\frac{1}{2},\frac{1}{2}\right) + \frac{4}{12} I\left(\frac{1}{2},\frac{1}{2}\right) \right] = 0$$

Escolha de testes de atributos

E do atributo **Fome**?

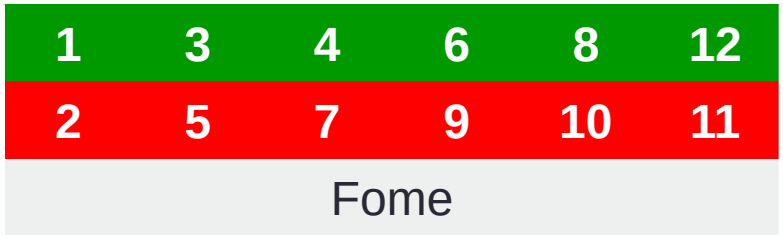
Escolha de testes de atributos



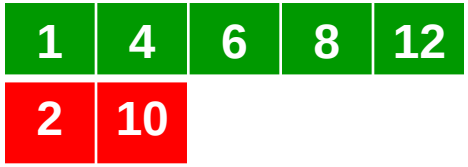


Exemplo	Alternativo	Bar	Sex/Sab	fome	Cliente	Preço	Chuva	Res	Tipo	Tempo	Conclusão (Vai esperar)?
X1	Sim	Não	Não	Sim	Alguns	RRR	Não	Sim	Francês	0-10	Sim
x2	Sim	Não	Não	Sim	Cheio	R	Não	Não	Tailandês	30-60	Não
x3	Não	Sim	Não	Não	Alguns	R	Não	Não	Hamburger	0-10	Sim
x4	Sim	Não	Sim	Sim	Cheio	R	Sim	Não	Tailandês	10-30	Sim
X5	Sim	Não	Sim	Não	Cheio	RRR	Não	Sim	Francês	>60	Não
X6	Não	Sim	Não	Sim	Alguns	RR	Sim	Sim	Italiano	0-10	Sim
X7	Não	Sim	Não	Não	Nenhum	R	Sim	Não	Hamburger	0-10	Não
X8	Não	Não	Não	Sim	Alguns	RR	Sim	Sim	Tailandês	0-10	Sim
X9	Não	Sim	Sim	Não	Cheio	R	Sim	Não	Hamburger	>60	Não
X10	Sim	Sim	Sim	Sim	Cheio	RRR	Não	Sim	Italiano	10-30	Não
X11	Não	Não	Não	Não	Nenhum	R	Não	Não	Tailandês	0-10	Não
X12	Sim	Sim	Sim	Sim	Cheio	R	Não	Não	Hamburger	30-60	Sim

Escolha de testes de atributos



Sim



Não

$$ganho(\,fome) = 1 - \left[\frac{7}{12} I(\frac{5}{7}, \frac{2}{7}) + \frac{5}{12} I\left(\frac{1}{5}, \frac{4}{5}\right) \right] \approx 0{,}196$$

Escolha de testes de atributos

Como medir a habilidade de um dado atributo discriminar as classes?

Existem muitas medidas!

Todas concordam em dois pontos:

– Uma divisão que mantêm as proporções de classes em todas as partições é inútil.

– Uma divisão onde em cada partição todos os exemplos são da mesma classe tem utilidade máxima

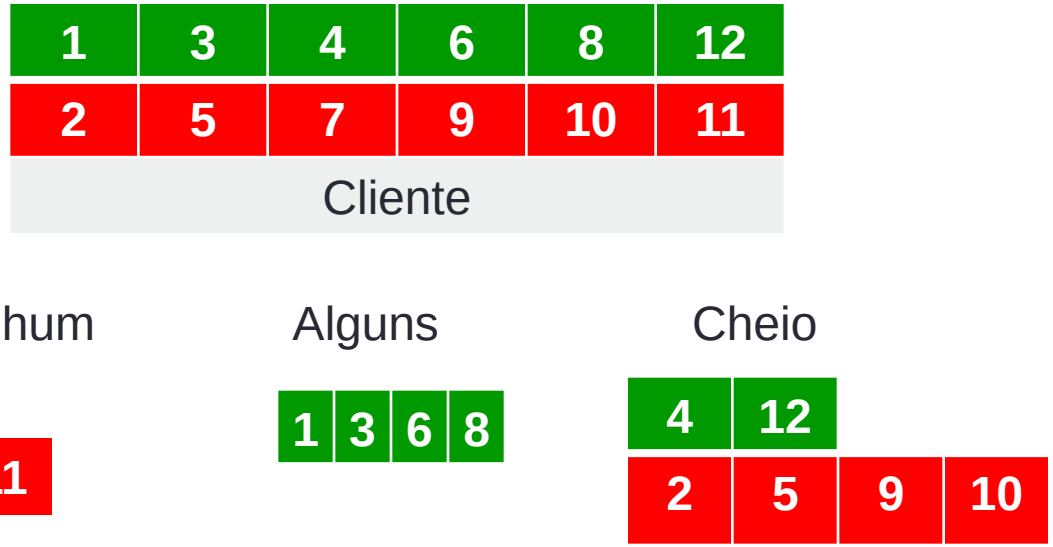
Resumindo: como funciona o algoritmo?

A ideia base do algoritmo é:

1. Escolher um atributo.
2. Estender a árvore adicionando **um ramo para cada valor do atributo**.
3. Passar os exemplos para as folhas (tendo em conta o valor do atributo escolhido)
4. Para cada folha
    1. Se todos os exemplos são da mesma classe, associar essa classe à folha 2.
    2. Senão repetir os passos 1 a 4

Exemplo

Considerando que o atributo **Cliente** estará na raiz da árvore (**aquele que teve o maior ganho**), que atributos estarão no segundo nível da árvore?



Exemplo

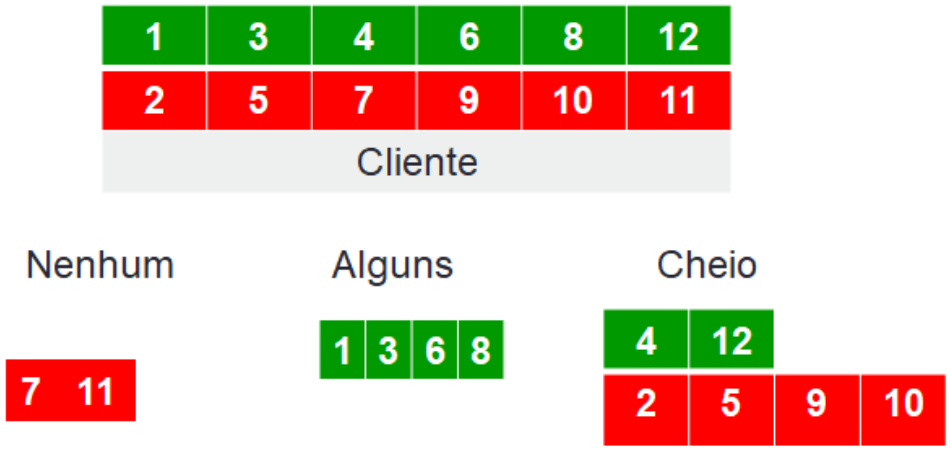
Em geral, depois que o primeiro teste de atributo separar os exemplos, cada resultado é um **novo problema de aprendizagem de árvore de decisão** em si, com menos exemplos e um atributo a menos.

Exemplo

Neste caso, quando o atributo **Cliente = Nenhum**, todas as instâncias tem a mesma classificação, todos são iguais a **Não**.

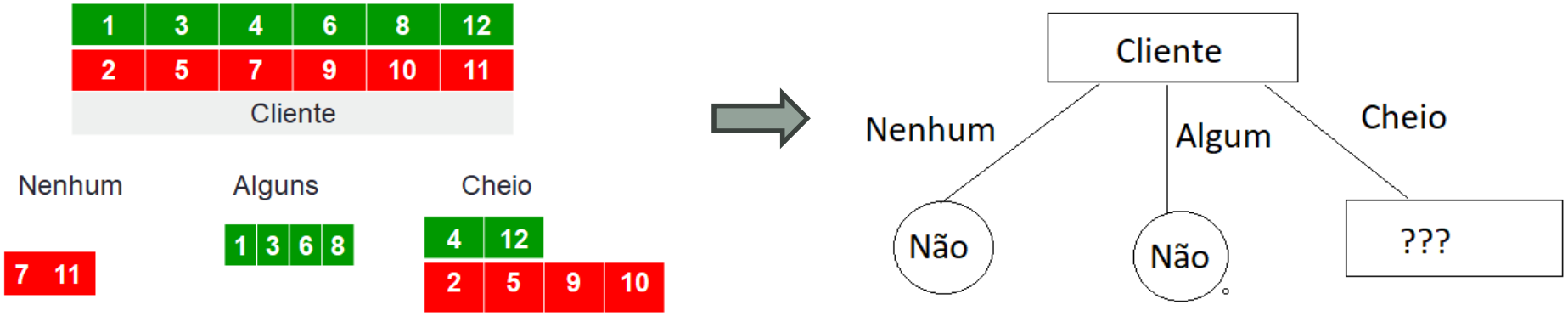
Quando o atributo **Cliente = Alguns**, todas as instâncias tem a mesma classificação, todos são iguais a **Sim**.

Mas quando o atributo **Cliente = Cheio**, temos 6 instâncias, sendo 2 da classe Sim e 4 da classe Não. Ou seja, neste caso, precisamos descobrir um outro atributo que possa separar estes exemplos.



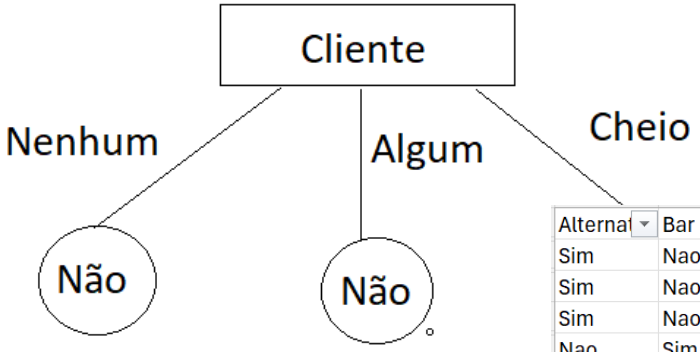
Exemplo

E como fazer isso? O processo de geração da árvore continuará apenas com estes 6 exemplos e precisaremos descobrir um novo atributo para ajudar a separá-los.



Exemplo

Ou seja, na ramificação da árvore para o tipo **Cliente=Cheio**, terão apenas 6 instâncias



Alternat	Bar	SexSab	fome	Cliente	Preco	Chuva	Res	Tipo	Tempo	Conclusao
Sim	Nao	Nao	Sim	Cheio	R	Nao	Nao	Tailandes	30-60	Nao
Sim	Nao	Sim	Sim	Cheio	R	Sim	Nao	Tailandes	10-30	Sim
Sim	Nao	Sim	Nao	Cheio	RRR	Nao	Sim	Frances	>60	Nao
Nao	Sim	Sim	Nao	Cheio	R	Sim	Nao	Hamburger	>60	Nao
Sim	Sim	Sim	Sim	Cheio	RRR	Nao	Sim	Italiano	10-30	Nao
Sim	Sim	Sim	Sim	Cheio	R	Nao	Nao	Hamburger	30-60	Sim

E agora executaremos o algoritmo considerando apenas estas 6 instâncias. Será que o novo atributo será fome? Preço? Chuva?

Façam os cálculos e descubra!

Árvore de Decisão – Algoritmo ID3

O Algoritmo ID3 (Iterative Dichotomiser 3)*

- Foi criado por Ross Quinlan, na universidade de Sydnei, Austrália em 1979
- O ID3 usa entropia e ganho de informação para construir a árvore de decisão
- É o algoritmo pioneiro em indução de árvore de decisão.
- É recursivo, baseado em busca gulosa, onde procura um conjunto de atributos que melhor dividem os exemplos (amostras), gerando sub-árvores.

* O nome deriva do fato de que o algoritmo construiu árvores de decisão dividindo iterativamente um conjuntos de dados até que eles possam ser classificados em uma de duas categorias distintas. Versões posteriores permitiram a classificação em mais de duas categorias, mas o "D" persistiu no nome.

Algoritmo ID3

ID3(*Exemplos*, *Atributo-objetivo*, *Atributos*)

// **ID3** retorna uma árvore de decisão que classifica corretamente os *Exemplos* determinados

// *Exemplos* são os exemplos de treinamento.

// *Atributo-objetivo* é o atributo cujo valor deve ser predito pela árvore.

// *Atributos* são uma lista de outros atributos que podem ser testados pela árvore de decisão.

Início

Crie um nodo *Raiz* para a árvore

Se todos os *Exemplos* são positivos

Então retorna a *Raiz* da árvore com o rótulo = **sim**

Se todos os *Exemplos* são negativos

Então retorna a *Raiz* da árvore com o rótulo = **não**

Se *Atributos* for vazio

Então retorna a *Raiz* da árvore com o rótulo = valor mais comum do *Atributo-objetivo* em *Exemplos*

Senão

A ← um atributo de *Atributos* que melhor classifica *Exemplos* (atributo de decisão)

Raiz ← **A** (rótulo = atributo de decisão **A**)

Para cada possível valor v_i de **A** faça

Acrescenta um novo arco abaixo da *Raiz*, correspondendo à resposta **A** = v_i

Seja $Exemplos_{vi}$ o subconjunto de *Exemplos* que têm valor v_i para **A**

Se $Exemplos_{vi}$ for vazio

Então acrescenta na extremidade do arco um nodo folha

com rótulo = valor mais comum do *Atributo-objetivo* em *Exemplos*

Senão acrescenta na extremidade do arco a sub árvore

ID3($Exemplos_{vi}$, *Atributo-objetivo*, *Atributos* – {**A**})

Retorna *Raiz* (aponta para a árvore)

Fim

Algoritmo ID3

Machine Learning 1: 81–106, 1986

Induction of Decision Trees

J.R. QUINLAN (munnari!nswitgould.oz!quinlan@seismo.css.gov)
Centre for Advanced Computing Sciences, New South Wales Institute of Technology, Sydney 2007, Australia

(Received August 1, 1985)

Key words: classification, induction, decision trees, information theory, knowledge acquisition, expert systems

Abstract. The technology for building knowledge-based systems by inductive inference from examples has been demonstrated successfully in several practical applications. This paper summarizes an approach to synthesizing decision trees that has been used in a variety of systems, and it describes one such system, ID3, in detail. Results from recent studies show ways in which the methodology can be modified to deal with information that is noisy and/or incomplete. A reported shortcoming of the basic algorithm is discussed and two means of overcoming it are compared. The paper concludes with illustrations of current research directions.

https://link.springer.com/article/10.1007/BF00116251

Algoritmo ID3

Limitações do Algoritmo ID3

- A principal limitação do ID3 é que ele só lida com atributos discretos, não sendo possível apresentar a ele um conjunto de dados com atributos contínuos. Neste caso, os atributos devem ser discretizados.

- O ID3 também não apresenta nenhuma forma de tratar valores desconhecidos, ou seja, todos os exemplos do conjunto de treinamento deve ter valores conhecidos para todos os seus atributos

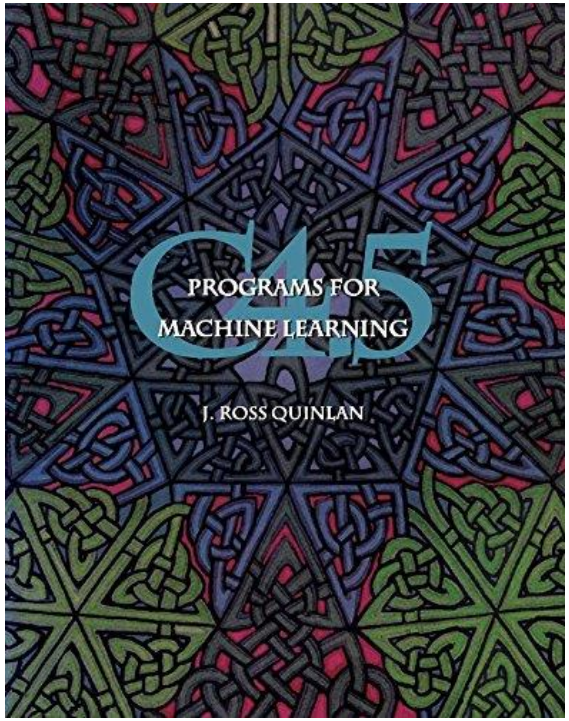
Algoritmo ID3

Limitações do Algoritmo ID3

- O algoritmo também não lida com nenhum mecanismo pós-poda, o que poderia amenizar em árvores mais complexas

Algoritmo C4.5

C4.5 é uma extensão do algoritmo ID3, ambos desenvolvido por Ross Quinlan.
Ele apresenta uma série de melhorias do algoritmo ID3.



Algoritmo C4.5

Algumas destas melhorias do algoritmo C4.5 em relação ao ID3 são:

- Lidar com atributos contínuos e discretos. No intuito de lidar com atributos contínuos, o algoritmo C4.5 cria um valor limiar e então particiona o conjunto de dados em dois subconjuntos dos quais um contém valores de atributos maiores do que aquele valor limiar e o outro conjunto contém valores menores ou iguais aquele valor limiar.

- Lidar com dados de treinamento com atributos incompletos. O algoritmo C4.5 permite que atributos sejam rotulados como *?* para casos onde os valores não estejam presentes. Valores de atributos que não estejam presentes não são utilizados em cálculos de entropia ou ganho de informação.

Algoritmo C4.5

Algumas destas melhorias do algoritmo C4.5 em relação ao ID3 são:

•Poda de árvores após a criação. O algoritmo C4.5 retrocede pela árvore quando esta é criada e tenta remover ramificações que não ajudam no processo de decisão e substitui estes ramos por nós folha.

•Implementa o 'gain ratio' ao invés do 'ganho de informação' tradicional.

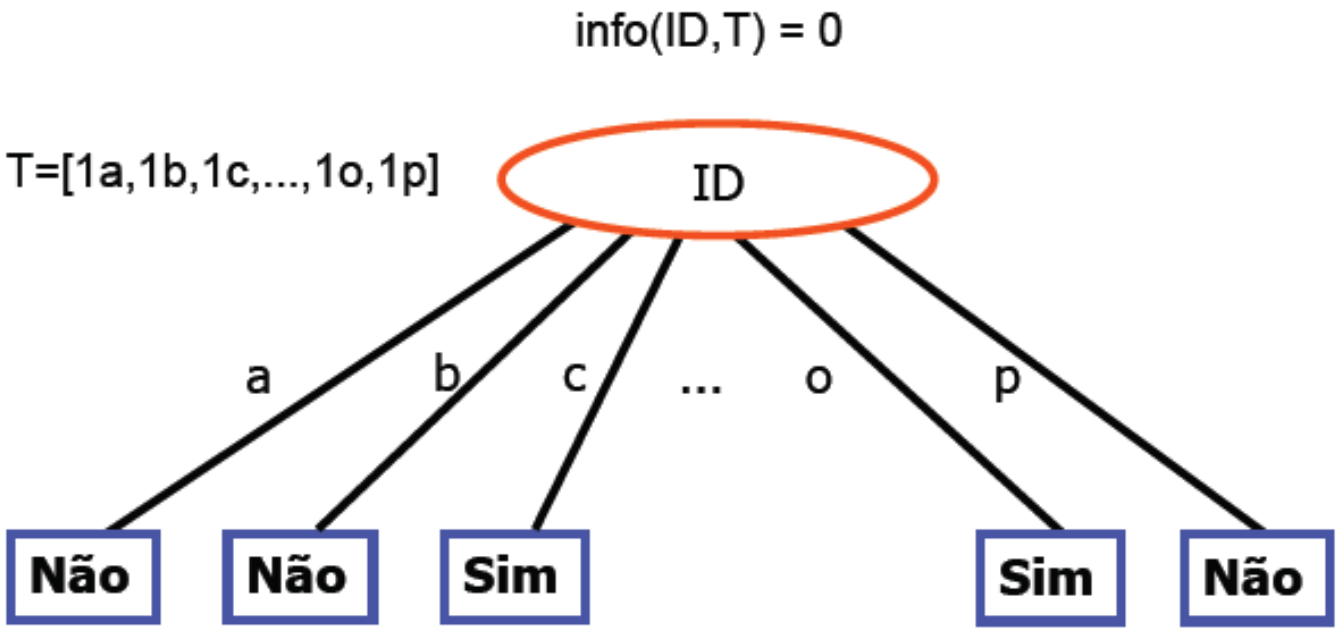
A **Razão de Ganho** foi introduzida no algoritmo **C4.5** para corrigir o viés do Ganho de Informação. Ela normaliza o Ganho de Informação ao considerar o número de valores distintos de um atributo.

Razão de ganho (gain ratio)

Uma vez que cada código ID é único, particionando o conjunto de treinamento nos valores deste atributo levará a um grande número de subconjuntos, cada um contendo somente um caso

Como todos os subconjuntos (de 1 elemento) necessariamente contêm exemplos de uma mesma classe, **info(ID,T)=0**, assim o ganho de informação deste atributo será máximo

Razão de ganho (gain ratio)



Razão de ganho (gain ratio)

Para solucionar esta situação, em analogia à definição de info(T), vamos definir a informação potencial gerada pela partição de T em r subconjuntos

$$\text{split-info}(X,T) = -\sum_{i=1}^{r} \frac{|T_i|}{|T|} \log_2 \frac{|T_i|}{|T|}$$

A razão de ganho é definida como:

$$\text{gain-ratio}(X,T) = \frac{\text{gain}(X,T)}{\text{split-info}(X,T)}$$

A **razão de ganho** expressa a proporção de informação gerada pela partição que é útil, ou seja, que aparenta ser útil para a classificação

Razão de ganho (gain ratio)

Usando o exemplo anterior para o atributo Clientes que produz três subconjuntos com 2, 4 e 6 exemplos, respectivamente

$$ganho(clientes) = 1 - \left[\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I\left(\frac{2}{6}, \frac{4}{6}\right) \right] \approx 0{,}541\, bits$$

$$split - \inf o(Clientes, T) = -\frac{2}{12}\log_2 \frac{2}{12} - \frac{4}{12}\log_2 \frac{4}{12} - \frac{6}{12}\log_2 \frac{6}{12}$$

$$split - \inf o(Clientes, T) = -\frac{2}{12}(-2.58496) - \frac{4}{12}(-1.58496) - \frac{6}{12}(-1) = 1.459$$

Razão de ganho (gain ratio)

Para este teste, cujo ganho é gain(Clientes,T) =0.541 (mesmo valor anterior), a razão de ganho é:

gain-ratio(Clientes,T) = $\frac{0,541}{1,459} = 0,37$

Assim, a **razão de ganho** expressa a proporção de informação gerada pela partição que é útil, ou seja, que aparenta ser útil para a classificação

Como o C4.5 lida com valores numéricos?

Um teste em um atributo numérico produz uma partição binária do conjunto de exemplos:

- Exemplos onde valor_do_atributo <= ponto_referência
- Exemplos onde valor_do_atributo > ponto_referência

Como o C4.5 lida com valores numéricos?

Escolha do ponto de referência:

- Ordenar os exemplos por ordem crescente dos valores do atributo numérico
- Qualquer ponto intermediário entre dois valores diferentes e consecutivos dos valores observados no conjunto de treino pode ser utilizado como possível ponto de referência.
  - ✓ É usual considerar o valor médio entre dois valores diferentes e consecutivos.
  - ✓ Soma-se o valor menor e o valor maior e divide-se pelo número de classes
- Fayyard e Irani (1993) mostram que de todos os possíveis pontos de referência aqueles que maximizam o ganho de informação separam dois exemplos de classes diferentes.

Como lidar com atributos numéricos em algoritmos de árvore de decisão?

Veja o artigo que está no CANVAS:

A_comparative_study_of_decision_tree_ID3_and_C4.5.pdf

Como utilizar o algoritmo de árvore de decisão em Python?

Rode os códigos sobre árvore que estão no CANVAS e façam as listas de exercícios

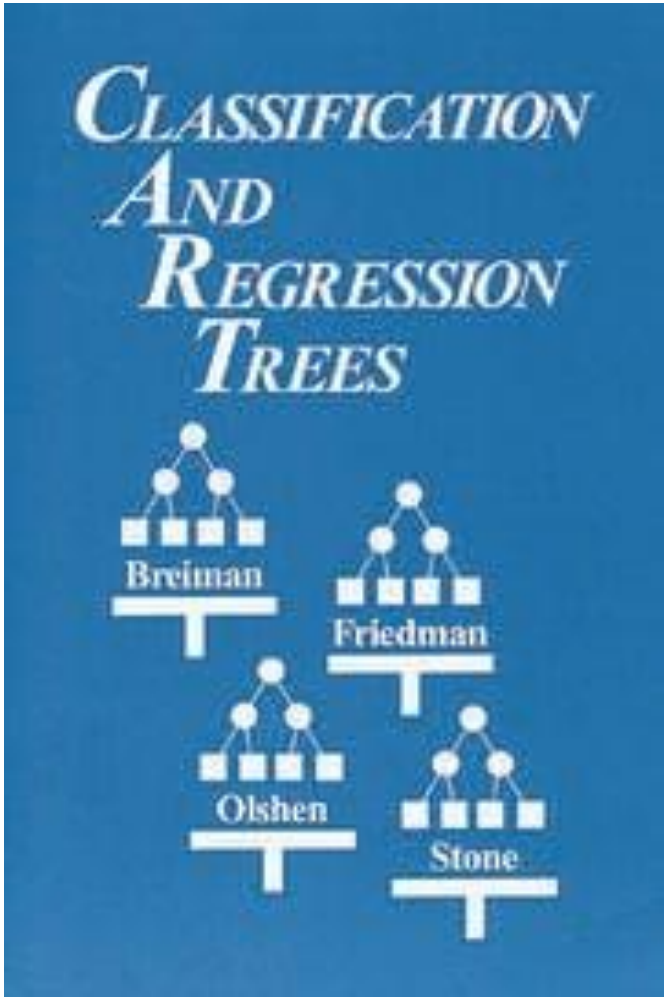
Algoritmo de árvore de decisão CART

O algoritmo de árvore de decisão **CART** (Classification and Regression Trees) foi desenvolvido por **Leo Breiman**, juntamente com Jerome Friedman, Richard Olshen, e Charles Stone. Eles introduziram o algoritmo no livro clássico "Classification and Regression Trees" publicado em 1984.

O algoritmo CART é amplamente utilizado tanto para tarefas de classificação quanto para regressão. Uma de suas principais características é que ele constrói árvores binárias (onde cada nó interno tem exatamente dois filhos) e utiliza critérios como o **índice de Gini** (para classificação) ou o erro quadrático médio (para regressão) para determinar as divisões nos nós.

Algoritmo de árvore de decisão CART

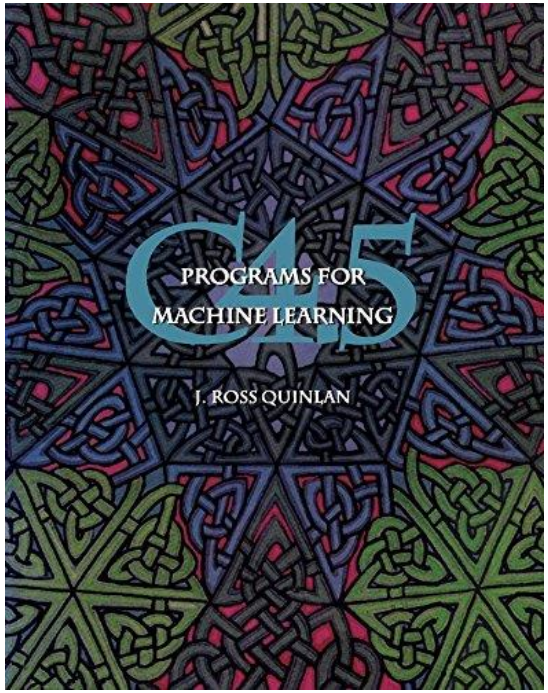
1928-2005



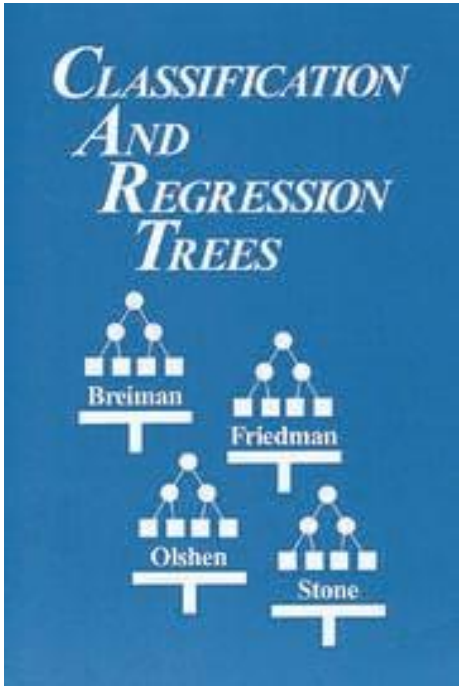
Algoritmo de árvore de decisão CART

Investigue o funcionamento do índice GINI

Referências



https://www.sciencedirect.com/book/9780080500584/c4-5#book-description



https://www.taylorfrancis.com/books/mono/10.1201/9781315139470/classification-regression-trees-leo-breiman-jerome-friedman-olshen-charles-stone