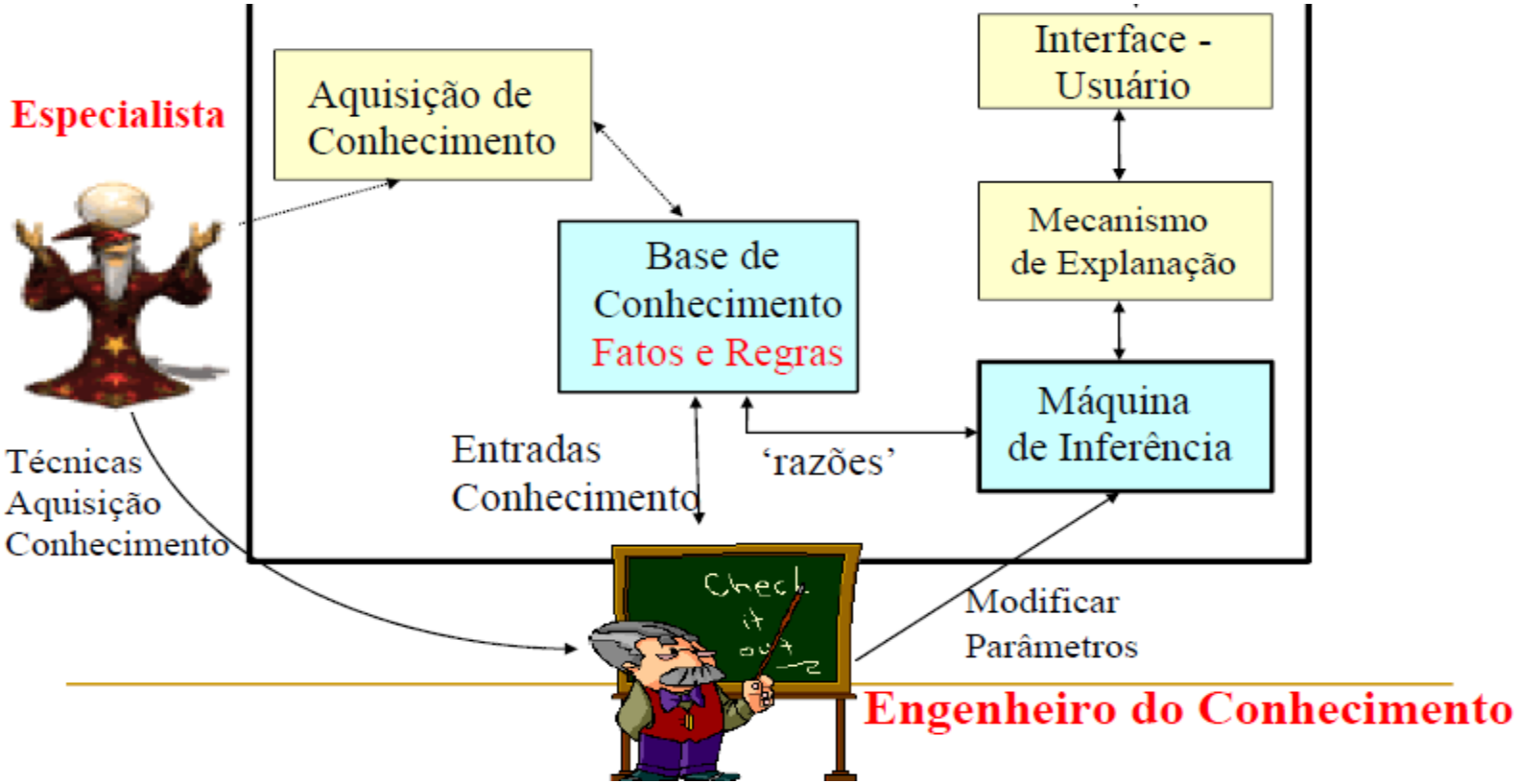
Aprendizagem a partir de observações

Capítulo 18 - Russell & Norvig

Prof. Lucas Cambuim

Problema com Sistemas Especialistas

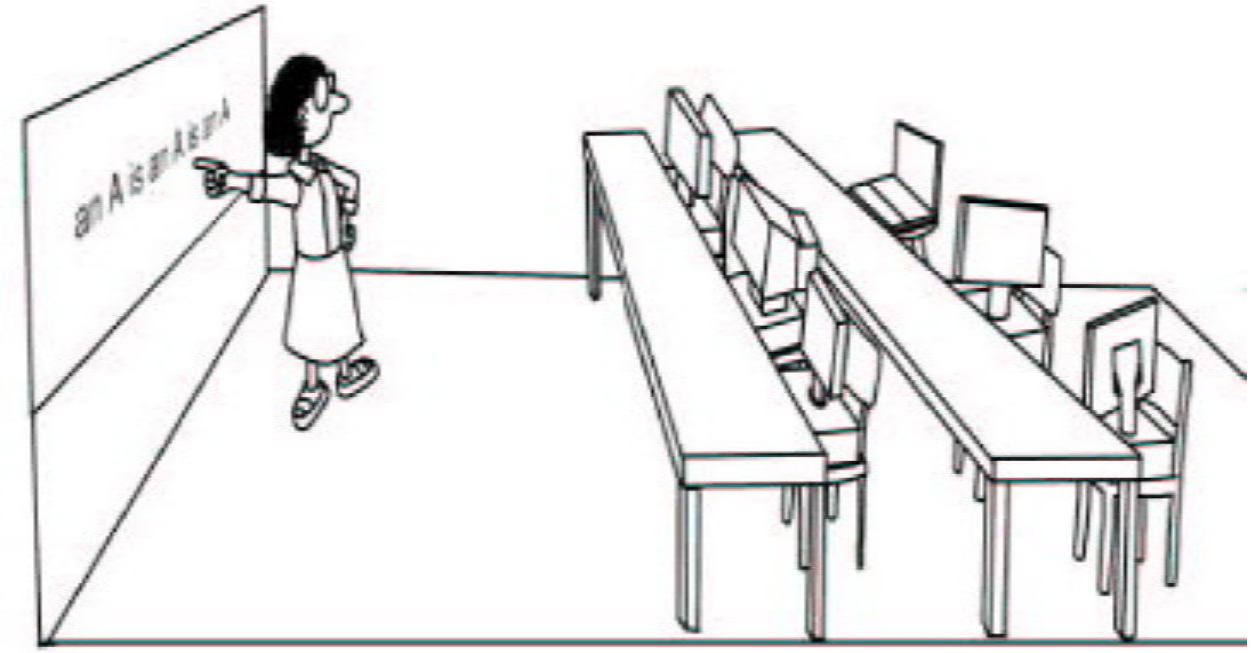
- Nem sempre é fácil obter todas informações do especialista



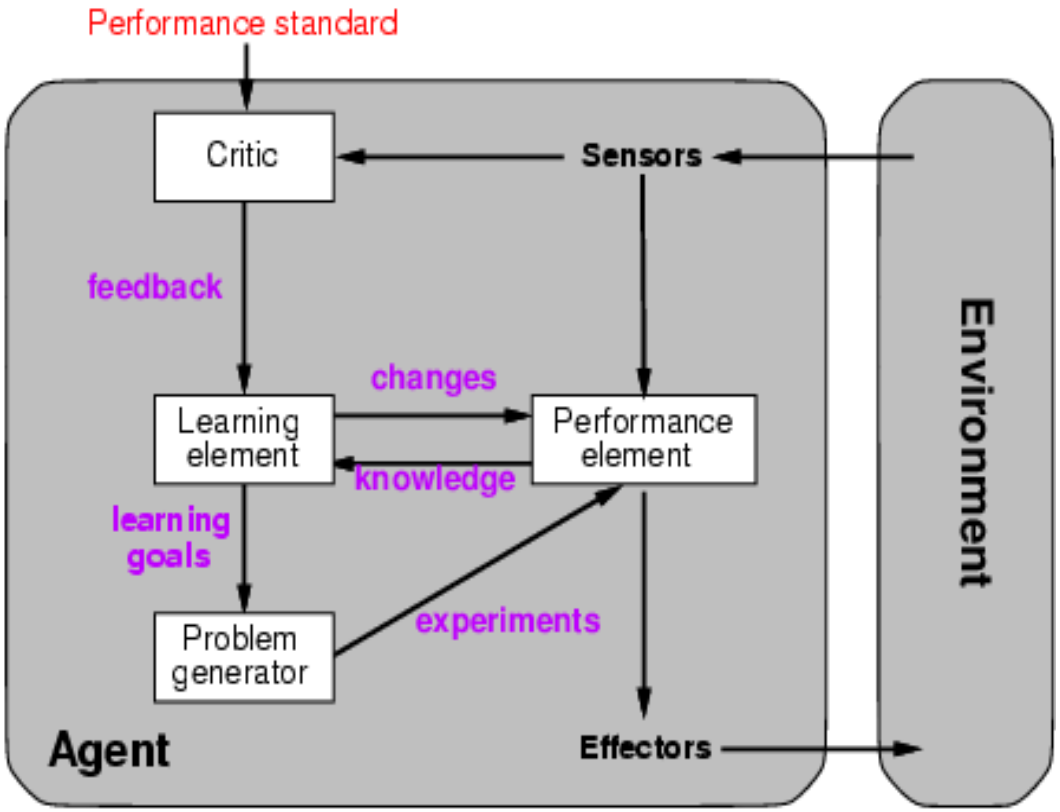
Aprendizagem

- é essencial para ambientes desconhecidos,
  - i.e., quando o projetista não prevê tudo
- Útil como um método de construção de sistemas
  - i.e., expor o agente à realidade ao invés de "representá-la" totalmente
- Modifica os mecanismos de decisão de um agente para melhorar sua performance.

Aprendizado de Máquina (do inglês, *Machine Learning*) é a área de Inteligência Artificial cujo objetivo é o desenvolvimento de **técnicas computacionais sobre processo de aprendizado** (BISHOP, 2007).



Agente de aprendizagem



- Elemento de desempenho:
  - decide que ações utilizar;
- Elemento de aprendizagem:
  - modifica o elemento de desempenho para que ele tome decisões melhores;

A idéia por trás da aprendizagem é que as percepções devem ser usadas não apenas para agir, mas, também para melhorar a habilidade do agente (RUSSEL & NORVIG, 2004).

Elemento de aprendizagem

- O projeto de um elemento de aprendizagem é afetado por:
  - Os componentes do elemento de desempenho que devem ser aprendidos;
  - A realimentação que estará disponível para aprender estes componentes;
  - A representação que será utilizada para os componentes

Elemento de aprendizagem

- Tipos de realimentação (*feedback*):
  - Aprendizagem supervisionada:
    - Aprendizagem de uma função a partir de exemplos de entrada e saída
    - respostas corretas para cada exemplo
  - Aprendizagem não-supervisionada:
    - Aprendizagem de padrões de entrada, quando não há valores de saída específicos
    - respostas corretas não são dadas
  - Aprendizagem por reforço:
    - aprendizagem dado recompensas ocasionais

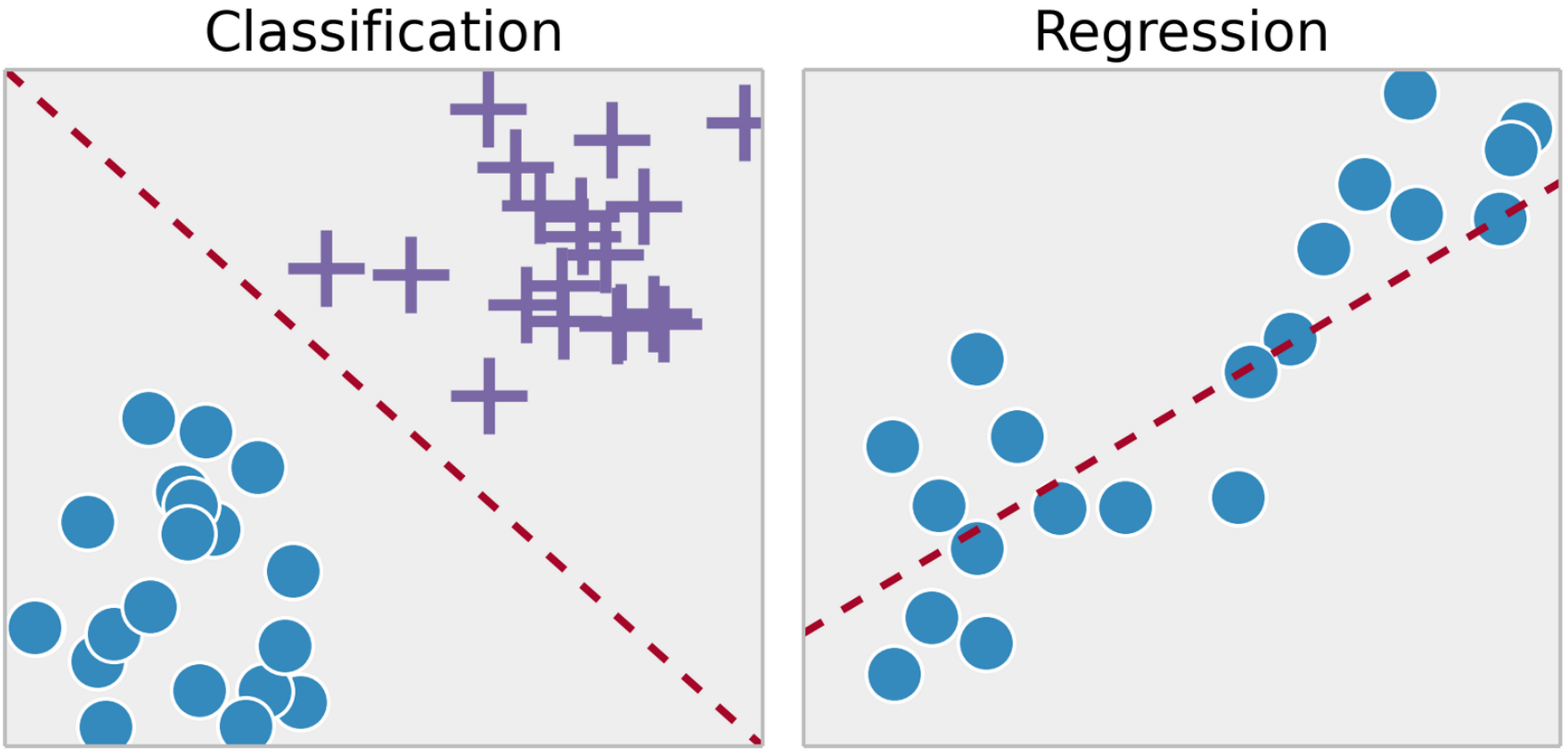
Aprendizado Supervisionado

- É fornecida uma referência do objetivo a ser alcançado:
  - o algoritmo de aprendizado recebe o valor de saída desejado para cada conjunto de dados de entrada apresentado.

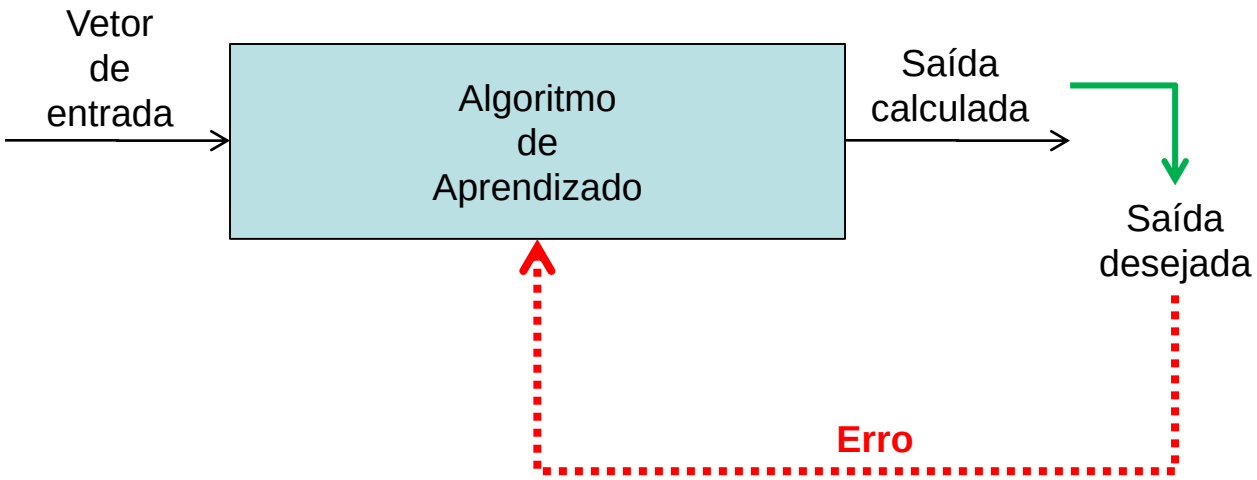
- Envolve o aprendizado de uma função a partir de exemplos de sua entrada e saída.

- Para rótulos discretos, esse problema é chamado de classificação e para valores contínuos como regressão.

- Exemplos de algoritmos: Árvores de Decisão, Redes Neurais (BP), SVM, TBL.


Classification
Regression

Aprendizado Supervisionado



Aprendizado Não-Supervisionado

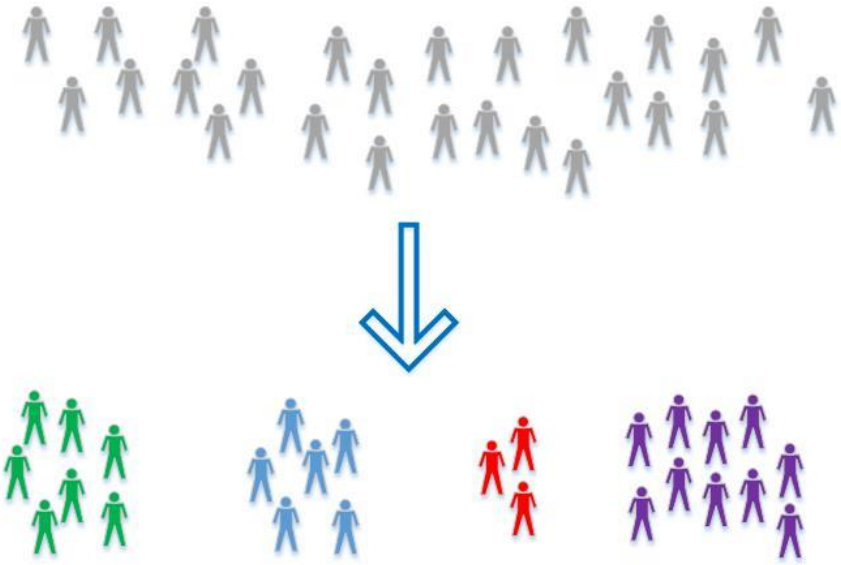
- É fornecido somente o conjunto de dados de entrada:
  - Não existe “a” saída desejada.

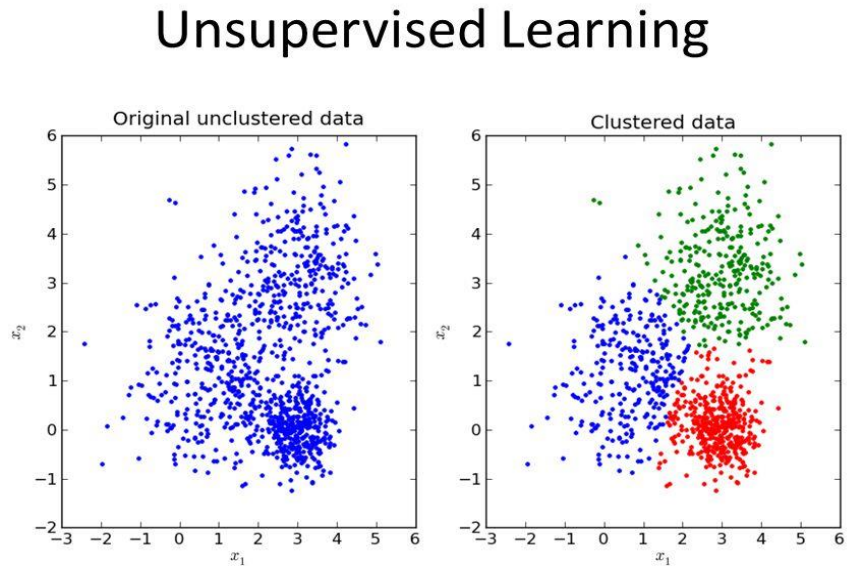
- Envolve a aprendizagem de padrões na entrada quando não são apresentados valores de saída específicos.

- Em geral, é utilizado para encontrar aglomerados de conjuntos de dados semelhantes entre si (*clusters*).

- Exemplos de algoritmos: C-means, K-means, KNN, Redes Neurais (SOM).

Aprendizado Não-Supervisionado

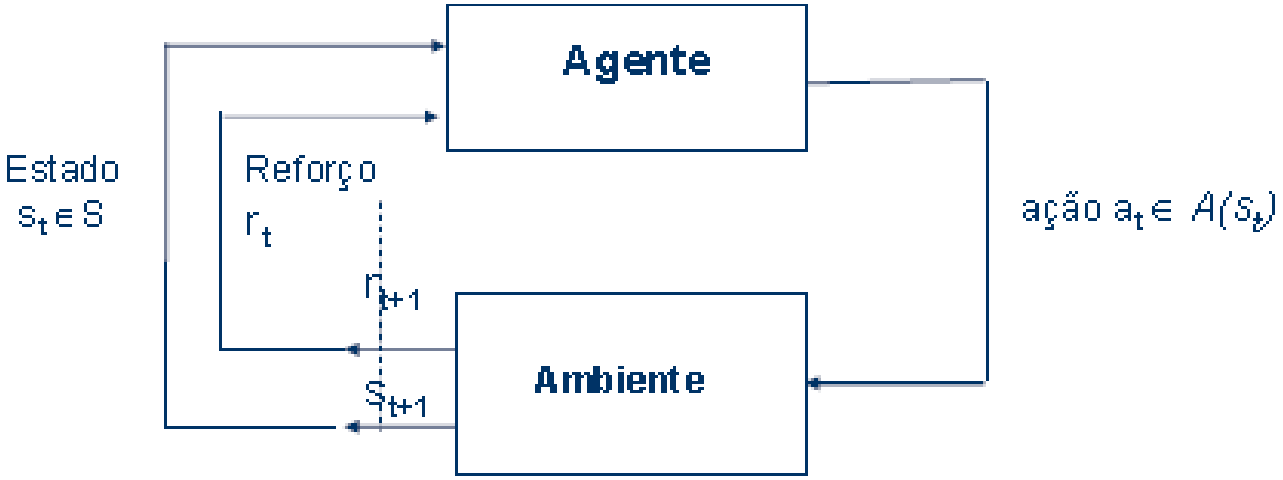




Aprendizado por Reforço

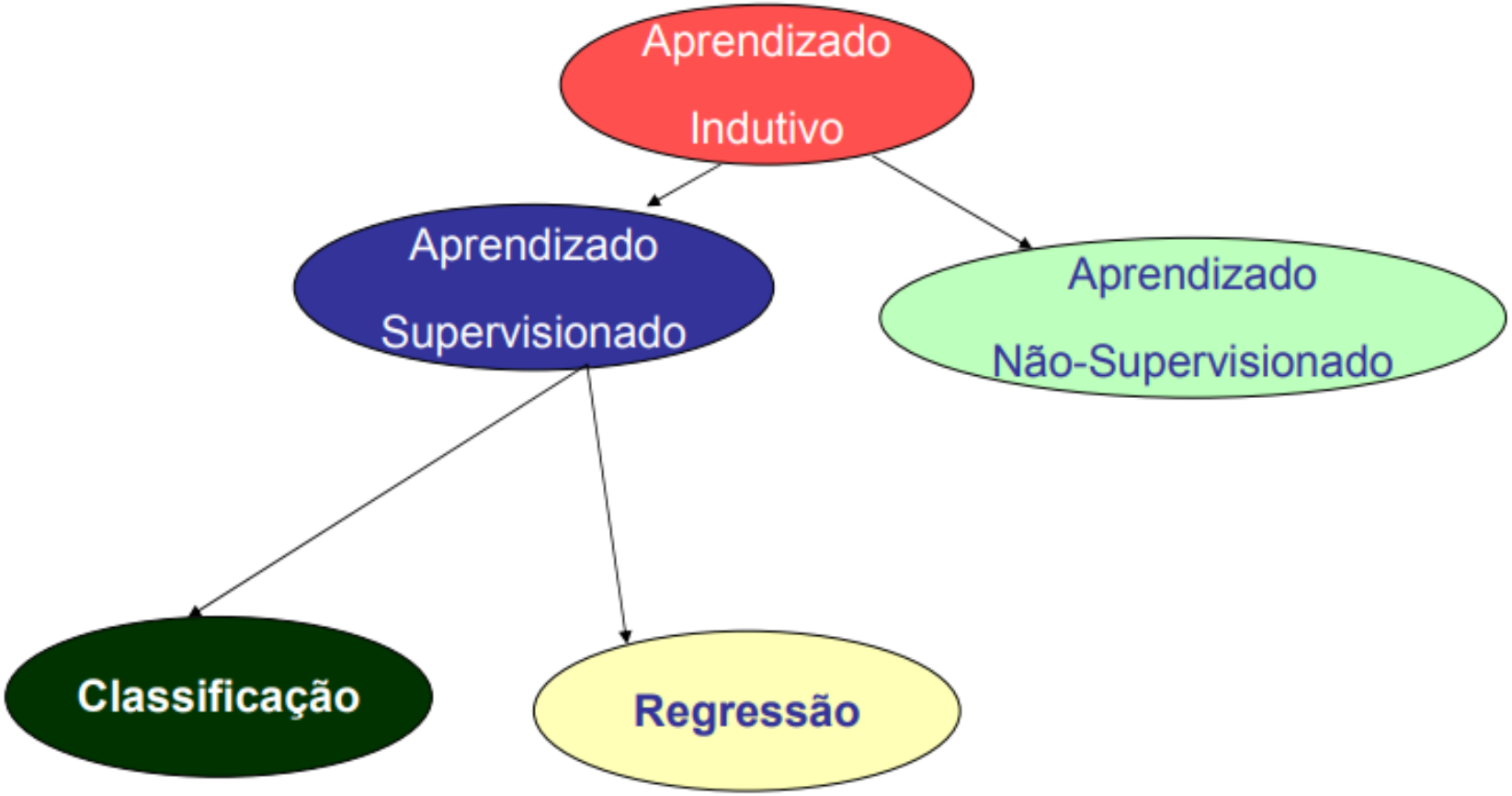
- Aprendizado a partir da interação "*learner–environment*":
  - Muitas vezes é impraticável o uso de aprendizagem supervisionada.
- Baseado em "tentativa e erro".
- Existe processo de busca (exploration) no espaço
- Aprende a escolher ações apenas interagindo com o ambiente.
- Através das interações, o agente descobre as relações de causa e efeito.

Aprendizado por Reforço



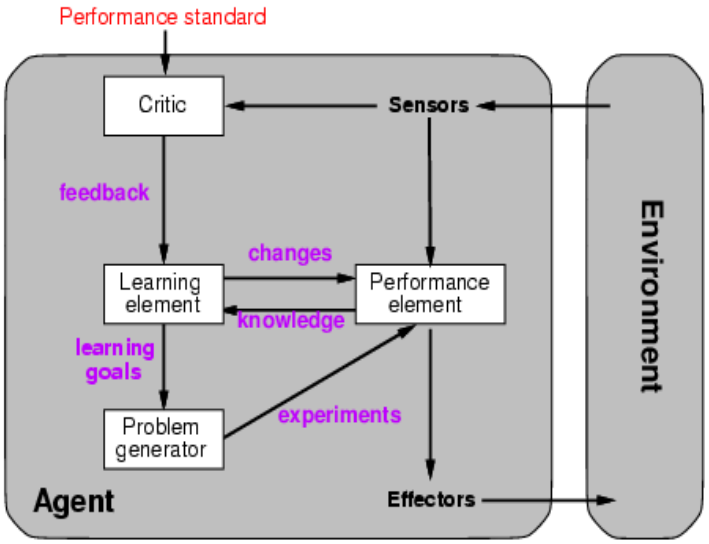
O agente recebe do ambiente um valor de resposta (recompensa/reforço).

Esta recompensa avalia o desempenho do agente durante o processo de aprendizado.


Aprendizado
Indutivo
Aprendizado
Supervisionado
Aprendizado
Não-Supervisionado
Classificação
Regressão

Elemento de aprendizagem

- Representação das informações aprendidas:
  - determina como o algoritmo de aprendizagem deve funcionar
    - polinômios lineares para funções de utilidade em jogos
    - lógica proposicional ou de 1a ordem
    - redes bayesianas
    - etc...



Aprendizagem Indutiva Clássica

- recebe como entrada o valor correto de uma função desconhecida para entradas específicas e tenta recuperar a função
- Dada uma coleção de exemplos de **f**, retornar uma função **h** que se aproxime de **f**
  - f é a função alvo
  - h é a hipótese
  - (x, f(x)) são exemplos de entrada

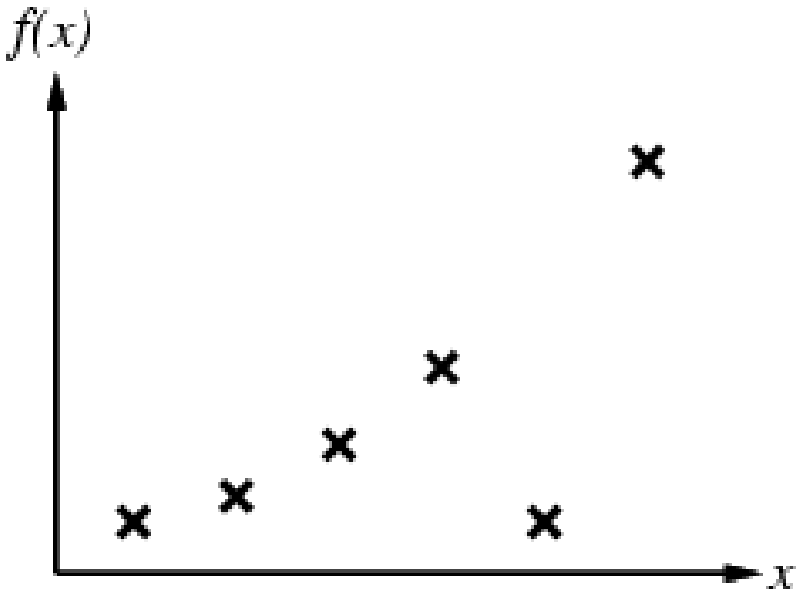
(Este é um modelo extremamente simplificado de aprendizagem indutiva onde:
  - não há conhecimento de fundo (*background knowledge*)
  - Assume que os exemplos são dados

Exemplos de Aplicações de Aprendizagem

- Detecção de spam
  - $\boldsymbol{x}$: descreve o conteúdo de mensagens de e-mail, em geral utiliza-se uma dimensão por palavra, além de outros atributos como número de palavras erradas, número de links, etc.
  - $\boldsymbol{f(x)}$: indica se a mensagem é spam ou não (em geral fornecido pelos usuários).
- Reconhecimento de caracteres
  - $\boldsymbol{x}$: pixels da imagem em que o caracter aparece.
  - $\boldsymbol{f(x)}$: indica o caracter.
- Diagnóstico de doenças
  - $\boldsymbol{x}$: descreve o resultado de exames do paciente.
  - $\boldsymbol{f(x)}$: indica se o paciente tem a doença ou não.
- Classificação de proteínas
  - $\boldsymbol{x}$: seqüência de aminoácidos da proteína.
  - $\boldsymbol{f(x)}$: indica a família a qual a proteína pertence.

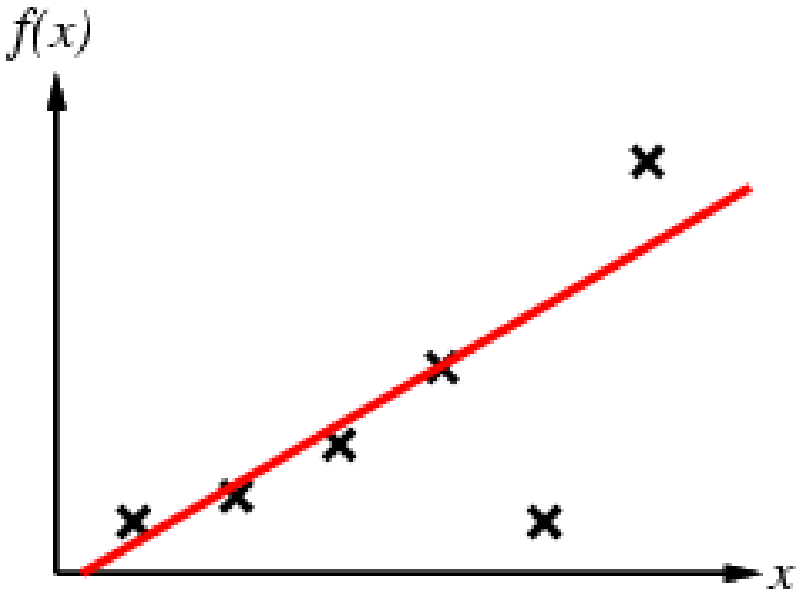
Aprendizagem indutiva

- Construir/adjustar *h* que concorde com *f* no conjunto de treinamento
- ***h* é consistente se concorda com todos os exemplos dados**
- E.g., ajuste de curva:



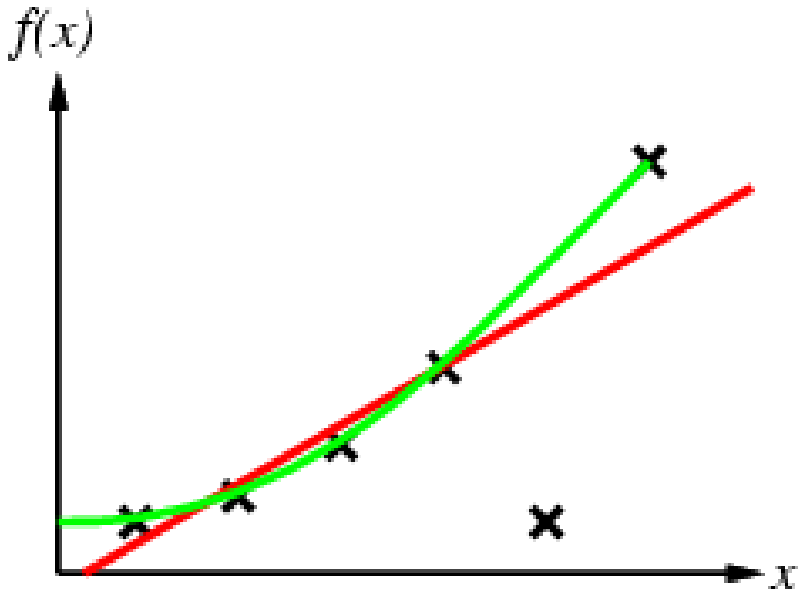
Aprendizagem indutiva

- Construir/adjustar *h* que concorde com *f* no conjunto de treinamento
- ***h* é consistente se concorda com todos os exemplos dados**
- E.g., ajuste de curva:



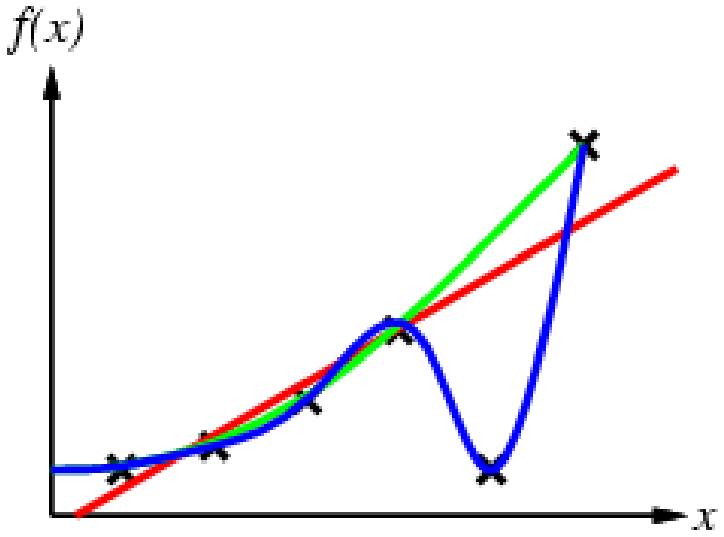
Aprendizagem indutiva

- Construir/adjustar *h* que concorde com *f* no conjunto de treinamento
- ***h* é consistente se concorda com todos os exemplos dados**
- E.g., ajuste de curva:



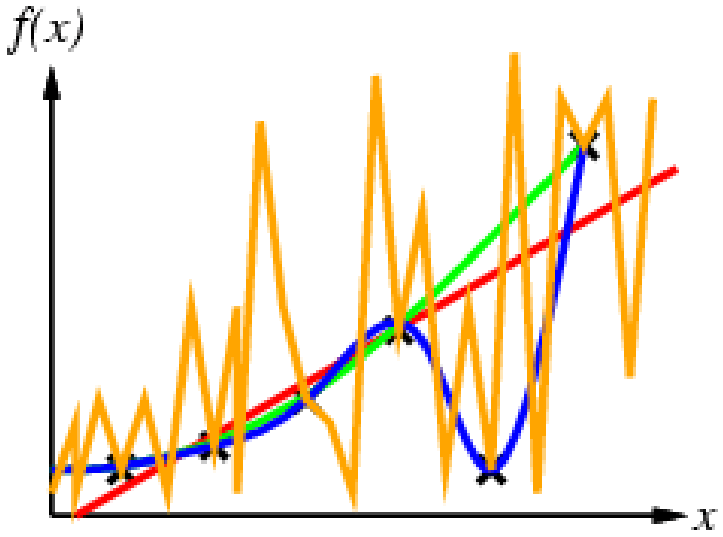
Aprendizagem indutiva

- Construir/adjustar *h* que concorde com *f* no conjunto de treinamento
- ***h* é consistente se concorda com todos os exemplos dados**
- E.g., ajuste de curva:



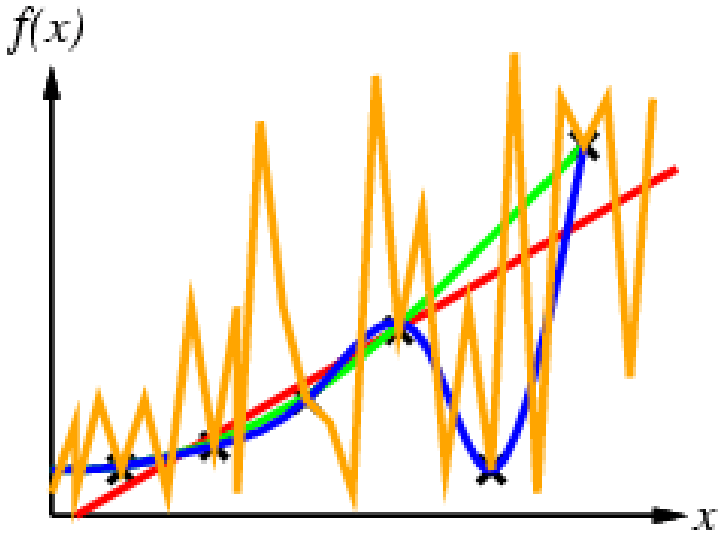
Aprendizagem indutiva

- Construir/adjustar *h* que concorde com *f* no conjunto de treinamento
- ***h* é consistente se concorda com todos os exemplos dados**
- E.g., ajuste de curva:



Aprendizagem indutiva

- Como escolher entre as várias hipóteses disponíveis?
  - **lâmina de Ockam (*Ockam's razor*)**: prefira a hipótese ***mais simples*** consistente com os dados



hipóteses que são mais complexas que os dados estão deixando de extrair algum padrão dos dados!

Lâmina de Ockam

- talvez seja melhor ajustar uma simples linha reta que não seja exatamente consistente, mas possa fazer previsões razoáveis;
- I.e. aceitar que a verdadeira função não seja determinística
  - as verdadeiras entradas não sejam completamente observadas
- Há um compromisso inevitável entre a complexidade da hipótese e o grau de ajuste aos dados

Espaço de hipótese

- Deve-se ter em mente que a possibilidade ou impossibilidade de encontrar uma hipótese simples e consistente depende do espaço de hipótese escolhido.
  - Ex. ax+ b + c sen x
- Um problema de aprendizagem é **realizável** se o espaço de hipótese contém a função verdadeira
  - nem sempre é possível de decidir pois a função verdadeira não é conhecida

Realizável?

- utilizar *conhecimento anterior* para derivar um espaço de hipóteses em que sabemos que a função verdadeira existe (cap. 19)
- Utilizar o maior espaço de hipóteses possível:
  - por que não **H** = classe de todas as máquinas de Turing? (já que toda função computável pode ser representada por uma máquina de Turing)

H = classe de todas as maquinas de Turing

- Existe um compromisso entre a expressividade de um espaço de hipóteses e a complexidade de encontrar hipóteses simples e consistentes dentro deste espaço...
  - ajuste de retas é fácil, ajustar polinômios de graus elevados é mais difícil e ajustar máquinas de Turing à realidade é muito difícil!
    - determinar se uma máquina de Turing é consistente com os dados não é nem mesmo decidível em geral.

H = classe de todas as maquinas de Turing

  - Segunda razão para espaços de hipóteses mais simples: as hipóteses resultantes podem ser mais simples de usar.

- Por isso, a maior parte do trabalho em aprendizagem se concentra em representações simples!

Aprendizagem em árvores de decisão

- **Indução de árvores de decisão:** forma mais simples (e mais bem sucedidas) de algoritmos de aprendizagem
- **árvore de decisão**: toma como entrada um objeto ou situação descritos por um conjunto de **atributos** e retorna uma decisão -- valor de saída previsto, dada uma entrada

Árvores de Decisão

- Ampla classe de algoritmos de aprendizado
  - Exemplo: ID3, C4.5, CART,...

- Conhecimento representado em uma *árvore de decisão*, em geral, na linguagem da lógica de atributos

Exemplo: Árvore de Decisão Booleana

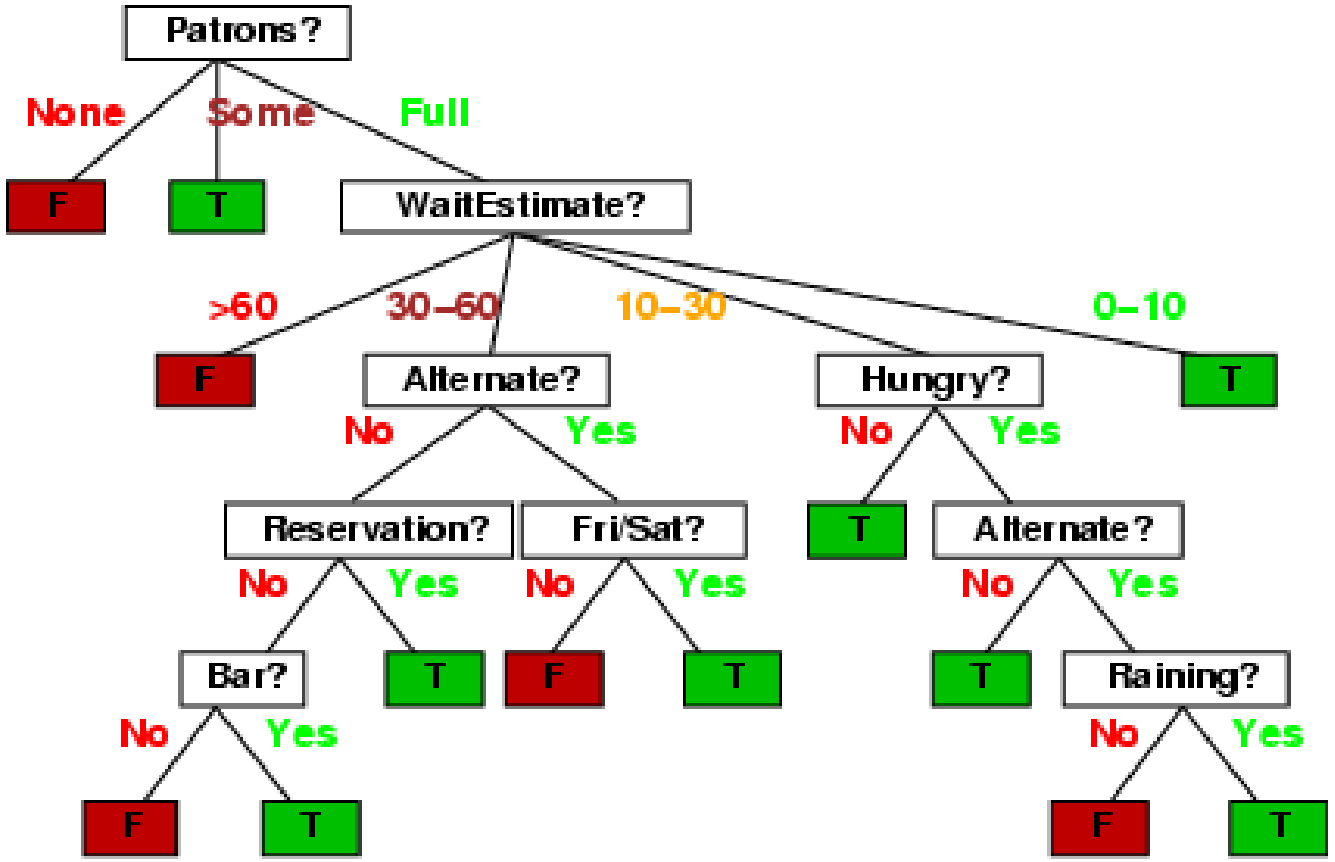
- Cada exemplo é classificado como verdadeiro ou falso
- Equivalente lógico à lógica proposicional em forma normal disjuntiva
  - Objetivo ⇔ (Caminho1 v Caminho2 v ...)
    - Onde cada caminho é uma conjunção de testes de atributo

árvores de decisão

Exemplo: decidir se devo esperar por uma mesa em um restaurante, dados os atributos:

1. ***Alternate***: há um restaurante alternativo na redondeza?
2. ***Bar***: existe um bar confortável onde se esperar?
3. ***Fri/Sat***: hoje é sexta ou sábado ?
4. ***Hungry***: estou com fome?
5. ***Patrons***: numero de pessoas no restaurante (***None***, ***Some***, ***Full***)
6. ***Price***: faixa de preços ($, $$, $$$)
7. ***Raining***: está a chover?
8. ***Reservation***: temos reserva?
9. ***Type***: tipo do restaurante (***French***, ***Italian***, ***Thai***, ***Burger***)
10. ***WaitEstimate***: tempo de espera estimado (0-10, 10-30, 30-60, >60)

árvores de decisão



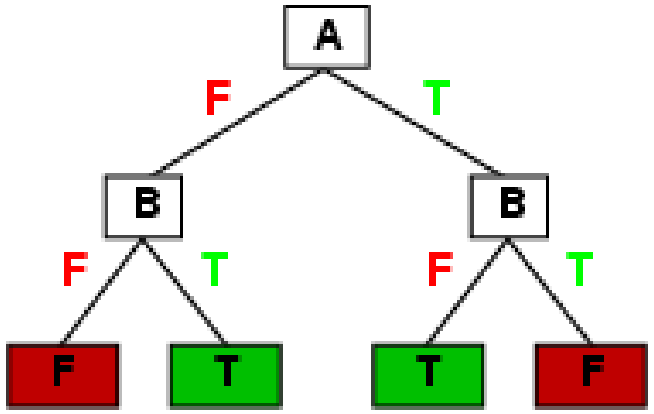
árvores de decisão

- Uma árvore de decisão alcança sua decisão executando uma sequência de testes. Cada nó interno da árvore corresponde a um teste do valor de uma das propriedades, e as ramificações a partir do nó são identificadas com os valores possíveis do teste. Cada nó de folha especifica o valor a ser retornado se aquela folha for alcançada.

Expressividade

- Qualquer função booleana pode ser escrita como uma árvore de decisão
  E.g., cada linha na tabela verdade de uma função Booleana pode corresponder a um caminho na árvore:

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



- Trivialmente, há uma árvore de decisão consistente para qqr conjunto de treinamento com um caminho (da raiz a uma folha) para cada exemplo...
  - Isso seria uma representação em uma árvore exponencialmente grande, pois a tabela verdade tem exponencialmente muitas linhas
- Devemos procurar por árvores de decisão mais compactas

Espaço de hipóteses

- Existe alguma espécie de representação que seja eficiente para TODOS os tipos de funções?

NÃO

Quantas árvores de decisão distintas existem para n atributos?

= numero de funções Booleanas (cada função Booleana tem 2^n linhas)

= numero de tabelas verdade distintas com 2^n linhas = 2^{2^n}

- Ou seja, com 6 atributos Booleanos, tem-se 18,446,744,073,709,551,616 árvores distintas!
- Ou seja, com 10 atributos Booleanos, tem-se 2^{1024} árvores ou seja, cerca de 10^{308} árvores distintas! (isto é, 1 seguido de 308 zeros!)

- O espaço de hipótese é enorme!
- É preciso de algoritmos “bem espertos” para conseguir encontrar um bom modelo nesse espaço

Induzindo árvores de decisão a partir de exemplos

- Exemplos descritos por valores de attributos (Booleanos, discretos, ou contínuos)
- E.g., exemplos de situações em que eu não esperarei 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 em positivo (T) ou negativo (F)

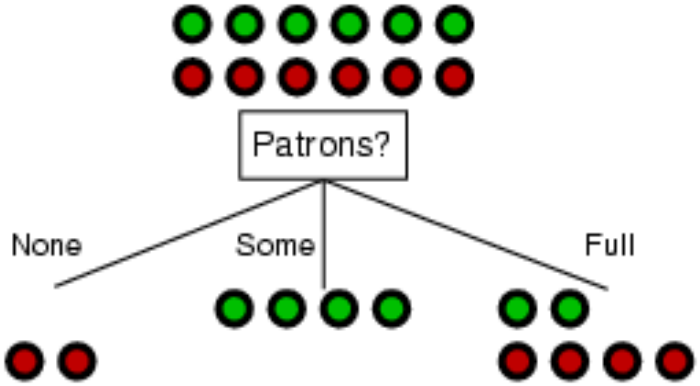
Aprendizagem por árvores de decisão

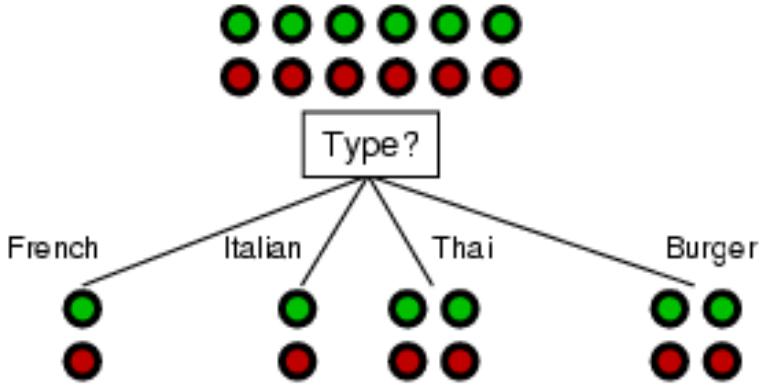
- Objetivo: encontrar a menor árvore que seja consistente com o número de exemplos
- Idéia: (recursivamente) encontre o atributo “mais significante” como raiz da sub-árvore. Isto é: estratégia gulosa baseada no atributo de maior importância a cada etapa.

```
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 atributos

- Idéia: um bom atributo divide os exemplos em subconjuntos que (preferivelmente) são “todos positivos" ou ”todos negativos"



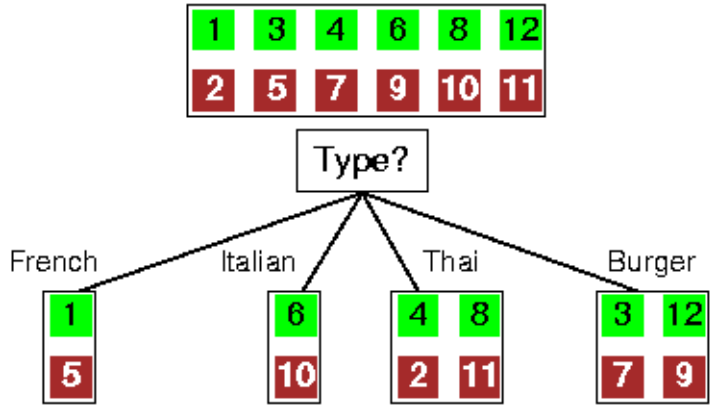


- *Patrons?* é um atributo melhor do que *Type* para ser raiz.

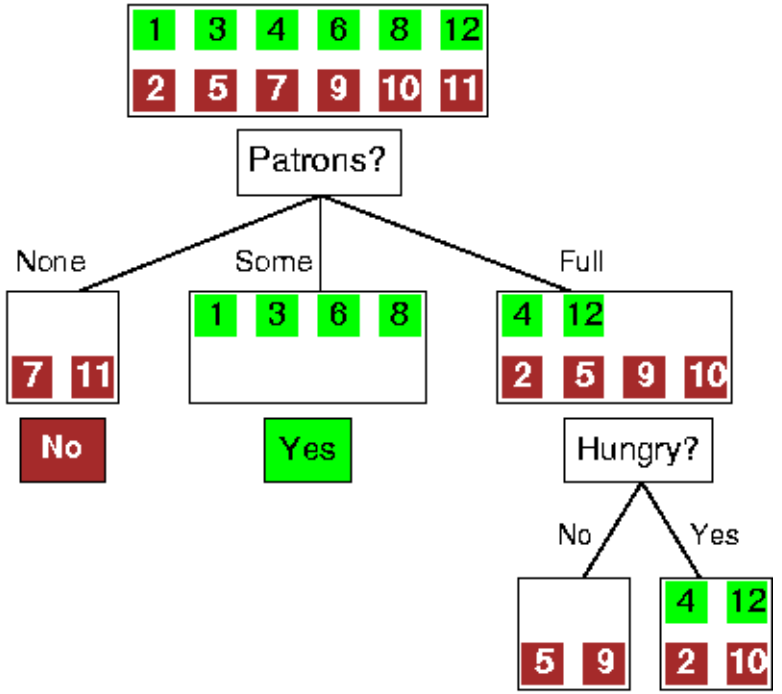
Algoritmo (informal)

- Se existem alguns exemplos positivos e alguns negativos, escolha o **melhor** atributo para dividi-los;
- Se todos os exemplos restantes forem positivos (ou todos negativos), então terminamos: podemos responder *Sim* ou *Não;*
- Se não resta nenhum exemplo, nenhum exemplo desse tipo foi observado. Então retorna-se um valor-padrão calculado a partir da classificação de maioria no pai do nó;

Escolha de atributos



(a)



(b)

Algoritmo (informal)

- Se não resta nenhum atributo, mas há exemplos positivos e negativos, temos um problema. Isso quer dizer que esses exemplos tem exatamente a mesma descrição, mas classificações diferentes. Isso acontece quando
  - alguns dados estão incorretos; dizemos que existe **ruído** nos dados;
  - os atributos não fornecem informações suficientes para descrever a situação completamente;
  - o domínio não é completamente determinístico
  - **saída simples: utilizar uma votação pela maioria**

Como definir o que é um atributo melhor ?

- A escolha de atributos deve minimizar a profundidade da árvore de decisão;
  - Escolher um atributo que vá o mais longe possível na classificação exata de exemplos;
  - Um atributo perfeito divide os exemplos em conjuntos que são todos positivos ou todos negativos.
- Solução: medir os atributos a partir da quantidade esperada de informações fornecidas por ele.

Como definir o que é um atributo melhor ?

- O atributo “patrons” não é perfeito, mas é bastante bom; o atributo “type” é completamente inútil.
- Precisamos definir uma medida formal de “bastante bom” e “bastante ruim”.
- A medida deve ter seu valor máximo quanto o atributo for perfeito e seu valor mínimo quando o atributo for inútil.

Utilizando teoria da informação

- Em geral, se as respostas possíveis v_i tem probabilidade $P(v_i)$, então o conteúdo de informação ***I*** da resposta real é dado por:

Conteúdo de Informação (Entropia):

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

- No caso do lançamento de uma moeda imparcial:
- $I(1/2,1/2) = - 1/2 \log_2 1/2 - 1/2 \log_2 1/2 =$ **1 bit**

- **Entropia é uma medida da incerteza de uma variável randômica; aquisição de informação corresponde a uma redução na entropia.**

Utilizando teoria da informação

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

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

No exemplo do restaurante temos n=p=6, portanto precisamos de 1 bit de informação no total.

Ganho de Informação

- Dado um atributo A, podemos medir quantas informações ainda precisamos *depois* deste atributo;
  - Qqr atributo A divide o conjunto de treinamento E em subconjuntos $E_1, \ldots, E_v$ *de acordo com seus valores para A, onde A pode ter v valores distintos.*
  - Cada subconjunto E_i tem p_i exemplos positivos e n_i exemplos negativos;
  - Assim seguindo essa ramificação, precisaremos de

$$I(p_i/(p_i + n_i), n_i/(p_i + n_i))$$

bits de informação para responder a pergunta.

Ganho de informação

- Um exemplo escolhido ao acaso a partir do conjunto de treinamento tem o i-esimo valor para o atributo com probabilidade $(p_i + n_i)/(p+n)$ (“peso do atributo”).
- e assim, em média, depois de testar o atributo A, temos:

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

bits de informação para classificar o exemplo...

Ganho de informação

- Um exemplo escolhido ao acaso a partir do conjunto de treinamento tem o i-esimo valor para o atributo com probabilidade $(p_i + n_i)/(p+n)$ (“peso do atributo”).
- e assim, em média, depois de testar o atributo A, temos:

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

bits de informação para classificar o exemplo...

“Dentre os exemplos deste atributo, qual é o grau de discernimento dele.”

Ganho de informação

- O ganho de informação a partir do teste deste atributo é a diferença entre o *requisito de informações original* e o *novo requisito*:

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

- A heurística é então escolher o atributo com o maior valor de ganho de informação (IG).

Ganho de informação

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

Considerando os atributos *Patrons* e *Type*, e os outros tb:

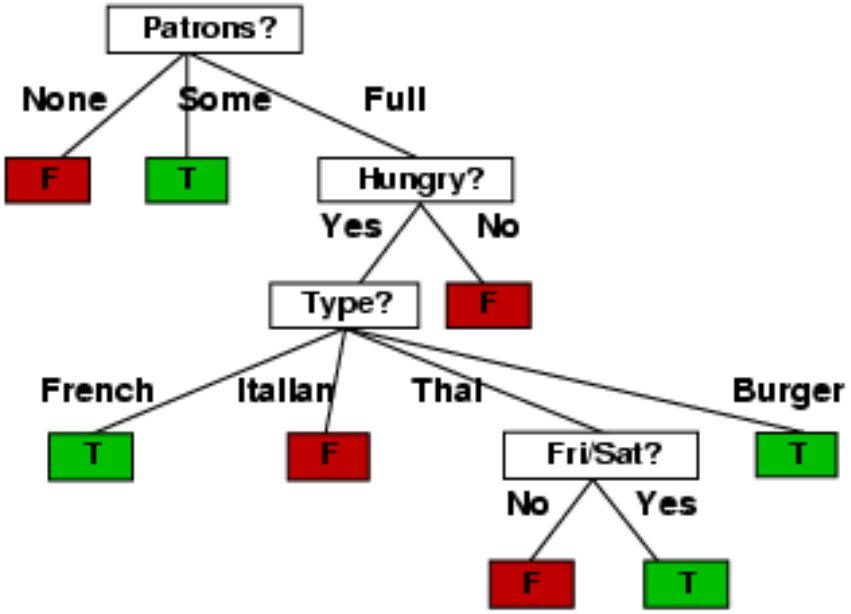
$$IG(Patrons) = 1 - [\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I(\frac{2}{6}, \frac{4}{6})] = .0541 \text{ bits}$$

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

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

Resolvendo o exemplo

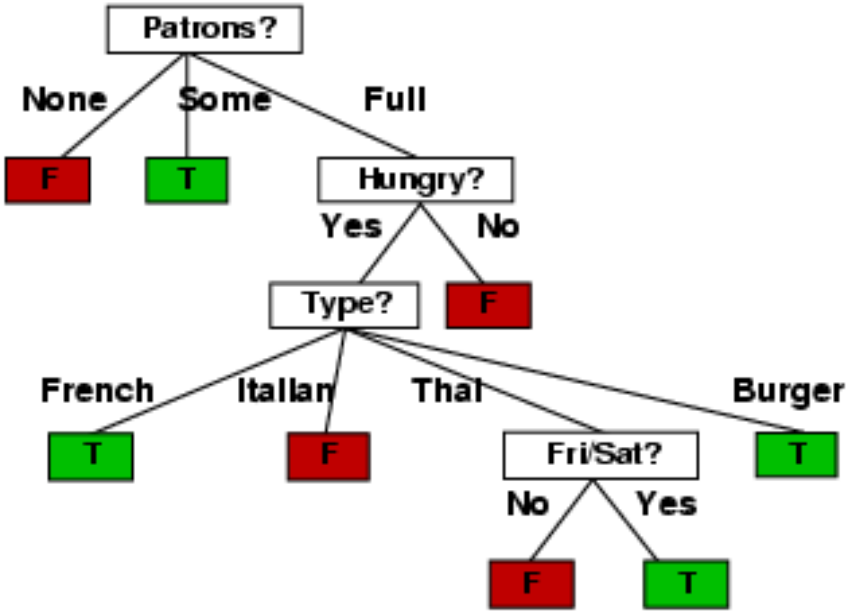
- Árvore de decisão aprendida dos 12 exemplos:



- Substacialmente mais simples do que a árvore “verdadeira”;
- Não há nenhuma razão para uma solução mais complexa (e.g incluindo *Rain* e *Res*), pois todos os exemplos já foram classificados.

Resolvendo o exemplo

- Árvore de decisão aprendida dos 12 exemplos:



- Com mais exemplos seria possível induzir uma árvore mais semelhante à árvore original;
- Esta árvore nunca viu um exemplo de espera 0-10
  - portanto, pode cometer um engano...
  - Como avaliar o desempenho do algoritmo?

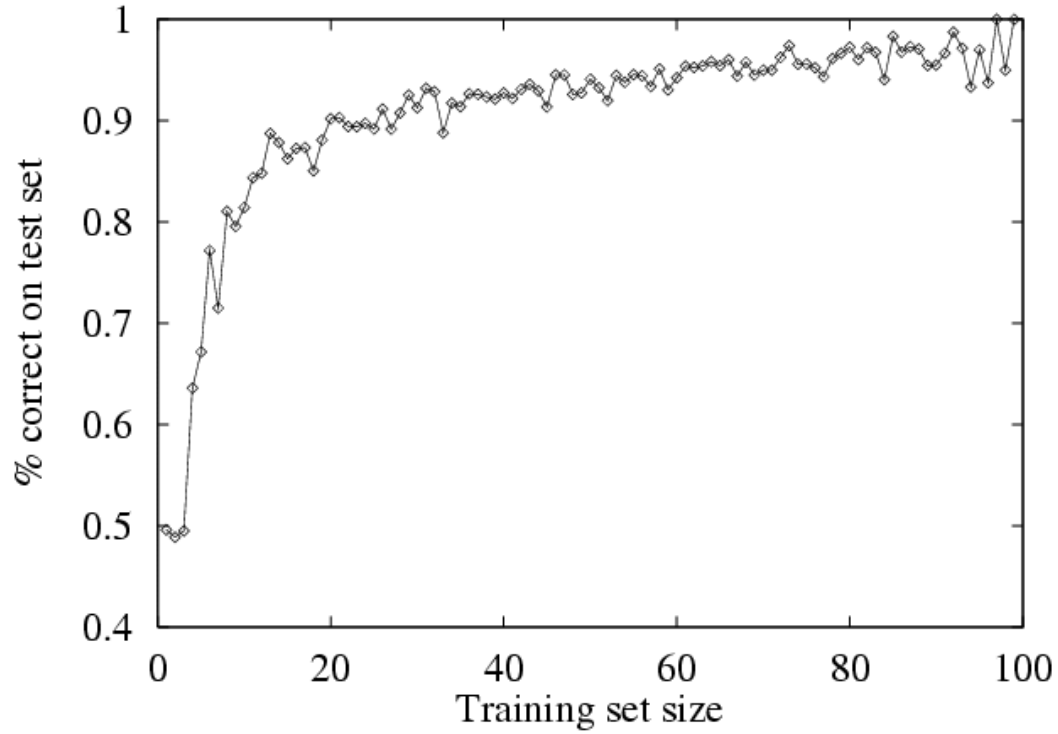
Avaliação de desempenho

- Como sabemos que $h \approx f$?
  1. Um algoritmo de aprendizagem é bom se produz hipóteses que fazem um bom trabalho de previsão de classificações de exemplos não vistos

1. Método de validação:
   1. Coletar um grande número de exemplos;
   2. Dividi-lo em **conjunto de treinamento** e **conjunto de teste;**
   3. Aplicar o algoritmo de aprendizagem ao conjunto de treinamento, gerando uma hipótese ***h***
   4. Medir a porcentagem de exemplos no conjunto de teste que são corretamente classificados por ***h***
   - Repetir as etapas 1 a 4 para diferentes tamanhos de conjuntos de treinamento e diferentes conjuntos de teste de cada tamanho selecionados aleatoriamente

Avaliação de desempenho

- Experimente *h* em novos conjuntos de teste.
- Curva de aprendizagem = % de classificações corretas sobre o conjunto de teste como uma função do tamanho do conjunto de treinamento



Árvores de Decisão – Critérios de Parada

- Totalidade (ou alternativamente, a maioria) do exemplos do nó pertencem a mesma classe
- Profundidade máxima para o nó
- Número mínimo de exemplos no nó
- Ganho pouco significativo no critério de separação
  - Obs.: valores definidos como *parâmetros* do aprendizado

Árvores de Decisão - Exemplo

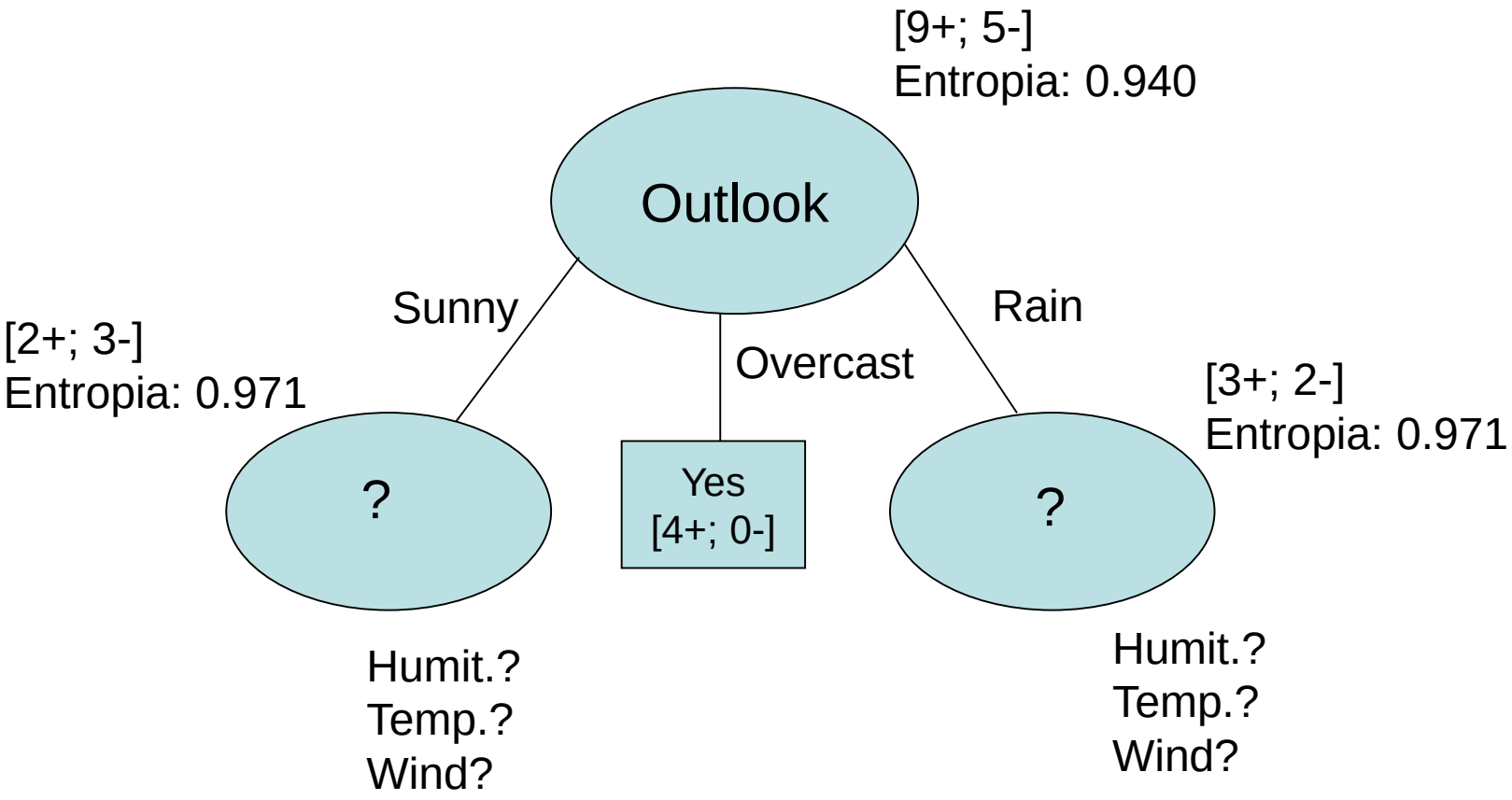
- Raíz: [9+; 5-]
  - Entropia = - 9/14*$\log_2$(9/14) - 5/14*$\log_2$(5/14) = 0.940
- Considerando teste com atributo Outlook
  - Outlook = Sunny: [2+;3-]
    - Entropia = - 2/5*$\log_2$(2/5) - 3/5*$\log_2$(3/5) = 0.971
  - Outlook = Overcast: [4+;0-]
    - Entropia = - 4/4*$\log_2$(4/4) - 0/4*$\log_2$(0/4) = 0.0
  - Outlook = Rain: [3+;2-]
    - Entropia = - 3/5*$\log_2$(3/5) - 2/5*$\log_2$(2/5) = 0.971
  - Média: 5/14*0.971 + 4/14*0.0 + 5/14*0.971 = 0.694
  - Ganho de Informação: 0.940 - 0.694 = 0.246

Árvores de Decisão - Exemplo

- Considerando os outros atributos:
  - Ganho(Outlook, D) = 0.246
  - Ganho(Humit., D) = 0.151
  - Ganho(Wind, D) = 0.048
  - Ganho(Temp., D) = 0.029

  - Atributo Outlook é o escolhido na raíz

Árvores de Decisão - Exemplo



Árvores de Decisão
- Poda

- Árvores de decisão podem super-ajustar os dados de treinamento (*overfitting*)
  - Sempre é possível crescer a árvore de forma a classificar corretamente todos os dados de treinamento
  - Mas isto não quer dizer que a arvore conseguiu generalizar bem para novos dados
- Overfitting torna-se mais provável a medida que o espaço de hipóteses e o número de atributos de entradas crescem e menos provável a medida que nós aumentamos o número de exemplos de treinamento.

Árvores de Decisão - Poda

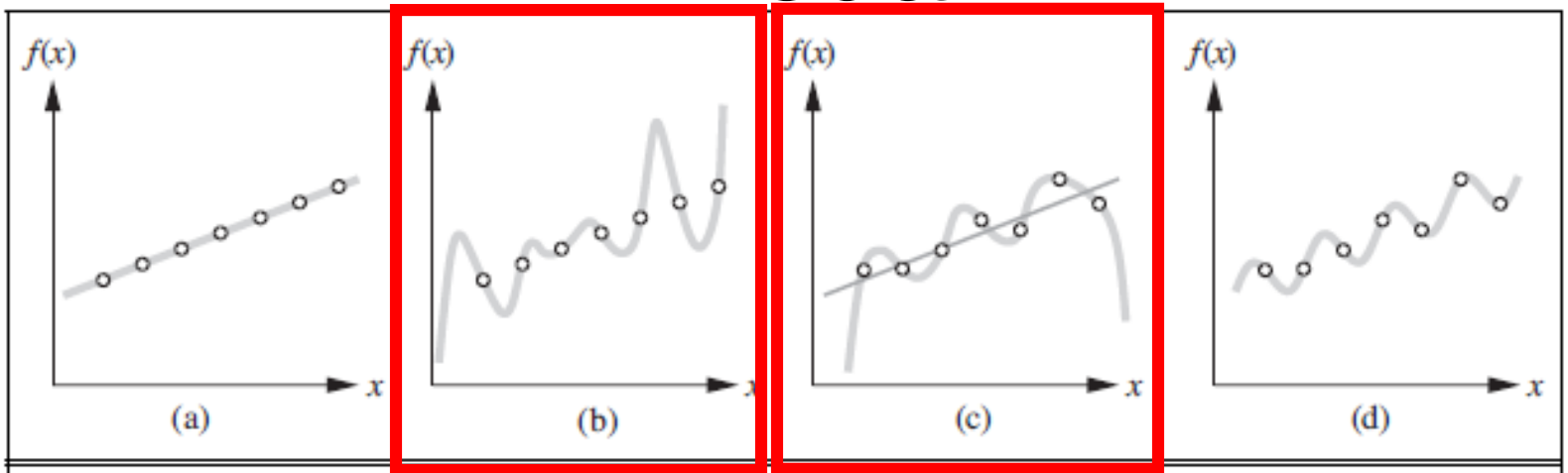


Figure 18.1 (a) Example $(x, f(x))$ pairs and a consistent, linear hypothesis. (b) A consistent, degree-7 polynomial hypothesis for the same data set. (c) A different data set, which admits an exact degree-6 polynomial fit or an approximate linear fit. (d) A simple, exact sinusoidal fit to the same data set.

- Ná prática, a árvore é *podada*, i.e., nós desnecessários são cortados

Árvores de Decisão
- Poda por validação

- Procedimento:
  - Passo 1: Separe um *conjunto de validação* diferente do conjunto de treinamento
  - Passo 2: Gere a árvore de decisão completa usando o conjunto de treinamento
  - Passo 3: Para cada nó da árvore:
    - Passo 3.1: Computar o erro no conjunto de validação obtido se a sub-árvore do nó fosse cortada da árvore
    - Passo 3.2: Efetue a eliminação da sub-árvore, caso o erro de validação se mantenha ou diminua

Árvores de Decisão - Discussão

- Vantagens:
  - Geram modelos dos dados (i.e., método *eager*)
  - Conhecimento interpretável
  - Pouca sensibilidade a atributos irrelevantes
    - Uma vez que implementam seleção de atributos
- Desvantagens:
  - Em geral, menos precisos comparados com algoritmos como redes neurais e SVMs

Árvores de Decisão

- Diferentes versões de algoritmos podem ser encontradas na literatura
  - Algoritmos ID3, C4.5 – versões mais clássicas de AD. Assumem que os dados cabem na memória principal. OK para até centenas de milhares de exemplos.
  - Algoritmos SPRINT, SLIQ – múltiplos scans sequenciais nos dados. OK para até milhões de exemplos.
  - Algoritmo VFDT – no máximo um scan sequencial nos dados. OK para até bilhões de exemplos.