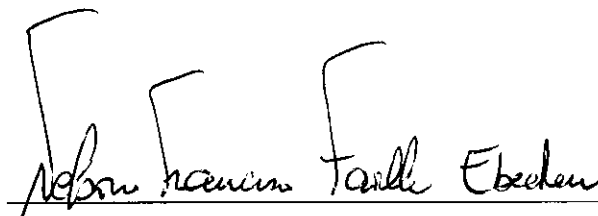


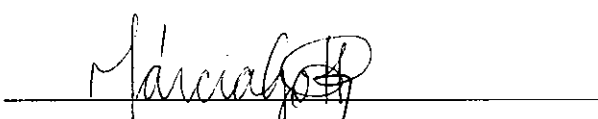
UM ESTUDO SOBRE MÉTODOS DE EXTRAÇÃO DE CONHECIMENTO
EM BANCO DE DADOS

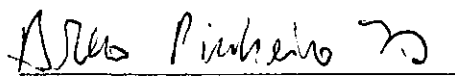
Maria Célia Santos Lopes

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS
PROGRAMAS DE PÓS-GRADUAÇÃO EM ENGENHARIA DA UNIVERSIDADE
FEDERAL DO RIO DE JANEIRO COMO PARTE INTEGRANTE DOS
REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM
CIÊNCIAS EM ENGENHARIA CIVIL.

Aprovada por:


Prof. Nelson Francisco Favilla Ebecken, D.Sc.


Prof. Márcia de Paiva Bastos Gottgroy, D.Sc.


Prof. Breno Pinheiro Jacob, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

FEVEREIRO DE 1998

LOPES, MARIA CÉLIA SANTOS

Um Estudo sobre Métodos de Extração de
Conhecimento em Bancos de Dados [Rio de
Janeiro] 1998

VII, 203 p. 29,7 cm (COPPE/UFRJ, M.Sc.,
Engenharia Civil, 1998)

Tese - Universidade Federal do Rio de
Janeiro, COPPE

1. Métodos de Extração de Conhecimento

I. COPPE/UFRJ II. Título (série)

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para obtenção do grau de Mestre em Ciências (M.Sc.).

UM ESTUDO SOBRE MÉTODOS DE EXTRAÇÃO DE CONHECIMENTO EM BANCO DE DADOS

Maria Célia Santos Lopes

Fevereiro/1998

Orientador: Nelson Francisco Favilla Ebecken

Programa: Engenharia Civil - Estruturas

Esse trabalho apresenta uma abordagem de alguns métodos selecionados de extração de conhecimento em bancos de dados. É mostrado como esses métodos se propõem a lidar com problemas que aparecem comumente em bancos de dados como: grandes massas de dados, dados errôneos, ou ainda, ausência de dados. Além disso, é apresentada também uma análise de alguns bancos de dados, feita a partir dos resultados da aplicação de algumas das técnicas descritas.

Abstract of Thesis presented to COPPE/UFRJ as partial fulfillment of the requirements of the degree of Master of Science (M.Sc.).

A STUDY ABOUT METHODS OF KNOWLEDGE DISCOVERY IN
DATABASES

Maria Célia Santos Lopes

February/1998

Advisor: Nelson Francisco Favilla Ebecken

Department: Civil Engineering - Structures

This work presents an approach of some selected methods of data mining in databases. It is shown how these methods deal with the most common problems in databases such as: huge databases, noisy data, or missing data. Besides, it is presented an analysis of a few databases done as a result of the application of some of the techniques described.

Índice

I- Introdução:	1
I.1. Objetivo:	1
I.2. Histórico:	1
I.3. Relevância:	2
II. Métodos de Extração de Conhecimento em Banco de Dados:	5
II.1. Data Mining:	5
II.1.1. Espaço de Procura:	5
II.1.2. Classes:	6
II.1.3. Agrupamentos:	6
II.1.4. Regras:	7
II.2. Descoberta de Conhecimento em Banco de Dados (KDD):	8
II.2.1. Definição:	9
II.2.2. O Processo de Descoberta de Conhecimento em Bancos de Dados (KDD):	10
II.3. Noção Geral sobre os métodos de extração de conhecimento:	12
II.3.1. A Tarefa Primária da Extração de Conhecimento:	12
II.3.2. Os Componentes dos Algoritmos de Extração de conhecimento	16
II.4. Uma abordagem sobre métodos mais populares de extração de conhecimento:	17
II.4.1. Árvores de Decisão e Regras	17
II.4.2. Regressão Não-Linear e Métodos de Classificação	18
II.4.3. Métodos Baseados em Exemplos	19
II.4.4. Modelos Gráficos de Dependência Probabilística	20
II.4.5. Modelos de Aprendizagem Relacional	20
II.5. Desafios em Pesquisa e Aplicações para a Descoberta de Conhecimento dos Bancos de Dados	21
II.6. Aspectos Mais Complexos do Processo KDD	23
II.6.1. Descoberta da Tarefa	23
II.6.2. Descoberta dos Dados	24
II.6.3. Triagem dos Dados	24
II.6.4. Conhecimento de Apoio	25
III. Uma Abordagem Geral sobre Sistemas de Extração de Conhecimento	27
III.1. ID3	27

III.1.1. Espaço de Procura.....	28
III.1.2. Algoritmo de Procura	28
III.1.3. Atributos Numéricos.....	30
III.1.4. Valores de Atributos Agrupados	30
III.1.5. Ruído.....	31
III.1.6. Valores de Atributos Ausentes	32
III.1.7. Janelas.....	33
III.1.8. Aprendizado Experimental	33
III.1.9. Conclusão	34
III.2. AQ15.....	34
III.2.1. Espaço de Procura.....	35
III.2.2. Algoritmo de Procura	36
III.2.3. Dados Inconsistentes e Com Ruído	37
III.2.4. Indução Construtiva.....	38
III.2.5. Aprendizado Incremental.....	38
III.2.6. Conclusão	39
III.3 DBLearn.....	39
III.3.1 Espaço de Procura.....	39
III.3.2. Algoritmo de Aprendizagem: regras consistentes	41
III.3.3. Ruído.....	41
III.3.4. Aprendizado Incremental.....	42
III.3.5. Conclusão	43
III.4. CN2.....	44
III.4.1. Resumo:	44
III.4.2. Introdução	44
III.4.3. O CN2 e os Outros Algoritmos Para Estudo Comparativo	46
III.4.3.1. O ASSISTANT	46
III.4.3.2. O AQR.....	48
III.4.3.3. Um Classificador Bayesiano.....	52
III.4.3.5. Algoritmo CN2.....	54
III.4.3.6. Algoritmo de Procura	60
III.4.3.7. Conclusão	61
III.5. O Pacote de Algoritmos Tooldiag	61
III.5.1. Introdução	62

III.5.2. Aquisição dos Dados:	64
III.5.3. Seleção de Características:	64
III.5.4. Estimativa de Erro:	64
III.5.5. Geração do Classificador	66
III.5.5.1. Classificador do Vizinho mais Próximo:.....	66
III.5.6. Visualização dos Dados Multidimensionais.....	75
III.5.7. Implementação.....	75
III.5.8. Conclusões.....	76
III.6. Sistemas Fuzzy	76
III.6.1. Introdução:.....	76
III.6.1.1. Uma Noção Geral Sobre Conjuntos Fuzzy.....	76
III.6.1.2. Variáveis Linguísticas.	77
III.6.1.3. Razão de Aproximação.....	79
III.6.1.4. Representação do Conhecimento:.....	80
III.6.1.5. Os Problemas da Engenharia de Conhecimento	80
III.6.1.6. Regras de Produção Obtidas a Partir de Modelos Verbais	81
III.6.2. Vantagens dos Sistemas Fuzzy	82
III.6.3. Limitações dos Modelos de Sistemas Fuzzy	83
III.6.3.1. Sistemas Lineares com um Modelo Matemático Bem Definido	83
III.6.3.2. O Modelo Envolve Variáveis Fuzzy.....	84
III.6.3.3. O Modelo Usa Dados Imprecisos	85
III.6.3.4. Revisão Regular do Modelo	88
III.6.3.5. Os Benefícios da Facilidade Explanatória.....	88
III.6.3.6. Manutenção do Modelo	89
III.6.4. Processamento da Linguagem Natural	89
III.6.5. Lidando com Informações Ausentes	89
III.6.5.1. Ruído (Incerteza) ao Invés de Dados Ausentes	90
III.6.5.2. Dados Desconhecidos ou Ausentes	92
III.6.6. As Idéias da Descoberta e Extração do Conhecimento	93
III.6.6.1. Os Benefícios da Extração de Conhecimento.....	93
III.6.6.2. Problemas com a Extração do Conhecimento	94
III.6.7. O Método de Descoberta de Regras de Wang-Mendel	96
III.6.7.1. O Processo de Descoberta de Regras.....	99
III.6.7.2. Um Modelo Simples de Sensibilidade de Preços	99

III.6.7.3. Representando e Usando os Dados de Comportamento do Modelo	100
III.6.8. Executando o Processo de Descoberta de Regras.....	102
III.6.8.1. Passo Um - Decompor Variáveis em Conjuntos Fuzzy.....	104
III.6.8.2. Passo Dois - Gera Regras de Tentativa Para os Dados.....	112
III.7. Métodos Adotados:.....	122
IV. Estudo de Casos:.....	123
IV.1. Introdução:.....	123
IV.2. Conjuntos de dados das Flores íris de Fisher	123
V.2.1. Resultados Comentados:.....	124
V.2.1.1. No Tooldiag:.....	124
V.2.1.2. No CN2	141
V.2.1.3. No Wang-Mendel	143
IV.3. Conjunto de Dados Climáticos do Aeroporto.....	152
Internacional do Galeão.....	152
V.3.1. Resultados Comentados:.....	155
V.3.1.1. No Tooldiag	155
V.3.1.2. No CN2	167
V. Conclusões:.....	172
V.1. CN2.....	173
V.2. Método Wang-Mendel	173
V.3. Tooldiag.	174
V.4. Sugestões e Trabalhos Futuros.....	174
Bibliografia:.....	176
Apêndice I:.....	181
Apêndice II	194
Apêndice III.....	200

I- Introdução:

I.1. Objetivo:

O objetivo da tese é abordar todo o processo de descoberta de conhecimento útil em banco de dados, incluindo as etapas de pré-processamento e interpretação dos resultados. Esse processo é tipicamente interativo e iterativo, e envolve a aplicação repetida de métodos ou algoritmos de extração de conhecimentos específicos e a interpretação dos modelos gerados por esses algoritmos.

O pré-processamento dos dados é quase sempre necessário, uma vez que dificilmente os dados se apresentam completos ou sem qualquer erro. Essa é a etapa de preparação dos dados para a posterior aplicação dos algoritmos.

Nesse trabalho, os algoritmos escolhidos já pertencem a pacotes que são utilizados atualmente em pesquisas de *data mining*.

Depois de aplicados os algoritmos selecionados, resta ainda a etapa de interpretação e visualização dos dados. O que vai ser feito nessa última etapa depende do tipo de dado que se está trabalhando e o que se espera como resultado. Por exemplo: se é um diagnóstico de doença, uma previsão de chuva ou uma tendência do mercado financeiro.

I.2. Histórico:

Nesta última década, tem-se visto um crescimento explosivo da capacidade de gerar e coletar dados. Os avanços na forma de coletar dados científicos (isto é, a partir de sensores remotos ou de satélites espaciais), a muito difundida introdução do código de barras para quase todos os produtos comerciais, e a computadorização de várias atividades (como as compras com cartão de crédito) e as transações governamentais (isto é, cobrança de impostos) têm gerado um enorme fluxo de dados. Os avanços nas formas de armazenamento dos dados (isto é, discos magnéticos, CD-ROMS), sistemas de gerenciamento de bancos de dados melhores, e a tecnologia de *data warehousing*, permitiram que se transformasse esse “dilúvio” de dados em “montanhas” de dados armazenados.

Exemplos representativos são fáceis de encontrar. No mundo dos negócios, um dos maiores bancos de dados foi criado por Wal-Mart (um varejista americano) que manuseia 20 milhões de transações por dia [1].

Quase todas as transações relativas à assistência médica nos Estados Unidos estão sendo guardadas em computadores, produzindo bancos de dados de muitos gigabytes. Esse tipo de acontecimento vem se repetindo em muitas outras empresas que já começam a analisar melhor esses dados para poder controlar custos e melhorar sua qualidade. A Companhia de Óleo Mobil está desenvolvendo uma *data warehouse* capaz de armazenar 100 terabytes de dados relacionados à exploração de petróleo [2].

Existem ainda bancos de dados científicos extraordinariamente grandes. O projeto do genoma humano [3] coletou gigabytes de dados do código genético humano e muito mais é esperado.

Um outro projeto ligado à astronomia registra bilhões de entradas de dados coletados na ordem de grandeza de terabytes relativos à observação de objetos do céu. Um sistema de observação da Terra criado pela NASA será capaz de gerar na ordem de 50 gigabytes de dados remotos de imagens, quando estiver operacional no final dos anos 90 ou início do próximo século.

Tais volumes de dados claramente descartam métodos manuais tradicionais tais como planilhas ou pesquisas *ad hoc*. Esses métodos podem criar relatórios informativos a partir dos dados, mas não podem analisar o conteúdo daqueles relatórios para se concentrar no que é conhecimento importante. Existe uma significativa necessidade de uma nova geração de técnicas e ferramentas com a habilidade de automaticamente e inteligentemente auxiliar na análise das grandes massas de dados, procurando por porções de dados consideradas úteis.

Essas técnicas e ferramentas são o assunto desse campo emergente: a descoberta de conhecimento em bancos de dados (KDD).

I.3. Relevância:

A Descoberta de Conhecimento em bancos de dados (KDD) vem preencher uma grande e antiga lacuna no tratamento de bancos de dados. Há muito tempo, o manuseio de grandes massas de dados tem sido um problema. Além disso, bancos de dados muito grandes comumente contém dados errôneos ou incompletos que

poderiam ser importantes para efeito de consulta e utilização dos mesmos. A KDD se propõe a solucionar esse e outros problemas a partir de um tratamento desses dados, associados, quando for o caso, ao conhecimento de especialistas.

Um grande número de aplicações interessantes e importantes na descoberta de conhecimento em banco de dados tem sido desenvolvidas. Exemplos em áreas de aplicação em ciências incluem:

- **Astronomia:** O sistema SKICAT da JPL/Caltech é usado por astrônomos para identificar estrelas e galáxias automaticamente, fazendo um reconhecimento do céu em larga escala. Os dados extraídos são catalogados e analisados cientificamente.
- **Biologia Molecular:** Vários sistemas têm sido desenvolvidos para encontrar padrões em estruturas moleculares e em dados genéticos.
- **Modelagem da Mudança Global de Clima:** padrões espaço-tempo tais como um ciclone são automaticamente encontrados em grandes bancos de dados simulados e observados.

Existem ainda várias outras aplicações envolvendo KDD, entre elas:

- O chamado " banco de dados de *marketing* ", que é um método de analisar bancos de dados de clientes, procurando por padrões entre as preferências dos clientes e usando esses padrões para uma seleção mais objetiva em clientes futuros.
- **Análise e seleção de estoques.** Já existem no mundo várias companhias de investimento que controlam seus estoques utilizando técnicas de extração de conhecimento avançadas.
- **Prevenção e detecção de fraudes** é uma outra área onde a KDD atua.

O crescimento explosivo dos bancos de dados científicos, governamentais, financeiros, etc. já ultrapassaram a capacidade de interpretar e digerir esses dados, criando a necessidade de uma nova geração de ferramentas e técnicas para uma

análise automatizada e inteligente dos bancos de dados. Algumas dessas ferramentas e técnicas são assunto dessa tese.

II. Métodos de Extração de Conhecimento em Banco de Dados:

II.1. Data Mining:

Data mining é uma procura por descrições, dentro do banco de dados. Uma vez que a quantidade de dados é, normalmente, finita, nem todos os conceitos podem ser corretamente extraídos. De fato, mesmo com bancos de dados que cresçam indeterminadamente, alguns conceitos podem não ser corretamente extraídos, uma vez que alguns problemas são insolúveis. Entretanto alguns desses conceitos podem ser aproximados.

II.1.1. Espaço de Procura:

O banco de dados, usualmente chamado de conjunto de treinamento S para os algoritmos de aprendizado, representa informações sobre o ambiente. Em princípio existem vários formalismos de representação possíveis, entretanto, uma vez que a maior parte dos Sistemas de Gerenciamento de Banco de Dados (DBMS) correntes são relacionais ou pelo menos suportam uma interface SQL, pode-se assumir que o Banco de Dados em questão seja relacional. Em outras palavras objetos de um ambiente são representados por tuplas no banco de dados.

Pode-se assumir que as tuplas no banco de dados apenas representam propriedades dos objetos e não relações entre objetos. Isto é, cada objeto no ambiente é representado por uma tupla e uma tupla representa um ou mais objetos no ambiente. Uma tupla só pode representar mais de um objeto se a distinção entre esses objetos é considerada irrelevante. Pode-se assumir também que o banco de dados consiste de uma única tabela e que cada domínio é finito.

II.1.2. Classes:

O banco de dados é o ambiente no qual se aplica a ferramenta de extração de conhecimento, isto é, a ferramenta de extração de conhecimento tem que inferir um modelo a partir do banco de dados. No caso do aprendizado supervisionado, isso requer que o usuário defina uma ou mais classes, também conhecidas como ‘conceitos’ [4], no banco de dados.

II.1.3. Agrupamentos:

Se o usuário não define classes, o aprendizado é não-supervisionado. Por isso, o sistema tem que descobrir suas próprias classes, isto é, o sistema agrupa os dados no banco de dados. Pode-se mostrar isso na figura II.1. Primeiro, o sistema tem que descobrir subconjuntos de objetos relacionados no conjunto de treinamento, e assim encontrar descrições (isto é D1, D2, D3) que descrevam cada um dos subconjuntos.

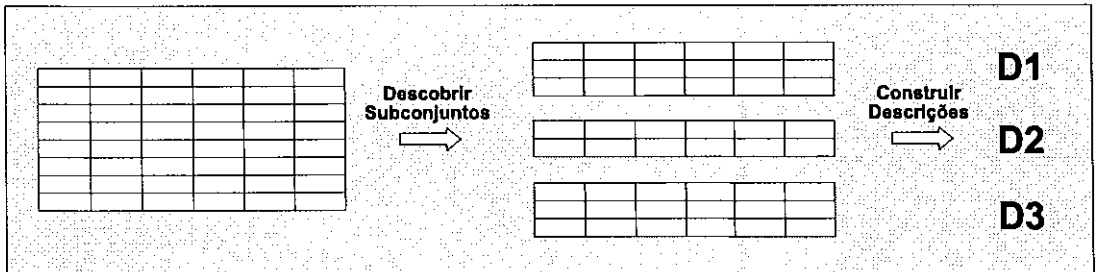


Figura II.1. A descoberta de Agrupamentos e Descrições em um Banco de Dados

Para ilustrar a complexidade do aprendizado não-supervisionado, dá-se o número de agrupamentos possíveis de um banco de dados com N tuplas. Se existem m agrupamentos no banco de dados, particiona-se o conjunto de N tuplas em m subconjuntos disjuntos não-vazios. Seja denotado por $P(N,m)$ o número de modos nos quais isso pode ser feito. Em geral o número de maneiras de particionar N elementos em m subconjuntos é dado por [5]:

$$P(N, m) = \frac{1}{m!} \sum_{j=0}^m \binom{m}{j} (-1)^j (m-j)^N$$

$P(N, m)$ é uma função que cresce exponencialmente com N , de fato, para um N grande têm-se $P(N, m) \approx m^N / m! \approx m^{N-m} e^m \sqrt{2\pi m}$. O número total de maneiras nas quais o banco de dados de n tuplas pode ser agrupado, denotado por $C(N)$ é:

$$C(N) = \sum_{m=1}^N P(N, m) = \sum_{m=1}^N \frac{1}{m!} \sum_{j=0}^m \binom{m}{j} (-1)^j (m-j)^N$$

Por exemplo, para $N=8$, tem-se:

m	1	2	3	4	5	6	7	8
$P(8, m)$	1	127	966	1701	1050	266	28	1

Assim, $C(8) = \sum_{m=1}^8 P(8, m) = 4140$. Claramente, o processo de extração de conhecimento só é bem sucedido se o número de classes no banco de dados é pequeno. Em outras palavras, $C(N)$ é grande demais como uma estimativa. Entretanto, mesmo se for assumido que o número de classes esperado no banco de dados é pequeno comparado ao tamanho do banco de dados, pode-se dizer 10, o número de agrupamentos, isto é, $\sum_{m=1}^{10} P(N, m)$, ainda é confuso.

II.1.4. Regras:

Quando as classes são definidas, sistemas baseados em regras devem inferir regras que governem a classificação. Isto é, o sistema deve encontrar a descrição de cada classe. Essas descrições devem, naturalmente, referirem-se apenas aos atributos prognosticáveis do conjunto de treinamento. Idealmente, todos os exemplos positivos deveriam satisfazer a descrição e nenhum dos exemplos negativos deveria satisfazer.

Se apenas atributos de objetos são armazenados no conjunto de treinamento, não devem existir relações entre os objetos, e assim as descrições só podem consistir de condições sobre esses atributos. Na verdade, de uma forma geral as classificações são feitas baseadas apenas nas contribuições que um atributo pode fazer. Um determinado atributo pode influenciar muito na classificação ou pode não ter qualquer efeito na mesma, independentemente do tipo de banco de dados com o qual se está trabalhando.

II.2. Descoberta de Conhecimento em Banco de Dados (KDD):

A Descoberta de Conhecimento em Bancos de Dados é de interesse para pesquisadores em aplicações como aprendizado por máquina, reconhecimento de padrão, banco de dados, estatística, inteligência artificial, aquisição de conhecimento para sistemas especialistas, e visualização de dados. A KDD trabalha utilizando métodos, algoritmos, e técnicas nesses diversos campos. A única meta é extrair conhecimento a partir de grandes bancos de dados. Esse conhecimento deve ser aproveitado em suas áreas específicas para as mais diversas finalidades e em particular para diminuir a quantidade de dados importantes ou necessários a serem armazenados. Se para uma determinada aplicação, apenas metade dos atributos contribuem para atingir o objetivo final, tem-se uma economia de 50% em termos de leituras e armazenamentos de dados.

A KDD tem muito a ver com a estatística e comumente apresenta procedimentos estatísticos específicos para modelar dados e tratar o ruído dentro de uma estrutura de descoberta de conhecimento.

II.2.1. Definição:

A descoberta de conhecimento em bancos de dados é um processo não-trivial de identificação de padrões válidos, singulares, potencialmente úteis e, principalmente, inteligíveis, nos dados.

É necessário, agora, esclarecer alguns termos mais detalhadamente:

- *Dados* são um conjunto de fatos F (isto é, casos num banco de dados).
- *Padrão* é uma expressão E numa linguagem L descrevendo fatos num subconjunto F_E de F . E é chamado de padrão se ele é mais simples que a enumeração de todos os fatos em F_E . Ele descreve algo em comum a todos os dados do subconjunto F_E de F .
- *Processo*: usualmente o processo de KDD é um processo de vários passos, que envolve a preparação dos dados, a procura por padrões, a avaliação do conhecimento, e o refinamento envolvendo iteração após modificação. O processo é assumidamente não-trivial. Essas etapas a serem cumpridas advêm dos tipos de dados e das aplicações a que esses dados servem.
- *Validade*: Os padrões descobertos devem ser válidos em relação aos novos dados com alguma certeza. Uma medida de certeza é uma função C que mapeia expressões em L para um espaço de medidas parcialmente ou totalmente ordenado M_C . Uma expressão E em L sobre um subconjunto $F_E \subset F$ pode ser associado a uma certa medida $c = (E, F)$.
- *Singularidade*: Os padrões são singulares (ao menos ao sistema). A singularidade pode ser medida em relação às mudanças nos dados (pela comparação dos valores correntes com os prévios ou com os esperados) ou no conhecimento (como uma nova descoberta é relacionadas às antigas). Em geral, é assumido que isso pode ser medido por uma função $N(E, F)$, que pode ser uma função booleana ou uma medida do grau de singularidade ou inesperabilidade.
- *Potencialmente Úteis*: Os padrões deveriam potencialmente levar a algumas ações úteis, como medido por alguma função utilitária. Tal função U mapeia expressões em L para um espaço de medida parcialmente ou totalmente ordenado M_U : assim, $u = U(E, F)$.

- **Principalmente Inteligíveis:** Um objetivo da KDD é fazer com que os padrões extraídos sejam facilmente compreendidos de forma a facilitar um melhor entendimento dos dados básicos. Embora isso seja difícil de medir precisamente, um substituto frequente é a medida de simplicidade. Existem várias medidas de simplicidade, e elas variam da sintática pura (isto é, o tamanho do padrão em bits) até a semântica (isto é, fácil entendimento em qualquer situação). Assume-se que isso é medido, quando possível, por uma função S que mapeia expressões E em L para um espaço de medidas parcialmente ou totalmente ordenadas M_S : assim, $s = S(E,F)$.

A Descoberta de Conhecimento em Banco de Dados é um processo que utiliza métodos (algoritmos) de extração de conhecimento para identificar e extrair o que é considerado conhecimento de acordo com especificações de medidas e limites, usando uma base de dados F acompanhada de qualquer pré-processamento, sub-amostragem, e transformações de F requeridos. A componente de extração de conhecimento do processo de KDD diz respeito principalmente aos meios pelos quais os padrões são extraídos e enumerados a partir dos dados. A descoberta do conhecimento envolve a avaliação e a possível interpretação dos padrões para tomar a decisão do que constitui ou não conhecimento. Inclui também pré-processamento, amostragem, e projeções dos dados.

II.2.2. O Processo de Descoberta de Conhecimento em Bancos de Dados (KDD):

O processo de KDD é um processo constituído de várias etapas envolvendo decisões a serem tomadas por parte do usuário. Brachman e Anand [6] apresentam uma visão prática do processo de KDD enfatizando sua natureza interativa. A seguir resumem-se em linhas gerais alguns de seus passos básicos:

1. Desenvolver um entendimento do domínio da aplicação, o conhecimento relevante a priori, e as metas do usuário final.

2. Criar um conjunto alvo de dados: selecionar o conjunto de dados, ou focalizar um subconjunto de variáveis ou amostras de dados, nos quais a descoberta deve ser realizada.

3. Limpeza dos dados e pré-processamento: operações básicas tais como remover o ruído, coletar a informação necessária para modelar ou calcular o ruído, decidir sobre estratégias para manusear campos de dados ausentes.

4. Redução e projeção de dados: encontrar ferramentas úteis para representar os dados que variam de acordo com a meta da tarefa. Usar a redução de dimensão ou os métodos de transformação para reduzir o número efetivo de variáveis sob consideração ou encontrar representações invariantes para os dados.

5. Escolher a tarefa de extração de conhecimento a ser realizada: decidir se a meta do processo de KDD é classificação, regressão, agrupamento, etc.

6. Escolher o(s) algoritmo(s) de extração de conhecimento: selecionar o(s) método(s) a serem usados na procura por padrões nos dados. Isso inclui decidir quais modelos ou parâmetros podem ser apropriados e combinar os vários recursos de extração de conhecimento entre si e comparar os resultados.

7. Procurar por modelos interessantes a uma forma de representação particular ou um conjunto de tais representações: classificações de regras ou árvores, regressão, agrupamento e assim por diante. O usuário pode ajudar significativamente o processo de extração de conhecimento executando corretamente os passos anteriores.

8. Interpretar modelos extraídos, com um possível retorno a qualquer dos passos 1-7 para uma iteração adicional.

9. Consolidar o conhecimento descoberto: incorporar esse conhecimento ao sistema de desempenho, ou simplesmente documentá-lo e reportá-lo às partes interessadas. Isso inclui também checagem e resolução de conflitos potenciais com o conhecimento previamente confiável (ou extraído).

II.3. Noção Geral sobre os métodos de extração de conhecimento:

A maior parte dos métodos de extração de conhecimento são baseados em conceitos de aprendizado por máquina, reconhecimento de padrão e estatística: classificação, agrupamento, modelos gráficos e etc. A exibição de diferentes algoritmos para a solução desses problemas pode, frequentemente, ser bastante confusa tanto para um analista de dados experiente quanto para um novato. Apresenta-se a seguir uma noção dos métodos de extração de conhecimento, tentando particularmente transmitir a noção de que a maior parte (senão todos) os métodos podem ser vistos como extensões ou híbridos de alguns poucos princípios e técnicas básicas.

II.3.1. A Tarefa Primária da Extração de Conhecimento:

Os dois primeiros objetivos primários da extração de conhecimento na prática tendem a ser o prognóstico e a descrição. O prognóstico envolve usar algumas variáveis ou campos no banco de dados para prever valores futuros ou desconhecidos de outras variáveis de interesse. A descrição concentra-se em encontrar padrões humanamente interpretáveis descrevendo os dados. A importância relativa do prognóstico e da descrição para uma aplicação em extração de conhecimento particular pode variar consideravelmente. No conceito de KDD que é visto a seguir, entretanto, a descrição tende a ser mais importante que o prognóstico. Os objetivos de prognóstico e de descrição são alcançados usando as seguintes tarefas de extração de conhecimento primárias:

- *Classificação* é o aprendizado da função que classifica um item de dados em uma de várias classes pré-definidas. Exemplos de métodos de classificação usados como parte das aplicações utilizadas para descoberta de conhecimento incluem classificar tendências do mercado financeiro e identificação automática de objetos de

interesse em grandes bancos de dados de imagens. A figura II.2 mostra um particionamento simples dos dados de empréstimos em duas regiões de classes. Note que não é possível separar as classes perfeitamente usando uma divisa linear de decisão. O banco gostaria de usar as regiões de classificação para decidir automaticamente se será concedido empréstimos a novos pretendentes.

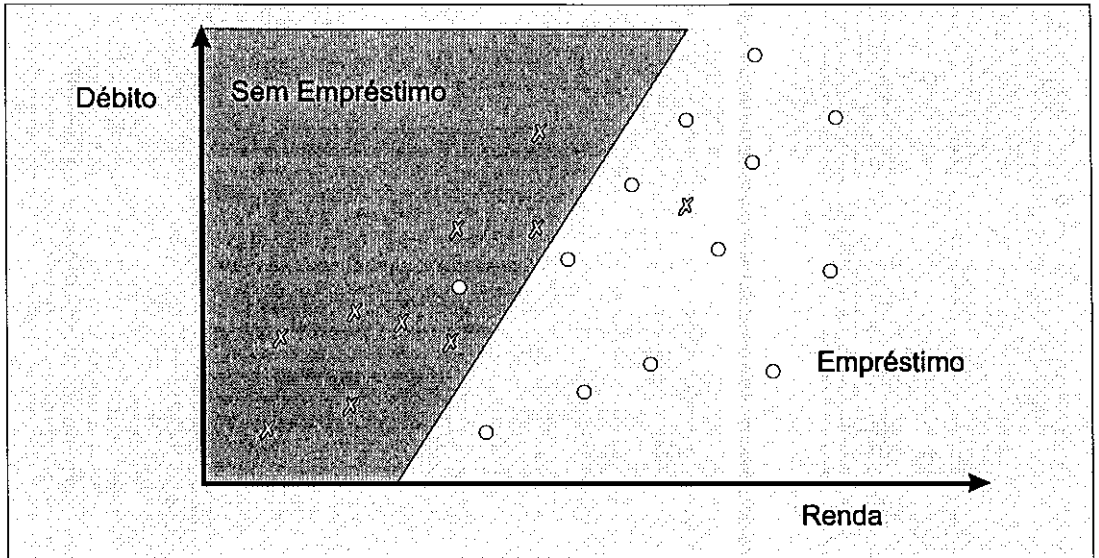


Figura II.2. Uma Fronteira Linear para o Conjunto de Dados Relativos à Empréstimos

- *Regressão* é o aprendizado da função que classifica um item de dados em uma variável de prognóstico de valor real. Existem várias aplicações para a regressão, como por exemplo, prognosticar a quantidade de biomassa presente numa floresta dadas as medidas feitas remotamente a partir de micro-ondas, estimar a probabilidade que um paciente vá morrer dados os resultados de um conjunto de exames médicos, prognosticar a demanda para consumo de um novo produto como função dos gastos com propaganda. A figura II.3 mostra o resultado de uma regressão linear simples onde o “débito total” é ajustado como uma função linear de “rendimentos”: o ajuste é pobre uma vez que a correlação entre as duas variáveis é pobre.

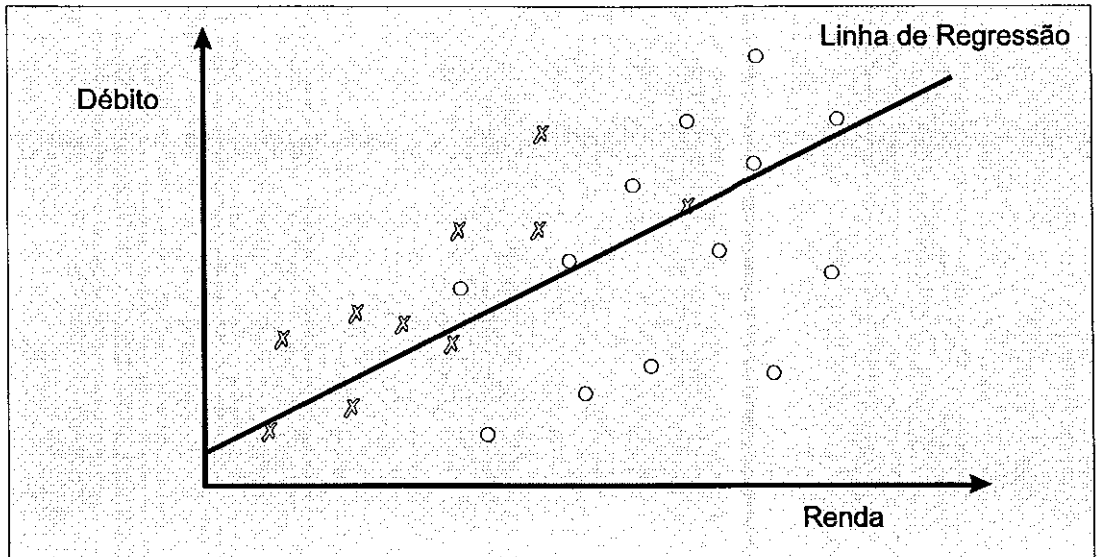


Figura II.3. Uma Regressão Linear para o Conjunto de Dados Relativos à Empréstimos

- *Agrupamento* (Clustering) é uma tarefa de descrição comum onde se procura identificar um conjunto finito de categorias ou classes para descrever os dados. As categorias podem ser mutuamente excludentes, ou consistir de uma representação mais complexa como as categorias hierárquicas ou superpostas. Exemplos de aplicações de agrupamentos num contexto voltado para a descoberta de conhecimento incluem a descoberta de sub-populações homogêneas para consumidores em bancos de dados de *marketing* e identificação de sub-categorias de espectro a partir de medidas do céu em infra-vermelho. A figura II.4 mostra um possível agrupamento do conjunto de dados de empréstimo em três agrupamentos: note que os agrupamentos se superpõem permitindo que alguns pontos dos dados pertençam a mais de um agrupamento. As legendas das classes originais (denotadas por \times e o nas figuras anteriores) foram substituídas por $+$ para indicar que a pertinência às classes não é mais assumida como conhecida. Muito próximo à tarefa de agrupamento está a tarefa de estimativa da densidade probabilística, que consiste de técnicas para estimar a partir dos dados, a função de densidade probabilística de junção multivariada de todos os campos/variáveis no banco de dados.

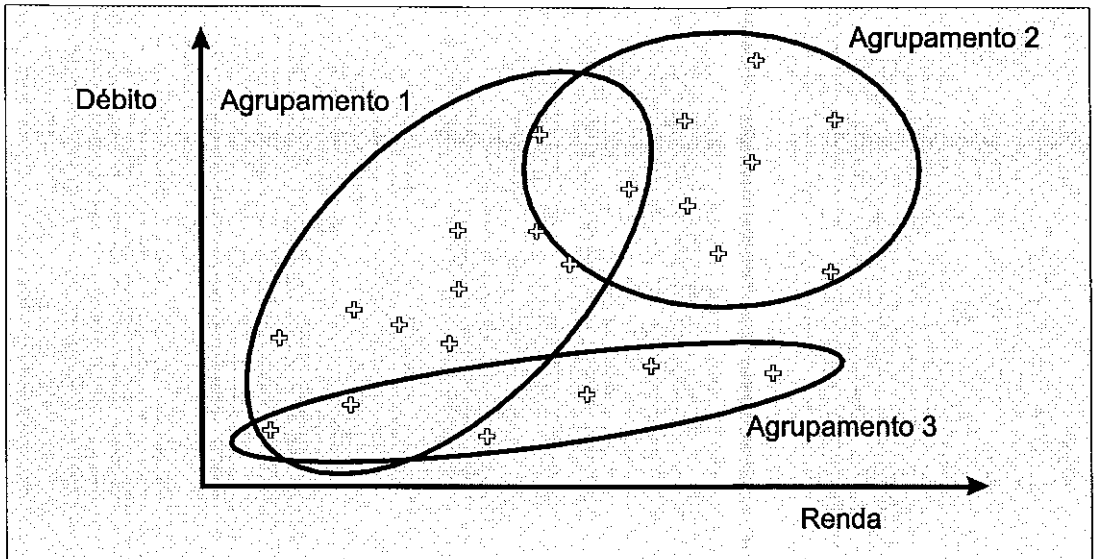


Figura II.4. Uma Divisão em Três Agrupamentos do Conjunto de Dados Relativos a Empréstimos. Note que as legendas originais foram substituídas por +’s.

- *Sumarização* (resumo) envolve métodos para encontrar uma descrição compacta para um subconjunto de dados. Um exemplo simples seria tabular os desvios médios e padrões para todos os campos. Métodos mais sofisticados envolvem técnicas de visualização multivariada, e a descoberta de relações funcionais entre as variáveis. Técnicas de sumarização são frequentemente aplicadas à análise de dados exploratória interativa e a geração de relatórios automatizada.

- *Modelagem Dependente* consiste em encontrar um modelo que descreva *dependências* significativas entre variáveis. Modelos dependentes existem em dois níveis: o nível estrutural do modelo especifica que variáveis são localmente dependentes uma da outra, ao passo que o nível quantitativo do modelo especifica o grau de dependência usando alguma escala numérica. Por exemplo, redes de dependência probabilística usam independência condicional para especificar o aspecto estrutural do modelo e probabilidades ou correlações para especificar a força das dependências. Redes de dependência probabilística estão cada vez mais encontrando aplicações em áreas diversas como o desenvolvimento de sistemas especialistas probabilísticos médicos a partir de bancos de dados, recuperação de informação, e modelagem do código genético humano.

- *Detecção de Mudança e Desvio* concentra-se na descoberta das mudanças mais significativas nos dados em relação a medidas prévias ou valores normativos.

II.3.2. Os Componentes dos Algoritmos de Extração de conhecimento

Depois de dar uma idéia das tarefas primárias de extração de conhecimento, o próximo passo é construir um algoritmo para realizá-las. Pode-se identificar três componentes primárias em qualquer algoritmo de extração de conhecimento: *representação do modelo*, *avaliação do modelo*, e *procura*. Essa visão simplista não é necessariamente completa, ou melhor, é apenas uma maneira conveniente de expressar conceitos-chave dos algoritmos de extração de conhecimento de uma maneira relativamente unificada e compacta. (Cheeseman [7] esboça uma estrutura similar).

- A *Representação do Modelo* é a linguagem L para descrever padrões que podem ser descobertos. Se a representação do modelo é muito limitada, mesmo que se empregue muito tempo de treinamento ou muitos exemplos não se conseguirá produzir um modelo exato para os dados. Assim, é necessário que o analista de dados compreenda totalmente as suposições representacionais que podem ser inerentes a um método particular. É igualmente importante que aquele que desenvolve um algoritmo declare claramente quais são as suposições representacionais que estão sendo feitas por esse algoritmo em particular. Note que, por outro lado, representações muito completas do modelo aumentam o risco de se ajustarem demais aos dados de treinamento resultando numa exatidão de prognósticos reduzida em dados ainda não vistos. Além do mais a procura se torna muito mais complexa e a interpretação do modelo muito mais difícil.

- A *Avaliação do Modelo* estima o quanto um exemplo em particular (um modelo e seus parâmetros) se adequa a um critério do processo KDD. A avaliação da exatidão do prognóstico (validade) é baseada na validação cruzada (cross validation). A avaliação da qualidade descritiva envolve a exatidão do prognóstico, a singularidade, e a inteligibilidade do modelo escolhido. Ambos os critérios, lógico e estatístico podem ser usados para avaliação do modelo. Por exemplo, o princípio de

probabilidade máxima escolhe para o modelo, parâmetros que forneçam o melhor ajuste aos dados de treinamento.

- O *Método de Procura* consiste de dois componentes: *Procura de Parâmetros* e *Procura de Modelos*. Na *procura de parâmetros*, o algoritmo deve procurar pelos parâmetros que otimizam o critério de avaliação do modelo, dada uma representação fixa do modelo e dos dados observados. Para problemas relativamente simples não existe procura: a estimativa do parâmetro ótimo pode ser obtida em forma fechada. Tipicamente para modelos mais gerais, a solução em forma fechada não está disponível: métodos iterativos mais complexos são, então, utilizados. A procura do modelo ocorre como um loop sobre o método de procura de parâmetros: a representação do modelo é mudada para que a família de modelos seja considerada. Para cada representação específica do modelo, o método de procura de parâmetros é instanciado para avaliar a qualidade daquele modelo em particular. As implementações dos métodos de procura de modelos tendem a usar técnicas de procura heurística uma vez que o tamanho do espaço dos modelos possíveis frequentemente impede a procura exaustiva e as soluções em forma fechada não são facilmente obtidas.

II.4. Uma abordagem sobre métodos mais populares de extração de conhecimento:

Existe uma grande variedade de métodos de extração de conhecimento. Nesse estudo, porém, serão apresentadas apenas um subconjunto dessas técnicas. Cada método é discutido no contexto de representação do modelo, de avaliação e procura do modelo.

II.4.1. Árvores de Decisão e Regras

Árvores de decisão e regras que usam divisão univariável têm uma forma de representação simples, fazendo com que o modelo inferido seja relativamente fácil de compreender pelo usuário. Entretanto, qualquer restrição na representação de uma

árvore ou a introdução de uma regra particular pode restringir significativamente a forma funcional e assim o poder de aproximação do modelo.

II.4.2. Regressão Não-Linear e Métodos de Classificação

Esses métodos consistem de uma família de técnicas para prognósticos que ajustam combinações de funções de base lineares e não-lineares (sigmoide, *spline*, polinomiais) à combinações de variáveis de entrada. Exemplos incluem redes neurais *feed forward*, métodos adaptativos *spline*, e assim por diante. Considere a rede neural, por exemplo. A figura II.5 ilustra o tipo de fronteira não-linear que uma rede neural pode encontrar para o conjunto de dados relativos a empréstimos. Em termos de avaliação do modelo, enquanto redes de um tamanho apropriado podem universalmente aproximar funções homogêneas para qualquer grau de exatidão, relativamente pouco é conhecido sobre as propriedades de representação das redes de tamanho fixo estimadas a partir de conjuntos de dados finitos. Métodos de regressão não-linear, embora poderosos em força representacional, podem ser muito difíceis de interpretar.

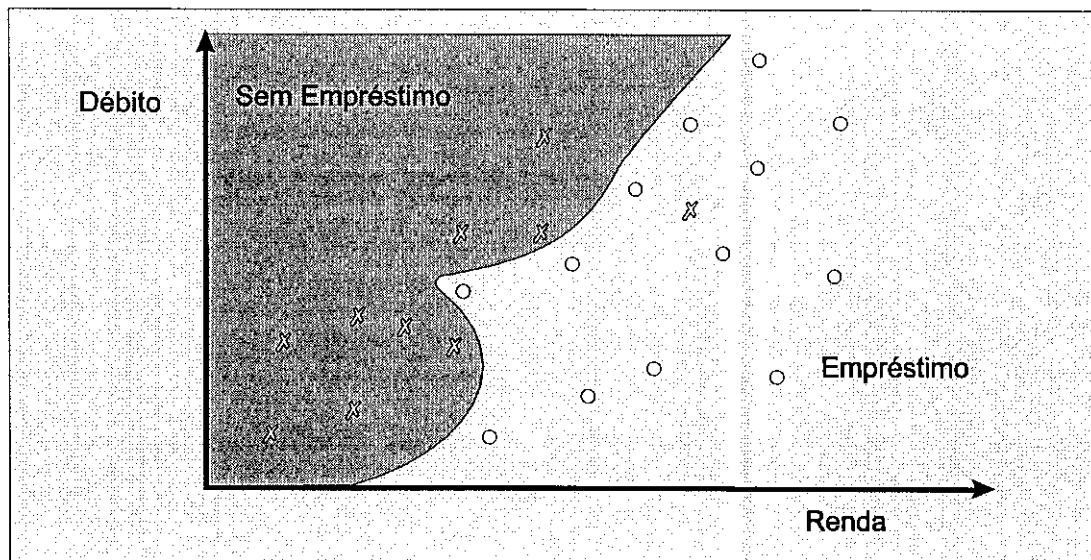


Figura II.5. Um Exemplo de Fronteira de Classificação Aprendida por um Classificador Não-Linear

II.4.3. Métodos Baseados em Exemplos

A representação é simples: usa-se exemplos representativos do banco de dados para aproximar um modelo, isto é, prognósticos em exemplos novos são derivados de propriedades de exemplos "similares" no modelo em que o prognóstico já se consumou. As técnicas incluem o modelo de classificação do vizinho mais próximo e algoritmos de regressão. A figura II.6 ilustra o uso do classificador do vizinho mais próximo para o conjunto de dados de empréstimos: a classe em qualquer ponto novo em um espaço bi-dimensional é o mesmo que a classe do ponto mais próximo do conjunto de dados de treinamento original.

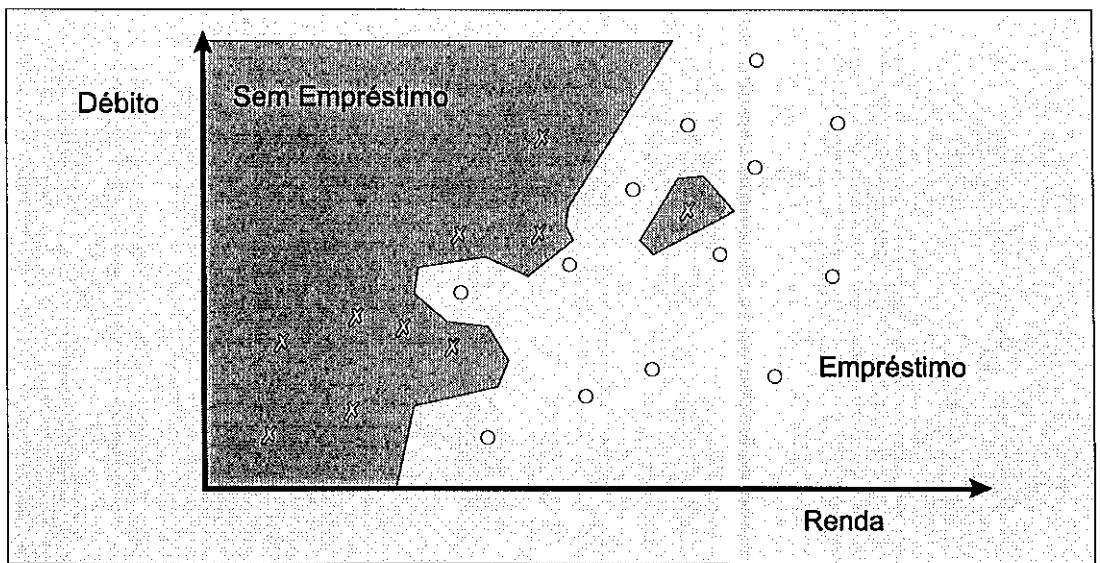


Figura II.6. As Fronteiras de Classificação para o Classificador do Vizinho Mais Próximo para o Conjunto de Dados Relativos a Empréstimos.

Uma desvantagem potencial dos métodos baseados em exemplos (comparados com métodos baseados em árvores, por exemplo) é que é requerida uma distância métrica bem definida para a avaliação da distância entre os pontos de dados. A avaliação do modelo é normalmente baseada em estimativas de validação cruzada de um erro de prognóstico: "parâmetros" do modelo a serem estimados podem incluir o número de vizinhos a usar o prognóstico e a própria distância métrica. Da mesma forma que os métodos de regressão não-lineares, os métodos baseados em exemplos são muitas vezes assintoticamente poderosos em função das propriedades de aproximação, mas em contraposição podem ser difíceis de interpretar uma vez que o

modelo está implícito aos dados e não formulado explicitamente. Técnicas correlatas incluem estimativa da densidade do kernel e modelagem mista.

II.4.4. Modelos Gráficos de Dependência Probabilística

Modelos gráficos especificam as dependências probabilísticas que estão implícitas num modelo particular usando uma estrutura gráfica. Na sua forma mais simples, o modelo especifica que variáveis são diretamente dependentes uma da outra. Tipicamente esses modelos são usados com variáveis incondicionais ou de valores discretos, mas extensões para casos especiais, tais como as densidades Gaussianas, e variáveis de valor real são também possíveis. Os critérios de avaliação do modelo são tipicamente Bayesianos na forma e a estimativa de parâmetros pode ser uma mistura de uma estimativa de forma fechada e métodos iterativos dependendo se uma variável é diretamente observada ou escondida. A procura do modelo pode consistir de métodos como o *hill-climber* sobre várias estruturas gráficas. Métodos que utilizem conhecimentos a priori, tais como ordenação parcial das variáveis baseadas em relações causais, podem ser bastante úteis em termos da redução do espaço de procura do modelo. Embora ainda primariamente na fase de pesquisa, métodos gráficos são de particular interesse para a KDD uma vez que a forma gráfica se torna fácil para a interpretação humana.

II.4.5. Modelos de Aprendizagem Relacional

Enquanto árvores de decisão e regras tem uma representação restrita à lógica proposicional, a aprendizagem relacional (também conhecida como programação de lógica indutiva) usa a linguagem de padrões mais flexível da lógica de primeira ordem. Um aprendiz relacional pode facilmente encontrar fórmulas tais como $X = Y$. A maior parte das pesquisas em métodos de avaliação de modelos para aprendizado relacional são lógicas por natureza. A força de representação extra dos modelos relacionais vem da significativa demanda computacional em termos da procura.

Existe assim, um amplo espectro de métodos e algoritmos de extração de conhecimento, o que torna essa resumida abordagem inevitavelmente limitada em alcance: existem várias técnicas de extração de conhecimento, métodos particularmente especializados para tipos particulares de dados e domínios, que não foram especificamente mencionados nessa discussão.

II.5. Desafios em Pesquisa e Aplicações para a Descoberta de Conhecimento dos Bancos de Dados

Mostra-se a seguir o esboço de algumas das situações com as quais os profissionais em KDD estão sempre lutando:

- **Grandes Bancos de Dados** - Bancos de dados com centenas de campos e tabelas, milhões de registros, e tamanhos de muitos gigabytes são bastante comuns e de terabytes estão começando a aparecer. Já existem algoritmos eficientes para enumerar todas as regras de associação, levando-se em conta limites de sigilo sobre grandes banco de dados. Outras soluções possíveis incluem amostragem, métodos de aproximação, e processamento paralelo em massa.
- **Grande Dimensão** - Além de existir um grande número de registros no banco de dados, pode existir também um grande número de campos (atributos, variáveis) fazendo com que a dimensão do problema seja grande. Um conjunto de dados dimensionalmente grande cria problemas em termos do aumento do espaço de procura para indução do modelo, fazendo-o tender a uma forma combinatorialmente explosiva. Em adição, aumenta as chances de um algoritmo de extração de conhecimento encontrar padrões espúrios que não são válidos. As abordagens a esses assuntos incluem métodos de reduzir a dimensão efetiva do problema e a utilização de conhecimento prévio para identificar variáveis irrelevantes.
- **Super-Adequação** - Quando um algoritmo procura pelos melhores parâmetros para um modelo particular usando um conjunto limitado de dados, ele pode se super-adequar aos dados, resultando em um desempenho pobre do modelo

nos dados de teste. As soluções possíveis incluem validação cruzada, regularização, e outras estratégias estatísticas sofisticadas.

- **Determinação do Significado Estatístico** - Um problema relacionado a super-adequação ocorre quando o sistema está procurando entre vários modelos possíveis. Por exemplo, se um sistema testa N modelos a um nível de significância de 0.001, então na média, com dados puramente randômicos, $N/1000$ desses modelos vão ser aceitos como significativos. Esse ponto é frequentemente esquecido por várias considerações iniciais em KDD. Uma maneira de lidar com esse problema é usar métodos que ajustem a estatística de teste com uma função da procura.

- **Mudança de Dados e Conhecimento** - A mudança rápida dos dados pode tornar padrões previamente descobertos em padrões inválidos. Em adição, as variáveis avaliadas em uma dada aplicação podem ser modificadas, apagadas, ou acrescidas de novos valores através do tempo. As soluções possíveis incluem métodos incrementais para a atualização de padrões e tratamento de mudanças apenas para a procura de mudanças de padrões.

- **Dados Incompletos ou Com Ruído** - Esse problema é especialmente crítico em bancos de dados comerciais. Atributos importantes podem estar faltando se o banco de dados não foi planejado levando em conta a descoberta de conhecimento. As soluções possíveis incluem estratégias estatísticas mais sofisticadas para identificar variáveis escondidas e dependências.

- **Relacionamento complexo entre os campos** - Atributos ou valores hierarquicamente estruturados, relações entre atributos, e outros meios mais sofisticados para a representação do conhecimento do conteúdo de um banco de dados requerem algoritmos que possam utilizar eficientemente tal conhecimento. Historicamente, algoritmos de extração de conhecimento têm sido desenvolvidos para registros de valor de atributos simples, embora novas técnicas para derivar relações entre variáveis tenham sido desenvolvidas.

- **Inteligibilidade dos Padrões** - Em várias aplicações é importante fazer com que os padrões sejam bem compreendidos. As soluções possíveis incluem representações gráficas, estruturação de regras com gráficos acíclicos diretos, geração de linguagem natural, e técnicas para a visualização dos dados e do conhecimento.
- **Interação com o usuário e conhecimento prévio** - Vários métodos e ferramentas KDD correntes não são verdadeiramente interativos e não podem incorporar facilmente o conhecimento prévio sobre um problema exceto de uma maneira simples. O uso do conhecimento do domínio é importante em todos os passos do processo KDD. As Aproximações Bayesianas utilizam probabilidades e distribuições prévias dos dados como uma forma de codificar o conhecimento prévio.
- **Integração com outros sistemas** - Um sistema de descoberta *stand-alone* pode não ser muito útil. Integrações típicas incluem a integração com um DBMS (via uma interface query), integração com planilhas e ferramentas de visualização.

II.6. Aspectos Mais Complexos do Processo KDD

II.6.1. Descoberta da Tarefa

Uma verdade sobre as explorações em KDD é que o cliente define sempre o problema ou o objetivo como se ele fosse claro e focalizado, mas sempre se justifica uma investigação mais profunda. Em outras palavras, os requisitos da tarefa – e assim, para qualquer aplicação que possa resultar, uma vez que a tarefa KDD básica é concluída – devem ser planejados empregando-se o tempo com o usuário e várias partes da organização do cliente.

Para que o objetivo real da tarefa apareça é necessário um aprofundamento em questões já levantadas, e que se empregue tempo peneirando dados em estado natural e se entendendo a sua forma, o seu conteúdo, o seu papel organizacional, e as suas origens. Embora a pessoa ou organização que precisa que a descoberta seja feita tenha ocasionalmente as perguntas certas a princípio, o que foi considerado inicialmente o objetivo é apenas o ponto de partida.

Esse processo inicial, que pode ser chamado de preparatório tende a tomar muito tempo e ser bastante difícil, mas sem ele, é muito fácil perder tempo respondendo às perguntas erradas.

II.6.2. Descoberta dos Dados

Como um complemento para o tempo empregado com o cliente para o entendimento de qual é a necessidade de aplicação da descoberta do conhecimento, analistas precisam inevitavelmente empregar tempo fazendo uma triagem dos dados, para ter uma idéia de como os dados são e qual o campo que eles cobrem. Embora o objetivo principal do empreendimento KDD venha do subprocesso de descoberta de tarefa já descrito, ele não pode ser completamente fechado sem um entendimento detalhado da estrutura, abrangência e qualidade dos dados.

II.6.3. Triagem dos Dados

Uma outra verdade sobre as aplicações reais do KDD é que os dados dos clientes praticamente sempre apresentam problemas. Os dados podem ter sido coletados de uma forma onde campos de registros não preenchidos sejam encontrados, erros na entrada dos dados tenham sido cometidos, etc. Como resultado, um processo KDD não pode ser bem sucedido sem um esforço sério para a “limpeza” dos dados. Sem a fase da descoberta dos dados já discutida, o analista não teria idéia se a qualidade dos dados poderia suportar a tarefa; normalmente é necessário um trabalho sério para colocar os dados em forma para a análise.

Pesquisando-se entre os analistas, foi descoberto que o método mais comumente utilizado para verificar a exatidão dos dados é extrair (se possível) o mesmo elemento a partir de diversas fontes e comparar os resultados. O conhecimento de apoio do especialista tem um papel crucial na triagem dos dados resultante das comparações. Em algumas situações também é possível implementar procedimentos de “edição dos dados” que limpem os dados antes que eles sejam carregados no banco de dados.

A triagem dos dados é uma faca de dois gumes. Ela é quase sempre necessária por causa da qualidade inevitavelmente baixa dos dados, mas ocasionalmente o que parece uma anomalia a ser desprezada mostra ser um indicador crucial de um fenômeno interessante dentro do domínio. Em outras palavras, o que parece uma anomalia a ser desconsiderada pode na realidade ser um ponto no qual vale a pena focalizar.

A triagem dos dados é também um problema de lidar com sintomas que podem acontecer de novo se algum processo básico para a coleta dos dados apresenta defeito. Se os dados são estatísticos e não são atualizados, o processo de limpeza desse estágio vai atuar e resolver o problema. Mas se ele é atualizado da mesma forma em que o banco de dados inicial foi criado, problemas contínuos de qualidade dos dados ocorrerão. Como resultado, o que parece superficialmente um simples exercício de extração de conhecimento poderia na realidade induzir a um exame organizacional e estrutural mais aprofundado para produzir dados que possam ser coletados e analisados de modo fidedigno.

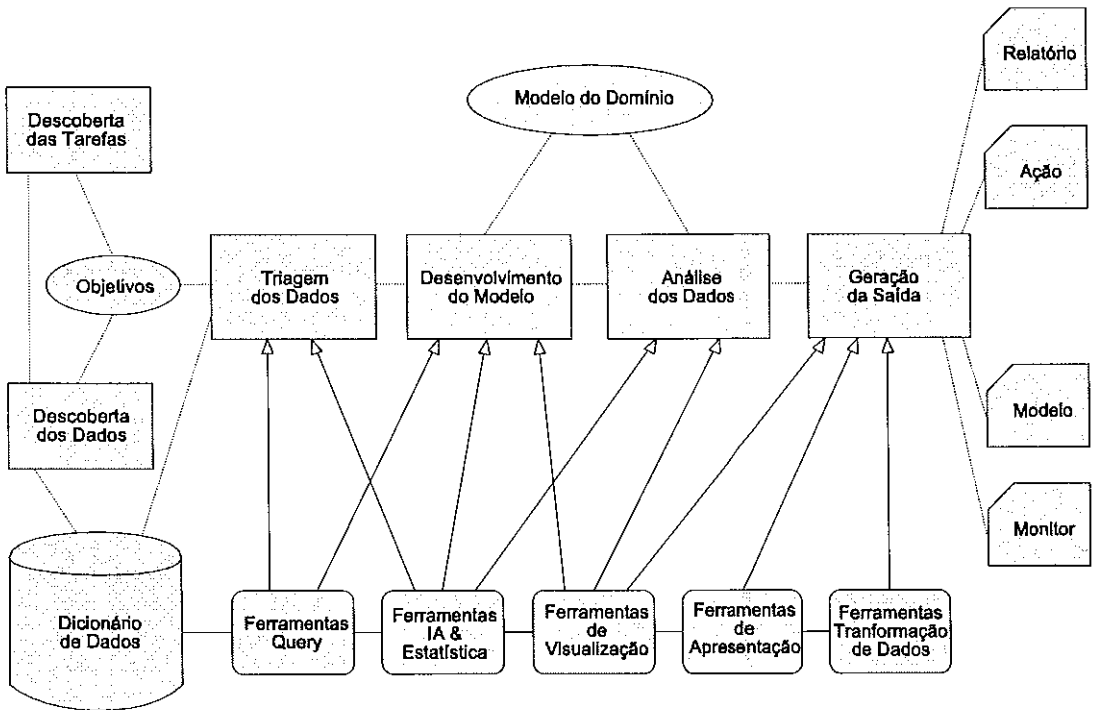
II.6.4. Conhecimento de Apoio

Finalmente, é importante incluir o papel do conhecimento de apoio em um modelo mental do domínio do processo KDD. Grande parte desse conhecimento encontra-se apenas na mente dos especialistas, mas algumas técnicas de análise podem tirar proveito do conhecimento formalmente representado no processo de adequação dos dados a um modelo.

Pode-se considerar, de alguma forma, um dicionário de dados, restrições de integridade, e várias formas de meta-dados a partir do DBMS, como conhecimento de apoio ao processo, mesmo se eles são usados apenas informalmente pelo analista. Às vezes apenas pela habilidade de representar meta-dados, ou resumir descrições de conceitos ou elementos dos dados, dá ao usuário a habilidade de fornecer entradas mais ricas aos métodos de análise dos dados.

A figura II.6 ilustra todo o processo KDD como foi descrito até aqui. Ele inclui entradas no processo, tais como os dados em estado natural, dicionários de dados, e um modelo de domínio de apoio; várias saídas; os passos do processo-chave; e as

ferramentas de suporte genéricas. Na figura II.6 as setas mostram o fluxo geral através do processo, mas está claro que os passos podem ser repetidos várias vezes e loops ao longo do processo são comuns.



Legenda

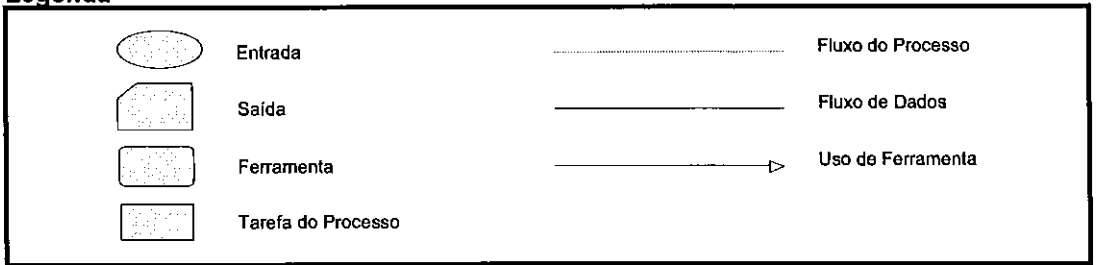


Figura II.6. O processo KDD completo

III. Uma Abordagem Geral sobre Sistemas de Extração de Conhecimento

O objetivo do *data mining* é gerar sistemas que sozinhos sejam capazes de extrair informações úteis dos bancos de dados. *Data mining* é, portanto, uma forma de extração de conhecimento que vai além de simples *queries* e fornece ao final da análise dos dados, informações sobre sua estrutura, vantagens e desvantagens a serem aproveitadas ou ignoradas. Além disso, é possível associar a essa análise o conhecimento do especialista.

Os estudos para extração de conhecimento em bancos de dados começaram utilizando como base métodos já conhecidos, como as árvores de decisão e a formulação de regras. Algoritmos bastante conhecidos e utilizados nessa área, como o ID3 e o C4.5 se basearam neles. Conforme a pesquisa foi avançando, outros métodos também foram utilizados, como os algoritmos genéticos, as redes neurais, os sistemas fuzzy, além dos métodos probabilísticos, etc.

Essa abordagem, entretanto, não tem a pretensão de dissertar sobre todos os métodos de extração de conhecimento. Apenas alguns métodos foram selecionados e são descritos a seguir.

III.1. ID3

O sistema que teve o maior impacto na pesquisa a respeito de aprendizado por máquina nos últimos anos é o ID3, desenvolvido na primeira metade dos anos 80 por Quinlan [8,9]. O ID3 utiliza o Método de Indução em Árvores de Decisão, e é um sistema de aprendizado supervisionado que constrói árvores de decisão a partir de um conjunto de exemplos. Esses exemplos são tuplas, onde o domínio de cada atributo nessas tuplas é limitado a uma pequena quantidade de valores, simbólicos ou numerais.

Versões anteriores do sistema ID3 geravam descrições para duas classes, isto é, exemplos positivos e negativos, mas essa restrição foi removida em sistemas mais atuais. As classes têm que ser mutuamente disjuntas: não existem exemplos

inconsistentes. O ID3 gera descrições que classificam cada objeto no conjunto de treinamento corretamente, isto é, geram regras de classificação fortes.

III.1.1. Espaço de Procura

O conhecimento é representado como uma árvore de conhecimento. O espaço de procura para um problema particular consiste de todas as árvores que podem ser construídas com atributos e valores no conjunto de teste. Para atravessar esse espaço, uma operação de transformação é definida: estender a árvore substituindo uma folha por uma nova sub-árvore (de profundidade um).

Entre todas as árvores no espaço de procura, o sistema precisa achar a melhor árvore. A função de qualidade, isto é, a qualidade de uma árvore, depende tanto da exatidão da classificação, quanto do tamanho da árvore. Entre árvores que classificam todos os objetos no conjunto de teste corretamente, as que são mais simples, são preferíveis. A racionalidade por trás desse último conceito é que a árvore de decisão captura alguns relacionamentos significativos entre as classes de objetos e os valores de seus atributos.

III.1.2. Algoritmo de Procura

O sistema ID3 usa uma estratégia *top-down* que procura apenas parte do espaço de procura, garantindo que uma simples – mas não necessariamente a mais simples – árvore seja encontrada. Uma árvore é construída como a seguir:

1. um atributo é selecionado como a raiz de uma árvore, e os ramos são criados para todos os valores diferentes que esse atributo possa ter;
2. a árvore é usada para classificar o conjunto de treinamento. Se todos os exemplos numa folha particular pertencem à mesma classe, a folha é classificada com a classe. Se todas as folhas são classificadas com a classe, o algoritmo termina;

3. de outra forma, o nó é classificado com o atributo que não ocorre no caminho para a raiz, e ramos são criados para todos os valores possíveis. O algoritmo continua com o passo 2.

O algoritmo acima sempre cria uma árvore que classifica todos os objetos num conjunto de exemplos corretos, mas essa árvore não é necessariamente simples. Uma árvore simples pode ser gerada por uma seleção adequada de atributos. No ID3, uma heurística baseada na informação é usada para selecionar esses atributos. A heurística seleciona o atributo fornecendo o maior ganho de informação, isto é, o atributo que minimiza a informação necessária nas sub-árvores resultantes para classificar os elementos.

Definem-se duas classes: a classe P contendo todos os exemplos positivos, e uma classe N para os exemplos negativos. Fazendo-se com que o conjunto de exemplos S contenha p elementos da classe P e n elementos da classe N . Na teoria da informação, uma medida é definida para a quantidade de informação necessária para decidir se um exemplo arbitrário em S pertence a P ou a N :

$$I(p,n) = -\frac{p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n}$$

Note que $I(p,n)$ depende de p e n apenas, e que $I(p,n) = 0$ para $p=0$ ou $n=0$, e $I(p,n) > 0$ caso contrário. Assume-se que usar a atribuição A como a raiz da árvore particionará S em conjuntos $\{S_1, S_2, \dots, S_v\}$. Se S_i contém p_i exemplos de P e n_i exemplos de N , a informação necessária para decidir se um elemento pertence a P ou N é $I(p_i, n_i)$. Assim a informação -- necessária para classificar um elemento de S usando uma árvore com atributo A como raiz -- é a média ponderada da informação, necessária para classificar objetos em todas as sub-árvores S_i :

$$E(A) = \sum_{i=1}^v \frac{p_i + n_i}{p+n} I(p_i, n_i)$$

O atributo A é selecionado de forma que o ganho de informação é máximo, ou seja, $E(A)$ é mínimo. Entretanto, um problema conhecido com relação a esse critério é que ele tende a favorecer atributos com vários valores.

III.1.3. Atributos Numéricos

Uma condição de um atributo, isto é, um nó interno de uma árvore de decisão, é um teste no valor de um atributo, com ramos para todos os valores possíveis. Embora isso seja conveniente para atributos simbólicos, seria útil se pudesse ser testado em variações de atributos numéricos. Uma extensão do algoritmo ID3, chamada C4.5 [10] permite testes sobre as desigualdades dos atributos numéricos, tal que $A_i \leq N$ e $A_i > N$ com duas saídas possíveis (ramos).

O ganho de informação de tal teste é computado da seguinte maneira: os exemplos são primeiramente sorteados considerando os valores dos atributos. Existe apenas um número finito desses atributos, digo $\{v_1, v_2, \dots, v_m\}$. Nesse conjunto, existe $m-1$ separações possíveis do atributo, que devem ser todas examinadas. Pode parecer dispendioso examinar todas as $m-1$ separações possíveis, mas, quando os exemplos forem sorteados como acima, isso pode ser executado em um passo, atualizando as distribuições de classes para a esquerda e para a direita do limite em cada rodada. Para cada limite possível, o ganho de informação é computado, e usado no processo de seleção do próximo teste, como descrito acima.

III.1.4. Valores de Atributos Agrupados

Um outro teste implementado no sistema C4.5 é um teste se o valor de um atributo pertence a um conjunto específico de valores, tais como $A_i \in \{v_1, v_2, \dots, v_n\}$. O nó é classificado com o atributo, e os ramos são classificados preferencialmente com conjuntos de valores, que com valores sozinhos. Para algumas situações, isso pode reduzir a complexidade da árvore resultante, quando os valores são relacionados de alguma forma.

Entretanto, para m valores diferentes, existem $2^{m-1}-1$ partições binárias diferentes, o que exclui a possibilidade de uma procura exaustiva para o melhor

particionamento. O C4.5 usa uma procura *bottom up*, baseada na fusão iterativa de grupos. Os grupos iniciais são apenas valores individuais do atributo em consideração e, a cada ciclo, o C4.5 avalia as consequências de fundir cada par de grupos. (Qualquer divisão de valores em grupos é refletida na separação de exemplos por essa atribuição, e assim na correspondente informação de separação e ganho). O processo continua até que apenas dois grupos de valores permaneçam, ou até que o ganho não possa ser melhorado pela fusão.

III.1.5. Ruído

O ruído pode afetar tanto os valores dos atributos quanto as informações sobre as classes [11]. Um conjunto de exemplos corrompidos causa dois problemas:

1. Não é mais possível gerar uma árvore que classifica todos os exemplos corretamente, isto é, a condição de consistência é violada. Quando as classes se superpõem, no passo 3 do algoritmo, ocorrerá um conjunto S_i de objetos que não pertencem à mesma classe, e não existem atributos que possam ser usados para levar o ramo até a folha.

A solução consiste em classificar a folha com a classe que domina S_i , ou com todas as classes em S_i , junto com as suas probabilidades.

2. Exemplos corrompidos podem fazer com que a árvore cresça para acomodar esses exemplos. Por exemplo, geralmente o algoritmo não vai ramificar num atributo A que reduza pouco a informação necessária após a ramificação. Entretanto, se os valores para esse atributo estão corrompidos para alguns exemplos, ramificando em A poderia dar um aparente ganho de informação, embora valores em A sejam randômicos, e não formem indicação para a classificação.

Para superar esse problema, pode-se exigir que o ganho de informação exceda um limite. Entretanto, experimentos sugerem que um limite que é alto o suficiente para desclassificar atributos irrelevantes, também desclassificará relevantes.

Uma alternativa é o teste χ^2 : se um atributo A é irrelevante para a classe de um objeto em S, isto é, 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), seria:

$$p'_i = p \frac{p_i + n_i}{p + n} = p \frac{|S_i|}{|S|}$$

Se n'_i é o valor correspondente esperado de n_i , a estatística

$$\sum_{i=1}^v \frac{(p_i - p'_i)^2}{p'_i} + \frac{(n_i - n'_i)^2}{n'_i}$$

é aproximadamente χ^2 com $v-1$ graus de liberdade. O procedimento de construção da árvore é modificado de forma que apenas atributos cuja irrelevância possa ser rejeitada com um nível de confiabilidade bastante alto vão ser utilizados. Uma aplicação desse teste é a árvore de poda.

III.1.6. Valores de Atributos Ausentes

Um outro problema diz respeito aos valores ausentes dos atributos, resultando em problemas na construção de uma árvore de decisão, e em problemas surgidos na tentativa de classificar um objeto com valores ausentes.

1. Para a construção de uma árvore, vários métodos foram propostos para lidar com os valores ausentes dos atributos. Pode-se tratar valores ausentes como o valor que mais aparece na sua classe, ou simplesmente descartar exemplos com valores ausentes, ou ainda tratar valores ausentes como um valor especial *desconhecido*. Entretanto, a última técnica aumenta o ganho de informação esperado para um atributo se alguns valores forem *desconhecidos*, o que não é uma propriedade desejável. Por essa razão, uma técnica melhor é proposta, onde a distribuição de valores desconhecidos é assumida como estando em proporção à frequência relativa desses valores em S.

2. Durante a classificação, objetos com valores de atributo ausentes não podem ser classificados, se a ramificação em um desses atributos é necessária. Todos os ramos para esse atributo são então explorados, e para cada caminho resultante, o ID3 estima a probabilidade para que esta seja a escolha correta de valores para os atributos. Essa probabilidade é o produto das frequências relativas dos valores escolhidos para os atributos desconhecidos. Assim, para todas as classes, essas probabilidades são somadas, e o resultado da classificação é a classe com a maior probabilidade. Esse procedimento oferece uma degradação muito suave quando a incidência dos valores desconhecidos aumenta.

III.1.7. Janelas

A cada ciclo do algoritmo, o conjunto de treinamento deve ser questionado para determinar o ganho de informação para os atributos. Note que o questionamento deve ser feito apenas no subconjunto de exemplos que não foram ainda classificados. Ao invés do conjunto de treinamento inteiro, um subconjunto é escolhido randomicamente – a chamada janela pode ser usada. Usando essa janela, uma árvore é gerada, e todos os exemplos no conjunto de treinamento são classificados usando a árvore. Enquanto nem todos os exemplos são classificados corretamente, alguns dos exemplos classificados incorretamente são adicionados à janela e o processo continua: uma nova árvore é gerada para esses exemplos, entretanto, essa nova árvore é gerada desde o começo.

Resultados experimentais mostram que são encontradas árvores de decisão corretas em poucas iterações, e que esse método é usualmente mais rápido do que formar a árvore usando o banco de dados inteiro.

III.1.8. Aprendizado Experimental

O algoritmo ID3 executa bem qualquer tarefa quando o conjunto de treinamento inteiro é fornecido de uma só vez. Entretanto se os exemplos são fornecidos um de cada vez, o algoritmo ID3 pode ainda ser usado, porém construiria uma nova árvore de decisão a partir do início, toda vez que um novo exemplo for

observado. Para tais “tarefas de aprendizado serial”, seria preferível um algoritmo incremental, assumindo que é mais eficiente revisar uma árvore existente que gerar uma nova árvore cada vez.

Uma adaptação do algoritmo ID3, chamado ID5R [12], não cria uma nova árvore para cada novo exemplo, mas ao invés disso reestrutura a árvore para fazê-la consistente com o exemplo corrente e todos os prévios. Esses exemplos prévios são guardados, mas não são reprocessados, são usados apenas para reestruturar finalidades. ID5R constrói exatamente a mesma árvore que o sistema ID3 para o mesmo conjunto de dados.

Mesmo para tarefas de treinamento não-seriais, algoritmos incrementais podem ter um desempenho superior. Experimentos mostram que selecionar situações de treinamento uma de cada vez, baseadas em inconsistências na árvore corrente, leva a uma árvore menor que seleciona várias situações de uma vez, como feito no ID3. Esse último encontrará a árvore idêntica se a janela inicial contém apenas um exemplo e a janela é acrescida de uma situação de cada vez. Mas como é comentado acima essa aproximação é proibitivamente onerosa.

III.1.9. Conclusão

O sistema tem um desempenho muito bom para várias áreas de aplicação, tais como área médica, aplicações em inteligência artificial, etc. A exatidão da classificação é alta. Entretanto, o sistema não faz qualquer uso do conhecimento do domínio. Além disso, árvores não são fáceis de entender, entretanto, podem ser transformadas em regras de decisão.

III.2. AQ15

O sistema AQ15 de Michalski [13,14] é um sistema de aprendizado indutivo que gera regras de decisão, onde a parte condicional é uma fórmula lógica. Uma característica especial desse sistema é a descrição construtiva, isto é o uso do conhecimento do domínio para gerar novos atributos, que não estão presentes nos dados de entrada.

Como o ID3, e vários outros sistemas de aprendizado por máquina, o AQ15 é primeiramente projetado para a construção de regras fortes (strong rules), isto é, para cada classe, uma regra de decisão é produzida de forma a cobrir todos os exemplos positivos e nenhum dos negativos. O sistema manuseia exemplos incompletos e inconsistentes pelo pré e pós-processamento. O número e tamanho das regras descobertas é dramaticamente reduzido pela aplicação de uma técnica de pós-processamento, chamada truncamento de regras, que não afeta a exatidão da classificação.

Exemplos no conjunto de treinamento são vetores dos valores de atributos. Atributos podem ser de três tipos: nominal, linear ou hierarquicamente estruturado.

III.2.1. Espaço de Procura

As regras são representadas em notação VL₁(Sistema Lógico de Valor Variável 1), cálculos atribucionais lógicos de múltiplo valor com variáveis típicas. Um seletor é uma condição de valor de atributo que relaciona um atributo a um valor ou a uma disjunção de valores, isto é, 'cor = vermelho \wedge verde'.

Uma conjunção de seletores é chamada um complexo (isto é uma descrição do conjunto). A parte condicional de uma regra de decisão é formada de um complexo de disjunções, chamada de uma *cobertura*.

Quando se constrói uma regra de decisão, o AQ desempenha uma procura heurística pelo espaço de todas as expressões lógicas para determinar àquelas a serem levadas em conta para todos os exemplos positivos e nenhum dos negativos. Porque existem (nos problemas de máquina de aprendizado) usualmente várias regras fortes, o objetivo do AQ é encontrar a mais preferível. O *critério de preferência* é definido pelo usuário, para definir as necessidades do domínio de aplicação.

As operações na estrutura do conhecimento são a adição de um complexo à cobertura de uma regra, e a *interseção* de um conjunto de complexos com um conjunto de seletores. A interseção de dois conjuntos A e B é um conjunto de todas as combinações de conjunções de elementos de ambos os conjuntos, isto é $\{x \wedge y \mid x \in A, y \in B\}$.

III.2.2. Algoritmo de Procura

O sistema gera uma regra de decisão para cada classe como retorno, usando um algoritmo *top down* de procura (na realidade uma procura de suporte: vários *hill climbers* em paralelo). A cada passo do algoritmo, o melhor complexo é adicionado à cobertura. Cada passo começa concentrando a atenção em um exemplo positivo selecionado – chamado a “semente”. O algoritmo gera um conjunto de todos os complexos (uma “estrela”) que “cobre” a semente e não “cobre” quaisquer exemplos negativos e então seleciona o melhor complexo da estrela de acordo com o critério definido pelo usuário. Esse complexo é adicionado à cobertura. O algoritmo de cobertura básica é:

While a cobertura parcial não cobre todos os exemplos positivos
do

1. selecione uma semente, isto é um exemplo positivo não coberto,
2. gere uma estrela, isto é, determine o máximo possível, os complexos gerais cobrindo a semente e nenhum exemplo negativo (veja abaixo),
3. selecione o melhor complexo, como definido pela função de qualidade,
4. adicione o complexo à cobertura.

O algoritmo começa com uma cobertura inicial que é ou vazia, previamente aprendida, ou fornecida pelo usuário. O passo 2, gerar uma estrela, consiste de:

While uma estrela parcial cobre exemplos negativos
do

1. selecione um exemplo negativo coberto,

2. gere uma estrela parcial: todos os seletores gerais máximos que cobrem a semente e excluem o exemplo negativo,
3. gere uma estrela parcial nova pela interseção da estrela parcial corrente com a estrela parcial construída até então,
4. balanceie a estrela parcial, isto é retenha apenas os melhores complexos da maxi-estrela.

Se o procedimento de geração de estrela trabalhasse exaustivamente, o espaço de procura para cobertura poderia crescer muito rapidamente. Por essa razão, o usuário pode definir uma *maxi-estrela* de parâmetro, que controla quantos complexos disjuntos podem ser guardados em uma estrela parcial. Apenas os melhores complexos são retidos, de acordo com o critério especificado pelo usuário tal como ‘maximizar o número de exemplos positivos cobertos e exemplos negativos excluídos’, ou ‘minimizar o número de seletores’.

Assim, para uma estrela parcial dada, o procedimento acima computa a qualidade dos complexos na interseção, retém a maxi-estrela desses (a largura de suporte), e faz a interseção desses de novo, até que nenhum dos complexos negativos seja coberto.

III.2.3. Dados Inconsistentes e Com Ruído

O conjunto de treinamento pode ser inconsistente, isto é, alguns exemplos positivos e negativos são idênticos, assim uma regra correta pode nunca ser construída. AQ fornece três opções: ele trata esses exemplos inconsistentes, como exemplos positivos, exemplos negativos, ou simplesmente os abandona. Se a informação estatística sobre a probabilidade dos exemplos inconsistentes está disponível, eles são pré-classificados de acordo com a probabilidade máxima.

Quando o conjunto de treinamento contém exemplos incorretos, pode ser vantajoso aplicar as regras construídas de uma forma probabilística. Assim, à parte de uma combinação exata, onde se testa se uma situação satisfaz a descrição, pode-se

distinguir um outro método de testar a associação de classes: combinação *flexível*, onde o grau de similaridade (ou proximidade conceitual) determina a classe.

Se a combinação flexível é usada, é possível simplificar a descrição pela remoção de um ou mais complexos. Essa técnica, chamada *truncamento*, remove o complexo que cobre a menor parte dos exemplos. Se o conjunto de treinamento possui ruído, esses complexos podem ser um indicativo de erros nos dados. A regra resultante não é mais a correta, mas a probabilística, uma vez que similaridades para classes diferentes são comparadas. Essas regras, porém, são simples de entender, e o tamanho da base de conhecimento reduz.

A cada passo do truncamento, a exatidão da classificação é medida. Cada passo produz uma relação diferente entre a complexidade da descrição e a exatidão da regra. Em algum passo, o melhor resultado geral pode ser obtido. Vários experimentos mostram que regras podem ser truncadas sem afetar a exatidão da classificação.

III.2.4. Indução Construtiva

O sistema AQ usa o conhecimento do domínio, expresso em forma de regras, para gerar novos atributos. Essas regras são de dois tipos: regras lógicas que definem valores de novas variáveis, e regras aritméticas que introduzem novas variáveis como funções de atributos numéricos. O sistema tenta usar essas novas regras para produzir regras de decisão melhores.

As regras lógicas representam definições de classe de segundo plano, restrições entre classes, hierarquias de generalização, dependências causais, etc. Classes e suas descrições, aprendidas pelo programa, são adicionadas ao estoque de regras.

III.2.5. Aprendizado Incremental

O AQ15 tem uma facilidade de aprendizado incremental. O usuário pode fornecer hipóteses de decisão como regras iniciais. O sistema implementa o método de aprendizado com memória completa, onde ele lembra todos os exemplos que foram vistos até então, assim como as regras que ele formou. Por esse método, ao contrário

do aprendizado com memória parcial, novas regras de decisão são garantidamente corretas com relação a todos (antigos e novos) exemplos de aprendizado.

III.2.6. Conclusão

O AQ15 concentra-se na descoberta de regras fortes (strong rules), e dados inconsistentes e com ruído são manuseados pelo pré-processamento (exemplos inconsistentes), e pós-processamento (truncamento de regras).

O sistema tem sido testado para vários domínios médicos, tais como diagnóstico em linfografias, prognóstico do câncer de mama, e a localização de tumor primário. Ele descobriu regras de decisão no nível de exatidão dos especialistas [14]. Os conjuntos de treinamento são pequenos, tipicamente umas poucas centenas de exemplos.

Um resultado potencialmente significativo desses testes é a aplicação do método proposto de truncamento de regras e combinação flexível (isto é, construindo alguns tipos de regras probabilísticas), diminuindo drasticamente a complexidade das regras sem afetar a exatidão da classificação.

III.3 DBLearn

O sistema DBLearn, desenvolvido por Cai *et al.* [15], usa o conhecimento do domínio para gerar descrições para subconjuntos pré-definidos em uma base de dados relacional. As características especiais desse sistema são sua estratégia de procura *bottom-up*, o uso do conhecimento do domínio na forma de hierarquias dos valores de atributos, e o uso da álgebra relacional. O conjunto de exemplos é uma tabela relacional, isto é, um conjunto de n tuplas.

III.3.1 Espaço de Procura

O sistema usa tabelas relacionais como estruturas de conhecimento: para cada classe, ele constrói uma tabela relacional, cujos nomes dos atributos (colunas) são um subconjunto dos nomes dos atributos no conjunto de exemplos. Uma tupla pode ser

vista como uma fórmula lógica, formada pela conjunção de seus pares atributo-valor. Uma tabela, um conjunto de tuplas, pode assim ser vista como uma disjunção dessas conjunções. Portanto, o nó de partida do espaço de procura é o conjunto de exemplos e o objetivo é generalizar essa tabela para uma descrição da classe: uma tabela muito menor, cobrindo todos os exemplos pertencentes a essa classe. O número máximo de tuplas na tabela resultante é especificado por um limite definido pelo usuário.

Existe uma relação no tamanho desse limite. Um limite pequeno leva a regras simples com poucas disjunções, mas pode resultar em super generalização e perda de algumas informações valiosas. Entretanto, um limite grande pode preservar algumas informações úteis, mas isso pode resultar em regras relativamente complexas com várias disjunções e alguns resultados inadequadamente generalizados.

Os domínios de alguns atributos são parcialmente ordenados, e formam uma treliça semi-superior, isto é todos os valores são menores que o maior valor 'ANY'. Os valores formam uma hierarquia IS_A, onde valores denotam generalizações dos valores abaixo deles.

Duas operações de generalização são usadas para gerar uma descrição da classe de um conjunto de exemplos:

1. Dados Relevantes da Tarefa são selecionados do banco de dados, isto é, uma simples tabela, contendo todos os exemplos positivos possíveis é recuperada.
2. A indução orientada a atributo consiste em aplicar operações de generalização na tabela. Uma heurística simples é utilizada: se existem vários valores distintos num atributo, e um valor de nível mais alto é fornecido, o atributo é generalizado pela substituição de cada valor com o seu valor de nível mais alto. Por outro lado, se existem vários valores distintos para um atributo, mas nenhum valor é fornecido, o atributo é simplesmente reduzido em todas as tuplas da tabela.

As tuplas duplicadas são removidas da tabela, e o processo de generalização continua até que o número de tuplas não é mais que um limite especificado.

3. Se possível, a tabela resultante é simplificada, isto é, transformada numa tabela cobrindo o mesmo conjunto. Por exemplo, se algumas tuplas têm valores idênticos para todos, exceto um, os atributos e os valores para esses atributos formam todos os

descendentes de um símbolo na hierarquia, essas tuplas podem ser substituídas por uma simples tupla, onde o atributo variante tem o símbolo como valor.

4. A tabela é transformada em uma fórmula lógica.

III.3.2. Algoritmo de Aprendizagem: regras consistentes

O outro algoritmo desenvolvido por Cai *et al.* [16] é o aprendizado de regras consistentes que não são necessariamente completas, isto é todos os exemplos cobertos pela regra pertencentes à classe, mas não necessariamente todos os exemplos positivos são cobertos. Esse algoritmo usa tanto exemplos positivos quanto negativos.

O algoritmo é similar ao algoritmo para aprendizado de regras características, apenas o segundo passo é levemente diferente. Ao invés de usar uma simples tabela para exemplos positivos, usa duas tabelas, uma contendo exemplos positivos e outra contendo exemplos negativos. Entretanto, ambas as tabelas podem compartilhar tuplas. Essas tuplas – chamadas de tuplas superpostas – são marcadas, e essas marcas são herdadas quando as tuplas são generalizadas. Uma vez que a generalização pode produzir novas superposições, uma checagem deve ser executada após cada operação de generalização. Finalmente, as tuplas não marcadas em cada tabela formam as descrições de classe para ambas as classes.

III.3.3. Ruído

Ambos os algoritmos são estendidos para lidar com o ruído e exceções. Um pequeno número de exemplos não usuais são considerados como ruído e dados excepcionais, e devem ser descartados. O algoritmo é estendido pela adição de informação quantitativa para o processo de generalização, e a eliminação das tuplas da tabela é baseada em informação.

Um voto especial para o atributo é adicionado para cada tupla na tabela. Esse atributo registra o número de tuplas na tabela inicial que são generalizadas para essa

tupla na tabela corrente. Baseados nos votos, dois pesos são definidos para cada tupla na tabela generalizada:

1. O peso-t é uma medida para a generalidade da descrição representada por essa tupla. Esse peso é a proporção do número de tuplas coberto pela tupla q_j para o número de tuplas cobertas pela tabela inteira $\{q_1, \dots, q_n\}$:

$$peso_t(q_j) = \frac{votos(q_j)}{\sum_{i=1}^n votos(q_i)}$$

Um peso-t alto implica que a tupla é induzida a partir da maioria dos dados, e um peso-t baixo implica que a tupla é induzida de alguns casos raros, excepcionais. As tuplas com um peso-t baixo são portanto removidas da tabela.

2. O peso-d é uma medida para a habilidade discriminativa da descrição representada pela tupla. Esse peso é a proporção do número de tuplas que pertencem à classe e são cobertas por uma tupla, para todas as tuplas que são cobertas por essa tupla.

O peso-d só pode ser construído se o número de classes é dois ou mais, isto é, apenas para regras consistentes. Um peso-d alto indica que uma tupla é generalizada principalmente das tuplas originais na classe de referência, um peso-d baixo indica que a tupla é principalmente derivada da classe contrastante e de apenas uns poucos (provavelmente excepcionais) casos na classe de referência. Por isso, apenas tuplas com um peso-d alto são incluídas na regra de classificação.

III.3.4. Aprendizado Incremental

O Aprendizado Incremental é implementado pelo uso da informação dos votos. Se uma tupla é adicionada ao banco de dados, ela é generalizada para a mesma classe das tuplas da tabela generalizada. Se a tupla já está na tabela, a informação dos votos para essa tupla é incrementada. De outra forma, ela é inserida como uma nova

tupla na tabela generalizada, e se o número de tuplas nessa tabela excede o limite, uma generalização da tabela a mais é executada.

III.3.5. Conclusão

O sistema DBLearn é um sistema relativamente simples, usando duas operações de generalização para construir descrições. É um exemplo instrutivo do uso do conhecimento do domínio no processo de generalização.

O sistema usa um processo de generalização orientado a atributo. Portanto, uma descrição é uma disjunção de conjunções de valor de atributo. Cada uma dessas conjunções consiste de condições sobre os mesmos atributos. Isso limita o conjunto de descrições que pode ser construído. O desempenho do sistema é bom, a complexidade-tempo do algoritmo é $O(N \log N)$, onde N é o número de tuplas na relação inicial.

No aprendizado inicial, apenas a informação dos votos é atualizada, e possivelmente novas tuplas são adicionadas à descrição. Não é claro se esse tipo de aprendizado incremental é correto, isto é, a descrição resultante não precisa ser a mesma que a descrição que seria construída do começo usando a base de dados atualizada. No último caso, uma outra sequência de operações de generalização poderia ser executada (porque o banco de dados foi atualizado), resultando em uma descrição diferente, que não pode ser construída pela atualização da informação dos votos na descrição inicial.

A aproximação DBLearn pode ser muito bem integrada com as operações do banco de dados, uma vez que operações de generalização são orientadas a conjunto, e ambos, dados e conhecimento são representados como tabelas relacionais. De fato, ambas as operações de generalização são idênticas: cair uma condição é equivalente a substituir os valores por um simples valor de topo qualquer, assim ascendendo numa árvore de generalização. Uma extensão útil não seria generalizar todos os valores para um atributo particular, mas permitir uma generalização parcial, isto é, substituir apenas alguns dos valores por um valor de alto nível.

O sistema foi estendido para lidar com o ruído, e construir regras probabilísticas (usando informação de votos). Técnicas para lidar com o ruído são

primitivas, aplicação de regras estatísticas, tal como o teste χ^2 , pode melhorar os resultados. Não existem técnicas para lidar com valores de atributos ausentes.

III.4. CN2

III.4.1. Resumo:

Sistemas que induzem descrições de conceitos a partir de exemplos são ferramentas valiosas que auxiliam os sistemas especialistas na tarefa de aquisição de conhecimento. Este ensaio apresenta uma descrição e uma avaliação empírica de um novo sistema de indução, o CN2, projetado por Clark e Niblett [17] para induzir eficientemente regras de produção simples e compreensíveis em áreas de linguagem descritiva pobre e/ou onde o ruído pode estar presente.

III.4.2. Introdução

Na tarefa de criar sistemas especialistas, sistemas que induzem descrições de conceitos a partir de exemplos provaram-se úteis ao facilitarem a aquisição de conhecimento [18]. Duas famílias de sistemas, baseadas nos algoritmos ID3 [8] e AQ [19], foram especialmente bem-sucedidas. Estes algoritmos básicos assumem que não há ruídos no domínio, buscando uma descrição de conceito que classifique dados de treinamento perfeitamente. Contudo, para que sistemas baseados nestes algoritmos possam ser aplicados no mundo real, são necessários métodos para o tratamento de dados com ruído. Em particular, são necessários mecanismos que evitem a noção de super-adequação entre a descrição do conceito induzido e os dados, relaxando-se a restrição de que a descrição induzida deve classificar os dados de treinamento perfeitamente.

O algoritmo ID3 presta-se a modificações simples, permitindo que esta restrição seja diminuída, devido ao seu tipo de busca, que parte do geral para o específico. Três das técnicas de poda (p.e. [20, 21]), que são utilizadas, por exemplo, nos sistemas C4 [22] e ASSISTANT [23], provaram ser métodos eficazes para se

evitar essa super-adequação. O algoritmo AQ, contudo, não é tão simples de ser modificado, pois depende de exemplos de treinamento específicos durante sua busca. Já estão sendo feitas implementações (p.e. AQ11 [24] e AQ 15 [13]) que tratam de dados com ruído utilizando técnicas de pré e pós-processamento, sem alterar o algoritmo AQ básico. O objetivo ao se criar o CN2 é modificar o algoritmo AQ de modo que esta dependência a exemplos específicos seja anulada e o espaço de regras pesquisado seja ampliado. Como resultado, técnicas estatísticas análogas às utilizadas em poda de árvores podem ser, então, aplicadas na geração de regras if-then, atingindo-se, assim, um algoritmo mais simples.

Pode-se identificar vários requisitos que os sistemas de aprendizagem devem atender caso queiram provar-se úteis em várias situações do mundo real:

Classificação precisa: As regras induzidas devem classificar os novos exemplos de forma precisa, mesmo na presença de ruído.

Regras simples: Para que haja compreensibilidade, as regras induzidas devem ser as mais breves possíveis. Entretanto, na presença de ruído, as regras super-adequadas tendem a ser muito longas. Assim, para induzir regras mais curtas, deve-se deixar de lado a premissa de que as regras induzidas devem ser consistentes com os dados de treinamento. Escolher até que ponto esta premissa pode ser desconsiderada envolve chegar a um meio termo entre a precisão e a simplicidade [25].

Geração eficiente de regras: Caso o conjunto de exemplos seja grande, é importante que o algoritmo esteja preparado para situações mais complexas. Na prática, o desejado é que o tempo usado para a geração de regras seja proporcional ao tamanho do conjunto de exemplos.

Com estes requisitos em mente, este ensaio apresenta uma descrição e uma avaliação empírica do CN2. Ele combina a eficiência e a habilidade de manipular os dados com ruído do ID3 utilizando regras if-then e a estratégia de busca flexível da família AQ. A representação de saída das regras pelo CN2 é um conjunto ordenado de regras if-then, também conhecido como “lista de decisão” [26]. O CN2 usa uma função heurística que encerra a busca durante a construção da regra, baseada em uma estimativa do ruído presente nos dados. Isto resulta em regras que não

necessariamente classificam todos os exemplos de treinamento corretamente, mas que têm bons resultados com dados novos.

Na seção a seguir, descreve-se o CN2 e três outros sistemas utilizados para estudo comparativo. Dentre eles estão: o AQR, a reconstrução que o autor fez do algoritmo AQ de Michalski *et al* [13]; o ASSISTANT de Kononenko *et al* [23], uma variante do ID3; e um classificador Bayesiano simples, utilizado como uma referência para o desempenho dos outros algoritmos. Em cada caso, considera-se a complexidade de tempo dos vários algoritmos.

III.4.3. O CN2 e os Outros Algoritmos Para Estudo Comparativo

O CN2 e os outros algoritmos serão apresentados a seguir. Como o CN2 foi desenvolvido a partir do estudo do ID3 e do AQ, será apresentado em primeiro lugar o sistema baseado em ID3, o ASSISTANT, e o sistema baseado em AQ, o AQR, antes de apresentar o CN2 e abordar sua relação com estes algoritmos.

Caracteriza-se os sistemas com base em três dimensões, a saber:

- A linguagem de representação do conhecimento induzido;
- O mecanismo de execução e avaliação do desempenho das regras;
- O algoritmo de aprendizagem e as heurísticas de busca associadas.

III.4.3.1. O ASSISTANT

O algoritmo ASSISTANT [23] é um descendente do ID3 de Quinlan [8] e incorpora seu mecanismo de poda de árvores para o tratamento de dados com ruído.

Considerando: E um conjunto de exemplos

A um conjunto de atributos descrevendo exemplos

$TE(E)$ um critério de terminação

$IDM(a_i, E)$ uma função de avaliação onde $a_i \in A$

Procedimento ASSISTANT que retorna ÁRVORE:

If: E satisfaz o critério de terminação $TE(E)$ then **retorne** um nó de folha para $ÁRVORE$, rotulado com a classe mais comum de exemplos em E .

Else: determine o atributo $a_{best} \in A$ com o maior valor da função $IDM(a_{best}, E)$. Então, para cada valor v_j do atributo a_{best} , gere sub-árvores usando o $ASSISTANT(E_j)$ onde E_j sejam os exemplos em E com o valor v_j para o atributo a_{best} .

Retorne um nó rotulado como teste no atributo a_{best} com as sub-árvores anexadas.

Tabela 1: O núcleo do algoritmo ASSISTANT

a. A Linguagem de Descrição de Conceitos e sua Interpretação

O ASSISTANT representa o conhecimento adquirido sob a forma de árvores de decisão. Um nó interno de uma árvore especifica um teste de um atributo, sendo que cada galho corresponde a um possível resultado deste teste. Os nós de folha representam a classificação a ser atribuída a um exemplo.

Para classificar um novo exemplo, é traçado um caminho da raiz da árvore de decisões até um nó de folha. A cada nó interno alcançado, segue-se o ramo que corresponde ao valor do atributo testado naquele nó. A classe no nó da folha representa o prognóstico da classe para aquele exemplo.

b. O Algoritmo de Aprendizagem

O ASSISTANT induz uma árvore de decisões, especializando repetidamente os nós de folha de uma árvore que, inicialmente, apresentavam uma única folha. A operação de especialização envolve a substituição de um nó de folha por um teste de atributo e a adição de novas folhas ao nó que corresponde aos possíveis resultados desse teste. A heurística determinará que atributo deverá ser testado e quando a especialização deverá ser encerrada. A Tabela 1 resume este algoritmo.

c. Funções Heurísticas

O ASSISTANT usa uma medida de entropia para orientar o crescimento da árvore de decisões, como foi descrito por Quinlan [8]. Isso corresponde à função *IDM* na Tabela 1. Além disso, o algoritmo pode aplicar um método de corte de árvore com base em uma estimativa de precisão de classificação máxima. Esta técnica estima se o acréscimo de mais um ramo reduziria a precisão da classificação e, em caso afirmativo, encerra a busca (neste cálculo não há parâmetros que possam ser alterados pelo usuário). Este critério de corte corresponde à função *TE* na Tabela 1. Se o ASSISTANT deve gerar uma árvore não-podada, o critério de terminação *TE(E)* será satisfeito se todos os exemplos *E* tiverem o mesmo valor de classe.

III.4.3.2. O AQR

O AQR é um sistema de indução que usa o algoritmo AQ [19] básico para gerar um conjunto de regras de classificação. Muitos sistemas usam esse algoritmo de forma mais sofisticada que o AQR para tornar os prognósticos mais precisos e as regras mais simples (p.e., o AQ11 [24] usa um método mais complexo de interpretação de regras que envolve graus de confirmação). O AQR é uma reconstrução bastante objetiva de um sistema baseado em AQ.

a. A Linguagem de Descrição de Conceitos e sua Interpretação

O AQR induz um conjunto de regras de decisão, uma para cada classe. Cada regra está no formato 'if <cobertura> then prognostica <classe>', onde <cobertura> é uma combinação booleana de testes de atributo, como estão sendo descritos agora. O teste básico de um atributo é chamado de *seletor*. Abaixo, estão exemplos de seletores:

⟨Nublado = sim⟩
⟨Tempo = úmido ∧ chuvoso⟩
⟨Temp > 60⟩

O AQR permite testes no conjunto $\{=, \leq, >, \neq\}$. Um conjunto de seletores é denominado *complexo* e uma disjunção de complexos é chamada uma *cobertura*. Dizemos que uma expressão (um seletor, complexo ou cobertura) *abrange* um exemplo se a expressão for verdadeira para o exemplo. Assim, o complexo vazio (conjunto de testes de atributo zero) abrange todos os exemplos e a cobertura vazia (disjunção ou zeros complexos) não abrange nenhum exemplo. Uma cobertura é armazenada com um valor de classe associado, que representa a classe mais comum dos exemplos de treinamento que ela abrange.

No AQR, um novo exemplo é classificado localizando-se as regras induzidas cujas condições sejam satisfeitas pelo exemplo. Se o exemplo satisfizer a apenas uma regra, a classe prognosticada pela regra é atribuída ao exemplo. Se o exemplo satisfizer a mais de uma regra, a classe mais comum de exemplos de treinamento que foram abrangidos pelas regras será prognosticada. Se o exemplo não for abrangido por nenhuma regra, será atribuído por padrão à classe que ocorreu com mais frequência nos exemplos de treinamento.

b. O Algoritmo de Aprendizagem

O algoritmo de geração de regras AQ já foi descrito várias vezes (e.g. [24, 27, 28]), e o sistema AQR é um exemplo do algoritmo geral. O algoritmo gera uma regra de decisão para uma classe de cada vez. Escolhida uma classe em que se concentrar, se forma uma disjunção de complexos (a cobertura) para servir como a condição da regra para essa classe. Esse processo ocorre em estágios; cada estágio gera um complexo simples, e então remove os exemplos que ele cobre a partir do conjunto de treinamento. Esse passo é repetido até que sejam encontrados complexos suficientes para cobrir todos os exemplos da classe escolhida. O processo total é repetido para uma classe de cada vez. A tabela 2 resume o algoritmo AQR.

c. Funções Heurísticas

As funções heurísticas particulares usadas pelo algoritmo AQ são dependentes da implementação. A heurística usada pelo AQR para escolher o melhor complexo é “maximizar o número de exemplos positivos cobertos”. A heurística usada para aparar a estrela parcial durante a geração de um complexo é “maximizar a soma dos exemplos positivos cobertos e exemplos negativos excluídos”. No caso de haver um laço para cada heurística, o sistema prefere complexos com menos seletores. As sementes são escolhidas randomicamente e exemplos negativos são escolhidos de acordo com a sua distância da semente (os mais próximos são apanhados primeiro, onde a distância é o número de atributos com valores diferentes na semente e no exemplo negativo). No caso de contradições (isto é, a semente e o exemplo negativo terem valores de atributo idênticos) o exemplo negativo é ignorado e um outro é escolhido, uma vez que o complexo não pode ser especializado para excluí-lo, mas ainda pode incluir a semente.

Procedimento AQR(POS, NEG) retornando COBERTURA:

faça COBERTURA ser uma cobertura vazia;

enquanto cobertura não cubra todos os exemplos positivos em POS

 selecione uma semente, isto é um exemplo positivo não coberto pela
 cobertura;

 chame procedimento ESTRELA(SEMENTE,NEG) PARA GERAR A

 ESTRELA (um conjunto) de complexos que cobrem a semente mas nenhum
 exemplo em NEG;

 selecione o melhor complexo BEST a partir da estrela de acordo com o critério
 definido pelo usuário;

 adicione BEST como uma disjunção extra à cobertura;

retorne cobertura.

Procedimento ESTRELA(SEMENTE, NEG) retornando ESTRELA:

faça ESTRELA ser o conjunto contendo o complexo vazio;

enquanto um ou mais complexos em ESTRELA cobrem alguns exemplos negativos
em NEG,

 selecione um exemplo negativo E_{neg} coberto por um complexo na ESTRELA;

Especializar complexos em ESTRELA para excluir E_{neg} por:

 faça EXTENSÃO ser todos os seletores que cobrem a SEMENTE mas
 não E_{neg} ;

 faça ESTRELA ser o conjunto $\{x \wedge y \mid x \in ESTRELA, y \in$
 EXTENSÃO $\}$;

 Remova os piores complexos da ESTRELA

 até que o tamanho da ESTRELA seja menor ou igual ao máximo definido
 pelo usuário (maxiestrela).

Retorne ESTRELA.

Tabela 2. O algoritmo de cobertura AQR: Gerando uma cobertura para a classe C

III.4.3.3. Um Classificador Bayesiano

É implementado também um simples classificador bayesiano e o seu comportamento é comparado ao dos outros algoritmos.

a. Linguagens de Descrições de Conceito e suas Interpretações

Esse classificador representa sua 'regra de decisão' como uma matriz de probabilidades $p(v_j/C_k)$ especificando a probabilidade de ocorrência de cada valor de atributo, dada cada classe. Para classificar um novo exemplo, pode-se aplicar o teorema de Bayes

$$p(C_i | \wedge v_j) = \frac{p(\wedge v_j | C_i) p(C_i)}{\sum_k p(\wedge v_j | C_k) p(C_k)}$$

onde a conclusão está em torno das n classes e $p(C_i | v_j)$ denota a probabilidade de que o exemplo é da classe C_i dado v_j (\wedge é o símbolo para conjunção, $\wedge v_j$ denota a conjunção de valores de atributos todos ocorrendo em um exemplo). Pode-se calcular essa probabilidade para cada classe, e então seleciona-se a classe com a mais alta probabilidade. O termo $p(C_k)$ é estimado a partir da distribuição dos exemplos de treinamento pelas classes. Se for assumida a independência de atributos, $p(v_j \wedge C_k)$ pode ser calculado usando

$$p(\wedge v_j | C_k) = \prod_j p(v_j | C_k)$$

e os valores $p(v_j | C_k)$ a partir da matriz de probabilidade.

Deve-se notar que existem aplicações mais sofisticadas das regras de Bayes nas quais os testes de atributo são ordenados [29]. Essa técnica sequencial adiciona a contribuição de cada teste; quando essa pontuação excede um limite, o algoritmo produz um prognóstico de classes.

b. O Algoritmo de Aprendizado

O Método de Aprendizado Bayesiano constrói a matriz $p(v_j|C_k)$ a partir dos exemplos de treinamento, examinando a frequência dos valores de cada classe. Pode-se computar essa matriz incrementalmente, incorporando um exemplo por vez, ou não-incrementalmente, usando todos os dados desde o princípio.

c. A Heurística

Algumas vezes o valor de zero é calculado a partir dos dados de treinamento para alguns elementos da matriz $p(v_j|C_k)$. Como todos os elementos da matriz, este número está sujeito a erro devido ao conjunto de dados de treinamento disponível ser finito. Entretanto, como a classificação de novos exemplos envolve a multiplicação de elementos, um elemento zero pode ter um efeito drástico, anulando o efeito de todas as outras probabilidades na multiplicação. Para evitar isso, assume-se que zero elementos na matriz, fornecidos mais dados, convergiriam num valor pequeno, diferente de zero e assim substitui-se os zeros por alguma estimativa mais apropriada.

d. A Regra *Default*

Finalmente, um algoritmo que simplesmente atribui a classe que mais comumente ocorre a todos os novos exemplos, sem nenhuma referência a seus atributos foi usada para comparação com os outros algoritmos. Curiosamente, esse procedimento simples era de desempenho comparável a outros algoritmos em algumas aplicações, e assim provou ser um algoritmo extra, útil a ser considerado.

III.4.3.5. Algoritmo CN2

Será mostrado agora o algoritmo CN2, primeiro descrevendo como o algoritmo geral surge, naturalmente, a partir da consideração dos algoritmos ID3 e AQ e então descrevendo seus detalhes.

O ID3 é facilmente adaptável para manipular dados com ruído em virtude da sua abordagem de geração de árvores do tipo top-down. Durante a indução, todos os testes de atributo possíveis são considerados quando se aumenta um nó de folha da árvore, e a entropia é usada para selecionar o melhor valor de atributo para tomar o lugar naquele nó. A super-adequação nas árvores de decisão pode ser evitada, impedindo-se o crescimento da árvore quando mais nenhuma informação significativa pode ser obtida pela continuação do crescimento. Seria interessante poder aplicar um método similar para a produção de regras *if-then*.

O algoritmo AQ, quando gera um complexo, também executa uma procura que parte do geral para o específico para encontrar o melhor complexo. Entretanto, apenas especializações que excluem algum exemplo negativo particular coberto pelo complexo, enquanto se assegura que algum exemplo ‘semente’ positivo particular que permaneça coberto sejam consideradas, iterando até que todos os exemplos negativos sejam excluídos. Como resultado, o AQ executa a procura apenas no espaço dos complexos completamente consistentes com os dados de treinamento. O algoritmo AQ emprega uma procura “irradiada”, que pode ser vista como várias procuras *hill-climbing* em paralelo.

Para o algoritmo CN2, foi mantido o método de procura “irradiada” do algoritmo AQ mas foi removida primeiramente sua dependência a exemplos específicos durante a procura e secundariamente foi estendido seu espaço de procura para incluir regras que não tenham um desempenho perfeito nos dados de treinamento. Isso é conseguido pela simples ampliação do processo de especialização de forma a examinar todas as especializações de um complexo, similar à maneira com que o ID3 considera todos os testes de atributo quando aumenta o nó em uma árvore. Na realidade, com uma largura “irradiada” de um, o algoritmo CN2 se comporta equivalentemente ao ID3 quando aumenta um simples ramo da árvore. Assim, pela execução de uma procura *top-down* para os complexos, é possível agora aplicar um método de poda similar à poda de árvore de decisão para deter a especialização quando não é mais possível encontrar especializações estatisticamente significantes.

Finalmente, nota-se que as regras que o CN2 produz tem mais a disposição de uma lista ordenada de regras if-then, que um conjunto desordenado de regras como as produzidas pelos sistemas baseados no método AQ. Ambas as representações têm suas respectivas vantagens e desvantagens para a compreensibilidade – regras independentes da ordem requerem algum mecanismo adicional a ser provido para resolver quaisquer conflitos de regras que possam ocorrer (assim contornando uma interpretação lógica muito rigorosa das regras), enquanto as regras ordenadas também sacrificam um pouco da compreensibilidade uma vez que a interpretação de uma simples regra é dependente de quais outras regras a precederam na lista. É possível fazer o CN2 produzir regras if-then desordenadas modificando apropriadamente a função de avaliação.

a. A Linguagem de Descrição de Conceitos e Sua Interpretação

As regras da lista ordenada que o CN2 induz são cada uma da forma ‘if <complexo> then prognostica <classe>, onde <complexo> tem a mesma definição que a do AQR; nominalmente um conjunto de testes de atributo. Essa representação das regras ordenadas é uma versão do que Rivest [26] chamou de regras de decisão. A última regra é uma regra *default* que simplesmente prognostica a classe que mais comumente ocorreu nos dados de treinamento para todos os exemplos novos.

Para usar as regras induzidas para classificar novos exemplos, o CN2 usa uma representação na qual cada regra é testada em ordem até que seja encontrada uma cujas condições sejam satisfeitas pelo exemplo sendo classificado. O prognóstico da classe resultante da regra é então determinado como a classe daquele exemplo. Assim, a ordenação das regras é importante. Se nenhuma das regras é satisfeita, a regra *default* final determina a classe mais comum para o exemplo novo.

b. Algoritmo de Aprendizado

A tabela 3 apresenta um resumo do algoritmo CN2. Ele funciona de uma forma iterativa, cada iteração procurando por um complexo cobrindo um grande número de exemplos de uma simples classe *C* e uns poucos exemplos de outras classes . O complexo deve ser ao mesmo tempo prognosticador e fidedigno, como determinado

pelas funções de avaliação do CN2. Tendo encontrado um bom complexo, aqueles exemplos que ele cobre são removidos do conjunto de treinamento e a regra 'if <complexo> então prognostica C' é adicionada ao final da lista de regras. Esse processo itera até que não existam mais complexos satisfatórios que possam ser encontrados.

O sistema procura por complexos executando uma procura para a realização da poda partindo do geral para o específico. A cada estágio da procura, o CN2 retém um conjunto limitado em tamanho ou estrela S dos 'melhores complexos encontrados até então'. O sistema examina apenas especializações desse conjunto, executando uma procura irradiada no espaço dos complexos. Um complexo é especializado ou pela adição de um novo termo conjuntivo ou pela remoção de um elemento disjuntivo em um de seus seletores. Cada complexo pode ser especializado de muitas formas, e o CN2 gera e avalia todas essas especializações. A estrela é aparada depois da conclusão desse passo pela remoção de seus elementos com pior pontuação, medidos por uma função de avaliação que será brevemente descrita.

Nessa implementação, o passo de especialização consiste em "cruzar" repetidamente o conjunto de todos os seletores possíveis com a estrela corrente, eliminando todos os elementos vazios e não modificados no conjunto resultante de complexos. (Um complexo vazio é aquele que contém um par de seletores incompatíveis, e.g. $\text{grande}=\text{y} \wedge \text{grande}=\text{n}$). O CN2 lida com atributos contínuos de uma maneira similar ao ASSISTANT – pela divisão da faixa de valores de cada atributo em sub-faixas discretas. Os testes em cada atributo examinam se o valor é maior ou menor (ou igual) que os valores nos limites das sub-faixas. A faixa de valores completa e o tamanho de cada sub-faixa é fornecido pelo usuário.

Para lidar com valores de atributo desconhecidos, o CN2 usa um método simples de substituir valores desconhecidos pelos valores que ocorrem mais comumente (ou o valor intermediário da sub-faixa que ocorre mais comumente, no caso dos atributos numéricos) para aquele atributo nos dados de treinamento.

c. Heurística

O algoritmo CN2 deve tomar duas decisões heurísticas durante o processo de aprendizado, e ele emprega duas funções de avaliação para ajudar nessas decisões.

Primeiro ele deve avaliar a qualidade dos complexos, determinando se um complexo deve substituir o ‘melhor complexo’ encontrado até então e também quais complexos na estrela S devem ser descartados se o tamanho máximo é excedido. Essa computação envolve primeiro encontrar o conjunto E' de exemplos que um complexo cobre (i.e., que satisfaz todos os seus seletores) e a distribuição de probabilidade $P = (p_1, \dots, p_n)$ de exemplos em E' entre classes (onde n = número de classes representada nos dados de treinamento). O CN2 usa a medida de entropia teórica da informação

$$Entropia = -\sum_i p_i \log_2(p_i)$$

para avaliar a qualidade do complexo (quanto mais baixa a entropia melhor o complexo). Essa função prefere dessa forma, complexos cobrindo um grande número de exemplos de uma simples classe e uns poucos exemplos de outras classes. Por isso, tais complexos são bem quotados no arquivo de dados quando usados para prognosticar a classe majoritária coberta. Deve ser notado que essa função foi usada preferencialmente a uma simples medida correta de percentual (e.g. tomando o $\max(P)$), principalmente porque a entropia vai distinguir as distribuições de probabilidade tais como $P = (0.7, 0.1, 0.1, 0.1)$ e $P = (0.7, 0.3, 0, 0)$ em favor do último enquanto que o $\max(P)$ não vai. Isso é desejável, uma vez que existem mais maneiras de especializar o último para um complexo identificando apenas uma classe; se os exemplos da classe majoritária são excluídos pela especialização, as distribuições se tornam $P = (0, 0.33, 0.33, 0.33)$ e $P = (0, 1, 0, 0)$ respectivamente. Em adição, a medida de entropia tende a dirigir a procura na direção das regras mais significantes; empiricamente, regras de entropia alta também tendem a ter uma significância alta.

A segunda função de avaliação testa se um complexo é *significativo*. Por isso, faz-se uma referência a um complexo que localize uma regularidade improvável de acontecer por acaso, e assim reflita uma correlação genuína entre os valores de atributos e classes. Para determinar a significância, o CN2, compara a distribuição *observada* entre classes de exemplos que satisfazem ao complexo com a distribuição *esperada* que seria resultante se o complexo selecionasse exemplos randomicamente. Algumas diferenças dessas distribuições vão resultar da variação randômica.

Faça: E ser um conjunto de exemplos de treinamento;

Procedimento CN2(E) retornando LISTA_REGRAS:

faça LISTA_REGRAS ser lista vazia;

repita

faça MELHOR_CPX ser ENCONTRE_MELHOR_COMPLEXO(E);

se MELHOR_CPX não é zero então

Faça E' ser os exemplos cobertos por MELHOR_CPX;

Remova de E os exemplos E' cobertos por MELHOR_CPX;

Faça C ser a classe mais comum de exemplos em E';

Adicione a regra 'se MELHOR_CPX então classe=C' ao fim da LISTA_REGRAS

até que MELHOR_CPX seja zero ou E esteja vazio.

retorne LISTA_REGRAS.

Procedimento ENCONTRE_MELHOR_COMPLEXO(E) retornando MELHOR_CPX:

faça o conjunto ESTRELA conter apenas o complexo vazio;

faça MELHOR_CPX ser vazio;

faça SELETORES serem o conjunto de todos os possíveis seletores;

enquanto ESTRELA não está vazia,

especialize todos os complexos em ESTRELA como a seguir:

faça NOVA_ESTRELA ser o conjunto $\{x \wedge y \mid x \in \text{ESTRELA}, y \in \text{SELETORES}\}$;

Remova todos os complexos em NOVA_ESTRELA que estão também em ESTRELA (isto é, os não-especializados) ou são nulos (por exemplo, grande = y \wedge grande = n)

para cada complexo Ci na NOVA_ESTRELA:

se Ci é estatisticamente significativo quando testado em E,

então substitua o valor corrente do MELHOR_CPX por Ci;

repita remova piores complexos a partir de NOVA_ESTRELA

até que tamanho da NOVA_ESTRELA é \leq o máximo definido pelo usuário;

faça ESTRELA ser NOVA_ESTRELA;

retorne MELHOR_CPX.

Tabela 3. O algoritmo CN2

A questão é se as diferenças observadas são grandes demais para serem explicadas satisfatoriamente pelo acaso. Se for assim, o CN2 assume que o complexo reflete uma correlação genuína entre atributos e classes.

Para testar a significância, o sistema usa a taxa de probabilidade estatística [30]. Isso é dado por

$$2 \sum_{i=1}^n f_i \log(f_i / e_i),$$

onde a distribuição $F = (f_1, \dots, f_n)$ é a distribuição de frequência observada de exemplos entre classes que satisfazem um complexo dado e $E = (e_1, \dots, e_n)$ é a distribuição de frequência esperada do mesmo número de exemplos sob a suposição de que o complexo seleciona exemplos randomicamente. Isso é tomado como os $N = \sum f_i$ exemplos cobertos distribuídos entre classes com a mesma probabilidade que aquela dos exemplos do conjunto de treinamento inteiro. Essa estatística fornece uma medida teórica de informação da distância (não-comutativa) entre as duas distribuições. Sob suposições adequadas, pode-se mostrar que essa estatística é distribuída aproximadamente como χ^2 com $n-1$ graus de liberdade. Isso fornece uma medida que indica a significância – quanto menor a pontuação, mais provável que a aparente regularidade seja devido ao acaso.

Assim essas duas funções – entropia e significância – servem para determinar se complexos encontrados durante a procura são ao mesmo tempo ‘bons’ (isso é, exatidão alta quando se faz o prognóstico da classe majoritária coberta) e ‘confiáveis’ (alta exatidão nos dados de treinamento não é apenas devido ao acaso). O CN2 usa essas funções para a procura repetida dos melhores complexos que também superem algum limite mínimo de confiabilidade até que não se possa mais encontrar complexos confiáveis .

III.4.3.6. Algoritmo de Procura

O algoritmo CN2 gera uma lista de decisão de uma forma iterativa, construindo uma regra a cada iteração. Uma regra é construída pela procura por um complexo que cubra um grande número de exemplos de uma classe arbitrária C_i e poucas de outras classes (a construção de tal complexo é descrita abaixo). Tendo achado um bom complexo, o algoritmo remove aqueles exemplos que ele cobre do conjunto de treinamento e adiciona a regra 'if $\langle \text{complex} \rangle$ then faça o prognóstico C_i ' para o fim da lista de decisão. Para o conjunto remanescente, uma nova regra é construída, até que não sejam mais encontrados complexos de qualidade suficiente.

O melhor complexo num conjunto de treinamento é encontrado usando uma procura de apoio, isto é, a construção de um gráfico de procura, onde a cada iteração, o gráfico é estendido a quase todas as folhas da maxi-estrela. A largura da maxi-estrela de apoio é definida pelo usuário. A qualquer momento durante a procura, as folhas (isto é complexos) são armazenados em um conjunto de tamanho limitado, chamado *estrela*. O sistema também mantém registro do melhor complexo encontrado até então.

Inicialmente, a estrela contém apenas a raiz do gráfico de procura, que é um complexo vazio, cobrindo o conjunto de treinamento inteiro. Todas as especializações de todos os complexos na estrela são examinadas pela computação da interseção da estrela e o conjunto de todos os possíveis seletores. Complexos cuja significância não exceda um limite estabelecido são descartados. Dos complexos remanescentes, os mais interessantes – como definido pelo critério de qualidade – formam a nova estrela.

O processo de procura do melhor complexo termina quando não se possa encontrar complexos que passem pelo teste de significância. O melhor complexo, encontrado no gráfico de procura é retornado, e forma a parte condicional de uma nova regra. Se nenhum complexo é encontrado, uma nova regra não pode ser construída e a regra default é adicionada ao final da lista de decisões.

III.4.3.7. Conclusão

O CN2 constrói regras de produção simples e compreensíveis em domínios onde o ruído pode estar presente. Ele constrói regras de probabilidade, isto é, as partes condicionais cobrem exemplos de uma simples classe, mas possivelmente uns poucos exemplos de outras classes também. O resultado não é dependente da ordem na qual exemplos particulares são escolhidos, como no sistema AQ, e ele procura uma área maior do espaço de procura, uma vez que ele considera todas as possíveis extensões de uma estrela. Mas a principal vantagem do CN2 sobre o AQ15 é que uma vez que o primeiro suporta um mecanismo de corte, ele não restringe a sua procura para apenas aquelas regras que são consistentes com o conjunto de exemplos.

O desempenho do ID3, AQ, e CN2 foram comparados em domínios médicos e aplicações em inteligência artificial. As estruturas de conhecimento descobertas são equivalentes em termos de exatidão e complexidade.

III.5. O Pacote de Algoritmos Tooldiag

O Tooldiag é uma ferramenta desenvolvida por Rauber [31] para manusear grandes quantidades de dados. Sua tarefa básica é conseguir extrair desses dados apenas a quantidade mínima suficiente para representá-los.

O pesquisador enfrenta frequentemente o problema de não poder usar os dados em sua totalidade: dados incompletos não podem ser manuseados facilmente – o próprio Tooldiag não trabalha com dados nessa situação – e existe também o caso dos dados com ruído que podem dificultar a utilização e avaliação dos mesmos. Além disso, existem os problemas devidos a parâmetros desconhecidos na modelagem do problema.

Os dados podem ser de naturezas bastante diferentes, como dados visuais ou padrões de som, medidas de temperatura, pressão, etc. A ferramenta de análise deve, portanto, ser flexível em relação à variedade dos dados e fazer o menor número de suposições possível sobre o comportamento específico desses dados.

Apresenta-se a seguir um conjunto de ferramentas enfocando principalmente as seguintes tarefas:

- Seleção de Características:

A dimensão de um vetor de amostras que caracterize uma classe é reduzido a um tamanho razoável.

Vários algoritmos são propostos.

- Estimativa de Erros:

Com o conjunto de características reduzido, o desempenho de um classificador baseado nessas características é prognosticado. Isso significa que se pode fazer previsões do quanto uma classificação de situações de um conjunto de dados desconhecido pode ser bem realizada no futuro.

- Visualização 2D:

A distribuição das classes no espaço das características pode ser visualizada no plano x-y com a ajuda de uma técnica de mapeamento não-linear. O comportamento do erro estimado em relação ao número de características selecionadas também pode ser plotado. Além disso, a correlação linear entre duas características diferentes pode ser testada e visualizada.

A filosofia principal dessa ferramenta de análise é dar ao pesquisador uma idéia sobre os seus dados: Vale a pena caracterizar uma determinada situação? Qual o subconjunto particular de dados é melhor que o outro? Qual a proximidade de certas características em relação à outras? Etc.

III.5.1. Introdução

O Tooldiag trabalha apenas com registros completos. Por isso, se os dados em estado natural se apresentam incompletos – o que é comum – antes de aplicar os algoritmos do Tooldiag é preciso um pré-processamento.

Os dados podem, também, apresentar ruído. Nesse caso é necessário saber o quanto esses dados estão alterados pelo ruído. Se os dados apresentam pouco ruído, o próprio processo de seleção de características vai se incumbir de expurgá-los naturalmente já que seriam considerados como exceções dentro dos dados, ou seja, dados fora do padrão. Já no caso desses dados apresentarem muito ruído, seria muito

mais difícil o manuseio, porque uma característica poderia ser tão distorcida pelo ruído que o próprio ruído poderia se tornar o padrão ou simplesmente não se chegar a um padrão.

Depois que os dados estão prontos para serem manuseados pelo Tooldiag começa então o processo de descoberta de conhecimento. Sejam quais forem os tipos de dados com os quais se está trabalhando, algumas peculiaridades podem ser associadas a eles. Cada registro pode ser visto como um vetor contendo valores de atributos relacionados a uma determinada situação ou classe. Sendo assim, pode-se ver cada linha do conjunto de dados como um vetor multidimensional de características.

Uma tarefa essencial no caso de grandes quantidades de dados a serem analisados é a capacidade de extrair o conhecimento estrutural.

O próximo passo é executar uma redução dos dados, que é feita utilizando-se técnicas de seleção ou de extração de características. O processo de seleção/extração de características se resume à redução do tamanho dos vetores de características a um tamanho suficiente para descrever bem o modelo. Para isso, é necessário identificar que partes dos dados melhor descrevem uma classe. Isso leva ao problema de reconhecimento de padrão e à realização de um aprendizado supervisionado [32]. O algoritmo de seleção de características deve ser de finalidade geral.

Pode-se ainda desejar saber se o conjunto das características escolhidas vai desempenhar bem a classificação. Essa é a tarefa do módulo de estimativa de erro.

Uma outra parte importante desse trabalho é a visualização das amostras compostas de características que foram tomadas durante o aprendizado e classificação. Uma vez que esses vetores de características são normalmente de uma dimensão que mesmo em uma forma seleta são maiores que a capacidade da imaginação humana, é preciso fazer um mapeamento para duas dimensões. Isso permite que se tenha uma idéia se certas características podem distinguir bem as classes.

O algoritmo de Sammon [33] mapeia um conjunto de distribuições de classes multidimensionais para o plano x-y.

III.5.2. Aquisição dos Dados:

O primeiro passo para a análise dos dados é a definição e a aquisição de amostras que descrevam perfeitamente a situação a ser avaliada. O processo de aquisição dos dados gera um vetor de características multidimensionais e o classifica com a respectiva classe (normal ou defeito).

A interface com o módulo de análise pode ser on-line ou via arquivos. Os arquivos têm a vantagem de poder manter a informação de entrada para experimentos posteriores com outras estratégias.

III.5.3. Seleção de Características:

É necessário executar uma redução de dados horizontal, isto é, de todas as características apenas um número reduzido vai ser retido; aquelas características que possuem o mais alto poder discriminativo serão usadas mais tarde para a classificação. Uma seleção de características deve ser capaz de encontrar as características que contêm as informações mais significativas dos dados originais. A formulação geral do problema de seleção de característica é portanto:

Para o vetor original de dados $\underline{x} = (x_1, \dots, x_D)$ ache um $\underline{x}' = (x_1, \dots, x_d)$ com uma dimensão reduzida $d < D$ que represente as classes mais compactamente.

Um dos problemas fundamentais da seleção de características é que a solução ótima não pode ser encontrada por procura exaustiva. Para uma dimensão ainda relativamente pequena, o número de combinações possíveis tornam impossível uma análise [34]. Além disso, na maior parte dos casos, é necessária a utilização da aproximação heurística. A distribuição estatística das características também não é conhecida a priori. Isso significa que, se modelos paramétricos são usados, mais suposições devem ser feitas.

III.5.4. Estimativa de Erro:

A estimativa de erro *Hold-out* divide um simples conjunto de dados em um conjunto de treinamento e um conjunto de teste. Um classificador é induzido a partir

do conjunto de treinamento e o conjunto de teste é então alimentado ao classificador. A taxa de erro do teste é reportada. Essa divisão em conjuntos de treinamento e teste, a geração do classificador e o teste podem ser repetidos várias vezes. Usualmente o erro estimado geral é a média dessas duas partes.

A estimativa de erro *leave-one-out* é dada quando se repete essa divisão e testagem tantas vezes quanto o número de amostras. Foi pega uma simples amostra, utilizado um classificador no resto das amostras e testado se o classificador identifica corretamente ou não. Esse processo é repetido para todas as amostras e o número absoluto de classificações erradas é dividido pelo número de amostras. O esquema seguinte lista a técnica de estimativa de erro:

Seja n o número de amostras do conjunto de dados S .

Seja e o número de erros.

- 1.) $e \leftarrow 0$
- 2.) Para todas as amostras i do conjunto de dados $i=1...n$
 - 1.1) Gere o classificador com todas as j amostras do conjunto de dados $j=1...n, j \neq i$
 - 1.2) Classifique a amostra i
 - 1.3) Se i foi classificado errado incremente e : $e \leftarrow e+1$
- 3.) Taxa de erro estimada $\leftarrow e/n$

Tabela 4. O algoritmo para a estimativa de erro *Leave-one-out*

A estimativa de erro *leave-one-out* tende sutilmente ao pessimismo. O custo computacional é relativamente alto, isso depende do quadrado do número de amostras vezes o custo da geração do classificador e o do passo de classificação. Por outro lado, isso dá um prognóstico confiável do desempenho de um classificador de todos os métodos *hold-out*.

III.5.5. Geração do Classificador

O classificador pode ser escolhido entre várias arquiteturas de classificação. Alguns deles necessitam de alguma especificação especial como por exemplo o k do k-vizinho mais próximo. A escolha do classificador, porém, não exerce qualquer influência na seleção ou extração de características, sendo necessária apenas nos módulos de “estimativa de erro” e “classificação de dados independentes”. Dá-se a seguir uma idéia sucinta sobre os modelos de classificador disponíveis no pacote.

III.5.5.1. Classificador do Vizinho mais Próximo:

Essa é a técnica de classificador não-paramétrico mais simples e é usada para a finalidade de estimativa de erro. Uma amostra desconhecida é considerada como pertencente a uma certa classe c se os seus k vizinhos mais próximos num espaço Euclidiano multidimensional pertencem a mesma classe (K-Vizinho Mais Próximo). "Mais Perto" indica que uma medida de distância está disponível, o que dá um valor numérico contínuo para duas amostras e satisfaz às propriedades de distância. Normalmente a norma Euclidiana é escolhida como a medida de distância, mas dependendo da situação, qualquer outra métrica pode ser aplicável. Assim, para duas amostras x_i e x_j a distância euclidiana é definida como

$$d(x_i, x_j) = \|x_i - x_j\|$$

Não se executa mais nenhuma redução sobre o conjunto de treinamento. Todas as amostras são consideradas protótipos [35]. Isso mantém a estimativa de erro livre de influências negativas do passo de compressão do conjunto de treinamento, mas gera mais cálculo, porque uma amostra desconhecida deve ser comparada com todas as amostras de treinamento.

III.5.5.2. Quadratic Gaussian Classifier

Uma forma comum de modelar probabilidades associadas às classes no espaço n -dimensional é assumir distribuições Gaussianas normais definidas na fórmula:

$$N(v, \mu, K) = \frac{1}{\sqrt{(2\pi)^N \det K}} \exp[-1/2(v - \mu)^T K^{-1}(v - \mu)]$$

A notação $N(v, \mu, K)$ com três argumentos v , μ e K expressa o fato que por um lado $N(v, \mu, K)$ é a função de densidade de probabilidade para alguma variável randômica v e por isso deve ter v como seu argumento e, por outro lado a forma específica da função de densidade $N(v, \mu, K)$ é determinada por dois vetores de parâmetros do meio e a matriz de covariância K . Esses dois parâmetros permitem posicionar o centro de $N(v, \mu, K)$ – o ponto de densidade probabilística máxima – em qualquer lugar no espaço n -dimensional e controlar a largura junto com a forma da função de densidade. Para essa discussão, é suficiente descrever as K probabilidades específicas de classe diferentes pelas K funções de distribuição normais cada uma com seu próprio conjunto de parâmetros μ_k, K_k :

$$prob(v|k) = N(v, \mu_k, K_k)$$

Se uma distribuição simples do tipo descrito acima não é flexível o suficiente para se adequar à distribuição empírica da classe k correspondente ao grau de precisão desejado, então uma combinação linear de um certo número, normalmente pequeno, de funções de distribuições normais podem ser empregados para “renderizar” uma adequação melhor para uma distribuição mista.

Enquanto em um caso de modelagem de $prob(v|K)$ por uma distribuição normal simples, a estimativa dos parâmetros μ e K é uma tarefa direta e é realizada pelo cálculo do meio da amostra e covariância da amostra como simples médias aritméticas a partir do conjunto de treinamento das classes, a tarefa de adequação das distribuições mistas é muito mais complicada. Isso leva a loops de otimização aninhados, mas podem ser acelerados por decisões de projetos acertadas sobre o

número de distriuições de componente ou sobre sua respectiva posição μ_i e matrizes de covariância K_i .

III.5.5.3. Radial Basis Function Network

A rede de função de base radial consiste de unidades executando funções lineares ou não-lineares de atributos, seguidos de uma camada de conexões a nós ponderadas cujas saídas têm a mesma forma que os vetores de referência. Essa rede apresenta algumas vantagens como um treinamento linear de regras, uma vez que os lugares no espaço dos atributos das funções não-lineares foi determinado, e um modelo subentendido envolvendo funções localizadas no espaço dos atributos. Um aprendizado de regras linear evita problemas associados com o mínimo local; em particular, ele fornece uma grande habilidade para fazer declarações sobre a acuidade da interpretação probabilística das saídas.

A figura III.1 mostra a estrutura de uma função de base radial; as não-linearidades compreendem uma posição no espaço dos atributos no qual a função está localizada (frequentemente referido como “centro” da função) e uma função não-linear da distância de um ponto de entrada a partir do centro, que pode ser qualquer função. Escolhas comuns incluem a função gaussiana de resposta, $\exp(-x^2)$ e multiquadráticas inversas ($[z^2+c^2]^{-1/2}$), etc. Embora pareça intuitivo tentar conseguir uma função de interpolação usando funções não-localizadas, é mais frequente ter propriedades de interpolação na região onde estão os dados de treinamento.

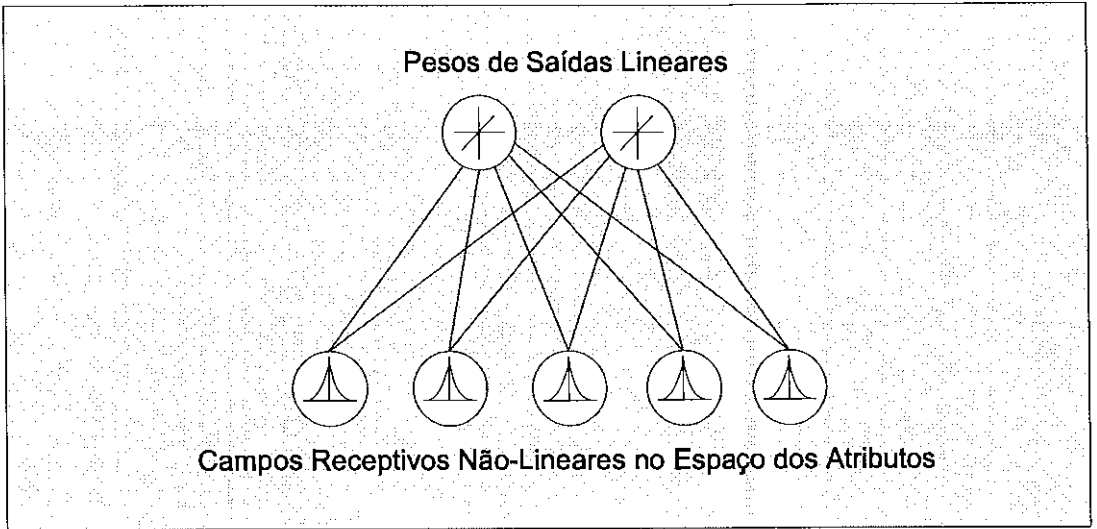


Figura III.1. Uma Rede de Função de Base Radial

A abordagem da função de rede de base radial envolve a expansão ou pré-processamento de vetores de entrada em um espaço dimensional grande. Ela tenta explorar o teorema de Cover [37] que diz que uma modelagem de problema de classificação em um espaço dimensional grande é mais provável de ser linearmente separável do que seria no caso de um espaço dimensional pequeno.

a. Escolhendo os Centros e as Não-Linearidades

Vários métodos podem ser utilizados para escolher os centros para uma rede de função de base radial. É importante que as distribuições de centros no espaço dos atributos possa ser similar, ou ao menos cubra a mesma região que os dados de treinamento. Assume-se que os dados de treinamento são representativos do problema, de outra forma, não se pode esperar um bom desempenho sobre amostras novas.

Uma técnica de primeira ordem para escolher centros é pegar pontos num *grid* quadrado abrangendo a região do espaço dos atributos coberta pelos dados de treinamento. Alternativamente, um desempenho melhor pode ser esperado se os centros forem amostrados randomicamente a partir dos dados de treinamento, usando algumas ou todas as amostras, desde que as regiões mais densamente povoadas do espaço dos atributos tenham um modelo de resolução melhor que as regiões esparsas. Nesse caso, é importante assegurar-se que ao menos uma amostra de cada classe é usada como o centro do protótipo. O número de amostras requeridas a partir de cada

classe foi calculado antes da amostragem, assegurando-se assim que a condição foi encontrada.

Quando as posições de centro são escolhidas para a rede de função de base radial com funções não-lineares localizadas tais como os campos receptivos de Gaussian, é importante calcular variâncias adequadas. Isso assegura que regiões mais amplas do espaço não ocorram entre centros, onde nenhum centro responde a padrões, e nenhum par de centros respondem da mesma forma a todos os padrões. Esse problema é particularmente verdadeiro em espaços de atributos dimensionalmente grandes, porque o volume depende sensivelmente do raio. Para uma discussão quantitativa desse ponto, veja Prager & Fallside [38]. Os desvios padrões das funções gaussianas foram tomados separadamente para cada direção de coordenada para a distância ao centro mais próximo naquela direção, multiplicado por um parâmetro de escala arbitrário (definido inicialmente como 1).

Outros métodos incluem a técnica de agrupamento baseada em “princípios” para posicionar os centros, tais como Modelo Misto de Gaussian ou rede de Kohonen.

b. Otimizando os Pesos

As redes de função da base radial são treinadas simplesmente pela solução de um sistema linear. O mesmo problema aparece na regressão linear ordinária, sendo que a única diferença é que a entrada para o sistema linear é a saída da camada escondida da rede, e não os próprios atributos.

Faça $y_{ki}^{(H)}$ a saída da k^{th} função de base radial do exemplo i^{th} . A saída do nó de referência j é computada usando-se os pesos w_{jk} como

$$y_{ik} = \sum_k w_{jk} y_{ki}^{(H)}$$

Faça a saída desejada para o exemplo i no nó de referência j ser Y_{ji} . A medida de erro escrita em sua forma completa é

$$E(w) = \frac{1}{2} \sum_{ji} \left(\sum_k w_{jk} y_{ki}^{(H)} - Y_{ji} \right)^2$$

com o seu mínimo onde a derivada

$$\frac{dE}{dW_{rs}} = \sum_k \sum_i w_{rk} y_{ki}^{(H)} y_{ji}^{(H)} - \sum_i Y_{ri} y_{si}^{(H)}$$

tende a zero. Faça R ser a matriz de correlação das saídas da função de base radial,

$$R_{jk} = \sum_i y_{ki}^{(H)} y_{ji}^{(H)}$$

A matriz de peso w^* que minimiza E se encontra onde o gradiente tende a zero:

$$w_{jk}^* = \sum_r \sum_i Y_{ji} y_{ri}^{(H)} (R^{-1})_{rk}$$

Assim, o problema é resolvido pela inversão da matriz R quadrada $H \times H$, onde H é o número da função de base radial.

A inversão da matriz pode ser conseguida por métodos padrões como a decomposição LU (Renals & Rohwer [39]) se R não é singular, nem está próximo de ser. Esse é o caso típico, mas nem sempre é assim. Se dois centros de função de base radial estão muito próximos um do outro, isso resultará em uma matriz singular, e uma matriz singular é garantida se o número de amostras de treinamento não é ao menos igual a H . Não existe nenhuma forma prática de assegurar uma matriz de correlação não-linear. Conseqüentemente, o mais certo seria usar um método de decomposição computacionalmente mais oneroso. Tais métodos fornecem uma inversão aproximada pela diagonalização da matriz, inversão apenas dos eigenvalues

que excedem zero por uma margem de parâmetro específico, retorno às coordenadas originais.

III.5.5.4. Parzen Window

É um método clássico que surgiu a partir do reconhecimento de padrão estatístico. A técnica *Parzen Window* é uma estimativa não-paramétrica das funções de densidade de probabilidade.

Uma função kernel está localizada no lugar de cada um ponto dos dados do conjunto de treinamento. Para cada classe a soma ponderada de todos os kernels pertencentes a essa classe, dividido pela soma de todas as funções de kernels para todas as classes, dá a probabilidade estimada dependente a posteriori da classe (condicionada à medida). A comparação direta das probabilidades estimadas a posteriori para cada classe determina a decisão de classificação.

Vários tipos de kernels são conhecidos. Os três mais importantes são implementados:

- a) kernel hipercúbico
- b) kernel hipertriangular
- c) kernel hiperesférico
- d) kernel Gaussiano
- e) kernel exponencial
- f) kernel de Lorenz

Se o kernel hiperesférico é usado, o classificador converge para o método do k-vizinho mais próximo. Dentro de um certo subespaço do espaço de características, todos os pontos de dados contribuem para o valor da função de densidade estimada no centro desse sub- espaço (*Parzen window*). No caso do kernel hiperesférico as bordas do subespaço são dadas pela distância ao centro do sub-espaço. Esse é o mesmo princípio do método do k-vizinho mais próximo. Entretanto, no último caso, a extensão do sub-espaço depende das distâncias das k amostras mais próximas, isto é, uma região inicial é aumentada até que k amostras tenham sido encontradas. Na

técnica Parzen, as bordas são estáticas; a largura do subespaço, a largura da janela, é pré-determinada.

O kernel hipercúbico é baseado na distância de Chebcheff, enquanto que o kernel Gaussiano é definido sobre o espaço de características inteiro. Nesse caso, o parâmetro “largura da janela” determina a extensão e a amplitude das funções de kernel.

Similar ao kernel Gaussiano, o kernel exponencial e o Lorenz – onde a influência de uma amostra cai exponencialmente com funções crescentes (distância Euclidiana), são definidas por todo o espaço de características. A forma do kernel de Lorenz é similar à função Gaussiana e definida como

$$Kernel(dist) = (P_i/window_width) * 1 / (1 + \text{sqr}(dist/window_width))$$

Como uma forma simplificada das funções de kernel com contribuições decrescentes para distâncias crescentes, tem sido implementado o kernel “triangular”, dado por

$$Kernel(dist) = const * (window_width - dist)$$

A exatidão do classificador depende de uma escolha apropriada do parâmetro *window width* (largura da janela) que pode ser diferente ou igual para todas as classes. Ele tem que ser escolhido em percentagem, referindo-se à distância mais ampla entre dois vetores numa classe particular. Se o parâmetro é definido como sendo muito pequeno, a estimativa da função de densidade resultante se torna aguda; se o parâmetro é definido como um valor muito alto, as estimativas refletem uma grande quantidade de atenuação. A largura da janela portanto representa a quantidade de atenuação que é necessária para calcular a finidade dos dados; motivo pelo qual ele é frequentemente chamado de “parâmetro de atenuação”.

No programa, o parâmetro deve ser definido como um percentual pequeno da extensão do espaço de características. O valor ótimo tem que ser encontrado em uma base heurística.

III.5.5.5. Q* Algorithm

É um algoritmo simples desenvolvido para o agrupamento de classes. Cada agrupamento é representado por um protótipo. Não é preciso fazer nenhuma suposição sobre a distribuição de probabilidade dos dados. O algoritmo classifica os dados, usando o conjunto corrente de protótipos. O classificador de 1-Vizinho Mais Próximo é utilizado quando se deseja uma classificação que atribua a classe do protótipo mais próximo à amostra classificada. Se a classificação não é satisfatória, novos protótipos são dinamicamente criados. As amostras que são classificadas positivamente influenciam o protótipo mais próximo por deslocá-lo para o meio (média) das amostras. Quando não ocorrem mais criações de protótipos ou deslocamentos de protótipos, a auto-organização dos protótipos pára. Nesse ponto, o Q* algorithm já aprendeu então a classificar corretamente todas as amostras de treinamento existentes pelos protótipos criados dinamicamente. Ele formou um conjunto de protótipos representativos que comprime os dados consideravelmente. A totalidade das amostras é comprimida em um pequeno conjunto de protótipos representativos. O esforço para obter um conjunto ótimo de protótipos é proporcional à complexidade da distribuição estatística dos dados. Por exemplos, a classe “setosa” dos dados de íris precisam apenas de um protótipo, porque ela é bem separada das outras duas classes. As outras duas classes “virginica” e “versicolor” precisam de mais protótipos porque se superpõem.

O algoritmo original atualiza os novos protótipos como o meio de todas as amostras que foram classificadas corretamente.

Agora também a “média marginal” e a “média vetorial” podem ser escolhidas como o protótipo atualizado:

média marginal - O vetor da média unidimensional de cada característica.

média vetorial - A média vetorial das n amostras multi-dimensionais é a amostra com a mínima soma das distâncias euclidianas para outras amostras.

III.5.6. Visualização dos Dados Multidimensionais

Dados multidimensionais não podem ser visualizados se a dimensão é maior que 2 ou 3 no máximo. Um ponto, num espaço de características de 17 dimensões, que está muito longe de um outro ponto no mesmo espaço poderia ser idêntico a um subespaço, isto é, um subespaço de duas dimensões de suas duas primeiras componentes.

O algoritmo de Sammon [33] calcula a distância multidimensional mútua entre as amostras e arranja as amostras iterativamente no plano x-y. Isso permite ter uma idéia de como as amostras podem estar afastadas entre si. Pode-se esperar que classes bem distintas num espaço de muitas dimensões mantenham a sua qualidade também num espaço com poucas dimensões.

O objetivo principal desse módulo é fornecer informações sobre a natureza quantitativa e qualitativa dos mesmos. Ele mostra um mapeamento dos dados n-dimensionais para duas ou três dimensões.

Como um pequeno módulo adicional, a correlação linear entre duas características diferentes pode ser calculada e visualizada. Isso pode ser feito para uma ou mais classes.

III.5.7. Implementação

Todos os conceitos apresentados até aqui foram programados em linguagem C bastante portátil. O conjunto de ferramentas foi compilado em uma modalidade de plataforma UNIX e em sistema DOS.

A aquisição de dados é feita através de arquivos em formato pré-determinado. A seleção de características e a estimativa de erro é implementada numa parte que foi desenvolvida mais recentemente. A ferramenta fornece uma interface para o programa. Incluído no pacote está um programa utilitário que executa o mapeamento de Sammon. Nesse trabalho há uma interface desse módulo para um pacote bastante conhecido, fácil de usar e potente que é o GNUPLOT. Assim, é possível acumular dados, selecionar características, visualizar dados, e gerar um classificador on-line.

III.5.8. Conclusões

O pacote foi concebido para a caracterização de classes. O Tooldiag vê os registros como vetores multidimensionais contínuos de características. O objetivo principal é a seleção do subconjunto mais discriminativo das características que são fornecidas. Pode-se calcular a estimativa de erro de um classificador baseado nessas características. Pode-se ainda, visualizar dados multidimensionais fazendo um mapeamento em apenas duas dimensões.

O pacote tenta fazer o menor número de suposições sobre os dados disponíveis quanto possível. É fornecida uma interface para um módulo de aprendizado de classes.

III.6. Sistemas Fuzzy

III.6.1. Introdução:

III.6.1.1. Uma Noção Geral Sobre Conjuntos Fuzzy

Conjuntos fuzzy são na realidade funções que cobrem todo o domínio de uma variável, mapeando os valores assumidos pela variável em graus de pertinência. Um grau “0” significa que o valor não pertence ao conjunto, e um grau “1” significa que o valor é completamente representativo do conjunto. Como um simples exemplo, pode-se utilizar a idéia de projeto LONGO. A figura III.2 ilustra como esse conceito pode ser representado.

Os membros desse conjunto são períodos de duração, em semanas, para um projeto. O conjunto fuzzy indica para que grau um projeto de uma duração específica é um membro do conjunto dos projetos ‘longos’. Conforme o número de semanas aumenta a expectativa de que o projeto seja LONGO cresce. Um projeto de duas semanas de duração não seria considerado LONGO, mas um projeto de 10 semanas de duração teria uma pertinência moderada no conjunto dos projetos ‘longos’, e um

projeto de 15 semanas é mais certamente um projeto LONGO. É claro que a definição do que é um projeto LONGO depende do contexto no qual ela é usada.

PROJETO LONGO

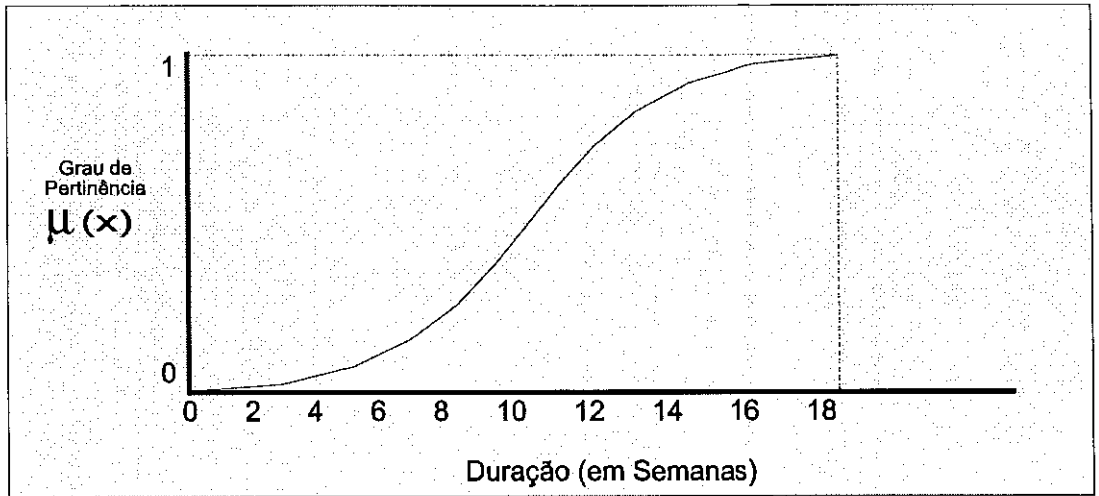


Figura III.2. A Idéia de um Projeto Longo

III.6.1.2. Variáveis Linguísticas.

O centro da técnica de modelagem fuzzy é a idéia da variável linguística. Pode-se dizer que, a variável linguística é o nome do conjunto fuzzy [40]. No exemplo anterior, o conjunto fuzzy LONGO é simplesmente uma variável linguística e poderia ser usado num sistema baseado em regras para tomar decisões apoiadas na duração de um projeto em particular:

```
if duração.projeto é LONGA
then risco.conclusão é AUMENTADO;
```

Mas uma variável linguística também carrega consigo o conceito de qualificadores dos conjuntos fuzzy, também chamados HEDGES. Esses qualificadores mudam a forma dos conjuntos fuzzy de maneiras previsíveis e funcionam do mesmo modo que advérbios e adjetivos na nossa língua. A figura III.3 ilustra a estrutura da variável linguística.

Isso permite que se escreva declarações expressivas sobre conceitos relacionados. Assim obtém-se variáveis linguísticas usando o conjunto fuzzy LONGO

– muito LONGO, pouco LONGO, levemente LONGO, e não muito LONGO. Interpreta-se essas expressões usando as mesmas regras de precedência do português; não muito LONGA e muito não-LONGA são declarações diferentes. Variáveis Linguísticas permitem à linguagem de modelagem fuzzy expressar diretamente os matizes dos significados semânticos usados pelos especialistas.

Isso é ilustrado pela seguinte regra,

```
if duração.projeto é não muito LONGA
then risco.conclusão é pouco REDUZIDO;
```

Os tipos de qualificadores que podem ser aplicados aos conjuntos fuzzy também incluem modificadores para usualidade, frequência, e contadores.

Uma variável linguística engloba as propriedades de conceitos de aproximação ou imprecisão de uma forma sistemática e computacionalmente útil.

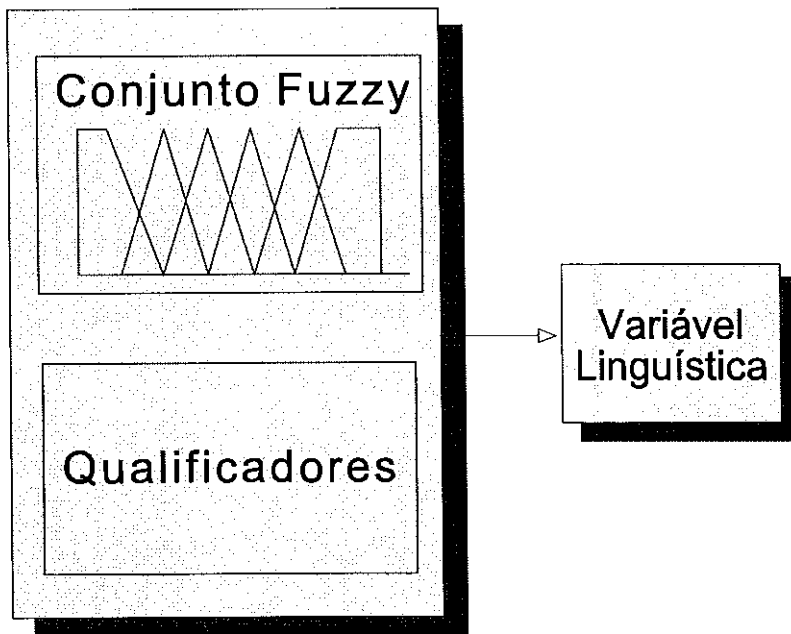


Figura III.3. Estrutura de uma Variável Linguística

III.6.1.3. Razão de Aproximação

A teoria dos conjuntos fuzzy, e de uma forma mais ampla a própria lógica fuzzy, não é um método específico para qualquer aplicação particular mais do que a lógica Booleana ou a probabilidade. A teoria dos conjuntos fuzzy sustenta uma teoria mais geral da lógica fuzzy ou cálculo de declarações fuzzy, que por sua vez sustenta as estruturas lógicas usadas para criar e manipular sistemas fuzzy. Isso é chamado de um forma mais geral de razão aproximada. A figura III.4 mostra como essas plataformas estão relacionadas.

Nas considerações feitas nesse trabalho, lógica fuzzy e razão aproximada são sempre usadas indiferentemente para indicar o processo de expressar conceitos imprecisos ou aproximados e relações. Em quase todos os casos importantes essa equivalência permanece. Entretanto é necessário estar consciente que lógica fuzzy é uma representação mais formal da teoria dos conjuntos fuzzy que a razão aproximada. Como um método de codificar o conhecimento, tanto através de regras fuzzy condicionais como incondicionais, a razão aproximada lida não apenas com a matemática implícita da lógica fuzzy, mas também incorpora um conjunto de heurísticas que estão fora da teoria dos conjuntos fuzzy, mas parecem trabalhar consistentemente e previsivelmente.

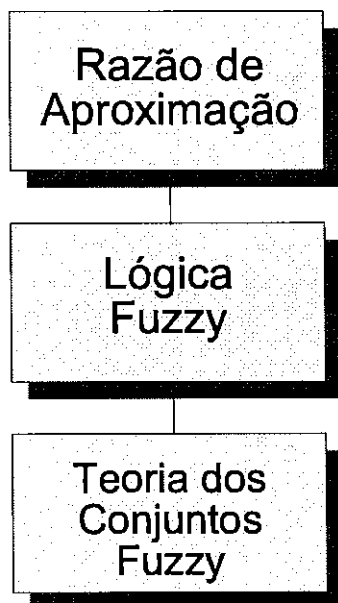


Figura III.4. Níveis de Lógica que sustentam a Razão de Aproximação

III.6.1.4. Representação do Conhecimento:

Uma representação é definida como um conjunto de convenções para descrever coisas. A maior parte das pessoas que trabalham com Inteligência Artificial concorda que desenvolver uma boa representação é a chave para a solução de problemas difíceis.

O conhecimento em uma área especializada é geralmente de três tipos: fatos, regras consideradas boas (heurística), e avaliações.

III.6.1.5. Os Problemas da Engenharia de Conhecimento

A Engenharia de Conhecimento é definida como um sub-campo da Inteligência Artificial que trata da aquisição, representação e aplicação do conhecimento ou como a disciplina da engenharia na qual o conhecimento é integrado aos sistemas para resolver problemas complexos que normalmente requerem um alto nível de perícia humana [41].

Qualquer um que desejar criar uma base de conhecimento, indiferente ao domínio específico envolvido, deve se deparar com os seguintes problemas fundamentais:

1. Os problemas de *representação do conhecimento*: Como se pode representar o conhecimento humano em termos de estruturas de dados que uma máquina possa processar?
2. O problema da *geração de inferência* : Como se pode usar essas estruturas de dados abstratas para gerar informação útil no contexto de um caso específico?

Para resolver esses problemas simultaneamente, deve-se satisfazer a dois requisitos:

1. *O requisito modular* : Durante o crescimento da base de conhecimento, os especialistas não são capazes de apresentar todos os fatos e relações necessárias para um desempenho de especialista no domínio. Sendo humanos, os especialistas tendem a esquecer ou simplificar detalhes sobre seu conhecimento, e os sistemas têm que aumentar seu conhecimento posteriormente. Uma vez que o conhecimento passado ao sistema é bastante empírico e o conhecimento dos domínios é aumentado rapidamente, os sistemas precisam ser capazes de fazer mudanças facilmente e de uma forma incremental ou modular.
2. *O requisito de programação* : A utilização fundamental de um sistema não é criar sequências de instruções para executar tarefas, mas para expressar e manipular descrições de processos computacionais e os objetos nos quais eles são aplicados.

Os requisitos acima são compatíveis e levam ao “paradigma natural da linguagem”. Os especialistas devem ser capazes de usar diretamente uma base de conhecimento durante a aquisição e exploração do conhecimento. Assim, o sistema de conhecimento sustentará uma representação numa linguagem de alto nível na qual os primeiros elementos são mais provavelmente atributos e associações de um problema de domínio particular que conceitos de programação. Linguagens não-estruturadas, incluindo a linguagem natural, tem a intenção de serem compreensíveis para o especialista com pouca ou nenhuma experiência anterior.

A idéia de utilização direta da base de conhecimento pelos especialistas usando a linguagem natural motiva a pesquisa sobre a teoria dos sistemas fuzzy.

III.6.1.6. Regras de Produção Obtidas a Partir de Modelos Verbais

A Engenharia de Conhecimento é governada por dois princípios fundamentais: O primeiro é que o poder de solução do problema de um sistema de conhecimento simbólico é primeiramente uma consequência de sua base de conhecimento e apenas

secundariamente uma consequência do método de inferência usado. Em outras palavras, sistemas especialistas devem ser ricos em conhecimento mesmo se eles são limitados em métodos.

O segundo princípio é que esse conhecimento, como a experiência tem mostrado, é bastante heurístico, isso é, ponderado, experimental, e incerto. Esse conhecimento pode ser extraído aos poucos de um especialista. Cada parte se torna uma regra de produção que representa uma resposta do especialista a questões levantadas. As questões podem ser colocadas pelo especialista, pelo projetista do sistema, ou pelo próprio sistema.

III.6.2. Vantagens dos Sistemas Fuzzy

Os Sistemas Fuzzy combinam flexibilidade com uma representação do conhecimento de apoio à decisão quase sempre de uma forma mais simples. Eles provaram habilidade como aproximadores universais conjugados com a habilidade de lidar com sistemas complexos, não-lineares e com muito ruído. Provaram-se uma ferramenta poderosa em projetos e na construção de uma nova geração de sistemas inteligentes de apoio à decisão no campo da informação.

Em geral, quando comparados a outros sistemas especialistas convencionais e sistemas de apoio à decisão, sistemas de lógica fuzzy têm as seguintes propriedades:

- São fáceis de construir.
- São fáceis de entender.
- São fáceis de verificar, validar, e ajustar.
- São mais estáveis intrinsecamente.
- São mais flexíveis e robustos.
- Podem aproximar sistemas altamente não-lineares.
- Podem ser adaptativos.
- Podem ser prototipados em um tempo significativamente reduzido.
- Podem lidar com informações vagas, incertas ou imprecisas.
- Podem incorporar conhecimentos conflitantes vindos de diferentes especialistas.

III.6.3. Limitações dos Modelos de Sistemas Fuzzy

Sistemas Fuzzy não são uma panacéia para os problemas encontrados no mundo real de informações e modelagem de processos. Eles têm limitações e restrições de representação que os tornam difíceis de usar em algumas situações. Além disso, alguns desenvolvedores, que não estão familiarizados com a lógica fuzzy, acreditam que essa tecnologia é uma maneira de calcular arbitrariamente a confiabilidade de dados com ruído ou ausentes. A seguir, comenta-se algumas das limitações encontradas nos modelos fuzzy.

III.6.3.1. Sistemas Lineares com um Modelo Matemático Bem Definido

Sistemas fuzzy são heurísticos por natureza. Isto é, eles representam o conhecimento de um especialista na aproximação de algum processo. Quando esse processo é bem comportado, ou seja, existe uma função linear bem definida que pode ser descrita matematicamente, o uso de lógica fuzzy ou qualquer outra tecnologia para sistemas especialistas é normalmente mais cara e menos precisa. Sabendo por exemplo que uma função dependente do tempo segue uma função matemática:

$$x(t) = a + mB(t - 1)$$

Pode-se escrever um modelo matemático direto que resolve $x(t)$ para cada uma das variáveis independentes. O modelo fuzzy baseado em regras que represente o mesmo processo poderia ser mais difícil de construir e de manter. Assim, para sistemas lineares de complexidade modesta, para o qual se possa escrever (ou descobrir) um modelo matemático, sistemas fuzzy são uma “segunda melhor” solução. Existem, entretanto algumas considerações na decisão de quando aplicar a lógica fuzzy para tais sistemas. Alguns dos pontos importantes a considerar no uso de um modelo fuzzy onde um modelo matemático é conhecido, inclui o que vem a seguir.

III.6.3.2. O Modelo Envolve Variáveis Fuzzy

Um número surpreendente de modelos em economia, finanças, planejamento urbano, manufatura, vendas, e transporte envolve variáveis fuzzy. Projetistas de modelos, que não estão acostumados a usar variáveis aproximadas, escrevem seus modelos usando a matemática tradicional. Como um exemplo, num modelo de orçamento capitalista poderia-se achar uma declaração como a seguinte:

```
if Caixa < ReservasMínimas do
    Débito=Débito( t-1)+ ReservasMínimas-Caixa
    Caixa=ReservasMínimas
enddo
```

Indicado isso, se o balanço do caixa (Caixa) cair abaixo de um nível de reserva de capital mínimo (ReservasMínimas), a linha de crédito da corporação (Débito) é automaticamente aumentada da quantia deficitária. Isso faz parte do sistema de orçamento capitalista atual usado em crescimento urbano. Em qualquer modelo real que é tolerante a mudanças em economias incorporadas, seria desejável ter procedimentos previamente preparados para serem executados caso sua reserva de caixa caia. Como um exemplo, poderia-se alterar a condição da parte de controle dessa regra como:

se o Caixa está muito baixo faça...

A espera para que o valor da variável atinja um limite é um processo reacionário. Em modelos que usam variáveis *crisp*, frequentemente é preciso aumentar esse limite (bufeizar o processo) e assim as ações são disparadas antes que o nível atual possa ser reconhecido pelo especialista. Num modelo fuzzy, pode-se rastrear o processo dinâmico e invocar ações do modelo baseadas no grau em que uma variável está em um determinado estado.

III.6.3.3. O Modelo Usa Dados Imprecisos

Alguns modelos lineares simples , numa análise mais cuidadosa, apresentam aparentemente uma parte de seus dados envolvendo informação imprecisa. Isto é, eles usam dados fuzzy. Devido às restrições das linguagens de computador convencionais, dados imprecisos são frequentemente agrupados em conjuntos precisos e assim essa característica passa despercebida. Como um exemplo, pode-se citar um sistema de identificação criminal que recebe fatores tais como a idade, peso e altura do suspeito. Um sistema convencional recebe valores 'crisp' e procura por suspeitos que casam com esses fatores. Como um exemplo, uma sessão de identificação poderia aparecer assim:

ENTRE COM OS SEGUINTE DADOS
SEXO: MASCULINO
IDADE: 45
PESO: 90
ALTURA: 1.80

Nesse modelo convencional, o valor da IDADE é usado para mostrar na tela o banco de dados para suspeitos que estejam dentro de uma faixa específica em torno de um valor específico. A faixa de valores poderia ser 45 ± 10 para que se pudesse selecionar indivíduos incluídos dentro do espectro de idade de 35 a 55. A figura III.5 mostra como esse conjunto de valores existem em um sistema convencional.

35 ≤ IDADE ≤ 55

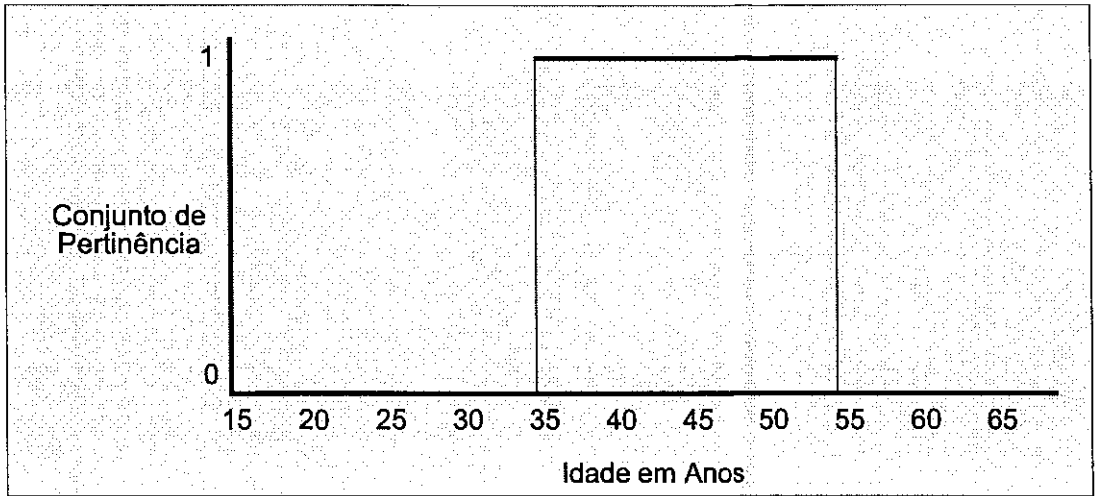


Figura III.5. A definição crisp da idade em torno de 45

Nessa abordagem, é usado um sistema para procurar na tabela de suspeitos, percorrendo registros do banco de dados criminal. Um conjunto parcial de comandos para esse processo poderia aparecer como:

```
para cada suspeito onde sexo="MASCULINO";
candidato=0;
se idade.suspeito>=(IDADE-10) ou idade.suspeito<=(IDADE+10) então
    candidato++;
•
•
```

De fato, a idade do suspeito não precisa realmente ser um valor numérico, pode ser uma estrutura conceitual definida por uma série de conjuntos fuzzy, tais como: NOVO, MEIA_IDADE, VELHO. Seguindo essa lógica, poderia-se responder ao sistema de mostra de suspeitos usando as categorias semânticas:

```
ENTRE COM OS SEGUINTE DADOS
SEXO:  MASCULINO
IDADE: MEIA_IDADE
PESO:  PESADO
ALTURA: ALTO
```

Nessa abordagem substitui-se os valores 'crisp' por nomes dos conjuntos fuzzy. Cada conjunto fuzzy define a faixa de valores para o conceito, assim como o grau de pertinência para qualquer valor característico de um suspeito. A figura III.6 mostra como o conjunto fuzzy para MEIA IDADE é definido.

MEIA-IDADE

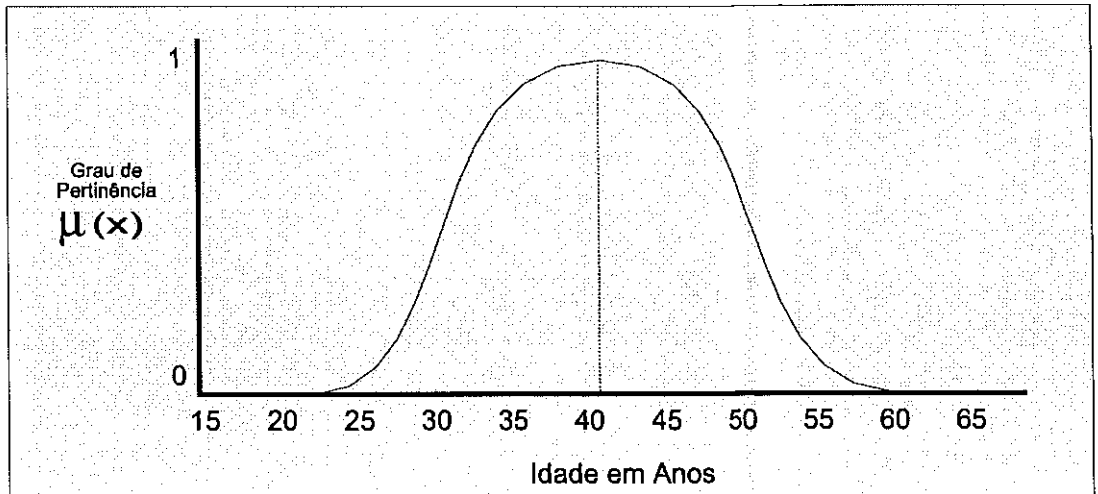


Figura III.6. A definição fuzzy para meia-idade

Usa-se um sistema que procure nos registros do banco de dados criminal da tabela de suspeitos. Para cada registro de suspeito selecionado, o valor do banco de dados é mapeado contra o conjunto fuzzy especificado. A verdade desse mapeamento (sua compatibilidade com o conjunto fuzzy) é guardada e usada para dar um valor composto para o quanto esse suspeito casa com a descrição fuzzy completa. Um conjunto parcial dos comandos para esse processo poderia aparecer como:

```
para cada suspeito onde sexo= "MASCULINO";
candidato=0;
  se idade.suspeito é MEIA IDADE então faça
    candidato++
    pertinência[candidato]=verdade
  enddo
```

-
-

O sistema protótipo de identificação criminal é típico de vários sistemas que aceitam dados de entradas crisp mais são na realidade variáveis de processamento fuzzy. Modelos em economia, *marketing*, manufaturas, controle de inventário, planejamento urbano, e assim por diante, processam tipicamente variáveis imprecisas. Nesses casos, mesmo onde o comportamento do sistema é bem entendido (e mesmo

rotina), o uso da lógica fuzzy pode prover um aumento significativo em inteligência e flexibilidade.

III.6.3.4. Revisão Regular do Modelo

O modelo pode estar sujeito a revisões regulares devido a fatores tais como mudanças regulatórias, sensibilidades econômicas, incorporações ou aquisições, flutuações monetárias internacionais, ou uma versão base do modelo localiza-se em áreas geográficas significativamente diferentes onde o desempenho do modelo é influenciado por fatores ambientais (temperatura, vibração, vento, e assim por diante). Assim a robustez, a facilidade de manutenção, e a tolerância a faltas dos sistemas fuzzy devem fornecer uma alternativa melhor para a dificuldade de “codificar” o modelo de sistema crisp convencional ou matemático. Nesses casos, o trabalho extra necessário para desenvolver e colocar para funcionar um modelo fuzzy é compensado pelo tempo necessário para reparar ou para revisar sistemas convencionais.

III.6.3.5. Os Benefícios da Facilidade Explanatória

Um sistema fuzzy, como muitos sistemas especialistas, durante seu processo de tomada de decisão, tem uma capacidade de guardar um log de todo o exame. O log do exame fornece um mecanismo para explicar seu raciocínio. Essa facilidade explanatória permite ao usuário final descobrir como o sistema chegou às suas conclusões. Isso é importante se a recomendação do sistema é inesperada. Para alguns sistemas, especialmente aqueles que fazem recomendações envolvendo o movimento de grandes quantias de dinheiro, que afetam o bem-estar dos seres humanos, ou podem aumentar consideravelmente o risco de um empreendimento planejado, a facilidade explanatória excede em importância qualquer outra consideração de desempenho, custos de desenvolvimento, e manutenção.

III.6.3.6. Manutenção do Modelo

Mesmo que exista um modelo linear bem comportado, a escolha de sua representação pode ser influenciada pela avaliação original dos engenheiros de conhecimento, desenvolvedores, e documentadores. Mesmo para modelos de complexidade modesta, um sistema fuzzy baseado em regras é normalmente mais fácil de entender, manter, e melhorar que um sistema correspondente escrito como um modelo matemático ou como um módulo de programa convencional. Quando o sistema deve ser mantido pelo usuário final ou pela equipe técnica que não era inicialmente responsável pelo modelo, um sistema fuzzy baseado em regras pode ser mais facilmente entendido que a maior parte dos sistemas convencionais.

III.6.4. Processamento da Linguagem Natural

Muita atenção tem sido direcionada ao uso da lógica fuzzy no processamento de linguagem natural, assim como as *queries* em bancos de dados para reduzir ou eliminar ambiguidades semânticas. Enquanto, a princípio, isso pareça uma aplicação natural da lógica fuzzy, dada a ambiguidade inerente à linguagem, a lógica fuzzy é de pouca aplicação nessa área. Diferentemente da ambiguidade associada com o domínio numérico das variáveis contínuas do modelo, a ambiguidade da linguagem não é computacional.

III.6.5. Lidando com Informações Ausentes

A lógica Fuzzy tem apenas habilidades limitadas para lidar com dados que estão faltando ou incompletos. Sem um mecanismo para completar os dados ausentes dentro de uma descrição do contexto real, a lógica fuzzy tem dificuldade de fazer previsões sobre as informações que estão faltando.

III.6.5.1. Ruído (Incerteza) ao Invés de Dados Ausentes

Parece ser uma noção comum que a lógica fuzzy lida com informação duvidosa, onde a dúvida é baseada em lacunas e erros (ruído) nos dados utilizados. Por isso, dados que possuem ruído podem ser manuseados pressupondo-se que esses pontos dos dados são números fuzzy. Números fuzzy representam ruído ou incerteza de um valor real de um elemento dos dados (ou um conjunto de pontos dos dados). Os dados não estão faltando, mas estão difusos em torno de um valor central possível. A figura III.7 mostra conceitos sublinhados associados com a RENDA.

Renda

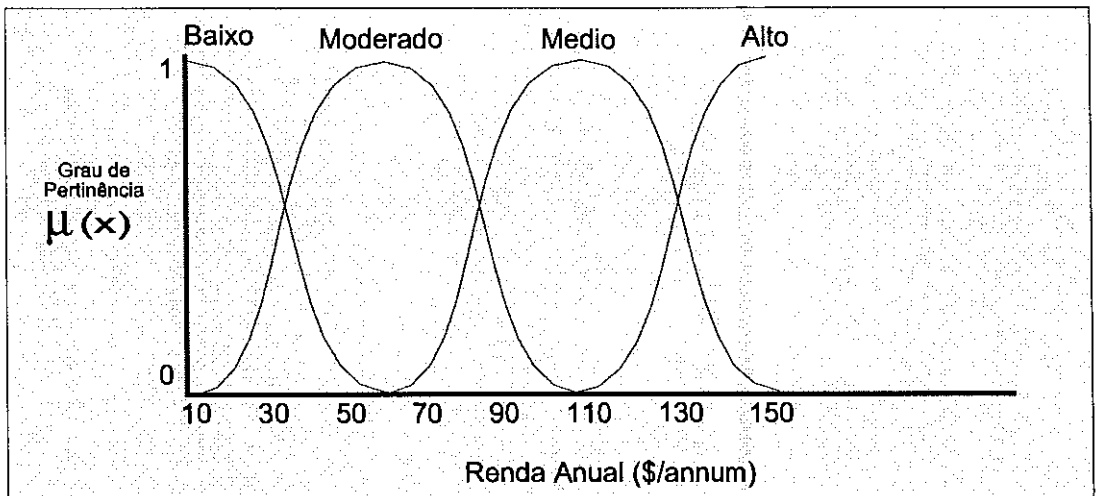


Figura III.7. Decomposição Fuzzy da variável RENDA

Pela avaliação de onde está, dentro do domínio dos conjuntos fuzzy, a parte dos dados onde existe ruído, pode-se fazer uma estimativa a respeito de seu valor possível. A figura III.8 mostra um número fuzzy representando o rendimento anual. O número é incerto devido a alguns fatores como: o suporte à documentação, a idoneidade da fonte, o tempo da leitura dos dados, e assim por diante. A largura da curva fuzzy em torno do número define a incerteza sobre essa informação. Quanto mais estreita a curva, menor é o seu grau fuzzy, e consequentemente, maior é a nossa crença de que os dados representam um simples número.

Dados Relativos à Renda

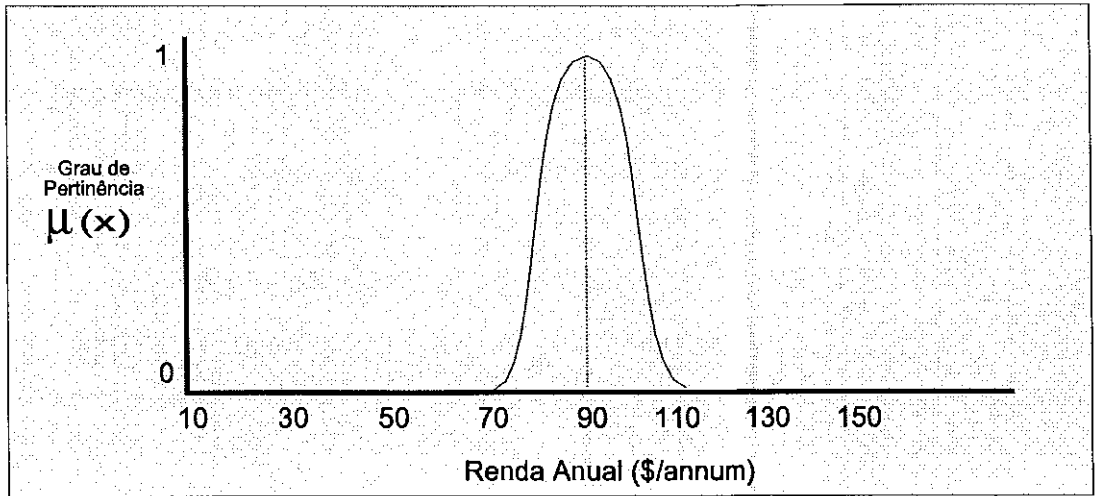


Figura III.8. Uma porção fuzzy de dados relativos à renda

Resolver os dados fuzzy envolve encontrar um encaixe entre as regiões fuzzy sob a curva de uma variável e a máxima pertinência verdadeira dos elementos dos dados fuzzy. A figura III.9 ilustra como o ruído ou a renda anual fuzzy em torno de R\$90.000,00/ano é convertido para um valor preciso de R\$94.700,00/ano com uma confiabilidade de [.93].

Renda

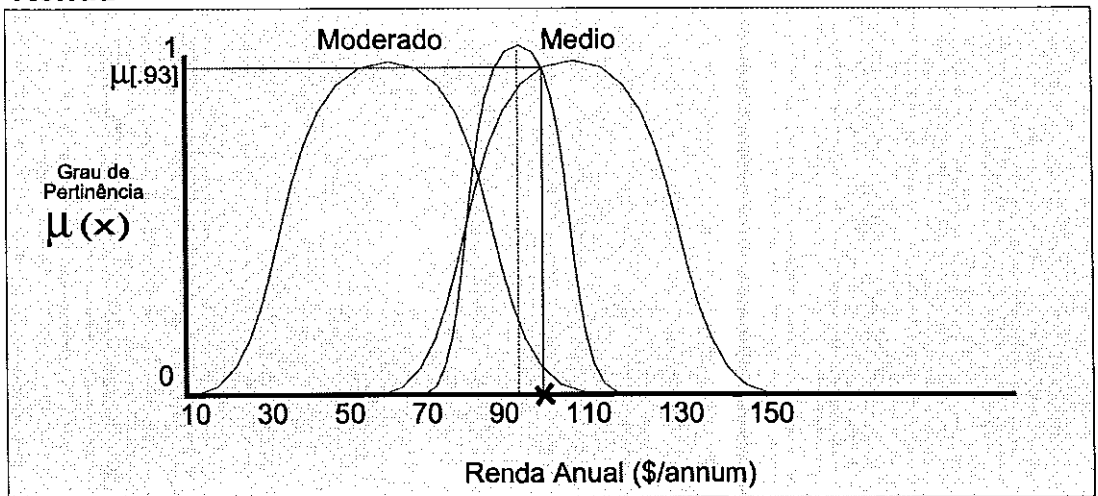


Figura III.9. Encontrando o valor esperado de um dado fuzzy

Na figura III.9, o conjunto de termos fuzzy que tem o máximo valor de interseção com os dados de entrada é usado como o especificador. Mas o ruído em torno do valor de rendimento é suficientemente grande para fazer interseção com mais de um

conjunto fuzzy. Nesse caso, pode-se usar uma técnica da média ponderada que balanceia o valor entre os vários espaços de termos fuzzy. A equação abaixo ilustra como essa média ponderada das regiões fuzzy que fazem interseção é calculada.

$$\bar{x}_w = \frac{\sum_{i=1}^n x_i \cdot m_i}{\sum_{i=1}^n m_i}$$

A equação a seguir mostra como o valor preciso para um ponto dos dados de investimento é calculado usando a média ponderada dos conjuntos fuzzy MODERADO e MÉDIO.

$$\bar{x}_w = \frac{94.7 \cdot [.93] + 82.3 \cdot [.80]}{[.93] + [.80]} = \frac{153.911}{1.73} = 88.96$$

A média ponderada move o ponto dos dados com o qual se está trabalhando para a esquerda, centrando-o entre os dois conjuntos de termos fuzzy. O valor é menor que a medida central do número fuzzy do rendimento, mas parece representar um bom encaixe baseado no grau de superposição das variáveis de rendimento dos conjuntos fuzzy. Se os dados ausentes caem abaixo do nível de ruído dos dados, então a lógica fuzzy pode ser usada para compensar os elementos que faltam. Em geral, entretanto, outras técnicas mais convencionais devem ser usadas para resolver esse problema.

III.6.5.2. Dados Desconhecidos ou Ausentes

Fornecidos pontos de referência claramente desconhecidos, a lógica fuzzy pouco pode ajudar. Ao mesmo tempo, parece óbvio que a lógica fuzzy seja frequentemente selecionada como critério para a aproximação do modelo. O objetivo do modelo a seguir foi relatado por um gerente de uma grande indústria química.

Quanto do produto X deve-se produzir, levando em conta que não se conhece o mercado competitivo, a demanda do mercado sazonal, e o custo da produção?

Embora as qualificações fuzzy forneçam uma extensão valiosa na análise da sensibilidade e da procura de objetivos, ela não é, por si mesma, uma metodologia para manusear um conjunto complexo de desconhecidos. Com informação histórica suficiente, um modelo fuzzy pode, quando combinado com técnicas de prognósticos de séries de tempo, ajudar na descoberta de valores prováveis para parâmetros do modelo que estão faltando (ausentes). A Lógica Fuzzy diz respeito à expressão e ao uso da incerteza e da imprecisão associadas com a descrição de um parâmetro, e não com a ausência de informação sobre um parâmetro (como é frequente no caso de áreas como o da análise de tolerância).

III.6.6. As Idéias da Descoberta e Extração do Conhecimento

Recentemente parece ser um tema comum entre os pesquisadores fuzzy: como se pode criar um sistema fuzzy de bancos de dados específicos. Essa abordagem, eles argumentam, resolveria vários problemas: ela descobriria e criaria os conjuntos fuzzy e formaria e escreveria as regras do sistema real. Será discutida a seguir uma abordagem a essas questões. O método adotado foi desenvolvido por Li-Xi Wang e Jerry Mendel [36] na University of Southern California num trabalho de produção de conjuntos fuzzy e regras de um conjunto descoberto a partir dos dados de desempenho do modelo. Essa técnica tem algumas de suas origens no trabalho matemático e analítico de Bart Kosk e Donald Specht.

III.6.6.1. Os Benefícios da Extração de Conhecimento

Os sistemas fuzzy comerciais podem ser complexos e difíceis de desenvolver usando aproximações convencionais à engenharia de sistemas e aquisição de conhecimento. Usando o comportamento do próprio sistema que se reflete nos dados, um modelo fuzzy pode ser desenvolvido para fornecer uma primeira aproximação do sistema final. A figura III.10 ilustra o processo geral de geração de conjuntos fuzzy e a descoberta de regras de comportamento.

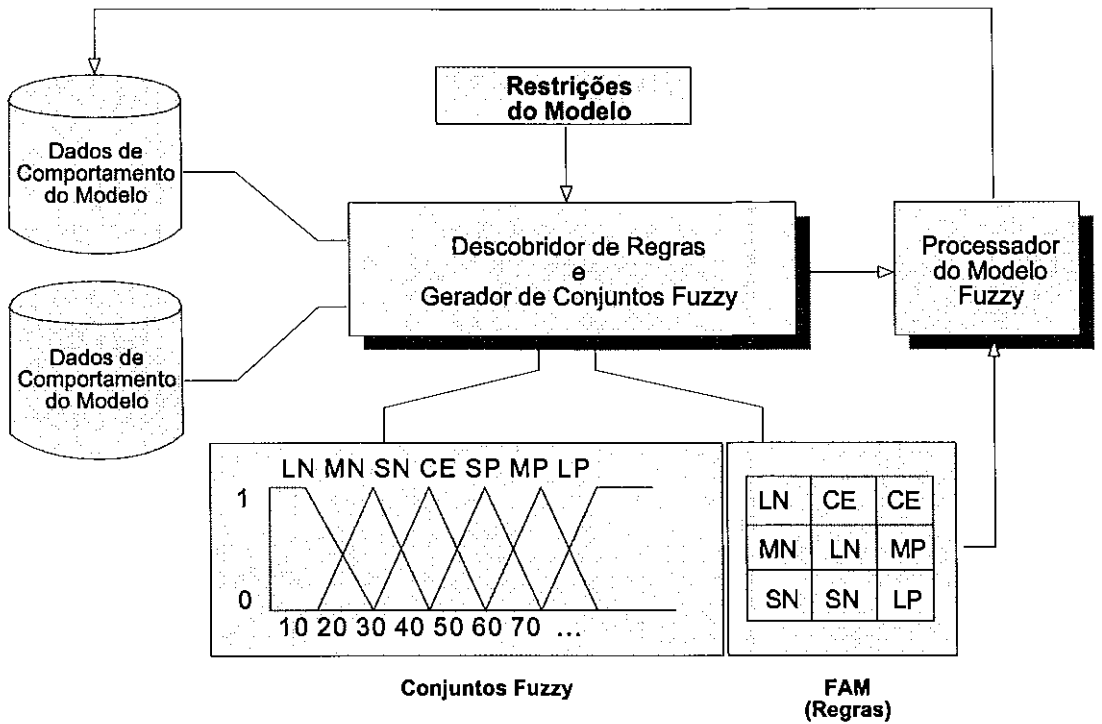


Figura III.10. Processo de construção do modelo geral de descoberta de regras

Dos conjuntos de dados de comportamento e restrições de estrutura do modelo, o mecanismo de descoberta de regras gera todos os conjuntos de termos fuzzy necessários para cada variável e então produz as regras que descrevem o comportamento do modelo. O próprio processador do modelo pode influenciar esse processo por mudar uma ou mais restrições ou modificar as características dos dados de comportamento. Esse protótipo é um mecanismo importante para a estruturação da natureza geral do sistema. Desse modelo desenvolvido especificamente para os dados, um analista pode isolar irregularidades no fluxo do processo descrito, consegue descobrir variáveis de controle ausentes ou redefinir o uso de variáveis dentro do modelo, e pode prover um mecanismo de realimentação orientado por dados que ajuste constantemente o sistema.

III.6.6.2. Problemas com a Extração do Conhecimento

A geração dos sistemas fuzzy associados à extração de conhecimento tem ganho um forte apoio na engenharia de controle e processos. Usando uma

combinação de redes neurais e algoritmos genéticos, os projetistas de sistemas de controle têm tentado reduzir o universo do problema para coleções de conjuntos fuzzy e regras que reflitam o comportamento do sistema dentro do domínio desses conjuntos. Isso parece funcionar bem para aplicações de engenharia porque os sistemas são geralmente aproximações fuzzy de modelos matemáticos e não interpretações fuzzy de sistemas comerciais complexos, conectados a bancos de dados. Isso não retira o poder e a eficiência de se produzir modelos fuzzy a partir dos dados, apenas trata de uma importante consequência no desenvolvimento de sistemas fuzzy: a representação do significado ou intenção semântica do modelo. Quando conjuntos fuzzy são gerados mecanicamente a partir do domínio de uma variável, eles são intencionalmente deficientes de alguma semântica. Eles servem de uma forma funcional simples – uma maneira de decompor a largura do domínio da variável em uma coleção de regiões fuzzy que podem ser endereçadas pelas regras. A figura III.11 ilustra como uma variável (ou uma coluna do banco de dados) pode ser arbitrariamente dividida em conjuntos fuzzy.

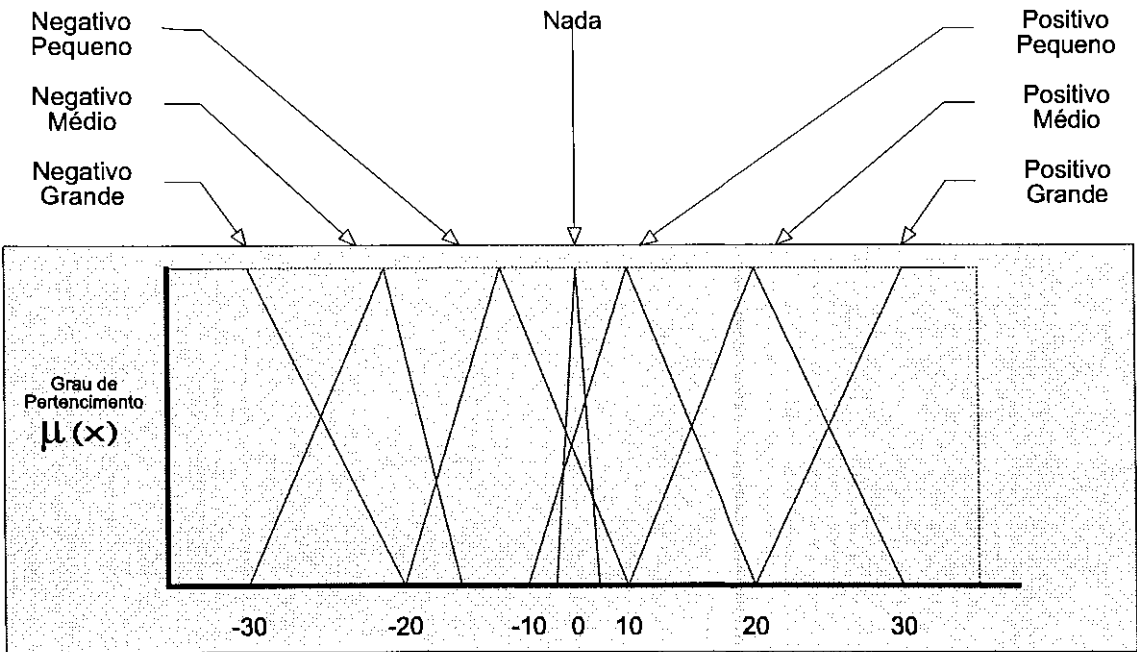


Figura III.11. Um Particionamento Mecânico em conjuntos fuzzy funcionais

Esse tipo de descoberta de dados e mapeamento é comum em assuntos de engenharia devido a natureza diferencial das variáveis de controle do modelo. Isso é, vários dos parâmetros de controle em problemas comuns de engenharia – tais como

balanço do pêndulo de duplo estágio, controle de navegação, reconhecimento de sinal digital – envolvem a medida da magnitude das mudanças de estado da variável. Os conjuntos fuzzy medem se isso é uma grande mudança, uma mudança moderada, ou uma pequena mudança tanto na direção positiva ou negativa. Entretanto, quando conjuntos fuzzy são gerados mecanicamente, como são na maior parte dos sistemas de extração de conhecimento e de descoberta de regras, existe um relacionamento muito pequeno entre o domínio de um conjunto fuzzy e a semântica do modelo como poderia ser entendida por um especialista.

III.6.7. O Método de Descoberta de Regras de Wang-Mendel

A idéia central por trás do Método Wang-Mendel [36] é simples e direta (embora sua implementação seja um tanto complexa). Um projetista de modelos enfrenta frequentemente o problema do controle da quantificação do modelo e dos parâmetros de solução do sistema em termos de sua representação linguística. Se o modelo é bem entendido e as descrições matemáticas são robustas e determinísticas, então se pode descrever o comportamento do sistema com confiança. Como um exemplo, num sistema econômico poderia-se ter um modelo de vendas baseado nos rendimentos disponíveis (D) e no custo do capital (k) do mercado consumidor. A equação a seguir mostra a função básica:

$$S = S(D, k)$$

A partir disso, a taxa de mudança nas vendas baseada nos rendimentos disponíveis e no custo do capital (a taxa de interesse dos empréstimos de consumidores), se a função é homogênea e diferenciável em qualquer ponto, é dada pela equação diferencial parcial na equação abaixo:

$$\partial S = \frac{\partial S \partial D}{\partial D} + \frac{\partial S \partial K}{\partial k} dk$$

Com essa equação em mãos, e conhecendo as mudanças nos rendimentos disponíveis (ΔD) ou o custo do capital (Δk) para qualquer período em particular, pode-se calcular uma mudança no volume de vendas projetado. Quando uma variável é fixada (ela se torna uma constante) e uma outra variável vai mudando, uma nova diferencial parcial fornece um meio de estimar o comportamento do modelo sob certas condições. A equação a seguir mostra a diferencial parcial produzida quando o custo do capital (taxas de interesse) são uma constante.

$$\partial S = \left(\frac{\partial S \partial D}{\partial D} \right)_{K \rightarrow \text{constante}}$$

Usando essas equações, pode-se fornecer valores exploratórios para cada uma das variáveis independentes e observar o comportamento da função. Isso é a essência de um modelo matemático do sistema.

Muito frequentemente, entretanto, os fenômenos que compõem o modelo — suas relações causais subentendidas — não são bem compreendidos, são altamente não-lineares, e sujeitos a múltiplas interpretações pelo conjunto de especialistas. Esses problemas são muito difíceis de modelar. Em muitos exemplos, entretanto, existem dados suficientes para modelar o caso e assim se tem o conhecimento fundamental sobre o nível de saída quando se tem uma série de entradas. A figura III.12 ilustra a entrada básica, o processo, e o modelo de saída.

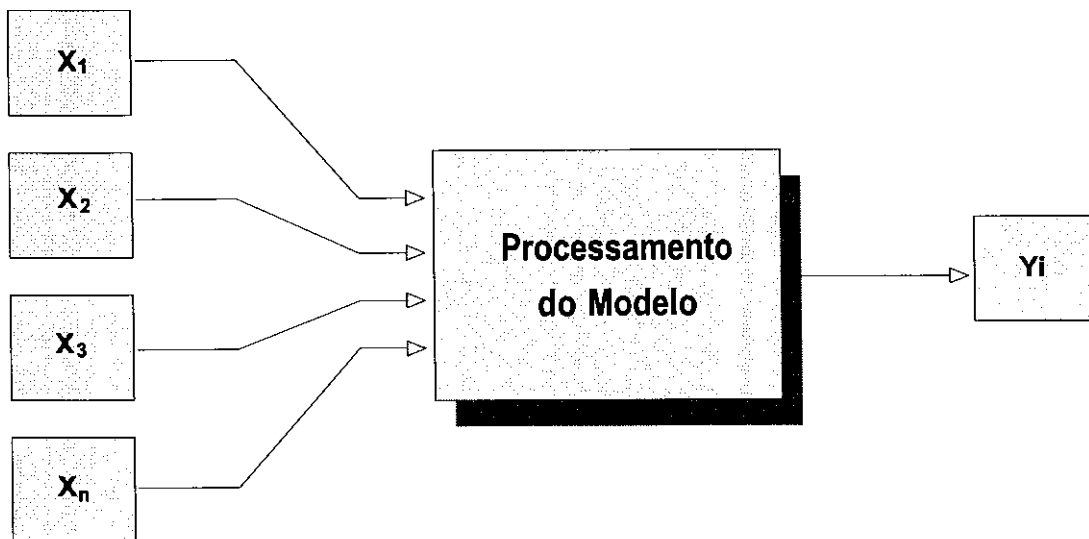


Figura III.12. O processo de entrada e saída de um modelo fuzzy

Nessa situação tem-se um mapeamento entre o estado inicial do modelo representado pelo vetor de dados quantizado $\{x_1 \dots x_n\}$ e o estado de solução do problema representado pela variável de estado decomposta $\{y_i\}$ para o i -ésimo caso do modelo. Isso significa, como mostrado na figura III.13, que provavelmente existe alguma função de transferência (g) entre esses estados.

$$\begin{aligned}
 y_1 &\leftarrow g(x_1, \dots, x_n) \\
 &\cdot \\
 &\cdot \\
 &\cdot \\
 y_k &\leftarrow g(x_{k1}, \dots, x_{kn})
 \end{aligned}$$

Figura III.13. O modelo da função de transferência

Os mecanismos reais do Processo do Modelo (veja figura III.12) são desconhecidos ou, talvez, apenas pobremente entendidos. Usando essa relação, gostaria-se de obter um conjunto de associações fuzzy condicionais ou regras na forma:

$$\text{if } x_1 \text{ é } P_{i1} \bullet x_2 \text{ é } P_{i2} \bullet \dots x_j \text{ é } P_{ij} \text{ então } y \text{ é } C_i$$

onde “P” é uma região fuzzy do predicado e “C” é uma região fuzzy consequente. Essas regiões fuzzy são geradas diretamente dos dados como aproximações dos domínios tomados como base. O processo de descoberta de regras encontra a melhor combinação de predicados e consequências consistentes com os pontos de dados. Isso significa que é preciso algum método de comparar as regras conflitantes, consolidar regras redundantes, e determinar a prioridade das regras baseadas em fatores tais como a confiabilidade nos dados e a pertinência aos conjuntos fuzzy.

III.6.7.1. O Processo de Descoberta de Regras

O processo de descoberta de regras de Wang-Mendel consiste de quatro passos básicos (embora o *software* atual combine a geração de regras e a filtragem numa simples fase). A figura III.14 mostra e explica o esquema do processo.

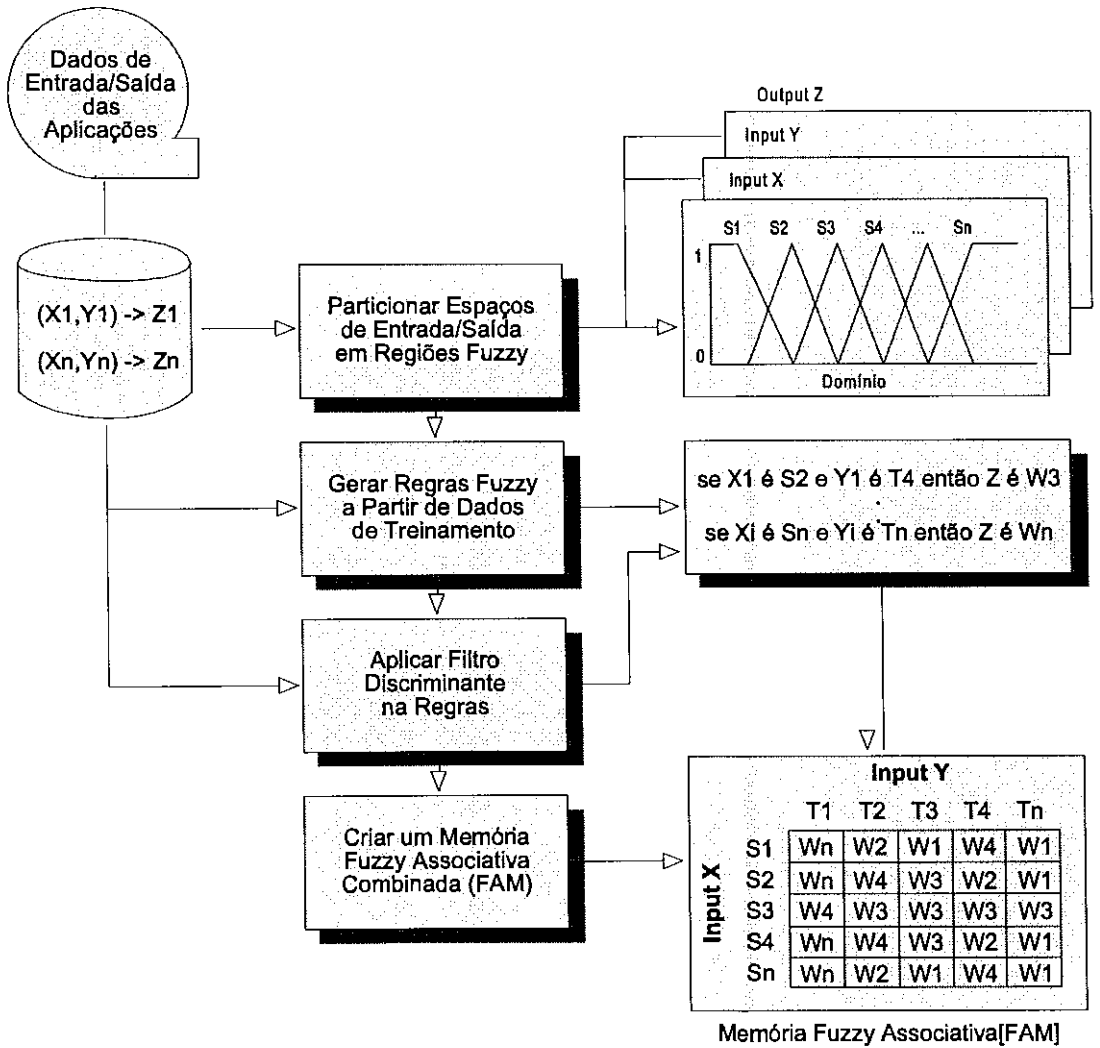


Figura III.14. O Método de Descoberta de Regras de Wang-Mendel (uma visão geral)

III.6.7.2. Um Modelo Simples de Sensibilidade de Preços

Para poder seguir o processo de descoberta de regras básicas de um processo comercial, será usado um volume de vendas simples (e admitidamente irrealista) para

o modelo de sensibilidade de preços. A partir de um número de regiões demográficas, tem-se os volumes de vendas conforme o preço do varejo do produto cresce constantemente. O modelo tem três variáveis: tempo, preço, e vendas com a relação indicada em:

$$vendas_{tempo} = f(\text{preço}_{tempo}, tempo)$$

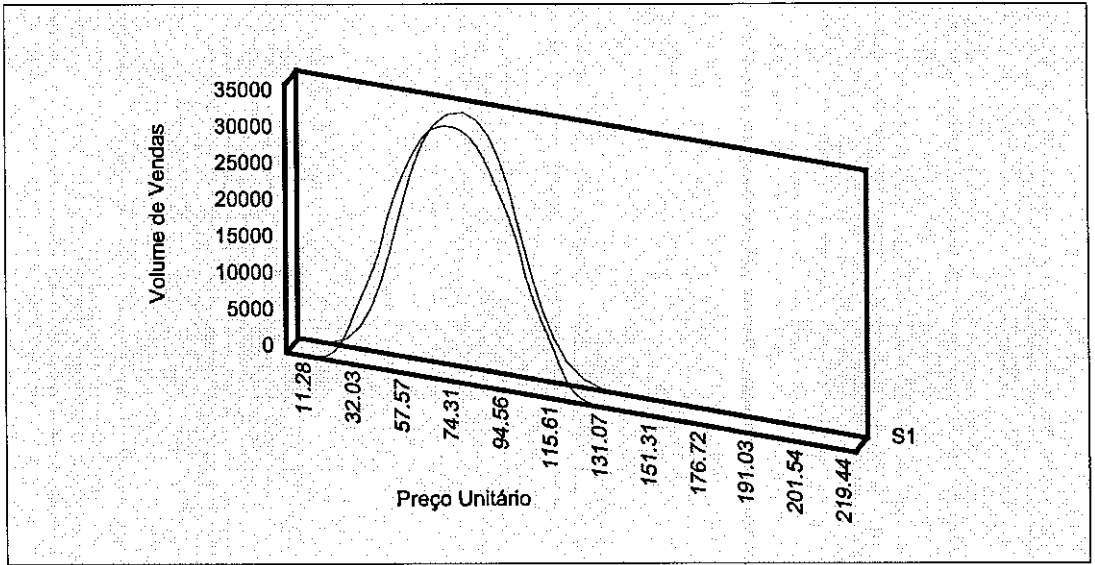


Figura III.15. A variação das vendas em função do preço

A figura III.15 mostra a curva para uma região. Esses dados são obtidos do volume de vendas e das informações de preço através dos 36 meses em uma das quatro regiões demográficas. As vendas parecem relativamente insensíveis ao preço do produto até que o preço do produto alcance um limite. Acima desse ponto, conforme o preço aumenta o volume de vendas cai rapidamente.

III.6.7.3. Representando e Usando os Dados de Comportamento do Modelo

O método de Wang-Mendel usa o comportamento do sistema para descobrir as regras fuzzy descritivas. Esses dados são colocados em um arquivo de dados e lidos pelo programa principal. O arquivo de dados tem um formato geral de:

$$x_1, x_2, x_3, \dots, x_n, y; d$$

onde “ x_i ” uma variável independente ou controlada do modelo, “ y ” é a variável solução ou dependente do sistema, e “ d ” é o grau de contribuição (fator de confiabilidade dos dados) associado com as regras implicadas pelos elementos de dados. O fator de confiabilidade dos dados pode ser omitido e é assumido como 1. Como um exemplo, a figura III.16 mostra uma parte do arquivo de comportamento para o modelo de sensibilidade de preços discutido anteriormente.

// tempo	preço	vendas
//-----	-----	-----
1.	11.28,	40.96
2.	17.45,	336.82
3.	28.75,	574.15
4.	32.03,	1066.51
5.	40.29,	2003.77
6.	49.92,	3646.68
7.	57.57,	6418.38
8.	58.81,	10607.24
9.	66.19,	15860.38
10.	74.31,	21624.00
11.	81.06,	27379.11
12.	89.42,	31538.08
13.	94.56,	32066.22
14.	101.49,	29985.85
15.	104.64,	23837.87
16.	115.61,	16075.54

Figura III.16. Um modelo de comportamento do arquivo de treinamento

As linhas que começam com uma barra dupla (//) são comentários e são ignorados pelo sistema de descoberta de regras. O arquivo de treinamento pode conter qualquer número de registros. Pode-se especificar um grau de contribuição para elementos dos dados como indicado na figura III.17.

// tempo	preço	vendas	
//-----	-----	-----	
1.	11.28,	40.96	: .4
2.	17.45,	336.82	: .4
3.	28.75,	574.15	: .4
4.	32.03,	1066.51	: .4
5.	40.29,	2003.77	: .4
6.	49.92,	3646.68	: .7
7.	57.57,	6418.38	: .7
8.	58.81,	10607.24	: .7

Figura III.17. Arquivo de comportamento com os graus de contribuição

Cada linha do arquivo de comportamento implica em uma regra sobre o comportamento do sistema. A finalidade do mecanismo de descoberta de regras é aprender o comportamento agregado do sistema a partir das relações entre as variáveis dependentes e independentes.

III.6.8. Executando o Processo de Descoberta de Regras

Uma vez que um arquivo de dados foi criado, pode-se invocar a descoberta de regras com o comando *wmdriver*. A figura III.18 mostra como tal sessão apareceria usando o arquivo de comportamento de vendas.

```
wmdriver
Enter name of Data File: sales.dat
1.0- -Partition Variables into Fuzzy Sets.
1.1- - -Generating Term Set for 'time'.
1.1- - -Generating Term Set for 'price'.
1.1- - -Generating Term Set for 'sales'.
2.0- -Generate Intermediate Rules from Trainig Data.
2.1- - -Learning Rules from Data File: 'sales.dat'
2.2- - -Formed 144 Rules.
3.0- -Create Combined FAM and write Production Rules.
3.1- - -Generating FAM from Rule File: 'B:WMrules.FIL'.
3.2- - -Combined FAM contains 23 Rules.
```

Figura III.18. Executando o processo de descoberta de regras

De uma entrada contendo 144 linhas (cada linha é uma regra implícita), o processo de descoberta cria uma Memória Fuzzy Associativa (FAM-Fuzzy Associative Memory) combinada contendo apenas as 23 regras verdadeiras. Essas regras podem ser usadas pelo sistema de modelagem fuzzy. Durante o processamento, o sistema de descoberta de regras cria uma série de arquivos de auditoria dos logs contendo informação detalhada sobre o estado do processo em cada fase, e também cria arquivos intermediários – tais como a matriz de regras – que é lida por outras partes do sistema durante os passos subsequentes. A figura III.19 mostra o alto nível esquemático dos vários passos do programa de descoberta de regras de Wang-Mendel e os arquivos que são escritos ou lidos durante a execução.

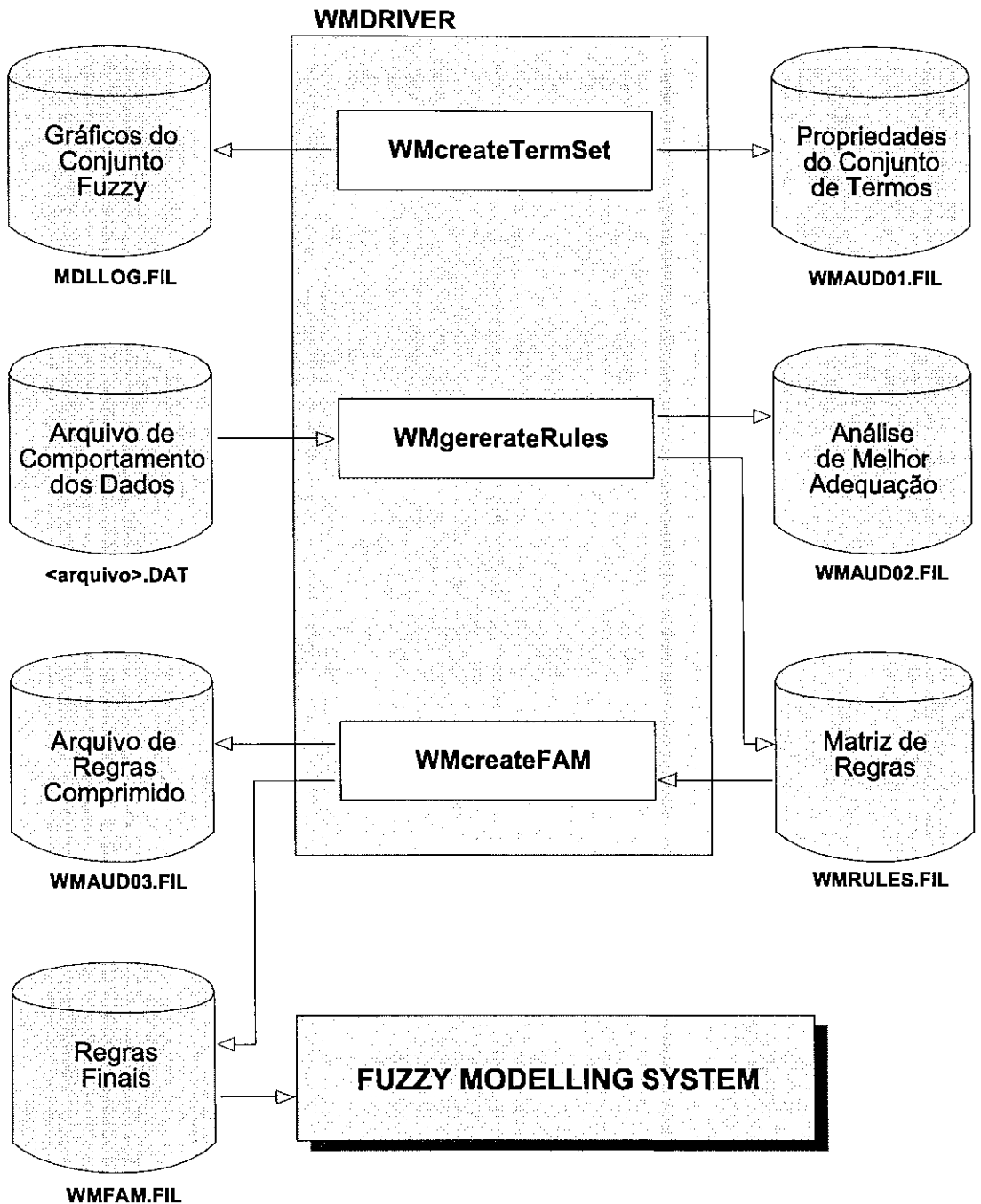


Figura III.19. O fluxograma do processo para o método de descoberta de regras Wang-Mendel

Pode-se usar esse tipo de sistema para gerar um modelo fuzzy a partir de vários conjuntos de dados diferentes (que se espera que estejam relacionados). Em particular, a geração de regras real é dirigida pelo conteúdo da matriz de regras descomprimida (wmrules.fil). Salvando esse arquivo durante várias execuções da descoberta de regras (com dados periódicos, regionais, avançados, ou novos como

exemplo) pode-se criar uma matriz de regras combinadas. Alimentar essa matriz de regras no programa WMcreateFAM gera um conjunto de regras das informações combinadas e atualizadas.

III.6.8.1. Passo Um - Decompor Variáveis em Conjuntos Fuzzy

Um processo de descoberta de regras começa com a geração de conjuntos fuzzy para cada uma das variáveis de entrada e saída. O Universo de Discurso completo da variável é decomposto em um conjunto de curvas em forma de sino (curvas PI) com os pontos limite à direita e à esquerda representados como conjuntos fuzzy com bordas. Essa decomposição é direcionada pelo analista de conhecimento—o número de conjuntos fuzzy a ser utilizado é baseado na escolha de quantas regiões seriam necessárias para representar corretamente o domínio da variável. Isso seria um número ímpar maior que três. Existe sempre um grande número de regiões fuzzy devido à forma em que o processo de decomposição fixa um conjunto fuzzy central e então move para a esquerda e para a direita para cobrir o resto do domínio da variável. O processo começa como ilustrado na figura III.20, pela criação de um conjunto fuzzy no centro do domínio da variável.

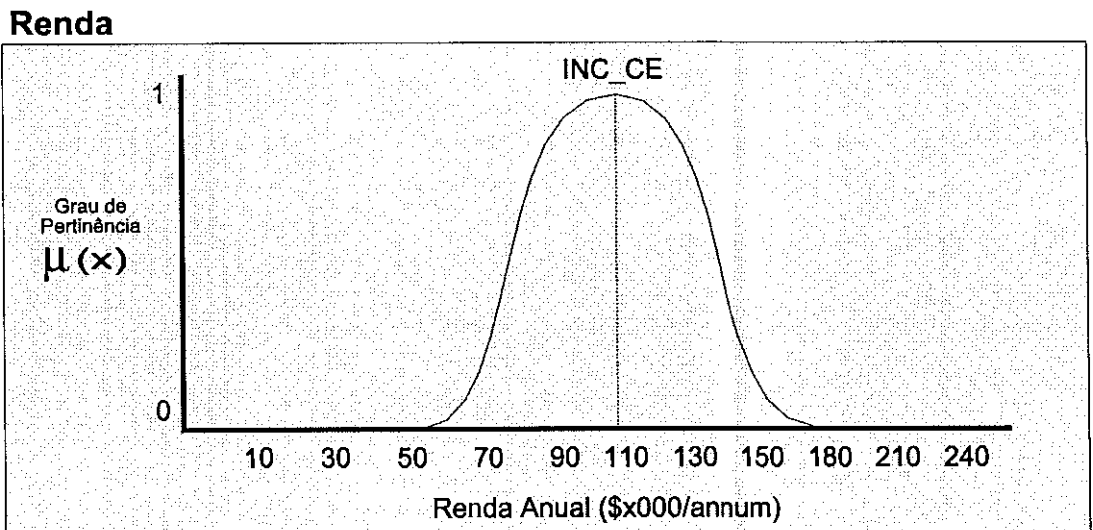


Figura III.20. Gerando o conjunto fuzzy central em um domínio de variável

Grande parte da decomposição é manuseada pelo próprio processo de descoberta. Depois que o analista especifica o número de conjuntos fuzzy básicos de uma variável, a largura de cada conjunto fuzzy em forma de sino, o grau de superposição, e os pontos de inflexão dos conjuntos de curvas S usados como os pontos limites à esquerda e à direita são determinados pelo processo metodológico.

Quando o conjunto fuzzy central foi instalado, os conjuntos fuzzy remanescentes são adicionados pelo movimento à esquerda e à direita ao longo do resto do domínio explícito da variável. As posições fuzzy finais são posicionadas à direita e à esquerda das curvas S que já foram instaladas. A figura III.21 mostra a primeira parte do processo para a porção esquerda da variável RENDA.

Renda

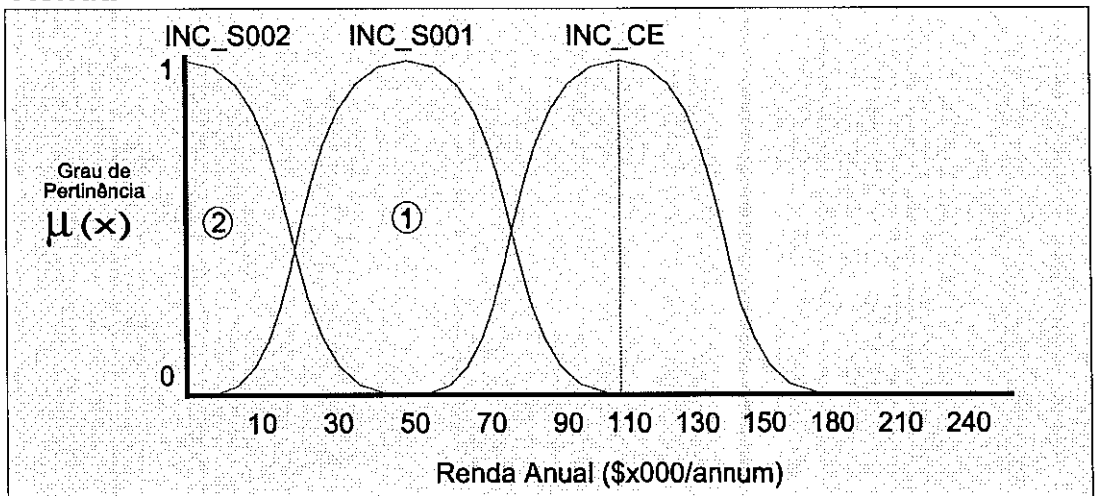


Figura III.21. Gerando a vizinhança e os conjuntos fuzzy básicos

O processo de decomposição da descoberta de regras rotula automaticamente os conjuntos fuzzy para a esquerda do centro, com S001 até S999 (indicando que os valores são inferiores que o centro) e àqueles à direita do centro com B001 até B999 (indicando que os valores são maiores que o centro). Na figura III.19 o conjunto fuzzy INC_S001(①) é adicionado imediatamente para a esquerda do centro. Existe uma superposição de 50% significando que a sua borda direita com pertinência zero está no centro do conjunto fuzzy (o valor do domínio com pertinência 1(um)). O conjunto fuzzy INC_S002 (②) é então adicionado à esquerda de INC_S001. Esse é um

conjunto fuzzy na forma de uma curva-S; e tem também uma superposição de 50% com INC_S001.

Pode-se observar que a decomposição não é relacionada a qualquer estrutura semântica explícita (ou implícita) da variável. Isso é, as divisões não correspondem a conceitos tais como: rendimentos baixos, moderados, médios, ou altos. Isso não significa que o processo de descoberta de regras deva estar desacoplado de tal conhecimento, significa apenas que a geração automática dos conjuntos fuzzy é geralmente feita sem qualquer meta-conhecimento sobre as propriedades e natureza semântica da variável. O processo trabalharia igualmente bem se o analista substituísse os conjuntos fuzzy automáticos por aqueles criados pela aquisição de conhecimento.

A decomposição das variáveis é executada na primeira fase do processo de descoberta de regras usando os contadores dos conjuntos fuzzy providos pelos sistemas ou pelo engenheiro de conhecimento. Esse número pode ser diferente para cada variável. A referência é “wmpartv.cpp” que cria o conjunto fuzzy dos termos para cada variável. O processo de criação dos conjuntos fuzzy é registrado no arquivo “wmaud01.fil” (arquivo de auditoria) enquanto os gráficos dos conjuntos fuzzy são armazenados no arquivo “mdllog.fil”(arquivo de auditoria do modelo padrão). Nas próximas figuras os conteúdos dos arquivos de auditoria e log do modelo foram combinados (e rearranjados) para ilustrar como a decomposição do processo é refletida em ambos os arquivos.

A figura III.22 mostra o processo usado pelo mecanismo de descoberta de regras para decompor a variável TIME em nove conjuntos fuzzy básicos.

```
---PI FuzzySet 'TIM_CE' created.  
---PI FuzzySet 'TIM_B001' created.  
---PI FuzzySet 'TIM_B002' created.  
---PI FuzzySet 'TIM_B003' created.  
---Growth S-Curve FuzzySet 'TIM_B004' created.  
---PI FuzzySet 'TIM_S001' created.  
---PI FuzzySet 'TIM_S002' created.  
---PI FuzzySet 'TIM_S003' created.  
---Decline S-Curve FuzzySet 'TIM_S004' created.
```

Figura III.22. Criando os conjuntos fuzzy para a variável TIME

O processo cria o centro do domínio dos conjuntos fuzzy, completa o espaço à direita do centro (os conjuntos “Bxxx”), e então completa o espaço à esquerda do centro (os Sxxx). A análise completa de como uma variável que foi decomposta é guardada no conjunto de log. A figura III.23 mostra o conteúdo desse relatório.

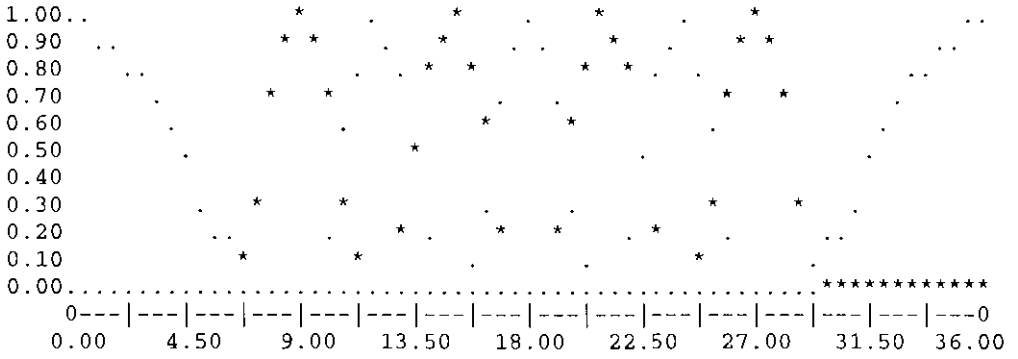
```
RD---[001]: Rule Discovery. Fuzzy Term Set Generation
Variable : time
Domain :      0.00 to   36.00
INITIAL PARAMETERS
DomRange :    36.00
MidPoint :    18.00
Partitions:    9
PartWidth :    6.00
```

FuzzySet	Curve Edges		Curve Surface Parameters		
	Left	Right	P1	P2	P3
1. TIM_S004	0.00	9.00	0.00	4.50	9.00
2. TIM_S003	6.00	12.00	9.00	3.00	9.00
3. TIM_S002	9.00	15.00	12.00	3.00	12.00
4. TIM_S001	12.00	18.00	15.00	3.00	15.00
5. TIM_CE	15.00	21.00	18.00	3.00	18.00
6. TIM_B001	18.00	24.00	21.00	3.00	21.00
7. TIM_B002	21.00	27.00	24.00	3.00	24.00
8. TIM_B003	24.00	30.00	27.00	3.00	27.00
9. TIM_B004	27.00	36.00	27.00	31.50	36.00

Figura III.23. Relatório detalhando as propriedades do conjunto de termos para TIME

O DomRange (extensão do domínio) é a diferença entre os valores baixos e altos do domínio da variável. O MidPoint é o valor de domínio que corresponde ao meio do caminho do domínio e é o centro do conjunto fuzzy central (TIM_CE, no exemplo). As Partições indicam o número de conjuntos fuzzy no conjunto de termos das variáveis. Esse número é fornecido pelo engenheiro de conhecimento (se o número é par, ele é incrementado de um para torná-lo ímpar). A PartWidth é a largura da partição e indica o comprimento da base de cada conjunto fuzzy em forma de sino. A metade da PartWidth forma a extensão da curva PI. Nessa abordagem, CurveEdges indicam as bordas esquerda e direita dos conjuntos fuzzy enquanto Curve Surface Parameters indicam os parâmetros de especificação para cada conjunto fuzzy PI ou curva-S. A figura III.24 mostra os conjuntos fuzzy produzidos por esse método de decomposição.

time
 Domain [UofD]: 0.00 to 36.00



- . FuzzySet: TIM_S004
 Support : 0.00, 4.50, 9.00
- * FuzzySet: TIM_S003
 Support : 9.00, 3.00
- . FuzzySet: TIM_S002
 Support : 12.00, 3.00
- * FuzzySet: TIM_S001
 Support : 15.00, 3.00
- . FuzzySet: TIM_CE
 Support : 18.00, 3.00
- * FuzzySet: TIM_B001
 Support : 21.00, 3.00
- . FuzzySet: TIM_B002
 Support : 24.00, 3.00
- * FuzzySet: TIM_B003
 Support : 27.00, 3.00
- . FuzzySet: TIM_B004
 Support : 27.00, 31.50, 36.00

Figura III.24. Os conjuntos fuzzy associados com a variável TIME

Durante a fase de decomposição da variável, cada variável do modelo é dividida em um coleção de conjuntos fuzzy. Uma regra fuzzy só pode especificar uma ação dentro de uma região fuzzy definida. As figuras a seguir mostram a forma com que o processo de descoberta de regras decompõem as variáveis PRICE e SALES em conjuntos de termos fuzzy. A figura III.25 é o relatório do arquivo de auditoria sobre a variável preço.

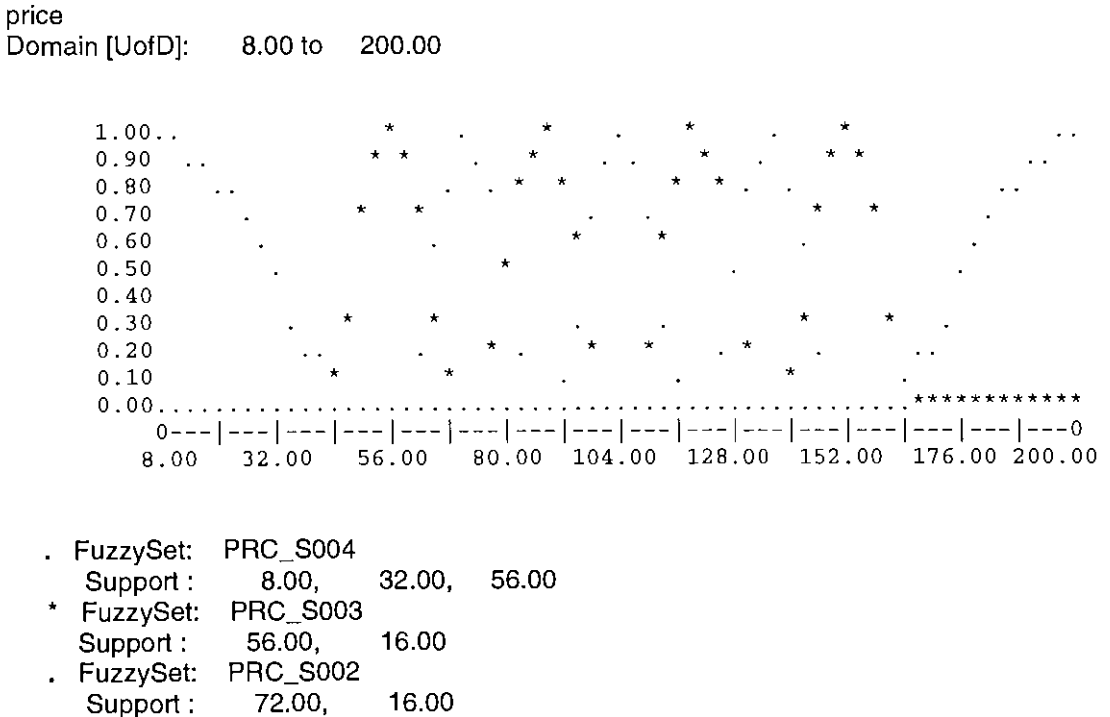
RD---[001]: Rule Discovery. Fuzzy Term Set Generation

Variable : price
 Domain : 8.00 to 200.00
 INITIAL PARAMETERS
 DomRange : 192.00
 MidPoint : 104.00
 Partitions: 9
 PartWidth : 32.00

FuzzySet	Curve Edges		Curve Surface Parameters		
	Left	Right	P1	P2	P3
1. PRC_S004	8.00	56.00	8.00	32.00	56.00
2. PRC_S003	40.00	72.00	56.00	16.00	56.00
3. PRC_S002	56.00	88.00	72.00	16.00	72.00
4. PRC_S001	72.00	104.00	88.00	16.00	88.00
5. PRC_CE	88.00	120.00	104.00	16.00	104.00
6. PRC_B001	104.00	136.00	120.00	16.00	120.00
7. PRC_B002	120.00	152.00	136.00	16.00	136.00
8. PRC_B003	136.00	168.00	152.00	16.00	152.00
9. PRC_B004	152.00	200.00	152.00	176.00	200.00

Figura III.25. Relatório detalhando as propriedades do conjunto de termos para PRICE

O domínio de PRICE é \$8.00 a \$200.00 com uma variação de domínio de \$192.00. Essa variável também é decomposta em nove conjuntos fuzzy. A figura III.26 mostra os conjuntos fuzzy produzidos pelo processo de decomposição.



```

* FuzzySet: PRC_S001
  Support : 88.00, 16.00
. FuzzySet: PRC_CE
  Support : 104.00, 16.00
* FuzzySet: PRC_B001
  Support : 120.00, 16.00
. FuzzySet: PRC_B002
  Support : 136.00, 16.00
* FuzzySet: PRC_B003
  Support : 152.00, 16.00
. FuzzySet: PRC_B004
  Support : 152.00, 176.00, 200.00

```

Figura III.26. Os conjuntos fuzzy associados com a variável PRICE

Uma vez que a variável SALES é uma variável de saída (solução) e a natureza da curva requer um grau de granularidade moderado, a variável é decomposta em dezessete conjuntos fuzzy. A figura III.27 mostra o relatório da auditoria na decomposição de vendas em seus conjuntos de termos.

```

RD---[001]: Rule Discovery. Fuzzy Term Set Generation
Variable : sales
Domain : 0.00 to 90000.00
INITIAL PARAMETERS
DomRange : 90000.00
MidPoint : 45000.00
Partitions: 17
PartWidth : 7941.18
Curve Edges      Curve Surface Parameters
FuzzySet         Left    Right   P1      P2      P3
-----
1. SAL_S008      0.00  17205.88  0.00  8602.94  17205.88
2. SAL_S007     13235.29  21176.47  17205.88  3970.59  17205.88
3. SAL_S006     17205.88  25147.06  21176.47  3970.59  21176.47
4. SAL_S005     21176.47  29117.65  25147.06  3970.59  25147.06
5. SAL_S004     25147.06  33088.24  29117.65  3970.59  29117.65
6. SAL_S003     29117.65  37058.82  33088.24  3970.59  33088.24
7. SAL_S002     33088.24  41029.41  37058.82  3970.59  37058.82
8. SAL_S001     37058.82  45000.00  41029.41  3970.59  41029.41
9. SAL_CE       41029.41  48970.59  45000.00  3970.59  45000.00
10. SAL_B001    45000.00  52941.18  48970.59  3970.59  48970.59
11. SAL_B002    48970.59  56911.76  52941.18  3970.59  52941.18
12. SAL_B003    52941.18  60882.35  56911.76  3970.59  56911.76
13. SAL_B004    56911.76  64852.94  60882.35  3970.59  60882.35
14. SAL_B005    60882.35  68823.53  64852.94  3970.59  64852.94
15. SAL_B006    64852.94  72794.12  68823.53  3970.59  68823.53
16. SAL_B007    68823.53  76764.71  72794.12  3970.59  72794.12
17. SAL_B008    72794.12  90000.00  72794.12  81397.06  90000.00

```

Figura III.27. Relatório detalhando as propriedades do conjunto de termos para SALES

As extremidades esquerda e direita da função de volume de vendas tem valores muito pequenos enquanto a porção central da função tem valores entre aproximadamente 35.000 e 85.000 unidades. Assim a variação da variável SALES é bem grande, indo de 0 até 90.000. A figura III.28 mostra os conjuntos fuzzy associados com a variável SALES.

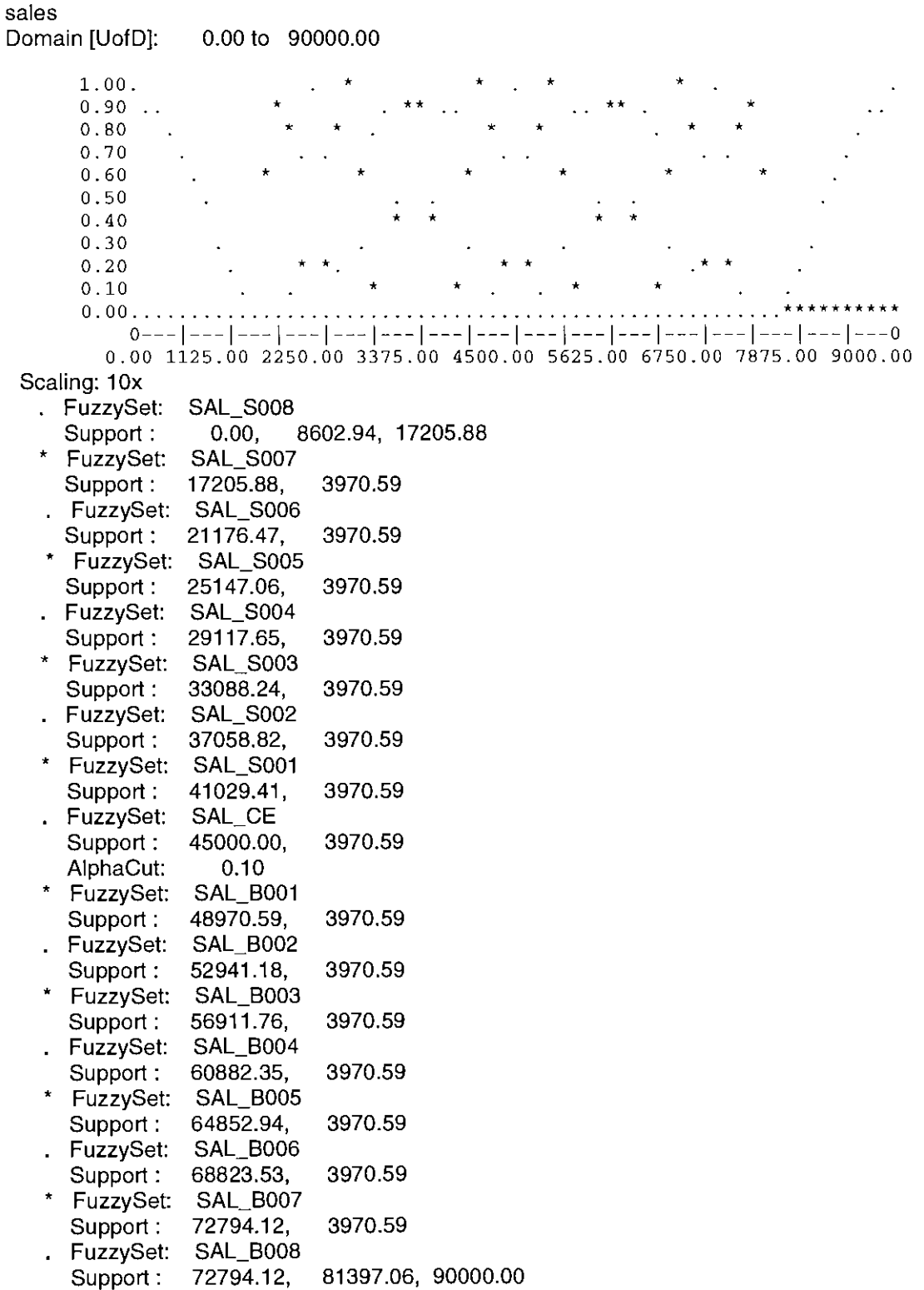


Figura III.28. Os conjuntos fuzzy associados com a variável SALES

O processo de decomposição cria um aglomerado de conjuntos fuzzy que ainda assim se encontram espaçados através do domínio da variável (ou Universo do Discurso). Cada conjunto fuzzy tem uma superposição de 50% com os vizinhos adjacentes e assim quando a pertinência em um conjunto fuzzy é totalmente verdadeira, a pertinência em um vizinho superposto é completamente falsa. Várias modificações razoáveis ao processo de geração dos conjuntos fuzzy são úteis em várias situações de modelagem. Uma mudança óbvia envolve alterar a forma na qual os conjuntos fuzzy estão situados no domínio da variável. Em várias situações, especialmente quando se lida com variáveis de saída não-lineares, poderia-se querer a densidade dos conjuntos fuzzy localizada em torno de alguma medida central que seja maior que a densidade ou alcance dos conjuntos fuzzy conforme se afaste desse valor. Fornecer um vetor de fatores de densidade ou difusão permitiria ao processo gerar uma distribuição fuzzy não-uniforme. Uma outra modificação que se mostra muito útil é mudar o tipo de conjunto fuzzy instalado em alguns pontos através do domínio. Além da mudança óbvia para formas triangulares, o uso dos conjuntos fuzzy trapezoidais é algumas vezes benéfico. Conjuntos fuzzy trapezoidais são adequados para representar mudanças fuzzy abruptas através do domínio de uma variável (especialmente funções de grau que tenha dicotomias implícitas na sua face). Finalmente, é ocasionalmente necessário mudar o grau de superposição entre os conjuntos fuzzy vizinhos. Em algumas aplicações de controle e reconhecimento de padrão, as bordas de uma área de controle ou imagem podem ser mais bem gerenciadas se a superposição do conjunto fuzzy é maior que 50%.

III.6.8.2. Passo Dois - Gera Regras de Tentativa Para os Dados

Depois que a variável do modelo foi decomposta em conjuntos fuzzy, pode-se começar o processo de produção de regras. A produção de regras é atualmente feita em vários passos, cada passo desenvolvido para encontrar o melhor conjunto de regras que reflitam o comportamento geral do modelo. Nesse passo, o processo de descoberta de regras cria uma matriz de tentativa de regras baseada no máximo encaixe dos dados com os conjuntos fuzzy básicos. Essa fase começa lendo o comportamento do modelo ou os dados de treinamento. A figura III.29 mostra parte dos dados de treinamento dos modelos de venda.

//time	price	sales
//-----	-----	-----
1,	11.28,	40.94
2,	17.45,	336.82
3,	28.75,	574.15
4,	32.03,	1066.51
5,	40.29,	2003.77
6,	49.92,	3646.68
7,	57.57,	6418.38
8,	58.81,	10607.24
9,	66.19,	15860.38
10,	74.31,	21624.00
11,	81.06,	27379.11
12,	89.42,	31538.08
13,	94.56,	32066.22
14,	101.49,	29985.87
•		
•		
•		

Figura III.29. Uma parte do arquivo de comportamento das vendas

Cada linha do conjunto de dados de treinamento (como um exemplo, sales.dat) contém uma entrada do modelo e o comportamento de saída. Por convenção, o último elemento dos dados é a variável dependente ou solução. O arquivo de treinamento do comportamento representa o processamento do modelo na forma de regra fuzzy,

se tempo é TIM_xxxx e preço é PRC_xxxx então vendas são SAL_xxxx

onde xxxx representa um dos conjuntos fuzzy básicos da variável especificada. É claro que são pontos dentro dos próprios dados que ajudam o processo de descoberta a formar uma regra de tentativa para cada linha procurando a relação que melhor encaixe cada valor dos dados e um conjunto fuzzy no conjunto de termos associados à variável. A idéia de melhor encaixe significa simplesmente o conjunto fuzzy onde os elementos dos dados têm valor de pertinência máximo. Assim, para um elemento dos dados, x , associado a uma variável V , o melhor encaixe entre os conjuntos fuzzy nas P partições, é encontrado.

A equação abaixo mostra esse processo:

$$\mu(x)_{max} = g(x_i, \bigvee_{j=0}^N P_j)$$

Como um exemplo, para um dado valor da variável PREÇO, digo \$63.00, pode-se ver na figura III.30 que ela tem uma pertinência inicial nos conjuntos fuzzy PRC_S003 e PRC_S002. O melhor encaixe, ou o seu máximo encaixe, é PRC_S003.

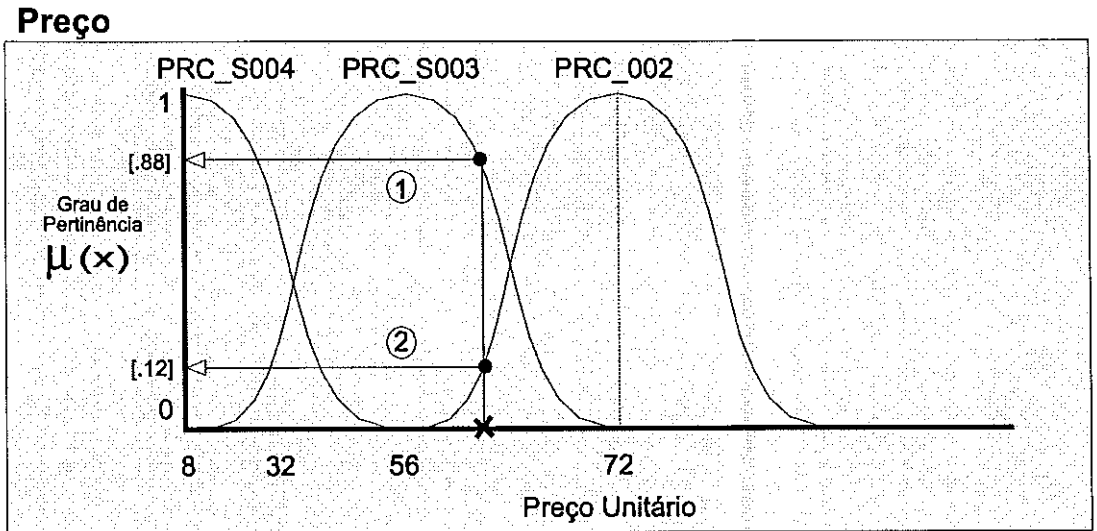


Figura III.30. Encontrando o máximo (“Melhor Encaixe”) para um ponto dos dados

Dessa avaliação de pertinência, a descoberta de regras forma uma proposição de tentativa fuzzy, preço é PRC_S003 e também armazena o seu grau de compatibilidade (.88). Quando cada elemento de entrada e saída é avaliado para encontrar a sua máxima pertinência, uma regra de tentativa é gerada:

se tempo é TIM_S001 e preço é PRC_S003 então vendas são SAL_S001

Essa regra de tentativa, junto com os graus de pertinência para cada um dos predicados assim como as proposições consequentes são armazenados na matriz intermediária. A matriz de regras intermediária forma a base para todo o processo de descoberta de regras. Conforme cada registro é lido do arquivo de comportamento dos dados, os valores dos dados são usados para achar a máxima pertinência em um dos conjuntos fuzzy associados com a variável. Essas pertinências máximas são usadas para calcular o Grau de Eficácia e, finalmente, o Grau de Contribuição. As regras são eventualmente filtradas baseadas em seus relativos graus de contribuição para a descrição do comportamento do sistema.

Solucionando o Grau de Eficácia para Cada Regra

Como cada linha do arquivo de comportamento dos dados gera uma entrada na matriz de regras intermediária, o processo de descoberta cria várias regras potenciais. Várias dessas regras especificam atitudes conflitantes e outras têm contribuições de peso relativamente fracas. Para poder fornecer um mecanismo que a consolidação de regras possa utilizar para selecionar uma regra baseada na sua força de descrição do comportamento do modelo, calcula-se para cada regra um grau de eficácia. A eficácia básica de uma regra (designada $Eff(x)$) é o produto do grau de pertinência máximo por cada componente da regra. A equação a seguir é a definição do grau de eficácia:

$$Eff(rule) = \prod_{i=0}^N \mu_p(x_i) \cdot \mu_C(y)$$

Isso significa que se multiplica a pertinência do predicado e a pertinência consequente para obter uma eficácia final. Como exemplo, considere a regra:

se x_1 é W e x_2 é Y e y é Z ,

onde W, Y , e Z são conjuntos fuzzy fora do conjunto de termos da variável, a eficácia é computada como indicado na equação abaixo

$$Eff(rule) = \mu_W(x_1) \cdot \mu_Y(x_2) \cdot \mu_Z(y)$$

O grau de eficácia é um peso para a resolução de conflitos, atuando como um filtro que permite apenas às regras com as maiores eficácias participarem da descoberta de regras. Usar uma medida de eficácia explora não apenas a verdade máxima dos vários componentes das regras mas também a medida de utilidade geral das regras. Em vez de uma indicação de peso explícita, pode-se reprocessar o conjunto de treinamento para encontrar as regras que exibem o melhor comportamento sob às restrições impostas ao conjunto de treinamento. Na maior parte dos casos reais, os dados de treinamento produzem várias regras com o mesmo predicado e consequente. Essas regras disputam pela participação no modelo. Uma

forma simples de discriminar entre várias regras é determinar um grau de classificação para cada regra e selecionar a regra com a mais alta classificação. Essa é a finalidade da métrica da eficácia.

Determinando e Usando o Grau de Contribuição

Em alguns exemplos poderia-se querer ajustar também a classificação da produção de regras usando o conhecimento dos dados. Baseados em precisões de medidas, crença na validade dos dados, ou alguma outra métrica, um índice de credibilidade (Bel) pode também ser associado aos pares de treinamento. Uma vez que a credibilidade é determinada em uma base registro por registro, ela é também conhecida por Unidade de Peso da Confiabilidade – especificando a confiabilidade geral que se tem nos dados. Para dados altamente confiáveis esse fator se move em direção ao valor 1 e para dados altamente suspeitos esse fator se move em direção a 0. Esse índice de credibilidade é combinado com o grau de eficácia da regra para gerar um Grau de Contribuição (Doc) geral. A equação de filtragem das regras, mostrada na equação anterior, se torna:

$$Doc(rule) = Eff(rule).Bel(rule)$$

Esse processo incorpora agora fatores que influenciam os dados com a experiência do especialista, resultando em um sistema que é suscetível aos diferentes graus de confiabilidade dos dados. No mundo real da composição de modelos, os dados têm variados graus de ruído, aplicação e importância. Os pesos de contribuição se ajustam às regras baseados em quanta confiança se tem nos dados tomados como base. Pode-se notar que é possível remover a subjetividade humana e o relacionamento de utilidade dos dados simplesmente fazendo com que todos os valores de $Bel(rule_i)$ sejam 1. Se o índice de credibilidade não é especificado com um registro de comportamento, um valor de 1 é sempre assumido.

Avaliando o Arquivo de Comportamento dos Dados

O recurso de geração de regra primária (WMgenerateRules) lê o arquivo de comportamento para construir uma matriz de regras potenciais. Essa informação de avaliação, em uma base registro a registro, é armazenada no arquivo de auditoria de análise do melhor encaixe (wmaud02.fil). A figura III.31 mostra o conteúdo desse arquivo:

```
RD---[002]: Rule Discovery. Initial Rule Generation
  Data File : sales.dat
  Rule File : a:WMrules.fil
  Audit File : a:WMaud02.fil

1. //time price sales
2. //-----
3. 1, 11.28, 40.94

RD---[003]: Rule Generation Statistics from Data:
  'time' ( 1.00) has max m[0.9761] in termset 'TIM_S004'
  'price' ( 11.28) has max m[0.9922] in termset 'PRC_S004'
  'sales' ( 40.94) has max m[1.0000] in termset 'SAL_S008'
  Unit ConfWeight: 1.0000, RuleDegree: 0.9684
4. 2, 17.45, 336.82

RD---[003]: Rule Generation Statistics from Data:
  'time' ( 2.00) has max m[0.9043] in termset 'TIM_S004'
  'price' ( 17.45) has max m[0.9297] in termset 'PRC_S004'
  'sales' ( 336.82) has max m[1.0000] in termset 'SAL_S008'
  Unit ConfWeight: 1.0000, RuleDegree: 0.8407
5. 3, 28.75, 574.15
```

Figura III.31. Parte da análise registro-a-registro a partir dos dados de vendas

A análise para cada linha calcula o Grau de Contribuição (RuleDegree). Esse é o produto do Grau de Eficácia e a Unidade do Índice de Credibilidade ou Unidade do Peso da Confiabilidade (Unidade ConfWeight). Se o fator de Unidade ConfWeight não é especificado pelo registro, um valor de 1.0 é assumido e o Grau de Contribuição é simplesmente o Grau de Eficácia. Para encontrar a eficácia da regra, o conjunto fuzzy com o máximo grau de pertinência para o valor do dado novo é isolado. O produto desses valores é a eficácia da regra.

Entendendo a Matriz de Regras Intermediária

Quando o arquivo de comportamento dos dados foi completamente processado, uma matriz de regras intermediária é criada e salva (no arquivo `wmrules.fil`). A figura III.32 mostra o conteúdo parcial da matriz de regras para o modelo de sensibilidade de preços.

```
time, TIM_S004, 0.9761, price, PRC_S004, 0.9922, sales, SAL_S008, 1.0000, 0.9684
time, TIM_S004, 0.9043, price, PRC_S004, 0.9297, sales, SAL_S008, 1.0000, 0.8407
time, TIM_S004, 0.7847, price, PRC_S004, 0.6440, sales, SAL_S008, 0.9992, 0.5049
time, TIM_S004, 0.6172, price, PRC_S004, 0.5000, sales, SAL_S008, 0.9925, 0.3063
time, TIM_S004, 0.4106, price, PRC_S004, 0.2153, sales, SAL_S008, 0.9791, 0.0866
time, TIM_S004, 0.2363, price, PRC_S003, 0.6440, sales, SAL_S008, 0.9165, 0.1395
time, TIM_S003, 0.1763, price, PRC_S003, 0.9824, sales, SAL_S008, 0.7295, 0.1263
time, TIM_S003, 0.7188, price, PRC_S003, 0.9604, sales, SAL_S008, 0.2996, 0.2068
time, TIM_S003, 1.0000, price, PRC_S002, 0.6948, sales, SAL_S007, 0.7565, 0.5256
time, TIM_S003, 0.7847, price, PRC_S002, 0.9688, sales, SAL_S006, 0.9908, 0.7532
time, TIM_S002, 0.7637, price, PRC_S001, 0.5894, sales, SAL_S005, 0.5308, 0.2389
time, TIM_S002, 0.9995, price, PRC_S001, 0.9922, sales, SAL_S003, 0.5894, 0.5845
time, TIM_S002, 0.8047, price, PRC_S001, 0.6948, sales, SAL_S003, 0.8476, 0.4739
```

Figura III.32. Uma parte da matriz intermediária

Cada linha na matriz consiste de vários grupos de três tuplas, uma para cada predicado e outra para cada conseqüente. As três tuplas contêm o nome da variável, o nome do conjunto fuzzy que tem o valor máximo para o elementos dos dados associados com essa variável, e o grau de pertinência para cada elemento dos dados. A última entrada em cada linha é o Grau de Contribuição calculado para cada regra. Essa matriz é o conjunto de regras produzido a partir do arquivo de comportamento. A próxima fase do processo de descoberta, formando as regras fuzzy reais, é gerada a partir da matriz intermediária de regras.

III.6.8.3. Passo Três - Cria as Regras Fuzzy Combinadas

Como se pode ver na Figura III.32, a matriz de regras intermediária contém várias regras redundantes e conflitantes. Essa é uma consequência natural do processo de aprendizado de regras associado com os dados de comportamento. A fase

final da descoberta de regras envolve o uso do Grau de Contribuição para selecionar aquelas regras que melhor descrevem o desempenho do sistema. Embora os valores de pertinência máximos sejam carregados com as regras, eles são mantidos para os propósitos de validação e análise, mas não participam da compressão de regras.

Processo de Compressão de Regras

A combinação de regras envolve a inserção de regras temporárias na memória fuzzy associativa MxN usando os predicados dos conjuntos fuzzy. Para qualquer combinação dos predicados dos conjuntos fuzzy (tais como aqueles para TEMPO e PREÇO) existe um conjunto de soluções fuzzy (tal como aquele para a variável dependente VENDAS). A memória associativa fuzzy retangular é usada para consolidar as regras intermediárias, mas o mecanismo de descoberta escreve produções convencionais if-then. A figura III.33 mostra o conteúdo do arquivo de auditoria da compressão de regras (wmaud03.fil) para o modelo de vendas.

RD---[003]: Rule Discovery. Combined FAM Generation

Audit File : a:WMaud03.fil

Rule File : a:WMrules.FIL

FAM File : a:WMfam.fil

Record: 1. Rule ADDED: TIM_S004xPRC_S004: SAL_S008 (0.9684)
Record: 2. Rule REJECTED: TIM_S004xPRC_S004: SAL_S008 (0.8407 versus 0.9684)
Record: 3. Rule REJECTED: TIM_S004xPRC_S004: SAL_S008 (0.5049 versus 0.9684)
Record: 4. Rule REJECTED: TIM_S004xPRC_S004: SAL_S008 (0.3063 versus 0.9684)
Record: 5. Rule REJECTED: TIM_S004xPRC_S004: SAL_S008 (0.0866 versus 0.9684)
Record: 6. Rule ADDED: TIM_S004xPRC_S003: SAL_S008 (0.1395)
Record: 7. Rule ADDED: TIM_S003xPRC_S003: SAL_S008 (0.1263)
Record: 8. Rule UPDATES: TIM_S003xPRC_S003: SAL_S008 (0.2068)
Record: 9. Rule ADDED: TIM_S003xPRC_S002: SAL_S007 (0.5256)
Record: 10. Rule UPDATES: TIM_S003xPRC_S002: SAL_S006 (0.7532)
Record: 11. Rule ADDED: TIM_S002xPRC_S001: SAL_S005 (0.2389)
Record: 12. Rule UPDATES: TIM_S002xPRC_S001: SAL_S003 (0.5845)
Record: 13. Rule REJECTED: TIM_S002xPRC_S001: SAL_S003 (0.4739 versus 0.5845)
Record: 14. Rule ADDED: TIM_S001xPRC_CE: SAL_S004 (0.6383)
Record: 15. Rule UPDATES: TIM_S001xPRC_CE: SAL_S005 (0.6770)
Record: 16. Rule ADDED: TIM_S001xPRC_B001: SAL_S007 (0.5133)
Record: 17. Rule ADDED: TIM_CExPRC_B001: SAL_S008 (0.2380)
Record: 18. Rule ADDED: TIM_CExPRC_B002: SAL_S008 (0.5603)
Record: 19. Rule UPDATES: TIM_CExPRC_B002: SAL_S008 (0.6124)
Record: 20. Rule ADDED: TIM_B001xPRC_B002: SAL_S008 (0.6270)

Figura III.33. Parte do relatório de auditoria a partir da geração de regras fuzzy

Nesse arquivo de auditoria, a ação de compressão é mostrada para cada regra. As regras temporárias são mostradas na forma retangular ou algébrica, assim, interpreta-se a especificação da regra:

TIM_B002xPRC_B003: SAL_S008

como a regra: se tempo é TIM_B002 e preço é PRC_B003 então vendas são SAL_S008. A forma retangular mostra os eixos vertical e horizontal da memória fuzzy associativa com o conjunto de solução fuzzy ocupando uma célula na interseção da linha e da coluna.

Existem várias ações que a lógica de compressão de regras pode invocar quando se encontra uma nova regra da matriz intermediária de regras. A regra pode ser ADICIONADA, significando que ela é uma nova regra (a interseção dos conjuntos fuzzy TIME e PRICE está vazia). Uma nova regra também ATUALIZA uma célula existente se o Grau de Contribuição é maior que o peso de contribuição da regra que já está nessa célula. E uma nova regra é também REJEITADA se o seu Grau de Contribuição é menor que o peso da contribuição da regra que já está na célula (a análise mostra também a razão para essa rejeição, mostrando os dois pesos de confiabilidade). Quando a matriz de regras completa foi processada, o relatório estatístico indica o número de regras combinadas geradas e o percentual que isso representa do conjunto de regras inicial.

Compondo a Memória Associativa Fuzzy

A figura III.34 mostra a Memória Associativa Fuzzy para o modelo de sensibilidade de vendas. Os conjuntos fuzzy de TIME se situam ao longo do eixo vertical e os conjuntos fuzzy de PRICE se situam ao longo do eixo horizontal. Na interseção de TIME e PRICE estão os conjuntos fuzzy SALES correspondentes.

	PRC_S004	PRC_S003	PRC_S002	PRC_S001	PRC_CE	PRC_B001	PRC_B002	PRC_B003	PRC_B004
TIM_S004I	SAL_S008	SAL_S008							
TIM_S003I		SAL_S008	SAL_S002						
TIM_S002I			SAL_S007	SAL_S003					
TIM_S001I			SAL_S007	SAL_B002	SAL_S001	SAL_S007			
TIM_CE I				SAL_S006	SAL_S008	SAL_S007	SAL_S008		
TIM_B001I						SAL_S008	SAL_S008	SAL_S008	
TIM_B002I							SAL_S008	SAL_S008	SAL_S008
TIM_B003I								SAL_S008	SAL_S008
TIM_B004I									SAL_S008

Figura III.34. A Memória Associativa Fuzzy final para o modelo de vendas

A Memória Fuzzy Associativa (FAM) é esparsa para esse problema. Isso acontece quando pequenas quantidades de dados são usadas para treinar o sistema ou, como no caso do modelo de vendas, usa-se conjuntos repetidos de 36 períodos de tempo, cada um situado ao longo do eixo de tempo de um até trinta e seis. Grandes espectros de dados descrevendo mais do comportamento do sistema levam a uma melhor saturação do espaço de regras e fornece uma aproximação mais próxima do sistema atual.

Gerando as Regras Fuzzy Combinadas

O processo de descoberta de regras termina quando as regras finais são geradas a partir da Memória Fuzzy Associativa. Elas são armazenadas no arquivo wmfam.fil. Cada regra fuzzy é da forma *if-then* e é prefaciada pelo número da regra entre colchetes. A figura III.35 mostra as regras produzidas para o modelo de vendas a partir da memória fuzzy associativa já consolidada.

[R001]: if time is TIM_S004 and price is PRC_S004 then sales is SAL_S008;
[R002]: if time is TIM_S004 and price is PRC_S003 then sales is SAL_S008;
[R003]: if time is TIM_S003 and price is PRC_S003 then sales is SAL_S008;
[R004]: if time is TIM_S003 and price is PRC_S002 then sales is SAL_S002;
[R005]: if time is TIM_S002 and price is PRC_S002 then sales is SAL_S007;
[R006]: if time is TIM_S002 and price is PRC_S001 then sales is SAL_S003;
[R007]: if time is TIM_S001 and price is PRC_S002 then sales is SAL_S007;
[R008]: if time is TIM_S001 and price is PRC_S001 then sales is SAL_B002;
[R009]: if time is TIM_S001 and price is PRC_CE then sales is SAL_S001;
[R010]: if time is TIM_S001 and price is PRC_B001 then sales is SAL_S007;
[R011]: if time is TIM_CE and price is PRC_S001 then sales is SAL_S006;
[R012]: if time is TIM_CE and price is PRC_CE then sales is SAL_S008;
[R013]: if time is TIM_CE and price is PRC_B001 then sales is SAL_S007;
[R014]: if time is TIM_CE and price is PRC_B002 then sales is SAL_S008;
[R015]: if time is TIM_B001 and price is PRC_B001 then sales is SAL_S008;
[R016]: if time is TIM_B001 and price is PRC_B002 then sales is SAL_S008;
[R017]: if time is TIM_B001 and price is PRC_B003 then sales is SAL_S008;
[R018]: if time is TIM_B002 and price is PRC_B002 then sales is SAL_S008;
[R019]: if time is TIM_B002 and price is PRC_B003 then sales is SAL_S008;
[R020]: if time is TIM_B002 and price is PRC_B004 then sales is SAL_S008;
[R021]: if time is TIM_B003 and price is PRC_B003 then sales is SAL_S008;
[R022]: if time is TIM_B003 and price is PRC_B004 then sales is SAL_S008;
[R023]: if time is TIM_B004 and price is PRC_B004 then sales is SAL_S008;

Figura III.35. As regras fuzzy finais produzidas pelo modelo de vendas

Essas regras representam o melhor encaixe entre os dados de comportamento em estado natural e suas funções explícitas ou relações das regras tomadas coletivamente. Um importante aspecto do processo de descoberta de regras é a habilidade de combinar regras geradas pelos computadores com as geradas pelos especialistas. Essa combinação pode vir de um dos dois pontos: através da inserção de regras potenciais no arquivo de regras intermediário (com os correspondentes graus de contribuição), ou pela inserção de regras completas no conjunto de regras consolidadas mostrado na Figura III.35. Dessa forma, pode-se usar o método de descoberta de regras para melhorar o processo de aquisição do conhecimento, mantendo o processo mecânico próximo ao conhecimento dos especialistas.

III.7. Métodos Adotados:

Os Métodos adotados para os estudos de caso são o CN2, o Método de Descoberta de Regras de Wang-Mendel, e alguns métodos fornecidos pelo Tooldiag.

IV. Estudo de Casos:

IV.1. Introdução:

Como ilustração de algumas das técnicas mostradas anteriormente, estudou-se dois bancos de dados, um deles considerado um clássico na área de reconhecimento de padrão, que é o Conjunto de Dados das Flores Íris de Fisher; e o outro é um conjunto de dados climáticos fornecidos pelo Aeroporto Internacional do Galeão. Os dados das flores íris já trazem várias informações associadas e está preparado para ser utilizado, sem a necessidade de pré-processamento. É interessante observar que essa não é uma situação comum em problemas práticos. Ao contrário, o banco de dados climáticos representa melhor os problemas reais enfrentados no processo de extração de conhecimento.

IV.2. Conjuntos de dados das Flores íris de Fisher

Esse é talvez o melhor banco de dados conhecido que se pode encontrar na literatura de reconhecimento de padrão. O paper de Fisher é um clássico no campo e é frequentemente referido até hoje. O conjunto de dados contém 3 classes de 50 exemplos cada, onde cada classe refere-se a um tipo de íris. A primeira classe é linearmente separável das outras duas; a última não é linearmente separável de cada uma das outras. Atributo prognosticado: Classe da planta.

Esse é um domínio bastante simples e consiste de 150 exemplos divididos igualmente pelas três classes (50 em cada uma das três classes). Cada exemplo é constituído de quatro atributos numéricos e a classe. Os atributos numéricos são mostrados a seguir:

1º . comprimento da sépala em cm

2º . largura da sépala em cm

3º . comprimento da pétala em cm

4º . largura da pétala em cm

As denominações de classe utilizadas são: Íris Setosa, Íris Versicolor, e Íris Virginica e as amostras se encontram distribuídas 33% para cada classe.

Sumário das Estatísticas:

	Min	Max	Meio	SD	Correlação das Classes
comprimento da sépala	: 4.3	7.9	5.84	0.83	0.7826
largura da sépala	: 2.0	4.4	3.05	0.43	-0.4194
comprimento da pétala	: 1.0	6.9	3.76	1.76	0.9490 (alto!)
largura da pétala	: 0.1	2.5	1.20	0.76	0.9565 (alto!)

V.2.1. Resultados Comentados:

V.2.1.1. No Tooldiag:

Quando se utiliza o arquivo de dados das flores íris, percebe-se que para alguns critérios de seleção de características como o mostrado na figura IV.1, os atributos *petal_length* e *petal_width* são selecionados como as duas melhores características . Para esse grupo de critério de seleção, o *Estimated minimal error probability*, os valores contidos nos resultados que aparecem à esquerda das características se confundem com valores relativos à exatidão de classificação. Nesse caso, os dois primeiros atributos apresentam uma ótima exatidão para a classificação e bem superior às dos outros atributos e por isso é correto dizer que nesse e em casos similares tais atributos são suficientes para a classificação dos dados. No caso mostrado na figura IV.1, a estratégia de procura é a *Best Features*.

```

iris3c-111.fea
#
#----- FEATURE SELECTION -----
# Search strategy: Best Features
# Selection criterion group: Estimated minimal error probability
# - Classifier: 1-Nearest Neighbor, using raw data
# - Cross validation: Leave-One-Out
#-----
iris.UCI
4
# Features Selection criterion (-1 = EMPTY)
3 0.880000 petal_length
4 0.880000 petal_width
1 0.586667 sepal_length
2 0.480000 sepal_width

```

Figura IV.1. Arquivo de seleção de características iris3c-111.fea

Acontece, porém, que isso não é verdade para todas as situações dos diferentes processos de seleção de características apresentados pelo Tooldiag e na sequência, utilizando-se a mesma estratégia de procura, mas com o grupo de seleção de critério *Inter-class distance*, todas as métricas de distâncias apresentam a mesma ordem de seleção características, diferente da ordem apresentada anteriormente. Uma delas é mostrada abaixo na figura IV.2.

```

iris3c-2121.fea
#
#----- FEATURE SELECTION -----
# Search strategy: Best Features
# Selection criterion group: Inter-class distance
# Distance metric:Minkowski of order s=2.000000
# Calculating mutual distance between ALL classes
#-----
iris.UCI
4
# Features Selection criterion (-1 = EMPTY)
3 0.982205 petal_length
1 0.469942 sepal_length
4 0.431503 petal_width
2 0.240235 sepal_width

```

Figura IV.2. Arquivo de seleção de características iris3c-2121.fea

A próxima sequência ainda utiliza o método *Best Features*, agora, entretanto, utilizando o grupo de critério de seleção *Probabilistic Distance*, apresentando sete opções diferentes de métricas. Para essas métricas, as duas primeiras características selecionadas são sempre as mesmas, porém, para quatro dessas métricas, o primeiro

atributo selecionado é o *petal_length* como mostra a figura IV.3a e para as outras três, é o *petal_width* como mostra a figura IV.3b.

```
iris3c-2131.fea
#
#----- FEATURE SELECTION -----
#      Search strategy:           Best Features
#      Selection criterion group:  Probabilistic distance
#      Distance metrik:Chernoff with s = 0.500
#-----
iris.UCI
4
# Features      Selection criterion (-1 = EMPTY)
3      2.396339      petal_length
4      1.810675      petal_width
1      0.224921      sepal_length
2      0.077387      sepal_width
```

Figura IV.3a. Arquivo de seleção de características iris3c-2131.fea

```
iris3c-2137.fea
#
#----- FEATURE SELECTION -----
#      Search strategy:           Best Features
#      Selection criterion group:  Probabilistic distance
#      Distance metrik:Normalized distance between means
#                               = 1 - (s1+s2)^2 / (m1-m2)^2
#-----
iris.UCI
4
# Features      Selection criterion (-1 = EMPTY)
4      0.268800      petal_width
3      0.254453      petal_length
1      -0.153877      sepal_length
2      -1.154346      sepal_width
```

Figura IV.3b. Arquivo de seleção de características iris3c-2137.fea

Para a estratégia de procura *Sequential Forward Search* (Procura Sequencial a Frente) com o grupo de critério de seleção *Estimated minimal error probability* e validação cruzada do tipo *leave-one-out* as características selecionadas foram *petal_length* e *petal_width*, porém a segunda aparece com a mesma pontuação que a característica *sepal_length* como mostra a figura IV.4.

```

iris3c-221.fea
#
#----- FEATURE SELECTION -----
# Search strategy: Sequential Forward Search
# Selection criterion group: Estimated minimal error probability
# - Classifier: 1-Nearest Neighbor, using raw data
# - Cross validation: Leave-One-Out
#-----
iris.UCI
4
# Features Selection criterion (-1 = EMPTY)
3 0.880000 petal_length
4 0.953333 petal_width
1 0.953333 sepal_length
2 0.960000 sepal_width

```

Figura IV.4. Arquivo de seleção de características iris3c-221.fea

Já para a sequência da estratégia *Sequential Forward Search*, porém com grupo de critério de seleção *Interclass Distance*, para as cinco métricas apresentadas, os atributos selecionados foram *petal_length* e *sepal_length* como mostra uma das saídas apresentadas na figura IV.5.

```

iris3c-2223.fea
#
#----- FEATURE SELECTION -----
# Search strategy: Sequential Forward Search
# Selection criterion group: Inter-class distance
# Distance metric: EUCLIDEAN DISTANCE
# Calculating mutual distance between ALL classes
#-----
iris.UCI
4
# Features Selection criterion (-1 = EMPTY)
3 0.982205 petal_length
1 1.126588 sepal_length
4 1.217152 petal_width
2 1.263404 sepal_width

```

Figura IV.5. Arquivo de seleção de características iris3c-2223.fea

Na sequência da estratégia de procura ‘Sequential Forward Search’ com grupo de critério de seleção ‘Probabilist Distance’ (distância probabilística) tem-se um resultado diferente quase que para cada métrica. Para as duas primeiras Chernoff e Bhattacharyya, a seleção é idêntica inclusive com os mesmos valores. A figura IV.6 mostra o resultado para as duas distâncias mencionadas acima.

```

iris3c-2231.fea
#
#----- FEATURE SELECTION -----
# Search strategy: Sequential Forward Search
# Selection criterion group: Probabilistic distance
# Distance metrik:Chernoff with s = 0.500
#-----
iris.UCI
4
# Features Selection criterion (-1 = EMPTY)
3 2.396339 petal_length
1 3.302497 sepal_length
4 3.957121 petal_width
2 4.484632 sepal_width

```

```

iris3c-2232.fea
#
#----- FEATURE SELECTION -----
# Search strategy: Sequential Forward Search
# Selection criterion group: Probabilistic distance
# Distance metrik:Bhattacharyya distance
#-----
iris.UCI
4
# Features Selection criterion (-1 = EMPTY)
3 2.396339 petal_length
1 3.302497 sepal_length
4 3.957121 petal_width
2 4.484632 sepal_width

```

Figura IV.6. Arquivo de seleção de características iris3c-2231.fea e iris2232.fea

Para a distância métrica Matusita, o resultado varia bastante na ordem e os valores para todas as características se encontram muito próximos uns dos outros, como mostra a figura IV.7.

```

iris3c-2233.fea
#
#----- FEATURE SELECTION -----
# Search strategy: Sequential Forward Search
# Selection criterion group: Probabilistic distance
# Distance metrik:Matusita distance
#-----
iris.UCI
4
# Features Selection criterion (-1 = EMPTY)
4 0.442258 petal_width
3 0.451043 petal_length
2 0.457616 sepal_width
1 0.459969 sepal_length

```

Figura IV.7. Arquivo de seleção de características iris3c-2233.fea

Para a métrica *Divergence*, a ordem de seleção de torna a se alterar significativamente. Dessa vez, tem-se novamente um distanciamento dos valores para cada uma das características e a ordem da primeira e segunda características se invertem em relação à anterior como mostra a figura IV.8.

```
iris-2234.fea
#
#----- FEATURE SELECTION -----
#      Search strategy:           Sequential Forward Search
#      Selection criterion group:  Probabilistic distance
#      Distance metrik:Divergence
#-----
iris.UCI
4
# Features      Selection criterion (-1 = EMPTY)
3      51.770252      petal_length
4      60.917572      petal_width
2      73.794250      sepal_width
1      77.032867      sepal_length
```

Figura IV.8. Arquivo de seleção de características iris3c-2134.fea

Para a distância Mahalanobis, a seleção volta a se apresentar disposta como as primeiras dessa sequência, porém os valores para essa métrica são um pouco diferentes como mostra a figura IV.9.

```
iris3c-2235.fea
#
#----- FEATURE SELECTION -----
#      Search strategy:           Sequential Forward Search
#      Selection criterion group:  Probabilistic distance
#      Distance metrik:Mahalanobis distance
#-----
iris.UCI
4
# Features      Selection criterion (-1 = EMPTY)
3      18.725349      petal_length
1      25.949768      sepal_length
4      30.839724      petal_width
2      34.887352      sepal_width
```

Figura IV.9. Arquivo de seleção de características iris3c-2235.fea

Finalizando essa sequência, com o emprego da distância de Patrick-Fisher, a ordem é modificada novamente como é mostrado na figura IV.10

```

iris3c-2236.fea
#
#----- FEATURE SELECTION -----
# Search strategy: Sequential Forward Search
# Selection criterion group: Probabilistic distance
# Distance metrik:Patrick-Fisher
#-----
iris.UCI
4
# Features Selection criterion (-1 = EMPTY)
4 1.099114 petal_width
3 1.416709 petal_length
2 1.271819 sepal_width
1 1.393285 sepal_length

```

Figura IV.11. Arquivo de seleção de características iris3c-2236.fea

Terminada a primeira abordagem ao módulo de seleção de características do Tooldiag, escolhe-se um conjunto de características selecionadas, para a demonstração do módulo de estimativa de erro. Optou-se pelo primeiro conjunto de características selecionado pela estratégia de procura *Best Features* e Grupo de Critério de Seleção *Estimated minimal error probability*, que é lembrado abaixo, para efeito didático, na figura IV.12.

```

iris3c-111.fea
#
#----- FEATURE SELECTION -----
# Search strategy: Best Features
# Selection criterion group: Estimated minimal error probability
# - Classifier: 1-Nearest Neighbor, using raw data
# - Cross validation: Leave-One-Out
#-----
iris.UCI
4
# Features Selection criterion (-1 = EMPTY)
3 0.880000 petal_length
4 0.880000 petal_width
1 0.586667 sepal_length
2 0.480000 sepal_width

```

Figura IV.12. Arquivo de seleção de características iris3c-111.fea.

O módulo de estimativa de erro é executado sobre as características selecionadas no módulo de seleção de características. Além disso, é necessária a escolha do modelo de classificador. Começa-se com as características selecionadas acima, o modelo de classificador *1-Nearest Neighbor* e finalmente a seleção do

primeiro método de estimativa de erro *Resubstitution*. Para essa combinação, o valor do erro começa em 10,67% com a introdução da primeira característica, diminui com a introdução da segunda, e atinge 0% com a introdução da terceira característica, permanecendo em 0% após a introdução da quarta característica, como mostra a figura IV.13. Dependendo do limite escolhido para o nível de erro, pode-se descartar a utilização de uma ou mais características dentre as características selecionadas. Para esse exemplo, se 90% de exatidão de classificação fosse considerado suficiente para o processo, poderia-se descartar as duas últimas características. Em qualquer caso, entretanto, a utilização da quarta característica é desnecessária, já que adiciona processamento e não existe a possibilidade de melhorar a exatidão de classificação.

```
iris11-111.err
### ERROR ESTIMATION PROTOCOL ###
Universe: iris.UCI
----- CROSS VALIDATION -----
      NAME:          Resubstitution
              with increasing number of features
      CLASSIFIER MODEL: 1-Nearest Neighbor, using raw data
-----
Estimation for increasing number of features!

=====
=== Number of selected features: 1 ===
Nr. Features: 1 - Estimated error:      10.67 % - Accuracy: 89.33 %
=====

=====
=== Number of selected features: 2 ===
Nr. Features: 2 - Estimated error:       1.33 % - Accuracy: 98.67 %
=====

=====
=== Number of selected features: 3 ===
Nr. Features: 3 - Estimated error:       0.00 % - Accuracy: 100.00 %
=====

=====
=== Number of selected features: 4 ===
Nr. Features: 4 - Estimated error:       0.00 % - Accuracy: 100.00 %
=====
```

Figura IV.13. Arquivo de estimativa de erro para o arquivo de seleção de características iris3c-111.fea: o iris11-111.err

Mantendo-se o classificador e os atributos já selecionados, e variando-se o método de estimativa de erro para *Holdout*, percebe-se que o valor da estimativa de erro vem diminuindo até a introdução da terceira característica. Após a introdução da

quarta característica, porém, o valor da estimativa de erro aumenta sutilmente, como mostra a figura IV.14. Isso demonstra que a introdução da última característica, interferiu negativamente para a exatidão da classificação, além de adicionar tempo de processamento à execução desse módulo.

```
iris12-111.err
### ERROR ESTIMATION PROTOCOL ###
Universe: iris.UCI
----- CROSS VALIDATION -----
NAME: Holdout
      splitting into 70.00% training data and 30.00% test data
      taking the mean over 5 runs
      with increasing number of features
CLASSIFIER MODEL: 1-Nearest Neighbor, using raw data
-----
Estimation for increasing number of features!

=====
=== Number of selected features: 1 ===
--- AVERAGE STATISTICS OF ERROR ESTIMATION ---
Mean over 5 runs has been taken

STATISTICAL PARAMETERS OF THE ESTIMATED ERROR:
MEAN=11.1111% STANDARD DEVIATION=0.04216370 VARIANCE=0.00177778
MINIMUM=4.444% MAXIMUM=15.556%
Nr. Features: 1 - Estimated error: 11.11 % - Accuracy: 88.89 %
=====

=====
=== Number of selected features: 2 ===
--- AVERAGE STATISTICS OF ERROR ESTIMATION ---
Mean over 5 runs has been taken

STATISTICAL PARAMETERS OF THE ESTIMATED ERROR:
MEAN=5.3333% STANDARD DEVIATION=0.02666667 VARIANCE=0.00071111
MINIMUM=2.222% MAXIMUM=8.889%
Nr. Features: 2 - Estimated error: 5.33 % - Accuracy: 94.67 %
=====

=====
=== Number of selected features: 3 ===
--- AVERAGE STATISTICS OF ERROR ESTIMATION ---
Mean over 5 runs has been taken

STATISTICAL PARAMETERS OF THE ESTIMATED ERROR:
MEAN=2.6667% STANDARD DEVIATION=0.02591534 VARIANCE=0.00067160
MINIMUM=0.000% MAXIMUM=6.667%
Nr. Features: 3 - Estimated error: 2.67 % - Accuracy: 97.33 %
=====

=====
=== Number of selected features: 4 ===
--- AVERAGE STATISTICS OF ERROR ESTIMATION ---
Mean over 5 runs has been taken

STATISTICAL PARAMETERS OF THE ESTIMATED ERROR:
MEAN=4.8889% STANDARD DEVIATION=0.03265986 VARIANCE=0.00106667
MINIMUM=2.222% MAXIMUM=11.111%
Nr. Features: 4 - Estimated error: 4.89 % - Accuracy: 95.11 %
=====
```

Figura IV.14. Arquivo de estimativa de erro iris12-111.err

Ainda para o mesmo modelo de classificador, variando-se mais uma vez o método de estimativa de erro, dessa vez para *Leave-one-out*. O erro estimado começa em 12% com a introdução da primeira característica, diminuindo para 4.67% com a introdução da segunda característica, mantendo-se a mesma com a introdução da terceira característica, alterando-se levemente para 4% com a introdução da última característica. Vale observar que a introdução das duas últimas características praticamente não teve efeito sobre o valor do erro estimado. Se o cálculo do erro tivesse sido realizado sobre uma seleção composta apenas de duas características, a diferença final do valor do erro teria sido de 0.89%, o que para muitas aplicações no mundo real não justificaria a utilização das duas últimas características para classificação. A figura IV.15 mostra o resumo dos resultados para o método *leave-one-out*.

```

iris13-111.err
### ERROR ESTIMATION PROTOCOL ###
Universe: iris.UCI
----- CROSS VALIDATION -----
      NAME:                Leave-One-Out
                with increasing number of features
      CLASSIFIER MODEL:    1-Nearest Neighbor, using raw data
-----
Estimation for increasing number of features!

=====
=== Number of selected features:  1 ===
--- AVERAGE STATISTICS OF ERROR ESTIMATION ---
Mean over 150 runs has been taken

      STATISTICAL PARAMETERS OF THE ESTIMATED ERROR:
      MEAN=12.0000% STANDARD DEVIATION=0.32496154      VARIANCE=0.10560000
      MINIMUM=0.000%      MAXIMUM=100.000%
Nr. Features:  1 - Estimated error:  12.00  % - Accuracy:  88.00  %
=====

=====
=== Number of selected features:  2 ===
--- AVERAGE STATISTICS OF ERROR ESTIMATION ---
Mean over 150 runs has been taken

      STATISTICAL PARAMETERS OF THE ESTIMATED ERROR:
      MEAN=4.6667% STANDARD DEVIATION=0.21092390      VARIANCE=0.04448889
      MINIMUM=0.000%      MAXIMUM=100.000%
Nr. Features:  2 - Estimated error:  4.67  % - Accuracy:  95.33  %
=====

=====
=== Number of selected features:  3 ===
--- AVERAGE STATISTICS OF ERROR ESTIMATION ---
Mean over 150 runs has been taken

      STATISTICAL PARAMETERS OF THE ESTIMATED ERROR:
      MEAN=4.6667% STANDARD DEVIATION=0.21092390      VARIANCE=0.04448889
      MINIMUM=0.000%      MAXIMUM=100.000%
Nr. Features:  3 - Estimated error:  4.67  % - Accuracy:  95.33  %
=====

=====
=== Number of selected features:  4 ===
--- AVERAGE STATISTICS OF ERROR ESTIMATION ---
Mean over 150 runs has been taken

      STATISTICAL PARAMETERS OF THE ESTIMATED ERROR:
      MEAN=4.0000% STANDARD DEVIATION=0.19595918      VARIANCE=0.03840000
      MINIMUM=0.000%      MAXIMUM=100.000%
Nr. Features:  4 - Estimated error:  4.00  % - Accuracy:  96.00  %
=====

```

Figura IV.15. Arquivo de estimativa de erro iris13-111.err

Faz-se uma nova escolha de método de estimativa de erro, o *Rotation*. Com a introdução da primeira característica o valor é 10.67%. Introduzindo-se a segunda característica, esse valor passa a ser de 4%, aumentando sutilmente para 4.67% com a introdução da terceira característica e mantendo-se a mesma, após a introdução da

quarta característica. Nesse método, fica ainda mais claro que as duas últimas características não são necessárias ao processo de classificação. A figura IV.16 mostra o resumo dos resultados para a combinação selecionada.

```
iris14-111.err
### ERROR ESTIMATION PROTOCOL ###
Universe: iris.UCI
----- CROSS VALIDATION -----
NAME:           Rotation alias K-fold cross validation
                splitting samples into 3 subsets
                (3-fold cross-validation, Leave-50-Out)
                with increasing number of features
CLASSIFIER MODEL: 1-Nearest Neighbor, using raw data
-----
Estimation for increasing number of features!

=====
=== Number of selected features:  1 ===
--- AVERAGE STATISTICS OF ERROR ESTIMATION ---
Mean over 3 runs has been taken

STATISTICAL PARAMETERS OF THE ESTIMATED ERROR:
MEAN=10.6667% STANDARD DEVIATION=0.06599665   VARIANCE=0.00435556
MINIMUM=6.000%   MAXIMUM=20.000%
Nr. Features:  1 - Estimated error: 10.67  % - Accuracy:  89.33  %
=====

=====
=== Number of selected features:  2 ===
--- AVERAGE STATISTICS OF ERROR ESTIMATION ---
Mean over 3 runs has been taken

STATISTICAL PARAMETERS OF THE ESTIMATED ERROR:
MEAN=4.0000% STANDARD DEVIATION=0.01632993   VARIANCE=0.00026667
MINIMUM=2.000%   MAXIMUM=6.000%
Nr. Features:  2 - Estimated error:  4.00  % - Accuracy:  96.00  %
=====

=====
=== Number of selected features:  3 ===
--- AVERAGE STATISTICS OF ERROR ESTIMATION ---
Mean over 3 runs has been taken

STATISTICAL PARAMETERS OF THE ESTIMATED ERROR:
MEAN=4.6667% STANDARD DEVIATION=0.00942811   VARIANCE=0.00008889
MINIMUM=4.000%   MAXIMUM=6.000%
Nr. Features:  3 - Estimated error:  4.67  % - Accuracy:  95.33  %
=====

=====
=== Number of selected features:  4 ===
--- AVERAGE STATISTICS OF ERROR ESTIMATION ---
Mean over 3 runs has been taken

STATISTICAL PARAMETERS OF THE ESTIMATED ERROR:
MEAN=4.6667% STANDARD DEVIATION=0.03399346   VARIANCE=0.00115556
MINIMUM=0.000%   MAXIMUM=8.000%
Nr. Features:  4 - Estimated error:  4.67  % - Accuracy:  95.33  %
=====
```

Figura IV.16. Arquivo de estimativa de erro iris14-111.err

Escolhendo-se agora o último método de estimativa de erro, para esse modelo de classificador, confirma-se mais uma vez que são necessárias apenas as duas primeiras características uma boa classificação. Mostra-se apenas um resumo desse resultado na figura IV.17 para ilustrar o método *Bootstrap*.

```
iris15-111.err
### ERROR ESTIMATION PROTOCOL ###
Universe: iris.UCI
----- CROSS VALIDATION -----
      NAME:                Bootstrap
              with increasing number of features
      CLASSIFIER MODEL:    1-Nearest Neighbor, using raw data
-----
Estimation for increasing number of features!

=====
=== Number of selected features:  1 ===
--- AVERAGE STATISTICS OF ERROR ESTIMATION ---
Mean over 100 runs has been taken

      STATISTICAL PARAMETERS OF THE ESTIMATED ERROR:
      MEAN=10.5349% STANDARD DEVIATION=0.03914467    VARIANCE=0.00153231
      MINIMUM=3.279%    MAXIMUM=20.690%
Nr. Features:  1 - Estimated error:  10.53  % - Accuracy:  89.47  %
=====

=====
=== Number of selected features:  2 ===
--- AVERAGE STATISTICS OF ERROR ESTIMATION ---
Mean over 100 runs has been taken

      STATISTICAL PARAMETERS OF THE ESTIMATED ERROR:
      MEAN=4.9351%  STANDARD DEVIATION=0.02253044    VARIANCE=0.00050762
      MINIMUM=1.667%    MAXIMUM=10.714%
Nr. Features:  2 - Estimated error:  4.94  % - Accuracy:  95.06  %
=====

=====
=== Number of selected features:  3 ===
--- AVERAGE STATISTICS OF ERROR ESTIMATION ---
Mean over 100 runs has been taken

      STATISTICAL PARAMETERS OF THE ESTIMATED ERROR:
      MEAN=4.3639%  STANDARD DEVIATION=0.01924253    VARIANCE=0.00037027
      MINIMUM=1.613%    MAXIMUM=9.434%
Nr. Features:  3 - Estimated error:  4.36  % - Accuracy:  95.64  %
=====

=====
=== Number of selected features:  4 ===
--- AVERAGE STATISTICS OF ERROR ESTIMATION ---
Mean over 100 runs has been taken

      STATISTICAL PARAMETERS OF THE ESTIMATED ERROR:
      MEAN=4.3156%  STANDARD DEVIATION=0.02154278    VARIANCE=0.00046409
      MINIMUM=0.000%    MAXIMUM=10.000%
Nr. Features:  4 - Estimated error:  4.32  % - Accuracy:  95.68  %
=====
```

Figura IV.17. Arquivo de estimativa de erro iris15-111.err

Todos os métodos de estimativa de erro disponíveis no pacote foram executados para o modelo de classificador *1-Nearest Neighbor*. Passa-se então ao segundo modelo de classificador, o *Quadratic Gaussian Classifier*, mantendo-se ainda as mesmas características selecionadas. Começa-se, então, com o primeiro método de estimativa de erro, o *Resubstitution*. Com a introdução da primeira característica, o erro apresentou-se baixo e igual a 4.67%. Quando se introduz a segunda característica a exatidão de classificação se mantém constante. Quando se introduz a terceira característica, porém, esse erro aumenta para 38% e permanece mesmo após a utilização da quarta característica, como mostra a figura IV.18. Conclui-se, portanto, que para essa combinação de métodos, apenas a primeira característica seria suficiente para uma boa classificação, as demais características adicionam apenas aumentam o tempo de processamento e o erro estimado.

```
iris21-111.err
### ERROR ESTIMATION PROTOCOL ###
Universe: iris.UCI
----- CROSS VALIDATION -----
NAME: Resubstitution
with increasing number of features
CLASSIFIER MODEL: Quadratic Gaussian Classifier
-----
Estimation for increasing number of features!
=====
=== Number of selected features: 1 ===
Nr. Features: 1 - Estimated error: 4.67 % - Accuracy: 95.33 %
=====

=====
=== Number of selected features: 2 ===
Nr. Features: 2 - Estimated error: 4.67 % - Accuracy: 95.33 %
=====

=====
=== Number of selected features: 3 ===
Nr. Features: 3 - Estimated error: 38.00 % - Accuracy: 62.00 %
=====

=====
=== Number of selected features: 4 ===
Nr. Features: 4 - Estimated error: 38.00 % - Accuracy: 62.00 %
=====
```

Figura IV.18. Arquivo de estimativa de erro iris21-111.err

Os outros métodos de estimativa de erro, quando combinados com esse classificador, não acrescentaram maiores informações à análise apresentada até aqui. Resolveu-se, portanto, passar para o terceiro modelo de classificador, o *Radial Basis*

Function Network. Para o primeiro método de estimativa de erro, o *Resubstitution*, houve uma mudança interessante, pois essa combinação só apresenta um erro baixo para o primeiro atributo. Antes da introdução da segunda característica, o erro estimado é de 11.33%, e depois é de 66.7%. Com a introdução da terceira característica, esse erro se torna ainda pior, passando para 86.67%. Embora esse erro tenha diminuído um pouco com a introdução da quarta característica, ainda se manteve alto e igual a 68%, como mostra a figura IV.19. Resumindo, essa combinação de métodos só selecionaria a primeira característica.

```
iris31-111.err
### ERROR ESTIMATION PROTOCOL ###
Universe: iris.UCI
----- CROSS VALIDATION -----
      NAME:                Resubstitution
              with increasing number of features
      CLASSIFIER MODEL:    Radial Basis Function Network
-----
Estimation for increasing number of features!
=====
=== Number of selected features:  1 ===
Nr. Features:  1 - Estimated error:    11.33  % - Accuracy:  88.67  %
=====

=====
=== Number of selected features:  2 ===
Nr. Features:  2 - Estimated error:    66.67  % - Accuracy:  33.33  %
=====

=====
=== Number of selected features:  3 ===
Nr. Features:  3 - Estimated error:    86.67  % - Accuracy:  13.33  %
=====

=====
=== Number of selected features:  4 ===
Nr. Features:  4 - Estimated error:    68.00  % - Accuracy:  32.00  %
=====
```

Figura IV.19. Arquivo de estimativa de erro iris31-111.err

Os demais métodos de estimativa de erro associados a esse e outros classificadores não apresentam grandes variações em termos de resultados, dos demais comentados até esse ponto da análise. Receia-se o excesso de comentários sobre resultados até certo ponto redundantes. Todas as combinações de classificadores e métodos de estimativa de erro possíveis foram aplicados ao banco de dados das flores íris. Não se vê, porém, a necessidade de mostrá-los todos nessa análise, já que o objetivo final da mesma que é comprovar que as características que

foram selecionadas pelo módulo de seleção de características apresentam um baixo erro estimado, foi atingido.

Esse é um estudo interessante que mostra o efeito do Tooldiag num conjunto de dados clássico de Fisher [10]. Um conjunto de 150 amostras com uma dimensão de 4 características que descrevem três classes de flores diferentes, 50 amostras por classe. Os valores particulares das características são mostrados na tabela II. Na tabela III, o resultado para a seleção de características e estimativa de erro são apresentados. É interessante mostrar que todas as três estratégias selecionam quase o mesmo conjunto de características.

TABLE II
IRIS FLOWER DATA SET

Sample Nr.	Species											
	Iris setosa				Iris versicolor				Iris virginica			
	Features				Features				Features			
	sepal length	sepal width	petal length	petal width	sepal length	sepal width	petal length	petal width	sepal length	sepal width	petal length	petal width
1	5.1	3.5	1.4	0.2	7.0	3.2	4.7	1.4	6.3	3.3	6.0	2.5
2	4.9	3.0	1.4	0.2	6.4	3.2	4.5	1.5	5.8	2.7	5.1	1.9
3	4.7	3.2	1.3	0.2	6.9	3.1	4.9	1.5	7.1	3.0	5.9	2.1
4	4.6	3.1	1.5	0.2	5.5	2.3	4.0	1.3	6.3	2.9	5.6	1.8
5	5.0	3.6	1.4	0.2	6.5	2.8	4.6	1.5	6.5	3.0	5.8	2.2
6	5.4	3.9	1.7	0.4	5.7	2.8	4.5	1.3	7.6	3.0	6.6	2.1
7	4.6	3.4	1.4	0.3	6.3	3.3	4.7	1.6	4.9	2.5	4.5	1.7
8	5.0	3.4	1.5	0.2	4.9	2.4	3.3	1.0	7.3	2.9	6.3	1.8
9	4.4	2.9	1.4	0.2	6.6	2.9	4.6	1.3	6.7	2.5	5.8	1.8
10	4.9	3.1	1.5	0.1	5.2	2.7	3.9	1.4	7.2	3.6	6.1	2.5
11	5.4	3.7	1.5	0.2	5.0	2.0	3.5	1.0	6.5	3.2	5.1	2.0
12	4.8	3.4	1.6	0.2	5.9	3.0	4.2	1.5	6.4	2.7	5.3	1.9
13	4.8	3.0	1.4	0.1	6.0	2.2	4.0	1.0	6.8	3.0	5.5	2.1
14	4.3	3.0	1.1	0.1	6.1	2.9	4.7	1.4	5.7	2.5	5.0	2.0
15	5.8	4.0	1.2	0.2	5.6	2.9	3.6	1.3	5.8	2.8	5.1	2.4
16	5.7	4.4	1.5	0.4	6.7	3.1	4.4	1.4	6.4	3.2	5.3	2.3
17	5.4	3.9	1.3	0.4	5.6	3.0	4.5	1.5	6.5	3.0	5.5	1.8
18	5.1	3.5	1.4	0.3	5.8	2.7	4.1	1.0	7.7	3.8	6.7	2.2
19	5.7	3.8	1.7	0.3	6.2	2.2	4.5	1.5	7.7	2.6	6.9	2.3
20	5.1	3.8	1.5	0.3	5.6	2.5	3.9	1.1	6.0	2.2	5.0	1.5
21	5.4	3.4	1.7	0.2	5.9	3.2	4.8	1.8	6.9	3.2	5.7	2.3
22	5.1	3.7	1.5	0.4	6.1	2.8	4.0	1.3	5.6	2.8	4.9	2.0
23	4.6	3.6	1.0	0.2	6.3	2.5	4.9	1.5	7.7	2.8	6.7	2.0
24	5.1	3.3	1.7	0.5	6.1	2.8	4.7	1.2	6.3	2.7	4.9	1.8
25	4.8	3.4	1.9	0.2	6.4	2.9	4.3	1.3	6.7	3.3	5.7	2.1
26	5.0	3.0	1.6	0.2	6.6	3.0	4.4	1.4	7.2	3.2	6.0	1.8
27	5.0	3.4	1.6	0.4	6.8	2.8	4.8	1.4	6.2	2.8	4.8	1.8
28	5.2	3.5	1.5	0.2	6.7	3.0	5.0	1.7	6.1	3.0	4.9	1.8
29	5.2	3.4	1.4	0.2	6.0	2.9	4.5	1.5	6.4	2.8	5.6	2.1
30	4.7	3.2	1.6	0.2	5.7	2.6	3.5	1.0	7.2	3.0	5.8	1.6
31	4.8	3.1	1.6	0.2	5.5	2.4	3.8	1.1	7.4	2.8	6.1	1.9
32	5.4	3.4	1.5	0.4	5.5	2.4	3.7	1.0	7.9	3.8	6.4	2.0
33	5.2	4.1	1.5	0.1	5.8	2.7	3.9	1.2	6.4	2.8	5.6	2.2
34	5.5	4.2	1.4	0.2	6.0	2.7	5.1	1.6	6.3	2.8	5.1	1.5
35	4.9	3.1	1.5	0.2	5.4	3.0	4.5	1.5	6.1	2.6	5.6	1.4
36	5.0	3.2	1.2	0.2	6.0	3.4	4.5	1.6	7.7	3.0	6.1	2.3
37	5.5	3.5	1.3	0.2	6.7	3.1	4.7	1.5	6.3	3.4	5.6	2.4
38	4.9	3.6	1.4	0.1	6.3	2.3	4.4	1.3	6.4	3.1	5.5	1.8
39	4.4	3.0	1.3	0.2	5.6	3.0	4.1	1.3	6.0	3.0	4.8	1.8
40	5.1	3.4	1.5	0.2	5.5	2.5	4.0	1.3	6.9	3.1	5.4	2.1
41	5.0	3.5	1.3	0.3	5.5	2.6	4.4	1.2	6.7	3.1	5.6	2.4
42	4.5	2.3	1.3	0.3	6.1	3.0	4.6	1.4	6.9	3.1	5.1	2.3
43	4.4	3.2	1.3	0.2	5.8	2.6	4.0	1.2	5.8	2.7	5.1	1.9
44	5.0	3.5	1.6	0.6	5.0	2.3	3.3	1.0	6.8	3.2	5.9	2.3
45	5.1	3.8	1.9	0.4	5.6	2.7	4.2	1.3	6.7	3.3	5.7	2.5
46	4.8	3.0	1.4	0.3	5.7	3.0	4.2	1.2	6.7	3.0	5.2	2.3
47	5.1	3.8	1.6	0.2	5.7	2.9	4.2	1.3	6.3	2.5	5.0	1.9
48	4.6	3.2	1.4	0.2	6.2	2.9	4.3	1.3	6.5	3.0	5.2	2.0
49	5.3	3.7	1.5	0.2	5.1	2.5	3.0	1.1	6.2	3.4	5.4	2.3
50	5.0	3.3	1.4	0.2	5.7	2.8	4.1	1.3	5.9	3.0	5.1	1.8

Tabela II

TABLE III
FEATURE SELECTION AND ERROR ESTIMATION

Classes	Nr. 1 "setosa"			Nr. 2 "virginica"			Nr. 3 "versicolor"		
Nr. of samples	50			50			50		
Selection Strategy									
Selected feature Nr.	Univar			M-Covar			MinErr		
	Frequency Nr.	Criterion	Estimated error [%]	Frequency Nr.	Criterion	Estimated error [%]	Frequency Nr.	Criterion	Estimated error [%]
1	3	0.81	12.00	3	107.2	12.00	2	12.00	= Criterion
2	2	0.77	4.67	2	25.65	4.67	3	4.67	
3	0	-0.43	4.67	0	11.66	4.67	0	4.67	
4	1	-3.32	4.00	1	4.07	4.00	1	4.00	

Tabela III

Qualitativamente todas as estratégias consideram que as duas características *petal_length* (comprimento da pétala) e *petal_width* (largura da pétala) são características boas; o que reduz o erro estimado da classificação consideravelmente. As outras duas características *sepal_length* (comprimento da sépala) e *sepal_width* (largura da sépala) não contribuem para melhorias perceptíveis na taxa de erro. Elas são características ruins.

A figura IV.20 mostra a visualização do conjunto de dados pelo algoritmo de Sammon.

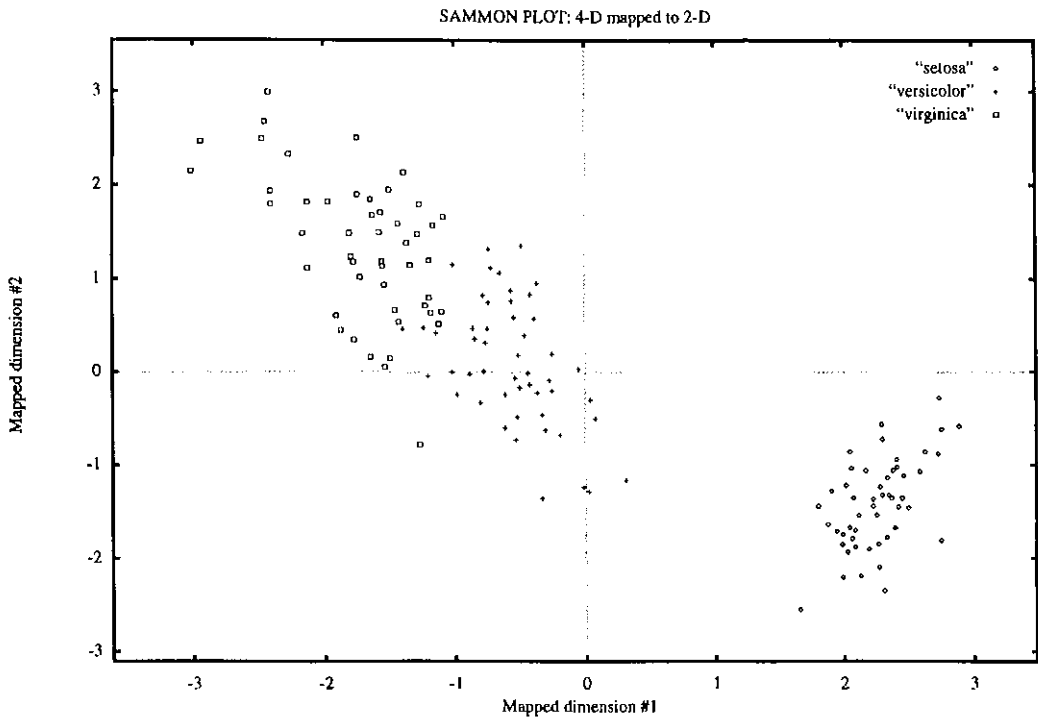


Figura IV.20. A saída do Sammon plot para banco de dados das flores íris

Esse exemplo de análise de dados mostra bem a idéia de se reunir um conjunto de estratégias dentro de um único pacote de forma a ter um meio comparativo entre as várias técnicas, ou seja, o uso de estratégias diferentes levando a resultados similares. Isso permite conclusões sobre a qualidade das características, e assim faz sentido usar essas características que foram consideradas boas para classificar as três espécies de flores. Além disso, se só 50% das características são consideradas boas e são suficientes para uma classificação satisfatória, isso significa dizer que é desnecessário armazenar as outras características. Nesse caso, tem-se uma economia de 50% no armazenamento de dados.

V.2.1.2. No CN2

É interessante notar que para o conjunto das flores íris, o CN2 forma poucas regras simples (como mostra a figura IV.21), porém, capazes de classificar esses dados com uma excelente exatidão de classificação. A exatidão conseguida é de 100% para as classes íris-setosa e íris-virginica, e a classe íris-versicolor apresenta uma exatidão de 94%. A exatidão geral é de 98% como mostra a figura IV.21.

```

**RULE FILE**
@
Time: [ Thu Apr 10 12:31:49 1997 ]
Examples: iris.exs
Algorithm: UNORDERED
Error_Estimate: LAPLACIAN
Threshold: 0.00
Star: 5
@

*UNORDERED-RULE-LIST*

IF  petal-length < 2.45
THEN class = Iris-setosa [50 0 0]

IF  sepal-length > 4.95
  AND 2.45 < petal-length < 4.95
  AND petal-width < 1.75
THEN class = Iris-versicolor [0 46 0]

IF  2.35 < sepal-width < 2.45
THEN class = Iris-versicolor [0 3 0]

IF  sepal-length > 6.25
  AND sepal-width > 2.45
  AND petal-length > 5.05
THEN class = Iris-viginica [0 0 35]

IF  sepal-width < 3.10
  AND petal-width > 1.75
THEN class = Iris-viginica [0 0 28]

IF  sepal-length < 6.15
  AND sepal-width < 2.65
  AND petal-length > 4.45
THEN class = Iris-viginica [0 0 4]

IF  petal-length > 5.25
THEN class = Iris-viginica [0 0 32]

(DEFAULT) class = Iris-setosa [50 50 50]

-----
PREDICTED
ACTUAL  Iris-se Iris-ve Iris-vi Accuracy
Iris-set    50    0    0  100.0 %
Iris-ver     3   47    0   94.0 %
Iris-vig     0    0   50  100.0 %
Overall accuracy: 98.0 %
Default accuracy: 33.3 %

```

Figura IV.21. Arquivos de regras e saída da exatidão de classificação das flores íris.

V.1.2.3. No Wang-Mendel

Para descrever as classes conta-se com quatro variáveis. O modelo é, portanto, composto de cinco variáveis descritas pela seguinte relação:

$$\text{Classes} = f(\text{petal_length}, \text{petal_width}, \text{sepal_length}, \text{sepal_width})$$

O próximo passo é colocar os dados no formato “entendido” pelo programa. Como exemplo, a figura IV.22 mostra uma parte do arquivo de comportamento para o banco de dados das flores íris.

```
//SEPAL  SEPAL  PETAL  PETAL  CLASS
//LENGTH WIDTH LENGTH WIDTH
//
5.1,    3.5,    1.4,    0.2,    Iris-setosa
4.9,    3.0,    1.4,    0.2,    Iris-setosa
4.7,    3.2,    1.3,    0.2,    Iris-setosa
4.6,    3.1,    1.5,    0.2,    Iris-setosa
5.0,    3.6,    1.4,    0.2,    Iris-setosa
5.4,    3.9,    1.7,    0.4,    Iris-setosa
4.6,    3.4,    1.4,    0.3,    Iris-setosa
5.0,    3.4,    1.5,    0.2,    Iris-setosa
4.4,    2.9,    1.4,    0.2,    Iris-setosa
4.9,    3.1,    1.5,    0.1,    Iris-setosa
```

Figura IV.22. Parte do arquivo de comportamento dos dados

Depois de lidos os dados, pode-se executar o programa de descoberta de regras. A figura IV.23 mostra a forma do relatório do programa para o arquivo de dados das flores íris.

```

wmdriver
Enter name of Data File: flores.txt
1.0- -Partition Variables into Fuzzy Sets.
1.1- - - -Generating Term Set for 'SepalLength'.
1.1- - - -Generating Term Set for 'SepalWidth'.
1.1- - - -Generating Term Set for 'PetalLength'.
1.1- - - -Generating Term Set for 'PetalWidth'.
1.1- - - -Generating Term Set for 'Classes'.
2.0- -Generate Intermediate Rules from Trainig Data.
2.1- - - -Learning Rules from Data File: 'flores.txt'
2.2- - - -Formed 150 Rules.
3.0- -Create Combined FAM and write Production Rules.
3.1- - - -Generating FAM from Rule File: 'a:WMrules.FIL'.
.....
.....
3.2- - - -Combined FAM contains 51 Rules.
FIM.

```

Figura IV.23. Relatório gerado pelo programa wmdriver alterado.

As diferenças encontradas no arquivo de saída indicam onde foram feitas alterações para que o método de descoberta de regras pudesse trabalhar com o arquivo de dados das flores íris.

Das 150 regras geradas o sistema selecionou 51 regras verdadeiras.

Passo Um - Decompor as variáveis em conjuntos fuzzy

A decomposição das variáveis fuzzy é feita nessa primeira parte do processo de descoberta de regras. Cada variável é decomposta em cinco conjuntos fuzzy: o do centro, dois à esquerda e dois à direita. O número de conjuntos fuzzy a serem criados depende do conhecimento de um especialista, e pode ser diferente para cada variável. No caso desse conjunto de dados, é coerente usar o mesmo desdobramento para todas as variáveis, já que elas apresentam a mesma unidade de medida e os dados se encontram bem distribuídos ao longo do domínio. Escolheu-se dividir espaço das variáveis em 5 conjuntos fuzzy. Na figura IV.24, exemplifica-se o processo de criação dos conjuntos fuzzy para a variável *sepal_length* como aparece no relatório de *log* do programa.

```

-----
RD---[001]: Rule Discovery. Fuzzy Term Set Generation
Variable : SepalLength
Domain : 4.00 to 8.00
INITIAL PARAMETERS
DomRange : 4.00
MidPoint : 6.00
Partitions: 5
PartWidth : 1.20

```

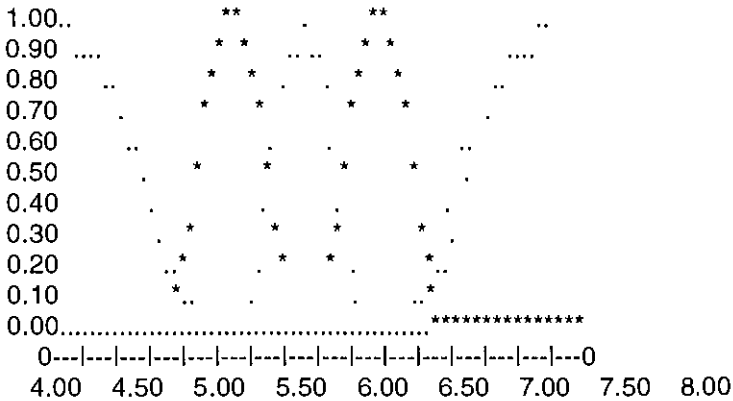
FuzzySet	Curve Edges		Curve Surface Parameters		
	Left	Right	P1	P2	P3
1. SPL_S002	4.00	5.40	4.00	4.70	5.40
2. SPL_S001	4.80	6.00	5.40	0.60	5.40
3. SPL_CE	5.40	6.60	6.00	0.60	6.00
4. SPL_B001	6.00	7.20	6.60	0.60	6.60
5. SPL_B002	6.60	8.00	6.60	7.30	8.00

Figura IV.24. Relatório detalhando as propriedades para o conjunto de termos para a variável SEPAL_LENGTH

A distribuição dos conjuntos fuzzy da variável *sepal_length* em sua forma gráfica é mostrada na figura IV.25.

SepalLength

Domain [UofD]: 4.00 to 8.00



- . FuzzySet: SPL_S002
Description:
Support : 4.00, 4.70, 5.40
AlphaCut: 0.10
- * FuzzySet: SPL_S001
Description:
Support : 5.40, 0.60
AlphaCut: 0.10
- . FuzzySet: SPL_CE
Description:
Support : 6.00, 0.60
AlphaCut: 0.10
- * FuzzySet: SPL_B001
Description:
Support : 6.60, 0.60
AlphaCut: 0.10
- . FuzzySet: SPL_B002
Description:
Support : 6.60, 7.30, 8.00
AlphaCut: 0.10

Figura IV.25. Os conjuntos fuzzy associados com a variável *sepal_length*

Um conjunto fuzzy faz uma superposição de 50% a seus vizinhos à esquerda e à direita. Dessa forma, se a pertinência a um conjunto fuzzy é um, a pertinência a seu vizinho é zero. Durante a modelagem de um problema real, essa condição pode apresentar-se alterada, ou seja, é necessária uma superposição maior ou menor que 50%. É possível também, alterar a forma dos conjuntos fuzzy para conjuntos fuzzy triangulares ou trapezoidais de modo a se adaptarem melhor às condições do problema.

Passo Dois – Gera Regras de Tentativa para os Dados.

O passo de produção de regras é feito em várias partes. Primeiramente, cada linha do arquivo de comportamento dá origem a uma regra de tentativa. A última variável que aparece em cada linha do arquivo de regras é a variável dependente ou solução do problema. Essas regras se apresentam como no exemplo abaixo.

```
if SepalLength is SPL_S001 and SepalWidth is SPW_CE and PetalLength is PTL_S002 and  
PetalWidth is PTW_S002 then Class is CLA_S002
```

Dentro de cada regra de tentativa é associado um grau de pertinência máximo a cada predicado e consequente. Esse grau de pertinência é obtido a partir dos conjuntos fuzzy. Depois de obtidos os graus de pertinência, esses valores são multiplicados gerando o grau de eficácia da regra. Esse valor aparece após o último grau de pertinência como exemplificado abaixo.

```
SepalLength, SPL_S001, 0.4794, SepalWidth, SPW_CE, 1.0000, PetalLength, PTL_S002,  
0.9280, PetalWidth, PTW_S002, 0.9280, Classes, CLA_S002, 1.0000, 0.4128
```

A transição do arquivo de comportamento dos dados para a regra com seus respectivos graus de eficácia é colocada em um arquivo de log, o wmaud02.fil, como mostra a figura IV.26.

02/04/1998 10.30am

RD---[002]: Rule Discovery. Initial Rule Generation

Data File : flores.txt

Rule File : a:WMrules.fil

Audit File : a:WMAud02.fil

1. 5.1, 3.5, 1.4, 0.2, 1

RD---[003]: Rule Generation Statistics from Data:

'SepalLength' (5.10) has max m[0.4794] in termset 'SPL_S001'

'SepalWidth' (3.50) has max m[1.0000] in termset 'SPW_CE'

'PetalLength' (1.40) has max m[0.9280] in termset 'PTL_S002'

'PetalWidth' (0.20) has max m[0.9280] in termset 'PTW_S002'

'Classes' (1.00) has max m[1.0000] in termset 'CLA_S002'

Unit ConfWeight: 1.0000, RuleDegree: 0.4128

2. 4.9, 3.0, 1.4, 0.2, 1

RD---[003]: Rule Generation Statistics from Data:

'SepalLength' (4.90) has max m[0.2648] in termset 'SPL_S002'

'SepalWidth' (3.00) has max m[0.9713] in termset 'SPW_S001'

'PetalLength' (1.40) has max m[0.9280] in termset 'PTL_S002'

'PetalWidth' (0.20) has max m[0.9280] in termset 'PTW_S002'

'Classes' (1.00) has max m[1.0000] in termset 'CLA_S002'

Unit ConfWeight: 1.0000, RuleDegree: 0.2215

3. 4.7, 3.2, 1.3, 0.2, 1

RD---[003]: Rule Generation Statistics from Data:

'SepalLength' (4.70) has max m[0.5177] in termset 'SPL_S002'

'SepalWidth' (3.20) has max m[0.7914] in termset 'SPW_S001'

'PetalLength' (1.30) has max m[0.9641] in termset 'PTL_S002'

'PetalWidth' (0.20) has max m[0.9280] in termset 'PTW_S002'

'Classes' (1.00) has max m[1.0000] in termset 'CLA_S002'

Unit ConfWeight: 1.0000, RuleDegree: 0.3666

4. 4.6, 3.1, 1.5, 0.2, 1

RD---[003]: Rule Generation Statistics from Data:

'SepalLength' (4.60) has max m[0.6403] in termset 'SPL_S002'

'SepalWidth' (3.10) has max m[0.9843] in termset 'SPW_S001'

'PetalLength' (1.50) has max m[0.8901] in termset 'PTL_S002'

'PetalWidth' (0.20) has max m[0.9280] in termset 'PTW_S002'

'Classes' (1.00) has max m[1.0000] in termset 'CLA_S002'

Unit ConfWeight: 1.0000, RuleDegree: 0.5206

5. 5.0, 3.6, 1.4, 0.2, 1

RD---[003]: Rule Generation Statistics from Data:

'SepalLength' (5.00) has max m[0.2222] in termset 'SPL_S001'

'SepalWidth' (3.60) has max m[0.9132] in termset 'SPW_CE'

'PetalLength' (1.40) has max m[0.9280] in termset 'PTL_S002'

'PetalWidth' (0.20) has max m[0.9280] in termset 'PTW_S002'

'Classes' (1.00) has max m[1.0000] in termset 'CLA_S002'

Unit ConfWeight: 1.0000, RuleDegree: 0.1748

6. 5.4, 3.9, 1.7, 0.4, 1

Figura IV.26. Análise registro-a registro

O grau de contribuição final da regra é igual ao grau de eficácia da regra multiplicado pelo índice de credibilidade. O índice de credibilidade é um valor que pode ou não ser definido. Caso ele não seja definido, é assumido como 1. Depois de

calculado o grau de contribuição para cada regra, é gerado um arquivo cujas linhas contêm os valores de cada variável, acompanhada do nome do conjunto fuzzy contendo o máximo grau de pertinência, e o respectivo grau. Ao fim da linha, é adicionado o grau de contribuição. Mostra-se na figura IV.27 uma parte da matriz de regras intermediária.

```
SepalLength, SPL_S001, 0.4794, SepalWidth, SPW_CE, 1.0000, PetalLength, PTL_S002, 0.9280, PetalWidth, PTW_S002, 0.9280, Classes, CLA_S002, 1.0000, 0.4128
SepalLength, SPL_S002, 0.2648, SepalWidth, SPW_S001, 0.9713, PetalLength, PTL_S002, 0.9280, PetalWidth, PTW_S002, 0.9280, Classes, CLA_S002, 1.0000, 0.2215
SepalLength, SPL_S002, 0.5177, SepalWidth, SPW_S001, 0.7914, PetalLength, PTL_S002, 0.9641, PetalWidth, PTW_S002, 0.9280, Classes, CLA_S002, 1.0000, 0.3666
SepalLength, SPL_S002, 0.6403, SepalWidth, SPW_S001, 0.9843, PetalLength, PTL_S002, 0.8901, PetalWidth, PTW_S002, 0.9280, Classes, CLA_S002, 1.0000, 0.5206
SepalLength, SPL_S001, 0.2222, SepalWidth, SPW_CE, 0.9132, PetalLength, PTL_S002, 0.9280, PetalWidth, PTW_S002, 0.9280, Classes, CLA_S002, 1.0000, 0.1748
SepalLength, SPL_S001, 0.9995, SepalWidth, SPW_B001, 0.9737, PetalLength, PTL_S002, 0.7905, PetalWidth, PTW_S002, 0.7120, Classes, CLA_S002, 1.0000, 0.5478
SepalLength, SPL_S002, 0.6403, SepalWidth, SPW_CE, 0.8901, PetalLength, PTL_S002, 0.9280, PetalWidth, PTW_S002, 0.8443, Classes, CLA_S002, 1.0000, 0.4465
SepalLength, SPL_S001, 0.2222, SepalWidth, SPW_CE, 0.8901, PetalLength, PTL_S002, 0.8901, PetalWidth, PTW_S002, 0.9280, Classes, CLA_S002, 1.0000, 0.1634
```

Figura IV.27. A matriz de regras intermediária

Passo Três – Cria as Regras Fuzzy Combinadas.

Para resolver problemas redundantes e conflitantes que podem aparecer como resultado do passo 2, utiliza-se o grau de contribuição. A comparação é feita para a mesma variável, o mesmo conjunto fuzzy, para todas as variáveis e finalmente para o grau de contribuição. Embora os graus de pertinência tenham sido mantidos na matriz intermediária eles não são utilizados na comparação. O resultado dessa parte do processo é o arquivo de compressão de regras wmaud03.fil mostrado na figura IV.28.

RD---[003]: Rule Discovery. Combined FAM Generation

Audit File : a:WMaud03.fil

Rule File : a:WMrules.FIL

FAM File : a:WMfam.fil

RD---[005]: Record: 2. Rule ADDED: SPL_S002xSPW_S001xPTL_S002xPTW_S002: CLA_S002 (0.2215)

RD---[005]: Record: 3. Rule UPDATES: SPL_S002xSPW_S001xPTL_S002xPTW_S002: CLA_S002 (0.3666)

RD---[005]: Record: 4. Rule UPDATES: SPL_S002xSPW_S001xPTL_S002xPTW_S002: CLA_S002 (0.5206)

RD---[007]: Record: 5. Rule REJECTED: SPL_S001xSPW_CExPTL_S002xPTW_S002: CLA_S002 (0.1748 versus 0.4128)

RD---[005]: Record: 6. Rule ADDED: SPL_S001xSPW_B001xPTL_S002xPTW_S002: CLA_S002 (0.5478)

RD---[005]: Record: 7. Rule ADDED: SPL_S002xSPW_CExPTL_S002xPTW_S002: CLA_S002 (0.4465)

RD---[007]: Record: 8. Rule REJECTED: SPL_S001xSPW_CExPTL_S002xPTW_S002: CLA_S002 (0.1634 versus 0.4128)

RD---[005]: Record: 9. Rule UPDATES: SPL_S002xSPW_S001xPTL_S002xPTW_S002: CLA_S002 (0.5447)

RD---[007]: Record: 10. Rule REJECTED: SPL_S002xSPW_S001xPTL_S002xPTW_S002: CLA_S002 (0.2283 versus 0.5447)

RD---[005]: Record: 11. Rule UPDATES: SPL_S001xSPW_CExPTL_S002xPTW_S002: CLA_S002 (0.5020)

Figura IV.28. Parte do relatório de auditoria a partir da geração de regras fuzzy

A partir desse resultado, monta-se o arquivo contendo as linhas selecionadas pelo processo de compressão de regras na forma *if-then*. Para o exemplo das flores íris foram selecionadas 51 das 150 regras geradas. Essas regras são mostradas a seguir na figura IV.29.

[R001]: if SepalLength is SPL_S001 and SepalWidth is SPW_CE and PetalLength is PTL_S002 and PetalWidth is PTW_S002 then Classes is CLA_S002;
[R002]: if SepalLength is SPL_S002 and SepalWidth is SPW_S001 and PetalLength is PTL_S002 and PetalWidth is PTW_S002 then Classes is CLA_S002;
[R003]: if SepalLength is SPL_S001 and SepalWidth is SPW_B001 and PetalLength is PTL_S002 and PetalWidth is PTW_S002 then Classes is CLA_S002;
[R004]: if SepalLength is SPL_S002 and SepalWidth is SPW_CE and PetalLength is PTL_S002 and PetalWidth is PTW_S002 then Classes is CLA_S002;
[R005]: if SepalLength is SPL_CE and SepalWidth is SPW_B001 and PetalLength is PTL_S002 and PetalWidth is PTW_S002 then Classes is CLA_S002;
[R006]: if SepalLength is SPL_S001 and SepalWidth is SPW_B002 and PetalLength is PTL_S002 and PetalWidth is PTW_S002 then Classes is CLA_S002;
[R007]: if SepalLength is SPL_S001 and SepalWidth is SPW_S001 and PetalLength is PTL_S002 and PetalWidth is PTW_S002 then Classes is CLA_S002;
[R008]: if SepalLength is SPL_S002 and SepalWidth is SPW_S002 and PetalLength is PTL_S002 and PetalWidth is PTW_S002 then Classes is CLA_S002;
[R009]: if SepalLength is SPL_B001 and SepalWidth is SPW_S001 and PetalLength is PTL_B001 and PetalWidth is PTW_CE then Classes is CLA_CE;
[R010]: if SepalLength is SPL_S001 and SepalWidth is SPW_S002 and PetalLength is PTL_CE and PetalWidth is PTW_CE then Classes is CLA_CE;
[R011]: if SepalLength is SPL_S001 and SepalWidth is SPW_S001 and PetalLength is PTL_B001 and PetalWidth is PTW_CE then Classes is CLA_CE;
[R012]: if SepalLength is SPL_CE and SepalWidth is SPW_CE and PetalLength is PTL_B001 and PetalWidth is PTW_CE then Classes is CLA_CE;
[R013]: if SepalLength is SPL_S002 and SepalWidth is SPW_S002 and PetalLength is PTL_S001 and PetalWidth is PTW_S001 then Classes is CLA_CE;

[R044]: if SepalLength is SPL_B001 and SepalWidth is SPW_CE and PetalLength is PTL_B002 and PetalWidth is PTW_B001 then Classes is CLA_B002;
[R045]: if SepalLength is SPL_CE and SepalWidth is SPW_S001 and PetalLength is PTL_B001 and PetalWidth is PTW_B001 then Classes is CLA_B002;
[R046]: if SepalLength is SPL_B002 and SepalWidth is SPW_S001 and PetalLength is PTL_B002 and PetalWidth is PTW_CE then Classes is CLA_B002;
[R047]: if SepalLength is SPL_CE and SepalWidth is SPW_S001 and PetalLength is PTL_B001 and PetalWidth is PTW_CE then Classes is CLA_B002;
[R048]: if SepalLength is SPL_CE and SepalWidth is SPW_S002 and PetalLength is PTL_B002 and PetalWidth is PTW_CE then Classes is CLA_B002;
[R049]: if SepalLength is SPL_B002 and SepalWidth is SPW_S001 and PetalLength is PTL_B002 and PetalWidth is PTW_B002 then Classes is CLA_B002;
[R050]: if SepalLength is SPL_B001 and SepalWidth is SPW_CE and PetalLength is PTL_B002 and PetalWidth is PTW_B002 then Classes is CLA_B002;
[R051]: if SepalLength is SPL_CE and SepalWidth is SPW_CE and PetalLength is PTL_B001 and PetalWidth is PTW_B002 then Classes is CLA_B002;

Figura IV.29. Regras finais para o modelo das flores íris de Fisher

IV.3. Conjunto de Dados Climáticos do Aeroporto Internacional do Galeão

Os dados climáticos do Galeão são resultado de uma coleta feita ao longo de 10 anos. São, portanto, dez arquivos que vão de 1951 a 1960. Cada um deles contém registros compostos por 39 campos – também chamados nessa análise, de atributos ou características – diferentes. Vários desses atributos apresentaram-se bastante incompletos, acima de 98%. Outros ainda, não se apresentavam tão incompletos, porém, como restringiam a quantidade de registros completos dentro dos arquivos de dados e principalmente não eram classificados como bons pelo método de avaliação utilizado (módulo de seleção de características do pacote Tooldiag), foram também eliminados do arquivo de dados. Assim, das 39 características iniciais, 20 características foram eliminadas na primeira triagem e a análise dos dados começou a ser feita em arquivos com 19 atributos.

O objetivo da análise dos dados era a previsão de nevoeiro. A classificação dos dados foi feita baseada em um dos atributos iniciais, o campo “tempo presente” que mais tarde foi retirado da análise. A primeira classificação dividiu os dados em sete classes: nevoeiro, pré-nevoeiro1h, pré-nevoeiro2h, pré-nevoeiro3h, pré-nevoeiro4h, pré-nevoeiro5h, normal. As primeiras análises foram feitas considerando essas sete classes, porém, concluiu-se que a quantidade de dados era insuficiente para classificar bem sete classes. Mesmo após a retirada de mais quatro características da

análise, resultando em um aumento de quase 30% nos dados, a exatidão geral de classificação ainda se apresentava baixa. Como não havia mais a possibilidade de aumentar a quantidade de dados para a análise, a solução encontrada foi diminuir o número de classes. Houve, então, um agrupamento de classes, resultando em cinco classes: nevoeiro, pré-nevoate2h (resultante da união das classes pré-nevoeiro1h e pré-nevoeiro2h), pré-nevo2ha4h (resultante da união das classes pré-nevoeiro3h e pré-nevoeiro4h) e pré-nevoeiro5h. Os resultados apresentaram uma pequena melhora, porém, as exatidões de classificação ainda se apresentavam baixas. Houve, então, uma nova junção de classes, resultando em três classes finais a saber: nevoeiro, pré-nevoeiro, normal.

O primeiro arquivo a ser trabalhado foi o G1951R19.txt (relativo ao ano de 1951 com dezenove características) e apresentava baixa exatidão no “módulo” escolhido para o teste nos dados. Como havia muitos atributos e várias classes dentro do arquivo, a primeira conclusão foi a de que faltavam dados para uma boa classificação.

O próximo passo foi fazer como sugerido nos processos de extração de conhecimento: a idéia era ter dois arquivos, um para treinamento dos dados e outro para teste. Optou-se, então, por fazer dois grandes arquivos juntando no conjunto de treinamento os cinco primeiros arquivos (referentes aos anos 1951 a 1955) e no teste os outros cinco (referentes aos anos 1956 a 1960). Durante os testes, porém, percebeu-se que a exatidão no processo de seleção de características ainda se apresentava muito baixa (em torno de 19%). Durante a execução do módulo de estimativa de erro do Tooldiag, percebeu-se que apenas em alguns trechos dos dados, o valor do erro estimado crescia consideravelmente e a hipótese mais provável para a baixa exatidão do conjunto de treinamento dos dados passou a ser discrepância ou erro nos dados. Retirando-se o arquivo referente a 1951, percebeu-se imediatamente uma melhoria para a faixa de 70% nas melhores características selecionadas, sendo que o erro estimado em alguns casos caiu para quase 0% quando se utilizava todas as cinco melhores características.

Durante a utilização do arquivo de teste, porém, aconteceu o mesmo problema e dessa vez vários trechos dos dados apresentaram um aumento considerável no erro estimado. Presumindo que a mesma situação ocorrida nos dados de treinamento se repetia, tentou-se novamente isolar o problema e dessa vez verificou-se que vários arquivos estavam apresentando erros de leitura. Os arquivos 1957, 1958, 1959, 1960,

mostraram uma exatidão muito baixa , inferior a 40%, no teste de exatidão feito pelo Tooldiag.

Como vários arquivos apresentaram uma baixa exatidão, optou-se por retirá-los da análise e criar um grande arquivo com os arquivos restantes 1952, 1953, 1954, 1955, 1956, chamado de G2a6R19.txt, e analisar a exatidão encontrada, a fim de determinar se essas análises apontavam o caminho certo. Isso foi confirmado quando o G2a6R19.txt apresentou uma exatidão em torno de 70% no teste de avaliação escolhido.

Voltou-se ao banco de dados dos 19 atributos e verificou-se que o mesmo ainda continha atributos mais de 50% incompletos e para aumentar a quantidade de amostras, optou-se por cortar mais quatro atributos, sendo eles: altura das nuvens da 1ª camada, quantidade de nuvens da 2ª camada, tipo de nuvens da 2ª camada, altura das nuvens da 2ª camada, reduzindo o número de características para 15. Isso só pôde ser feito porque essas características não se apresentavam fundamentais ao processo de extração de conhecimento, ou seja, exatidão baixa. As 15 características finais são relacionadas a seguir:

- 1°. mês
- 2°. dia
- 3°. hora
- 4°. total de nuvens
- 5°. direção do vento
- 6°. velocidade do vento em nós
- 7°. visibilidade horizontal
- 8°. quantidade de nuvens da 1 camada
- 9°. tipo de nuvens da 1 camada
- 10°. altura das nuvens da 1 camada em decâmetros
- 11°. ajuste do altímetro em milibares e décimos
- 12°. temperatura do ponto de orvalho
- 13°. pressão em atm ao nível do mar
- 14°. temperatura do ar do bulbo seco
- 15°. temperatura do ar do bulbo úmido

Com a eliminação dos quatro atributos mencionados, conseguiu-se um aumento de quase 30% da quantidade de amostras e retornou-se à análise. Dessa vez, porém, optou-se por começar com um teste arquivo a arquivo para verificar que características cada um selecionava e com que exatidão, contornando assim, o

problema de mais tarde ter que particionar o arquivo de treinamento ou o de teste para saber a que arquivos pertencem as amostras problemáticas. Houve uma união temporária de todos os arquivos para saber como a redução dessas quatro características havia alterado a exatidão da classificação. Finalmente, os arquivos foram divididos em um arquivo de treinamento contendo os arquivos referentes aos anos 1952, 1953 e 1954 e um arquivo de testes contendo os arquivos 1955 e 1956, todos com 15 atributos.

V.3.1. Resultados Comentados:

V.3.1.1. No Tooldiag

O TOOLDIAG só manuseia registros completos. No caso dos dados climáticos, houve um grande trabalho inicial para formatá-los de maneira a serem utilizados pelo pacote. Utilizou-se o aplicativo DTMining para o pré-processamento dos dados. Finalizado o primeiro trabalho de preparação dos dados, partiu-se para a análise dos mesmos. A análise a seguir é feita para 3 classes: normal, nevoeiro, pré-nevoeiro.

Os dados climáticos quando submetidos aos diversos métodos do pacote Tooldiag apresentaram resultados bastante variados embora a característica que melhor se classificou tenha se mantido a mesma para quase todos os métodos. Para o primeiro método apresentado: estratégia de procura *Best Features*, com o critério de seleção *Estimated minimal error probability*; a seleção apresentou características com exatidões de classificação bem altas. Os valores das características selecionadas apresentaram-se bem próximos uns aos outros, como mostra a figura IV.30.

```

#
#----- FEATURE SELECTION -----
# Search strategy: Best Features
# Selection criterion group: Estimated minimal error probability
# - Classifier: 1-Nearest Neighbor, using raw data
# - Cross validation: Leave-One-Out
#-----
g2a4r15ori.txt
5
# Features Selection criterion (-1 = EMPTY)
7 0.722972 visibihoriz
10 0.722329 altnuv1c
2 0.722044 dia
3 0.722044 hora
4 0.722044 totnuvens

```

Figura IV.30. O arquivo de seleção de características g2a6R15-111.fea

Esse resultado é bastante alto se for considerada o número de atributos diferentes e o volume de dados em condições de serem manuseadas pelo pacote. Acontece, porém, que conforme se diminui o número de amostras da classe ‘normal’ essa exatidão diminui consideravelmente. No caso de uma redução em que se mantém um para cada oito registros da classe ‘normal’ essa exatidão cai para a faixa dos 24%. A figura IV.31. confirma o relato acima.

```

#
#----- FEATURE SELECTION -----
# Search strategy: Best Features
# Selection criterion group: Estimated minimal error probability
# - Classifier: 1-Nearest Neighbor, using raw data
# - Cross validation: Leave-One-Out
#-----
g2a6j15.txt
5
# Features Selection criterion (-1 = EMPTY)
7 0.249718 visibihoriz
14 0.246033 temparbseco
10 0.245828 altnuv1c
13 0.244087 presatmmar
6 0.242040 velocivento

```

Figura IV.31. O arquivo de seleção de características g2a6J15-111.fea

Dando continuidade a apresentação dos diversos métodos de seleção de características do tooldiag, passa-se a sequência seguinte. Para a mesma estratégia de procura anterior, passa-se a trabalhar com o critério de seleção *interclass distance* para o qual o pacote apresenta cinco métricas diferentes. As quatro primeiras

métricas: minkowsky, city block, euclidean distance e chebychev, apresentam os mesmos resultados. Foi escolhida apenas uma para apresentar os resultados que valem para o grupo como mostra a figura IV.32.

```
#
#----- FEATURE SELECTION -----
# Search strategy: Best Features
# Selection criterion group: Inter-class distance
# Distance metric:EUCLIDEAN DISTANCE
# Calculating mutual distance between ALL classes
#-----
g2a4r15ori.txt
5
# Features Selection criterion (-1 = EMPTY)
7 261.864655 visibihoriz
10 62.875679 altnuv1c
11 21.755684 ajustaltimetro
13 21.673048 presatmmar
14 14.255381 temparbseco
```

Figura IV.32. O arquivo de seleção de características g2a6R15-2121.fea

Para a métrica *parzen and hyperspheric kernel*, a seleção aparece de forma bem diferente das anteriores e apresenta uma exatidão quase uniforme para todas as características, como mostra a figura IV.33.

```
#
#----- FEATURE SELECTION -----
# Search strategy: Best Features
# Selection criterion group: Inter-class distance
# Distance metric:Nonlinear (Parzen & hyperspheric kernel)
# Calculating mutual distance between ALL classes
#-----
g2a4r15ori.txt
5
# Features Selection criterion (-1 = EMPTY)
11 20.162636 ajustaltimetro
13 20.161617 presatmmar
14 19.965725 temparbseco
15 19.880363 temparbumido
2 19.624216 dia
```

Figura IV.33. O arquivo de seleção de características g2a6r15-2125.fea

Mudando o critério de seleção para *probabilistic distance*, nota-se que a ordem das características selecionadas se mantém a mesma para as quatro primeiras métricas, Chernoff, Bhattacharyya, Matusita, Divergence, e Mahalanobis sendo representada

por apenas uma delas; a sexta e sétima métricas variam bastante as características selecionadas como mostrado na figura IV.34.

```

g2a4r15-2131.fea
#
#----- FEATURE SELECTION -----
#      Search strategy:           Best Features
#      Selection criterion group:  Probabilistic distance
#      Distance metrik:Chernoff with s = 0.500
#-----
g2a4r15ori.txt
5
# Features      Selection criterion (-1 = EMPTY)
7      0.086985   visibihoriz
14     0.018514   temparbseco
6      0.012509   velocivento
3      0.009421   hora
10     0.007725   altnuv1c

g2a4r15-2136.fea
#
#----- FEATURE SELECTION -----
#      Search strategy:           Best Features
#      Selection criterion group:  Probabilistic distance
#      Distance metrik:Patrick-Fisher
#-----
g2a4r15ori.txt
5
# Features      Selection criterion (-1 = EMPTY)
6      0.003462   velocivento
9      0.002561   tiponuv1c
4      0.002292   totnuvens
3      0.001521   hora
8      0.001123   quantnuv1c

g2a4r15-2137.fea
#
#----- FEATURE SELECTION -----
#      Search strategy:           Best Features
#      Selection criterion group:  Probabilistic distance
#      Distance metrik:Normalized distance between means
#                               = 1 - (s1+s2)^2 / (m1-m2)^2
#-----
g2a4r15ori.txt
5
# Features      Selection criterion (-1 = EMPTY)
7      -2.194054   visibihoriz
6      -22.458981  velocivento
10     -30.445976   altnuv1c
14     -98.876198  temparbseco
4      -246.891617  totnuvens

```

Figura IV.34. Três arquivos de seleção de características comparados entre si

Várias combinações de métodos e métricas apresentados nessa análise mostram seleções de características idênticas ou muito semelhantes. Isso demonstra uma coerência entre os métodos de seleção de características.

Depois de mostrar todas essas técnicas e métricas pertencentes ao módulo de seleção de características, apresenta-se a seguir uma abordagem a alguns métodos de estimativa de erro, a partir de um conjunto de características escolhido que se manterá o mesmo durante toda a análise. Para efeito didático, achou-se conveniente rememorar o conjunto de características com as quais se vai trabalhar, como mostra a figura IV.35.

```
#
#----- FEATURE SELECTION -----
# Search strategy: Best Features
# Selection criterion group: Estimated minimal error probability
# - Classifier: 1-Nearest Neighbor, using raw data
# - Cross validation: Leave-One-Out
#-----
g2a4r15ori.txt
5
# Features Selection criterion (-1 = EMPTY)
7 0.722972 visibihoriz
10 0.722329 altnuv1c
2 0.722044 dia
3 0.722044 hora
4 0.722044 totnuvens
```

Figura IV.35. O arquivo de seleção de características g2a6R15-111.fea

A meta, agora, é mostrar alguns resultados de métodos de estimativa de erro a partir das características selecionadas acima. O módulo de estimativa de erro serve como uma avaliação do método de seleção de características utilizado. Para esse módulo, é necessária a escolha de um modelo de classificador. O primeiro classificador escolhido foi o *1-Nearest Neighbor*. O próximo passo é a escolha do método de estimativa de erro. O método escolhido foi o *Resubstitution*. O cálculo do erro estimado foi feito levando-se em consideração apenas as cinco características selecionadas pelo módulo de seleção de características, introduzindo-as uma a uma. O resultado dessa avaliação é mostrado na figura IV.36.

```

erro11-111.fea
### ERROR ESTIMATION PROTOCOL ###
Universe: g2a4r15ori.txt
----- CROSS VALIDATION -----
NAME: Resubstitution
with increasing number of features
CLASSIFIER MODEL: 1-Nearest Neighbor, using raw data

```

Estimation for increasing number of features!

```

=====
=== Number of selected features: 1 ===
Nr. Features: 1 - Estimated error: 27.60 % - Accuracy: 72.40 %
=====

=====
=== Number of selected features: 2 ===
Nr. Features: 2 - Estimated error: 25.64 % - Accuracy: 74.36 %
=====

=====
=== Number of selected features: 3 ===
Nr. Features: 3 - Estimated error: 15.30 % - Accuracy: 84.70 %
=====

=====
=== Number of selected features: 4 ===
Nr. Features: 4 - Estimated error: 2.95 % - Accuracy: 97.05 %
=====

=====
=== Number of selected features: 5 ===
Nr. Features: 5 - Estimated error: 1.00 % - Accuracy: 99.00 %
=====

```

Figura IV.36. O arquivo de estimativa de erro erro11-111.err

Utilizando-se o mesmo classificador, as mesmas características selecionadas, e variando-se o método de estimativa de erro para *Hold-out*, há uma boa variação no resultado final. Para esse método, o erro estimado é de 27,65%, após a introdução da primeira característica, melhora sutilmente com a introdução da segunda e da terceira características, e piora sutilmente com a introdução da quarta e da quinta características como mostra a figura IV.37. Para essa combinação de classificador e método de estimativa de erro, a quarta e a quinta características podem ser dispensadas da análise, pois não conseguem diminuir o erro estimado e acrescentam processamento.

ERROR ESTIMATION PROTOCOL

Universe: g2a4r15ori.txt

----- CROSS VALIDATION -----

NAME: Holdout
splitting into 70.00% training data and 30.00% test data
taking the mean over 5 runs
with increasing number of features
CLASSIFIER MODEL: 1-Nearest Neighbor, using raw data

Estimation for increasing number of features!
=====

=== Number of selected features: 1 ===

--- AVERAGE STATISTICS OF ERROR ESTIMATION ---

Mean over 5 runs has been taken

STATISTICAL PARAMETERS OF THE ESTIMATED ERROR:

MEAN=27.6499% STANDARD DEVIATION=0.00456539

VARIANCE=0.00002084

MINIMUM=27.046% MAXIMUM=28.069%

Nr. Features: 1 - Estimated error: 27.65 % - Accuracy: 72.35 %

=====

=== Number of selected features: 2 ===

--- AVERAGE STATISTICS OF ERROR ESTIMATION ---

Mean over 5 runs has been taken

STATISTICAL PARAMETERS OF THE ESTIMATED ERROR:

MEAN=27.3739% STANDARD DEVIATION=0.00197392

VARIANCE=0.00000390

MINIMUM=27.141% MAXIMUM=27.712%

Nr. Features: 2 - Estimated error: 27.37 % - Accuracy: 72.63 %

=====

=== Number of selected features: 3 ===

--- AVERAGE STATISTICS OF ERROR ESTIMATION ---

Mean over 5 runs has been taken

STATISTICAL PARAMETERS OF THE ESTIMATED ERROR:

MEAN=26.5033% STANDARD DEVIATION=0.00398963

VARIANCE=0.00001592

MINIMUM=26.047% MAXIMUM=27.165%

Nr. Features: 3 - Estimated error: 26.50 % - Accuracy: 73.50 %

=====

=== Number of selected features: 4 ===

--- AVERAGE STATISTICS OF ERROR ESTIMATION ---

Mean over 5 runs has been taken

STATISTICAL PARAMETERS OF THE ESTIMATED ERROR:

MEAN=27.8735% STANDARD DEVIATION=0.00340265

VARIANCE=0.00001158

MINIMUM=27.569% MAXIMUM=28.520%

Nr. Features: 4 - Estimated error: 27.87 % - Accuracy: 72.13 %

=====

=== Number of selected features: 5 ===

--- AVERAGE STATISTICS OF ERROR ESTIMATION ---

Mean over 5 runs has been taken

STATISTICAL PARAMETERS OF THE ESTIMATED ERROR:

MEAN=28.0637% STANDARD DEVIATION=0.00637313

VARIANCE=0.00004062

MINIMUM=26.832% MAXIMUM=28.687%

Nr. Features: 5 - Estimated error: 28.06 % - Accuracy: 71.94 %

=====

Figura IV.37. O arquivo de estimativa de erro erro12-111.err

Alterando-se o método do classificador para *Quadratic Gaussian Classifier*, e utilizando-se o método de estimativa de erro *Resubstitution*, nota-se nitidamente que o erro estimado tem um valor constante e razoável em termos de classificação para a primeira e segunda características. Quando se introduz a terceira característica, ele diminui sutilmente, porém, quando se introduz a quarta e quinta características, o valor do erro aumenta significativamente, tornando-se alto o suficiente para condenar as duas últimas características, como mostra a figura IV.38.

```

erro21-111.err
### ERROR ESTIMATION PROTOCOL ###
Universe: g2a4r15ori.txt
----- CROSS VALIDATION -----
      NAME:                Resubstitution
              with increasing number of features
      CLASSIFIER MODEL:    Quadratic Gaussian Classifier
-----
Estimation for increasing number of features!

=====
=== Number of selected features:  1 ===
Nr. Features:  1 - Estimated error:    23.60  % - Accuracy:  76.40  %
=====

=====
=== Number of selected features:  2 ===
Nr. Features:  2 - Estimated error:    23.60  % - Accuracy:  76.40  %
=====

=====
=== Number of selected features:  3 ===
Nr. Features:  3 - Estimated error:    27.80  % - Accuracy:  72.20  %
=====

=====
=== Number of selected features:  4 ===
Nr. Features:  4 - Estimated error:    55.63  % - Accuracy:  44.37  %
=====

=====
=== Number of selected features:  5 ===
Nr. Features:  5 - Estimated error:    57.02  % - Accuracy:  42.98  %
=====

```

Figura IV.38. O arquivo de estimativa de erro erro21-111.err

Dando sequência a análise, alterou-se novamente o modelo de classificador para *Radial Basis Function Network*, porém, a implementação do tooldiag para o método não foi capaz de manusear esses dados. Passou-se então para o quarto modelo de classificador apresentado que é o *Parzen Window*. Para esse classificador,

conjugado com o método de estimativa de erro *Resubstitution*, a exatidão mostra uma melhora ínfima de 0.02% da primeira para a segunda característica, mantém-se constante até a introdução da quarta característica ao cálculo do erro, mas cai 9,01% quando se introduz a quinta característica. É, portanto, correto afirmar que para essa combinação de classificador e método de estimativa de erro, a quinta característica seria desnecessária, só contribuindo para diminuir a exatidão da classificação como demonstra a figura IV.39. Na realidade, depois da introdução da segunda característica, não houve mais melhorias no valor do erro estimado, ou seja, a terceira, quarta e quinta características seriam dispensáveis para essa análise.

```
err41-111.err
### ERROR ESTIMATION PROTOCOL ###
Universe: g2a4r15ori.txt
----- CROSS VALIDATION -----
                                NAME:           Resubstitution
                                with increasing number of features
                                CLASSIFIER MODEL: Parzen window
-----
Estimation for increasing number of features!

=====
=== Number of selected features: 1 ===
Nr. Features: 1 - Estimated error:    27.80  % - Accuracy:  72.20  %
=====

=====
=== Number of selected features: 2 ===
Nr. Features: 2 - Estimated error:    27.78  % - Accuracy:  72.22  %
=====

=====
=== Number of selected features: 3 ===
Nr. Features: 3 - Estimated error:    27.78  % - Accuracy:  72.22  %
=====

=====
=== Number of selected features: 4 ===
Nr. Features: 4 - Estimated error:    27.78  % - Accuracy:  72.22  %
=====

=====
=== Number of selected features: 5 ===
Nr. Features: 5 - Estimated error:    36.79  % - Accuracy:  63.21  %
=====
```

Figura IV.39. O arquivo de estimativa de erro erro41-111.err

Passa-se a seguir para o último modelo de classificador apresentado pelo pacote, que é o Q^* *Algorithm*. Associando-o ao método de estimativa de erro *Resubstitution*, nota-se um aumento vertiginoso do valor do erro estimado calculado para o primeiro atributo, e esse valor não consegue ser melhorado com a introdução das demais características. O único ponto de acordo entre esse método e os demais, é continuar considerando a quinta característica dispensável, já que a sua adição ao processo piorou ainda mais o resultado da estimativa de erro como mostra a figura IV.40.

```
err51-111.err
### ERROR ESTIMATION PROTOCOL###
Universe: g2a4r15ori.txt
----- CROSS VALIDATION -----
      NAME:                Resubstitution
                with increasing number of features
      CLASSIFIER MODEL:   Q* algorithm
-----
Estimation for increasing number of features!

=====
=== Number of selected features:  1 ===
Nr. Features:  1 - Estimated error:    87.30  % - Accuracy:  12.70  %
=====

=====
=== Number of selected features:  2 ===
Nr. Features:  2 - Estimated error:    87.30  % - Accuracy:  12.70  %
=====

=====
=== Number of selected features:  3 ===
Nr. Features:  3 - Estimated error:    87.30  % - Accuracy:  12.70  %
=====

=====
=== Number of selected features:  4 ===
Nr. Features:  4 - Estimated error:    87.30  % - Accuracy:  12.70  %
=====

=====
=== Number of selected features:  5 ===
Nr. Features:  5 - Estimated error:    93.03  % - Accuracy:   6.97  %
=====
```

Figura IV.40. O arquivo de estimativa de erro erro51-111.err

Depois de selecionadas as características, e comprovada a sua exatidão através do módulo de estimativa de erro, pode-se testar a exatidão de classificação das características selecionadas em um novo arquivo de dados com o mesmo formato daquele do qual se extraiu as características. Para isso, é preciso primeiramente

selecionar um modelo de classificador. Selecionado o classificador *1-Nearest neighbor*, deve-se carregar o arquivo de dados de teste. Isso é feito no módulo de “Classificação de Dados Independentes”. Os dados de teste são resultado da união dos arquivos referentes aos anos 1955 e 1956, e é chamado de G5e6R15.txt. Agora, testa-se a acuidade das características selecionadas para o arquivo de dados de treinamento G2a4R15.txt (junção dos arquivos referentes aos anos 1952,1953 e1954) nos dados de teste. Para o primeiro classificador, a exatidão de classificação foi de 64,07%, como mostra a figura IV.41, o que pode ser considerada uma boa classificação.

```
Indep1-111.sai
Load benchmark data from file? g5e6r15.txt
----- CLASSIFIER MODEL -----
NAME:      1-Nearest Neighbor, using raw data
-----
No classifier available - Induce it ? (y/n)y
That's easy. Nothing to do, just keeping all raw samples!
Verbose the reasoning process? (y/n)n
Show confusion matrix? (y/n)n

#### Using 1-Nearest-Neighbor Classifier ####

--- RESULT for benchmark file g5e6r15.txt:

ERROR RATE = 35.93% --- ACCURACY 64.07% --- Recognized 6365 of 9934
```

Figura IV.41. O arquivo de exatidão de classificação de dados independentes
indep1-111.err

Selecionado o segundo classificador, o *Quadratic Gaussian Classifier*, mantendo os mesmos dados de teste, com as mesmas características selecionadas, a exatidão de classificação subiu para 70,14%, como mostra a figura IV.42.

```

indep2-111.sai
Load benchmark data from file? g5e6r15.txt
----- C L A S S I F I E R M O D E L -----
NAME:      Quadratic Gaussian Classifier
SIMPLIFICATIONS:
* Having identical covariance matrices: NO
-----
No classifier available - Induce it ? (y/n)y

--- Inducing a 5-variate Gaussian model ---

Verbose learning of decision functions? (y/n)n
Verbose the reasoning process? (y/n)n
Show confusion matrix? (y/n)n

#### Using Bayes Classifier ####

--- RESULT for benchmark file g5e6r15.txt:

ERROR RATE = 29.59% --- ACCURACY 70.41% --- Recognized 6995 of 9934

```

Figura IV.42. O arquivo indep2-111.err

Como foi comentado anteriormente, o terceiro modelo de classificador, o *Radial Basis Function Network*, apresenta problemas para trabalhar com os dados climáticos. No teste dos dados, esse modelo de classificador, também não conseguiu gerar resultados .

O Quarto modelo de classificador, o *Parzen Window*, utilizando o kernel hipercúbico, apresentou uma exatidão de classificação muito próxima a do classificador *1- Nearest Neighbor*, como mostra a figura IV.43.

```

indep41-111.sai
Load benchmark data from file? g5e6r15.txt
----- CLASSIFIER MODEL -----
NAME:      Parzen window
Parzen Window Kernel Type: Hypercubic
Window width (as a fraction of total range of feature values): 15.00%
-----
No classifier available - Induce it ? (y/n)y

--- Inducing a 5-variate Parzen Classifier ---

Verbose learning of decision structure? (y/n)n

Verbose the reasoning process? (y/n)Show confusion matrix? (y/n)n

#### Using Parzen Window Classifier ####

--- RESULT for benchmark file g5e6r15.txt:

ERROR RATE = 35.32% --- ACCURACY 64.68% --- Recognized 6425 of 9934

```

Figura IV.43. O arquivo indep41-111.err

V.3.1.2. No CN2

O CN2 é um algoritmo de indução e execução das regras induzidas produzindo um percentual de exatidão na execução das regras sobre uma base de dados, que pode ser individual para cada regra ou geral. O CN2 manuseia não só registros completos como também registros incompletos. Testa-se o comportamento do CN2 para os dois tipos de arquivos. Logo de saída, nota-se que para registros incompletos o CN2 é muito mais lento na análise de dados e indução de regras. Quando os dados se apresentam incompletos, os valores inexistentes são substituídos por "?". Quando os atributos são texto devem estar entre aspas.

Depois de induzidas as regras para cada classe, deve-se executá-las sobre a base de dados que as gerou para verificar a exatidão das mesmas. Pode-se também executar essas regras sobre uma nova base de dados que contenha as mesmas características da base de dados que a criou.

Quanto maior o número de classes, mais dados são necessários para uma boa classificação. Quando os dados com os quais se está trabalhando se encontram muito incompletos e possuem muitos atributos mesmo depois da triagem e não se introduz qualquer conhecimento do especialista para auxiliar os programas, é preciso entregar a maior quantidade de dados possível para que o pacote seja capaz de extrair padrões (regras ou conjuntos de atributos) que possam classificar com uma exatidão aceitável.

Existe, porém o risco de que os dados não convirjam para padrões e aí seria necessário o conhecimento do especialista para induzir o pacote a uma conclusão. Isto está relacionado com vários fatores, entre eles: a forma com que as classes se apresentam ou seja, se as classes são altamente superpostas é muito difícil determinar atributos ou características que distingam perfeitamente as classes.

Quando uma classe está bem definida, o CN2 consegue formar regras associando vários atributos dentro de uma mesma regra, por exemplo:

```
Best Rules:  
IF    totnuvens > 0.5  
      AND  velocvent > 6.5  
      AND  visibhoriz >1900.00  
      AND  altnuv1c > 245.00  
      AND  ajualt > 71.50  
      AND  presatmmar < 107.50  
THEN classes = normal
```

Entretanto, se o CN2 formasse muitas regras complexas de forma que todas as situações dos dados fossem retratadas, então correria-se o risco de que essas regras se superadequassem aos dados e apresentassem uma classificação pobre para quaisquer outros dados. O CN2 consegue, porém, evitar que isso ocorra.

Quando uma classe não é muito bem definida pelos dados, as regras geradas são pobres em sua totalidade e a exatidão da classificação das regras é baixa. Abaixo, mostra-se um exemplo dessa situação:

```
Best Rules:  
IF    visibhoriz > 1550.00  
      AND  70 < altnuv1c <145.00  
      AND  ajualt >82.00  
      AND  244.50 < temparbumido < 247.50  
THEN classes = normal
```

O arquivo G2a4R15.txt é resultante da união dos arquivos referentes ao anos 1952, 1953 e 1954, e foi utilizado para treinamento dos dados. Desse mesmo arquivo, foi feito primeiramente uma variação em dois arquivos, um arquivo de três e outro de cinco classes, já que se concluiu que não havia dados suficientes para se trabalhar com as sete classes determinadas inicialmente. Depois cada um desses arquivos deu origem a mais três, resultantes da diminuição do número de registros da classe normal. Isso foi feito porque a classe normal apresentava um número muito superior de amostras em relação às outras classes. A idéia era verificar se diminuindo o

número de amostras da classe normal melhoraria o desempenho do pacote. Considerava-se que não havia a necessidade de tantas amostras de uma mesma classe. A primeira redução que manteve um de cada dois registros da classe normal gerou a versão b, a segunda redução manteve um de cada três registros da classe normal e gerou a versão c, e a terceira manteve um de cada quatro registros lidos e gerou a versão d.

Para a sequência de arquivos de três classes, pode-se notar que a versão b apresenta uma boa exatidão geral de 73,8%, embora a classe 'pré_nevoeiro' apresente uma exatidão baixa de 34,2% com mostra a figura IV.44.

```
WRITE> Filename? 3cg2a4b.rul
Writing current rules to 3cg2a4b.rul
CN> Execute
EVAL> all
Executing rules...
  PREDICTED
ACTUAL   normal  nevoeir pre_nev Accuracy
normal   4313   430   316   85.3 %
nevoeiro  121   1571   84   88.5 %
pre_nevo  852   542   725   34.2 %
Overall accuracy: 73.8 %
Default accuracy: 56.5 %
```

Figura IV.44. O arquivo g2a4r15b-3c.out

Na versão c, nota-se que embora o número de amostras tenha diminuído a quantidade de dados, já que foram retiradas mais amostras da classe normal, a exatidão geral de classificação aumentou para 75,4%, por causa da melhoria da exatidão da classificação da classe 'pré_nevoeiro'. Embora a princípio só tenha sido alterada a classe normal, isso pode ter reflexos em outras classes. É possível, por exemplo que a eliminação de amostras que se encontravam entremeadas com as amostras de uma outra classe, melhore a definição da fronteira entre as duas classes. A figura IV.45 mostra o resultado do CN2 para a versão c.

```

WRITE> Filename? 3cg2a4c.rul
Writing current rules to 3cg2a4c.rul
CN> Execute
EVAL> all
Executing rules...
  PREDICTED
ACTUAL  normal  nevoeir pre_nev Accuracy
Normal   3085    287     0   91.5 %
nevoeiro  204    1572     0   88.5 %
pre_nevo  727     542    850   40.1 %
Overall accuracy: 75.8 %
Default accuracy: 46.4 %

```

Figura IV.45. O arquivo g2a4r15c-3c.out

Com a nova redução do arquivo de dados, nota-se novamente uma melhoria da exatidão geral de classificação para 77,1% e novamente isso se deveu ao aumento da exatidão da classe ‘pré_nevoeiro’ como mostra a figura.

```

WRITE> Filename? 3cg2a4d.rul
Writing current rules to 3cg2a4d.rul
CN> Execute
EVAL> all
Executing rules...
  PREDICTED
ACTUAL  normal  nevoeir pre_nev Accuracy
normal   2529     0     0  100.0 %
nevoeiro  437    1339     0   75.4 %
pre_nevo 1037     0    1082   51.1 %
Overall accuracy: 77.1 %
Default accuracy: 39.4 %

```

Figura IV.46. O arquivo g2a4r15d-3c.out

Quando se aumenta o número de classes de três para cinco, e se compara a mesma versão b dos dois arquivos, percebe-se que já há uma queda da exatidão geral de classificação de 73.8% para 66.9%. Nota-se ainda que para essa sequência de arquivos acontece basicamente o mesmo processo da sequência anterior. Aqui, quando se diminui o número de amostras da classe normal, há uma melhoria sutil da exatidão de classificação de todas as classes, exceto a classe normal que se manteve constante em 100%. Porém, como a quantidade de amostras pertencentes a classe normal vai diminuindo bastante, o percentual de exatidão de classificação associado a essa classe (100%) passa a ter um peso menor sobre o total das amostras e, portanto,

sobre a exatidão geral de classificação que também diminui. Isso demonstra que não há dados suficientes para classificar nem cinco classes. Na figura IV.46, mostra-se os resultados do CN2 para a sequência de arquivos de cinco classes.

```
WRITE> Filename? 5cg2a4b.rul
Writing current rules to 5cg2a4b.rul
CN> Execute
EVAL> all
Executing rules...
  PREDICTED
ACTUAL  normal  nevoeir pre_nev pre_nev pre_nev Accuracy
normal   5059    0      0      0      0    100.0 %
nevoeiro  859     917    0      0      0    51.6 %
pre_nevo  979      8      5      0      0     0.5 %
pre_nevo  762      5      0      2      0     0.3 %
pre_nevo  355      0      0      0      3     0.8 %
Overall accuracy: 66.9 %
Default accuracy: 56.5 %
```

```
WRITE> Filename? 5cg2a4c.rul
Writing current rules to 5cg2a4c.rul
CN> Execute
EVAL> all
Executing rules...
  PREDICTED
ACTUAL  normal  nevoeir pre_nev pre_nev pre_nev Accuracy
normal   3372    0      0      0      0    100.0 %
nevoeiro  602    1174    0      0      0    66.1 %
pre_nevo  969      1     22     0      0     2.2 %
pre_nevo  764      0      0      5      0     0.7 %
pre_nevo  354      0      0      0      4     1.1 %
Overall accuracy: 63.0 %
Default accuracy: 46.4 %
```

```
WRITE> Filename? 5cg2a4d.rul
Writing current rules to 5cg2a4d.rul
CN> Execute
EVAL> all
Executing rules...
  PREDICTED
ACTUAL  normal  nevoeir pre_nev pre_nev pre_nev Accuracy
normal   2171    358     0      0      0    85.8 %
nevoeiro  318    1458     0      0      0    82.1 %
pre_nevo  531    385    76     0      0     7.7 %
pre_nevo  396    360     1     12     0     1.6 %
pre_nevo  200    154     0      0      4     1.1 %
Overall accuracy: 57.9 %
Default accuracy: 39.4 %
```

Figura IV.46. Os arquivos g2a4r15b-5c.out, g2a4r15c-5c.out, g2a4r15d-5c.out

V. Conclusões:

Utilizou-se dois bancos de dados para avaliação dos algoritmos escolhidos. O primeiro foi o das flores íris de Fisher e conseguiu-se uma ótima exatidão para o CN2. Para o Tooldiag, as características selecionadas como melhores se diversificaram um pouco ao longo da análise, o que está relacionado com a dificuldade de alguns métodos em tratar superposições de classes. O sistema Wang-Mendel teve que ser modificado para trabalhar com o banco de dados das flores íris. Depois de concluído todo o processo de descoberta de regras gerou 51 regras fuzzy.

A expectativa maior surgiu em torno dos dados climáticos, pois tratava-se de dados problemáticos que exigiam pré-processamento e avaliação. O CN2 apresentou uma precisão geral considerável, embora só duas das três classes definidas tenham apresentado a desejável exatidão de classificação. No tooldiag, métodos e métricas diferentes selecionaram conjuntos variados de cinco características cada. A característica selecionada em primeiro lugar, entretanto se manteve sempre a mesma. O estudo de erro estimado foi feito utilizando-se o primeiro grupo de características selecionado, empregando novamente diversos métodos. De maneira geral, o valor do erro se manteve baixo (chegando às vezes a 0% com a utilização dos cinco melhores atributos), o que demonstrou um bom desempenho do módulo de seleção de características.

É interessante notar que essa análise prova que os métodos de extração de conhecimento permitem partir de um banco de dados qualquer e fazendo-se apenas uma “limpeza” nos dados e formatando-os para se adequarem aos algoritmos de extração de conhecimento, é possível obter-se várias informações sobre os dados. Caso esses dados se mostrem confiáveis ao longo da análise, é possível também obter classificações boas como mostrou a análise dos dados climáticos. Essa foi uma primeira abordagem sobre métodos de extração de conhecimento, onde o objetivo não foi dissertar sobre todas as técnicas existentes, mas selecionar algumas, descrevê-las e utilizá-las para avaliar suas facilidades e deficiências. Além disso, existem vários outros bancos de dados que permitiriam um estudo rico em termos de avaliação das técnicas de extração de conhecimento, nas mais diversas áreas como médica, financeira, etc. Fica assim a sugestão para estudos futuros.

A seguir comenta-se sobre as falhas apresentadas pelo CN2, método Wang-Mendel e o pacote Tooldiag.

V.1. CN2

O CN2 é um método baseado em regras fácil de ser usado e embora suporte dados contendo vários atributos, apresentou uma limitação em número de registros que não ultrapassa os 10.000 (dez mil). Quando essa situação ocorre, o CN2 simplesmente interrompe o processamento apresentando uma mensagem de erro. As regras geradas podem ser salvas em um arquivo e aplicadas em qualquer banco de dados com o mesmo formato daquele que o gerou. A geração das regras é a parte do processo que mais exige esforço computacional, podendo demorar horas para um arquivo de 10.000 registros. Depois de formadas as regras, porém, sua “execução” sobre o banco de dados é rápida, gerando um resultado para a exatidão de classificação das regras geradas quase instantâneo.

V.2. Método Wang-Mendel

Esse método de formação de regras fuzzy apresenta uma biblioteca contendo uma grande quantidade de funções que podem ser utilizadas facilmente e servem como base para qualquer trabalho envolvendo conjuntos fuzzy. Porém, o modelo como um todo está longe de poder ser mais amplamente utilizado. Primeiramente, ele só prevê duas variáveis de entrada e uma de saída, o que o torna inadequado a quase todas as aplicações do mundo real. Para que o sistema fosse aplicado ao exemplo das flores íris de Fisher, várias alterações foram feitas e uma parte teve que ser reprogramada. Como originalmente o programa trabalha com alocação dinâmica de memória, para um conjunto de dados um pouco maior que o exemplo apresentado, a memória do micro computador não é capaz de suportar. Esse problema de alocação dinâmica de memória seria facilmente contornado para sistemas que possuíssem paginação de memória em disco.

Além disso, a utilização do Método Wang-Mendel apresenta como resultado regras fuzzy, sendo necessária a utilização de um interpretador para que essas regras sejam melhor entendidas e avaliadas, o que se sugere como extensão desse trabalho.

V.3. Tooldiag.

O Tooldiag contém um conjunto de métodos dentro de um único pacote, o que o torna uma boa opção em termos de recursos e de comparação entre métodos. Contudo, o pacote em si apresenta um *help* precário o que dificulta tanto a utilização quanto o entendimento de seus resultados. Foi necessária uma extensa pesquisa para se compreender vários de seus métodos e métricas. A saída do software não apresenta maiores explicações sobre os resultados e a interface com o usuário deixa bastante a desejar. Além disso, o pacote não tem muitos recursos e qualquer problema durante a sua execução, ocasiona a volta ao sistema operacional.

Para que esse pacote pudesse lidar com banco de dados maiores, vários métodos teriam que ser otimizados, pois algumas combinações de métodos e métricas levaram muitas horas para fornecer os resultados, trabalhando com um banco de dados de aproximadamente 10.000 linhas. Outras combinações sequer foram concluídas.

Resumindo, para que esse pacote pudesse ter uma aplicação mais ampla, seria necessário vários ajustes e otimizações, além de uma *interface* de usuário com mais recursos.

V.4. Sugestões e Trabalhos Futuros

A maior parte dos métodos abordados nessa tese se baseiam em métodos clássicos como as regras e árvores de decisão, *K-nearest neighbor*, etc. Outros já incorporam conceitos mais recentes como a Rede de função de base radial e o Parzen Window. Existem, porém, várias outras técnicas que podem ser utilizadas na extração de conhecimento, além das técnicas apresentadas: redes neurais, redes polinomiais,

projection pursuit, estimativa de densidade, etc. Implementações robustas desses métodos permitem o manuseio de arquivos de dados da ordem de giga e até terabytes.

Os bancos de dados apresentados nesse trabalho serviram como base para demonstrar toda a variedade de aplicações e situações em que se pode extrair conhecimento: de um simples banco de dados de 150 amostras, completo e sem erros; até um banco de dados incompleto, com erros e discrepâncias, com muitas amostras e vários atributos. Existem, ainda, muitas outras aplicações nas mais diversas áreas, onde poderia-se realizar a extração de conhecimento.

Como continuação natural do trabalho apresentado, sugere-se a utilização de algumas técnicas mais recentes aplicadas a grandes bancos de dados, em outras áreas como a financeira, comercial, médica, etc.

Bibliografia:

- [1] BABCOCK, C., "Parallel Processing Mines Retail Data", *Computer World*, v. 6, pp. 62-71, Sept. 1994.
- [2] HARRISON, D., "Backing up", *Network Computing*, pp. 98-104, Oct. 1993.
- [3] FASMAN, K. H., CUTICCHIA, A. J., and KINGSBURRY, D. T., 1994, "The GDB (TM) Human Genome Database Anno 1994", *Nucl. Acid. R.*, v. 17, pp. 3462-3469, 1994.
- [4] MICHALSKI, R. S., CARBONELL, J. G., and MITCHELL, T. M., "Machine Learning, An Artificial Intelligence Approach", v.1, 1 ed., San Mateo, California Morgan Kaufmann, 1983.
- [5] DURAN, B. S., and ODELL, P. L., "Cluster Analysis: a survey", *Lectures Notes in Economics and Matematical Systems*, v. 100, Spinger-Verlag, 1974.
- [6] BRACHMAN, R. J., ANAND, T., "The Process of Knowledge Discovery in Databases: A Human Centered Approach". In: Fayad, U. M., Piatesky-Shapiro, G., Smyth, P., and Uthurasamy, R. S. (eds), *Advances in Knowledge Discovery and Data Mining*, 1 ed., chapter 1, Menlo Park, Calif, The AAAI Press, 1996.
- [7] CHEESEMAN, P., "On Finding the Most Probable Model". In: Shrager, J. and Langley, P. (eds), *Computacional Models of Scientific Discovery and Theory Formation*,. 1 ed, San Francisco, Morgan Kaufmann, pp. 73-95, 1990.
- [8] QUINLAN, J. R., "Learning efficient classification procedures and their application to chess endgames". In: Carbonell, J.G., Michaelski, R. S., Mitchell, T. M.(eds), *Machine Learning*, v. 1, Palo Alto, Calif, The AAAI Press, 1983.
- [9] QUINLAN, J. R., "Induction for decision trees". In: *Machine Learning*, v. 1, pp. 81-106, The AAAI Press, 1986.

- [10] QUINLAN, J. R., “C4.5: Programs for Machine Learning”, ed.1, San Mateo, Calif, Morgan Kaufmann, 1992.
- [11] QUINLAN, J. R. “The effect of noise on concept learning”. In: Michalski et al. [], pages 149-166.
- [12] UTGOFF, P. E., “Incremental Induction of decision trees”. In: *Machine Learning*, v. 4, pp. 161-186, The AAAI Press, 1989.
- [13] MICHALSKI, R. S., MOZATIC, I., HONG, J., LAVRAC, N., “The AQ15 inductive learning system: an overview and experiments”. In: *Proceedings of IMAL* Orsay, France, Université de Paris-Sud, 1986.
- [14] MICHALSKI, R. S., MOZATIC, I., HONG, J., LAVRAC, N., “The multi-purpose incremental learning system AQ15 and its testing application to three medical domains”. In: *Proceedings of the 5th national conference on Artificial Intelligence*, pp. 1041-1045, Philadelphia, 1986.
- [15] CAI, Y., CERCONE, N., HAN, J., “Attribute-oriented induction in relational databases”. In: Piatetsky-Shapiro and Frawley [...], páginas 213-228.
- [16] HAN, J., CAI, Y., CERCONE, N., “Knowledge discovery in databases: An attribute-oriented approach”. In: *Proceedings of the 18th VLDB Conference*, pp. 547-559, Vancouver, British Columbia, Canada, 1992.
- [17] CLARK, P.; NIBLETT, T.; “The CN2 induction algorithm”. In: *Machine Learning*, v. 3, pp. 261-283, 1989.
- [18] MOWFORTH, P., “Some Applications with inductive expert system shells”. v. 86-002, Turing Institute, Glasgow, UK, 1986.

- [19] MICHALSKI, R. S., "On the quasi-minimal solution of the general covering problem". In: *Proceedings of the 5th international symposium of Information Processing (FCIP 69)*, v. A3, pp. 125-128, Bled, Yugoslavia, 1969.
- [20] QUINLAN, J. R., "Simplifying decision trees". In: *Journal of Man-Machine Studies*, v. 3, pp. 221-234, Sept. 1987.
- [21] NIBLETT, T., "Constructing decision trees in noisy domains". In: *Proceedings of the 2nd European Working Session on Learning*, pp. 67-78, Wilmslow, UK, 1987.
- [22] QUINLAN, J. R., COMPTON, P. J., HORN, K. A., LAZARUS, L., "Inductive knowledge acquisition: a case study". In: *Applications of Expert Systems*, pp. 157-173. Addison-Wesley, Wokingham, UK, 1987.
- [23] KONONENKO, I.; BRATKO, I.; ROSKAR, E., "Experiments in automatic learning of medical diagnostic rules". In: *Technical Report, Faculty of Electrical Engineering*, E. Kardelj University, Ljubljana, 1984.
- [24] MICHALSKI, R. S.; and LARSON, J., "Incremental generation of v_1 hypotheses: the underlying methodology and the description of program aq11". ISG 83-5. Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign, Urbana, 1983.
- [25] IBA, W., WOGULIS, J., and LANGLEY, P., "Trading off simplicity and coverage in incremental concept learning". In: *Proceedings of the 5th International Conference On Machine Learning*, pp. 73-79, Calif, 1988.
- [26] RIVEST, R. L., "Learning decision lists". In: *Machine Learning*, v.2, pp. 229-246, 1987.
- [27] MICHALSKI, R. S., CHILAUSSKY, R., "Learning by being told and learning from examples: an experimental comparison of two methods of knowledge acquisition in the context of developing an expert system for soybean diagnosis". In: *Policy Analysis and Information Systems*, v. 4, pp. 125-160, 1980.

- [28] O’RORKE, P., “A comparative study of inductive learning systems AQ11P and ID3 using a chess endgame test problem”. ISG 82-2, Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign, Urbana, 1982.
- [29] WALD, A., “Sequential Analysis”, Wiley, New York, 1947.
- [30] KALBFLEISH, J., “Probability and Statistical Inference”, v. 2, ed. 1, NY, USA, Spring Verlag, 1979.
- [31] RAUBER, T. W., “A Toolbox for Analysis and Visualization of Sensor Data in Supervision”, In: *Proceedings of the International Conference on Fault Diagnosis*, Toulouse, France, April 1993.
- [32] DEVIJVER, P.A., KITTLER, J., “Pattern Recognition: A Statistical Approach”, Prentice/Hall Inc., London, 1982.
- [33] SAMMON, J. W., “A Non-linear Mapping for Data Structures Analysis”, IEEE Trans. of Computers.
- [34] BERLIOUX, P., BIZARD, B., “Algorithms 2: Data Structures and Search Algorithms, John Wiley & Sons, 1990.
- [35] Tou, J. T., Gonzalez, R. C., “Pattern Recognition Principles”, Addison-Wesley, 1974.
- [36] WANG, L., MENDELL, J. M., “Generating Fuzzy Rules from Numerical Data, with Applications”, In: *Report of University of Southern California*, Los Angeles, 1991.
- [37] COVER, T. M., “Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition”, IEEE Transactions on Electronic Computers, 14, 326-334, 1965.

- [38] PRAGER, R. W., FALLSIDE, F., "The Modified Kanerva Model for Automatic Speech Recognition", *Computer Speech and Language*, 3, 61-82, 1989.
- [39] RENALS, S. ROHWER, R., "Phoneme Classification Experiments Using Radial Basis Function", In: *Proceedings of the International Joint Conference on Neural Networks*, vol. I, pp. 461-468, 1989.
- [40] COX, E. D., "Fuzzy Logic for Business and Industry", 1ed., Rockland, Charles River Media Inc., 1995.
- [41] TUTHILL, S., "Human Knowledge, Machine Knowledge, and Knowledge-Based Systems", 1 ed., Tab Books Inc., 1990.
- [42] MICHALSKI, R. S., "A Theory and Metodology of Inductive Learning", ed. 1, California, 1983.

Apêndice I:

Tooldiag: Descrição do Sistema

```
>>>          TOOLDIAG          <<<
>>>          Pattern recognition toolbox      <<<
>>>-----<<<
(1)  Classifier model
(2)  Feature selection
(3)  Feature extraction
(4)  Classifier induction alias supervised leaning
(5)  Error Estimation alias cross validaton
(6)  Classify from idependent data
(7)  Sammon Plot
(8)  Miscellaneous
(9)  Interfacing
(10) Load universe
(11) Normalize data
(Q)  uit
```

Choice:

(1) Classifier Model:

Escolhe uma das várias arquiteturas de classificação. Alguns classificadores precisam de alguma especificação especial, por exemplo o K do K-Vizinho mais Próximo.

Assim na estimativa de erro e na classificação de dados independentes, utiliza-se a arquitetura de classificador escolhida.

----- SPECIFICATION OF CLASSIFIER MODEL -----

-
- (1) 1-Nearest Neighbor, using raw data ← Actual
 - (2) Quadratic Gaussian Classifier
 - (3) Radial Basis Function Network
 - (4) Parzen Window
 - (5) Q* algorithm
 - (6) “Your Classifier”

Choice:

- (1) 1-Nearest Neighbor, using raw data

Atribui a uma amostra desconhecida a classe a qual seu vizinho mais próximo pertence. A similaridade é definida por uma distância métrica, normalmente a norma Euclidiana. Se for considerado apenas um vizinho, temos um classificador de 1-Vizinho mais Próximo. Se K vizinhos são considerados, tem-se um classificador do K-Vizinho Mais Próximo. Essa classe é então atribuída à amostra desconhecida que tem a maioria entre os k vizinhos.

- (2) Quadratic Gaussian Classifier

A suposição básica é que os dados obedecem uma distribuição multidimensional normal. Tem um mau desempenho se as amostras de uma classe aparecem em mais de um agrupamento (distribuição multi-modal).

- (3) Radial Basis Function Network

Rede de Função de Base Radial. Um paradigma de classificador da área de redes neurais artificiais.

A rede FBR pode ser considerada como a notação gráfica para a solução generalizada do problema de regularização, como descrito em (Haykin, 1994). Ela consiste de três camadas: uma camada de entrada d-dimensional, onde d é a dimensão do espaço de procura, uma camada escondida, e uma camada de saída com uma saída para cada

classe. A camada escondida é formada por um conjunto de vetores de referência que representam a função de distribuição básica. Na forma generalizada, o número dessas unidades escondidas é consideravelmente menor que o número de pontos de dados. Sob essa condição, a rede de regularização pode ser interpretada como a solução para o problema de interpolação multivariável (Powel, 1987) que é dado pela superposição linear de um conjunto de funções de base, representadas por um subconjunto de pontos de dados.

Para cada vetor de entrada, a rede FBR fornece a resposta correspondente a sua saída, isto é, para cada classe separadamente. Pela comparação das ativações de saída para todas as classes, toma-se uma decisão sobre a classificação.

O mapeamento entrada-saída tem que ser “aprendido” primeiro. Aprender, nesse contexto, é equivalente a encontrar a superfície de um espaço multidimensional que fornece o “melhor ajuste” aos dados de treinamento, isto é a melhor aproximação possível para a função de densidade de probabilidade verdadeira da distribuição básica. Os parâmetros a serem aprendidos são:

- os pesos lineares associados com a camada de saída
- as posições das unidades escondidas (vetores de referência)
- as extensões das e unidades escondidas (vetores de referência)
- as extensões das funções de unidades (base) escondidas

Existem duas estratégias básicas de aprendizado:

- aprendizado supervisionado de todos os parâmetros, e.g. nas bases do algoritmo correção de erros (um procedimento de gradiente descendente), que é uma forma generalizada do bem conhecido algoritmo do Quadrado-Menos-Significativo (Least-Mean-Square)
- determina as posições e as extensões das funções de base primeiro, e então calcula os pesos lineares e a camada de saída pela aplicação da solução de regularização; os centros das unidades escondidas podem ser encontrados pelo algoritmo de agrupamento

Para conseguir alcançar resultados razoáveis, os seguintes parâmetros podem ser escolhidos:

agrupar os vetores de referência ? : yes

→ o conjunto dos vetores de referência é agrupado (algoritmo c-means)

número de vetores de referência : 20 por cento

→ o número de unidades escondidas fora do conjunto inteiro de vetores de treinamento; se 100 por cento é usado, a solução converge para a rede de regularização pura; tem que ser encontrado um compromisso entre um alto número de unidades escondidas necessárias para um alto grau de aproximação (referência a propriedade de melhor aproximação) e um pequeno número de unidades escondidas para evitar o mau-condicionamento e tempo de computação excessivo; valores razoáveis estão entre 10 e 30 por cento

algoritmo de treinamento : Regularização (1):

parâmetro de regularização: 0.5..2

→ Lambda; se esse parâmetro é considerado zero, obtém-se a matriz pseudoinversa

Penrose-Moore ($(G^T * G)^{-1} * G^T$); para qualquer valor de Lambda sendo maior que zero, o efeito atenuador reduz o risco de mau-condicionamento; se Lambda é escolhido como sendo alto, então a aproximação da função de densidade se torna atenuada também, e o erro de aproximação cresce; o valor ótimo para Lambda depende da forma com que os dados são distribuídos e se ocorrem superposições excessivas quando não se aplica atenuação adicional;

Valores pequenos desse parâmetro foram considerados úteis.

Variâncias ajustadas individualmente ? : yes

fator : 0.1..0.3

→ a variância de cada unidade escondida depende da distância para o vetor de referência mais próximo (hidden unit), o fator de parâmetro determina a quantidade de superposição; se o fator é considerado maior que 0.5, então a superposição se torna crítica; valores razoáveis são, portanto, os menores que 0.5

variâncias ajustadas individualmente ? : no

variância independente da distância : 10

→ esse parâmetro é considerado arbitrário para um valor fora do intervalo $(0..10^5)$, para que a função de unidade escondida sejam estendidas ou encolhidas como desejado; se o mau-condicionamento deve ser evitado, valores altos deveriam optar por um algoritmo de treinamento : Aprendizado da Correção de Erros (2)

pesos da taxa de aprendizado : 0.0005..0.001

extensões da taxa de aprendizado : 1

centros da taxa de aprendizado : 0.005

→ as taxas de aprendizado determinam o quanto algoritmo se aproxima rápido do ótimo desejado; entretanto, se as taxas de aprendizado são consideradas valores altos, o algoritmo pode facilmente tornar-se instável; isso porque valores pequenos devem ser preferidos; por outro lado, se as taxas de aprendizado são escolhidas propositamente pequenas, então o algoritmo pode ficar cravado num mínimo local.

O aprendizado de correção de erro é muito sensível às escolhas das taxas de aprendizado. Os parâmetros podem ser ajustados um por um, isto é, primeiro apenas os pesos, depois extensões e centros, novamente os pesos lineares, etc. O algoritmo de aprendizado de correção de erro é implementado apenas a título de ilustração, e só pode ser usado no método de substituição de estimativa de erro, e apenas se o classificador foi incluído antes (opção do menu principal, “Classifier induction alias supervised learning”).

À parte do problema central, o cálculo dos pesos lineares, a escolha das variâncias ótimas deve ser vista como a mais importante, uma vez que ela regula a quantidade de superposição. A escolha das variâncias ótimas é considerada mais crucial que uma grande quantidade de atenuação alcançada por um parâmetro de regularização alto.

Existe a possibilidade de verificar se os parâmetros apropriados foram escolhidos tais que o mau-condicionamento não ocorre. Usando a opção 4 do menu principal “Classifier induction alias supervised learning” e escolhendo “Verbose learnig of decision structure”, a seguinte saída deve aparecer na tela:

checking the matrix inversion :

Condition Number of the Green's Matrix : 50.415134

DET=4571886595.471016

Deviation from Identity matrix = 0.000000

No caso ideal, o número condição da matriz de Green (estritamente falando o inverso de $(G^T * G + \text{Lambda} * G_0)$) se torna unitário, ao passo que o determinante vai para o

infinito. Esse caso nunca vai ocorrer, entretanto uma vez que as funções de base Gaussianas têm valores diferentes de zero em cada ponto do espaço de características. Por outro lado, se o número condição se aproxima da unidade, então as funções de base tomam mais e mais a forma aguda, assim as funções de densidade aproximada se tornam mais próximas de zero nas regiões entre os pontos de referência. Como pode ser concluído, deveria existir um ótimo para a quantidade de superposição. Infelizmente, o número-condição não indica se uma boa aproximação é alcançada ou não; se mostra apenas o mau-condicionamento é provável de acontecer, isto é, para que extensão as colunas da matriz são ortogonais umas às outras. Uma vez que a multiplicação de uma matriz com o seu inverso fornece a matriz identidade, a exatidão da inversão pode ser verificada computando-se o desvio do produto $(G^{-1} * G)$.

(4) Parzen window

Um método clássico a partir do reconhecimento de padrão estatístico. Uma estimativa não-paramétrica das funções de densidade de probabilidade.

Uma simples amostra tomada a partir de um conjunto de treinamento descreve um tipo de experiência que, estritamente falando, é válida apenas para um ponto distinto, no espaço n-dimensional em que ela foi observada. Essa validade limitada localmente é trazida por uma função de propagação do ponto para uma certa vizinhança local. Assim, de uma forma simples o efeito da interpolação é alcançado, limitando ao mesmo tempo o efeito extrapolador de uma forma facilmente controlável. Fazendo a soma das funções de atenuação em cada ponto, a abordagem total transforma a representação amostrada de uma certa função em uma versão contínua atenuada.

(5) Q* algorithm

Um algoritmo caseiro para o agrupamento de classes. Cada classe deve ter vários agrupamentos. Cada agrupamento é representado por um protótipo. Nenhuma suposição sobre a distribuição de probabilidade dos dados é necessária.

A idéia original do Q* algorithm: Dino Coltuc

(6) “Your Classifier”

Projete sua própria arquitetura de classificador para o aprendizado supervisionado e o integre no módulo de extração de características e estimativa de erro. Isso permite a comparação do desempenho de classificação do seu algoritmo com outras arquiteturas de classificador.

Choice:

(2) Feature Selection

Seleção de características = redução da dimensão pela deleção de características.

Diferentes estratégias de procura são combinadas com diferentes critérios de seleção.

Exceto pelas funções auxiliares, como carregar e salvar características selecionadas.

----- FEATURE SELECTION -----

----- Search Strategy -----

- (1) Best Features
- (2) Sequential Forward Search
- (3) Sequential Backward Search
- (4) Branch and Bound
- (5) Exhaustive Search

(T)ools and old algorithms

(Q)uit

Choice:

(1) Best Features

Essa estratégia de procura assume que todas as características são independentes.

Analisa uma característica depois da outra por um critério de seleção univariável.

Depois classifica seguindo esse critério. Escolha as primeiras N características desse ordenamento como as selecionadas.

(2) Sequential Forward Search

--- Sequential Forward Search alias Forward Stepwise Selection ---

Do grupo de características disponíveis teste todas as características JUNTO com as já selecionadas. Junte essa característica com as já selecionadas que deram o melhor critério (junto com as selecionadas). Remova-a do grupo de características ainda disponíveis.

Comece com um conjunto vazio como o já selecionado e todas as características como as disponíveis.

(3) Sequential Backward Search

--- Sequential Backward Search alias Backward Stepwise Elimination ---

Delete aquela característica das disponíveis que deram o mais baixo critério até que o número até que o número de características desejado é alcançado. Comece com todas as características disponíveis.

(4) Branch and Bound

Para critério de seleção com propriedade de monotonicidade, essa técnica de procura de 'Branch and Bound' (Ramificar e Limitar) pode ser aplicada. Implicitamente todos os subconjuntos possíveis (de tamanho 'd') fora de todo o conjunto de características (de tamanho 'D') são analisados, sem procurar explicitamente todos os 'd' sobre 'D' possibilidades.

(5) Exhaustive Search

Faça uma procura exaustiva de todos os subconjuntos possíveis, procurando todas as 'd' sobre 'D' combinações. É bastante possível que ocorra uma explosão combinatória.

A procura pode ser feita para um número incremental de características que permitem a você encontrar o ótimo global do critério de seleção. Por exemplo, ele permite encontrar o erro estimado mais baixo para todas as características e todas as combinações possíveis para um certo número de características. Por exemplo para 4 características os seguintes conjuntos são testados:

1 característica: {1}, {2}, {3}, {4},

2 características: {1,2}, {1,3}, {1,4}, {2,3}, {2,4}, {3,4}

3 características: {1,2,3} {1,2,4}, {2,3,4}

4 características: {1,2,3,4}

(T)ools and old algorithms

Originalmente apenas três algoritmos de seleção de características foram implementados. A área de aplicação era o monitoramento e a supervisão dos processos de produção, usando o aprendizado indutivo das situações do processo. A referência desses três algoritmos originais é:

Grupo de critério de seleção: Erro estimado mínimo.

Isso é uma combinação da seleção de característica e da estimativa de erro. Escolha aquela(s) característica(s) que prognosticam o erro mais baixo para um certo tipo de classificador. Por exemplo, pode-se usar um classificador 1-Vizinho Mais Próximo para prognosticar o erro do conjunto de características atual. Dependendo do erro adiciona-se ou omite-se certas características.

Grupo de critério de seleção: Distância Interclasses.

Aquelas características que são boas, maximizam a distância entre as amostras das classes (e minimiza a distância dentro da mesma classe). Portanto você está procurando características onde as amostras de uma classe estão longe das amostras de uma outra classe no conjunto de características atual. A medida da distância é crucial para o cálculo da distância média.

Grupo de critério de seleção: Distância Probabilística.

Escolha aquelas características que separam bem funções de densidade probabilística condicionadas às classes. Características que causam uma superposição mínima das classes são boas para seleção, aquelas que causam uma alta superposição não são adequadas para seleção.

Choice: 3

(3) Feature Extraction

Extração de Características = redução da dimensão pela combinação de todas as características

– linear

Uma matriz que mapeia lineamente as amostras originais para as novas amostras com uma dimensão menor é calculada. As novas características são combinações lineares das velhas características.

– Não-lineares

Geram um novo conjunto de dados no qual as amostras originais são mapeadas para as novas amostras em um espaço dimensional menor. O mapeamento é executado de uma forma não-linear.

----- FEATURE EXTRACTION -----

----- Method -----

- (1) Linear discriminant analysis
- (2) Principal Component Analysis alias Karhunen-Loeve Expansion
- (3) Sammon Mapping (a non linear method)

----- Tools for linear extractors -----

- (S)ave linear fearture extractor matrix to file
- (L)oad linear feature extractor matrix from file
- S(H)ow actual matrix
- (E)xtract samples in memory with actual linear extractor

Choice:

(1) Linear Discriminant Analysis

Executa análise Discriminativa Linear para a extração de características.

(2) Principal Component Analysis alias Karhunen-Loeve Expansion

Análise da Principal Componente se aproxima da redução de dimensão.

Também conhecida como Transformação Karhunen-Loeve.

(3) Sammon Mapping (a non linear method)

Mapeia os pontos de dados D-dimensional para 2 ou 3 dimensões, o usuário pode ver o relacionamento estrutural dos dados. Dados que são 2-D ou 3-D a priori podem ser visualizados diretamente. Apenas as características selecionadas são usadas .

Classes altamente superpostionadas não fornecem muita informação a partir do gráfico.

Nesse caso deve ser repetido o experimento apenas com duas classes, por exemplo. Então uma separação melhor pode ser visível. É um método não-linear.

Choice:

(4) Classifier induction alias supervised learning

Induz um classificador seguindo o atual modelo de classificador.

Também conhecido como aprendizado supervisionado.

Todas as amostras de treinamento são usadas para gerar uma estrutura de decisão.

Dependendo do tipo de classificador, a estrutura de decisão pode ser bem diferente. Por exemplo, para o Classificador Gaussiano Quadrático, o classificador consiste de vetores de média, matrizes de covariância inversas e alguns parâmetros auxiliares. No caso da Rede de Função de Base Radial, o classificador consiste de um conjunto de vetores de referência e pesos associados.

(5) Error estimation alias cross validation

Use os métodos de estimativa de erro – cognome validação cruzada (cross validation) ou contador de erro (error counting). O desempenho de classificação do classificador atual é estimado usando o conjunto de dados atual. O conjunto de dados total é dividido em um conjunto de treinamento e um conjunto de teste. O papel do conjunto de treinamento e do conjunto de teste é trocado por algumas técnicas de estimativa de erro (Leave-and-Out e Rotation).

(6) Classify from independent data

Usa o classificador atual para identificar dados a partir de um arquivo.

Os dados de teste devem ser independentes dos dados de treinamento.

(7) Sammon Plot

Mapeia pontos de dados D-dimensionais para pontos de dados d-dimensionais com $d < D$.

Bom para extração de características e visualização.

a) Extração de características:

O mapeamento de Sammon pode ser usado para a extração de características, isto é a redução de dimensão não-linear de D para d sob a preservação do relacionamento estrutural dos dados.

b) Visualização:

Se os dados D-dimensionais são mapeados em 2 ou 3 dimensões, o usuário pode ver o relacionamento estrutural dos dados. Dados que são 2 ou 3-D a priori podem ser visualizados diretamente. Apenas características selecionadas são usadas.

Classes altamente superposicionadas não fornecem muita informação a partir do gráfico.

Nesse caso deveria-se repetir os experimentos com apenas duas classes, por exemplo. Então uma separação melhor pode ser visível.

(8) Miscellaneous

Algumas funções úteis.

(9) Interfacing

Geram arquivos de entrada de dados para outros pacotes.

(10) Load Universe

a) Carrega o universo a partir de um diretório.

Um diretório vai ser escaneado a procura de arquivos que contenham as amostras para uma classe. O diretório pode conter apenas arquivos de dados. O formato de um arquivo de dados é como se segue:

```
CLASS_NAME  
DIMENSION  
NUMBER_OF_SAMPLES
```

sample1
sample2
...
sampleN

b) Carrega o universo a partir de um arquivo.

O arquivo deve ter o seguinte formato:

DIMENSION

<Feature_1> ... <Feature_D> CLASS_NAME

<Feature_2> ... <Feature_D> CLASS_NAME

(11) Normalize data

Os valores das características diferentes podem variar consideravelmente.

Por exemplo, se a primeira característica fosse algum comprimento, medido em metros, então seu valor poderia estar entre 0.0001 e 0.0005.

A segunda característica poderia ser algum peso medido em gramas e está entre 2500 e 12000. Se isso é uma distância Euclidiana calculada entre duas amostras a segunda característica vai ser enfatizada por natureza, embora a primeira pudesse ser eventualmente mais discriminativa para classificação. Portanto, a escala das características deveria ser mais ou menos a mesma para todas as características para não ocultar uma informação importante de uma característica somente porque seu valor é pequeno.

A normalização às vezes destrói informações valiosas.

Apêndice II

Pré-Processamento dos Dados - Pacote DTMining

Como foi colocado anteriormente, o Tooldiag só trabalha com dados completos. Como frequentemente esses dados não se apresentam num formato adequado para ser trabalhado pelo Tooldiag, desenvolvi alguns programas básicos reunidos num pequeno pacote. Essa ferramenta simples é responsável pelo pré-processamento dos dados. A figura mostra a tela de entrada do pré-processador.

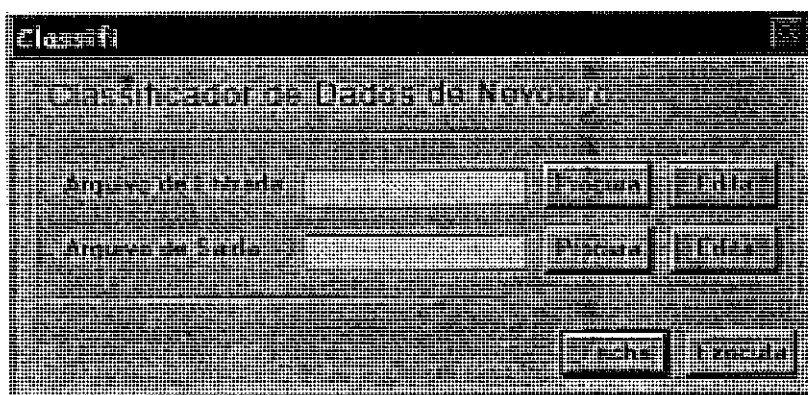


Na sequência de desenvolvimento dos programas surgiram alguns problemas e alguns desses programas tiveram que ser modificados e alguns outros ainda criados.

O primeiro programa a ser desenvolvido foi o programa Galeão. O programa Galeão classificava um arquivo de dados em três classes pré-definidas: pré-nevoeiro, nevoeiro, e normal. A classificação era feita da seguinte forma: o programa ia lendo todas as linhas do arquivo de dados. A cada leitura igual a 4 no campo de dados 'tempo presente', classificava-se o registro corrente como nevoeiro, retornava-se aos cinco registros imediatamente anteriores e os classificava como pré-nevoeiro, todos os outros registros que não estivessem em nenhuma dessas situações eram classificados como 'normal'. Como os dados devem estar completos para o processo de classificação era necessário primeiramente eliminar os registros incompletos. Dessa

forma, porém, não se poderia mais manter a classificação das cinco linhas anteriores às linhas de nevoeiro como linhas correspondentes à pré-nevoeiro pois não se poderia saber ao certo quais linhas foram retiradas do banco de dados. A solução encontrada foi completar “artificialmente” os campos incompletos, realizar a classificação – o que só é possível fazer com certeza porque o campo ‘tempo presente’ está completo – e posteriormente transferir essa coluna (campo) relativo à classificação para o banco de dados incompleto. Agora, é possível eliminar as linhas (registros) incompletas da base de dados sem afetar o encadeamento do processo de classificação.

A nova versão do primeiro programa, o ‘classifi’, traduz a solução encontrada para o problema de classificação dos dados e veio substituir o programa Galeão. Como foi descrito anteriormente toda a classificação gira em torno do campo de dados ‘tempo presente’. As modificações ocorrem a partir da identificação de um valor 4 nesse campo. O registro que contém esse valor será classificado como nevoeiro. As cinco linhas anteriores porém serão classificadas independentemente como pré-nevoeiro de 1h, pré-nevoeiro de 2h, pré-nevoeiro de 3h, pré-nevoeiro de 4h e pré-nevoeiro de 5h. Todos os registros que fogem a essas classificações também são classificados como normal. Depois de feita essa classificação, como no programa Galeão, não há mais a necessidade de trabalhar com o arquivo em ordem para avaliar o padrão de pré-nevoeiro. Fazendo-se a análise de pré-nevoeiro hora a hora, porém, tem-se uma análise mais precisa, que deve tender mais facilmente a um padrão.

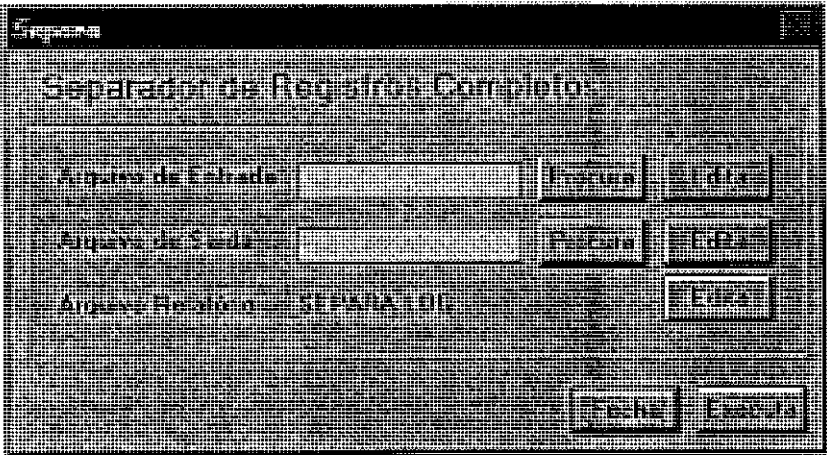


O segundo programa chamado ‘analisa’ foi desenvolvido para verificar quais atributos da base de dados se encontram incompletos, e o quanto em percentual esses atributos se apresentam incompletos, gerando um relatório. Baseado nesse relatório,

pode-se escolher, baseado na heurística ou em bom senso, um limite acima do qual esses atributos podem ser desconsiderados. Por exemplo: uma coluna que se encontra 99% incompleta não poderá ajudar muito numa classificação baseada em atributo. A escolha do valor percentual a ser utilizado como limite se torna meio duvidosa quando o percentual de incompletos cai para 67% ou 55%. Deve-se ou não utilizar esses atributos? Para tentar avaliar se os dados restantes (os 33% ou 45%) são suficientes para conduzir a classificação é preciso separá-los e para isso foi desenvolvido o 3º programa.

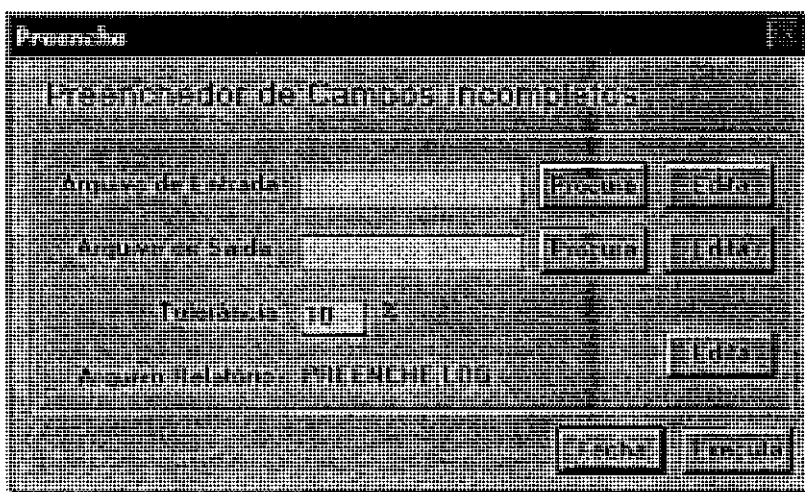


O terceiro programa, o 'separa', foi desenvolvido para verificar e separar em um arquivo novo todos os registros completos de um banco de dados. A figura mostra o menu referente a esse programa.

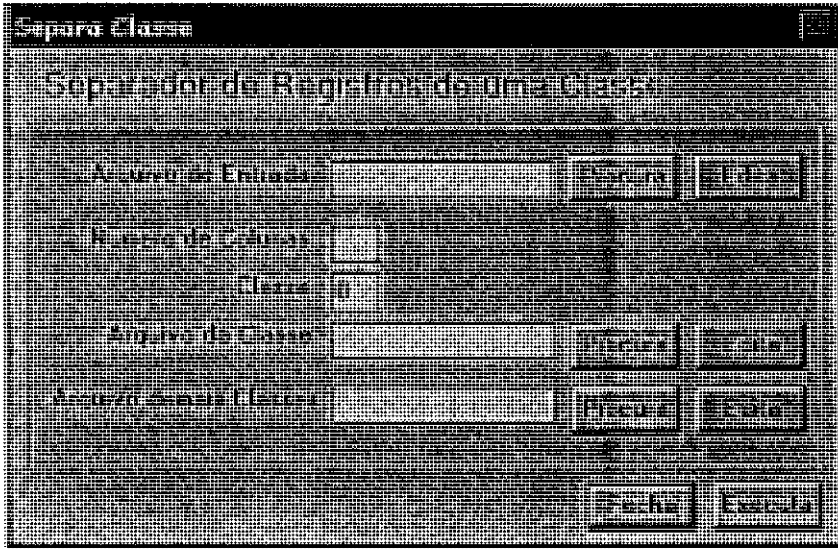


O quarto programa, o 'preenche', tem o objetivo de tentar preencher campos de dados (atributos) incompletos pela comparação registro a registro, campo a campo. Quando, numa comparação entre dois registros, o programa encontra todos os campos

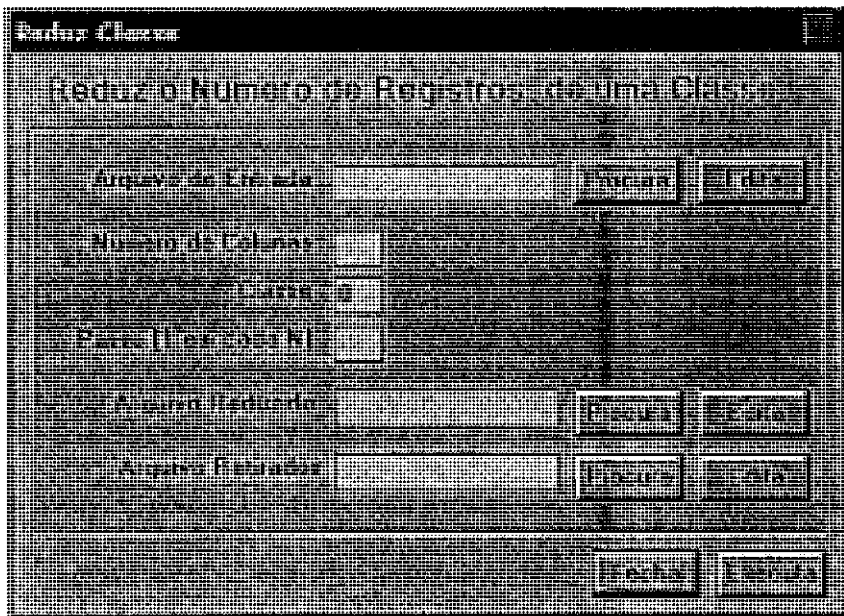
completos diferindo no máximo do valor da tolerância (definida pelo usuário), ele preenche imediatamente os campos incompletos com os valores dos mesmos campos do registro completo. Numa primeira versão, essa tolerância era aplicada sobre o valor corrente lido no campo. Foi observado porém, que para alguns campos como 'hora', em alguns horários do dia essa variação era de 100% (de 1:00h para 2:00hs) e resolveu-se calcular essa tolerância como um percentual da faixa de variação do atributo. No caso do atributo 'hora', que vai de 0:00h até 23:00hs, seria de 23 a variação da faixa e a tolerância seria independente da leitura do dado corrente.



O quinto programa, o 'retira', foi desenvolvido para separar todas as amostras de uma determinada classe de um arquivo de dados. Isso funciona quando se quer, por exemplo agregar mais um arquivo à um arquivo maior em que já existem amostras suficientes de uma determinada classe. Assim, antes de juntá-los executa-se o programa no arquivo novo, eliminando-se as amostras da classe da qual não se deseja mais amostras.



Acontece, porém, que às vezes só se observa que existem mais amostras que o necessário de uma determinada classe depois que os arquivos já foram agregados. Nesse caso, pode-se executar o sexto programa, o 'reduz', sobre a base de dados total. Porém, não deve ser interessante retirar todas as amostras de uma determinada classe, assim o programa permite não só escolher a classe, como a proporção de amostras a serem eliminadas pelo programa.



Um oitavo programa, o 'coluna', foi ainda adicionado ao pacote. Esse programa foi desenvolvido visando a agilização na eliminação de colunas incompletas da base de dados sem a necessidade de abrir uma planilha.



Nota-se que os programas de classificação de dados apresentam dificuldades para manusear e classificar dados quando existem poucas amostras para cada classe. Quanto maior o número de amostras maior deveria ser a probabilidade de se conseguir fazer uma boa classificação. No entanto, isso nem sempre é verdade, pois, para alguns casos, mesmo que se tenha um grande número de amostras, elas não levam a um padrão definido. Isso pode ocorrer porque existem classes superpostas; porque as classes não foram bem definidas, ou seja, o que foi apresentado primeiramente como dados de treinamento para o programa possui erros; ou ainda, faltam informações (atributos ou leituras) que sejam fundamentais para a classificação.

Apêndice III

Relatório de Compilação:

O compilador utilizado para criar o executável do Tooldiag na versão MS-DOS foi o WATCOM. Durante o processo de criação do executável surgiram alguns problemas que são relatados a seguir:

- DIR.C → dir.h teve que ser substituído por direct.h.
- DIAGNOS.C → alterado de 2 para 4 o 'define sigint' por causa do signal.h do WATCOM.
- DIR.C → alterado código-fonte para adaptar as rotinas FINDFIRST e FINDNEXT do WATCOM.
- DEF H → retirado o comentário da linha #DEFINE ONLY_RAND
- RANDOM → alterada a linha i=RANDOM(MAX) para i=RAND() para compatibilizar como o WATCOM.
- FEATEXTR.C, PARZEN.C, QSTAR.C, RBF.C, SAMMON.C, MATRIX.C → alterados os nomes das funções MATRIX_ALLOC e MATRIX_FREE para MMATRIX_ALLOC e MMATRIX_FREE respectivamente para diferenciar das funções com o mesmo nome.