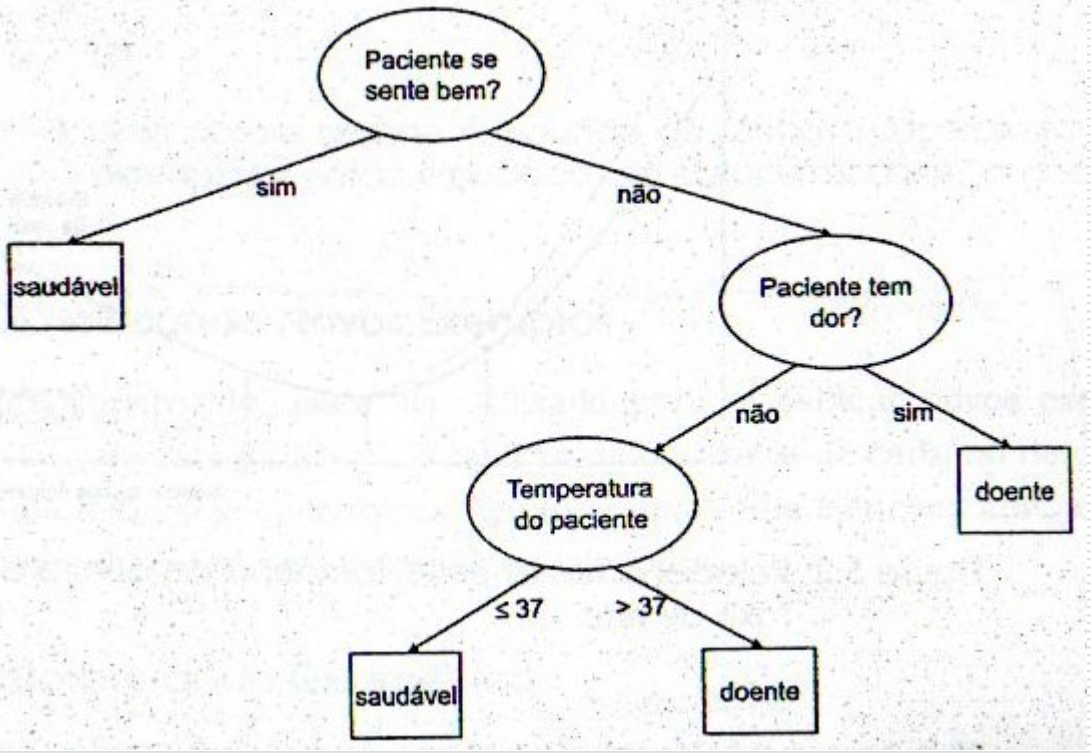
Árvores de Decisão

Prof. Flávio Varejão
Departamento de Informática
Programa de Pós-Graduação em Informática
Universidade Federal do Espírito Santo

Indução de Árvores de Decisão

- Algoritmos que induzem Árvores de Decisão pertencem à família de algoritmos *Top Down Induction of Decision Trees* – TDIDT
- Uma árvore de decisão(ou AD) é uma estrutura de dados definida recursivamente como
  - Um nó folha que corresponde a uma classe
  - Um nó de decisão que contém um teste sobre algum algoritmo
    - Para cada resultado do teste existe uma aresta para uma subárvore

Árvores de Decisão



Árvores de Decisão

- A árvore pode ser representada como um conjunto de regras. Cada regra tem seu início na raiz da árvore e caminha até uma de suas folhas

se paciente se sente bem = sim **então**
    classe = saudável
senão
    se paciente tem dor = não **então**
      se temperatura do paciente ≤ 37 **então**
          classe = saudável
      senão {temperatura do paciente > 37}
          classe = doente
      fim se
    senão {paciente tem dor = sim}
      classe = doente
    fim se
fim se

Árvores de Decisão

- Outra forma de representação:

se paciente se sente bem = sim **então**
  classe = saudável
fim se
se paciente se sente bem = não **and** paciente tem dor = não
and temperatura do paciente ≤ 37 **então**
  classe = saudável
fim se
se paciente se sente bem = não **and** paciente tem dor = não
and temperatura do paciente > 37 **então**
  classe = doente
fim se
se paciente se sente bem = não **and** paciente tem dor = sim **então**
  classe = doente
fim se

Construindo uma Árvore de Decisão

- **Passo 1:** *T* contém um ou mais exemplos, todos pertencentes à mesma classe *Cj*
  - Nesse caso, a árvore de decisão para *T* é um nó folha identificando a classe *Cj*
- **Passo 2:** *T* não contém exemplos
  - Novamente, nessa situação, a árvore é uma folha, mas a classe associada à folha deve ser determinada a partir da informação além de *T*
    - Por exemplo, a classe mais freqüente para o nó pai desse nó pode ser utilizada

Construindo uma Árvore de Decisão

- **Passo 3:** *T* contém exemplos que pertencem a várias classes.
  - Refina-se *T* em suconjuntos de exemplos que são (ou aparentam ser) conjuntos de exemplos pertencentes a uma única classe
  - Um teste é escolhido baseado em um único atributo que possui resultados mutuamente exclusivos
    - Os possíveis resultados do teste são denotados por {*O1, O2, ...,Or*}
    - *T* é então particionado em subconjuntos *T1, T2, ...,Tr,* nos quais cada *Ti* contém todos os exemplos em *T* que possui como resultado daquele teste o valor *Oi*
    - A AD para *T* consiste em um nó interno identificado pelo teste escolhido e uma aresta para cada um dos resultados possíveis

Construindo uma Árvore de Decisão

- **Passo 4:** Os passos 1, 2 e 3 são aplicados recursivamente para cada subconjunto de exemplos de treinamento de maneira que, em cada nó, as arestas levam para as subárvores construídas a partir do subconjunto de exemplos *Ti*
- **Passo 5:** Após a construção da AD, a poda pode ser realizada para melhorar a capacidade de generalização da AD

Escolha do Melhor Atributo para Particionar

- Critérios utilizados para escolher o atributo que particiona o conjunto de exemplos em cada interação
  - Aleatório
  - Menos valores
  - Mais valores
  - Ganho máximo

Indutores de Árvores de Decisão

- Método mais conhecido C5 de Quinlan
- Usa Ganho Máximo
  - Baseado no ganho máximo de informação ou redução máxima de entropia
  - Entropia é medida estatística que determina o grau de desorganização dos dados
  - Minimiza a informação necessária para classificar os exemplos
    - Minimiza o número esperado de testes para classificar um caso

Escolha do Melhor Atributo para Particionar

- Etapas do Processo de Ganho Máximo
  1. Obter a entropia do conjunto de exemplos inicial
     - $I(S) = - \Sigma\ p_i \log_2 (p_i)$

       $1 \leq i \leq m$

       m é o total de classes do conjunto S

       $p_i = s_i / s$

Escolha do Melhor Atributo para Particionar

- Etapas do Processo de Ganho Máximo
  1. Calcular a entropia do particionamento do conjunto S para cada atributo A a ser medido
     - $E(A) = \Sigma\ p_j\ I(S_j)$

       $1 \leq j \leq v$

       v é o total de valores distintos de A

       $p_j = s_j / s$

       $I(S_j) = - \Sigma\ p_{ij}\ \log_2 (p_{ij})$

       $p_{ij} = s_{ij} / s_j$

Exemplo

Exemplo Nº	Aparência	Temperatura	Umidade	Ventando	Viajar ?
T1	sol	25	72	sim	vá
T2	sol	28	91	sim	não_vá
T3	sol	22	70	não	vá
T4	sol	23	95	não	não_vá
T5	sol	30	85	não	não_vá

Escolha do Melhor Atributo para Particionar

$I(S) = - \Sigma\ p_i \log_2 (p_i)$

$p_{va} = 2/5 = 0.4 :: p_{nao\text{-}va} = 3/5 = 0.6$

$I(S) = - (0.4 \log_2 (0.4) + 0.6 \log_2 (0.6))$

$I(S) = - (0.4 \times -1.322 + 0.6 \times -0.737)$

$I(S) = 0.5288 + 0.4422 = 0.971$

Exemplo

- A = ventando

  $E(A) = \Sigma\ p_j\, I(S_j)$

  $p_{sim} = 2/5 = 0.4 :: p_{nao} = 3/5 = 0.6$

  $p_{va/sim} = 1/2 = 0.5 :: p_{nao\text{-}va/sim} = 1/2 = 0.5$

  $I(S=sim) = - (0.5 \log_2 (0.5) + 0.5 \log_2 (0.5))$

  $I(S=sim) = - \log_2 (0.5) = 1.0$

  $p_{va/nao} = 1/3 = 0.333 :: p_{nao\text{-}va/nao} = 2/3 = 0.666$

  $I(S=nao) = - (0.333 \log_2 (0.333) + 0.666 \log_2 (0.666))$

  $I(S=nao) = 0.528 + 0.391 = 0.919$

  $E(A) = 0.4 \times 1.0 + 0.6 \times 0.919 = 0.9514$

Exemplo

$G(A) = I(S) - E(A)$

$G(A) = 0.971 - 0.9514 = 0.0196$

$K(A) = - \Sigma\ p_j \log_2 (p_j)$

$K(A) = - (0.4 \log_2 (0.4) + 0.6 \log_2 (0.6))$

$K(A) = 0.971$

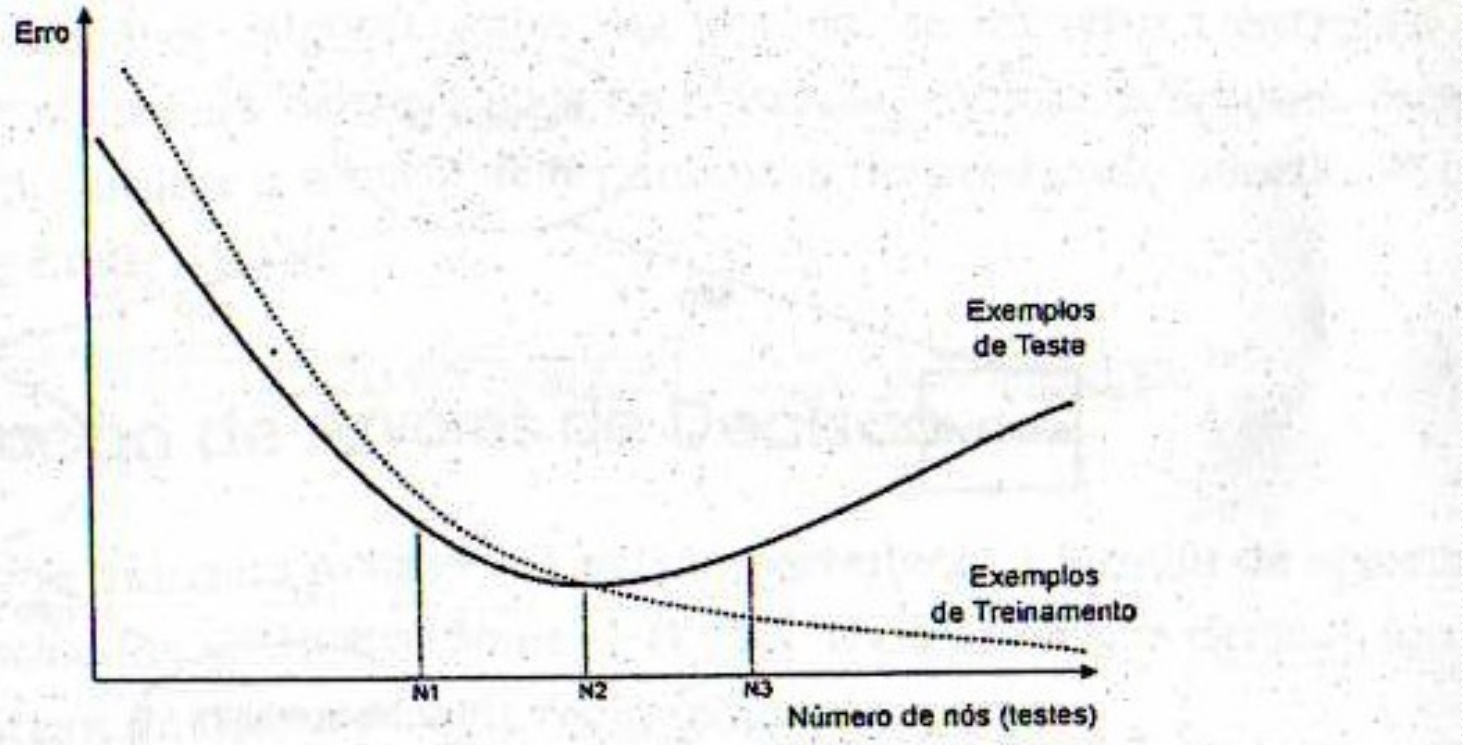
$RG(A) = G(A)/K(A) = 0.0196/0.971$

$RG(A) = 0.0202$

Poda

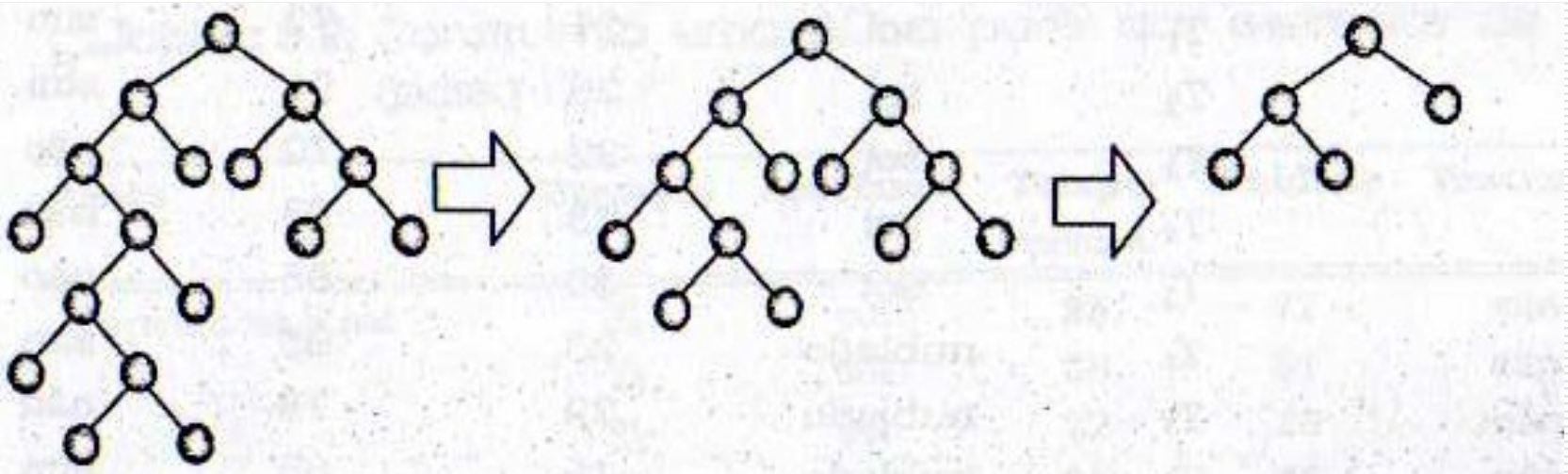
- É possível que a AD induzida seja muito específica para o conjunto de treinamento
- Diz-se que o classificador superajustou os dados de treinamento, ou seja, ocorreu um *overfitting*
  - Foram adicionadas na árvore nós que melhoram seu desempenho nos dados de treinamento, mas que pioram seu desempenho em um conjunto de teste

Poda



Relacionamento entre tamanho da árvore de decisão e a taxa de erro

Poda



Uma árvore grande é induzida de forma a superajustar os exemplos e então é podada até obter uma árvore menor

Poda

- Existem várias estratégias
  - Substituição de subárvore por nó folha
  - Elevação de subárvore
- Estratégia
  - Separar subconjunto de treinamento independente para poda
  - Estimar erro da árvore com ou sem poda para o subconjunto de poda
    - Podar se erro diminui

Classificando Novos Exemplos

- Uma AD pode classificar novos exemplos iniciando-se da raiz e caminhando através de cada nó de decisão até que uma folha seja encontrada, e a classe do novo exemplo é dada pela classe daquela folha

Interpretação Geométrica

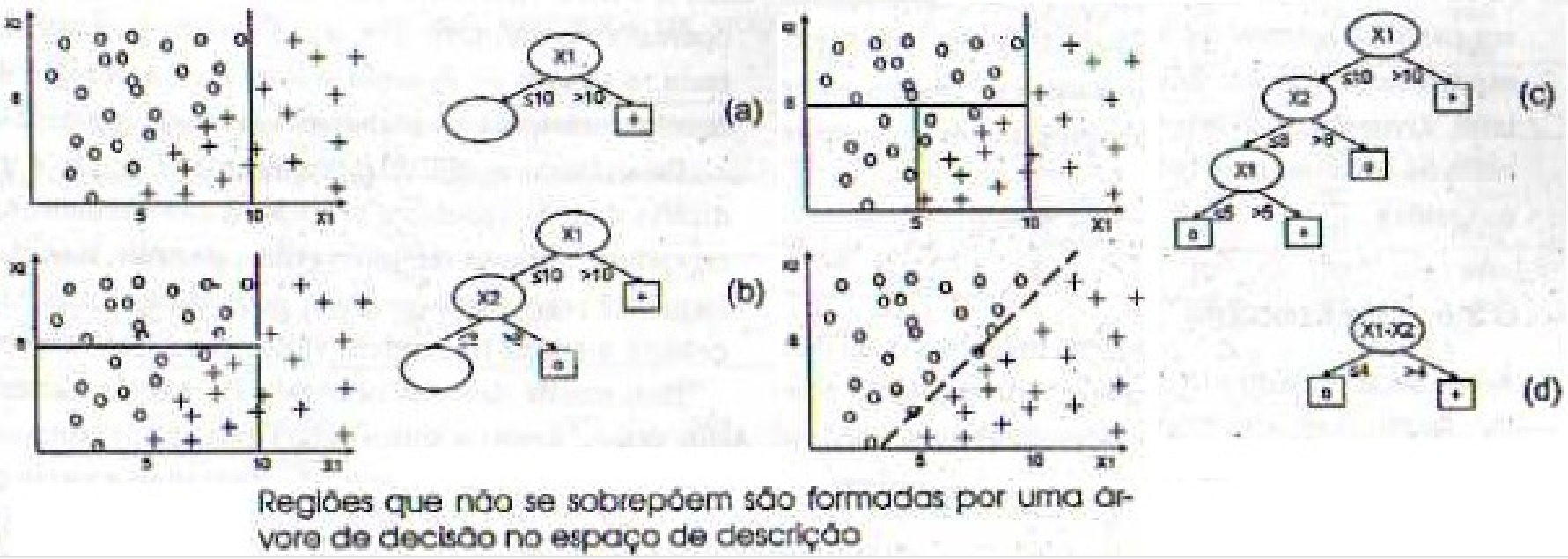
- Exemplos = vetor de *m* atributos, localizado no espaço *m*-dimensional dos atributos.
- AD corresponde a uma divisão deste espaço em regiões (classes)
- Atributo-Valor
  - Ilustrando dois atributos reais e duas classes (o e +).
  - Teste: Xi op Valor, onde

    Xi ∈ {X1, X2}

    op é um operador no conjunto {≤, >}

    Valor é um valor válido para os atributos X1 ou X2

Interpretação Geométrica



Regiões que não se sobrepõem são formadas por uma árvore de decisão no espaço de descrição

Vantagens do Método de AD

- Rápido para aprendizado de conceitos
- Simples de implementar
- Permite transformar seus resultados em forma de regras interpretáveis
- Pode tratar exemplos com ruído
- É uma tecnologia madura utilizada em vários produtos comerciais

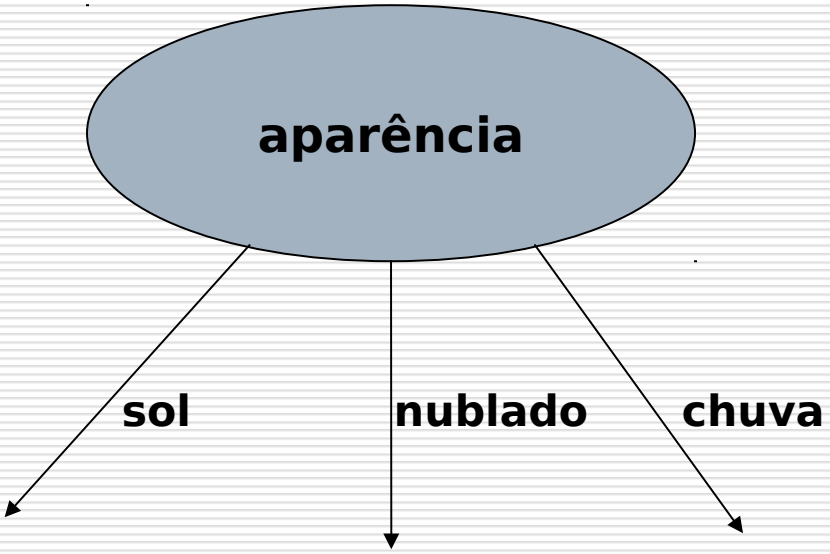
Desvantagens do Método de AD

- Árvores muito grandes são geralmente difíceis de serem lidas
- Árvores univariadas, nas quais apenas um atributo é utilizado em cada nó interno de teste, são limitadas a partições paralelas aos eixos no espaço de descrição, limitando o conceito que pode ser aprendido
- Árvores multivariadas podem utilizar mais de um atributo em cada nó interno, mas requerem maiores recursos computacionais para serem induzidas

Exemplo de Construção de uma AD

Exemplo Nº	Aparência	Temperatura	Umidade	Ventando	Viajar ?
T1	sol	25	72	sim	vá
T2	sol	28	91	sim	não_vá
T3	sol	22	70	não	vá
T4	sol	23	95	não	não_vá
T5	sol	30	85	não	não_vá
T6	nublado	23	90	sim	vá
T7	nublado	29	78	não	vá
T8	nublado	19	65	sim	não_vá
T9	nublado	26	75	não	vá
T10	nublado	20	87	sim	vá
T11	chuva	22	95	não	vá
T12	chuva	19	70	sim	não_vá
T13	chuva	23	80	sim	não_vá
T14	chuva	25	81	não	vá
T15	chuva	21	80	não	vá

Exemplo de Construção de uma AD



Passo1: Construindo uma AD a partir dos exemplos de viagem

Exemplo de Construção de uma AD

Teste	Exemplo	Aparência	Tempe-ratura	Umidade	Ventando	Viajar?
if aparência = sol	T_1	sol	25	72	sim	vá
	T_2	sol	28	91	sim	não_vá
	T_3	sol	22	70	não	vá
	T_4	sol	23	95	não	não_vá
	T_5	sol	30	85	não	não_vá
if aparência = nublado	T_6	nublado	23	90	sim	vá
	T_7	nublado	29	78	não	vá
	T_8	nublado	19	65	sim	não_vá
	T_9	nublado	26	75	não	vá
	T_{10}	nublado	20	87	sim	vá
if aparência = chuva	T_{11}	chuva	22	95	não	vá
	T_{12}	chuva	19	70	sim	não_vá
	T_{13}	chuva	23	80	sim	não_vá
	T_{14}	chuva	25	81	não	vá
	T_{15}	chuva	21	80	não	vá

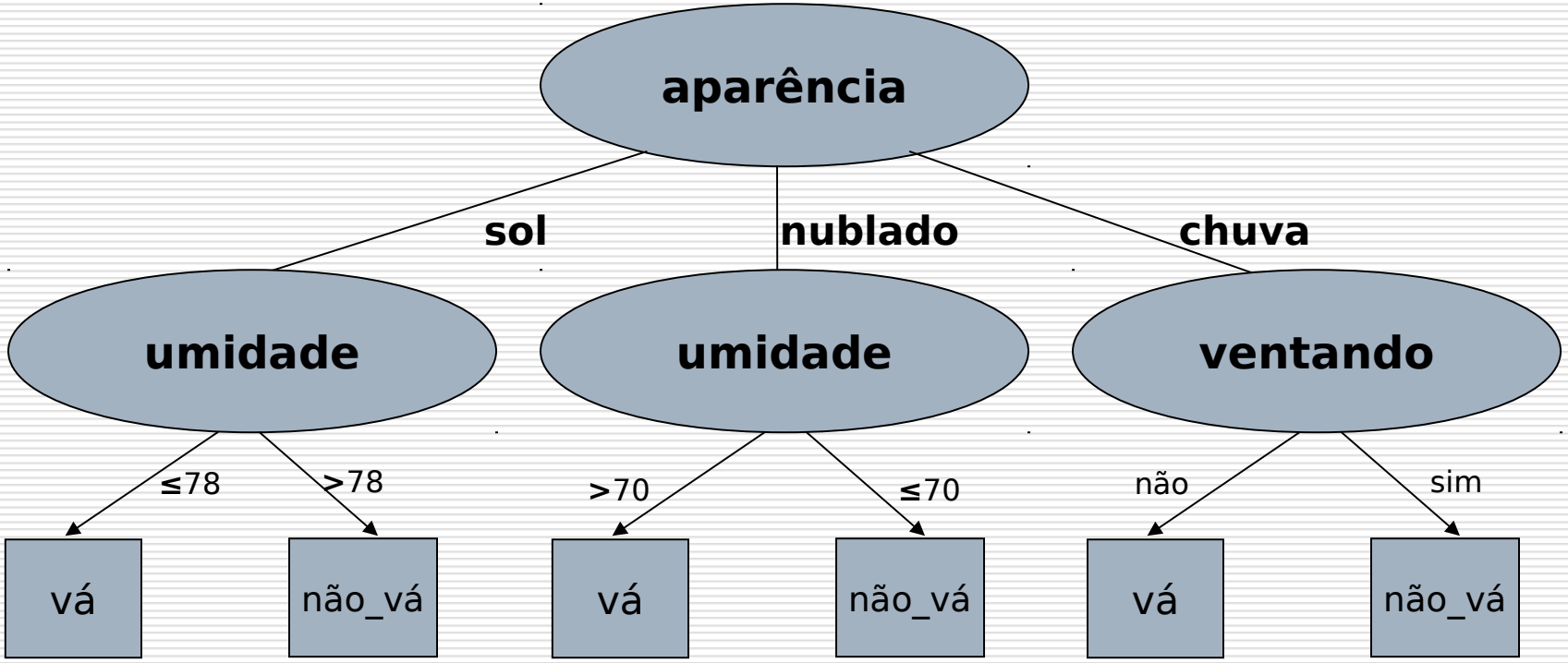
Passo 1: Construindo uma AD a partir dos exemplos de viagem

Exemplo de Construção de uma AD

- Cada subconjunto ainda contém exemplos pertencentes a várias classes, assim é necessário escolher um outro teste baseado em um único atributo. Foi selecionado o atributo **umidade** para as subárvores “sol” e “nublado” e **ventando** para a subárvore “chuva”
- Após a construção da AD completa, considera-se a seguinte subárvore:

se aparência = nublado **então**
    se umidade > 70 **então**
        classe = vá (exemplos cobertos: *T6, T7, T9, T10*}
    senão {umidade ≤ 70}
        classe = não_vá {Exemplo coberto: *T8*}
    fim se
fim se

Exemplo de Construção de uma AD



Passo 2: Construindo uma AD a partir dos exemplos de viagem

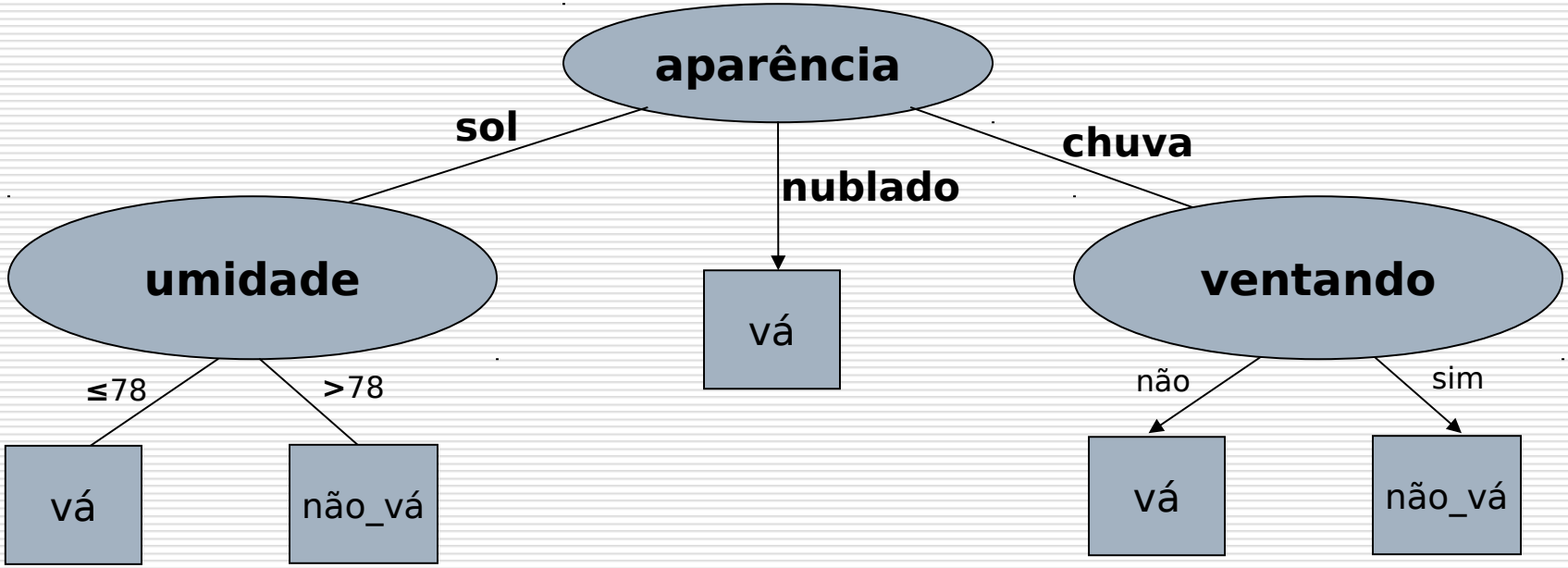
Exemplo de Construção de uma AD

Teste	Exemplo	Aparência	Tempe-ratura	Umidade	Ventando	Viajar?
if aparência = sol	T_1	sol	25	72	sim	vá
e umidade ≤ 78	T_2	sol	22	70	não	vá
if aparência = sol	T_2	sol	28	91	sim	não_vá
e umidade > 78	T_4	sol	23	95	não	não_vá
	T_5	sol	30	85	não	não_vá
if aparência = nublado	T_6	nublado	23	90	sim	vá
e umidade > 70	T_7	nublado	29	78	não	vá
	T_9	nublado	26	75	não	vá
	T_{10}	nublado	20	87	sim	vá
if aparência = nublado						
e umidade ≤ 70	T_8	nublado	19	65	sim	não_vá
if aparência = chuva	T_{11}	chuva	22	95	não	vá
e ventando = não	T_{14}	chuva	25	81	não	vá
	T_{15}	chuva	21	80	não	vá
if aparência = chuva	T_{12}	chuva	19	70	sim	não_vá
e ventando = sim	T_{13}	chuva	23	80	sim	não_vá

Passo 2: Construindo uma AD a partir dos exemplos de viagem

Exemplo de Construção de uma AD

- Apenas um exemplo (*T8*) satisfaz o teste “umidade ≤ 70”; todos os outros exemplos para a subárvore **nublado** pertencem à “classe = vá”. Isso pode indicar um *overfitting* dos dados e o indutor pode podar essa subárvore



Passo 3: Construindo uma AD a partir dos exemplos de viagem

Exemplo de Construção de uma AD

Teste	Exemplo	Aparência	Tempe-ratura	Umidade	Ventando	Viajar?
if aparência = sol	T_1	sol	25	72	sim	vá
e umidade ≤ 78	T_3	sol	22	70	não	vá
if aparência = sol	T_2	sol	28	91	sim	não_vá
e umidade > 78	T_4	sol	23	95	não	não_vá
	T_5	sol	30	85	não	não_vá
if aparência = nublado	T_6	nublado	23	90	sim	vá
	T_7	nublado	29	78	não	vá
	T_8	nublado	19	65	sim	não_vá
	T_9	nublado	26	75	não	vá
	T_{10}	nublado	20	87	sim	vá
if aparência = chuva	T_{11}	chuva	22	95	não	vá
e ventando = não	T_{14}	chuva	25	81	não	vá
	T_{15}	chuva	21	80	não	vá
if aparência = chuva	T_{12}	chuva	19	70	sim	não_vá
e ventando = sim	T_{13}	chuva	23	80	sim	não_vá

Passo 3: Podando a AD a partir dos exemplos de viagem

Poda

- A poda da AD pode, em geral, melhorar o desempenho para exemplos não vistos
  - No exemplo anterior, a poda descarta alguma informação (exemplo *T8*)
- Quando o aprendizado ocorre em exemplos contendo ruído, um grau adequado de poda pode melhorar o desempenho em exemplos não vistos
  - A poda, em geral, elimina erros provenientes de ruídos em vez de descartar informação relevante

Algoritmos de Árvores de Decisão

- ID3 - Quinlan (1986)
- C4.5 - Quinlan (1993)
  - J48 de Weka é uma implementação

Algoritmo ID3

- Proposto por Quinlan (1986), é aplicável para conjuntos de objetos com atributos discretos ou discretizados
- Algoritmo possui uma implementação simples e um bom desempenho, o que levou a ser este um dos mais populares
- A inovação alcançada por Quinlan reside no critério de seleção de atributos para o particionamento, que é o Critério do Ganho de Informação

Características do ID3

- Só atributos nominais
- Árvore ajustada ao conjunto de treinamento
  - Sem poda
  - Acerta 100%
- Não realiza *backtracking* na busca pela melhor árvore
  - Uma vez escolhido um atributo de teste em um nível particular da árvore, ele nunca retrocede a este nível para reconsiderar esta escolha
  - Há o risco da solução encontrada corresponder a uma solução ótima local
- Bias
  - Ganho de Informação (Árvores amplas)

Algoritmo C4.5

- Uma extensão do ID3
  - Tratamento de instâncias com atributos contínuos
  - Uso de razão de ganho
  - Poda de árvore
  - Trata valores ausentes

Algoritmo C4.5 Atributos Contínuos

- Tratamento de instâncias com atributos contínuos
  - Substituição de valores contínuos por intervalos
  - Usa valores dos exemplos para calcular valores limites
    - Valores limites candidatos são a média de dois valores adjacentes
- Exemplo de discretização em valores booleanos
  - Escolher um valor limite c e determinar que

    A < c -> verdadeiro e A ≥ c -> falso, ou vice-versa
  - Valor c deve ser tal que produza o maior ganho de informação no caso de seleção deste atributo

Algoritmo C4.5 Atributos Contínuos

- Em um exemplo real, ordenando os valores contínuos do atributo no conjunto de treino permitirá a identificação de intervalos cujos valores estão relacionados com um mesmo valor do atributo de classificação

Temperature	40	48	60	72	80	90
PlayTennis	No	No	Yes	Yes	Yes	No

Valores Limite Candidatos : (48+60)/2 and (80+90)/2

Algoritmo C4.5 Atributos Contínuos

- Uma vez decidido que o ponto de corte (valor limite) implica na maior razão de ganho de informação para uma discretização do atributo A, este atributo A discretizado terá ainda que competir com os demais para decidir qual implica na maior razão de ganho para o nó a ser particionado

Algoritmo C4.5 Atributos Contínuos

- Uma vez decidido que o ponto de corte (valor limite) implica na maior razão de ganho de informação para uma discretização do atributo A, este atributo A discretizado terá ainda que competir com os demais para decidir qual implica na maior razão de ganho para o nó a ser particionado

Algoritmo C4.5 Valores Ausentes

- Duas situações
  - Valor ausente é significativo
    - Valor só é obtido sob condições especiais
      - Exame condicionado a resultado de outro exame
    - Trata como valor possível (absent)
  - Valor ausente obtido por falha na coleta
    - Efeito no treinamento e no teste

Algoritmo C4.5 Valores Ausentes

- Treinamento
  - Desconsidera no cálculo do ganho
  - Se atributo não é escolhido, nada há a fazer
  - Senão, exemplo deve ser usado em cada ramo contando proporcionalmente
    - Surgem os numeros fracionários no fator de confiança de cada folha usado para poda e para classificação
- Teste
  - Deve-se seguir cada ramo possível escolhendo o de melhor fator de confiança

Regras

Estratégias de Geração de Regras

- A partir de árvore de decisão
  - A lista de regras é obtida para cada ramo da árvore de decisão
  - Há uma generalização das regras desconsiderando condições supérfluas, sem afetar a precisão e mantendo as regras mais relevantes

Estratégias de Geração de Regras

- A partir dos exemplos
  - Assim que uma regra é encontrada, os exemplos de treinamento são removidos e o processo se repete
  - As regras resultantes podem ser refinadas por meio da remoção de condições irrelevantes de acordo com algum teste estatístico

Método de Regras

- É um processo iterativo, cada iteração procura por um complexo que cobre um grande número de exemplos de uma mesma classe *Ci* e poucos de outras classes Cj, j≠i
- Ao encontrar um complexo, os exemplos cobertos que pertencem à classe Ci sendo aprendida (assim como, eventualmente, alguns exemplos de outras classes Cj, j≠i também cobertos pelo mesmo complexo) são removidos do conjunto de treinamento
- A regra “if <complexo> then class = Ci” é adicionada no final da lista de regras
- Esse processo se repete até que nenhum complexo possa ser encontrado

Classificando Novos Exemplos

- O classificador de regras tenta cada regra em ordem até encontrar uma, cujas condições sejam satisfeitas pelo novo exemplo
  - A classe do novo exemplo é associada à classe predita pela regra, o que torna a ordem de regras fundamental
- Uma regra isolada, exceto a primeira, não tem validade por si própria
- Se nenhuma regra é satisfeita, existe a regra *default* que, em geral, atribui ao novo exemplo a classe mais comum no conjunto de treinamento

Exemplo

	Regra	CC	CI
R1	if umidade < 83 then classe=vá	T1,T3,T7,T8,T9,T14,T15	T12,T13
R2	else if temperatura ≥ 23 then classe=vá	T2,T4,T5	T6
R3	else if aparência = chuva then classe=vá	T11	
R4	else classe=não_vá	T10	

Classificadores de Regras

- ZeroR
  - Sempre classifica na classe majoritária
  - Boa linha de base para comparação de classificadores
    - Desempenho sempre igual (bases balanceadas) ou superior (bases desbalanceadas) ao classificador aleatório
    - Acerto
      - Balanceadas: 1/c
      - Desbalanceadas: m/n

(c numero de classes, m exemplos da majoritaria, n total de exemplos)

Classificadores de Regras

- OneR
  - Sempre classifica segundo o atributo mais preditivo

Algorítmo

For each predictor,

For each value of that predictor, make a rule as follows;

Count how often each value of target (class) appears

Find the most frequent class

Make the rule assign that class to this value of the predictor

Calculate the total error of the rules of each predictor

Choose the predictor with the smallest total error.

Árvores x Regras

- Regras se destacam pela sua compreensibilidade e pouco espaço requerido de armazenamento
- O método de Regras é
  - Mais lento que o de Árvore de Decisão
  - Há mais parâmetros a serem ajustados