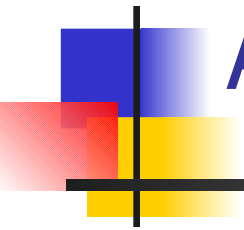
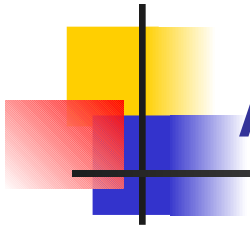


Aprendizagem

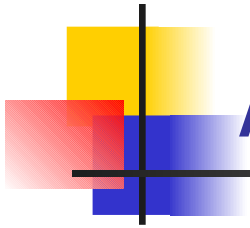
Profa. Josiane M. Pinheiro Ferreira

Nov/2005



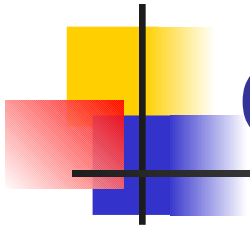
A idéia por trás da aprendizagem

- 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



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

١. Um mapeamento de condições -> ações

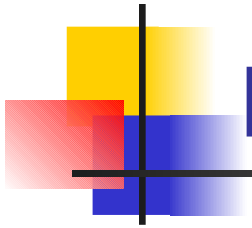
٢. Um meio para deduzir propriedades do mundo a partir de seqüências de ações

٣. Informações sobre o modo como o mundo evolui e sobre o resultados das ações possíveis que o agente pode executar

٤. Informações de utilidade indicando a desejabilidade de estados do mundo

٥. Informações de valores de ações indicando a desejabilidade de ações

٦. Metas que descrevem classes de estados cuja realização maximiza a utilidade



Realimentação

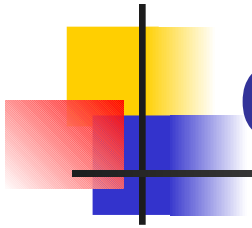
- Cada componente pode ser aprendido a partir da realimentação apropriada

٢. Toda vez que o instrutor gritar “Freie!” o agente poderá aprender uma regra de condição-ação

٣. Ao ver muitas imagens que lhe dizem ser um ônibus o agente poderá aprender a reconhecê-los

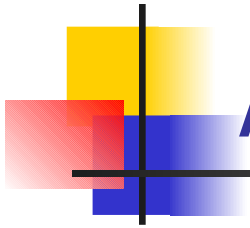
٤. Freando bruscamente em uma estrada molhada o agente poderá aprender o resultado de suas ações

٥. 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
- Aprendizagem **supervisionada**
- Aprendizagem **não supervisionada**
- Aprendizagem **por reforço**



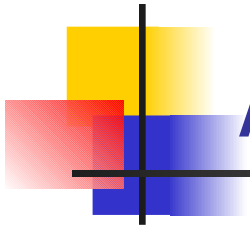
Aprendizagem supervisionada

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

٢. Agente aprende a regra condição-ação
   - Função de estados para uma saída booleana (frear ou não)

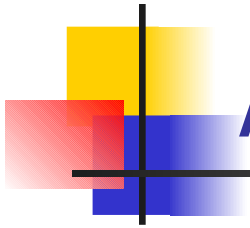
٣. Agente aprende a reconhecer um ônibus
   - Função a partir de imagens para uma saída booleana (a imagem contém ou não o ônibus)

٤. Agente aprende a teoria de frear
   - Função de estados e ações para, por exemplo, a distância de parada



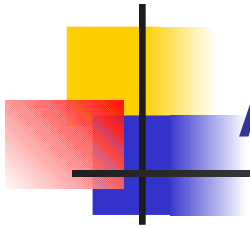
Aprendizagem supervisionada

- 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



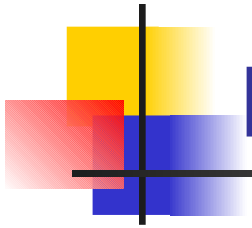
Aprendizagem não-supervisionada

- 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"
  - Sem jamais ter recebido exemplos de cada um deles
- 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 deu comportamento é indesejável
- Inclui o subproblema de aprender como o ambiente funciona

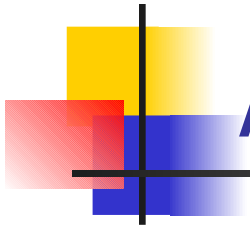


Representação da informações

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

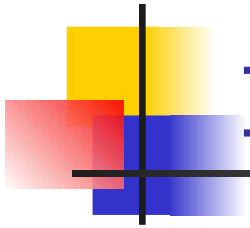
Disponibilidade de conhecimento anterior

- 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
  - Ex: Físico e crítico de arte examinando uma pilha de fotografias



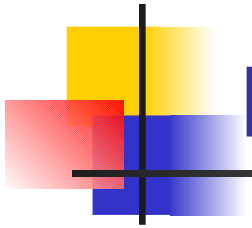
Aprendizagem indutiva

- 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)

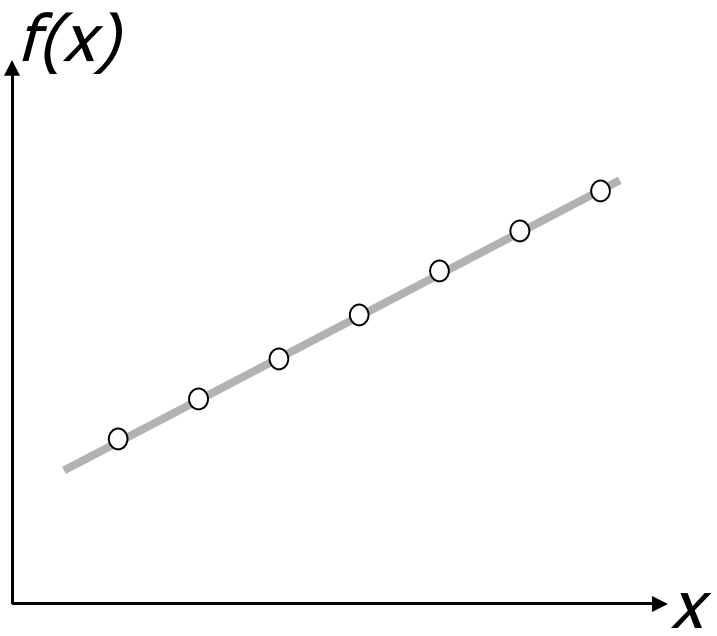
- Dada uma coleção de exemplos de *f*, retornar uma função *h* que se aproxime de *f*.
- A função *h* é chamada **hipótese**
  - Não é fácil saber se uma *h* específica é uma boa aproximação de *f*
  - Uma boa *h* irá **generalizar** bem – prever corretamente exemplos ainda não vistos
  - Esse é o **problema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



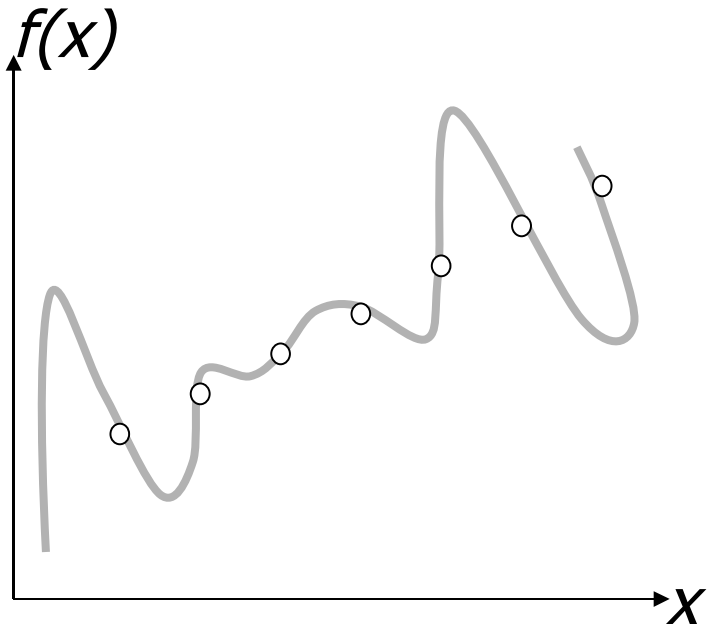
Exemplo de indução

- 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 $x^{17} - 4x^3$

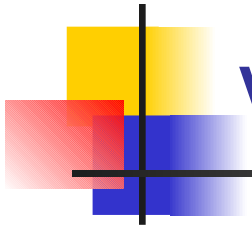
Exemplos de hipóteses I



Uma hipótese linear consistente



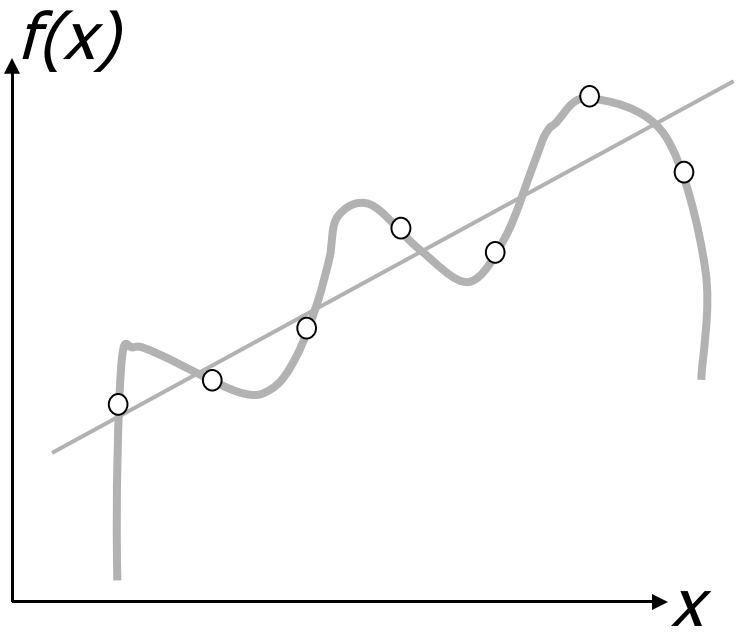
Uma hipótese de polinômio de grau 7 consistente



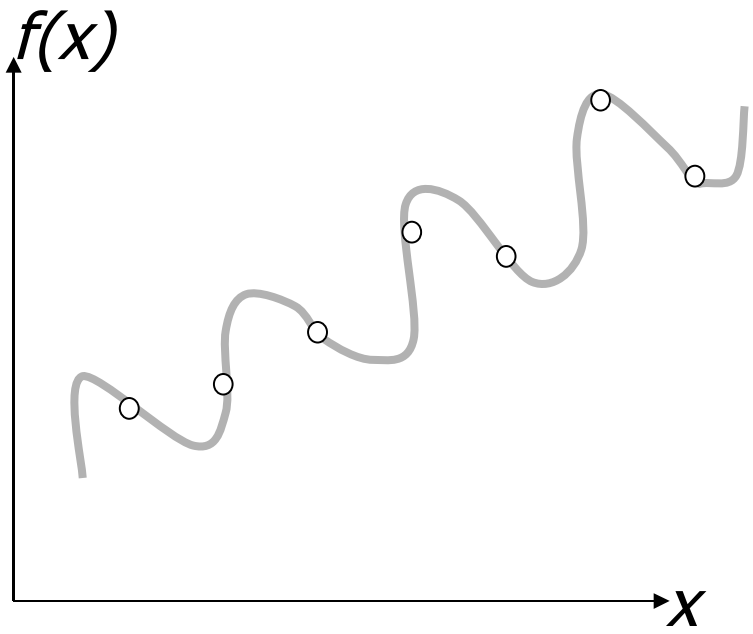
Várias hipóteses consistentes

- 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*
- Mas, um polinômio de grau 1 é mais simples do que um polinômio de grau 12

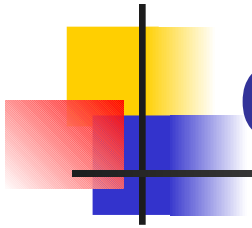
Exemplos de hipóteses II



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

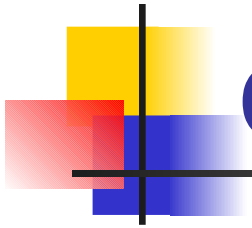


Um ajuste senoidal exato para o mesmo conjunto de dados



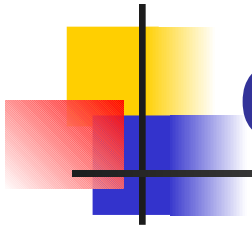
O ajuste dos dados

- 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



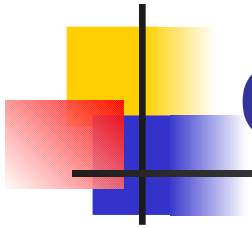
O espaço de hipóteses escolhido

- Os dados podem ser ajustados exatamente por uma função simples do tipo ax + b + c sen x
- Um espaço de hipóteses de polinômios de grau finito não representa funções senoidais com precisão
  - Um agente com este espaço de estados 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

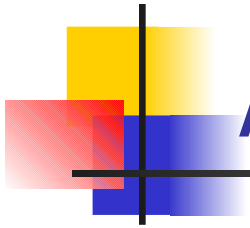
- Nem sempre podemos dizer se o problema é realizável ou não pois a fç verdadeira é desconhecida
  - Solução: conhecimento anterior
- Porque não utilizar o maior espaço de estados possíveis?
- Porque não fazer de **H** o espaço de estados de todas as máquinas de Turing?
  - Toda função computável pode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



Complexidade computacional

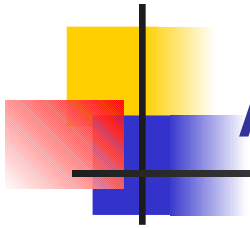
- Expressividade de **H** X a complexidade de encontrar *h*'s simples e consistentes dentro de **H**
  - Ajuste de linhas retas é fácil
  - Ajuste de polinômios é mais difícil
  - Ajuste de máquinas de Turing é muito difícil
  - Determinar se uma máquina de Turing é consistente com os dados não é nem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



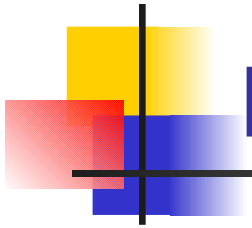
Aprendizagem em árvores de decisão

- 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



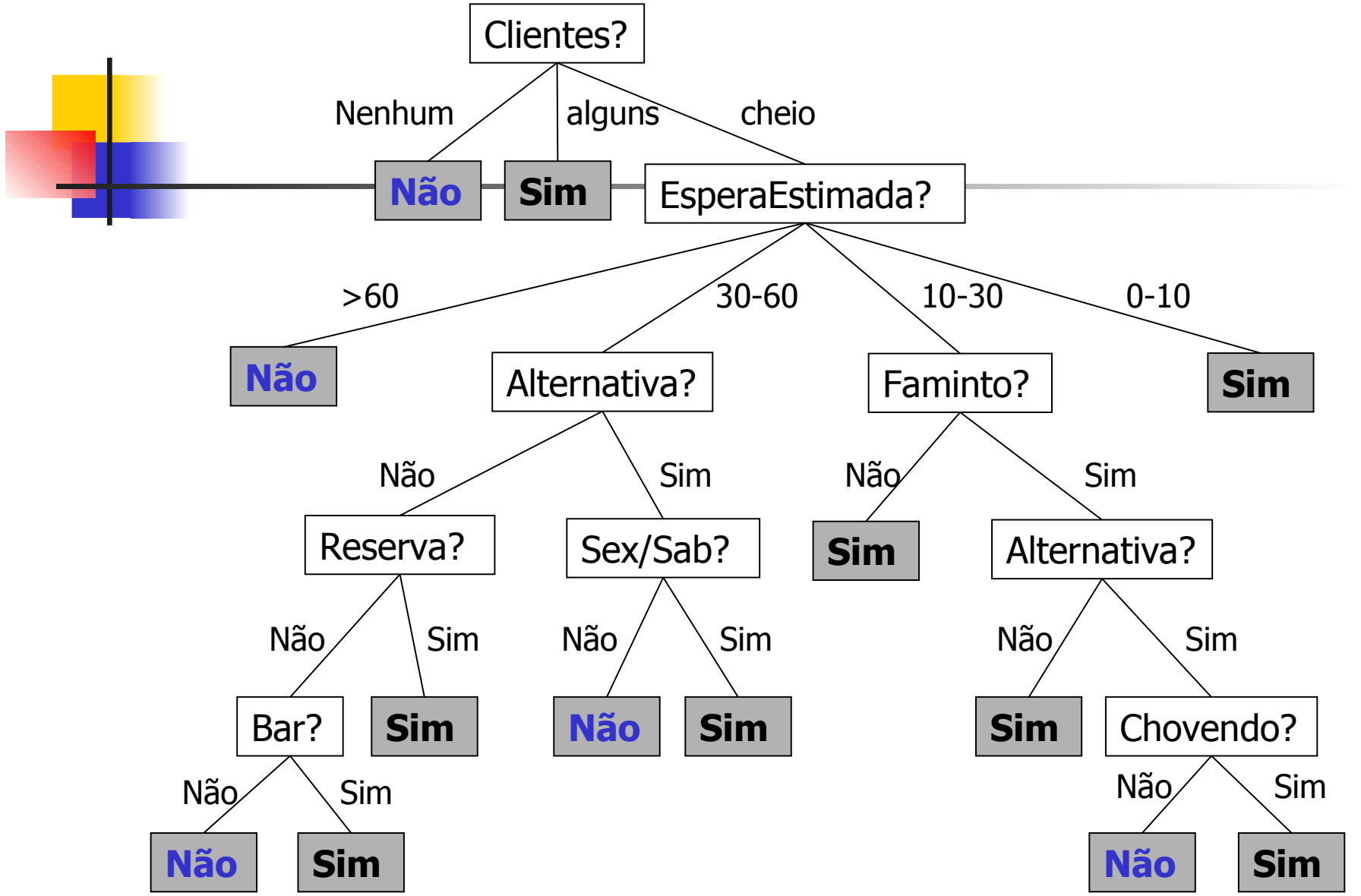
Árvores de decisão

- 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?
- 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: A espera estimada pelo o gerente

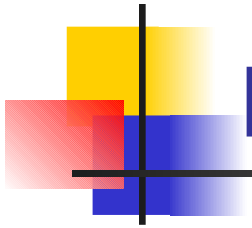

Clientes?
Nenhum
alguns
cheio
Não
Sim
EsperaEstimada?
>60
30-60
10-30
0-10
Não
Alternativa?
Faminto?
Sim
Não
Sim
Não
Sim
Reserva?
Sex/Sab?
Sim
Alternativa?
Não
Sim
Não
Sim
Não
Sim
Não
Sim
Bar?
Sim
Não
Sim
Sim
Chovendo?
Não
Sim
Não
Sim
Não
Sim

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:

  $\forall s\ \mathit{VaiEsperar}(s) \Leftrightarrow (P_1(s) \vee P_2(s) \vee \ldots \vee P_n(s))$

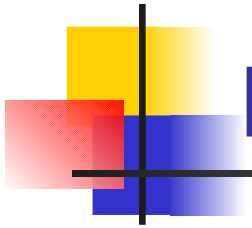
- Onde cada condição $P_i(s)$ é uma conjunção de testes que pode corresponde a um caminho da raiz até uma folha da árvore com resultado positivo



Expressividade de árvores de decisão

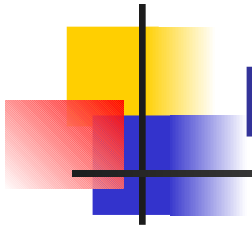
- Uma árvore de decisão
  - Descreve um relacionamento entre o predicado meta
  - E alguma combinação lógica de atributos
- Não podemos utilizar árvores de decisão para representar testes que se referem a dois ou mais objetos diferentes

$\exists r_2$ Perto(r_2, r) ^ Preço(r, p) ^ Preço(r_2, p_2) ^ MaisBarato(p_2, p)



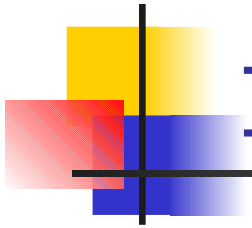
Expressividade de árvores de decisão

- 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



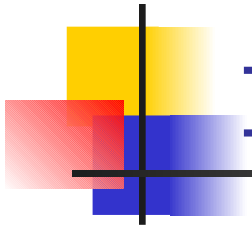
Expressividade de árvores de decisão

- 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



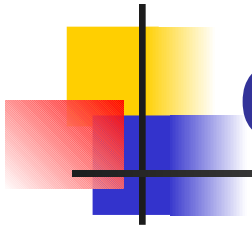
Induzindo árvores de decisão

- 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 qq 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 de um exemplo
  - Obter a classificação correta com pq nº de testes = caminhos curtos e árvore pq

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

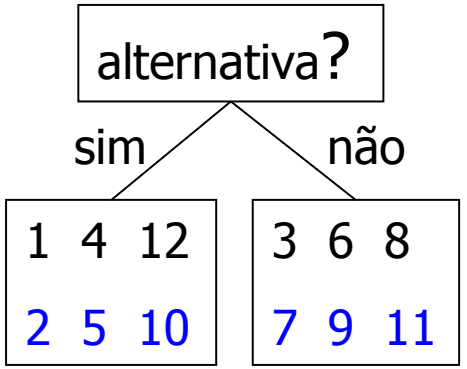
Aplicação do algoritmo de aprendizagem

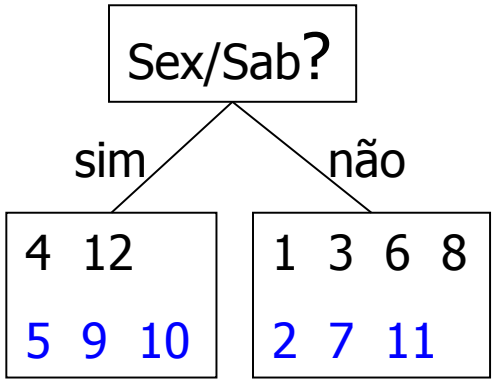
Ex	Alt	Bar	Sex
X_1	Sim	Não	Não
X_2	Sim	Não	Não
X_3	Não	Sim	Não
X_4	Sim	Não	Sim
X_5	Sim	Não	Sim
X_6	Não	Sim	Não
X_7	Não	Sim	Não
X_8	Não	Não	Não
X_9	Não	Sim	Sim
X_{10}	Sim	Sim	Sim
X_{11}	Não	Não	Não
X_{12}	Sim	Sim	Sim

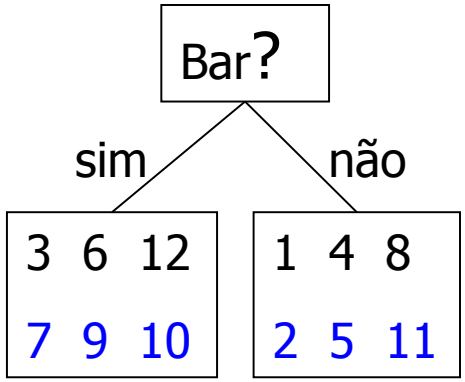
Meta

Sim 1 3 4 6 8 12

Não 2 5 7 9 10 11

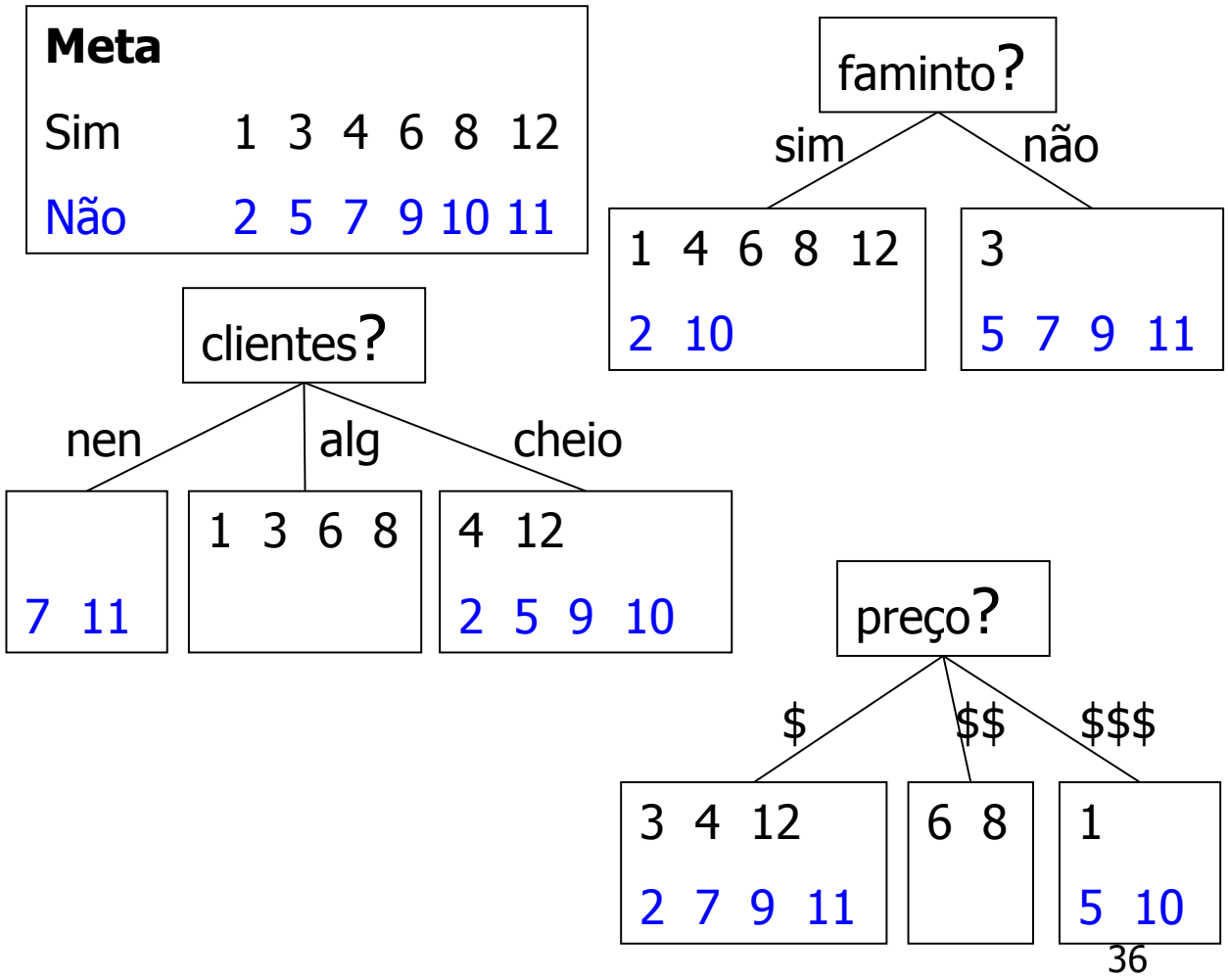


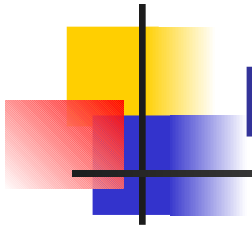




Aplicação do algoritmo de aprendizagem

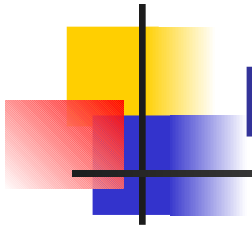
Ex	Fam	Cli	Pre
X_1	Sim	Alg	$$$
X_2	Sim	Che	$
X_3	Não	Alg	$
X_4	Sim	Che	$
X_5	Não	Che	$$$
X_6	Sim	Alg	$$
X_7	Não	Ne	$
X_8	Sim	Alg	$$
X_9	Não	Che	$
X_{10}	Sim	Che	$$$
X_{11}	Não	Ne	$
X_{12}	Sim	Che	$





Exercício

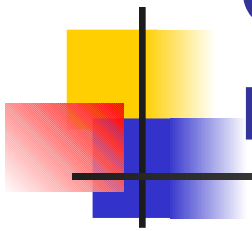
- Fazer a representação dos atributos abaixo da forma especificada anteriormente
  - Chovendo?
  - Reserva?
  - Tipo?
  - Tempo de espera?
- Qual é o atributo que separa melhor os exemplos?



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 e um atributo a menos
- **Exercício**: Aplique o algoritmo de aprendizagem de ad para os exemplos anteriores

Casos a considerar na escolha de um novo atributo



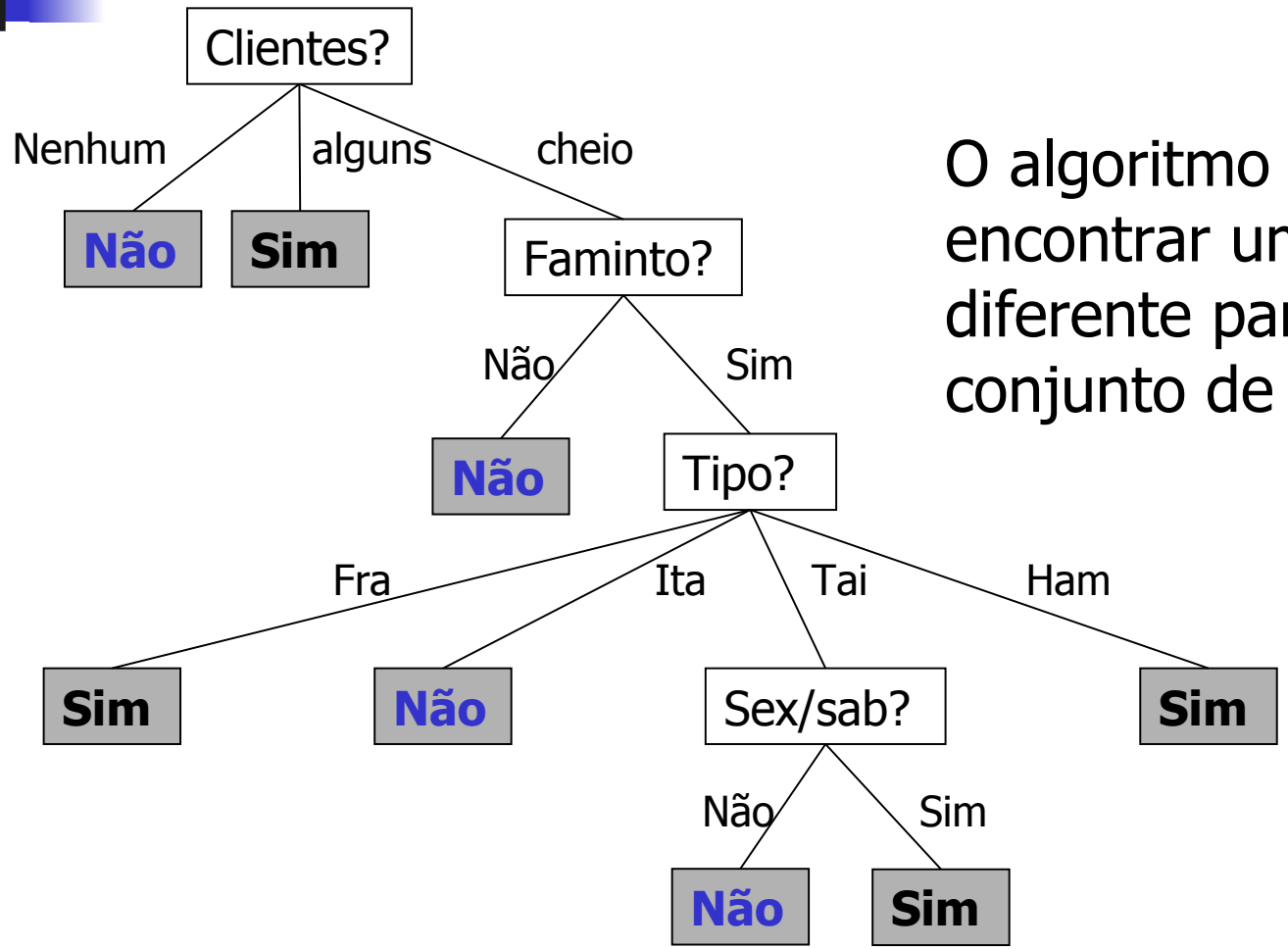
- Se existem alguns exemplos positivos e alguns negativos, escolha o melhor atributo para dividi-los
- Se todos os exemplos restantes forem positivos (ou negativos), terminamos
- Se não resta nenhum exemplo, nenhum exemplo deste tipo foi observado, retornamos a maioria do nó pai
- Não resta nenhum atributo mas há exemplos positivos e negativos – descrições iguais com classificações diferentes = **ruído** nos dados

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

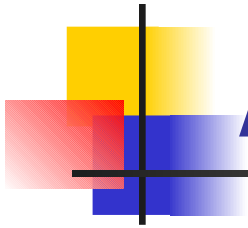
se *exemplos* é vazio **então retornar** padrão
senão se todos os *exemplos* têm a mesma classificação **então retornar** a classificação
senão se *atributos* é vazio **então retornar** VALOR-DA-MAIORIA(*exemplos*)
senão
melhor <- ESCOLHER-ATRIBUTO(*atributos, exemplo*)
árvore <- uma nova ad com teste 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



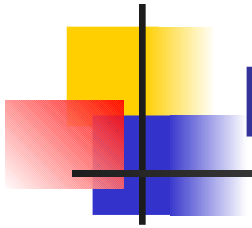
O algoritmo poderia encontrar uma ad diferente para o mesmo conjunto de treinamento?

Qual seria?



A árvore resultante

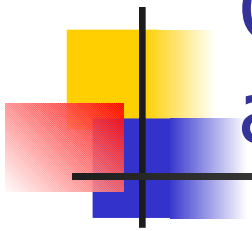
- É diferente da árvore original
- Mas a hipótese concorda com todos os exemplos
- E é consideravelmente mais simples do que a árvore original
- *Chovendo* e *Reserva* ficaram de fora por a árvore não necessita deles para classificar os exemplos
- Mas nunca viu um caso de espera de 0-10 min e o restaurante cheio
  - Para um caso onde faminto é falso a ad informa que não devemos esperar



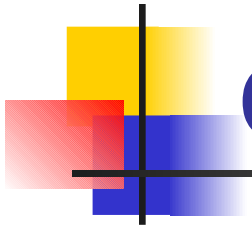
Escolha de testes de atributos

- 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



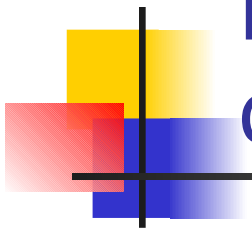
- Como medimos um "atributo muito bom" ou um "atributo realmente inútil"
- A medida
  - Deve ter o valor máximo quando o atributo é perfeito
  - E o valor mínimo quando o atributo for completamente inútil
- Uma medida apropriada seria a **quantidade** esperada de **informações** fornecidas pelo atributo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



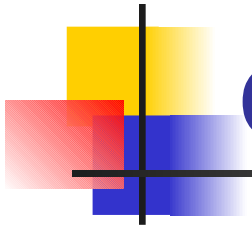
O algoritmo de aprendizagem

- 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
- Aplicar o algoritmo ao conjunto de treinamento, gerando uma hipótese *h*
- Medir a quantidade de exemplos do conjunto de teste classificados corretamente por *h*
5. Repetir as etapas 2 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