Capítulo 2- Árvores de Decisão

Baseado essencialmente nos Caps. 1, 2 e 3 de "Machine Learning", Tom Mitchell,

Página web do curso de Tom Mitchel, Carnegie Mellon University,

http://www.cs.cmu.edu/~tom/mlbook.html 8 setembro 2023

Complementa o conteúdo dos slides da aula teórica. Agradece-se qualquer comentário ou sugestão para a sua melhoria.

Coimbra, 12 de setembro de 2023.

Aprendizagem em árvores de decisão (mais em Cap. 3 Mitchell**)**

1 Introdução

A aprendizagem em árvores de decisão é um método de aproximação de funções-alvo discretas (*discrete-value target functions*), representadas numa árvore de decisão. As árvores aprendidas também podem ser representadas como conjuntos de regras *if-then* para aumentar a interpretabilidade humana (*human readability*). Estes métodos de aprendizagem contam-se entre os algoritmos mais populares de inferência indutiva e foram já aplicados com sucesso a um largo espectro de tarefas desde o diagnóstico médico à gestão do risco de créditos de clientes bancários.

2 Representação de conceitos por árvores de decisão

As árvores de decisão classificam instâncias ordenando-as descendo a árvore desde a raiz até algum nó folha, que fornece a classificação da instância. Cada nó na árvore executa um teste sobre algum atributo da instância, e cada aresta (*branch,* ramo) descendente daquele nó corresponde a um dos possíveis valores daquele atributo. O processo de classificação de uma instância inicia-se na raiz da árvore, testando o atributo especificado por este nó, descendo pelo ramo da árvore correspondente ao valor do atributo no caso em exemplo, e indo para o nó extremo desse ramo, onde se faz o teste do seu atributo, e se continua, até se atingir uma folha da árvore.

Consideremos o exemplo (de Mitchel) relativo à classificação das manhãs de sábado relativamente à sua adequação para jogar ténis, usando a classificação meteorológica segundo três atributos (céu, humidade, vento):

Outlook (estado do céu): Sunny (Soalheiro), Overcast (Nublado), Rain (Chuva)

Himidity (Humidade): High (Alta), Normal

Wind (Vento): Strong (Forte), Weak (Fraco)

Sunny (Soalheiro): com muita humidade, não adequado

ou com humidade normal, adequado

Overcast (Nublado): é adequado (independentemente do resto).

Rain (Chuva): com vento fraco, adequado

com vento forte, não adequado.

Especificando todos estes atributos e o valor final, temos definido o conceito jogar ténis.

Podemos representar este conceito pela árvore da Figura 1.

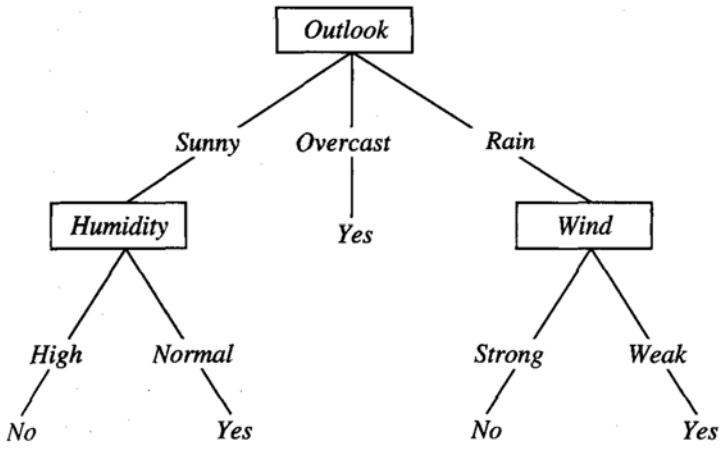


Figura 1. Uma árvore de decisão para o conceito PlayTennis (de Mitchell, pag. 53)).

Por exemplo para a instância

<Outlook=Sunny, Temperature=Hot, Humidity=High, Wind=Strong>

percorre-se pelo ramo mais à esquerda e será classificada como uma instância negativa (ou seja, a árvore prediz que PlayTenis=No). Note-se que a instância tem o atributo Temperature que não existe na árvore, mas isso não constitui problema, dado que com os outros atributos se chega a uma decisão. Esta situação é corrente em árvores de decisão: pode ser possível classificar uma instância apenas com uma parte dos seus atributos.

Geralmente as árvores de decisão representam uma disjunção de conjunções de restrições sobre os valores dos atributos das instâncias: cada caminho desde a raiz a uma folha representa uma conjunção de atributos de teste, e a árvore ela mesma é uma disjunção dessas conjunções. A figura 1 corresponde à expressão

(Outlook=Sunny $\wedge$ Humidity=Normal) ($\wedge$ AND, $\vee$ OR)

$\vee$ (Outlook=Overcast)

$\vee$ (Outlook=Rain $\wedge$ Wind=Weak)

Que é a disjunção das conjunções que levam à classificação yes. Esta expressão define o conceito PlayTenis.

3 Problemas apropriados para resolução por árvores de decisão

Foram (e continuam a ser) desenvolvidos muitos métodos de aprendizagem de árvores de decisão, com diferentes capacidades e características. No entanto consideram-se genericamente mais apropriadas para problemas com as características seguintes:

- *As instâncias são representadas por pares atributo-valor*. As instâncias são descritas por um conjunto fixo finito de atributos e estes por um conjunto finito de valores (Humidade: alta ou normal, por exemplo). A situação mais simples é quando os atributos podem tomar um pequeno número de valores disjuntos (quente, moderada, fria). No entanto algumas extensões do algoritmo base permitem tratar atributos com valores reais (por exemplo representando a temperatura por valores numéricos).

- *A função alvo (target) tem valores discretos de saída.* Na fig. 1 atribui-se uma classificação booleana (sim, não) a cada exemplo. É fácil estender ao caso de três valores, e mesmo a valores reais (mas neste caso é pouco frequente usar árvores de decisão).

- *Podem ser necessárias descrições disjuntivas*. Como vimos as árvores de decisão representam naturalmente expressões disjuntivas.

- *Os dados de treino podem conter erros.* Os métodos de treino de árvores de decisão são robustos em relação a erros, sejam eles na classificação de exemplos de treino sejam eles nos valores dos atributos que classificam esses exemplos.

- *Os dados de treino podem conter valores de atributos em falta (missing values).* As árvores de decisão podem usar-se quando alguns exemplos de treino têm valores desconhecidos em alguns atributos (por exemplo a humidade do dia só é conhecida para alguns exemplos de treino).

Estas características encontram-se felizmente em muitas aplicações práticas. As árvores de decisão foram por isso aplicadas a problemas como aprender a classificar doentes pelas suas doenças, avarias de equipamentos pelas suas causas, a clientes bancários pelo seu potencial de falharem os pagamentos. Estes problemas, nos quais a tarefa consiste em classificar exemplos numa entre um conjunto discreto de categorias possíveis, chamam-se frequentemente *problemas de classificação*.

4 Algoritmo básico de aprendizagem em árvores de decisão

A maior parte dos algoritmos de aprendizagem em árvores de decisão são variações de um algoritmo nuclear (core) que usa uma técnica de busca de cima para baixo, gulosa (greedy), através do espaço de todas as possíveis árvores de decisão. Esta técnica é ilustrada pelo algoritmo ID3 (Iterative Dichotomiser 3) (Ross Quinlan 1986) e o seu sucessor C4.5 (R. Quinlan 1993). As árvores estão invertidas, sendo a raiz o vértice mais elevado.

O nosso algoritmo básico, ID3, aprende as árvores de decisão construindo-as de cima (raiz) para baixo (folhas), iniciando-se com a pergunta "qual o atributo que deve ser testado na raiz?" Para

responder a esta questão usa-se um teste estatístico para determinar quanto bem um atributo, por si só, classifica os exemplos de treino. Esse teste diz-nos qual o melhor atributo a usar no nó raiz da árvore. De seguida traçam-se ramos descendentes a partir da raiz, cada um para o valor que o atributo da raiz pode assumir, e os exemplos de treino descem pelo ramo apropriado até ao próximo nó. Todo o processo se repete usando o exemplo de treino associado a cada nó descendente para selecionar o melhor atributo para testar nesse ponto da árvore. Este processo constitui uma busca gulosa (greedy), que escolhe o "mais sumarento", na qual o algoritmo nunca volta atrás para reconsiderar escolhas anteriores. O caso mais simples de aprendizagem de funções booleanas (ou seja, aprendizagem de um conceito), pode esquematizar-se como na Tabela. 1.

Tabela1. Algoritmo ID3 Pseudo-código (Mitchell, pág. 56).

Summary of the ID3 algorithm specialized to learning boolean-valued functions. ID3 is a greedy algorithm that grows the tree top-down, at each node selecting the attribute that best classifies the local training examples. This process continues until the tree perfectly classifies the training examples, or until all attributes have been used.

ID3(*Examples*, *Target_attribute*, *Attributes*)
Examples are the training examples. Target_attribute is the attribute whose value is to be predicted by the tree. Attributes is a list of other attributes that may be tested by the learned decision tree. Returns a decision tree that correctly classifies the given Examples.

- Create a *Root* node for the tree
- If all *Examples* are positive, Return the single-node tree *Root*, with label = +
- If all *Examples* are negative, Return the single-node tree *Root*, with label = −
- If *Attributes* is empty, Return the single-node tree *Root*, with label = most common value of *Target_attribute* in *Examples*
- Otherwise Begin
  - $A \leftarrow$ the attribute from *Attributes* that best* classifies *Examples*
  - The decision attribute for *Root* $\leftarrow A$
  - For each possible value, v_i, of A,
    - Add a new tree branch below *Root*, corresponding to the test $A = v_i$
    - Let $Examples_{v_i}$ be the subset of *Examples* that have value v_i for A
    - If $Examples_{v_i}$ is empty
      - Then below this new branch add a leaf node with label = most common value of *Target_attribute* in *Examples*
      - Else below this new branch add the subtree
        ID3($Examples_{v_i}$, *Target_attribute*, *Attributes* − {A}))
- End
- Return *Root*

4.1 Qual dos atributos é o melhor classificador?

Saber qual o atributo a testar em cada nó da árvore é a escolha central do algoritmo ID3. Naturalmente que queremos escolher o atributo que seja mais útil na classificação dos exemplos.
Qual será a medida quantitativa da capacidade discriminante de um atributo? Ou dito de outro modo, qual a capacidade informativa de um atributo para separar os exemplos de treino de acordo com o alvo (target) da classificação?

Precisamos de introduzir aqui uma métrica para o *ganho de informação* de um atributo, que mede precisamente a sua capacidade discriminante. Trata-se de uma propriedade estatística, que mede o quão bem um dado atributo separa os exemplos de treino de acordo com a classificação alvo. ID3 usa este ganho de informação para selecionar o atributo candidato em cada passo, enquanto a árvore cresce.

4.1.1 A entropia mede a homogeneidade do conjunto dos exemplos

Para se definir precisamente o ganho de informação, usamos aqui uma métrica comum em teoria da informação, chamada *entropia*, que caracteriza a (im)pureza de uma dada coleção de exemplos. Dada uma coleção S, contendo exemplos positivos e negativos de um conceito alvo, a entropia S relativa a esta classificação booleana é definida por (1)

$$\begin{aligned} &Entropy(S) \equiv -p_{\oplus}\log_2 p_{\oplus} - p_{\odot}\log_2 p_{\odot} \\ &\text{em que} \\ &p_{\oplus} \text{ é a proporção de exemplos positivos em } S \\ &p_{\odot} \text{ é a proporção de exemplos negativos em } S \\ &\text{Considera-se em todos os cálculos que } 0\log 0 = 0 \end{aligned} \tag{1}$$

Suponhamos que temos uma coleção *S* de 14 exemplos de um conceito booleano, sendo 9 positivos e 5 negativos (usaremos a notação 9+ e 5- para sumariar um tal conjunto de dados).
A entropia de *S* relativa a esta classificação booleana será

$$Entropy[9+,5-] = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0,940 \tag{2}$$

Se tivéssemos todos os exemplos numa classe, por exemplo a positiva, seria

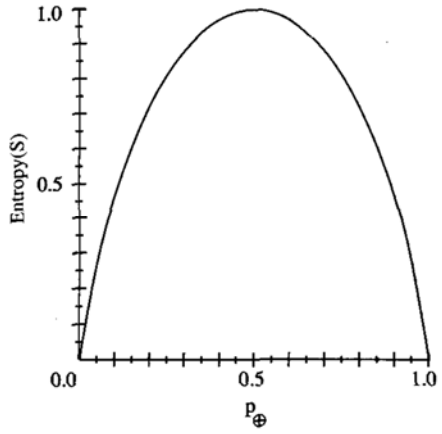
$$Entropy[14+,0-] = -(14/14)\log_2(14/14) - (0/14)\log_2(0/14) = -1\times 0 - 0\log_2 0 = 0 \tag{3}$$

E o mesmo se obteria no caso de 14-. Este resultado compreende-se: se todos os exemplos pertencem à mesma classe então não temos qualquer informação para classificar. Note-se que uma coleção *S* de exemplos representa um conceito para o qual existem outros exemplos não pertencentes a *S*.

Se *S* contém o mesmo número de elementos positivos e negativos, teremos

$$\begin{aligned} Entropy[7+,7-] &= -(7/14)\log_2(7/14) - (7/14)\log_2(7/14) = \\ &= -(1/2)\log_2(1/2) - (1/2)\log_2(1/2) == -\log_2(1/2) = -(-1) = 1 \end{aligned} \tag{4}$$

Este caso é extremo: existe a quantidade máxima de informação, dado que existem tantos exemplos positivos como negativos.
Se o número de exemplos positivos e negativos são diferentes, como é o caso mais usual, a entropia de *S* está entre 0 e 1. A Fig. 2 ilustra a variação da entropia em função da proporção dos exemplos positivos que varia entre 0 e 1.



(Mitchell, pag 57)

FIGURE 3.2
The entropy function relative to a boolean classification, as the proportion, $p_{\oplus}$, of positive examples varies between 0 and 1.

Se a classificação não for booleana, mas tiver mais de duas classes, sejam c classes, a entropia é definida por (5),

$$Entropy(\mathrm{S}) = \sum_{i=1}^{c} -p_i \log_2 p_i \qquad (5)$$

em que p_i é a proporção de S que pertence à classe i

Se o conceito alvo pode ter c valores possíveis, admitindo que a coleção S tem a mesma proporção 1/c de cada valor, a entropia será

$$\begin{aligned} Entropy(S) &= -(1/c)\log_2(1/c) - (1/c)\log_2(1/c) .... - (1/c)\log_2(1/c) = \text{ (c parcelas)} \\ &= (1/c)\log_2 c + (1/c)\log_2 c + ... + (1/c)\log_2 c = \\ &= (1/c + 1/c + ... + 1/c)\log_2 c = 1\log_2 c = \log_2 c \end{aligned} \qquad (6)$$

Assim se tivermos 4 valores possíveis para o alvo, a máxima entropia de qualquer conjunto de exemplos é $\log_2(4)=2$.

4.1.2 O ganho de informação mede a expectativa de redução da entropia

A entropia mede o grau de "impureza" (grande entropia grande impureza) ou de homogeneidade (pequena entropia grande homogeneidade) de uma coleção de exemplos de treino. Ao descermos a árvore, da raiz para as folhas, interessa-nos que o grau de entropia vá diminuindo, de modo que a homogeneidade vá aumentando até se chegar a uma classe bem definida. Este objetivo permite-nos definir uma medida de quanto apropriado é um atributo para classificar os dados de treino. Essa medida é o **ganho de informação**, que é precisamente a redução expectável da entropia provocada pela partição dos exemplos segundo este atributo. Mais precisamente, o ganho de informação *Gain (S,A)* de um atributo A, em relação a uma coleção de exemplos S, define-se por (7)

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \qquad (7)$$

Em que *Values(A)* é o conjunto de todos os valores possíveis do atributo A, e S_v é o subconjunto de S para o qual o atributo A tem o valor de v, ou seja

$$S_v = \{ s \in S \,|\, A(s) = v \}.$$

A primeira parcela de (7) é a entropia inicial da coleção de exemplos *S*; a segunda parcela é o valor esperado da entropia depois de particionar S usando o atributo *A*. A entropia esperada, na segunda parcela, é a soma das entropias de cada subconjunto S_v (note-se que *v* pode tomar diversos valores), ponderadas pela fração dos exemplos $\frac{|S_v|}{|S|}$ que pertencem a S_v. Note-se que no cálculo de *Entropy*(S_v) usa-se a fração de exemplos que assumem o valor de *v* do atributo *A*.
A entropia vai diminuindo descendo de nó para nó, e essa diferença é quantificada por *Gain (S,A),* a diminuição da entropia produzida pelo conhecimento do atributo *A* (quanto mais se conhece, menor a entropia). Podemos dizer também que *Gain(S,A)* é a informação fornecida acerca do valor da função alvo, dado o valor de algum atributo *A*.

Vejamos um exemplo (Tabela 2)
Seja S um conjunto de exemplos de treino do conceito *PlayTenis* descrito pelo atributo *Wind* que pode ter os valores *Weak* e *Strong*. Admitamos que S contém 14 exemplos, [9+,5-]. Desses 14 exemplos, 6 dos positivos e 2 dos negativos têm *Wind= Weak* e os restantes (3 positivos e 3 negativos) têm *Wind=Strong*.
O ganho de informação alcançado pelo ordenamento dos 14 exemplos originais pelo atributo *Wind* calcula-se assim (Mitchell, pág. 58):

$$\begin{aligned}
Values(Wind) &= Weak, Strong \\
S &= [9+, 5-] \\
S_{Weak} &\leftarrow [6+, 2-] \\
S_{Strong} &\leftarrow [3+, 3-] \\
Gain(S, Wind) &= Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v) \\
&= Entropy(S) - (8/14) Entropy(S_{Weak}) \\
&\quad - (6/14) Entropy(S_{Strong}) \\
&= 0.940 - (8/14)0.811 - (6/14)1.00 \\
&= 0.048
\end{aligned}$$

Note-se que *Entropy(S)*=0,94 já foi calculada anteriormente em (2). Por outro lado temos 8 instâncias com *Wind=Weak* e 6 com *Wind=Strong*.

$$Entropy(S_{Strong}) = -(3/6)\log_2(3/6) - (3/6)\log_2(3/6) = (1/2)\log_2 2 + (1/2)\log_2 2 = 1/2 + 1/2 = 1,00$$
$$Entropy(S_{Weak}) = -(6/8)\log_2(6/8) - (2/8)\log_2(2/8) = 0,811$$

O algoritmo ID3 usa o ganho da informação para selecionar o melhor atributo em cada nó.

Consideremos agora também o atributo *Humidity* para o conceito *PlayTenis,* com os valores *High* e *Normal*. É *High* em 3 positivos e 4 negativos e é *Normal* em 6 positivos 1 negativo.

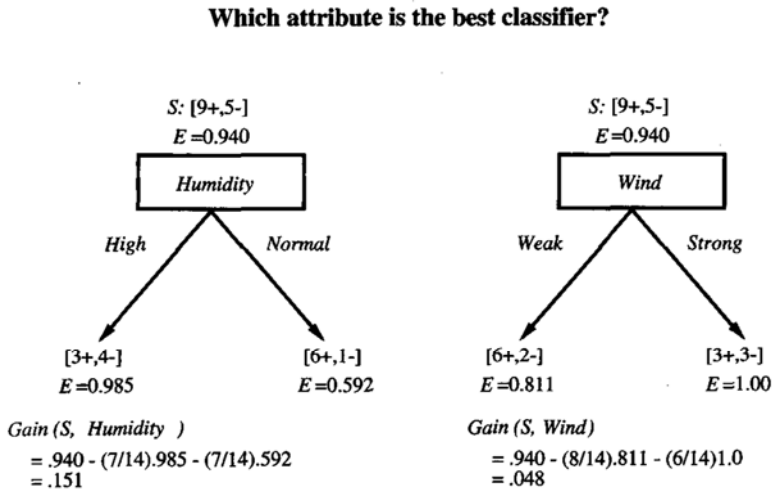


FIGURE 3.3
Humidity provides greater information gain than *Wind*, relative to the target classification. Here, E stands for entropy and S for the original collection of examples. Given an initial collection S of 9 positive and 5 negative examples, [9+, 5−], sorting these by their *Humidity* produces collections of [3+, 4−] ($Humidity = High$) and [6+, 1−] ($Humidity = Normal$). The information gained by this partitioning is .151, compared to a gain of only .048 for the attribute *Wind*.

Fazendo os cálculos, como na Fig. 3 (Mitchell pág. 59), conclui-se que o *Humidity* dá um maior ganho de informação do que o *Wind*.

5. Um exemplo ilustrativo

Considere-se o treino do conceito *PlayTenis* através de um conjunto de 14 exemplos da tabela 2.

Tabela 2. Exemplos de treino do conceito *PlayTenis* (Mitchell, pág. 59).

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Cada instância tem 4 atributos, dois deles (Outlook e Temperature) com três valores possíveis, e os outros dois (Humidity e Wind) com dois valores possíveis.
Nestas condições o número total de instâncias possíveis em que os 4 atributos assumem um valor concreto são 3x3x2x2=36.
Note-se, no entanto, que se podem definir instâncias semanticamente válidas, isto é, a que nós damos um significado, mas nas quais alguns atributos podem não assumir um valor específico. Por exemplo,

D15 = *(*, Mild, Normal, Weak, Yes*).

Qualquer que seja o céu, se a temperatura for *mild* (suave), a humidade normal, o vento fraco, o dia é bom para jogar ténis. O asterisco quer dizer isso mesmo: qualquer que seja. Portanto o atributo *Outlook* pode assumir 4 valores , *Outlook*={*Sunny, Rain, Overcast*, *). E analogamente para os outros atributos.
Temos ainda a situação em que os atributos são representados por um conjunto vazio,

D16=*(∅,∅,∅,∅, Yes*)

Quer dizer que nenhum dia é apropriado para jogar ténis.

Assim o número total de instâncias com significado semântico, neste exemplo, é:

4x4x3x3 + 1=145 (o 1 corresponde a D16).

(Noutros casos faz-se um cálculo semelhante).

E vamos treinar a árvore com 14 exemplos. Por isso é uma aprendizagem indutiva.

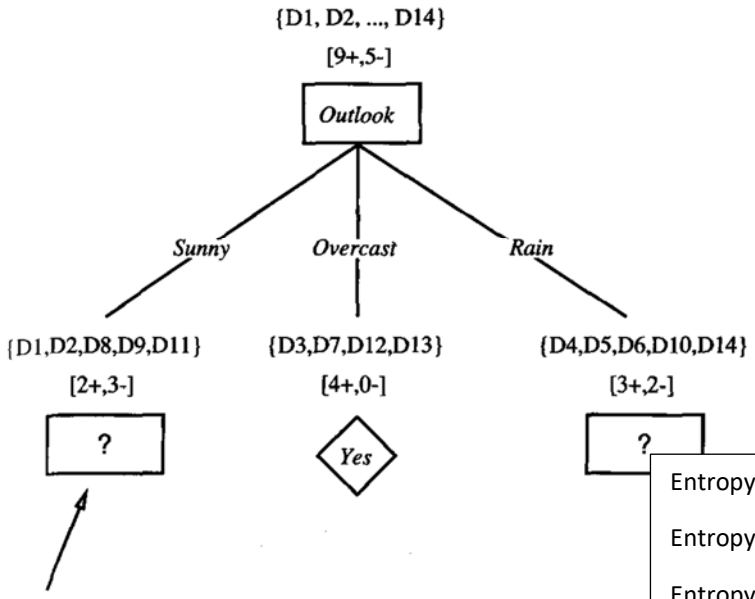
Primeiro passo: criar a raiz da árvore.

Para isso é necessário calcular o ganho de informação para os 4 atributos e escolher o que der maior valor. Já vimos na Fig. 3 como se calcula para a *Temperature* e *Wind*. Para o *Outlook* e *Humidity* procede-se de modo análogo, contando na Tabela 2 os diversos valores para + e -. Obtém-se (confira os cálculos em falta):

$$Gain(S, Outlook) = 0.246$$
$$Gain(S, Humidity) = 0.151$$
$$Gain(S, Wind) = 0.048$$
$$Gain(S, Temperature) = 0.029$$

(pág. 60)

Assim escolhe-se o *Outlook* como o atributo do nó raiz da árvore, aquele que dá a melhor predição do atributo alvo. Desenham-se os ramos correspondentes aos valores do *Outlook* e assim temos a primeira etapa concluída e a (sub)árvore como o aspeto da Fig. 4 (Mitchell, pág. 61).



Entropy([2+,3-)]=-2/5log$_2$(2/5)-3/5log$_2$(3/5)=0,970

Entropy([4+,0-)]=-4/4log$_2$(4/4)-0/4log$_2$(0/4)=0,000

Entropy([3+,2-)]=-3/5log$_2$(3/5)-2/5log$_2$(1/5)=0,970

S_{sunny} = {D1,D2,D8,D9,D11}

$Gain(S_{sunny}, Humidity) = .970 - (3/5)\,0.0 - (2/5)\,0.0 = .970$

$Gain(S_{sunny}, Temperature) = .970 - (2/5)\,0.0 - (2/5)\,1.0 - (1/5)\,0.0 = .570$

$Gain(S_{sunny}, Wind) = .970 - (2/5)\,1.0 - (3/5)\,.918 = .019$

Humidity High =[0+,3-], Normal=[2+,0-]
Temp. Hot =[[0+,2-], Mild=[1+,1-], Cold=[1+,0-]
Wind Weak=[1+,2-] Strong=[1+,1-]

FIGURE 3.4
The partially learned decision tree resulting from the first step of ID3. The training examples are sorted to the corresponding descendant nodes. The *Overcast* descendant has only positive examples and therefore becomes a leaf node with classification *Yes*. The other two nodes will be further expanded, by selecting the attribute with highest information gain relative to the new subsets of examples.

Note-se, na árvore parcial da Fig. 4, que todos os exemplos em que *Outlook=Overcast* são positivos em *PlayTenis* e por isso a entropia do nó correspondente é zero. Este nó é por isso uma folha da árvore com a classificação *PlayTenis=Yes* e não terá descendentes.
No caso dos dois outros nós já assim não acontece e é preciso continuar.

Segundo passo: selecionar um atributo novo (que não exista no caminho para cima) e particionar os exemplos de treino para cada nó não-terminal, mas agora usando apenas os exemplos de treino associados a esse nó. Está calculado na Fig. 4 para o nó *Sunny*.

A árvore final está na Fig. 1 que aqui se repete. Convida-se o leitor a completar os cálculos.

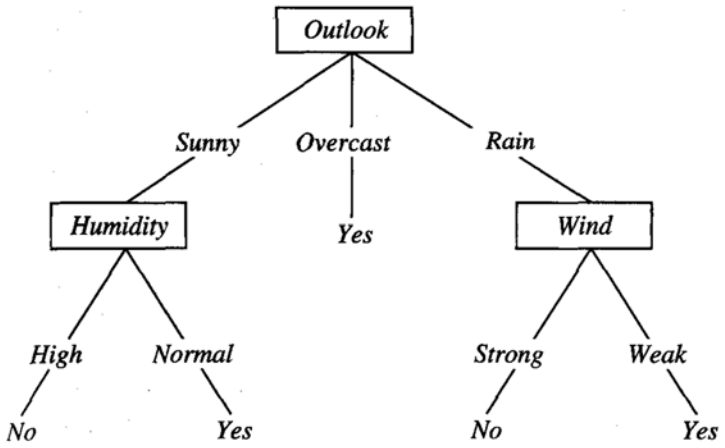


Fig 1 (repetida) Árvore de decisão final para o exemplo.

Podemos usá-la para instâncias não contidas no conjunto de treino. Por exemplo

(*Outlook, Temperature, Humidity, Wind) PlayTenis*

(*Overcast, Cool, High, Weak*) dá *Yes*
(*Sunny, Cool, High, Strong*) dá *No*

Note-se que em cada caminho da raiz até às folhas não é obrigatório que existam todos os atributos. Se falta algum, quer dizer que de acordo com a aprendizagem indutiva a partir do conjunto de exemplos, esse atributo é irrelevante para a classificação por esse caminho.

5. Procura no espaço das hipóteses na aprendizagem de árvores de decisão.

O algoritmo ID3 procura uma árvore no espaço das hipóteses (de árvores) que se ajuste aos exemplos de treino. O espaço de hipóteses é o espaço das árvores de decisão possíveis. O ID3 faz uma procura do simples para o complexo, de subida da montanha (*hill-climbing*) através do espaço das hipóteses, inicializando-se com a árvore vazia, e considerando sucessivamente hipóteses mais elaboradas na procura de uma árvore de decisão que classifique corretamente todos os dados de treino. O critério de avaliação que orienta essa busca é a medida do ganho de informação. A Fig. 5 (Mitchell, pág. 62) ilustra, esquematicamente, essa busca.

Esta terminologia vem da teoria da otimização.

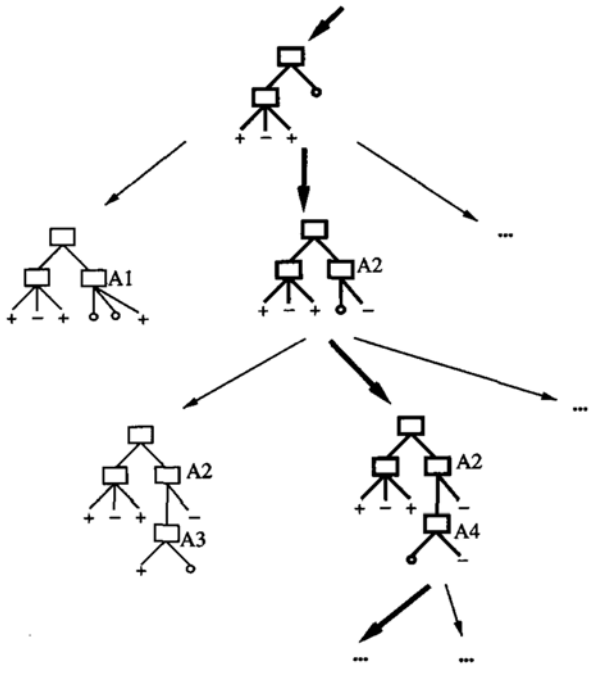


FIGURE 3.5
Hypothesis space search by ID3. ID3 searches through the space of possible decision trees from simplest to increasingly complex, guided by the *information gain heuristic.*

Olhando para o ID3 em termos do seu espaço de procura e da estratégia de procura, pode-se induzir algumas das suas capacidades e limitações:

- O espaço de hipóteses de todas as árvores de decisão do iD3 é um espaço completo de funções de valores discretos, relativos aos atributos disponíveis. Por isso está garantido que a solução está no espaço das hipóteses.

- À medida que procura, o ID3 mantém uma única hipótese corrente, contrariamente a outros algoritmos que mantêm o conjunto de todas as hipóteses consistentes com os exemplos de treino disponíveis. Este facto pode ser considerado uma limitação do ID3, dado não ser capaz de indicar quantas árvores alternativas são compatíveis com os dados disponíveis, ou elaborar novas instâncias (*queries*) que escolham de forma ótima entre essas hipóteses competitivas.

- ID3 não volta para trás (backtracking) na sua busca. Por esse facto pode cair num ótimo local que não seja ótimo global. Poderia ter havido uma outra sequência de busca que levasse a uma melhor solução.

- ID3 usa todos os exemplos de treino em cada passo, na sua busca para tomar decisões, estatisticamente fundamentadas, relativas ao modo de refinar as suas hipóteses atuais. Contrasta com outros métodos que tomam decisões incrementais usando exemplos individuais. Neste caso o fundamento estatístico é o ganho de informação. Tem como vantagem que, devido a esse fundamento estatístico, a busca resultante é muito menos sensível a erros nos exemplos individuais de treino.

6 Polarização indutiva na aprendizagem de árvores de decisão

Como vimos anteriormente o ID3 é capaz de generalizar de exemplos de treino para instâncias não vistas. Com faz isso? Qual a polarização indutiva (*inductive bias*) que lhe permite fazer isso?

Chama-se polarização indutiva (*inductive bias*) ao conjunto de suposições que, conjuntamente com o conjunto de treino, justificam dedutivamente a classificação atribuída pelo aprendiz (*learner*) a instâncias futuras.

Dado um conjunto de exemplos de treino, tipicamente existem muitas árvores de decisão consistentes com eles. Descrever a polarização indutiva do ID3 resume-se a descrever o fundamento pelo qual ele escolhe uma dessas hipóteses consistentes, e não outra. Ele escolhe a primeira árvore aceitável, que encontra na sua busca do simples para o complexo, subindo a montanha através do espaço das árvores possíveis. Ou seja, basicamente, a estratégia de busca do ID3 (i) prefere as árvores mais curtas em desfavor das mais longas e (ii) seleciona as árvores que colocam os atributos com maior ganho de informação mais próximos da raiz. Não sendo possível caracterizar precisamente a polarização indutiva do ID3 (devido à subtileza da interação entre a heurística de seleção dos atributos e os exemplos de treino particulares que o ID3 encontra), podemos caracterizar essa polarização ,de forma aproximada, como a **preferência por árvores de decisão pequenas sobre árvores de decisão grandes.**

Polarização indutiva aproximada do ID3: As árvores mais pequenas são preferidas em relação às maiores.

No entanto não está garantido que o ID3 encontra sempre a árvore mais pequena consistente com o conjunto de treino, por favorecer as árvores que colocam os atributos com maior ganho de informação mais perto da raiz.

Uma melhor aproximação para a polarização indutiva do ID3: As árvores mais curtas são preferidas sobre as mais longas. As árvores que colocam os atributos com maior ganho de informação mais próximos da raiz são preferidas sobre as que não o fazem.

Porquê preferir hipóteses mais curtas?

Será que esta polarização indutiva de favorecer as árvores mais curtas é uma abordagem sólida para a capacidade de generalização para além dos exemplos de treino? Esta questão genérica tem sido discutida pelos filósofos e cientistas ao longo de séculos e o debate mantém-se até aos nossos dias sem resolução. William de Occam (ou Ockham) foi um dos primeiros a discutir a questão, por volta de 1320, e por isso esta polarização é frequentemente enunciada como a lâmina de Occam (Occam's razor).

A lâmina de Occam (Occam's razor): Prefira a hipótese mais simples que se ajusta aos dados.

Nota: Guilherme de Ockham foi um lógico e frade franciscano inglês do Séc. XIV.

"O princípio afirma que a explicação para qualquer fenómeno deve assumir apenas as premissas estritamente necessárias à explicação do mesmo e eliminar todas as que não causariam qualquer diferença aparente nas predições da hipótese ou teoria. O princípio é frequentemente designado pela expressão latina **Lex Parsimoniae** (Lei da Parcimónia) enunciada como:"*entia non sunt multiplicanda praeter necessitatem*" (as entidades não devem ser multiplicadas além da necessidade).

O princípio recomenda assim que se escolha a teoria explicativa que implique o menor número de premissas assumidas e o menor número de entidades.

Sendo originalmente um princípio da filosofia reducionista do nominalismo, é hoje tido como uma das máximas heurísticas (regra geral) que aconselham economia, parcimónia e simplicidade, especialmente nas teorias científicas.

“Se em tudo o mais forem idênticas as várias explicações de um fenómeno, a mais simples é a melhor"— **Guilherme de Ockham**

A navalha de Occam é antecessora do chamado princípio KISS, Keep It Simple, Stupid, ou em português “simplifique, estúpido” , uma vulgarização da máxima de Albert Einstein de que "*tudo deve ser feito da forma mais simples possível, mas não mais simples que isso*", também expressa por Antoine de Saint-Exupéry como "a perfeição não é alcançada quando já não há mais nada para adicionar, mas quando já não há mais nada que se possa retirar".

Ver em https://pt.wikipedia.org/wiki/Navalha_de_Occam , um interessante texto que passa também por Leibniz (1646-1716), Kant (1724-1804), Karl Menger (século XX), que discordaram da navalha de Occam.
O termo navalha vem, segundo alguns (Mitchell), do facto de Occam ter formalizado o princípio enquanto se barbeava.

Esta questão de Lâmina de Occam pode dar azo a grandes discussões, mesmo no contexto de aprendizagem em árvores de decisão (ver Mitchell, pp 65/66). Todos os links anteriores estavam válidos em 12 de setembro de 2023.

7 Desafios na aprendizagem de árvores de decisão

Os desafios práticos na aprendizagem de árvores de decisão são vários:

- Quão fundo deve crescer a árvore de decisão?
- Como tratar atributos contínuos?
- Como selecionar uma métrica adequada para a seleção dos atributos?
- Como tratar com dados de treino com valores de atributos em falta?
- Como tratar atributos com diferentes custos?
- Como aumentar o desempenho computacional?

Existem extensões do ID3 que respondem a esses desafios, resultando no algoritmo C4.5. Ver Mitchell para mais informação.

7.1. Evitar sobre ajuste (overfiting) dos dados

O algoritmo ID3 faz crescer cada ramo da árvore apenas o necessário para classificar perfeitamente os exemplos de treino. Esta estratégia é razoável, mas pode criar dificuldades quando existe ruído nos dados ou quando o número de exemplos de treino é demasiado pequeno para produzir uma amostra representativa da função alvo. Nestes casos o algoritmo pode produzir árvores que sobre ajustam (overfit) aos exemplos de treino.

Uma hipótese sobre ajusta os dados se houver uma outra hipótese que não se ajusta tão bem aos dados de treino mas, em contrapartida, tem um melhor desempenho considerando toda a distribuição de instâncias (incluindo instâncias para além do conjunto de treino). Formalmente

Definição: Dado um espaço de hipóteses H, uma hipótese $h \in H$ diz-se que sobre ajusta (overfits) os dados de treino se existir alguma hipótese alternativa $h' \in H$, tal que h tem um menor erro do que h' ao longo dos exemplos de treino, mas h' tem menor erro do que h ao longo da distribuição completa das instâncias.

A Fig. 6 (Mitchell, pág. 67)) ilustra o impacto do sobre ajuste numa aplicação típica de aprendizagem de árvores de decisão.

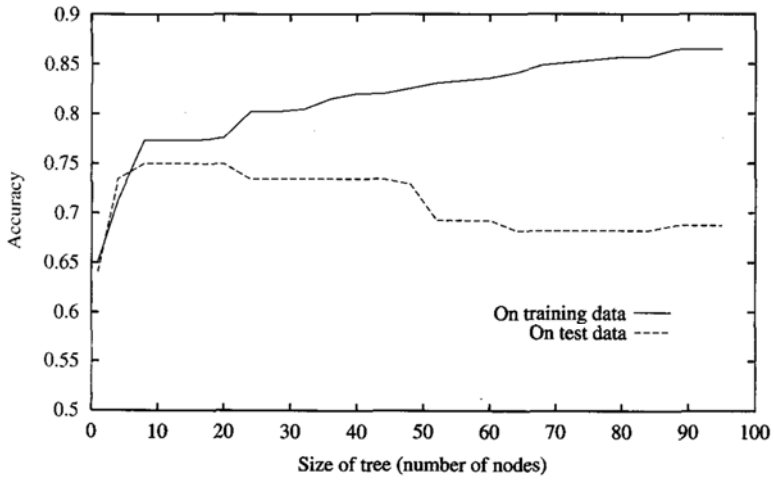


FIGURE 3.6
Overfitting in decision tree learning. As ID3 adds new nodes to grow the decision tree, the accuracy of the tree measured over the training examples increases monotonically. However, when measured over a set of test examples independent of the training examples, accuracy first increases, then decreases. Software and data for experimenting with variations on this plot are available on the World Wide Web at http://www.cs.cmu.edu/~tom/mlbook.html.

No caso trata-se de aprender quais os doentes que têm uma forma de diabetes. O eixo horizontal da figura indica o número de nós da árvore à medida que vai sendo construída. O eixo vertical indica a precisão das predições feitas pela árvore (no estado atual de construção) para as instâncias de dois conjuntos:
- o conjunto de treino (traço contínuo),
- o conjunto de teste, independente do de treino (a tracejado).
Note-se que a precisão no conjunto de treino aumenta monotonicamente até ao fim, enquanto que a precisão no conjunto de teste começa por aumentar, até cerca de 25 nós, e diminui a partir daí.

O sobre ajuste é um problema prático comum a todos os métodos de aprendizagem computacional e que por isso reencontraremos em capítulos futuros. No caso das árvores de decisão, existem diversas abordagens para gerir este problema:
- Terminar o crescimento da árvore antes que ela atinja o ponto em que classifica perfeitamente todos os dados de treino (admitindo assim que se engane nalguns)

- Construir a árvore completa com sobre ajuste, e depois podar a árvore eliminando nós. Esta técnica tem sido mais bem sucedida na prática, dado que na primeira é difícil determinar o passo em que se deve parar.
Em qualquer caso (seja por paragem prematura, seja por poda pós-aprendizagem) uma questão essencial é saber qual o tamanho ótimo da árvore final. Podem-se usar diversas abordagens (ver Mitchell, p. 69) mas a mais usada na prática (aqui e na generalidade das técnicas de aprendizagem computacional) é:

Evitar sobre ajuste: usar um conjunto de dados para treino e um conjunto de dados para validação, sendo os dois independentes um do outro.

Os dados para treino servem para construir a hipótese aprendida; os dados de validação servem para avaliar a precisão da hipótese aprendida noutros dados novos.

Claro que é necessário que estejam disponíveis muitos exemplos para que se possam dividir nesses dois conjuntos. Uma regra heurística é reservar cerca de dois terços para treino e cerca de um terço para validação, também conhecida com a regra 70-30.

Sobre a técnica de poda de nós ver Mitchell 69/70.

8. Melhorias do ID3 em direção ao C4.5

Foram introduzidas, ao longo do tempo, modificações ou extensões do ID3 nomeadamente:
- A incorporação de atributos com valores numéricos contínuos.
- Outras medidas para a seleção de atributos para cada nó (que não o ganho de informação de (7)).
- Manuseio de exemplos de treino com valores de atributos em falta.
- Manuseio de atributos com diferentes custos.

Essas contribuições visam suprir deficiências do ID3 e levaram ao algoritmo C4.5. Podem ser estudadas em Mitchell 72/76.

As árvores de decisão estão incluídas nos modelos CART (*Classification And Regression Trees*), com lhes chama Murphy.

Bibliografia:

Mitchell, T., Machine Learning, McGraw-Hill Science/Engineering/Math; (March 1, 1997).

Marsland, Stephen, Machine Learning, An Algorithm Perspective, CRC Press 2008.

Murphy, Kevin P., Machine Learning, A Probabilistic Perspective, MIT Press 2012.

Podgorelec V., P. Kokol, B. Stiglic, I. Rozman, Decision trees: an overview and their use in medicine, Journal of Medical Systems, Kluwer Academic/Plenum Press, Vol. 26, Num. 5, pp. 445-463, October 2002 . https://pubmed.ncbi.nlm.nih.gov/12182209/ 12 setembro 2023