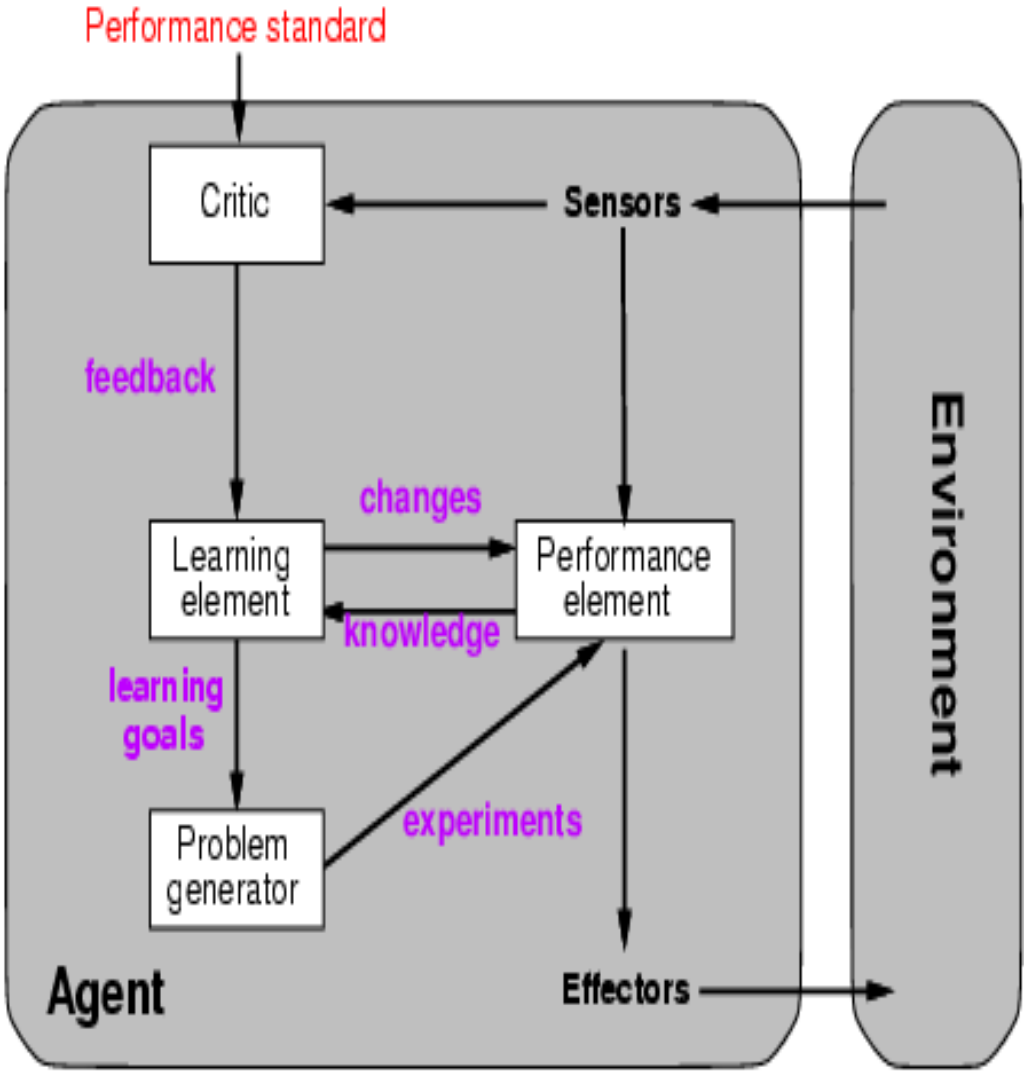
Aprendizagem a partir de observações

Capítulo 18 - Russell & Norvig

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.

Agente de aprendizagem



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

Elemento de aprendizagem:

Busca Alpha-beta, agentes lógico, baseado em utilidade e reflexo simples

- O projeto de um elemento de aprendizagem é afetado por:
  - Os componentes do elemento de desempenho que devem ser aprendidos
    - Função de avaliação, ação, modelo de transição;
  - A realimentação que estará disponível para aprender estes componentes;
    - Vitória/Derrota, Conclusão, Ação correta
  - A representação que será utilizada para os componentes
    - Função linear com pesos, Rede Neural, Rede de Bayes, axiomas

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

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...

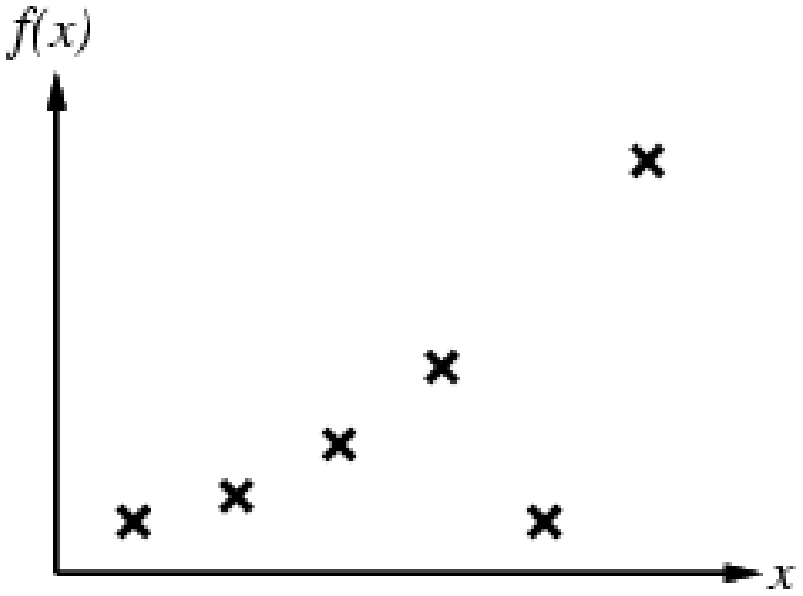
Aprendizagem Indutiva Clássica

- 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

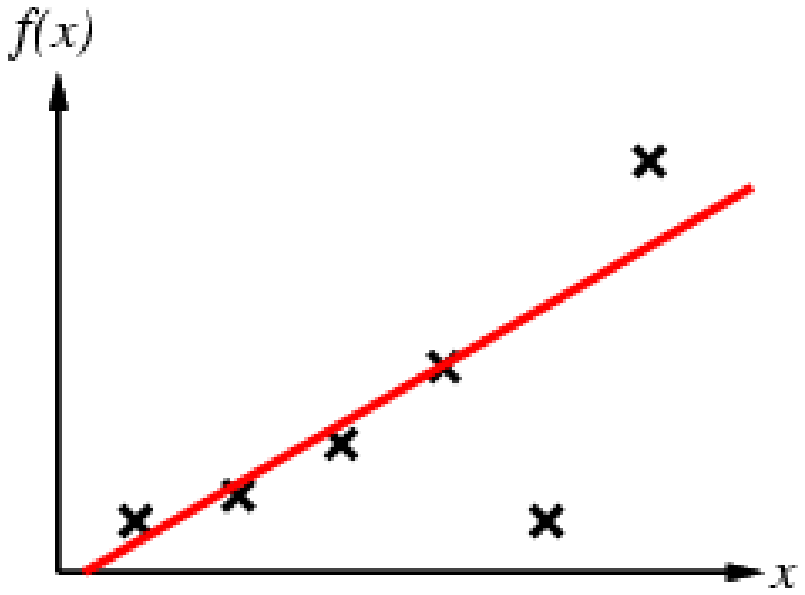
Aprendizagem indutiva

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



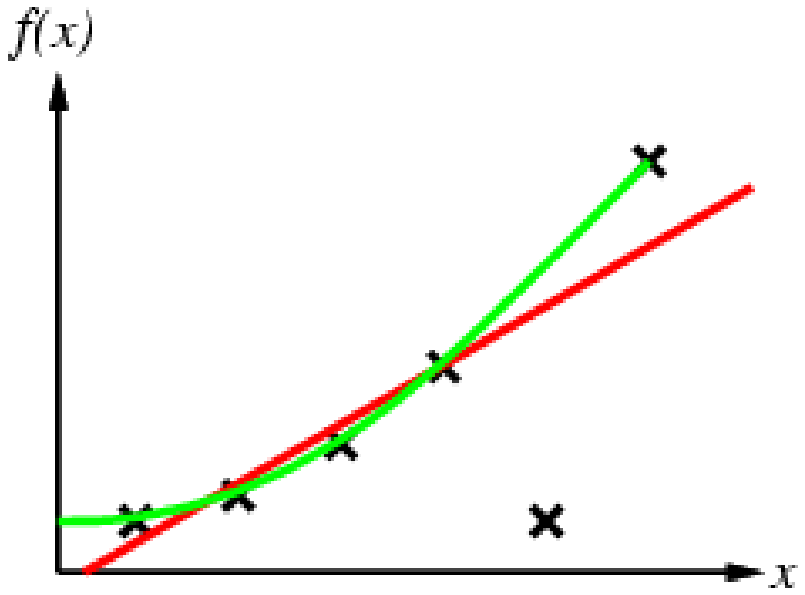
Aprendizagem indutiva

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



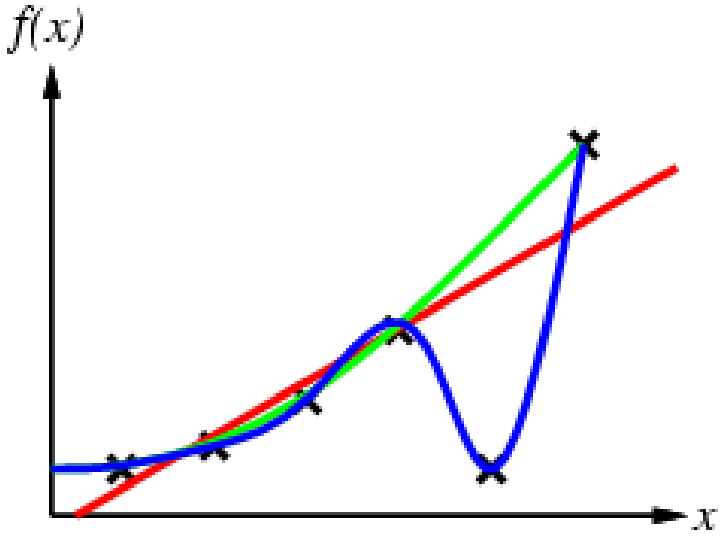
Aprendizagem indutiva

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



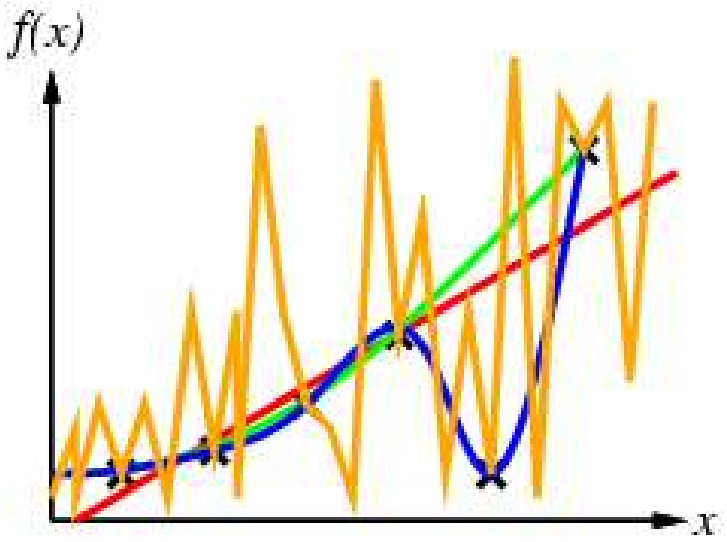
Aprendizagem indutiva

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



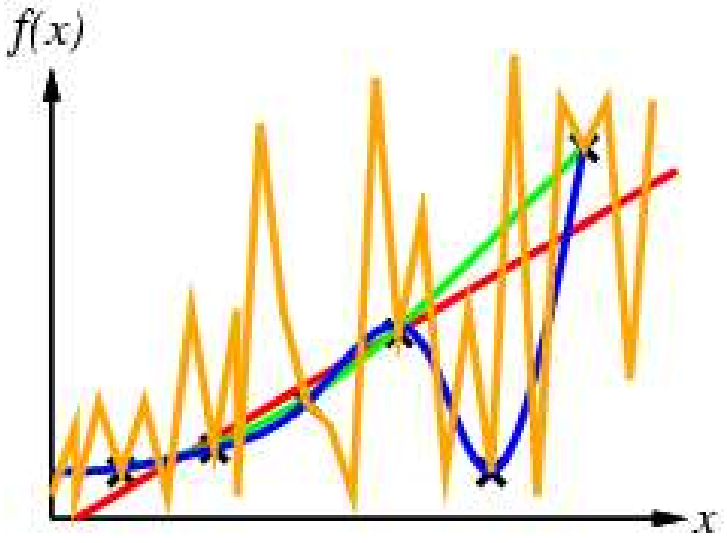
Aprendizagem indutiva

- Construir/ajustar *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!

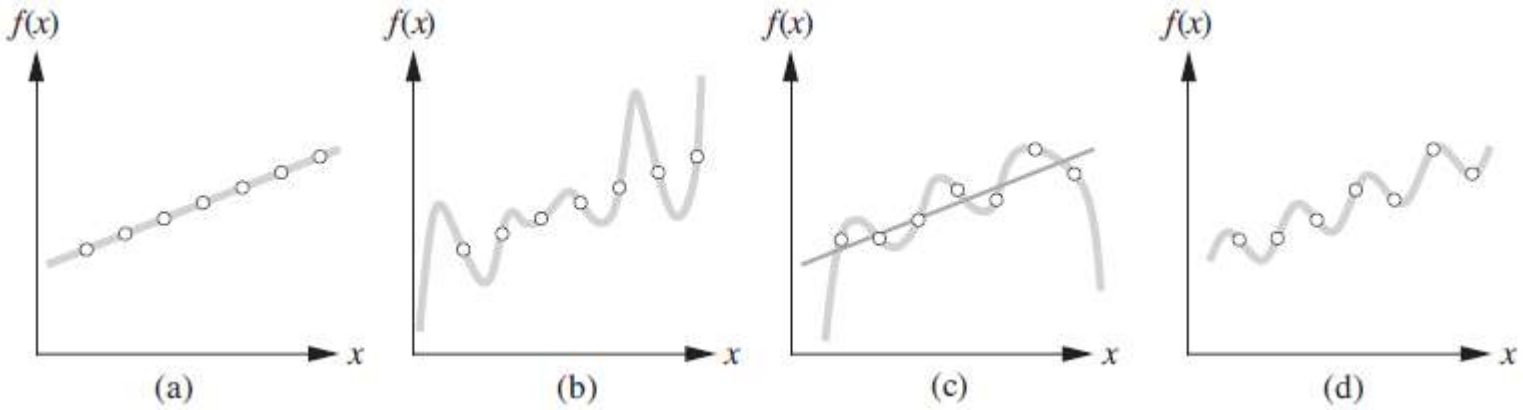
Lâmina de Ockam

- talvez seja melhor, nesse exemplo específico, 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
- compromisso entre a complexidade da hipótese e o grau de ajuste aos dados
- a maior parte do trabalho em aprendizagem se concentra em representações simples!

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:

f(x)= c + a.x *f(x)= c + b.x+ a.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 pode não ser conhecida

Aprendizagem em árvores de decisão

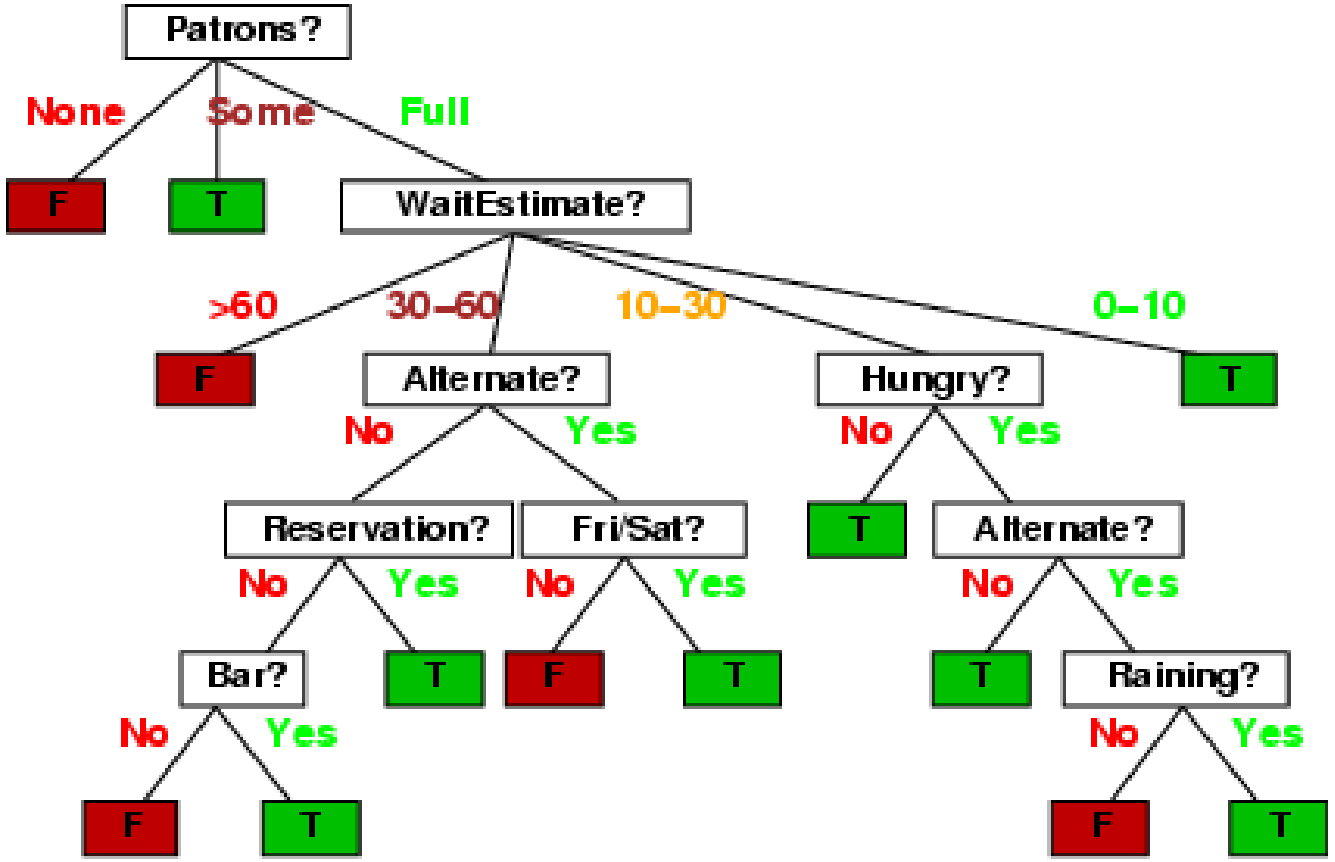
- **Indução de árvores de decisão:** forma mais simples (e mais bem sucedida) de algoritmos de aprendizagem
- **árvore de decisão**: toma como entrada um objeto ou situação descritos por um conjunto de **atributos** e retorna uma decisão -- valor de saída previsto, dada uma entrada

Árvores de decisão

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



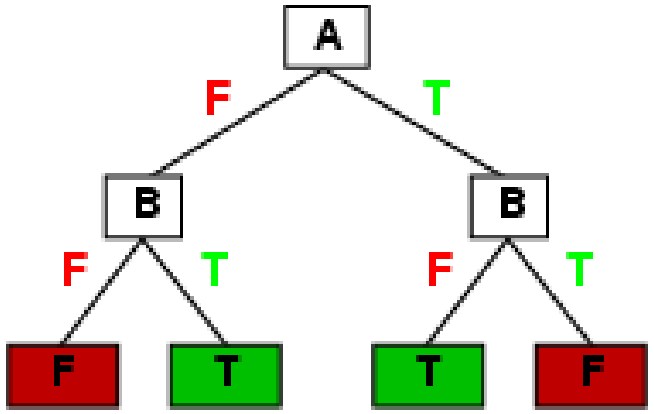
Árvores de decisão

- 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. 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



- Há uma árvore de decisão para qualquer 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 representação de conhecimento que seja eficiente para TODOS os tipos de funções?

Espaço de hipóteses

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

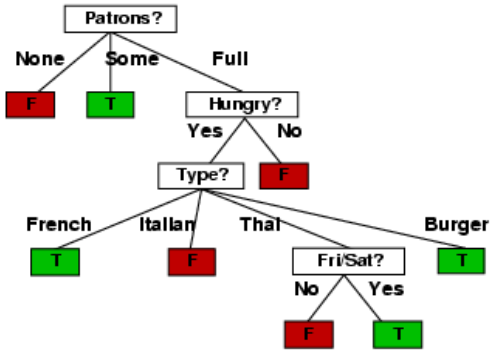
NÃO

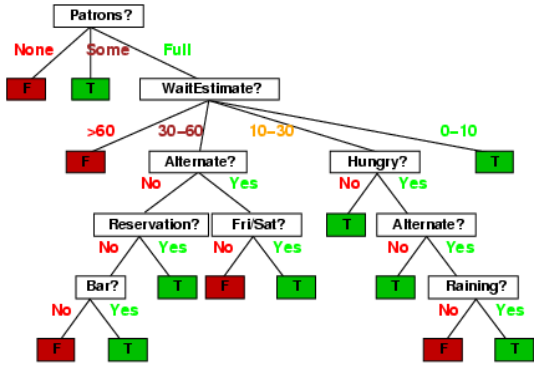
Quantas árvores de decisão distintas existem para n atributos?

= numero de funções booleanas (cada função booleana tem 2^n linhas)

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

- Exemplo: com 6 atributos Booleanos, tem-se 18,446,744,073,709,551,616 árvores distintas!





Induzindo árvores de decisão a partir de exemplos

- Exemplos descritos por valores de atributos (Booleanos, discretos, ou contínuos)
- Exemplos de situações em que eu não esperarei por uma mesa para jantar:

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)

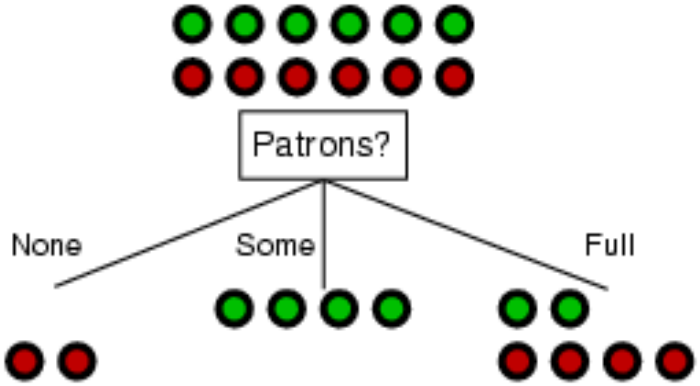
Aprendizagem por árvores de decisão

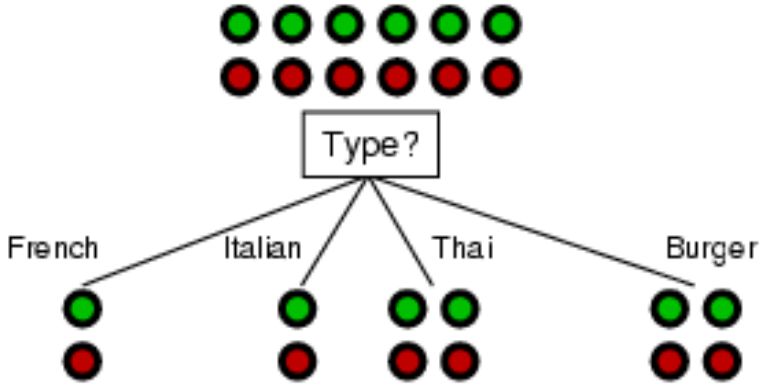
- Objetivo: encontrar a menor árvore que seja consistente com o número de exemplos
- Ideia: (recursivamente) encontre o atributo “mais significante” como raiz da subárvore

```
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

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



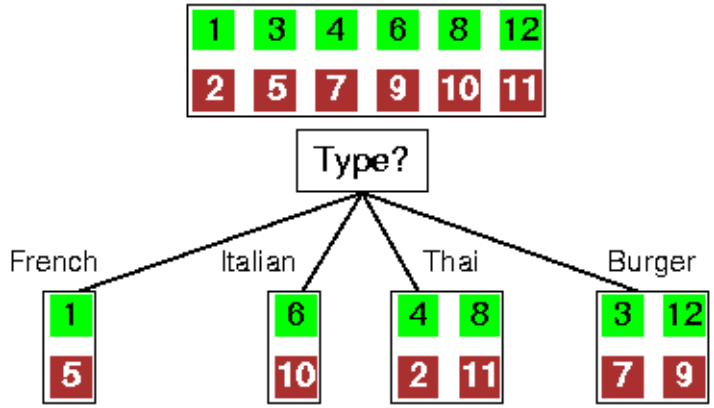


- *Patrons?* é um atributo melhor do que *Type* para ser raiz.

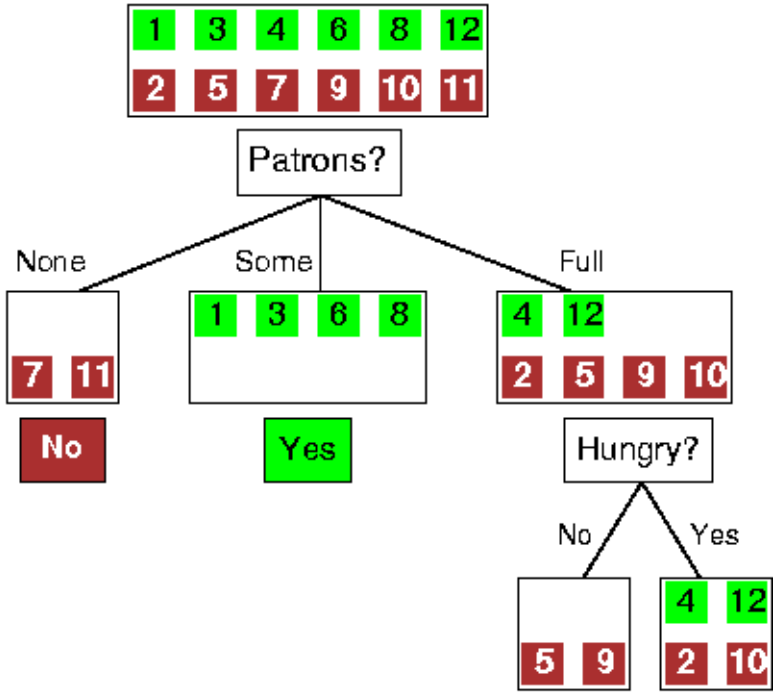
Algoritmo (informal)

- 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)

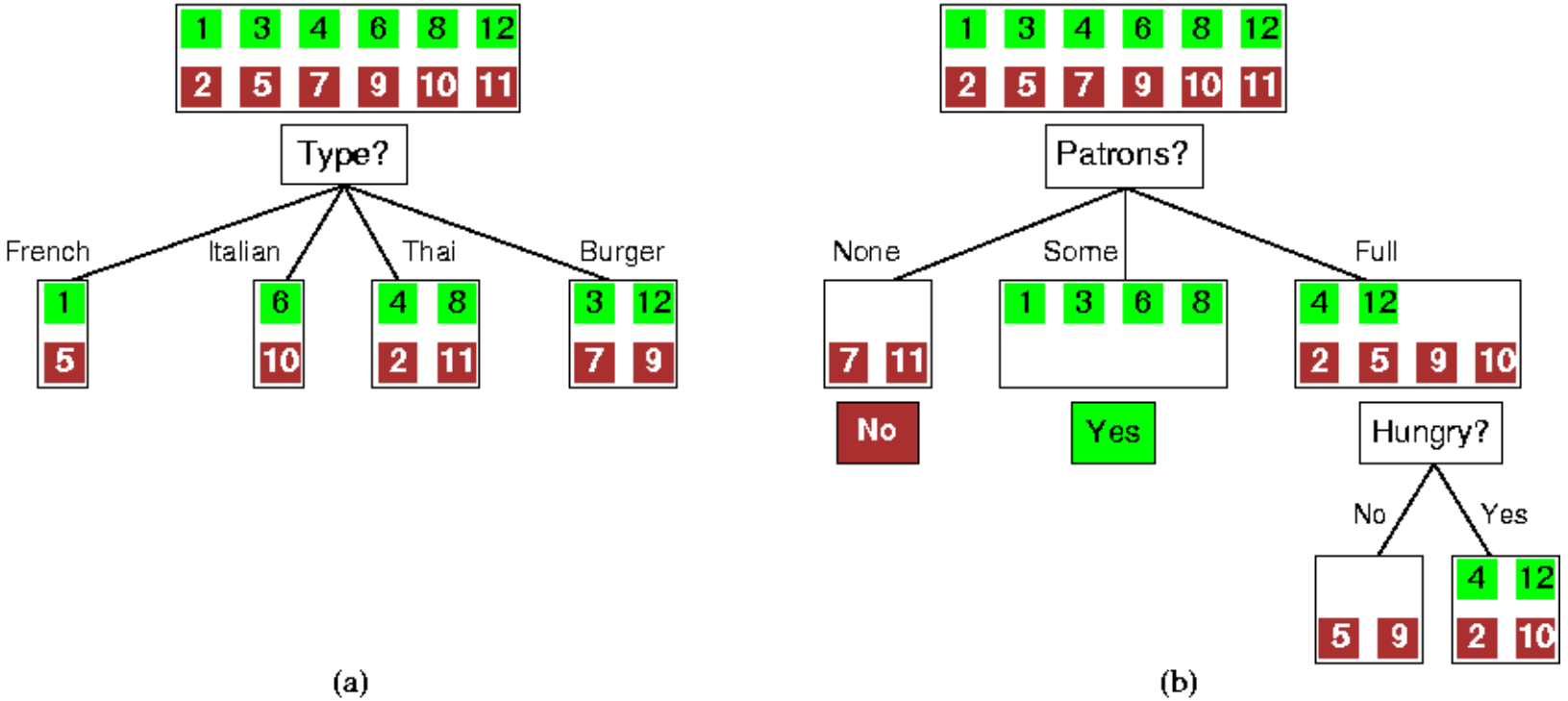
- Se não resta nenhum atributo, mas há exemplos positivos e negativos, temos um problema. Isso quer dizer que esses exemplos tem exatamente a mesma descrição, mas classificações diferentes. Isso acontece quando
  - alguns dados estão incorretos; dizemos que existe **ruído** nos dados;
  - os atributos 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.



Como definir o que é um atributo melhor ?

- 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

- Medida utilizada em `Choose-Attribute` no algoritmo DTL
- A teoria da informação mede o conteúdo de informação em bits.
  - um bit é o suficiente para responder a uma pergunta sim/não sobre a qual não se tem nenhuma ideia a respeito (como o lançamento de uma moeda)

```
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
```

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**

Utilizando teoria da informação

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

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

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 entrar na árvore de decisão;
  - Um 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 ésimo 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 ésimo 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:

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

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

bits de informação para classificar o exemplo ...

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*:

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

- A heurística é então escolher o atributo com o maior valor de Ganho de Informação (*IG*).

Ganho de informação

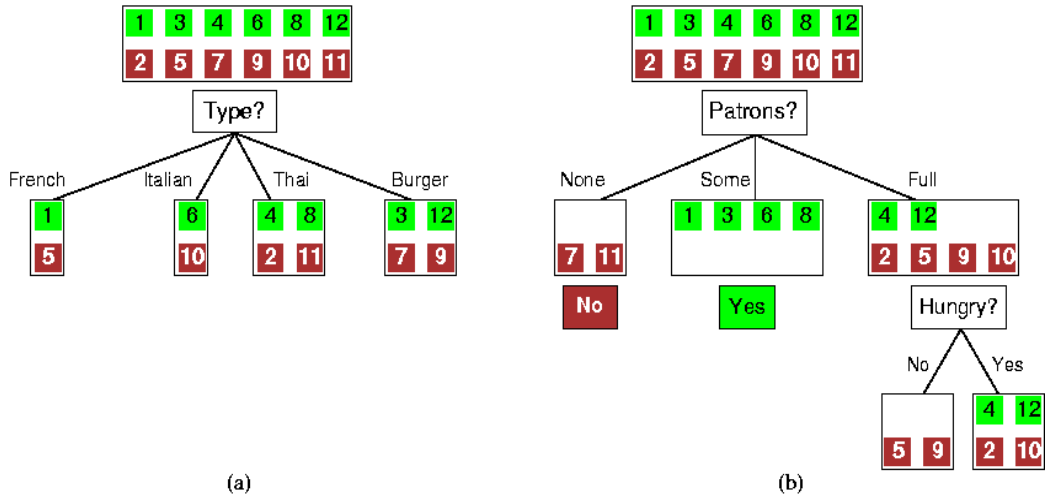
Para o conjunto de treinamento, $p = n = 6$, $I(6/12, 6/12) = 1$ bit

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

$$IG(Patrons) = 1 - \left[\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I\left(\frac{2}{6}, \frac{4}{6}\right)\right] = 0.541 \text{ bits}$$

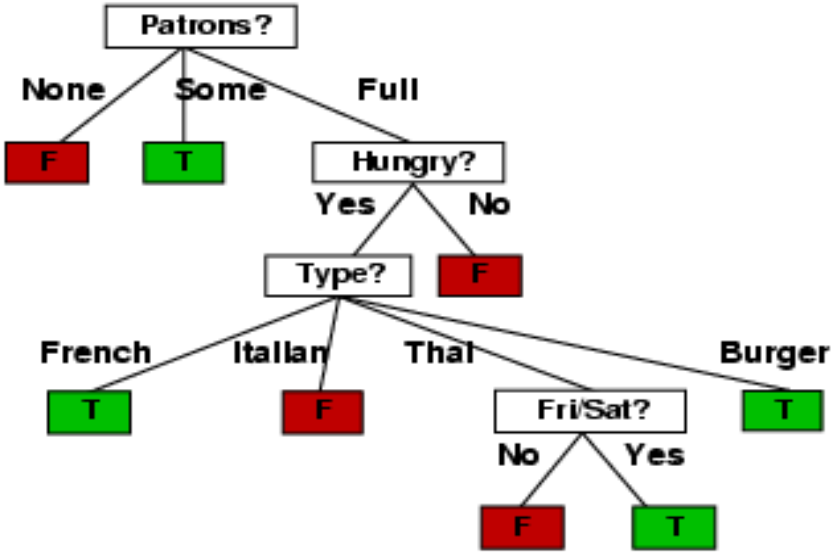
$$IG(Type) = 1 - \left[\frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right)\right] = 0 \text{ bits}$$

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

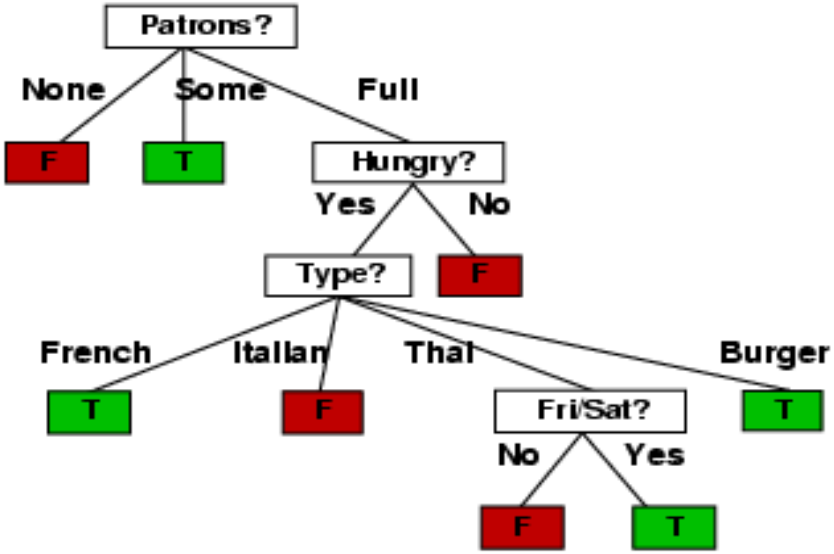
- Árvore de decisão aprendida dos 12 exemplos:



- Substancialmente mais simples do que a árvore completa;
- 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 enganos
- Como avaliar o desempenho do algoritmo?

Avaliação de desempenho

Como sabemos que $h \approx f$?

Um algoritmo de aprendizagem é bom se produz hipóteses que fazem um bom trabalho de previsão de classificações de exemplos não vistos

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;***
5) 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