APRENDER A PARTIR DE EXEMPLOS

Capítulo 18, secção 3

Resumo

- Agentes aprendizes
- Problemas de aprendizagem
- Abordagens
- Aprendizagem Indutiva
- Aprendizagem de Conceitos
- Aprendizagem baseada em Exemplos

Árvores de Decisão

- Árvores de Decisão:
  - Representam uma função que toma como input um vector de atributos e retorna uma "decisão" – um valor de output único.
    - Iremos ver o caso de input com valores discretos e output booleano
  - Atingem a decisão efetuando uma sequência de testes.
    - Cada nó interno da árvore corresponde a um teste ao valor de um atributo, A_i, e os ramos desse nó são rotulados com os possíveis valores desse atributo $A_i=v_i$.
    - Cada nó terminal (folha) especifica um valor a ser devolvido pela função.
  - A representação de funções através de árvores de decisão é compreensível para humanos
    - Há manuais inteiros (e.g. Reparação de automóveis) escritos como uma gigante árvore de decisão.
- A indução de árvores de decisão é uma das mais simples e bem sucedidas formas de aprendizagem automática.

Aprender árvores de decisão

Problema: decidir se se espera por uma mesa num restaurante baseado nos seguintes atributos:

1. Alternate: existe um restaurante alternativo próximo?
2. Bar: existe uma área com um bar para esperar?
3. Fri/Sat: hoje é Sexta ou Sábado?
4. Hungry: temos fome?
5. Patrons: número de pessoas no restaurante (None, Some, Full)
6. Price: nível de preço ($, $$, $$$)
7. Raining: está a chover na rua?
8. Reservation: fizemos uma reserva?
9. Type: tipo de restaurante (French, Italian, Thai, Burger)
10. WaitEstimate: tempo estimado de espera (0-10, 10-30, 30-60, >60)

Representações Baseadas em Atributos

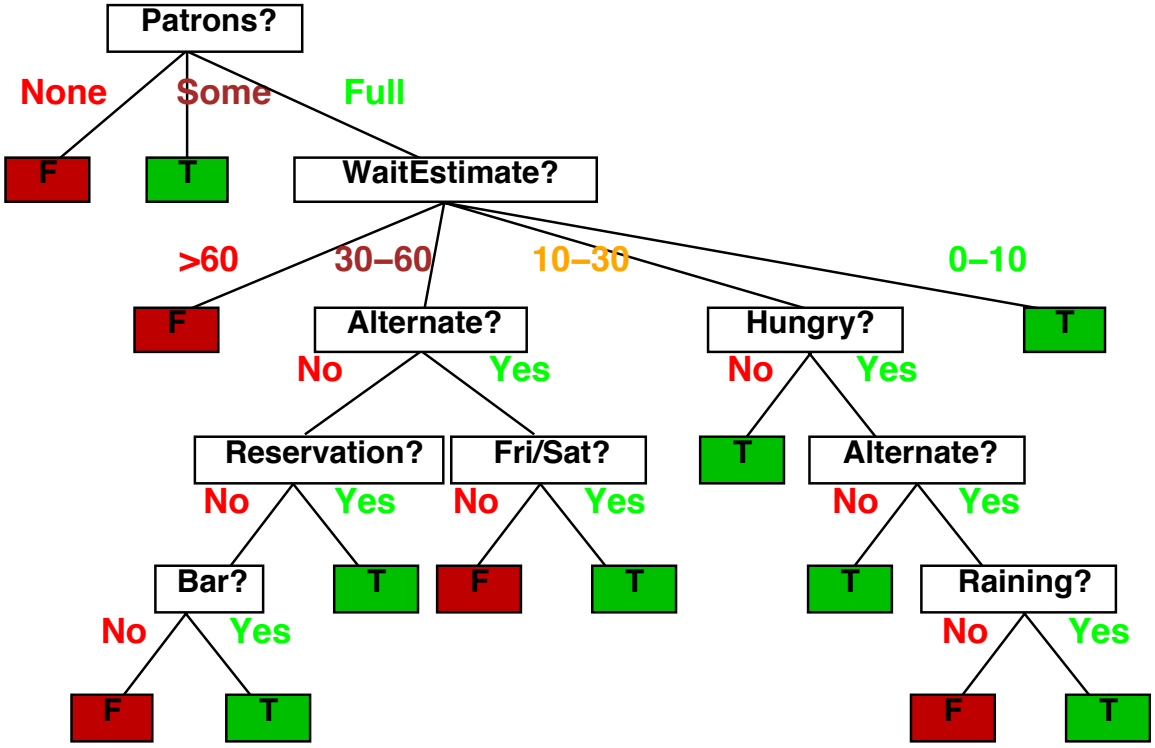
- Exemplos descritos por valores de atributos (Booleanos, discretos, contínuos)
- E.g., situações em que espero/não espero por uma mesa:

Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

- Classificação dos exemplos é positiva (T) ou negativa (F)

Árvores de Decisão

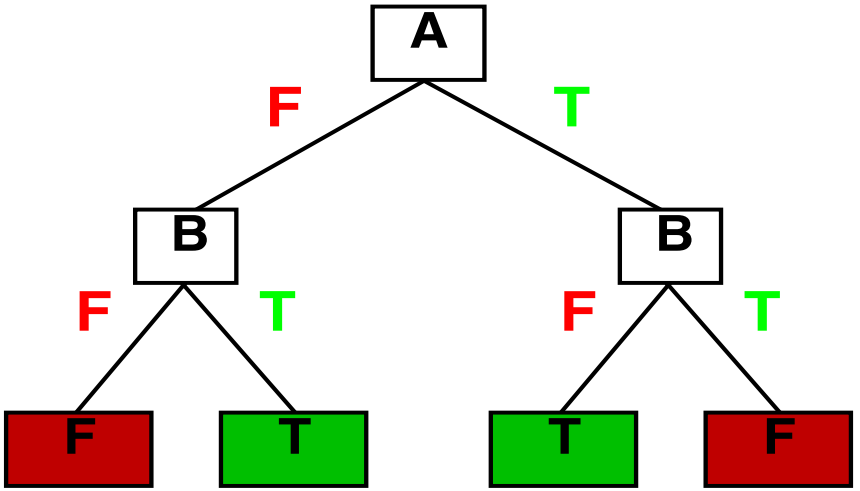
- Esta árvore de decisão é um dos elemento do espaço de hipóteses (todas as árvores com os atributos e valores possíveis).
- Coincide com a árvore de decisão "real", mas que é desconhecida à priori.



Expressividade

- Para as funções Booleanas, linha na tabela de verdade → caminho até à folha:

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



- Obviamente, existe uma árvore consistente para qualquer conjunto de treino tendo um caminho para a folha para cada exemplo (a não ser que f seja não determinista em x) mas provavelmente não é generalizável para novos exemplos

- Será preferível encontrar árvores de decisão mais compactas

Espaço de hipóteses

- Quantas árvores de decisão com n atributos Booleanos existem?
  - ≥ número de funções Booleanas
  - ≥ número distinto de tabelas de verdade com 2^n linhas = 2^{2^n}

- Exemplo, com 6 atributos Booleanos existem 18,446,744,073,709,551,616 funções Booleanas (e ainda um maior número de árvores!)

- Quantas hipóteses conjuntivas existem (e.g., Hungry ∧ ¬Rain)?
  - Cada atributo pode ocorrer positivamente, negativamente, ou de fora
    - ⇒ 3^n hipóteses conjuntivas distintas

- O espaço de hipóteses das árvores de decisão é mais expressivo
  - Aumenta a possibilidade que a função alvo seja expressível
  - Aumenta o número de hipóteses consistentes com o conjunto de treino
  - ⇒ pode resultar em predições piores!
  - É importante termos algoritmos engenhosos para encontrar boas hipóteses num espaço tão grande...

Utilizações típicas das árvores de decisão

- Exemplos são representados por pares atributos-valor
- A função alvo tem valores de saída discretos (existem extensões para lidar funções contínuas)
- Descrições disjuntivas podem ser necessárias
- O conjunto de treino pode conter erros
- O conjunto de treino pode conter alguns atributos sem valor (veremos apena o caso em que isto não acontece)

Indução de Árvores de Decisão

- Objectivo: encontrar uma árvore pequena consistente com os exemplos de treino

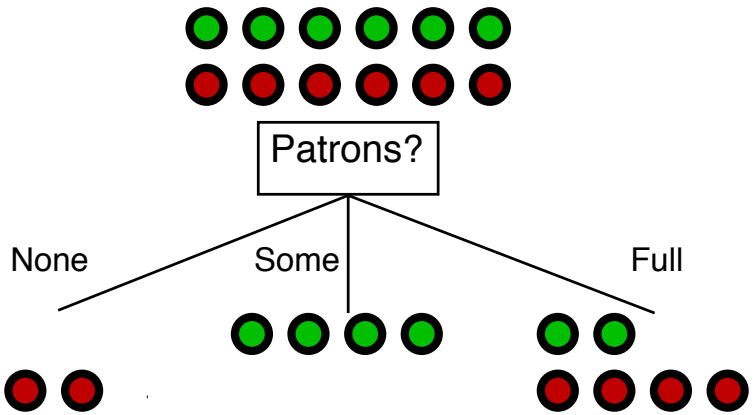
- Ideia: escolher (recursivamente) o atributo "mais significativo" como raiz da sub(árvore)

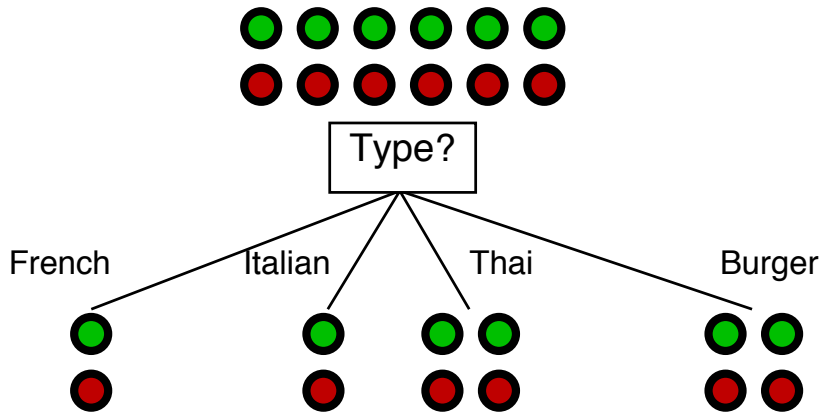
- Casos a tratar:
  1. Se todos os exemplos remanescentes são positivos ou negativos, devolvemos Yes/No, respectivamente.
  2. Se não existirem mais exemplos, devolve-se um valor por omissão (o que tiver mais ocorrências no nó pai – moda).
  3. Se existirem exemplos positivos e negativos, escolher o atributo mais significativo.
  4. Se não existirem mais atributos e houver exemplos positivos e negativos, temos um problema (ruído, não-determinismo, …). Utiliza-se a moda do conjunto de exemplos remanescente.

O algoritmo DTL (ou ID3)

```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value v_i of best do
            examples_i ← {elements of examples with best = v_i}
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```

Escolha de um atributo





- Qual a melhor escolha?
  - Patrons? Porquê?
- Ideia: um bom atributo divide os exemplos em subconjuntos que (idealmente) são “todos positivos" ou “todos negativos”

Utilizando teoria da informação

- Como implementar `Choose-Attribute` no algoritmo DTL?
- Conteúdo de Informação (Entropia – medida de incerteza):

$$H\left(P(v_1),\ldots,P(v_d)\right)=\sum_{i=1}^{d}P(v_i)\log_2\frac{1}{P(v_i)}=\sum_{i=1}^{d}-P(v_i)\log_2 P(v_i)$$

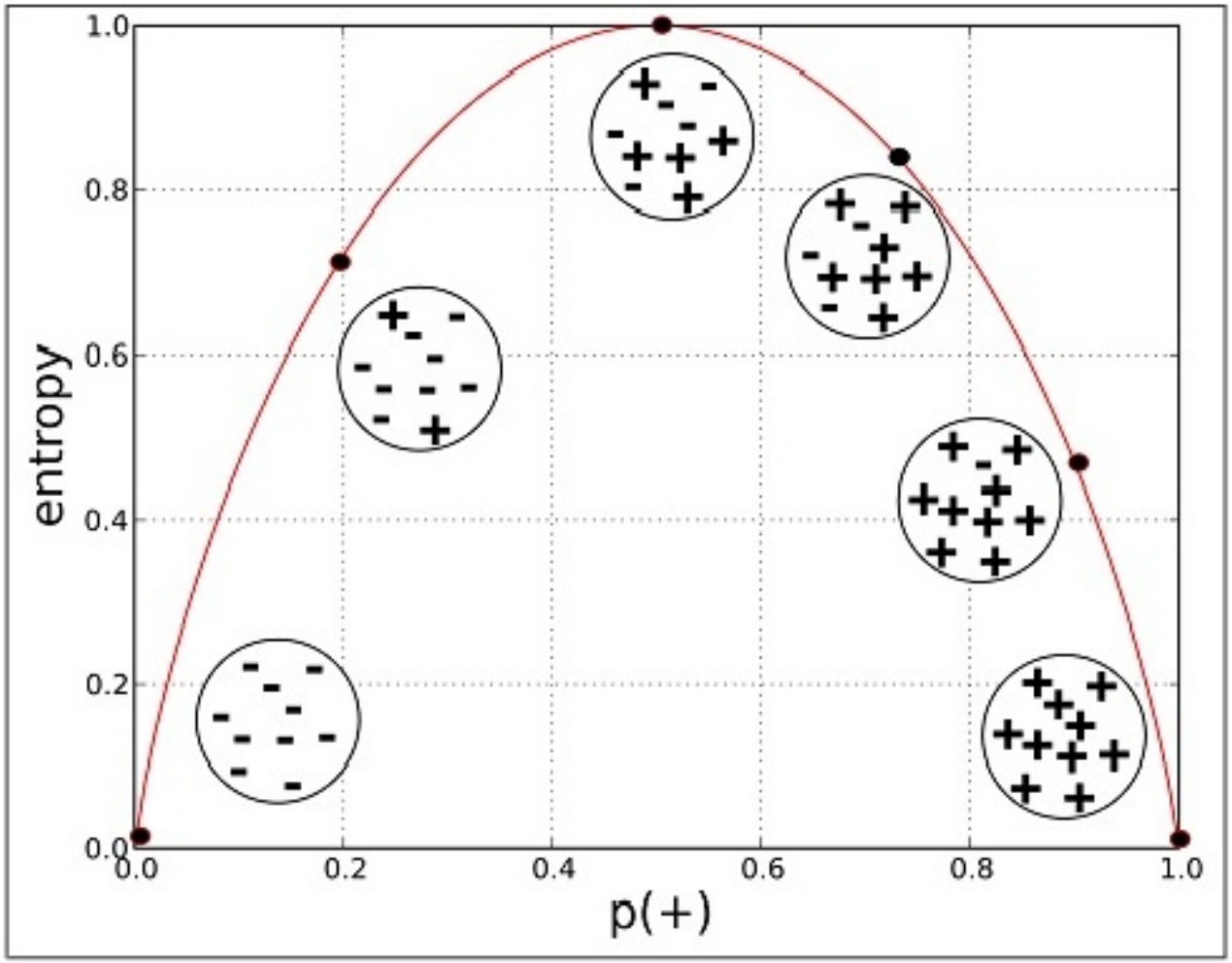
- No caso de um booleano, e para um conjunto de treino contendo p exemplos positivos e n exemplos negativos:

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

- Com $q=(p/p+n)$ = probabilidade da variável ser verdadeira:

$$H\left(q,1-q\right)=-q\log_2 q-\left(1-q\right)\log_2\left(1-q\right)$$

Curva de entropia (binária)



Ganho de Informação

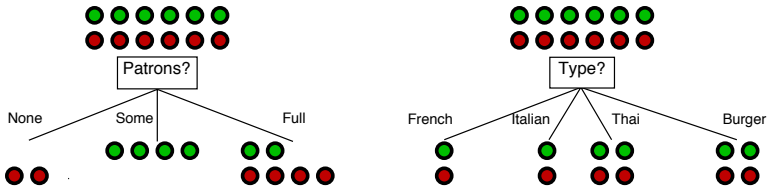
- Um atributo *A* escolhido divide o conjunto de treino *E* em subconjuntos $E_1, \ldots, E_d$ de acordo com os valores que *A* toma, em que *A* tem d valores distintos.

$$remainder(A) = \sum_{i=1}^{d} \frac{p_i + n_i}{p + n} H\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

- Ganho de Informação (IG) ou redução de entropia no atributo de teste:

$$IG(A) = H\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - remainder(A)$$

- Escolher o atributo com maior IG



Ganho de Informação

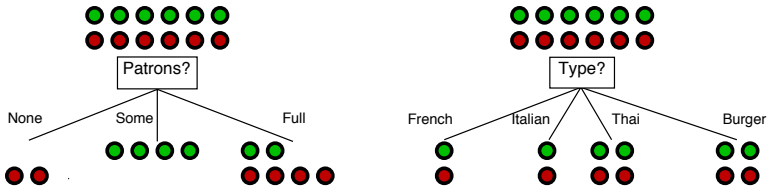
Para o conjunto de treino, $p = n = 6$, $H(6/12, 6/12) = 1$ bit

Consideremos os atributos *Patrons* e *Type* (assim como os outros…):

$$IG(Patrons) = 1 - \left[\frac{2}{12}H\left(\frac{0}{2},\frac{2}{2}\right) + \frac{4}{12}H\left(\frac{4}{4},\frac{0}{4}\right) + \frac{6}{12}H\left(\frac{2}{6},\frac{4}{6}\right)\right] = .0541 \text{ bits}$$

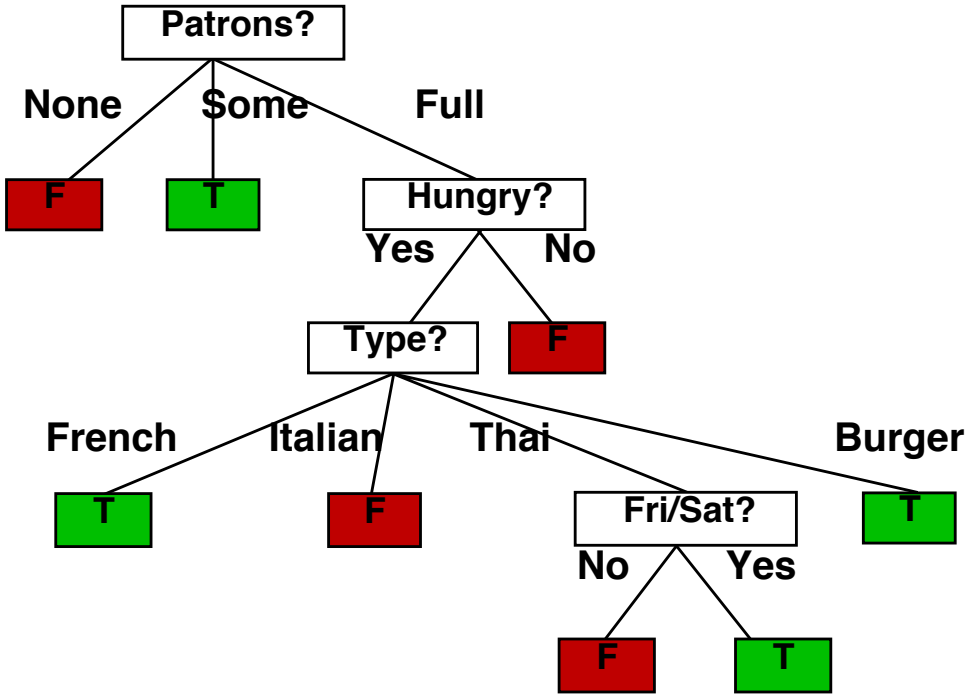
$$IG(Type) = 1 - \left[\frac{2}{12}H\left(\frac{1}{2},\frac{1}{2}\right) + \frac{2}{12}H\left(\frac{1}{2},\frac{1}{2}\right) + \frac{4}{12}H\left(\frac{2}{4},\frac{2}{4}\right) + \frac{4}{12}H\left(\frac{2}{4},\frac{2}{4}\right)\right] = 0 \text{ bits}$$

Patrons tem o maior IG de todos os atributos e portanto é escolhido pelo algoritmo DTL como raiz da árvore de decisão



Exemplo (continuação)

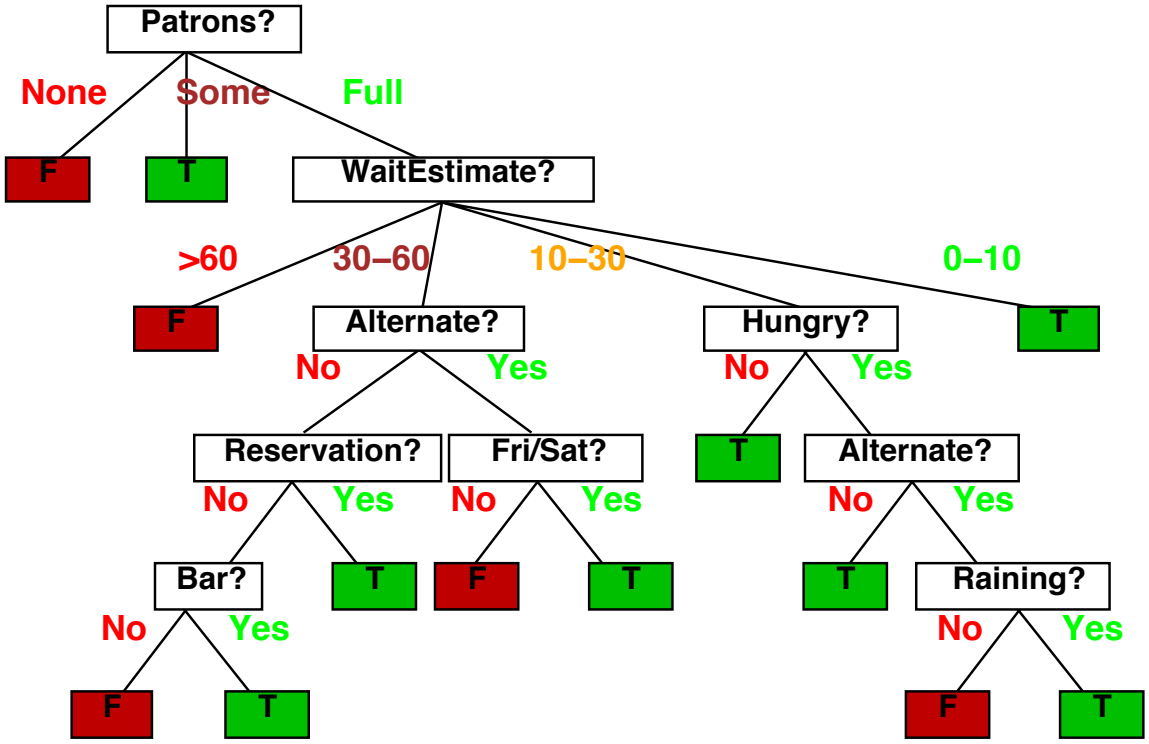
- Árvore de Decisão aprendida a partir dos 12 exemplos:



- Muito mais simples que a árvore "real" – uma hipótese mais complexa não é justificada pelo pequeno conjunto de dados

Árvores de Decisão

- Recordemos a árvore de decisão "real" para decidir se se espera:



Medida de desempenho

- Como é que sabemos que h ≈ f ?
  1. Usar teoremas da teoria da aprendizagem computacional/ estatística
  2. Aplicar h num novo conjunto de exemplos de teste

- Curva de aprendizagem = % de testes corretos em função da dimensão do conjunto de treino

Como avaliar a performance

1. Coleccionar um grande conjunto de dados
2. Dividir esses dados arbitrariamente em dois conjuntos disjuntos: conjunto de treino e conjunto de teste
3. Aplicar o algoritmo de aprendizagem ao conjunto de treino, gerando uma hipótese *h*.
4. Medir a percentagem de exemplos no conjunto de teste classificados correctamente por *h*.
5. Repetir os passos 1 a 4 para diferentes tamanhos dos conjuntos de treino e conjuntos de teste seleccionados aleatoriamente para cada tamanho.

Técnicas de validação

- Holdout validation
  Escolhem-se aleatoriamente exemplos do conjunto de dados para formar o conjunto de teste (habitualmente menos de 1/3 dos valores são utilizados como conjunto de teste)

- K-fold cross-validation
  Particionam-se os dados em K conjuntos de dimensão idêntica. Escolhe-se um desses como conjunto de teste e os restantes como conjuntos de treino. Repete-se o processo para cada subconjunto K e efectua-se a média dos resultados.

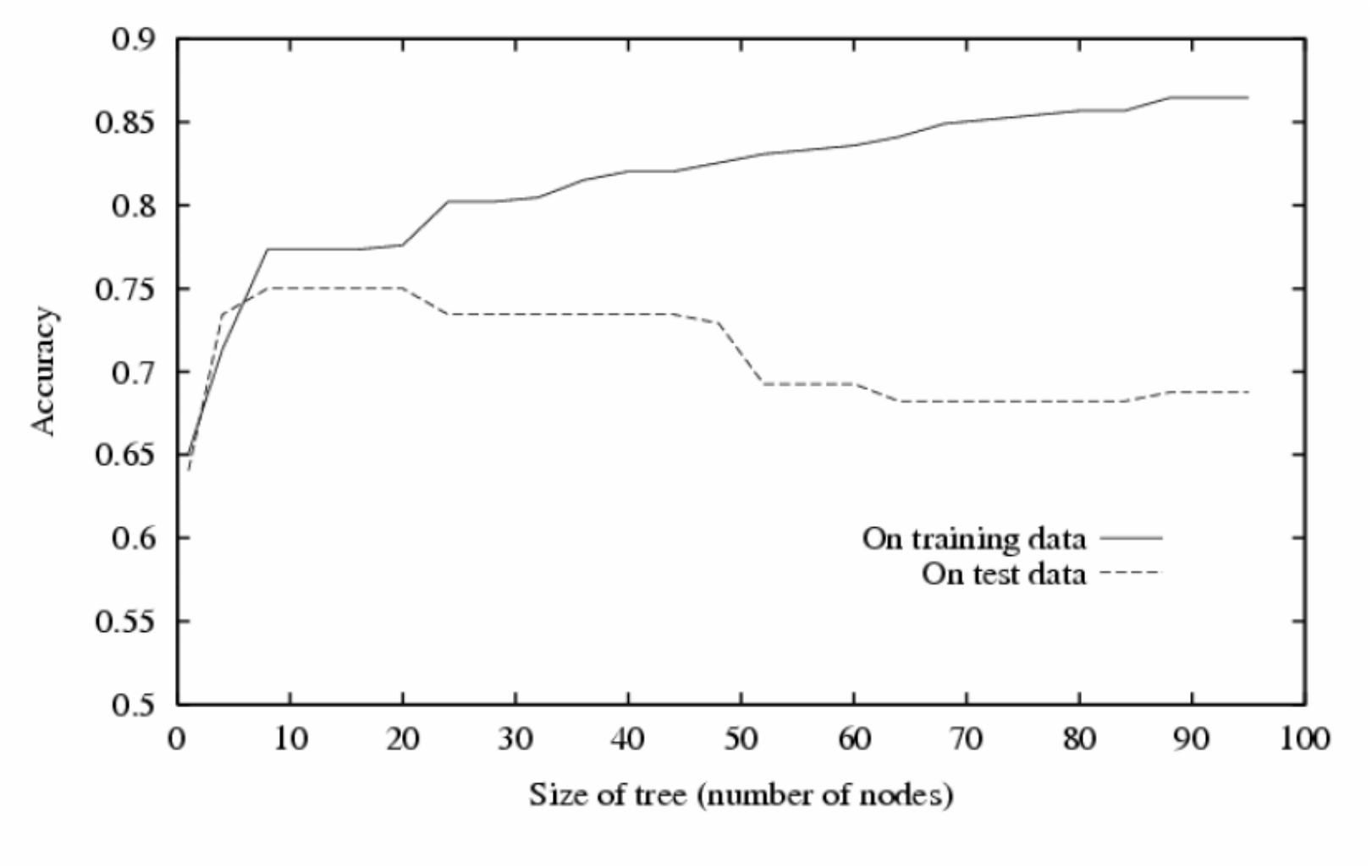
- Leave-one-out cross-validation
  Deixa-se um exemplo de fora e treina-se com todos os restantes exemplos. Repete-se para cada exemplo (K-fold cross-validation com K igual ao número de exemplos…)

Aspectos Práticos de Utilização

- Se existirem muitos atributos irrelevantes, ou ruído, é mais fácil encontrar uma hipótese exacta.
- Essa hipótese é totalmente espúria
- Neste caso estamos na presença de sobreajustamento na árvore de decisão
- Existem algoritmos mais sofisticados para lidar com estes casos (e.g. C4.5)

- Exemplo:
  - Podemos juntar um número irrelevante de atributos ao nosso problema (Cor do casaco, Transporte, Jornal) de maneira que exista uma única instância para cada caso.
  - Nestas situações o algoritmo DTL encontrará uma hipótese exacta, mas não generalizará bem para o conjunto de validação.

Exemplo de curva de aprendizagem



Extensões

- Omissão de valores para atributos ?
  - Como classificar uma instância com algum valor desconhecido num atributo de teste
  - Como alterar a fórmula de ganho de informação?
- Como lidar com atributos multivalor ?
  - Exemplo típico é o atributo Data
  - Necessita de outra medida: Rácio do Ganho
- Como lidar com atributos inteiros ou contínuos ?
  - Pontos de separação (splitting)
- Como tratar atributos de saída contínuos ?
  - Árvore de regressão – cada folha é uma função linear sobre um subconjunto dos atributos numéricos

Sumário

- Aprendizagem de árvores de decisão utiliza o ganho de informação
- Desempenho do algoritmo de aprendizagem = precisão no(s) conjunto(s) de teste.