DEBORAH RIBEIRO CARVALHO

DATA MINING ATRAVÉS DE INDUÇÃO DE REGRAS E ALGORITMOS GENÉTICOS

Dissertação apresentada como requisito para obtenção do grau de Mestre no Programa de Pós-Graduação em Informática Aplicada da Pontifícia Universidade Católica do Paraná - PUC-PR.

Orientador: Prof. Dr. Bráulio Coelho Ávila

CURITIBA

ABRIL 1999

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
Pró-Reitoria de Pós-Graduação, Pesquisa e Extensão

ATA DA SESSÃO PÚBLICA DE EXAME DE DISSERTAÇÃO DO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA APLICADA DA PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ.

Exame de dissertação nº 003

Aos 14 dias do mês de abril de 1999, realizou-se a sessão pública de defesa de dissertação "**DATA MINING ATRAVÉS DE INDUÇÃO DE REGRAS E ALGORITMOS GENÉTICOS**", apresentada por Deborah R. Carvalho Guimarães, ano de ingresso 1996, para obtenção do título de Mestre em Ciências. A Banca Examinadora foi composta pelos seguintes professores:

MEMBROS DA BANCA	ASSINATURA
Presidente:. **Prof. Dr. Bráulio Coelho Ávila (PUC-PR)**	
Prof. Dr. Júlio Cesar Nievola (PUC-PR)	
Prof. Dr. Maurício Figueiredo (UEM)	
Prof. Dr. Nelson Ebecken (COPPE / UFRJ)	

De acordo com as normas regimentais a Banca Examinadora deliberou sobre os conceitos a serem atribuídos e que foram os seguintes:

MEMBROS DA BANCA	CONCEITOS
Presidente:. **Prof. Dr. Bráulio Coelho Ávila (PUC-PR)**	A
Prof. Dr. Júlio Cesar Nievola (PUC-PR)	A
Prof. Dr. Maurício Figueiredo (UEM)	A
Prof. Dr. Nelson Ebecken (COPPE / UFRJ)	A
Conceito Final	A

Observações da Banca Examinadora

NÃO HÁ

Profº Júlio Cesar Nievola
Coordenador do Programa de Pós-Graduação em Informática Aplicada–PUC-PR

Rua Imaculada Conceição, 1155 – Prado Velho – CEP 80215-901 – Telefone: +55 41 330-1669 / + 55 41 330-1654
Fax: + 55 41 330- 1669 – Curitiba – Paraná – Brasil - E-mail:ppgia@rla01.pucpr.br

AGRADECIMENTOS

Agradeço ao Prof. Dr. Bráulio Coelho Ávila pelo constante apoio durante todo o período que estivemos em curso. Ao Prof. Dr. Alex Freitas pelo incansável acompanhamento desde junho de 1998, quando, mesmo sem ainda fazer parte do corpo docente da PUC-PR, dispôs de tempo para discussões, orientações e esclarecimentos de dúvidas sobre este trabalho.

Agradeço ao pessoal do LASIN – Laboratório de Sistemas Inteligentes da PUC-PR, principalmente na pessoa do Alexandre Thieme, pelo apoio no desenvolvimento e implementação de partes deste trabalho.

Agradeço aos meus colegas de mestrado por estarem sempre disponíveis para troca de idéias sobre algumas questões conceituais ou mesmo de apresentação.

E, finalmente, agradeço às pessoas da minha família, ao meu marido e aos meus 4 filhos, pois hoje estou convencida de que somente as pessoas que dispõem de tal apoio concluem uma atividade com um nível de dedicação tão alto.

SUMÁRIO

RESUMO vi
ABSTRACT vii
LISTA DE TABELAS viii
LISTA DE QUADROS x
LISTA DE GRÁFICOS xi
LISTA DE FIGURAS xii
1 INTRODUÇÃO 1
1.1 TRABALHOS RELACIONADOS 3
1.2 ORGANIZAÇÃO DO TEXTO 7
2 AQUISIÇÃO DE CONHECIMENTO 8
2.1 CONSIDERAÇÕES INICIAIS 8
2.2 O PROCESSO DE AQUISIÇÃO DE CONHECIMENTO 8
2.3 ABORDAGENS DA AQUISIÇÃO DE CONHECIMENTO 11
2.3.1 Conceito E Estratégias De Aprendizado 12
2.3.2 Fases Do Aprendizado De Máquina Indutivo 17
2.4 PARADIGMAS DA AQUISIÇÃO AUTOMÁTICA DE CONHECIMENTO 18
2.4.1 Indução De Regras 18
2.4.2 Redes Neurais 19
2.4.3 Aprendizado Baseado Em Casos 20
2.4.4 Algoritmos Genéticos 20
2.5 CONSIDERAÇÕES FINAIS 21
3 O PROCESSO DE KDD E DATA MINING 23
3.1 CONSIDERAÇÕES INICIAIS 23
3.2 O PROCESSO DE KDD 23
3.2.1 Etapas Do Processo De KDD 26
3.3 DATA MINING 29
3.3.1 Objetivos De Data Mining 30
3.4 CONSIDERAÇÕES FINAIS 30
4 ESTRATÉGIAS DE ALGORITMOS PARA DATA MINING 31
4.1 CONSIDERAÇÕES INICIAIS 31
4.2 CLASSIFICAÇÃO 31
4.2.1 Modelos De Classificação 32

4.3 ALGORITMOS PARA CLASSIFICAÇÃO 35
4.3.1 ID3 35
4.3.2 AQ 45
4.3.3 CN2 49
4.3.4 C4.5 56
4.3.5 C5.0 66
4.3.6 Conclusão 72
5 METODOLOGIA ADOTADA NO PROJETO EXPERIMENTAL 73
5.1 CONSIDERAÇÕES INICIAIS 73
5.2 BASES DE DADOS 73
5.2.1 BDPL - Bases De Dados Padrão Da Literatura 73
5.2.2 BDR - Base De Dados Do Censo Demográfico Relativa Aos Municípios Do Paraná 74
5.3 Avaliação De Desempenho Dos Algoritmos 89
5.3.1 Precisão Na Classificação 90
5.3.2 Simplicidade Nas Regras 92
5.3.3 Eficiência Na Geração Das Regras 92
5.3.4 Grau De Interesse Das Regras 92
6 RESULTADOS 93
6.1 BASES DE DADOS PADRÃO DA LITERATURA. 93
6.2 BDR 99
6.2.1 Experimentos Com O Valor Do Erro Médio Como Função Objetivo 100
6.2.2 Experimentos Com O Valor Da Taxa De Acerto Sensível Ao Desbalanceamento De Classes Como Função Objetivo 104
6.2.3 Comparação Dos Resultados Obtidos Com Relação Ao Critério De Repetição De Exemplos 107
7 CONCLUSÃO 111
7.1 COMPARAÇÃO DOS ALGORITMOS CN2, C4.5 E C5.0 111
7.2 UMA NOVA SOLUÇÃO PARA O PROBLEMA DE DESBALANCEAMENTO DE CLASSES EM UMA BASEDE DADOS DO MUNDO REAL 112
7.3 TRABALHOS FUTUROS 113
APÊNDICE B - DEFINIÇÕES 122
REFERÊNCIAS BIBLIOGRÁFICAS 127

RESUMO

Este trabalho tem dois objetivos principais. O primeiro é comparar a eficácia de 3 (três) algoritmos para aquisição de conhecimento baseados no paradigma de indução de regras, a saber CN2, C4.5 e C5.0. Esses algoritmos produzem listas de decisão, regras de produção ou árvores de decisão. Justifica esse objetivo o fato de não haver disponível na literatura textos que comparem a eficácia de outros algoritmos em relação ao algoritmo C5.0. Como resultado da análise realizada, a comparação da execução dos algoritmos CN2, C4.5 e C5.0 confirma o que já foi dito: "não existe um algoritmo que possa ser definido como sendo o melhor algoritmo independentemente do domínio e da base de dados sobre a qual se esteja processando". Porém, os resultados dos experimentos realizados neste trabalho demonstram que a adoção da opção *boosting*, disponível apenas para o algoritmo C5.0, melhora, em média, as taxas de erro se comparado com os resultados obtidos através do CN2 e C4.5 em suas diferentes versões. Outro objetivo deste trabalho é propor, implementar e testar uma nova solução para o problema de desbalanceamento de classes em uma base de dados do mundo real. Essa solução é baseada no uso de um método híbrido algoritmo genético e árvore de decisão (C5.0). O algoritmo genético age como *wrapper* usando a saída do algoritmo de árvore de decisão, estado da arte, C5.0 para obtenção da função objetivo (*fitness)* das populações de indivíduos (soluções candidatas ao problemas de classes desbalanceadas). Esta solução é avaliada sobre um estudo de caso no domínio dos dados do Censo Demográfico de 1991. Neste caso, a conclusão da análise desenvolvida neste trabalho, é a de que a expansão do conjunto de treinamento, i.e. a repetição dos exemplos de treinamento, objetivando uma distribuição mais eqüitativa entre os exemplos das classes, obtida através da abordagem *wrapper*, pode ser considerada uma solução promissora para o problema de desbalanceamento de classes. Os experimentos computacionais demonstram que esta abordagem – processo híbrido envolvendo algoritmo genético e árvore de decisão (C5.0) – obteve resultados que são significativamente melhores que os resultados obtidos pelo C5.0 sobre a base original, antes do processo de expansão.

RESUMO

Este trabalho tem dois objetivos principais. O primeiro é comparar a eficácia de 3 (três) algoritmos para aquisição de conhecimento baseados no paradigma de indução de regras, a saber CN2, C4.5 e C5.0. Esses algoritmos produzem listas de decisão, regras de produção ou árvores de decisão. Justifica esse objetivo o fato de não haver disponível na literatura textos que comparem a eficácia de outros algoritmos em relação ao algoritmo C5.0. Como resultado da análise realizada, a comparação da execução dos algoritmos CN2, C4.5 e C5.0 confirma o que já foi dito: "não existe um algoritmo que possa ser definido como sendo o melhor algoritmo independentemente do domínio e da base de dados sobre a qual se esteja processando". Porém, os resultados dos experimentos realizados neste trabalho demonstram que a adoção da opção *boosting*, disponível apenas para o algoritmo C5.0, melhora, em média, as taxas de erro se comparado com os resultados obtidos através do CN2 e C4.5 em suas diferentes versões. Outro objetivo deste trabalho é propor, implementar e testar uma nova solução para o problema de desbalanceamento de classes em uma base de dados do mundo real. Essa solução é baseada no uso de um método híbrido algoritmo genético e árvore de decisão (C5.0). O algoritmo genético age como *wrapper* usando a saída do algoritmo de árvore de decisão, estado da arte, C5.0 para obtenção da função objetivo (*fitness*) das populações de indivíduos (soluções candidatas ao problemas de classes desbalanceadas). Esta solução é avaliada sobre um estudo de caso no domínio dos dados do Censo Demográfico de 1991. Neste caso, a conclusão da análise desenvolvida neste trabalho, é a de que a expansão do conjunto de treinamento, i.e. a repetição dos exemplos de treinamento, objetivando uma distribuição mais eqüitativa entre os exemplos das classes, obtida através da abordagem *wrapper*, pode ser considerada uma solução promissora para o problema de desbalanceamento de classes. Os experimentos computacionais demonstram que esta abordagem – processo híbrido envolvendo algoritmo genético e árvore de decisão (C5.0) – obteve resultados que são significativamente melhores que os resultados obtidos pelo C5.0 sobre a base original, antes do processo de expansão.

ABSTRACT

This dissertation has two objectives: the first one is to evaluate the effectiveness and the efficiency of three rule induction algorithms (CN2, C4.5 and C5.0). This evaluation was performed by using the Lachenbruch (leave-one-out) method to compute the classification error rate and by measuring the processing time of those algorithms. The second objective of this dissertation is to propose a new approach for coping with the problem of unbalanced classes, where some class(es) is (are) much less frequent than the other(s). The proposed approach is a hybrid genetic algorithm / decision tree system. The genetic algorithm acts as a wrapper, using the output of a decision tree algorithm (the state-of-the-art C5.0) to compute the fitness of population individuals (candidate solutions to the problem of unbalanced classes). The proposed system was evaluated on a case study application domain about census data.

ABSTRACT

This dissertation has two objectives: the first one is to evaluate the effectiveness and the efficiency of three rule induction algorithms (CN2, C4.5 and C5.0). This evaluation was performed by using the Lachenbruch (leave-one-out) method to compute the classification error rate and by measuring the processing time of those algorithms. The second objective of this dissertation is to propose a new approach for coping with the problem of unbalanced classes, where some class(es) is (are) much less frequent than the other(s). The proposed approach is a hybrid genetic algorithm / decision tree system. The genetic algorithm acts as a wrapper, using the output of a decision tree algorithm (the state-of-the-art C5.0) to compute the fitness of population individuals (candidate solutions to the problem of unbalanced classes). The proposed system was evaluated on a case study application domain about census data.

LISTA DE TABELAS

2.1 NOMES, IDADES E OBRIGATORIEDADE DO VOTO 16

4.1 CÁLCULO DA ENTROPIA PARA DOIS CORTES DISTINTOS 38

4.2 PONDERAÇÃO DOS VALORES DA ENTROPIA PARA DOIS CORTES DISTINTOS 39

4.3 DE CONTINGÊNCIA 60

4.4 EXEMPLO DE SAÍDA PARA A EXECUÇÃO DO C5.0 COM OPÇÃO DE *BOOSTING* 69

4.5 EXEMPLO DE SAÍDA PARA A EXECUÇÃO DO C5.0 COM OPÇÃO DE *CROSS-VALIDATION* 71

4.6 EXEMPLO DE SAÍDA PARA A EXECUÇÃO DO C5.0 COM OPÇÃO DE *CROSS-VALIDATION* E *BOOSTING* 71

5.1 FREQÜÊNCIA DAS CLASSES POR CATEGORIAS E SITUAÇÃO DE DOMICÍLIO PARA OS MUNICÍPIOS PARANAENSES 80

5.2 DISTRIBUIÇÃO DOS EXEMPLOS DA BDR NOS TRÊS SUBCONJUNTOS 84

6.1 - TAXA DE ERRO OBTIDA ATRAVÉS DO MÉTODO LACHENBRUCH SOBRE OS EXPERIMENTOS COM A BDPL E BDR 93

6.2 - TEMPO DESPENDIDO COM OS EXPERIMENTOS ATRAVÉS DO MÉTODO LACHENBRUCH SOBRE A BDLP E BDR 96

6.3 NÚMERO DE CONDIÇÕES NOS ANTECEDENTES DAS REGRAS DESCOBERTAS 97

6.4 MÉDIA DE ATRIBUTOS NA PREMISSA 98

6.5 ATRIBUTO PREVISOR SELECIONADO PARA COMPOR O NÓ RAIZ 99

6.6 - RESULTADOS DAS 3 VERSÕES DO C5.0 SOBRE A BDR ORIGINAL 100

6.7 - RESULTADOS MEDIDO NAS 12 EXECUÇÕES DO AG COM O VALOR DO ERRO MÉDIO COMO FUNÇÃO OBJETIVO (DADOS DE VALIDAÇÃ0) 101

6.8 - VALOR DO ERRO MÉDIO OBTIDO ATRAVÉS DO C5.0 A PARTIR DO CONJUNTO DE TREINAMENTO EXPANDIDO PELO AG (TODOS) – MEDIDO NOS DADOS DE TESTE 103

6.9 - VALOR DO ERRO MÉDIO OBTIDO ATRAVÉS DO C5.0 A PARTIR DO CONJUNTO DE TREINAMENTO EXPANDIDO PELO AG (TRES-QUATRO) - MEDIDO NOS DADOS DE TESTE 103

6.10-RESULTADOS MEDIDO NAS 12 EXECUÇÕES DO AG COM A TAXA DE ACERTO SENSÍVEL AO DESBALANCEAMENTO DE CLASSES COMO FUNÇÃO OBJETIVO (DADOS DE VALIDAÇÃO) 105

6.11-TAXA DE ACERTO SENSÍVEL AO DESBALANCEAMENTO DE CLASSES OBTIDA ATRAVÉS DO C5.0 A PARTIR DO CONJUNTO DE TREINAMENTO EXPANDIDO PELO AG (TODOS) – MEDIDO NOS DADOS DE TESTE 106
6.12 TAXA DE ACERTO SENSÍVEL AO DESBALANCEAMENTO DE CLASSES OBTIDA ATRAVÉS DO C5.0 A PARTIR DO CONJUNTO DE TREINAMENTO EXPANDIDO PELO AG (TRES-QUATRO) – MEDIDO NOS DADOS DE TESTE 107
A.1 DADOS DE ENTRADA PARA A TAREFA DE CLASSIFICAÇÃO 116
B.1 COBERTURA DA REGRA *VERSUS* PRECISÃO 125

LISTA DE QUADROS

2.1 ENGENHARIA DO CONHECIMENTO *VERSUS* PROGRAMAÇÃO CONVENCIONAL 11
2.2 INDUÇÃO V*ERSUS* DEDUÇÃO 17
5.1 RELAÇÃO DE BASES PADRÃO DA LITERATURA 74
5.2 ALGUMAS OBSERVAÇÕES DA BDR 80

LISTA DE GRÁFICOS

6.1 NÚMERO DE VEZES QUE OS ALGORITMOS OBTIVERAM O MELHOR RESULTADO (MENOR ERRO) 94
6.2 NÚMERO DE VEZES QUE OS ALGORITMOS OBTIVERAM O MELHOR RESULTADO – SEM *BOOSTING* (MENOR ERRO) 95
6.3 NÚMERO DE VEZES QUE OS ALGORITMOS C5.0 E C4.5 OBTIVERAM O MELHOR RESULTADO - SEM *BOOSTING* (MENOR ERRO) 95
6.4 TEMPO DESPENDIDO COM OS EXPERIMENTOS ATRAVÉS DO MÉTODO LACHENBRUCH SOBRE A BDLP E BDR 97
6.5 RESULTADOS ANTERIORES E APÓS A EXPANSÃO DA BDR SEGUNDO FUNÇÃO DE FITNESS ERRO MÉDIO E CRITÉRIO DE REPETIÇÃO DE EXEMPLOS (MEDIDOS SOBRE OS DADOS DE TESTE) 108
6.6 RESULTADOS ANTERIORES E APÓS A EXPANSÃO DA BDR SEGUNDO FUNÇÃO DE FITNESS TAXA DE ACERTO SENSÍVEL AO DESBALANCEAMENTO DE CLASSES E CRITÉRIO DE REPETIÇÃO DE EXEMPLOS (MEDIDOS SOBRE OS DADOS DE TESTE) 109
6.7 CURVAS DE EVOLUÇÃO SEGUNDO FUNÇÃO DE FITNESS E CRITÉRIO DE REPETIÇÃO DE EXEMPLOS (MEDIDOS SOBRE OS DADOS DE VALIDAÇÃO) 110
6.8 DISTRIBUIÇÃO DAS CLASSES ANTES E DEPOIS DA EXPANSÃO DA BDR SEGUNDO A FUNÇÃO DE FITNESS E O CRITÉRIO DE REPETIÇÃO DOS EXEMPLOS 112

LISTA DE FIGURAS

1.1 - REPRESENTAÇÃO DE DESBALANCEAMENTO ENTRE CLASSES EM BASES DE EXEMPLOS 3
3.1 ÁREAS DE INTERDISCIPLINARIDADE DO KDD 24
3.2 ETAPAS DO PROCESSO KDD 26
3.3 O PROCESSO DE KDD *VERSUS* DATA MINING 28
4.1 ÁRVORE DE DECISÃO QUE REPRESENTA A CLASSIFICAÇÃO NA TABELA A.1 33
4.2 - PROCEDIMENTOS PARA A CONSTRUÇÃO DA ÁRVORE DE DECISÃO ADOTADOS PELO ALGORITMO CLS 36
4.3 PROCEDIMENTOS PARA A CONSTRUÇÃO DA ÁRVORE DE DECISÃO 41
4.4 ALGORITMO BÁSICO PARA A CONSTRUÇÃO DO *COVER* 46
4.5 PROCEDIMENTOS PARA GERAR UMA LISTA DE REGRAS ORDENADAS PELO CN2 54
4.6 PROCEDIMENTOS PARA GERAR UMA LISTA DE REGRAS NÃO-ORDENADAS PELO CN2 55
4.7 TRANSFORMAÇÃO DE ÁRVORES DE DECISÃO EM REGRAS DE PRODUÇÃO 58
4.8 PROCEDIMENTOS PARA GERAÇÃO DE VÁRIOS CLASSIFICAÇÕES 68
5.1 EXEMPLO DE SAÍDA DO C5.0 85
5.2 PROCESSO DE GERAÇÃO DE NOVAS POPULAÇÕES DE INDIVÍDUOS NO AG 88
5.3 EXEMPLO DE GENOMA DA MELHOR TAXA OBTIDA ATRAVÉS DO AG 92
B.1 UMA REGRA COMPLETA E CONSISTENTE 126

1 INTRODUÇÃO

O ser humano continuamente toma decisões, ou simplesmente chega a determinadas conclusões baseadas no conhecimento que ele acumula ao longo de sua vida.

Aliado ao fato do conhecimento fazer parte de nosso dia-a-dia, vários motivos contribuíram para que a aquisição de conhecimento se tornasse objeto de pesquisa e investimentos na área da computação. Dentre estes motivos, podemos citar os seguintes: (DECKER, 1995)

- disponibilidade de grande capacidade de processamento a baixo custo. Quando os métodos estatísticos foram desenvolvidos nos anos 60 e 70, o poder dos equipamentos mais desenvolvidos eqüivaleria atualmente à linha PC;
- os instrumentos científicos geram facilmente volumes de dados muito grandes, tornando-os impossíveis de serem analisados pelos métodos tradicionais de armazenamento e recuperação de dados. A questão que se torna cada vez mais presente é a do tamanho deste conjunto e o da sua dimensionalidade. Um técnico pode trabalhar eficientemente com alguns milhares de observações, onde cada uma apresente não mais de uma dezena de atributos. Porém, trabalhar com alguns milhões de observações, onde cada uma possua dezenas ou centenas de atributos, passa a ser um outro tipo de problema, o que pode conduzir a uma situação extrema de "quanto maior a quantidade de dados, menor o volume de informação" (HAN, 1995);
- a introdução de um novo conjunto de métodos lógicos desenvolvidos pela comunidade de Inteligência Artificial, pela Estatística[1] e também pela Física.

---

[1] Historicamente, o desenvolvimento da Estatística tem resultado em um grande número de métodos de análise, que têm sido amplamente aplicados com o objetivo de encontrar correlações e dependências nos conjuntos de dados.

> A ciência da computação tem apresentado alguns métodos de aquisição de conhecimento, como por exemplo, ferramentas para a Descoberta de Conhecimento em Bases de Dados (KDD).[2] Este conhecimento, adquirido a partir do processamento de um conjunto de dados, pode ser estendido para outros conjuntos, a partir do momento que se assume que estes dois conjuntos possuem estruturas de dados similares.

As máquinas ainda estão longe de possuírem as habilidades humanas, como, por exemplo, a capacidade de síntese de uma tecnologia, formulação de hipóteses e a criação de modelos. Este processo de percepção de um fenômeno e a condução de uma análise investigativa ainda é uma exclusividade dos seres humanos. No entanto, as máquinas assumem um papel importante no auxílio destes processos, uma vez que podem reduzir grandes volumes de dados através de métodos de classificação, segmentação, etc.

Este trabalho tem dois objetivos principais. O primeiro é comparar a eficácia de 3 (três) algoritmos para aquisição de conhecimento baseados no paradigma de indução de regras, a saber CN2, C4.5 e C5.0. Esses algoritmos produzem listas de decisão, regras de produção ou árvores de decisão. Outro objetivo deste trabalho é propor e testar uma nova solução para o problema de desbalanceamento de classes em uma base de dados do mundo real. Essa solução é baseada no uso de um método híbrido algoritmo genético e árvore de decisão (C5.0). O algoritmo genético age como *wrapper* usando a saída do algoritmo de árvore de decisão, estado da arte, C5.0 para obtenção da função objetivo (*fitness)* das populações de indivíduos (soluções candidatas ao problemas de classes desbalanceadas). Esta solução é avaliada sobre um estudo de caso no domínio dos dados do Censo Demográfico de 1991.

O que justifica o primeiro objetivo é o fato de não haver disponível na literatura textos que comparem a eficácia de outros algoritmos em relação ao algoritmo C5.0.

Já para o segundo objetivo a justificativa é exposta no item a seguir.

---

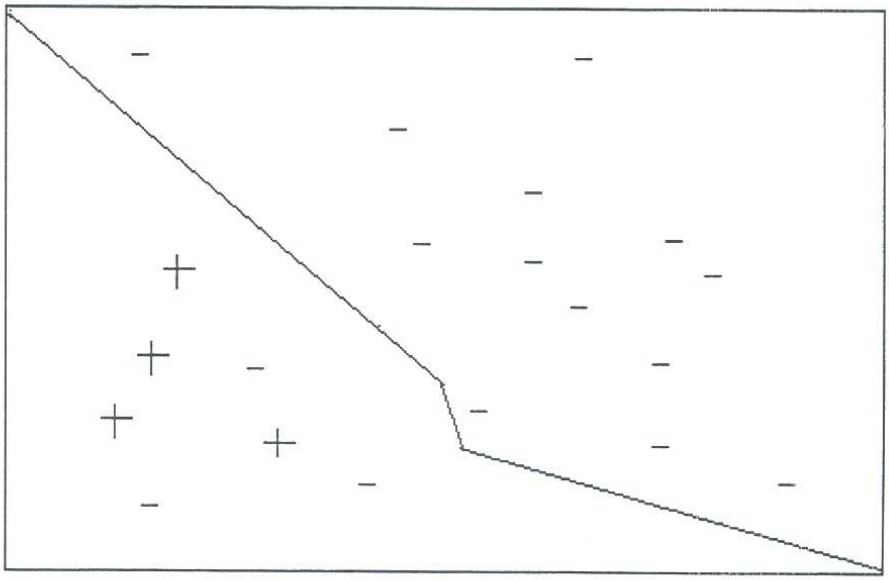
[2]Knowledge Discovery in Databases. Conceito proposto na primeira conferência internacional de KDD, em Montreal, em 1995 (FAYYAD, 1996).

1.1 TRABALHOS RELACIONADOS

As árvores de decisão são conhecidas como sendo classificadores universais: qualquer dicotomia de pontos em uma posição geral de um espaço contínuo n-dimensional pode ser representado por uma árvore de decisão.

No caso da figura 1.1, cada ponto é incluído em uma ramificação da árvore. Como pode ser observado, a representação dos exemplos negativos é bem maior que a representação dos exemplos positivos, dado que a freqüência destes últimos é significativamente menor.

FIGURA 1.1 - REPRESENTAÇÃO DE DESBALANCEAMENTO ENTRE CLASSES EM BASES DE EXEMPLOS



FONTE: (Kubat, 1997)

A indução de árvores de decisão, bem como outros paradigmas de aprendizado de máquina, sofre com o problema de desbalanceamento de classes em bases de dados. Quando o conjunto de treinamento (vide Apêndice B) apresenta uma distribuição não equilibrada do número de exemplos pertencentes às classes do domínio da aplicação, é mais fácil induzir as regras que cubram os exemplos da classe mais freqüente do que as regras que cubram os exemplos da classe menos freqüente (FREITAS, 1998).

Discutindo esta questão de desbalanceamento e questões afins existem alguns textos disponíveis, como, por exemplo, os seguintes:

- Kononenko (KONONENKO, 1991) e Provost (PROVOST, 1997) têm em comum o fato de discutirem métodos de avaliação das regras descobertas que são apropriados para problemas com desbalanceamento de classes. Entretanto, esses autores não propõem nenhuma solução para o problema de como descobrir regras a partir de dados com classes desbalanceadas. Em outros palavras, os métodos de avaliação propostos por estes autores, não envolvem mudança nem no algoritmo nem nos dados;
- Roberts (ROBERTS, 1995) e Turney (TURNEY, 1995) propõem métodos para lidar com classes desbalanceadas que consistem em modificar o algoritmo de *data mining* alvo; e
- Tomek[3], Kubat (KUBAT, 1997) e Chan (CHAN, 1998) propõem métodos que envolvem mudança nos dados sendo minerados. Mais precisamente, a idéia central desses 3 trabalhos é, em alto nível de abstração, remover exemplos da classe mais freqüente a fim de obter um conjunto de dados reduzido com classes mais balanceadas.

O método proposto e implementado neste trabalho segue a mesma idéia básica destes três últimos, porém sem remover exemplos da classe mais freqüente, ou criar vários conjuntos de treinamento onde os exemplos das classes menos freqüentes ocorrem em todos eles e os exemplos da classe mais freqüente apenas ocorre em um. Este método duplica os exemplos de algumas classes na tentativa de encontrar o balanceamento que promova os melhores resultados. Além disso, este trabalho utiliza um método de busca mais global e mais robusto que o utilizado por Kubat e Chan.

A seguir serão discutidos cada um dos trabalhos mencionados acima.

Kononenko propõe uma forma de avaliação de desempenho de classificadores no trato de problemas com distribuição desbalanceada de classes. Esta

---

[3] O texto de (TOMEK, 1976) foi citado em (KUBAT, 1997).

forma de avaliação apresenta a característica interessante de considerar a probabilidade prévia das classes. Para tal o cálculo é realizado da seguinte forma:

$$I_r = I_a / E * 100\% \qquad (1.1)$$

Sendo

E = a entropia das classes $\left(E = -\sum_i^n P(C_i) * \log_2 P(C_i)\right)$

$I_a = 1/T \sum_i^T I(j)$ $I_a = 1/T \sum_j^T I(j)$

onde I_r é o *relative information score*

I_a é a expectativa de informação necessária para a correta classificação

n é o número de classes

C é a classe correta para um determinado exemplo

T é o número de exemplos testados

Provost apresenta uma técnica para avaliar a performance de classificadores sujeitos a imprecisão na distribuição das classes e aos custos de erro de classificação denominada de ROC (Receiver Operating Characteristics) *convex*. Este método é derivado da curva ROC, que é uma representação gráfica utilizada para avaliar a performance de um grande número de classificadores. Em um gráfico ROC no eixo Y são representados os verdadeiro-positivos e no eixo X os falso-positivos. (KONONENKO, 1991). A novidade introduzida por Provost é a decomposição da performance do classificador a partir das classes específicas e de suas distribuições de custos (com o estabelecimento de custos para os falso-positivos e para os falso-negativos).

Roberts apresenta uma abordagem para integrar os custos de classificação errônea em algoritmos de árvore de decisão. Esta abordagem é baseada na geração de uma árvore de decisão sensível a custos. Para tal, são realizadas duas alterações no algoritmo de construção da árvore: no critério de seleção do nó folha que identifica a classe e no critério de seleção dos cortes dos nós internos.

A idéia básica é atribuir às classes mais raras um maior custo de classificação errônea. Como o algoritmo minimiza o custo de classificação errônea, o algoritmo

efetivamente tende a se concentrar mais nas classificação das classes mais raras. Essa é uma abordagem indireta para resolver o problema de classes desbalanceadas.

Turney propõe um algoritmo que constrói classificadores sensíveis a custos. Este algoritmo envolve Algoritmo Genético e Árvore de Decisão (o algoritmo C4.5 modificado). Para o AG é disponibilizada uma população de *biases* para o algoritmo de indução da AD, onde um bias é essencialmente um vetor com n posições, onde cada posição i, $i = 1, ..., n$ especifica o custo do i-ésimo atributo. A função objetivo do AG é a média de custos de classificação quando a AD é usada incluindo os custos dos testes (atributos, medidas) e de erros de classificação.

Analogamente ao trabalho de Roberts, o algoritmo tenta minimizar os custos de classificação errônea, o que indiretamente faz o algoritmo se concentrar nas classes mais raras.

Tomek propôs uma técnica para tratar o problema de bases com classes desbalanceadas a partir da remoção aleatória de exemplos da classe mais freqüente e mantendo inalterados os exemplos da classe menos freqüente.

Kubat sugere uma outra abordagem alterando a técnica de Tomek, não apenas com a remoção aleatória de exemplos da classe mais freqüente e a manutenção dos exemplos da classe menos freqüente, mas também com a remoção dos exemplos da classe mais freqüente que constituem um ruído na classe menos freqüente (exemplos identificados como "-", localizados abaixo da linha divisória na figura 1.1). A esta técnica, ele chamou de seleção unilateral de exemplos (*one-side selection of examples)*.

Chan propõe um multi-classificador baseado em meta-aprendizado capaz de atender ao desbalanceamento de classes e a uma distribuição não uniforme de custos por erro. Para tal são criados subconjuntos dos dados, e os resultados de algoritmos *Data Mining* aplicados a esses subconjuntos de dados são integrados através de meta-aprendizado (Meta-learning). Por exemplo, assume-se que uma aplicação com um balanceamento de classes de 20 / 80 deveria ser de 50 / 50. O grupo de 80 exemplos é dividido aleatoriamente em 4 subgrupos, cada um com 20 exemplos, sendo cada um desses 4 subgrupos concatenado como os 20 exemplos da classe mais rara. Logo, passam

a existir 4 conjuntos de treinamento, cada um deles com 50% de exemplos da classe mais rara e 50% da classe mais freqüente. Dessa forma são construídos 4 classificadores distintos, para a seguir gerar um único classificador a partir da combinação destes 4 classificadores usando o conceito de *meta-learning*. Esse conceito se baseia na predição dos classificadores sobre um conjunto de validação considerando a predição mais efetiva através do processo de voto (*voting-based techniques*).

1.2 ORGANIZAÇÃO DO TEXTO

O presente trabalho, além desta Introdução, contempla o Capítulo 2 onde são apresentadas e discutidas as principais formas de aquisição de conhecimento, os recursos humanos e as ferramentas necessárias, a relação existente entre o conhecimento e o aprendizado, e as formas de aprendizado. Apresenta-se também as vantagens e desvantagens destas formas de aquisição.

No capítulo 3 é apresentado e discutido o processo de descoberta de conhecimento, o KDD, e uma das etapas deste processo que efetivamente cumpre o papel desta descoberta, o Data Mining.

No capítulo 4 é analisado o processo de classificação, com base em métodos de indução. Também são descritos e discutidos alguns algoritmos que se enquadram no paradigma de indução de regras, o ID3, o AQ, o CN2, o C4.5 e o C5.0. Esses três últimos são objetos de comparação neste trabalho.

O capítulo 5 apresenta as bases de dados do mundo real e de exemplos disponíveis na literatura sobre as quais são realizados os experimentos, bem como a metodologia adotada para a realização dos mesmos. Também é proposto e testado o método para resolver o problema de desbalanceamento de classes em uma base do mundo real por meio de um processo híbrido integrando um algoritmo genético e um algoritmo de árvore de decisão (C5.0).

Finalmente, no Capítulo 6 são apresentados os resultados dos testes e no Capítulo 7 as conclusões, além das linhas de pesquisa que podem dar continuidade ao presente trabalho.

2 AQUISIÇÃO DE CONHECIMENTO

2.1 CONSIDERAÇÕES INICIAIS

Neste capítulo são apresentadas e discutidas as principais formas de aquisição de conhecimento, os recursos humanos e as ferramentas necessárias, a relação existente entre conhecimento e aprendizado e as formas de aprendizado. Também são apresentadas as vantagens e desvantagens destas formas de aquisição.

2.2 O PROCESSO DE AQUISIÇÃO DE CONHECIMENTO

Em geral, os processos que envolvem as atividades, os inventos e as ações dos seres vivos através dos tempos constituem de forma geral o conhecimento.

O ser humano tem como preocupação contínua a obtenção deste conhecimento e, com este objetivo, observa o significado inserido nessas atividades, inventos e ações. A partir da observação são geradas propostas que reformulam ou recriam as observações de um fenômeno, bem como determinados conceitos que possam ser aceitos pelos critérios de validação (MATURANA, 1987).

Define-se o conhecimento como um sistema de fatos e conceitos, onde um conceito é um símbolo que determina as características comuns ou relações compartilhadas entre os objetos e/ou os eventos (McGRAW, 1989). Conceitos são desenvolvidos no dia-a-dia pelas pessoas, de forma contínua, tornando-as capazes de agir de forma mais eficaz e/ou eficiente.

O "conhecimento pode ser definido como a informação armazenada, ou os modelos usados pela pessoa ou máquina para interpretar, predizer e responder apropriadamente ao mundo exterior" (ARARIBOIA, 1988).

O conhecimento, mesmo sendo um sistema de fatos e conceitos, não é sinônimo de informação. O conhecimento é a informação já interpretada, categorizada, aplicada e revisada.

Especificamente, o conhecimento consiste de uma descrição de definições e de relações, bem como de procedimentos para manipular estas descrições.

Existem várias atividades do cotidiano que exigem processos de tomada de decisão baseados no conhecimento. Alguns destes processos são objeto de estudo da área de Inteligência Artificial, onde pesquisas de novas metodologias são desenvolvidas visando a sua automação através de sistemas computacionais ou Sistemas Inteligentes Baseados em Conhecimento. Estes Sistemas dependem fundamentalmente de uma base de conhecimento e de mecanismos de raciocínio que possam resolver os problemas propostos.

Dentro do conjunto de Sistemas Inteligentes Baseados em Conhecimento existe um subconjunto denominado Sistemas Especialistas. O termo "especialista" sugere que estes sistemas sejam comparáveis aos especialistas humanos que existem em várias áreas do conhecimento. Essas pessoas não apenas são profundos conhecedores de fatos e procedimentos envolvidos em determinado domínio, mas também são pessoas que ao longo de anos de experiência construíram um conhecimento consolidado que as permite informar e dirigir suas decisões. Muitas vezes elas são guiadas pela intuição, não sendo muito claro o que as levou a tal decisão.

O especialista, independentemente de ser humano ou um Sistema Computacional, deve possuir as seguintes características (HART, 1989):

- Eficácia – usar o conhecimento com uma taxa de sucesso aceitável, de forma consistente, sem conflitos;
- Eficiência – não é suficiente ter condições de resolver o problema, mas sim resolvê-lo de forma rápida e adequada, deduzindo as soluções mais prováveis e determinando as informações mais relevantes;
- Percepção das limitações – ter a habilidade de identificar quando a demanda supera a capacidade de resposta, ou seja, os seus limites de conhecimento;
- Versatilidade – trabalhar bem em situações novas, eventualmente com incertezas ao estar diante de situações distintas das situações mais comuns.

Um Sistema Especialista soluciona problemas que, normalmente, são solucionados por especialistas humanos. Para solucionar tais problemas estes sistemas necessitam de (HART, 1989):

- acesso a uma base de conhecimento que represente o domínio da aplicação. Este acesso deve prever a recuperação, o raciocínio a partir dessa base e a aquisição de mais conhecimento;
- um ou mais mecanismos de raciocínio que permitam aplicar o conhecimento aos problemas a serem solucionados. Por exemplo, os seres humanos possuem vários métodos de raciocínio, entre os quais se destacam os raciocínios dedutivo, indutivo e analógico. O raciocínio dedutivo teve sua origem entre os gregos e tornou-se, durante toda a história da ciência, o método mais seguro para obtenção do conhecimento a partir de conhecimentos prévios. O raciocínio indutivo é muito usado nas ciências experimentais. Consiste em enunciar uma lei geral a partir de várias ocorrências particulares de um determinado fenômeno. O raciocínio por analogia tem como princípio estabelecer uma correspondência entre os elementos de dois sistemas distintos;
- uma forma de explicar aos usuários como foi encontrada ou determinada a solução para o problema.

Em um Sistema Especialista, o conhecimento está contido em bases de conhecimento e o processo de construção destas bases é chamado de engenharia do conhecimento.[4]

De forma a esclarecer o papel e o escopo da engenharia do conhecimento apresenta-se no quadro 2.1 uma analogia entre a engenharia de conhecimento e uma atividade bastante conhecida: a programação convencional de computadores (RUSSEL, 1995).

---

[4]Neste processo de construção, o Engenheiro de Conhecimento é a pessoa responsável pela pesquisa em um determinado domínio, pela determinação de quais conceitos são os mais importantes sobre o domínio e pela criação de uma representação formal dos conceitos e das relações sobre os conceitos que compõem o domínio. O Engenheiro do Conhecimento deve apresentar bom desempenho em: comunicação, conceituação, tato e diplomacia; persistência; e raciocínio lógico. O trabalho do engenheiro é sempre acompanhado pelo especialista do domínio para juntos desenvolverem a tarefa de aquisição do conhecimento.

QUADRO 2.1 - ENGENHARIA DO CONHECIMENTO *VERSUS* PROGRAMAÇÃO CONVENCIONAL

ENGENHARIA DO CONHECIMENTO	PROGRAMAÇÃO CONVENCIONAL
Seleção de uma forma de representação do conhecimento	Seleção de uma linguagem de programação
Construção de uma base de conhecimento	Escrever um programa
Implementar uma teoria de prova	Escrever ou selecionar um compilador
Inferência de novos fatos	Executar um programa

2.3 ABORDAGENS DA AQUISIÇÃO DE CONHECIMENTO

Basicamente, existem duas abordagens adotadas na aquisição de conhecimento[5]: a abordagem manual e a abordagem automática.

Na forma manual, o domínio do conhecimento é obtido através de processos interativos entre o especialista humano e o engenheiro do conhecimento. O engenheiro de conhecimento deve construir a base de conhecimento. Do especialista humano não será apenas exigido profundo conhecimento sobre o assunto em questão, mas também clareza e objetividade ao expressar este conhecimento.

Os processos interativos são baseados em entrevistas, a partir das quais o engenheiro de conhecimento, cuidadosamente, armazena toda a informação que é relatada para posterior verificação e análise. Da mesma forma, deve estimular o especialista humano a descrever a sua experiência da maneira mais objetiva possível, podendo para isso alternar as técnicas de entrevista.

Depois que um modelo inicial estiver pronto, este deve ser refinado até aproximar-se do nível de desempenho semelhante ao do especialista.

Pode-se perceber que esta abordagem é cara e lenta, aliado ao fato de seu sucesso estar totalmente dependente da inter-relação entre os profissionais envolvidos.

Na abordagem automática, a aquisição ocorre através de programas computacionais que extraem o conhecimento especializado a partir de métodos de aprendizado de máquina. Esses métodos deflagraram uma verdadeira explosão de pesquisas nos últimos anos, principalmente devido aos seguintes fatores:

---

[5] Não faz parte do escopo deste trabalho analisar a questão da aquisição semi-autimática do conhecimento no que se refere aos Sistemas Especialistas.

- criação de grupos de pesquisa especializados nas áreas de aprendizado de máquina, na teoria do aprendizado computacional, em redes neurais e no reconhecimento de padrões;
- aplicação de técnicas de aprendizado de máquina em problemas tradicionais, como o reconhecimento de padrões (voz, escrita e face), a análise de dados médicos, etc.;
- aplicação de técnicas de aprendizado de máquina em novos tipos de problemas, a exemplo da descoberta de conhecimento em bases de dados, do processamento de linguagem natural, do controle de robôs e da otimização combinatória.

Antes de tratar especificamente do aprendizado de máquina no item 2.3.2 do presente Capítulo, é importante apresentar as idéias principais que estão embutidas no conceito e nas estratégias de aprendizado.

2.3.1 Conceito e Estratégias de Aprendizado

O aprendizado não se resume apenas na percepção de um fenômeno a ser utilizado no momento atual, mas requer também a possibilidade de orientar uma atitude/decisão futura. O "aprendizado é o resultado da interação entre o agente aprendiz[6] e o mundo real, capacitando este agente para a tomada de suas próprias decisões a partir de outras observações previamente feitas" (RUSSEL, 1995).

O processo de aprendizado pode ocorrer através de diferentes estratégias, que são apresentadas a seguir, segundo a ordem crescente de complexidade com relação à inferência realizada pelo aprendiz. No entanto, se estas estratégias podem ser assim classificadas, deve-se sempre ter em mente que em todo ato prático de aquisição de conhecimento mais de uma estratégia pode ser utilizada simultaneamente.

Segundo Castiñeira, podem ser identificadas as seguintes estratégias de aprendizado (CASTIÑEIRA, 1990):

---

[6]O agente aprendiz é o agente que aprende, podendo ser uma pessoa ou mesmo um sistema computacional.

- implantação direta do conhecimento ou aprendizado por hábito - nenhuma inferência ou qualquer outro tipo de transformação do conhecimento é requerido. O conhecimento fornecido é diretamente aceito pelo aprendiz;
- aprendizado por instrução - o conhecimento é adquirido através de um instrutor ou outra fonte organizada. O aprendiz seleciona os fatos mais relevantes, transformando-os para o seu formato de representação interna;
- aprendizado por dedução - os conceitos são adquiridos através de dedução sobre conhecimento já possuído. Um caso desta abordagem é o aprendizado por explicação, "explanation-based learning", onde a definição abstrata de um conceito é transformada em uma definição operacional, utilizando um conceito-exemplo como guia;
- aprendizado por analogia - é freqüentemente adotada pelos seres humanos, na medida que acreditam que uma determinada situação que estão vivendo é semelhante à alguma outra já experimentada. Se a situação anterior sugeria uma determinada ação, então esta também deve ser a ação para a situação presente;
- aprendizado por indução - o sistema aprendiz adquire os conceitos através da utilização de inferências indutivas realizadas sobre fatos fornecidos ou observados. Este trabalho tratará especificamente desta estratégia de aprendizado.

Aprendizado por Indução *versus* Dedução

No aprendizado por indução, o aprendiz induz a descrição do conceito através de exemplos e, opcionalmente, de contra-exemplos do conceito. Ou seja, dado um conjunto de exemplos positivos (objetos pertencentes a uma classe pré-definida) e um conjunto de exemplos negativos (objetos não pertencentes àquela classe), o aprendiz observa os conjuntos e reconhece as similaridades entre os exemplos procurando identificar uma descrição da classe, também chamada de conceito, que possa cobrir todos os exemplos positivos e excluir todos os exemplos negativos.

Pode-se compreender melhor o método de indução por meio de sua comparação com o método dedutivo.

Num processo de dedução, são oferecidos os fatos ou as regras para que sejam feitas as deduções, conforme o definido no exemplo 2.1:

EXEMPLO 2.1 - OBRIGATORIEDADE DO VOTO

Regra

- Toda pessoa com idade entre 18 e 64 anos tem obrigação de votar.

Fato

- Mário tem 20 anos de idade.

Dedução

- Mário tem obrigação de votar

Se as regras não são conhecidas, então é necessária uma investigação para que as regras sejam induzidas. É procurada uma descrição que constitua a melhor regra. É escolhida uma descrição inicial, por exemplo, uma descrição que cubra todo o universo de exemplos. Esta descrição é transformada através de operações de generalização e especialização até que atinja um determinado critério previamente exigido.

Na generalização, o objetivo é identificar as situações comuns ao longo da estrutura, enquanto que na especialização o objetivo é identificar as diferenças significativas.

Definição 2.1 Generalização - um conjunto de descrição $D = (A_1 \in S_1 \wedge \ldots \wedge A_i \in S_i \wedge \ldots \wedge A_n \in S_n)$ é generalizado pela expansão do conjunto de valores para um atributo em particular A_i para S'_i onde $S_i \subset S'_i \subseteq Dom_i$[7].

[7]Conforme Holsheimer (HOLSHEIMER, 1994).

A aplicação da operação de generalização para uma descrição D^8, resulta uma nova descrição D' que cobre um número maior de objetos, ou seja, $\sigma_D(S) \subseteq \sigma_{D'}(S)$. Desta forma, se um objeto é coberto pela descrição D, ele também será coberto pela descrição D'.

No entanto, o inverso não é verdadeiro. A operação de generalização não é preservadora do acerto da classificação, isto é, se uma regra classifica perfeitamente os objetos, então uma generalização da regra não necessariamente produzirá uma classificação tão boa. A generalização das regras preserva o erro de classificação, ou seja, se uma regra classificava erroneamente um objeto, a sua generalização continuará classificando-o da mesma forma.

A operação de especialização é o inverso da operação de generalização, onde o conjunto S_i é substituído pelo conjunto S'_i sendo que $S_i \supset S'_i$.

A indução é um processo inverso ao da dedução, ou seja, parte-se do específico para o geral. Nesta abordagem a descrição inicial é construída a partir de um conjunto formado por todos os exemplos pertencentes a uma determinada classe. Esta descrição é completa, mas por outro lado muito complexa. Para reduzir a complexidade, a descrição é modificada a partir de sucessivas generalizações e especializações. Este processo resulta em uma regra mais geral, idealmente em condições de classificar os exemplos corretamente.

A indução ocorre a partir da descrição contida no conjunto de exemplos e contra-exemplos fornecidos como entrada, também chamado de conjunto de treinamento. Se as regras geradas forem de boa qualidade, elas não só expressam os conceitos contidos nos exemplos apresentados, mas também podem ser aplicadas para as novas situações que venham a ocorrer.

De forma semelhante ao exemplo 2.1 pode-se esclarecer o processo de indução com base nos dados apresentados pela tabela 2.1.

---

[8]Uma descrição do espaço D é o conjunto de todas as descrições construídas, na representação em questão. Para cada descrição D em $\mathcal{D}$ (conjunto de todas as descrições) corresponde a um subconjunto de S, chamada de *cover* (cobertura) $\sigma_D(S)$

TABELA 2.1 - NOMES, IDADES E OBRIGATORIEDADE DO VOTO

NOME	IDADE	OBRIGATORIEDADE DE VOTO
Mário	20	Sim
Maria	5	Não
Flávia	16	Não
Sérgio	19	Sim
Luís	4	Não
Ana	5	Não
Júlio	63	Sim
Mercedes	70	Não

O resultado de um processo de indução a partir destes dados é a regra: **se a pessoa tiver idade compreendida no intervalo entre 19 e 63 tem obrigação de votar.**

Torna-se evidente o problema com relação às idades de 17 e 18 anos, bem como 64 e 69 anos, pois a regra induzida pode levar a uma previsão errada, em virtude destas idades não estarem representadas no conjunto de treinamento. Como produto da indução, é obtida uma regra não necessariamente correta. Desta forma, pode-se concluir que os conjuntos de treinamento, fornecidos como entrada para os processos de indução de regras, devem ser os mais completos possíveis e considerar os aspectos típicos mais relevantes do domínio que representam. Quanto mais completo e adequado for o conjunto de treinamento, mais confiável será a regra induzida.

No processo de dedução, o conhecimento é expresso pelas premissas e conclusões das regras, e pelas operações lógicas, enquanto que, no processo indutivo, o conhecimento será "descoberto" a partir do conjunto de treinamento.

A eficácia na dedução depende diretamente da qualidade das premissas, enquanto que os processos de indução dependem da qualidade do conjunto de treinamento e do algoritmo[9] reproduzido pelo programa computacional. Se o conjunto de exemplos omite certa informação, ou não a representa corretamente, o algoritmo não poderá expressar um conceito que seja sensível a ela.

O quadro 2.2 sintetiza as principais diferenças entre os métodos indutivo e dedutivo.

---

[9]Estes algoritmos procuram variar nos seguintes aspectos (HART, 1989): na forma pela qual eles pesquisam o conjunto de treinamento; na forma pela qual eles generalizam e na forma pela qual eles tratam os erros e os ruídos do conjunto de entrada.

QUADRO 2.2 - INDUÇÃO *VERSUS* DEDUÇÃO

ITEM	INDUÇÃO	DEDUÇÃO
Característica	Específico → Geral	Geral → Específico
Eficácia	Depende da qualidade do Conjunto de treinamento	Depende da qualidade das Premissas

A questão da indução pode ser tratada como sendo f um valor função $\{0, 1\}$, definido sobre o conjunto de n variáveis $\{x_1, x_2, ..., x_n\}$, onde cada uma destas variáveis representa alguma propriedade ou atributo dos objetos/fatos do domínio e onde o valor da função para um objeto/fato corresponda à respectiva classe. No caso do exemplo 2.1 (Obrigatoriedade do Voto) para uma determinada pessoa X_1, os atributos $\{x_1, x_2, ..., x_n\}$ e $f(X_1)$ resulta: sim (1) para o caso de ter a obrigatoriedade do voto; ou não (0) para o caso de não ter a obrigatoriedade do voto.

O par $<X_1, f(X_1)>$ representa um par de situação-resposta ou um exemplo da função f. Uma coleção $S=\{<X_1, f(X_1)>, <X_2, f(X_2)>, <X_2, f(X_2)>,..., <X_m, f(X_m)>\}$ é chamada de amostra da função f e m é o número de exemplos da amostra ou o tamanho da amostra.

Desta forma, pode-se concluir que um processo indutivo é categorizado por componentes fornecidos e por componentes procurados. Os componentes fornecidos podem ser dados por uma amostra f de tamanho razoável e os componentes procurados resultam de um procedimento que simule f. A saída produzida por um processo induzido é dito hipótese do processo e f é dito rótulo (*label)* do conceito.

2.3.2 Fases do Aprendizado de Máquina Indutivo

O processo de aprendizado deve obedecer ao seguinte ciclo de fases:

- **observação** - tornar disponível um conjunto de observações;
- **análise** - procurar encontrar padrões nestas observações;
- **teoria** - ao serem identificadas as regularidades, é formulada uma hipótese para explicá-las;

- **predição** - a hipótese deve prever uma nova ocorrência do fenômeno, podendo ser verificada através do processamento de uma nova observação.

Nesta última fase, a predição pode estar correta, o que corrobora a hipótese; ou pode estar incorreta, o que exige que novas observações sejam analisadas e uma nova hipótese seja estabelecida.

Existe uma grande variedade de técnicas capazes de implementar em computadores o conceito de aprendizado de acordo com este ciclo de fases. Porém, a maioria delas são baseadas na idéia que um algoritmo de aprendizado "aprenda" e produza uma hipótese tão completa e consistente[10] quanto possível.

2.4 PARADIGMAS DA AQUISIÇÃO AUTOMÁTICA DE CONHECIMENTO

Os programas que implementam em computadores o conceito de aprendizado se encontram em diversas áreas de pesquisa. No entanto, contemplam alguns objetivos comuns e metodologias semelhantes de avaliação.

Com base nestas similaridades, os pesquisadores tendem a organizá-los atendendo a 4 paradigmas que se diferem basicamente na forma de representação, performance e nos algoritmos empregados (FREITAS, 1998).

2.4.1 Indução de Regras

Estes programas geram descrições de conceitos através de generalizações e especializações a partir de exemplos e, opcionalmente, também de contra-exemplos do mesmo conceito, previamente organizados e fornecidos como entrada para estes programas.

Assim, a título de exemplo, o conceito de ave é formulado através de vários exemplos positivos, tais como: papagaio, pardal, avestruz, galo, galinha, etc. Na

---

[10] Uma hipótese é completa se ela é capaz de reconhecer todas as suas instâncias. Ou seja, ela não realiza predições equivocadas e não desconsidera "bons exemplos". Pelo contrário, ela os trata como sendo ocorrências da hipótese. Uma hipótese é dita consistente se ela não classifica qualquer exemplo negativo como uma ocorrência da hipótese.

formulação do conceito podem também ser usados contra-exemplos, tais como avião, borboleta, mosquito, etc. A partir destes exemplos e contra-exemplos, o sistema induz que as aves são animais com penas, ovíparos e vertebrados de acordo com os respectivos atributos agregados aos exemplos e contra-exemplos. O programa também é capaz de distinguir as aves de outros objetos e/ou animais.

Conforme foi visto anteriormente, o aprendizado através de exemplos é normalmente chamado de indução.

2.4.2 Redes Neurais

As Redes Neurais Artificiais (RNA) funcionam conceitualmente de forma similar ao cérebro humano, tentando reconhecer regularidades e padrões de dados, e são capazes de aprender com a experiência e fazer generalizações baseadas no seu conhecimento previamente acumulado.

Uma RNA é composta de um grupo de processadores (unidades ou neurônios), cada um possuindo uma pequena quantidade de memória. Os neurônios são conectados através de ligações (conexões). Cada conexão tem um peso numérico associado a ela. Os neurônios operam apenas em sua área de dados local e sobre os dados recebidos pelas conexões de entrada. Alguns neurônios são ligados ao ambiente externo e podem ser designados como unidades de entrada ou de saída.

Cada neurônio tem um conjunto de conexões de entrada, um conjunto de conexões a outros neurônios, um nível corrente de ativação e um significado para o processamento desse nível. A idéia é que cada neurônio processe de forma local, baseado nas entradas a partir de seus vizinhos, mas sem a necessidade de qualquer controle global sobre o conjunto de neurônios como um todo.

As RNA´s trabalham com o conceito de treinamento por meio dos pesos, os quais são ajustados através dos dados de entrada (BIGUS, 1996). Os pesos são modificados ao tentar trazer o comportamento da rede mais próximo do comportamento dos dados de entrada. Em outras palavras, as RNA´s adquirem o conhecimento a partir de exemplos e exibem capacidade para a generalização (além dos dados de treinamento em si).

2.4.3 Aprendizado Baseado em Casos

Os sistemas baseados em casos representam o conhecimento na forma de experiências e confiam na flexibilidade dos métodos para medir similaridades entre casos, para recuperar estes casos e aplicá-los a novas situações. "Um especialista, quando encontra um novo problema, em geral se lembra de casos semelhantes vistos no passado, dos resultados desses casos, e talvez do raciocínio por trás deles. Os problemas novos são resolvidos por analogia com os antigos, e as explicações são colocadas em termos de experiências passadas" (KOLODNER, 1993).

Estes sistemas baseados em casos utilizam exemplos representativos a partir da base de dados para aproximar um modelo, podendo realizar predições sobre novos exemplos, derivando as propriedades de situações similares identificadas no conjunto de experiências armazenadas. Uma das técnicas mais usadas é a classificação a partir do "vizinho mais próximo".

A eficiência de um Sistema Baseado em Casos depende de cinco fatores (KOLODNER, 1993):

- do conjunto de experiências passadas;
- da habilidade de entender novas situações em termos destas experiências passadas;
- da capacidade de adaptação;
- da capacidade de avaliação; e
- da habilidade de armazenar novas experiências ao conjunto.

2.4.4 Algoritmos Genéticos

Este é um paradigma baseado no princípio da seleção natural, onde os indivíduos (soluções candidatas) evoluem para melhores indivíduos (melhores soluções) (GOLDBERG, 1989).

No contexto de aquisição de conhecimento, os algoritmos genéticos podem ser usados para descobrir regras de produção. A interpretação mais comum deste conhecimento emprega um processo de combinação lógica associando às regras (indivíduos de uma população) e uma medida de qualidade, *fitness*, de um indivíduo.

O operador padrão de aprendizado neste paradigma gera novas regras candidatas a partir dos "pais" que tenham *fitness* altas, onde os pesos refletem alguma medida de performance nos casos de treinamento (LANGLEY, 1996).

Este paradigma reflete um comportamento de busca paralela, onde vários indivíduos de uma população representam soluções em diferentes pontos do espaço de busca.

2.5 CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentadas e discutidas as abordagens para aquisição de conhecimento (manual e automática), as várias estratégias de aprendizado, bem como as fases do aprendizado de máquina. Foram, também, apresentados os quatro principais paradigmas de aquisição automática de conhecimento e um comparativo entres esses paradigmas.

Um argumento que favorece a abordagem automática em relação a abordagem manual na questão da aquisição do conhecimento é que na abordagem automática é possível, em certos domínios, que o sistema exceda a capacidade do especialista humano na identificação das relações entre os conceitos, existindo assim a possibilidade de serem descobertas algumas descrições ou mesmo relações nunca antes imaginadas. Essa vantagem se justifica pelo fato que, na abordagem manual, dificilmente será adquirido algum conhecimento que possa vir a ser surpreendente. Isso porque o especialista tende a fornecer o conhecimento já adquirido por ele.

É importante salientar que nenhuma das duas abordagens dispensa o trabalho do especialista humano. Pelo contrário, será sempre o especialista humano quem fornecerá a validação final do conhecimento extraído tanto na abordagem manual quanto na abordagem automática.

Outra vantagem da aquisição automática de conhecimento é a de permitir que o especialista humano se concentre na elaboração do conjunto de treinamento que será fornecido como entrada para o processo, ao invés de dedicar tempo e esforço na transmissão de conhecimento ao engenheiro do conhecimento.

Uma terceira vantagem é que alguns programas computacionais que realizam a tarefa de aquisição de conhecimento podem prever em suas rotinas o tratamento de possíveis inconsistências. No caso da abordagem manual, esta consistência fica a cargo do engenheiro de conhecimento, lembrando que, se o processo de aquisição de conhecimento envolver mais de um especialista humano, a tendência de ocorrerem inconsistências é muito maior.

3 O PROCESSO DE KDD E DATA MINING

3.1 CONSIDERAÇÕES INICIAIS

A descoberta automática de conhecimento a partir dos dados – usar os computadores para descobrir novos significados de informações – é um dos objetivos mais fascinantes da ciência da computação.

Cada vez mais, os volumes de informação excedem a capacidade de sua análise pelos métodos tradicionais (planilhas, consultas e gráficos). Estes métodos podem gerar relatórios a partir dos dados, mas não conseguem analizá-los sob o enfoque do conhecimento. Para atender a esta necessidade foram pesquisadas e desenvolvidas novas técnicas e ferramentas, que permitem a extração de conhecimento a partir de grandes volumes de dados.

Neste capítulo é apresentado e discutido o processo de descoberta automática de conhecimento, o KDD, e mais especificamente uma de suas etapas, a que efetivamente cumpre o papel de descoberta de conhecimento, denominada Data Mining.

3.2 O PROCESSO DE KDD

Historicamente, a noção de encontrar "pepitas - *nuggets*" de conhecimento em um conjunto de dados tem recebido diversos nomes: descoberta de conhecimento em bases de dados, Data Mining, extração de conhecimento, descoberta de informação, "colheita" de informação, arqueologia de dados e processamento de padrões de dados.

Em 1989, foi convencionado o termo KDD - *Knowledge Discovery in Databases* como uma referência geral a processos de descoberta de conhecimento em bases de dados (FAYYAD, 1998).

O processo de KDD é interdisciplinar e envolve áreas relativas a aprendizado de máquina, reconhecimento de padrões, bases de dados, estatística e matemática, aquisição de conhecimento para sistemas especialistas e visualização de dados. Este

processo utiliza métodos, algoritmos e técnicas oriundos destas diversas áreas, com o objetivo principal de extrair conhecimento a partir de grandes bases de dados.

A interdisciplinaridade de áreas no processo KDD pode ser visualizada pela figura 3.1:

FIGURA 3.1 - ÁREAS DE INTERDISCIPLINARIDADE DO KDD

FONTE: (ADRIAANS, 1996)

Aprendizado de Máquina

Como foi visto no capítulo 2, nesta área são utilizados modelos cognitivos ou estratégias de aprendizado de máquina, bem como os paradigmas para a aquisição automática de conhecimento.

Reconhecimento de Padrões

Nesta área concentram-se estudos sobre as teorias e os algoritmos para extração de padrões e modelos,[11] principalmente a identificação de padrões especiais, chamados de conhecimento útil e interessante sobre os dados. Os modelos trabalham com todo o conhecimento inferido, porém, um especialista humano deve avaliar se estes modelos refletem ou não um conhecimento útil e interessante, conforme já foi colocado no capítulo 2.

---

[11]Os termos padrões e modelos muitas vezes são usados de forma quase que indistinta. Porém, vale lembrar que um padrão pode ser definido como sendo uma instância de um modelo. Ex.: $f(x) = 3x2 + x$ é um padrão. $f(x)=\alpha x2 + \beta x$ é um modelo.

Bases de Dados

Na área de bases de dados existem tecnologias específicas, bem como uma série de pesquisas que objetivam melhor explorar as características dos dados a serem trabalhados. Como, por exemplo, pesquisas que trabalham iterativamente com bases de dados relacionais de clientes em atividades de marketing (FU, 1996).

Estatística e Matemática

É freqüente que modelos matemáticos ou estatísticos sejam construídos para a geração de regras, padrões e regularidades. Como por exemplo, a construção de uma rede de Bayes[12] a partir de um determinado conjunto de treinamento, onde as relações entre os objetos podem ser extraídas a partir dos parâmetros e das ligações da própria rede. Os nós desta rede representam as variáveis ou estados e os arcos representam as dependências entre os nós de forma direcionada a partir da causa para o efeito.

No caso específico da Estatística, essa disponibiliza um grande número de procedimentos técnicos e resultados de testes para as tarefas de Data Mining, como por exemplo, para verificar se estimativas e procedimentos de pesquisa estão consistentes sob determinados critérios de avaliação e identificar o grau de incerteza. (GLYMOUR, 1997).

Sistemas Especialistas

Os sistemas especialistas são programas complexos de Inteligência Artificial criados para resolver problemas do mundo real (capítulo 2). Inicialmente, estes sistemas ofereciam apenas mecanismos para a representação do conhecimento, raciocínio e explicações. Posteriormente foram incorporadas ferramentas para a aquisição do conhecimento.

Visualização de Dados

A Visualização de Dados assume um papel importante já que em vários momentos existe a necessidade de interação entre o processo de descoberta e o ser

---

[12]A noção básica da teoria de Bayes é a probabilidade condicional, onde P (H / E) sendo P a probabilidade, H a hipótese e E a evidência (RUSSEL, 1995).

humano. Pode-se citar como exemplo, a análise prévia dos dados que vão ou não fazer parte do processo, onde são realizadas algumas consultas usando ferramentas de análise ou mesmo de visualização de dados.

Para a visualização, pode-se recorrer a distintas formas, tais como: gráficos, ícones e figuras.

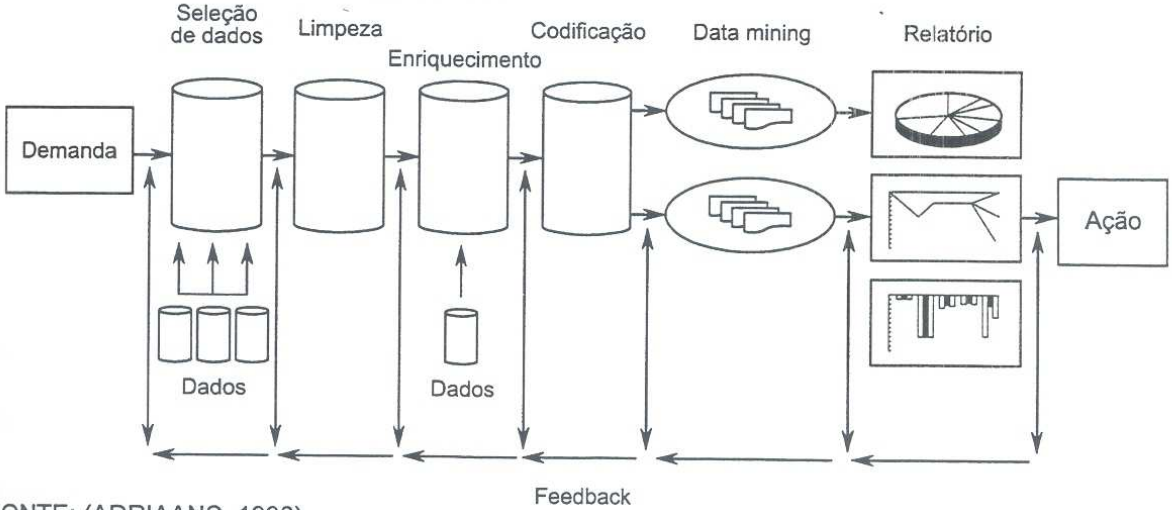
3.2.1 Etapas do Processo de KDD

O processo de KDD é um conjunto de atividades contínuas que compartilham o conhecimento descoberto a partir de bases de dados. Este conjunto (figura 3.2) é composto de 6 etapas (FAYYAD, 1996): seleção dos dados; limpeza; enriquecimento; codificação; Data Mining; e relatórios e consolidação do conhecimento descoberto.

Seleção de Dados

Uma vez definido o domínio sobre o qual se pretende executar o processo de descoberta, o próximo passo é selecionar e coletar o conjunto de dados ou variáveis necessárias. A maioria das empresas já possui bases de dados. Porém, nem sempre todos os dados necessários estão disponíveis nestas bases, o que exige um trabalho de compatibilização. Ou seja, os dados que são necessários, os que já estão disponíveis e a possibilidade de obtenção daqueles que não estão disponíveis.

FIGURA 3.2 - ETAPAS DO PROCESSO KDD



FONTE: (ADRIAANS, 1996)

Limpeza e Enriquecimento

É a atividade pela qual os ruídos – dados estranhos ou inconsistentes – são tratados e onde são estabelecidas as estratégias para resolver o problema de ausência de dados. As causas que levam à situação de ausência de dados são a não disponibilidade do dado ou a inexistência do mesmo. Uma situação de não disponibilidade ocorre quando da não divulgação do dado, como por exemplo, dados de renda de pessoa física em função da obrigatoriedade de sigilo. A inexistência do dado ocorre, por exemplo, quando são necessários dados sobre determinados municípios, incluídos no domínio do processo KDD, que, no momento de organização da base original, ainda não haviam sido criados.

Codificação

O processo de codificação se faz necessário quando existem dados que estão apresentados em um nível de detalhe não adequado para o processamento. Por exemplo, se forem processadas as datas de nascimento ao invés de idades ou faixas etárias, um algoritmo provavelmente irá tratar as pessoas (observações) em grupos de uma mesma data de nascimento (dia/mês/ano) o que, normalmente, gera um nível de especialização muito maior do que o necessário. Na grande maioria das aplicações, o nível de exigência é o de grupos etários, podendo chegar ao máximo a idades individuais, mas raramente ocorrem demandas para as datas de nascimento no formato dia/mês/ano. Desta forma, a data de nascimento necessita de codificação para ser transformada em idade ou mesmo em intervalos de idade dependendo da natureza da aplicação.

Data Mining

A atividade de descoberta do conhecimento é uma das mais fascinantes, onde são processados os algoritmos de aprendizado de máquina e de reconhecimento de padrões. A maioria dos métodos de Data Mining é baseada em conceitos de aprendizagem de máquina, reconhecimento de padrões, estatística, classificação, *clustering*, modelos gráficos, etc.

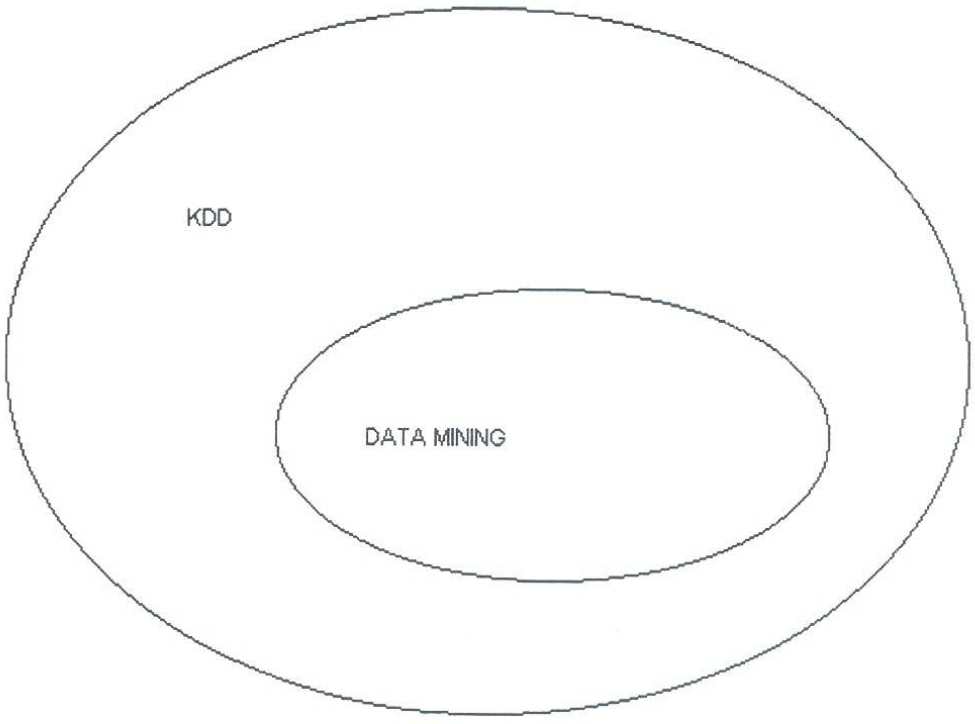
Relatórios e Consolidação do Conhecimento Descoberto

Os resultados do processo de descoberta do conhecimento podem ser mostrados de diversas formas. Porém, estas formas devem possibilitar uma análise criteriosa para identificar a necessidade de retornar a qualquer um dos estágios anteriores do processo de KDD.

Esta apresentação das atividades pode sugerir que exista uma trajetória linear do processo KDD. No entanto, isso geralmente não se verifica, uma vez que em cada etapa pode ser identificada a necessidade de retorno para qualquer uma das etapas anteriores. Por exemplo, se na atividade de codificação ou mesmo na de Data Mining, é identificado que os dados não estejam plenamente consistidos, ou se for verificada a necessidade de um dado que não havia sido previsto anteriormente, isso pode levar ao retorno para a fase de consistência ou mesmo de seleção dos dados.

A figura 3.3 representa graficamente a relação entre o processo de KDD e Data Mining, onde KDD se refere a todo o processo de descoberta do conhecimento e Data Mining compreende a etapa referente à aplicação de algoritmos para extração de padrões a partir dos dados.

FIGURA 3.3 - O PROCESSO DE KDD *VERSUS* DATA MINING



FONTE: (ADRIAANS, 1996)

Data Mining é uma das fases do processo de descoberta de conhecimento, para a qual têm sido descrito vários métodos. Dentre estes métodos destacam-se: Técnicas Estatísticas, Visualização, Raciocínio Baseado em Casos (Vizinho mais próximo), Árvores de Decisão, Regras de Associação, Redes Neurais e Algoritmos Genéticos.

É importante observar que, no capítulo 2, foram apresentados 4 paradigmas de aprendizado de máquina e neste ponto do trabalho são apresentados esses paradigmas como métodos adotados para Data Mining. Segundo Freitas (FREITAS, 1998b), é possível afirmar que o processo de Data Mining constitui uma evolução em vez de uma revolução sobre o processo de aprendizado de máquina. Esta afirmação se baseia no fato de que em Data Mining a compreensibilidade do conhecimento descoberto geralmente é ainda mais importante do que em aprendizado de máquina. Além disso, em Data Mining há uma ênfase na descoberta de conhecimento novo e interessante, o que em geral não é tão enfatizado em aprendizado de máquina.

3.3 DATA MINING

Antes de tratar mais especificamente de *Data Mining,* é importante também estabelecer um paralelo entre as áreas de pesquisa de *Data Mining* e de *Data Warehouse.*

As ferramentas de Data Mining exploram fenômenos ainda desconhecidos sobre os dados. Já a área de *Data Warehouse* está diretamente relacionada à disponibilização de várias bases de dados para a recuperação *on-line*, ou seja, trata de coletar, consistir e tornar os dados disponíveis para processos de análise e tomada de decisão. Alguns sistemas que implementam o conceito de *Data Warehouse* disponibilizam formas de consulta aos dados, tais como: *query*, relatórios, etc., que trabalham com fenômenos previamente conhecidos sobre os dados. Uma das abordagens mais comuns para análise de dados em *warehouses* tem sido a OLAP, a qual oferece uma análise multi-dimensional sobre os dados.

O *Data Mining,* bem como o *Data Warehouse,* adotam o conceito de meta-data, o qual descreve a estrutura do conteúdo das bases de dados com informações, tais como: qual dado existe, onde o dado está localizado, qual é o tipo e o formato do dado, como este dado está relacionado a outro dentro das bases, qual é a fonte do dado e a quem pertence.

3.3.1 Objetivos de Data Mining

Data Mining é uma metodologia para encontrar uma descrição lógica ou matemática, eventualmente de natureza complexa, de padrões e regularidades em um determinado conjunto de dados. Para isso, desenvolve fundamentalmente dois tipos de processos: generalizações a partir de conjuntos de observações conhecidas e detalhamentos desta estrutura a partir de suas descrições.

As várias tarefas desenvolvidas em Data Mining (vide Apêndice A) têm como objetivo primário a predição e a descrição. A predição usa atributos para predizer o desconhecido ou os valores futuros de outras variáveis de interesse. A descrição contempla o que foi descoberto nos dados sob o ponto de vista da interpretação humana (FAYYAD, 1996).

3.4 CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentado o processo de descoberta de conhecimento e como a tarefa de Data Mining se insere neste processo.

O Data Mining é um tipo especial de aprendizado de máquina onde o ambiente observado é uma base de dados do mundo real (HOLSHEIMER, 1994).

4 ESTRATÉGIAS DE ALGORITMOS PARA DATA MINING

4.1 CONSIDERAÇÕES INICIAIS

Um grande número de aplicações na área de Inteligência Artificial trabalha com a construção de modelos que sejam capaz de reproduzir o conhecimento sobre um determinado domínio. Este conhecimento, como foi tratado no capítulo 2, pode ser obtido através de métodos específicos de aquisição de conhecimento, entre os quais as técnicas de aprendizado de máquina.

Em geral, o processo de aprendizado de máquina pode ocorrer sob diversas estratégias de aprendizado e os seus algoritmos podem ser agrupados em quatro paradigmas distintos, conforme também foi tratado no capítulo 2.

Este capítulo descreve e discute o processo de classificação, com base na estratégia de indução e a partir dos algoritmos CN2, C4.5 e C5.0.

4.2 CLASSIFICAÇÃO

No processo de classificação, os algoritmos de indução de regras resolvem problemas do tipo: "descobrir regras para a previsão das classes (valores de uma variável dependente) de novos exemplos do domínio".

Como esta descoberta ocorre ao analisar experiências passadas, é necessário nesse processo, que um conjunto de exemplos dessas experiências esteja disponível, de forma que a variável dependente esteja tratada em k classes disjuntas, onde k representa o número de valores possíveis desta variável. Assim sendo, um novo exemplo poderá ser classificado, através da indução, em uma destas k classes.

Os processos de classificação trabalham com alguns elementos tais como: modelos de classificação, conjunto de treinamento, classes, regras e classificadores (HOLSHEIMER, 1994).

4.2.1 Modelos de Classificação

Em um modelo de classificação, a conexão entre as classes e os atributos previsores pode ser definida de forma simples, por grafos (*flowchart*) ou mesmo por um conjunto de procedimentos estruturados.

Estas conexões, expressas tanto nos grafos como nos conjuntos de procedimentos, devem contemplar as inúmeras classificações identificadas e reproduzidas no modelo, construído indutivamente através de generalizações e de especializações dos exemplos (ver seção 2.3.1).

Os algoritmos do paradigma de indução de regras geram classificadores que podem ser expressos como árvores de decisão, como conjuntos de regras de produção ou como listas de decisão. Estas formas de representação restringem a descrição da classe em uma expressão lógica que possui como primitiva as declarações sobre os valores dos atributos previsores.

Árvores de Decisão

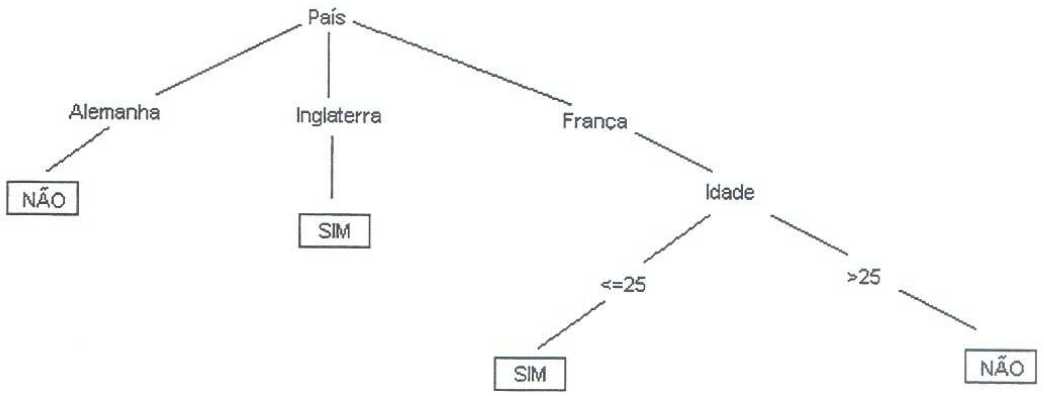
Uma árvore de decisão é uma representação simples de classificadores utilizada por diversos sistemas de aprendizado de máquina, como por exemplo, o ID3 e alguns de seus sucessores (C4.5 e C5.0), que serão apresentados no item 4.3 deste capítulo.

Uma árvore de decisão é induzida a partir de um conjunto de exemplos onde as classes são previamente conhecidas. Esta estrutura é organizada de tal forma que os seus nós internos representam as condições (atributos) e os nós folhas os resultados[13] previstos, possibilitando assim a classificação de novas observações. A figura 4.1 representa a classificação expressa na tabela A.1, onde esta árvore pode classificar corretamente todos as observações da tabela.

---

[13] São também chamados de classes ou de valores de atributo meta

FIGURA 4.1 - ÁRVORE DE DECISÃO QUE REPRESENTA A CLASSIFICAÇÃO NA TABELA A.1



FONTE: (Freitas, 1998)

Na figura 4.1 cada nó interno representa um corte sobre um atributo, como por exemplo País e Idade. Cada propriedade corresponde a uma ramificação da árvore, a citar País igual a Inglaterra. Os nós folhas representam as classificações, tal como Sim ou Não.

O processo de classificação ocorre ao caminhar pela árvore, a partir do nó raiz, procurando percorrer os arcos que unem os nós, de acordo com as condições que estes mesmos arcos representam. Ao atingir um nó folha, é identificada a classe à qual este exemplo pertence.[14]

Na tabela A.1 os exemplos são representados por um conjunto de atributos fixos e das classes às quais pertencem. Em geral, a forma mais simples desta representação ocorre quando os atributos estão representados por um domínio de poucos valores discretos. Porém, existem algoritmos capazes de processar sobre atributos de valores contínuos. As classes são representadas por valores discretos.

Regras de Produção

Devido ao fato das árvores de decisão apresentarem a tendência de crescimento muito rápido em aplicações do mundo real e, desta forma, aumentar a dificuldade de sua interpretação, alguns algoritmos transformam as árvores de decisão

[14]Outros aspectos relativos ao conceito de árvore de decisão serão tratados no capítulo 4.

em outras formas de representação, tais como as regras de produção.

As regras de produção que resultam da transformação de árvores de decisão têm as seguintes vantagens (HOLSHEIMER, 1994):

- são uma forma de representação do conhecimento amplamente utilizada em sistemas especialistas;
- possuem maior facilidade de interpretação pelo ser humano;
- melhoram a precisão (vide Apêndice B) de classificação pela eliminação de cortes que expressem peculiaridades pouco generalizáveis do conjunto de treinamento.

Listas de Decisão

O classificador apresentado através de listas de decisão resulta da generalização de representações da Forma Normal Conjuntiva[15] e da Forma Normal Disjuntiva,[16] sendo representado por uma lista de pares do tipo (HOLSHEIMER, 1994):

$$(\phi_1 , C_1), (\phi_2 , C_2), \ldots , (\phi_r , C_r)$$

onde cada ϕ é uma descrição elementar e cada C_i é uma classe, sendo a última descrição ϕ_r uma constante verdadeira.

A forma de leitura de uma lista de decisões é dada por: se ϕ_1 então C_1 caso-contrário se ϕ_2 então C_2 ... caso-contrário C_r

A classe de um objeto x é C_j se j for o menor índice da descrição ϕ_j que cobre o objeto x. Caso nenhuma descrição cubra o exemplo, será assumida a última classe da lista por se tratar da classe *default*.

O algoritmo CN2 pode usar listas de decisão como forma de representação do classificador, conforme será tratado no item 4.3.

---

[15]Um exemplo da Forma Normal Conjuntiva pode ser dado na seguinte forma : "(função = aluno ∨ função = professor) ∧ sede = centro", ou seja, uma conjunção de cláusulas, que sejam disjunções de condições de valores dos atributos. Esta expressão pode ser representada também sob a forma de conjuntos "função ∈ {aluno, professor} ∧ sede ∈ {centro}".

[16]A Forma Normal Disjuntiva é uma disjunção de termos, onde os termos são conjunções de condições sobre os valores dos atributos.

4.3 ALGORITMOS PARA CLASSIFICAÇÃO

Neste item são apresentados alguns algoritmos de aprendizado de máquina baseados na estratégia de indução e que se enquadram no paradigma de indução de regras.

Existem duas famílias de algoritmos de indução – a do ID3 e a do AQ - que têm sido bem aceitas em processos de aquisição de conhecimento.

A seguir, são apresentadas as características destas duas famílias de algoritmos, por se tratar de algoritmos base para o desenvolvimento de outros algoritmos, como os CN2, C4.5 e C5.0, cuja comparação constitui um dos objetivos deste trabalho.

4.3.1 ID3

O algoritmo ID3 (*Iterative Dichotomiser 3*) é a base de funcionamento dos algoritmos C4.5 e C5.0, bem como para o CN2 naquilo que se refere ao tratamento de ruídos.

Trata-se de um exemplo de algoritmo de estrutura TDIDT - *Top-Down Induction of Decision Trees* (QUINLAN, 1986), desenvolvido por Ross Quinlan, na década de 70, a partir do algoritmo CLS (*Concept Learning System*), também chamando de algoritmo de Hunt, desenvolvido por Marin Hunt e Stone em 1966 (QUINLAN, 1996).

Para a construção *top-down* (recursivo) de árvores de decisão, o ID3 utiliza o método *greedy* (guloso) que possui uma estratégia de "dividir para conquistar.[17]"

Pelo método *greedy,* a árvore inicia com um único nó que contém os exemplos do conjunto de treinamento. Se os exemplos forem todos de uma mesma classe, então o nó torna-se um nó folha e é rotulado com a respectiva classe. Caso contrário, o algoritmo estabelece um novo corte onde uma ramificação é gerada para cada valor do atributo de

---

[17]Esta estratégia quebra o problema em vários subproblemas que são similares ao problema original, mas de menor tamanho. Resolve os problemas de forma recursiva, para então combinar estas soluções para criar uma solução para o problema originalmente proposto (CORMEN, 1996).

decisão. Para este atributo de decisão é selecionado o atributo previsor com o maior ganho de informação. O algoritmo usa o mesmo processo para a construção de cada partição da árvore de decisão, finalizando a recursividade quando todos os exemplos de cada partição pertencerem a uma mesma classe, ou quando não existir mais forma de particionar os atributos dos exemplos remanescentes (KAMBER, 1997).

Assim, a partir de um conjunto *S* de exemplos de treinamento e sendo C_1, C_2, ..., C_n as classes do respectivo domínio do problema, a base de funcionamento do CLS é dada pelos procedimentos indicados pela figura 4.2 (QUINLAN, 1993):

FIGURA 4.2 - PROCEDIMENTOS PARA A CONSTRUÇÃO DA ÁRVORE DE DECISÃO ADOTADOS PELO ALGORITMO CLS

1) Se *S* contém apenas exemplos da classe C_i, então a árvore de decisão para *S* é uma folha que identifica C_i;
2) Se *S* não contém exemplos, então a árvore de decisão é uma folha. Assim, a classe deve ser definida por outros elementos que não S. Por exemplo, a classe mais freqüente no nó antecessor;
3) Se S contém exemplos de mais de uma classe, então deve ser selecionado um atributo para subdividir S.

No desenvolvimento do ID3 duas são as modificações introduzidas por Quinlan sobre o CLS: a aplicação do processo conhecido como *windowing* e a introdução de uma função heurística a partir da teoria da informação, a saber, a entropia (medida de homogeneidade dos exemplos) ou também chamada de medida ganho de informação (WU, 1995).

CLS ——————————→ ID3

Windowing + entropia

Segundo Quinlan (QUINLAN, 1993), quando ele começou a desenvolver o ID3 os computadores com memória virtual não eram comuns e os programas possuíam restrições quanto ao tamanho. Os conjuntos de treinamento dos primeiros experimentos eram grandes, contendo em torno de 30.000 exemplos com 20 atributos em média, o que

facilmente excedia a memória nos equipamentos disponíveis. Consequentemente, foi necessário desenvolver um método que construísse árvores de decisão para grandes conjuntos de dados.

Desta forma, foi desenvolvido o método denominado *windowing,* onde um subconjunto de exemplos – denominado *window* – é selecionado randomicamente e uma árvore de decisão é construída a partir dele. Esta árvore é utilizada para classificar os exemplos do conjunto de treinamento que ainda não tenham sido incluídos no *window*. É freqüente que, como resultado, restem alguns exemplos classificados erroneamente. Esses exemplos são agregados ao *window* original e uma nova árvore de decisão é construída a partir deste subconjunto, a qual é novamente submetida à classificação sobre o conjunto de treinamento. Este ciclo é repetido enquanto a árvore construída a partir deste *window* não classificar corretamente todos os exemplos contidos no conjunto completo de treinamento.

A entropia permite identificar como as classes estão distribuídas nos cortes definidos a partir dos dados do conjunto de treinamento a cada estágio de crescimento da árvore. Este crescimento decorre da avaliação dos atributos previsores e de seus respectivos valores de corte, tomando por base a entropia que um ou mais cortes oferecem.

O ganho de informação representa a diferença entre a quantidade de informação necessária para uma predição correta e as correspondentes quantidades acumuladas dos segmentos resultantes dos cortes. Para essa avaliação são considerados dois momentos: um, antes da inserção de um corte e, o outro, depois da sua inserção. Se a quantidade de informação requerida é muito menor depois que o corte é introduzido, isso indica que a inclusão deste corte reduz a desordem (entropia) do segmento original.

A entropia é uma medida bem-definida da desordem ou da informação encontrada nos dados. A introdução da entropia no processo de construção de árvores de decisão possibilita que sejam criadas árvores menores e mais eficientes (WU,1995).

A forma de obtenção da entropia é dada por (WU, 1995):

- $T = PE \cup NE$ onde PE é o conjunto de exemplos positivos e NE é o conjunto de exemplos negativos

- $p = |PE|$ e $n = |NE|$ onde $|PE|$ e $|NE|$ representam a cardinalidade de *PE* e *NE* respectivamente
- para cada exemplo será determinada uma probabilidade p/(p+n) e uma probabilidade n/*(p+n)*.

Assim, a entropia é definida pela quantidade de informação necessária para decidir se um exemplo pertence a *PE* ou a *NE*, segundo a seguinte fórmula (WU, 1995):

$$entropia(p,n) = - p/(p+n) \log_2 p/(p+n) - n/(p+n) \log_2 n/(p+n) \text{ para } p \neq 0 \text{ e } n \neq 0$$
$$entropia(p,n) = 0 \text{ caso contrário} \qquad (4.3)$$

Note-se que *entropia* (*p,n*) depende apenas de *p* e de *n*.

A entropia dos segmentos descendentes de um nó pai da árvore é acumulada de acordo com o peso de suas contribuições na entropia total do corte, ou seja, de acordo com o número de exemplos cobertos pelo corte.

Como avaliar entre dois cortes distintos qual deles diminui a entropia ao máximo, promovendo assim um maior ganho?

A tabela 4.1 (BERSON,1997) apresenta um exemplo onde é possível verificar que o corte *A* é melhor que o *B* porque ele separa mais os dados, mesmo considerando que o corte *B* cria um novo segmento perfeitamente homogêneo, ou seja, *com entropia 0 (zero). O problema é que este segmento de entropia perfeita cobre* apenas uma única observação, o que indica ser um corte que não gera uma árvore de decisão importante ou que venha a ser útil.

TABELA 4.1 - CÁLCULO DA ENTROPIA PARA DOIS CORTES DISTINTOS

CORTES	CORTE A ESQUERDA	CORTE A DIREITA	ENTROPIA ESQUERDA	ENTROPIA DIREITA
Corte *A*	++++-	+----	(-4/5 $\log_2$ (4/5)) + (-1/5 $\log_2$ (1/5)) =0.72	(-1/4 $\log_2$ (1/5)) + (-4/5 $\log_2$ (4/5))=0.72
Corte *B* -	+++++--	-	(-5/9 $\log_2$ (5/9)) + (-4/9 $\log_2$ (4/9)) =0.99	(-1/1 $\log_2$ (1/1)) + (-0/1lg (0/1))=0

A métrica que é usada para escolher o melhor corte deve avaliar o "quanto de desordem será reduzido com o novo segmento e como será a ponderação da desordem em cada segmento" (BERSON, 1997).

Para avaliar o quanto de desordem é reduzida através de um novo corte basta calcular a entropia a cada novo segmento. Calcula-se a entropia ponderada pela seguinte fórmula:

$$\text{entropia\_ponderada}(X) = \sum_{i=1}^{N} (p_i+n_i)/(p+n)\ \text{entropia}\ (p_i,n_i) \qquad (4.4)$$

onde N é o número de partições (segmentos) criados pelo corte

Como pode ser observado na tabela 4.2 (BERSON, 1997), se for analisada apenas a entropia média (aritmética simples) para cada novo segmento, o corte selecionado seria o corte *B*, dado que a sua média é 0.5 e a média do corte *A* é 0.72. No entanto, se, além disso, for ponderada a contribuição de cada novo segmento com o seu respectivo tamanho,[18] é obtida uma medida denominada de entropia ponderada (média ponderada).

TABELA 4.2 - PONDERAÇÃO DOS VALORES DA ENTROPIA PARA DOIS CORTES DISTINTOS

CORTES	CORTE A ESQUERDA	CORTE A DIREITA	ENTROPIA MÉDIA	ENTROPIA PONDERADA
Corte *A*	++++-	+----	0.72=(0.72+0.72)/2	0.72=(1/2*0.72) + (1/2)*0.72
Corte *B*	+++++----	-	0.50=(0.99+0)/2	0.89=(9/10)*0.99+ (1/10)*0.0

Comparando os valores apresentado na tabela 4.2, pode-se observar que, para o corte *A* não houve alteração entre a entropia média e a entropia ponderada. Porém o mesmo não aconteceu com o corte *B*, onde o valor encontrado para a entropia ponderada é maior do que o valor da entropia média.

Em resumo, existem duas opções de análise para escolher entre o corte *A* e o corte *B*. Se for analisada a entropia média, o corte *B* será selecionado. Se for analisada a entropia ponderada, o corte *A* será selecionado.

O algoritmo ID3 utiliza o método de seleção através da entropia ponderada.

---

[18] Este tamanho se refere ao número de exemplos contemplados pelo corte.

Espaço de Busca

Sendo o conhecimento representado por uma árvore de decisão, o espaço de busca é compreendido de todas as árvores que possam ser construídas sobre os exemplos do conjunto de treinamento. A construção ocorre em operações contínuas de transformação, pelas quais a árvore é estendida através da substituição de um nó folha por uma nova subárvore de profundidade 1(um).

Com todas estas árvores e subárvores compondo o espaço de busca é preciso definir a melhor estrutura de representação. Isso pode ser feito através da definição de uma função de qualidade que depende de dois elementos:

- da precisão na classificação;
- do tamanho da árvore.

As árvores simples que classificam corretamente todos os exemplos do conjunto de teste são as preferidas. Este critério da preferência pela simplicidade foi estabelecido a partir da versão mais contemporânea do princípio *Occam's razor* (*Prefer the simplest hypothesis that fits the data*),[19] um dos princípios fundamentais da teoria do aprendizado de máquina. Este princípio estabelece que a hipótese deve ser tão simples quanto possível para obter uma melhor taxa de erro. Afirma também que a taxa de erro tende a crescer na medida que o número de parâmetros que descrevem a hipótese também cresce.

Porém vários pesquisadores têm realizado experimentos que contradizem este princípio. Domingos (DOMINGOS, 1998) afirma estar crescendo o número de sistemas de aprendizado de máquina que obtém reduções na taxa de aprendizado ao construir modelos mais complexos. Esse autor também apresenta várias razões pelas quais a versão acima do princípio de *Occam* é essencialmente falso. (Note que simplicidade é desejável por si mesma, o que é falso é a afirmação de que uma hipótese mais simples tende a ser, em geral, mais correta.)

---

[19] O princípio conhecido por Occam's Razor foi inicialmente estabelecido pelo lógico medieval William of Occam em 1324. "It is vain to do with more what can be done with less ... Entities should not be multiplied beyond necessity".

Algoritmo de Busca

Como já foi visto anteriormente, o algoritmo ID3 usa a estratégia *top-down* que pesquisa apenas parte do espaço de busca na tentativa de garantir que uma árvore mais simples seja construída. Este processo ocorre conforme os procedimentos (proc) e condições (cond) descritas pela figura 4.3 (HOLSHEIMER, 1994):

FIGURA 4.3 - PROCEDIMENTOS PARA A CONSTRUÇÃO DA ÁRVORE DE DECISÃO

Proc 1	-	um atributo é selecionado como sendo o nó raiz da árvore e ramificações são criadas conforme o número de valores possíveis do domínio do atributo;
Proc 2	-	uma árvore é construída para classificar o conjunto de treinamento.
Cond 2.1	-	se todos os exemplos cobertos por um determinado nó folha pertencerem a uma mesma classe, este nó folha será tratado como um label de classe.
Cond 2.2	-	se todos os nós folhas forem tratados como label de classes, o algoritmo é encerrado;
Proc 3	-	caso contrário (Cond 2.2), o nó será tratado como sendo um nó atributo, novamente com ramificações criadas conforme o número de valores possíveis do domínio. O algoritmo retorna ao Proc 2.

Os procedimentos descritos na figura 4.3 criam uma árvore que classifica todos os exemplos do conjunto de treinamento corretamente. No entanto, a árvore criada não necessariamente é simples.

Considerando o espaço de busca do ID3, podem ser destacadas algumas habilidades e restrições (MITCHELL, 1997):

- o espaço de todas as árvores de decisão é um espaço completo de funções de valores discretos finitos relativos aos atributos disponíveis;
- ID3 mantém apenas uma única hipótese ao longo da construção da árvore de decisão. Desta forma, perde a capacidade de considerar todas as hipóteses consistentes. Assim, ele não tem a habilidade de determinar quantas árvores de decisão são consistentes em relação ao conjunto de treinamento;

- ID3 em sua forma original não realiza *backtracking* na sua busca. Uma vez selecionado um atributo para testar um determinado nível na árvore de decisão, ele não torna a considerá-lo para compor um outro teste;
- ID3 usa todos os exemplos a cada processo para definir uma nova hipótese. Isso o torna diferente de outros métodos que constróem árvores de decisão de forma incremental. Para a definição da hipótese, o ID3 usa testes estatísticos sobre todos os exemplos, o que determina a construção de um classificador menos sensível a erros, permitindo, assim, a manipulação de ruídos na base de treinamento.

Atributos Simbólicos

Um nó interno da árvore de decisão, ou seja, uma condição definida pelo atributo previsor, constitui um corte sobre o valor do atributo, com as respectivas ramificações para todos os valores possíveis no domínio dos atributos. Este processo é conveniente para o processamento sobre os atributos simbólicos.

Partição Baseada em Valores dos Atributos

Se o atributo X com um domínio $\{v_1, \ldots, v_N\}$ é usado como raiz da árvore de decisão, a árvore terá então N partições de T $\{T_1, \ldots, T_N\}$, onde T_i conterá aqueles exemplos em T que possuam o valor v_i de X. Dado que T_i contém p_i exemplos de PE e n_i exemplos de NE, a expectativa de informação requerida para a subárvore T_i é dada pela *entropia* (p_i,n_i) (WU, 1995).

A medida de ganho de informação – $G(X)$, obtida pela ramificação em X, é dada por:

$$G(X) = \text{entropia}\ (p,n) - \text{entropia\_ ponderada}\ (X) \qquad (4.5)$$

O ID3 examina todos os atributos previsores candidatos, escolhe o X que maximize $G(X)$ e constrói a árvore, repetindo o processo de forma recursiva para a construção das árvores de decisão com subconjuntos residuais $T_1, \ldots, T_N$.

Para cada T_i ($i = 1, ..., N$), tem-se que

- se todos os exemplos em T_i são positivos, é criado um nó folha "SIM" e interrompe;
- se todos os exemplos em T_i são negativos, é criado o nó folha "NÃO" e interrompe;
- caso contrário, seleciona outro atributo.

A construção da árvore é iniciada por um nó raiz vazio e o conjunto de todos os exemplos. O ID3 seleciona o atributo previsor que gera o maior valor para o ganho de informação (expressão 4.5), definindo-o, assim, como sendo o nó raiz. As ramificações são geradas de acordo com o número de valores possíveis para o respectivo atributo e o conjunto de treinamento é particionado em subconjuntos de acordo com essas ramificações geradas. Este mesmo processo é repetido para cada um dos subconjuntos até que todos os exemplos de um subconjunto pertençam a uma única classe (MITCHELL, 1997).

Ruído

O ruído pode interferir tanto na questão dos valores dos atributos previsores quanto na classe, aumentando a árvore de forma a acomodar os exemplos inconsistentes. Para superar este problema, é utilizada a medida ganho de informação.

No entanto, podem ocorrer situações onde esta medida não apenas elimine atributos irrelevantes, mas também elimine algum atributo relevante.

Uma forma alternativa para estes casos é o uso do teste X^2 (*chi-squared)*[20]. Se o atributo A é irrelevante para a classe de um exemplo em S, ou seja, o atributo A e a classe são estatisticamente independentes, o valor esperado p_i' de p_i (o número de elementos da classe P em S_i) deve ser

$$p' = p * ((p_i + n_i) / (p + n)) = p\ |S_i| / |S| \qquad (4.6)$$

e, se n_i' é o correspondente valor esperado de n_i, a estatística

$$\Sigma_{i=1}^{V} (p_i - p_i')^2 / p_i' + (n_i - n_i')^2 / n_i' \qquad (4.7)$$

---

[20] O teste X^2 é um teste de significância estatística desenvolvido por um estatístico inglês, Karl Pearson, em 1900. O X^2 é definido como sendo a soma dos quadrados das diferenças de freqüência entre o esperado e o obtido para os exemplos. O teste é uma medida da probabilidade pela qual uma associação seja possível, ou que uma diferença entre os exemplos seja possível.

é o valor aproximado de X^2 com v-1 graus de liberdade. O procedimento de construção da árvore é modificado de forma que apenas os atributos irrelevantes sejam eliminados. A aplicação deste teste resulta na poda da árvore produzida (HOLSHEIMER, 1994).

Ausência de Valores para os Atributos

Um problema que pode ocorrer nos exemplos que compõem o conjunto de treinamento é a ausência de valores para os atributos previsores.

Para a construção de árvores de decisão, vários métodos têm sido propostos a fim de tratar este problema. Por exemplo: a possibilidade de tratar os valores ausentes como sendo os valores com a maior freqüência nesta classe, ou simplesmente descartar o exemplo onde os valores estejam ausentes, ou tratá-los como tendo um valor especial que identifique a ausência.

Um dos melhores métodos propõe que os valores dos atributos ausentes sejam assumidos como sendo uma proporção da freqüência relativa dos correspondentes valores dos atributos em S (HOLSHEIMER, 1994).

Durante a fase de classificação, não é possível classificar de imediato os exemplos com valores de atributos previsores ausentes, na hipótese de haver uma ramificação específica para estes valores. Para viabilizar a classificação, o ID3 explora todas as ramificações para estes atributos, cria um caminho para cada ramificação e estima a probabilidade considerando a opção mais correta dos valores dos atributos. Esta probabilidade é dada pelo produto da freqüência relativa dos valores escolhidos para os atributos ausentes. Desta forma, para todas as classes, estas probabilidades são calculadas, sendo selecionada a classe com a maior probabilidade como o resultado da classificação. Este procedimento degrada a performance de execução onde a incidência dos valores desconhecidos seja alta.

Resumo

Durante a indução, no ID3 todos os possíveis atributos são testados e considerados quando do particionamento de um nó da árvore, sendo utilizada a entropia para melhor escolher o atributo previsor usado para realizar essa partição.

Dado que este algoritmo trabalha com atributos previsores discretos, o número de ramificações decorrentes de uma condição é determinado pelo número de valores possíveis no domínio deste atributo.

4.3.2 AQ

Da mesma forma que o ID3, o AQ foi inicialmente desenvolvido para construir regras consistentes, isto é, uma regra é construída para cada classe, sendo capaz de cobrir todos os exemplos positivos e de não cobrir os exemplos negativos.

Uma diferença entre o AQ e o ID3 é o fato do AQ permitir sobreposição de regras e o ID3 não, pois esse último cria partições mutuamente exaustivas e exclusivas.

O algoritmo AQ manipula exemplos incompletos e inconsistentes através de técnicas de pré e pós-processamento (HOLSHEIMER, 1994). Sem que a precisão da classificação seja afetada, o número e o tamanho das regras descobertas pelo AQ é reduzido, dada a aplicação da técnica de pós-processamento, chamada *rule truncating.*

Espaço de Busca

As regras são representadas na notação VL_1 (*Variable-value Logic system* 1), um cálculo de atributos com múltiplos valores lógicos *(a multiple-value logic attributional calculus)* de variáveis tipadas.

Um *selector* é definido para relacionar um atributo a um valor ou a uma disjunção de valores, como por exemplo 'cor = vermelho $\vee$ verde'. Uma conjunção de seletores é chamada um *complex.*[21] A parte condicional de uma regra de decisão, formada por uma disjunção de *complexes,* é chamada de *cover.*

Na construção de uma regra de decisão, o AQ executa uma heurística de busca através do espaço de todas as expressões que determinam quais expressões consideram todos os exemplos positivos e não consideram os exemplos negativos. Em função de existirem muitas destas regras, o AQ tem como objetivo encontrar a regra

---

[21] Um *complex* é uma descrição do conjunto.

"preferida". Este critério de preferência é definido pelo usuário a partir de uma reflexão sobre a necessidade do domínio de aplicação (ver exemplos deste critério no item seguinte).

Algoritmo de Busca

O AQ gera regras de decisão adicionando, a cada passo, o melhor *complex* ao *cover*. Cada passo inicia com a atenção focada em um exemplo positivo chamado *seed*. O algoritmo gera um conjunto de todos os *complexes*, chamado *star,* que cobre o *seed* e não cobre nenhum exemplo negativo e, então, seleciona o melhor *complex* a partir do *star* de acordo com o critério de preferência. Ao término deste processo, este *complex* selecionado é adicionado ao *cover*.

O algoritmo básico para o *cover* pode ser descrito na forma da figura 4.4 (HOLSHEIMER, 1994).

FIGURA 4.4 - ALGORITMO BÁSICO PARA A CONSTRUÇÃO DO *COVER*

Enquanto o *cover* parcial não cobre todos os exemplos positivos, faça:
Proc 1 - um *seed*, um exemplo positivo ainda não coberto, é selecionado;
Proc 2 - é gerado um *star*, ou seja, é determinado o número máximo de *complexes* que cobrem o *seed* e não cobrem exemplos negativos;
Proc 3 - seleciona o melhor *complex* ;
Proc 4 - adiciona o *complex* no *cover*.
Enquanto o *cover* parcial cobre exemplos negativos, faça:
Proc 1 - um exemplo negativo coberto é selecionado;
Proc 2 - é gerado um *star* parcial, ou seja, o conjunto máximo de seletores que cobrem o *seed* e excluem o exemplo negativo;
Proc 3 - gera um novo *star* parcial pela interseção do *star* parcial corrente com o *star* anteriormente gerado;
Proc 4 - organiza o *star* parcial mantendo o *maxstar* com os melhores *complexes*.

O procedimento de geração do *star* é executado de forma exaustiva, o que acarreta o crescimento muito rápido do espaço de busca para os *covers*. Entretanto, o usuário pode definir um parâmetro que controle quantas disjunções de *complexes*

devam ser armazenadas no *star* parcial. Apenas os melhores *complexes* devem ser armazenados, segundo um critério especificado pelo usuário, a exemplo dos seguintes: "maximize o número de exemplos positivos cobertos e o número de exemplos negativos excluídos" ou "minimize o número de *selectors*".

Desta forma, para um dado *star* parcial, o algoritmo processa a qualidade dos *complexes* na interseção, retém o *maxstar* destes e intersecta-os novamente até que nenhum dos exemplos negativos seja coberto.

Ao gerar um *complex,* este algoritmo realiza uma busca do melhor dos *complexes*. Se, por um lado, as especializações excluem alguns exemplos negativos que são cobertos a partir do *complex*, por outro, garantem que sejam considerados alguns *seeds* cobrindo exemplos positivos. Este processo ocorre de forma iterativa até que todos os exemplos negativos sejam excluídos. Desta forma, o algoritmo AQ pesquisa apenas em um espaço de complexes perfeitamente consistentes com o conjunto de treinamento.

O algoritmo AQ utiliza o procedimento *beam search*, o qual pode ser interpretado como várias pesquisas *hill-climbing* em paralelo (CLARK, 1989).

Dados Inconsistentes e Ruídos

Para tratar o problema de inconsistências (exemplos positivos e negativos idênticos), o AQ tem três opções (HOLSHEIMER, 1994):

- tratar os exemplos inconsistentes, sejam positivos ou negativos; ou
- simplesmente ignorá-los; ou
- pré-classificar estes exemplos inconsistentes de acordo com a probabilidade máxima, se estiver disponível a informação estatística sobre a probabilidade destes exemplos.

Também podem ocorrer situações onde o conjunto de treinamento contenha exemplos incorretos e não exista nenhuma vantagem na aplicação de tratamento probabilístico para a construção de regras. Neste caso, o teste de verificação se uma

instância satisfaz ou não a descrição ocorre através de outro método, como por exemplo, o método da combinação flexível, onde graus de similaridade são ou não satisfeitos na determinação de uma classe.

O uso da combinação flexível possibilita simplificar a descrição através da remoção de um ou mais *complexes*. Esta técnica, chamada de *truncation*, remove o *complex* que cobre o menor número de exemplos. Se um conjunto de treinamento é inconsistente, este tipo de *complex* pode ser um indicativo de erro nos dados. A regra resultante deixa de estar correta, mas probabilisticamente as similaridades para as diferentes classes estão contempladas. Além disso, a regra se torna mais compreensível e de menor tamanho.

A cada processo de *truncation,* a precisão de classificação é calculada. Diversas iterações produzem diferentes negociações entre a complexidade da descrição e a precisão da regra, até que seja encontrada a negociação que gere o melhor de todos os resultados obtidos.

Indução Construtiva

O algoritmo AQ usa o conhecimento sobre o domínio, expresso na forma de regras, para gerar novos atributos que auxiliem na produção de melhores regras. Estas regras podem ser de dois tipos (Holsheimer,1994):

- **lógicas** - que definem valores de novas variáveis; e
- **aritméticas** - que introduzem novas variáveis como funções de atributos numéricos.

As regras lógicas representam o *background* da definição das classes, as restrições das classes, a generalização de hierarquias e as dependências causais. As classes e as descrições "aprendidas" pelo algoritmo são adicionadas ao conjunto de regras.

Aprendizado Incremental

A partir de hipóteses de decisão, fornecidas inicialmente pelo usuário na forma de regras, o algoritmo implementa o método de aprendizado através do conceito *memória total,* onde são referenciados todos os exemplos tratados na etapa de formulação das

regras. Por outro lado, existe um outro método de aprendizado, oposto ao *memória total*, que utiliza o conceito *memória parcial*, onde as novas regras de decisão tem a garantia de estarem corretas em relação a todos os exemplos, quer sejam os exemplos utilizados na etapa de formulação das regras, quer sejam os exemplos testados (Holsheimer,1994).

Resumo

O algoritmo AQ constrói um classificador contendo uma regra para cada classe de forma a cobrir todos os exemplos positivos e nenhum exemplo negativo, permitindo a sobreposição de regras.

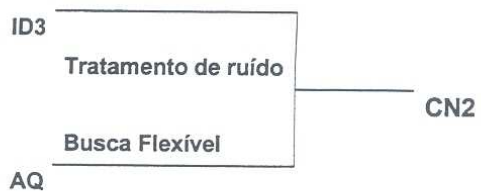
Este algoritmo trata o ruído e inconsistências através de técnicas de pré e pós-processamento, inclusive controlando o número e o tamanho das regras descobertas através de *rule-truncation*.

4.3.3 CN2

O algoritmo CN2 induz uma lista de decisão e foi desenvolvido por Clark e Niblet em 1989 e aperfeiçoado por Clark e Boswell em 1991 (FAYYAD, 1996), a partir dos sistemas ASSISTANT e AQR, ambos baseados nos algoritmos ID3 e AQ, respectivamente.

Do ID3, o CN2 herda a habilidade para o tratamento de ruído, usando técnicas de poda.

Já do AQ, o CN2 apropria a estratégia de busca flexível ao não gerar um conjunto de regras numa ordem específica. O CN2 também passa a incluir no espaço de busca as regras que não eram executadas perfeitamente sobre a base de treinamento.



O CN2 possibilita gerar o conjunto de regras de forma ordenada ou de forma não-ordenada. Isso tem vantagens e desvantagens, dentre as quais as seguintes:

- as regras geradas de forma não-ordenada requerem mecanismos adicionais para a administração de conflitos; e
- as regras ordenadas são mais difíceis de serem interpretadas, já que a interpretação de uma regra depende das regras anteriores na seqüência.

No processo de geração de regras, o algoritmo CN2 desenvolve dois procedimentos básicos:

- uma função que mede a qualidade de uma regra; e
- um algoritmo de controle da execução de busca repetitiva.

Espaço de Busca

O CN2 é projetado para induzir e medir a precisão de um conjunto de regras de classificação do tipo "se.. então..." em domínios onde podem ocorrer ruídos. Estas regras estão dispostas em lista e, embora sejam semelhantes às regras do AQ (se <condições> então <classe>), são diferentes, pois a parte condicional que apresentam é um *complex* e não uma disjunção de *complexes*.

Durante o processo de busca, os *complexes* são especializados pela adição de um *selector* conjuntivo ou pela remoção de um valor disjuntivo em um dos *selectors*. O algoritmo CN2 gera todas as possíveis especializações do conjunto de *complexes* (*star*) pela interseção deste com o conjunto de todos os possíveis *selectors*.

O critério de avaliação da qualidade dos *complexes* consiste de dois testes:

- determinar se o *complex* é preciso[22]; e
- avaliar a sua significância[23].

A precisão implica em encontrar o conjunto E de exemplos que são cobertos pelo *complex* e a probabilidade de distribuição $P = (p_1, p_2, \ldots, p_n)$ dos exemplos em E entre as n classes.

---

[22]Tem-se uma alta precisão sobre o conjunto de treinamento quando a predição da maioria da classe é coberta.

[23]Se a alta precisão sobre o conjunto de treinamento é ou não devida

Uma das alternativas que o CN2 apropria para o cálculo da precisão é a medida de entropia[24] (também utilizada pelo ID3), definida de acordo com a seguinte fórmula (CLARK, 1991):

$$entropia = -\sum_{i=1}^{n} \rho_i \log_2 (\rho_i) \qquad (4.8)$$

Esta medida avalia a qualidade dos *complexes*: quanto menor for a entropia melhor será a qualidade do *complex* (CLARK, 1989). A medida de entropia prefere os *complexes* que cobrem um grande número de exemplos de uma única classe e poucos exemplos de outras classes.

Para evitar a seleção de regras com alta especificidade, o CN2 usa o teste de significância. O *complex* deve ter um grau de significância tal que identifique uma real correlação entre os valores dos atributos previsores e as respectivas classes. O CN2 testa a significância das regras, através da seguinte fórmula de distância entre duas distribuições de probabilidade (HOLSHEIMER, 1994):

$$2 \sum_{i=1}^{n} f_i \log_2 (f_i / e_i) \qquad (4.9)$$

onde $F = (f_1, ..., f_n)$ é a distribuição de freqüência observada

$E = (e_1, ..., e_n)$ é a distribuição de freqüência esperada.

O menor dos escores, o que mais próximo estiver da regularidade, é adotado como escolha. Apenas os *complexes* que atendam ao mínimo de significância inicialmente definido pelo usuário são considerados para o processamento.

O teste de significância pode ser ainda utilizado para podar as regras mais especializadas da lista (menos freqüentemente aplicáveis), reduzindo, assim, a complexidade da lista sem grandes prejuízos de precisão e de predição.

---

[24]Esta medida da entropia também é utilizada pelo ID3 (CLARK, 1991).

Entretanto, mesmo que este teste elimine as regras que estejam abaixo de um certo nível de significância, persiste o problema de somente as regras que atendam a este teste sejam as preferidas, ao invés de outras mais gerais e mais confiáveis, porém, aparentemente menos precisas (*downward bias*).

Essas observações relativas à precisão e ao teste de significância das regras, podem ser melhor esclarecidas através do exemplo 4.1 (CLARK, 1991).

EXEMPLO 4.1 – EXEMPLO DE TESTE DE SIGNIFICÂNCIA

Um domínio qualquer contendo duas classes similares C_1 e C_2, e três regras R_1, R_2 e R_3 de forma que:

- R_1 cobre 1000 exemplos da classe C_1 e (1) um exemplo da C_2 [1000 , 1]
- R_2 cobre 5 exemplos de C_1 e 0 exemplos da classe C_2 [5 , 0]
- R_3 cobre [1 , 0]

Como pode ser observado no exemplo 4.1, o algoritmo deveria selecionar a regra R_1 dado que a sua precisão é a melhor - as regras R_2 e R_3 apenas cobrem poucos exemplos e a sua aparente precisão de 100% não reflete, necessariamente a performance sobre um novo teste. Entretanto, embora um teste com significância de 99% elimine R_3, R_2 será selecionado ao invés de R_1. Desta forma, deve existir um mecanismo que aceite a regra R_1 (a mais geral), mesmo possuindo este nível de significância.

Uma outra alternativa adotada pelo CN2, para medir a precisão, é a estimativa de precisão de Laplace que é dada pela fórmula (CLARK, 1991):

$$precisão = (n_c + 1) / (n_{tot} + k) \tag{4.10}$$

onde

k é o número de classes no domínio;

n_c é o número de exemplos cobertos pela regra e que têm a classe C predita pela regra;

n_{tot} é o número total de exemplos cobertos pela regra.

Considerando as mesmas informações do exemplo 4.1, a estimativa de precisão de Laplace para a predição de classes seria de 99,8% para R_1, 85,7% para R_2 e 66,6% para R_3.

Desta forma a regra R_1 seria considerada a mais precisa e, assim, a estimativa de precisão de Laplace evita o indesejável *downward bias* da medida de entropia, referida anteriormente (CLARK, 1991).

O comportamento do teste de significância com a estimativa de precisão de Laplace é qualitativamente diferente se comparado com a entropia. No caso da entropia, o aumento do nível de significância faz com que o CN2 selecione as regras mais gerais durante o processo de indução. No caso da estimativa de precisão de Laplace, a seleção tende a favorecer as regras mais gerais e o teste de significância altera o ponto de parada do CN2 na busca de mais regras. Assim o teste de significância é usado como uma forma de pré-poda (*pre-pruning*) das regras. Cabe ressaltar que, em geral, pós-poda (*post-pruning*) tende a apresentar melhores resultados que pré-poda, já que o primeiro tem acesso a mais informação (regra completa) para realizar a poda. (BREIMAN, 1984) e (QUINLAN, 1993).

Em resumo, com a entropia o teste de significância influi na decisão de quais regras são selecionadas como sendo as melhores, e com a estimativa de Laplace afeta o critério de término para o algoritmo (CLARK, 1991).

Algoritmo de Busca

Conforme já foi visto anteriormente, o CN2 permite gerar o conjunto de regras de forma ordenada ou de forma não-ordenada.

Na forma ordenada, o CN2 gera uma lista de decisão de forma iterativa, construindo uma regra a cada iteração. Durante a execução do algoritmo as regras encontradas são avaliadas, com o objetivo de selecionar a melhor delas. Uma regra é construída pela busca de um *complex* que cubra um grande número de exemplos de uma classe C_i e poucos exemplos de outra classe. Ao encontrar um bom *complex*, o algoritmo remove estes exemplos cobertos do conjunto de treinamento e adiciona a regra "se <*complex*> então prediz C_i" ao final da lista de decisão. Para o conjunto

restante uma nova regra é construída, até que mais nenhum *complex,* de qualidade suficiente, possa ser definido.

Para a classificação de um novo exemplo, cada regra da lista é testada até que uma destas regras seja satisfeita, atribuindo a respectiva classe como sendo a classe deste novo exemplo. Nesta situação, um exemplo é classificado por apenas uma regra na lista, não necessitando assim heurísticas para o tratamento de conflitos entre as regras.

Os procedimentos para a geração de uma lista ordenada de regras pelo CN2 podem ser observados na figura 4.5 (CLARK, 1991):

FIGURA 4.5 - PROCEDIMENTOS PARA GERAR UMA LISTA DE REGRAS ORDENADAS PELO CN2

```
Procedure CN2ordenado(exemplos, classes):
let lista_regras = [ ]
Repeat
Call FindBestCondition(exemplos) to find melhorcond
  If melhorcond is not null
  Then let classe = a classe mais comum dos exemplos cobertos pela melhorcond
       & add regra "if melhorcond then predict classe" para o final da lista _regras
       & remove de exemplos todos os exemplos cobertos pela melhorcond
until melhorcond is null
return lista_regras
```

Entretanto, o fato das regras serem ordenadas implica em regras menos compreensíveis, dado que para analisar uma regra em específico, é necessário avaliar as *n* regras antecedentes. Nos casos onde existam várias regras antecedentes, a dificuldade do especialista humano em entender o significado da regra se acentua.

Já no caso para a geração de regras não-ordenadas, o CN2 remove apenas os exemplos da classe em questão cobertos pela regra quando esta é criada.

Os procedimentos para a geração de um conjunto não-ordenado de regras pelo CN2 podem ser observados na figura 4.6 (CLARK, 1991):

FIGURA 4.6 - PROCEDIMENTOS PARA GERAR UMA LISTA DE REGRAS NÃO-ORDENADAS PELO CN2

```
Procedure CN2não-ordenado(exemplos, classes):
let lista_regras = [ ]
for cada classe no conjunto de classes
  gerar as regras através de CN2ForOneClass (todos os exemplos, classe)
  add regras no conjunto de regras
return conjunto de regras


procedure CN2ForOneCLass(exemplos,classe)
let regras = [ ]
Repeat
  Call FindBestCondition(exemplos, classe) to find melhorcond
  If melhorcond is not null
  Then add regra "if melhorcond then predict classe" para regras
        & remove de exemplos todos os exemplos da classe cobertos pela
        melhorcond
until melhorcond is null
return regras
```

De forma distinta ao processo de criação de lista ordenada de regras, na criação de conjunto não-ordenado os exemplos negativos permanecem no conjunto de exemplos, dado que cada regra deve, de forma independente, tratar os exemplos negativos. Os exemplos positivos continuam sendo retirados do conjunto de exemplos.

Para classificar um novo exemplo através do conjunto não-ordenado de regras, o CN2 procura por todas as regras cujas condições são satisfeitas pelo exemplo.

Nos casos que forem identificados conflitos (com a predição de mais de uma classe por regras diferentes) é calculada a distribuição de classes cobertas pelas regras e a agregação destas distribuições identifica a classe mais provável.

Resumo

O CN2 é um algoritmo baseado nas características dos algoritmos ID3 e AQ e é utilizado para construir, de forma simples e compreensível, listas de regras em domínios onde o ruído pode estar presente. Sua construção de regras é probabilística.

O CN2 gera listas de decisão de forma iterativa, construindo uma regra a cada iteração, onde uma regra é construída pela buscas de um *complex* que cubra um grande número de exemplos de uma classe C_i e poucos exemplos de outra classe.

O CN2 tem procedimentos específicos para a geração de regras ordenadas e não-ordenadas.

Para construir "boas" regras ordenadas, o CN2 avalia a precisão pela entropia e, para resolver o problema de alta especificidade das regras, usa o teste de significância. No caso de regras não-ordenadas, o CN2 utiliza a estimativa de precisão Laplaciana ao invés da entropia

O resultado produzido pelo algoritmo não é dependente da ordem na qual os exemplos são apresentados, como ocorre no AQ. O CN2 realiza a pesquisa em um grande espaço de busca, uma vez que considera todas as possibilidades de extensões de um *star*.

4.3.4 C4.5

O C4.5 é um algoritmo baseado no ID3 e também foi desenvolvido por Ross Quinlan (QUINLAN, 1993).

Implementações sobre o ID3

No desenvolvimento do C4.5 foram implementadas várias modificações sobre o ID3, tais como:

- poda (*postpruning*) em árvores de decisão e tratamento de ausência de valores de atributos;
- transformação de árvores de decisão em regras de produção;
- um novo critério de seleção de atributos para a construção da árvores de decisão;
- tratamento de atributos numéricos contínuos.

Poda em Árvores de Decisão

O algoritmo ID3 tende a gerar uma árvore de decisão exata, ou seja, classifica corretamente todos os exemplos que constam do conjunto de treinamento. Entretanto, na grande maioria das aplicações isto não é desejável dada a ocorrência de ruídos e de incertezas.

Assim sendo, no C4.5 foram desenvolvidos mecanismos de poda para tratar desta questão. O mecanismo de poda em árvores de decisão adotado pelo C4.5 é baseado na comparação das taxas de estimativa de erro[25] de cada subárvore e do nó folha. São processados sucessivos testes a partir do nó raiz da árvore, de forma que, se a estimativa de erro indicar que a árvore será mais precisa se os nós descendentes (filhos) de um determinado nó *n* forem eliminados, então estes nós descendentes serão eliminados e o nó *n* passará a ser o novo nó folha. Com essa poda, a árvore se torna mais precisa (de acordo com a estimativa de precisão adotada) a cada processo (WU, 1995).

Transformação de Árvores de Decisão em Regras de Produção

O ID3 constrói uma árvore de decisão, porém não tem a capacidade de transformá-la em regras de produção.

Para encontrar um conjunto de regras que dê cobertura (vide Apêndice B) ao conjunto de treinamento, o C4.5 extrai este conjunto de regras a partir de árvores de decisão previamente geradas.

---

[25] Pode-se definir a taxa de estimativa de erro da seguinte forma: se N observações são satisfeitas até determinado nó folha e E dentre estas N são classificadas de forma incorreta, então a taxa de estimativa de erro desta folha é E / N (BERRY, 1997).

O C4.5, por outro lado, é capaz de transformar uma árvore de decisão em regras de produção. A idéia básica é que em uma árvore de decisão quando um exemplo é classificado por um determinado nó folha, as condições que devem ser satisfeitas podem ser encontradas através do rastreamento de todos os cortes (nós internos) que se encontram no caminho compreendido entre o nó raiz e o nó folha em questão, podendo, desta forma, definir regras de produção. Assim, o C4.5 extrai as regras de árvores de decisão previamente construídas e o objetivo da transformação é produzir um conjunto não-ordenado de regras do tipo "se .... então .....".

Visando esclarecer as vantagens de transformar uma árvore de decisão em regras de produção, pode-se recorrer ao exemplo representado na figura 4.7 (QUINLAN, 1993):

FIGURA 4.7 - TRANSFORMAÇÃO DE ÁRVORES DE DECISÃO EM REGRAS DE PRODUÇÃO

```
F=0:
    J=0 : NÃO
    J=1:
      K=0: NÃO
      K=1: SIM
F=1:
    G=1:SIM
    G=0:
      J=0:NÃO
      J=1:
      K=0:NÃO
      K=1:SIM
```

Como pode ser observado, ao se analisar o caminho para atingir a folha SIM mais profunda,[26] é verificada a condição:

$$F = 1, G = 0, J = 1 \text{ e } K = 1$$

[26] A folha mais distante em relação ao nó raiz da árvore.

que pode transformada para a seguinte regra:

se F=1 e

G=0 e

J=1 e

K=1

então classe SIM.

Como conseqüência da transformação da árvore original em regras de produção, a condição *se* passa a ser uma regra dentre um conjunto de regras mutuamente exclusivas e exaustivas, e a ordem de apresentação das regras não interfere no resultado final.

Rescrever uma árvore de decisão como uma coleção de regras pode não resultar num produto mais simples do que a própria árvore, uma vez que existirá uma regra para cada nó folha.

Entretanto, é possível verificar se alguns dos nós antecedentes contêm condições irrelevantes e portanto podem ser podados.

Assim, pode-se observar no exemplo da Figura 4.7, que a classe SIM também pode ser obtida apenas considerando as condições baseadas nos atributos previsores J e K, a partir de uma composição que poda as condições relativas aos atributos F e G, estabelecendo a seguinte regra (QUINLAN, 1993):

se J=1 e

K=1

então classe SIM

No entanto, se isto pode ser feito, surge a seguinte questão: como decidir se uma condição pode ou não ser eliminada de uma regra? Esta decisão pode ser tomada avaliando a importância de uma condição X. A importância de uma condição X em uma regra A na definição de uma classe C pode ser identificada através de uma heurística *greedy*.

Este método avalia a possibilidade de cada condição ser eliminada através de uma matriz de contingência (SALZBERG, 1994), na forma em que é apresentada na tabela 4.3 (QUINLAN, 1993):

TABELA 4.3 - TABELA DE CONTINGÊNCIA

CONDIÇÃO	CLASSE C	OUTRAS CLASSES
Exemplos que satisfazem a condição *X*	Y_1	E_1
Exemplos que não satisfazem a condição *X*	Y_2	E_2

Onde:

- Y_1 é o número de exemplos que satisfazem à condição *X* e que pertencem à classe C
- E_1 é o número de exemplos que satisfazem à condição *X* e que pertencem às demais classes
- Y_2 é o número de exemplos que não satisfazem à condição *X* e que pertencem à classe C
- E_2 é o número de exemplos que não satisfazem à condição *X* e que pertencem às demais classes.

A condição X é mantida apenas nos casos onde os números apresentados na matriz de contingência sejam insatisfatórios (QUILAN, 1993).

Esta avaliação é feita com base na análise da taxa de erro. Para esta análise, o C4.5, de forma a considerar as diferentes contingências na eliminação da condição X, substitui a taxa de erro (E/N) pela taxa de erro relativa a um intervalo de confiança, dada por $U_{CF}(E,N)$. Esta taxa é definida em um nível de confiança especificado por CF.

Assim, a estimativa da taxa de erro das regras "que satisfazem a condição *X*" é dada por $U_{CF}(E_1, Y_1+E_1)$ e a das regras "que não satisfazem a condição *X*" é dada por $U_{CF}(E_1+E_2, Y_1+Y_2+E_1+E_2)$.

A necessidade da manutenção da condição *X* é verificada a partir da análise dos exemplos de treinamento para a construção da árvore de decisão: se a taxa estimada de erro da regra sem a condição *X* for maior que a taxa estimada com a condição *X*, então a condição *X* não deve ser eliminada (QUINLAN, 1993).

Após a conversão da árvore em regras de produção generalizadas, é possível chegar a um conjunto de regras que não seja mutuamente exaustivo e exclusivo e, desta forma, a uma situação onde uma observação não seja coberta por nenhuma regra ou cobertura por mais de uma regra. Esse ultimo caso pode ser resolvido simplesmente pela adoção de uma das regras, quando mais de uma for aplicável. Para a situação de inexistência de uma regra que se adapte a um exemplo, pode ser adotado o critério da classe *default* (por exemplo, a classe da maioria dos exemplos de treinamento).

O conjunto de regras é derivado apenas do exame da árvore de decisão e dos casos de treinamento a partir dos quais ela foi gerada. Uma vez criado o conjunto de regras a partir da árvore de decisão, o C4.5 agrupa as regras de cada classe e elimina aquelas que não necessariamente contribuem para a precisão do conjunto como um todo. Após a criação do conjunto de regras é identificada o valor da classe mais freqüente a ser atribuído como classe *default* (BERRY, 1997).

Os procedimentos para a geração de regras pelo C4.5 são os seguintes (KUFRIN, 1997):

- Poda *(postprunning)* – um conjunto inicial de regras é construído pela conversão de cada caminho do nó raiz da árvore até um nó folha, de forma que cada condição da regra corresponda a um nó interno de corte. A árvore de decisão com l folhas inicialmente será transformada em um conjunto de l regras. Cada uma das condições é examinada e aquela que não contribuir significativamente para a precisão da regra será retirada, resultando assim uma regra podada de caráter mais geral, desde que essa regra já não exista no conjunto de treinamento;
- Seleção – as regras já construídas e podadas compõem um conjunto para cada uma das k classes do conjunto de treinamento. Cada um destes conjuntos é avaliado para selecionar o subconjunto que maximize a precisão de predição para a classe à qual se associa;
- Ordenação – Os k conjuntos são ordenados de acordo com a freqüência dos erros falso positivos (exemplos cobertos pela regra, mas com classe

diferente da prevista pela regra). Então uma classe *default* é estabelecida para classificar possíveis exemplos do conjunto de treinamento que não estejam cobertos por nenhuma das regras do conjunto. A classe mais freqüente destes exemplos é a classe designada como sendo a classe *default*;

- Avaliação – O conjunto de regras como um todo é avaliado contra o conjunto de treinamento para determinar se alguma regra afeta negativamente a taxa de erro de classificação. Ao ser identificada alguma regra nestas condições, ela é removida do conjunto e a avaliação é repetida até que mais nenhuma regra recaia nesta condição.

O C4.5 mostra o resultado da avaliação de cada regra construída durante a fase de treinamento, constando (QUINLAN, 1993):

- a identificação da regra;
- o tamanho (número de condições que a compõem);
- a taxa de erro de predição;
- o número de vezes que a mesma foi utilizada em classificações durante o treinamento dos exemplos;
- o número de erros encontrados no conjunto de treinamento;
- o que poderia ocorrer se esta regra fosse omitida da lista. Isso é mostrado pelo C4.5 no formato a (b | c), onde b representa o número de casos classificados corretamente em função da existência desta regra, c representa o número de casos que, ao não serem classificados pela omissão desta regra, poderiam ser classificados por regras subsequentes ou pela opção *default* e a é igual a $b - c$;
- finalizando este sumário de performance de cada regra, é apresentado um quadro onde constam os casos que não foram classificados.

Um Novo Critério para a Construção de Árvores de Decisão

O ID3 trata de forma insatisfatória a existência de atributos com um número bastante variado de possíveis valores discretos, gerando árvores de decisão com alto grau de ramificação.

Isso pode ser verificado no caso de um simples diagnóstico na área médica. Um dos atributos é a identificação do paciente. No entanto, o ID3, ao não considerar a irrelevância da unicidade deste atributo, criará um número grande de subconjuntos, cada um com apenas um exemplo. Porém, tal divisão não teria uso em um processo de classificação (BERRY, 1997).

O C4.5 supera esta característica do ID3 através de uma outra forma de avaliar o ganho de informação. Ao invés da medida $G(X)$, dada pela fórmula 4.5, o C4.5 usa a *gain ratio* (taxa de ganho total de informação).

Esta abordagem, *gain ratio,* é definida pela seguinte fórmula (Quinlan, 1993):

$$gain\ ratio(X) = G(X) / split\ info(X) \tag{4.11}$$

ou

$$gain\ ratio(X) = gain(X) / split\ info(X) \tag{4.12}$$

onde

$$gain(X) = info(T) - info_x(T) \tag{4.13}$$

$$info_x(T) = \sum_{i=1}^{n} |T_i| / |T| * info(T_i) \tag{4.14}$$

$$split\ info(X) = - \sum_{i=1}^{n} |T_i| / |T| * log_2 (|T_i| / |T|) \tag{4.15}$$

$info_x(T)$ é a informação esperada da árvore tendo T como raiz

split info(X) é a informação esperada para distinguir as classes

A medida *split info(X)* representa uma informação em potencial pelo fato de particionar T em n subconjuntos, podendo, assim, avaliar como as classes estão distribuídas em um determinado nó.

O critério *gain ratio* expressa a proporção de informação gerada pelo corte que parece ser útil para o processo de classificação. Se o corte for trivial, o valor *split info* será muito pequeno e o *gain ratio* será instável. Para evitar esta situação, o critério *gain ratio* seleciona um corte que maximize o seu próprio valor, sujeito à restrição que o corte escolhido tenha um ganho de informação pelo menos maior que a média de ganho de informação sobre todos os cortes avaliados (QUINLAN, 1993).

O fato de incluir o critério *gain ratio* não substitui completamente a medida de entropia, pois em muitos casos este novo critério não melhora expressivamente a árvore produzida.

Tratamento de Atributos Contínuos

Uma extensão do ID3 para tratar os atributos contínuos, a partir da medida ganho de informação, identifica o limite de corte com o maior grau de informação. Segundo U. M. Fayyad & K. B. Irani, para a obtenção do valor máximo para o ganho de informação é sempre considerado um ponto de corte, dito *midpoint*, entre os valores de dois exemplos de diferentes classes (WU, 1995)[27].

Segundo J. Catlet (WU, 1995)[28] cada $x = (x_i + x_{i+1}) /2$, onde $i = 1 , ... , n-1$, é um possível ponto de corte se x_i e x_{i+1} forem exemplos de classes distintas. Todos estes pontos são testados, usando o ganho de informação, para identificar o melhor ponto de corte. Esta verificação de cada ponto é executada recursivamente para a ramificação esquerda e direita de cada nó com o objetivo de promover mais cortes. Este processo pode incorrer em um número muito grande de intervalos, caso os atributos não sejam suficientemente significativos.

Assim sendo, a interrupção deste processo recursivo de cortes, é dado por:

- interromper se o valor do ganho de informação sobre todos os cortes for o mesmo;
- interromper se o número de exemplos cobertos pelo corte for inferior a determinado padrão previamente estabelecido;
- limitar o número de intervalos de acordo com um número previamente estabelecido.

Já o C4.5 adota a seguinte abordagem para o cálculo do ganho de informação (Wu, 1995):

---

[27] página 85.

[28] página 85.

- cada ponto de corte possível não é o *midpoint* entre os dois valores mais próximos, e sim o maior valor em todo o conjunto de treinamento, desde que não exceda o *midpoint*. Este fato assegura que todos os valores limites ocorram no conjunto de treinamento. Assim sendo, cada valor limite não necessariamente é o menor valor entre dois valores vizinhos (próximos) porque todos os valores do conjunto de treinamento são avaliados;
- adota a *gain ratio* ao invés da ganho de informação;
- "binariza" os atributos de valores contínuos.

O ID3 não binariza os atributos numéricos. Já o C4.5 permite cortes com inequações binárias sobre os atributos numéricos, tais como $A_i <= N$ ou $A_i > N$, tratando-as como duas ramificações para o mesmo nó interno.

O *gain ratio* para tal situação é obtido da seguinte forma (WU, 1995):

- os exemplos são primeiramente ordenados pelo atributo que está sendo considerado;
- existe um número finito destes valores $\{v_1, ... , v_n\}$;
- neste conjunto, existem n-1 ramificações possíveis. Pode parecer oneroso examinar todas as n-1 repartições. No entanto, como os exemplos são ordenados, isto pode ser executado num só passo, atualizando a distribuição de classes da esquerda para a direita e do princípio para o final;
- para cada determinação de corte possível dado por $((v_i + v_{i+1}) / 2)$, o ganho de informação é processado e usado para a seleção do próximo corte.
- é pesquisado o corte possível que irá proporcionar a melhor entropia e o *gain ratio*.

Windowing

Tendo em vista o aumento rápido e acentuado da capacidade de processamento e armazenamento dos computadores atuais, surge a pergunta: por que então manter esta estratégia do *windowing*, que existe no ID3, também no C4.5? Quinlan responde a esta questão afirmando que, para certas situações, este recurso

pode aumentar a velocidade na construção da árvore de decisão, bem como produzir árvores mais precisas(QUINLAN, 1993).

O aumento da velocidade ocorre desde que o conjunto de treinamento não contenha um alto nível de inconsistência ou ruído, pois caso contrário pode tornar o processo ainda mais lento, requerendo um grande número de ciclos até que a árvore produzida atinja um nível de erro aceitável.

Quanto a questão da precisão, o segmento inicial que deflagra o processo do *window* é determinado de forma aleatória, ou seja, a árvore de decisão construída pode ser diferente todas as vezes que o processamento for reiniciado. Esta característica de geração de múltiplas árvores de decisão possibilita que seja selecionada a árvore de decisão com a menor taxa de erro ao invés da criação de um único classificador(QUINLAN, 1993).

Resumo

O algoritmo C4.5 é um algoritmo baseado no algoritmos ID3.

As mais importantes melhorias do C4.5 sobre o ID3 são as seguintes:

- a heurística de *gain ratio* para os testes de seleção, o que evita a tendência do ID3 de gerar árvores de decisão com alto grau de ramificação;
- mecanismo de *postpruning* em árvores de decisão, que se baseia na comparação das taxas de estimativa de erro de cada subárvore e do nó folha;
- a transformação opcional de árvores de decisão em regras de produção, pois parte do princípio que quando um exemplo é classificado por um nó folha, as condições que devem ser satisfeitas podem ser identificadas pelo rastreamento de todos os nós internos desde a raiz até este nó folha.

4.3.5 C5.0

O algoritmo C5.0 resulta de melhorias realizadas sobre o seu antecessor, o algoritmo C4.5, e foi também desenvolvido por Ross Quinlan (QUINLAN, 1997).

Da mesma forma que o C4.5, o algoritmo C5.0 aceita atributos discretos (assumindo um pequeno conjunto de valores possíveis) ou contínuos (valores numéricos).

Estes algoritmos fornecem classificadores na forma de árvore de decisão e de um conjunto de regras de produção como resultado do processo de transformação dessas árvores. Eles incluem métodos de generalização de regras, onde são removidas as condições julgadas irrelevantes e/ou redundantes, ao contrário de outros métodos que geram as regras a partir do simples acúmulo de todos os cortes entre o nó raiz e o nó folha correspondente.

As principais melhorias realizadas do C5.0 sobre o C4.5 são (QUINLAN, 1997):

- o uso da técnica *boosting;*
- o tratamento de diferentes custos para os diferentes tipos de erros na classificação;
- o suporte para *cross-validation.*

Boosting

Recentemente tem crescido o interesse por algoritmos de aprendizado que obtenham uma maior precisão através da implementação de *voting methods*, como por exemplo, o método *boosting,* desenvolvido por Robert Schapire e Yoav Freund (SCHAPIRE, 1997).

Com o objetivo de melhorar a precisão do algoritmo, a incorporação do *boosting* fundamenta-se na geração de várias classificações ao invés de somente uma. Os algoritmos *boosting* operam ponderando e reponderando os exemplos do conjunto de treinamento e re-executando os algoritmos de aprendizado sobre estes exemplos reponderados. A técnica de *boosting* força que o algoritmo de aprendizado se concentre sobre os exemplos, de forma que a combinação final de hipóteses seja um voto ponderado sobre as hipóteses mais "fracas". No C5.0, quando um novo exemplo está para ser classificado, cada classificação vota pela sua classe predita e os votos são contabilizados para determinar a classe.

Os procedimentos adotados por este algoritmo para gerar as várias classificações a partir de um conjunto de dados podem ser observados na figura 4.8:

FIGURA 4.8 - PROCEDIMENTOS PARA GERAÇÃO DE VÁRIOS CLASSIFICAÇÕES

- Proc. 1 - construir uma única árvore de decisão ou um conjunto de regras de produção, da mesma forma que os algoritmos apresentados anteriormente;
- Proc. 2 - gerar uma segunda classificação de forma a tratar os N exemplos que causaram erro na primeira classificação, a fim de que estes N exemplos não gerem erro;
- Proc. 3 - gerar uma terceira classificação de forma a tratar os N' exemplos que causaram erro na segunda classificação, a fim de que estes N' exemplos não gerem erro;
- Repetir o Proc. 3 até que um número pré-determinado de ciclos seja executado.

Geralmente, a construção de um maior número de classificações demanda um tempo maior de processamento que a geração de uma única classificação. Porém, segundo Quinlan (QUINLAN, 1997), a adaptação de 10 classificações pelo método *boosting* reduz a margem de erro em 25%.

A performance da classificação construída em cada ciclo de execução do C5.0, executado com a opção *boosting,* é sumarizada em uma tabela. Conforme pode ser observado na tabela 4.4, as linhas representam os diversos ciclos e nas colunas são apresentados os dados referentes ao tamanho da árvore de decisão (o número de nós folhas) e a sua respectiva taxa de erro. A última linha da tabela indica o valor da taxa média de erro.

TABELA 4.4- EXEMPLO DE SAÍDA PARA A EXECUÇÃO DO C5.0 COM OPÇÃO DE *BOOSTING*

```
Evaluation on test data (129 cases):

Trial                Decision Tree
-----                ----------------
                     Size      Errors

  0                  69        54(41.9%)
  1                  41        48(37.2%)
  2                  70        65(50.4%)
  3                  63        62(48.1%)
  4                  47        55(42.6%)
  5                  68        61(47.3%)
  6                  67        67(51.9%)
  7                  50        77(59.7%)
  8                  66        61(47.3%)
  9                  72        58(45.0%)
boost                          44(34.1%)          <<
```

Diferentes Custos para o Tratamento dos Erros de Classificação

Os algoritmos ID3 e C4.5 tentam minimizar o número de classificações erradas, mas tratam todas as ocorrências de erro com o mesmo peso, ou seja, consideram somente o número de ocorrências de erro, ignorando que existe uma diferença relativa nos custos destes erros. Por exemplo, o custo de aplicar dinheiro em uma "má" alternativa é maior do que não aplicar dinheiro em uma "boa" aplicação (ROBERTS, 1995).

O algoritmo C5.0 permite atribuir custos para os casos onde não haja classificação correta de um exemplo, sendo este custo dependente de uma valoração previamente estabelecida, da seguinte forma: classe1, classe2 : 5.5, o que significa que o custo do erro em classificar uma observação da classe1 como sendo da classe2 "vale" 5.5 unidades, ou seja, este tipo de erro é 5.5 vezes mais "oneroso".

Suporte para *Cross-Validation*

O C5.0 disponibiliza a obtenção de uma taxa média de erro a partir da realização de vários testes sobre a mesma base de treinamento.

Para isso, os dados são separados aleatoriamente em n subconjuntos de exemplos, que mantêm a mesma distribuição de classes do conjunto original. Para

cada subconjunto é construído um classificador usando *n-1* conjuntos para treinamento e o subconjunto restantes, que não foi visto durante a fase de treinamento para teste. Desta forma, cada exemplo é utilizado uma única vez para teste e n-1 vezes para treinamento. A taxa de erro, a partir de todos os exemplos, é estimada pela divisão do número de todos os erros ocorridos nas fases de teste pelo número total de exemplos.

Os resultados obtidos a partir da *cross-validation* dependem da forma aleatória na qual o conjunto original de treinamento seja particionado em subconjuntos. Esta partição pode mudar a cada processo, possibilitando, assim, que, a cada execução, ocorra uma taxa média de erro distinta para um mesmo conjunto original de exemplos.

Por exemplo, se a uma aplicação for atribuído o fator de particionamento de 5 subconjuntos para a opção de *cross-validation* e, para cada classificação seja apropriado um fator de 10 para a opção de *boosting*, ao todo são geradas 50 árvores de decisão para o cálculo da estimativa de erro da aplicação (QUINLAN, 1997). Esta afirmação é avaliada e comentada no capítulo 6.

Segundo Quinlan, a taxa média de erro produzida a partir do método de *croos-validation* é significativamente mais confiável se comparada a uma taxa gerada a partir de uma única classificação.

A performance da classificação construída a partir da execução do C5.0, com a opção *cross-validation,* também é sumarizada em uma tabela. Conforme pode ser observado na tabela 4.5, as linhas representam os diversos ciclos e nas colunas são os dados referentes ao tamanho da árvore de decisão (o número de nós folhas), pois e a sua respectiva taxa de erro. A penúltima linha da tabela indica o valor da taxa média de erro e a última mostra o valor SE (*the standard errors of the means)* que constitui uma estimativa de variabilidade destes resultados.

TABELA 4.5 - EXEMPLO DE SAÍDA PARA A EXECUÇÃO DO C5.0 COM OPÇÃO DE *CROSS-VALIDATION*

Fold	Decision Tree	
	Size	Errors
0	72.0	32.8%
1	73.0	40.6%
2	61.0	45.3%
3	76.0	51.6%
4	75.0	40.6%
5	52.0	38.5%
6	73.0	41.5%
7	79.0	30.8%
8	67.0	36.9%
9	83.0	38.5%
Mean	71.1	39.7%
SE	2.9	1.9%

A performance da classificação construída a partir da execução do C5.0, com as opções *cross-validation* e *boosting,* é sumarizada em uma tabela com algumas diferenças, conforme pode ser observado na Tabela 4.6, as linhas representam os diversos ciclos e nas colunas não são mais apresentados os dados referentes ao tamanho da árvore de decisão, pois os classificadores não são mais uma única árvore, e a sua respectiva taxa de erro.

TABELA 4.6 - EXEMPLO DE SAÍDA PARA A EXECUÇÃO DO C5.0 COM OPÇÃO DE *CROSS-VALIDATION* E *BOOSTING*

Xval	Decision Tree	
	Size	Errors
0	*	23.5%
1	*	23.0%
2	*	24.8%
3	*	23.4%
4	*	24.3%
5	*	23.8%
6	*	24.2%
7	*	23.3%
8	*	22.5%
9	*	25.4%
Mean		23.8%
SE		0.3%

Resumo

O algoritmo C5.0 é um algoritmo que constitui uma extensão do algoritmo C4.5 implementando a técnica de *boosting*, diferentes custos para os casos de erros na classificação e suporte para *cross-validation*.

4.3.6 Conclusão

Os três algoritmos descritos neste capítulo e que constituem o objeto de testes deste trabalho (CN2, C4.5 e C5.0) são algoritmos que funcionam de forma *top-down*.

O CN2 constrói listas de decisão. Já os algoritmos C4.5 e C5.0 controem árvores de decisão e regras de produção.

Estes algoritmos são chamados de *partitioning classifiers*, pois particionam o conjunto de exemplos em partições. O ID3, o C4.5 e o C5.0 constróem árvores de decisão a partir de particionamentos recursivos do conjunto de treinamento e o AQ e CN2 usam regras disjuntivas para o particionamento, porém de forma não recursiva (BUNTINE, 1992).

5 METODOLOGIA ADOTADA NO PROJETO EXPERIMENTAL

5.1 CONSIDERAÇÕES INICIAIS

Conforme já foi colocado anteriormente, este trabalho tem dois objetivos principais. O primeiro é comparar a eficácia de 3 (três) algoritmos para aquisição de conhecimento baseados no paradigma de indução de regras, a saber CN2, C4.5 e C5.0, aplicados sobre dois tipos de bases diferentes: bases padrão da literatura e uma base de dados do mundo real.

Um outro objetivo deste trabalho é propor e testar, no projeto experimental, uma solução para o problema de desbalanceamento entre as classes, muitas vezes encontrado em bases do mundo real.

Neste capítulo são apresentadas as bases de dados sobre as quais são realizados os experimentos, o método pelo qual são comparados os resultados obtidos e a descrição da abordagem proposta para resolver a questão do desbalanceamento das classes, a qual foi tratada em uma base de dados do mundo real.

5.2 BASES DE DADOS

Neste trabalho, para testar os "classificadores" gerados pelo CN2, C4.5 e C5.0, são utilizadas as seguintes bases de dados:

- BDPL : Bases de dados padrão da literatura;
- BDR: Base de dados do Censo Demográfico relativa aos municípios do Paraná (IBGE - 1991) – mundo real;

5.2.1 BDPL - Bases de Dados Padrão da Literatura

A identificação, a fonte, o número de observações (exemplos), número de atributos previsores, o número de classes e a existência ou não de valores de atributos previsores ausentes das bases de dados padrão da literatura a serem utilizadas neste trabalho podem ser observadas no quadro 5.1.

5.2.2 BDR - Base de dados do Censo Demográfico relativa aos Municípios do Paraná

Esta base real é constituída de dados oriundos do Censo Demográfico de 1991 (IBGE - 1991) e do Sistema de Indicadores Analíticos do IPARDES – Instituto Paranaense de Desenvolvimento Econômico e Social (IPARDES, 1993).

Esta base disponibiliza os atributos previsores (variáveis) referentes aos municípios paranaenses, segundo a condição de domicílio (urbano e rural).

QUADRO 5.1 - RELAÇÃO DE BASES PADRÃO DA LITERATURA

IDENTIFICAÇÃO	FONTE	OBSERV.	N. ATRIB.	CLASSES	VALOR AUSENTE
Allbp	New South Wales Institute	2800	29	3	Não
Balance	Balance Scale Weight & Distance database	625	4	5	Não
Bupa	Bupa Medical Research Ltd.	345	6	2	Não
Crx	Credit Approval	690	15	2	Sim
Dis	New South Wales Institute	2 800	29	2	Não
Glass	Central research Establishment Home Office Forensic Science Service	214	9	7	Não
House	Congressional Quarterly Almanac 98th Congress	435	16	2	Sim
Imports	1985 Model Import Car and Truck Specifications	205	25	7	Sim
Ionos	Johns Hopkins University Ionosphere database	351	34	2	Não
Iris	Iris Plants Database	150	3	4	Não
Seg	Image Segmentation data University of Massachusetts	210	19	7	Não
Sick	New South Wales Institute	2 800	29	2	Não
Soybean	Large Soybean database	307	35	19	Sim
Yeast	Institute of Molecular and Cellular Biology - Japan	1 484	8	10	Não
Wine	An Extendible Package for Data Exploration, Classification and Correlation. Institute of Pharmaceutical	178	12	3	Não

FONTE: http://www.ics.uci.edu/~llearn/MLRepository.html

Assim, como em 1991 existiam no Paraná 323 municípios, essa base é composta por 645 observações. Para cada município, tem-se duas observações referentes a condição de domicílio (urbano e rural). No entanto, o município de Curitiba participa com apenas uma observação, dado que o mesmo não possui área rural.

Cada observação é composta por 16 atributos, conforme listados adiante, e a respectiva classe. Os 16 atributos são oriundos de variáveis do Censo Demográfico de 1991, dos quais 14 são contínuos e dois discretos.

Classes

O atributo meta (classe) é obtido a partir do Sistema de Indicadores Analíticos, desenvolvido pelo IPARDES. Este Sistema classifica os municípios paranaenses segundo uma graduação discreta de suas condições relativas de qualidade de vida do morador e do domicílio (boa, razoável, crítica e muito crítica) referentes a uma série de variáveis obtidas a partir de diversas fontes, a seguir discriminadas. A partir dos indicadores referentes a cada uma dessas variáveis, é elaborado um indicador síntese para cada município, por condição urbana e rural, tornando-se possível discriminá-los em classes.

Este Sistema pressupõe a disponibilidade de informação para todos os municípios do Estado e vem sendo usado correntemente para a definição de prioridades visando a orientação de ações dos setores público ou privado.

O processo para a obtenção destes indicadores decorre de várias etapas de trabalho manual, desenvolvidas pelos pesquisadores do IPARDES, assim descritas:

a) os dados referentes a cada variável, como, por exemplo, saúde, são selecionados e devidamente coletados;
b) esses dados são inicialmente avaliados com o objetivo de verificar se permitem ou não que os municípios sejam analisados comparativamente;
c) caso o resultado do item (b) seja afirmativo, são estabelecidos cortes analíticos referentes a cada variável. Visando o enquadramento do município segundo a sua posição relativa, esses cortes são definidos de

acordo com parâmetros que possam classificar os valores de cada variável segundo os critérios bom, razoável, crítico e muito crítico. Esses critérios, por sua vez, têm por base padrões preestabelecidos de avaliação qualitativa ou quantitativa do atributo ou área a qual a variável se refere. E tais padrões são estabelecidos a partir de referências internacionais relativas à avaliação que possa ser feita de cada variável ou segundo conceitos de análise estabelecidos pelo analista. Como exemplo, pode-se tomar a variável coeficiente de mortalidade infantil, definido com o número de óbitos de menores de 1 ano de idade por mil nascidos vivos, cujos cortes analíticos são os seguintes: coeficiente preconizado pela Organização Mundial de Saúde-OMS, coeficiente médio do Estado do Paraná e coeficiente médio do Brasil;

d) os cortes analíticos, assim definidos, são hierarquizados segundo os critérios bom, razoável, crítico e muito crítico, logicamente de acordo com a positividade ou negatividade de avaliação a que se referem, construindo desta forma uma escala de indicadores ou parâmetros de enquadramento de cada município, apresentada sob a forma de quatro classes hierárquicas, 4, 3, 2 e 1. A escala de valor 4 representa a melhor posição qualitativa e quantitativa da variável de referência e o valor 1 representa a pior posição. De acordo com o exemplo anterior, a posição 4 se refere a coeficientes de valores superiores aos estabelecidos pela OMS; a posição 3 tem como limite superior o valor do coeficiente preconizado pela OMS e como limite inferior o valor médio desse coeficiente correspondente ao Estado do Paraná; a posição 2 tem como extremos esse valor e o valor do coeficiente relativo à média verificada para o Brasil; e a posição 1, valores inferiores a essa média;

e) o processo envolvendo os itens (a), (b), (c) e (d) são repetidos várias vezes de acordo com os atributos e áreas selecionadas;

f) no momento em que vários indicadores para uma mesma área ou atributo estejam estabelecidos, torna-se possível construir um indicador síntese que é o resultado da combinação dos *n* indicadores disponíveis para a área ou atributo. O indicador síntese também classifica os municípios em quatro classes hierárquicas, 4, 3, 2 e 1. No caso dos experimentos descritos neste trabalho, a partir da Base de Dados Real, está sendo utilizado um indicador síntese que é o resultado da combinação dos indicadores individuais relativos às variáveis condições do domicílio, condições do morador e condições de saneamento (IPARDES, 1993).

Atributos Previsores

Os atributos previsores selecionados a partir do Censo Demográfico seguiu orientação dos técnicos do IPARDES responsáveis pelo Sistema de Indicadores Analíticos, e podem ser descritos conforme a seguir:

- PA013_A009: participação percentual do número de casas em aglomerados subnormais[29] sobre o número total de domicílios.[30]
- PA017_A009: participação percentual do número de apartamentos em aglomerados subnormais sobre o número total de domicílios.
- PA020_A009: participação percentual do número de casas com abastecimentos de água por rede geral[31] sobre o número total de domicílios.
- PA023_A009: participação percentual do número de casas com abastecimentos de água sem canalização interna sobre o número total de domicílios.

---

[29]Aglomerado subnormal é um conjunto constituído por unidades habitacionais (barracos, casas,...), ocupando ou tendo ocupado, até o período recente, terrenos de propriedade alheia (pública ou particular), dispostos, em geral, de forma desordenada e densa, e carentes, em sua maioria, de serviços públicos essenciais. O que caracteriza um aglomerado subnormal é a ocupação desordenada e a inexistência de posse legal da terra ou título de propriedade, quando da sua implantação.

[30]Considerou-se como domicílio a moradia estruturalmente independente, constituída por um ou mais cômodos, com entrada privativa. Por extensão, edifícios em construção, embarcações, veículos, barracas, tendas e outros locais que estavam, na data do Censo, servindo de moradia, também foram considerados como domicílios.

[31]Rede geral – quando o domicílio é servido de água proveniente de uma rede geral de abastecimento.

- PA024_A009: participação percentual do número de apartamentos com abastecimentos de água por rede geral sobre o número total de domicílios.
- PA028_A009: participação percentual do número de casas com instalação sanitária só no domicílios ligados a rede geral rede geral sobre o número total de domicílios.
- PA046_A009: participação percentual do número de domicílios com condição de ocupação próprio sobre o número total de domicílios
- PA054_A009: participação percentual do número de domicílios com lixo coletado sobre o número total de domicílios.
- PA067_A009: participação percentual do número de domicílios particulares improvisados[32] sobre o número total de domicílios.
- PA071_A202: participação percentual do número de moradores em domicílios particulares improvisados sobre a população total no domicílio.[33]
- PA074_A202: participação percentual do número total de chefes em domicílios particulares permanentes sobre a população total no domicílio.
- PA081_A202: participação percentual do número total de chefes[34] com renda[35] até ½ salário mínimo (SM) sobre a população total no domicílio.
- PA082_A202: participação percentual do número total de chefes com renda entre ½ e 1 SM sobre a população total no domicílio.

---

[32]Classificou-se o domicílio particular em: permanente, assim considerado o construído para fim residencial; e improvisado, o que não atendia à referida condição, embora servisse de moradia na data do Censo, tal como o localizado em unidade (loja, fábrica, etc.) que não possuísse dependências destinadas exclusivamente à moradia, prédios em construção servindo de moradia a pessoal de obra, embarcações, carroças, vagões de estrada de ferro, tendas, barracas, grutas, etc.

[33]A população foi constituída pelos moradores habituais no domicílio, ou seja, as pessoas que tinham o domicílio como local de residência habitual, quer estivesses presentes ou ausentes na data de referência. As pessoas moradoras habituais do domicílio que estavam ausentes na data de referência foram recenseadas, desde que sua ausência não tenha sido superior a 12 meses em relação àquela data.

[34]Considerou-se como chefe do domicílio a pessoa, homem ou mulher, responsável pelo domicílio.

[35]A investigação do rendimento nominal médio mensal limitou-se aos chefes dos domicílios. Considerou-se como rendimento nominal médio mensal a soma de todos os rendimentos do chefe do domicílio: o rendimento bruto do mês de agosto de 1991 da ocupação habitual; o rendimento bruto do mês de agosto de 1991 de outras ocupações; o rendimento bruto do mês de agosto de 1991 proveniente de aposentadoria e/ou pensão; e o rendimento bruto do mês de agosto de 1991 ou a média dos últimos 12 meses, corrigida monetariamente, para outros rendimentos que não se enquadrasse em nenhuma das categorias relacionadas anteriormente.

- PA083_A202: participação percentual do número total de chefes com renda entre 1 e 2 SM sobre a população total no domicílio.
- Situação: situação do domicílio[36] (1 urbano, 2 rural).
- Categoria: faixas de tamanho dos centros urbanos (ANDRADE, 1998).

  1 - < 5.000

  2 - Entre 5.000 e 10.000

  3 - Entre 10.000 e 20.000

  4 - Entre 20.000 e 50.000

  5 - Entre 50.000 e 100.000

  6 - Entre 100.000 e 250.000

  7 - > 250.000

As faixas de tamanho de municípios foram definidas segundo a dimensão de seus respectivos centros urbanos, ou seja, sua população urbana. Este critério se justifica na medida em que reflete de forma mais adequada a realidade sócio-econômica local do que a que pode ser feita a partir da variável população total. Este critério é de uso consensual e reconhecido como sendo o mais apropriado para os Municípios do Estado do Paraná. Por outro lado, a definição do tamanho dos municípios segundo a população total implicaria em um viés decorrente das diferentes proporções entre as populações urbana e rural municipais.

Dessa forma, definida a BDR, o quadro seguir apresenta um extrato da forma como se encontram sistematizadas as suas observações.

---

[36] Segundo a localização do domicílio, a situação pode ser urbana ou rural, definida por lei municipal em vigor em 1 de setembro de 1991. Na situação urbana, consideraram-se as pessoas e domicílios recenseados nas áreas urbanizadas ou não, correspondentes às cidades (sedes municipais), às vilas (sedes distritais) ou às áreas urbanas isoladas. A situação rural abrange a população e os domicílios recenseados em todas a área situada fora desses limites, inclusive os aglomerados rurais de extensão urbana, os povoados e os núcleos.

QUADRO 5.2 - ALGUMAS OBSERVAÇÕES DA BDR

```
0,0,0.154043646,0.264441592,0.017971759,0.073170732,0.333761232,0.0975609
76,0.002567394,0.000893921,0.232121573,0.012514899,0.075089392,0.09475566
2,2,3,tres
0,0,0.745573159,0.165890028,0.080149115,0,0.682199441,0.727865797,0.00186
3933,0.002472799,0.265331355,0.06206726,0.087784372,0.047725025,1,1,um
0,0,0.009698276,0.440732759,0,0,0.385775862,0.004310345,0.002155172,0.001
340842,0.248860284,0.049611156,0.140520247,0.039420756,2,1,um
0,0,0.897887324,0.100352113,0.084507042,0.711267606,0.647887324,0.9278169
01,0.00528169,0.003144654,0.255166217,0.040431267,0.107367475,0.049415993
,1,1,dois
0,0,0.105084746,0.4,0.050847458,0.010169492,0.318644068,0.010169492,0,0,0
.232833465,0.042620363,0.113654301,0.054459353,2,1,dois
```

NOTA: Os itens 2, 3 e três foram sublinhados, de forma a ressaltar a descrição da base.

Assim, conforme pode ser verificado no quadro 5.2, a primeira observação pertence a classe "tres", a sua categoria é 3 e tratam-se de dados rurais de um município do Estado do Paraná.

Já a tabela 5.1 apresenta a freqüência das classes da BDR, segundo categorias e condição do domicílio.

TABELA 5.1 - FREQÜÊNCIA DAS CLASSES POR CATEGORIAS E SITUAÇÃO DE DOMICÍLIO PARA OS MUNICÍPIOS PARANAENSES

CLASSE	CATEGORIA									TOTAL
	1	2	3	4	5	6	7	8	9	GERAL
Urbano										
Um	68	36	18	12	0	1	1	0	0	136
Dois	74	30	22	12	7	3	0	0	0	148
Tres	9	2	6	2	3	1	0	0	0	23
Quatro	5	2	2	1	3	2	1	0	0	16
TOTAL										323
Rural										
Um	68	36	18	12	0	1	1	0	0	136
Dois	74	30	22	12	7	3	0	0	0	148
Tres	9	2	6	2	3	1	0	0	0	23
Quatro	5	2	2	1	3	2	0	0	0	15
TOTAL										322

Conforme a metodologia do Sistema de Indicadores Analíticos descrita e sintetizada anteriormente e o apresentado na tabela 5.1 anterior, pode-se observar a seguinte distribuição de classes:

- Classe um: municípios em situação muito crítica Urbano e Rural: 42% .
- Classe dois: municípios em situação crítica Urbano e Rural: 46%.
- Classe tres: municípios em situação razoável Urbano e Rural: 7%.
- Classe quatro: municípios em situação boa Urbano e Rural: 5%.

Lidando com Desbalanceamento de Classes

Uma das dificuldades encontradas na realização de experimentos com bases de dados do mundo real é o balanceamento de suas classes para o treinamento dos algoritmos de aprendizado.

No caso da BDR, é possível observar que a participação percentual de municípios pertencentes às classes "tres" e "quatro" é significativamente inferior à participação percentual de municípios das classes "um" e "dois" (tabela 5.1).

Dado o fato de que esta diferença de participação entre as classes prejudica os resultados obtidos através dos três algoritmos testados neste trabalho (capítulo 6), foi criado um novo método para lidar com esse problemas, o qual se baseia da idéia de expansão da BDR. O objetivo desta expansão é procurar melhorar a taxa de acerto obtida através da execução do algoritmo de aprendizado C5.0 sobre a BDR.

Nessa expansão não foi criada nenhuma nova observação, ou seja, nenhum novo município. As observações das classes foram repetidas. Para esta repetição surge a seguinte questão: "quantas vezes cada observação deve ser repetida?".

Como forma de responder a esta pergunta, foi criado um processo *wrapper* envolvendo o método de Algoritmo Genético (AG) e Árvore de Decisão. Um *wrapper* é um algoritmo que trata um outro algoritmo como uma "caixa preta" e apropria a saída desse algoritmo em seu processamento (KOHAVI, 1996). Para a expansão, o Algoritmo Genético trata o método de Árvore de Decisão(estado da arte C5.0) como uma "caixa preta" para a obtenção da função objetivo, denominada *fitness* (TURNEY, 1995).

Balanceamento da Base

O motivo que determinou a escolha do método de Algoritmo Genético para expansão da BDR, em detrimento de outros métodos, foram os seguintes (GOLBERG, 1989):

- o AG não usa nenhuma heurística específica do domínio do problema, apenas usa a informação da função objetivo (*fitness*) e não de outras funções auxiliares; e
- o AG é flexível e genérico, o que o torna apropriado para problemas que não tenham um método bem definido ou problemas onde não se conhece o mecanismo de ajuste.

Uma desvantagem apresentada pelo AG é com relação ao tempo de processamento, pois é um método lento. Porém, no caso específico da expansão da BDR, não estão sendo incluídos atributos dinâmicos, isto é, atributos cujos conteúdos variam rapidamente com o tempo. Mas precisamente, os valores dos atributos da BDR só mudam a cada novo recenseamento, a cada 10 anos. Assim sendo a questão de tempo de processamento deixa de ser relevante, pois a classificação não tem a necessidade de ser repetida freqüentemente.

Os Algoritmos Genéticos requerem um conjunto de parâmetros ou variáveis oriundos do problema de otimização a ser resolvido. Esses parâmetros devem ser codificados, constituindo-se em indivíduos de uma população. No caso da expansão da BDR, o problema de otimização se resume em encontrar uma distribuição de freqüência das classes no conjunto de treinamento que minimize a taxa média de erro dos classificadores gerados pelo C5.0.

A idéia básica do AG é evoluir um conjunto de indivíduos, candidatos à solução de um problema. O processo de evolução gradualmente melhora a qualidade da população pela construção de novos indivíduos montados a partir de partes dos melhores indivíduos de gerações anteriores. O conjunto mais recente de indivíduos gerados compõem a geração atual G_a, a qual serve como base para a combinação dos melhores indivíduos para compor uma nova geração G_n. Este processo evolutivo é repetido até que a qualidade desejada nos indivíduos seja encontrada, ou até que um número máximo de geração seja alcançado.

Para a execução do AG é necessário gerar uma população inicial G_1 de indivíduos. Um indivíduo é composto por um conjunto de atributos, ou seja, pelo material genético do indivíduo, dito genoma do indivíduo, que, no caso da expansão da BDR é o número de ocorrências de cada exemplo das classes "um", "dois", "tres" e "quatro", segundo a situação de domicílio e a categoria. Para cada indivíduo da população é identificado um valor real que representa a função objetivo de sobrevivência ou o *fitness*.

Genoma

Os primeiros 14 genes do genoma pode ser apresentados da seguinte forma:

1. N. de vezes que cada observação da classe "um", situação 1, categoria 1 é repetida.
2. N. de vezes que cada observação da classe "um", situação 1, categoria 2 é repetida.
3. N. de vezes que cada observação da classe "um", situação 1, categoria 3 é repetida.
4. N. de vezes que cada observação da classe "um", situação 1, categoria 4 é repetida.
5. N. de vezes que cada observação da classe "um", situação 1, categoria 5 é repetida.
6. N. de vezes que cada observação da classe "um", situação 1, categoria 6 é repetida.
7. N. de vezes que cada observação da classe "um", situação 1, categoria 7 é repetida.
8. N. de vezes que cada observação da classe "um", situação 2, categoria 1 é repetida.
9. N. de vezes que cada observação da classe "um", situação 2, categoria 2 é repetida.
10. N. de vezes que cada observação da classe "um", situação 2, categoria 3 é repetida.

11. N. de vezes que cada observação da classe "um", situação 2, categoria 4 é repetida.
12. N. de vezes que cada observação da classe "um", situação 2, categoria 5 é repetida.
13. N. de vezes que cada observação da classe "um", situação 2, categoria 6 é repetida.
14. N. de vezes que cada observação da classe "um", situação 2, categoria 7 é repetida.

Esta estrutura é repetida da mesma forma para as classes "dois", "tres" e "quatro".

Na geração da população inicial, o número de vezes que cada observação da classe X_i, situação N_i e categoria K_i é repetida, é obtido de forma aleatória de acordo com o limite fornecido como parâmetro no momento do processamento.

Assim, para cada um dos indivíduos da população inicial é construída uma nova base de treinamento, conforme a respectiva distribuição do número de observações descrita em cada gene.

Dado que o número de observações pertencentes às classes é estático, os exemplos das classes são repetidos para a obtenção da nova base de treinamento expandida.

Para o processo de evolução do método AG, a BDR foi segmentada em três conjuntos: de treinamento (T_r), de validação(V) e de teste(T_s), conforme pode ser observado na tabela 5.2. No processo AG, o conjunto T_r é utilizado para a construção do classificador para cada indivíduo do AG e o conjunto V para o teste daquele classificador e cálculo da função objetivo daquele indivíduo. O conjunto T_s só será utilizado na fase de comparação dos resultados do algoritmo C5.0, após o AG ter terminado sua evolução.

TABELA 5.2 - DISTRIBUIÇÃO DOS EXEMPLOS DA BDR NOS TRÊS SUBCONJUNTOS

CONJUNTO DE TREINAMENTO			CONJUNTO DE VALIDAÇÃO			SUBCONJUNTO DE TESTE		
Um	163	0,42	um	55	0,42	Um	54	0,41
Dois	179	0,46	dois	58	0,44	Dois	59	0,45
Tres	27	0,07	tres	9	0,06	Tres	10	0,07
Quatro	18	0,05	quatro	7	0,05	Quatro	6	0,04
TOTAL	387	1	TOTAL	129	1	TOTAL	129	1

Foram adotadas duas formas de função objetivo *(fitness)* para a expansão da base: o valor do erro médio fornecido pela execução do C5.0 e a taxa de acerto sensível a desbalanceamento de classes. Para cada função objetivo, um conjunto de experimentos separado foi realizado.

A justificativa para adotar não apenas o valor do erro médio fornecido pelo sistema C5.0 como função objetivo, mas também outras medidas de avaliação se deve ao fato da base não estar balanceada em relação às classes (KONONENKO, 1991). Através da figura 5.1 pode ser observado que, apesar do algoritmo não ter conseguido classificar corretamente nenhum exemplo da classe "tres" e "quatro", o valor do erro médio fornecido pelo sistema é de 36.4%. Assim, este valor não deixa evidente que nenhum dos exemplos das classes menos freqüentes foi classificado corretamente.

FIGURA 5.1 - EXEMPLO DE SAÍDA DO C5.0

```
C5.0 INDUCTION SYSTEM [Release 1.06]   Thu Nov 12 11:13:20 1998
-----------------------------------
 Options: File stem <sub1235>

Read 516 cases (16 attributes) from sub1235.data

Evaluation on training data (516 cases):
                    Decision Tree
                    ----------------
                    Size       Errors

                      55     76(14.7%)      <<

                     (a)      (b)    (c)      (d)        <-classified as
                     ----    ----   ----     ----
                      193     24                1        (a): class um
                      11      224       1              (b): class dois
                       6       19      11        1      (c): class tres
                       3        9       1       12      (d): class quatro

Evaluation on test data (129 cases):
                    Decision Tree
                    ----------------
                     Size      Errors

                      55     47(36.4%)      <<

                      (a)     (b)    (c)      (d)        <-classified as
                      ----    ----   ----     ----
                       41      13                        (a): class um
                       15      41                4       (b): class dois
                        3       6                        (c): class tres
                        1       5                        (d): class quatro
```

A taxa de acerto sensível ao desbalanceamento de classes é obtida da seguinte forma:

- é gerado o classificador a partir da execução do C5.0 sobre a BDR estendida, conforme a composição descrita pelo genoma acima, gerado a partir do

conjunto T_r. Um possível resultado dessa execução pode ser observado na figura 5.1. Neste caso, por exemplo, eram previstos 54 (41 + 13) casos classificados como classe "um", sendo realmente classificados apenas 41 casos. Também foram previstos 60 (15 + 41 + 4) casos classificados como classe "dois", sendo realmente classificados apenas 41 casos. O mesmo raciocínio se repete para as classes "tres" e "quatro";

- C5.0 testa o classificador resultante sobre o conjunto V;
- conjunto V não é "visto" durante o treinamento do classificador;
- como resultado do teste, é gerada uma tabela contendo o que era previsto em comparação com o real;
- calcula-se a participação percentual dos acertos em cada classe. Por exemplo a participação percentual de acerto para a classe "um" é de 0.759% e assim sucessivamente;
- a taxa de acerto sensível ao desbalanceamento de classes é a raiz quarta da multiplicação das 4 respectivas taxas de acerto para cada classe. Esse cálculo efetivamente calcula a média geométrica das 4 taxas de acerto calculadas separadamente paras as 4 classes;
- para a construção do classificador está sendo invocado o C5.0 na sua forma *default*.

Formalmente, a taxa de acerto sensível ao desbalanceamento de classes é dada pela seguinte fórmula (KONONENKO, 1991):

$$\text{Taxa de acerto} = (pp_{c1} * pp_{c2} * pp_{c3} * pp_{c4}) ** (1 / 4) \qquad (5.1)$$

onde pp_{ci} é a participação percentual dos acertos

A vantagem em adotar a medida da taxa de acerto sensível ao desbalanceamento de classes para a função objetivo e não apenas adotar o valor do erro médio do C5.0, é que a primeira medida é uma medida mais justa para o problema

das classes desbalanceadas. Por exemplo, num conjunto de observações onde a classe mais freqüente ocorre em 99% dos casos, seria trivial obter um valor médio de erro de 1% e, portanto, bastaria construir um classificador que sempre previsse a classe mais freqüente. Entretanto isso não significa que o resultado deste classificador seja bom. Ao considerar o exemplo anterior (figura 5.1), a medida de taxa de acerto sensível ao desbalanceamento tem o valor 0 (zero), pois o classificador não conseguiu classificar corretamente nenhuma das observações das classes "tres" e "quatro". O que significa que o resultado do classificador construído é ruim.

Estratégias de Gerenciamento da População

Para evoluir da população inicial, chamada de geração inicial G_i, para uma nova geração G_n, são aplicados operadores genéticos que, a partir de indivíduos de uma geração, geram os novos indivíduos de uma geração subsequente, formando assim uma nova população. Todas as populações são mantidas em tamanho constante, com 100 indivíduos no caso deste trabalho. O número de indivíduos que compõem cada geração é determinado de forma empírica, dado que não existe nenhuma regra que indique que 100 seja pior do que 150 ou qualquer outro número. Quanto maior o número de indivíduos, melhor tende a ser a qualidade da solução encontrada pelo AG, porém maior será o tempo de processamento. O número de 100 indivíduos é um número comumente usado na literatura.

Para a seleção dos indivíduos que devem fazer parte da nova geração são adotados dois métodos: o "elitista"[37] com fator (2) e, para os demais 98 indivíduos restantes na geração atual, é adotado o método do "torneio"[38]. O fator (2), adotado no método elitista, é utilizado com o objetivo de não serem perdidos os dois melhores

---

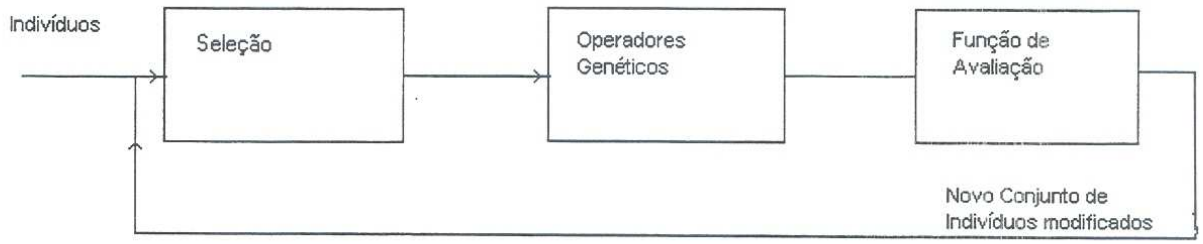
[37]O método elitista prevê que os melhores indivíduos de uma geração sejam mantidos inalterados como elementos da próxima geração.

[38]O método do torneio prevê que sejam escolhidos dois indivíduos de forma aleatória. O indivíduo com o melhor *fitness* é selecionado para reprodução. Novamente, mais dois indivíduos são escolhidos de forma aleatória, sendo novamente selecionado para reprodução o que tiver o melhor *fitness*.

indivíduos de uma determinada geração, dado que, pela aleatoriedade do método do torneio, não é possível garantir que o melhor ou os dois melhores indivíduos venham a compor as próximas gerações. Desta forma, fica preservada a reprodução do que se tem de melhor, em termos de função objetivo, de uma geração atual para uma nova geração. A este processo é dito "seleção por torneio com elitismo".

O AG não usa operações de generalização e de especialização conforme descrito no capítulo 2, mas sim dois outros operadores: o *crossover* e a mutação. A partir dos dois indivíduos selecionados através do método do "torneio", são utilizados os operadores de mutação[39] (com probabilidade de 1%) e de *crossover*[40] (com probabilidade de 80%). Após o processamento destes dois operadores sobre os dois indivíduos, estes passam a compor a população da nova geração.

FIGURA 5.2 - PROCESSO DE GERAÇÃO DE NOVAS POPULAÇÕES DE INDIVÍDUOS NO AG



É um consenso que a taxa da probabilidade do operador *crossover* deve ser muito superior à taxa de probabilidade do operador de mutação. No caso da mutação, é

---

[39] Para cada um dos atributos que compõem a estrutura do indivíduo é definido um número aleatório entre 0 e 1. Caso este número seja inferior à taxa de probabilidade definida pelo usuário, este atributo sofrerá mutação, ou seja, será definido um novo número que atenda aos limites inferior e superior. O limite inferior pode ser 1 e o limite superior é o resultado da razão entre o número de instâncias da classe mais freqüente pelo número de instâncias da classe mais rara.

[40] É gerado um número aleatório entre 0 e 1. Caso este número seja inferior à taxa de probabilidade definida pelo usuário é realizada a operação de crossover. Esta operação ocorre a partir de dois indivíduos previamente selecionados, onde são gerados dois novos indivíduos por troca de alguns atributos do primeiro indivíduo, que, assim, passam a ser também atributos do segundo indivíduo e vice-versa.

importante que exista alguma aleatoriedade, garantindo assim a introdução de novo material genético nos indivíduos da próxima população. Porém esta aleatoriedade não pode ser alta, pois caso contrário o processo estaria sempre retornando a novas populações iniciais, não garantindo a evolução para uma população que resulte em um valor de função objetivo adequado para a solução do problema.

O operador *crossover* possibilita que o resultado da função objetivo seja ainda melhor, dada a probabilidade de troca entre alguns genes dos dois indivíduos selecionados especificamente para esta operação.

Para os experimentos deste trabalho foi arbitrado, de forma empírica, o valor 100 como sendo o número máximo de gerações a serem geradas.

5.3 AVALIAÇÃO DE DESEMPENHO DOS ALGORITMOS

Na análise comparativa realizada com diferentes algoritmos, algumas características básicas devem ser atendidas, tais como:

- precisão na classificação – as regras induzidas devem ser capazes de classificar novas situações mesmo diante de ruídos;
- simplicidade nas regras – as regras devem ser as menores possíveis. No entanto, elas tendem a se tornar longas na presença de ruído. Sendo assim, para garantir regras curtas é necessário relaxar a exigência de que as regras induzidas sejam absolutamente consistentes para todo o conjunto de dados de treinamento. A forma de relaxamento dessa exigência é determinada pelo equilíbrio entre a precisão e a simplicidade desejadas;
- eficiência na geração das regras – onde o tempo necessário para a geração das regras deveria, idealmente, ser linear em relação ao tamanho do conjunto de treinamento;
- grau de interesse das regras – o quanto as regras devem apresentar algo de surpreendente e de interessante e que sejam, assim, reconhecidas pelo especialista humano.

As três primeiras características são comuns aos processos de aprendizado de máquina. A última característica envolve especificamente a aplicação de Data Mining.

5.3.1 Precisão na Classificação

Para a indução das árvores de decisão, as bases são tratadas de acordo com os critérios descritos a seguir.

BDPL

Nos testes com a BDPL foi escolhido o método Lachenbruch, também conhecido "deixa-um-fora" (*leave-one-out*), para a avaliação da precisão e do tempo despendido. Os procedimentos para a realização deste método são (JOHNSON, 1978):

- o i-ésimo exemplo do conjunto original é retirado;
- o classificador é gerado a partir dos (n – i-ésimo) exemplos restantes;
- este classificador resultante é testado sobre o i-ésimo exemplo;
- os procedimentos 1, 2, e 3 são repetidos alterando o valor de *i*, até que os *n* exemplos tenham sido retirados e respectivamente testados.

Neste método, a taxa de erro é obtida pela fórmula:

$$\text{Taxa} = \left(\left(\sum_{i=1}^{g} n_{iM(H)} \right) / \left(\sum_{i=1}^{g} n_i \right) \right) \qquad (5.2)$$

onde: $n_{iM}^{(H)}$ é o número de exemplos classificados erroneamente na iteração *i*

n_i é o número total de exemplos testados na classificação

g é o número de classes

Base de Dados Estendida com Exemplos Repetidos

Para calcular a precisão dos algoritmos CN2, C4.5 e C5.0 foram realizados 7 conjuntos de experimentos, 2 conjuntos para os algoritmos CN2 e C4.5, e para o C5.0 3 conjuntos. Para o CN2, os dois classificadores foram gerados a partir da opção de regras ordenadas e de regras não-ordenadas. No caso do C4.5, os classificadores foram gerados a partir do C4.5 padrão e do C4.5rules. E para o C5.0, os classificadores foram gerados a partir do C5.0 padrão e do C5.0 com a opção de *boosting*.

O objetivo destes experimentos foi avaliar em quanto a adoção do método AG melhora a taxa de acerto sensível a desbalanceamento de classes e o valor do erro médio, obtidos sobre o mesmo conjunto, antes e depois da expansão.

Ao todo foram executados 24 experimentos evolutivos a partir do processo *wrapper*, com diferentes "sementes" para a obtenção de 24 propostas distintas de expansão dos exemplos da BDR: 12 experimentos adotam a taxa de acerto sensível a desbalanceamento de classes como sendo a função objetivo e 12 adotam o valor do erro médio do C5.0.

Dentro destes dois grupos de 12 evoluções, que adotam distintas funções objetivo foram estabelecidos dois critérios diferentes quanto ao número de vezes que cada exemplo pode ser repetido[41]. Das 12 evoluções 6 permitem que todos os exemplos da base, independentemente da classe a qual pertence, possam ser replicados. As outras 6 evoluções permitem que apenas os exemplos pertencentes às classes raras ("tres" e "quatro") possam ser repetidos, ficando os demais exemplos, pertencentes às classes "um" e "dois", com o valor de repetição fixado em 1(um), ou seja, ocorrendo uma única vez na base de treinamento.

A partir destas 24 propostas são construídos, respectivamente, 72 classificadores, (24 x 3 versões do C5.0 – C5.0 *default*, C5.0 *boosting*, C5.0 *rules*), os quais são testados sobre o conjunto de teste (Ts), não visto durante a evolução do AG (tabela 5.2).

Para a construção dos 72 distintos classificadores, o novo conjunto de treinamento incorpora o conjunto de validação acrescido do conjunto de treinamento anterior (tabela 5.2), repetindo os exemplos conforme a distribuição descrita em cada gene correspondente ao genoma dos melhores indivíduos, obtidos através do processo com o AG.

A Figura 5.3 mostra um exemplo de um genoma do indivíduo contendo o valor de cada um dos 56 genes. Nesta figura o primeiro valor 1 significa que cada exemplo da classe "um", área urbana, faixa de população urbana 1 terá apenas uma cópia no conjunto de treinamento. O valor que segue, o número 5 significa que cada exemplo da classe "um", área urbana, faixa de população urbana 2 terá 5 cópias no

[41]Este número de vezes é expresso pelos genes que compõem o genoma de cada indivíduo.

conjunto de treinamento; e assim sucessivamente. Vale salientar que este exemplo de genoma reproduz uma situação onde todos os exemplos de todas as 4 classes podem ser repetidos durante o processo evolutivo.

FIGURA 5.3 - EXEMPLO DE GENOMA DA MELHOR TAXA OBTIDA ATRAVÉS DO AG

1 5 5 0 5 7 1 2 2 5 0 6 5 1 2 4 5 1 0 9 0 1 2 1 5 9 8 0 1 2 2 1 0 6 1 0 3 9 5 9 5 5 0 1 2 8 6 6 4 2 8 5 4 1 4 8 0 2

5.3.2 Simplicidade nas Regras

Para a análise da característica de simplicidade pode ser adotado o seguinte critério:

- Número total de regras;
- Número total de condições;
- Número médio de condições por regra;

5.3.3 Eficiência na Geração das Regras

Para a análise da eficiência na geração das regras foi adotado o critério tempo despendido para o processamento. Vale ressaltar que, nesta avaliação, a máquina onde os experimentos foram realizados não estava com uso exclusivo para esta atividade.

5.3.4 Grau de Interesse das Regras

A análise do grau de interesse das regras fica recomendada como trabalhos futuros, a ser realizado junto a um especialista humano.

6 RESULTADOS

6.1 BASES DE DADOS PADRÃO DA LITERATURA.

As Bases de Dados Padrão da Literatura (BDPL) utilizadas nos testes foram obtidas em http://www.ics.uci.edu/~mlearn/MLRepository.html

A Tabela 6.1 mostra os resultados dos testes de precisão obtidos a partir do método Lachenbruch sobre estas bases de dados e sobre a base de dados real (BDR).

Os experimentos foram realizados considerando os algoritmos CN2, C4.5 e C5.0. Para o CN2, foram executadas as duas variantes do algoritmo: uma descobrindo um conjunto não-ordenado de regras e uma descobrindo uma lista ordenada de regras. O C4.5 foi executado em suas versões C4.5 *default* e C4.5 *rules*. Finalmente, o C5.0 foi executado em suas versões *default*, *rules* e *boosting*. A base de dados Crx não pode ser testada para as duas versões do CN2 dada a sua forma de apresentação dos dados.

TABELA 6.1 - TAXA DE ERRO OBTIDA ATRAVÉS DO MÉTODO LACHENBRUCH SOBRE OS EXPERIMENTOS COM A BDPL E BDR

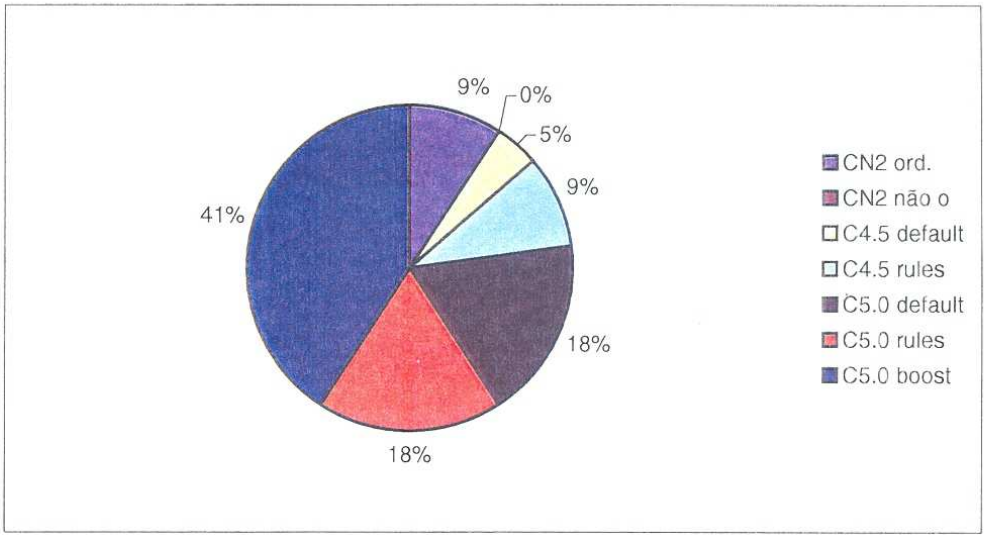
BASES	ERRO (%)						
	CN2 ord.	CN2 não o	C4.5 default	C4.5 rules	C5.0 default	C5.0 rules	C5.0 boost
Allbp	3,53	3,89	2,96	2,82	2,57	3,07	2,50
Balance	16,10	23,60	37,59	21,60	47,20	37,59	22,08
Bupa	38,90	35,73	38,55	33,04	28,11	26,66	29,85
Crx	NA	NA	17,94	15,62	16,35	15,91	13,02
Dis	1,17	1,60	1,17	1,10	0,09	0,78	0,09
Glass	2,31	4,16	2,33	2,33	2,33	2,33	2,33
House	5,46	5,72	4,13	3,67	3,21	3,21	4,59
Imports	20,28	21,73	18,53	26,34	15,12	14,63	12,68
Ionos	13,59	8,21	7,67	5,39	10,22	10,79	5,39
Iris	6,57	7,23	6,57	5,92	5,92	5,92	7,89
Seg	16,03	21,22	12,32	12,79	11,84	11,84	9,95
Sick	2,10	2,85	1,14	1,46	1,25	1,49	1,17
Soybean	11,00	14,23	15,96	21,82	14,65	14,33	6,84
Yeast	47,30	49,19	47,03	42,92	45,52	44,87	40,83
Wine	7,22	8,88	8,42	7,30	6,74	6,17	2,80
BDR	44,97	37,09	44,18	38,44	34,41	34,41	35,50
Média	15,77	16,36	16,66	15,16	15,35	14,63	12,34

NOTA: NA – não se aplica por incompatibilidade de valores para os atributos previsores.

Conforme pode ser observado na tabela 6.1, nenhum dos três algoritmos pode ser considerado o melhor algoritmo, ou seja, obteve a menor taxa de erro independentemente da base de dados sobre a qual esteja sendo processado.

Porém, a partir do gráfico 6.1 é possível concluir que, em média, o algoritmo C5.0 com a opção de *boosting* obteve a melhor performance em 41 % das bases ao serem comparadas as taxas de erro das execuções do CN2 (ordenado e não-ordenado), C4.5 (*default* e *rules*) e o C5.0 (*default*, *rules* e *boosting*). Enquanto o C5.0 obteve os melhores resultados, o CN2 na versão não-ordenado obteve o pior resultado, ou seja, não teve o melhor resultado em nenhuma das 16 bases testadas.

GRÁFICO 6.1 - NÚMERO DE VEZES QUE OS ALGORITMOS OBTIVERAM O MELHOR RESULTADO (MENOR ERRO)

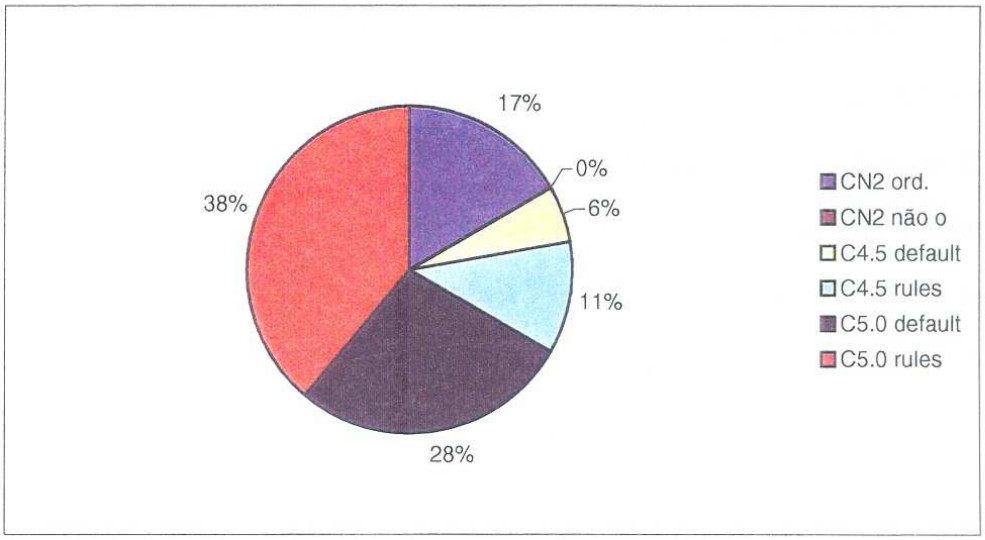


De certa forma este resultado não é surpreendente, dado que a execução do C5.0 com a opção *boosting* ocorreu pela construção de um classificador baseado em outros 10 classificadores, ou seja, no caso deste algoritmo não é somente um classificador que determina a classe a ser estabelecida para um novo exemplo, mas sim um conjunto de classificadores. Este fato tende a aumentar a taxa de precisão do classificador construído. Naturalmente, C5.0 *boosting* apresenta a desvantagem de requerer um tempo de execução maior, a fim de gerar 10 classificadores em vez de apenas um.

O gráfico 6.2 mostra os resultados dos experimentos sobre as 16 bases desconsiderando o resultado apresentado pelo C5.0 *boosting*, onde é possível observar que

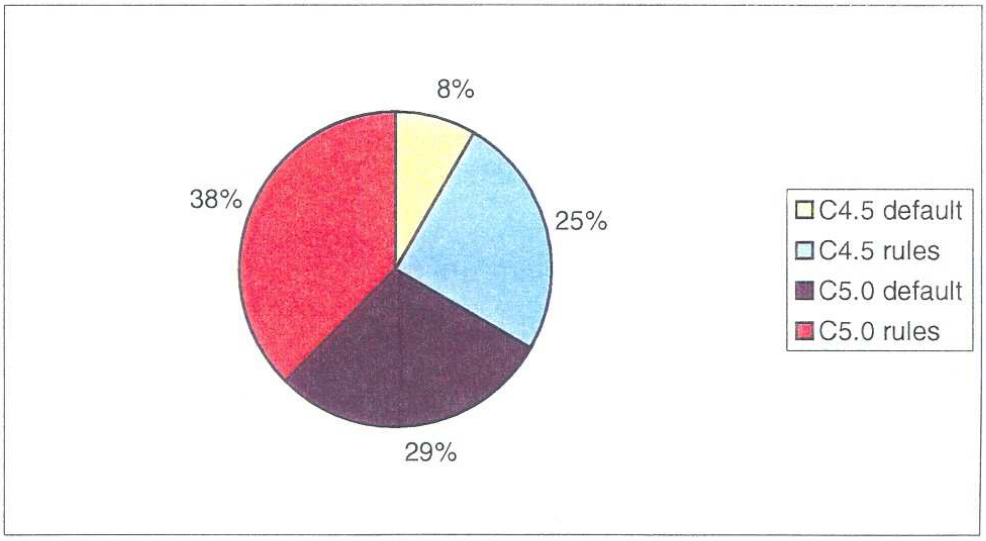
o algoritmo C5.0 – *default* (28%) e *rules* (38%) – também obteve, em média, melhores resultados e que o CN2 na sua versão não-ordenada continua obtendo o pior resultado.

GRÁFICO 6.2 - NÚMERO DE VEZES QUE OS ALGORITMOS OBTIVERAM O MELHOR RESULTADO – SEM *BOOSTING* (MENOR ERRO)



Já se a comparação for restrita aos resultados obtidos a partir dos algoritmos C5.0 com o C4.5 para as versões *default* e *rules*, o C5.0 continua obtendo em média os melhores resultados. Conforme pode ser observado através do gráfico 6.3 o C5.0 obteve as melhores taxas de erro em 29% e 38% paras as duas versões analisadas respectivamente.

GRÁFICO 6.3 - NÚMERO DE VEZES QUE OS ALGORITMOS C5.0 E C4.5 OBTIVERAM O MELHOR RESULTADO - SEM *BOOSTING* (MENOR ERRO)



Infelizmente, os critérios de desenvolvimento do algoritmo C5.0 não estão disponíveis na literatura, o que impede que estes resultados possam ser plenamente justificados. O que se tem divulgado em relação ao C4.5 e C5.0 (QUINLAN, 1997) é que houve uma preocupação muito grande com relação à redução do tempo necessário para a execução do C4.5 visando o desenvolvimento do algoritmo C5.0.

Conforme pode ser observado na Tabela 6.2[42], realmente houve uma redução de 35,65% no tempo médio de execução do C5.0 em relação ao C4.5, ambos na sua versão *default*. Já se a comparação for realizada para estes dois algoritmos, considerando a forma *rules*, a redução de tempo despendido chega a 74,04%. A redução de tempo de execução é ainda maior se forem comparados os tempos do C5.0 *default* em relação ao CN2 ordenado e CN2 não-ordenado: 97,08% e 94,58% respectivamente.

TABELA 6.2 - TEMPO DESPENDIDO COM OS EXPERIMENTOS ATRAVÉS DO MÉTODO LACHENBRUCH SOBRE A BDLP E BDR

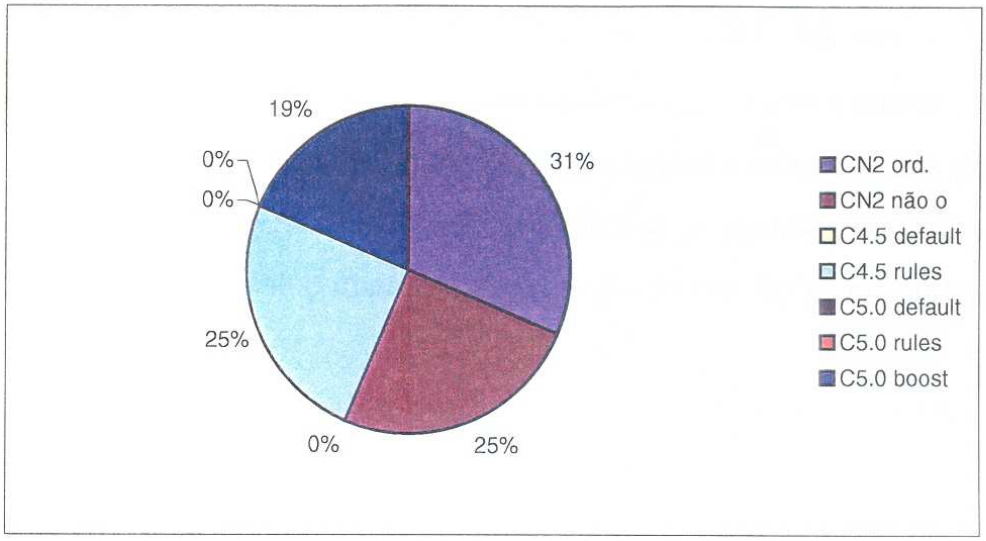
BASES	TEMPO (s)						
	CN2 ord.	CN2 não o	C4.5 default	C4.5 rules	C5.0 default	C5.0 rules	C5.0 boost
Allbp	788309	263322	26629	81549	17370	40695	191611
Balance	2591	11048	617	8757	631	840	1286
Bupa	440	2428	465	3673	307	603	1452
Crx	NA	NA	2091	16680	1444	2005	5037
Dis	583763	461337	22701	50080	12973	20012	135675
Glass	622	1572	348	412	201	178	578
House	2641	2771	452	2963	136	410	1343
Imports	5068	4104	549	2014	147	520	3061
Ionos	5297	7476	2685	5601	696	862	22985
Iris	276	325	172	55	183	28	433
Seg	1147	2045	712	716	770	291	3549
Sick	785998	446388	23284	65198	20849	23234	147759
Soybean	5745	15597	1204	2844	878	1128	5407
Yeast	74091	153542	17264	304341	12222	57889	104400
Wine	247	399	218	79	226	97	393
BDR	18024	33828	10638	52832	1768	6392	15289
Média	151617	93745	6877	37362	4425	9699	40016

NOTA: NA – não se aplica por incompatibilidade de valores para os atributos previsores.

O gráfico 6.4 evidencia que em mais 50% dos experimentos as duas versões do algoritmo CN2 despenderam o maior tempo para as suas respectivas execuções.

[42] Estes tempos devem ser considerados apenas em suas ordens de grandeza, dado que a máquina onde estavam sendo realizados os experimentos não foi de uso exclusivo para os mesmos.

GRÁFICO 6.4 - TEMPO DESPENDIDO COM OS EXPERIMENTOS ATRAVÉS DO MÉTODO LACHENBRUCH SOBRE A BDLP E BDR



A tabela 6.3 mostra o número de condições no antecedente das regras descobertas usando os algoritmos CN2 (ordenado e não-ordenado), C4.5 e C5.0 nas suas versões *rules* para as 15 bases que compõe o grupo de BDPL e para a base de dados real (BDR). As células sublinhadas correspondem à versão do algoritmo que obteve o melhor desempenho.

TABELA 6.3 - NÚMERO DE CONDIÇÕES NOS ANTECEDENTES DAS REGRAS DESCOBERTAS

BASES	CN2 O	CN2 NO	C4.5R	C5.0R
Allbp	112	157	37	27
Balance	122	576	57	52
Bupa	99	115	65	51
Crx	NA	NA	32	29
Dis	50	72	13	27
Glass	6	11	10	12
House	57	88	13	13
Imports	57	46	62	82
Ionos	27	48	16	26
Iris	11	16	6	5
Seg	23	43	22	38
Sick	99	113	35	69
Soybean	71	131	102	148
Yeast	500	647	152	671
Wine	10	21	8	10
BDR	189	117	44	201
Média	96	147	42	91

NOTA: NA – não se aplica por incompatibilidade de valores para os atributos previsores.

É possível observar que em 8 destas bases, quando ocorre uma sensível redução no número de cortes identificado por um classificador em comparação aos demais, verifica-se também que este classificador é o que obtém a menor taxa de erro.

Vale salientar que em nenhum dos experimentos relatados nos gráficos 6.1 e 6.2, o algoritmo CN2 versão não-ordenada obteve a menor taxa de erro, e este algoritmo de fato apresentou o maior número médio de condições nos antecedentes das regras (147) na tabela 6.3.

Uma possível explicação para estes resultados se deve ao fato de que um menor número de cortes reduz o problema de que a presença de ruídos nos dados interfiram no resultado do classificador como um todo.

Buscou-se identificar alguma relação que justificasse as menores taxas de erro, como por exemplo o número de regras, o número de atributos nas premissas das regras ou mesmo o número médio de atributos nas premissas, porém nenhum destes quesitos estabelece uma relação direta que justifique quando os algoritmos geram as menores taxas de erro.

Como exemplo, a tabela 6.4 mostra a média de atributos na premissa das regras, sendo que os valores sublinhados correspondem às versões dos algoritmos que obtiveram as menores taxas de erro. Para 56.3% das bases, a menor média não corresponde necessariamente à menor taxa de erro.

TABELA 6.4 - MÉDIA DE ATRIBUTOS NA PREMISSA

BASES	MÉDIA DE ATRIBUTOS NA PREMISSA		
	CN2 ord.	C4.5 rules	C5.0 rules
Allbp	3,61	3,36	2,70
Balance	1,85	1,97	1,86
Bupa	3,19	3,14	3,06
Crx	NA	2,67	2,42
Dis	2,78	2,17	2,45
Glass	1,20	1,67	2,00
House	2,85	2,17	2,33
Imports	3,17	2,95	2,96
Ionos	2,45	1,78	2,45
Iris	1,57	1,50	1,25
Seg	1,92	2,25	2,92
Sick	3,41	2,92	2,88
Soybean	3,04	3,64	4,41
Yeast	5,21	4,47	5,46
Wine	2,00	1,60	2,00
BDR	3,32	3,67	4,10

NOTA: NA – não se aplica pór incompatibilidade de valores para os atributos previsores.

A tabela 6.5 mostra que, em 12 das 16 bases testadas, o C4.5 *rules* e o C5.0 *rules* selecionaram o mesmo atributo como sendo o atributo do nó raiz. Este fato evidencia que, no que concerne a seleção deste atributo, não houve uma mudança significativa na evolução do C4.5 para o C5.0.

TABELA 6.5 - ATRIBUTO PREVISOR SELECIONADO PARA COMPOR O NÓ RAIZ

BASES	ATRIBUTO PREVISOR NA RAIZ	
	C4.5	C5.0
Allbp	Pregnant	Pregnant
Balance	Right-Weight	Right-Weight
Bupa	Sgpt	Gammagt
Crx	A9	A9
Dis	FTI	FTI
Glass	Id-number	Id-number
House	physician-fee-freese	physician-fee-freese
Imports	num-of-doors	num-of-doors
Ionos	A5	A5
Iris	Petal	Petal_width
Seg	Intensity-mean	region-centroid-row
Sick	T3	T3
Soybean	Plant-growth	Plant-growth
Yeast	Mcg	Mcg
Wine	Flavanoids	Flavanoids
BDR	PA74_A202	PA082_A202

6.2 BDR

Os procedimentos descritos a seguir foram realizados com o objetivo de determinar se a abordagem *wrapper* melhorou ou não os resultados obtidos com os classificadores gerados a partir do algoritmo C5.0 sobre a BDR desbalanceada (base original).

Inicialmente, antes da aplicação do processo *wrapper*, foram construídos 3 classificadores a partir da execução de três versões distintas do C5.0, denominadas C5.0 *default*, C5.0 com opção de *boosting* e C5.0 *rules*. Estes classificadores foram construídos a partir do conjunto de treinamento formado pelos subconjuntos T_r e V (tabela 5.2) e testados sobre o subconjunto T_s.

Para cada teste realizado foi obtido o valor do erro médio do algoritmo C5.0 e calculada a taxa de acerto sensível ao desbalanceamento de classes (expressão 5.1). Os resultados são apresentados na tabela 6.6.

TABELA 6.6 - RESULTADOS DAS 3 VERSÕES DO C5.0 SOBRE A BDR ORIGINAL

VERSÕES	TAXA DE ACERTO	ERRO MÉDIO
C5.0 default	31.35	50.40
C5.0 boosting	0.00	32.60
C5.0 rules	0.00	46.50

Como mostra a segunda coluna da tabela 6.6, apenas a opção C5.0 *default* obteve uma taxa de acerto sensível ao desbalanceamento de classes diferente de 0 (zero). As versões C5.0 *boosting* e C5.0 *rules* não classificaram corretamente nenhum dos exemplos das classes menos freqüentes (classes "tres" e "quatro").

Como pode ser observado na terceira coluna, a versão C5.0 *default* obteve o pior resultado dentre as três versões do C5.0. O melhor resultado foi obtido pelo C5.0 *boosting*.

De forma a avaliar a melhoria ou não destes resultados a partir da aplicação do processo *wrapper* foram realizados dois conjuntos de experimentos. A diferença entre os dois conjuntos está na definição da função objetivo adotada. O primeiro utiliza o valor do erro médio e o segundo usa a taxa acerto sensível ao desbalanceamento de classes. Os resultados destes experimentos são apresentados a seguir.

6.2.1 Experimentos com o Valor do Erro Médio como Função objetivo

Estes experimentos consistem de 12 execuções do AG, com diferentes populações iniciais, que variam aleatoriamente segundo a semente do AG e agrupados da seguinte forma:

- 6 execuções permitindo que todos os exemplos de todas as classes sejam repetidos conforme a determinação do processo aleatório. Este grupo de execuções será identificado, a partir de agora, como sendo TODOS; e
- 6 execuções permitindo que apenas possam ser repetidos os exemplos das classes menos freqüentes. Assim, os exemplos das classes mais freqüentes somente ocorrerão uma única vez na base de treinamento. Este grupo será identificado como TRES-QUATRO.

Uma vez realizado cada processo evolutivo (*wrapper*), os subconjuntos T_r e V são concatenados e então expandidos de forma a constituir um novo conjunto de treinamento. Assim, os exemplos deste novo conjunto de treinamento são repetidos de acordo com a distribuição associada ao genoma do melhor indivíduo encontrado pelo AG.

Foram construídos 3 classificadores (C5.0 *default*, C5.0 *boosting* e C5.0 *rules*) para cada um dos 12 conjuntos de dados expandidos. No total, foram construídos 36 classificadores.

A tabela 6.7 mostra os resultados das 12 execuções do AG para os grupos TODOS e TRES-QUATRO. As colunas 2 e 5 mostram o número total de exemplos, associado aos melhores indivíduos de cada execução do AG após a expansão da base original, para cada grupo. As colunas 4 e 7 mostram o número da geração onde houve a estabilização da função objetivo, ou seja, quando todos os indivíduos convergem para o mesmo genoma.

Como pode ser observado na tabela 6.7, tanto no caso dos experimentos do grupo TODOS, como no caso dos experimentos do grupo TRES-QUATRO, a especificação de um número de gerações superior a 100 provavelmente não significaria grandes alterações nos resultados, uma vez que a estabilização da população ocorreu, em média, por volta da 59ª geração para o primeiro grupo e da 68ª para o segundo.

TABELA 6.7 - RESULTADOS MEDIDO NAS 12 EXECUÇÕES DO AG COM O VALOR DO ERRO MÉDIO COMO FUNÇÃO OBJETIVO (DADOS DE VALIDAÇÃ0)

EXPERIMENTOS	TODOS			TRES-QUATRO		
	Número Exemplos	Valor Erro Médio	Geração	Número Exemplos	Valor Erro Médio	Geração
Exp 1	1469	27,90	99	1284	30,20	85
Exp 2	1737	31,00	3	1202	29,50	94
Exp 3	1826	27,90	1	1333	29,50	93
Exp 4	1761	24,80	78	1211	30,20	32
Exp 5	1681	25,60	92	1255	29,50	98
Exp 6	1679	28,70	85	1309	32,60	11
Média	1692,17	27,65	59,67	1265,67	30,25	68,83

Na tabela 6.7 também é possível observar, nas colunas 3 e 6, os melhores valores para o erro médio encontrados após a evolução do AG, ou seja, o valor da função objetivo correspondente ao melhor indivíduo em cada processo evolutivo. Cabe ressaltar que esse valor foi calculado com base nos dados de validação (V) e não sobre os dados de teste (Ts). Em contraste, os resultados mostrados na tabela 6.6 e os resultados a serem mostrados no restante deste item foram calculados sobre os dados de teste (Ts).

A tabela 6.8 mostra o valor do erro médio obtido através das 3 versões do C5.0 quando cada versão foi treinada sobre a base de treinamento expandida a partir do melhor indivíduo, onde o processo evolutivo permitiu que todos os exemplos de todas as classes fossem repetidos (grupo TODOS).

A primeira coluna da tabela 6.8 identifica a versão do algoritmo C5.0. A segunda coluna indica o valor do erro médio de cada versão do C5.0 treinado sobre a base de treinamento original, antes da expansão. Esta coluna, na verdade, é a última coluna da tabela 6.6 que foi copiada para a tabela 6.8 para maior conveniência do leitor. As 6 colunas seguintes da tabela 6.8 indicam o valor do erro médio do C5.0 treinado sobre a base expandida associada ao melhor indivíduo da execução do AG correspondente à versão do algoritmo. A 9ª coluna da tabela 6.8 mostra a média dos valores dos erros médios correspondentes às 6 colunas anteriores. Finalmente, a última coluna desta tabela mostra a variação percentual entre as colunas 9 e 2.

Como pode ser observado nesta última coluna, o processo *wrapper* foi significativamente útil para o C5.0 *default,* pois houve uma redução do valor do erro médio em 18,22% se comparado à execução do C5.0 *default* sobre a base original. Também na execução do C5.0 com a opção *rules* houve uma redução do valor do erro médio de 10,25%. Já com o C5.0 com a opção *boosting* houve uma pequena piora de 1,48% no valor do erro médio obtido. Talvez esta pequena piora se justifique devido ao fato do algoritmo de *boosting* já ter sido desenvolvido com o objetivo de priorizar os exemplos de maior dificuldade de classificação na construção do classificador, que no caso deste trabalho, ocorre pelo desbalanceamento das classes na base. Ou seja, o

classificador com a opção de *boosting* gerado a partir do conjunto de treinamento original já é altamente otimizado para lidar com exemplos críticos, e uma tentativa adicional de aumentar a importância desses exemplos críticos pode ser ineficaz.

TABELA 6.8 - VALOR DO ERRO MÉDIO OBTIDO ATRAVÉS DO C5.0 A PARTIR DO CONJUNTO DE TREINAMENTO EXPANDIDO PELO AG (TODOS) – MEDIDO NOS DADOS DE TESTE

VERSÕES	ANTES AG	EXP 1	EXP 2	EXP 3	EXP 4	EXP 5	EXP 6	MÉDIA	VARIAÇÃO(%)
C5.0 *default*	50,40	46,50	38,80	38,80	40,30	43,40	39,50	41,22	-18,22
C5.0 *boosting*	32,60	32,60	31,80	33,30	32,60	34,90	33,30	33,08	1,48
C5.0 *rules*	46,50	45,00	39,50	45,00	35,70	39,50	45,70	41,73	-10,25

A tabela 6.9 mostra o valor do erro médio obtido através das 3 versões do C5.0 quando cada versão foi treinada sobre a base de treinamento expandida a partir do melhor indivíduo, onde o processo evolutivo permitiu que apenas os exemplos das classes "tres" e "quatro" (classes raras) fossem repetidos (grupo TRES-QUATRO).

Nesta tabela (tabela 6.9), as colunas obedecem a mesma distribuição apresentada na tabela 6.8. Como pode ser observado na sua última coluna, o processo *wrapper* obteve resultados similares aos resultados apresentados na tabela 6.8, onde todos os exemplos puderam ser repetidos independentemente da classe à qual pertencem. Ocorreram apenas pequenas alterações nos valores obtidos para o erro médio. No caso do C5.0 *default* e do C5.0 *rules*, a redução foi de 13,62% e de 8,60% para os valores do erro médio, respectivamente. No caso do C5.0 com a opção de *boosting* houve uma piora mais acentuada que a anteriormente encontrada na tabela 6.8 de 6,60%.

TABELA 6.9 - VALOR DO ERRO MÉDIO OBTIDO ATRAVÉS DO C5.0 A PARTIR DO CONJUNTO DE TREINAMENTO EXPANDIDO PELO AG (TRES-QUATRO) - MEDIDO NOS DADOS DE TESTE

VERSÕES	ANTES AG	EXP 1	EXP 2	EXP 3	EXP 4	EXP 5	EXP 6	MÉDIA	VARIAÇÃO(%)
C5.0 *default*	50,40	44,20	41,10	45,70	45,70	45,70	38,80	43,53	-13,62
C5.0 *boosting*	32,60	34,90	31,00	41,10	31,00	39,50	31,00	34,75	6,60
C5.0 *rules*	46,50	45,70	40,30	39,50	41,90	46,50	41,10	42,50	-8,60

Sumarizando, o uso do processo *wrapper* proposto nesta dissertação, utilizando o valor do erro médio como função objetivo do AG, pode ser considerado

benéfico. Em geral, esse processo reduziu significativamente as taxas de erro dos algoritmos C5.0 *default* e C5.0 *rules*. Apenas no caso do algoritmo C5.0 *boosting* o processo *wrapper* levou a um pequeno acréscimo na taxa de erro.

6.2.2 Experimentos com o valor da taxa de acerto sensível ao desbalanceamento de classes como função objetivo

Estes experimentos, da mesma forma que no item 6.2.1, consistem de 12 execuções do AG para os grupos TODOS e TRES-QUATRO.

Do mesmo modo que nos experimentos do item 6.2.1, uma vez completado cada processo evolutivo, os subconjuntos T_r e V são concatenados e então expandidos constituindo o novo conjunto de treinamento, i.e., este novo conjunto de treinamento tem seus exemplos repetidos de acordo com a distribuição associada ao genoma do melhor indivíduo encontrado pelo AG.

Novamente, foram construídos 3 classificadores (C5.0 *default*, C5.0 *boosting* e C5.0 *rules*) para cada um dos 12 conjuntos de dados expandidos, somando um total de 36 classificadores.

A tabela 6.10 mostra os resultados das 12 execuções do AG, correspondentes à função objetivo taxa de acerto sensível ao desbalanceamento de classes. As colunas 2 e 5 mostram o número total de exemplos para os conjuntos TODOS e TRES-QUATRO, associado aos melhores indivíduos de cada execução do AG, respectivamente. As colunas 4 e 7 mostram o número da geração onde houve a estabilização.

De forma semelhante aos resultados da tabela 6.7, pode-se observar na tabela 6.10 que, em ambos os experimentos identificados como TODOS e como TRES-QUATRO, a especificação de um número de gerações superior a 100 não significaria grandes alterações nos resultados, uma vez que a estabilização da população ocorreu, em média, por volta da 36ª geração para o grupo TODOS e da 30ª para o grupo TRES-QUATRO. Vale salientar que neste grupo de experimentos, considerando a taxa de acerto sensível ao desbalanceamento de classes como função objetivo, a estabilidade ocorreu bem mais cedo (36ª e 30ª) do que no grupo de experimentos com o valor do erro médio como sendo esta função (59ª e 68ª).

TABELA 6.10 - RESULTADOS MEDIDO NAS 12 EXECUÇÕES DO AG COM A TAXA DE ACERTO SENSÍVEL AO DESBALANCEAMENTO DE CLASSES COMO FUNÇÃO OBJETIVO (DADOS DE VALIDAÇÃO)

EXPERIMENTOS	TODOS			TRES-QUATRO		
	Número Exemplos	Taxa de Acerto	Geração	Número Exemplos	Taxa de Acerto	Geração
Exp 1	1706	51,84	27	1143	35,70	2
Exp 2	1769	51,85	84	1383	55,60	28
Exp 3	1592	48,88	44	1020	50,17	28
Exp 4	1727	50,03	2	1222	48,84	1
Exp 5	1797	51,50	46	1406	49,58	46
Exp 6	1761	52,88	17	1322	50,96	76
Média	1725,33	51,16	36,67	1249,33	48,48	30,17

Na tabela 6.10 também é possível observar, nas colunas 3 e 6, os melhores valores para o erro médio encontrados após a evolução do AG, ou seja, o valor da função objetivo correspondente ao melhor indivíduo em cada processo evolutivo.

De forma análoga ao exposto no item 6.2.1, cabe ressaltar que os valores da taxa de acerto mostrados na tabela 6.10 foram calculados com base nos dados validação (V) e não sobre os dados de teste (TS). Em contraste, os resultados a serem mostrados no restante deste item foram calculados sobre os dados de teste.

A tabela 6.11 mostra a taxa de acerto sensível ao desbalanceamento de classes obtida através das 3 versões do C5.0 quando cada versão foi treinada sobre a base de treinamento expandida a partir do melhor indivíduo, onde o processo evolutivo permitiu que todos os exemplos de todas as classes fossem repetidos (grupo TODOS).

A primeira coluna da tabela 6.11 identifica a versão do algoritmo C5.0. A segunda coluna indica a taxa de acerto sensível ao desbalanceamento de classes de cada versão do C5.0 treinado sobre a base de treinamento original, antes da expansão. Esta coluna corresponde à segunda coluna da tabela 6.6 que foi copiada para a tabela 6.11. As 6 colunas seguintes da tabela 6.11 indicam os resultados do processo de abordagem híbrida algoritmo genético/árvore de decisão. Assim, estas 6 colunas indicam a taxa de acerto sensível ao desbalanceamento de classes. Finalmente, a última coluna da tabela 6.11 mostra a média da taxa de acerto sensível ao desbalanceamento de classes calculada a partir dos resultados das 6 colunas anteriores.

Como pode ser observado na última coluna da tabela 6.11, este processo de abordagem *wrapper* – com *fitness* taxa de acerto sensível ao desbalanceamento de classe para o grupo TODOS – foi significativamente benéfico para a execução do C5.0 *boosting* e do C5.0 *rules*. As taxas de acerto do C5.0 *boosting* e do C5.0 *rules* sem a abordagem *wrapper* eram 0%. Entretanto, após a execução deste processo estas taxa cresceram para 31,34% e 37% respectivamente. Para o caso do C5.0 *default* também houve uma melhora na taxa de acerto, porém não tão significativa como as taxas para o C5.0 *boosting* e C5.0 *rules,* ou seja passou de 31,35% antes do processo evolutivo para 39,82%.

Ainda pode ser observado na tabela 6.11, coluna 8, que o sexto experimento para a opção C5.0 *boosting*, mesmo após o processo evolutivo, continuou com uma taxa de acerto igual a 0%. Isto se deve ao fato de tratar-se de uma medida – expressão 5.1 – bastante rigorosa. Se o percentual de acerto de uma única classe chegar a 0%, então o valor da taxa de acerto sensível ao desbalanceamento de classes colapsa em 0% também. Este problema já é bastante complicado para um domínio de aplicação onde ocorram apenas 2 classes, sendo uma delas considerada rara. Considerando que a aplicação tratada neste trabalho tem 4 classes, sendo 2 classes consideradas raras, o problema torna-se ainda mais complicado. Em todo caso, o fato de ter sido obtida uma taxa de acerto de 0% com o uso do processo *wrapper* ocorreu em apenas 1 (um) dos 18 experimentos com o AG relatados na tabela 6.11. Logo, na grande maioria dos casos o AG consegue evitar o problema acima.

TABELA 6.11 - TAXA DE ACERTO SENSÍVEL AO DESBALANCEAMENTO DE CLASSES OBTIDA ATRAVÉS DO C5.0 A PARTIR DO CONJUNTO DE TREINAMENTO EXPANDIDO PELO AG (TODOS) – MEDIDO NOS DADOS DE TESTE

VERSÕES	ANTES AG	EXP 1	EXP 2	EXP 3	EXP 4	EXP 5	EXP 6	MÉDIA
C5.0 *default*	31,35	34,11	36,70	44,20	47,58	46,48	29,86	39,82
C5.0 *boosting*	0,00	37,47	30,88	39,29	44,29	36,08	0,00	31,34
C5.0 *rules*	0,00	38,50	33,13	34,28	42,90	43,31	29,86	37,00

De forma semelhante aos experimentos do item 6.2.1, a tabela 6.12 mostra a taxa de acerto sensível ao desbalanceamento de classes obtida através das 3 versões do

C5.0 quando cada versão foi treinada sobre a base de treinamento expandida a partir do melhor indivíduo, onde o processo evolutivo permitiu que apenas os exemplos das classes "tres" e " quatro" (classes raras) fossem repetidos (grupo TRES-QUATRO).

Nesta tabela (tabela 6.12) as colunas obedecem a mesma distribuição apresentada na tabela 6.11. Como pode ser observado na sua última coluna, o processo *wrapper* obteve resultados ainda melhores que os resultados apresentados na tabela 6.11, onde todos os exemplos puderam ser repetidos independentemente da classe à qual pertencem. As taxas de acerto do C5.0 *boosting* e do C5.0 *rules* sem a abordagem *wrapper* eram 0%. Entretanto, após a execução deste processo, estas taxa cresceram para 42.77% e 43.72%, respectivamente. Para o caso do C5.0 *default,* a taxa passou de 31,35% antes do processo evolutivo para 43,76%.

TABELA 6.12 - TAXA DE ACERTO SENSÍVEL AO DESBALANCEAMENTO DE CLASSES OBTIDA ATRAVÉS DO C5.0 A PARTIR DO CONJUNTO DE TREINAMENTO EXPANDIDO PELO AG (TRES-QUATRO) – MEDIDO NOS DADOS DE TESTE

VERSÕES	ANTES AG	EXP 1	EXP 2	EXP 3	EXP 4	EXP 5	EXP 6	MÉDIA
C5.0 default	31,35	34,34	52,82	45,20	38,85	41,33	50,02	43,76
C5.0 boosting	0,00	40,59	39,36	37,40	40,85	47,04	51,37	42,77
C5.0 rules	0,00	34,11	56,54	44,28	37,32	44,59	45,49	43,72

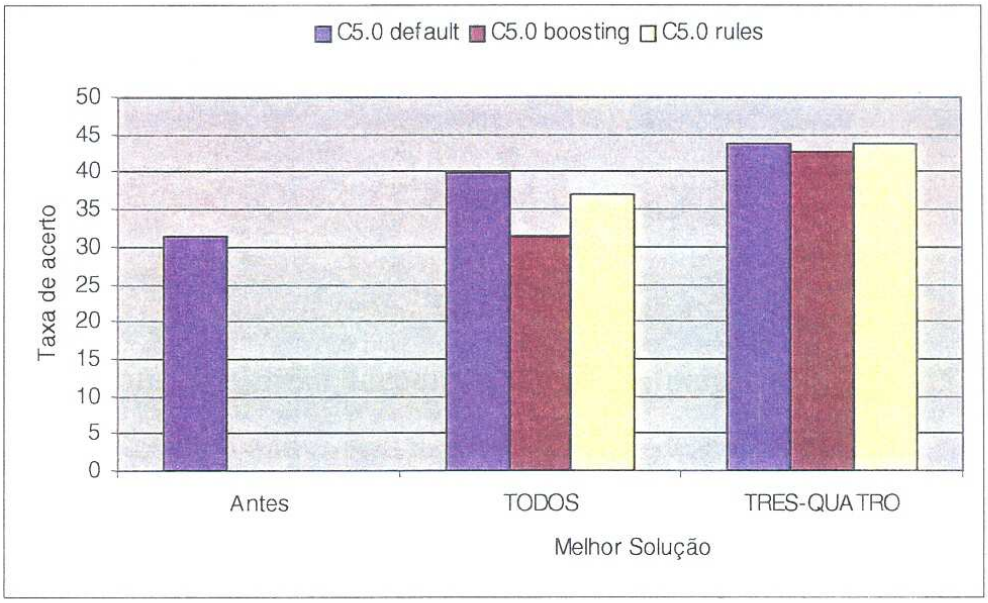
Sumarizando, o uso do processo *wrapper* proposto netas dissertação utilizando a taxa de acerto sensível ao desbalanceamento de classes como função objetivo pode ser considerado como bastante benéfico. De fato, o uso do processo *wrapper* aumentou significativamente a taxa de acerto para as versões do C5.0 utilizadas nos experimentos (C5.0 *default*, C5.0 *boosting* e C5.0 *rules*).

6.2.3 Comparação dos Resultados Obtidos com Relação ao Critério de Repetição de Exemplos

No que se refere ao fato da abordagem *wrapper* permitir a repetição de todos os exemplos da base (grupo TODOS) ou restringir que a repetição se dê apenas sobre os exemplos pertencentes às classes raras (grupo TRES-QUATRO), será comentado a seguir.

Se o processo evolutivo utilizar o valor do erro médio como sendo a função objetivo, conforme o gráfico 6.5, os resultados são melhores se todos os exemplos puderem ser replicados. A diminuição da média do valor do erro médio é da ordem de 5% nos casos do C5.0 *default* e do C5.0 *boosting* e de 2% para o C5.0 *rules*.[43]

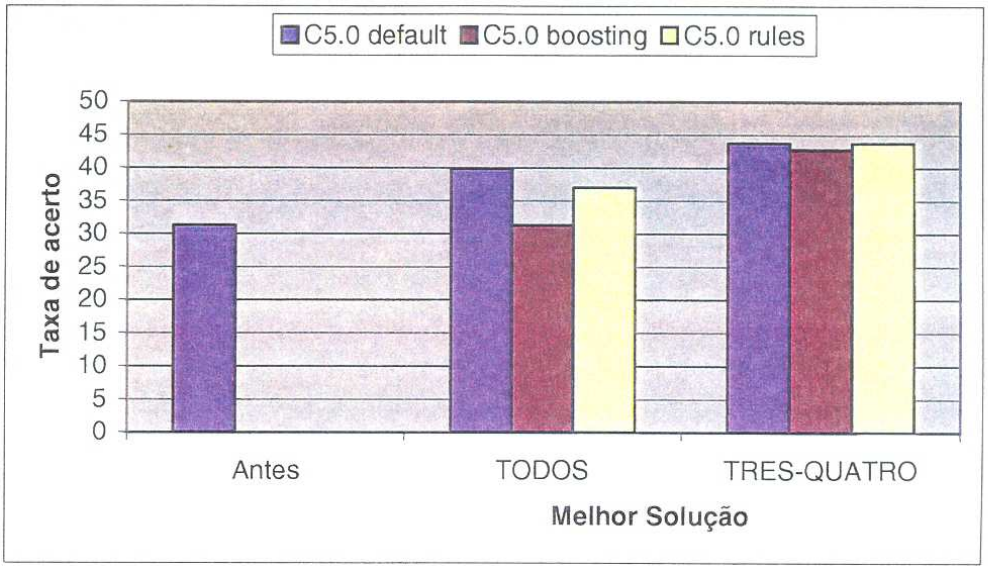
GRÁFICO 6.5 - RESULTADOS ANTERIORES E APÓS A EXPANSÃO DA BDR SEGUNDO FUNÇÃO DE FITNESS ERRO MÉDIO E CRITÉRIO DE REPETIÇÃO DE EXEMPLOS (MEDIDOS SOBRE OS DADOS DE TESTE)



Se o processo evolutivo utilizar a taxa de acerto sensível ao desbalanceamento de classes como sendo a função objetivo, os resultados são melhores, conforme o gráfico 6.6, se apenas os exemplos das classes raras puderem ser replicados. O aumento da taxa média de acerto sensível ao desbalanceamento é da ordem de 9,8%, 36,4% e de 18,1% para o C5.0 *default*, o C5.0 *boosting* e o C5.0 *rules*, respectivamente.

[43] Percentuais calculados a partir das colunas de médias apresentadas na Tabela 6.11 e na Tabela 6.12.

GRÁFICO 6.6 - RESULTADOS ANTERIORES E APÓS A EXPANSÃO DA BDR SEGUNDO FUNÇÃO DE FITNESS TAXA DE ACERTO SENSÍVEL AO DESBALANCEAMENTO DE CLASSES E CRITÉRIO DE REPETIÇÃO DE EXEMPLOS (MEDIDOS SOBRE OS DADOS DE TESTE)

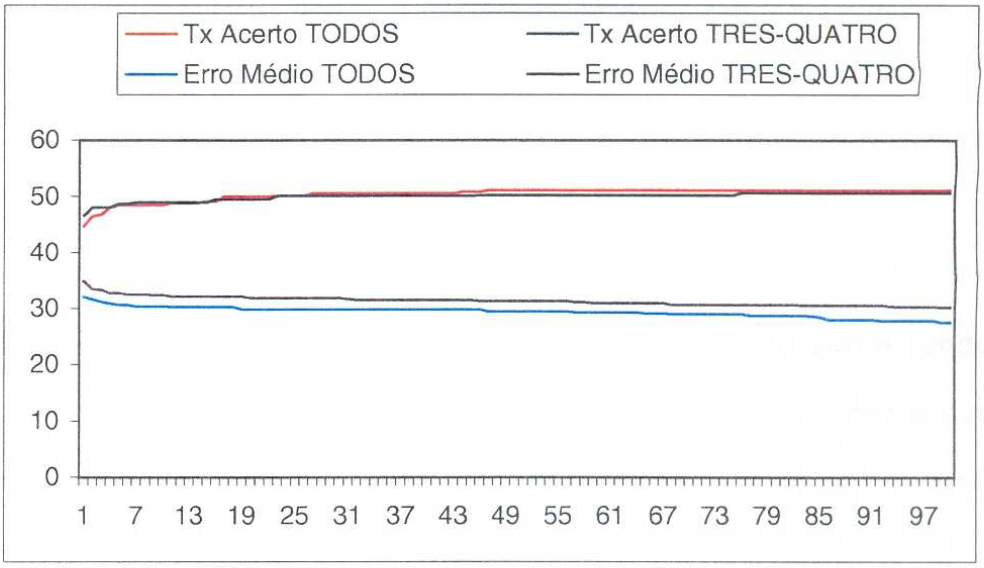


O aumento médio tão acentuado de 36,4% para o caso do C5.0 *boosting*, para os experimentos com a taxa de acerto sensível ao desbalanceamento de classes e a restrição de que apenas os exemplos das classes raras pudessem ser replicados, ocorreu devido ao fato de que, no experimento 6 correspondente (tabela 6.11), a taxa ter colapsado em 0%. O resultado tão baixo para este experimento forçou a média geral deste grupo de experimentos para baixo.

Conforme já foi colocado anteriormente quando da descrição da tabela 6.6, as taxas de acerto sensíveis aos desbalanceamento de classes para as versões *boosting* e *rules* são zeradas, por isso da não existência das respectivas colunas no gráfico 6.6.

Ainda sobre a comparação dos diferentes critérios de evolução adotados no processo *wrapper,* o gráfico 6.7 mostra as 4 curvas evolutivas. O gráfico 6.7 complementa as tabelas 6.7 e 6.10, porém trabalha com a média, geração a geração, enquanto estas duas tabelas trabalham com a média geral.

GRÁFICO 6.7 CURVAS DE EVOLUÇÃO SEGUNDO FUNÇÃO DE FITNESS E CRITÉRIO DE REPETIÇÃO DE EXEMPLOS (MEDIDOS SOBRE OS DADOS DE VALIDAÇÃO)



Este gráfico (gráfico 6.7) mostra duas curvas superiores correspondentes a evolução pela taxa de acerto sensível ao debalanceamento de classes e duas curvas inferiores correspondentes a evolução pelo erro médio como funções objetivo. Quando a função objetivo é a taxa de acerto é possível perceber que a estabilização ocorre mais cedo se o critério de repetição permite que TODOS os exemplos sejam repetidos. Já para o caso do erro médio como função objetivo, a estabilização ocorre próximo do término do processo evolutivo, independentemente do critério de repetição dos exemplos.

7 CONCLUSÃO

7.1 COMPARAÇÃO DOS ALGORITMOS CN2, C4.5 E C5.0

A comparação da execução dos algoritmos CN2, C4.5 e C5.0 confirma o que já foi dito: "não existe um algoritmo que possa ser definido como sendo o melhor algoritmo independentemente do domínio e da base de dados sobre a qual se esteja processando". Porém, os resultados dos experimentos realizados neste trabalho demonstram que a adoção da opção *boosting*, disponível apenas para o algoritmo C5.0, melhora, em média, as taxas de erro se comparado com os resultados obtidos através do CN2 e C4.5 em suas diferentes versões.

Ao se analisar os experimentos realizados com versões similares dos três algoritmos, ou seja, se forem excluídos os resultados do C5.0 com opção de *boosting*, é possível concluir que ainda assim o C5.0 continua obtendo em média os melhores resultados, porém na sua opção *rules*.

Conforme o que já foi divulgado sobre o C5.0 (QUINLAN, 1997), no desenvolvimento deste algoritmo houve uma grande preocupação com relação ao tempo de processamento. O C5.0 chega a ter uma redução do tempo despendido em torno de 74% em relação ao C4.5 e em torno de 95% em relação ao CN2.

Deve ser ressaltado que a avaliação de desempenho dos algoritmos é uma tarefa a ser recorrentemente realizada no âmbito da temática da inteligência artificial. A busca de simplicidade das regras e de eficiência na geração de regras tem motivos não só econômicos (redução do tempo de processamento, por exemplo), mas também é de fundamental importância para a difusão e ampliação do uso de técnicas de geração de conhecimentos por meio de máquinas.

Se, como já foi dito, "não existe um algoritmo que possa ser definido como sendo o melhor algoritmo independentemente do domínio e da base de dados sobre a qual se esteja processando", isso não descarta a possibilidade de que a avaliação

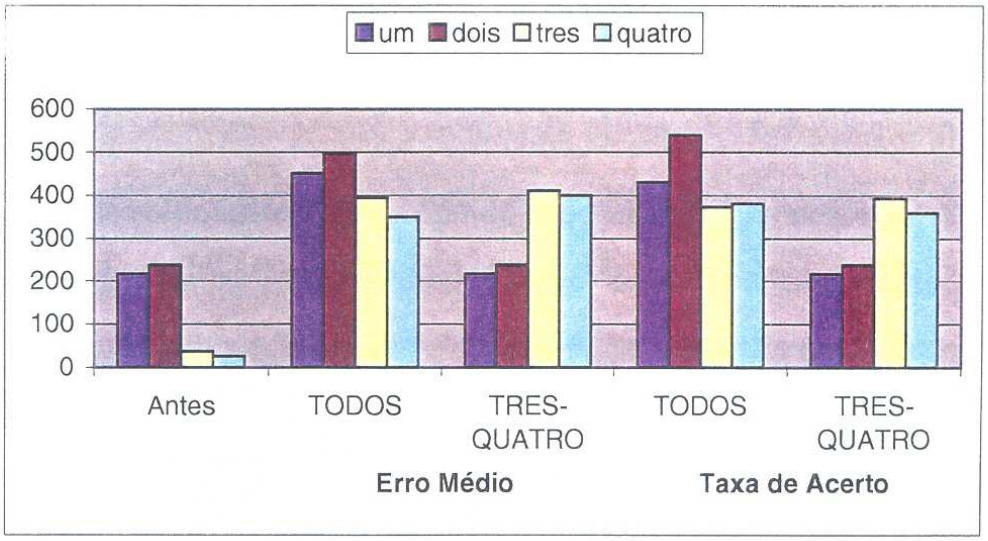
sistemática de desempenho dos algoritmos venha a permitir cada vez mais o mapeamento de afinidades entre grupos de algoritmos e de domínios, de forma a permitir indicações mais precisas das possibilidades genéricas de uso da cada um desses grupos.

7.2 UMA NOVA SOLUÇÃO PARA O PROBLEMA DE DESBALANCEAMENTO DE CLASSES EM UMA BASE DE DADOS DO MUNDO REAL

Concluindo, a expansão do conjunto de treinamento, i.e. a repetição seletiva de alguns dos exemplos de treinamento, obtida através da abordagem *wrapper* proposta e implementada neste trabalho, pode ser considerada uma solução promissora para o problema de desbalanceamento de classes. Os experimentos computacionais demonstram que esta abordagem – processo híbrido envolvendo algoritmo genético e árvore de decisão (C5.0) – obteve resultados que são significativamente melhores que os resultados obtidos pelo C5.0 sobre a base original, antes do processo de expansão.

Observando o gráfico 6.8 pode-se concluir que mesmo após a expansão permitindo que todos exemplos sejam repetidos, as classes "um" e "dois" continuam apresentando uma maior freqüência se comparadas com as classes mais raras ("tres" e "quatro").

GRÁFICO 6.8 - DISTRIBUIÇÃO DAS CLASSES ANTES E DEPOIS DA EXPANSÃO DA BDR SEGUNDO A FUNÇÃO DE FITNESS E O CRITÉRIO DE REPETIÇÃO DOS EXEMPLOS



A desvantagem desta abordagem híbrida AG/C5.0 é com relação ao tempo de processamento despendido. Ela é muito mais lenta se comparada à execução do C5.0 sobre a base original, que é da ordem de poucos segundos. Entretanto, para pequenas bases de dados esta questão de tempo é perfeitamente aceitável. Especificamente no caso dos experimentos apresentados neste trabalho, o tempo de execução de cada uma das evoluções do AG foi em torno de 12 horas (permitindo que TODOS os exemplos fossem repetidos) e 6 horas (permitindo que apenas os exemplos das classes TRES-QUATRO)[44].

É preciso considerar que, no caso deste domínio de aplicação, os dados são de natureza estática, uma vez que novos dados semelhantes são coletados somente a cada 10 anos, quando da realização de um novo Censo Demográfico. Sendo assim, este tempo demandado pode ser perfeitamente aceitável. Ou seja, um tempo de processamento de alguns dias para um processo de *Data Mining* pode ser aceitável se este processo conduzir a uma melhoria na precisão preditiva das regras descobertas.

Finalmente, apesar dos experimentos deste trabalho enfocarem os dados de Censo, acredita-se que este método híbrido envolvendo algoritmo genético e árvore de decisão seja suficientemente genérico para ser aplicado eficazmente sobre outras bases de dados.

7.3 TRABALHOS FUTUROS

Da discussão acima decorre a necessidade, no futuro, de avançar com a análise aqui realizada através das seguintes três alternativas:

a) repetir os experimentos para outras bases de dados, particularmente para bases com diferentes proporções de desbalanceamento entre as classes;
b) substituir o algoritmo adotado como "caixa preta" dentro processo *wrapper* – o algoritmo de árvore de decisão (C5.0) - por um outro algoritmo;

---

[44]Maquina : Ultra Sparc com 256Mb de RAM, 166 de clock, 90Gbytes de disco SCSI rodando Unix C5.0 versão 1.6.

c) repetir os experimentos não apenas com uma única partição de dados de treinamento e de teste, mas sim várias destas partições a partir da técnica *cross-validation.*

E como essas alternativas não são mutuamente excludentes e, portanto, podem ser combinadas, assim, abre-se a possibilidade de avaliar não só os resultados passíveis de serem obtidos em cada uma delas, *ceteris paribus*, mas também os de suas diferentes formas de combinação.

APÊNDICE A - TAREFAS DE DATA MINING

O objetivo de descrição, bem como o da predição, são atendidos através de algumas das tarefas principais do Data Mining (classificação, regressão, *clustering*, caracterização, regras de associação, evolução e *deviation mining*), que são descritas a seguir (ADRIAANS, 1996), (FAYYAD, 1996) e (FU, 1996).

Classificação

A classificação é a tarefa de Data Mining que tem sido mais estudada ao longo do tempo. A classificação é uma função de aprendizado de máquina que classifica um item de dado como pertencente a uma determinada classe dentre várias classes previamente definidas. O principal objetivo é descobrir algum tipo de relação entre os atributos[45] e as classes.[46] Um conjunto de dados de treinamento é disponibilizado e analisado, e um modelo de classificação é construído baseado sobre as características destes mesmos dados. Um conjunto de regras de classificação é gerado a partir do modelo de classificação, o qual poderá ser utilizado posteriormente para classificar outros dados.

Um exemplo bastante simples de um problema de classificação pode ser o de uma editora que, ao publicar um livro "X" e possuindo uma base de dados de seus clientes, deseja identificar quais destes constituem clientes em potencial para aquisição deste livro (FREITAS, 1998).

Para prever se o cliente vai ou não atender ao encarte de divulgação e comprar o livro, a editora necessita de alguns dados sobre a resposta de seus clientes cadastrados ao seu material de marketing. A partir destes dados, o algoritmo de classificação pode descobrir regras que auxiliem na predição se um cliente deverá comprar ou não o livro como resposta ao recebimento do material de divulgação.

A tabela A.1 mostra os valores de alguns atributos previsores e do atributo meta (Sim/Não) correspondente aos clientes que compram ou não o livro após o recebimento do material de divulgação da editora.

---

[45] Os atributos podem também ser chamados de atributos previsores (FREITAS, 1998).

[46] As classes podem também ser chamadas de atributos meta (FREITAS, 1998).

O algoritmo de classificação analisa os dados da tabela A.1 e determina os valores dos atributos que estão associados com cada valor correspondente às classes (Sim ou Não).

TABELA A.1 - DADOS DE ENTRADA PARA A TAREFA DE CLASSIFICAÇÃO

SEXO	PAÍS	IDADE	OBJETIVO COMPRAR
M	França	25	Sim
M	Inglaterra	21	Sim
F	França	23	Sim
F	Inglaterra	34	Sim
F	França	30	Não
M	Alemanha	21	Não
M	Alemanha	20	Não
F	Alemanha	18	Não
F	França	34	Não
M	França	55	Não

O conhecimento descoberto é geralmente representado na forma de regras "se.... então....", onde a interpretação é: "se os valores dos atributos satisfazem a condição da regra então a tupla pertence à classe prevista pela regra" (exemplo A.1).

EXEMPLO A.1 - REGRAS DE CLASSIFICAÇÃO OBTIDAS ATRAVÉS DOS DADOS DA TABELA A.1 (FREITAS, 1998)

se (País = "Alemanha") então (Compra = "Não")

se (País = "Inglaterra") então (Compra = "Sim")

se (País = "França" e Idade <=25) então (Compra = "Sim")

se (País = "França" e Idade >25) então (Compra = "Não")

Existem critérios para avaliar a qualidade das regras descobertas na tarefa de classificação, como por exemplo a precisão preditiva e a compreensibilidade do conhecimento descoberto. Estes critérios e outras características da tarefa de classificação são tratados no capítulo 4, dado que um dos principais objetivos deste trabalho é comparar algoritmos de classificação.

Regressão

A regressão é também uma função de aprendizado de máquina, que mapeia o dado em uma variável de predição.

Trata-se de uma tarefa conceitualmente similar à de classificação, onde a principal diferença é que na regressão o atributo a ser predito é contínuo. Isto é, ele pode representar valores reais ou inteiros ao invés de valores discretos, como na classificação.

A predição de valores numéricos pode ser obtida por métodos tradicionais de estatística, tal como a regressão linear. Estes métodos numéricos são baseados no conceito de otimização *theoretically-sound*, tal como encontrar o valor mínimo de uma função de erro. Entretanto, a regressão pode ser executada também através de métodos ditos de "alto nível" (ou simbólicos) (FREITAS, 1998).

Tanto a classificação como a regressão partem da análise e a generalização de exemplos de experiências passadas para prever o futuro. No caso da classificação, a resposta do processo geralmente é do tipo "verdadeiro ou falso", "atende ou não atende" às características de uma determinada classe. Em alguns casos, a resposta pode ser escolhida dentre mais de 2 (dois) valores, mais ainda, dentre um pequeno número de valores discretos (classes), tais como "bom, razoável, crítico e muito crítico". No caso da regressão, a resposta é um número, o valor da função de regressão, o qual é um valor contínuo, ao invés de discreto.

Clustering

O *clustering,* também chamado de classificação não-supervisionada (CHEN, 1996), é a identificação de um conjunto finito de classes ou *clusters*[47], baseada nos atributos dos objetos não previamente classificados. Os objetos são agrupados de tal forma que as similaridades intraclasse são maximizadas e as similaridades extraclasse (ou inter-classe) são minimizadas com base em alguns critérios. Uma vez definidos os *clusters*, os objetos são identificados com seu *cluster* correspondente e as características comuns dos objetos no *cluster* são sumarizadas para formar a descrição da classe. Por

[47]Um conjunto de objetos agrupados em função de sua similaridade ou proximidade.

exemplo, um conjunto de novas doenças pode ser agrupado em várias categorias baseado nas similaridades dos seus sintomas e os sintomas comuns das doenças da categoria podem ser usados para descrever a qual grupo de doenças pertencem.

Um método de *clustering* é o *conceptual clustering,* que foi primeiramente apresentado por Michalski e Steep (FU, 1996). Neste método, dado um conjunto de objetos, são identificadas as classes, tomando como referência o princípio de "proximidade conceitual" entre os objetos. Entende-se como objeto, uma tupla de aridade *n,* representada por um vetor de valores em um conjunto de *n* atributos do domínio. Desta forma, uma classe pode ser definida como sendo uma conjunção de predicados sobre estes atributos e valores.

No algoritmo Cluster/2, apresentado por Michalski e Steep em 1983 (LUGER, 1992), um pequeno número de objetos aleatórios, chamados de *seeds* (sementes), é selecionado como representativo de cada classe e a descrição da classe é feita com base nestes *seeds*. Em processo, outros objetos são introduzidos nas classes baseado no conceito de proximidade. Um *clustering* hierárquico pode ser construído pela divisão das classes em subclasses. A busca pela melhor classe, baseando-se em alguns critérios de *clustering*, é interrompida quando um ponto máximo é encontrado.

Os critérios de *clustering* têm as seguintes características:

- *fitness* – é a razão entre o número de objetos de uma classe e o espaço coberto pela classe;
- complexidade – medida pelo número de classes e o número de predicados em cada conjunção;
- cobertura – o número de objetos cobertos pela classe; e
- *disjointedness* – medida pelo número de objetos que são cobertos por mais de uma classe.

Um outro algoritmo foi apresentado por D. Fisher na AAAI Conference de 1987 (FU, 1996), o COBWEB, onde uma árvore de *clustering* é construída de tal forma que cada nó da árvore representa uma classe e cada subárvore representa as respectivas subclasses. Inicialmente, a árvore de *clustering* tem apenas um nó, o nó raiz, a partir do qual novos objetos são adicionados e ajustados à árvore. O algoritmo

COBWEB usa a medida *category utility,* que contribui para o aumento do número de objetos na classe estimada corretamente. As operações básicas executadas de acordo com a medida *category utility* são:

- adição de um objeto em uma classe já existente;
- criação de uma nova classe para posicionar o novo objeto;
- subdividir a classe em duas novas subclasses; ou
- realizar um *merge* de duas classes em uma única classe.

Caracterização

O objetivo desta tarefa é produzir uma descrição das características de cada uma das classes das tuplas no conjunto de dados. Ou seja, é a sumarização dos valores dos atributos das tuplas que compõem uma determinada classe. Esta descrição pode ser expressa na forma de regras "se.... então ....", que podem ser lidas como: se a tupla pertence a determinada classe (antecedente da regra) então a tupla atende a todas as propriedades mencionadas (conseqüente da regra) (FREITAS, 1998).

É importante observar que na tarefa de caracterização a classe pertence ao antecedente da regra, enquanto que na tarefa de classificação a classe pertence ao conseqüente da regra.

Um exemplo de caracterização pode ser obtido na identificação dos sintomas de uma doença específica, onde estes sintomas podem ser sumarizados através de um conjunto de regras características.

Regras de Associação

Estas regras se referem à descoberta de associações ou conexões entre objetos. A sintaxe de uma regra de associação obedece a seguinte forma:

$$A_1 \wedge .... \wedge A_i \rightarrow B_1 \wedge .... \wedge B_j$$

onde os objetos $B_1 \wedge .... \wedge B_j$ estão relacionados com os objetos $A_1 \wedge .... \wedge A_i$.

As regras de associação revelam quais são as relações relevantes dos objetos no conjunto de dados. Por exemplo, um conjunto de sintomas freqüentemente ocorre juntamente com um outro conjunto de sintomas.

Considerando, por exemplo, os dados de um supermercado, onde as compras de cada cliente estejam registradas por transação de compra, o departamento de planejamento pode, utilizando esta abordagem, identificar associações entre conjuntos de itens consumidos e continuidade de consumo dos clientes. Uma possível resposta seria que 90% das transações registradas para pão e manteiga são também acompanhadas de leite. Neste caso, é dito que o antecedente da regra é o pão e a manteiga, que o conseqüente é o leite e que o fator de confiança da regra é de 90%.

Neste caso, a partir do uso de regras de associação, várias questões podem ser levantadas:

- encontrar todas as regras que tem o produto "X" como conseqüente, procurando identificar os fatores que podem elevar a sua venda;
- encontrar todas as regras que tem o produto "Y" como antecedente, onde é possível antever o que ocorrerá se este "Y" deixar de constar nas prateleiras.

Evolução

Esta tarefa tem por objetivo a identificação de regularidades e suas respectivas evoluções para determinados objetos que apresentam modificações no comportamento ao longo do tempo. O desenvolvimento desta tarefa envolve outras tarefas, tais como: a caracterização, a classificação, as regras de associação e o *clustering* dos dados relacionados ao tempo.

Pode-se citar como exemplo de evolução a identificação de características de companhias que tenham aumentado acima de 10% seus preços de mercado ao longo dos últimos dois anos.

Deviation Mining

O *deviation mining* é a descoberta e a avaliação de desvios dos valores "padrões" dos atributos, para um certo grupo de objetos. Estes desvios ocorrem entre o valor atual do conjunto de dados e a sua respectiva expectativa de comportamento. Estas expectativas podem ser obtidas através de alguns pressupostos, tais como média ou crescimento linear.

Os dois principais tipos de desvios que ocorrem são (FREITAS, 1998):

- os desvios temporais – referem-se a descoberta das mudanças mais significativas nos dados previamente analisados, ou mesmo nos valores normativos, em relação a dimensão de tempo;
- os desvios de grupos – referem-se a diferenças entre os dados em dois subconjuntos de dados.

Um exemplo de *deviation* pode ser a avaliação de um conjunto de dados de movimentação de vendas de uma determina empresa. Esta avaliação pode identificar alguma forma de desvio no comportamento, dada a tendência de movimentação de vendas da maioria de outras empresas no mesmo segmento de mercado, durante certo período do tempo.

A partir desta tarefa também é possível detectar a ocorrência de erros nos dados ou desvios a partir de valores normativos.

APÊNDICE B - DEFINIÇÕES

Conjunto de Treinamento

O conjunto de treinamento é o conjunto de exemplos utilizado na fase de construção do modelo e contém as informações sobre o domínio a ser processado, ou seja, o conjunto de experiências passadas.

Definição B.1 Seja $A = \{A_1, ..., A_n\}$ o conjunto de atributos com domínios $Dom_1, ..., Dom_n$. Um conjunto de treinamento é uma tabela sobre A. Um exemplo ou uma observação é uma tupla no conjunto de treinamento. O universo U contem todas as relações sobre A, isto é, $U = Dom_1 \times .... \times Dom_n$. Assim sendo, cada conjunto de treinamento é um subconjunto finito de U[48].

A partir das informações contempladas pela tabela A.1 (dados de entrada para a tarefa de classificação), onde o primeiro atributo descreve o sexo do cliente, pode-se exemplificar o domínio como sendo $Dom_1 = \{M, F\}$. O segundo atributo da mesma tabela é país e o seu domínio pode ser dado por $Dom_2 = \{$Alemanha, França, Inglaterra$\}$. O terceiro atributo é idade, cujo domínio, Dom_3 é o conjunto dos números inteiros positivos. O quarto atributo é a condição se compra ou não: $Dom_4 = \{$Sim, Não$\}$.

Os exemplos do domínio são apresentados por tuplas do conjunto de treinamento que representam as propriedades dos exemplos e não as suas relações. É de fundamental importância que o conjunto de treinamento possua dados suficientes para os procedimentos de generalização indutiva, conforme foi ressaltado no capítulo 2 (exemplo 2.1 – obrigatoriedade do voto).

O volume de dados requerido para o conjunto de treinamento depende do número de atributos previsores e de classes, bem como da complexidade do modelo, dado que o processo de generalização ocorre através de procedimentos estatísticos, que necessitam de um grande número de exemplos para que sejam eficazes. Um modelo simples pode ser construído a partir de um pequeno conjunto de dados, mas um

---

[48]Conforme Holsheimer (HOLSHEIMER, 1994).

modelo de classificação detalhado requer centenas e, algumas vezes, milhares de exemplos para a sua respectiva construção.

Classes

As bases de dados determinam o ambiente a partir do qual o algoritmo constrói um modelo, segundo os atributos previsores e suas respectivas classes, também denominadas de valores do atributo meta.

Definição B.2 Uma classe C_i é um subconjunto do conjunto de treinamento S, constituído de todos os exemplos que satisfazem a descrição que determina a classe:

$C_i = \{ o \in S \mid conds(o)\}$[49]

As observações que satisfazem a descrição são ditas exemplos positivos ou instâncias da classe C_i. Por outro lado, as observações que não atendem à descrição são ditas exemplos negativos da classe C_i.

Regras

Ao construir o modelo, o algoritmo deve inferir as regras que determinam a classificação, isto é, devem ser identificadas as definições ou as descrições dos conceitos de cada classe (HOLSHEIMER, 1994).

Definição B.3 Descrição de conceitos – Dado A como sendo o conjunto dos atributos previsores, uma descrição elementar de conceitos é formulada como sendo $A_1 = c_1 \wedge ... \wedge A_n = c_n$ tal que[50]:

1. $A_i \in A$ e $i \neq j \rightarrow A_i \neq A_j$;
2. $C_i \in Dom\ (A_i)$

---

[49] Conforme Holsheimer(HOLSHEIMER, 1994).

[50] Conforme Holsheimer (HOLSHEIMER, 1994).

O conjunto de regras pode ser usado para descrever a relação entre os conceitos (ou classes) e as propriedades (ou atributos previsores). O "corpo" da regra é o resultado produzido a partir dos algoritmos de indução.

Um conjunto de regras consiste de uma coleção de declarações do tipo "se... então ..." , que é chamada de produção, regras de produção ou simplesmente regras.

Definição B.4 Regra – Uma regra de classificação consiste de uma descrição *D* e de um símbolo de classe *C* tal que:[51]

$\forall x \in U : x \in \sigma_D (S) \rightarrow x \in C$

É importante que as regras sejam acompanhadas de suas respectivas situações relativas à sua precisão (ou confiança) e a sua cobertura.

A precisão informa o quanto a regra é correta: **"se o antecedente é verdadeiro, então o conseqüente será verdadeiro"**.

Definição B.5 Precisão de classificação – é a probabilidade de uma regra classificar corretamente, ou seja, é a probabilidade de um objeto coberto pela descrição realmente ser pertencente à respectiva classe. A precisão de classificação é a porção relativa de σ_D (S) que é também coberta pela classe *C* [52]:

$$\text{Precisão de classificação} = \frac{|\sigma_D (s) \cap C|}{|\sigma_D (S)|}$$

Uma alta precisão indica uma regra altamente dependente entre o antecedente e o conseqüente da regra.

Já uma alta cobertura significa que a regra pode ser usada freqüentemente e que não se trata de uma exceção no conjunto de treinamento.

Definição B.6 A cobertura de uma descrição é a probabilidade de um objeto qualquer, pertencente a uma classe *C*, estar coberto pela descrição *D*, de tal forma que:
Cobertura = ($|\sigma_D (S) \cap C|$) / $|C|$ [53]

---

[51] Conforme Holsheimer (HOLSHEIMER, 1994)

[52] Conforme Holsheimer (HOLSHEIMER, 1994)

[53] Conforme Holsheimer (HOLSHEIMER, 1994).

Para avaliar uma regra, pode-se verificar não apenas a sua precisão e cobertura isoladamente, mas também a sua relação de precisão *versus* cobertura. A comparação entre a cobertura de uma regra e sua respectiva precisão pode ser observada através da tabela B.1 (BERSON, 1997).

TABELA B.1 - COBERTURA DA REGRA *VERSUS* PRECISÃO

COBERTURA	BAIXA PRECISÃO	ALTA PRECISÃO
Alta Cobertura	A regra raramente é correta, mas pode ser usada freqüentemente	A regra freqüentemente é correta e pode ser usada freqüentemente
Baixa Cobertura	A regra raramente é correta e raramente é usada	A regra freqüentemente é correta e raramente é usada

Com base nas definições anteriores, é possível distinguir algumas características especiais das regras, como a sua completeza e a sua consistência.

Definição B.7 Uma regra é dita completa se a sua respectiva cobertura é igual a 1(um), ou seja, qualquer exemplo pertencente à classe prevista pela regra é coberto pela respectiva definição, isto é, $C \subseteq \sigma_D (S)$[54].

O grau de completeza de uma regra é dado através da divisão do número de exemplos da classe cobertos pela regra pelo número total de exemplos da classe.

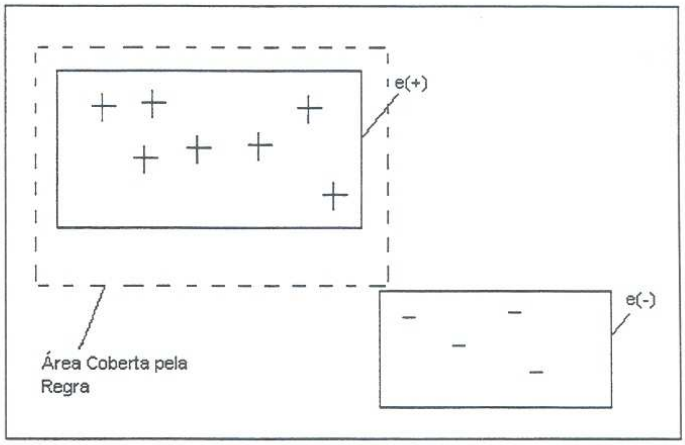
Definição B.8 Uma regra é dita consistente se a sua precisão é igual a 1(um), ou seja, ela sempre classifica corretamente. Qualquer exemplo coberto pela descrição pertence à classe $C \supseteq \sigma_D (S)$[55].

A figura B.1 (MONARD, 1997) mostra graficamente uma regra completa e consistente.

[54] Conforme Holsheimer (HOLSHEIMER, 1994).

[55] Conforme Holsheimer (HOLSHEIMER, 1994).

FIGURA B.1 - UMA REGRA COMPLETA E CONSISTENTE



A partir de sua precisão e cobertura, é possível também avaliar se uma regra pode ser dita correta ou não.

Definição B.9 Uma regra é dita correta se tanto a precisão quanto a cobertura forem iguais a 1(um)[56].

A situação ideal é a que o conjunto de regras induzido contemple todos os casos positivos e nenhum caso negativo (figura 4.1).

Classificador

Dado um conjunto de treinamento $S=\{<X_1, f(X_1)>, <X_2, f(X_2)>,..., <X_m, f(X_m)>\}$, um algoritmo de classificação deve produzir um classificador, ou seja, uma hipótese sobre os valores verdadeiros da função f. Ao serem fornecidos novos valores para x, o classificador prediz os correspondentes valores de $f(x)$.

Definição B.10 Um classificador é uma função $f(x)$ definida sobre S, de forma que, para cada x, $f(x)$ seja igual a um dos valores do domínio pré-definido para $f(x)$[57].

---

[56] Conforme Holsheimer (HOLSHEIMER, 1994).

[57] Conforme Breiman (BREIMAN, 1984).

REFERÊNCIAS BIBLIOGRÁFICAS

(ADRIAANS, 1996) Adriaans, Pieter; Zabtinge, Dolf, **DATA MINING**, Addison Wesley Longman - 1996, England.

(ANDRADE, 1998) Andrade, Thompson A.; Serra, Rodrigo Valente, **O Recente Desempenho das Cidades Médias no Crescimento Populacional Urbano Brasileiro**, IPEA-Instituto de Pesquisa Econômica Aplicada, Rio de Janeiro. Março de 1998.

(ARARIBOIA, 1988) Araribóia G., **Inteligência Artificial Um Curso Prático.** Rio de Janeiro: LTC, 1988.

(BERRY, 1997) Berry, Michael J. A.; Linoff, Gordon, **Data Mining Techniques: for marketing, sales, and customer support**, John Wiley &Sons, Inc., USA, 1997.

(BERSON, 1997) Berson Alex, Smith Stephen J., **Data Warehousing, Data Mining, and OLAP,** McGraw-Hill, USA. 1997.

(BIGUS, 1996) Bigus, Joseph P., **Data Mining with Neural Networks,** McGraw-Hill, USA. 1996.

(BREIMAN, 1984) Breiman, Leo; Friedman, Jerome H.; Olshen, Richard A.; Stone, Charles J., **Classification and Regression Trees,** Wadsworth & Brooks / Cole Advanced Books & Software, California. 1984.

(BUNTINE, 1992) Buntine, Wray, **Learning Classification Trees**, Statistics and Computing journal, Vol 2,. 1992.

(CASTIÑEIRA, 1990) Castiñeira, Maria I., **Aprendizado de Máquina por Exemplos usando Árvores de Decisão**, USP – São Carlos, Setembro de 1990.

(CHAN,1998) Chan, Philip K; Stolfo, Salvatore J., **Toward Scalable Learning with Non-uniform Class and Cost Distributions: A Case Study in Credit Card Fraud Detection,** KDD98 – Prod. Of the Fourth International Conference on Knowledge Discovery and Data Mining. AAAI Press, USA. 1998.

(CHEN, 1996) Chen, Ming-Syan; Han, Jiawei; Yu, Philip S., **Data Mining: An Overview from Database Perspective,** IEER Transactions on KDE, Vol 8, N. 6, December 1996.

(CLARK, 1989) Clark, Peter; Niblet, Tim, **The CN2 Induction Algorithm,** Machine Learning Journal, 3 (4), pp261-283, Netherlands : Kluwer, 1989. http://www.cs.utexas.edu/users/pclark/papers/cn2.ps 01/12/1997

(CLARK, 1991) Clark, Peter; Boswell, Robin, **Rule Induction with CN2: Some Recent Improvements,** Proceedings of the Fifth European Conference (EWSL-91), pp151-163, Ed: Yves Kodratoff, Berlin: Spriner-Verlag, 1991. http://www.cs.utexas.edu/users/pclark/papers/newcn.ps 01/12/1997

(CORMEN, 1996) Cormen, Thomas H; Leiserson, Charles E; Rivest, Ronald L, **Introduction to Algorithms,** McGraw-Hill, USA. 1996.

(DECKER, 1995) Decker, Karsten M.; Focardi, Sérgio, **Technology Overview : A Report on Data Mining**, CSCS TR-95-02, Swiss Scientific Computer Center, 1995.

(DOMINGOS, 1998) Domingos, Pedro, **Occam's Razors: The Sharp and the Blunt,** KDD98 – Prod. Of the Fourth International Conference on Knowledge Discovery and Data Mining. AAAI Press, USA. 1998.

(FAYYAD, 1996) Fayyad, Usama M; Piatetsky-Shapiro, Gregory; Smyth, Padhraic; Uthurusamy, Ramasamy. **Advances in Knowledge Discovery and Data Mining.** USA: American Association for Artificial Intelligence- 1996.

(FAYYAD, 1998) Fayyad, Usama, **Mining Databases: Towards Algorithms for Knowledge Discovery,** IEEE Computer Society, Bulletin of Technical Committee on Data Engineering, March 1998.

(FREITAS, 1998) Freitas, Alex A; Lavington, Simon H., **Mining Very Large Databases with Parallel Processing,** Kluwer Academic Publishers, USA 1998.

(FREITAS, 1998b) Freitas, Alex A, **Data Mining (Mineração de Dados)** Tutorial apresentado em maio/98 na PUCPR, Curitiba 1998.

(FU, 1996) FU Yongjian, **Discovery of Multiple-Level Rules from Large Databases,** Thesis of Doctor of Philosophy, Simon Fraser University, 1996.

(GOLDBERG, 1989) Goldberg, David E., **Genetic Algorithms in Search Optimization 7 Machine Learning,** Addison Wesley Longman, Inc.. USA. 1989.

(GLYMOUR, 1997) Glymour Clark, **Statistical Themes and Lessons for Data Mining.** Data Mining and Knowledge Discovery Vol 1, N. 1, March 1997.

(HAN, 1995) Han, Jiawei, **Mining Knowledge at Multiple Concept Levels,** Tutorial on Information and Knowledge Management, Maryland. 1995.

(HART, 1989) Hart, Anna, **Knowledge Acquisition for Expert Systems,** Billing and Son Ltd., Worcester, 1989

(HOLSHEIMER, 1994) Holsheimer Marcel; Siebes, Arno, **Data Mining The Search for Knowledge in Databases**, CS-R9406 1994.

(IPARDES, 1993) IPARDES, **Indicadores Analíticos,** Volume I, Referencial Urbano, Curitiba. 1993.

(JOHNSON, 1978) Johnson, R.,A.; Wichern, D.W., **Applied Multivariate Statistical Analysis,** Prentice Hall International Inc., Madison. 1978.

(KAMBER, 1997) Kamber, Micheline; Winstone, Lara; Gong, Wan; Chemg, Shan; Han, Jiawei. **Generalization and Decision Tree Induction: Efficient Classification in Data Mining.** Proc. of 1997, Workshop on Research Issues on Data Engineering (RIDE 97), Birmingham, England. 1997

(KOHAVI, 1996) Kohavi, Ron; Sommerfield, Dan; Dougherty, James, **Data Mining using *MLC++* A Machine Learning Library in C++.** Appeared in Tool With AI 1996. http://www.sgi.com/Technology/mlc 1996.

(KOLODNER, 1993) Kolodner, Janet L., **Case-Based Reasoning**, Morgan Kaufmann Publishers, Inc. USA. 1993.

(KONONENKO, 1991) Kononenko, Igor; Bratko, Ivan, **Information-Based Evaluation Criterion for Classifier's Performance,** Machine Learning (6). 1991.

(KUBAT,1997) Kubat, Miroslav; Martin, Stan, **Addressing the Curse of Imbalanced Training sets: One-Sided Selection.** Proc. of Fourteenth International Conference (ICML'97) Tenesse. 1997.

(KUFRIN, 1997) Kufrin, Richard. **Generating C4.5 production Rules In Parallel**. Proceedings of Fourthteenth National Conference on Artificial Intelligence (AAAI-97) 1997.

(LANGLEY, 1996) Langley Pat; Simon Herbert A, **Elements of Machine Learning,** Morgan Kaufmann Publishers, Inc. USA 1996

(LUGER, 1992) Luger, George F; Stubblefield, William A., **Artificial Intelligence Structures and Strategies for Complex Problem Solving,** The Benjamim / Cummings Publishing Company, Inc., USA 1992.

(MATURANA, 1987) Maturana, Humberto; Varela, Francisco, **The tree of knowledge: the biological roots of human understanding,** Shambala Publications, Inc., Boston, 1987.

(McGRAW, 1989) McGraw, Karen L.; Harbison-Briggs Karan, **Knowledge Acquisition: Principles and Guidelines.** Prentice-Hall, Inc., USA 1989.

(MITCHELL, 1997) Mitchell, Tom M., **Machine Learning**, MacGraw-Hill, USA. 1997.

(MONARD, 1997) Monard, Maria Carolina; Rezende, Solange O., **Desenvolvimento de Sistemas Inteligentes para a Tomada de Decisão**, 8° DBFORUM. 1997.

(PROVOST,1997) Provost, Foster; Fawcett, Tom, **Analisys and Visualization of Classifier Performance: Comparision under Imprecise Class and Cost Distributions.** KDD97 – Prod. Of the Thirth International Conference on Knowledge Discovery and Data Mining. AAAI Press, USA. 1997.

(QUINLAN, 1986) Quinlan, J. Ross, **Induction of Decision Trees,** Machine Learning, 1996.

(QUINLAN, 1993) Quinlan, J. Ross, **C4.5 Programs for Machine Learning,** Morgan Kaufmann Publishers, San Diego California, 1993.

(QUINLAN, 1996) Quinlan, J. Ross, **Improved Use of Continuous Attributes in C4.5**. Journal of Artificial Intelligence Research 4. 1996.

(QUINLAN, 1997) Quinlan, J. Ross, **Is C5.0 Better Than C4.5?,** 1997, htt://207.153.200.79/see5-comparison.html em 05/12/1997.

(ROBERTS, 1995) Roberts, Huw; Denby, Mark; Totton Ken, **Accounting for Misclassification Costs in Decision Tree Classifiers**, Intelligent Data Analysis, IDA 95. 1995.

(RUSSEL, 1995) Russel, Stuart ; Norving, Peter. **Artificial Intelligence - A Modern Approach**. Prentice Hall, New York, 1995.

(SALZBERG, 1994) Salzberg, Steven , **Book Review: C4.5 : Programs for Machine Learning., by J. Ross Quinlan.** Morgan Kaufmann Publishers, Inc., 1993, Machine Learning Journal, 1-6, Netherlands: Kluwer, 1994.

(SCHAPIRE, 1997) Schapire, Robert E.; Freund Yoav; Bartlett Peter; Lee, Wee Sun, **Boosting the Margin: A new Explanation for the Effectiveness of Voting Methods ,** Machine Learning: Proceedings of the Fourteenth International Conference, 1997. http://www.research.att.com/~schapire/publist.html em 12/03/1998

(TURNEY, 1995) Turney, Peter D, **Cost-Sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm,** Journal of Artificial Intelligence Research 2, Ottawa, Canada. March,1995.

(WU, 1995) Wu, Xindong, **Knowledge Acquisition from Databases,** Ablex Publishing Corporation, USA, 1995.