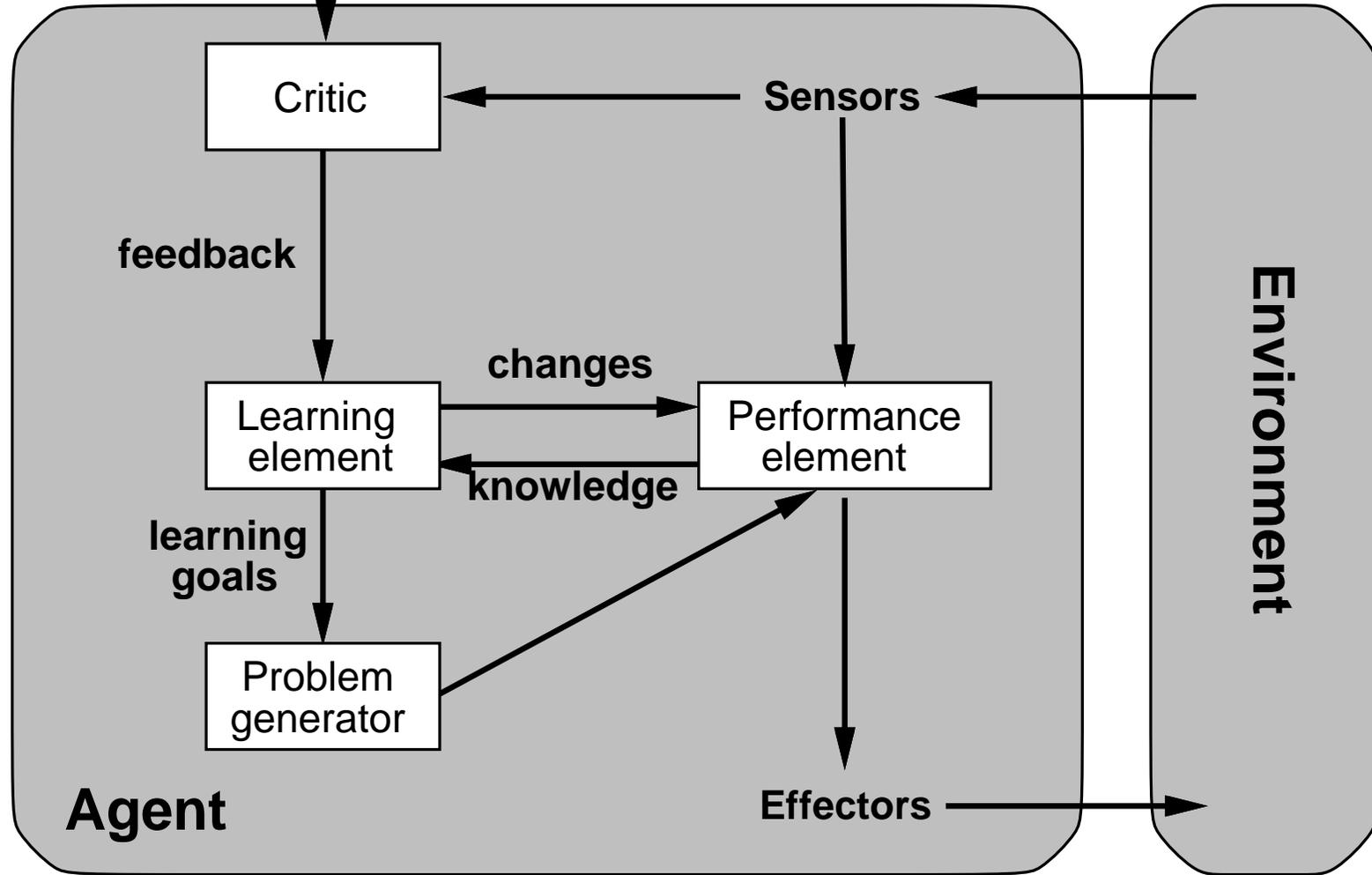


Cap. 18: Aprendendo através de observações

Performance standard



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 (conhecimento prévio) 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.

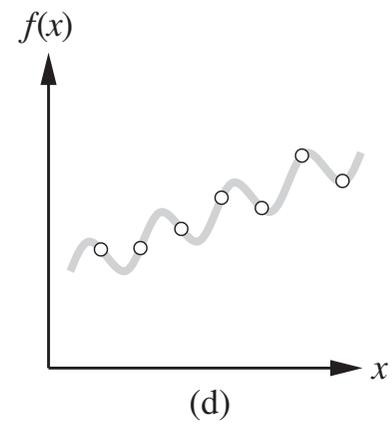
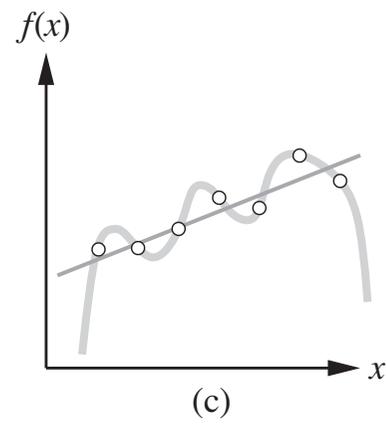
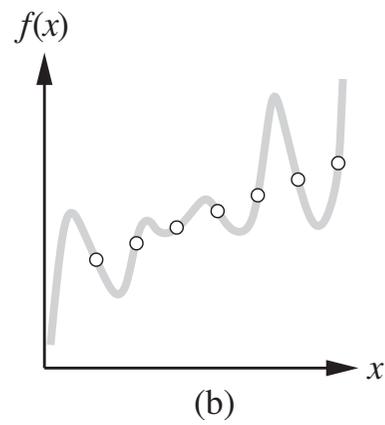
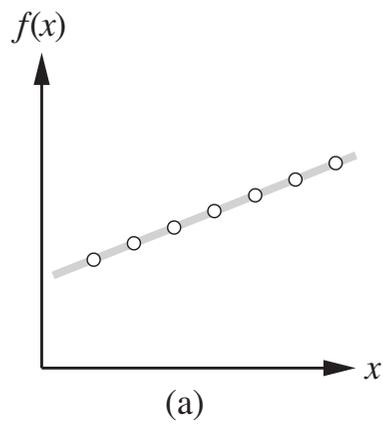
Aprendendo através de observações

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

Aprendizagem Indutiva (Inductive Learning)

- Em aprendizagem supervisionada o elemento de aprendizagem 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 e $f(x)$ 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.

Aprendizagem Indutiva (Inductive Learning)



global $examples \leftarrow \{\}$

function REFLEX-PERFORMANCE-ELEMENT($percept$) **returns** an action

if ($percept, a$) **in** $examples$ **then return** a

else

$h \leftarrow$ INDUCE($examples$)

return $h(percept)$

procedure REFLEX-LEARNING-ELEMENT($percept, action$)

inputs: $percept$, feedback percept

$action$, feedback action

$examples \leftarrow examples \cup \{(percept, action)\}$

Aprendizagem Indutiva (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 aprendizagem INDUCE sobre exemplos que viu até então.
- INDUCE retorna uma hipótese h que é usada para escolher uma ação.

Aprendizagem Indutiva (Inductive Learning)

- Alternativa: **aprendizagem 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 aprendizagem: 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 aprendizagem. É “representável” na linguagem? É eficiente (tratável)?

Aprendizagem Indutiva (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 experiências futuras e reflete corretamente experiências passadas.
- Problema filosófico: como é que podemos saber se um algoritmo de aprendizagem está produzindo uma teoria q prevê corretamente o futuro?

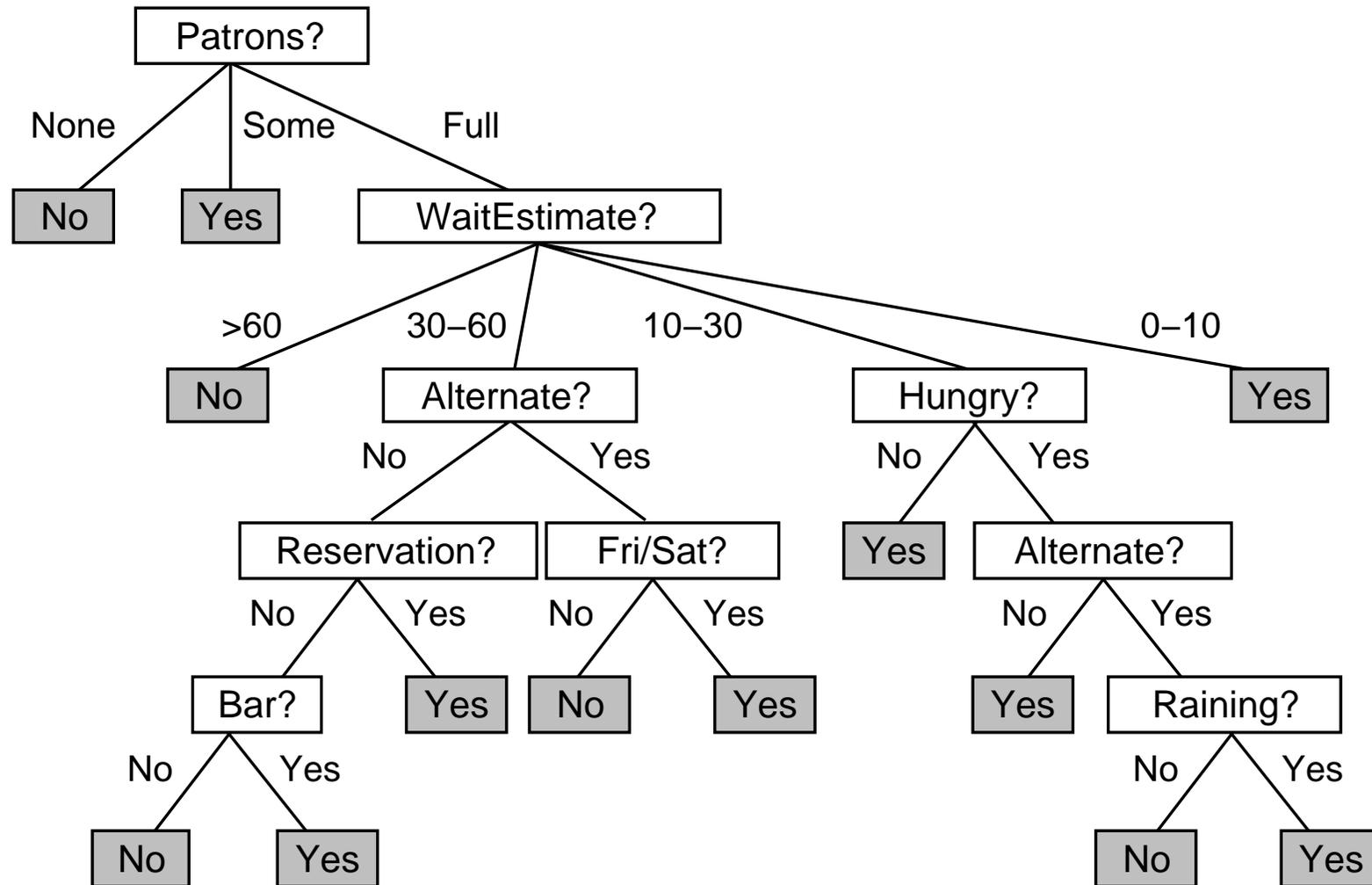
Árvores de Decisão

- 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.

Árvores de Decisão

- Decidimos (p eqto, pois isto poderia ser decidido pelo algoritmo de aprendizagem) 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.

Example	Attributes										Goal
	<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>WillWait</i>
<i>X</i> ₁	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>French</i>	<i>0–10</i>	<i>Yes</i>
<i>X</i> ₂	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Thai</i>	<i>30–60</i>	<i>No</i>
<i>X</i> ₃	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Some</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Burger</i>	<i>0–10</i>	<i>Yes</i>
<i>X</i> ₄	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Thai</i>	<i>10–30</i>	<i>Yes</i>
<i>X</i> ₅	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>Full</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>French</i>	<i>>60</i>	<i>No</i>
<i>X</i> ₆	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$</i>	<i>Yes</i>	<i>Yes</i>	<i>Italian</i>	<i>0–10</i>	<i>Yes</i>
<i>X</i> ₇	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>None</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Burger</i>	<i>0–10</i>	<i>No</i>
<i>X</i> ₈	<i>No</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$</i>	<i>Yes</i>	<i>Yes</i>	<i>Thai</i>	<i>0–10</i>	<i>Yes</i>
<i>X</i> ₉	<i>No</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>Full</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Burger</i>	<i>>60</i>	<i>No</i>
<i>X</i> ₁₀	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>Italian</i>	<i>10–30</i>	<i>No</i>
<i>X</i> ₁₁	<i>No</i>	<i>No</i>	<i>No</i>	<i>No</i>	<i>None</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Thai</i>	<i>0–10</i>	<i>No</i>
<i>X</i> ₁₂	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Burger</i>	<i>30–60</i>	<i>Yes</i>



Árvores de Decisão

- Em lógica:

$$\forall r \text{ Clientes}(r, \text{Cheio}) \wedge \text{EsperaEstimada}(r, 10 - 30) \wedge \\ \neg \text{ComFome}(r, N) \Rightarrow \text{VouEsperar}(r)$$

- Árvores de decisão não conseguem representar testes sobre 2 ou mais objetos diferentes (objetos devem ser “ground”).
- 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

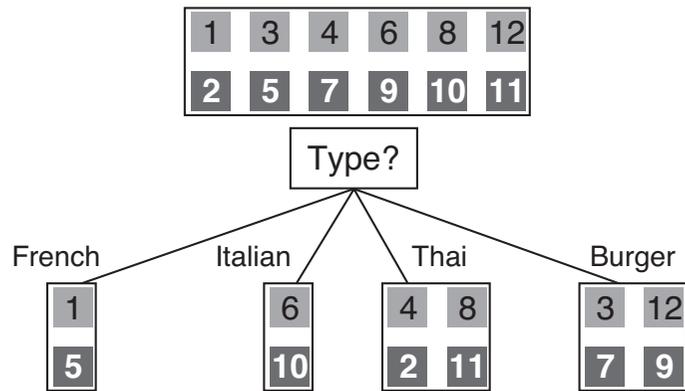
- **Exemplos:** valores dos atributos + valor do predicado desejado.
- **Classificação** do exemplo: valor do predicado.
- qdo valor é verdadeiro, exemplo **positivo**. Caso contrário, é um exemplo **negativo**.
- conj completo de exemplos: **conjunto de treinamento**.

Árvores de Decisão

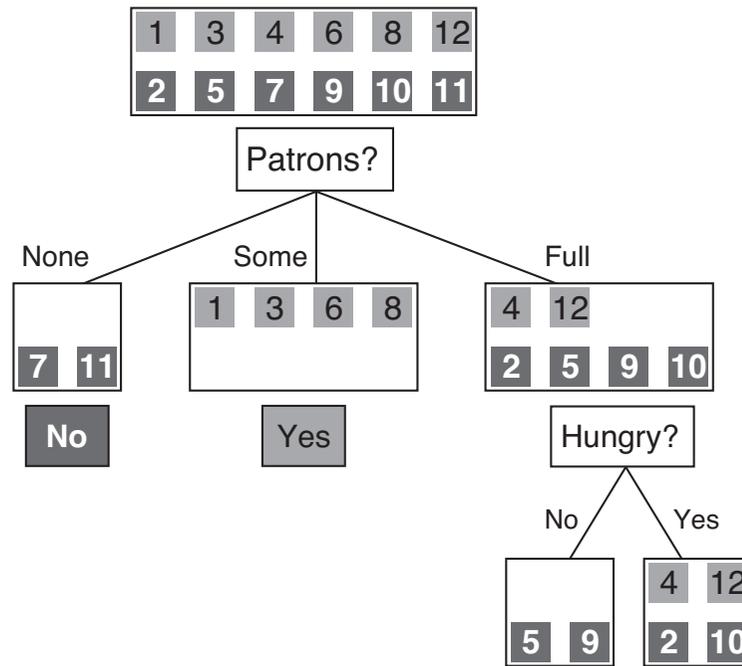
- Como induzir uma árvore de decisão através de exemplos?
- Cada exemplo pode ser um caminho diferente na árvore.
- limitação de representação, não deixa extrair nenhum outro padrão de informação além daquele descrito pelos exemplos já conhecidos.
- Extrair um padrão significa descrever um grande número de casos de forma concisa.
- Princípio geral de aprendizagem indutiva: **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.



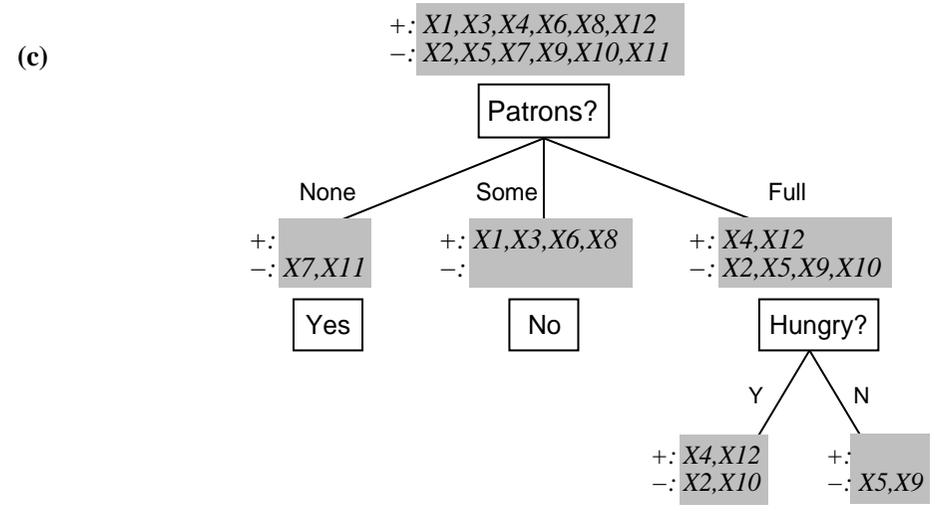
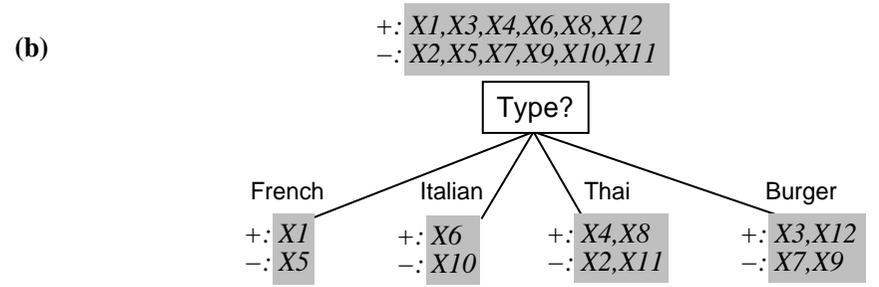
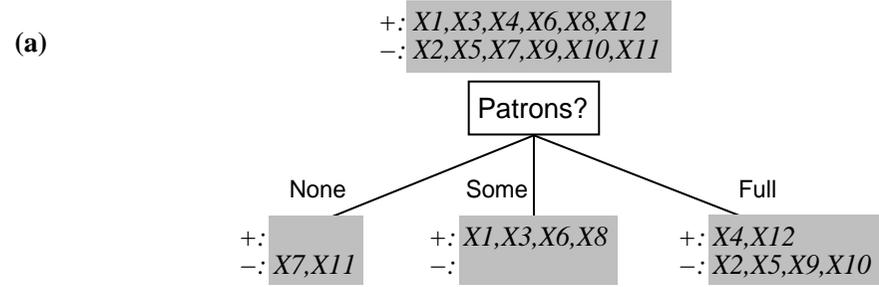
(a)

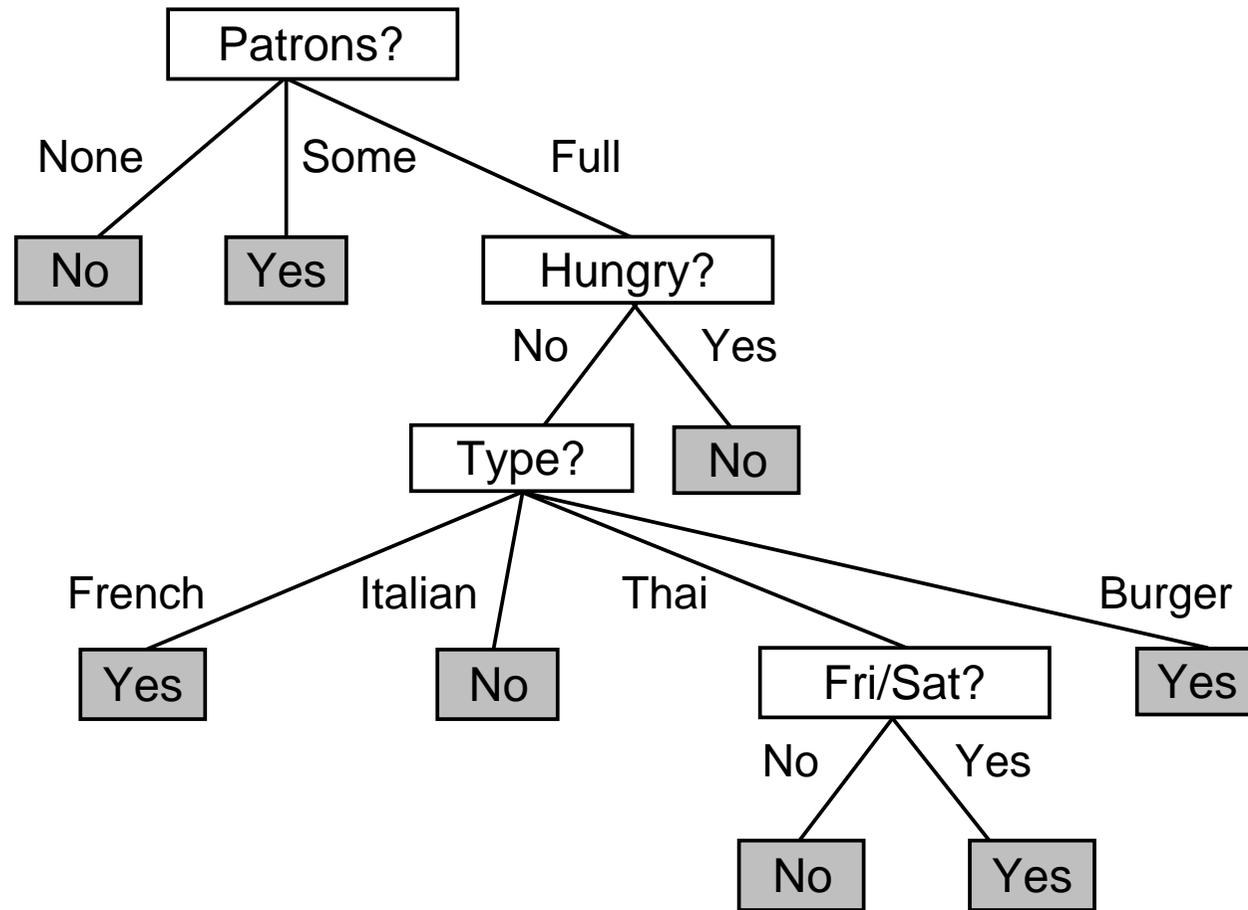


(b)

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.





Árvores de Decisão

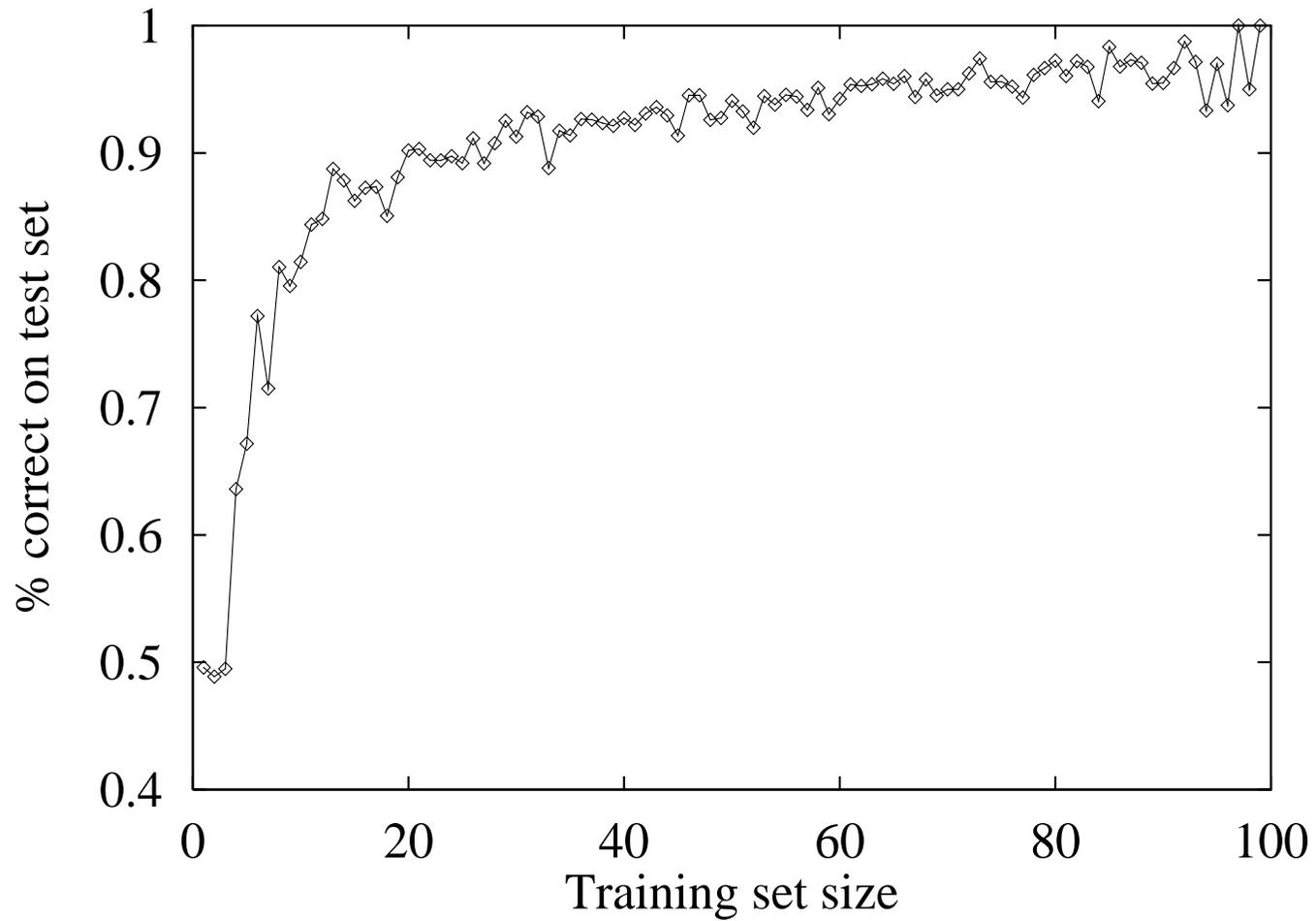
- Observações:
 - árvore induzida é diferente da árvore original, apesar do agente ter utilizado exemplos gerados por aquela árvore.
 - algoritmo pode concluir fatos que não estão muito evidentes dos exemplos: sempre esperar por um restaurante Tailandês, se for um fim de semana.
 - Por causa destes fatos: muito tempo gasto em depuração procurando por erros não existentes!
 - qto mais exemplos mais detalhada será a árvore de decisão.
 - Neste exemplo, a árvore 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 Aprendizagem

- Um algoritmo de aprendizagem 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 aprendizagem**.

Desempenho de um Algoritmo de Aprendizagem



Utilização Prática de Aprendizagem em Árvores de Decisão

- GASOIL, BP, sistema escrito a mão levaria 10 anos para ser completado. Usando algoritmo baseado em 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:

$$I(P(v_1), \dots, P(v_n)) = \sum_{i=1}^n -P(v_i) \log_2 P(v_i)$$

- Codificação da info com tamanho ótimo vai ter $\log_2 p$ bits para atributo que tenha prob p .

- Considerando exemplos positivos e negativos:

$$I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n}, \text{ estimativa da info contida em uma resposta correta.}$$

- **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.

Árvores de Decisão

```
ID3(Examples, Target_Attribute, Attributes)
  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 number of predicting attributes is empty,
    Return the single node tree Root,
      with label = most common value of the
      target attribute in the examples.
  Else
    A = Attribute that best classifies examples
    Decision Tree attribute for Root = A
    For each possible value, vi, of A,
      Add a new tree branch below Root,
        corresponding to the test A = vi.
      Let Examples(vi) be the subset of examples that
        have the value vi for A
      If Examples(vi) is empty
        below this new branch add a leaf node with
          label = most common target value in the examples
      Else
        below this new branch add the subtree
          ID3 (Examples(vi), Target_Attribute, Attributes - {A})
      EndIf
    EndFor
  EndIf
Return Root
```