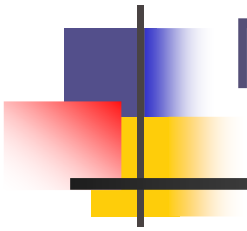
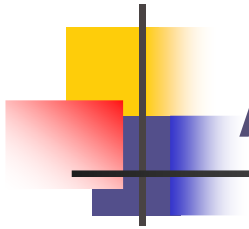


Modelo Geral de Aprendizagem

Profa. Josiane M. Pinheiro Ferreira

Stuart Russel e Peter Norving - “Inteligência Artificial” - cap 18.

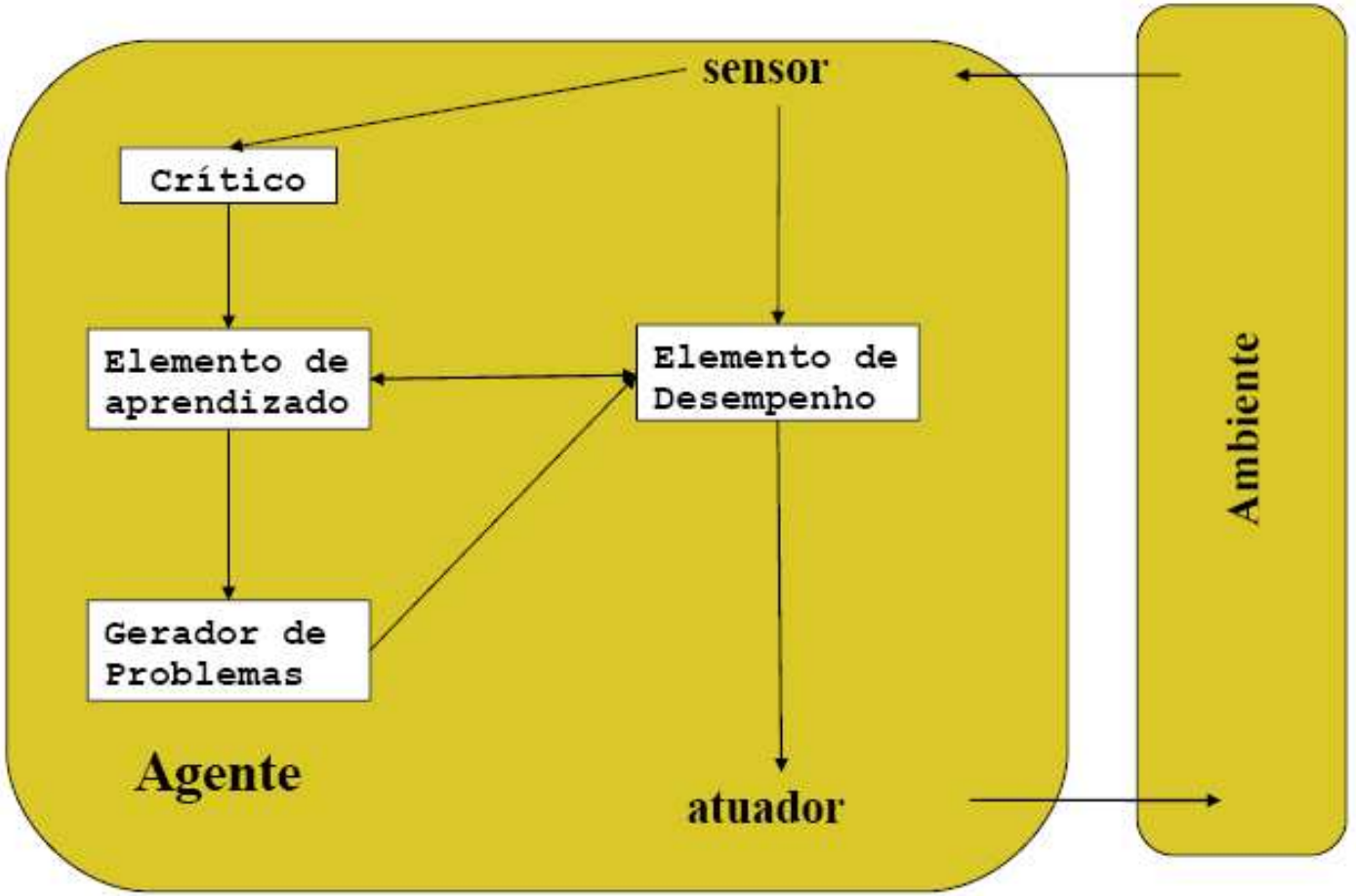
Outubro/2008

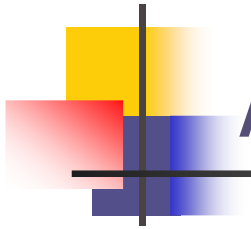


A idéia por trás da aprendizagem

- Percepções
  - Para agir
  - Para melhorar a habilidade do agente para agir no futuro
- Aprendizagem ocorre
  - Quando o agente observa suas interações como o mundo e com seus próprios processos de tomada de decisão

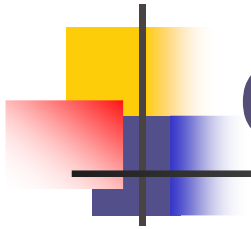
Agentes que aprendem – esquema geral





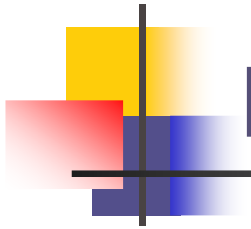
Agente de aprendizagem

- **Elemento de desempenho:** decide que ações executar
- **Elemento de aprendizagem:** modifica o elemento de desempenho para que ele tome decisões melhores
  - Existe uma grande variedade deles
  - Seu projeto é afetado por três questões importantes
    - Os **componentes** que devem ser aprendidos
    - A **realimentação** que estará disponível para aprender esses componentes
    - A **representação** que será usada para os componentes



Os componentes

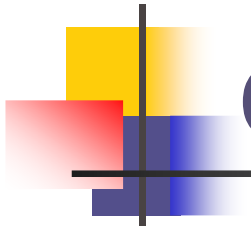
1. Um mapeamento de condições -> ações
2. Um meio para deduzir propriedades do mundo a partir de seqüências de ações
3. Informações sobre o modo como o mundo evolui e sobre o resultados das ações possíveis que o agente pode executar
4. Informações de utilidade indicando a desejabilidade de estados do mundo
5. Informações de valores de ações indicando a desejabilidade de ações
6. Metas que descrevem classes de estados cuja realização maximiza a utilidade



Realimentação

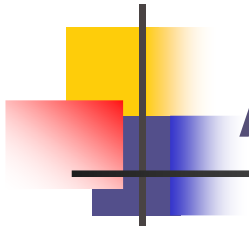
- Cada componente pode ser aprendido a partir da realimentação apropriada

1. Toda vez que o instrutor gritar “Freie!” o agente poderá aprender uma regra de condição-ação
2. Ao ver muitas imagens que lhe dizem ser um ônibus o agente poderá aprender a reconhecê-los
3. Freando bruscamente em uma estrada molhada o agente poderá aprender o resultado de suas ações
4. Se não receber nenhuma gorjeta dos passageiros que foram sacudidos o agente poderá aprender um componente útil de sua função de utilidade



O tipo de realimentação

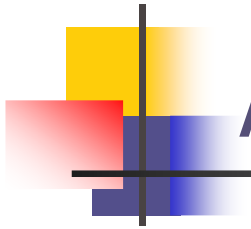
- É o fator mais importante na determinação da natureza do problema de aprendizagem
- Normalmente distinguimos 3 casos:
  - Aprendizagem **supervisionada**
  - Aprendizagem **não supervisionada**
  - Aprendizagem **por reforço**



Aprendizagem supervisionada

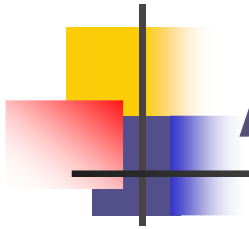
- Envolvem aprendizagem de uma **função** a partir de exemplos de suas entradas e saídas (casos 1, 2 e 3 anteriores)

1. Agente aprende a regra condição-ação
   - Função de estados para uma saída booleana (frear ou não)
2. Agente aprende a reconhecer um ônibus
   - Função a partir de imagens para uma saída booleana (a imagem contém ou não o ônibus)
3. Agente aprende a teoria de frear
   - Função de estados e ações para, por exemplo, a distância de parada



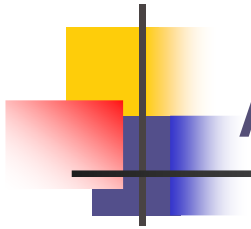
Aprendizagem supervisionada

- Exemplos 1 e 2 o valor correto da saída é dado por um instrutor
- Exemplo 3 saída disponível diretamente a partir das percepções do agente
- Ambientes completamente observáveis
- Ambientes parcialmente observáveis



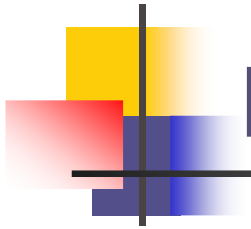
Aprendizagem não-supervisionada

- Envolve a aprendizagem de padrões na entrada, quando não são fornecidos valores de saída específicos
  - Agente pode desenvolver conceitos de "dias de tráfego bom" e "dias de tráfego ruim"
  - Mas sem saber determinar uma denominação para estes conceitos
- Agente não pode aprender o que fazer (sozinho)
  - Não tem nenhuma informação sobre uma ação correta ou estado desejável



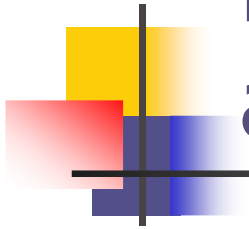
Aprendizagem por reforço

- O agente deve aprender a partir do reforço (recompensa)
- Por exemplo:
  - Falta de uma gorjeta no fim da viagem
  - Multa pesada por bater na traseira de outro carro
- Fornecem ao agente alguma indicação de que seu comportamento é (in)desejável
- Inclui o subproblema de aprender como o ambiente funciona



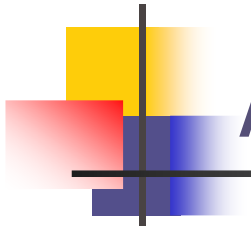
Representação da informações

- Os componentes podem ser representados com quaisquer um dos esquemas de representação
  - Sentenças lógicas proposicionais
  - Sentenças lógicas de primeira ordem
  - Redes Semânticas
  - Frames
  - Regras de produção
  - Descrições probabilísticas (redes bayesianas)
  - ...



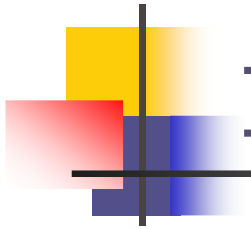
Disponibilidade de conhecimento anterior

- Na maioria das pesquisas, o agente começa sem nenhum conhecimento
  - Acesso apenas aos exemplos apresentados por experiência
- É um caso especial mas não o caso geral
  - Grande parte da aprendizagem humana ocorre no contexto de um bom conhecimento prático
- Conhecimento anterior pode ajudar muito na aprendizagem



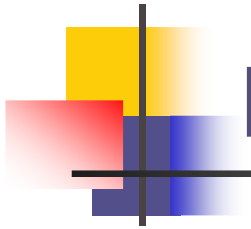
Aprendizagem indutiva

- Um algoritmo para aprendizagem supervisionada determinística
- Recebe como entrada
  - Entradas específicas
  - O valor correto da função para estas entradas
- Deve tentar recuperar a **função desconhecida** ou uma aproximação
- Um exemplo é um par (x, $f(x)$), onde:
  - x é a entrada e
  - $f(x)$ é a saída da função aplicada a x



Inferência indutiva pura (indução)

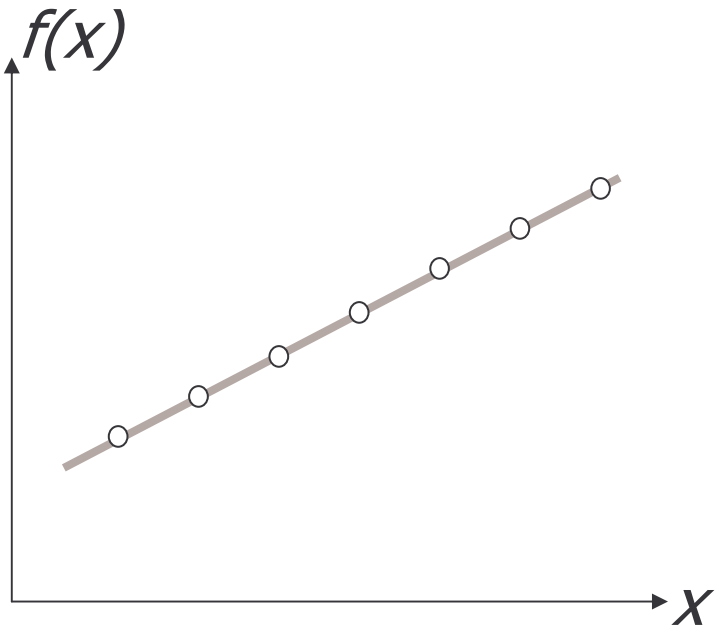
- **Dada uma coleção de exemplos de *f*, retornar uma função *h* que se aproxime de *f***
- A função *h* é chamada **hipótese**
  - Não é fácil saber se uma *h* específica é uma boa aproximação de *f*
  - Uma boa *h* irá **generalizar** bem – prever corretamente exemplos ainda não vistos
  - Esse é o **problema da indução fundamental**



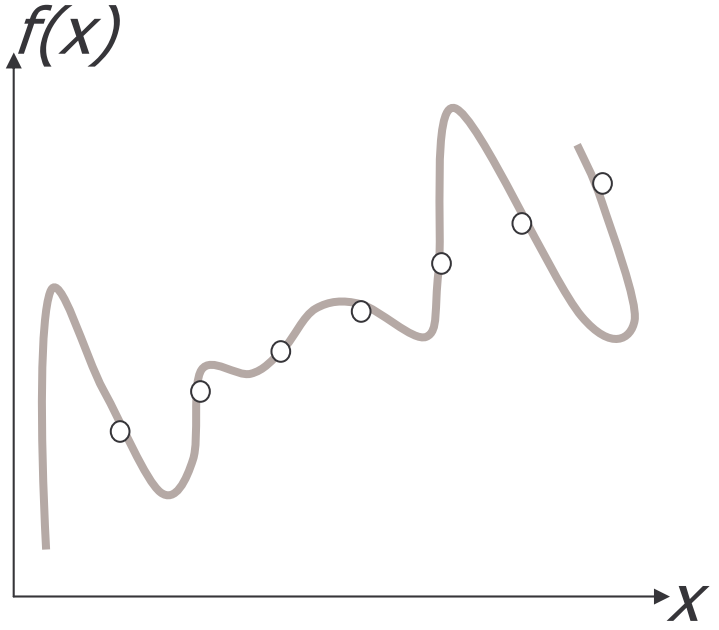
Exemplo de indução

- Ajustar uma função de uma única variável a alguns pontos de dados
- Exemplos são pares (*x, f(x)*) de números reais
- **Espaço de hipóteses (H)**
  - O conjunto de polinômios de grau máximo k
    - $3x^2+2$ ou $x^{17} – 4x^3$

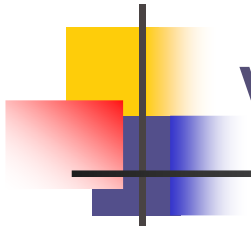
Exemplos de hipóteses I



Uma hipótese linear consistente



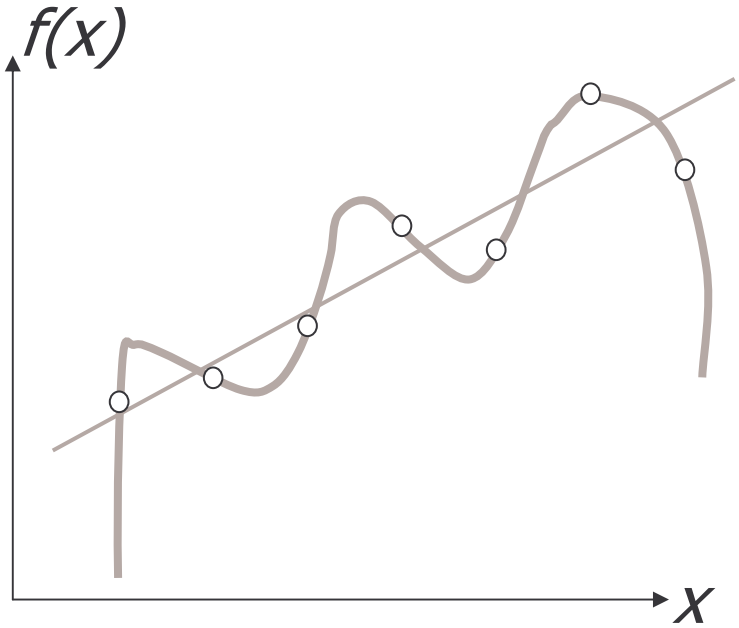
Uma hipótese de polinômio de grau 7 consistente



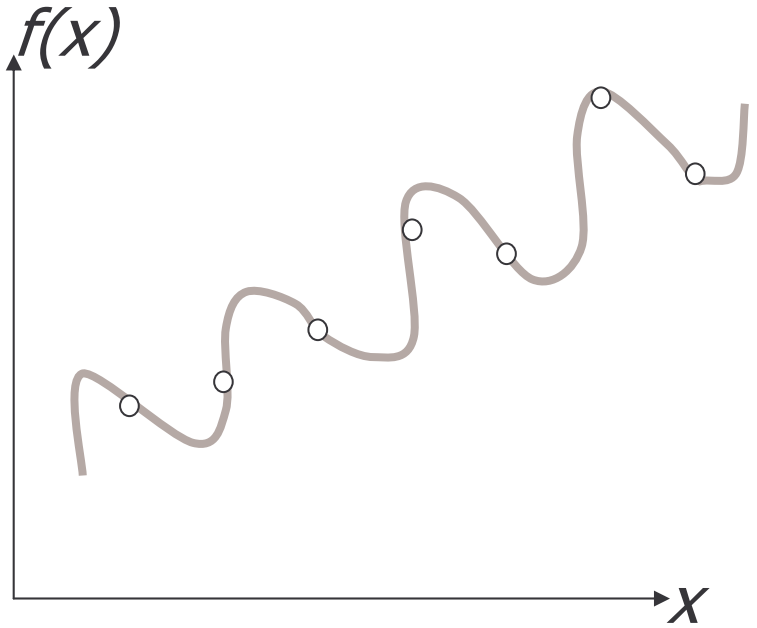
Várias hipóteses consistentes

- Qual hipótese escolher?
- **Lâmina de Ockham**
  - "Prefira a hipótese *mais simples* consistente com os dados"
- Hipóteses que não são simples deixam de extrair algum padrão dos dados
- Nem sempre é fácil determinar qual é a *mais simples*
- Um polinômio de grau 1 é mais simples do que um polinômio de grau 12

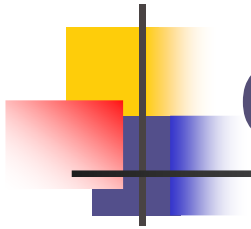
Exemplos de hipóteses II



Um ajuste de polinômio de grau 6 exato ou um ajuste linear aproximado

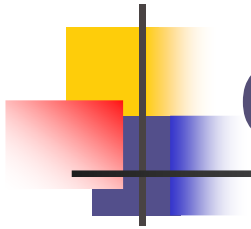


Um ajuste senoidal exato para o mesmo conjunto de dados



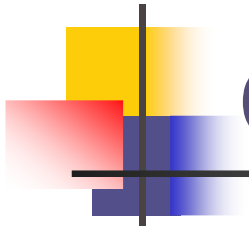
O ajuste dos dados

- Não existe nenhuma linha reta consistente para os dados
- Exige um polinômio de grau 6, com 7 parâmetros, para um ajuste exato
  - O polinômio tem tantos parâmetros quanto os pontos de dados
  - Não encontra um padrão, porém não generaliza bem
  - Às vezes é melhor ajustar uma hipótese que não seja completamente consistente, mas faça previsões razoáveis
  - A verdadeira função pode não ser determinística (entradas verdadeiras podem não ser completamente observadas)



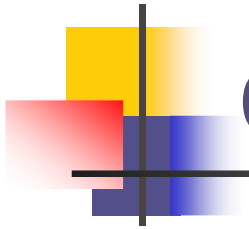
O espaço de hipóteses escolhido

- Os dados podem ser ajustados exatamente por uma função simples do tipo $ax + b + c \text{ sen } x$
- Um espaço de hipóteses de polinômios de grau finito não representa funções senoidais com precisão
  - Um agente com este espaço de hipóteses não será capaz de aprender a partir dos dados
- Se o espaço de hipóteses **contém** a função verdadeira
  - Temos um problema de aprendizagem **realizável**
  - Caso contrário, temos um problema de aprendizagem **irrealizável**



O espaço de hipóteses escolhido

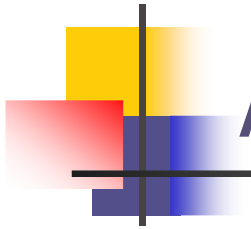
- Nem sempre podemos dizer se o problema é realizável ou não pois a fç verdadeira é desconhecida
  - Solução: conhecimento anterior
- Porque não utilizar o maior espaço de estados possíveis?
- Porque não fazer de **H** o espaço de estados de todas as máquinas de Turing?
  - Toda função computável pode ser representada por uma máquina de Turing



Complexidade computacional

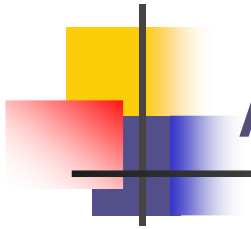
- (Expressividade de **H**) X (complexidade de encontrar *h*'s simples e consistentes dentro de **H**)
  - Ajuste de linhas retas é fácil
  - Ajuste de polinômios é mais difícil
  - Ajuste de máquinas de Turing é muito difícil
  - Determinar se uma máquina de Turing é consistente com os dados não é nem mesmo **decidível**

- **H** simples = *h* resultantes mais simples de usar
  - É mais **rápido** calcular $h(x)$ quando ela é uma fç linear do que quando ela é um programa de uma mq de Turing



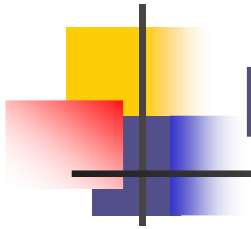
Aprendizagem em árvores de decisão

- Uma árvore de decisão
  - Toma como entrada um objeto ou situação descrito por um conjunto de **atributos**
  - Retorna uma "decisão" – o valor de saída previsto de acordo com a entrada
  - Atributos de entrada e o valor de saída podem ser discretos ou contínuos
  - **Classificação** – aprendizagem de uma fç de valores discretos
  - **Regressão** – aprendizagem de uma fç contínua



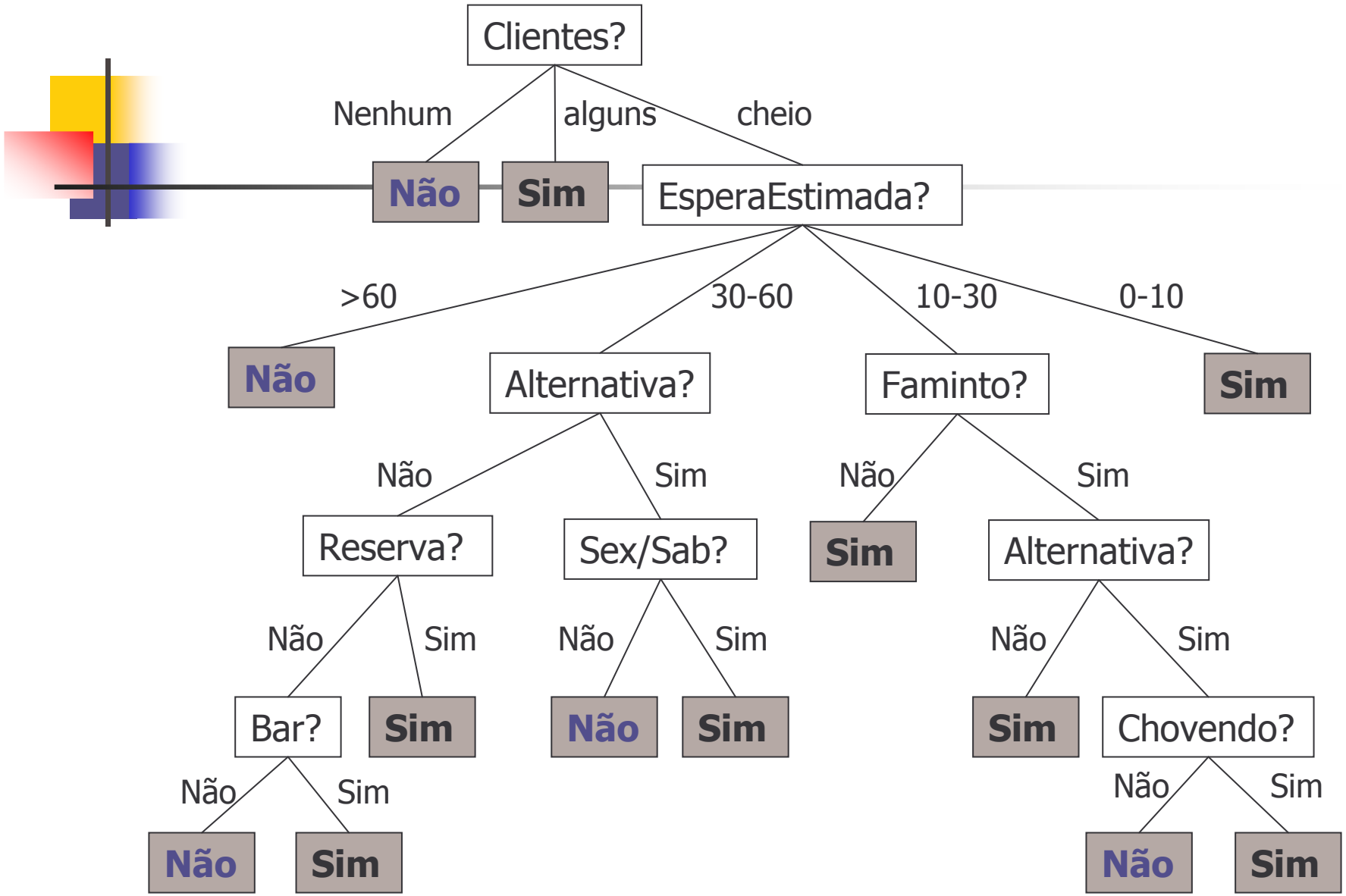
Árvores de decisão

- Alcança a decisão executando uma seqüência de testes
- Cada nó interno da árvore corresponde a um teste do valor de uma propriedade
- As ramificações do nó são os valores possíveis do teste
- Cada nó folha especifica o valor de saída se aquela folha for alcançada
- Representação bastante natural para os seres humanos



Exemplo de árvore de decisão

- Esperar ou não uma mesa em um restaurante?
- Objetivo: aprender uma definição para o **predicado de objetivo** *vaiEsperar*
- Atributos disponíveis
  - Alternativa: há um outro restaurante apropriado por perto?
  - Bar: o restaurante tem uma área de bar confortável para esperar?
  - Sex/Sab: verdadeiro às sextas e sábados
  - Faminto: Estamos com fome?
  - Clientes: Quantas pessoas estão no restaurante?
  - Preço: A faixa de preços do restaurante
  - Chovendo: Está chovendo do lado de fora?
  - Reserva: Fizemos uma reserva?
  - Tipo: Qual o tipo do restaurante?
  - Espera estimada: O tempo de espera estimado pelo o gerente


Clientes?
Nenhum
alguns
cheio
Não
Sim
EsperaEstimada?
>60
30-60
10-30
0-10
Não
Alternativa?
Faminto?
Sim
Não
Sim
Não
Sim
Reserva?
Sex/Sab?
Sim
Alternativa?
Não
Sim
Não
Sim
Não
Sim
Bar?
Sim
Não
Sim
Sim
Chovendo?
Não
Sim
Não
Sim
Não
Sim

Expressividade de árvores de decisão

- Qualquer hipótese de árvore de decisão específica para o predicado meta *VaiEsperar* pode ser vista como uma asserção da forma:

$$\forall s \; vaiEsperar(s) \Leftrightarrow (P_1(s) \vee P_2(s) \vee \ldots \vee P_n(s))$$

- Onde cada condição $P_i(s)$ é uma conjunção de testes que correspondem a um caminho da raiz até uma folha da árvore com resultado positivo

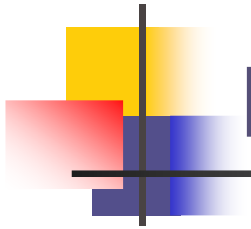
$$(\text{clientes} = \text{alguns}) \vee$$
$$(\text{clientes} = \text{cheio} \wedge \text{esperaEstimada} = 0\text{-}10) \vee$$
$$(\text{clientes} = \text{cheio} \wedge \text{esperaEstimada} = 10\text{-}30 \wedge \text{faminto} = \text{não}) \vee$$
$$(\text{clientes} = \text{cheio} \wedge \text{esperaEstimada} = 10\text{-}30 \wedge \text{faminto} = \text{sim} \wedge \text{alternativa} = \text{não}) \vee$$
$$(\text{clientes} = \text{cheio} \wedge \text{esperaEstimada} = 10\text{-}30 \wedge \text{faminto} = \text{sim} \wedge \text{alternativa} = \text{sim} \wedge \text{chovendo} = \text{sim}) \vee$$
$$(\text{clientes} = \text{cheio} \wedge \text{esperaEstimada} = 30\text{-}60 \wedge \text{alternativa} = \text{sim} \wedge \text{sex/sab} = \text{sim}) \vee$$
$$(\text{clientes} = \text{cheio} \wedge \text{esperaEstimada} = 30\text{-}60 \wedge \text{alternativa} = \text{não} \wedge \text{reserva} = \text{sim}) \vee$$
$$(\text{clientes} = \text{cheio} \wedge \text{esperaEstimada} = 30\text{-}60 \wedge \text{alternativa} = \text{não} \wedge \text{reserva} = \text{não} \wedge \text{bar} = \text{sim})$$

Expressividade de árvores de decisão

- Uma árvore de decisão
  - Descreve um relacionamento entre o predicado meta
  - E alguma combinação lógica de atributos
- Não podemos utilizar árvores de decisão para representar testes que se referem a dois ou mais objetos diferentes

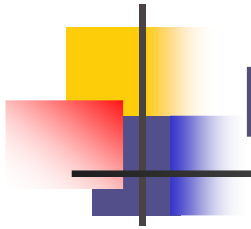
$$\exists\, r_2\ \mathit{Perto}(r_2, r) \wedge \mathrm{Pr}\,\mathit{eço}(r, p) \wedge \mathrm{Pr}\,\mathit{eco}(r_2, p_2) \wedge \mathit{MaisBarato}(p_2, p)$$

- Existe um restaurante mais barato por perto?



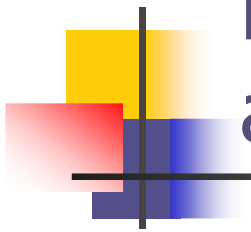
Expressividade de árvores de decisão

- Árvores de decisão são completamente expressivas dentro da classe de linguagens proposicionais
  - Qualquer função booleana pode ser escrita como uma árvore de decisão
- Cada linha da TV = um caminho na árvore
  - Representação exponencialmente grande
  - TV número exponencial de linhas
- Árvore de decisão exponenciais
  - Função paridade
  - Função maioria



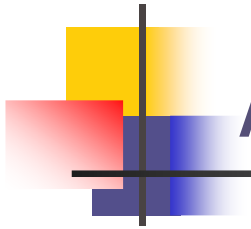
Expressividade de árvores de decisão

- Árvores de decisão servem para alguns tipos de funções e não são boas para outros
- Infelizmente não existe uma espécie de representação que seja eficiente para todos os tipos de funções



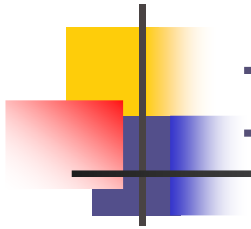
Problemas apropriados para o uso de árvores de decisão

- Instâncias são representadas através de pares atributo-valor
  - cada atributo assume um número pequeno de possíveis valores disjuntos (por exemplo, Temperatura = Quente, Moderado, Frio).
- A função tem valores discretos (Verdadeiro: sim ou Falso: não)
- Permitem descrições disjuntas
- Os dados de treinamento podem conter erros
- Os dados de treinamento podem conter valores de atributo indefinidos



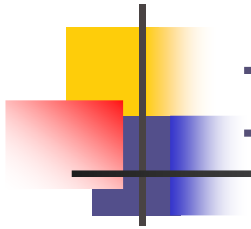
Aplicação de árvores de decisão

- Muitos problemas práticos possuem estas características
- Aprendizado utilizando árvore de decisão já foi aplicado a problemas como:
  - Classificar os pacientes médicos pela doença
  - Causa de maus funcionamentos de equipamentos
  - A probabilidade de candidatos a empréstimo ficarem inadimplentes
  - Entre outros...



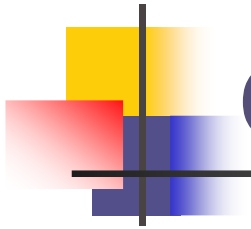
Induzindo árvores de decisão

- Induzindo árvores de decisão por meio de exemplos
  - Um **exemplo** é descrito pelo valor de seus atributos e o valor do **predicado meta**
  - **Valor do predicado meta** => classificação do exemplo
  - Predicado meta = verdadeiro => exemplo positivo
  - Predicado meta = falso => exemplo negativo
  - Um conjunto completo de exemplos é chamado de **conjunto de treinamento**



Induzindo árvores de decisão

- Encontrar uma ad ($h(x)$) que concorde com o conjunto de treinamento parece difícil, mas pode ser trivial
  - Podemos construir uma ad que tem um caminho para uma folha correspondente a cada exemplo
  - Infelizmente ela não terá muito a informar sobre qualquer outros casos (não fará boas generalizações)
  - Ela memoriza as observações, não extrai qualquer padrão



O problema com a ad trivial

- Pela Lâmina de Okham devemos encontrar a menor ad consistente com os exemplos (intratável)
  - Podemos resolver o problema intratável utilizando algumas heurísticas para encontrarmos árvores pequenas
- Algoritmo de aprendizagem em árvore de decisão
  - A idéia é testar o **atributo mais importante –** aquele que faz a maior diferença para a classificação dos exemplos
  - Obter a classificação correta com um pequeno número de testes = caminhos curtos e árvore pequenas

Conjunto de treinamento para o exemplo do restaurante

Ex	Alt	Bar	Sex	Fam	Cli	Pre	Chu	Res	Tipo	Tem	Meta
X_1	Sim	Não	Não	Sim	Alg	$$$	Não	Sim	Fra	0-10	Sim
X_2	Sim	Não	Não	Sim	Che	$	Não	Não	Tai	30-60	Não
X_3	Não	Sim	Não	Não	Alg	$	Não	Não	Ham	0-10	Sim
X_4	Sim	Não	Sim	Sim	Che	$	Sim	Não	Tai	10-30	Sim
X_5	Sim	Não	Sim	Não	Che	$$$	Não	Sim	Fra	>60	Não
X_6	Não	Sim	Não	Sim	Alg	$$	Sim	Sim	Ita	0-10	Sim
X_7	Não	Sim	Não	Não	Ne	$	Sim	Não	Ham	0-10	Não
X_8	Não	Não	Não	Sim	Alg	$$	Sim	Sim	Tai	0-10	Sim
X_9	Não	Sim	Sim	Não	Che	$	Sim	Não	Ham	>60	Não
X_{10}	Sim	Sim	Sim	Sim	Che	$$$	Não	Sim	Ita	10-30	Não
X_{11}	Não	Não	Não	Não	Ne	$	Não	Não	Tai	0-10	Não
X_{12}	Sim	Sim	Sim	Sim	Che	$	Não	Não	Ham	30-60	Sim

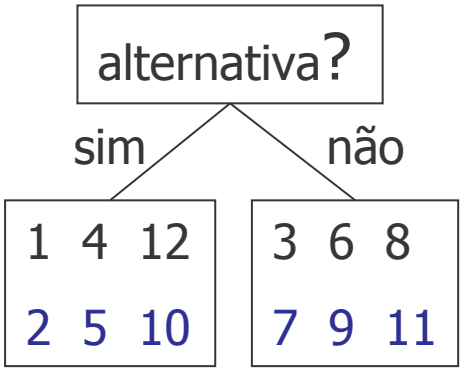
Aplicação do algoritmo de aprendizagem

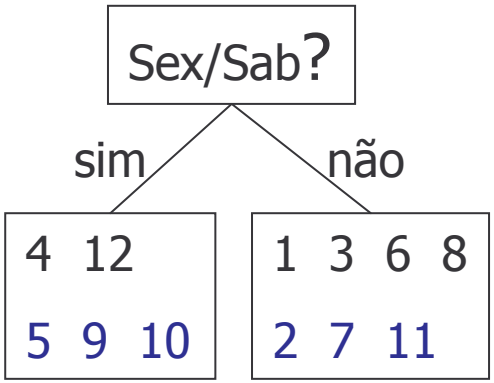
Ex	Alt	Bar	Sex
X_1	Sim	Não	Não
X_2	Sim	Não	Não
X_3	Não	Sim	Não
X_4	Sim	Não	Sim
X_5	Sim	Não	Sim
X_6	Não	Sim	Não
X_7	Não	Sim	Não
X_8	Não	Não	Não
X_9	Não	Sim	Sim
X_{10}	Sim	Sim	Sim
X_{11}	Não	Não	Não
X_{12}	Sim	Sim	Sim

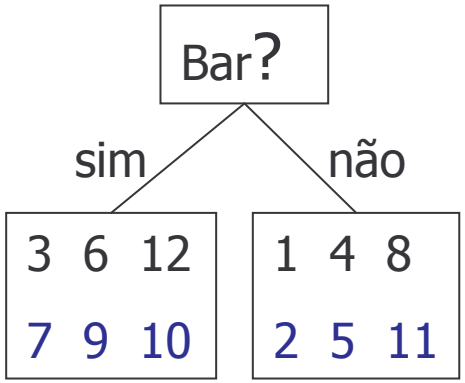
Meta

Sim 1 3 4 6 8 12

Não 2 5 7 9 10 11

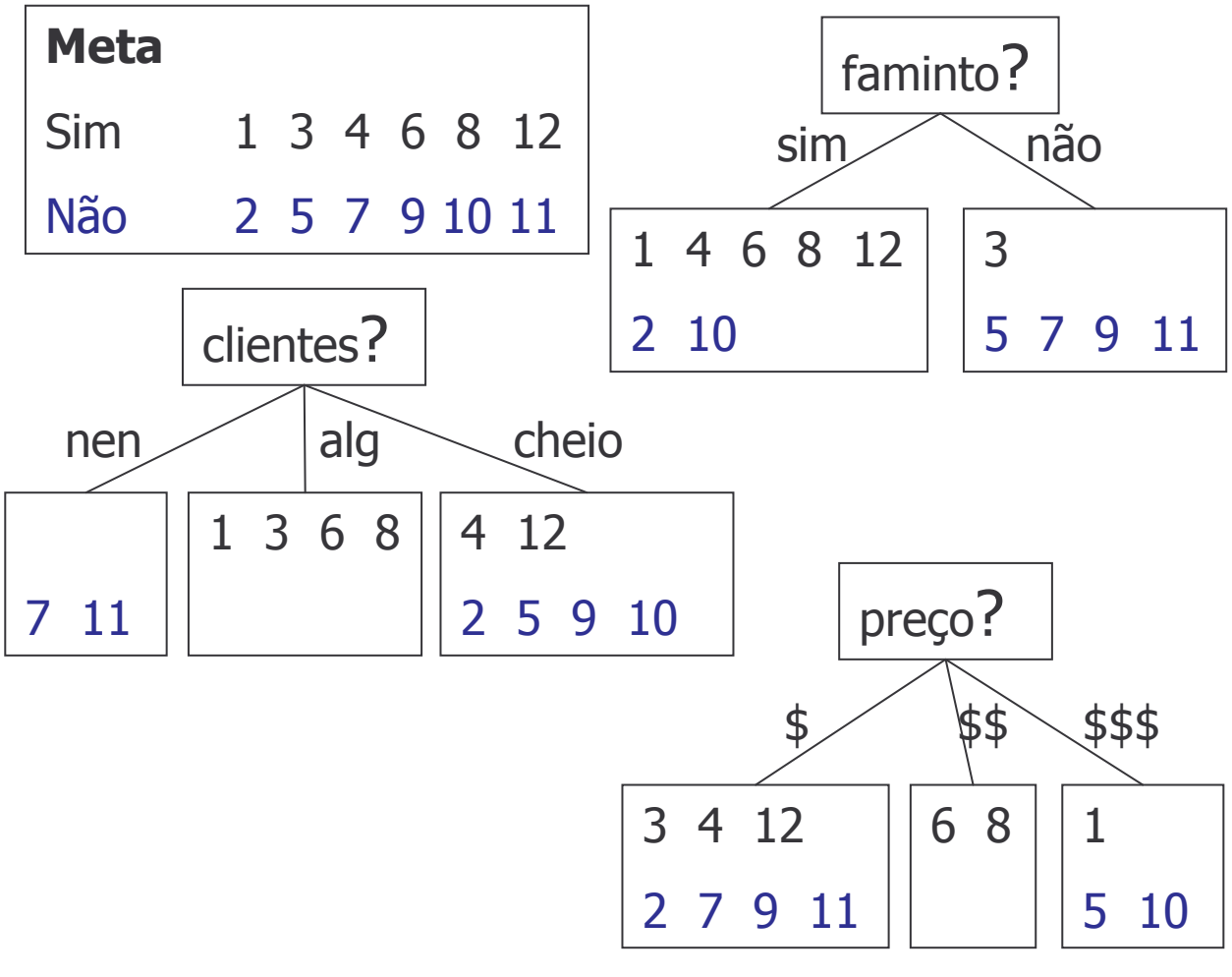


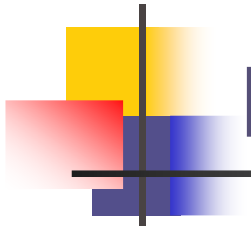




Aplicação do algoritmo de aprendizagem

Ex	Fam	Cli	Pre
X_1	Sim	Alg	$$$
X_2	Sim	Che	$
X_3	Não	Alg	$
X_4	Sim	Che	$
X_5	Não	Che	$$$
X_6	Sim	Alg	$$
X_7	Não	Ne	$
X_8	Sim	Alg	$$
X_9	Não	Che	$
X_{10}	Sim	Che	$$$
X_{11}	Não	Ne	$
X_{12}	Sim	Che	$





Exercício

- Fazer a representação dos atributos abaixo da forma especificada anteriormente
  - Chovendo?
  - Reserva?
  - Tipo?
  - Tempo de espera?
- Qual é o atributo que separa melhor os exemplos?

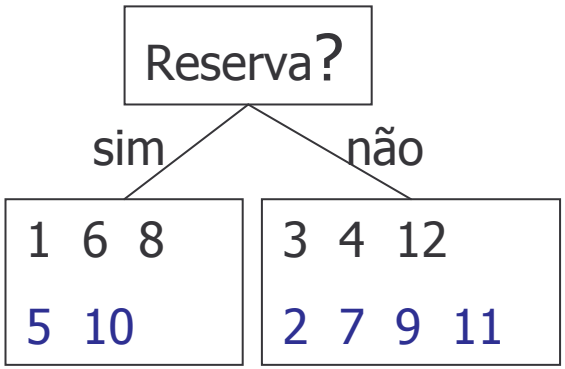
Aplicação do algoritmo de aprendizagem

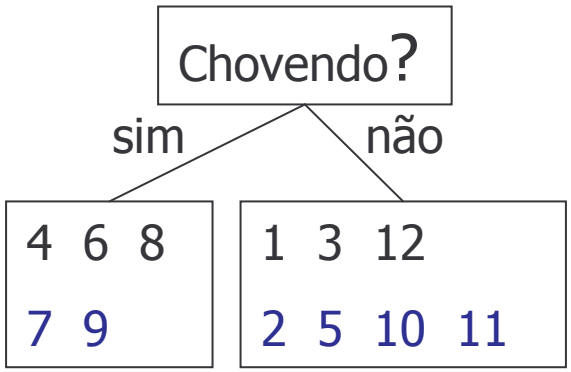
Ex	Cho	Res
X_1	Não	Sim
X_2	Não	Não
X_3	Não	Não
X_4	Sim	Não
X_5	Não	Sim
X_6	Sim	Sim
X_7	Sim	Não
X_8	Sim	Sim
X_9	Sim	Não
X_{10}	Não	Sim
X_{11}	Não	Não
X_{12}	Não	Não

Meta

Sim 1 3 4 6 8 12

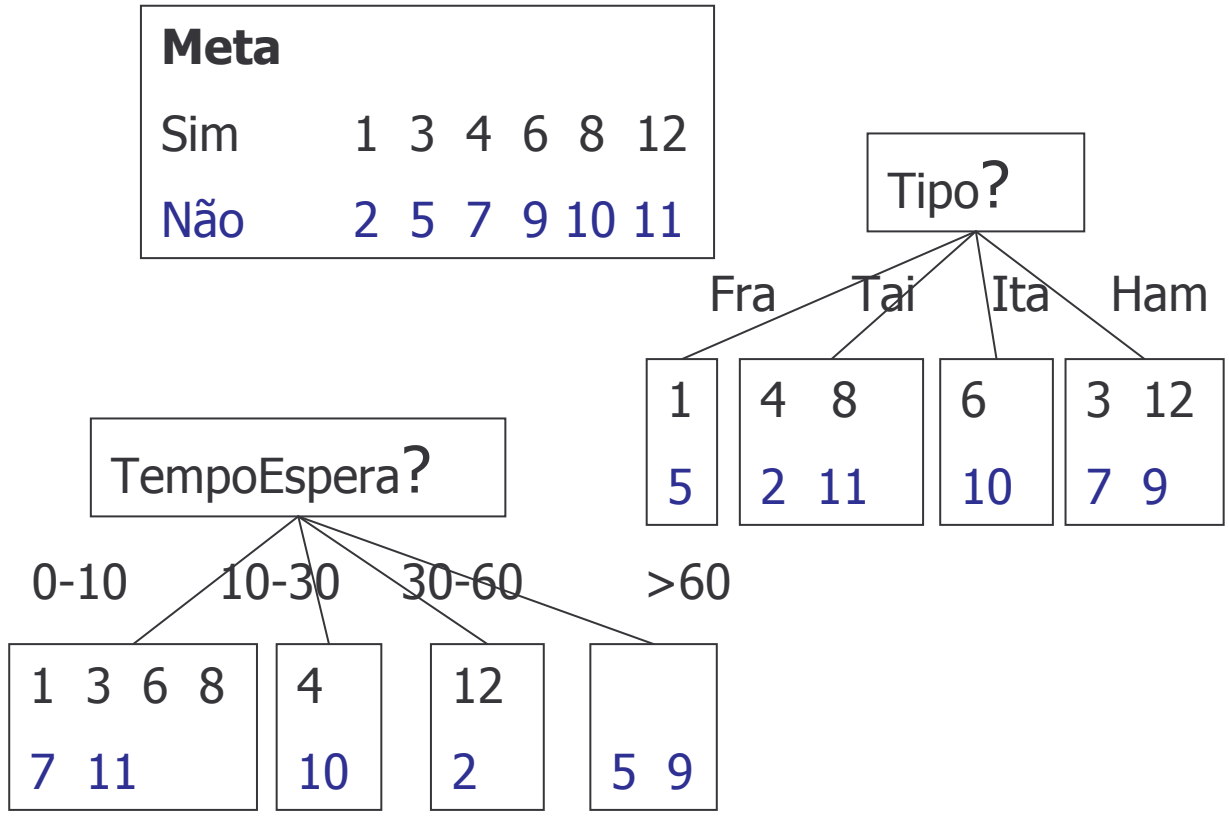
Não 2 5 7 9 10 11





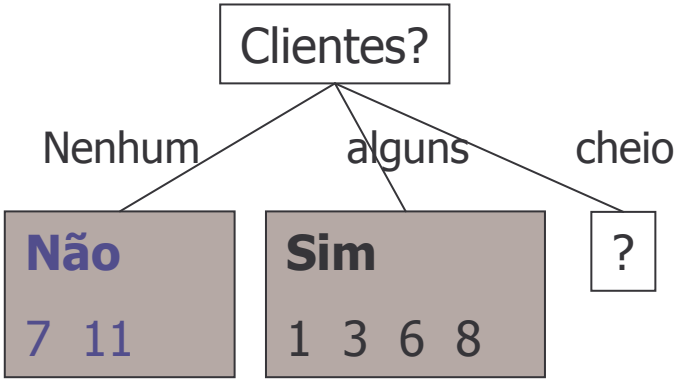
Aplicação do algoritmo de aprendizagem

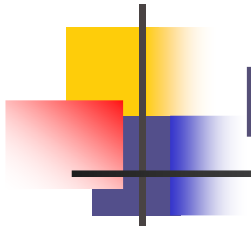
Ex	Tipo	Tem
X_1	Fra	0-10
X_2	Tai	30-60
X_3	Ham	0-10
X_4	Tai	10-30
X_5	Fra	>60
X_6	Ita	0-10
X_7	Ham	0-10
X_8	Tai	0-10
X_9	Ham	>60
X_{10}	Ita	10-30
X_{11}	Tai	0-10
X_{12}	Ham	30-60



Qual é o atributo que melhor separa os exemplos?

- O atributo *cliente* classifica 6 exemplos
- Então ele será a raiz da árvore de decisão

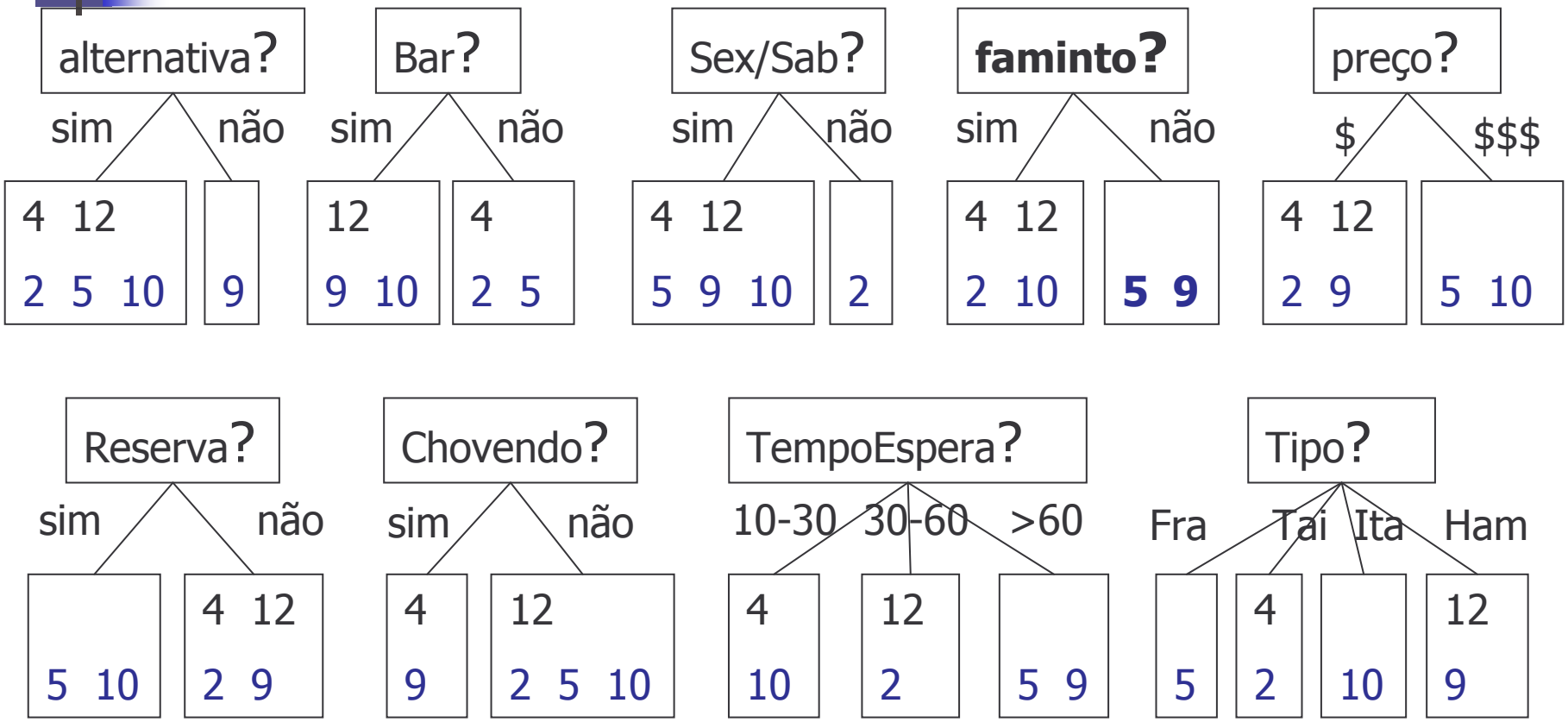




Escolha dos atributos

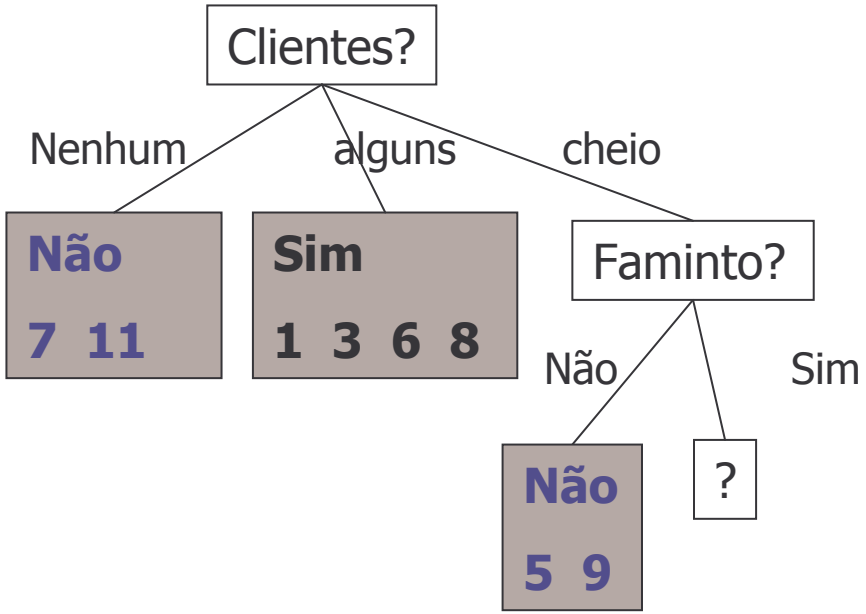
- Depois da escolha do atributo *clientes* ficamos com um conjunto misto de exemplo se o valor for *cheio*
- Depois que o primeiro teste de atributo separa os exemplos, cada resultado é um novo problema de aprendizagem em ad
  - Com menos exemplos (os que ainda não foram classificados (2 5 9 10 4 e 12) e um atributo a menos (tira-se clientes)
- **Exercício**: Aplique o algoritmo de aprendizagem de ad para os exemplos anteriores

Nova instância do problema com menos exemplos e menos atributos

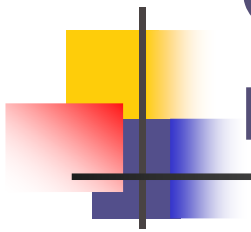


Qual é o atributo que melhor separa os exemplos restantes?

- Faminto, preço, reserva e tipo classificam 2 exemplos
  - Podemos escolher qualquer um deles (heurística gulosa)



Casos a considerar na escolha de um novo atributo



- Se existem alguns exemplos positivos e alguns negativos, escolha o melhor atributo para dividi-los
- Se todos os exemplos restantes forem positivos (ou negativos), terminamos
- Não resta nenhum atributo mas há exemplos positivos e negativos – descrições iguais com classificações diferentes = **ruído** nos dados

Algoritmo ID3 (Inductive Decision Tree)

Função APRENDIZAGEM-EM-AD (*exemplos, atributos, padrão*) **retorna** uma ad

entradas: *exemplos*, conjunto de exemplos

atributos, conjunto de atributos

padrão, valor-padrão para o predicado de objetivo

se *exemplos* é vazio **então retornar** padrão

senão se todos os *exemplos* têm a mesma classificação **então retornar** a classificação

senão se *atributos* é vazio **então retornar** VALOR-DA-MAIORIA(*exemplos*)

senão

melhor <- ESCOLHER-ATRIBUTO(*atributos, exemplo*)

árvore <- uma nova ad com teste de raiz *melhor*

m <- VALOR-DA-MAIORIA($exemplos_i$)

para cada valor v_i de *melhor* **faça**

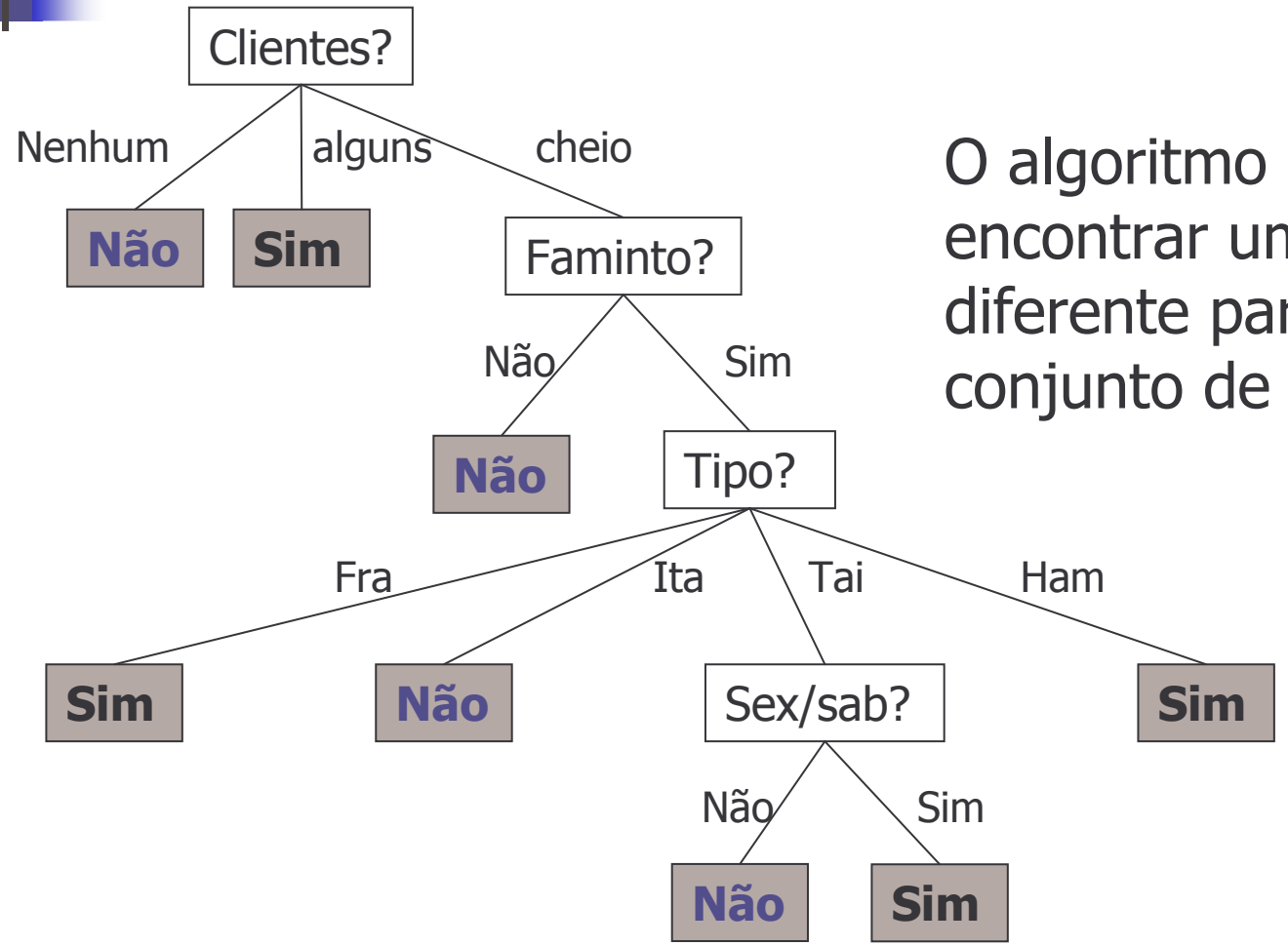
$exemplos_i$ <- {elementos de *exemplos* com melhor = v_i)

subárvore <- APRENDIZAGEM-EM-AD($exemplos_i$,*atributos-melhor*, *m*)

adicionar uma ramificação a *árvore* com rótulo v_i e subárvore *subárvore*

retornar *árvore*

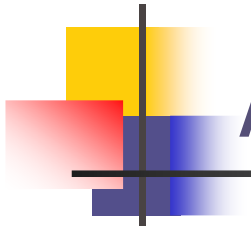
Árvore resultante da aplicação do algoritmo ID3



O algoritmo poderia encontrar uma ad diferente para o mesmo conjunto de treinamento?

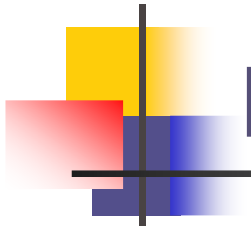
Por que?

Qual seria?



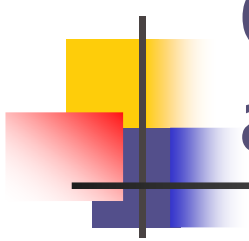
A árvore resultante

- É diferente da árvore original
- Mas a hipótese concorda com todos os exemplos
- E é consideravelmente mais simples do que a árvore original
- *Chovendo* e *Reserva* ficaram de fora porque a árvore não necessita deles para classificar os exemplos
- Se tivéssemos um conjunto de treinamento maior, com certeza a árvore ficaria mais parecida com a original



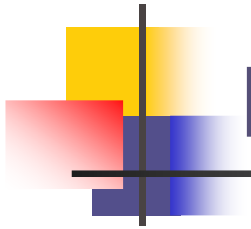
Escolha de testes de atributos

- Esquema de aprendizagem da ad
  - Projetado para minimizar a profundidade da árvore final
  - Idéia: escolher o atributo que melhor fornece uma classificação exata dos exemplos
- **Atributo perfeito**: divide os exemplos em conjuntos que são todos positivos ou todos negativos
  - *Clientes* não é perfeito, mas é "bastante bom"
- **Atributo inútil**: deixa os conjuntos de exemplos com a mesma proporção do conjunto original
  - *Tipo* é um atributo "realmente inútil"



Quantidade de informação fornecida pelo atributo

- Como medimos um "atributo muito bom" ou um "atributo realmente inútil"
- A medida
  - Deve ter o valor máximo quando o atributo é perfeito
  - E o valor mínimo quando o atributo for completamente inútil
- Uma medida apropriada seria a **quantidade** esperada de **informação** fornecida pelo atributo
  - Isto é feito por meio da teoria da informação



Entropia

- Medida comumente usada na teoria da informação
- Caracteriza a **impureza** de uma coleção de exemplos
- Seja *S* uma coleção de exemplos positivos e negativos de algum conceito objetivo lógico, a entropia de *S*:

$$\text{Entropia} \equiv - p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

onde $p_{\oplus}$ é a proporção de exemplos positivos em *S*

e $p_{\ominus}$ é a proporção de exemplos negativos em *S*

Cálculo de Entropia – Exemplo

- Suponha *S* uma coleção de 14 exemplos de algum conceito lógico
  - 9 exemplos são positivos e 5 são negativos [9+, 5-]

$$\text{Entropia}([9+, 5-]) \equiv -(9/14)\log_2(9/14) - (5/14)\log_2(5/14)$$

$$\text{Entropia}([9+, 5-]) = 0.940$$

- Entropia=0, se todos os exemplos de *S* pertencem à mesma classe (todos positivos ou todos negativos)
- Entropia=1, quando *S* contém um número igual de exemplos positivos e negativos (exemplo do restaurante)
- Se *S* contém números desiguais de exemplos positivos e negativos a entropia está entre 0 e 1

Medida de Ganho de informação

- A medida da efetividade de um atributo para classificar os dados de treinamento
- Ganho de informação = redução esperada da entropia causada pelo particionamento dos exemplos de *S* por um atributo *A*

$$\text{Ganho}(S, A) = \text{Entropia}(S) - \sum_{v \in \text{valores}(A)} \left(S_v / S\right) \text{Entropia}(S_v)$$

onde :

- valores(A) é o conjunto de todos os possíveis valores para o atributo A

- S_V é o subconjunto de S para o qual o atributo A tem valor v

Medida de Ganho de informação – Exemplo para o atributo *Clientes*

$S = [6+,6-] \qquad S_{nenhum} = [0+,2-] \quad S_{alguns} = [4+,0-] \quad S_{cheio} = [2+,4-]$

$$Ganho(S, Clientes) = Entropia(S) - \sum_{v \in (nenhum,\, alguns,\, cheio)} (S_v / S)\, Entropia(S_v)$$

$$Ganho(S, Clientes) = Entropia(S) - (2/12)\, Entropia(S_{nenhum}) - (4/12)\, Entropia(S_{alguns}) - (6/12)\, Entropia(S_{cheio})$$

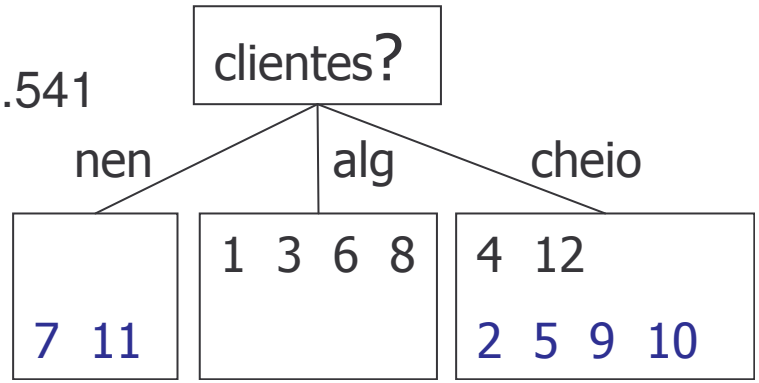
$Entropia(S) = = -(6/12)\log_2(6/12) - (6/12)\log_2(6/12) = 1$ (número de positivos e negativos é igual)

$Entropia(S_{nenhum}) = -(0/2)\log_2(0/2) - (2/2)\log_2(2/2) = 0$ (todos negativos)

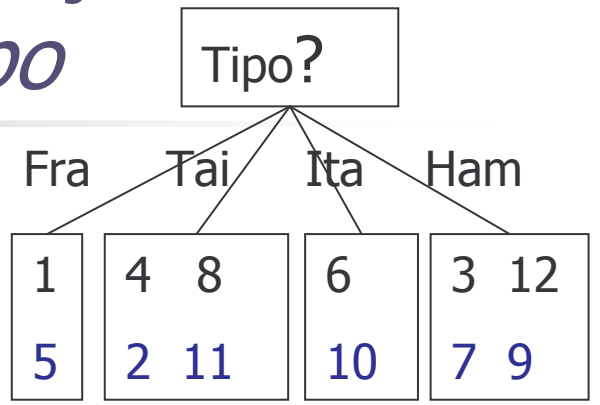
$Entropia(S_{alguns}) = -(4/4)\log_2(4/4) - (0/4)\log_2(0/4) = 0$ (todos positivos)

$Entropia(S_{cheio}) = -(2/6)\log_2(2/6) - (4/6)\log_2(4/6) = 0.918$

$Ganho(S, Clientes) = 1 - (2/12)\,0 - (4/12)\,0 - (6/12)\,0.918 = 0.541$



Medida de Ganho de informação – Exemplo para o atributo *Tipo*



$S = [6+,6-]$

$S_{Francês} = [1+,1-]$ $S_{Italiano} = [1+,1-]$

$S_{Tailandês} = [2+,2-]$ $S_{Hamburger} = [2+,2-]$

$$Ganho(S,Tipo) = Entropia(S) - \sum_{v \in (Francês, Italiano, Tailandês, Hamburger)} (S_v/S)\, Entropia(S_v)$$

$$Ganho(S,Tipo) = Entropia(S) - (2/12)\, Entropia(S_{Fra}) - (2/12)\, Entropia(S_{Ita}) - (4/12)\, Entropia(S_{Tai}) - (4/12)\, Entropia(S_{Ham})$$

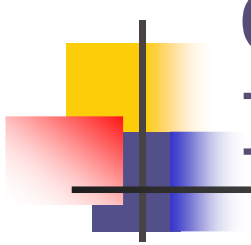
$Entropia(S) = 1$ (número de exemplos positivos e negativos é igual)

$Entropia(S_{Fra}) = Entropia(S_{Ita}) = Entropia(S_{Tai}) = Entropia(S_{Ham}) = 1$

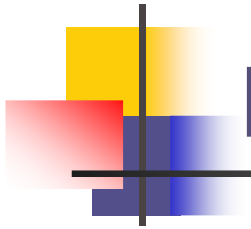
(em todos os subconjuntos o número de exemplos positivos e negativos é igual)

$$Ganho(S,Tipo) = 1 - (2/12)1 - (2/12)1 - (4/12)1 - (4/12)1 = 0$$

Conclusões sobre a Medida de Ganho de Informação



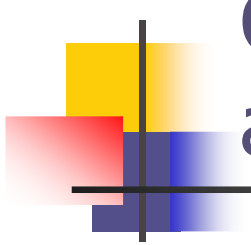
- O atributo *Clientes* é melhor do que o atributo *Tipo*
  - *Clientes* tem maior ganho de informação para separar os exemplos
  - *Tipo* é um atributo inútil para classificar os exemplos (Ganho(S, *Tipo*) = 0)
- O ganho de informação deve ser calculado para todos os atributos e o conjunto *S* de exemplos
  - Clientes será o atributo com maior ganho
- Após a escolha de *Clientes* como a raiz da árvore, o novo conjunto *S* se torna S_{cheio}[2+, 4-] e Entropia(S_{cheio})=0.918
  - Novamente o ganho de informação deve ser calculado para todos os atributos e o novo conjunto *S*



Exercício

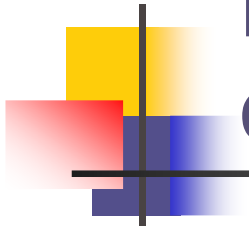
- Calcule o ganho de informação para os outros atributos do restaurante, considerando que o atributo *Clientes* já foi escolhido como raiz da árvore

- Continue o algoritmo de aprendizagem calculando o ganho de informação para os atributos, para provar a árvore de decisão que foi induzida



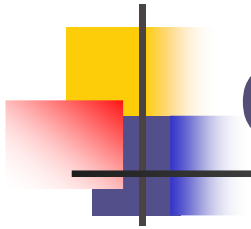
Como avaliar um algoritmo de aprendizagem?

- É um bom algoritmo se ele produz hipóteses que classificam (predizem) bem exemplos ainda não vistos
- A predição é boa se ela se torna verdadeira
- Como avaliar a qualidade de uma hipótese?
  - Podemos checar sua previsão com uma classificação correta que já conhecemos
  - Conjunto de exemplos conhecidos = **conjunto de testes**



Metodologia para avaliação de desempenho de um algoritmo

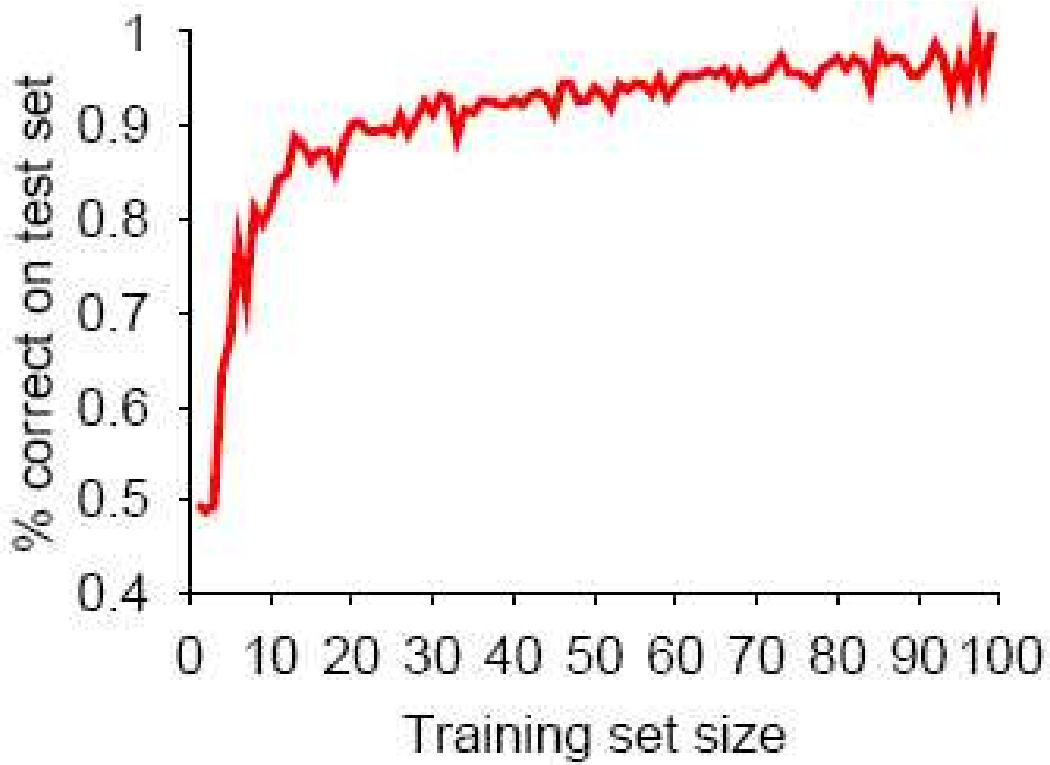
1. Coletar um grande conjunto de exemplos
2. Dividi-lo em dois conjuntos disjuntos
   - Conjunto de treinamento e conjunto de teste
3. Aplicar o algoritmo ao conjunto de treinamento, gerando uma hipótese *h*
4. Medir a quantidade de exemplos do conjunto de teste classificados corretamente por *h*
5. Repetir as etapas 1 a 4 para
   - Diferentes tamanhos de conjuntos de treinamento
   - Diferentes conjs. de treinamento de cada tamanho

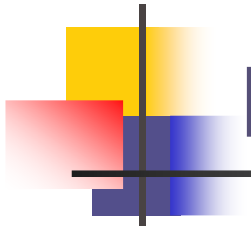


Curva de aprendizagem

- É traçada com o conjunto de dados obtidos da metodologia anterior
- Conjunto de treinamento aumenta => qualidade da previsão aumenta
- Bom sinal de que existe um padrão nos dados e o algoritmo está capturando este padrão

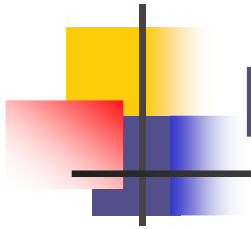
Curva de aprendizagem





Ruído e superadaptação (*overfitting*)

- O algoritmo ID3 faz crescer cada ramo da árvore o suficiente para classificar perfeitamente os exemplos de treino
- Problemas:
  - Quando existe **ruídos** ou **erros aleatórios** nos dados ou
  - Quando o **número de exemplos de treino é muito pequeno** não constituindo uma amostra representativa da verdadeira função objetivo
  - Nestes casos ID3 pode produzir árvores que ***se*** **superadaptam** os exemplos de treino - isto é, aprendem inclusive os ruídos e os erros

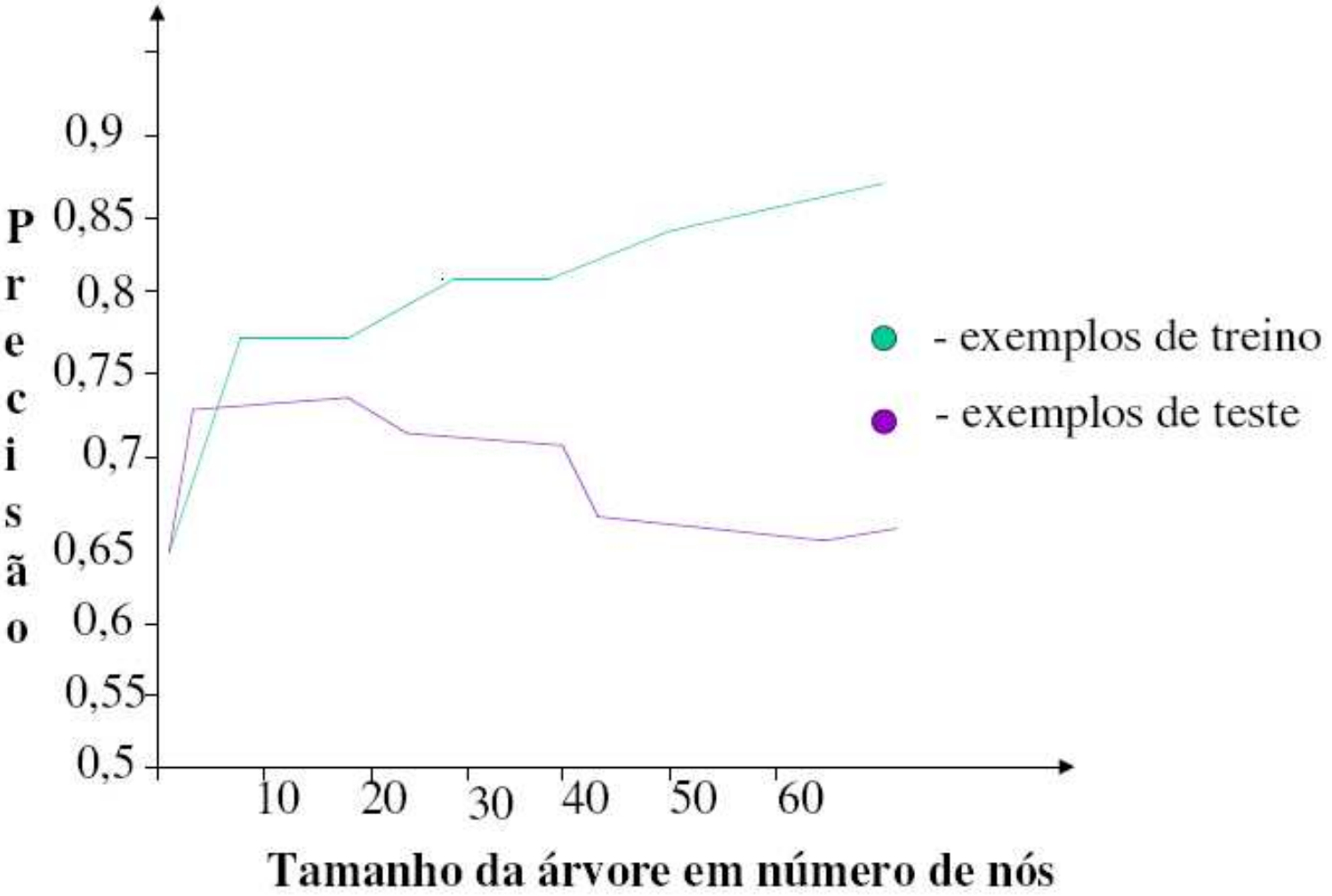


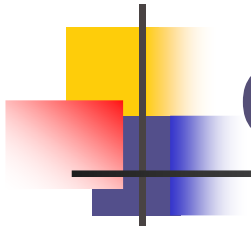
Definição de superadaptação

Dado um espaço de hipóteses H. Uma hipótese $h \in H$ *overfit* os dados de treino se existe alguma hipótese alternativa $h' \in H$, tal que h tenha **menor** erro do que h' sobre os exemplos de treino, **mas** h' tem **menor** erro do que h **sobre toda a distribuição dos exemplos** (i.e. Incluindo exemplos fora do conjunto de treinamento).

- A superadaptação aflinge todo tipo de algoritmo de aprendizagem, não apenas árvores de decisão

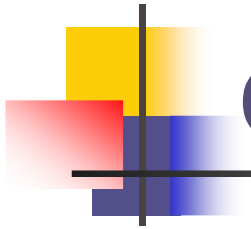
Exemplo de superadaptação





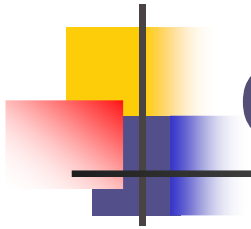
Como evitar a superadaptação

- Fazer a árvore parar de crescer antes que ela alcance o ponto onde ela classifique perfeitamente os exemplos de treino
  - Para isso temos que acompanhar as porcentagens de previsões corretas tanto no conjunto de teste quanto no conjunto de treinamento
  - Um atributo com ganho de informação perto de zero tem grandes indícios de ser um atributo irrelevante
  - Quando a partição dos dados não for estatísticamente significante, podemos parar o crescimento da árvore



Como evitar a superadaptação

- Permitir que a árvore "sobreajuste" os dados, e depois podar os atributos irrelevantes
  - Podar um nó de decisão: remover a sub-árvore enraizada naquele nó, tornando–o um nó folha
    - Atribuir a este nó, a classificação mais comum dos exemplos de treinamento afiliados com aquele nó
  - Nós são removidos somente se a árvore aparada resultante não apresenta um comportamento pior do que a original sobre o conjunto de validação (Erro de poda reduzido)



Como evitar a superadaptação

- Particionar os dados em conjuntos de validação e treinamento
  - Faça até que uma redução (poda) adicional seja prejudicial
  - 1. Avaliar o impacto sobre o conjunto de validação da poda de cada nó possível, mais aqueles abaixo dele
  - 2. Remover "gulosamente" aquele que melhora mais a precisão sobre o conjunto de validação