Inteligência Artificial

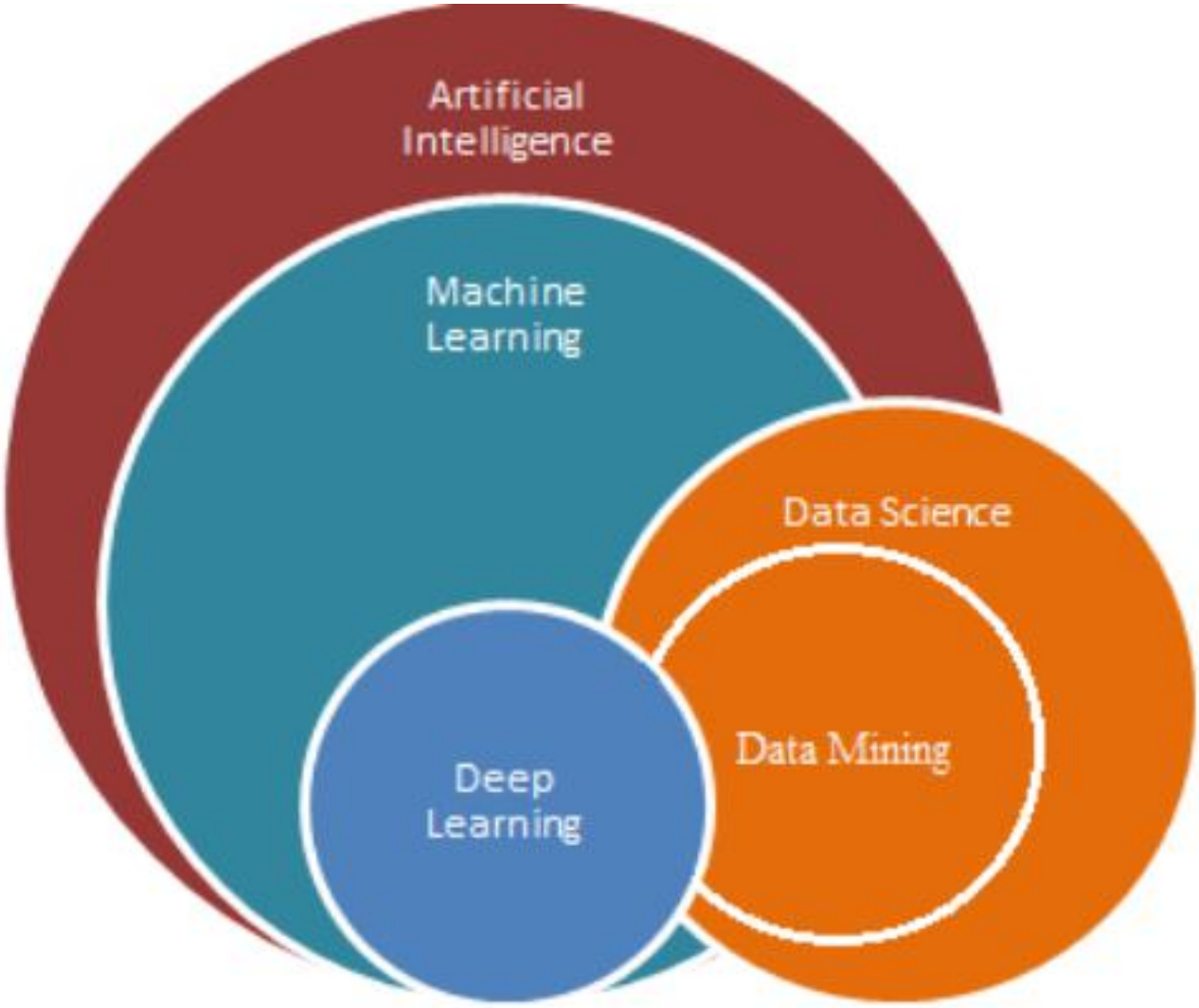
Introdução a Aprendizado de Máquina

Aprendizado – por que?

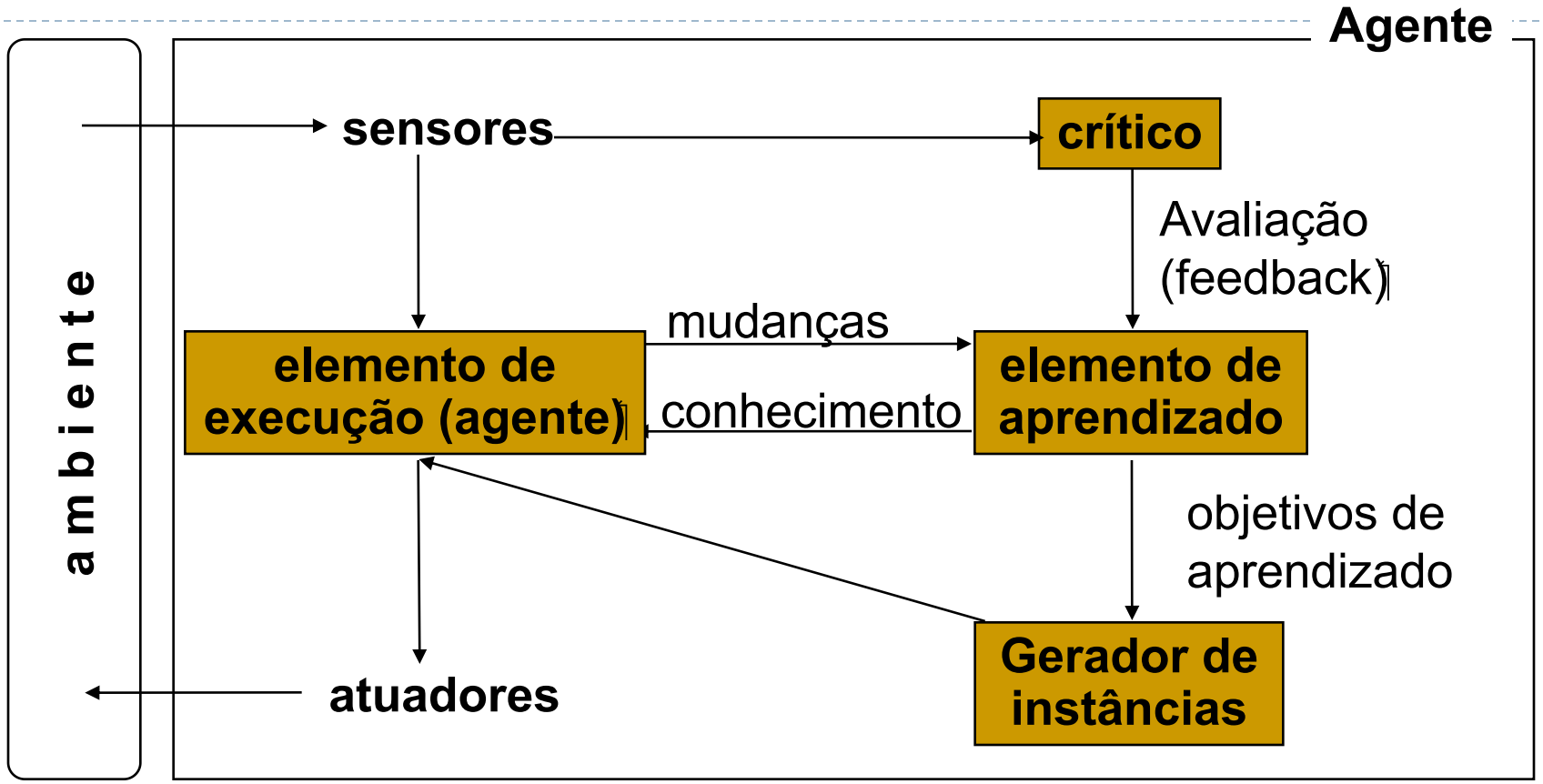
- Capacidade de aprender é parte fundamental do conceito de inteligência.
- Um agente aprendiz é mais flexível ➔ aprendizado permite lidar com situações novas (mundo é dinâmico). Dá **autonomia** ao agente.
- Aprendizado facilita tarefa do projetista ➔ programar apenas o essencial

Como construir programas (agentes) que automaticamente melhoram com a experiência?

A "Reasonable" Graph Representation of Intersections of Related Areas to AI



Agente que aprende



- **Aprendizado:** processo de modificação dos parâmetros do agente, de modo a maximizar uma medida de desempenho.

Um Modelo Geral

- Ambiente / Sensores / Atuadores
- Elemento de Execução: executa as ações de acordo com medida geral de desempenho.
- Crítico: comunica ao EA quão bem ou mal o agente está operando, de acordo com um critério fixo. Observações sensoriais nem sempre são boas indicadoras...
- Elemento de Aprendizado: armazena informação sobre como a modificação dos parâmetros do EE deve ser feita (algoritmos, estruturas de dados, medida de desempenho, conhecimentos a priori, etc).
- Gerador de Instâncias: sugere ações alternativas que podem ser tomadas pelo agente, com o fim de adquirir informação adicional.

Aprendizado - paradigmas

- Aprendizado supervisionado
  - O crítico comunica a EA o erro relativo entre a ação que deve ser tomada idealmente pelo EE e a ação efetivamente escolhida pelo agente. Pares (corretos) de entrada/saída podem ser observados (ou demonstrados por um supervisor).
- Aprendizado por reforço
  - O crítico comunica apenas uma indicação de desempenho (geralmente, indicação de quão bom ou ruim é o estado resultante), normalmente de modo intermitente e apenas quando situações dramáticas são atingidas (*feedback* indireto, com retardo).
- Aprendizado não-supervisionado
  - O crítico não envia nenhum tipo de informação ao EA, não há "pistas" sobre as saídas corretas (geralmente utiliza-se regularidades, propriedades estatísticas dos dados sensoriais)
    - Busca-se encontrar padrões ou estruturas / agrupamentos nos dados. Inclui por exemplo técnicas de clusterização

Ambientes e Aprendizado

- Técnicas de Aprendizado podem ser aplicadas a ambientes complexos: parcialmente observável, estocástico, seqüencial, dinâmico, contínuo e multiagente
- Mas via de regra, foca-se no caso **monoagente** pela suposição implícita de problema i.i.d (independente e identicamente distribuído). O que não ocorre no caso multiagente
  - Independente, significa: $P(E_j \mid E_{j-1}, E_{j-2} .....) = P(E_j)$
  - Identificamente: $P(E_j) = P(E_{j-1}) = P(E_{j-2}) = ...$
- Vamos assumir a hipótese i.i.d. salvo explicitamente dito o contrário....
- Além de casos multiagentes, o que mais poderia ser citado como **não** i.i.d?

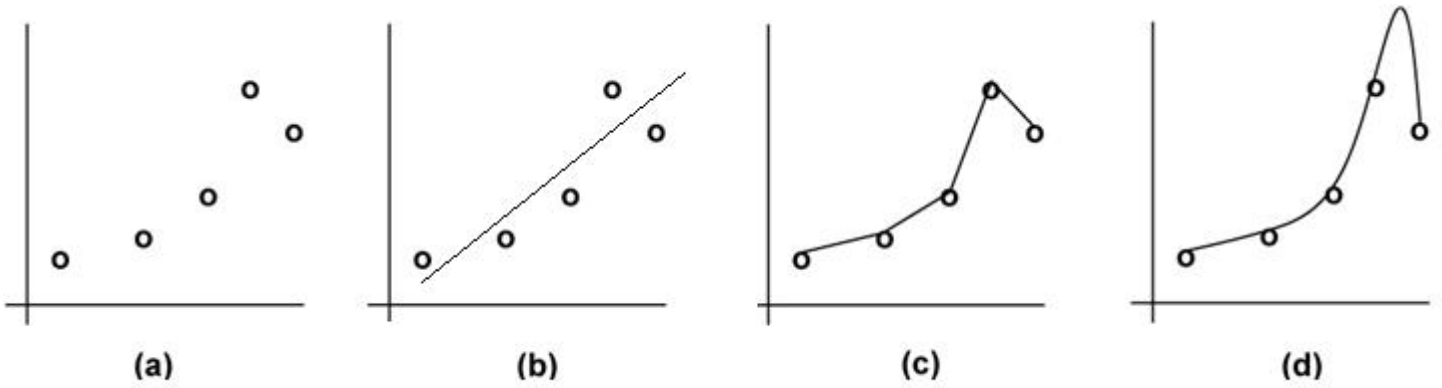
Como aprender?

- Infererir uma regra geral (hipótese) a partir de exemplos particulares (generalização).
- Aprendizado indutivo é uma forma de aprendizado supervisionado

- Precisão diretamente proporcional à quantidade de exemplos. Abordagens:
  - **não-incremental**: gera hipótese a partir de todo o conjunto de exemplos
    - eficiente, conceitualmente simples... Porém não é aplicável a muitos problemas práticos!
  - **incremental**: atualiza hipótese a cada novo exemplo
    - mais flexível, algoritmos *anytime*... Porém a ordem de apresentação é importante!

Aprendizado Indutivo

- *Problema: Dado um conjunto de exemplos de f, retornar uma função h que aproxime f*
- *x*: entrada; *f(x)*: saída desejada
- Exemplo (par de treinamento) = *(x, f(x))*
- Objetivo: aprender uma função *h* (hipótese) que aproxime *f*.

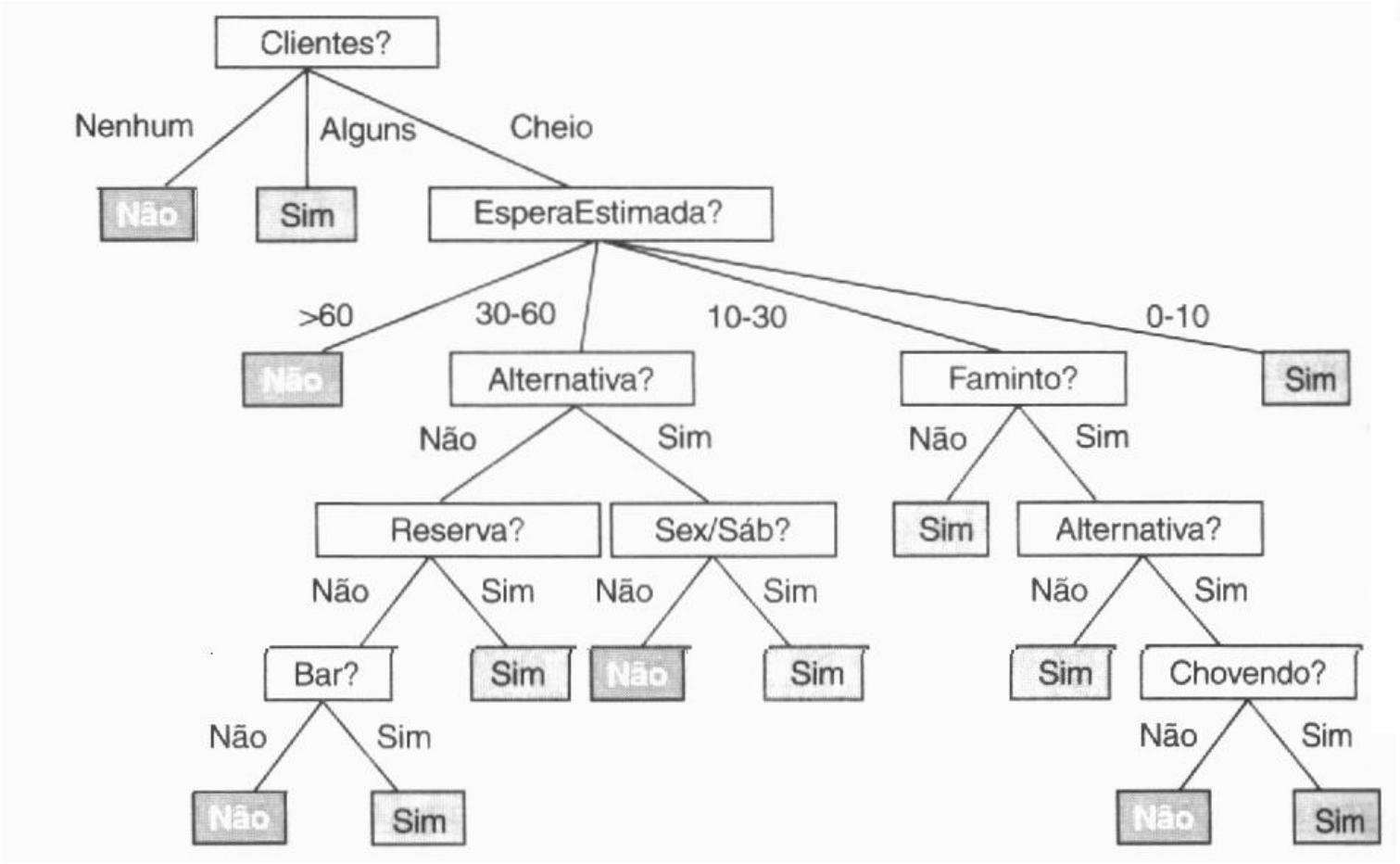


Técnica de aprendizado: Árvores de Decisão

- Entrada: situação descrita por um conjunto de propriedades;
- Saída: "decisão" SIM / NÃO
  - Podem ser criadas variantes com outros valores de saída
- Função aprendida: representada por uma árvore de decisão (ou conjunto de regras IF-THEN)
- Árvore de decisão representa funções booleanas
  - Nós não folha: testes para propriedades.
  - Ramos: valores dos testes.
  - Nó folha: decisão ou classe

Árvore de Decisão: Exemplo

- Problema: esperar por uma mesa vazia?



Aprendizado de Árvores de Decisão

Exemplo	Atributos										Meta
	Alt	*Bar*	*Sex*	*Fam*	*Cli*	*Preço*	Chuva	Res	Tipo	Estim	*VaiEsperar*
X_1	Sim	Não	Não	Sim	Alguns	$$$	Não	Sim	francês	0-10	Sim
X_2	Sim	Não	Não	Sim	Cheio	$	Não	Não	tailandês	30-60	Não
X_3	Não	Sim	Não	Não	Alguns	$	Não	Não	hambúrguer	0-10	Sim
X_4	Sim	Não	Sim	Sim	Cheio	$	Sim	Não	tailandês	10-30	Sim
X_5	Sim	Não	Sim	Não	Cheio	$$$	Não	Sim	francês	>60	Não
X_6	Não	Sim	Não	Sim	Alguns	$$	Sim	Sim	italiano	0-10	Sim
X_7	Não	Sim	Não	Não	Nenhum	$	Sim	Não	hambúrguer	0-10	Não
X_8	Não	Não	Não	Sim	Alguns	$$	Sim	Sim	tailandês	0-10	Sim
X_9	Não	Sim	Sim	Não	Cheio	$	Sim	Não	hambúrguer	>60	Não
X_{10}	Sim	Sim	Sim	Sim	Cheio	$$$	Não	Sim	italiano	10-30	Não
X_{11}	Não	Não	Não	Não	Nenhum	$	Não	Não	tailandês	0-10	Não
X_{12}	Sim	Sim	Sim	Sim	Cheio	$	Não	Não	hambúrguer	30-60	Sim

- Um exemplo = valores de atributos, objetivo.
- Objetivo positivo/negativo: exemplo positivo/negativo
- Idéia "ingênua": construir árvore com um caminho para cada exemplo. Mas e se aparecer um outro exemplo?
- Memorização de observações → generalização inexistente!

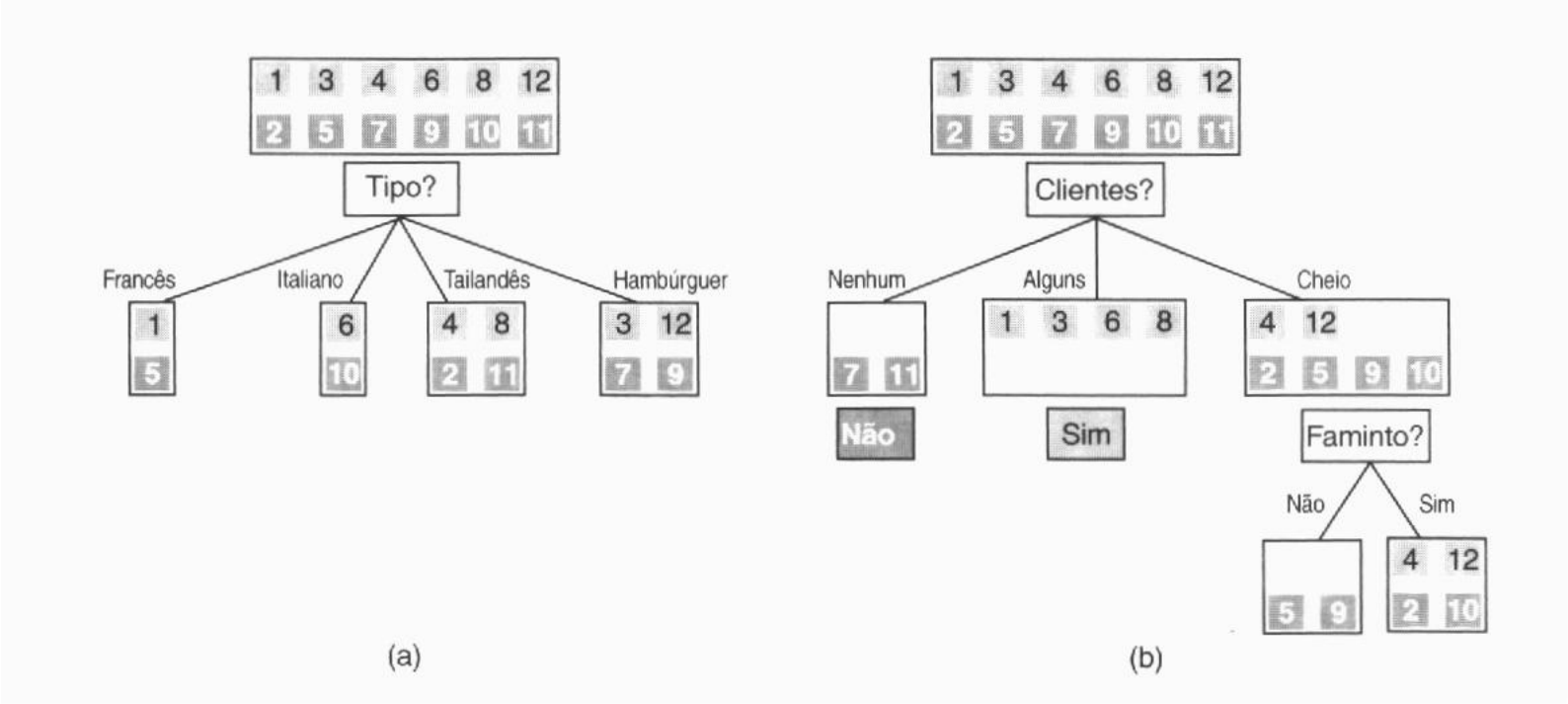
Princípio da Navalha de Occam

- A melhor hipótese é a mais simples entre aquelas consistentes com os pares de treinamento.

- Implementação recursiva:
  - Testo primeiro atributo que permita decisão imediata. Caso este não exista, testo aquele com maior poder de categorização, ou de acordo com algum outro critério, e divido a coleção de exemplos entre positivos e negativos para o atributo.
  - Exemplos restantes todos positivos (ou negativos): fim.
  - Nenhum atributo restante: uso classificação da maioria dos exemplos do nó pai.

  - Exemplos restantes, e nenhum atributo restante: exemplos com a mesma descrição e com classificações diferentes. Dados imprecisos (ruído, atributos ocultos).

Dividindo coleção de eventos: exemplos

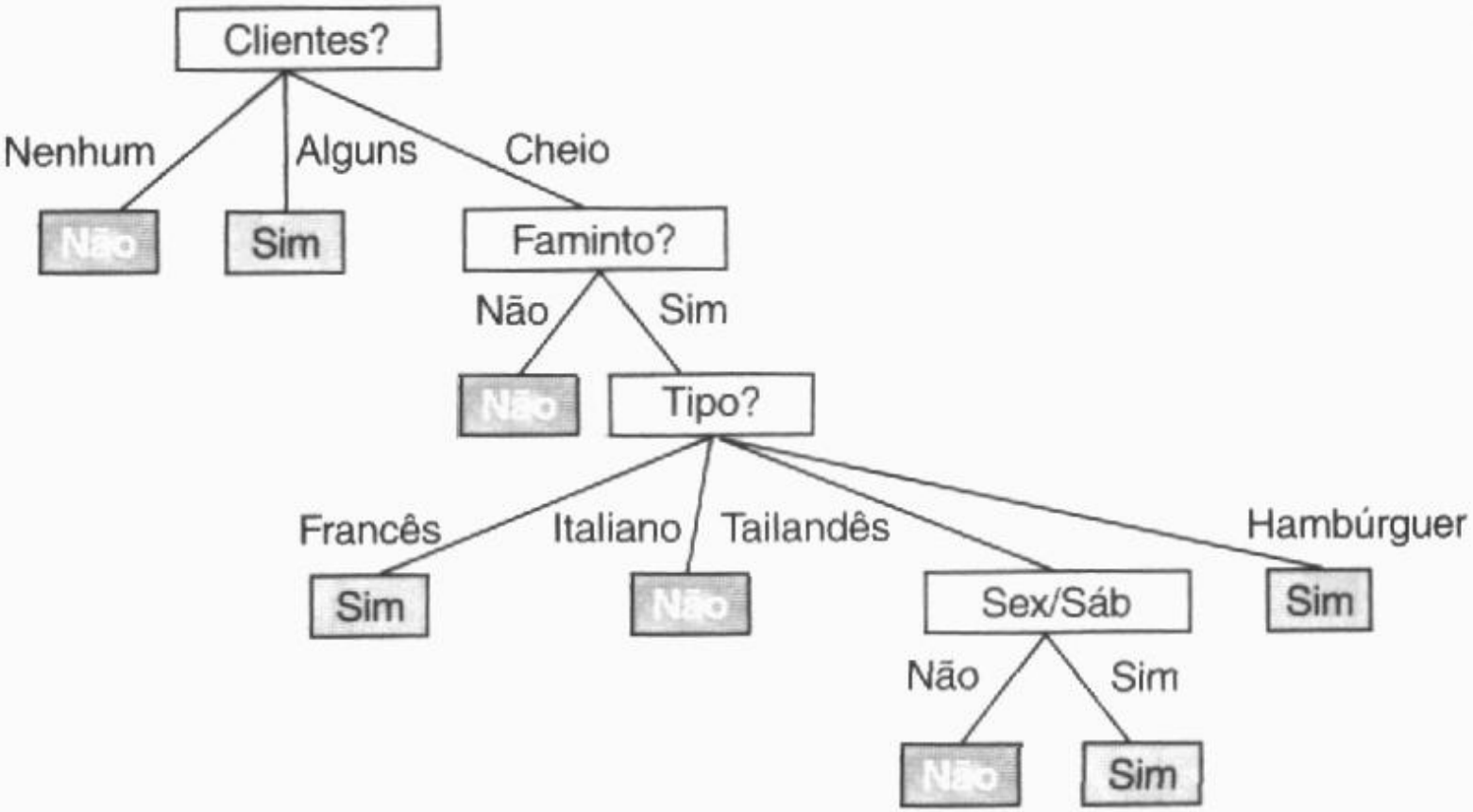


Algoritmo para determinação da árvore

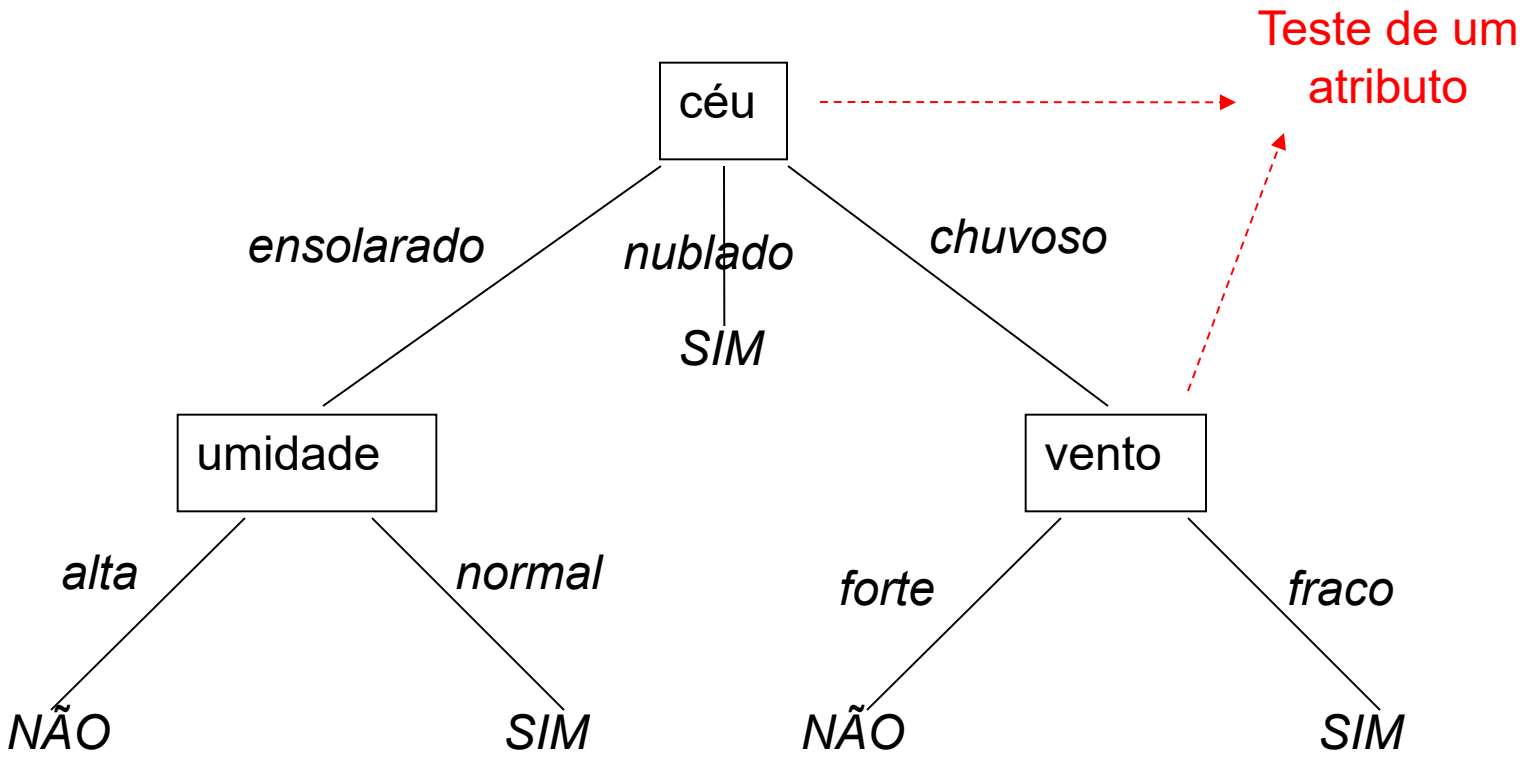
```
função APRENDIZAGEM-EM-ÁRVORE-DE-DECISÃO (exemplos, atributos, padrão) retorna uma árvore de decisão
  entradas: exemplos, conjunto de exemplos
            atributos, conjunto de atributos
            padrão, valor-padrão para o predicado de objetivo

  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, exemplos)
    árvore ← uma nova árvore de decisão 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-ÁRVORE-DE-DECISÃO(exemplos_i, atributos – melhor, m)
      adicionar uma ramificação a árvore com rótulo v_i e subárvore subárvore
  retornar árvore
```

Árvore de decisão induzida



Conceito a aprender: Devo jogar tênis?



Árvore de Decisão: disjunção de conjunções

- Devo jogar tênis quando:

$$(\text{céu} = \text{ensolarado} \wedge \text{umidade} = \text{normal}) \vee (\text{céu} = \text{nublado}) \vee (\text{céu} = \text{chuvoso} \wedge \text{vento} = \text{fraco})$$

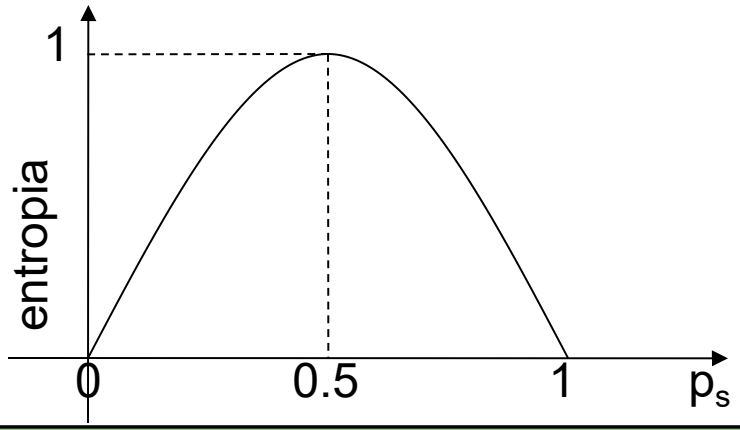
Qual atributo é o melhor classificador?

- Aquele que reduz a probabilidade de errar a classificação (reduz a confusão)...
- Medida baseada em **ganho de informação**, calculado pela **entropia**
- *Entropia*: medida de "impureza" numa coleção de exemplos de treinamento S
  - Entropia = 0: todos membros da mesma classe (todos sim, ou todos não)
  - Entropia = 1:coleção com mesmo número de (sim) e (não)

Entropia em coleção de exemplos S

- *Entropia*: medida de "impureza" numa coleção de exemplos de treinamento S
  - Entropia = 0: todos membros da mesma classe (todos sim, ou todos não)
  - Entropia = 1:coleção com mesmo número de (sim) e (não)

$$Entropia = \frac{-p_s}{p_s + p_n} * \log_2\left(\frac{p_s}{p_s + p_n}\right) + \frac{-p_n}{p_s + p_n} * \log_2\left(\frac{p_n}{p_s + p_n}\right)$$



p_s: No. de exemplos positivos em S

p_n : No. de exemplos negativos em S

Entropia

$$Entropia = \frac{-p_s}{p_s + p_n} * \log_2\left(\frac{p_s}{p_s + p_n}\right) + \frac{-p_n}{p_s + p_n} * \log_2\left(\frac{p_n}{p_s + p_n}\right)$$

- Ex: se S tem 14 exemplos, sendo 9 positivos e 5 negativos, vem:
- Entropia(S) = - (9/14) $\log_2$ (9/14) -
- - (5/14) $\log_2$ (5/14) = 0.940
- Se decisão da árvore pode ter C-valores, é possível generalizar para:

$$Entropia(S) = \sum_{i=1}^{C} \left[\frac{-p_i}{\sum_{j=1}^{C} p_j} * \log\left(\frac{p_i}{\sum_{j=1}^{C} p_j} \right) \right]$$

*OBS: 0*log$_2$ 0 = 0*

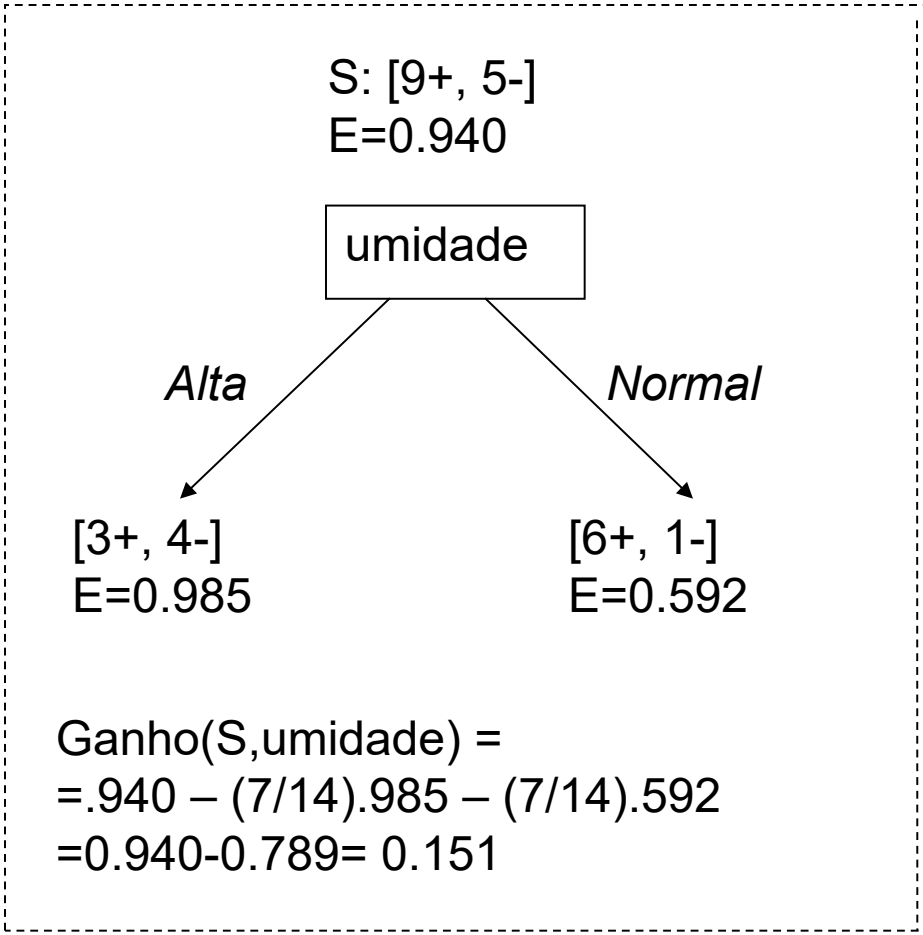
Ganho de Informação

- Mede a redução esperada na entropia, causada pela partição nos exemplos segundo um atributo.
- Ganho(S,A): ganho de informação de um atributo A, relativo à coleção de exemplos S.
- $$Ganho(S,A) = entropia(S) - \sum_{v \in valores(A)} (|Sv| / |S|)\ entropia(Sv)$$
- Sv: subconjunto de S no qual A tem valor v
- |Sv|: número de elementos de Sv
- valores(A): todos possíveis valores do atributo A
- $Sv = \{s \in S \mid A(s) = v\}$

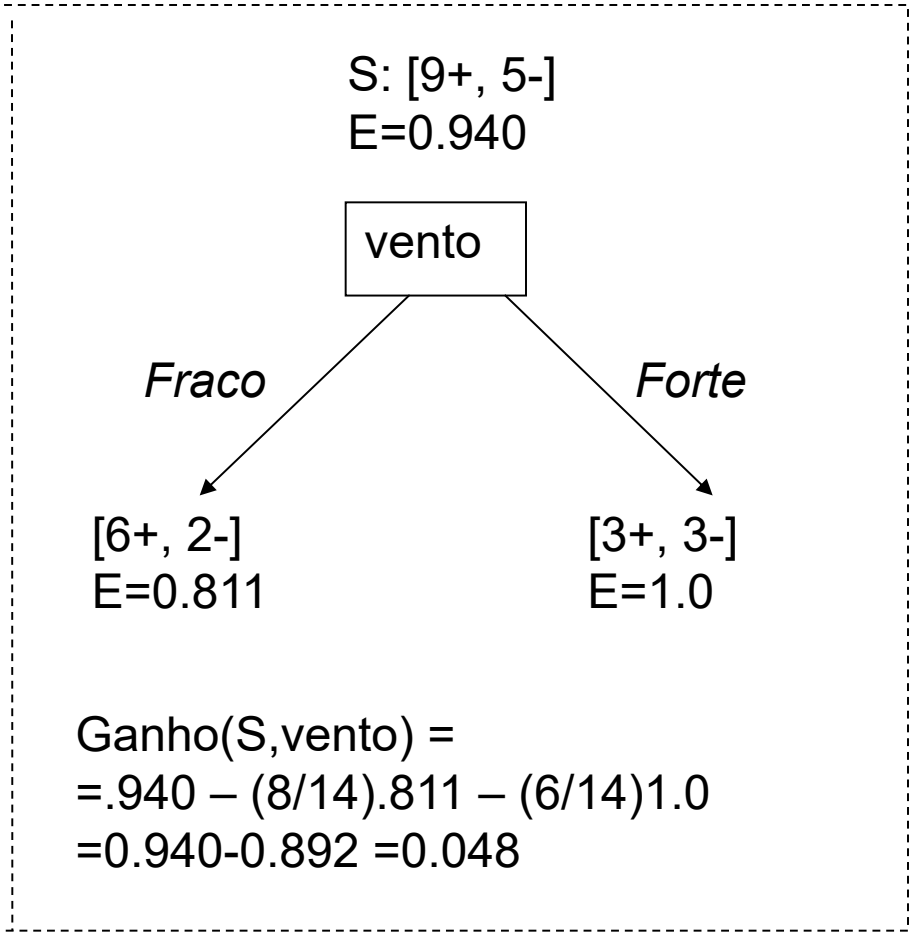
Exemplos de treinamento para o atributo-alvo JogarTênis

Ex	Céu	Temperatura	Umidade	Vento	JogarTênis
X1	Ensolarado	Quente	Alta	Fraco	NÃO
X2	Ensolarado	Quente	Alta	Forte	NÃO
X3	Nublado	Quente	Alta	Fraco	SIM
X4	Chuvoso	Boa	Alta	Fraco	SIM
X5	Chuvoso	Fria	Normal	Fraco	SIM
X6	Chuvoso	Fria	Normal	Forte	NÃO
X7	Nublado	Fria	Normal	Forte	SIM
X8	Ensolarado	Boa	Alta	Fraco	NÃO
X9	Ensolarado	Fria	Normal	Fraco	SIM
X10	Chuvoso	Boa	Normal	Fraco	SIM
X11	Ensolarado	Boa	Normal	Forte	SIM
X12	Nublado	Boa	Alta	Forte	SIM
X13	Nublado	Quente	Normal	Fraco	SIM
X14	Chuvoso	Boa	Alta	Forte	NÃO

Exemplo: qual é o melhor atributo classificador?



Ganho(S,umidade) =
=.940 – (7/14).985 – (7/14).592
=0.940-0.789= 0.151



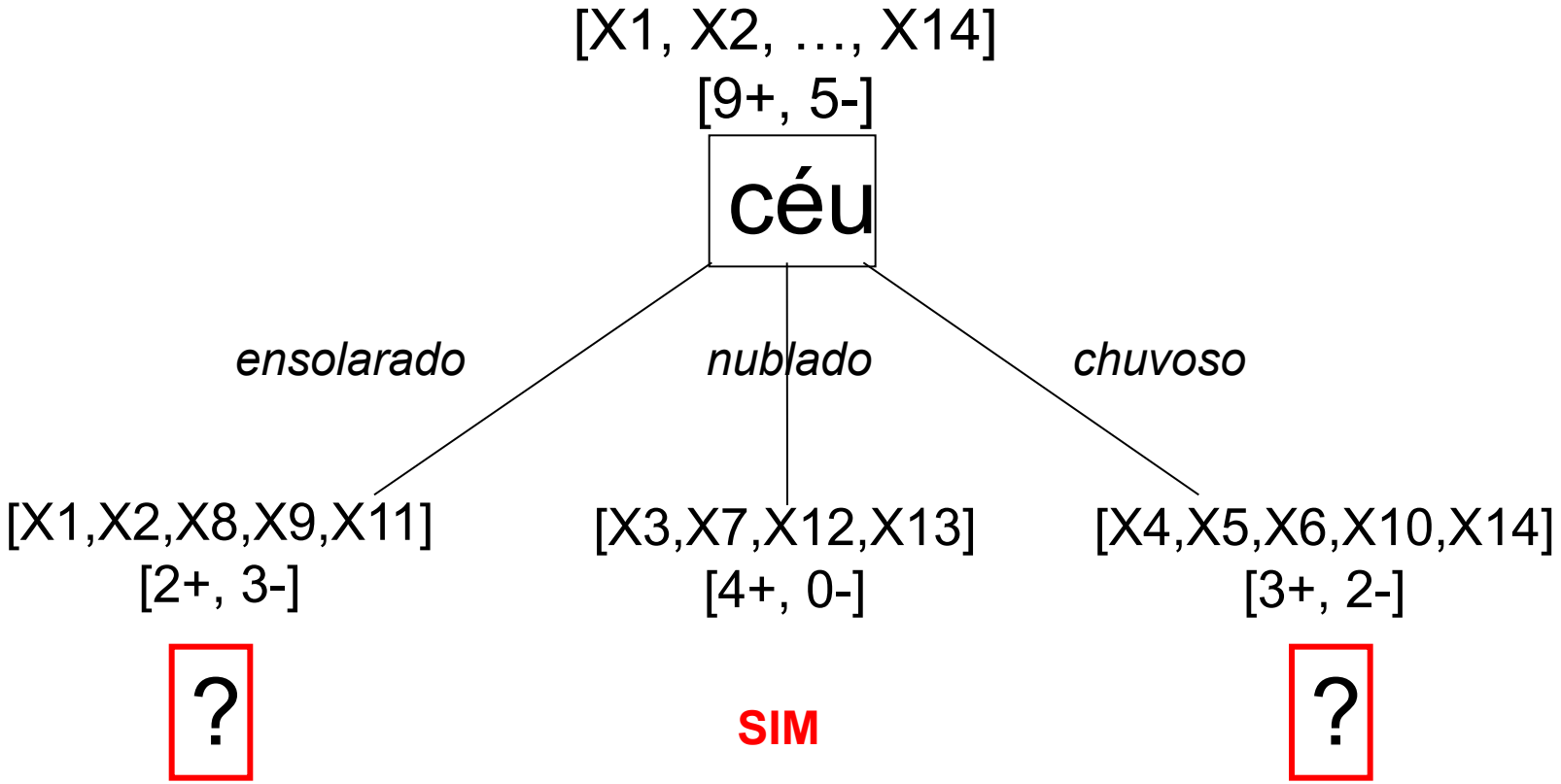
Ganho(S,vento) =
=.940 – (8/14).811 – (6/14)1.0
=0.940-0.892 =0.048

ID3: não-incremental - analisa todos os exemplos para decidir atributo classificador

Construção da árvore de decisão com ID3

Ganho(S,céu) = 0.246; Ganho(S,umidade) = 0.151
Ganho(S,vento) = 0.048; Ganho(S,temperatura) = 0.029

➔ Para S, céu é melhor!



Características do ID3

- Preferência por árvores pequenas:
  - sua busca no espaço de hipóteses aumenta a árvore somente até o tamanho necessário para classificar o conjunto de exemplos de treinamento disponível.
- Coloca mais perto da raiz aqueles atributos que oferecem o maior ganho de informação.

Problemas gerais

- Estratégia de aumentar a árvore o mínimo necessário pode trazer problemas quando:
  - Há ruído nos dados;
  - Número de exemplos de treinamento é pequeno (não representativo da função buscada)
- Problema: ruído nos dados
  - Ex: dois ou mais exemplos com mesma descrição (em termos dos atributos), mas classificação diferente.
  - Soluções possíveis: (i) cada folha é rotulada com a classificação majoritária, (ii) folhas indicam probabilidade de ocorrência de cada classificação (relativo à freqüência da classificação).

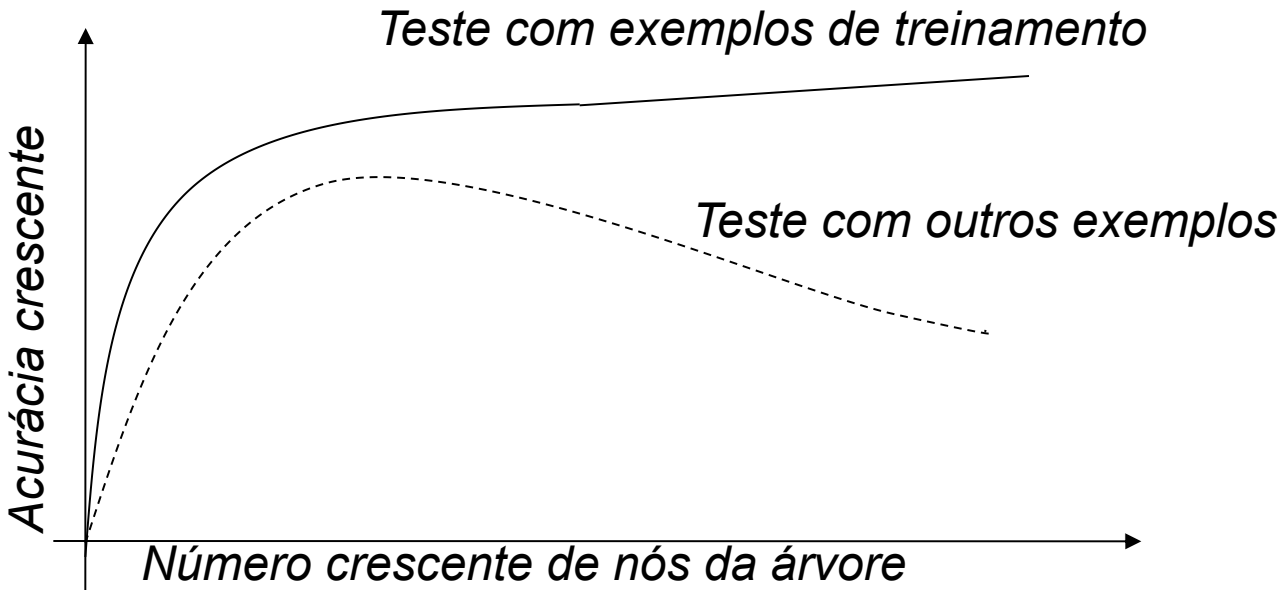
Overfitting (super-especialização)

➔ problema em todos algoritmos de aprendizado!!

Definição: dado um espaço de hipóteses H, uma hipótese $h \in H$ super-especializa (*overfits*) os dados de treinamento se existir uma outra hipótese $h' \in H$, tal que h tem menor erro que h' no conjunto de treinamento, mas h' tem um menor erro que h sobre a distribuição total de instâncias (incluindo instâncias fora do conjunto de treinamento).

- ➔ *Como detectar atributos irrelevantes?*
- ➔ *Quão grande deve ser o ganho de informação para que o correspondente atributo seja um nó na árvore?*

Impacto do *overfitting* num aprendizado por árvore de decisão:



Conforme ID3 adiciona mais nós para crescer a árvore de decisão, a acurácia da árvore, medida sobre os exemplos de treinamento, cresce monotonicamente. Entretanto, quando medida sobre um conjunto de dados independente do conjunto de treinamento, a acurácia primeiro cresce e, depois, decresce.

Overfitting – Solução 1

- Parar de crescer a árvore antes de alcançar o ponto de classificação "perfeita" dos exemplos de treinamento.
  - ➔ OK, mas: quando parar?
  - **Validação cruzada**: tenta estimar quão bem a hipótese corrente irá predizer dados ainda não recebidos ("vistos").

Validação Cruzada

- Algoritmo

  1) Divide o conjunto de exemplos em dois sub-conjuntos: conjuntos de treinamento (CT) e de validação (CV)¶

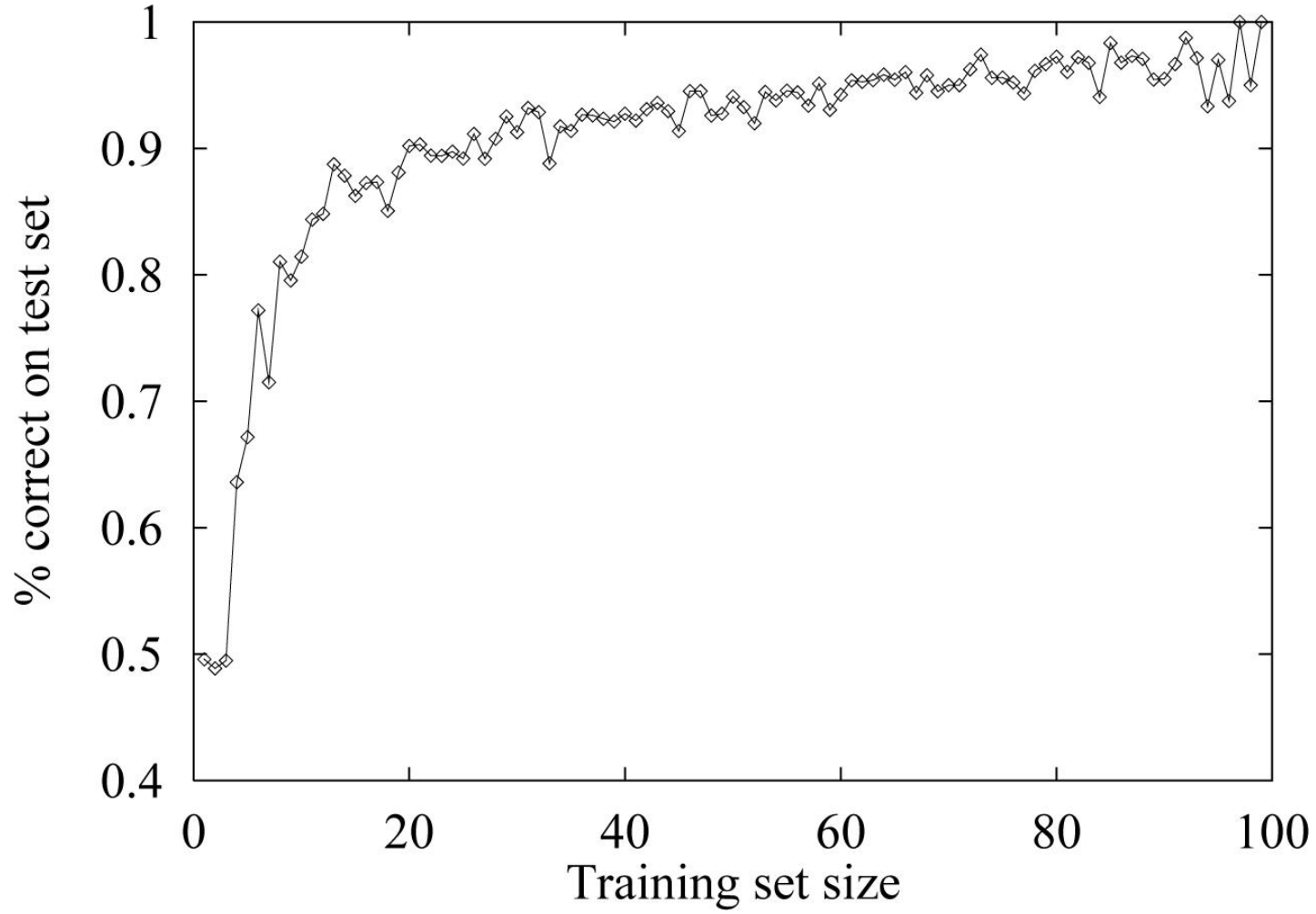
  2) Usa indução para gerar hipótese H sobre CT

  3) Mede percentagem de erro de H aplicada à CV

  4) Repete passos 1-3 com diferentes tamanhos de CV e CT, e tendo elementos escolhidos aleatoriamente

- Pode-se calcular a média com os dados resultantes e determinar a curva de aprendizado para o domínio em questão. Deseja-se que a qualidade da predição cresça com o tamanho do conjunto de treinamento.

Curva de Aprendizado



Cross-validation

- *K-fold cross-validation* avoids overlapping test sets
  - First step: split data into *k* subsets of equal size
  - Second step: use each subset in turn for testing, the remainder for training
  - This means the learning algorithm is applied to *k* different training sets
- Often the subsets are stratified before the cross-validation is performed to yield stratified *k*-fold cross-validation
- The error estimates are averaged to yield an overall error estimate; also, standard deviation is often computed
- Alternatively, predictions and actual target values from the *k* folds are pooled to compute one estimate
  - Does not yield an estimate of standard deviation

More on cross-validation

- Standard method for evaluation: stratified ten-fold cross-validation
- Why ten?
  - Extensive experiments have shown that this is the best choice to get an accurate estimate
  - There is also some theoretical evidence for this
- Stratification reduces the estimate's variance
- Even better: repeated stratified cross-validation
  - E.g., ten-fold cross-validation is repeated ten times and results are averaged (reduces the variance)

Overfitting – Solução 2

- Abordagens que provoquem o *overfitting* e depois podam a árvore (*pruning*)¶
  - Método do Erro Reduzido:
    - considera cada nó como candidato a folha (elimina sub-árvore abaixo dele), com classificação a ele associada como a mais comum; o nó se torna folha (nova árvore) sempre que a acurácia da classificação não diminuir em relação à árvore original, usando o conjunto de validação.

Aplicações

- GASOIL
  - Sistema de separação de gás-óleo em plataformas de petróleo
  - Construção do sistema especialista para tal projeto usaria 10 pessoas-ano (aproximadamente 2500 regras!)
  - Desenvolvido em 100 pessoas-dia, usando aprendizado de árvore de decisão.
- Piloto automático de um Cessna
  - Treinado por três pilotos
  - Obteve um desempenho melhor que os três
- Mineração de dados
- Recuperação de Informação
- Classificação de imagens, etc.

Regressão e Classificação

- Regressão é o problema clássico de calibrar os parâmetros de uma função (linear ou não linear) para ajustar os dados de um conjunto de treinamento. O aprendizado de máquina em ambientes contínuos pode ser visto como um caso de regressão não-linear. Sob esse ponto de vista, é isto que fazem **redes neurais** (regressão não linear)

- Chama-se de problema de **Classificação** aqueles que podem ser visto como a atribuição de uma classe (ou rótulo) a um determinado exemplar do qual se conhece algumas caracteristicas.

- Vejamos algumas medidas de qualidade para classificadores e regressores

Classificação e Matriz de Confusão

- Com classe binária

		Predicted class: yes	Predicted class: no
Actual class	yes	true positive	false negative
	no	false positive	true negative

- Exemplo com classe de vários (3) valores

		Predicted class: a	b	c	Total
Actual class	a	88	10	2	100
	b	14	40	6	60
	c	18	10	12	40
	Total	120	60	20	

Análise de Classificadores

- Não é difícil criar classificadores, mas pode ser muito difícil criar bons classificadores...

- Isso traz a questão: O que exatamente é um bom classificador?
- Taxa de acerto (acurácia)
  - Tx. Ac=(TP+TN)/(TP+TN+FP+FN) ou
  - Tx. Ac= ∑ (Elem da diagonal principal) / ∑ (Elem da matriz de confusão)
- É sempre uma boa medida?
  - Considere um sistema de detecção de fraude em transações de cartão de crédito que sempre diz que não há fraude….A taxa de acerto seria alta ou baixa? É um bom classificador?

Outras medidas...

- Fração dos itens classificados como verdadeiro que o são realmente: ***precision***=TP/(TP+FP)
- Fração dos itens classificados como verdadeiro sobre o total dos itens que **são** verdadeiros:
  recall =TP/(TP+FN) (a.k.a TP rate, sensitivity)

- Em diferents domínios, uma medida pode ter outros nomes. Em medicina, o termo **sensitivity** é usado ao invés de **recall**.
  - **Sensitivity**: proporção de pessoas com doença que tem um teste positivo: (TP / (TP + FN)). Quanto maior, melhor
  - **FP rate**: proporção de pessoas sem doença que tem um resultado positivo no teste: (FP/ (FP+TN)). Para a FP rate, quanto menor, melhor

Outras medidas 2...

- **Specificity**: proporção das pessoas sem doença que tem um resultado de teste negativo: (TN/ (FP + TN)). It is equal to (1-**FP rate**). Quanto maior, melhor.

- As vezes, utiliza-se o produto *sensitivity* × specificity como uma medida de qualidade

- *Também usada:* *F-measure*=(2 × recall × precision)/(recall+precision)
  - Equivalente a 2*TP/(2*TP+FN+FP)

- And the old success rate : (TP+TN)/(TP+FP+TN+FN)

Análise de Classificadores -2

- O problema com a taxa de acerto (e outras..) é que não levam em consideração os acertos por puro acaso..
- Uma alternativa: estatística kappa (Cohen's kappa)
  - $\kappa = (p_o - p_e)/(1-p_e)$
  - p_o é a concordância observada
  - p_e é a concordância esperada
- Kappa mensura o ganho em relação a distribuição esperada aleatória, 0 significa que não faz melhor do que ela e 1 significa perfeita acurácia.
- Um problema da estatística kappa (e também da taxa de acerto) é que não leva em consideração o custo dos erros...que podem ser diferentes, com erros bem mais significativos que outros
- Cada parâmetro de comparação é parcial e deve-se fazer uma análise vários parâmetros (há vários outros...precision, f-measure,etc) considerando as particularidades do domínio do problema

Exemplo

- O classificador A prevê uma distribuição de classes com <0.6;0.3;0.1> e se construímos um classificador com essa **distribuição esperada,** qual seria a matriz de confusão?

		Predicted class			
		a	b	c	Total
Actual class	a	88	10	2	100
	b	14	40	6	60
	c	18	10	12	40
	Total	120	60	20	

(a) Classifier A

		Predicted class			
		a	b	c	Total
Actual class	a	60	30	10	100
	b	36	18	6	60
	c	24	12	4	40
	Total	120	60	20	
		60%	30%	10%	

(b) Expected Distribution

- p_o = 88+ 40+12 = 140/200 (acurácia observada)
- p_e = 60+18+4= 82/200 (acurácia com a distribuição esperada)
- $\kappa = (p_o - p_e)/(1-p_e) = (140-82)/(200-82) = 52/118 = 49{,}2\%$

Comparação de classificadores

- *Kappa* statistic: (success rate of actual predictor - success rate of random predictor) / (1 - success rate of random predictor)

- Measures relative improvement on random predictor: 1 means perfect accuracy, 0 means we are doing no better than random

Associando custo aos erros

Default cost matrixes: (a) a two-class case and (b) a three-class case.

		Predicted class	
		yes	no
Actual class	yes	0	1
	no	1	0

(a)

		Predicted class		
		a	b	c
Actual class	a	0	1	1
	b	1	0	1
	c	1	1	0

(b)

When error is not uniform?

- Problem: Predicting return of financial investment (low, neutral, high). Is it uniform?

$$MatrizCustodeErro = \begin{bmatrix} L2 & L1 & N & H1 & H2 & \\ 0 & 1 & 1 & 1 & 1 & L2 \\ 1 & 0 & 1 & 1 & 1 & L1 \\ 1 & 1 & 0 & 1 & 1 & N \\ 1 & 1 & 1 & 0 & 1 & H1 \\ 1 & 1 & 1 & 1 & 0 & H2 \end{bmatrix}$$

Custos podem ser diferentes para erros diferentes...

$$MatrizCustoErroAjustado = \begin{bmatrix} L2 & L1 & N & H1 & H2 & \\ 0 & c_{1,2} & c_{1,3} & c_{1,4} & c_{1,5} & L2 \\ c_{2,1} & 0 & c_{2,3} & c_{2,4} & c_{2,5} & L1 \\ c_{3,1} & c_{3,2} & 0 & c_{3,4} & c_{3,5} & N \\ c_{4,1} & c_{4,2} & c_{4,3} & 0 & c_{4,5} & H1 \\ c_{5,1} & c_{5,2} & c_{5,3} & c_{5,4} & 0 & H2 \end{bmatrix}$$

- A taxa de erro ajustada é calculada ponderando o erro por seu custo e selecionar a decisão que minimiza o custo do erro

Given the errors and hits....

$$DistribuicacaoClassificador = \begin{bmatrix} L2 & L1 & N & H1 & H2 & \\ d_{1,1} & d_{1,2} & d_{1,3} & d_{1,4} & d_{1,5} & L2 \\ d_{2,1} & d_{2,2} & d_{2,3} & d_{2,4} & d_{2,5} & L1 \\ d_{3,1} & d_{3,2} & d_{3,3} & d_{3,4} & d_{3,5} & N \\ d_{4,1} & d_{4,2} & d_{4,3} & d_{4,4} & d_{4,5} & H1 \\ d_{5,1} & d_{5,2} & d_{5,3} & d_{5,4} & d_{5,5} & H2 \end{bmatrix}$$

$$ErroMedioClassificador = \frac{1}{\tau} * \frac{\sum_i \sum_j c_{i,j} * d_{i,j}}{c_{max}}$$

$$TaxaAcertoAjustada = 1 - \frac{1}{\tau} * \frac{\sum_i \sum_j c_{i,j} * d_{i,j}}{c_{max}}$$

Cost-sensitive classification

- Can take costs into account when making predictions
  - Basic idea: only predict high-cost class when very confident about prediction
- Given: predicted class probabilities
  - Normally, we just predict the most likely class
  - Here, we should make the prediction that minimizes the expected cost
    - Expected cost: dot product of vector of class probabilities and appropriate column in cost matrix
    - Choose column (class) that minimizes expected cost
- This is the minimum-expected cost approach to cost-sensitive classification

Cost-sensitive learning

- Most learning methods do not perform cost-sensitive learning
  - They generate the same classifier no matter what costs are assigned to the different classes
  - Example: standard decision tree learner
- Simple methods for cost-sensitive learning:
  - Resampling of instances according to costs
  - Weighting of instances according to costs

Evaluating Regression (numeric prediction)

- Same strategies: independent test set, cross-validation, significance tests, etc.
- Difference: error measures
- Actual target values: $a_1\ a_2\ \ldots a_n$
- Predicted target values: $p_1\ p_2\ \ldots\ p_n$
- Most popular measure: *mean-squared error*

$$\frac{(p_1-a_1)^2+\cdots+(p_n-a_n)^2}{n}$$

  - Easy to manipulate mathematically

Other measures

- The *root mean-squared error* :

$$\sqrt{\frac{(p_1 - a_1)^2 + \cdots + (p_n - a_n)^2}{n}}$$

- The *mean absolute error* is less sensitive to outliers than the mean-squared error:

$$\frac{|p_1 - a_1| + \cdots + |p_n - a_n|}{n}$$

- Sometimes *relative* error values are more appropriate (e.g. 10% for an error of 50 when predicting 500)

Improvement on the mean

- How much does the scheme improve on simply predicting the average?
- The *relative squared error* is:

$$\frac{(p_1 - a_1)^2 + \cdots + (p_n - a_n)^2}{(a_1 - \bar{a})^2 + \cdots + (a_n - \bar{a})^2}$$

(in this formula and the following two, $\bar{a}$ is the mean value over the training data)

- The *root relative squared error* and the *relative absolute error* are:

$$\sqrt{\frac{(p_1 - a_1)^2 + \cdots + (p_n - a_n)^2}{(a_1 - \bar{a})^2 + \cdots + (a_n - \bar{a})^2}} \qquad \frac{|p_1 - a_1| + \cdots + |p_n - a_n|}{|a_1 - \bar{a}| + \cdots + |a_n - \bar{a}|}$$

Correlation coefficient

- Measures the *statistical correlation* between the predicted values (p) and the actual values (a).
- $$\frac{S_{PA}}{\sqrt{S_P S_A}}, \text{ where } S_{PA} = \frac{\sum_i (p_i - \bar{p})(a_i - \bar{a})}{n-1}, S_P = \frac{\sum_i (p_i - \bar{p})^2}{n-1},$$
  $$S_A = \frac{\sum_i (a_i - \bar{a})^2}{n-1} \text{ (here, } \bar{a} \text{ is the mean value over the test data)}$$
- Scale independent, between –1 and +1. Zero means no correlation at all. 1 perfect correlation. Negative values mean correlated negatively
- Good performance leads to large values!

Which measure?

- Best to look at all of them
- Often it doesn't matter
- Example:

	A	B	C	D
Root mean-squared error	67.8	91.7	63.3	57.4
Mean absolute error	41.3	38.5	33.4	29.2
Root rel squared error	42.2%	57.2%	39.4%	35.8%
Relative absolute error	43.1%	40.1%	34.8%	30.4%
Correlation coefficient	0.88	0.88	0.89	0.91

- D best
- C second-best
- A, B arguable

Model selection criteria

- Model selection criteria attempt to find a good compromise between:
  - The complexity of a model
  - Its prediction accuracy on the training data
- Reasoning: a good model is a simple model that achieves high accuracy on the given data
- Also known as *Occam's Razor* :
  the best theory is the smallest one
  that describes all the facts

Elegance vs. errors

- Theory 1: very simple, elegant theory that explains the data almost perfectly
- Theory 2: significantly more complex theory that reproduces the data without mistakes
- Theory 1 is probably preferable
- Classical example: Kepler's three laws on planetary motion
  - Less accurate than Copernicus's latest refinement of the Ptolemaic theory of epicycles on the data available at the time

Errors and Machine Learning Models

- ML models generally suffer from three types of errors:
  - **Bias:** This error is caused by unrealistic assumptions.
    - When bias is high, the ML algorithm has failed to recognize important relations between features and outcomes.
    - In this situation, the algorithm is said to be "underfit."
  - **Variance:** This error is caused by sensitivity to small changes in the training set.
    - When variance is high, the algorithm has **overfit** the training set, and that is why even minimal changes in the training set can produce wildly different predictions.
    - Rather than modelling the general patterns in the training set, the algorithm has mistaken noise with signal.
  - **Noise**: This error is caused by the variance of the observed values, like unpredictable changes or measurement errors.
    - This is the irreducible error, which cannot be explained by any model

Ensemble Methods

- An ensemble method is a method that combines a set of weak learners, all based on the same learning algorithm, in order to create a (stronger) learner that performs better than any of the individual ones.

- Ensemble methods help reduce bias and/or variance.

Main Ensemble Methods

- Bagging (Bootstrap aggregation)
- Random Forest
- Boosting

Bagging (Bootstrap aggregation)

- Bootstrap aggregation (bagging) is an effective way of reducing the variance in forecasts.
- It works as follows:
  - First, generate *N* training datasets by random sampling *with replacement.*
  - Second, fit *N* estimators, one on each training set. These estimators are fit independently from each other, hence the models can be fit in parallel.
  - Third, the ensemble forecast is the *simple* average of the individual forecasts from the *N* models.
- In the case of multinomial variables, the probability that an observation belongs to a class is given by the proportion of estimators that classify that observation as a member of that class (majority voting).

Bagging vs Boosting (Training sets)

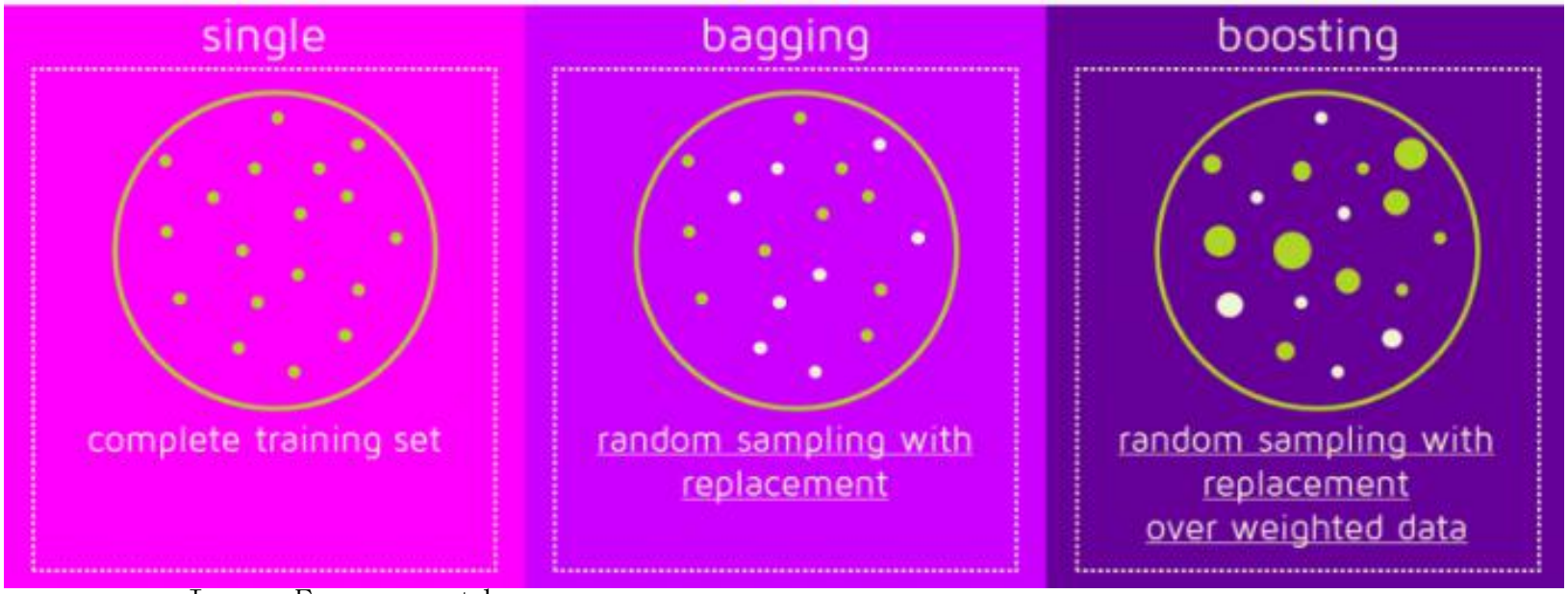


Image From quantdare.com

Boosting

- Unlike Bagging, Boosting tries to create new models to do well where previous models failed
- It does that by defining weights for the data to favor the most difficult cases
- In boosting, It is used an weighted average rather than an equally weighted average
- Both are good to reduce variance and give more stability, but boosting tries to reduce bias "also", therefore it can make overfitting worst!

Bagging vs Boosting (Training)

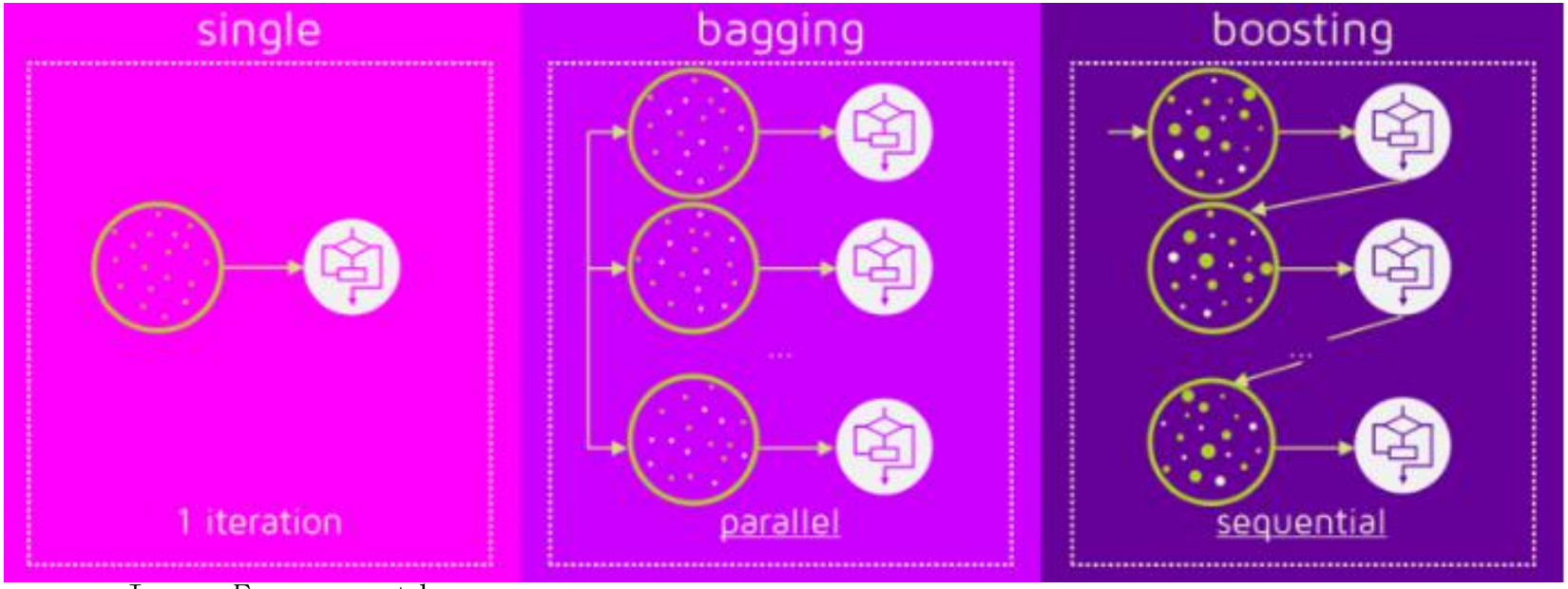


Image From quantdare.com

Bagging vs Boosting (Classification/estimation)

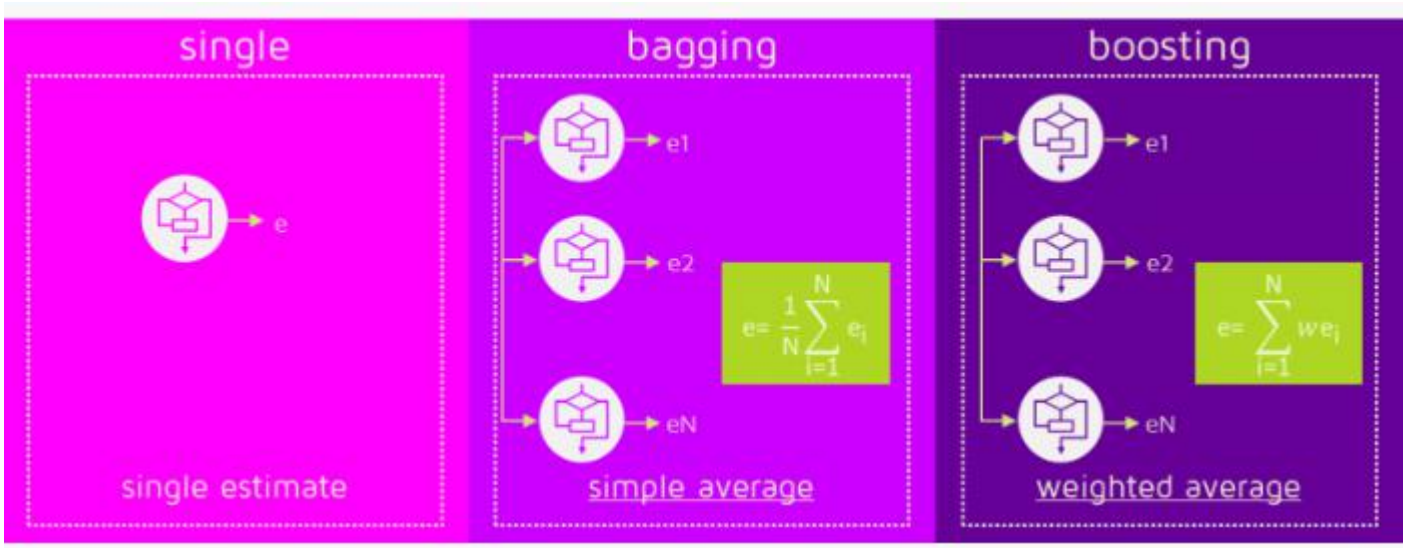


Image From quantdare.com

Bagging vs Boosting (Using/evaluating it)

Image From quantdare.com

Random Forest

- Decision trees are known to be prone to overfitting, which increases the variance of the forecasts.3 In order to address this concern, the random forest (RF) method was designed to produce ensemble forecasts with lower variance

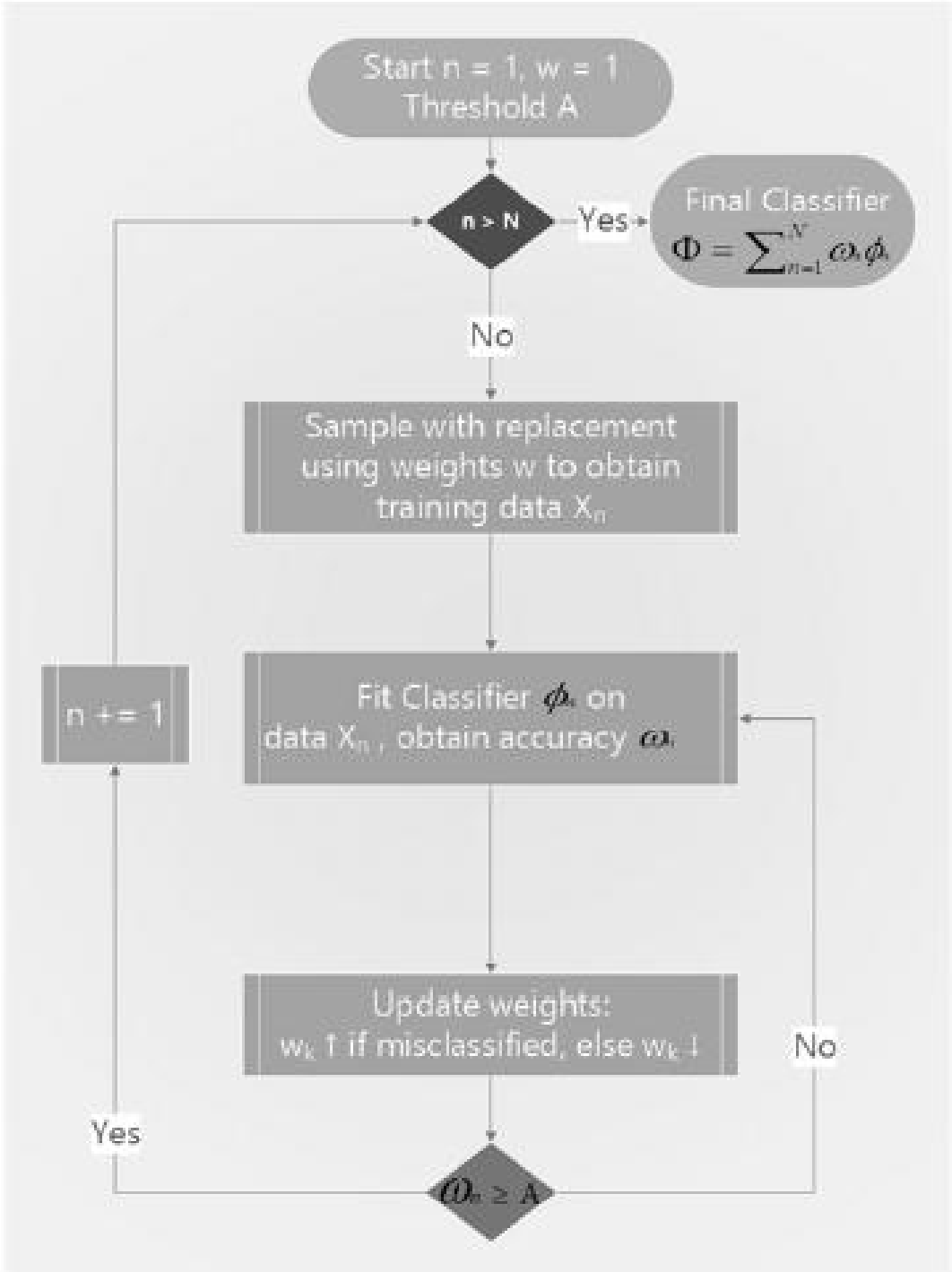
- RF shares some similarities with bagging, in the sense of training independently individual estimators over bootstrapped subsets of the data.

- The key difference with bagging is that random forests incorporate a second level of randomness: When optimizing each node split, only a random subsample (without replacement) of the attributes will be evaluated, with the purpose of further decorrelating the estimators

Boosting: From weak to stronger

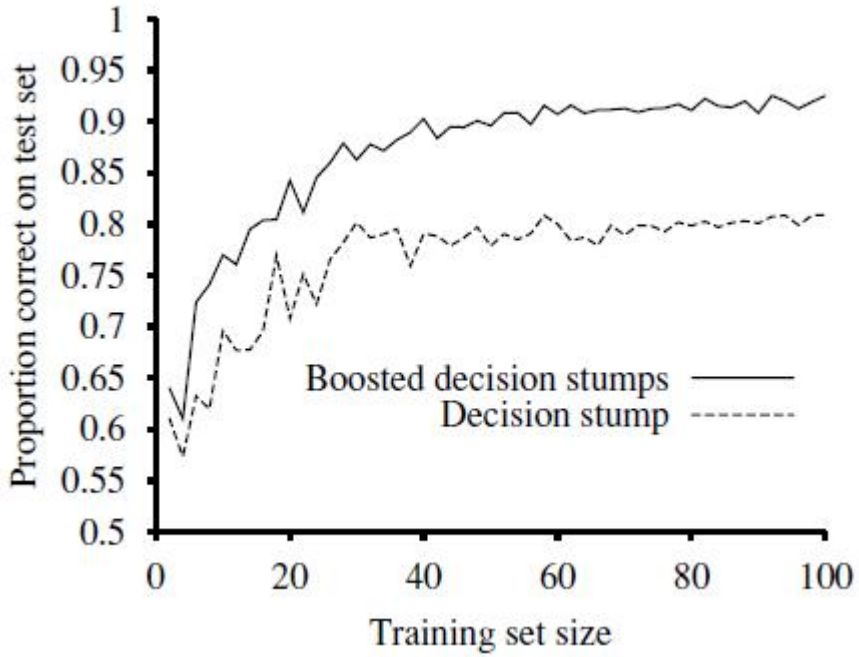
- Schapire [1990] demonstrated that it was possible to combine weak estimators in order to achieve one with high accuracy, using the procedure we today call boosting. In general terms, it works as follows:
  1. Generate one training set by random sampling with replacement, according to some sample weights (initialized with uniform weights)
  2. Fit one estimator using that training set
  3. if the single estimator achieves an accuracy greater than the acceptance threshold (e.g., 50% in a binary classifier, so that it performs better than chance), the estimator is kept, otherwise it is discarded.
  4. Give more weight to misclassified observations, and less weight to correctly classified observations.
  5. Repeat the previous steps until N estimators are produced.
  6. The ensemble forecast is the *weighted* average of the individual forecasts from the N models, where the weights are the model's accuracies

AdaBoost Decision Flow



Boosting

- Let us see how well boosting does on the restaurant data. The basic models are decision trees with just one test, at the root (called **Decision stump**)



Bagging vs Booting

- Some key aspects make boosting quite different from bagging
  - Individual classifiers are fit sequentially.
  - Poor-performing classifiers are dismissed.
  - Observations are weighted differently in each iteration.
  - The ensemble forecast is a weighted average of the individual learners
- Bagging's main advantage is that it reduces forecasts' variance, hence helping address **overfitting**
- Boosting reduces both **variance and bias** in forecasts
- However, correcting bias comes at the cost of greater risk of overfitting