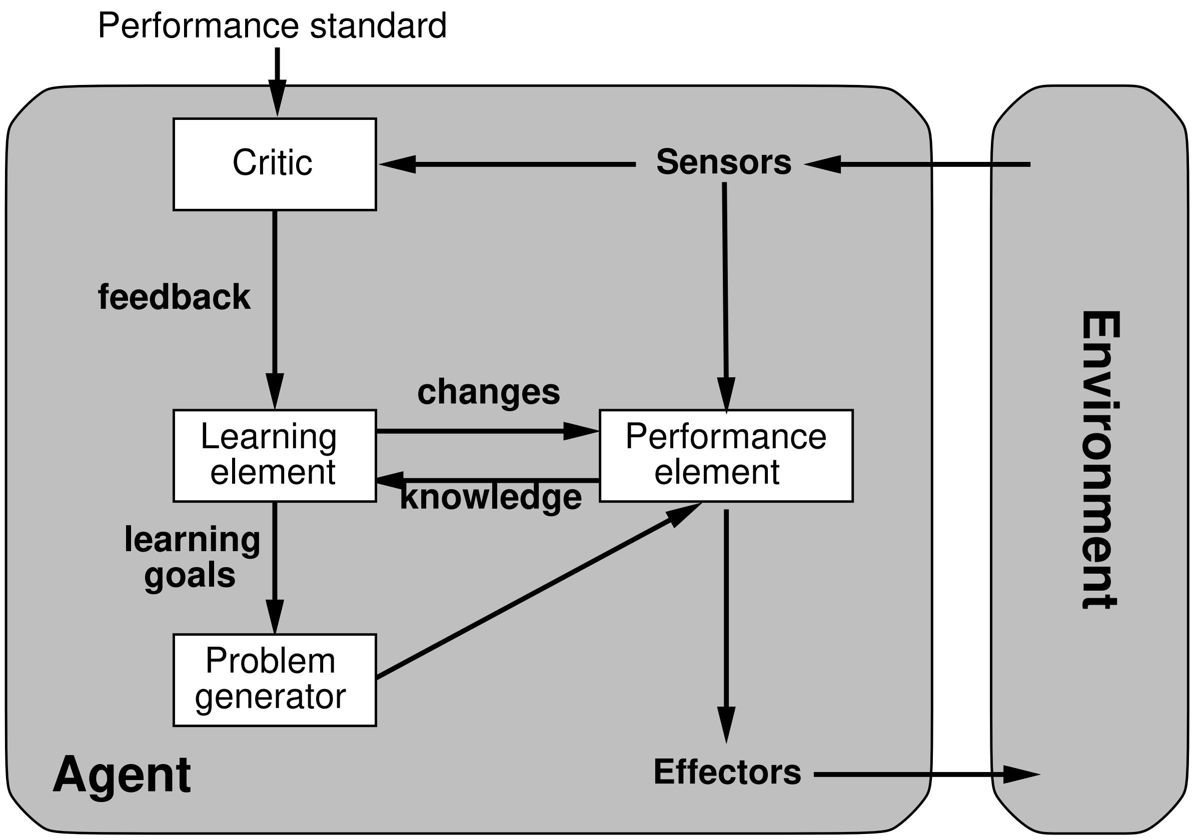
Cap. 18: Aprendendo através de observações


Performance standard
Critic
Sensors
feedback
changes
Learning element
Performance element
knowledge
learning goals
Problem generator
Effectors
Agent
Environment

Aprendendo através de observações

- Projeto de um elemento de performance é influenciado por 4 fatores:
  - que componentes devem ser melhorados.
  - que representação é usada para os componentes.
  - qual tipo de retro-alimentação está disponível.
  - que informação anterior é conhecida.

Aprendendo através de observações

- Componentes de um elemento de performance:
  - mapeamento direto de condições do estado corrente para as ações.
  - meios de inferir propriedades relevantes do ambiente através das percepções.
  - info sobre modificações no ambiente
  - info sobre os resultados de possíveis ações.
  - info de utilidade
  - info sobre valores de ações (prioridades) que indiquem o interesse naquela determinada ação em determinado estado.
  - Objetivos que descrevem classes de estados que maximizem a utilidade.

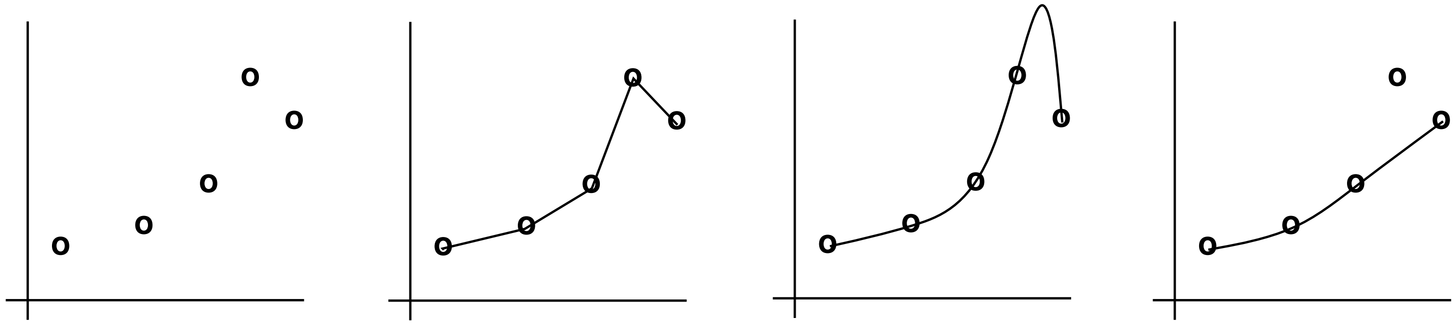
Aprendendo através de observações

- Representação dos componentes: pode ser feita utilizando qq esquema estudado.
- Retro-alimentação:
  - **aprendizado supervisionado:** entradas e saídas dos componentes são conhecidas. Agente pode prever qual será a saída.
  - **reinforcement**: agente recebe algum avaliação de sua ação, mas não conhece a ação correta.
  - **aprendizado não supervisionado**: qdo não há nenhuma dica sobre as saídas.
- Conhecimento anterior: necessário para melhorar aprendizado.

Aprendizado Indutivo (Inductive Learning)

- Em aprendizado supervisionado o elemento de aprendizado tem o valor correto ou aproximado da função das entradas.
- Modifica a representação da função para que esta se torne análoga à info fornecida por retro-alimentação.
- **Exemplo**: par $(x, f(x))$, onde x é entrada é f(x) é saída.
- **Inferência puramente indutiva** (ou simplesmente indução): dado um conj de exemplos de f, retorna uma função h (**hipótese**) que aproxima f.
- **Bias**: preferência por uma ou outra hipótese.

Aprendizado Indutivo (Inductive Learning)



(a) (b) (c) (d)

```
global examples ← {}
--------------------------------------------------------------------
function REFLEX-PERFORMANCE-ELEMENT( percept) returns an action

  if ( percept, a) in examples then return a
  else
     h ← INDUCE(examples)
     return h( percept)
--------------------------------------------------------------------
procedure REFLEX-LEARNING-ELEMENT(percept, action)
  inputs: percept, feedback percept
          action, feedback action

  examples ← examples ∪ {( percept,action)}
```

Aprendizado Indutivo (Inductive Learning)

- Algoritmo atualiza var global *examples*, lista de pares *percepção, ação*.
- Percepção pode ser uma situação no jogo de xadrez.
- Ação: melhor jogada de acordo com um grande mestre enxadrista.
- Se o agente percebe uma situação q já tenha visto antes, executa a ação correspondente.
- Caso contrário, utiliza o algoritmo de aprendizado INDUCE sobre exemplos que viu até então.
- INDUCE retorna uma hipótese h q é usada para escolher uma ação.

Aprendizado Indutivo (Inductive Learning)

- Alternativa: **aprendizado incremental**. Agente tenta atualizar a hipótese anterior sempre q um novo exemplo aparece, sem precisar induzir sobre *todos* os exemplos a cada nova previsão.
- Pode tb receber retro-alimentação sobre a qualidade das ações escolhidas.
- Forma em que hipóteses são representadas: livre.
- Algoritmos de aprendizado: mais uma vez, lógica!
- Duas abordagens para aprender sentenças lógicas: **árvores de decisão** e **version-space** (mais geral, menos eficiente).
- Problema: representação da função utilizada para o aprendizado. É "representável" na linguagem? É eficiente (tratável)?

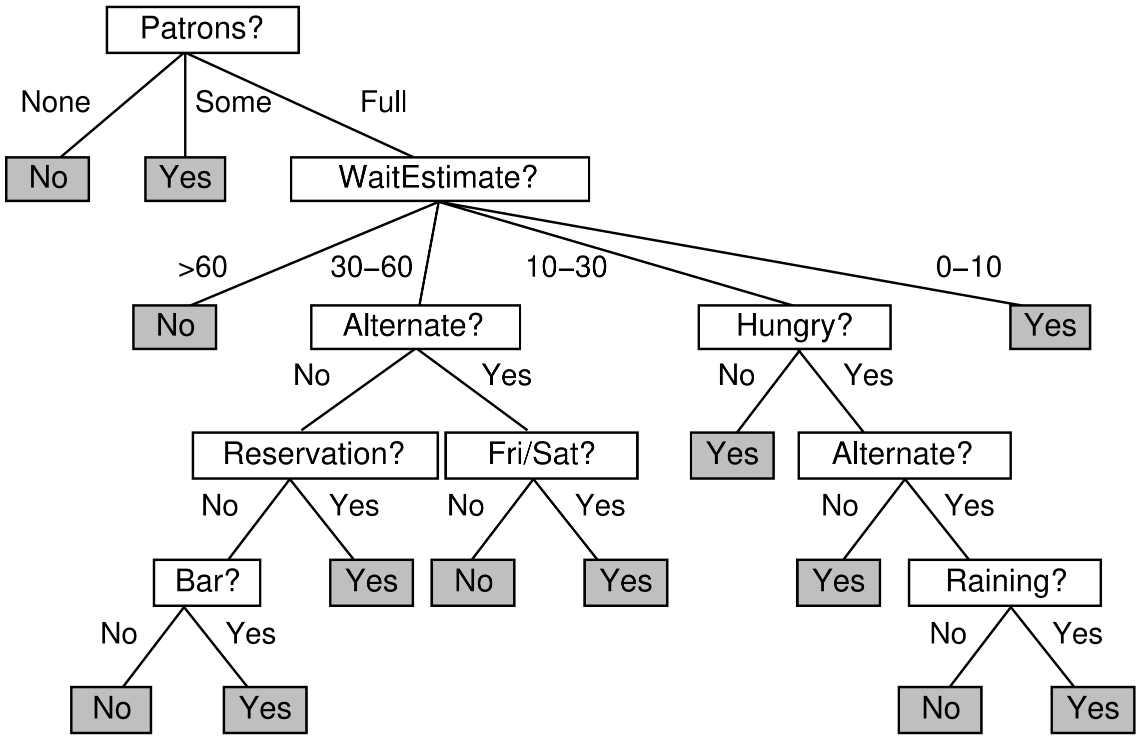
Aprendizado Indutivo (Inductive Learning)

- Lógica de primeira ordem: tempo de computação e número de exemplos necessários para aprender um "bom" conjunto de sentenças.
- "Bom" conj de sentenças: prevê corretamente experências futuras e reflete corretamente experiências passadas.
- Problema filosófico: como é que podemos saber se um algoritmo de aprendizado está produzindo uma teoria q prevê corretamente o futuro?

Árvores de Decisão

- Simples e fácil de implementar.
- Recebe como entrada um objeto ou situação descrita por um conj de propriedades e produz uma resposta "sim" ou "não". Representam funções booleanas.
- Exemplo: esperar por uma mesa num restaurante.
- Objetivo: aprender a definição do predicado "VouEsperar", com a definição expressa por uma árvore de decisão.
- Decidimos (p eqto, pois isto poderia ser decidido pelo algoritmo de aprendizado) as propriedades ou atributos:
  - Alternativo: algum restaurante alternativo perto?
  - Bar: restaurante tem uma área de espera?
  - Sex/Sab: V se for sexta ou sábado.
  - ComFome: estamos com fome?

- Clientes: número de pessoas no restaurante (Nenhuma, Algumas, Cheio).
- Preço: \$, \$\$, \$\$\$.
- Chovendo: chovendo do lado de fora.
- Reserva: temos reserva?
- Tipo: Francês, Italiano etc.
- EsperaEstimada: 0–10min, 10–30, 30–60, > 60.


Patrons?
None
Some
Full
No
Yes
WaitEstimate?
>60
30–60
10–30
0–10
No
Alternate?
Hungry?
Yes
No
Yes
No
Yes
Reservation?
Fri/Sat?
Yes
Alternate?
No
Yes
No
Yes
No
Yes
Bar?
Yes
No
Yes
Yes
Raining?
No
Yes
No
Yes
No
Yes
No
Yes

Árvores de Decisão

- Em lógica: $\forall r Clientes(r, Cheio) \wedge EsperaEstimada(r, 10-30) \wedge \neg ComFome(r, N) \Rightarrow VouEsperar(r)$
- Árvores de decisão não conseguem representar testes sobre 2 ou mais objetos diferentes.
- Limitações em representação.
- Qq função booleana pode ser representada como uma árvore de decisão.
- No entanto, representação em árvores de decisão devem ser + compactas, pq tabelas verdade têm crescimento exponencial.

Árvores de Decisão

- **Exemplos**: valores dos atributos + valor do predicado desejado.
- **Classificação** do exemplo: valor do predicado.
- qdo valor é verdadeiro, exemplo **positivo**. Caso contrário, é um exemplo **negativo**.
- conj completo de exemplos: **conjunto de treinamento**.

Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	*Yes*
X_2	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	*No*
X_3	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	*Yes*
X_4	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*10–30*	*Yes*
X_5	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	*No*
X_6	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	*Yes*
X_7	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	*No*
X_8	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	*Yes*
X_9	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	*No*
X_{10}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	*No*
X_{11}	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	*No*
X_{12}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	*Yes*

Árvores de Decisão

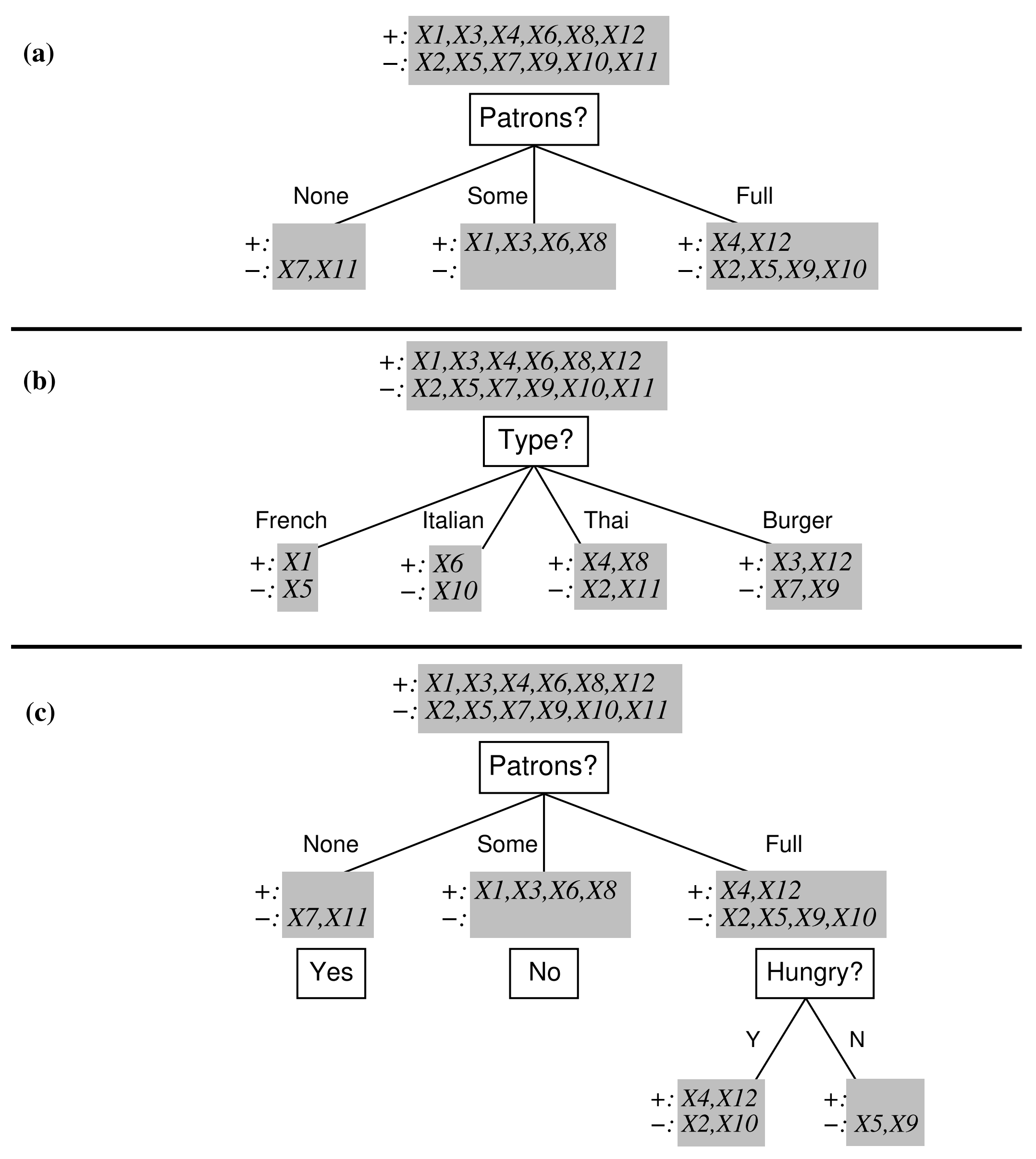
- Como induzir uma árvore de decisão através de exemplos?
- Cada exemplo pode ser um caminho diferente na árvore.
- limitação de representação, não deixa extrair nenhum outro padrão de informação além daquele descrito pelos exemplos já conhecidos.
- Extrair um padrão significa descrever um grande número de casos de forma concisa.
- Princípio geral de aprendizado indutivo: **Ockham's razor**. "A hipótese mais provável é a mais simples e consistente com todas as observações".
- encontrar a menor árvore de decisão é um problema intratável.
- heurísticas podem ajudar.

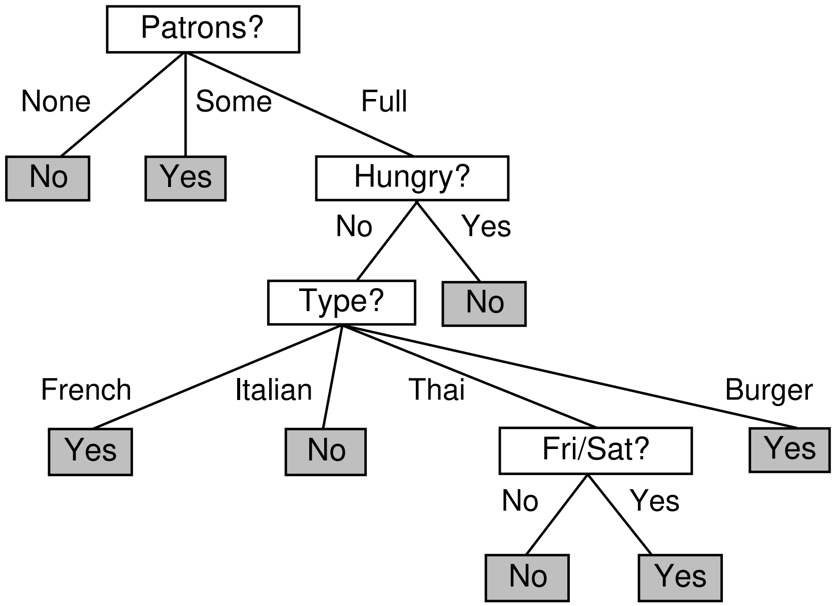
Árvores de Decisão

- idéia básica do algoritmo: testar os atributos "mais importantes" primeiro.
- O que é um atributo "mais importante"? É aquele que influencia mais a classificação do exemplo.
- Exemplo: 12 conjuntos de treinamento, separados em exemplos positivos e negativos.
- *Clientes* é um atributo importante: se valor igual a Nenhuma ou Algumas, o predicado sempre tem valor definido (Não e Sim).
- *Tipo*: atributo pobre.
- algoritmo escolhe atributo + forte e coloca como raiz da árvore.

Árvores de Decisão

- Restam subconjuntos de exemplos, algoritmo é aplicado recursivamente. 4 casos possíveis:
  - Se há alguns exemplos positivos e negativos, escolher o melhor atributo.
  - Se todos os exemplos restantes são positivos (ou todos negativos), podemos responder diretamente Sim ou Não.
  - Se não há mais exemplos, significa que nenhum exemplo foi observado para aquele caminho. Retorna valor default Sim ou Não dependendo da maioria das classificações do pai.
  - Se não há mais atributos, mas temos exemplos positivos e negativos, significa que estes exemplos têm exatamente a mesma descrição, mas diferentes classificações. Solução simples: voto majoritário.


(a)
+: X1,X3,X4,X6,X8,X12
−: X2,X5,X7,X9,X10,X11
Patrons?
None
Some
Full
+:
−: X7,X11
+: X1,X3,X6,X8
−:
+: X4,X12
−: X2,X5,X9,X10
(b)
+: X1,X3,X4,X6,X8,X12
−: X2,X5,X7,X9,X10,X11
Type?
French
Italian
Thai
Burger
+: X1
−: X5
+: X6
−: X10
+: X4,X8
−: X2,X11
+: X3,X12
−: X7,X9
(c)
+: X1,X3,X4,X6,X8,X12
−: X2,X5,X7,X9,X10,X11
Patrons?
None
Some
Full
+:
−: X7,X11
+: X1,X3,X6,X8
−:
+: X4,X12
−: X2,X5,X9,X10
Yes
No
Hungry?
Y
N
+: X4,X12
−: X2,X10
+:
−: X5,X9


Patrons?
None
Some
Full
No
Yes
Hungry?
No
Yes
Type?
No
French
Italian
Thai
Burger
Yes
No
Fri/Sat?
Yes
No
Yes
No
Yes

Árvores de Decisão

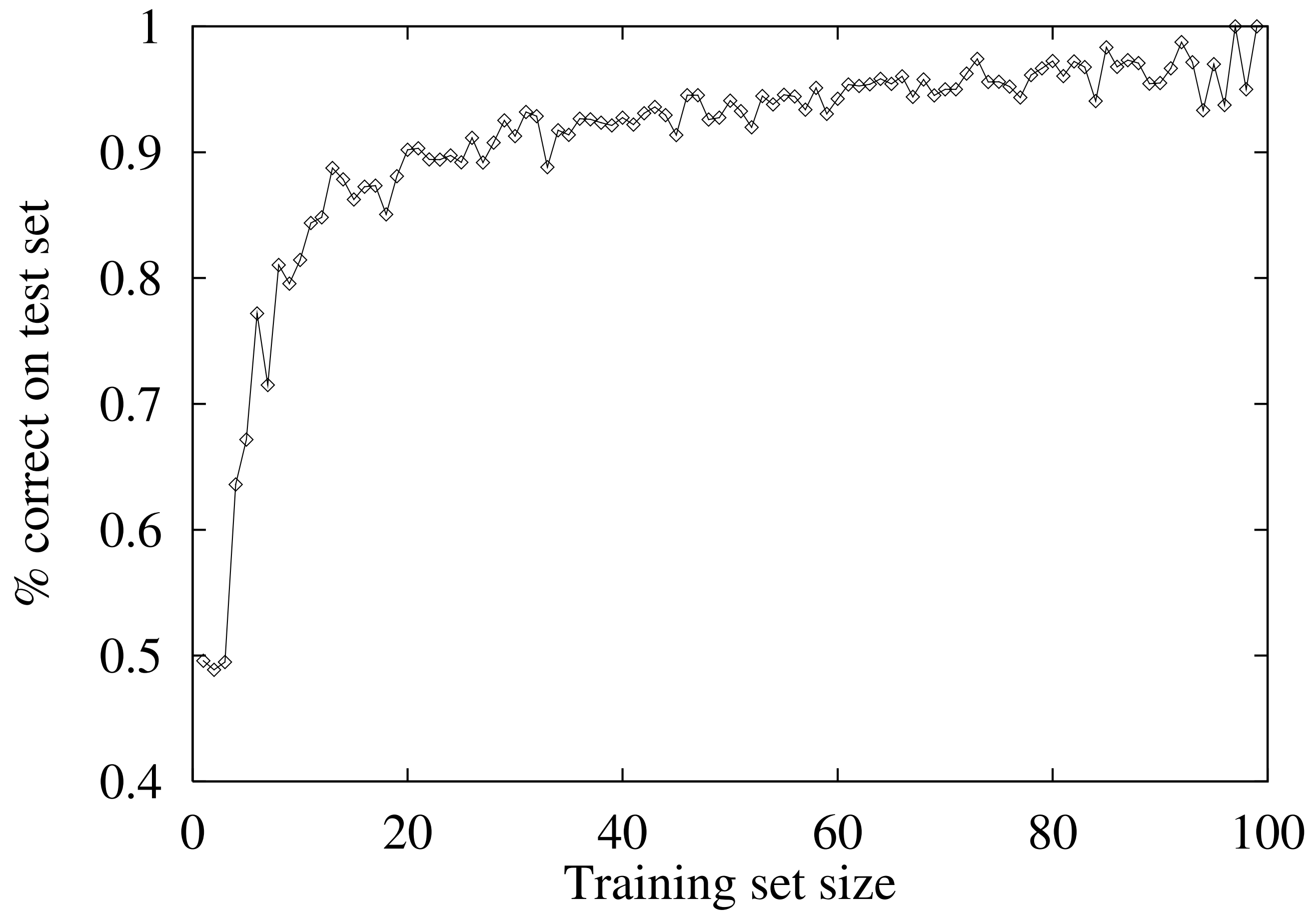
- Observações:
  - árvore induzida é diferente da árvore original, apesar do agente ter utilizado exemplos gerados por aquela árvore.
  - algoritmo pode concluir fatos que não estão muito evidentes dos exemplos: sempre esperar por um restaurante Tailandes, se for um fim de semana.
  - Por causa destes fatos: muito tempo gasto em depuração procurando por erros não existentes!
  - qto mais exemplos, mas detalhada será a árvore de decisão.
  - Figura 18.8 pode induzir erros já que nunca viu um caso onde o tempo de espera é 0-10mins, mas o restaurante está cheio.
- Questão: se o algoritmo induz uma árvore consistente, mas incorreta, através dos exemplos, quão incorreta é a árvore?

Desempenho de um Algoritmo de Aprendizado

- Um algoritmo de aprendizado é bom se produzir hipóteses que classificam bem exemplos ainda não vistos.
- duas formas de avaliar desempenho: após os fatos, em avanço.
- Método após os fatos: verificação das previsões de acordo com as classificações corretas num **conjunto de teste**.
  1. Escolher um conjunto grande de exemplos.
  2. Dividir este conj em conj de treinamento e conj de teste.
  3. Usar o algoritmo com o conj de treinamento como exemplo para gerar a hipótese H.
  4. Calcular a percentagem de exemplos no conj de teste que estejam corretamente classificados por H.
  5. Repetir passos 1 a 4 para tamanhos diferentes de conjs de treinamento e conjs de treinamento selecionados aleatoriamente para cada tamanho.

- Resultado do método: conj de dados q podem ser processados para produzir a **curva de aprendizado**.

Desempenho de um Algoritmo de Aprendizado


1
0.9
0.8
0.7
0.6
0.5
0.4
% correct on test set
0
20
40
60
80
100
Training set size

Utilização Prática de Aprendizado em Árvores de Decisão

- GASOIL, BP, sistema escrito a mão levaria 10 anos para ser completado. Usando algoritmo baseado de indução de árvores de decisão, foi desenvolvido em 100 dias! Considerado melhor do que um expert.
- Aprendendo a voar: C4.5 utilizado para extrair a árvore de decisão através de 90.000 exemplos obtidos por pilotos experientes num simulador. Árvore resultante convertida em C aprende e voa melhor do que os instrutores.

Teoria da Informação

- Encontrar medidas formais para classificar atributos como "bom" ou "razoável" ou "pobre" etc.
- Informação representada em número de bits. Se I(p) = 1, precisamos de 1 bit de info. Se I(p) = 0, não precisamos de info adicional.
- Assuma q atributo tenha como respostas possíveis v_i com probabilidade $P(v_i)$. Informação total: $I(P(v_1), \ldots, P(v_n)) = \sum_{i=1}^{n} -P(v_i) log_2 P(v_i)$
- Codificação da info com tamanho ótimo vai ter $log_2 p$ bits para atributo que tenha prob p.
- Considerando exemplos positivos e negativos: $I(\frac{p}{p+n}, \frac{n}{p+n}) = -\frac{p}{p+n} log_2 \frac{p}{p+n} - \frac{p}{p+n} log_2 \frac{n}{p+n}$, estimativa da info contida em uma resposta correta.

- **Ganho de informação**: diferença entre a info original e o novo requisito. $Ganho(A) = I(\frac{p}{p+n}, \frac{n}{p+n}) - Restante(A)$
- Heurística usada por CHOOSE-ATTRIBUTE escolhe o atributo com maior ganho.
- Ex:
  $Ganho(Clientes) = 1 - [\frac{2}{12}I(0,1) + \frac{4}{12}I(1,0) + \frac{6}{12}I(\frac{2}{6}, \frac{4}{6})] \approx 0.541$ bits.