

# Aprendizagem a partir de Observações



“Descrição de agentes que podem melhorar o seu comportamento a partir do estudo diligente das próprias experiências”

- Formas de aprendizagem
- Aprendizagem indutiva
- Aprendizagem com árvores de decisão
- Teoria de aprendizagem computacional
- Sumário



# Aprendizagem Automática

O Que é?



# Aprendizagem

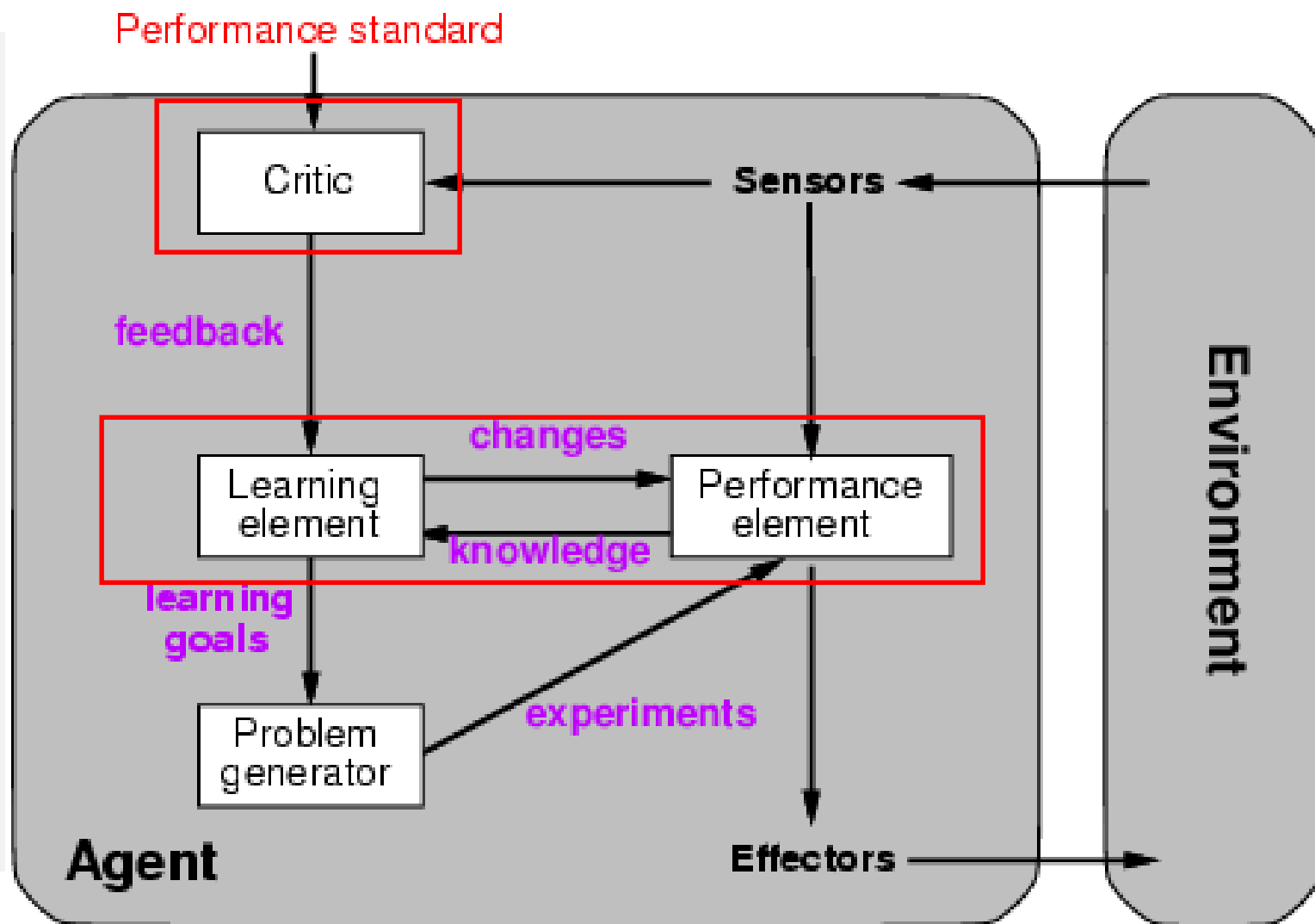
- A ideia por detrás da aprendizagem é que as percepções de um agente devem servir **não apenas para o agente agir, mas igualmente para melhorar a sua capacidade de acção no futuro**

# Aprendizagem

- Aprendizagem é essencial em **ambientes desconhecidos**
  - i.e., quando não há omnisciência
- Aprendizagem é útil como **método construtivo**
  - i.e., o agente é exposto à realidade em vez de ser completamente definido
- Aprendizagem **modifica o mecanismo de decisão dos agentes** de modo a melhorar o seu desempenho



# Agentes aprendizes



# Agente aprendiz

- Correspondem à ideia de máquina inteligente caracterizada por Turing (1950)
  - Agente pode actuar num mundo inicialmente desconhecido
- **Elemento de aprendizagem**
  - Torna o agente mais eficiente ao longo do tempo
  - Usa *feedback* da crítica que avalia actuação do agente de acordo com o desempenho espectável
- **Elemento de desempenho**
  - Responsável por seleccionar as acções do agente
- Elemento de geração de problemas
  - Sugere acções que podem trazer informação útil

# Elementos da aprendizagem

- Componentes do desempenho que podem/devem ser aprendidos (ex: agente a “aprender” a ser taxista)
  - **Mapeamento entre as condições do estado corrente e acções**
    - Se o instrutor lhe gritar muitas vezes “break” sempre que há um sinal vermelho/um carro em frente/etc., pode aprender este par condições/acção
  - **Propriedades do mundo que podem ser inferidas a partir de uma sequência de percepções**
    - Vendo muitas imagens com autocarros, pode aprender a reconhecê-los
  - **Informação sobre o modo como o mundo evolui e sobre o resultado de possíveis acções que o agente pode tomar**
    - Travando a fundo sobre piso molhado pode aprender as consequências dos seus actos
  - **Função de utilidade indicando quão desejável é um estado do mundo**
    - Se sempre que conduzir como um louco, não receber gorjeta, pode aprender uma componente muito útil para a sua função de utilidade
  - ...

# Elementos da aprendizagem

- Tipos de **feedback** (ex: agente a “aprender” a ser taxista)
  - **Aprendizagem supervisionada**: para cada exemplo é dado input e output (implica aprender uma função)
    - aprender a reconhecer autocarros: mostram-lhe imagens e dizem-lhe se a imagem contém ou não autocarros (a função a aprender recebe uma imagem e devolve um valor Booleano)
  - **Aprendizagem não supervisionada**: para cada exemplo só é dado input (implica aprender padrões a partir do input)
    - agente aprende a reconhecer (infere) o que é um bom/mau dia de tráfego
  - **Aprendizagem com reforço**: prémios ocasionais
    - gorjeta é consequência de um bom serviço

# Elementos da aprendizagem

- Representação usada para os componentes
  - Já vimos algumas
    - Funções de utilidade em jogos
    - Fórmulas lógicas para os componentes de um agente
    - ...

# Conhecimento já existente

- O último factor a ter em conta na criação de um sistema de aprendizagem é a possível existência de **conhecimento anterior**
  - Em muitos casos o agente inicialmente não tem conhecimento nenhum sobre o que vai aprender, apenas tem acesso aos exemplos apresentados na sua experiência

- Formas de aprendizagem
- **Aprendizagem Supervisionada**
- Aprendizagem com árvores de decisão
- Teoria de aprendizagem computacional
- Sumário

# Aprendizagem Supervisionada

- Considere-se um algoritmo determinístico de aprendizagem supervisionada
- Seja um **exemplo** um par  $(x, f(x))$ , em que:
  - $x$  é o input
  - $f(x)$  é o output da função quando aplicada a  $x$
- A tarefa de **inferência indutiva (ou indução) pura** define-se do seguinte modo:

“Dada uma colecção de exemplos (**conjunto de treino**) de  $f$ , encontrar uma função  $h$  (**hipótese**) que seja uma aproximação de  $f$  ( $h \approx f$ ).”



# Aprendizagem Supervisionada

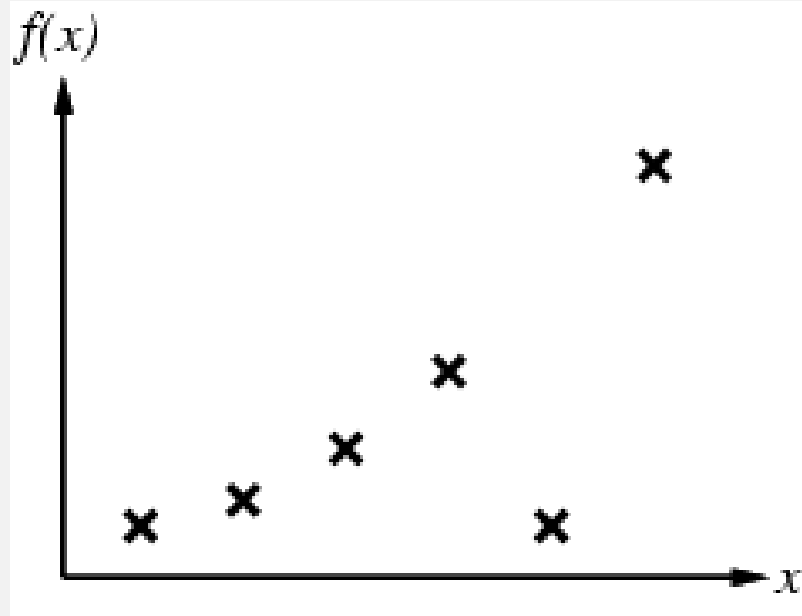
- Só que não é fácil dizer se uma dada função  $h$  é ou não uma boa aproximação de  $f$ .
- $h$  deve ser capaz de **generalizar bem**, ou seja, deve ser capaz de prever correctamente exemplos que nunca foram vistos.

Diz-se que  **$h$  é consistente** se está de acordo com  $f$  em todos os exemplos

# Aprendizagem

## Supervisionada: método

- Considere-se que são dados alguns pontos e há que adaptar uma função com uma única variável a esses dados.
- $h$  é **consistente** se está de acordo com  $f$  em todos os exemplos
- E.g., obter  $h$  para:



# Aprendizagem Supervisionada

Dado um conjunto de treino de  $N$  pares input-output

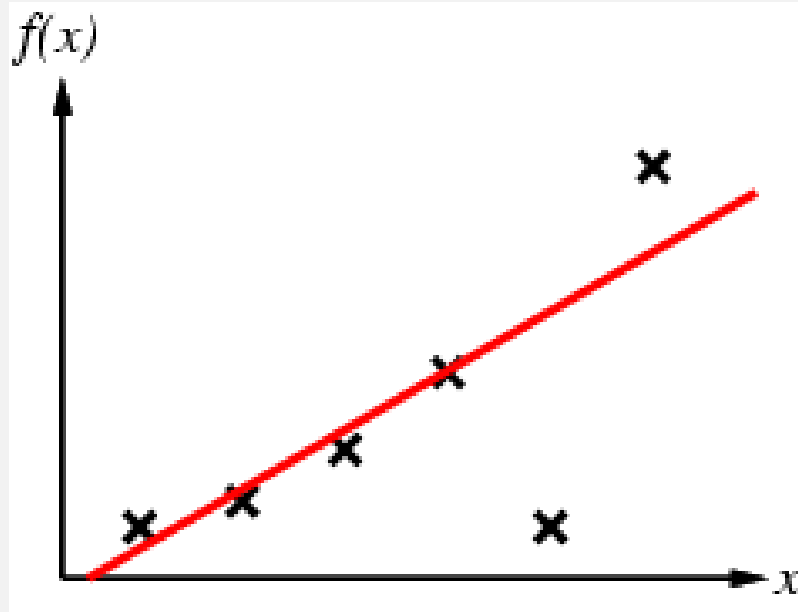
- $(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N),$

Em que  $y_k$  foi gerado por uma função desconhecida  $y=f(x)$

**OBJECTIVO:** Descobrir uma função  $h$  que se aproxima à função  $f$ .

# Aprendizagem indutiva exemplo

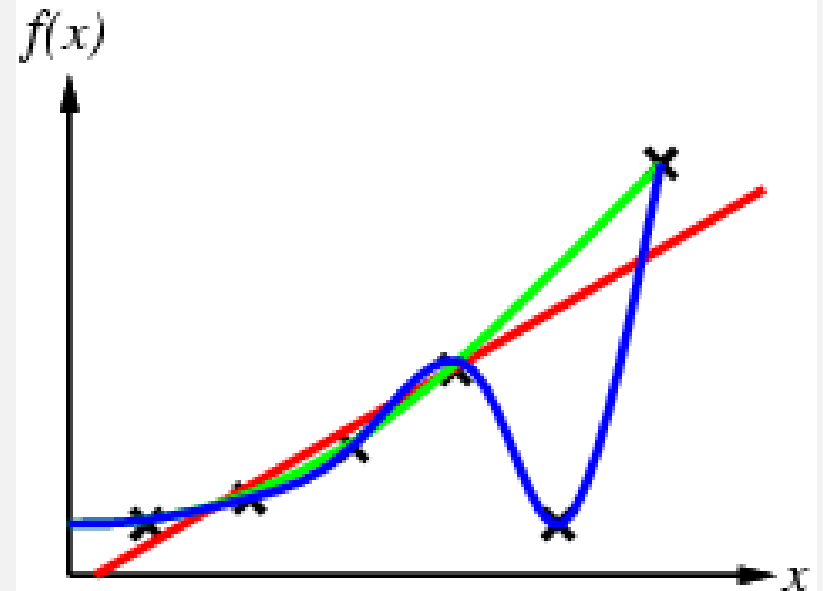
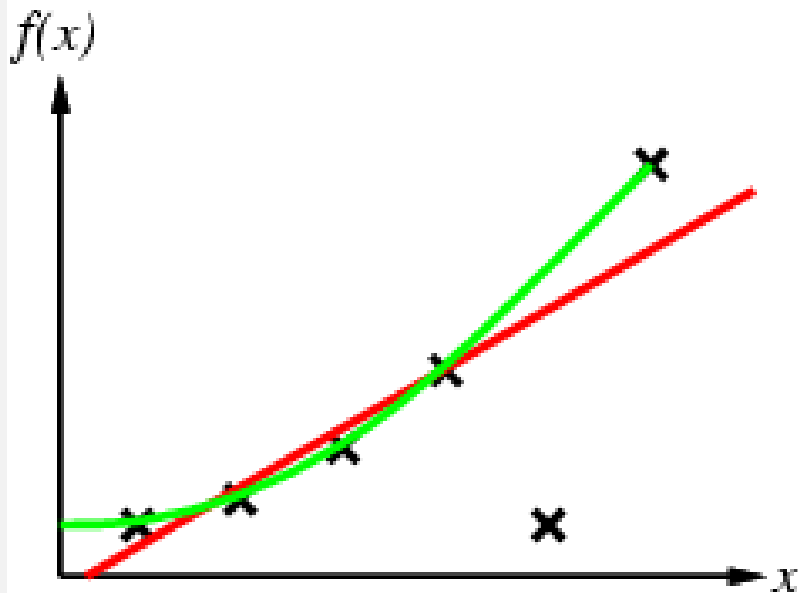
- Considerando um polinómio de grau 1



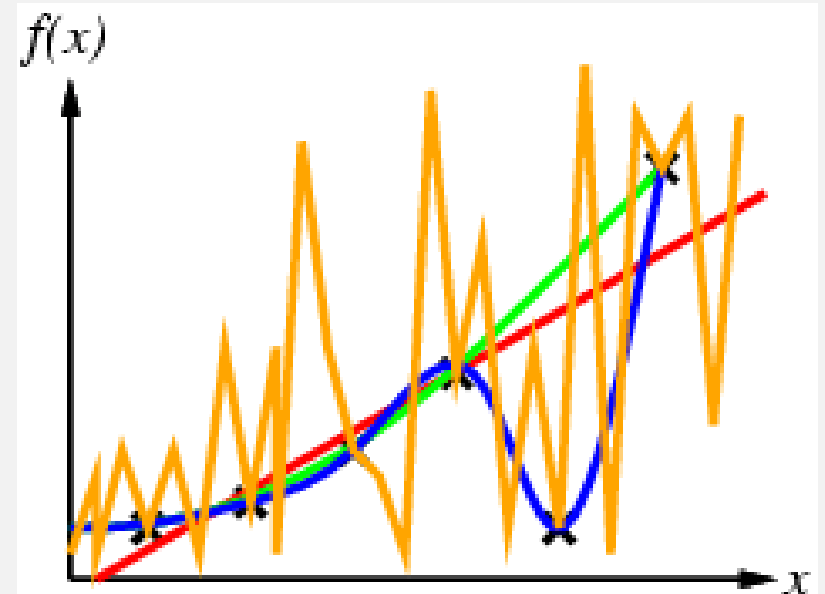
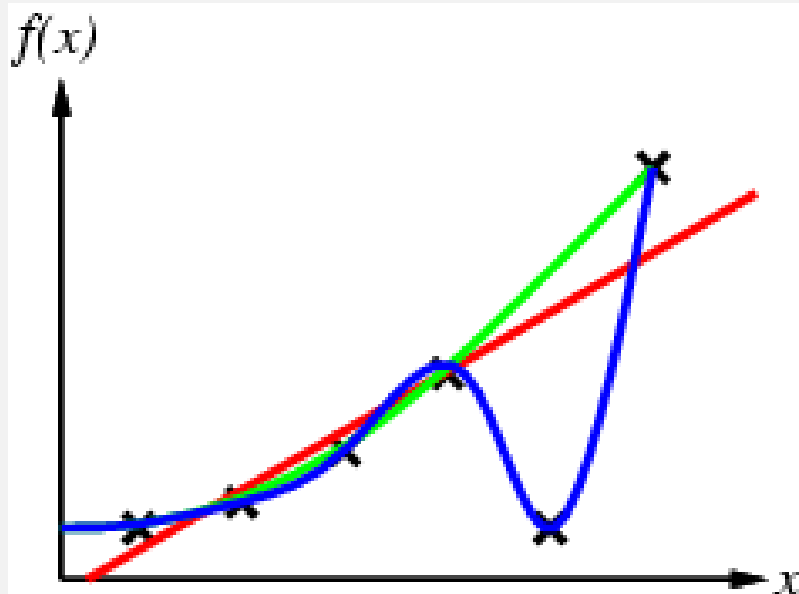
# Aprendizagem indutiva

## exemplo

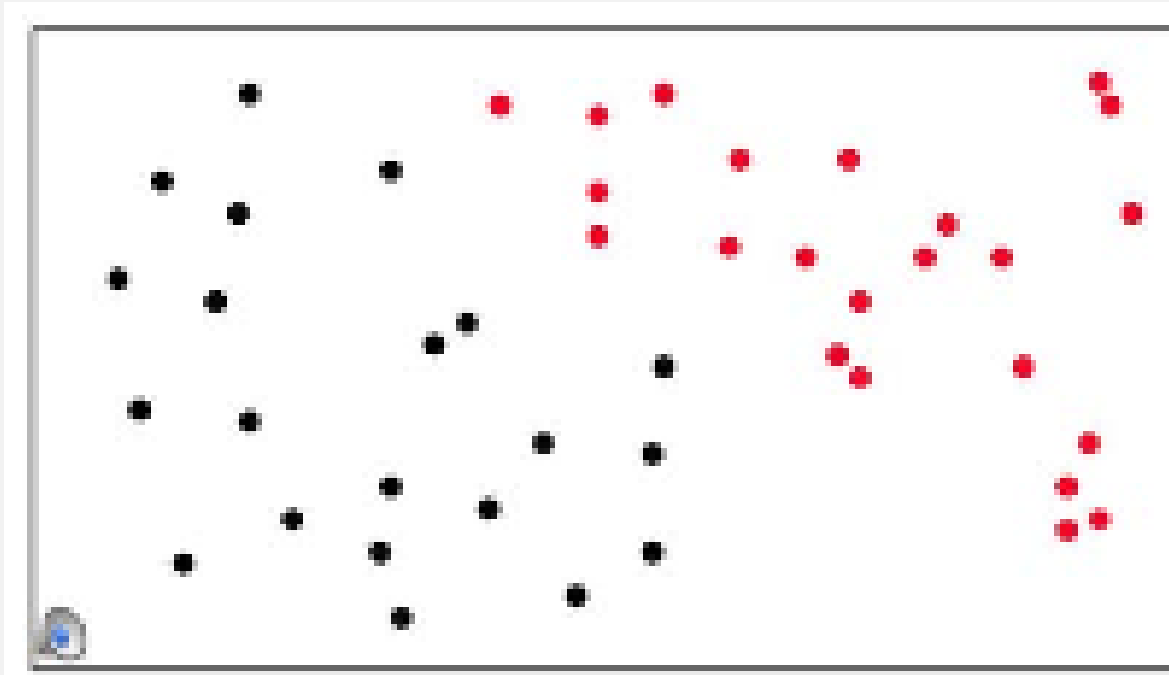
- Considerando polinómios de maior grau



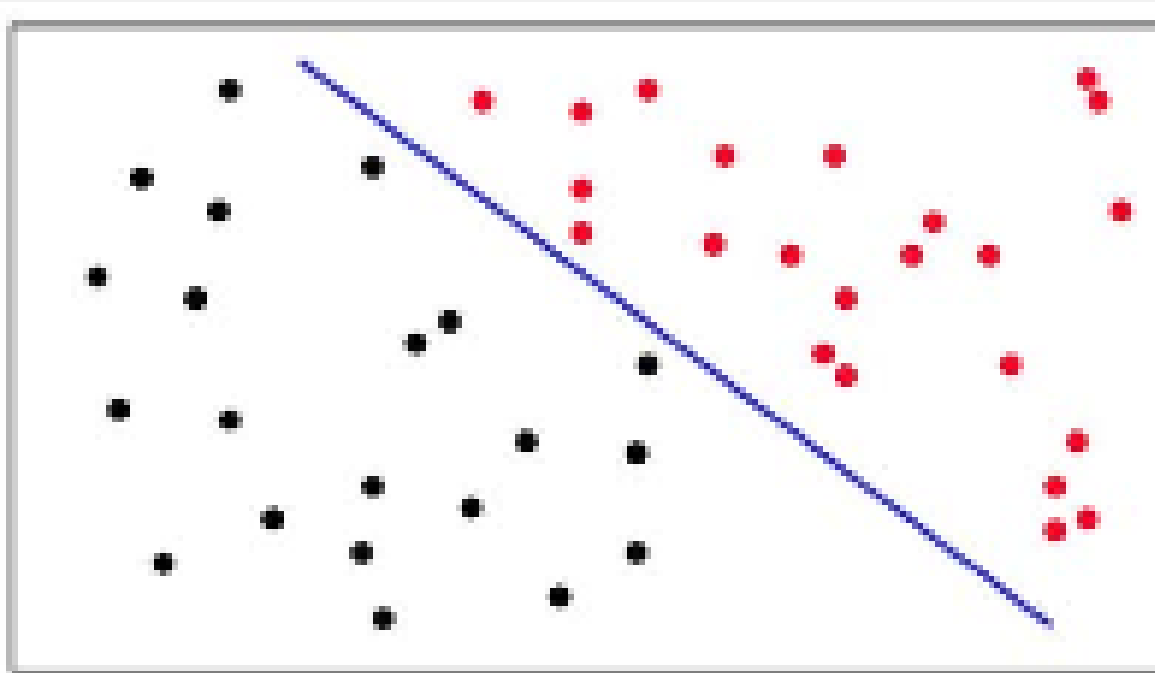
# Aprendizagem indutiva exemplo



# Outro exemplo: distinção entre dois valores possíveis

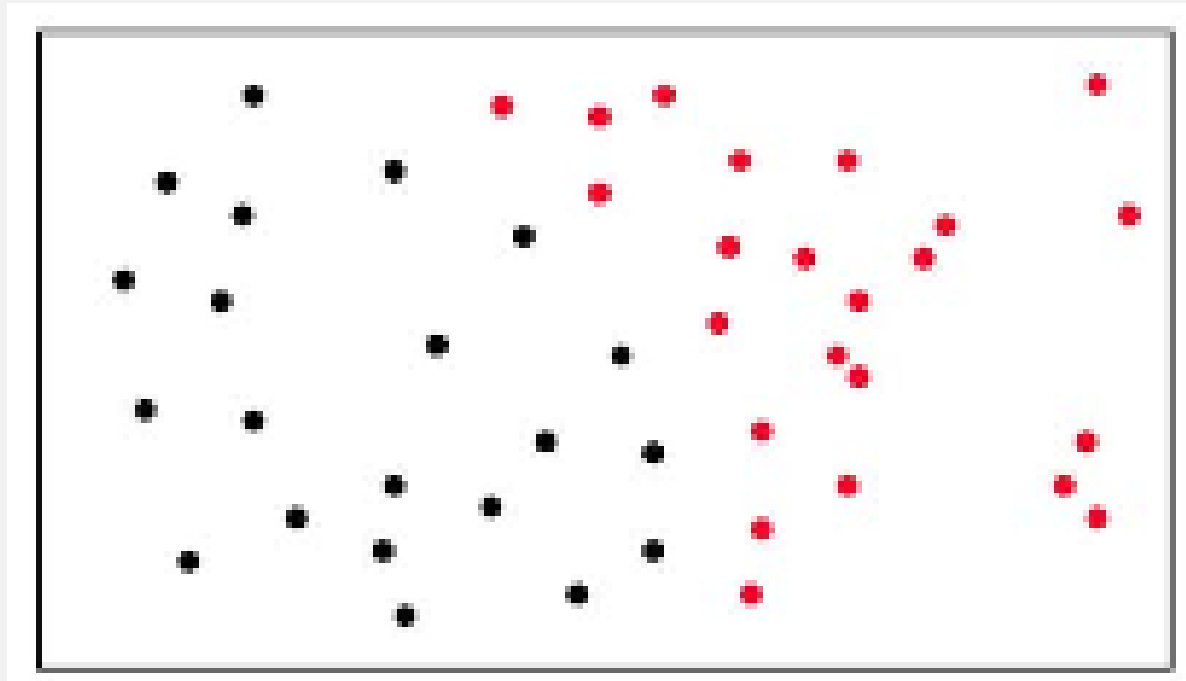


# Hipótese correcta?

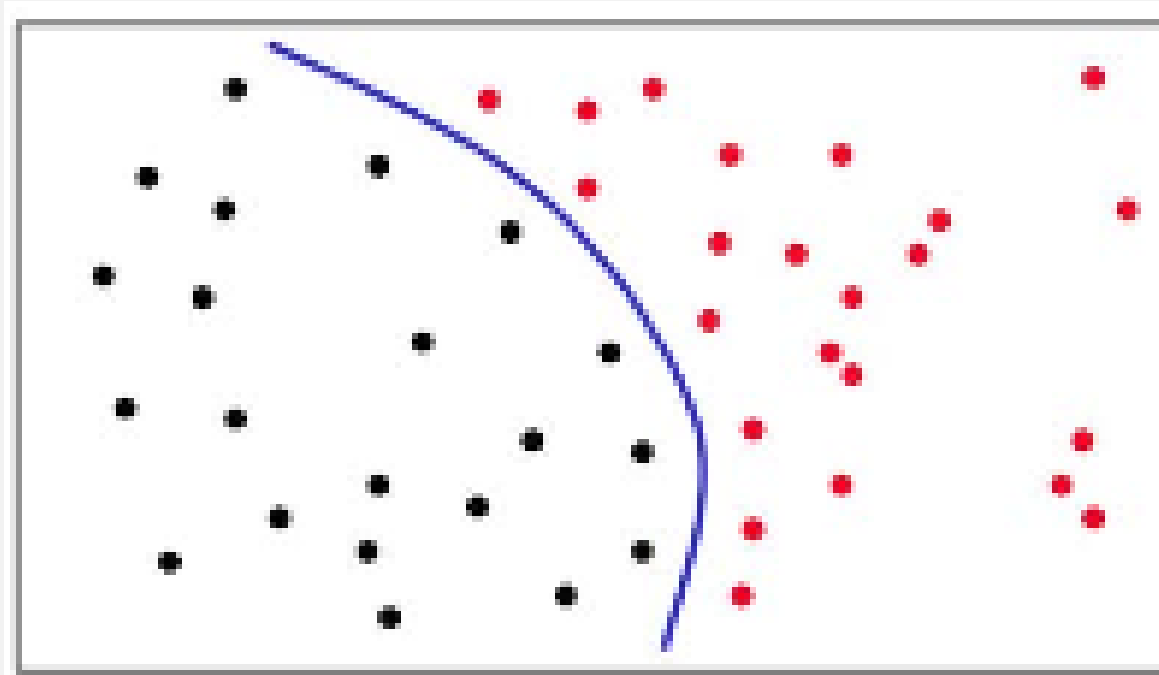




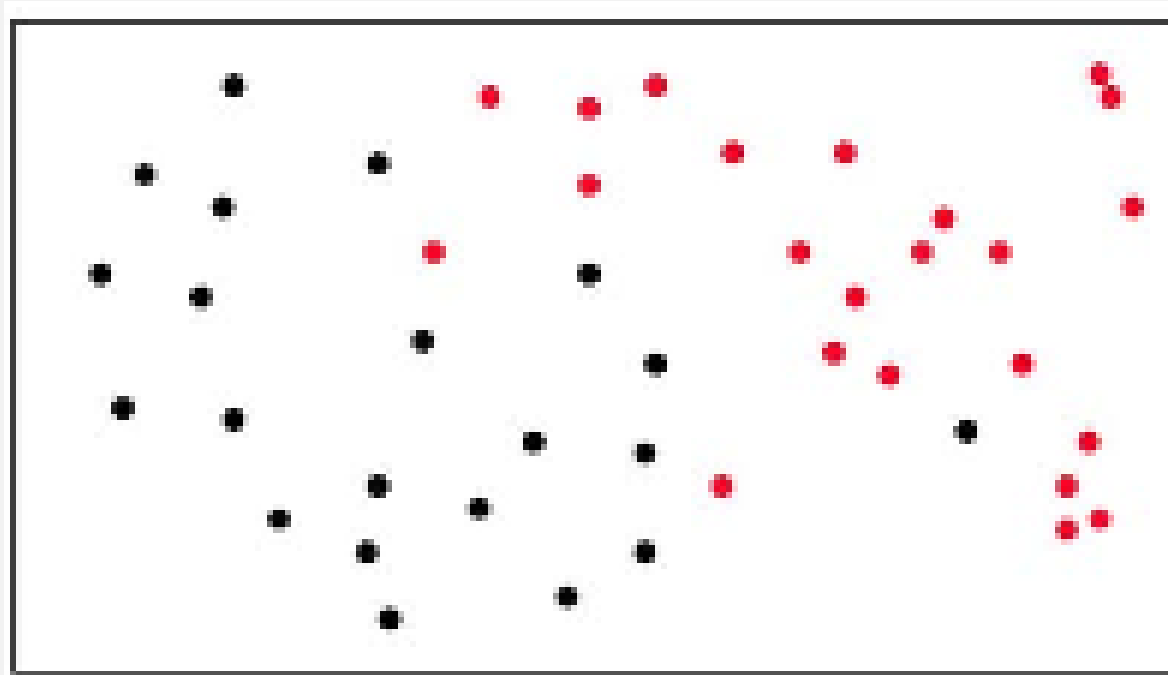
# E agora?



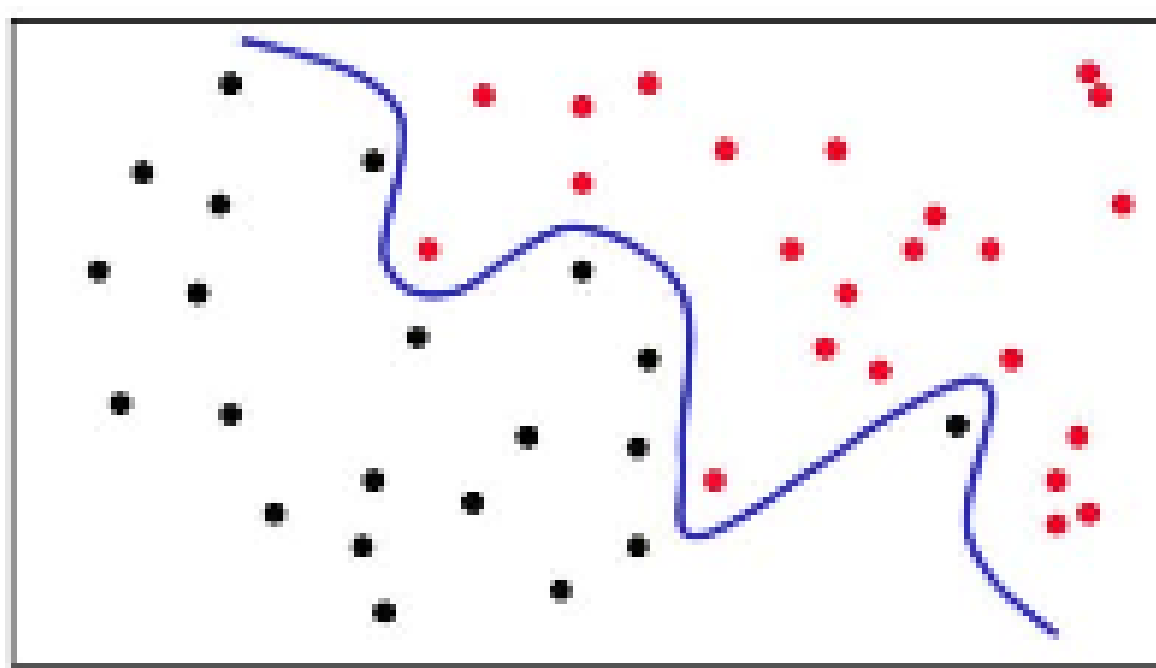
# Hipótese correcta?



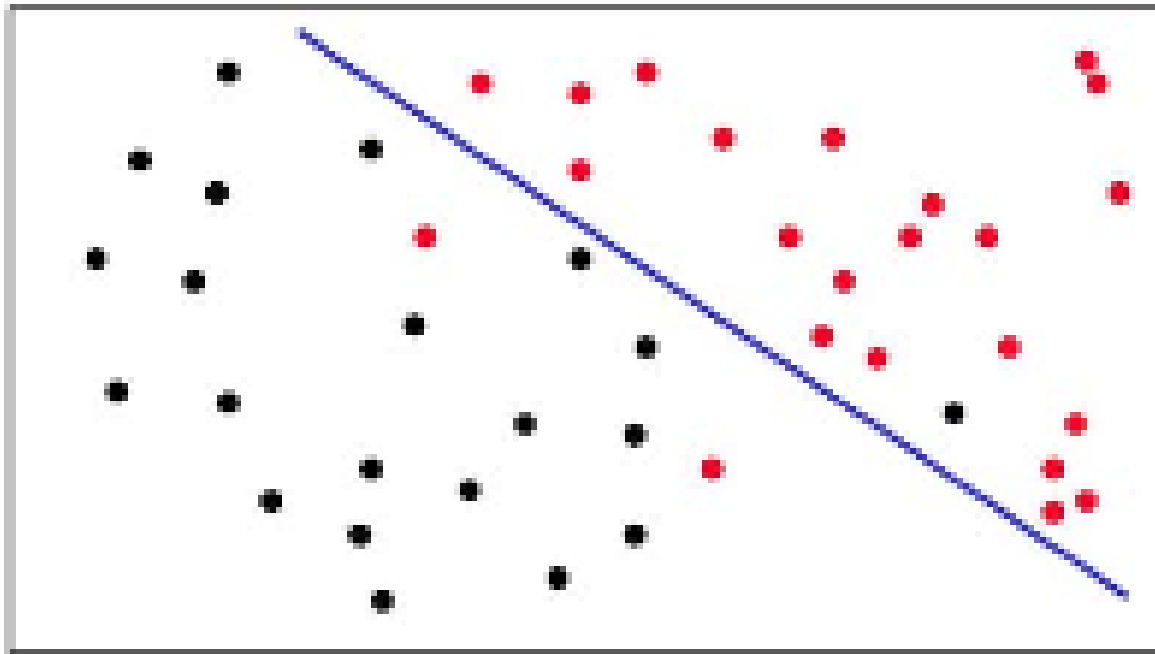
# E agora?



# Hipótese 1



# Hipótese 2



# Generalização

- A função  $h$  é chamada a hipótese

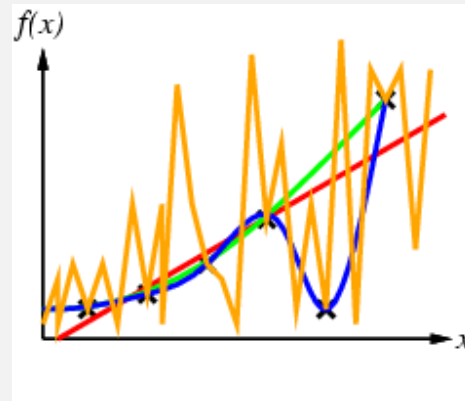
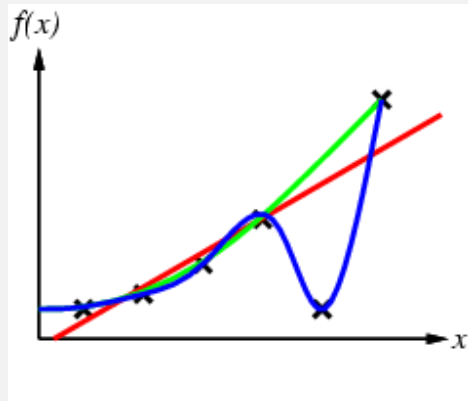
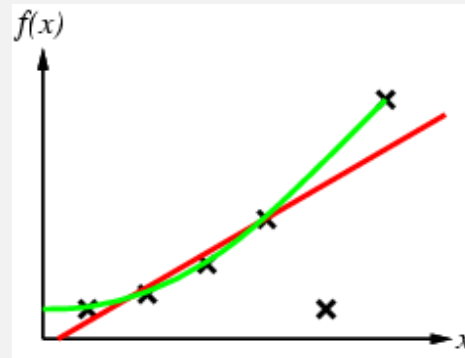
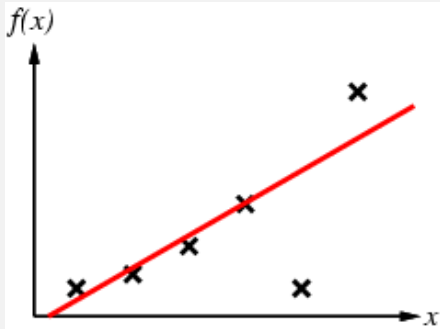
Aprendizagem é uma procura no espaço de possíveis hipóteses tentando encontrar aquela que tem melhor desempenho, mesmo em novos exemplos (alem do conjunto de treino).

Diz-se que uma hipótese generaliza bem se consegue prever o valor de  $y$  para novos exemplos.

# Classificação vs Regressão

- Quando o output  $y$  é um conjunto finito de valores (por exemplo, sunny, cloudy ou rainy) o problema de aprendizagem é denominado de **CLASSIFICAÇÃO**
- Quando o output  $y$  é um numero (por exemplo, uma temperatura) o problema de aprendizagem é denominado de **REGRESSÃO**

# Espaço de hipóteses



Como escolher?



# Aprendizagem Supervisionada

## Regra de Ockham

- Qual escolher?
  - Regra de Ockham: prefere-se a hipótese mais simples que seja **consistente** com os dados
    - Um polinómio de grau 1 é certamente mais simples que um de grau 12.

# Aprendizagem indutiva

## O espaço de hipóteses

- Há que ter em mente que a possibilidade (ou impossibilidade) de encontrar uma hipótese consistente depende do espaço de hipóteses escolhido.
  - Por exemplo, um espaço definido com polinómios pode não chegar a encontrar uma hipótese consistente
- Um problema de aprendizagem é **realizável** se o espaço de hipóteses contém a função  $f$ ; caso contrário é **não realizável**
- Usar o maior espaço de hipóteses?
  - Só que quando mais complexo for o espaço de hipóteses, mais complicado é encontrar funções que se adaptem aos dados

# Tradeoff



Em geral existe um “tradeoff” entre as hipóteses complexas que de facto se adequam bem aos dados e hipóteses simples, mas que generalizam melhor.....

- Formas de aprendizagem
- Aprendizagem supervisionada
- **Aprendizagem com árvores de decisão**
- Teoria de aprendizagem computacional
- Sumário

- **Aprendizagem com árvores de decisão**
  - Conceitos básicos
  - Expressividade das árvores de decisão
  - Indução de árvores de decisão a partir de exemplos
  - Escolha de testes atributo
  - Avaliação da performance do algoritmo de aprendizagem
  - Ruído e overfitting
  - Expansão da aplicabilidade das árvores de decisão

# Conceitos básicos



- Uma árvore de decisão é um exemplo muito simples (e de grande sucesso) de um algoritmo de aprendizagem que se enquadra na área de aprendizagem supervisionada

# Conceitos básicos

- Uma árvore de decisão recebe como **input** um objecto ou situação descritos **por um conjunto de atributos** e devolve uma “decisão” (o valor de output para o input recebido)
  - Os atributos de input podem ser discretos ou contínuos
  - O valor de output também pode ser discreto ou contínuo
  - Vamos tratar casos de classificação Booleana em que cada exemplo é classificado como positivo ou negativo
  - **Funciona como elemento de desempenho (responsável por seleccionar acções do agente)**

# Exemplo

**Problema:** decidir se  
esperamos (esperamos?)  
por uma mesa num  
restaurante....





# Conceitos Básicos (exemplo)

**Problema:** decidir se esperamos (esperamos?) por uma mesa num restaurante, com base nos seguintes atributos:

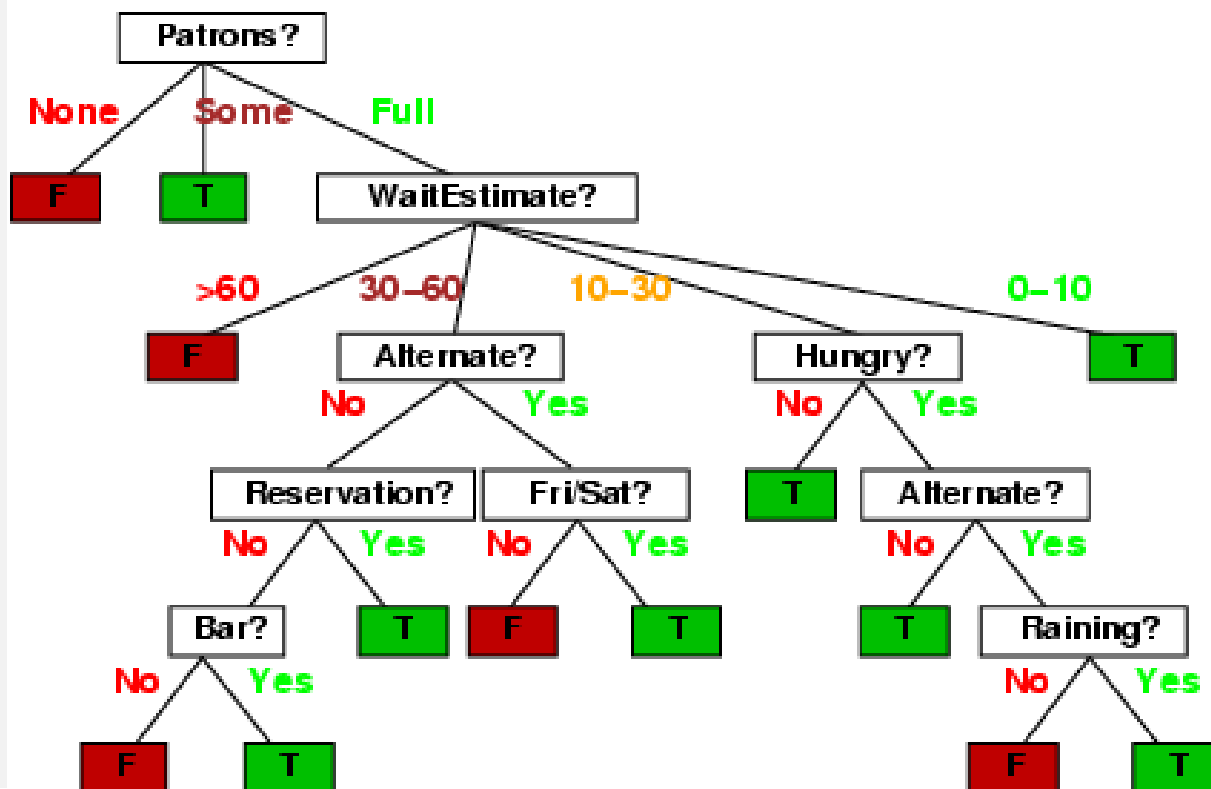
1. **Alternativa:** existe um restaurante alternativo próximo?
2. **Bar:** existe uma área de bar confortável para esperar?
3. **Sexta/Sábado:** hoje é Sexta ou Sábado?
4. **Fome:** temos fome?
5. **Clientes:** número de pessoas no restaurante (Nenhum, Algum, Cheio)
6. **Preço:** gama de preços (\$, \$\$, \$\$\$)
7. **Chuva:** está a chover lá fora?
8. **Reserva:** fizemos uma reserva?
9. **Tipo de restaurante:** Francês, Italiano, Tailandês, Burger, ...
10. Estimativa do **tempo** de espera: 0-10, 10-30, 30-60, >60

# Conceitos básicos

- Uma árvore de decisão chega a uma decisão executando uma sequência de testes.
  - Cada nó da árvore corresponde ao teste de um atributo;
  - Cada ramo está etiquetado com possíveis valores do atributo testado;
  - Cada folha representa o valor a ser devolvido se essa folha for atingida.

# Conceitos básicos (exemplo)

- Árvore para decidir se esperamos ou não (função *esperamos?*)



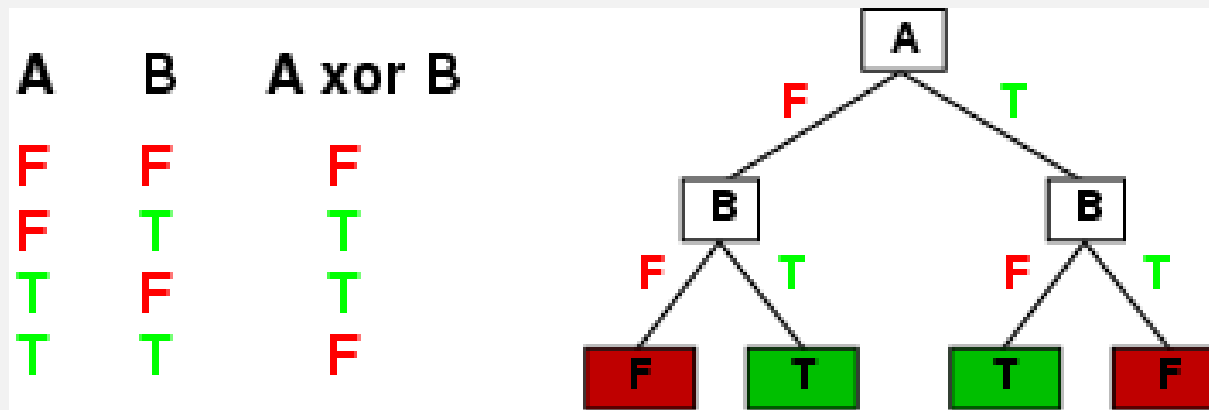
# Expressividade de árvores de decisão

- Qualquer árvore de decisão para o predicado objectivo **esperamos?** pode ser vista como a seguinte frase:
  - $\forall_s \text{esperamos?}(s) \Leftrightarrow (P_1(s) \vee P_2(s) \vee \dots \vee P_n(s))$

Cada  $P_i(s)$  é uma conjunção de testes correspondente a um caminho da raiz até uma folha com valor T

# Expressividade de árvores de decisão

- Na verdade, qualquer função Booleana pode ser representada como uma árvore de decisão
  - Cada linha da tabela de verdade é um caminho na árvore
    - Este processo leva a uma árvore de decisão de dimensão exponencial



- As árvores de decisão *podem* representar funções de modo muito mais compacto (recordar BDDs...)
  - No pior caso têm na mesma dimensão exponencial

# Expressividade de árvores de decisão (espaços de hipóteses)

- As árvores de decisão são boas para representar algumas funções e más para outras
  - Por exemplo, para representar as funções AND e OR é possível podar a árvore mas para representar as funções PARIDADE (verdadeiro sse  $n^o$  de entradas verdadeiras é par) e MAIORIA (verdadeiro sse maioria de entradas são verdadeiras) não é
- Será que existe alguma representação que é boa para todas as funções? **Não!**
  - Porque qualquer que seja a representação esta será sempre exponencial (ver slide seguinte) para algumas funções

# Expressividade de árvores de decisão (espaços de hipóteses)

- Considere-se o conjunto de todas as funções Booleanas com  $n$  atributos Booleanos
  - O número de possíveis inputs corresponde ao número de diferentes entradas da tabela de verdade ( $2^n$ )
  - O número de funções distintas (output) para  $n$  atributos é  $2^{2^n}$   
Exº para  $n=2$

Input	
X	Y
T	T
T	F
F	T
F	F

Output															
T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F
T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F
T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F
T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F

- E para 6 atributos Booleanos existem 18,446,744,073,709,551,616 árvores!

# Expressividade de árvores de decisão (espaços de hipóteses)

Quantas hipóteses puramente conjuntivas (e.g.,  $Hungry \wedge \neg Rain$ )?

- Cada atributo pode aparecer positivo ou negativo ou não aparecer  
⇒  $3^n$  hipóteses conjuntivas distintas
- Maior expressividade do espaço de estados
  - Aumenta a possibilidade da função alvo ser expressa
  - Aumenta o número de hipóteses consistentes com o conjunto de treino  
⇒ pode dar origem a previsões piores



# Como aprender árvores de decisão?

# Indução de árvores de decisão com base em exemplos

- **Conjunto de treino:**
  - exemplos descritos por **valores de atributos** (Booleanos, discretos, contínuos)
  - a **classificação** de exemplos é **positiva** (T) ou **negativa** (F)
  - objectivo: aprender **função objectivo**
- E.g., conjunto de treino para o caso **esperamos?**

Example	Attributes										Target
	<i>Alt</i>	<i>Bar</i>	<i>Fri</i>	<i>Hun</i>	<i>Pat</i>	<i>Price</i>	<i>Rain</i>	<i>Res</i>	<i>Type</i>	<i>Est</i>	<i>Wait</i>
$X_1$	T	F	F	T	Some	\$\$\$	F	T	French	0–10	T
$X_2$	T	F	F	T	Full	\$	F	F	Thai	30–60	F
$X_3$	F	T	F	F	Some	\$	F	F	Burger	0–10	T
$X_4$	T	F	T	T	Full	\$	F	F	Thai	10–30	T
$X_5$	T	F	T	F	Full	\$\$\$	F	T	French	>60	F
$X_6$	F	T	F	T	Some	\$\$	T	T	Italian	0–10	T
$X_7$	F	T	F	F	None	\$	T	F	Burger	0–10	F
$X_8$	F	F	F	T	Some	\$\$	T	T	Thai	0–10	T
$X_9$	F	T	T	F	Full	\$	T	F	Burger	>60	F
$X_{10}$	T	T	T	T	Full	\$\$\$	F	T	Italian	10–30	F
$X_{11}$	F	F	F	F	None	\$	F	F	Thai	0–10	F
$X_{12}$	T	T	T	T	Full	\$	F	F	Burger	30–60	T

# Indução de árvores de decisão com base em exemplos

- Pode ser extraída uma árvore de decisão consistente para qualquer conjunto de treino em que cada exemplo corresponde a uma caminho até uma folha, onde o caminho testa um atributo de cada vez e segue os valores do exemplo e cada folha tem a classificação do exemplo (a não ser que  $f$  seja não-determinística)
  - não há dúvida que seguindo este processo a árvore atribui os resultados correctos aos exemplos dados
  - mas provavelmente não será possível generalizá-la para novos exemplos (apenas memoriza os exemplos)
- Preferência dada a árvores de decisão **compactas**

# Indução de árvores de decisão com base em exemplos

- Objectivo: encontrar uma árvore tão pequena quanto possível consistente com os exemplos de treino
- Ideia: (recursivamente) escolher o atributo "mais significativo" como raiz da (sub-)árvore
  1. Se não restam exemplos, então tal situação nunca foi observada e devolvemos um valor de defeito calculado a partir da classificação maioritária do nó pai.
  2. Se os exemplos são todos positivos ou todos negativos podemos terminar (podemos logo dizer *sim* ou *não*)
  3. Se não restam atributos e ainda temos exemplos positivos e negativos então temos um problema...  $\Rightarrow$  ruído nos dados (tipicamente devolve-se valor em maioria)
  4. Se há exemplos positivos e negativos escolher o melhor atributo para os separar

# Algoritmo DTL

## (*Decision Tree Learning*)

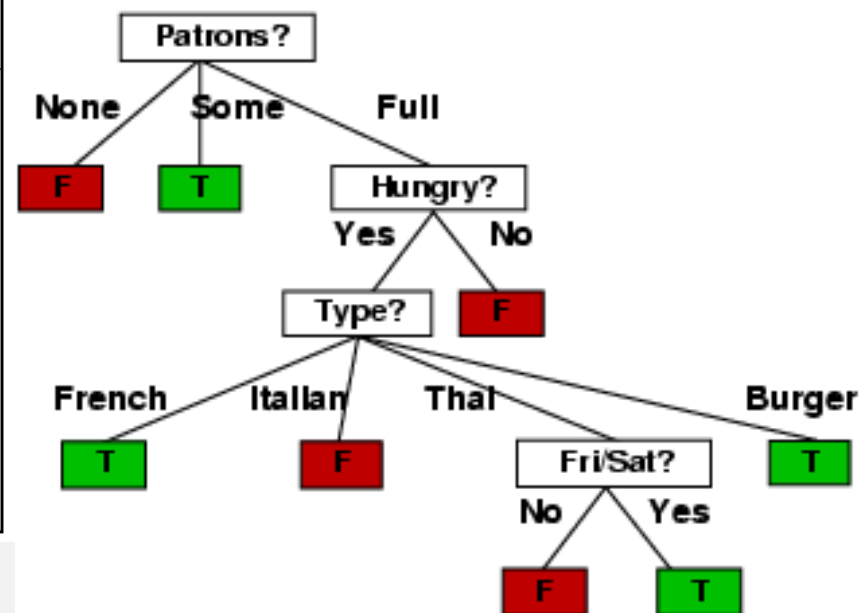
Função DTL (*exemplos, atributos, exp\_pais*) devolve árvore de decisão

- 1 se *exemplos* vazio então devolve VALOR-MAIORIA(*exp\_pais*)
  - 2 senão se todos os *exemplos* têm a mesma *classificação* então devolve *classificação*
  - 3 senão se *atributos* está vazio então devolve VALOR-MAIORIA(*exemplos*)
  - 4 senão
    - $A \leftarrow \operatorname{argmax}_{a \in \text{atributos}} \text{IMPORTÂNCIA}(a, \text{exemplos})$
    - árvore*  $\leftarrow$  nova árvore com a raiz teste A
    - para cada valor  $v_i$  de A
      - $\text{exs} \leftarrow \{e: e \in \text{exemplos} \text{ e } e.A = v_i\}$
      - sub-árvore*  $\leftarrow$  DTL(*exs*, *atributos* - A, *exemplos*)
      - adiciona um ramo à *árvore* com etiqueta ( $A = v_i$ ) e *sub-árvore*
- devolve *árvore*

# Indução de árvores de decisão com base em exemplos

- Árvore de decisão aprendida a partir dos 12 exemplos:

Example	Attributes										Target Wait
	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	
$X_1$	T	F	F	T	Some	\$\$\$	F	T	French	0-10	T
$X_2$	T	F	F	T	Full	\$	F	F	Thai	30-60	F
$X_3$	F	T	F	F	Some	\$	F	F	Burger	0-10	T
$X_4$	T	F	T	T	Full	\$	F	F	Thai	10-30	T
$X_5$	T	F	T	F	Full	\$\$\$	F	T	French	>60	F
$X_6$	F	T	F	T	Some	\$\$	T	T	Italian	0-10	T
$X_7$	F	T	F	F	None	\$	T	F	Burger	0-10	F
$X_8$	F	F	F	T	Some	\$\$	T	T	Thai	0-10	T
$X_9$	F	T	T	F	Full	\$	T	F	Burger	>60	F
$X_{10}$	T	T	T	T	Full	\$\$\$	F	T	Italian	10-30	F
$X_{11}$	F	F	F	F	None	\$	F	F	Thai	0-10	F
$X_{12}$	T	T	T	T	Full	\$	F	F	Burger	30-60	T



- Substancialmente mais compacta do que a árvore inicial

# Escolha de atributo

- IMPORTÂNCIA no algoritmo DTL
- Ideia: um bom atributo separa exemplos em subconjuntos que são (idealmente) “todos positivos” ou “todos negativos”



- *Cientes vs. Tipo? Cientes é melhor escolha!*

# Escolha de atributo

- Esta estratégia é a usada para implementar **IMPORTÂNCIA** no algoritmo DTL
- E como decidir se um atributo é bom ou não?
  - Uma boa medida seria medir a quantidade de informação dada pelo atributo
  - Em teoria de informação mede-se a quantidade de informação em **bits** sendo que a quantidade de informação depende do conhecimento pré-existente
    - E.g. um bit de informação é suficiente para responder a uma pergunta do tipo sim/não, sobre a qual não temos nenhuma ideia, como por exemplo o lançamento de uma moeda ao ar



# Teoria da Informação

- Entropia....

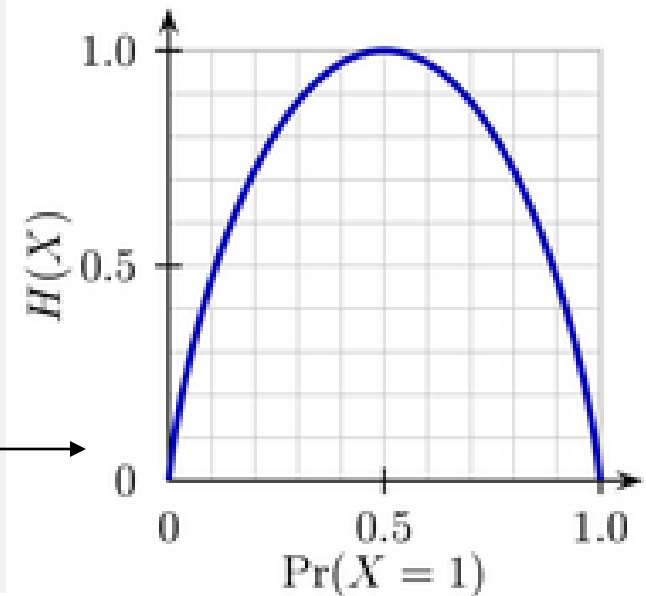
# Teoria da informação

- **Entropia** mede a quantidade de incerteza numa distribuição de **probabilidade**:

Considere-se o atirar ao ar uma moeda “tendenciosa”. Se atirmos ao ar a moeda muitas vezes, a frequência para *cara* é  $p$  e consequentemente a frequência para *coroa* é  $1-p$ . (Para uma moeda normal  $p=0.5$ )

A incerteza de um valor de output é dado pela **entropia**.

O valor da entropia é 0 se  $p=0$  ou  $p=1$  e é máximo ( $=1$ ) se  $p=0.5$ .



# Teoria da informação

- Se existem mais do que dois estados  $s=1,2,..n$  (por ex<sup>o</sup> se temos um dado) vem:

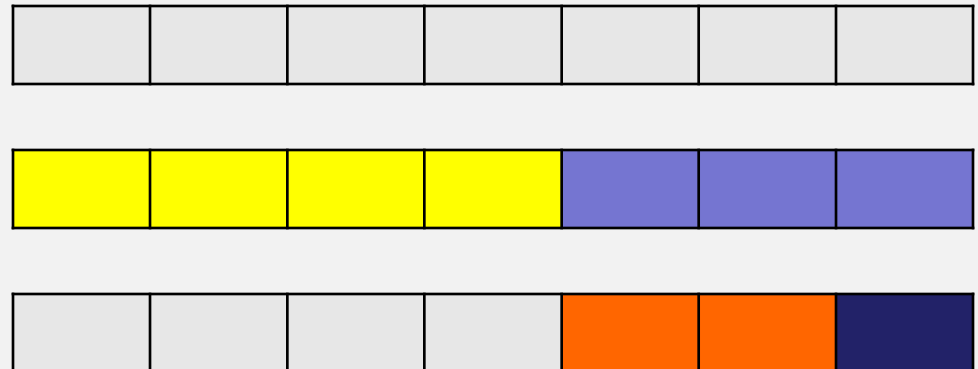
$$\begin{aligned} \text{Entropy}(p) = & -p(s=1)\log[p(s=1)] \\ & - p(s=2)\log[p(s=2)] \\ & \dots \\ & - p(s=n)\log[p(s=n)] \end{aligned}$$

$$\sum_{s=1}^n p(s) = 1$$

# Porquê log?

- Considere-se um conjunto de  $S$  elementos
  - $\log_2(|S|)$  corresponde ao número **mínimo** de vezes que temos de **dividir um conjunto ao meio** para garantidamente identificar **um elemento**
    - $\log_2(1) = 0$ ,
    - $\log_2(2) = 1$ ,
    - $\log_2(3) = 1$ ,
    - $\log_2(4) = 2$ ,
    - $\log_2(5) = 2$ ,
    - $\log_2(6) = 2$ ,
    - $\log_2(7) = 2$ ,
    - $\log_2(8) = 3, \dots$
  - Podemos considerar que o número de divisões corresponde ao número de questões que vamos ter de responder (testes da árvore)

## Exemplo para $S=7$



# Escolha de atributo

- Considere-se agora um conjunto de  $P+N$  elementos, sendo  $P/N$  os elementos classificados positivamente/negativamente, respectivamente. Temos que:
  - $\log_2(|P|)$  questões no caso do elemento pertencer a  $P$
  - $\log_2(|N|)$  questões no caso do elemento pertencer a  $N$
  - Número de questões:  $-P_P \log_2(P_P) - P_N \log_2(P_N)$  em que
    - $P_{P/N}$  é a probabilidade do elemento pertencer a  $P/N$ , respectivamente (ou seja, por exemplo, é  $P_P = P/(P+N)$ )
- **$I(P,N) = -P_P \log_2(P_P) - P_N \log_2(P_N)$**

# Escolha de atributo

- $I(P,N) = -P_P \log_2(P_P) - P_N \log_2(P_N)$

- Para o caso do restaurante:  $P=N=6$

$$I\left(\frac{6}{12}, \frac{6}{12}\right) = -\frac{6}{12} \log_2 \frac{6}{12} - \frac{6}{12} \log_2 \frac{6}{12} = 1$$

# Teoria da informação

- O que ganhamos em termos de informação se soubermos o valor de determinado atributo?
- Resposta:  
*incerteza antes de escolher o atributo*  
*menos incerteza depois de o escolher*

# Escolha de atributo

- Qualquer atributo  $A$  divide o conjunto de treino  $E$  em sub-conjuntos  $E_1, \dots, E_v$  de acordo com os valores para  $A$ , em que  $A$  tem  $v$  valores distintos
- Entropia depois da separação em função de  $A$  dada por:

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

- Ganho de informação (GI) ou redução de entropia para o atributo  $A$ :

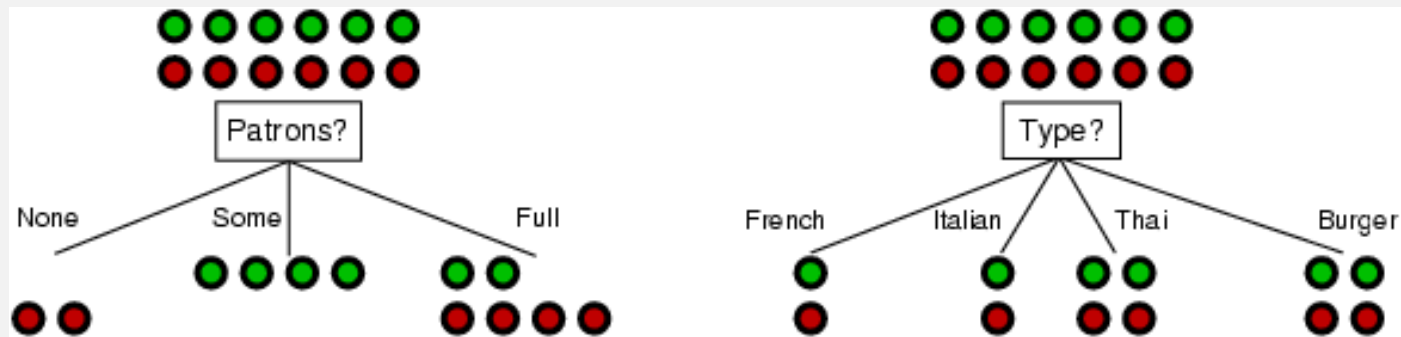
$$GI(A) = I\left(\frac{p}{p + n}, \frac{n}{p + n}\right) - resto(A)$$

- Escolher o atributo com o maior GI**



# Escolha de atributo

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



Considerar os atributos *Clientes* e *Tipo* (e todos os outros):

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

$$GI(Tipo) = 1 - \left[ \frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right) \right] = 0 \text{ bits}$$

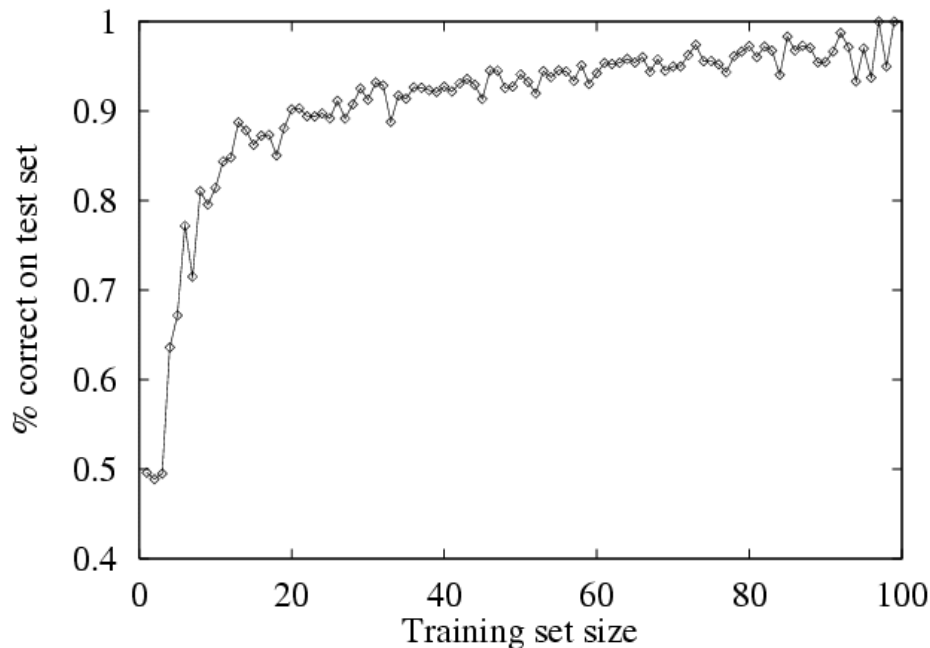
*Clientes* tem o GI mais elevado entre todos os atributos e portanto é escolhido pelo algoritmo DTL como raiz

# Avaliando a performance do algoritmo de aprendizagem (medida de desempenho)

- Como é que sabemos se  $h \approx f$ ?
  - Idealmente, verificando se a classificação prevista coincide com a classificação real para um **conjunto de teste** de exemplos
  - Na prática é adoptado um procedimento iterativo
    1. Obter um conjunto de exemplos
    2. Dividir o conjunto em 2 conjuntos disjuntos: conjunto de treino e conjunto de teste
    3. Aplicar o algoritmo de aprendizagem ao conjunto de treino, e obter uma hipótese  $h$
    4. Medir a percentagem de exemplos do conjunto de teste correctamente classificados por  $h$
    5. Repetir os passos 2-4 para diferentes conjuntos de treino de diferentes dimensões (obtidos aleatoriamente)

# Avaliando a performance do algoritmo de aprendizagem (medida de desempenho)

Curva de aprendizagem = %  $h \approx f$  no conjunto de teste em função da dimensão do conjunto de treino



# Ruído e Overfitting (Problemas com o DTL)

- Problema 1: se existem **dois ou mais exemplos com a mesma descrição** (em termos dos atributos), **mas com diferentes classificações**, o DTL não consegue encontrar uma árvore de decisão consistente com os dados
- Solução: fazer com que cada folha indique uma classificação por **maioria** para o seu conjunto de exemplos (no caso de ser necessário uma hipótese determinística) ou uma estimativa da **probabilidade** para cada classificação, usando frequências relativas

# Ruído e Overfitting (Problemas com o DTL)

- Problema 2: é possível que, mesmo quando **falta informação vital**, o DTL seja capaz de encontrar uma árvore de decisão consistente com todos os exemplos. Isto acontece quando o algoritmo usa **atributos irrelevantes** (caso existam) para fazer distinções erradas entre os exemplos
  - Exemplo: Considere-se a experiência de lançar um dado, várias vezes, sendo a experiência caracterizada pelos seguintes parâmetros:
    - Dia: dia em que foi lançado o dado (2<sup>a</sup>, 3<sup>a</sup>, 4<sup>a</sup>, 5<sup>a</sup>)
    - Mês: Mês em que foi lançado o dado (Janeiro ou Fevereiro)
    - Cor: cor do dado (azul ou vermelho)
- O DTL vai encontrar uma hipótese exacta mas incorrecta! Pretendia-se apenas uma árvore com um único nó e com 1/6 associado a cada face do dado...

# Ruído e Overfitting (Problemas com o DTL)

- Problema 3: Quando o **conjunto de hipóteses** consistentes com os exemplos **é muito grande**, é possível que se encontrem **regularidades sem sentido** nos dados. A este problema chama-se ***overfitting*** e é um problema geral nos algoritmos de aprendizagem.
- Solução:
  - Podar a árvore de decisão (decision tree pruning)
  - Validação cruzada (cross validation)

# Ruído e Overfitting (Problemas com o DTL)

- E em que consiste a poda da árvore de decisão?
  - Basicamente consiste em evitar a divisão em atributos que são irrelevantes

# Ruído e Overfitting (Problemas com o DTL)

- E em que consiste a validação cruzada?
  - Pode ser aplicado a qualquer algoritmo de aprendizagem e pode ser combinado com outras técnicas ou até usado na construção da árvore de decisão
  - A ideia básica é estimar quão bem uma hipótese vai estimar dados não vistos
    - Assim, considerando  $k$  iterações, parte dos dados são postos de lado ( $1/k$ ) e usados para testar a capacidade de previsão da hipótese induzida pelo resto dos dados.



# Alargando a aplicabilidade das árvores de decisão

- Problemas a ter em conta:
  - **Faltas de dados:** em muitos domínios, por vezes existem valores desconhecidos para os atributos, às vezes pode ser muito caro encontrar estes valores. Isto levanta dois problemas:
    - Dada uma árvore de decisão completa, como classificar um objecto do qual não se conhece algum valor dos atributos?
    - Como modificar a fórmula do ganho quando alguns exemplos têm valores desconhecidos para os atributos?
  - **Atributos com inúmeros valores:** nestes casos a medida usada para o ganho de informação dá uma indicação inapropriada da utilidade do atributo. Solução: usar o rácio do ganho.

# Alargando a aplicabilidade das árvores de decisão

- Problemas a ter em conta (cont):
  - **Atributos com domínios contínuos ou contínuos mas ilimitados:** é o caso de pesos ou alturas. Nestes casos não tem sentido ter um número infinito de ramos, mas encontrar um ponto de divisão dos valores que represente um maior ganho. Há muitos algoritmos eficientes a tratar este problema, mas isto representa um dos maiores custos destas aplicações a problemas reais
  - **Classificações não discretas:** se estamos a tentar prever um valor numérico, tal como o preço de uma obra de arte, mais do que uma classificação discreta precisamos é de uma árvore de regressão (*regression tree*). Esta árvore tem em cada folha uma função linear de um subconjunto de atributos numéricos, em vez de um único valor. O algoritmo tem de decidir quando deve parar para aplicar a regressão linear sobre os atributos restantes.
- Aplicações reais têm de considerar estes aspectos

- Formas de Aprendizagem
- Aprendizagem indutiva
- Aprendizagem com árvores de decisão
- **Teoria de Aprendizagem Computacional**
- Sumário

# Teoria de aprendizagem computacional

- Como é que se garante que um algoritmo de aprendizagem vai ser capaz de fazer previsões correctas?

# Teoria de aprendizagem computacional

- Abordagem da teoria de aprendizagem computacional (ramo que relaciona IA com estatística e ciências de computação):
  - Qualquer hipótese que está seriamente errada será encontrada com grande probabilidade depois de um pequeno número de exemplos, pois faz previsões incorrectas.
  - Assim, qualquer hipótese que é consistente com um **número significativo de exemplos** não poderá estar significativamente errada, ou seja, deve estar **aproximadamente correcta**

# Teoria de aprendizagem computacional

- Nota:
  - Qualquer algoritmo de aprendizagem que devolve hipóteses **provavelmente correctas** diz-se um algoritmo de aprendizagem PAC (*probably approximately correct*)
- A ter em conta:
  - **Relação entre o treino e os testes**
    - Assunção de estacionariedade: o treino e os testes são retirados aleatoria e independentemente da mesma população de exemplos com a mesma distribuição de probabilidades.
  - **Quantidade de exemplos**
    - É possível calcular um N em função do espaço de estados
      - Problema: se o espaço de espaços for muito grande
        - » Ou o algoritmo não aprende (não se consegue obter N)
        - » Ou se restringe o espaço de estados (só que podemos estar a deitar fora a verdadeira função)

- Formas de Aprendizagem
- Aprendizagem indutiva
- Aprendizagem com árvores de decisão
- Teoria de Aprendizagem Computacional
- **Sumário**

# Sumário

- Aprendizagem necessária para ambientes desconhecidos, desenho de agentes incompleto
- Agente aprendiz = desempenho + aprendizagem
- Para aprendizagem supervisionada, o objectivo é encontrar uma hipótese simples aproximadamente consistente com os exemplos de treino
- Aprendizagem com árvores de decisão usa ganho de informação
- Desempenho de aprendizagem = precisão da previsão medida com um conjunto de teste