# Aprender a partir de exemplos

Capítulo 18, secções 1 – 3

Capítulo 19, secção 1

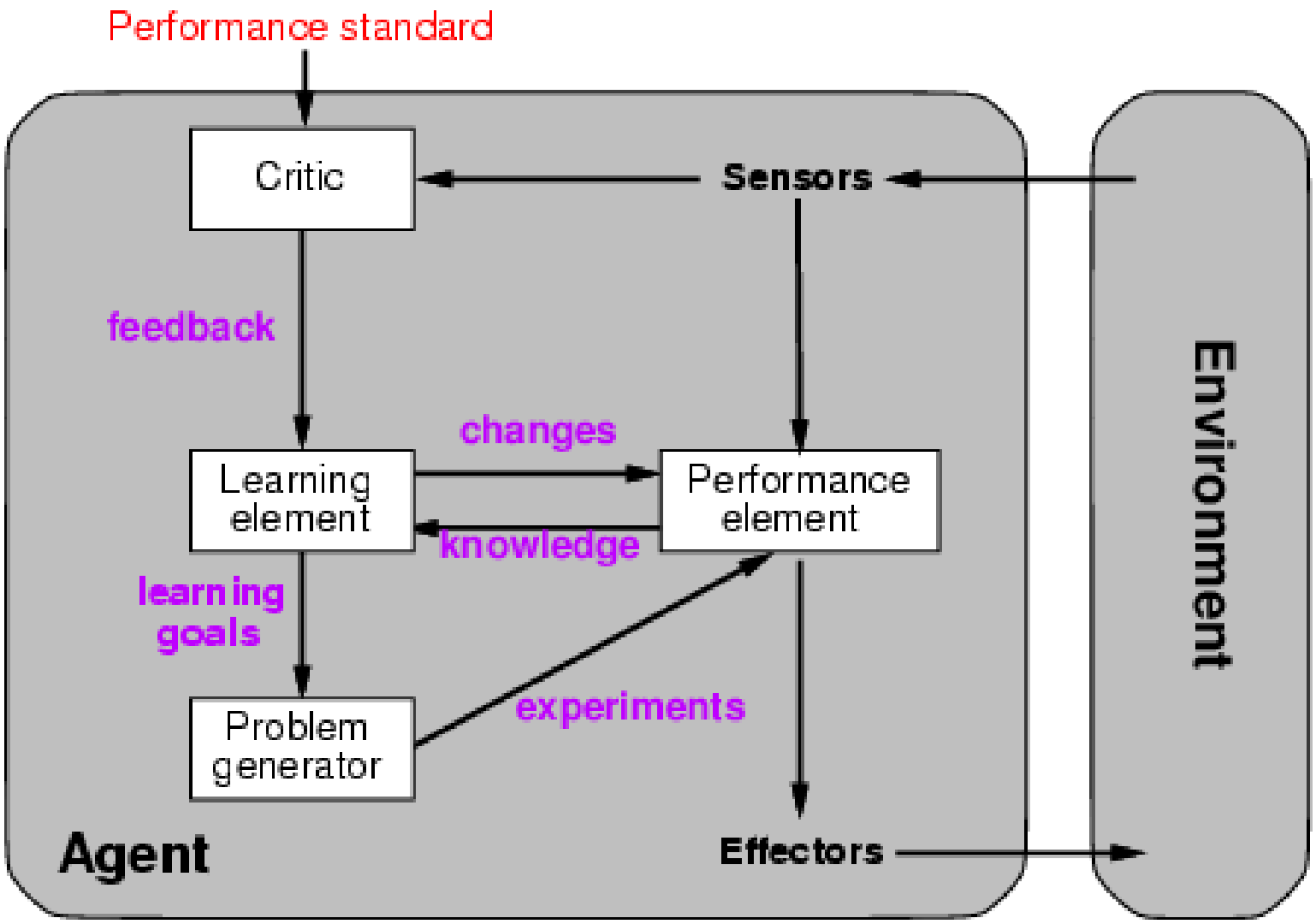
# Resumo

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

# Aprendizagem

- A aprendizagem é essencial em ambientes desconhecidos,
  - i.e., quando o projectista do sistema não possui omnisciência
- A aprendizagem é útil como um método de construção de sistemas,
  - i.e., expõe o agente à realidade em vez de tentar descrevê-lo
- A aprendizagem modifica os mecanismos de decisão do agente visando o aumento de desempenho

# Agentes aprendizes



# Elemento de aprendizagem

- O desenho do elemento de aprendizagem é ditado por
  - Quais os componentes do elemento de desempenho que devem ser aprendidos
  - Qual o feedback disponível para aprender esses componentes
  - Qual é a representação utilizada pelos componentes
- Tipo de feedback:
  - Aprendizagem Supervisionada: respostas correctas para cada exemplo
  - Aprendizagem Não Supervisionada : respostas correctas não são dadas
  - Aprendizagem por reforço: recompensas ocasionais

# Tipos de Aprendizagem

- Aprendizagem Supervisionada:
  - Aprendizagem de uma função a partir de exemplos de valores de entrada e respectivos resultados (respostas correctas para cada instância). Necessita de “professor”.
- Aprendizagem Não Supervisionada:
  - Descobrir padrões nos valores de entrada sem que sejam especificados os valores de saída (exemplos de treino não estão classificados, por exemplo).
- Aprendizagem por Reforço:
  - Aprendizagem for reforço, através de recompensas ocasionais. É a forma mais geral de aprendizagem.

# Aprendizagem Indutiva

Aprender uma função a partir de exemplos dados (supervisionada)

- *f* é uma função alvo

- Um exemplo é um par (*x*, *f(x)*)
-

Problema da Indução: encontrar uma hipótese *h* no *espaço de hipóteses*

tal que $h \approx f$

dado um conjunto de treino de exemplos

(encontrar uma hipótese h que generalize bem, ou seja, que prediga correctamente exemplos desconhecidos)

NOTA: *h* diz-se consistente quando concorda com *f* em todos os exemplos

Modelo muito simplificado da aprendizagem humana:

- Ignora conhecimento prévio
- Assume que os exemplos são fornecidos

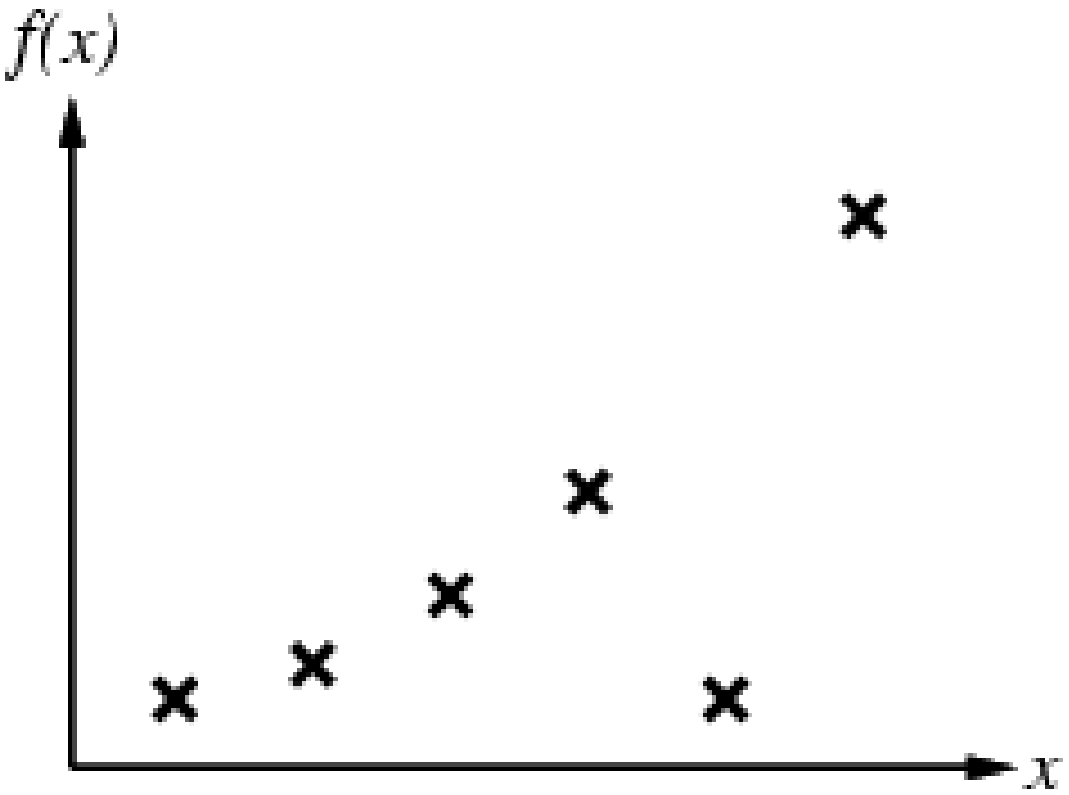
# Aprendizagem Indutiva

- Se a função alvo que estivermos a aprender for discreta estamos na presença de um problema de classificação.
- Em particular, se estivermos interessados em funções booleanas dizemos que estamos na presença de um problema de aprendizagem conceptual.
- O caso de aprendizagem de funções contínuas é designado por regressão (interpolação quando se impõe consistência)
- Todos eles assumem a Hipótese da Aprendizagem Indutiva

  *Uma hipótese h que aproxime bem a função alvo num conjunto de treino suficientemente grande, também aproximará bem a função alvo num conjunto de exemplos não observados previamente.*

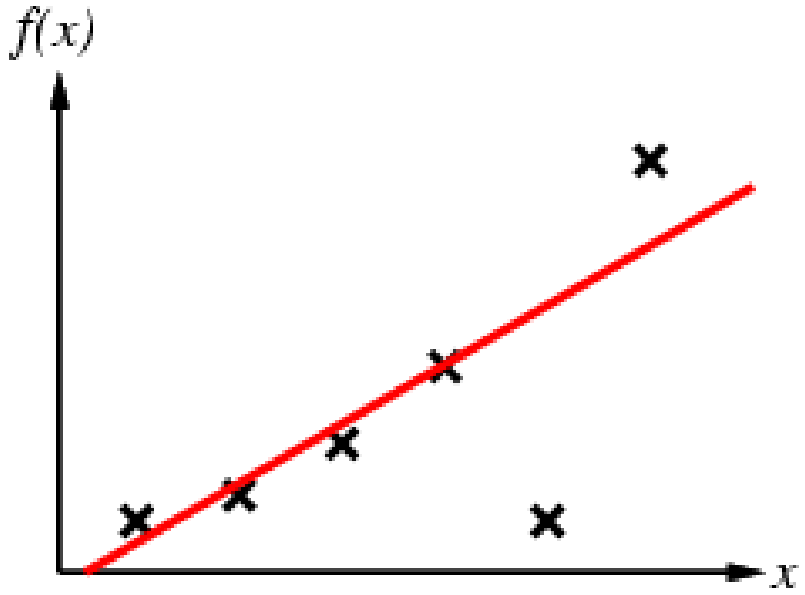
# Método da Aprendizagem Indutiva

- Construir/ajustar *h* tal que concorde com *f* no conjunto de treino
- E.g., regressão:
-



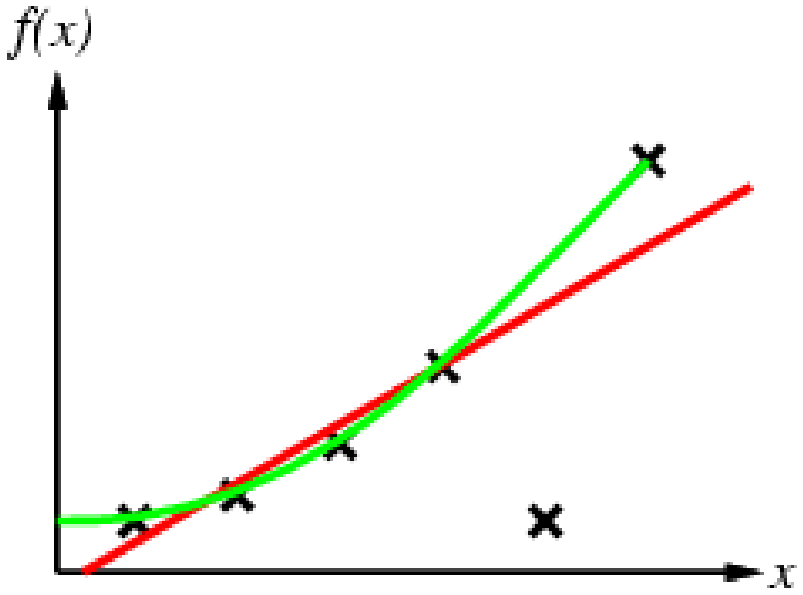
# Método da Aprendizagem Indutiva

- Construir/ajustar *h* tal que concorde com *f* no conjunto de treino
- E.g., regressão:



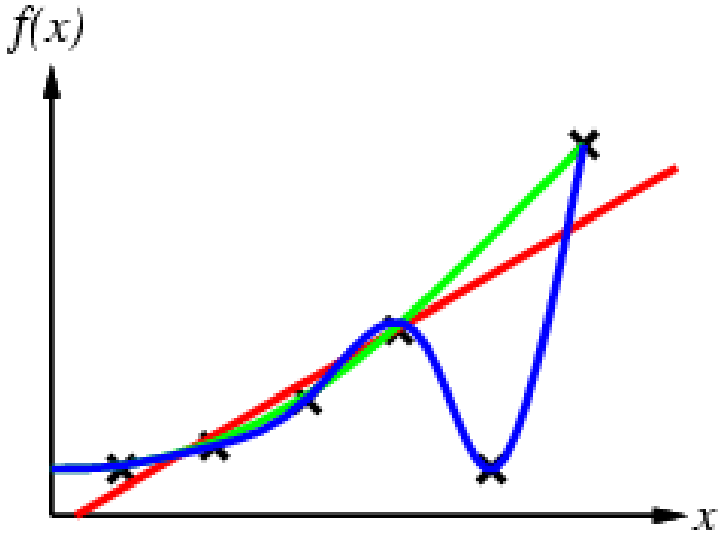
# Método da Aprendizagem Indutiva

- Construir/ajustar *h* tal que concorde com *f* no conjunto de treino
- E.g., regressão:
-



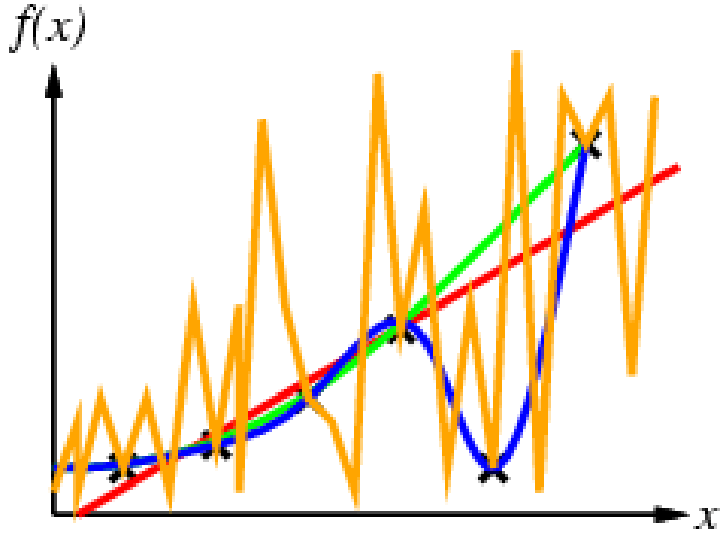
# Método da Aprendizagem Indutiva

- Construir/ajustar *h* tal que concorde com *f* no conjunto de treino
- E.g., regressão:



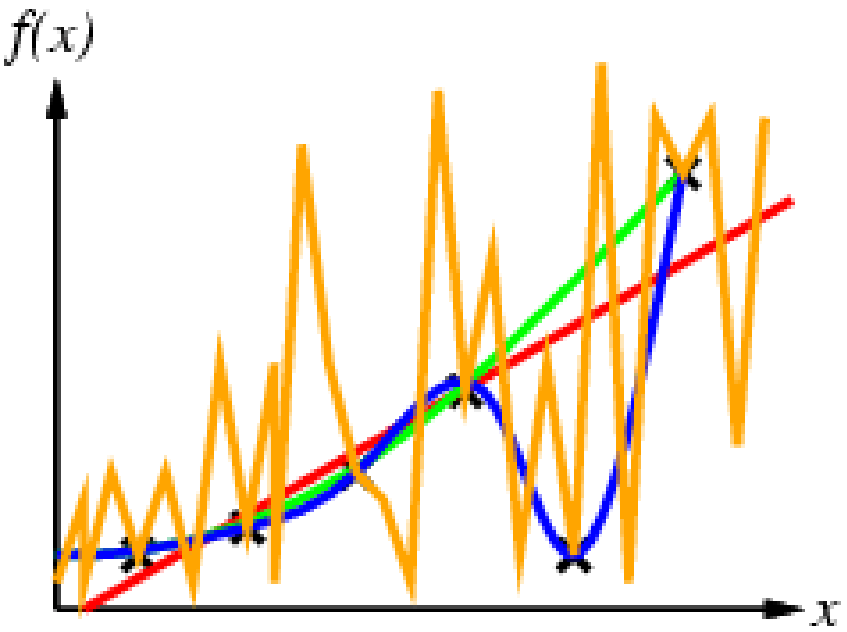
# Método da Aprendizagem Indutiva

- Construir/ajustar *h* tal que concorde com *f* no conjunto de treino
- E.g., regressão:



# Método da Aprendizagem Indutiva

- Construir/ajustar *h* tal que concorde com *f* no conjunto de treino
- E.g., regressão:




- Navalha de Ockham: preferir a hipótese mais simples consistente com os dados

# Aprendizagem Conceptual

- Dados
  - Instâncias X descritas por atributos
  - Espaço de hipóteses H
  - Conceito alvo c: X -> {0,1}
  - Exemplos de treino D positivos e negativos

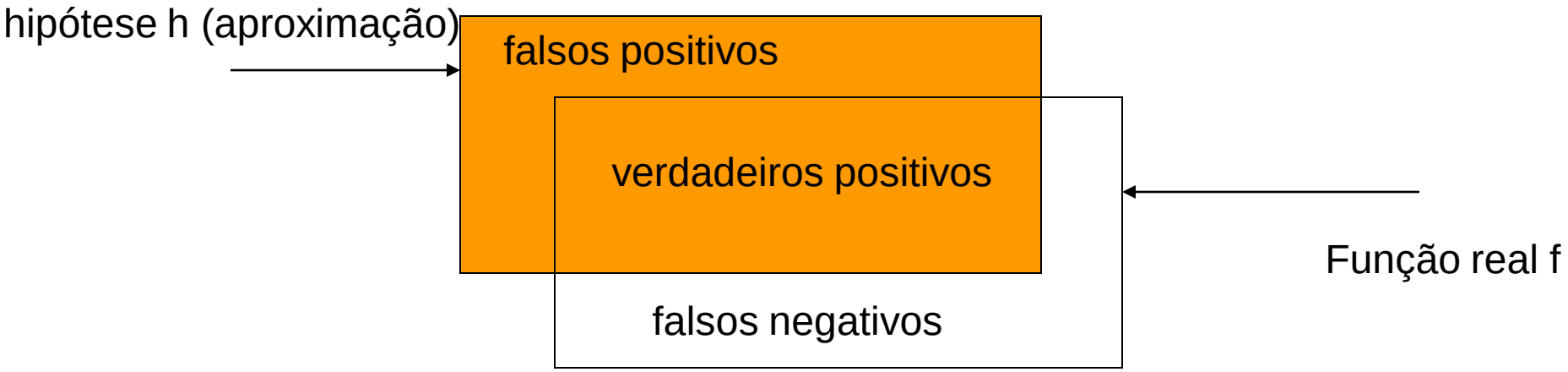
- Determinar

  Uma hipótese *h* no espaço de hipóteses H tal que *h(x)* = *c(x)* para todo o *x*.

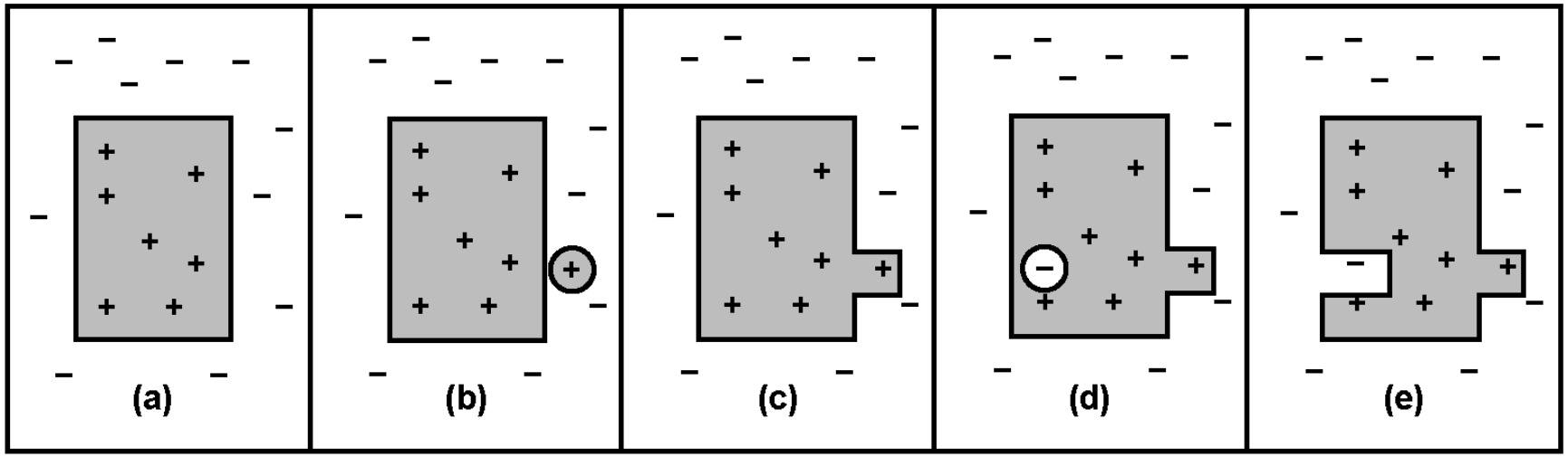
# Exemplo (T. Mitchell)

| Sky | AirTemp | Humidity | Wind | Water | Forecast | Sport? |
|---|---|---|---|---|---|---|
| Sunny | Warm | Normal | Strong | Warm | Same | 1 |
| Sunny | Warm | High | Strong | Warm | Same | 1 |
| Rainy | Cold | High | Strong | Warm | Change | 0 |
| Sunny | Warm | High | Strong | Cool | Change | 1 |

# Falsos positivos e negativos



# Tratar falsos positivos e negativos




Hipótese Consistente

Falso Negativo

Hipótese Consistente

Falso Positivo

Hipótese Consistente

Generalização da Hipótese

Especialização da Hipótese

# Espaço de Hipóteses

- Comecemos por considerar hipótese conjuntivas com restrições simples nos valores de atributos:
  - '?' qualquer valor possível para o atributo
  - '*valor*' só *valor* é possível para o atributo
  - '_' nenhum valor é aceitável

Exemplo:

h1=<?,Cold,High,?,?,?>

h2=<?,?,?,?,?,?>

h3=<_,_,_,_,_,_>

# Satisfação de hipóteses

- Uma instância x satisfaz uma hipótese h sse h(x) = 1

Exemplo

A hipótese h=<Sunny,?,High,?,?,?> é satisfeita pela instância

<Sunny,Warm,High,Strong,Cool,Same>

e não é satisfeita pela instância

<Sunny,Warm,Normal,Strong,Cool,Same>

# Ordenação parcial entre hipóteses

- Uma hipótese hi é mais geral do que uma hipótese hj sse todas as instâncias que satisfazem hj também satisfazem hi (hi $\geq$ hj)
- Esta ordem parcial é induzida pela ordem parcial entre restrições simples nos valores do atributo

Exemplo

h1=<Sunny,?,?,?,?,?>

h2=<Sunny,?,High,?,?,?>

h3=<Rainy,?,High,?,?,?>

Logo h1 $\geq$ h2 (h1 é mais geral do que h2)

Nota: duas hipóteses podem ser incomparáveis (h1 e h3)

# Espaço de Hipóteses

- Dado um conjunto de hipóteses qualquer, dizemos que h é maximamente específica, se não existir outra hipótese h’ tal que $h \geq h'$

- Dado um conjunto de hipóteses qualquer, dizemos que h é maximamente geral, se não existir outra hipótese h’ tal que $h' \geq h$

# Eliminação de Candidatos

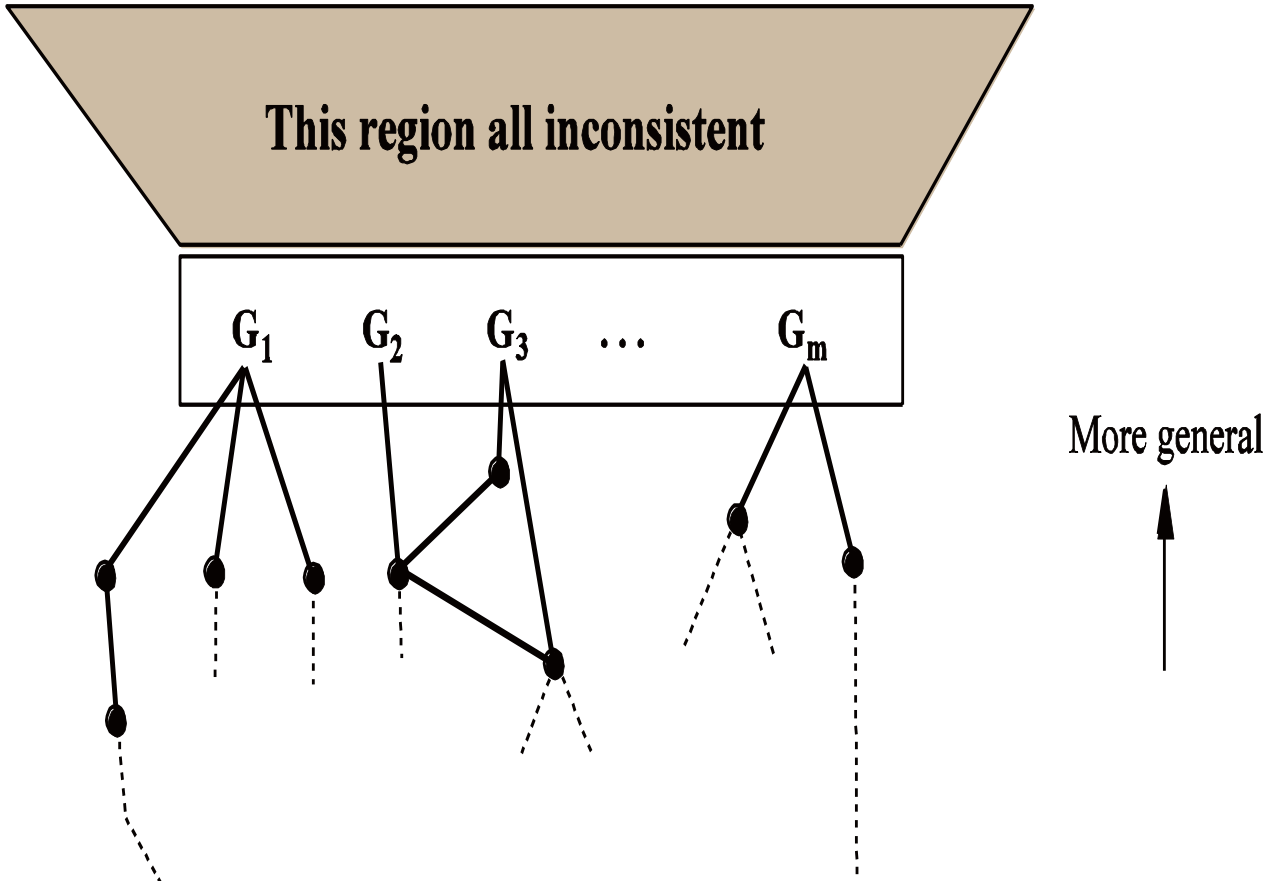
- Algoritmo proposto por Mitchell que efectua a procura no espaço de versões (Version Space - VS), o conjunto de todas as hipóteses consistentes com os exemplos apresentados.

- Uma hipótese *h* é consistente com um conjunto de treino *D* sse *h(x)=c(x)* para todo o exemplo de treino *<x,c(x)>* em *D*.

- O algoritmo funciona mantendo duas fronteiras
  - **G**: a fronteira mais geral das hipóteses consistentes com os exemplos D
  - **S**: a fronteira mais específica das hipóteses consistentes com os exemplos D

- Algoritmo utilizado no sistema META-DENDRAL para detectar regularidades em dados de espectroscopia de massa. Os resultados obtidos foram publicados numa revista de Química.

# O Espaço de Versões (VS)



# Evolução das fronteiras

Seja *Si* uma hipótese em *S* e um novo exemplo

- Se for um falso positivo para *Si*, então eliminar *Si* de *S* pois não se pode especializar *Si*.
- Se for um falso negativo para *Si* então substituímo-la pelas suas generalizações imediatas, desde que mais específicas do que algum elemento de *G*

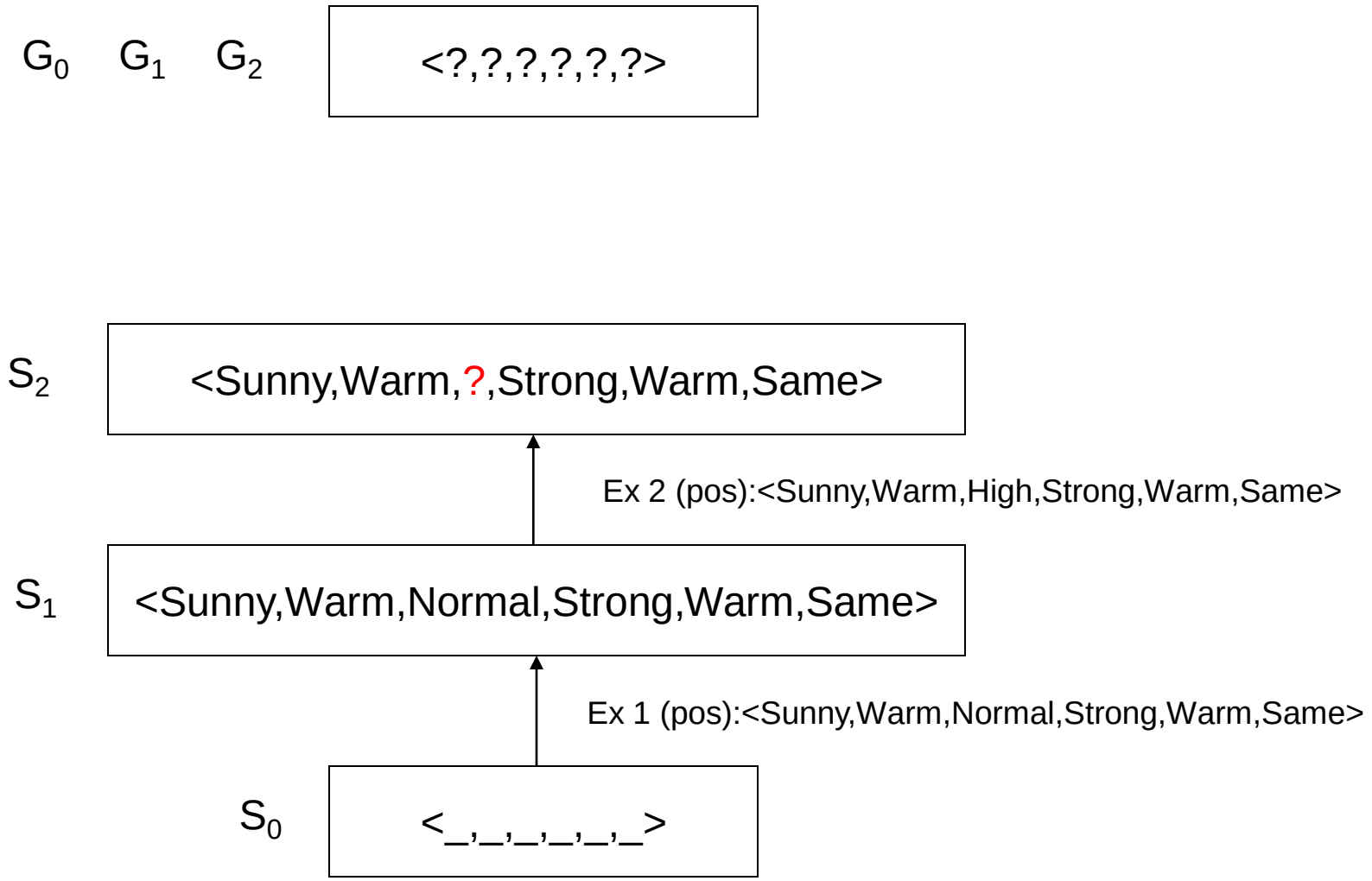
Seja *Gi* uma hipótese em *G* e um novo exemplo

- Se for um falso negativo para *Gi*, então eliminar *Gi* de *G* pois não se pode generalizar *Gi*.
- Se for um falso positivo para *Gi* então substituímo-la pelas suas especializações imediatas, desde que mais gerais do que algum elemento de *S*

# Algoritmo de Eliminação de Candidatos

- S contém a(s) hipóteses mais específicas de H
- G contém a(s) hipóteses mais gerais de H
- Para cada exemplo de treino d, fazer
  - Se d é um exemplo positivo
    - Remover de G qualquer hipótese inconsistente com d
    - Para cada hipótese s em S que não é consistente com d
      - Retirar s de S
      - Adicionar a S todas as generalizações minimais h de s tal que h é consistente com d, e algum membro de G é mais geral do que h
      - Retirar de S qualquer hipótese que seja mais geral do que outra hipótese em S
  - Se d é um exemplo negativo
    - Remover de S qualquer hipótese inconsistente com d
    - Para cada hipótese g em G que não é consistente com d
      - Retirar g de G
      - Adicionar a G todas as especializações minimais h de g tal que h é consistente com d e algum membro de S é mais específico do que h
      - Retirar de G qualquer hipótese que seja menos geral do que outra hipótese em G

# Exemplo


$G_0$ $G_1$ $G_2$
<?,?,?,?,?,?>
$S_2$
<Sunny,Warm,?,Strong,Warm,Same>
Ex 2 (pos):<Sunny,Warm,High,Strong,Warm,Same>
$S_1$
<Sunny,Warm,Normal,Strong,Warm,Same>
Ex 1 (pos):<Sunny,Warm,Normal,Strong,Warm,Same>
$S_0$
<_,_,_,_,_,_>

# Exemplo (cont)

$G_2$ <?,?,?,?,?,?>

Ex 3 (neg):<Rainy,Cold,High,Strong,Warm,Change>

$G_3$ <Sunny,?,?,?,?,?> <?,Warm,?,?,?,?> <?,?,?,?,?,Same>

~~<Cloudy,?,?,?,?,?> <?,?,Normal,?,?,?> <?,?,?,Weak,?,?> <?,?,?,?,Cool,?>~~

$S_2$ $S_3$ <Sunny,Warm,?,Strong,Warm,Same>

# Exemplo (cont)

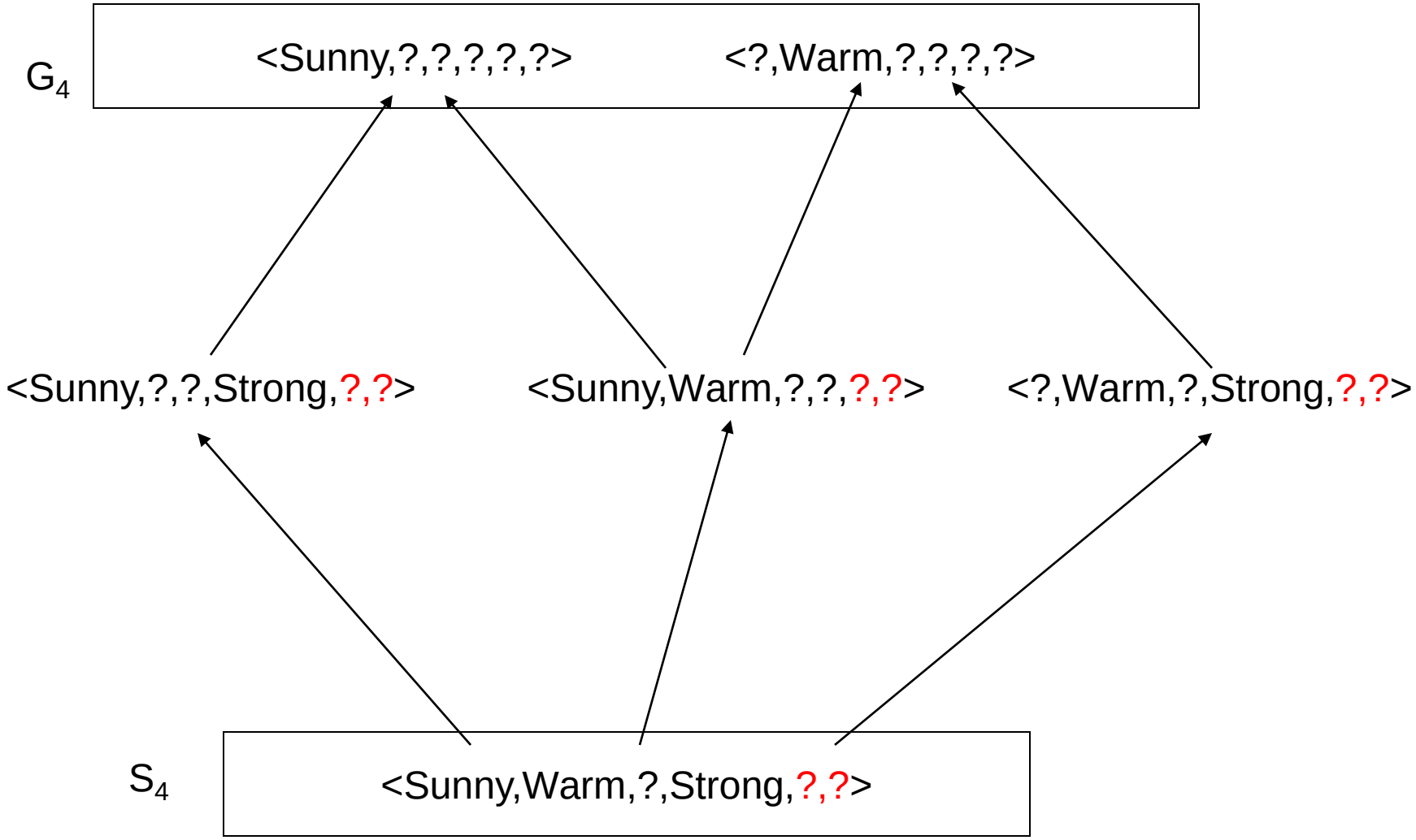
$G_3$ $G_4$ <Sunny,?,?,?,?,?> <?,Warm,?,?,?,?> ~~<?,?,?,?,?,Same>~~

$S_4$ <Sunny,Warm,?,Strong,?,?>

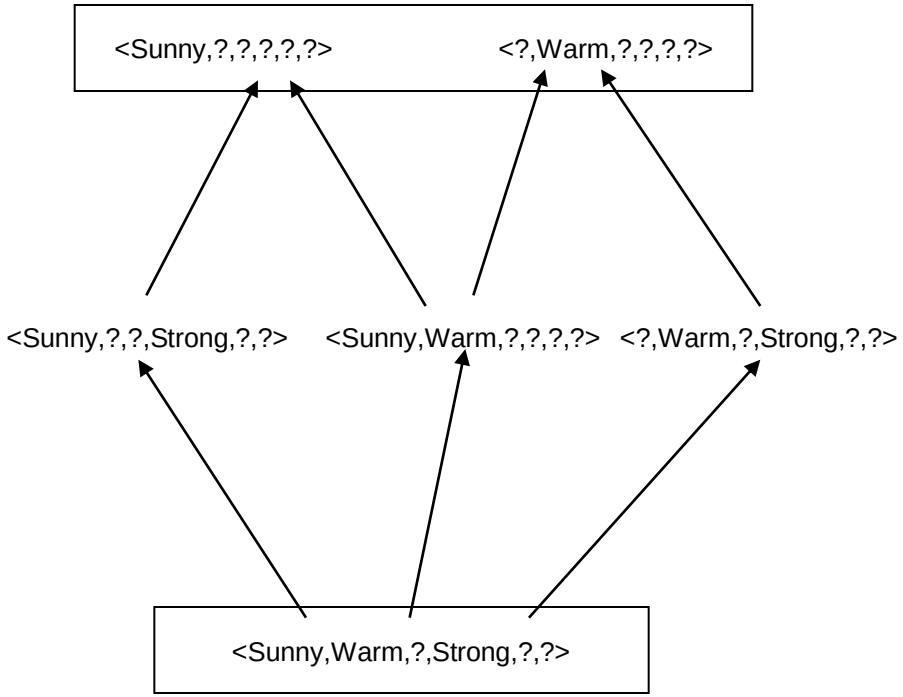
Ex 4 (pos):<Sunny,Warm,High,Strong,Cool,Change>

$S_3$ <Sunny,Warm,?,Strong,Warm,Same>

# Espaço de Versões Completo



# Classificação de novas instâncias




| | |
|---|---|
| <Sunny,Warm,Normal,Strong,Cool,Change> | Positivo (classificado + por todas as hipóteses) |
| <Rainy,Cold,Normal,Weak,Warm,Same> | Negativo (classificado - por todas as hipóteses) |
| <Sunny,Warm,Normal,Weak,Warm,Same> | Indefinido (classificado + por 1/2 das hipóteses) |
| <Sunny,Cold,Normal,Strong,Warm,Same> | Negativo? (classificado - por 2/3 das hipóteses) |

# Propriedades do algoritmo

- O algoritmo de eliminação de candidatos é incremental
- Efectua o menor compromisso (tal como no POP)
- O algoritmo converge para o conceito alvo pretendido se forem dados exemplos de treino suficientes (pelo menos $\log_2 |VS|$)

- Se existir ruído ou o domínio não contiver atributos suficientes para a classificação exacta, o espaço de versões colapsa (um dos conjuntos S ou G fica vazio)
- O algoritmo com o espaço de hipóteses apresentado não permite aprender conceitos disjuntivos.
- Caso seja permitida disjunção ilimitada no espaço de hipóteses, então o algoritmo só conseguirá classificar os exemplos dados (não generaliza).
- Para alguns espaços de hipóteses o número de elementos de S e de G podem crescer exponencialmente.

# 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

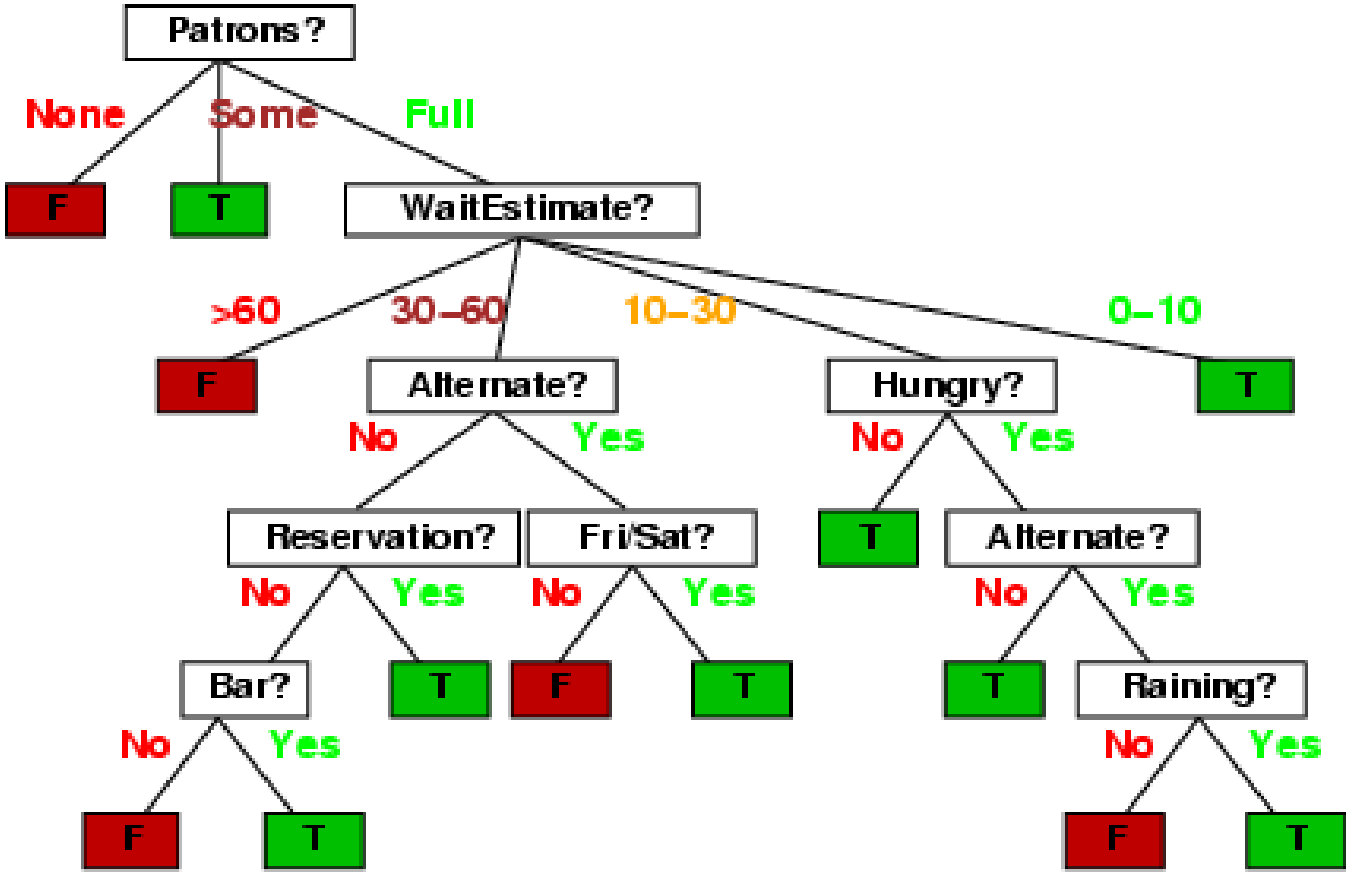
- Exemplos descritos por valores de atributos (Booleanos, discretos, contínuos)
- E.g., situações em que espero/não espero por uma mesa:

| Example | Attributes | | | | | | | | | | Target |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Alt* | *Bar* | *Fri* | *Hun* | *Pat* | *Price* | *Rain* | *Res* | *Type* | *Est* | *Wait* |
| $X_1$ | 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 |

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

# Árvores de Decisão

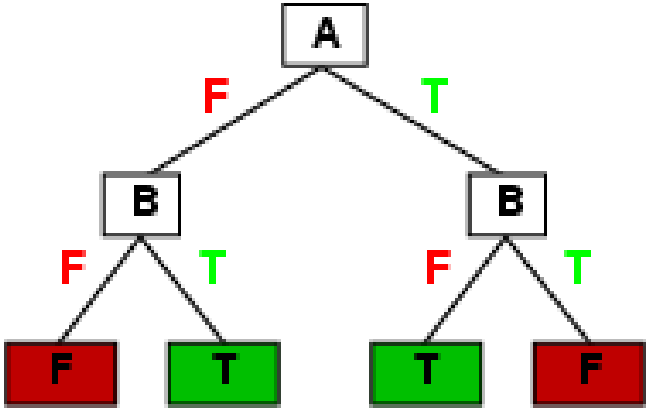
- Uma possível representação para a hipótese
- Abaixo é a árvore de decisão “real” para decidir se se espera:



# Expressividade

- Árvores de decisão podem expressar qualquer função dos seus atributos
- 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

- Preferir encontrar árvores de decisão mais compactas

# Espaço de Hipóteses

Quantas árvores de decisão com *n* atributos Booleanos existem?

$\geq$ número de funções Booleanas

$\geq$ 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* $\wedge \neg$*Rain*)?

- Cada atributo pode ocorrer positivamente, negativamente, ou de fora
  $\Rightarrow 3^n$ 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
    $\Rightarrow$ pode resultar em predições piores!

# 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

# Indução de Árvores de Decisão

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

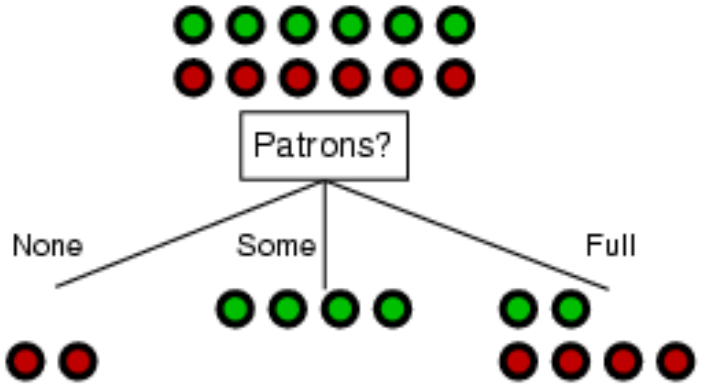
- Ideia: escolher (recursivamente) o atributo "mais significativo" como raiz da sub(árvore)

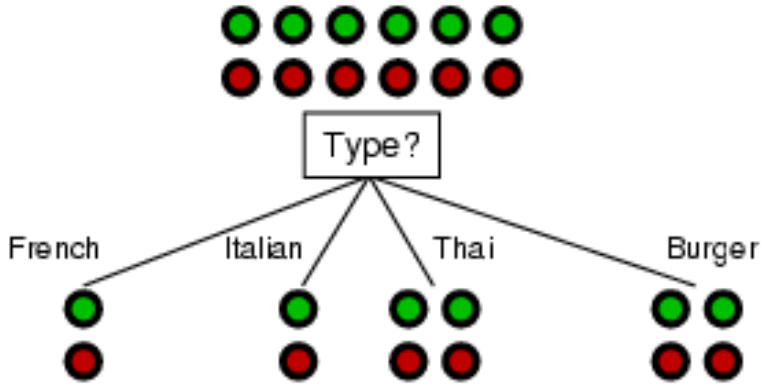
- Casos a tratar:
  1. Se existirem exemplos positivos e negativos, escolher o atributo mais significativo.
  2. Se todos os exemplos remanescentes são positivos ou negativos, devolvemos Yes/No, respectivamente.
  3. Se não existirem mais valores, devolve-se um valor por omissão (o que tiver mais ocorrências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




- *Patrons?* É a melhor escolha?
- 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?

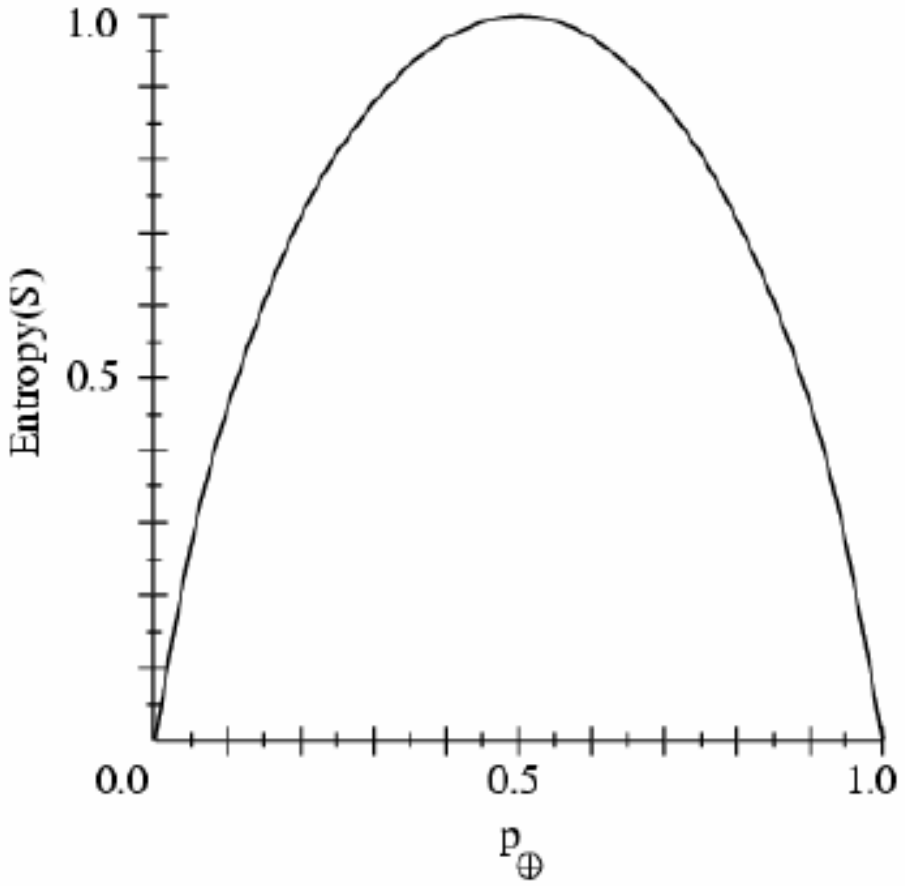
- Conteúdo de Informação (Entropia):

$$I(P(v_1), \ldots, P(v_n)) = \Sigma_{i=1} -P(v_i) \log_2 P(v_i)$$

- Para um conjunto de treino contendo *p* exemplos positivos e *n* exemplos negativos:

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

# Curva de entropia (binária)




[Tom M. Mitchell – Machine Learning]

# Ganho de Informação

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

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

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

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

- Escolher o atributo com maior IG

# Ganho de Informação

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

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

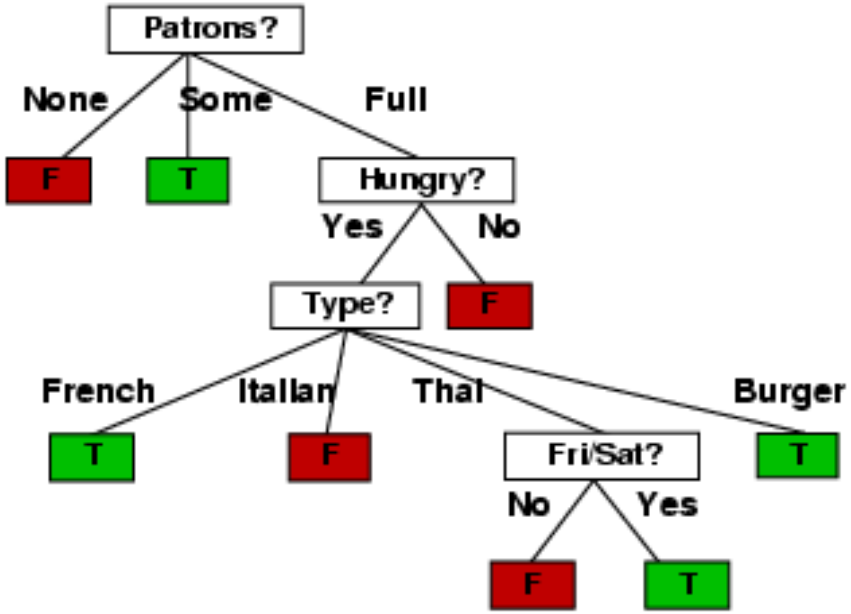
$$IG(Patrons) = 1 - [\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I(\frac{2}{6}, \frac{4}{6})] = .0541 \text{ bits}$$

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

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

# Exemplo (continuação)

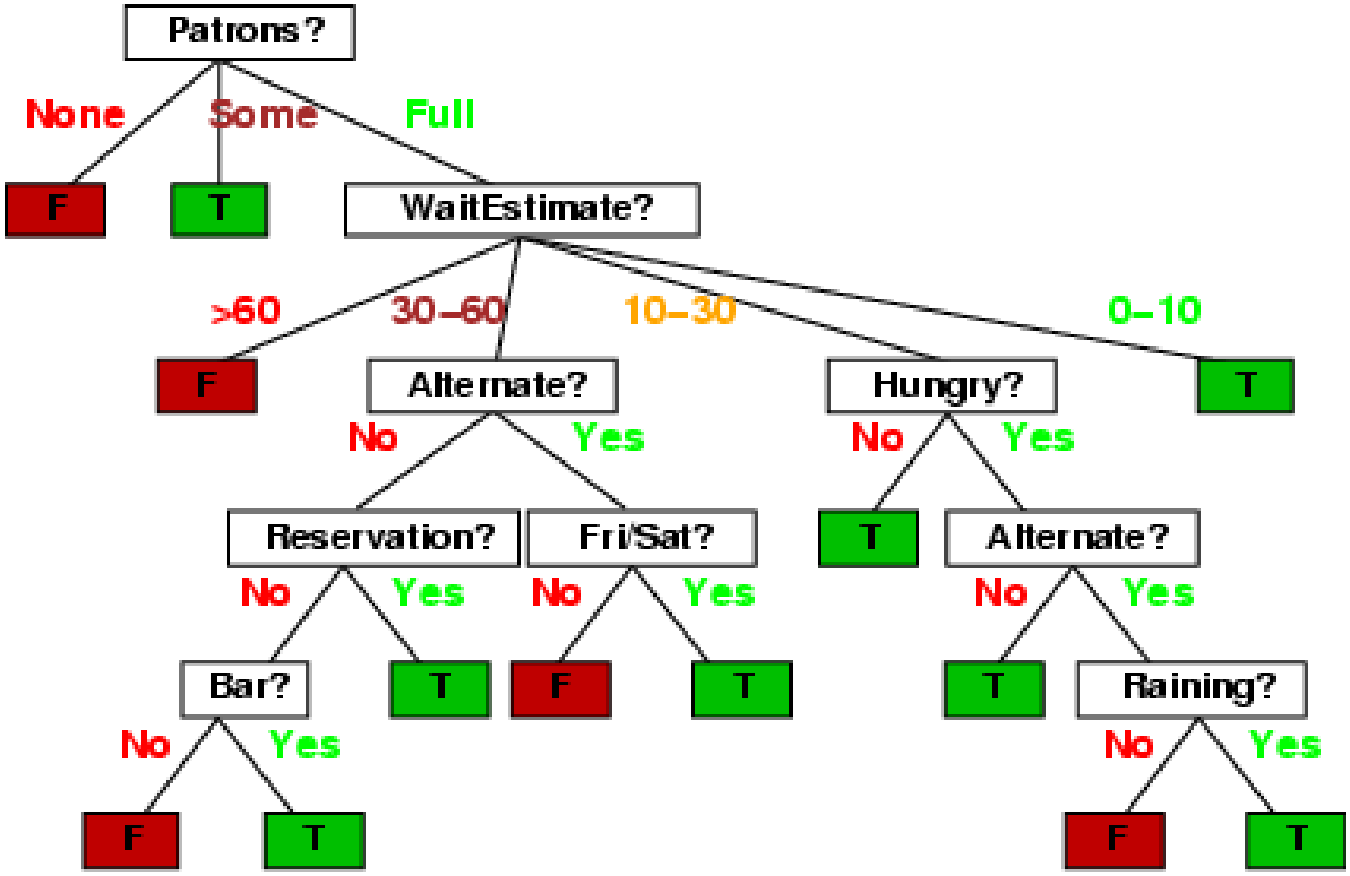
- Árvore de Decisão aprendida a partir dos 12 exemplos:




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

# Árvores de Decisão

- Uma possível representação para a hipótese
- Abaixo é a árvore de decisão “real” para decidir se se espera:

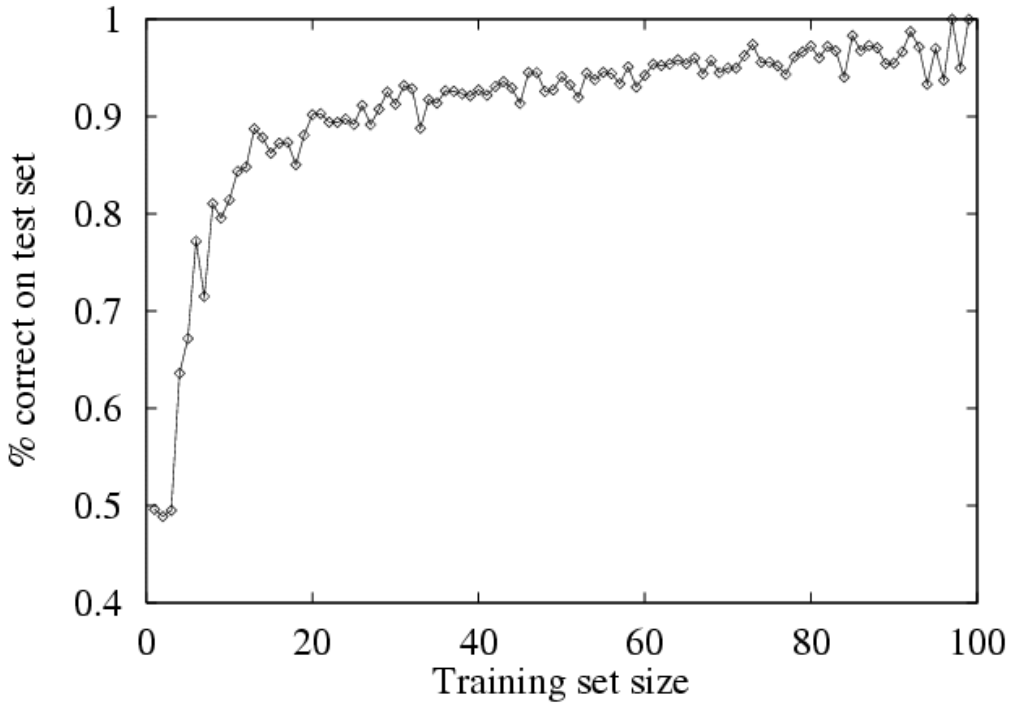


# Medida de desempenho

Como é que sabemos que $h \approx f$ ?

1. Usar teoremas da teoria da aprendizagem computacional/estatística
2. Aplicar *h* num novo conjunto de exemplos de teste

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



# Aspectos Práticos de Utilização

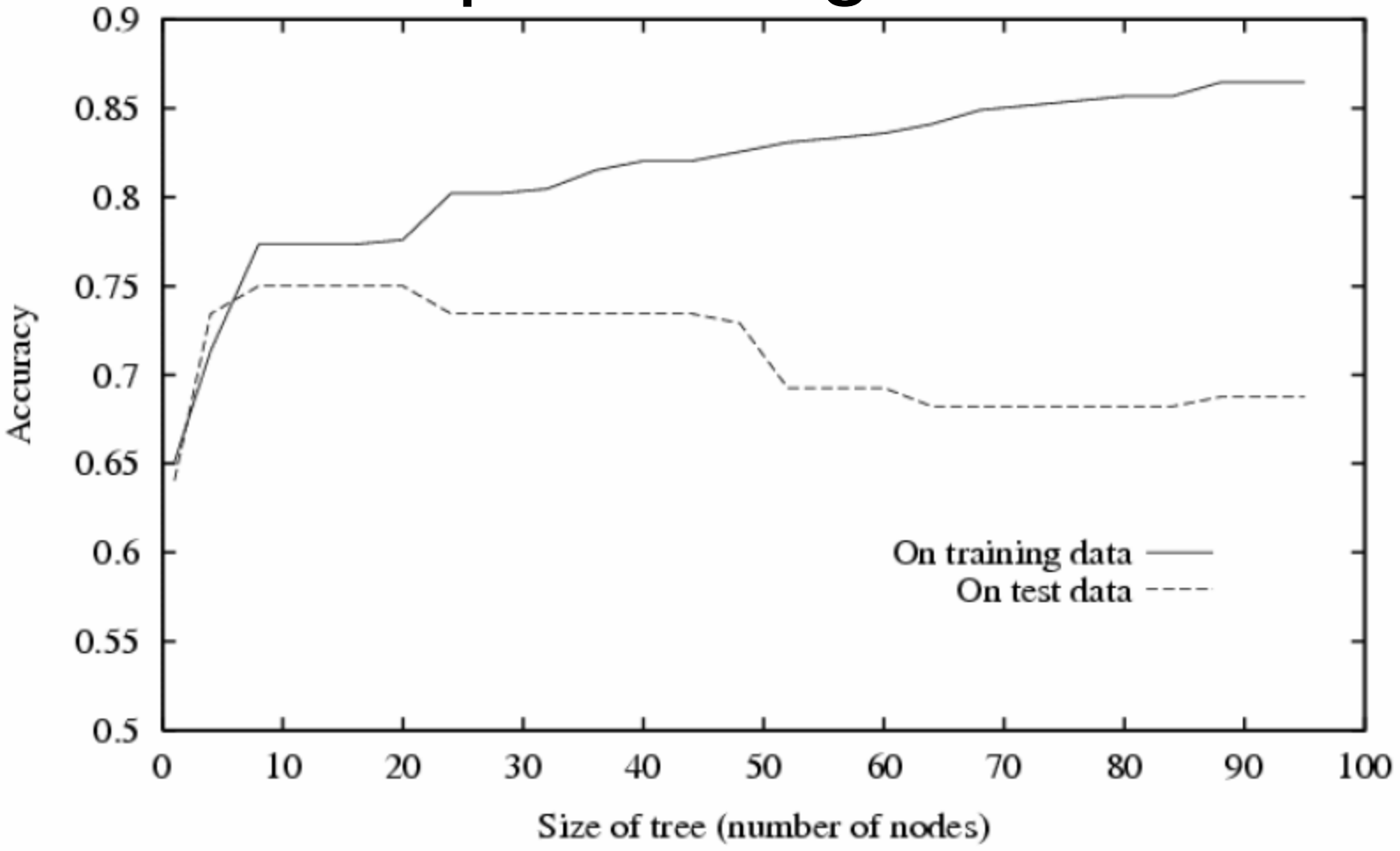
- Se existirem muitos atributos irrelevantes ou ruído é mais fácil encontrar uma hipótese exacta.
- Essa hipótese é totalmente espúria
- Neste caso estamos na presença de sobreajustamento na árvore de decisão
- Existem algoritmos mais sofisticados para lidar com estes casos (e.g. C4.5)

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




[Tom M. Mitchell – Machine Learning]

# 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…)

# 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 ?
- Como tratar atributos de saída contínuos ?

# Sumário

- Aprendizagem essencial para lidar com ambientes desconhecidos
- Agente aprendiz = elemento de desempenho + elemento de aprendizagem.
- No caso da aprendizagem indutiva, o objectivo consiste em encontrar uma hipótese simples que é aproximadamente consistente com os exemplos de treino.
- Aprendizagem conceptual é um caso particular de aprendizagem indutiva onde se pretende aprender uma função booleana a partir de exemplos dados.
- O algoritmo de eliminação de candidatos mantém as fronteiras de hipóteses maximamente específicas e maximamente gerais.
- Aprendizagem de árvores de decisão utiliza o ganho de informação
- Desempenho do algoritmo de aprendizagem = precisão no(s) conjunto(s) de teste.