Pós-Graduação em Ciência da Computação

Desenvolvimento de uma Plataforma Híbrida para Descoberta de Conhecimento em Bases de Dados

por

Bruno Pereira de Amorim

Dissertação de Mestrado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://www.cin.ufpe.br/~posgraduacao

RECIFE, MARÇO/2004

Bruno Pereira de Amorim

Desenvolvimento de uma Plataforma Híbrida para Descoberta de Conhecimento em Bases de Dados

Este trabalho foi submetido à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Germano Crispim Vasconcelos

Co-orientadora: Profa. Dra. Lourdes Mattos Brasil

Recife,
29 de Março de 2004

A Deus, pela sua infinita graça e bênçãos.
A meus pais, Adauto e Aparecida.

Agradecimentos

Constitui, sem dúvida, uma tarefa difícil expressar em tão poucas palavras a enorme gratidão ao apoio e incentivo recebidos ao longo do desenvolvimento desta dissertação. Meus sinceros agradecimentos a todos que contribuíram de forma direta ou indireta para a realização deste trabalho. Em especial gostaria de agradecer:

A Deus por ter me concedido sabedoria, discernimento, paciência e confiança para concluir este trabalho;

A Nossa Senhora pela intercessão e proteção;

A meus pais, Adauto e Aparecida, que, sem medir esforços, nem sacrifícios, me proporcionaram as melhores condições, visando a minha educação e formação profissional;

As minhas irmãs, Islânia e Ismênia, e familiares, em especial a tio Paulo, tia Adaura e Bebeu, pelo apoio que sempre me deram na busca da realização dos meus sonhos;

A Ismênia pelo companheirismo, apoio e paciência durante todo o desenvolvimento deste trabalho;

A Renata pela contribuição científica;

Aos amigos Cledja, Hélio, Tirza, Carla e Fred pela amizade e pelo conforto nos momentos mais difíceis;

Aos pesquisadores Mark Craven, Nikola Kasabov, Ruiping Li e Masao Mukaidono pela atenção que tiveram no decorrer do desenvolvimento deste trabalho;

Ao professor Germano Vasconcelos pela orientação durante o desenvolvimento deste trabalho;

A professora Lourdes Brasil, que, apesar da distância, mostrou-se extremamente atenciosa; pela amizade e contribuições científicas, fundamentais no meu ingresso ao Mestrado;

Ao Conselho Nacional de Pesquisa (CNPq) pelo apoio financeiro à atividade de pesquisa.

Resumo

As Redes Neurais Artificiais (RNA) têm sido utilizadas com sucesso em tarefas como mapeamento de funções complexas e reconhecimento de padrões. Este sucesso é resultado da habilidade das RNA em realizar cálculos com dados complexos ou imprecisos, aprender a partir de exemplos, generalizar a informação aprendida, extrair padrões e descobrir tendências. Apesar destas vantagens, geralmente não é muito fácil obter explicações de como uma RNA representa a solução de um problema. Devido a esta limitação, as RNA têm sido consideradas inadequadas para serem utilizadas em aplicações de KDD (*Knowledge Discovery in Databases*) em que o usuário deseja saber o raciocínio usado pela rede para obter uma dada conclusão.

Sistemas Híbridos Inteligentes (SHI) é uma abordagem de Inteligência Artificial que vem sendo bastante utilizada na resolução de problemas onde o emprego de uma única técnica não é suficiente para obter resultados satisfatórios. Tais sistemas se inspiram na integração de duas ou mais técnicas inteligentes com o intuito de suprir as limitações de cada técnica. A disseminação dos SHI tem contribuído para a emergência dos Sistemas Neurais Híbridos (SNH). O principal foco de pesquisa em SNH tem sido a integração de RNA, técnica fortemente baseada em dados, com técnicas que utilizam representação simbólica, como Lógica *Fuzzy* e algoritmos simbólicos convencionais. Os Sistemas Neuro-*Fuzzy* são um exemplo de SNH que combinam sistemas conexionistas com sistemas *fuzzy*. Nestes sistemas é aplicado algum método de extração de regras que permite a representação do conhecimento incorporado pela rede numa forma compreensível. Além das técnicas de extração de conhecimento simbólico associadas aos Sistemas Neuro-*Fuzzy*, diversas técnicas têm sido propostas para outros modelos neurais.

Esta dissertação tem como principais objetivos investigar o paradigma dos Sistemas Neuro-*Fuzzy* e as técnicas de extração de conhecimento simbólico de RNA como uma opção para tornar as RNA mais adequadas ao processo de KDD; e, como resultado da investigação, modelar e implementar uma ferramenta de *software*, a *Neural Mining*, baseada na abordagem neural híbrida. A ferramenta *Neural Mining* integra, em um único ambiente, o modelo *Perceptron* Multicamadas (*Multilayer Perceptron* - MLP); os modelos neuro-*fuzzy* FWD (*Feature-Weighted Detector*) e FuNN (*Fuzzy Neural Network*), juntamente com suas técnicas de extração de regras; e a técnica TREPAN (*Trees Parroting Networks*), que representa o conhecimento incorporado por uma RNA na forma de uma árvore de decisão. Os modelos e técnicas são avaliados e comparados com relação à capacidade de generalização e compreensibilidade do conhecimento extraído. Além da análise nas etapas de mineração de dados e apresentação do conhecimento, também são investigadas duas técnicas de seleção de atributos: a técnica do modelo FWD e através da árvore de decisão gerada por TREPAN.

A investigação experimental é realizada usando uma base de dados de um problema real e de larga escala no domínio de análise de crédito ao consumidor. Como os resultados obtidos demonstram que os ganhos decorrentes do uso de modelos neuro-*fuzzy* e técnicas de extração de conhecimento simbólico de RNA são bastante significativos, ao final da investigação, considerando as vantagens de cada modelo e técnica, são propostas duas soluções neurais híbridas para o processo de KDD.

Abstract

Artificial Neural Networks (ANN) have successfully been used in tasks as the mapping of complex functions and pattern recognition. This success is due to the ANN ability to make calculations of complicated and undetermined data, learn from examples, generalize the learned information, extract patterns and discover tendencies. Despite these advantages, it is generally not very easy to obtain explanations of how an ANN represents the solution of a problem. Due to this limitation, ANN have been considered inadequate to be used in applications of KDD (Knowledge Discovery in Databases) where the user wants to know the reasoning used by the network to obtain a conclusion.

Intelligent Hybrid Systems (IHS) is an approach of Artificial Intelligence that has been used in resolution of problems where the application of one isolated technique is not sufficient to get satisfactory results. These systems are based on the integration of two or more intelligent techniques, aiming at overcoming the limitations of each technique. The dissemination of IHS has contributed to the emergence of Hybrid Neural Systems (HNS). The main research focus in HNS has been the integration of ANN, strongly data-based technique, with techniques that use symbolic representation, such as Fuzzy Logic and conventional symbolic algorithms. Neuro-Fuzzy Systems are an example of HNS that combine connectionist systems with fuzzy systems. In these systems, some rule extraction technique is applied to represent the knowledge embedded by the neural network in a comprehensible way. In addition to the rule extraction techniques of Neuro-Fuzzy Systems, several techniques for extraction of symbolic knowledge from other neural models have been proposed.

The main goals of this work are investigating the paradigm of Neuro-Fuzzy Systems and the techniques for extraction of symbolic knowledge from ANN as an option to become ANN more adequate to the KDD process; and, as result of the investigation, modeling and developing a software tool, Neural Mining, based on the hybrid neural approach. The Neural Mining tool integrates, in an only environment, the neural model MLP (Multilayer Perceptron), the neuro-fuzzy models FWD (Feature-Weighted Detector) and FuNN (Fuzzy Neural Network), together with their rule extraction techniques, and the technique TREPAN (Trees Parroting Networks) that represents the knowledge embedded by an ANN in the form of a decision tree. The models and techniques are evaluated with respect to their generalization performance and comprehensibility of the extracted knowledge. In addition to the analysis in the data mining step and knowledge presentation step, two feature selection techniques are also investigated: the FWD technique and using the decision tree extracted by TREPAN.

The experimental investigation is performed using a large scale credit assessment database of a real problem. As the results acquired demonstrate that the benefits obtained from the use of neuro-fuzzy models and techniques for extraction of symbolic knowledge from ANN are highly expressive, in the end of the investigation, considering the advantages of each model and technique, two hybrid neural solutions are proposed to the KDD process.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

AG – Algoritmos Genéticos

ANFIS – *Adaptive Network-based Fuzzy Inference System* ou *Adaptive Neuro Fuzzy Inference System*

AREFuNN – *Aggregated Rule Extraction from a FuNN*

ARTMAP – *Adaptive Resonance Theory-supervised Predictive Mapping*

BIO-RE – *Binarized Input-Output Rule Extraction*

BRE – *Black-box Rule Extraction*

CANFIS – *Coactive Network-Fuzzy Inference System*

CART – *Classification and Regression Trees*

CEBP – *Constrained Error Backpropagation*

CMAC – *Cerebellar Model Arithmetic Computer*

COA – *Center of Area*

COG – *Center of Gravity*

DDA – *Dynamic Decay Adjustment*

DEDEC – *Decision Detection*

EDDA – *Extended Dynamic Decay Adjustment*

EN – *Explanation Facility*

FCM – *Fuzzy C-Means*

FNES – *Fuzzy Neural Expert System*

FNN – *Fuzzy Neural Network*

FP – Falso Positivo

Full-RE – *Full Rule Extraction*

FuNN – *Fuzzy Neural Network*

FWD – *Feature-Weighted Detector*

GG – *Gath-Geva*

GK – *Gustafson-Kessel*

GL – *Generalization Loss*

IA – Inteligência Artificial

KBANN – *Knowledge-Based Artificial Neural Networks*

KDD – *Knowledge Discovery in Databases*

KNN – *K-Nearest Neighbor*

LazyPOP – *Lazy Pseudo Outer-Product*

LRE – *Link Rule Extraction*

LRU – *Locally Responsive Units*

LSM – *Least Squares Method*

MLP – *Multilayer Perceptrons*

MOM – *Mean of Maximum*

MPG – *Modus Ponens* Generalizado

NEFCLASS – *Neuro Fuzzy Classification*

NEFCON – *Neuro Fuzzy Control*

NEFPROX – *Neuro Fuzzy Function Approximation*

OLAP – *On-Line Analytical Processing*

Partial-RE – *Partial Rule Extraction*

POP – *Pseudo Outer-Product*

POPFNN – *Pseudo Outer-Product based Fuzzy Neural Network*

QPROP – *Quick Propagation*

RBF – *Radial Basis Function*

REFuNN – *Rule Extraction from a FuNN*

RNA – Rede Neural Artificial

ROC – *Receiver Operating Characteristics*

RPROP – *Resilient Propagation*

SGBD – Sistemas Gerenciadores de Banco de Dados

SHI – Sistemas Híbridos Inteligentes

SM-BP – Treinamento por modos deslizantes

SNH – Sistemas Neurais Híbridos

SOM – *Smallest of Maximum*

SSE – *Sum of Squared Error*

TREPAN – *Trees Parroting Networks*

VP – Verdadeiro Positivo

Capítulo 1
Introdução

A principal área de concentração desta dissertação é a aplicação de técnicas de extração de conhecimento simbólico de Redes Neurais Artificiais (RNA) no processo de Descoberta de Conhecimento em Bases de Dados (*Knowledge Discovery in Databases* - KDD). Tais técnicas possibilitam que o conhecimento obtido pela rede neural seja representado numa forma compreensível. Deste modo, as RNA tornam-se mais adequadas para serem aplicadas ao processo de KDD, onde, na maioria das aplicações, a representação do conhecimento é um fator primordial. Como resultado da investigação realizada nesta dissertação, foi desenvolvida uma ferramenta de *software*, a *Neural Mining*, responsável por capturar dados já consolidados, aplicar algoritmos de mineração baseados na abordagem neural híbrida e apresentar o conhecimento descoberto em forma de regras. A ferramenta foi aplicada em um estudo de caso de análise de crédito, um problema real e de larga escala.

1.1. Motivação

Os avanços na coleta, armazenamento, processamento e distribuição de dados, aliados ao uso extensivo de Sistemas Gerenciadores de Banco de Dados (SGBD) e da tecnologia de *Data Warehousing* [Kimball et al. 1998], têm aumentado drasticamente a magnitude dos dados armazenados em diversos domínios de aplicação. No ramo de negócios e comércio, por exemplo, o processo de informatização tem ocorrido desde o simples controle de pagamentos até modernos e integrados ambientes de computação que oferecem total automação dos serviços. Conseqüentemente, as corporações passaram a armazenar, ao longo dos anos, um volume enorme de dados em seus repositórios sobre as mais diversas tarefas e operações ligadas às suas atividades.

Nestes grandes volumes de dados escondem-se informações valiosas que podem ser de importância estratégica para a tomada de decisões numa empresa ou grupo de interesse. Estas informações podem permitir, entre outras coisas, a detecção de fraudes no uso de linhas telefônicas e transações financeiras, a identificação de nichos de consumidores propensos ao uso de produtos ou serviços, e a previsão do comportamento de ações e índices financeiros em bolsas de valores.

Esta abundância de dados tem excedido a capacidade de análise humana, tornando completamente impraticável a análise manual dos dados em diversos domínios de aplicação [Fayyad et al. 1996a]. Além disso, os métodos tradicionais utilizados para manipular esses dados podem criar apenas relatórios informativos, não sendo capazes de analisar o conteúdo dos dados e destacar o

conhecimento importante. Estes métodos geram soluções para questões pré-estabelecidas, em que o usuário sabe de antemão o que deseja obter.

Os bancos de dados multidimensionais (*Data Warehouse*) e seu método de consulta *On-Line Analytical Processing* (OLAP) [Kimball et al. 1998] têm ajudado a melhorar a capacidade de análise dos dados, mas ainda não são suficientes [Bigus 1996]. Ferramentas de análise de dados adicionais são requeridas para realizar tarefas mais complexas, tais como classificação (encontrar um relacionamento entre os atributos de entrada e as classes para realizar a predição de novos exemplos), *clustering* (identificar um conjunto de grupos ou *clusters*, inicialmente desconhecidos, de tal forma que exemplos com características similares sejam agrupados no mesmo *cluster*) e caracterização de mudanças temporais.

Este cenário tem contribuído para o surgimento de técnicas e ferramentas capazes de analisar grandes volumes de dados e extrair conhecimento de forma inteligente e automática (ou semi-automática). Algumas dessas técnicas e ferramentas são resultantes do emergente campo de KDD [Fayyad et al. 1996a].

KDD é um processo de descoberta de características e relacionamentos nos dados que possam se transformar em conhecimento útil, estratégico e compreensível ao ser humano. O processo de KDD consiste das seguintes etapas [Han & Kamber 2001]:

- Integração dos dados: combinação das diversas fontes de dados;
- Seleção dos dados: recuperação dos dados relevantes;
- Limpeza dos dados: tratamento dos dados inconsistentes e incompletos;
- Transformação dos dados: mapeamento dos dados em formas apropriadas para a etapa de mineração;
- Mineração de dados: aplicação de métodos inteligentes para extrair padrões de comportamentos ou regularidades;
- Avaliação dos padrões: identificação de padrões interessantes [Hilderman & Hamilton 1999];
- Apresentação do conhecimento: uso de técnicas de visualização [Fayyad et al. 2001] e representação do conhecimento para apresentar o conhecimento minerado ao usuário.

Na etapa de mineração de dados, diversas técnicas podem ser empregadas. As RNA têm sido utilizadas com sucesso para executar tarefas como o mapeamento de funções complexas e reconhecimento de padrões [Amorim et al. 2001]. Este sucesso é resultado da habilidade das RNA de realizar cálculos com dados complexos ou imprecisos, aprender a partir de exemplos, generalizar a informação aprendida, extrair padrões e descobrir tendências. Tais características permitem que uma rede neural treinada para representar um conjunto de dados possa ser usada para fornecer novas projeções.

Embora as RNA tenham se mostrado uma técnica eficiente para a solução de um grande número de aplicações, não é correto afirmar que elas são suficientes para resolver qualquer tarefa. As RNA apresentam várias limitações que inviabilizam seu uso exclusivo para a solução de uma quantidade significativa de problemas [Braga et al. 2000].

Uma das maiores limitações das RNA é a dificuldade de gerar uma explicação de como a rede representa a solução de um problema, pois o conhecimento aprendido é representado de forma implícita pela topologia, valores dos pesos e limiares da rede; e o espaço de representação é dividido em regiões complexas por meio da combinação de várias funções matemáticas [Azevedo et al. 2000].

Como na maioria das aplicações reais, os usuários desejam saber o raciocínio utilizado pelo sistema para obter uma dada conclusão, é importante que a rede neural seja capaz de fornecer uma explicação de suas saídas [Amorim et al. 2001] [Huang & Xing 2002]. Desta forma, cada vez mais, pesquisadores, projetistas e usuários acreditam que todo o potencial das RNA não poderá ser completamente explorado enquanto não for acrescentado a estes modelos um mecanismo que permita a explicação de suas decisões. Esta necessidade é essencial principalmente em aplicações onde a segurança na operação é um aspecto crucial. Este é o caso de problemas como controle de usinas nucleares, diagnóstico médico e análise de crédito [Faifer & Janikow 1999].

Diversas abordagens têm sido propostas para superar a dificuldade das RNA de gerar representações compreensíveis da solução de um problema.

Sistemas Híbridos Inteligentes (SHI) é uma das abordagens de Inteligência Artificial que vem sendo bastante utilizada na resolução de problemas onde o emprego de uma única técnica não é suficiente para obter resultados satisfatórios [Goonatilake & Khebbal 1995]. Tais sistemas se inspiram na integração de duas ou mais técnicas inteligentes, como RNA, Algoritmos Genéticos e Sistemas *Fuzzy*, com o intuito de suprir as limitações de cada técnica [Silva et al. 2002]. Neste caso, a combinação de duas ou mais técnicas pode levar a uma solução mais robusta e eficiente. No entanto, a utilização de SHI não resulta necessariamente em uma melhoria de desempenho do sistema em comparação com as soluções obtidas pelas técnicas que foram combinadas quando utilizadas isoladamente.

Como resultado da rápida disseminação dos SHI, no contexto de RNA, um dos temas de pesquisa mais investigados atualmente é o desenvolvimento de Sistemas Neurais Híbridos (SNH) [Braga et al. 2000]. O principal foco de pesquisa em SNH tem sido o de combinar RNA, técnica fortemente baseada em dados, com técnicas que utilizam representação simbólica, como Sistemas *Fuzzy*, algoritmos simbólicos convencionais, Raciocínio Baseado em Casos e Sistemas Tutores.

As combinações de RNA com Sistemas *Fuzzy* geralmente procuram simular Lógica *Fuzzy* utilizando RNA. Nestes casos, a rede neural é empregada para ajustar um conjunto de parâmetros de um sistema *fuzzy*, sendo utilizada no lugar do sistema de inferência [Braga et al. 2000].

Um sistema *fuzzy* pode ser utilizado como sistema de apoio à decisão capaz de representar o conhecimento de especialistas e algumas características do raciocínio humano, e interpolar decisões a partir de entradas vagas ou imprecisas. Deste modo, um sistema *fuzzy* pode ser considerado como centrado no homem [Evsukoff & Almeida 2003].

Os benefícios oferecidos pelos Sistemas *Fuzzy* permitem a representação de conhecimento na forma de regras *fuzzy* Se-Então. No entanto, problemas surgem quando conceitos *fuzzy* têm que ser representados por graus de pertinência, que influenciam a conduta do sistema. Por esta razão, é bastante promissor aplicar procedimentos de aprendizagem que determinem estes valores automaticamente [Azevedo et al. 2000].

Considerando as características destes dois paradigmas, diversos pesquisadores vêm procurando reunir as potencialidades dos Sistemas *Fuzzy* e RNA nos chamados Sistemas Neuro-*Fuzzy* [Evsukoff & Almeida 2003]. Esta integração permite que o conhecimento adquirido pela rede neural seja representado na forma de regras *fuzzy* Se-Então, regras iniciais sejam usadas para definir a arquitetura inicial da rede e refinadas durante o treinamento, e os graus de pertinência sejam determinados automaticamente [Taha & Ghosh 1999]. Deste modo, o conhecimento adquirido pela rede passa a ser representado numa forma compreensível.

Entre os modelos neuro-*fuzzy* mais recentes, pode-se destacar o *Feature-Weighted Detector* (FWD) [Li et al. 2002]. Este modelo visa resolver dois dos principais problemas em Reconhecimento de Padrões (seleção de atributos relevantes e classificação de padrões) e possibilita a extração de regras *fuzzy* Se-Então.

Embora muitos esforços tenham sido realizados, bons resultados não foram alcançados com técnicas propostas anteriormente que lidem simultaneamente com classificação de padrões e seleção de atributos [Bezdek 1981] [Kuncheva 1992]. Por esta razão, o modelo FWD é potencialmente promissor para ser aplicado em problemas de reconhecimento de padrões.

Entre os modelos neuro-*fuzzy* mais consolidados, pode-se destacar a rede *Fuzzy Neural Network* (FuNN) [Kasabov et al. 1997], que representa um sistema *fuzzy* através de uma arquitetura neural adaptativa. FuNN é uma rede *Multilayer Perceptron* (MLP) que utiliza um algoritmo de treinamento baseado no *Backpropagation* [Rumelhart et al. 1986], através do qual as funções de pertinência dos predicados *fuzzy* e as regras *fuzzy* inseridas antes do treinamento são ajustadas aos dados.

A arquitetura da rede FuNN facilita a inserção e extração de regras *fuzzy*, e a aprendizagem a partir dos dados, possibilitando, desta forma, a utilização de duas fontes de dados (base de dados e regras iniciais). Este modelo também permite a aplicação de vários métodos de adaptação [Kasabov et al. 1997].

Os algoritmos de aprendizagem da rede FuNN usados em conjunto com técnicas de extração e inserção de regras demonstram que esta é uma abordagem promissora para construir sistemas inteligentes e adaptativos de processamento de informação que podem ser aplicados em diversos domínios. As áreas de aplicação incluem processamento de sinal e imagem, modelagem e predição de séries temporais, controle adaptativo, mineração de dados e aquisição de conhecimento [Kasabov et al. 1997].

Além das técnicas de extração de regras dos Sistemas Neuro-*Fuzzy*, diversas técnicas que permitem a extração de conhecimento simbólico de outros modelos neurais têm sido propostas. *Explanation Facility* (EN) [Pau & Gotzche 1992], *Trees Parroting Networks* (TREPAN) [Craven & Shavlik 1996], *Decision Detection* (DEDEC) [Tickle et al. 1996], *Binarized Input-Output Rule Extraction* (BIO-RE) [Taha & Ghosh 1999], *Partial Rule Extraction* (Partial-RE) [Taha & Ghosh 1999], *Full Rule Extraction* (Full-RE) [Taha & Ghosh 1999] e NeuroRule [Lu et al. 1995] são alguns exemplos.

TREPAN é uma das técnicas mais difundidas. Esta técnica faz perguntas a uma RNA treinada utilizando o conjunto de exemplos usado na fase de treinamento da rede e outro gerado a partir da distribuição dos dados de treinamento [Craven & Shavlik 1996]. As respostas são utilizadas para a construção de uma árvore de decisão que aproxima o conhecimento representado pela rede. Nenhuma restrição é feita com relação à arquitetura da rede ou ao algoritmo de treinamento aplicado. Na

realidade, esta técnica é bastante genérica, podendo ser aplicada a uma ampla variedade de modelos de aprendizagem, não sendo restrita apenas às RNA. Além disso, ela apresenta boa escalabilidade para problemas com bases de dados e RNA extensas [Craven 1996].

As principais vantagens provenientes da utilização de técnicas de extração de regras de RNA são [Nobre et al. 1998] [Brasil 1999] [Haykin 1999] [Taha & Ghosh 1999] [Braga et al. 2000]:

- ♣ Facilidade de compreensão: A representação do conhecimento em forma de regras apresenta uma grande proximidade com a linguagem humana. Por esta razão, a compreensão do conhecimento minerado é facilitada;
- ♣ Mineração de dados: Novos relacionamentos e características são descobertos, permitindo a formulação de novas teorias;
- ♣ Facilitar a aceitação pelo usuário: Uma das características mais importantes dos sistemas simbólicos é a capacidade de explicar suas decisões. Esta capacidade facilita a aceitação dos sistemas pelos usuários. No caso das RNA, a ausência de regras explícitas que justifiquem suas decisões dificulta sua aceitação e restringe o seu escopo de aplicação;
- ♣ Melhorar a generalização da rede: As regras podem ser usadas para descobrir as circunstâncias em que a rede pode cometer erros de generalização ou identificar regiões no espaço de entradas que não estejam suficientemente representadas no conjunto de treinamento. Nestes casos, dados adicionais podem ser usados para melhorar o desempenho da rede;
- ♣ Integração com Sistemas Simbólicos: Além de possibilitarem uma descrição simbólica concisa e apurada da rede, as regras facilitam a integração da rede com outros sistemas baseados em conhecimento, como os Sistemas Especialistas;
- ♣ Redefinir a rede: As regras podem ser utilizadas para verificar a adequação da arquitetura neural;
- ♣ Minimizar o problema de aquisição de conhecimento e refinar o conhecimento inicial adquirido dos especialistas do domínio;
- ♣ Facilitar a identificação dos atributos relevantes: As regras extraídas possibilitam um melhor entendimento do relacionamento entrada-saída, ajudando a identificar os atributos mais relevantes.

1.2. Objetivos

1.2.1. Objetivos Gerais

Motivada pela importância da utilização do processo de KDD na solução de problemas do mundo real e pelas vantagens obtidas através da aplicação de técnicas de extração de conhecimento simbólico de RNA, esta dissertação tem como objetivos gerais:

- ♣ Investigar o paradigma dos Sistemas Neuro-*Fuzzy* e as técnicas de extração de conhecimento simbólico de RNA como uma alternativa para tornar as RNA mais adequadas ao processo de KDD;

- Como resultado da investigação, modelar e implementar uma ferramenta de *software*, a *Neural Mining*, que integre, em um único ambiente, os modelos neuro-*fuzzy* e as técnicas de extração de conhecimento simbólico de RNA mais promissores.

1.2.2. Objetivos Específicos

Baseando-se nos objetivos gerais desta dissertação, os seguintes objetivos específicos podem ser definidos:

- Investigar o paradigma dos Sistemas Neuro-*Fuzzy* e as técnicas de extração de conhecimento simbólico de RNA;
- Modelar e implementar uma ferramenta de *software*, a *Neural Mining*, que abranja os modelos neuro-*fuzzy* e as técnicas de extração de conhecimento simbólico de RNA mais promissores;
- Aplicar um problema de larga escala no domínio de análise de crédito como ambiente de teste para a ferramenta *Neural Mining* e os algoritmos investigados;
- Avaliar e comparar os resultados alcançados por cada modelo e técnica implementados na ferramenta *Neural Mining* e identificar suas vantagens e deficiências quando aplicados na solução de um problema real e de larga escala.

  Os modelos neurais serão avaliados e comparados considerando os seguintes aspectos: desempenho em relação à generalização e capacidade de gerar conhecimento compreensível através de suas técnicas de extração de regras. O desempenho dos classificadores em relação à generalização será avaliado observando as taxas de classificação no conjunto de teste (total e por classe), analisando as curvas ROC (*Receiver Operating Characteristics*) [Provost & Fawcett 1997] e o impacto das decisões dos classificadores no contexto específico da aplicação investigada (análise de crédito ao consumidor). A compreensibilidade do conhecimento extraído será avaliada analisando a facilidade de interpretação e aplicação do conhecimento descoberto. Outro aspecto considerado na avaliação do conhecimento extraído é a precisão. Além da análise nas etapas de mineração de dados e apresentação do conhecimento, também serão investigadas duas técnicas de seleção de atributos: a técnica da rede FWD e através da árvore de decisão gerada pela técnica TREPAN;
- Ao final da investigação experimental, considerando as vantagens de cada modelo e técnica, propor soluções neurais híbridas para o processo de KDD.

1.3. Justificativas

Embora KDD seja uma área recente com muitos aspectos que ainda precisam ser pesquisados em profundidade, uma grande quantidade de ferramentas de *software* de KDD e mineração de dados já foi desenvolvida [Han & Kamber 2001]. Inicialmente, as ferramentas eram mais utilizadas em ambientes de pesquisa e experimentais. Nos últimos anos, tem-se observado um rápido crescimento de ferramentas sofisticadas voltadas para o meio empresarial [Goebel & Gruenwald 1999]. O domínio de aplicação destas ferramentas está constantemente evoluindo devido ao desenvolvimento de novos sistemas, incorporação de novas funcionalidades aos sistemas existentes e proposição de novos métodos de mineração de dados. Apesar dessa evolução, poucas ferramentas têm adotado a

abordagem dos SNH [Goebel & Gruenwald 1999] [Data 2003] [Data 2004]. Portanto, a principal vantagem da ferramenta desenvolvida nesta dissertação, a *Neural Mining*, é a capacidade de integrar vários métodos de aprendizagem neural híbrida num único ambiente. Além disso, *Neural Mining* foi projetada para solução de problemas de larga escala, apresenta uma interface amigável e, por ter sido desenvolvida em Java, possui todas as vantagens de um sistema orientado a objetos, tais como compatibilidade, reusabilidade e fácil manutenção [Deitel & Deitel 2001].

Os modelos neurais abordados pela ferramenta são: a rede MLP com o algoritmo de aprendizagem *Backpropagation* [Rumelhart et al. 1986] e as redes neuro-*fuzzy* FWD e FuNN. Com relação às técnicas de extração de conhecimento simbólico, foram implementadas as técnicas AREFuNN (*Aggregated Rule Extraction from a FuNN*) [Kasabov et al. 2001] e REFuNN (*Rule Extraction from a FuNN*) [Kasabov 1996], que extraem regras *fuzzy* Se-Então de uma rede FuNN; a técnica do modelo FWD, que extrai regras *fuzzy* Se-Então; e a técnica TREPAN, que extrai uma árvore de decisão de qualquer modelo.

O modelo MLP foi escolhido para ser aplicado em dois contextos: em conjunto com a técnica TREPAN e, separadamente, por ser um modelo tradicional, para comparar sua performance com os modelos neuro-*fuzzy* FWD e FuNN.

A motivação para investigar os modelos FWD e FuNN, e a técnica TREPAN se baseou no fato destes métodos serem bastante promissores para serem aplicados ao processo de KDD.

A rede FWD apresenta uma característica inovadora de englobar, dentro de uma mesma estrutura, três etapas do processo de KDD: seleção de dados, mineração de dados e apresentação do conhecimento. A investigação da rede FWD também é motivada pelo fato de que existem poucos resultados experimentais demonstrando sua viabilidade em aplicações reais [Li et al. 2002] [Amorim et al. 2003].

O modelo FuNN foi escolhido por apresentar as seguintes características: permite o uso de limiares no processo de extração de regras, possibilitando a redução do número de regras e condições na parte antecedente das regras; permite a inserção e o refinamento de regras usadas para definir a arquitetura inicial da rede, possibilitando o uso de duas fontes de dados (base de dados e regras iniciais); e tem apresentado resultados satisfatórios em diversos domínios de aplicação.

As técnicas de extração de regras dos modelos FuNN (AREFuNN e REFuNN) e FWD foram implementadas por possibilitarem a representação do conhecimento incorporado por estes modelos na forma de regras *fuzzy* Se-Então.

A técnica TREPAN foi escolhida por apresentar as seguintes vantagens: representa o conhecimento numa forma compreensível; pode ser utilizada em aplicações que envolvem tanto atributos discretos como contínuos; é capaz de produzir árvores de decisão sucintas de RNA grandes aplicadas em domínios de larga escala; é capaz de produzir árvores de decisão precisas, compreensíveis e com alto nível de fidelidade; é escalável em relação ao tamanho da base de dados, complexidade dos modelos aprendidos e tempo de execução (o usuário tem controle sobre a complexidade da árvore); e é genérica em sua aplicabilidade (não requer a utilização de um procedimento especial de treinamento e não impõe restrições sobre o modelo utilizado como oráculo).

Um problema de larga escala no domínio de análise de crédito será aplicado como ambiente de teste para a ferramenta desenvolvida e os algoritmos investigados. Este domínio foi escolhido por se tratar de um problema de larga escala, envolver dados reais com múltiplos atributos relacionados e ser de interesse de diversas instituições e empresas. Tais características permitirão a verificação da viabilidade prática dos modelos e técnicas selecionados.

1.4. Estrutura da Dissertação

Esta dissertação está organizada em oito capítulos e dois apêndices. O Capítulo 1 apresenta uma visão geral da área de KDD, dos Sistemas Neuro-*Fuzzy* e das técnicas de extração de conhecimento simbólico de RNA, e as motivações, objetivos e justificativas desta dissertação. O Capítulo 2 descreve detalhadamente todas as etapas do processo de KDD e destaca algumas áreas de aplicação de KDD. O Capítulo 3 aborda os principais aspectos referentes às RNA e a viabilidade de aplicá-las no processo de KDD. O Capítulo 4 aborda os Sistemas *Fuzzy* e Neuro-*Fuzzy*. Este capítulo também apresenta o estado da arte e uma breve descrição da perspectiva histórica dos Sistemas Neuro-*Fuzzy*, enfatizando os modelos FWD e FuNN. O Capítulo 5 aborda os principais aspectos sobre as técnicas de extração de conhecimento simbólico de RNA e descreve detalhadamente o algoritmo TREPAN e as técnicas específicas dos modelos FuNN e FWD. O Capitulo 6 apresenta a ferramenta *Neural Mining*, desenvolvida nesta dissertação. O Capítulo 7 descreve o estudo de caso em análise de crédito aplicado à ferramenta. O Capítulo 8 apresenta as conclusões, contribuições e trabalhos futuros. Por fim, o Apêndice A aborda os principais conceitos da Lógica *Fuzzy* e o Apêndice B descreve os formatos dos arquivos utilizados pela ferramenta *Neural Mining* para ler dados de entrada ou gravar as saídas.

Capítulo 2

Descoberta de Conhecimento em Bases de Dados

2.1. Introdução

A disponibilidade de recursos computacionais de alto desempenho, a diminuição do custo de armazenamento, o poder de processamento dos computadores atuais e a informatização em diversos setores da sociedade têm contribuído para que grandes quantidades de dados sejam armazenadas em diversos domínios de aplicação. Muitos desses dados possuem informações valiosas que poderiam ser usadas para melhorar o processo de tomada de decisões.

Apesar desta abundância de dados, a aquisição de conhecimento ainda é um problema na maioria das corporações, pois à medida que o volume de dados aumenta, a capacidade de análise humana decresce acentuadamente [Witten & Frank 2000]. Além disso, o conhecimento que está escondido nos dados pode não ser obtido através de ferramentas tradicionais de análise de dados, projetadas para gerar relatórios simplificados, capazes de satisfazer as necessidades básicas dos usuários. Essa dificuldade gerou a necessidade de ferramentas de análise e extração do conhecimento capazes de responder a perguntas mais complexas e transformar grandes volumes de dados em conhecimento útil [Goebel & Gruenwald 1999].

Data Warehousing é considerada um dos primeiros passos no aprimoramento da análise de grandes quantidades de dados [Kimball et al. 1998]. O objetivo básico desta tecnologia é criar um repositório (*Data Warehouse*) contendo dados limpos, agregados e consolidados que possam ser analisados por ferramentas OLAP. Estas ferramentas apresentam facilidades para a realização de consultas complexas em bases de dados multidimensionais. Normalmente, as consultas são dirigidas pelos usuários, que possuem hipóteses a serem validadas ou, simplesmente, executam-nas aleatoriamente [Rezende et al. 2003]. Através de várias iterações, o usuário pode formular uma nova hipótese ou refinar a hipótese atual. Portanto, a análise via OLAP é um processo dedutivo cuja habilidade de descobrir padrões e tendências é limitada pelas hipóteses e perguntas feitas pelos usuários. Hipóteses importantes nem sempre são óbvias ou se originam de padrões escondidos nos dados. Além disso, quando o repositório contém um grande número de atributos, a análise pode se tornar complexa e demandar bastante tempo para descobrir boas hipóteses. Por estas razões, padrões escondidos podem não ser encontrados nessa abordagem.

Este cenário impulsionou o desenvolvimento de novos métodos computacionais de análise e extração automática (ou semi-automática) de conhecimento a partir de uma base de dados. Alguns desses métodos são resultantes do emergente campo de KDD, uma área de pesquisa multidisciplinar que incorpora técnicas utilizadas em áreas como Reconhecimento de Padrões, Inteligência Artificial, Aquisição de Conhecimento, Estatística, Banco de Dados, *Data Warehousing* e Visualização de Dados.

Diferente das ferramentas OLAP, ao invés de verificar padrões hipotéticos, o processo de KDD usa a própria base de dados para descobrir os padrões, caracterizando-se como um processo indutivo [Two Crows 1999]. Devido aos benefícios proporcionados por KDD, esta área vem atraindo, nos últimos anos, um enorme interesse de pesquisadores e desenvolvedores de soluções para problemas do mundo real.

Este capítulo aborda inicialmente os principais fatores que contribuíram para o surgimento da área de KDD. Em seguida, todas as etapas do processo de KDD são descritas de forma detalhada. Por fim, com o objetivo de fornecer uma visão prática da aplicação de KDD em problemas reais, são apresentadas algumas de suas principais áreas de aplicação.

2.2. O Processo de KDD

KDD é um processo automático (ou semi-automático) de descoberta de características e relacionamentos nos dados que sejam válidos e novos, e possam se transformar em conhecimento útil, estratégico e compreensível ao ser humano [Fayyad et al. 1996a].

O processo de KDD é interativo (várias decisões são tomadas pelos usuários durante o processo), iterativo (melhorias consecutivas podem ser realizadas a fim de obter melhores resultados) e composto de várias etapas [Fayyad et al. 1996b]. KDD é mais do que simplesmente aplicar técnicas de mineração de dados. Uma abordagem passo a passo deve ser adotada.

Existem diversas propostas para a divisão do processo de KDD. [Fayyad et al. 1996a] propõem um processo em nove etapas. [Weiss & Indurkhya 1998] sugerem quatro etapas. Esta dissertação considera a divisão descrita em [Han & Kamber 2001], onde o processo de KDD consiste das seguintes etapas: integração dos dados, seleção dos dados, limpeza dos dados, transformação dos dados, mineração de dados, avaliação dos padrões e apresentação do conhecimento. Além destas etapas, foi considerada uma etapa inicial que se refere à identificação do problema (Figura 2.1).

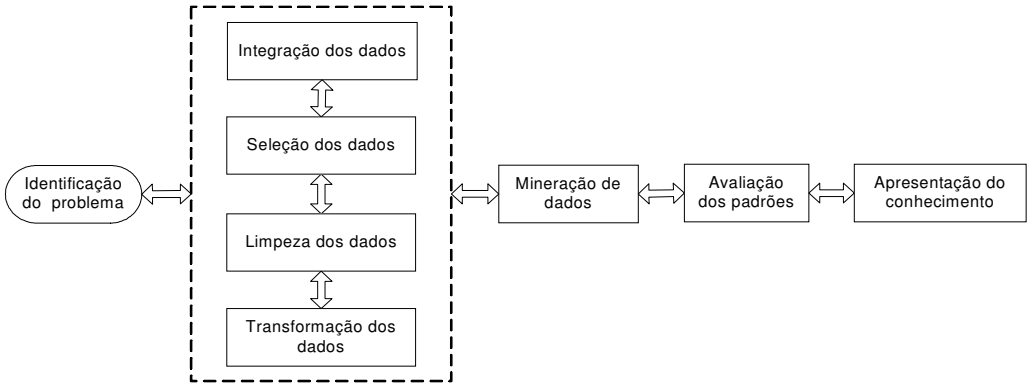

Figura 2.1 - Processo de KDD

É importante ressaltar que KDD é um processo contínuo, através do qual o conhecimento e o entendimento dos dados estão em constante melhoria [Adriaans & Zantinge 1996]. Além disso, sua aplicação não resulta na substituição dos especialistas do domínio. Ao invés disso, KDD deve ser usado como uma ferramenta para melhorar as atividades dos especialistas, confirmando observações empíricas e descobrindo novos padrões [Two Crows 1999].

As próximas seções descrevem detalhadamente todas as etapas do processo de KDD.

2.2.1. Identificação do Problema

Na primeira etapa do processo de KDD, identificação do problema, deve-se fazer um estudo sobre o domínio do problema investigado, sendo, portanto, essencial a participação de especialistas do domínio. Além de ser útil nesta fase, o conhecimento adquirido fornece um subsídio para as demais etapas, podendo auxiliar na definição dos valores válidos dos atributos, seleção dos atributos relevantes, escolha de um critério de preferência entre os modelos gerados, ajuste dos parâmetros do processo de mineração, validação do conhecimento minerado, geração de conhecimento inicial a ser fornecido como entrada para o algoritmo de mineração, entre outras atividades [Rezende et al. 2003].

Alguns aspectos que devem ser considerados nessa fase são: necessidades dos usuários, principais objetivos do processo, *hardware* e *software* existentes, critérios de desempenho, métricas de avaliação, nível de compreensibilidade ao final do processo, relação entre simplicidade e precisão do conhecimento extraído, e período de tempo em que as informações descobertas se tornam obsoletas, definindo, desta forma, quando o processo deve ser refeito [Adriaans & Zantinge 1996]. Outro aspecto que deve ser considerado é o entendimento dos dados que serão utilizados durante o processo de KDD. Este entendimento pode ser alcançado através da elaboração de um relatório com a descrição dos dados [Two Crows 1999]. O relatório deve conter propriedades como número de atributos e exemplos; total de dados ausentes e diferentes; nome, domínio, descrição, valores máximo e mínimo dos atributos; e descrição das fontes de dados.

Regras de negócios, meta-conhecimento, histogramas e análise através de ferramentas tradicionais de consulta são exemplos de técnicas que podem ser aplicadas durante esta etapa a fim de obter informações mais detalhadas sobre os dados [Adriaans & Zantinge 1996]. Técnicas de visualização [Fayyad et al. 2001] e ferramentas OLAP [Kimball et al. 1998] também podem ser usadas, uma vez que padrões, relacionamentos, dados não usuais (*outliers*) e valores ausentes normalmente são mais fáceis de serem percebidos quando mostrados graficamente.

2.2.2. Integração dos Dados

Uma vez estabelecidos os requisitos, o próximo passo é coletar os dados necessários.

Na maioria dos casos, os dados estão armazenados em bases de dados operacionais e a coleta nem sempre é uma tarefa trivial, pois pode envolver conversões de dados de baixo nível, tais como de arquivos texto para tabelas relacionais ou de sistemas hierárquicos para sistemas relacionais. Além disso, os dados usados pelos setores da organização podem variar com relação à qualidade e ao intervalo de atualizações [Adriaans & Zantinge 1996]. Em algumas situações, parte dos dados pode não pertencer à organização, sendo necessário adquiri-los de bases públicas ou proprietárias [Two Crows 1999].

Para que seja possível a realização do processo de KDD em ambientes onde os dados estão armazenados em locais e formatos diferentes, é necessário realizar a integração dos dados. Nesta etapa, devem-se considerar problemas como atributos representando o mesmo conceito com nomes diferentes, causando inconsistência e redundância; valores diferentes para uma mesma entidade; técnicas diferentes de codificação, representação e medição; atributos que podem ser inferidos de outros, o que pode retardar e prejudicar o processo de descoberta devido à redundância; e diferentes graus de agregação dos dados [Han & Kamber 2001].

Normalmente, os dados disponíveis após a integração não estão em um formato adequado para a extração de conhecimento. Desta forma, torna-se necessária a aplicação de métodos para tratamento dos dados antes da etapa de mineração. A execução das transformações deve ser guiada pelos objetivos do processo de KDD, a fim de que o conjunto de dados gerado apresente as características necessárias para que tais objetivos sejam alcançados [Rezende et al. 2003]. As etapas de seleção, limpeza e transformação dos dados são executadas para este fim.

Apesar de não se tratar de requisito fundamental, a construção de um *Data Warehouse* [Kimball et al. 1998] pode reduzir drasticamente a complexidade e duração do processo de KDD, pois algumas tarefas de preparação dos dados já terão sido previamente realizadas [Monteiro 1999].

2.2.3. Seleção dos Dados

A seleção dos dados ocorre em duas dimensões [Monteiro 1999]: representatividade dos exemplos e relevância dos atributos.

Apesar dos métodos de aprendizagem tentarem selecionar os atributos relevantes e ignorar os irrelevantes (não afetam a descrição do conceito alvo) ou redundantes (não adicionam qualquer informação para a descrição do conceito alvo), na prática, a performance dos métodos normalmente pode ser melhorada pela seleção de atributos [Witten & Frank 2000].

Embora o especialista do domínio possa escolher os atributos mais relevantes, essa tarefa pode ser difícil e demandar tempo. Por esta razão, diversos estudos têm sido direcionados para o desenvolvimento de métodos automáticos de análise de relevância de atributos. O objetivo destes métodos é aplicar alguma medida que seja usada para quantificar a relevância de um atributo com relação a uma classe ou conceito (Ex:. ganho de informação, índice de Gini e coeficiente de correlação) [Han & Kamber 2001]. Exemplos de métodos de análise de relevância de atributos são:

- Algumas técnicas, como as RNA, possuem a capacidade de exibir, após o processo de aprendizagem, o quanto cada atributo contribuiu para a solução de um problema [Gately 1996];
- Técnicas estatísticas, como Análise de Componentes Principais, podem ser utilizadas [Haykin 1999]. O objetivo dessas técnicas é extrair um conjunto de medidas características a partir dos atributos;
- Uma árvore de decisão é construída a partir da base de dados e os atributos que não aparecerem na árvore são considerados irrelevantes [Han & Kamber 2001];
- Métodos de aprendizagem baseados em exemplos também podem ser usados para selecionar atributos. Neste caso, deve-se amostrar aleatoriamente exemplos do conjunto de treinamento e verificar se os exemplos próximos pertencem à mesma classe. Se um exemplo próximo

pertencer à mesma classe e tiver um valor diferente para um determinado atributo, o atributo parece ser irrelevante e um peso associado ao atributo deve ser decrementado. Por outro lado, se um exemplo próximo pertencer a uma classe diferente e tiver um valor diferente para um determinado atributo, o atributo parece ser relevante e seu peso é incrementado. Depois de repetir este processo várias vezes, a seleção pode ser realizada, ou seja, apenas os atributos com pesos positivos são escolhidos [Witten & Frank 2000];

- Aplicar a indução construtiva, em que um novo atributo é criado a partir de outros. Caso os atributos originais utilizados na construção do novo atributo não estejam presentes no novo modelo, eles podem ser removidos. A utilização de indução construtiva pode aumentar consideravelmente a qualidade do conhecimento extraído [Rezende et al. 2003].

Além de melhorar o desempenho dos métodos de aprendizagem, a remoção de atributos irrelevantes resulta numa representação mais compacta do conceito alvo, focando a atenção do usuário para os atributos mais importantes [Witten & Frank 2000].

Com relação ao número de exemplos, a redução deve ser realizada através da geração de amostras representativas dos dados, ou seja, as características do conjunto de dados original devem estar presentes nas amostras geradas. Se a amostra não for representativa ou a quantidade de exemplos for insuficiente para caracterizar os padrões escondidos nos dados, os modelos obtidos não representarão a realidade. Além disso, uma quantidade relativamente pequena de exemplos pode resultar em *overfitting* (ajuste excessivo do modelo ao conjunto de treinamento, afetando a capacidade de generalização) [Fayyad et al. 1996b].

A abordagem mais utilizada para redução de exemplos é a amostragem aleatória, pois esta tende a gerar amostras representativas e, na maioria dos problemas cuja base de dados é muito grande, não resulta em perda de informação [Two Crows 1999].

A seleção de dados não é uma tarefa trivial e deve ser realizada com bastante cuidado, pois os conceitos que podem ser aprendidos pelos métodos de mineração são extremamente dependentes da qualidade dos dados [Monard & Baranauskas 2003].

2.2.4. Limpeza dos Dados

Depois que os dados são integrados e selecionados, o próximo passo é a limpeza.

É bastante comum que as bases de dados de problemas reais contenham dados imperfeitos. Isto ocorre devido a falhas no processo de geração, aquisição, integração e transformação dos dados. Nestes casos, diz-se que existe ruído nos dados [Monard & Baranauskas 2003].

Dados sem qualidade podem prejudicar a mineração de dados, resultando numa saída não confiável. Embora alguns métodos de mineração possuam procedimentos para lidar com dados ruidosos e incompletos, eles nem sempre são robustos [Han & Kamber 2001]. Portanto, é essencial realizar um processamento nos dados antes da etapa de mineração.

O objetivo da etapa de limpeza dos dados é minimizar o ruído presente na base de dados, tratando os dados ausentes (*missing data*), com valores fora dos limites aceitáveis ou muito diferentes dos valores usuais (*outliers*), incorretos, redundantes e inconsistentes. Não é realístico, entretanto,

achar que é possível remover todo o ruído antecipadamente. Algumas anomalias só serão descobertas posteriormente. Isto demonstra que KDD é um processo iterativo [Adriaans & Zantinge 1996].

Diversas técnicas podem ser empregadas no tratamento de valores inválidos e ausentes [Two Crows 1999] [Han & Kamber 2001]. A remoção de exemplos, substituição pela média (atributos contínuos), substituição pela moda (atributos nominais), substituição pela mediana (atributos ordinais), substituição pelo valor mais freqüente, atribuição de um valor baseado na distribuição dos valores presentes, preenchimento com um identificador ou utilização de um modelo de predição para prever o valor a ser preenchido são alguns exemplos. A construção de um modelo preditivo normalmente gera resultados melhores do que um simples cálculo, pois utiliza mais as informações presentes nos dados para prever os valores. No entanto, requer muito mais tempo. A utilização de um identificador não é recomendada, pois o algoritmo de mineração pode identificá-lo como um conceito. Em problemas de classificação, se for utilizada a substituição pela média, valor mais freqüente, moda, mediana ou valor baseado em uma distribuição, pode-se considerar apenas os valores dos exemplos que pertencem à mesma classe. Com relação à remoção de exemplos, se todos os exemplos que possuem valores ausentes forem removidos, a base de dados resultante pode ficar muito pequena ou apresentar uma representação imprecisa da base original.

Com relação aos *outliers*, podem-se aplicar técnicas como *clustering* e inspeção para identificá-los. Na técnica de *clustering*, os valores são organizados em grupos e os valores isolados são considerados *outliers* [Han & Kamber 2001].

Em algumas aplicações, como detecção de fraudes, a falta de informação ou a presença de *outliers* pode ser uma indicação valiosa de padrões interessantes, por isso não deveria ser removida ou preenchida com algum valor [Adriaans & Zantinge 1996].

A melhor maneira de evitar ruído é coletar os dados de forma apropriada. Neste caso, os sistemas que possuem conhecimento semântico dos atributos devem realizar checagem de consistência [Adriaans & Zantinge 1996].

2.2.5. Transformação dos Dados

Nesta fase, os dados são transformados em formas apropriadas para o algoritmo de mineração [Han & Kamber 2001]. Caso os dados não estejam num formato adequado, a aprendizagem pode ser prejudicada, resultando em modelos não representativos [Gately 1996]. Por esta razão, as transformações devem ser realizadas com bastante cuidado, garantindo que as informações presentes nos dados brutos permaneçam. Conhecimento *a priori* sobre o domínio do problema pode ajudar bastante na determinação de que transformações são mais adequadas.

Exemplos de técnicas que podem ser aplicadas nesta etapa são:

- Discretização: Reduz o número de valores de um atributo contínuo dividindo a amplitude do atributo em intervalos. Os rótulos dos intervalos substituem os valores. Esta técnica melhora o entendimento, mas pode causar perda de informação. A discretização é um caso particular da generalização, técnica em que os dados primitivos são substituídos por conceitos de ordem superior organizados em uma hierarquia [Han & Kamber 2001];

- Suavização: Diminui o número de valores de um atributo sem discretizá-lo. Nesse método, os valores do atributo são agrupados, mas, diferente da discretização, cada grupo é substituído por um valor numérico que o represente (Ex.: média) [Rezende et al. 2003];
- Agregação: Permite a redução do número de atributos, normalmente é resultante da aplicação de operações sobre os atributos (Ex.: soma, produto) e pode gerar resultados mais precisos [Two Crows 1999];
- Normalização: Assegura que os valores de um atributo estejam dentro de um determinado intervalo, minimizando, desta forma, os problemas oriundos do uso de unidades e dispersões distintas entre os atributos.

  A normalização pode melhorar a precisão e a eficiência dos algoritmos de mineração que lidam com medidas de distância (Ex.: RNA, classificadores *K-Nearest Neighbor* ou KNN, e *clustering*), pois quando os atributos são escalados para mesma ordem de magnitude, uma medida de distância confiável entre os diferentes exemplos é obtida. Caso contrário, os atributos que possuem uma ordem de magnitude maior exercerão mais influência [Adriaans & Zantinge 1996].

  Existem várias formas de realizar a normalização [Monteiro 1999]. A norma máxima é a mais comum, pois requer pouco processamento. Ela consiste em dividir cada valor pelo valor máximo. Existem diversas variações desta forma, pois ela não garante que os dados ocupam todo o intervalo [0, 1]. Como em alguns casos isso não é desejável, é mais usual utilizar a Equação 2.1.

$$V_n = k(V - V_{min})/(V_{max} - V_{min}) \quad (2.1)$$

  onde k é um fator de multiplicação, V_n representa o valor normalizado, V o valor original, V_{min} e V_{max} os valores mínimo e máximo do conjunto, respectivamente. Para k igual a 1, esta forma garante que os valores estarão no intervalo [0,1];
- Atributo do tipo data: Em geral, os valores dos atributos do tipo data são transformados em valores numéricos que representam, por exemplo, a quantidade de meses ou anos [Bigus 1996];
- Padronização: É bastante usual que o valor de um atributo nominal apresente erros tipográficos, sendo descrito por nomes diferentes. Neste caso, é importante que a lista de possíveis valores de cada atributo seja cuidadosamente examinada e, em seguida, os valores que representam o mesmo conceito sejam substituídos por um único valor [Witten & Frank 2000];
- Codificação binária: Atributos discretos e simbólicos podem ser representados através de uma codificação binária. O desafio é representar estes atributos de tal forma que o algoritmo de mineração seja capaz de diferenciar e relacionar a magnitude e ordenação dos valores. A decisão a ser tomada depende do problema investigado e da semântica do atributo. Exemplos de codificação binária são [Monteiro 1999]:

  1 de N: São utilizados *N* bits, onde *N* é igual ao número de possíveis valores para o atributo. Cada valor codificado tem *N-1* elementos com valor 0 e apenas o bit que representa o valor codificado com valor 1. As vantagens dessa codificação são: simplicidade, facilidade de uso e o fato de algoritmos, como RNA, poderem aprender facilmente a distinguir os valores. No entanto, para atributos com um conjunto de valores possíveis grande, esta codificação apresenta um custo muito alto com relação ao tamanho da representação. Conseqüentemente, o tempo para obtenção do modelo é aumentado e o poder de generalização pode ser prejudicado;

Padrão: Cada valor recebe seu valor binário correspondente. A dimensão da representação é igual ao inteiro imediatamente superior a $log_2(N)$, onde N é o valor máximo do atributo. Esta representação é bastante simples, mas distancia valores próximos;

M de N: Tenta resolver os problemas apresentados nas codificações padrão e *1 de N*. Nesta codificação *M* dos *N* bits são iguais a 1. Desta forma, todos os valores codificados têm a mesma representatividade. É uma das codificações mais aplicadas;

Termômetro: É utilizada quando existe uma relação entre os valores do atributo. Neste caso, pode ser necessário aproximar alguns valores e afastar outros, a fim de que o algoritmo de mineração faça uso dessas distâncias para achar uma solução melhor, de forma mais rápida. Como exemplo, considerando que um atributo pode assumir os valores *ruim*, *bom* e *ótimo*, e é desejável que o valor *ótimo* esteja distante de *ruim* e próximo de *bom*, a representação (100, 110, 111) seria adequada. Apesar de funcionar bem, esta técnica exige a aplicação de taxonomias e engenharia de conhecimento, e não é tão aplicada quanto as demais.

As etapas de preparação dos dados descritas anteriormente consomem de 50% a 90% do tempo e esforço necessário para a realização de todo o processo de KDD [Two Crows 1999]. Quanto mais completa e consistente for a preparação, melhor será o resultado da mineração [Bigus 1996].

2.2.6. Mineração de Dados

Depois que os dados são coletados e processados, a etapa de mineração pode ser iniciada. Nesta etapa são realizadas a escolha, configuração e execução de um ou mais algoritmos (isolados ou integrados) para extração automática (ou semi-automática) de conhecimento da base de dados. Normalmente, esta etapa é executada diversas vezes para ajustar o conjunto de parâmetros, visando à obtenção de resultados mais adequados aos objetivos estabelecidos [Rezende et al. 2003].

Tarefas de Mineração de Dados

Antes da escolha do algoritmo de mineração, deve-se definir a tarefa de mineração que será empregada. A escolha da tarefa é realizada de acordo com os objetivos estabelecidos na etapa de identificação do problema, pois cada tarefa pode extrair diferentes tipos de conhecimento. Como mostra a Figura 2.2 [Rezende et al. 2003], as tarefas de mineração de dados podem ser classificadas como preditivas e descritivas.

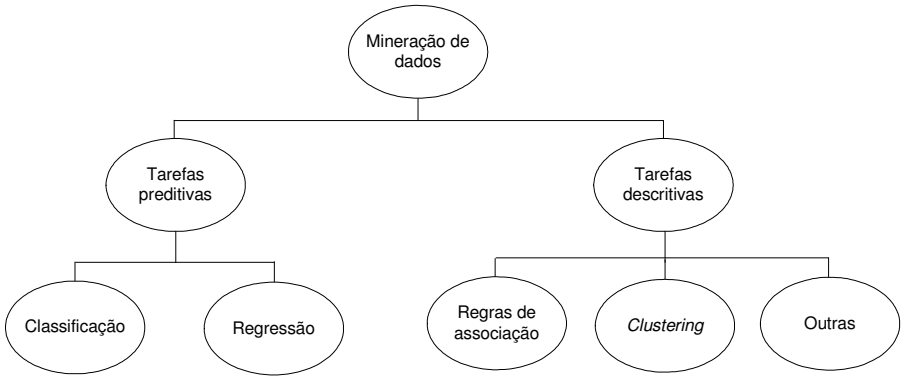

Figura 2.2 - Tarefas de mineração de dados

As tarefas preditivas consistem em utilizar um conjunto de dados com uma característica conhecida para prever a mesma característica em novos exemplos. As principais tarefas preditivas são classificação (encontra um relacionamento entre os atributos e as classes para realizar a predição) e regressão (é similar à classificação, no entanto, o atributo a ser predito é contínuo).

As tarefas descritivas fornecem uma descrição concisa dos dados. Ao final da realização deste tipo de tarefa, o conhecimento descoberto é representado numa forma compreensível [Fayyad et al. 1996a]. *Clustering* (identifica um conjunto de grupos ou *clusters*, inicialmente desconhecidos, de tal forma que exemplos com características similares sejam agrupados no mesmo *cluster*) e regras de associação (especifica as associações em que a presença de um conjunto de itens implica em outro conjunto) são exemplos de tarefas descritivas.

Algoritmos de Mineração de Dados

Depois que a tarefa de mineração é definida, deve-se escolher o algoritmo apropriado para executá-la. Os algoritmos de aprendizagem podem ser agrupados nas seguintes categorias [Fayyad et al. 1996a] [Goebel & Gruenwald 1999] [Monard & Baranauskas 2003]:

- Simbólico: Através da análise de um conjunto de exemplos, buscam aprender construindo representações simbólicas. As representações são compreensíveis e normalmente estão na forma de árvore de decisão ou regras;
- Estatístico: Extraem um conjunto de medidas características a partir dos dados. A idéia geral consiste em utilizar modelos estatísticos para encontrar uma boa aproximação do conceito alvo.

  Os métodos estatísticos tradicionais são poderosos e normalmente apresentam alta precisão. Suas principais limitações são: geralmente se baseiam em suposições que nem sempre são satisfeitas pelos dados do mundo real, os resultados podem ser difíceis de interpretar e exige conhecimento matemático para serem aplicados.

  Entre os métodos estatísticos, destacam-se os de aprendizado Bayesiano, que utilizam um modelo probabilístico baseado no conhecimento prévio do problema, o qual é combinado com os exemplos de treinamento para determinar a probabilidade final de uma hipótese;
- Baseado em Exemplos: Tenta resolver um dado problema utilizando soluções passadas. Desta forma, assume-se a existência de conhecimento *a priori* implícito na forma de um conjunto de exemplos. Predições de exemplos novos são derivadas de exemplos similares cuja predição é conhecida. Esse tipo de categoria necessita manter os exemplos na memória para realizar a predição. Assim, é muito importante saber quais exemplos de treinamento devem ser memorizados. Uma desvantagem desses métodos é a requisição de uma métrica de distância bem definida para avaliar as distâncias entre os exemplos. Além disso, podem ser difíceis de interpretar, pois o modelo está implícito nos dados. KNN e Raciocínio Baseado em Casos são as técnicas mais conhecidas;
- Conexionista: Construções matemáticas simplificadas (RNA) inspiradas no modelo biológico do sistema nervoso. A representação de uma RNA envolve unidades de processamento altamente interconectadas. Como no cérebro humano, a intensidade das conexões pode mudar em resposta a um estímulo apresentado ou a uma saída obtida, permitindo à rede aprender;

- Evolutivo: O modelo consiste de uma população de indivíduos que competem para fazer a predição. Indivíduos que possuem uma performance fraca são descartados, enquanto os mais fortes proliferam, produzindo variações de si mesmos. Possui uma analogia direta com a teoria de Darwin, na qual sobrevivem os mais bem adaptados ao ambiente.

Existem controvérsias em relação à classificação de alguns algoritmos. As RNA, por exemplo, são consideradas por alguns autores como métodos estatísticos paramétricos, uma vez que seu treinamento geralmente significa encontrar valores apropriados para os pesos e *bias* [Monard & Baranauskas 2003]. KNN, por sua vez, também faz parte das técnicas estatísticas modernas ou não paramétricas, pois não exigem conhecimento *a priori* acerca das funções de densidade de probabilidade [Michie et al. 1994].

Seleção do Algoritmo de Mineração de Dados

A escolha do algoritmo não é uma tarefa fácil, pois não existe um único algoritmo que apresente o melhor desempenho para todos os problemas. Diferentes tarefas requerem diferentes tipos de algoritmos. As RNA, por exemplo, são melhores para classificação, enquanto que os Algoritmos Genéticos (AG) são mais adequados para resolver tarefas *problem-solving* (tarefas cuja solução é pesquisada em um espaço de estados até que o estado atual satisfaça o objeto desejado) [Adriaans & Zantinge 1996].

Além da tarefa de mineração de dados, os seguintes aspectos que devem ser considerados quando selecionando um algoritmo de mineração [Adriaans & Zantinge 1996]:

- Número de exemplos: Alguns algoritmos são melhores para manipular grandes quantidades de exemplos;
- Número de atributos: A performance de alguns algoritmos, como RNA e AG, deteriora-se consideravelmente à medida que o número de atributos aumenta;
- Tipos de atributos que podem ser manipulados;
- Representação do conhecimento: Alguns algoritmos, como as RNA, não fornecem explicação de suas respostas;
- Capacidade de aprender de forma incremental: Quando novos dados tornam-se disponíveis, o algoritmo é capaz de revisar suas teorias, sem refazer completamente o processo de aprendizagem. Isto pode ser de grande relevância em aplicações com bases grandes;
- Habilidade de estimar a significância estatística dos resultados: Em alguns algoritmos, como RNA e AG, normalmente é muito difícil avaliar os resultados estatisticamente;
- Performance.

Metodologia de Avaliação dos Algoritmos de Mineração de Dados

É importante adotar alguma metodologia de avaliação para estimar o desempenho dos algoritmos e compreender suas vantagens e limitações. A metodologia de avaliação baseada em amostragem é bastante utilizada [Monard & Baranauskas 2003].

Para estimar uma medida (Ex.: precisão) de um indutor, normalmente utiliza-se um conjunto de exemplos de treinamento e outro de teste. A fim de assegurar que as medidas obtidas, utilizando o

conjunto de teste, sejam estatisticamente válidas, é necessário que os conjuntos sejam disjuntos. O conjunto de treinamento deve conter informações significativas que possam ajudar na solução do problema. Ele é utilizado para permitir que o indutor aprenda informações relevantes, tais como parâmetros estatísticos, agrupamentos, características discriminantes ou estruturas básicas dos padrões [Monteiro 1999]. Outro conjunto de dados, conjunto de validação, pode ser utilizado durante o processo de aprendizagem para verificar a capacidade de generalização do indutor, e, a partir desta análise, decidir se o treinamento deve continuar [Beale & Jackson 1991] [Haykin 1999]. Nos casos em que o conjunto de validação é usado, os conjuntos também devem ser particionados independentemente. Ao terminar a avaliação do algoritmo, todos os dados podem ser usados para construir o modelo final. Geralmente, quanto maior o conjunto de treinamento, melhor o indutor e quanto maior o conjunto de teste, mais precisa a estimativa da performance [Witten & Frank 2000].

Os métodos mais difundidos para estimar uma medida verdadeira são [Han & Kamber 2001] [Monard & Baranauskas 2003]:

- Re-substituição: constrói o indutor e testa seu desempenho no mesmo conjunto de exemplos. Este método fornece uma medida altamente otimista, pois o bom desempenho no conjunto de treinamento, em geral, não se estende a conjuntos independentes de teste;
- *Cross-validation*: na *r-fold cross-validation* os exemplos (total igual a *n*) são aleatoriamente divididos em *r* partições mutuamente exclusivas (*folds*) de tamanho aproximadamente igual a *n/r*. *(r-1)* partições são usadas para treinamento e a hipótese induzida é testada na partição remanescente. Este processo é repetido *r* vezes, cada vez considerando uma partição diferente para teste. A precisão é estimada pela média das precisões de cada iteração. É um dos métodos mais utilizados, principalmente em problemas cujo tamanho da base de dados é limitado;
- *Stratified cross-validation*: similar ao *cross-validation*, mas, ao gerar as partições, a distribuição das classes presente na base de dados é considerada. Em geral, recomenda-se o método *stratified 10-fold cross-validation* por apresentar viés e variância relativamente baixos [Witten & Frank 2000].
- *Holdout*: caso especial da *cross-validation*, onde os dados são aleatoriamente particionados em dois conjuntos independentes (treinamento e teste). Normalmente, dois terços dos dados são alocados para o conjunto de treinamento e o restante para o conjunto de teste. Devido à grande quantidade de dados disponíveis na maioria das aplicações reais, a utilização deste método é suficiente para estimar a performance do modelo [Beale & Jackson 1991];
- *Leave-one-out*: caso especial da *cross-validation*, onde a partição utilizada em cada teste é composta de um exemplo. É computacionalmente dispendioso e freqüentemente usado em amostras pequenas. A precisão é estimada pela média das precisões de cada teste;
- *Boostrap*: repete o processo diversas vezes utilizando um novo conjunto de treinamento, obtido por amostragem com reposição do conjunto original. Os exemplos que não aparecerem no conjunto de treinamento constituem o conjunto de teste. A precisão é estimada pela média das precisões de cada iteração.

Além de estimar a precisão, normalmente deseja-se saber se os algoritmos possuem capacidade de generalização equivalente. Uma maneira óbvia seria comparar os resultados obtidos dos métodos

de estimação descritos anteriormente. No entanto, tal alternativa apresenta o problema de variância das estimativas. Conseqüentemente, testes estatísticos, como *student paired t-test* [Goulden 1956], são requeridos.

Nos casos em que a precisão do sistema precisa ser melhorada, técnicas como *bagging* [Breiman 1996], *boosting* [Freund & Schapire 1997], *stacking* e *error-correcting output codes* [Dietterich & Bakiri 1995] podem ser utilizadas [Witten & Frank 2000]. Tais técnicas combinam um conjunto de modelos com o objetivo de criar um modelo com melhor precisão.

Em algumas aplicações, a avaliação direta da precisão não é suficiente para selecionar o melhor modelo, sendo necessário considerar aspectos como tipos de erro, comumente relatados através de uma matriz de confusão, e custos associados a cada tipo de erro [Two Crows 1999].

Uma matriz de confusão especifica o número de classificações corretas versus o número de classificações preditas para cada classe sobre um conjunto de exemplos. Portanto, é uma ferramenta muito útil para interpretar os resultados de problemas de classificação, pois além de mostrar a precisão global do modelo, especifica detalhadamente os tipos de erro, permitindo uma verificação mais abrangente de sua precisão [Two Crows 1999].

Como mostra a Tabela 2.1 [Monard & Baranauskas 2003], os resultados de uma matriz de confusão são totalizados em duas dimensões: classes verdadeiras e classes preditas, para *k* classes $\{C_1, C_2, ..., C_k\}$. Cada elemento $Num(C_i,C_j)$ da matriz representa o número de exemplos que pertencem à classe C_i, mas foram classificados como sendo da classe C_j. O número de acertos para cada classe está localizado na diagonal principal. Os demais elementos representam erros na classificação.

Tabela 2.1 - Matriz de confusão

Classe	**Predita C_1**	**Predita C_2**	**...**	**Predita C_k**
Verdadeira C_1	Num(C_1, C_1)	Num(C_1, C_2)	...	Num(C_1, C_k)
Verdadeira C_2	Num(C_2, C_1)	Num(C_2, C_2)	...	Num(C_2, C_k)
...	...	...	...	...
Verdadeira C_k	Num(C_k, C_1)	Num(C_k, C_2)	...	Num(C_k, C_k)

Em algumas aplicações do mundo real, os diferentes tipos de erros incorrem em diferentes custos. Assim, em vez de projetar um algoritmo que minimize a taxa de erro global, procura-se gerar um modelo que minimize o custo de classificação incorreta [Monard & Baranauskas 2003]. Portanto, um modelo com precisão menor pode ser preferível a um modelo que apresenta melhor precisão, mas tem um custo associado maior devido aos tipos de erro cometidos [Two Crows 1999].

Uma alternativa natural para os casos onde cada tipo de classificação incorreta possui um custo diferente consiste em associar um custo para cada tipo de erro. No cálculo utilizando custos, os erros são convertidos em custos através da multiplicação do erro pelo custo correspondente [Monard & Baranauskas 2003]. Neste caso, o algoritmo de mineração não é sensível ao custo, pois a informação sobre os custos não está sendo usada durante a fase de aprendizagem. Performances melhores podem ser obtidas se o algoritmo for sensível ao custo. Uma forma simples de tornar o algoritmo sensível ao custo é gerar amostras de dados de treinamento com proporções diferentes para as classes. Quanto maior a proporção de uma classe, mais o algoritmo tenderá a evitar erros associados à classe, pois a penalização será mais freqüente [Witten & Frank 2000].

Técnicas visuais como *lift chart* também são bastante utilizadas em aplicações onde o custo associado a cada tipo de erro é considerado durante a etapa de mineração [Witten & Frank 2000].

Diversos métodos têm sido propostos para os casos em que a distribuição das classes e os custos associados aos tipos de erro não são precisamente conhecidos. A análise comparativa de classificadores através de curvas ROC (*Receiver Operating Characteristics*) é um exemplo de método que vem sendo bastante adotado [Provost & Fawcett 1997]. As curvas ROC mostram a relação das taxas de falsos positivos (FP) e verdadeiros positivos (VP) através da variação de um limiar. Esta relação prediz o comportamento dos classificadores independentemente dos custos e da distribuição das classes. Numa curva ROC, o eixo das ordenadas (*y*) representa VP e o eixo das abscissas (*x*) representa FP. As Equações 2.2 e 2.3 mostram como essas taxas são calculadas.

$$VP = \frac{positivos\ classificados\ corretamente}{total\ positivos} \tag{2.2}$$

$$FP = \frac{negativos\ classificados\ incorretamente}{total\ negativos} \tag{2.3}$$

Além da precisão, os algoritmos de mineração podem ser analisados com relação a diversos aspectos, tais como velocidade de treinamento, robustez, escalabilidade, sensibilidade, especificidade e interpretabilidade [Han & Kamber 2001].

2.2.7. Avaliação dos Padrões

A obtenção do conhecimento não é o passo final do processo de KDD. Os algoritmos de mineração podem gerar uma quantidade enorme de padrões, muitos dos quais podem ser irrelevantes para o usuário. Fornecer ao usuário uma grande quantidade de padrões não é produtivo [Rezende et al. 2003]. Portanto, surge a necessidade de distinguir o conhecimento óbvio e irrelevante do que é efetivamente útil [Hilderman & Hamilton 1999]. [Han & Kamber 2001] consideram um padrão como sendo interessante se ele é fácil de ser compreendido, válido em dados novos com algum grau de certeza, potencialmente útil e novo.

Diversas medidas para avaliação de conhecimento têm sido pesquisadas. Estas medidas ordenam o conhecimento descoberto segundo um nível de importância, reduzindo o número de padrões que precisam ser considerados [Hilderman & Hamilton 1999]. Elas podem ser divididas em duas categorias: desempenho e qualidade. Algumas medidas de desempenho são precisão, erro, suporte, confiança, sensibilidade e especificidade. Compreensibilidade e interessabilidade (*interestingness*) são exemplos de medidas de qualidade.

As medidas de interessabilidade podem ser divididas em objetivas e subjetivas. As medidas objetivas estão relacionadas com a estrutura dos padrões e conjunto de teste. Elas não consideram fatores específicos do usuário e do conhecimento do domínio, e normalmente estão associadas a um limiar que pode ser controlado pelo usuário. Exemplos de medidas objetivas são custo de classificação incorreta, modelos de regras e cobertura mínima das regras. Como diferentes usuários podem ter diferentes graus de interesse para um determinado padrão, medidas subjetivas são necessárias. Estas medidas consideram fatores específicos do conhecimento do domínio e de interesse do usuário. Exemplos de medidas subjetivas são inesperabilidade e utilidade. Num ambiente para avaliação de conhecimento, medidas objetivas podem ser utilizadas como um primeiro filtro. Em seguida, as

medidas subjetivas podem ser aplicadas como um filtro final para selecionar o conhecimento realmente interessante [Rezende et al. 2003].

Após a avaliação do resultado do processo com relação aos objetivos estabelecidos inicialmente, pode ser necessário repetir o processo realizando tarefas como adição de atributos ou exemplos, aplicação de outras técnicas de transformação e ajuste de parâmetros no algoritmo de mineração a fim de obter resultados melhores na próxima iteração [Monard & Baranauskas 2003].

2.2.8. Apresentação do Conhecimento

Um dos principais objetivos do processo de KDD é que o usuário possa compreender e utilizar o conhecimento descoberto [Rezende et al. 2003]. Portanto, não adianta apenas apresentar os resultados obtidos, é preciso que o conhecimento seja representado numa forma que o usuário possa entender. Para isso, são utilizados recursos como relatórios, gráficos e planilhas, e formas de representação como regras e grafos. Tais recursos facilitam a interpretação dos resultados através da apresentação do conhecimento descoberto numa forma mais compacta, contribuindo para que o mesmo seja incorporado ao processo de tomada de decisões [Witten & Frank 2000].

As técnicas e ferramentas de visualização [Fayyad et al. 2001] também podem ser aplicadas nas demais etapas do processo de KDD, melhorando a compreensão dos resultados intermediários [Rezende et al. 1998].

2.3. Aplicações de KDD

A disponibilidade de grandes quantidades de dados e a eminente necessidade de transformar estes dados em conhecimento útil têm sido os principais motivos pelos quais KDD tem atraído tanta atenção da indústria da informação nos últimos anos. O conhecimento descoberto pode ser usado em aplicações que vão desde o gerenciamento de negócios, controle de produção e análise de mercado, até projetos de engenharia e pesquisas científicas [Han & Kamber 2001].

A seguir são destacadas algumas das principais áreas de aplicação de KDD. Além das aplicações citadas, novas estão constantemente sendo descobertas [Monteiro 1999].

2.3.1. Lojas de Varejo

As lojas de varejo vendem uma grande variedade de produtos. Neste setor, são indesejáveis custos excessivos de transporte e estoque, e a impossibilidade da venda de um produto por falta no estoque. A aplicação de KDD para previsão de vendas é muito importante para este setor, pois permite a otimização de dois pontos fundamentais do negócio: as vendas e o estoque [Monteiro 1999].

Outra aplicação bastante usual é a análise *market-basket*, que descobre associações entre produtos que são comprados em conjunto ou seqüência [Fayyad et al. 1996a].

2.3.2. Segmentação de Mercado

A segmentação de mercado baseia-se na idéia de que um produto não pode satisfazer as necessidades e desejos de todos os consumidores, pois eles estão dispersos em inúmeras regiões, têm hábitos de compras variados e divergem em suas necessidades e preferências. Por esta razão, a

prática moderna de *Marketing* sugere dividir o mercado em segmentos, avaliá-los, selecionar alguns como alvo e definir a posição da empresa em relação a cada um deles. Esta segmentação ajuda a empresa a compreender melhor o perfil de seus clientes e realizar um *marketing* dirigido que proporciona um maior retorno financeiro para empresa e maior satisfação para os clientes [Witten & Frank 2000].

2.3.3. Setor Financeiro

Na maioria dos ambientes de decisão financeira utilizam-se informações provenientes de diversas fontes. Os gerentes financeiros analisam essas informações de forma subjetiva e dificilmente conseguem formalizar seus processos de decisão. Além disso, esses ambientes são dinâmicos e as decisões devem ser tomadas rapidamente. Este cenário tem requerido a utilização de tecnologias mais sofisticadas para satisfazer usuários cada vez mais exigentes. Assim, o mercado financeiro está buscando a utilização de novas técnicas computacionais, entre elas as técnicas inteligentes.

KDD tem sido aplicado com bastante sucesso no setor financeiro [Bigus 1996]. Análise de propostas de cartões de crédito, previsão de índices de bolsas de valores, desenvolvimento de modelos de fidelização de consumidores, detecção de fraudes e tendências estão entre as aplicações mais usuais [Monteiro 1999].

2.3.4. Setores Químico e Farmacêutico

Dois domínios bastante promissores para aplicação de KDD são os setores químico e farmacêutico. As atividades inerentes a estes setores requerem processamento intenso nas bases de dados. Tal processamento pode ser realizado de forma mais eficiente através da incorporação de técnicas de mineração de dados.

Atualmente, os setores químico e farmacêutico estão aplicando KDD em grandes bases de dados de componentes químicos para descobrir novas substâncias e inter-relações entre componentes químicos, e aprimorar fórmulas e componentes já desenvolvidos. Os resultados do processo de KDD podem ser úteis em diversas aplicações, como por exemplo, tratamento de doenças e produção de produtos agrícolas [Two Crows 1999].

2.3.5. Análise de Dados de DNA

Pesquisas recentes em análise de dados de DNA têm sido conduzidas para prevenção e tratamento de doenças, descoberta de medicamentos e métodos para diagnóstico, e identificação das causas genéticas de doenças.

Os seres humanos possuem em torno de 100.000 genes, que normalmente são compostos de centenas de nucleotídeos. Existe quase um número ilimitado de formas de ordenação e seqüenciamento dos nucleotídeos. Devido a esta complexidade, é um desafio identificar padrões específicos de seqüências de genes que têm alguma relação com uma determinada doença. Com o desenvolvimento de técnicas de mineração para análise de padrões seqüenciais e busca por similaridade, KDD tem se tornado bastante promissor nesta área, podendo contribuir substancialmente em atividades como análise de associação (identificação de seqüências de genes co-relacionadas),

análise de caminho (influência dos genes nos diversos estágios da doença), busca por similaridade e comparação entre seqüências de DNA [Han & Kamber 2001].

2.3.6. Análise na *Web*

A *World Wide Web* vem despertando muito interesse para aplicação de KDD [Witten & Frank 2000]. Os *sites* e bases de dados associadas aos sistemas *web* possuem grande quantidade de dados que pode ser minerada. Dados sobre os acessos a um *site*, por exemplo, podem ser armazenados automaticamente em arquivos *log*. Tais arquivos podem ser utilizados num processo de KDD para descobrir comportamentos padrões dos visitantes. No caso do comércio eletrônico, onde os consumidores são altamente móveis e demonstram uma lealdade mais vulnerável do que no comércio tradicional, o principal desafio é identificar e entender essa nova base de consumidores. Com a aplicação do processo de KDD, comportamentos e preferências dos consumidores podem ser descobertos, tornando as empresas virtuais mais competitivas.

2.3.7. Medicina

Os centros médicos estão investindo cada vez mais na integração de equipamentos de monitoramento e coleta de dados com os sistemas de informação. Como resultado, grandes quantidades de dados estão sendo armazenadas nas bases de dados médicas [Lavrač 1999).

As bases de dados médicas possuem dados históricos de doenças contraídas, exames laboratoriais e clínicos. Estes dados são caracterizados por incompletude, incorretude, registros não representativos e falta de exatidão. Além disso, os dados coletados pelos equipamentos de monitoramento envolvem valores medidos em intervalos distintos e, conseqüentemente, requerem que o aspecto temporal seja considerado durante a análise dos dados (Lavrač 1999).

O aumento no volume das bases de dados médicas e as características inerentes a este domínio têm dificultado a extração de conhecimento útil para tomada de decisões, requerendo ferramentas efetivas para gerenciamento e análise dos dados. Portanto, as tecnologias provenientes da área de KDD são bastante úteis à Medicina. Estudos epidemiológicos; predição de quais pacientes têm maior probabilidade de contrair determinada doença, em função de dados históricos; predição da efetividade de procedimentos cirúrgicos, teste médicos e medicamentos são alguns exemplos de aplicações em que KDD pode ser empregado [Two Crows 1999].

2.4. Considerações Finais

Nos últimos anos, os benefícios de KDD têm sido demonstrados em diversos domínios de aplicação. Como resultado, esta área de pesquisa é considerada bastante promissora.

Inicialmente as etapas do processo de KDD eram validadas com pequenos conjuntos de dados. No entanto, a realidade de um mundo imperfeito e *terabytes* de dados inconsistentes revelaram as limitações de alguns métodos que vinham sendo desenvolvidos. Apesar das evoluções realizadas até o momento, diversos desafios estão continuamente surgindo. O desenvolvimento de algoritmos e técnicas que lidem de forma mais eficiente com volumes de dados maiores, proposição de técnicas que reduzam a dimensionalidade do problema, criação de algoritmos incrementais (lidam com dados não estacionários), facilidade de incorporação do conhecimento prévio sobre o domínio do problema,

integração com outros sistemas, aplicação do processo de KDD em ambientes de rede e distribuídos, e a descoberta de conhecimento a partir de dados mais complexos (espaciais, orientados a objetos, multimídia, etc.) são alguns exemplos [Fayyad et al. 1996a] [Rezende et al. 2003].

Independente das evoluções, o sucesso nas aplicações de KDD sempre dependerá, em parte, da participação de especialistas do domínio. Assim, inicialmente, é importante realizar uma análise a fim de adquirir conhecimento sobre o problema. O conhecimento adquirido irá fornecer um subsídio para as demais etapas do processo [Rezende et al. 2003]. Além disso, mais do que aprender indutivamente a partir de bases de dados, o processo de KDD deve criar modelos compreensíveis que possam ser utilizados na tomada de decisões. Este requisito pode ser alcançado através do uso de ferramentas e técnicas de visualização [Witten & Frank 2000] [Fayyad et al. 2001].

Capítulo 3
Redes Neurais Artificiais

3.1. Introdução

Nas últimas décadas, muita atenção tem sido voltada ao estudo do paradigma conexionista. Este paradigma surgiu como uma tentativa de emular em computadores as atividades de armazenamento e processamento de informação realizadas pelo sistema neural biológico [Beale & Jackson 1991] [Haykin 1999]. Como resultado, diversos modelos extremamente simplificados, sobre o ponto de vista da neurofisiologia das estruturas biológicas correspondentes, foram propostos. Estes modelos, denominados de Redes Neurais Artificiais (RNA), compartilham diversas características dos modelos neurais biológicos, tais como aprendizagem a partir da experiência, generalização baseada em exemplos, processamento paralelo e distribuído, armazenamento associativo de informação e abstração de informações essenciais a partir de dados irrelevantes. Devido a estas características, as RNA têm atraído bastante atenção de vários setores da sociedade e pesquisadores de áreas como Neurociência, Ciência Cognitiva, Estatística e Inteligência Artificial [Monteiro 1999].

As aplicações de RNA se estendem por uma vasta diversidade de áreas. Inicialmente, as aplicações foram concentradas na exploração e reprodução de tarefas realizadas pelo cérebro humano, tais como visão, fala, processamento de informação e coordenação motora. Em seguida, as RNA passaram a ser aplicadas em tarefas como problemas de otimização combinatória, reconhecimento de padrões, modelagem de sistemas de controle e aproximação de funções.

Este capítulo aborda as principais características das RNA, considerações relevantes para projetos de RNA e a viabilidade de aplicar RNA ao processo de KDD. O modelo *Perceptron* Multicamadas (*Multilayer Perceptron* – MLP) é discutido de forma detalhada devido às seguintes razões: este modelo será usado em conjunto com a técnica TREPAN (abordada no Capítulo 5) e servirá como base para comparação com os modelos neuro-*fuzzy* FWD e FuNN, selecionados para investigação nesta dissertação (discutidos no Capítulo 4).

3.2. Principais Características

Uma RNA consiste de um conjunto de unidades de processamento interconectadas, os neurônios artificiais. Cada neurônio (nó) executa uma função simples, mas a RNA como um todo tem capacidade computacional para resolver problemas complexos [Braga et al 2003]. Os neurônios operam sobre

informações locais e são dispostos em uma ou mais camadas interligadas por conexões, geralmente unidirecionais. Na maioria dos modelos, estas conexões estão associadas a pesos, os quais armazenam o conhecimento representado no modelo e servem para ponderar a entrada recebida por cada neurônio. Em geral, a aprendizagem da rede é realizada modificando os valores desses pesos.

Os modelos neurais apresentam diversas características em comum [Beale & Jackson 1991] [Haykin 1999] [Braga et al. 2003].

Uma das principais características das RNA é a capacidade de fornecer respostas coerentes para dados não-conhecidos. Tal característica, denominada de generalização, é uma demonstração de que a capacidade das RNA vai muito além de simplesmente mapear relações entrada-saída. A generalização ocorre porque as RNA são capazes de detectar características relevantes que não são explicitamente representadas nos dados de treinamento e permitem a descoberta de uma relação mais geral.

As RNA são aproximadores universais de funções multivariáveis contínuas, com custo computacional que cresce apenas linearmente com o número de variáveis.

Os modelos neurais são robustos, pois são capazes de manipular dados incompletos e imperfeitos. Esta característica é bastante importante, pois, no mundo real, os dados normalmente são ruidosos.

Outra característica atraente das RNA é a capacidade de aprendizagem e adaptação a partir de um conjunto de dados. Diferente das técnicas clássicas da Inteligência Artificial (IA), as RNA não dependem da disponibilidade *a priori* de uma base de conhecimento na forma de regras, mas de um conjunto de dados do qual o conhecimento é extraído automaticamente.

Diferente das técnicas estatísticas, as RNA descobrem relacionamentos entre os atributos de entrada e saída sem fazer suposições sobre a distribuição dos dados.

Com relação ao processamento, as RNA atuam de forma distribuída e paralela, pois cada neurônio é capaz de aprender uma parte do problema. Como resultado, as redes podem ser implementadas em máquinas paralelas para melhorar o tempo de aprendizagem e se tornar mais tolerantes a falhas (entradas incompletas ou com ruído podem ser reconhecidas e uma rede danificada pode ser capaz de funcionar de forma satisfatória).

Outras características das RNA são:

- ♣ Implementação rápida;
- ♣ Capacidade de manipular apenas atributos numéricos;
- ♣ Os parâmetros normalmente são determinados de forma empírica;
- ♣ Não são facilmente interpretadas. Tal característica é uma das principais críticas às RNA;
- ♣ Devido ao grande número de parâmetros, tendem a se ajustar aos dados de treinamento;
- ♣ Tendem a funcionar melhor quando o conjunto de dados é suficientemente grande e a razão sinal por ruído é razoavelmente alta;
- ♣ Capacidade de auto-organização e processamento temporal. Estas características não estão presentes em todos os modelos neurais;

- A menos que o problema seja muito pequeno, requerem tempo de treinamento extenso. No entanto, depois de treinadas, fornecem predições de forma bastante rápida.

A maioria das características especificadas faz das RNA uma ferramenta computacional extremamente poderosa e atrativa para solução de problemas complexos.

3.3. Aprendizagem

Aprendizagem em RNA é um aspecto extremamente importante que continua a ser submetido a pesquisas intensas. No contexto da computação neural, aprendizagem pode ser definida como um processo de atualização iterativa dos parâmetros ajustáveis da rede, de tal forma que a rede seja capaz de realizar eficientemente uma tarefa [Haykin 1999]. Assim, durante a aprendizagem, a rede extrai informações relevantes do conjunto de treinamento e, ao final do processo, os parâmetros codificam o conhecimento adquirido [Braga et al. 2003].

Para projetar um processo de aprendizagem deve-se ter inicialmente um modelo do ambiente no qual a rede irá operar, ou seja, é necessário saber quais informações estarão disponíveis para a rede. Este modelo é conhecido como paradigma de aprendizagem [Haykin 1999]. Os principais paradigmas de aprendizagem são: supervisionado, não-supervisionado e por reforço.

A aprendizagem supervisionada, cujo esquema é mostrado na Figura 3.1 [Braga et al. 2000], é o paradigma mais comum. Neste paradigma é fornecido ao algoritmo de aprendizagem um conjunto de exemplos para os quais a saída desejada é conhecida. A entrada e a saída desejada são fornecidas por um supervisor (professor) externo. A aprendizagem é realizada de forma que a saída da rede se aproxime da saída desejada. Este tipo de aprendizagem é mais utilizado em problemas de classificação e predição.

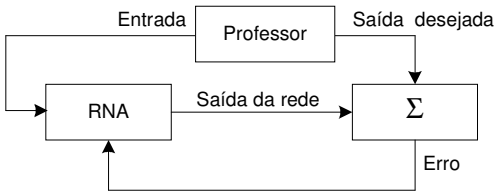

Figura 3.1 - Esquema da aprendizagem supervisionada

Na aprendizagem não-supervisionada, cujo esquema é mostrado na Figura 3.2 [Braga et al. 2000], as saídas desejadas não são conhecidas *a priori* e são descobertas a partir de correlações, características e regularidades presentes nos exemplos de entrada. O algoritmo de aprendizagem analisa os exemplos fornecidos e verifica se eles podem ser agrupados de alguma forma. A partir do momento em que a rede estabelece uma harmonia com as regularidades estatísticas dos exemplos, ela desenvolve uma habilidade de formar representações internas para codificar características e criar novos grupos automaticamente [Haykin 1999]. Após a determinação dos grupos, é necessária uma análise para determinar o que cada grupo representa no contexto do problema investigado. Este tipo de aprendizagem é mais utilizado em problemas de segmentação e *clustering*, e só se torna possível quando existe redundância nos dados de entrada, o que possibilita encontrar padrões ou características regulares.

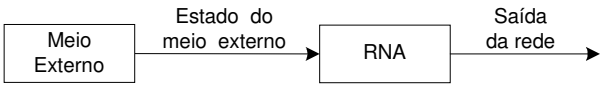

Figura 3.2 - Esquema da aprendizagem não-supervisionada

Na aprendizagem por reforço, ilustrada na Figura 3.3 [Braga et al. 2000], a rede tenta aprender o mapeamento entrada-saída através da interação contínua com o ambiente externo, visando maximizar um índice de desempenho chamado sinal de reforço. Se a ação tomada pela rede é seguida de estados satisfatórios, então a tendência da rede produzir esta ação particular é reforçada. Por outro lado, se não for seguida de estados satisfatórios, a tendência da rede produzir esta ação é enfraquecida. Neste tipo de aprendizagem, a rede recebe apenas um valor que informa se a saída está correta ou não, ou seja, a saída desejada não é fornecida para a rede. A aprendizagem por reforço é mais utilizada em aplicações de controle e robótica.

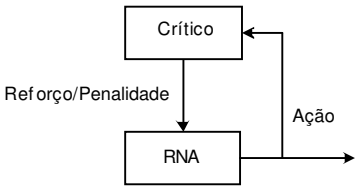

Figura 3.3 - Esquema da aprendizagem por reforço

3.4. O Modelo *Perceptron* Multicamadas

A idéia de RNA surgiu a partir do MCP, modelo artificial de um neurônio biológico proposto por Waren McCulloch e Walter Pitts [McCulloch & Pitts 1943]. Posteriormente, Frank Rosenblatt [Rosenblatt 1962] demonstrou com seu novo modelo (*perceptron*) que, se fossem acrescidas de pesos ajustáveis, as RNA com nós MCP poderiam ser treinadas para classificar certos tipos de padrões.

O *perceptron* possui três camadas: a primeira recebe as entradas do exterior e possui conexões fixas; a segunda recebe impulsos da primeira camada através de conexões fixas e envia suas saídas para a terceira camada (resposta). Inicialmente, a saída da rede é aleatória, mas, através dos ajustes dos pesos, a rede é treinada para fornecer saídas de acordo com os dados de treinamento. Embora a topologia original do *perceptron* possua três camadas, ela é conhecida como *perceptron* de uma camada, pois apenas a camada de saída possui propriedades adaptativas [Braga et al. 2000].

A Figura 3.4 [Braga et al. 2000] mostra a estrutura de um nó MCP, que corresponde a unidade de processamento do *perceptron*. Nesta figura, $X = [x_1, x_2, ..., x_n]$ é o vetor de entrada, $W = [w_1, w_2, ..., w_n]$ é o vetor de pesos associados às entradas e y é a saída do nó. O vetor de pesos determina como o nó avalia a combinação dos sinais de entrada e o somatório ponderado representa o estímulo recebido pelo nó. Este estímulo é passado como parâmetro para uma função de ativação, que determina o nível de atividade do nó. No caso do *perceptron*, é utilizada a função degrau com limiar θ cujo valor tem que ser alcançado ou excedido para o nó produzir um sinal. Desta forma, se a soma for maior que θ, o nó fornece a resposta 1, caso contrário, 0. Por questões de simplicidade notacional, considera-se o limiar como outro peso $w_0 = -\theta$ ligado a um nó com uma entrada constante $x_0 = 1$.

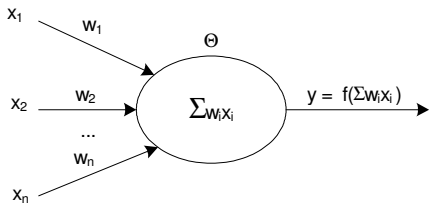

Figura 3.4 - Estrutura do MCP

A Tabela 3.1 descreve a regra de aprendizagem do *perceptron*.

Tabela 3.1 - Regra de aprendizagem do *perceptron*

1. Iniciar as conexões com pesos aleatórios;
2. Selecionar um vetor de entrada *x* do conjunto de treinamento;
3. Se a saída da rede (*y*) for diferente da saída desejada (*d*), modificar os pesos w_i de acordo com a equação: $\Delta w_i = \eta.e.x_i$, onde η é o taxa de aprendizagem e $e = (d - y)$;
4. Parar se $e = 0$ para todos os vetores do conjunto de treinamento ou outro critério de parada for satisfeito. Senão, voltar para o passo 2.

O *perceptron* funciona como um classificador linear capaz de diferenciar duas classes que tenham seus elementos dispostos numa forma que possam ser separados por um hiperplano cuja posição é determinada pelos pesos e limiares (Figura 3.5 [Beale & Jackson 1991]). Expandindo o número de nós, o *perceptron* é capaz de diferenciar mais de duas classes. No entanto, as classes devem ser linearmente separáveis. Desta forma, apesar da atenção voltada para o surgimento da área, este modelo apresentava uma grande limitação: capacidade de resolver apenas problemas linearmente separáveis.

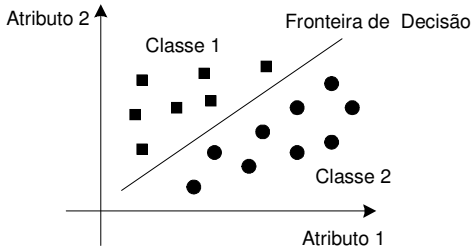

Figura 3.5 - Fronteira de decisão de um classificador linear

Posteriormente observou-se que se uma combinação linear de vetores resultantes de outros *perceptrons* for fornecida como entrada a um *perceptron*, as regiões de decisão passam a ter formas variadas. Como resultado, problemas não linearmente separáveis podem ser resolvidos. No entanto, as funções de ativação devem ser não lineares, pois uma rede com mais de uma camada cujos nós utilizam funções de ativação lineares é equivalente a uma rede de uma camada [Mitchell 1997].

A partir desta constatação, surgiram os modelos MLP, redes adaptativas cujos nós são dispostos em camadas (Figura 3.6 [Haykin 1999]). As redes MLP não possuem conexões entre nós de uma mesma camada e entre camadas que não são adjacentes. Nestes modelos, o processamento realizado por cada nó é definido pela combinação dos processamentos realizados pelos nós da

camada anterior conectados a ele. Por esta razão, quando se segue da primeira camada intermediária em direção à camada de saída, as funções implementadas se tornam cada vez mais complexas.

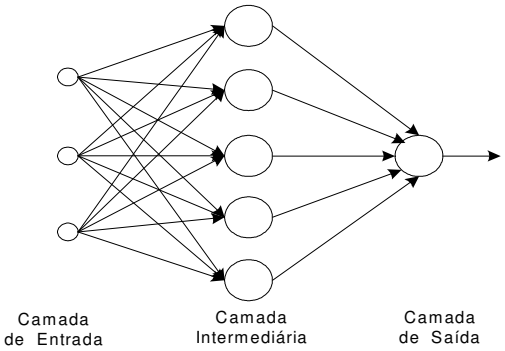

Figura 3.6 - Rede MLP

Nas redes MLP, cada camada tem uma função específica. A camada de entrada é responsável pela propagação dos valores de entrada para as camadas seguintes. As camadas intermediárias ou escondidas funcionam como extratoras de características cujos pesos são uma codificação das características presentes nos exemplos de entrada [Two Crows 1999]. As características capturadas pelos nós intermediários não são explícitas no conjunto de dados e são bastante relevantes para a rede aprender a função alvo. A camada de saída recebe os estímulos das camadas intermediárias e fornece a resposta da rede.

Os primeiros modelos MLP foram criados baseando-se no teorema de Kolmogorov que afirma que três camadas de *perceptrons* são suficientes para a representação de qualquer região de decisão [Kolmogorov 1957]. Apesar deste teorema retratar a potencialidade da estrutura em resolver problemas, ele não descreve como é possível fazer com que a estrutura aprenda a resolvê-los. Portanto, uma das mais importantes contribuições que impulsionou decisivamente o interesse pelas RNA foi o algoritmo de aprendizagem *Backpropagation* [Rumelhart et al. 1986]. *Backpropagation* é um dos métodos mais gerais e simples para treinamento supervisionado de RNA *feedforward* (acíclica) com múltiplas camadas [Duda et al. 2000].

No *Backpropagation*, o treinamento ocorre em duas fases: *forward* e *backward*. Na fase *forward*, um exemplo é apresentado à camada de entrada. A atividade resultante flui através da rede, camada por camada, até que a resposta seja produzida pela camada de saída. Na fase *backward*, a resposta fornecida pela rede é comparada com a saída desejada para o exemplo atual. Se a saída da rede não estiver correta, o erro é calculado e retropropagado até a primeira camada intermediária. O erro é usado para realizar ajustes nos pesos de cada nó de forma proporcional aos erros cometidos individualmente. O processo é repetido por várias iterações (ou épocas) para todo o conjunto de treinamento até que a saída da rede seja próxima à saída desejada. Os pesos são ajustados através de uma extensão do método gradiente descendente, visando minimizar a soma do erro quadrático (Equação 3.1, onde *d* se refere aos exemplos de treinamento e *k* aos nós de saída) entre a saída fornecida pela rede (o_{kd}) e a saída desejada (t_{kd}) [Mitchell 1997]. O erro quadrático médio (Equação 3.2, onde *N* é o número de exemplos de treinamento) também é bastante utilizado no processo de minimização.

$$E(w) \equiv \frac{1}{2} \sum_{d} \sum_{k} (t_{kd} - o_{kd})^2 \tag{3.1}$$

$$E(w) \equiv \frac{1}{2N} \sum_{d} \sum_{k} (t_{kd} - o_{kd})^2 \tag{3.2}$$

De maneira genérica, o ajuste dos pesos pode ser descrito pela Equação 3.3:

$$\Delta w_{ij}(n) = - \eta \frac{\partial E}{\partial w_{ij}} \tag{3.3}$$

onde η é a taxa de aprendizagem e $\partial E/\partial w_{ij}$ é a derivada parcial do erro E em relação ao peso w_{ij}. Como o gradiente especifica a direção e o sentido que produz o maior aumento na taxa de erro e o objetivo é mover o vetor de pesos na direção que minimize esta taxa, utiliza-se o sinal negativo antes da derivada. η assume valores no intervalo [0,1] e controla a magnitude dos ajustes dos pesos. Assim, a taxa de aprendizagem afeta a velocidade de convergência e a estabilidade da rede durante o treinamento. A fim de evitar grandes oscilações nos valores dos pesos, normalmente η assume valores pequenos.

Em alguns casos, a equação de ajuste dos pesos é acrescida do termo *momentum*, que tende a fazer com que os ajustes mantenham sempre a mesma direção, considerando os ajustes realizados anteriormente. O termo *momentum* normalmente assume valores no intervalo [0,1]; aumenta a velocidade de aprendizagem, acelerando o treinamento em regiões muito planas da superfície de erro; e evita perigo de instabilidade, suprimindo oscilações em vales e ravinas [Jang et al. 1997]. O ajuste dos pesos acrescido do termo *momentum* (α) é descrito pela Equação 3.4.

$$\Delta w = - \eta \frac{\partial E}{\partial w} + \alpha \, \Delta w_{anterior} \tag{3.4}$$

Após o treinamento, a rede está apta a ser testada. Nesta fase, exemplos diferentes dos utilizados durante o treinamento são apresentados à rede a fim de verificar sua capacidade de generalização [Monteiro 1999].

Como qualquer algoritmo baseado no método gradiente descendente, o *Backpropagation* tem problemas com convergência e eficiência (normalmente é lento, passível de estacionar em um mínimo local e seu desempenho piora sensivelmente para problemas maiores e mais complexos).

Com o intuito de gerar soluções mais eficientes e eficazes, diversas heurísticas e variações do *Backpropagation* foram propostas. *Backpropagation* com *momentum* [Rumelhart & McClelland 1986], *Quick Propagation* (QPROP ou Quickprop) [Fahlman 1988], *Resilient Propagation* (RPROP) [Riedmiller & Braun 1993], *Levenberg-Marquadt* [Hagan & Menhaj 1994] e treinamento por modos deslizantes (SM-BP) [Parma et al. 1998] são alguns exemplos de variações. Apesar da proposição de diversas variações, o algoritmo *Backpropagation* ainda é o mais amplamente utilizado devido à sua simplicidade e amplo escopo de aplicação [Braga et al. 2003].

3.5. Projetos de RNA

3.5.1. Funções de Ativação

Um dos principais aspectos relacionados a um projeto de RNA diz respeito à função de ativação, que limita a amplitude da saída de um nó [Haykin 1999]. A Figura 3.7 [Haykin 1999] [Braga et al. 2000] ilustra alguns exemplos de função de ativação.

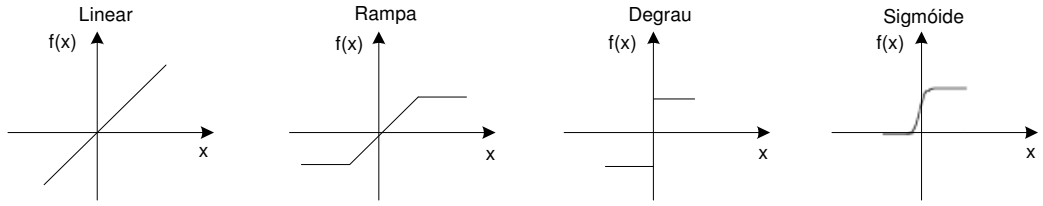

Figura 3.7 - Funções de ativação

A função de ativação linear é definida pela Equação 3.5, onde *k* é um número real positivo (inclinação da reta) que define a saída linear *y* para os valores de entrada *x*.

$$y(x) = k\,x \tag{3.5}$$

A função linear pode ser restringida para produzir valores constantes em um intervalo [-γ,+γ]. Neste caso, obtém-se a função rampa, descrita pela Equação 3.6.

$$y(x) = \begin{cases} +\gamma, \ se \ \ x \geq +\gamma \\ x, se \ |x| < +\gamma \\ -\gamma, \ se \ \ x \leq -\gamma \end{cases} \tag{3.6}$$

A função degrau ou passo, definida pela Equação 3.7, é similar à função sinal, pois produz a saída $+\gamma$ para os valores de *x* maiores que 0 e $-\gamma$, caso contrário.

$$y(x) = \begin{cases} +\gamma, \ se \ \ x > 0 \\ -\gamma, \ se \ \ x \leq 0 \end{cases} \tag{3.7}$$

Diversas funções de ativação têm sido propostas para as redes multicamadas. Estas funções são não-lineares e diferenciáveis. No caso dos algoritmos baseados no método gradiente descendente, as funções precisam ser diferenciáveis para que o gradiente possa ser calculado. As funções sigmóides são as mais utilizadas. Entre as funções sigmóides, destacam-se a logística e a tangente hiperbólica, definidas pelas Equações 3.8 e 3.9, respectivamente. A saída da função logística está contida no intervalo [0,1] e aumenta monotonicamente com sua entrada. A tangente hiperbólica assume valores no intervalo [-1,1] e pode tornar o treinamento mais rápido [Haykin 1999].

$$y(x) = \frac{1}{1 + e^{-x}} \tag{3.8}$$

$$y(x) = tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.9}$$

As funções de ativação dos nós de saída podem ser diferentes das funções de ativação dos nós intermediários. O mesmo se estende para os nós individuais, onde cada nó pode apresentar uma função de ativação diferente [Duda et al. 2000].

3.5.2. Arquitetura da Rede

A arquitetura de uma RNA é definida pelos seguintes parâmetros: número de camadas, número de nós em cada camada, tipo de conexão entre os nós e topologia da rede.

O arranjo dos nós e conexões tem um impacto profundo na capacidade de processamento das RNA, pois a estrutura da rede afeta diretamente sua performance e restringe o tipo de problema que pode ser resolvido [Bigus 1996]. Redes com uma camada de nós MCP, por exemplo, só conseguem resolver problemas linearmente separáveis. Redes recorrentes, por sua vez, são mais adequadas para resolver problemas que envolvem processamento temporal.

Quanto ao número de camadas, podem-se ter redes com uma ou mais camadas intermediárias. Redes com uma camada intermediária são capazes de aproximar qualquer função contínua ou booleana [Cybenko 1989], enquanto que redes com duas camadas intermediárias podem aproximar qualquer função matemática [Cybenko 1988]. Apesar da utilização de duas ou mais camadas intermediárias poder facilitar o treinamento da rede em alguns problemas, não é recomendado a utilização de várias camadas intermediárias, pois cada vez que o erro medido durante o treinamento é propagado para a camada anterior, ele se torna menos útil e preciso [Braga et al. 2000].

A determinação do número de nós de entrada e saída depende da dimensionalidade dos vetores de entrada e saída, respectivamente. O número de nós intermediários depende da complexidade da função a ser aprendida, número de exemplos de treinamento e quantidade de ruído. Quanto maior o número de nós intermediários, mais complexas são as funções mapeadas pela rede. No entanto, um número excessivo pode levar a resultados indesejáveis, como *overfitting*. Por outro lado, um número pequeno pode impedir que a rede aprenda (*underfitting*) [Duda et al. 2000].

Diversas pesquisas têm sido direcionadas para determinar automaticamente a quantidade de camadas e nós intermediários. Escolha baseada nos dados, inicialização com redes grandes e remoção gradual de nós e pesos irrelevantes, inicialização com redes pequenas e adição gradual de nós e pesos, e aplicação de Algoritmos Genéticos são alguns exemplos de técnicas propostas [Bigus 1996]. Embora diversas técnicas tenham sido desenvolvidas, a determinação do número de camadas e nós intermediários ainda é um problema em aberto [Azevedo et al. 2000].

Com relação ao tipo de conexão, os nós podem ter conexões do tipo *feedforward* ou *feedback*. Nas redes *feedforward* (acíclica, não-recorrente ou direta), a saída de um nó na *i*-ésima camada não pode ser usada como entrada de nós em camadas de índice menor ou igual a *i*. Nas redes com *feedback* (cíclica, recorrente com realimentação ou com retroação), a saída de um nó na *i*-ésima camada é usada como entrada de nós em camadas de índice menor ou igual a *i* [Azevedo et al. 2000].

Quanto à conectividade, as RNA podem ser classificadas como fracamente (ou parcialmente) conectada ou completamente conectada.

3.5.3. Critérios de Parada

Uma dúvida que normalmente surge durante a realização de experimentos com uma RNA é quando parar o treinamento da rede. Existem vários métodos para determinar o momento em que o treinamento deve ser encerrado. Estes métodos são chamados critérios de parada. A escolha deste critério é muito importante, pois poucas iterações podem reduzir o erro de forma insuficiente e muitas podem conduzir ao *overfitting* [Mitchell 1997]. Os critérios de parada mais utilizados são [Braga et al. 2000] [Duda et al. 2000]:

- ♣ Erro de treinamento abaixo de uma constante;
- ♣ Aplicação de algum critério baseado no conjunto de validação;
- ♣ Porcentagem de classificações corretas acima de uma constante;
- ♣ Máximo de épocas (número de vezes que o conjunto de treinamento é apresentado à rede);
- ♣ Combinação dos métodos acima.

Como o algoritmo de treinamento pode ser susceptível a *overfitting*, a escolha de um critério de parada que se baseia apenas no conjunto de treinamento pode não ser adequada [Mitchell 1997].

3.5.4. Mínimos Locais e Regiões Planas na Superfície de Erro

A forma irregular da superfície de erro (mínimos locais e regiões planas), decorrente das não-linearidades das funções de ativação, causa dificuldades para o treinamento de RNA com algoritmos baseados no gradiente descendente, como o *Backpropagation* [Braga et al. 2003].

Diversas técnicas têm sido propostas para evitar regiões de gradiente nulo (apresentam uma solução estável, embora não seja a melhor). O ajuste adaptativo da taxa de aprendizagem e a adição do termo *momentum* à equação de ajuste dos pesos são alguns exemplos dessas técnicas.

Pequenos valores da taxa de aprendizagem η resultam em maior sensibilidade às variações da superfície, enquanto valores maiores podem evitar regiões planas e mínimos locais. O valor de η influencia também a velocidade de convergência. Se η é muito grande, os ajustes dos pesos são de maior magnitude e o erro tende a diminuir mais rapidamente. No entanto, se η excede um valor crítico, a rede fica instável [Haykin 1999]. Por outro lado, se η é muito pequena, a convergência se torna lenta e o treinamento poder ficar preso em mínimos locais. Outra observação se refere à utilização de taxas de aprendizagem diferenciadas. Como os sinais de erro das primeiras camadas tendem a ser menores, as taxas de aprendizagem dos nós destas camadas deveriam ser maiores [Jang et al. 1997].

Outra alternativa para evitar regiões de gradiente nulo é a adição do termo *momentum* à equação de ajuste dos pesos [Phansalkar & Sastry 1994].

O termo *momentum* é responsável pelo acúmulo dos ajustes anteriores, o que resulta em um termo residual quando o ajuste atual é nulo por causa do gradiente. Dependendo do valor do *momentum*, regiões planas e mínimos locais podem ser evitados. A utilização do *momentum* pode resultar em melhor qualidade de treinamento, no entanto é mais um parâmetro a ser ajustado pelo usuário. Seu valor também pode ser ajustado durante o treinamento [Braga et al. 2003].

Além da utilização de taxa de aprendizagem decrescente e do termo *momentum*, outras técnicas utilizadas para acelerar a aprendizagem e reduzir a incidência de mínimos locais são [Beale & Jackson 1991] [Mitchell 1997]:

- ♣ Atualizar os pesos após o cálculo do erro para cada exemplo de treinamento;
- ♣ Treinar várias redes com os mesmos dados e pesos iniciais diferentes;
- ♣ Inserção de ruído aos dados. O ruído deve ser suficiente para impedir *overfitting*, mas não deve ser excessivo, para não confundir a rede;
- ♣ Mínimos locais podem ocorrer quando duas ou mais classes são categorizadas como a mesma. Isto pode ser conseqüência de uma representação ruim nos nós intermediários. Assim, a adição de nós intermediários permitirá uma melhor codificação das entradas e reduzirá a ocorrência desses mínimos.

3.5.5. Medidas de Desempenho

Antes da escolha da melhor configuração da rede, diversos experimentos são realizados, gerando uma quantidade significativa de redes alternativas. A fim de verificar qual rede apresenta o melhor desempenho é preciso utilizar alguma medida que possibilite especificar o desempenho de cada rede.

Uma medida de desempenho bastante utilizada quando as RNA são aplicadas em problemas de classificação é a taxa de erro. Normalmente, a taxa de erro é obtida comparando a classe verdadeira de cada exemplo com o rótulo atribuído pela rede [Monard & Baranauskas 2003].

Além da taxa de erro, outras medidas podem ser usadas. Uma medida bastante utilizada é o erro quadrático, que fornece o valor do erro para cada exemplo [Duda et al. 2000]. Normalmente utiliza-se a soma ou a média destes valores para todo o conjunto de exemplos.

Existem vários métodos para estimar uma medida de erro verdadeira. Os métodos mais difundidos foram descritos na Seção 2.2.6.

3.5.6. Underfitting, Overfitting e Overtuning

Underfitting, *overfitting* e *overtuning* são alguns dos problemas que podem ocorrer durante o treinamento de uma RNA [Monard & Baranauskas 2003].

Quando a rede neural realiza a indução a partir dos exemplos, é possível que a hipótese obtida seja muito específica para o conjunto de treinamento. Como o conjunto de treinamento é apenas uma amostra, é possível que as hipóteses induzidas melhorem o desempenho da rede neste conjunto, enquanto pioram em exemplos diferentes. Neste caso, diz-se que a rede ajusta-se em excesso ao conjunto de treinamento, gravando suas peculiaridades e ruídos, ou que houve *overfitting*. O *overfitting* pode ocorrer quando a rede tem mais parâmetros ajustáveis do que o necessário para a resolução do problema.

Uma forma de evitar *overfitting* é estimar o erro de generalização durante o treinamento. Para isto, o conjunto de dados é dividido em conjunto de treinamento e conjunto de validação. O conjunto de treinamento é utilizado para ajustar os pesos e o conjunto de validação para estimar a capacidade de generalização da rede durante o processo de aprendizagem. O treinamento deve ser interrompido quando o erro do conjunto de validação começar a subir [Prechelt 1994]. Embora esta alternativa seja

eficiente em algumas situações, sua utilização é limitada para os casos em que um conjunto de treinamento grande está disponível, pois os dados de validação não podem ser utilizados para o treinamento. Outras soluções são a aplicação de técnicas de poda, que eliminam pesos e nós irrelevantes, e a redução dos valores dos pesos durante cada iteração, evitando a formação de superfícies de decisão muito complexas [Mitchell 1997].

Overtuning pode ocorrer quando um modelo é bastante ajustado com o intuito de otimizar seu desempenho em todos os exemplos disponíveis. Da mesma forma que o *overfitting*, *overtuning* pode ser detectado e evitado utilizando uma parte dos exemplos disponíveis para a construção do indutor e o restante para um teste do modelo induzido.

Outro tipo de problema pode ocorrer quando poucos exemplos representativos são fornecidos ou a rede tem menos parâmetros ajustáveis do que o necessário. Neste caso, é provável que o modelo encontrado não apresente bom desempenho tanto nos exemplos de treinamento como nos exemplos de teste, situação conhecida como *underfitting* [Rezende et al. 2003]. Portanto, o objetivo do treinamento deve ser encontrar o ajuste ideal na fronteira entre o *overfitting* e o *underfitting*. No entanto, estimar o número de parâmetros é uma tarefa difícil que requer conhecimento sobre a complexidade do problema. Este conhecimento normalmente não está disponível, principalmente em problemas multidimensionais [Braga et al. 2003].

Várias abordagens tentam resolver estes problemas, entre elas estão os métodos construtivos [Fahlman & Lebiere 1998] e os algoritmos de poda [Mozer & Smolensky 1989] [LeCun et al. 1990]. Os algoritmos construtivos visam à construção gradual da rede por meio da adição de nós até que um critério de parada envolvendo erro de treinamento e generalização seja alcançado. Os algoritmos de poda, por sua vez, visam à diminuição da estrutura da rede pela eliminação gradativa de pesos e nós. Estas abordagens não garantem convergência, pois sofrem dos mesmos problemas de treinamento inerentes às redes de múltiplas camadas. Além disso, os algoritmos requerem o ajuste de parâmetros de treinamento adicionais pelo usuário, aos quais o modelo resultante é bastante sensível.

3.5.7. Freqüência de Atualização dos Pesos

Outro aspecto que precisa ser observado em projetos de RNA é a freqüência de atualização dos pesos, pois esta influencia o desempenho do modelo. Duas abordagens têm sido utilizadas com relação à freqüência de ajuste dos pesos: por padrão (*on-line*, seqüencial ou estocástico) e por ciclo (*batch* ou *off-line*) [Haykin 1999] [Braga et al. 2000]. A escolha da abordagem a ser utilizada depende da aplicação e da distribuição estatística dos dados.

Na abordagem por padrão, os pesos são atualizados após a apresentação de cada exemplo de treinamento. Esta abordagem é estável se a taxa de aprendizagem for pequena, sendo, portanto, aconselhável reduzir a taxa progressivamente. A abordagem por padrão requer menos memória e geralmente é mais rápida, sobretudo se o conjunto de treinamento for grande e redundante.

Na abordagem por ciclo, os pesos são atualizados após todos os exemplos terem sido apresentados. Esta abordagem normalmente é mais estável. No entanto, pode ser lenta se o conjunto de treinamento for grande e redundante.

3.6. Redes Neurais e KDD

Na maioria das aplicações de KDD, o grau de compreensibilidade é considerado um fator bastante importante durante a escolha da técnica de mineração de dados. De acordo com este grau, os sistemas de aprendizagem podem ser classificados como [Monard & Baranauskas 2003]:

- Sistemas tipo caixa-preta: desenvolvem uma representação interna que pode não ser facilmente interpretada e não fornecem explicação do processo de reconhecimento. As RNA é um exemplo deste tipo de sistema. Conseqüentemente, as RNA têm sido consideradas inadequadas para serem utilizadas em aplicações de KDD onde a representação do conhecimento numa forma compreensível é um fator primordial [Lu et al. 1995] [Craven & Shavlik 1998] [Han & Kamber 2001];
- Sistemas orientados a conhecimento: objetivam a criação de estruturas simbólicas compreensíveis.

As RNA são conhecidas pelo bom desempenho que geralmente obtêm quando utilizadas em uma grande variedade de aplicações [Ludermir et al. 2003]. Entretanto, em várias aplicações reais, é importante não apenas o desempenho obtido, mas também a facilidade do usuário compreender como a rede alcança suas decisões, pois este poderá aprender e incorporar o conhecimento em suas atividades, e terá mais confiança nos resultados alcançados [Adriaans & Zantinge 1996] [Amorim et al. 2001] [Huang & Xing 2002]. Essa demanda por explicação torna-se mais evidente em sistemas que primam pela segurança na operação. Este é o caso de problemas como controle de usinas nucleares, controle de navegação de aeronaves, auxílio a cirurgias médicas entre outros [Ludermir et al. 2003]. Nestes tipos de problemas, o processo de KDD não deve finalizar quando o modelo neural é definido. Após esta fase, é necessário representar o conhecimento incorporado pela rede numa forma compreensível. Isto pode ser alcançado através da aplicação de técnicas representam o conhecimento da rede neural na forma de visualizações, regras e relacionamentos facilmente compreensíveis. Estas técnicas podem ser divididas em diversas abordagens, tais como [Bigus 1996]:

- Tratar a rede como uma caixa-preta, apresentando as entradas e registrando as saídas. Esta abordagem é conhecida como análise de sensibilidade à entrada e permite identificar o impacto de um atributo de entrada na saída de um modelo. Para determinar este impacto, é necessário manter os valores dos outros atributos constantes e variar apenas os valores do atributo analisado. Se durante a variação a saída é consideravelmente alterada, então o atributo é importante. O conhecimento obtido por este tipo de análise pode ser representado em regras do tipo *"Se x decresce 5% Então y aumenta 8%"* [Han & Kamber 2001]. Uma abordagem mais automática para realizar esta análise em redes treinadas com a retropropagação do erro é manter os erros computados. Computando o erro até a camada de entrada, obtém-se o grau para o qual cada entrada contribui para o erro da saída. Assim, a entrada com maior erro tem o maior impacto na saída. Acumulando os erros e normalizando-os, no final do processo de aprendizagem é possível computar a contribuição de cada entrada para os erros da saída;
- Apresentar os dados de entrada à rede e gerar um conjunto de regras que descrevam as funções realizadas pela rede, baseando-se nos seus estados internos e pesos. Esta provisão pode ser obtida a partir da integração das RNA com técnicas da IA Simbólica, resultando nos Sistemas Neurais Híbridos [Haykin 1999];

- Representar a rede visualmente, a fim de que o usuário contribua para o processo de aprendizagem e obtenha alguma informação sobre o conhecimento incorporado pela rede.

  Os pesos da rede contêm informação sobre a importância relativa dos atributos de entrada e a correlação entre eles. De alguma forma, a magnitude absoluta e o sinal dos pesos podem ser usados como uma indicação da importância das entradas. Uma das técnicas de visualização mais utilizadas é o diagrama de Hinton, uma coleção de caixas cujo tamanho representa a magnitude relativa do peso e cuja cor depende do sinal [Bigus 1996].

  Ao contrário de técnicas como o diagrama de Hinton, se a tarefa de mineração de dados é de alto nível, os resultados devem ser apresentados sobre uma perspectiva de alto nível.

Essas técnicas, que permitem a representação do conhecimento incorporado pelas RNA numa forma compreensível, contribuem para uma maior aceitação das RNA como uma alternativa bastante viável para ser usada em diversas aplicações de KDD [Han & Kamber 2001].

3.7. Considerações Finais

Embora as RNA tenham se mostrado uma técnica eficiente para a solução de um grande número de aplicações, não é correto afirmar que elas são suficientes para resolver qualquer tarefa. As RNA apresentam várias limitações que inviabilizam seu uso exclusivo para a solução de uma quantidade significativa de problemas [Braga et al. 2000].

Uma das maiores limitações das RNA é a dificuldade de gerar uma explicação de como a rede representa a solução de um problema, pois o conhecimento aprendido é representado de forma implícita pela topologia, valores dos pesos e limiares da rede; e o espaço de representação é dividido em regiões complexas por meio da combinação de várias funções matemáticas [Azevedo et al. 2000].

Diversas técnicas para extração do conhecimento de RNA têm sido desenvolvidas. A aplicação dessas técnicas permite que o conhecimento incorporado pela rede seja representado numa forma compreensível. Conseqüentemente, as RNA tornam-se mais adequadas para o processo de KDD, ampliando seu escopo de aplicação. Entre as soluções propostas, os Sistemas Neuro-*Fuzzy* têm mostrado resultados satisfatórios. Esses sistemas permitem que o conhecimento adquirido pela rede seja representado na forma de regras *fuzzy* Se-Então. Outras técnicas utilizam uma RNA treinada e um conjunto de exemplos para extrair uma representação simbólica do conhecimento codificado pela rede. Os Sistemas Neuro-*Fuzzy* e as técnicas de extração de conhecimento simbólico de RNA são descritos nos Capítulos 4 e 5, respectivamente.

Capítulo 4

Sistemas Neuro-*Fuzzy*

4.1. Introdução

A dificuldade ou impossibilidade de obter informações e equacionar a realidade imprecisa do mundo real levou alguns pesquisadores a propor lógicas alternativas às lógicas clássicas que seriam mais propícias à representação do conhecimento [Azevedo et al. 2000]. Uma destas proposições é a Lógica *Fuzzy* [Zadeh 1965], que é capaz de lidar lingüisticamente com informações incompletas e imprecisas presentes na linguagem natural, sem diminuir a expressividade.

A Lógica *Fuzzy* fornece um mecanismo sistemático para realizar computação numérica, usando termos lingüísticos associados a funções de pertinência, e permite que o conhecimento seja representado na forma de regras *fuzzy* Se-Então [Zimmermann 1991]. Por ser estruturado próximo à linguagem natural, este tipo de regra constitui uma forma clara e eficiente de representação do conhecimento. Conseqüentemente, sistemas baseados na Lógica *Fuzzy* podem modelar, de forma mais simples que a matemática convencional, problemas complexos [Ramos 2001].

Apesar de apresentar várias potencialidades, os Sistemas *Fuzzy* não são capazes de se adaptar. Uma solução que vem sendo bastante adotada para superar esta limitação é o desenvolvimento de Sistemas Neuro-*Fuzzy*, resultantes da integração de Sistemas *Fuzzy* e RNA [Almeida & Evsukoff 2003]. Além de possibilitar o ajuste dos parâmetros dos conjuntos e regras *fuzzy* a partir de um conjunto de dados, tais sistemas permitem a incorporação de conhecimento na fase de definição da rede e a extração do conhecimento refinado após o treinamento.

Este capítulo aborda o paradigma dos Sistemas Neuro-*Fuzzy* como uma opção para extração de conhecimento simbólico de RNA. Inicialmente, são apresentados uma introdução sobre a Lógica *Fuzzy* e os conceitos fundamentais associados aos Sistemas *Fuzzy*. Em seguida, são fornecidas uma introdução dos Sistemas Neuro-*Fuzzy* e uma descrição detalhada dos modelos FuNN e FWD, que devido às suas funcionalidades foram escolhidos para serem investigados nesta dissertação. Os resultados apresentados por tais modelos serão comparados com os obtidos por uma rede MLP e TREPAN, técnica de extração de conhecimento de RNA que será abordada no Capítulo 5, juntamente com as técnicas de extração de regras dos modelos FuNN e FWD.

4.2. Lógica *Fuzzy*

Dois dos principais aspectos da imperfeição da informação são a imprecisão e a incerteza. As teorias mais conhecidas para tratar destes aspectos são Teoria dos Conjuntos Clássicos e Teoria de Probabilidades. Estas teorias, embora sejam úteis em várias aplicações, nem sempre conseguem captar a riqueza da informação fornecida pelos seres humanos. A Teoria dos Conjuntos Clássicos não é capaz de tratar o aspecto vago da informação e a Teoria de Probabilidades é mais adequada para tratar informações usuais [Sandri & Correa 1999]. Percebendo as limitações dessas teorias, [Zadeh 1965] expandiu a idéia dos conjuntos clássicos e deu origem aos conjuntos *fuzzy*.

A Lógica *Fuzzy*, também conhecida como Lógica Nebulosa ou Lógica Difusa, se baseia na Teoria dos Conjuntos *Fuzzy* e consiste de um conjunto de princípios matemáticos que modela a informação através de graus de pertinência. Os graus de pertinência permitem uma transição gradual entre a pertinência total e a não-pertinência e fornecem flexibilidade aos conjuntos *fuzzy* para modelar expressões lingüísticas [Almeida & Evsukoff 2003]. Por ser capaz de lidar com o aspecto vago da informação, a Lógica *Fuzzy* permite a representação de algumas características do raciocínio humano [Brasil 1999]. Desta forma, seus conceitos podem ser utilizados para traduzir em termos matemáticos a informação imprecisa presente na linguagem natural.

A Lógica *Fuzzy* tem sido cada vez mais aplicada em sistemas que utilizam informações fornecidas por seres humanos para automatizar processos. Seu escopo de aplicação é bastante amplo, indo do controle de eletrodomésticos ao controle de satélites, do mercado financeiro à medicina, e tende a crescer cada vez mais, sobretudo em sistemas híbridos que incorporam abordagens conexionistas e evolutivas (*soft computing*) [Sandri & Correa 1999].

O Apêndice A apresente uma comparação entre a Teoria dos Conjuntos *Fuzzy* e a Teoria dos Conjuntos Clássicos, e os principais conceitos e operações associados à Lógica *Fuzzy*.

4.3. Sistemas *Fuzzy*

Os Sistemas *Fuzzy* representam a principal ferramenta de modelagem baseada na Lógica *Fuzzy* e têm sido aplicados com sucesso em diversas áreas, tais como controle, diagnose, análise de decisão, robótica, reconhecimento de padrões e predição de séries temporais [Jang et al. 1997].

Parte do sucesso dos Sistemas *Fuzzy* é proveniente das variáveis lingüísticas e dos conjuntos *fuzzy* que fornecem um *framework* intuitivo para representação e incorporação do conhecimento de especialistas na fase de modelagem do sistema [Jang et al. 1997]. Tal conhecimento é adquirido simbolicamente, mas é representado e processado numericamente.

Os modelos *fuzzy* de Mandani, Sugeno e Tsukamoto são alguns exemplos de sistemas *fuzzy* que têm sido amplamente utilizados [Almeida & Evsukoff 2003]. A diferença entre esses sistemas está na especificação das conclusões das regras e, conseqüentemente, nos mecanismos de raciocínio, agregação e defuzzificação [Jang 1993].

De um modo geral, as principais características dos Sistemas *Fuzzy* são:

- Representação do conhecimento na forma de regras *fuzzy* Se-Então;
- Uso de um mecanismo de inferência responsável pelo raciocínio *fuzzy*;

- Capacidade de integrar o mundo lingüístico (simbólico) com o numérico, o que facilita a interface homem-máquina;
- Capacidade de lidar com as complexidades resultantes das não-linearidades dos sistemas modelados (as regras *fuzzy* podem simular qualitativamente o comportamento de um sistema para o qual não existe um modelo matemático preciso disponível).

4.3.1. Processamento

A maioria dos sistemas baseados na Lógica *Fuzzy* compreende as seguintes etapas [Marsh et al. 1992]: fuzzificação, avaliação de regras e defuzzificação. A Figura 4.1 [Azevedo et al. 2000] mostra como estas etapas interagem.

A fuzzificação é o primeiro passo no processamento de um sistema *fuzzy*. Esta etapa recebe valores do meio externo e, através das funções de pertinência associadas à parte antecedente das regras, efetua o mapeamento destes valores para graus de pertinência.

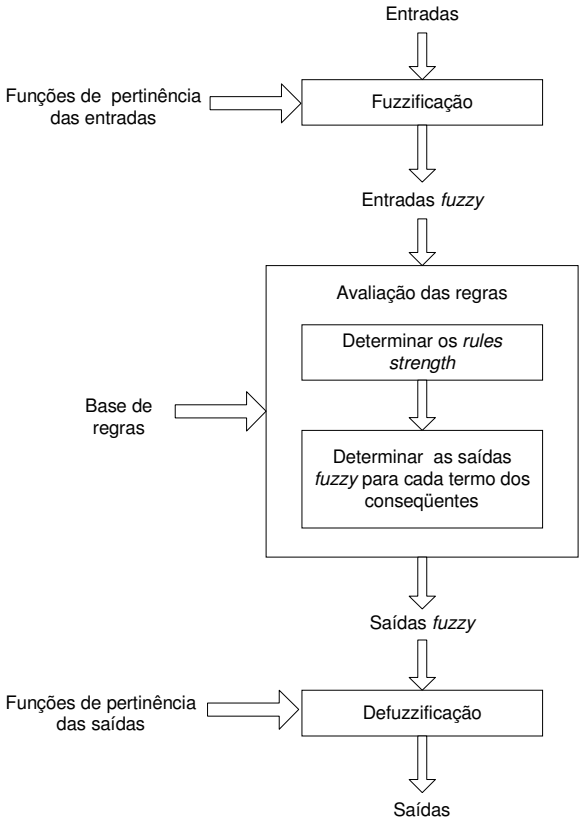

Figura 4.1 - Etapas do processamento de um sistema *fuzzy*

A avaliação das regras, também conhecida como inferência *fuzzy*, é o segundo passo. O processo de avaliação pode ser descrito da seguinte forma [Jang et al. 1997]:

- ♣ Encontrar o grau de verdade (*rule strength* ou *firing strength*) para cada regra: O grau de verdade indica o grau para o qual a parte antecedente da regra é satisfeita. Seu valor é obtido através da aplicação do operador *fuzzy* presente na parte antecedente da regra;
- ♣ Determinar a saída *fuzzy*: Esta fase compreende a aplicação de um operador de agregação em todas as regras que envolvem o mesmo termo do conseqüente. O máximo e a soma disjunta são os operadores mais utilizados. Existe uma saída *fuzzy* para cada conjunto *fuzzy* da saída.

A base de regras *fuzzy* é formada por estruturas do tipo Se-Então com a seguinte sintaxe [Zimmermann 1991]:

$$Se\ condição_1\ op\ ...\ op\ condição_n\ \ Então\ conclusão_1\ op\ ...\ op\ conclusão_m$$

onde *op* é um operador *fuzzy* e as condições e conclusões apresentam a forma *variável = termo* ou, lingüisticamente, *variável é termo*, onde *termo* é um dos termos lingüísticos dos conjuntos *fuzzy* associados à *variável*.

A defuzzificação é o último passo no processamento de um sistema *fuzzy*. Sua função é combinar todas as saídas *fuzzy* e gerar um resultado não *fuzzy*. Portanto, esta etapa é necessária apenas em problemas em que uma solução não *fuzzy* é requerida.

Os métodos de defuzzificação mais utilizados são [Cox 1994]:

- ♣ Primeiro Máximo (*Smallest of Maximum* – SOM): Encontra o valor de saída através do ponto em que o grau de pertinência atinge o primeiro valor máximo;
- ♣ Média dos Máximos (*Mean of Maximum* – MOM): Encontra o ponto médio entre os valores que têm o maior grau de pertinência inferido pelas regras;
- ♣ Centro da Área (*Center of Area* – COA), Centróide ou Centro da Gravidade (*Center of Gravity* - COG): O valor de saída é o centro de gravidade da função de pertinência resultante.

Os métodos de defuzzificação apresentam diferenças em termos de velocidade e eficiência. Portanto, a seleção do método mais apropriado está diretamente relacionada com os requisitos da aplicação.

4.3.2. Modelagem e Concepção

A concepção de um sistema *fuzzy* engloba as seguintes tarefas [Jang et al. 1997]:

- ♣ Selecionar os atributos de entrada e saída relevantes;
- ♣ Escolher o tipo de sistema de inferência *fuzzy*;
- ♣ Determinar o número de termos lingüísticos associados a cada atributo;
- ♣ Projetar um conjunto de regras *fuzzy* Se-Então;
- ♣ Escolher as funções de pertinência;
- ♣ Determinar os parâmetros das funções de pertinência;
- ♣ Refinar os parâmetros das funções de pertinência, aplicando técnicas de regressão e otimização que utilizam um conjunto de dados.

Estas tarefas são realizadas com o auxílio de especialistas do domínio ou por tentativa e erro.

Um ponto crítico na modelagem *fuzzy* é o processo de aquisição de conhecimento, pois a performance de um sistema *fuzzy* é extremamente dependente do grau de precisão alcançado neste processo. Normalmente, o conhecimento é extraído de um especialista ou de um equipamento existente que realiza, de forma adequada, a tarefa alvo. Entretanto, especialistas não estruturam seus processos de tomada de decisões numa maneira formal. Como resultado, o processo de transcrição do conhecimento em uma base de regras e funções de pertinência adequadas é custoso e não-trivial, sendo normalmente realizado de forma empírica. Desta forma, é necessário desenvolver métodos de aquisição de conhecimento mais eficientes e sistemáticos [Medeiros 1996].

Pesquisas recentes estão sendo direcionadas para concepção e projeto de sistemas *fuzzy* com capacidade de aprendizagem a partir de exemplos. Sistemas Neuro-*Fuzzy*, resultantes da integração de Sistemas *Fuzzy* com RNA, têm sido bastante utilizados para este fim. Neste caso, a RNA é responsável por ajustar os parâmetros do sistema *fuzzy* a partir de um conjunto de dados [Bigus 1996]. Estes sistemas são descritos na próxima seção. Outra maneira de se aprender os parâmetros dos Sistemas *Fuzzy* consiste no uso de Algoritmos Genéticos [Sandri & Correa 1999].

4.4. Sistemas Neuro-*Fuzzy*

No Capítulo 3 foi visto que as RNA são capazes de aprender e generalizar o conhecimento contido no conjunto de treinamento. Apesar de ser centrada nos dados e ter apresentado bons resultados em diversos domínios, a abordagem conexionista geralmente dificulta a incorporação do conhecimento dos especialistas e a extração do conhecimento resultante do processo de aprendizagem [Evsukoff & Almeida 2003].

Na seção anterior foram introduzidos os fundamentos dos Sistemas *Fuzzy*. Um sistema *fuzzy* pode ser utilizado como sistema de apoio à decisão capaz de representar o conhecimento de especialistas e interpolar decisões a partir de entradas incertas. O formalismo matemático desses sistemas permite a representação de algumas características do raciocínio humano. Portanto, um sistema *fuzzy* pode ser considerado como centrado no homem [Evsukoff & Almeida 2003].

Assim como as RNA e os Sistemas *Fuzzy*, toda técnica de IA tem suas vantagens e deficiências para resolver diferentes tipos de problemas. Por esta razão, existem muitos problemas complexos do mundo real que não podem ser facilmente resolvidos com as técnicas atuais. Tais problemas requerem sistemas inteligentes que combinem conhecimento, técnicas e metodologias de diversas fontes, resultando na construção de Sistemas Híbridos Inteligentes (SHI) [Jang et al. 1997]. Apesar de integrar diversos métodos, os SHI apresentam pouca redundância, pois normalmente os métodos não tentam resolver o mesmo problema em paralelo. Ao invés disso, eles atuam de forma mutuamente complementar [Shapiro 2002].

Apesar de suas potencialidades, os Sistemas *Fuzzy* não são capazes de se adaptarem [Nauck 1997]. Conseqüentemente, diversas pesquisas têm sido direcionadas para a integração das potencialidades de Sistemas *Fuzzy* e RNA nos chamados Sistemas Neuro-*Fuzzy*, um subgrupo dos SHI. Tais sistemas surgem como uma alternativa capaz de solucionar as limitações apresentadas pelos paradigmas em que se baseiam quando utilizados isoladamente e, ao mesmo tempo, incorporar suas vantagens [Shapiro 2002]. Como um sistema *fuzzy*, os Sistemas Neuro-*Fuzzy* podem explicar seu

comportamento através de regras, realizar inferência e tomada de decisões, e incorporar conhecimento sobre o domínio do problema. Como uma RNA, são capazes de reconhecer padrões e se adaptar (ajustar os parâmetros dos conjuntos e regras *fuzzy*) a partir de um conjunto de dados [Sandri & Correa 1999]. Devido a essas vantagens, tais sistemas têm sido utilizados com sucesso em diversos domínios de aplicação.

Os Sistemas Neuro-*Fuzzy* normalmente são representados como uma RNA multicamadas *feedfoward* cuja estrutura é organizada de modo que as entradas da rede correspondam às entradas do sistema *fuzzy*, os nós de saída correspondam às saídas do sistema *fuzzy* e os nós intermediários representem funções de pertinência e regras *fuzzy* [Medeiros 1996] [Nauck 1997].

Nos Sistemas Neuro-*Fuzzy*, a Lógica *Fuzzy* é empregada nas unidades de processamento ou nas representações dos pesos das conexões [Bigus 1996]. A Lógica *Fuzzy* também pode ser usada para monitorar os parâmetros da rede neural. A taxa de aprendizagem e o termo *momentum*, por exemplo, podem ser modificados através de funções de pertinência derivadas do erro total de treinamento e da variação do erro entre iterações consecutivas. A utilização de regras *fuzzy* no ajuste dos parâmetros pode reduzir significativamente o tempo de treinamento [Shapiro 2002].

Os Sistemas Neuro-*Fuzzy* normalmente envolvem quatro fases [Taha & Ghosh 1999]: representação simbólica do conhecimento inicial, mapeamento deste conhecimento em uma arquitetura conexionista, treinamento da rede e extração de regras da rede treinada. Nos casos em que não existe um conhecimento prévio sobre o domínio do problema, define-se uma arquitetura inicial, realiza-se o treinamento e, de acordo com os resultados obtidos, testam-se outras arquiteturas.

Existe uma grande variedade de Sistemas Neuro-*Fuzzy*. Esses sistemas diferem principalmente nos seguintes aspectos [Kasabov 1996] [Kasabov et al. 1997] [Kasabov et al. 2001]:

- Forma de mapeamento de um conjunto de regras na estrutura da rede neural;
- Forma de observação de alguns parâmetros da rede treinada para explicar a inferência através de um conjunto de regras *fuzzy*;
- Modo de operação: reflete na maneira pela qual os pesos e as funções de ativação são modificados durante o treinamento;
- Tipo de regra *fuzzy*: reflete na estrutura conexionista. Regras simples de Zadeh-Mamdani, regras ponderadas e regras de Takagi-Sugeno são alguns exemplos;
- Método de inferência: reflete na seleção de parâmetros e funções da rede neural, e na forma como os pesos são inicializados e interpretados depois do treinamento.

4.4.1. Modelos Investigados

Estão seção aborda os modelos neuro-*fuzzy* mais difundidos. Além desses modelos, outros foram propostos, como por exemplo, FCM (*Fuzzy C-Means*) [Kronszynski & Zhou 1998], *Fuzzy*-CMAC (*Fuzzy Cerebellar Model Arithmetic Computer*) [Rezende et al. 2003] e CANFIS (*Coactive Network-Fuzzy Inference System*) [Jang et al. 1997], uma extensão do ANFIS (*Adaptive Network-based Fuzzy Inference System* ou *Adaptive Neuro Fuzzy Inference System*) para múltiplas saídas com regras *fuzzy* não-lineares.

Devido aos benefícios oferecidos pela abordagem Neuro-*Fuzzy*, diversos modelos estão constantemente sendo propostos ou aprimorados. Por estão razão, esta seção também descreve modelos propostos recentemente.

ANFIS

ANFIS [Jang et al. 1997] é o modelo neuro-*fuzzy* mais difundido na literatura [Evsukoff & Almeida 2003]. Este modelo representa uma classe de redes adaptativas que são funcionalmente equivalentes a um sistema de inferência *fuzzy*.

Usando um conjunto de dados, o modelo ANFIS constrói um sistema de inferência *fuzzy* cujos parâmetros das funções de pertinência são ajustados através de um algoritmo *backpropagation* ou de um algoritmo de aprendizagem híbrida que combina o *backpropagation* e o método LSM (*Least Squares Method*) [Malhotra & Malhotra 2002].

A Figura 4.2 [Jang et al. 1997] mostra a arquitetura de um modelo ANFIS com duas regras, duas entradas (*x* e *y*) associadas a dois conjuntos *fuzzy* (A_1 e A_2, B_1 e B_2, respectivamente) e uma saída (*f*). Tal modelo é funcionalmente equivalente ao modelo *fuzzy* de primeira ordem de Sugeno, onde a conclusão das regras é uma combinação linear das entradas, ao invés de conjuntos *fuzzy*. Os nós representados por um quadrado são adaptativos (possui um conjunto de parâmetros ajustáveis) e os nós representados por um círculo são fixos. As conexões entre os nós indicam apenas a direção do fluxo dos sinais. Diferentemente da maioria das RNA, não existem pesos associados às conexões [Shapiro 2002].

Como pode ser visto na Figura 4.2, cada camada é responsável pela execução de uma determinada função.

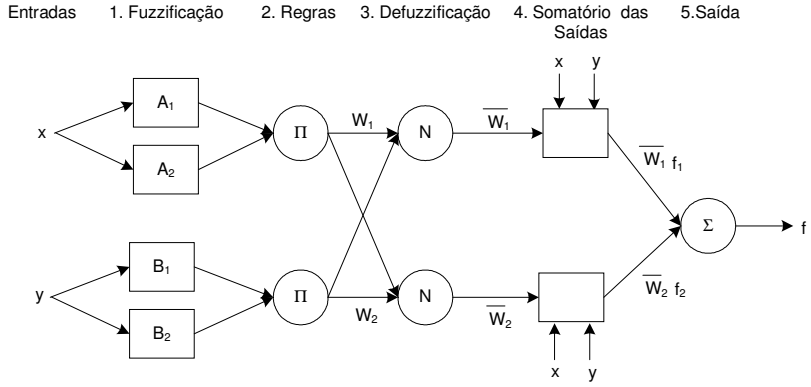

Figura 4.2 - Arquitetura de um modelo ANFIS

A estrutura apresentada na Figura 4.2 não é única. As camadas 3 e 4 podem ser combinadas, resultando em uma rede equivalente com quatro camadas. Outras mudanças possíveis seriam: realização da normalização dos pesos na última camada, aplicação de um operador *t-norma* na camada 2 e funções de pertinência diferentes, utilização de regras conectadas pelo operador lógico *OU* e múltiplas saídas. ANFIS correspondentes aos modelos de Tsukamoto e Mamdani também podem ser obtidos [Jang et al. 1997].

As principais desvantagens do ANFIS são: não possui mecanismo de aprendizagem de regras, a camada de saída é composta apenas por um nó e, apesar de permitir a aplicação de técnicas de otimização, geralmente o número de regras extraídas é muito grande.

NEFCLASS

NEFCLASS (*Neuro Fuzzy Classification*) [Nauck & Kruse 1995] é um sistema neuro-*fuzzy* para classificação de padrões que deriva regras *fuzzy* a partir dos dados e usa um algoritmo de aprendizagem supervisionada baseado na retropropagação *fuzzy* do erro.

Este sistema utiliza um modelo genérico de *perceptron fuzzy* [Nauck 1994], apresenta arquitetura similar à rede MLP de três camadas e permite a incorporação de conhecimento sobre o domínio do problema. Apenas os pesos, as entradas dos nós e as ativações dos nós de saída são modelados como conjuntos *fuzzy* [Nauck 1997].

A Figura 4.3 [Nauck & Kruse 1995] mostra um modelo NEFCLASS com duas entradas, cinco regras e duas classes. As elipses em torno de algumas conexões representam o compartilhamento de peso. Este compartilhamento garante que para cada predicado (Ex.: x_1 *é grande*), existe um conjunto fuzzy associado (Ex.: $\mu_1^{(1)}$), ou seja, o predicado tem a mesma interpretação para todos os nós regra (Ex.: R_1 e R_2). Os pesos entre a camada de regra e a camada de saída são fixados em 1 por razões semânticas (evitar regras ponderadas) [Nauck & Kruse 1999].

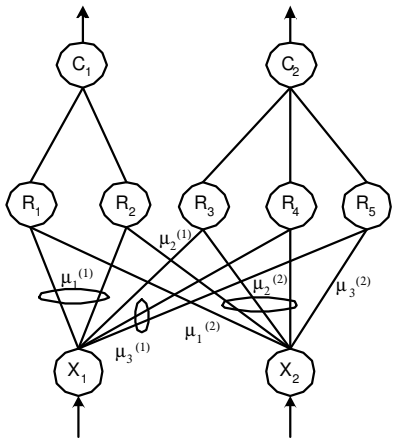

Figura 4.3 - Arquitetura de um modelo NEFCLASS

As regras *fuzzy* extraídas de um modelo NEFCLASS apresentam a seguinte estrutura:

Se x_1 *é* μ_1 *E* x_2 *é* μ_2 *E ... E* x_n *é* μ_n
Então o exemplo $(x_1, x_2, ..., x_n)$ *pertence a classe i*

onde $\mu_1, \mu_2, ..., \mu_n$ são conjuntos *fuzzy*. Como pode ser observado, os conseqüentes das regras não são associados a conjuntos *fuzzy* [Nauck & Kruse 1995].

Num sistema NEFCLASS, o usuário deve definir o total de funções de pertinência para cada entrada e o número máximo de nós regra. No início do treinamento, a camada intermediária não possui nós. Durante a primeira apresentação do conjunto de treinamento, os nós regra são criados. Uma regra é criada encontrando para um dado exemplo a combinação de conjuntos *fuzzy* que resulta no maior

grau de pertinência para cada atributo. Se a combinação encontrada não for idêntica aos antecedentes de uma regra existente e o número máximo de regras não for alcançado, um novo nó é criado. O conseqüente para cada regra é determinado somando, separadamente para cada classe, os graus de satisfação de todos os exemplos em relação ao antecedente. O conseqüente é estabelecido para o rótulo da classe que obtém a maior soma. Quando a base de regras é criada, o algoritmo de aprendizagem adapta as funções de pertinência dos antecedentes.

Outra forma de gerar as regras é não restringir o número máximo de regras e, ao final do processo, se o total de regras criadas for maior do que o limite estabelecido, apenas as melhores regras são consideradas. Além de selecionar as melhores regras, as técnicas de simplificação de regras diminuem a quantidade de condições por regra e removem atributos ou conjuntos *fuzzy*. Em seguida, os parâmetros das funções de pertinência são novamente ajustados [Nauck & Kruse 1999]. Esta forma pode demandar bastante processamento, não garantir a obtenção de uma interpretação satisfatória dos dados e gerar uma base de regras bastante extensa, principalmente em problemas com um grande número de atributos de entrada.

Além do NEFCLASS, também foram propostos os modelos NEFCON (*Neuro Fuzzy Control*) [Nauck & kruse 1993] para aplicações de controle e o NEFPROX (*Neuro Fuzzy Function Approximation*) [Nauck & Kruse 1999] para aproximação de funções contínuas baseada no Sistema *Fuzzy* de Mamdani.

RBF-*Fuzzy*

O modelo RBF-*Fuzzy* (*Radial Basis Function*) [Sun & Jang 1992] consiste de uma rede adaptativa que ajusta seus parâmetros através de um algoritmo de aprendizagem híbrido [Jang 1993].

A rede RBF-*Fuzzy* é composta de três camadas. Os nós da primeira camada são responsáveis pela fuzzificação dos dados de entrada. Normalmente, as funções de pertinência têm a forma de uma distribuição gaussiana. Os nós da segunda camada, denominada camada RBF, são fixos e computam o produto escalar (*E* da Lógica *Fuzzy*) dos valores recebidos. A saída dos nós da segunda camada corresponde ao grau de verdade do antecedente de uma regra. A terceira camada possui um número de nós igual ao número de classes. A saída de cada um destes nós é um número entre 0 e 1, representando o grau para o qual o vetor de entrada pertence à classe associada ao nó.

A configuração da rede RBF-*Fuzzy* (número de regras e funções de pertinência por entrada) pode ser automaticamente definida através do algoritmo DDA (*Dynamic Decay Adjustment*) [Berthold & Diamond 1995]. As vantagens do DDA são: é construtivo (novas funções gaussianas são adicionadas, quando necessário, durante o treinamento), rápido, apresenta convergência garantida e possui dois parâmetros não críticos (a convergência não depende da escolha desses parâmetros).

A Figura 4.4 [Conde 2000] mostra uma RBF-*Fuzzy* com duas entradas, associadas a três e dois conjuntos *fuzzy*, seis nós na camada intermediária e três nós na camada de saída.

A base de regras extraída de uma rede RBF-*Fuzzy* tem o tamanho máximo igual ao número de nós na camada RBF multiplicado pelo número de nós de saída. O conseqüente das regras apresenta apenas uma saída e as regras extraídas não possuem graus de confiança.

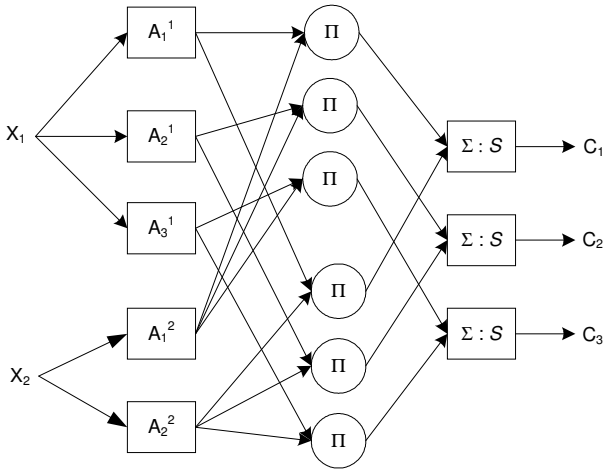

Figura 4.4 - Arquitetura de uma rede RBF-*Fuzzy*

As principais limitações da rede RBF-*Fuzzy* estão relacionadas ao tamanho da base de regras extraída. Dependendo da qualidade e quantidade dos dados, a rede pode crescer muito. Como resultado, uma base de regras extensa pode ser extraída. Além disso, não existe nenhum mecanismo para reduzir o número de premissas ou regras. Por esta razão, este modelo não é muito adequado para problemas em que a dimensão do vetor de entrada é grande.

Fuzzy ARTMAP

A rede *Fuzzy* ARTMAP (*Fuzzy Adaptive Resonance Theory-supervised Predictive Mapping*) [Carpenter et al. 1992] é um modelo construtivo, ou seja, é capaz de aprender novos padrões e manter o reconhecimento das informações aprendidas anteriormente sem a necessidade de ser re-treinado com os padrões antigos.

As redes *Fuzzy* ARTMAP são baseadas nas redes ARTMAP. Desta forma, características, arquitetura, parâmetros e algoritmo de aprendizagem das redes ARTMAP também são válidos para as redes *Fuzzy* ARTMAP. Estes modelos diferem no modo como a classificação é realizada. A rede ARTMAP classifica exemplos através de um vetor de valores binários que indicam a presença ou ausência de um conjunto de características. A rede *Fuzzy* ARTMAP aprende a classificar os padrões de entrada através de um conjunto *fuzzy* de características representado por graus de pertinência que variam entre 0 e 1, indicando o grau com que cada característica está presente no padrão de entrada.

A base de regras extraída de uma rede *Fuzzy* ARTMAP depende diretamente da arquitetura do modelo, que pode crescer muito, dependendo da qualidade e quantidade dos dados. Uma característica particular deste tipo de rede é que as regras extraídas não são *fuzzy*.

FuNN

FuNN (Fuzzy Neural Network) [Kasabov et al. 1997] é uma rede MLP que utiliza um algoritmo de treinamento baseado no *Backpropagation*, através do qual as funções de pertinência e regras *fuzzy* são ajustadas de acordo com os dados de treinamento.

Além da integração da Lógica *Fuzzy* com RNA, FuNN também permite a utilização de AG. Esta rede tem produzido bons resultados em uma ampla variedade de problemas. Suas principais áreas de aplicação são aproximação de funções e interpretação e adaptação de regras *fuzzy*.

A arquitetura da rede FuNN facilita a aprendizagem a partir de um conjunto de dados, a adaptação, o raciocínio aproximado e a inserção e extração de regras *fuzzy* [Kasabov 1996]. Desta forma, duas fontes de dados podem ser utilizadas: base de dados e regras iniciais.

A rede FuNN é composta de cinco camadas com conexões *feedforward* parciais. A Figura 4.5 [Kasabov et al. 1997] mostra a estrutura de uma rede FuNN com duas regras iniciais:

$$R_1: Se\ X_1\ é\ A_1\ (DI_{1,1})\ E\ X_2\ é\ B_1\ (DI_{2,1})\ Então\ y\ é\ C_1\ (CF_1)$$

$$R_2: Se\ X_1\ é\ A_2\ (DI_{1,2})\ E\ X_2\ é\ B_2\ (DI_{2,2})\ Então\ y\ é\ C_2\ (CF_2)$$

onde *DI* são os graus de importância das condições e *CF* são os fatores de certeza das conclusões. *DI* representa a importância de cada condição para ativar a regra, enquanto que *CF* representa o fator de confiança da regra quando inferindo a saída *fuzzy*.

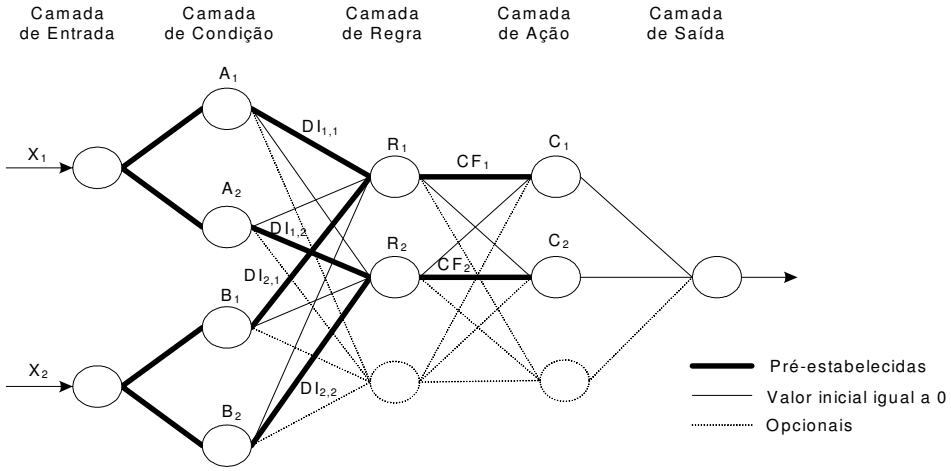

Figura 4.5 - Arquitetura de uma rede FuNN

POPFNN

POPFNN (*Pseudo Outer-Product based Fuzzy Neural Network*) [Quek & Zhou 2001] é um modelo neuro-*fuzzy* que realiza as etapas de fuzzificação, inferência *fuzzy* e defuzzificação. Este modelo foi proposto com dois algoritmos de aprendizagem para identificação de regras *fuzzy* relevantes, o *Pseudo Outer-Product* (POP) e o *Lazy Pseudo Outer-Product* (LazyPOP). POP é um algoritmo simples, no entanto, considera todas as regras possíveis no início do processo de aprendizagem. LazyPOP foi proposto para superar esta limitação. Este algoritmo identifica apenas as regras que são relevantes para o conjunto de treinamento e não usa um método de seleção através do qual as regras irrelevantes são eliminadas de um conjunto de regras iniciais. Além de apresentar performance superior ao POP, LazyPOP é capaz de ajustar a estrutura da rede através da remoção de atributos inválidos, baseando-se nas regras *fuzzy* identificadas. LazyPOP e POP são executados em um passo.

FWD

FWD (*Feature-Weighted Detector*) [Li et al. 2002] é um modelo neuro-*fuzzy* que apresenta uma estrutura bastante diferente dos modelos tradicionais e cujo principal objetivo é resolver os dois principais problemas em reconhecimento de padrões: seleção de atributos relevantes e classificação de padrões. Algumas técnicas foram propostas para resolver simultaneamente estes problemas [Bezdek 1981] [Kuncheva 1992]. No entanto, estas técnicas apresentam várias limitações, as quais motivaram a proposição do modelo FWD, que também permite a extração de regras *fuzzy* Se-Então diretamente dos pesos da rede.

A rede FWD, como mostra a Figura 4.6 [Li et al. 2002], consiste de quatro camadas: camada de entrada (E), camada de casamento (C), camada de detecção (D) e camada de saída (S).

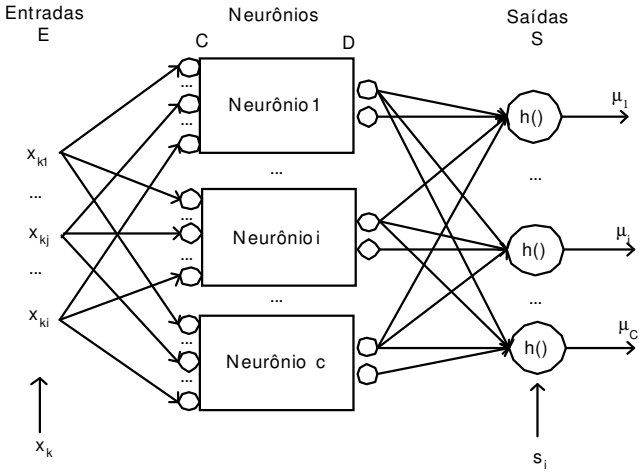

Figura 4.6 - Arquitetura da rede FWD

FNN

FNN (*Fuzzy Neural Network*) [Kuo et al. 2002] é uma rede neural *fuzzy* composta por pesos *fuzzy*, capaz de aprender regras *fuzzy* Se-Então e eliminar pesos irrelevantes durante o treinamento, contribuindo para uma melhor convergência. O modelo FNN é composto de três camadas: entrada, intermediária e saída.

O algoritmo de aprendizagem deste modelo é similar ao *Backpropagation*. O algoritmo supõe que as entradas, os pesos e os limiares são fuzzificados; as entradas são números *fuzzy* positivos; e os números *fuzzy* são representados por funções gaussianas assimétricas.

4.4.2. Modelos Escolhidos

FuNN

Entre os modelos neuro-*fuzzy* mais difundidos, o modelo FuNN foi escolhido para ser investigado nesta dissertação devido às seguintes razões:

- ♣ Tem apresentado resultados satisfatórios em diversos domínios de aplicação;

- Permite o uso de limiares no processo de extração de regras, possibilitando a redução do número de regras e atributos nos antecedentes das regras;
- Permite a inserção e o refinamento de regras usadas para definir a arquitetura inicial da rede, possibilitando, desta forma, o uso de duas fontes de dados (base de dados e regras iniciais).

Como foi apresentado na Figura 4.5, o modelo FuNN é composto de cinco camadas que desempenham funções específicas.

Os nós da camada de entrada representam os atributos de entrada. Os valores de entrada são propagados para a camada de condição, que pode ser expandida durante a fase de adaptação e cujos nós representam os conjuntos *fuzzy* dos atributos de entrada. Os atributos de entrada podem ter números diferentes de conjuntos *fuzzy*.

As funções de pertinência associadas aos nós condição podem mudar ou permanecer fixas durante o treinamento. Os valores de ativação destes nós representam os graus de pertinência dos valores de entrada [Kasabov 1996]. A fuzzificação é realizada usando funções de pertinência triangulares cujos centros são representados pelos pesos das conexões com os nós de entrada. Estes pesos apresentam valores entre 0 e 1, pois assume-se que os dados estão normalizados neste intervalo. Os pontos máximo e mínimo dos triângulos são estabelecidos para os centros dos triângulos adjacentes, ou são *shouldered* no caso da primeira e última funções de pertinência. Esta abordagem baseada no centro é simples, eficiente e evita o problema de regiões não cobertas no espaço de entrada.

As funções de pertinência podem ser não-simétricas e qualquer valor de entrada pertence a duas funções com graus diferentes de zero, com exceção do caso em que o valor de entrada se posiciona exatamente no centro de uma função. Neste caso, apenas essa função será ativada e com valor 1. A soma dos graus de pertinência para um determinado valor de entrada sempre é igual a 1.

Inicialmente as funções de pertinência são igualmente distribuídas sobre o espaço dos pesos. A fim de manter o significado semântico das funções, algumas restrições devem ser satisfeitas durante a adaptação, garantindo que os centros permaneçam dentro de partições válidas. Isto evita problemas de violação de ordenamento semântico dos termos lingüísticos. Desta forma, os termos associados inicialmente aos pesos permanecerão válidos após o treinamento. A Figura 4.7 [Kasabov et al. 1997] mostra um exemplo onde as linhas sólidas representam as funções de pertinência iniciais e as linhas tracejadas representam as funções de pertinência após a adaptação.

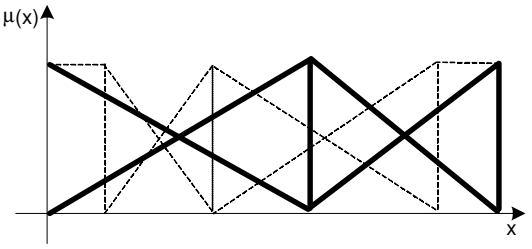

Figura 4.7 - Adaptação das funções de pertinência de uma rede FuNN

Na camada de regra, cada nó representa uma regra *fuzzy*. Esta camada pode ser expandida, permitindo que mais regras sejam adicionadas à medida que a rede se adapta. A função de ativação dos nós regra é a sigmóide logística, definida pela Equação 3.8.

A ativação de um nó regra representa o grau para o qual os dados de entrada se associam ao antecedente da regra associada ao nó. Os pesos entre a camada de condição e a camada de regra representam semanticamente os graus de importância dos nós condição para a ativação do nó regra.

Os valores dos pesos que chegam e que saem da camada de regra podem ser limitados durante a fase de treinamento, introduzindo, desta forma, não linearidade aos pesos. Esta opção elimina a necessidade de normalização dos pesos e deve ser implementada com um coeficiente de ganho apropriado. A limitação dos pesos garante que as entradas nas regras permaneçam no intervalo [-1,+1] e o fator de ganho permite regras apenas para valores de saída dentro do intervalo [0,1; 0,9]. Isto melhora a semântica das regras e evita saturação.

Quando a rede FuNN é usada para implementar um conjunto de regras iniciais, as conexões entre a camada de condição e a camada de regra são estabelecidas de acordo com os graus de importância normalizados do antecedente das regras. Se os graus de importância não estiverem associados aos elementos do antecedente de uma regra R_i, os pesos w_{ij} para o nó regra R_i são uniformemente calculados para cada conexão como [Kasabov 1996]:

$$w_{ij} = \frac{Net_i}{n} \tag{4.1}$$

onde n é o número de elementos no antecedente da regra R_i, Net_i é uma constante que define qual deveria ser a entrada do nó R_i, a fim de disparar a regra. Os outros pesos são inicializados com 0. Nós adicionais podem ser inseridos com pesos iniciais iguais a 0. Isto fornece mais flexibilidade à estrutura para ajustar as regras iniciais e, possivelmente, capturar novas regras.

Na camada de ação, cada nó representa um termo *fuzzy* de um atributo de saída (elemento do conseqüente de uma regra). A ativação do nó corresponde ao grau para o qual sua função é suportada pelos dados apresentados à rede. Os pesos entre a camada de regra e a camada de ação representam conceitualmente os fatores de confiança das regras quando inferindo os valores da saída *fuzzy*. Estes pesos estão sujeitos às mesmas restrições dos pesos da camada de regra, obtendo as mesmas vantagens semânticas. A função de ativação dos nós ação é a sigmóide logística com o mesmo coeficiente de ganho da camada anterior.

Quando um conjunto de regras iniciais está disponível, os pesos entre os nós regra e os nós ação são inicializados com os fatores de confiança normalizados das regras. Os demais pesos são inicializados com 0. Nós adicionais podem ser usados para capturar novos termos durante o treinamento [Kasabov 1996].

A camada de saída, cujos nós representam os atributos de saída, realiza uma defuzzificação baseada no método COG. *Singletons*, representando os centros das funções de pertinência triangulares, são associados aos pesos entre a camada de ação e a camada de saída. Inicialmente, *singletons* uniformemente distribuídos podem ser usados. Funções de ativação lineares são utilizadas nesta camada.

A adaptação na camada de saída corresponde a mover os centros. A soma dos graus de pertinência para um valor de saída deve ser igual a 1. Da mesma forma que ocorre nas funções de pertinência das entradas, para cada centro existe uma partição em que ele pode se mover. Mais de um atributo de saída pode ser usado e atributos de saída diferentes podem ter números diferentes de funções de pertinência.

A inicialização uniformemente distribuída dos centros das funções de pertinência dos atributos de entrada e saída é usada na ausência de informações sobre o domínio do problema. Se um conjunto de regras está disponível, ele pode ser usado para inicializar a estrutura da rede. Assim, a rede pode aprender de forma mais rápida e precisa.

Formas de Adaptação

Existem três formas de adaptação na rede FuNN [Kasabov et al. 1997]:

- Parcialmente adaptativa 1: As funções de pertinência dos atributos de entrada e saída não mudam durante o treinamento e uma versão modificada do algoritmo *Backpropagation* é usada para adaptar as regras. Esta forma é adequada para problemas onde as funções de pertinência são conhecidas *a priori*;
- Parcialmente adaptativa 2: As funções de pertinência são adaptadas através da aplicação de AG. Neste caso, as regras não são alteradas;
- Totalmente adaptativa: As regras e funções de pertinência são adaptadas através de uma versão modificada do algoritmo *Backpropagation* que impõe as restrições necessárias para manter o significado semântico. Este algoritmo será explicado na próxima seção, pois essa forma foi escolhida para ser investigada nesta dissertação.

Estas formas podem ser combinadas de acordo com o problema.

Algoritmo de Treinamento

Uma versão estendida do algoritmo *Backpropagation* é utilizada para realizar o treinamento totalmente adaptativo da rede FuNN [Kasabov et al. 1997]. Desta forma, o algoritmo é executado em duas fases: *forward* e *backward*. Para facilitar a notação, um índice sobrescrito é utilizado para indicar a camada e um índice subscrito para indicar os pesos entre as camadas.

Na fase *forward* são calculados os valores da ativação de todos os nós da rede, da primeira à quinta camada.

- Camada de entrada: Os nós transmitem os valores de entrada para a camada de condição sem realizar qualquer processamento;
- Camada de condição: A saída de cada nó corresponde ao grau de pertinência da entrada com relação ao conjunto *fuzzy* que o nó representa. As funções de ativação para um nó *i* são:

$$Se\ a_i < x < a_{i+1},\ então\ At_i^c = 1 - (x - a_i) / (a_{i+1} - a_i),\ senão$$
$$Se\ a_{i-1} < x < a_i,\ então\ At_i^c = 1 - (a_i - x) / (a_i - a_{i-1}),\ senão$$
$$Se\ x = a_i,\ então\ At_i^c = 1$$

onde *a* é o centro da função de pertinência.

♣ Camada de regra: Os pesos das conexões entre a camada de condição e a camada de regra podem ser inicializados com valores aleatórios e alterados no treinamento, ou inicializados de acordo com um conjunto inicial de regras. Os *net inputs* e as ativações dos nós desta camada são estabelecidos como:

$$Net^r = \sum w_{rc} \,.\, At^c \tag{4.2}$$

$$At^r = \frac{1}{1+e^{-g \cdot Net^r}} \tag{4.3}$$

♣ Camada de ação: O processamento é similar ao da camada de regra:

$$Net^a = \sum w_{ar} \,.\, At^r \tag{4.4}$$

$$At^a = \frac{1}{1+e^{-g \cdot Net^a}} \tag{4.5}$$

♣ Camada de saída: É utilizada a seguinte função de ativação:

$$Net^o = \sum w_{oa} \,.\, At^a \tag{4.6}$$

$$At^o = \frac{Net^o}{\sum At^a} \tag{4.7}$$

Na fase *backward*, o objetivo é minimizar a função de erro:

$$Erro = \frac{\sum (y^d - y^a)^2}{2} \tag{4.8}$$

onde y^d é a saída desejada e y^a é a saída da rede. Para isso, é utilizada a seguinte regra de atualização dos pesos:

$$| \quad w_{t+1} = \eta.\delta.At + \alpha.| \quad w_t \tag{4.9}$$

onde η é a taxa de aprendizagem e α é o termo *momentum*.

♣ Camada de saída: Devem-se considerar as restrições impostas aos centros das funções de pertinência. Quando um peso se desloca para fora de sua partição ele é retornado para o limite apropriado.

$$\delta^o = y^d - y^a \tag{4.10}$$

Se $w_t + | \quad w_{t+1} < Partição_k$, *então* $w_{t+1} = Partição_k$, *senão*

Se $w_t + | \quad w_{t+1} > Partição_{k+1}$, *então* $w_{t+1} = Partição_{k+1}$, *senão*

$w_{t+1} = w_t + | \quad w_{t+1}$

♣ Camada de ação: O erro de cada nó é calculado individualmente baseando-se no erro da saída e na sua ativação.

$$\delta^a = At^a.(1 - At^a).(d^a - At^a) \tag{4.11}$$

Se $a_i < y < a_{i+1}$, *então* $d^a = 1 - (y - a_i) / (a_{i+1} - a_i)$, *senão*

Se $a_{i-1} < y < a_i$, *então* $d^a = 1 - (a_i - y) / (a_i - a_{i-1})$, *senão*

Se $y = a_i$, *então* $d^a = 1$

♣ Camada de regra:

$$\delta^r = At^r.(1 - At^r).\Sigma(w_{ar}.\delta^a) \tag{4.12}$$

♣ Camada de condição:

$$\delta^c = At^c.\Sigma(w_{rc}.\delta^r) \tag{4.13}$$

A regra de atualização dos pesos desta camada é:

$$w_{ic_{t+1}} = \eta\delta^c + \alpha\Delta w_{ic_t} \tag{4.14}$$

Da mesma forma que na camada de saída, devem-se considerar as restrições referentes aos centros das funções de pertinência.

FWD

Entre os modelos neuro-*fuzzy* propostos recentemente, FWD foi escolhido para ser investigado nesta dissertação. A rede FWD apresenta uma característica inovadora de englobar, dentro de uma mesma estrutura, três etapas do processo de KDD: seleção de dados, mineração de dados e apresentação do conhecimento. Esta é a principal razão da escolha deste modelo. A investigação da rede FWD também é motivada pelo fato de que existem poucos resultados experimentais demonstrando sua viabilidade em aplicações reais [Li et al. 2002] [Amorim et al. 2003].

Para facilitar a compreensão do modelo, será considerado o funcionamento de um neurônio isoladamente, como mostra a Figura 4.8 [Li et al. 2002].

A camada *E* é responsável por fornecer os exemplos de entrada à rede. Cada neurônio recebe todos os valores de entrada.

A camada *C* efetua o casamento dos exemplos de entrada com as conexões de memória m_{ji}. Por esta razão, esta camada recebe entradas através das conexões com as camadas *E* e *D*. A função de ativação comparativa, *f(.)*, da camada *C* é descrita pela Equação 4.15.

$$y_{ij} = (x_{kj} - m_{ji}), \qquad j = 1,2,...,p;\ i = 1,2,...,c;\ k = 1,2,\cdots,N \tag{4.15}$$

Na Equação 4.15, *p* representa o número de atributos, *c* o número de classes e *N* o número de exemplos.

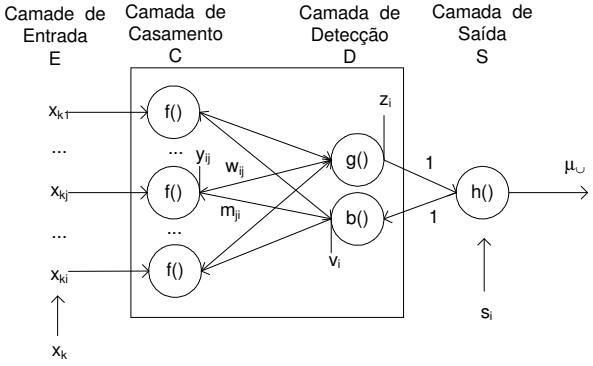

Figura 4.8 - Estrutura de um neurônio da rede FWD

De acordo com o fluxo de processamento, a camada *D* possui dois tipos de nós: propagação e retropropagação. Os nós de propagação recebem *p* entradas através das conexões w_{ij} com a camada *C* e fornecem dados para a camada *S* a partir de conexões de pesos fixos com valor 1. A função de ativação da camada *D* é a função gaussiana *g(.)*, descrita pela Equação 4.16.

$$z_j = exp(-\frac{1}{2\sigma^2} \cdot \sum_{j=1}^{p} w_{ij}^2 \cdot y_{ij}^2) \tag{4.16}$$

Os nós de retropropagação recebem entradas das conexões com a camada *S* com pesos fixos em 1. A função de ativação *b(.)* destes nós é uma função identidade.

A camada *S* é responsável por fornecer a classificação da rede. Esta camada funciona como uma função de pertinência *fuzzy*, efetuando a normalização de todos os valores provenientes da camada *D*. A camada *S* recebe *c+1* entradas, *c* através das conexões com a camada *D* e uma chamada de sinal. A função de ativação da camada *S*, *h(.)*, é descrita pela Equação 4.17.

$$u_i = z_i / \sum_{j=1}^{c} z_j \tag{4.17}$$

Aprendizagem

A rede FWD utiliza aprendizagem supervisionada e não-supervisionada. O ajuste das conexões de memória m_{ji}, responsáveis pela classificação *fuzzy*, é realizado de modo não-supervisionado. Tal atualização é descrita pela Equação 4.18.

$$\Delta m_i = \alpha \cdot (1 - t/T) \cdot u_i(x_k) \cdot (x_k - m_i) \tag{4.18}$$

Nesta equação, x_k representa a *k*-ésima entrada e $\alpha \in [0,1]$ se refere à taxa de aprendizagem temporal cujo valor é definido pelo usuário. *T* corresponde ao número de épocas de treinamento e *t* indica o passo atual de treinamento.

O ajuste dos pesos w_{ij}, responsáveis pela seleção dos atributos mais relevantes para a classificação, é realizado de modo supervisionado de acordo com a Equação 4.19.

$$\Delta w_{ij} = \frac{\beta}{\sigma^2 \cdot s^2} \cdot (u_i(x_k) - d_i) \cdot (s - z_i) \cdot w_{ij} \cdot z_i \cdot y_{ij}^2 \tag{4.19}$$

Nesta equação, $s = \sum_{i=1}^{c} z_i$, $\beta > 0$ é a taxa de aprendizagem, $\sigma > 0$ representa a nebulosidade no agrupamento e d_i é o valor da saída desejada para o neurônio *i*. Para facilitar a compreensão, $0 \le w_{ij} \le 1$.

A medida do erro de treinamento da fase supervisionada é dada pela soma do erro quadrático (Equação 3.1).

Durante a fase de treinamento, a rede tenta identificar os atributos relevantes, enquanto mantém a taxa de classificação máxima.

Inicialização dos Pesos

Na rede FWD, as conexões de memória são inicializadas com valores aleatórios (Tabela 4.1), considerando o conjunto de treinamento, e as conexões peso são inicializadas com 1.

Tabela 4.1 - Algoritmo de inicialização das conexões de memória do modelo FWD

```
Entrada: vetor de números reais dadosTreinamento[ ][ ]
real somaPertinencia;
real soma;
real pertinencia;
para ( inteiro s = 0; s < número de nós de saída; s = s + 1 )
  para ( inteiro e = 0; e < número de nós de entrada; e = e + 1 ) {
    somaPertinencia = 0;
    soma = 0;
    para ( inteiro p = 0; p < tamanho do vetor dadosTreinamento; p = p + 1 ) {
      pertinencia = obterNumeroAleatoirioEntreZeroUm( );
      somaPertinencia += pertinencia;
      soma += pertinencia * dadosTreinamento[p][e];
    }
    memoria[s][e] = soma/somaPertinencia;
    peso[s][e] = 1;
}
```

Seleção de Atributos

O processo de seleção de atributos da rede FWD é realizado a partir das conexões peso w_{ij}, cujos valores representam o grau de contribuição do atributo *j* para a classe *i*. Por exemplo, $w_{ij} = 0$ indica que o atributo *j* não tem contribuição alguma na formação da classe *i* e poderia ser retirado da base sem prejuízo para a classificação.

4.5. Considerações Finais

Os Sistemas Neuro-*Fuzzy* apresentam diversas vantagens [Bigus 1996] [Kasabov et al. 1997]: aprendizagem rápida e precisa (conhecimento sobre o domínio do problema é utilizado para definir a estrutura inicial da rede), boa capacidade de generalização, facilidade de explicação na forma de regras *fuzzy*, habilidade de se adaptar a um conjunto de dados e incorporar o conhecimento de especialistas, e, mais provavelmente, convergem para um mínimo global sobre condições iniciais arbitrárias. Devido a essas vantagens, tais sistemas têm sido utilizados com sucesso em diversos domínios de aplicação.

Apesar de superar as deficiências das RNA e Sistemas *Fuzzy*, e apresentar várias potencialidades, os Sistemas Neuro-*Fuzzy* podem apresentar algumas limitações quando aplicados em problemas complexos de grande dimensionalidade. Exemplos de limitações e algumas soluções que podem ser aplicadas para minimizá-las são [Jang 1993] [Medeiros 1996] [Jang et al. 1997] [Malhotra & Malhotra 2002] [Evsukoff & Almeida 2003]:

- A quantidade de regras geradas e o tamanho dos antecedentes podem ser muito grandes. A aplicação de métodos de seleção de atributos a fim de reduzir a dimensionalidade das entradas e a utilização de um método sofisticado de particionamento do espaço de entrada são exemplos de soluções que podem ser adotadas para minimizar este problema;

♣ A modelagem de um sistema baseado na Lógica *Fuzzy* deveria ser realizada com o auxílio de um especialista. No entanto, nem sempre é possível a presença do especialista. Conseqüentemente, é bastante usual modelar o sistema empiricamente, através de técnicas de visualização que auxiliam na análise dos dados ou por tentativa e erro.

Para aplicações reais, onde a dimensão da entrada é grande e/ou os dados de treinamento das diversas classes estão bastante entrelaçados, a dificuldade persiste mesmo quando técnicas de visualização são utilizadas. Nestes casos, a dimensão, quantidade e distribuição no espaço de entrada restringem e inviabilizam o emprego de técnicas de visualização. Como resultado, na maioria das vezes, é adotada a estratégia "tentativa e erro".

Nas sucessivas tentativas, o número de funções de pertinência uniformemente distribuídas é aumentado progressivamente. Em adição, premissas conjuntivas contendo todos os atributos de entrada são consideradas, de forma que o espaço de entrada seja totalmente coberto. Além de depender de várias tentativas, esta estratégia pode gerar uma explosão combinatória do número de regras quando a dimensionalidade do vetor de entrada for grande. Isso ocorre porque o número de regras cresce exponencialmente com a dimensão da entrada. Desta forma, métodos de identificação e seleção de regras iniciais são fortemente requisitados.

Embora seja possível superestimar o número de funções de pertinência, esta alternativa não é recomendada, pois torna o processo de aprendizagem mais lento e degrada o poder de generalização.

Pesquisas recentes em Sistemas Neuro-*Fuzzy* apontam para a determinação automática do número de regras e funções de pertinência por entrada. O algoritmo EDDA (*Extended Dynamic Decay Adjustment*) e as técnicas CART (*Classification and Regression Trees*) e *clustering* são exemplos de métodos que podem ser usados para este fim. Outra solução que vem sendo bastante investigada para este problema é a aplicação de AG para otimização do número de termos *fuzzy*.

Os problemas especificados anteriormente são minimizados pelos modelos FuNN e FWD. O modelo FuNN permite a utilização de limiares na etapa de extração de regras, possibilitando, desta forma, a redução do número de regras e de condições nos antecedentes. Além disso, na presença de um especialista ou de algum conhecimento *a priori* sobre o domínio do problema, este modelo permite a inserção de regras para definir a arquitetura inicial da rede. O modelo FWD, por sua vez, fornece um mecanismo de seleção de atributos que possibilita a remoção dos atributos irrelevantes sem prejudicar a performance de classificação, resultando numa representação mais compacta do conceito alvo. Este modelo não apresenta o problema referente ao tamanho da base de regras, pois ele gera apenas uma regra por classe.

Capítulo 5

Técnicas de Extração de Conhecimento de Redes Neurais Artificiais

5.1. Introdução

Embora as RNA tenham se mostrado uma técnica eficiente para a solução de um grande número de problemas, não é correto afirmar que elas são suficientes para resolver qualquer tarefa. As RNA apresentam várias limitações que inviabilizam seu uso exclusivo na solução de uma quantidade significativa de problemas. Uma das maiores limitações das RNA é a dificuldade de gerar uma representação compreensível da solução de um problema, pois o conhecimento aprendido é representado de forma implícita pela topologia, valores dos pesos e limiares; e o espaço de representação é dividido em regiões complexas por meio da combinação de várias funções matemáticas [Azevedo et al. 2000]. Devido a esta limitação, pesquisadores, projetistas e usuários acreditam que todo o potencial das RNA não poderá ser completamente explorado enquanto não for acrescentado a estes modelos um mecanismo que explique suas decisões. Como resultado, muitos pesquisadores têm concentrado seus esforços no desenvolvimento de técnicas que tornem as RNA mais compreensíveis. Essas técnicas são denominadas técnicas de extração de conhecimento de RNA [Milaré & Carvalho 2001].

Além da integração de Sistemas *Fuzzy* e RNA, que permite a extração de regras de uma rede neural, existem outras formas de extrair conhecimento simbólico de RNA. Diversas técnicas foram propostas para este fim. Este capítulo aborda algumas dessas técnicas e descreve detalhadamente TREPAN (*Trees Parroting Networks*) [Craven 1996] e as técnicas específicas dos modelos FuNN [Kasabov 1996] e FWD [Li et al. 2002], por terem sido escolhidas para investigação nesta dissertação.

5.2. Extração de Conhecimento de RNA

Quando técnicas de extração de conhecimento de RNA são consideradas, o conhecimento não precisa necessariamente estar na forma de regras. Embora isto aconteça na maioria dos casos, outras formas de representação podem ser extraídas, como valores estatísticos, protótipos de *clusters* e autômatos de estados finitos [Craven & Shavlik 1998] [Braga et al. 2000].

As técnicas de extração de conhecimento de RNA diferem entre si pelas seguintes características [Craven & Shavlik 1998]:

- ♣ Requisitos arquiteturais e de treinamento impostos aos modelos neurais;
- ♣ Linguagem de representação usada para descrever o conhecimento incorporado pela rede. Regras Se-Então, regras *m-de-n*, regras *fuzzy*, árvores de decisão e autômatos de estados finitos são alguns exemplos de linguagem;
- ♣ Estratégia de mapeamento do modelo representado pela rede em um modelo na nova linguagem de representação. A estratégia define como a técnica explora o espaço de descrições candidatas e o nível de descrição usado para caracterizar a rede.

Ao final do processo de extração de conhecimento, as técnicas podem ser avaliadas de acordo com os seguintes aspectos [Craven & Shavlik 1999]:

- ♣ Compreensibilidade da representação extraída;
- ♣ Precisão (habilidade da representação extraída em realizar predições corretas);
- ♣ Fidelidade (nível de precisão que a representação extraída modela o conhecimento incorporado pela rede);
- ♣ Escalabilidade (habilidade de a técnica ser escalável em redes com espaço de entrada extenso e grande quantidade de nós).

5.3. Extração de Regras Se-Então

Regras Se-Então é o formalismo mais difundido para representação de conhecimento devido às seguintes vantagens [Turban & Aronson 2000]:

- ♣ Forma natural e compreensível de expressar o conhecimento;
- ♣ Cada regra define uma parte do conhecimento, ou seja, as regras são modulares. No entanto, quando as regras são combinadas e aplicadas em um mecanismo de inferência, o resultado alcançado é melhor do que a soma dos resultados das regras individuais;
- ♣ Como conseqüência da modularidade, novas regras podem ser adicionadas e regras antigas podem ser alteradas de forma relativamente independente das demais. Deste modo, a manutenção das regras é facilitada. Esta facilidade não é observada quando o conhecimento a ser modelado é complexo. Neste caso, muitas regras são geradas, dificultando o uso e a manutenção das mesmas;
- ♣ São estruturas explícitas que podem ser usadas, internamente, para raciocínio e aprendizagem e, externamente, para explicação dos resultados.

Devido a estas vantagens, a extração de regras Se-Então é usualmente aceita como a melhor maneira de extrair o conhecimento incorporado pelas RNA [Brasil 1999]. Conseqüentemente, a maioria dos esforços têm sido direcionados no sentido de apresentar as explicações da rede na forma de um conjunto de regras expressas na lógica simbólica convencional. Um esforço substancial também tem sido feito para codificar o conhecimento da RNA usando conceitos da Lógica *Fuzzy*. Como resultado, o conhecimento extraído é expresso numa forma mais próxima do mundo real, pois as regras *fuzzy* são capazes de lidar com verdades parciais [Amorim et al. 2001].

As principais vantagens decorrentes da integração de técnicas de extração de regras e RNA são [Nobre et al. 1998] [Brasil 1999] [Taha & Ghosh 1999] [Braga et al. 2000]:

- ♣ Facilidade de compreensão do conhecimento minerado;
- ♣ As regras podem ser utilizadas para verificar a adequação da arquitetura neural;
- ♣ Facilitar a aceitação pelo usuário e, como resultado, ampliar o escopo de aplicação das RNA;
- ♣ Descoberta de novos relacionamentos e características, possibilitando a formulação de novas teorias;
- ♣ As regras extraídas contribuem para um melhor entendimento do relacionamento entrada-saída, ajudando a identificar os atributos que são mais relevantes;
- ♣ Além de possibilitarem uma descrição simbólica concisa e apurada da rede, as regras facilitam a integração das RNA com Sistemas Simbólicos, como os Sistemas Especialistas;
- ♣ Melhorar a generalização da rede, utilizando as regras para descobrir as circunstâncias em que a rede pode cometer erros de generalização ou identificar regiões no espaço de entradas que não estejam suficientemente representadas no conjunto de treinamento. Neste caso, dados adicionais podem ser usados para melhorar o desempenho da rede;
- ♣ Minimizar o problema de aquisição de conhecimento e refinar o conhecimento inicial adquirido pelos especialistas do domínio. O problema de aquisição de conhecimento é minimizado, pois é mais fácil para o especialista fazer comentários sobre as regras do que criar um conjunto consistente de regras com suas respectivas medidas de confiança.

5.4. Classificação das Técnicas de Extração de Regras

A extração de regras de uma RNA é realizada utilizando as funções aprendidas pelos nós da rede ou a classificação fornecida pela rede para os exemplos. De acordo com a fonte da qual as regras são extraídas, as técnicas podem ser subdivididas nas seguintes abordagens: decomposicional, pedagógica, eclética e composicional [Andrews et al. 1996] [Taha & Ghosh 1999] [Azevedo et al. 2000] [Braga et al. 2000] [Milaré & Carvalho 2001].

5.4.1. Decomposicional

Na abordagem decomposicional, também denominada de local, estrutural ou *Link Rule Extraction* (LRE), a principal fonte das regras é a estrutura da rede. Um conjunto de regras é extraído para descrever cada nó intermediário e de saída em relação aos nós que os antecedem. As regras extraídas para cada nó são então combinadas em um conjunto de regras que descreve a rede como um todo.

Uma condição básica para as técnicas desta categoria é que as saídas de cada nó intermediário e de saída devem ser mapeadas como um resultado booleano, o qual corresponde ao conseqüente de uma regra. Assim, cada nó intermediário e de saída pode ser interpretado como uma função degrau ou uma regra booleana, o que reduz o problema de extração de regras em um problema de determinar as situações nas quais a regra é verdadeira, ou seja, um conjunto de conexões cuja soma dos pesos garante que o limiar do nó seja excedido independentemente do valor da ativação presente em outras conexões.

As condições da regra para um dado nó podem ser expressas de duas formas:

- ♣ Em termos dos nós diretamente conectados: Neste caso, a extração da regra é facilitada;
- ♣ Apenas pelos atributos de entrada da rede: Pode ser a melhor opção para o caso em que a informação está distribuída na rede de tal forma que a informação associada a cada nó não tenha sentido.

Uma vantagem dessa abordagem é que ela produz termos intermediários, os quais resultam em descrições simples [Nobre et al. 1998]. No entanto, impõe restrições às arquiteturas e métodos de aprendizagem das redes, é computacionalmente cara e faz suposições simplistas sobre o comportamento da rede (requer que nós intermediários sejam aproximados por nós *threshold*), contribuindo para que as regras extraídas não forneçam uma representação muito apurada da rede [Faifer & Janikow 1999].

As técnicas KT [Fu 1994], MofN [Towell & Shavlik 1993], SUBSET [Towell & Shavlik 1993], NeuroRule [Lu et al. 1995], RuleNet [McMillan et al. 1991], EN [Pau & Gotzche 1992], RULEX [Andrews & Geva 1995], Partial-RE e Full-RE [Taha & Ghosh 1999] se enquadram nesta abordagem.

5.4.2. Pedagógica

Na abordagem pedagógica, também denominada de global, funcional, didática ou *Black-box Rule Extraction* (BRE), a rede é vista como uma caixa-preta. A extração de conhecimento é uma tarefa de aprendizagem cujo objetivo é aprender a função computada pela rede. Assim, as técnicas procuram extrair regras que mapeiam a entrada diretamente na saída, sem se preocupar com os passos intermediários.

As técnicas pedagógicas normalmente são usadas em conjunto com algum algoritmo de aprendizagem simbólico e não requerem qualquer tipo de informação sobre a arquitetura e pesos da rede. As desvantagens apresentadas pela abordagem decomposicional são minimizadas na abordagem pedagógica [Faifer & Janikow 1999]. Exemplos de técnicas desta abordagem são: RULENEG [Pop et al. 1994], BRAINNE [Sestito & Dillon 1995], DectText [Boz 2002], TREPAN [Craven 1996], *FuzzyTrepan* [Faifer & Janikow 1999] e BIO-RE [Taha & Ghosh 1999].

5.4.3. Eclética

A abordagem eclética, também denominada de combinacional, é uma combinação das abordagens decomposicional e pedagógica. Portanto, as técnicas ecléticas utilizam conhecimento sobre a arquitetura interna e/ou os vetores de pesos da rede para complementar um algoritmo de aprendizagem simbólico que utiliza os dados de treinamento.

5.4.4. Composicional

As técnicas composicionais realizam a extração de conhecimento utilizando *clusters* de nós ao invés de nós individuais. Um exemplo dessa abordagem é o método desenvolvido por [Schellharmmer et al. 1997], que extrai regras gramaticais de redes recorrentes.

5.5. Técnicas de Extração de Regras Clássicas

Esta seção descreve as principais técnicas de extração de regras clássicas de RNA.

5.5.1. DEDEC

O método DEDEC (*Decision Detection*) [Tickle et al. 1996] é aplicado para extração de regras Se-Então de redes MLP treinadas com o algoritmo *Backpropagation* ou *Cascade Correlation*. Este método procura combinar a robustez das RNA com a capacidade de explicação dos algoritmos de aprendizagem indutivo.

Para construir as regras, DEDEC seleciona um subconjunto de exemplos, utilizando a rede treinada e informações extraídas dos pesos. O subconjunto de exemplos pode incluir exemplos não utilizados no treinamento da rede. As entradas da rede são ordenadas de acordo com suas contribuições para as saídas. A ordenação é realizada examinando os vetores de pesos, o que coloca DEDEC na fronteira entre as abordagens decomposicional e pedagógica. No passo seguinte, as entradas são agrupadas e cada *cluster* é usado para gerar um conjunto de regras binárias que descreve as dependências funcionais entre os atributos do *cluster* e as saídas da rede.

5.5.2. SUBSET

A idéia básica do SUBSET [Towell & Shavlik 1993] é procurar subconjuntos de pesos para cada nó cuja soma supera seu limiar. Esta técnica considera que os pesos podem assumir valores positivos e negativos, e que os nós estão na ativação máxima (próxima de 1) ou inativos (próxima de 0).

O algoritmo SUBSET pode ser descrito da seguinte forma:

Tabela 5.1 - Algoritmo SUBSET

Para cada nó intermediário e de saída: Formar S_p subconjuntos, combinando pesos positivos cujo somatório supera seu limiar; Para cada elemento *P* dos subconjuntos S_p : Formar S_n subconjuntos de *N* elementos, considerando as combinações mínimas de pesos negativos, de forma que a soma absoluta destes pesos seja maior do que a soma de *P* menos o limiar do nó; Formar a regra: *Se P e não N Então <nome do nó>*

Depois do processo de extração, as regras que são muito específicas (cobrem uma porção muito pequena da base de dados) são removidas.

A Figura 5.1 [Azevedo et al. 2000] mostra um exemplo de aplicação do algoritmo SUBSET.

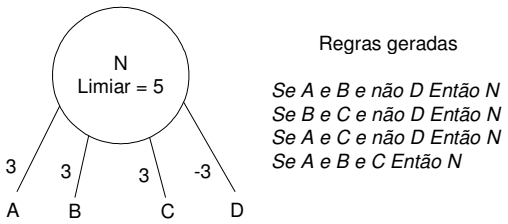

Figura 5.1 - Exemplo do algoritmo SUBSET

Apesar das regras extraídas serem simples, a técnica SUBSET apresenta as seguintes limitações:

- ♣ Pode gerar regras repetidas;
- ♣ Não garante que todo conhecimento útil incorporado na rede seja extraído;
- ♣ As regras podem ter antecedentes com muitas condições, tornando proibitivo o seu uso prático;
- ♣ Para uma grande quantidade de conexões, encontrar todos os subconjuntos é computacionalmente caro. Conseqüentemente, o método é mais apropriado para redes pequenas. Em alguns casos, é utilizado um limite no tamanho dos subconjuntos. No entanto, esta alternativa não é adequada para aplicações do mundo real que podem requerer um número grande de antecedentes nas regras extraídas.

5.5.3. MofN

O algoritmo MofN [Towell & Shavlik 1993] extrai conhecimento de uma rede KBANN (*Knowledge-Based Artificial Neural Network*) [Towell & Shavlik 1994]. Este algoritmo foi proposto com o objetivo de reduzir as limitações da técnica SUBSET, diferindo desta, basicamente, na forma de representação das regras. O formato das regras é:

Se (M das N condições são verdadeiras) Então ...

Neste tipo de representação, deve-se supor que grupos de condições formam classes equivalentes, nas quais cada condição tem a mesma importância e podem ser trocadas entre si.

A Tabela 5.2 descreve os passos do algoritmo MofN:

Tabela 5.2 - Algoritmo MofN

1. Para cada nó intermediário e de saída, formar grupos de pesos numericamente similares;
2. Atribuir aos pesos de cada grupo o valor médio calculado para o grupo;
3. Eliminar qualquer grupo que não tenha efeito significativo para a ativação do nó (valor médio pequeno);
4. Mantendo os pesos constantes, otimizar os valores dos limiares de todos os nós através do algoritmo de retropropagação;
5. Extrair uma regra a partir dos grupos, considerando a soma dos pesos dos grupos e o valor dos limiares otimizados;
6. A partir da regra geral, simplificar pesos e combinações possíveis.

5.5.4. RULEX

RULEX [Andrews & Geva 1995] foi desenvolvida para extrair conhecimento de RNA treinadas com o algoritmo CEBP (*Constrained Error Backpropagation*).

Os nós intermediários da rede CEBP são unidades localmente suscetíveis (*Locally Responsive Units* – LRU) baseadas na função sigmóide, que tem o efeito de particionar os dados em um conjunto de regiões distintas representadas por um nó intermediário.

Cada LRU é composta de um conjunto de arestas, uma para cada entrada. Uma aresta produz uma saída plausível se o valor apresentado como entrada estiver dentro da faixa ativa da aresta. A saída da LRU é a soma do limite das ativações das arestas. Assim, para que um vetor de entrada seja

classificado por uma LRU, cada componente do vetor deve estar dentro da faixa ativa da aresta. Desta forma, é possível extrair regras proposicionais das LRU, com o seguinte formato:

Se Aresta$_1$ é ativada e Aresta$_2$ é ativada e ... e Aresta$_n$ é ativada
Então o exemplo pertence à classe C

A técnica RULEX também possui mecanismos para manter antecedentes negativos e remover antecedentes e regras redundantes. Diferentemente de outros métodos decomposicionais, tais como KT e SUBSET, os quais empregam variações de técnicas de *procura e teste*, RULEX extrai as regras interpretando diretamente os pesos da rede. Como resultado, RULEX não apresenta os problemas computacionais existentes nas outras técnicas decomposicionais.

5.5.5. EN

O método *Explanation Facility* (EN) [Pau & Gotzche 1992] fornece recursos para responder, em forma de regras, a perguntas do tipo *Por quê?* e *Como?* feitas a uma RNA treinada. O primeiro tipo define por que a rede gerou uma dada resposta. O segundo explica como uma saída foi produzida para um conjunto de valores de entrada.

Para prover explicações, o método EN seleciona um conjunto de nós baseando-se nos valores absolutos dos pesos ligados a cada nó. Deste conjunto de nós selecionados, apenas os mais significativos são utilizados para a definição das regras.

Este método possui uma implementação simples e pode ser utilizado em quase todos os modelos neurais.

5.5.6. TREPAN

TREPAN (*Trees Parroting Networks*) [Craven 1996] trata a extração de conhecimento como uma tarefa de aprendizagem indutiva, em que o conceito alvo é a função representada por um oráculo e a hipótese produzida é uma árvore de decisão que aproxima o seu comportamento [Craven & Shavlik 1998]. Uma vez que a função alvo nesta dissertação é o conhecimento adquirido por uma RNA, a própria rede é usada como oráculo (Figura 5.2).

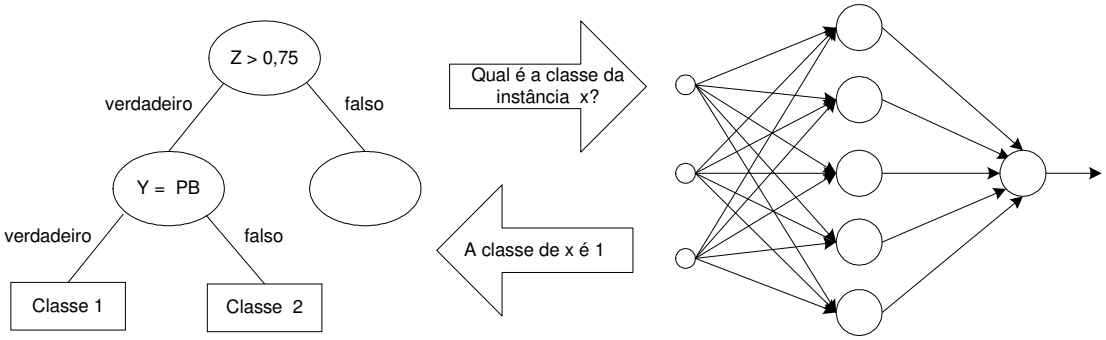

Figura 5.2 - Interação de TREPAN com uma RNA

Durante o processo de aprendizagem, TREPAN faz perguntas à RNA utilizando o conjunto de treinamento e exemplos gerados artificialmente [Craven & Shavlik 1996]. O papel da rede é determinar a classe de cada exemplo apresentado como consulta. As respostas da rede são utilizadas para a construção da árvore.

Como a interação com a rede consiste apenas de consultas de pertinência, TREPAN não faz nenhuma restrição com relação à arquitetura da rede ou ao algoritmo de treinamento utilizado. Na realidade, TREPAN é uma técnica bastante genérica, podendo ser aplicada a uma ampla variedade de modelos de aprendizagem, não sendo restrita apenas às RNA. Além disso, TREPAN apresenta boa escalabilidade para problemas com bases de dados e RNA extensas [Craven & Shavlik 1999].

Diversos estudos têm demonstrado que nas aplicações em que as RNA apresentam melhor capacidade preditiva que os algoritmos convencionais de árvore de decisão TREPAN é capaz de extrair árvores que aproximam bastante as hipóteses aprendidas pelas RNA. Conseqüentemente, as árvores geradas por TREPAN fornecem capacidade preditiva superior às obtidas diretamente por algoritmos como o C4.5 [Craven & Shavlik 1998].

5.6. Técnicas de Extração de Regras *Fuzzy*

Paralelo ao desenvolvimento de técnicas de extração de regras clássicas, diversas técnicas de extração de regras *fuzzy* têm sido propostas [Azevedo et al. 2000].

[Masuoka et al. 1991] desenvolveram um dos primeiros trabalhos nesta área, usando uma abordagem decomposicional. Antes da extração de regras de uma RNA, três fases são executadas. Na primeira fase, a função de pertinência de cada antecedente é representada. As operações *fuzzy* {*E*, *OU*, *NÃO*} nos antecedentes e as funções de pertinência que constituem os conseqüentes são representadas na segunda e terceira fases, respectivamente. Um conjunto compacto de regras é obtido através da remoção dos pesos que são menores que um valor limite.

De modo similar, [Berenji 1991] demonstrou o uso de uma RNA para refinar uma base de conhecimento (regras *fuzzy*) usada como parte de um controlador. A característica marcante desta técnica é que o conjunto inicial de regras é conhecido. Além disso, a RNA é usada para modificar as funções de pertinência dos antecedentes e conseqüentes.

[Horikawa et al. 1992] desenvolveram três RNA *fuzzy* capazes de identificar automaticamente regras *fuzzy* relevantes e refinar as funções de pertinência, modificando os pesos através do algoritmo *Backpropagation*. Nesta abordagem, a base de regras iniciais é criada usando o conhecimento de um especialista ou iterando seletivamente através de possíveis combinações dos atributos de entrada e funções de pertinência.

O FNES (*Fuzzy Neural Expert System*) [Hayashi 1989] e o modelo *fuzzy* para RNA multicamadas de [Mitra et al. 1994] são direcionados especificamente para o problema de prover ao usuário uma explicação de como uma conclusão foi atingida. Em ambas as técnicas, o antecedente das regras é determinado pela análise e classificação dos pesos. Entretanto, enquanto o FNES lida com o envolvimento de um especialista na fase inicial de conversão dos dados de entrada, no modelo *fuzzy* para RNA multicamadas, este processo foi automatizado.

[Huang & Xing 2002] apresentam uma abordagem para representar atributos de entrada contínuos através de termos lingüísticos baseados na Lógica *Fuzzy* (discretização) e extrair as regras *fuzzy* mais relevantes de uma RNA binária. Esta abordagem aplica uma técnica de poda para simplificar as regras e gera uma regra para cada nó da camada de saída. Sua principal limitação é a restrição da aplicação da técnica de extração de regras apenas para redes sem nós intermediários, limitando seu escopo

para resolução de problemas linearmente separáveis. Uma forma de superar esta limitação é aumentar a inclinação da função sigmóide dos nós intermediários, a fim de que ela se aproxime de uma função de ativação binária. O algoritmo é então aplicado separadamente para as camadas intermediária e de saída. Em seguida, as regras extraídas são agregadas. As regras *fuzzy* extraídas fornecem uma representação simples de processos complexos e refletem um tipo de conhecimento que pode ser aplicado em problemas reais. A interpretação destas regras pode ser feita através de modelos neuro-*fuzzy* ou métodos convencionais de inferência *fuzzy*.

[Kasabov 1996] e [Kasabov et al. 2001] propuseram, respectivamente, os métodos de extração de regras *fuzzy* REFuNN (*Rule Extraction from a FuNN*) e AREFuNN (*Aggregated Rule Extraction from a FuNN*). Estes métodos são bastante simples; podem extrair regras simplificadas, agregadas e ponderadas; e foram propostos especificamente para o modelo FuNN. No método REFuNN, cada nó regra é representado, no mínimo, por uma regra (usualmente por mais de uma). Por outro lado, no método AREFuNN, a quantidade de regras não é maior do que o número de nós regra.

[Li et al. 2002] propuseram um mecanismo de extração de regras para a rede FWD. Tal mecanismo possibilita a representação do conhecimento na forma de regras *fuzzy* Se-Então. O processo de extração de regras é realizado de forma direta através das conexões de memória m_{ji}, obtidas durante a fase de treinamento.

5.7. Técnicas Selecionadas

Os principais fatores que devem ser considerados durante a escolha da técnica de extração de conhecimento a ser aplicada em um determinado problema são [Taha & Ghosh 1999]:

- ♣ Precisão;
- ♣ Fidelidade;
- ♣ Formato de representação;
- ♣ Capacidade de generalização;
- ♣ Tipos de dados que podem ser manipulados;
- ♣ Compreensibilidade do conhecimento extraído;
- ♣ Granularidade ou nível de detalhe das explicações;
- ♣ Transparência (quão bem as conclusões podem ser explicadas);
- ♣ Complexidade e escalabilidade (aspectos relevantes para problemas de larga escala);
- ♣ Portabilidade ou generalidade (capacidade de ser aplicada em diferentes tipos de modelos).

Considerando estes fatores, a técnica TREPAN foi escolhida para investigação nesta dissertação. Esta técnica apresenta as seguintes vantagens:

- ♣ Representação do conhecimento numa forma compreensível;
- ♣ Pode ser utilizada em aplicações que apresentam tanto atributos discretos como contínuos;
- ♣ Capacidade de produzir árvores de decisão sucintas de redes grandes aplicadas em domínios de larga escala;

- ♣ Capacidade de produzir árvores de decisão precisas, compreensíveis e com alto nível de fidelidade;
- ♣ Escalável em relação ao tamanho da base de dados, complexidade dos modelos aprendidos e tempo de execução (o usuário tem controle sobre a complexidade da árvore);
- ♣ Genérico em sua aplicabilidade (não requer a utilização de um procedimento especial de treinamento e não impõe restrições sobre o modelo utilizado como oráculo).

As técnicas de extração de regras *fuzzy* dos modelos FWD e FuNN (REFuNN e AREFuNN) também serão investigadas nesta dissertação.

As técnicas REFuNN e AREFuNN fornecem um mecanismo de definição de limiares para as condições e conclusões. Através deste mecanismo, o usuário pode controlar a complexidade da base de regras. Os valores dos limiares definem quais condições irão compor cada regra, determinando, desta forma, o número de regras e a qualidade das mesmas. Como resultado, a interpretação das regras é facilitada. Dependendo dos valores estabelecidos para os limiares, as regras podem ser apresentadas em diferentes níveis de abstração. Quanto maior o limiar, mais genéricas são as regras e melhor a qualidade de explicação (nível de compreensibilidade). No entanto, a qualidade de interpretação (precisão dos resultados quando as regras são usadas para raciocínio e inferência em novos dados) é afetada. Portanto, regras genéricas podem ser usadas para finalidades de explicação e entendimento do conhecimento contido nos dados de treinamento, enquanto que regras precisas e específicas podem ser usadas para inferência sobre novos dados. Outra vantagem destas técnicas é a apresentação de graus de confiança nas regras.

O processo de extração de regras do modelo FWD é realizado de forma direta através das conexões de memória m_{ji}, obtidas durante a fase de treinamento. Nesta técnica, para cada classe, é gerada uma regra, que contém todos os atributos de entrada no antecedente.

5.7.1. TREPAN

TREPAN [Craven 1996] trata a extração de conhecimento como uma tarefa de aprendizagem indutiva, em que o conceito alvo é a função representada por um oráculo e a hipótese produzida é uma árvore de decisão que aproxima o seu comportamento [Craven & Shavlik 1998]. Uma vez que a função alvo nesta dissertação é o conhecimento adquirido por uma RNA, a própria rede é usada como oráculo.

Os passos do algoritmo TREPAN [Craven 1996] são descritos na Tabela 5.3.

Consultas de Pertinência

A generalidade de TREPAN deriva do fato de sua interação com o oráculo consistir apenas de consultas de pertinência (perguntas compostas de um exemplo) [Craven & Shavlik 1998].

As consultas de pertinência são usadas para obter da rede a classificação dos exemplos de treinamento ou artificiais. A classificação não é necessariamente correta. No entanto, uma vez que o interesse é induzir uma descrição da rede, a classificação fornecida por ela é considerada verdadeira. Esta consideração garante a fidelidade da descrição extraída com o conhecimento incorporado pela rede [Faifer & Janikow 1999].

Tabela 5.3 - Algoritmo TREPAN

```
Entradas: oráculo( ), conjunto de treinamento S, conjunto de atributos F, número mínimo de
amostras min_amostra e número máximo de nós internos num_máximo
Para cada exemplo x ∈ S
  rótulo_classe_x = oráculo(x)
Inicializar a raiz, R, da árvore como um nó folha
Construir um modelo M da distribuição das instâncias cobertas por R
query_instances_R = obterAmostra( { }, min_amostra - |S|, M )
Usar S e query_instances_R para determinar rótulo_classe_R
Inicializar fila com a tupla < R, S, query_instances_R, { } >
Enquanto fila não estiver vazia e num_máximo não for atingido
  Remover < nó N, S_N, query_instances_N, restrições_N > da cabeça de fila
  T = obterTeste( F, S_N ∪ query_instances_N )
  Tornar N um nó interno com teste T
  Para cada resultado, t, do teste T
    Tornar C um nó filho de N
    restrições_C = restrições_N ∪ {T = t}
    S_C = membros de S_N com resultado t no teste T
    Construir um modelo M da distribuição de instâncias cobertas pelo nó C
    query_instances_C = obterAmostra( restrições_C, min_amostra - |S_C|, M )
    Usar S_C e query_instances_C para determinar rótulo_classe_C
    Se critério de parada local não for satisfeito então
      Colocar <C, S_C, query_instances_C, restrições_C> na fila
    Senão
      Inserir o nó C na árvore como um nó folha
Retornar árvore com raiz R
```

Exemplos Artificiais

Uma das maiores limitações dos algoritmos convencionais de árvore de decisão é que a quantidade de dados de treinamento usada para definir o teste dos nós internos ou rotular folhas diminui com a profundidade da árvore. Desta forma, normalmente os testes e rótulos dos nós próximos da base são pobremente escolhidos, pois tais decisões são baseadas em poucos dados. Isso pode causar *overfitting* [Faifer & Janikow 1999]. Diferentemente desses algoritmos, através do uso de exemplos artificiais, em adição aos exemplos de treinamento, TREPAN sempre toma decisões baseando-se em amostras grandes, ou seja, TREPAN garante que existe pelo menos uma quantidade mínima de exemplos antes de estabelecer o rótulo ou teste de um nó [Craven & Shavlik 1996].

A utilização dos exemplos artificiais, que estão sujeitos a um conjunto de restrições determinadas pela localização do nó na árvore, permite que alguma informação adicional sobre o comportamento da rede seja adquirida [Craven 1996] [Faifer & Janikow 1999].

A forma como o conjunto de restrições é convertido em um exemplo a ser usado como consulta de pertinência é um ponto-chave de TREPAN. TREPAN constrói um modelo da distribuição dos dados e o utiliza para gerar os exemplos. O modelo é construído através de uma abordagem simples baseada na modelagem de distribuições marginais. A distribuição marginal se refere à distribuição de um atributo independente dos demais [Craven 1996].

TREPAN usa distribuições empíricas para modelar atributos discretos e estimativas de densidade de Kernel [Silverman 1986] para modelar atributos contínuos. A distribuição empírica é a distribuição dos valores do atributo que ocorrem em uma dada amostra. O método de estimação de densidade de Kernel modela a função de densidade de probabilidade para atributos contínuos *x* como:

$$f(x) = \frac{1}{m}\sum_{j}^{m}\left[\frac{1}{\sqrt{2\pi\sigma}}e^{-\left(\frac{x-\mu_j}{2\sigma}\right)^2}\right] \tag{5.1}$$

onde *m* é o número de exemplos de treinamento usados na estimativa, μ_j é o valor do atributo para o *j*-ésimo exemplo e σ é a largura do Kernel gaussiano. TREPAN considera σ como sendo $1/\sqrt{m}$.

Este tipo de modelagem de distribuições apresenta uma limitação. Como são estimadas apenas distribuições marginais, as dependências entre os atributos não são consideradas. Negligenciar tais dependências pode resultar em árvores grandes que apresentam relacionamentos sem importância e gerar exemplos que podem não ser observados no mundo real [Faifer & Janikow 1999]. TREPAN supera parcialmente esta limitação, estimando as distribuições localmente à medida que a árvore é gerada. Desta forma, algumas das dependências condicionais são capturadas e um modelo mais preciso da distribuição real é obtido [Craven 1996].

Outro problema da geração de exemplos artificiais é não considerar a idéia de que os exemplos artificiais deveriam ser usados apenas para melhorar a extração de conhecimento em sub-espaços onde a probabilidade de encontrar exemplos é relativamente alta [Faifer & Janikow 1999].

[Milaré & Carvalho 2001] demonstraram através de vários experimentos que a geração de exemplos artificiais realizada por TREPAN pode não ser uma boa estratégia. Embora seja possível verificar se cada exemplo artificial tem valores válidos para os atributos individualmente, é muito difícil certificar que cada combinação de valores seja válida. Conseqüentemente, alguns dos exemplos gerados podem não ser observados no mundo real. Os experimentos realizados sugerem que os exemplos gerados com o intuito de aumentar a fidelidade entre TREPAN e a rede podem reduzir a fidelidade sobre exemplos reais ou não acarretar diferença.

Expansão da Árvore

TREPAN mantém uma fila de nós que são inseridos na árvore à medida que são removidos da fila. Para cada nó na fila, são armazenados [Craven & Shavlik 1996]:

- O subconjunto dos exemplos de treinamento que alcançam o nó;
- O conjunto de exemplos artificiais usados, juntamente com o subconjunto dos exemplos de treinamento, para selecionar o teste, se o nó é interno, ou determinar o rótulo da classe, se o nó é folha;
- O conjunto de restrições que os exemplos devem satisfazer para alcançar o nó. Este conjunto é usado na geração dos exemplos artificiais.

O processo de expansão é muito parecido com o utilizado nos algoritmos convencionais de árvore de decisão: um teste é selecionado para um nó e um filho é criado para cada resultado do teste. Cada filho torna-se folha ou é colocado na fila para posterior expansão. No entanto, diferente da maioria dos

algoritmos de árvore de decisão, que gera a árvore numa forma *depth-first* (expande o nó mais profundo), TREPAN gera a árvore usando uma expansão *best-first* (expande o melhor nó segundo alguma heurística). A motivação de expandir a árvore na forma *best-first* é que tal abordagem fornece ao usuário algum controle sobre o tamanho da árvore a ser retornada (o processo de expansão pode ser interrompido em qualquer momento) [Craven & Shavlik 1998].

Durante a expansão, o melhor nó é aquele que apresenta o maior potencial para aumentar a fidelidade da árvore extraída com a rede. A função usada para avaliar um nó *N* é:

$$f(N) = alcance(N)(1 - fidelidade(N)) \tag{5.2}$$

onde *alcance(N)* é a fração estimada de exemplos de treinamento e artificiais que alcançam *N* e *fidelidade(N)* é a fração estimada dos exemplos para os quais a árvore e a rede concordam em suas predições [Craven & Shavlik 1999].

O algoritmo de aprendizagem expande a árvore até que os exemplos de treinamento sejam suficientemente separados ou algum critério de parada seja satisfeito [Craven 1996].

Teste dos Nós Internos

Definir o teste para um nó interno de uma árvore de decisão implica em decidir como o subconjunto do espaço de exemplos coberto pelo nó será particionado. Além dos testes envolvendo apenas um atributo, TREPAN também usa expressões *m-de-n* para gerar descrições mais compactas. Uma expressão *m-de-n* é satisfeita quando pelo menos *m* de seus *n* literais booleanos são satisfeitos [Craven & Shavlik 1996].

A escolha do melhor teste binário de um nó é feita usando o critério de ganho de informação [Quinlan 1993]. Para atributos cujo domínio é composto de dois valores, um teste separa os exemplos de acordo com esses valores (Ex.: sexo = F?, sexo = M?). Para atributos discretos com mais de dois valores, é gerado um teste para cada valor (Ex.: profissão = analista?, profissão = professor?, ...). Para atributos contínuos, TREPAN compõe os testes com limiares (Ex.: salário > 1.500). O conjunto de limiares candidatos para um nó é determinado ordenando os valores presentes nos exemplos de treinamento que alcançam o nó e obtendo os *midpoints* entre valores adjacentes cujos rótulos da classe são diferentes [Craven 1996].

Critério de Parada

A maioria dos algoritmos de árvore de decisão usa critério de parada local, ou seja, ao decidir se o nó será expandido em uma sub-árvore ou tornar-se-á uma folha, a decisão é local, baseando-se apenas nas características do nó. TREPAN, por outro lado, usa critérios de parada global e local [Craven & Shavlik 1998].

O critério de parada local usado por TREPAN é baseado na pureza do conjunto de exemplos cobertos pelo nó. O nó torna-se folha se, com alta probabilidade, ele cobre apenas exemplos de uma classe.

Um dos critérios de parada global (considera o estado da árvore inteira) é um limite no tamanho da árvore que TREPAN retorna. Este parâmetro, que é especificado em termos de nós internos, fornece ao usuário algum controle sobre a compreensibilidade das árvores produzidas. TREPAN também é capaz de usar um conjunto de validação. Como TREPAN gera árvores numa forma *best-first*, o

processo de extração pode ser considerado como produzindo uma seqüência aninhada de árvores em que cada árvore difere de sua predecessora apenas pela sub-árvore que corresponde ao nó expandido no último passo. Quando um conjunto de validação é fornecido, TREPAN utiliza-o para medir a fidelidade de cada árvore nesta seqüência. Em seguida, a árvore que tem o maior nível de fidelidade é retornada.

Poda

Depois que algum critério de parada é satisfeito, TREPAN emprega uma forma muito simples de poda antes de retornar a árvore final. A finalidade deste passo é detectar sub-árvores que prevêem a mesma classe em todas as suas folhas e uni-las em um único nó folha. As modificações removem nós internos irrelevantes e não alteram o comportamento preditivo da árvore [Craven 1996].

Classificação

Para realizar a classificação de um exemplo, os valores dos atributos do exemplo são utilizados na execução dos testes dos nós internos da árvore. A verificação inicial é feita com o teste do nó raiz. À medida que os testes são realizados, o exemplo percorre um caminho na árvore. O processo de classificação termina quando o exemplo alcança um nó folha, cujo rótulo é considerado a classe do exemplo [Faifer & Janikow 1999].

Representação dos Dados

Diferente da maioria das abordagens de extração de conhecimento onde a representação é feita na forma de regras, as descrições produzidas por TREPAN são representadas na forma de árvores de decisão.

Existem várias vantagens de usar árvores de decisão como forma de representação, ao invés de regras. A representação na forma de árvore de decisão fornece ao algoritmo de extração algum grau de controle sobre a complexidade e fidelidade da representação extraída. Inicialmente, TREPAN extrai uma descrição muito simples (um nó) de uma rede treinada e, em seguida, refina sucessivamente esta descrição para melhorar sua fidelidade. Outra vantagem é que uma árvore de decisão fornece uma cobertura completa do espaço de exemplos, ou seja, prevê a classe de todos os pontos no espaço de instâncias [Craven & Shavlik 1999]. Além dessas vantagens, uma árvore de decisão pode ser facilmente convertida em um conjunto de regras e apresentar os resultados numa forma gráfica que pode ser facilmente visualizada e compreendida [Quinlan 1993].

Variações

Diversas soluções baseadas na técnica TREPAN foram propostas. [Milaré & Carvalho 2001] utilizaram os princípios básicos de TREPAN e aplicaram os algoritmos simbólicos C4.5 [Quilan 1993], C4.5*rules* [Quilan 1993] e CN2 [Clark e Nibeltt 1989] para extrair representações simbólicas de RNA. A idéia principal de [Milaré & Carvalho 2001] é desenvolver um procedimento geral que, como TREPAN, possa ser aplicado para qualquer modelo de rede, independente de sua arquitetura, processo de treinamento, tamanho e valores de entrada. Diferente do TREPAN, o procedimento proposto não gera exemplos artificiais.

[Faifer & Janikow 1999] propuseram a técnica *FuzzyTrepan* que estende TREPAN em dois aspectos: usa uma representação *fuzzy* (árvores de decisão *fuzzy*) na etapa de extração do conhecimento e heurísticas adicionais no processo de geração de exemplos artificiais. *FuzzyTrepan* difere de TREPAN em outros aspectos, como por exemplo, critérios de parada e criação de nós folhas.

5.7.2. REFuNN

O algoritmo REFuNN (*Rule Extraction from a FuNN*) [Kasabov 1996] é um método simples de extração de regras *fuzzy* de uma rede FuNN. Neste método, cada nó regra da rede é representado, no mínimo, por uma regra (usualmente por mais de uma) [Kasabov et al. 2001].

No método REFuNN, um conjunto de regras é extraído da seguinte forma:

- Extrair o conjunto inicial de regras ponderadas: Todas as conexões para um nó ação que contribuem significativamente para sua possível ativação (seus valores são maiores que um limiar Th_a) são escolhidas e os nós regra são analisados. Apenas os nós condição que contribuem significativamente para a ativação do nó regra (os pesos das conexões estão acima de um limiar Th_c) são usados no antecedente da regra. O número de predicados *fuzzy* permitidos no antecedente da regra não é maior do que o número de atributos de entrada (no máximo um predicado *fuzzy* por atributo). Os pesos entre os nós condição e os nós regras são tomados como graus de importância das proposições *fuzzy* do antecedente. Os pesos entre um nó regra e um nó ação definem o grau de certeza. O limiar Th_c pode ser calculado usando a equação:

  $$Th_c = Net_{max}/k \tag{5.3}$$

  onde Net_{max} é o valor desejado do *net input* para disparar a regra correspondente e k é o número de atributos de entrada.

  O operador lógico *E* é usado para conectar as condições e conclusões das regras;

- Extrair o conjunto de regras simples a partir do conjunto de regras ponderadas: O conjunto inicial de regras ponderadas pode ser convertido em um conjunto de regras simples removendo os graus de importância e certeza. Algumas condições, entretanto, podem disparar as regras sem qualquer suporte das demais condições, ou seja, seus graus de importância são maiores do que o limiar $Th_{OU} = Net_{max}$. Tais condições formam regras separadas. Esta transformação é análoga à decomposição de regras com conectivo *OU* em regras com conectivo *E*;

- Agregar as regras ponderadas iniciais: Todas as regras ponderadas iniciais que têm as mesmas condições e conclusões, diferindo apenas nos graus de importância, são agregadas em uma regra. Os graus de importância são calculados para todas as condições de uma regra como a soma normalizada dos graus de importância das condições correspondentes nas regras iniciais.

Uma opção adicional no método REFuNN é a aplicação do conectivo *NÃO*. Neste caso, os pesos negativos cujos valores absolutos estão acima dos limiares Th_c e Th_a são considerados e os rótulos das entradas correspondentes são incluídos nas regras com o conectivo *NÃO*.

Dependendo dos valores dos limiares, o método REFuNN pode ser usado para produzir regras com diferentes qualidades de explicação e interpretação. O problema com este método é que um número excessivo de regras com baixa qualidade de explicação pode ser produzido quando tentando produzir regras com alta qualidade de interpretação. O método AREFuNN foi proposto com o objetivo

de superar esta limitação. Desta forma, regras com alta qualidade de interpretação e explicação podem ser extraídas [Kasabov et al. 2001].

5.7.3. AREFuNN

O algoritmo AREFuNN (*Aggregated Rule Extraction from a FuNN*) [Kasabov et al. 2001] é um método simples de extração de regras *fuzzy* de uma rede FuNN. Neste método, o número de regras não é maior do que o número de nós regra da rede.

O método AREFuNN consiste dos seguintes passos [kasabov et al. 2001]:

♣ Os pesos entre a camada de condição e a camada de regra, $W_1(i,j)$, são normalizados:

$$W_{1n}(i,j) = \frac{W_1(i,j)}{\sum abs(W_1)}, \text{ para } i = 1, 2, ..., N_c \text{ e } j = 1, 2, ..., N_r \quad (5.4)$$

onde N_c é o número de nós condição e N_r é o número de nós regra;

♣ Os valores normalizados $W_{1n}(i,j)$ são limitados:

$$W_{1nt}(i,j) = W_{1n}(i,j), \text{ se } abs(W_{1n}(i,j)) > Thr_1$$
$$W_{1nt}(i,j) = 0, \text{ senão}$$

♣ Os pesos entre a camada de regra e a camada de ação, $W_2(j,k)$, são limitados de forma similar com o uso de um limiar Thr_2. Neste caso, $k = 1, 2, ..., N_a$, onde N_a é o número de nós ação;

♣ Uma regra R_j que representa um nó regra j é formada da seguinte maneira:

$$R_j: \text{Se } x_1 \text{ é } I_1 [W_{1nt}{}^1(i_1,j)] \text{ E } x_2 \text{ é } I_2 [W_{1nt}{}^2(i_2,j)] \text{ E } ... \; x_n \text{ é } I_n [W_{1nt}{}^n(i_n,j)]$$
$$\text{Então } y_1 \text{ é } L_1 [W_2{}^1(j,I_1)] \text{ E } y_2 \text{ é } L_2 [W_2{}^2(j,I_2)] \text{ E } ... \; y_m \text{ é } L_m [W_2{}^m(j,I_m)]$$

onde I_1, I_2, ..., I_n são rótulos *fuzzy* dos atributos de entrada x_1, x_2, ..., x_n, respectivamente, com os pesos mais altos para o nó regra j que estão acima do limiar Thr_1. L_1, L_2, ..., L_m são os predicados *fuzzy* dos atributos de saída y_1, y_2, ..., y_n, respectivamente, que são suportados pelo nó regra j com pesos acima do limiar Thr_2. Os valores $[W_{1nt}(i,j)]$ são interpretados como graus de importância associados às condições da regra R_j e os valores $[W_2(j,l)]$ são interpretados como graus de certeza associados às conclusões. Se $[W_{1nt}{}^k(i_k,j)]$ for menor que 0, a condição é interpretada como *"x_k NÃO é I_k"*;

♣ As regras que possuem antecedente e conseqüente iguais, diferindo apenas nos graus de importância e certeza, são agregadas tomando os valores máximos de cada condição ou conclusão.

5.7.4. FWD

O processo de extração de regras da rede FWD é realizado de forma direta através das conexões de memória m_{ji}, obtidas durante a fase de treinamento [Li et al. 2002]. Este processo utiliza a Equação 5.5, derivada das Equações 4.15 e 4.16.

$$z_i = exp[-\frac{1}{2\sigma^2} \cdot w_{i1}^2 \cdot (x_1 - m_{1i})^2] \cdot exp[-\frac{1}{2\sigma^2} \cdot w_{i2}^2 \cdot (x_2 - m_{2i})^2] \cdot ... \cdot exp[-\frac{1}{2\sigma^2} \cdot w_{in}^2 \cdot (x_n - m_{ni})^2]$$
$$= U_{A_{i1}}(x_1) \cdot U_{A_{i2}}(x_2) \cdot ... \cdot U_{A_{in}}(x_n), \qquad i = 1,2,...,c \quad n = 1,2,...,p \quad (5.5)$$

Nesta equação, z_i correspondem às saídas da rede para cada classe. O conjunto *fuzzy* do atributo *n* da classe *i* é representado por A_{in} e $U_{A_{in}}$ é a função de pertinência de A_{in}.

Cada saída da rede é normalizada utilizando a Equação 5.6, a fim de que o grau de pertinência do exemplo *x* para a classe c_i mantenha-se entre 0 e 1 e a soma das saídas seja igual a 1.

$$u_{ci} = U_{A_{i1}}(x_1) \cdot U_{A_{i2}}(x_2) \cdot \ldots \cdot U_{A_{in}}(x_n) / \sum_{j=1}^{c} U_{A_{j1}}(x_1) \cdot U_{A_{j2}}(x_2) \cdot \ldots \cdot U_{A_{jn}}(x_n), \quad i = 1,2,\ldots,n \qquad (5.6)$$

A construção das regras *fuzzy* é realizada da seguinte forma:

Se x_1 *é termo_lingüístico* m_{1i} *E ... E* x_n *é termo_lingüístico* m_{ni} $i = 1,2,\ldots,c$
Então x pertence à classe c_i $n = 1,2,\ldots,p$

O antecedente da regra corresponde ao lado direito da Equação 5.5 e o conseqüente ao lado esquerdo. Os termos *fuzzy termo_lingüístico* m_{1i} são representados pelos conjuntos *fuzzy* A_{i1}, ..., A_{in} desta equação.

5.8. Considerações Finais

Uma das maiores limitações das RNA é a dificuldade de gerar uma explicação de como a rede representa a solução de um problema, pois o conhecimento aprendido é representado de forma implícita pela topologia, valores dos pesos e limiares da rede; e o espaço de representação é dividido em regiões complexas por meio da combinação de várias funções matemáticas [Azevedo et al. 2000].

Como na maioria das aplicações reais os usuários desejam saber o raciocínio utilizado pelo sistema para obter uma dada conclusão, é importante que a rede seja capaz de fornecer uma explicação de suas saídas, a fim de que os usuários possam ter mais confiança nos resultados alcançados [Amorim et al. 2001] [Huang & Xing 2002]. Desta forma, cada vez mais, pesquisadores, projetistas e usuários acreditam que todo o potencial das RNA não poderá ser completamente explorado enquanto não for acrescentado a estes modelos um mecanismo que permita a explicação de suas decisões [Faifer & Janikow 1999].

Diversas abordagens têm sido propostas para superar esta limitação das RNA, integrando a capacidade de explicação dos sistemas que utilizam representação simbólica às funcionalidades das RNA [Nobre et al. 1998].

Este capítulo abordou as técnicas de extração de conhecimento simbólico de RNA. A aplicação dessas técnicas permite que o conhecimento incorporado pela rede seja representado numa forma mais compacta e compreensível. Como resultado, as RNA tornam-se mais adequadas para o processo de KDD, ampliando seu escopo de aplicação.

Capítulo 6

Neural Mining

6.1. Introdução

Embora KDD seja uma área recente com muitos aspectos que ainda precisam ser pesquisados em profundidade, uma grande quantidade de ferramentas de *software* de KDD e mineração de dados já foi desenvolvida [Han & Kamber 2001]. Inicialmente, as ferramentas eram mais utilizadas em ambientes de pesquisa e experimentais. Nos últimos anos, tem-se observado um rápido crescimento de ferramentas sofisticadas voltadas para o meio empresarial [Goebel & Gruenwald 1999]. O domínio de aplicação destas ferramentas está constantemente evoluindo devido ao desenvolvimento de novos sistemas, incorporação de novas funcionalidades aos sistemas existentes e proposição de novos métodos de mineração de dados. Apesar dessa evolução, poucas ferramentas têm adotado a abordagem dos Sistemas Neurais Híbridos [Goebel & Gruenwald 1999] [Data 2003] [Data 2004].

A proposta desta dissertação vai além do estudo do processo de KDD, dos Sistemas Neuro-*Fuzzy* e das técnicas de extração de conhecimento simbólico de RNA. Esta dissertação também visa ao desenvolvimento da ferramenta *Neural Mining* cujo principal objetivo é minorar o problema da pouca disponibilidade de ferramentas para KDD baseadas no paradigma neural híbrido.

Neural Mining é uma ferramenta de mineração de dados e apresentação do conhecimento responsável por capturar dados já consolidados, provenientes de uma base de dados bruta; aplicar algoritmos de mineração baseados na abordagem neural híbrida para geração de respostas ou decisões; e apresentar o conhecimento descoberto em forma de regras. A Figura 6.1 apresenta este fluxo de processos.

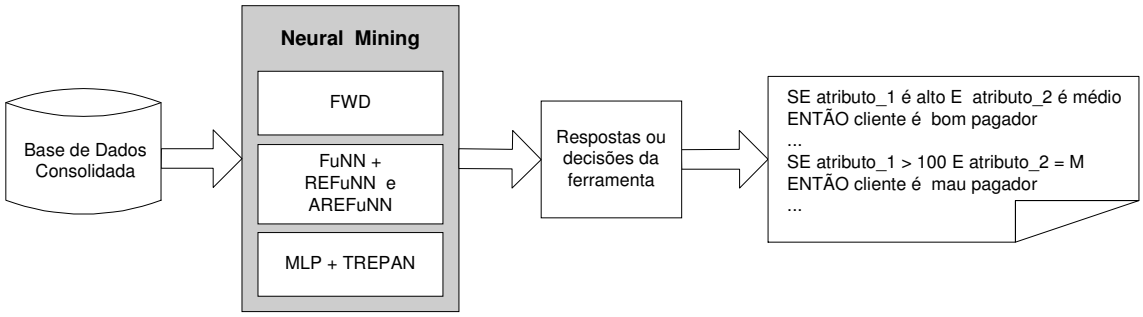

Figura 6.1 - Fluxo de processos da ferramenta *Neural Mining*

A ferramenta *Neural Mining* foi desenvolvida em Java e a interface foi implementada utilizando o pacote *Swing* [Deitel & Deitel 2001]. Devido às facilidades oferecidas pela linguagem Java, foram aplicados fundamentos da programação orientada a objetos, o que possibilitou a utilização dos conceitos de herança, encapsulamento e reusabilidade de código.

Este capítulo descreve a ferramenta *Neural Mining*, apresentando sua estrutura modular, os modelos e técnicas implementados e sua interface gráfica. Os formatos dos arquivos utilizados para armazenar as bases de dados, as configurações dos modelos e técnicas, e os resultados da classificação e extração de conhecimento são descritos no Apêndice B.

6.2. Algoritmos e Técnicas

As principais vantagens da ferramenta *Neural Mining* são a integração de vários métodos de aprendizagem neural híbrida num único ambiente e a característica de ter sido projetada para solução de problemas de larga escala. Os modelos e técnicas implementados na ferramenta *Neural Mining* são especificados nas próximas seções.

6.2.1. Algoritmos de Mineração de Dados

Os modelos neurais abordados pela ferramenta *Neural Mining* são: a rede MLP com o algoritmo de aprendizagem *Backpropagation* [Rumelhart et al. 1986] e as redes neuro-*fuzzy* FWD [Li et al. 2002] e FuNN [Kasabov et al. 1997].

O modelo MLP foi escolhido para ser aplicado em dois contextos: em conjunto com a técnica TREPAN [Craven & Shavlik 1996] e, separadamente, por ser um modelo tradicional, para comparar sua performance com os modelos neuro-*fuzzy* FWD e FuNN.

A motivação para investigar os modelos FWD e FuNN se baseou no fato destes métodos serem bastante promissores para serem aplicados ao processo de KDD. A rede FWD apresenta uma característica inovadora de englobar, dentro de uma mesma estrutura, três etapas do processo de KDD: seleção de dados, mineração de dados e apresentação do conhecimento. A investigação da rede FWD também é motivada pelo fato de que existem poucos resultados experimentais demonstrando sua viabilidade em aplicações reais [Li et al. 2002] [Amorim et al. 2003]. O modelo FuNN foi escolhido por apresentar as seguintes características: permite o uso de limiares no processo de extração de regras, possibilitando a redução do número de regras e condições nos antecedentes das regras; permite a inserção e o refinamento de regras usadas para definir a arquitetura inicial da rede, possibilitando o uso de duas fontes de dados (base de dados e regras iniciais); e tem apresentado resultados satisfatórios em diversos domínios de aplicação.

6.2.2. Técnicas de Extração de Regras

Com relação à extração de regras, *Neural Mining* aborda as seguintes técnicas: AREFuNN [Kasabov et al. 2001] e REFuNN [Kasabov 1996], que extraem regras *fuzzy* de uma rede FuNN; a técnica do modelo FWD, que extrai regras *fuzzy*; e a técnica TREPAN, que extrai uma árvore de decisão de qualquer modelo. Neste trabalho, TREPAN é utilizada em conjunto com a rede MLP.

A técnica TREPAN foi escolhida por apresentar as seguintes vantagens: permite representação do conhecimento numa forma compreensível; pode ser utilizada para aplicações envolvendo tanto atributos discretos como contínuos; possui a capacidade de produzir árvores de decisão sucintas de redes grandes aplicadas em domínios de larga escala; é capaz de produzir árvores de decisão precisas, compreensíveis e com alto nível de fidelidade; é escalável em relação ao tamanho da base de dados, complexidade dos modelos aprendidos e tempo de execução (o usuário tem controle sobre a complexidade da árvore retornada); e é genérica em sua aplicabilidade (não requer a utilização de um procedimento especial de treinamento e não impõe restrições sobre o modelo utilizado como oráculo).

Os detalhes funcionais dos modelos e técnicas abordados pela ferramenta não serão abordados neste capítulo porque os mesmos já foram descritos nos capítulos anteriores.

6.3. Estrutura Modular

Os módulos que compõem a ferramenta *Neural Mining* foram organizados de forma que permitissem uma fácil navegação. A divisão foi baseada nos modelos e técnicas selecionados para serem investigados nesta dissertação e nas suas funcionalidades, sendo criada uma estrutura modular de três níveis (Figura 6.2).

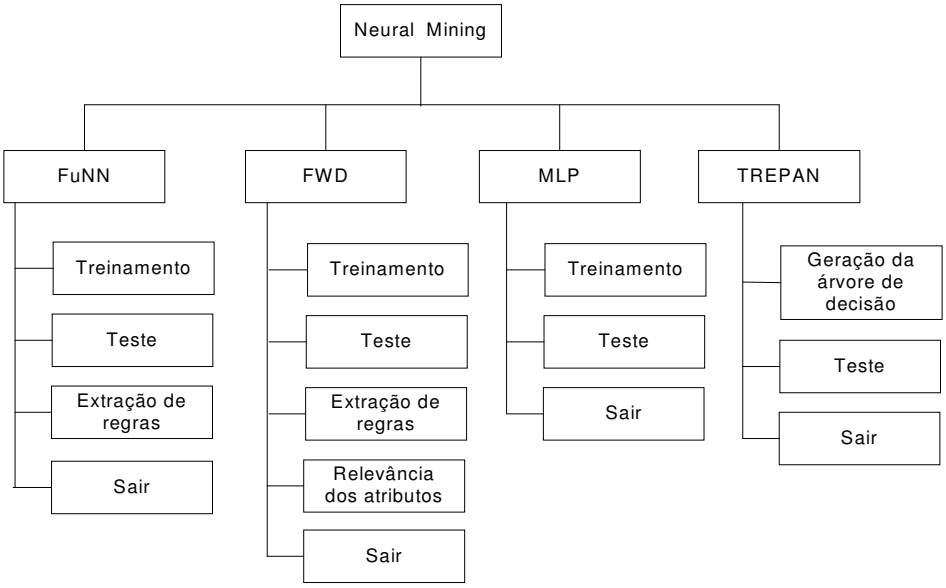

Figura 6.2 - Estrutura modular da ferramenta *Neural Mining*

O primeiro nível contém o módulo central, que é a tela principal da ferramenta. A tela principal, que fica visível durante toda execução, possui uma barra de menus que permite o acesso aos demais módulos da ferramenta.

O segundo nível foi estruturado de acordo com os modelos e técnicas selecionados para serem investigados nesta dissertação. Neste nível, os módulos servem apenas como menu de opções para as operações inerentes a cada modelo ou técnica. Estas operações estão localizadas no terceiro nível.

A finalidade de cada um destes módulos é apresentada com detalhes na próxima seção.

6.4. Interface

Esta seção apresenta uma breve descrição das telas da ferramenta *Neural Mining*.

6.4.1. Tela Principal

A tela principal da ferramenta é apresentada na Figura 6.3. Através da barra de menus presente nesta tela, o usuário tem acesso às operações específicas dos modelos e técnicas implementados na ferramenta.

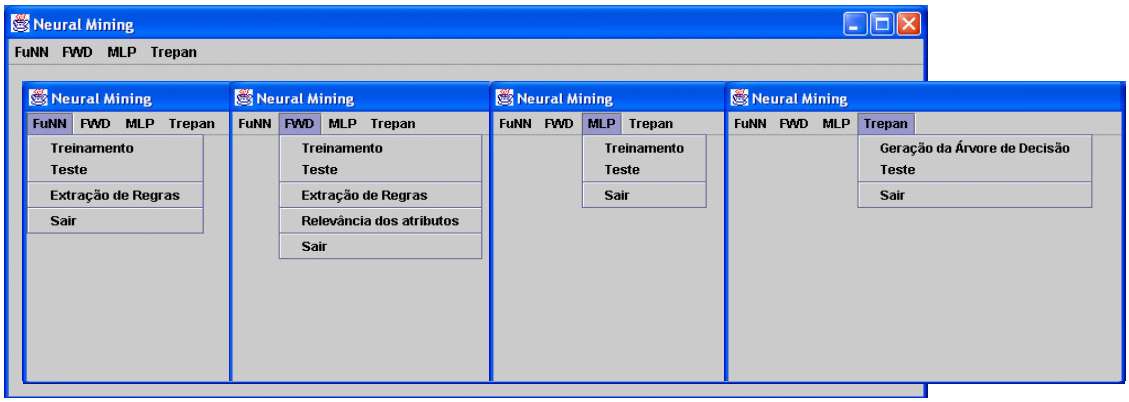

Figura 6.3 - Tela principal e menus de opções

6.4.2. FuNN

Os dados relacionados ao treinamento do modelo FuNN foram divididos em dois grupos: arquitetura da rede e parâmetros de treinamento. A definição da arquitetura é realizada a partir da tela apresentada na Figura 6.4. Esta definição pode ser feita de duas formas: a partir de arquivo texto ou preenchendo os componentes gráficos. No caso da primeira opção, os componentes gráficos são automaticamente preenchidos.

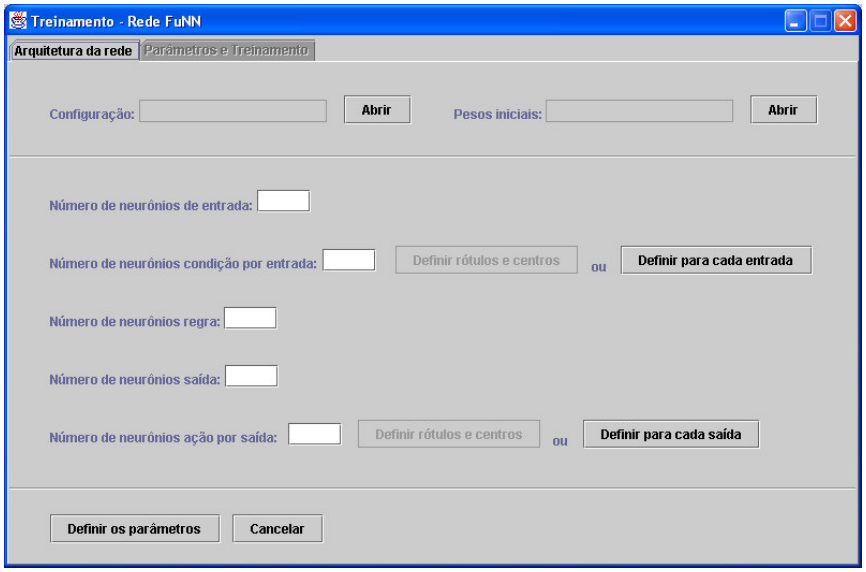

Figura 6.4 - Primeira tela de treinamento do modelo FuNN

A partir da tela de definição da arquitetura da rede, o usuário pode especificar a descrição dos atributos de entrada e saída, e os rótulos *fuzzy* (termos lingüísticos) e centros das funções de pertinência associadas aos atributos. Caso o número de nós condição seja igual para todos os nós de entrada e o número de nós ação seja igual para todos os nós de saída, o usuário pode definir, em uma única tela (Figura 6.5), a descrição dos atributos e os rótulos *fuzzy* e centros das funções de pertinência. Caso os números de nós condição e ação sejam diferentes, o usuário tem que definir, primeiramente, a descrição e o número de funções de pertinência para cada atributo (Figura 6.6) e, posteriormente, os rótulos *fuzzy* e centros das funções de pertinência (Figura 6.7).

Figura 6.5 - Tela de definição dos atributos e funções de pertinência do modelo FuNN

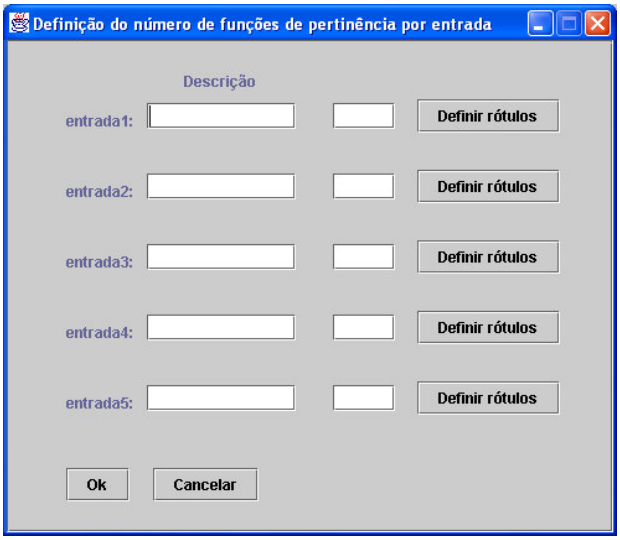

Figura 6.6 - Tela de definição do número de funções de pertinência e descrição dos atributos do modelo FuNN

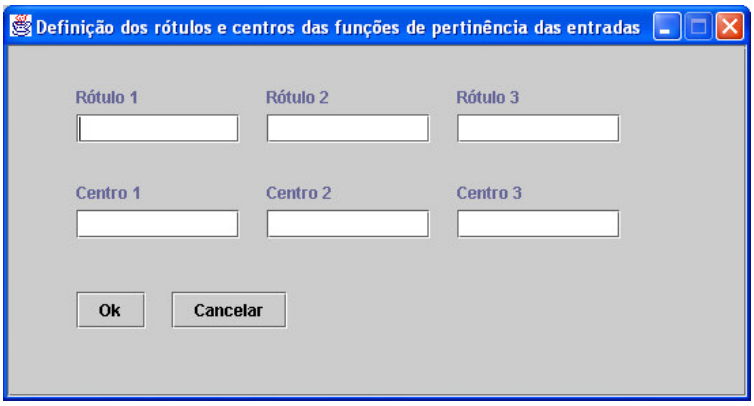

Figura 6.7 - Tela de definição das funções de pertinência do modelo FuNN

Após a definição da arquitetura da rede, os parâmetros de treinamento podem ser especificados a partir da tela apresentada na Figura 6.8.

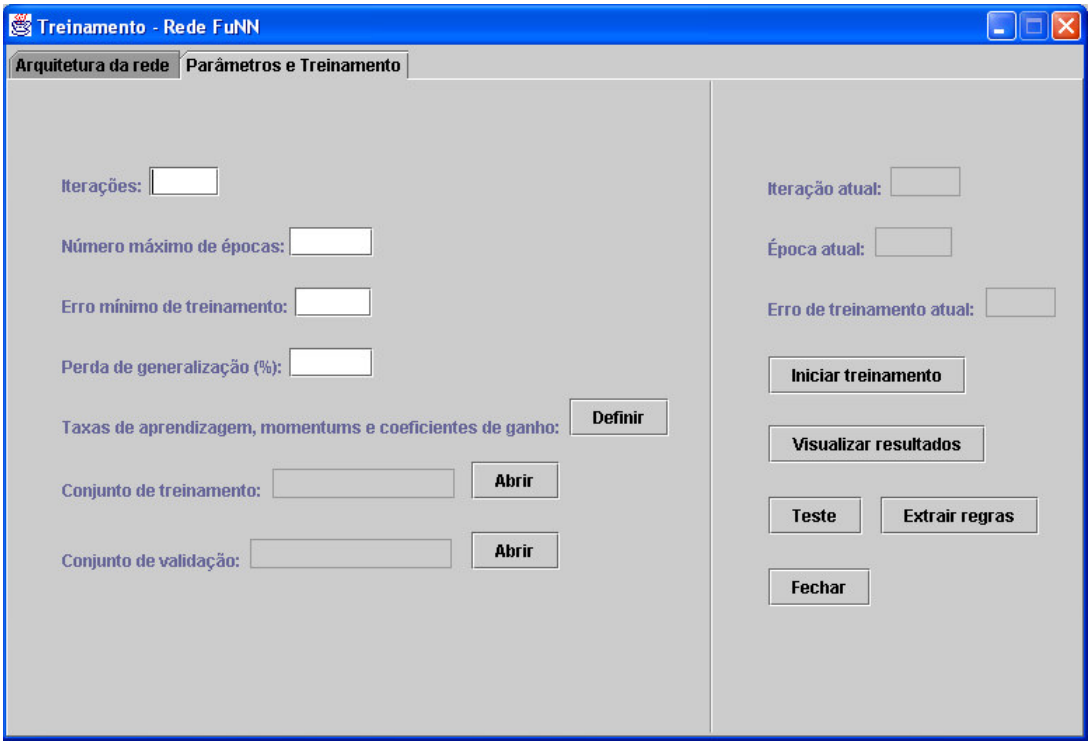

Figura 6.8 - Segunda tela de treinamento do modelo FuNN

Os coeficientes de ganho dos nós regra e ação, as taxas de aprendizagem e os termos *momentum* dos nós condição, regra, ação e saída são definidos separadamente (Figura 6.9). Se o usuário definiu a arquitetura da rede a partir de um arquivo texto, os componentes gráficos desta tela são automaticamente preenchidos.

Figura 6.9 - Tela de definição dos parâmetros de treinamento do modelo FuNN

Após a fase de treinamento, o usuário pode visualizar o resultado da classificação para os conjuntos de treinamento e validação (Figura 6.10), aplicar o modelo treinado no conjunto de teste (Figura 6.11) e extrair regras (Figura 6.12) do modelo resultante. Na tela de extração de regras (Figura 6.12), o usuário deve definir os limiares para as técnicas AREFuNN e REFuNN.

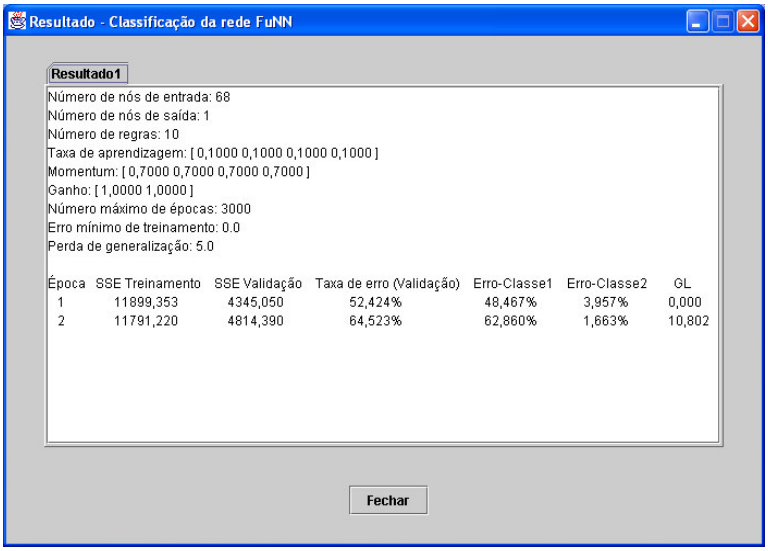

Figura 6.10 - Tela dos resultados da fase de treinamento do modelo FuNN

Figura 6.11 - Tela de teste direto dos modelos FuNN e FWD

Figura 6.12 - Tela de extração de regras direta do modelo FuNN

O teste do modelo gerado (Figura 6.13) e a extração de regras (Figura 6.14) também podem ser realizados a partir das opções da barra de menu referentes ao modelo FuNN. Nestes casos, o usuário deve primeiramente selecionar os arquivos de definição e pesos da rede. A extração de regras também pode ser solicitada a partir da tela de teste do modelo (Figura 6.13). Neste caso, é exibida a tela de extração de regras direta (Figura 6.12).

Figura 6.13 - Tela de teste dos modelos FuNN e FWD

Figura 6.14 - Tela de extração de regras do modelo FuNN

Os resultados do teste (Figura 6.15) e as regras extraídas pelas técnicas REFuNN e AREFuNN (Figura 6.16) podem ser visualizados pelo usuário.

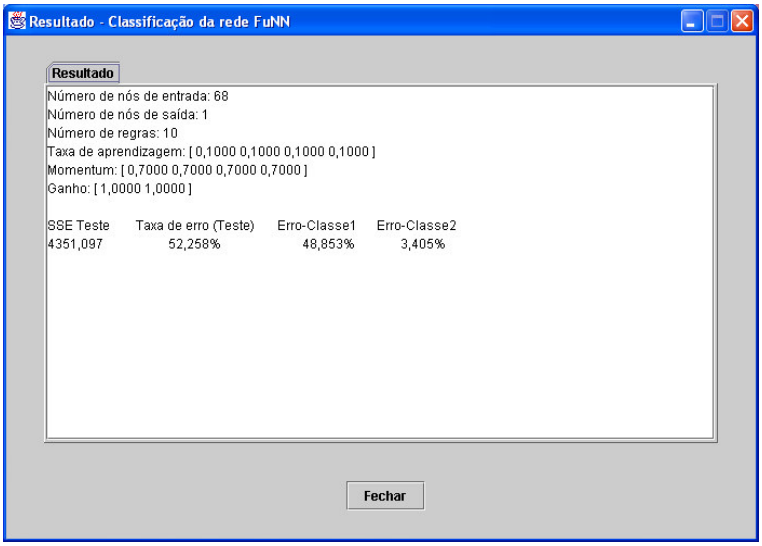

Figura 6.15 - Tela dos resultados do teste do modelo FuNN

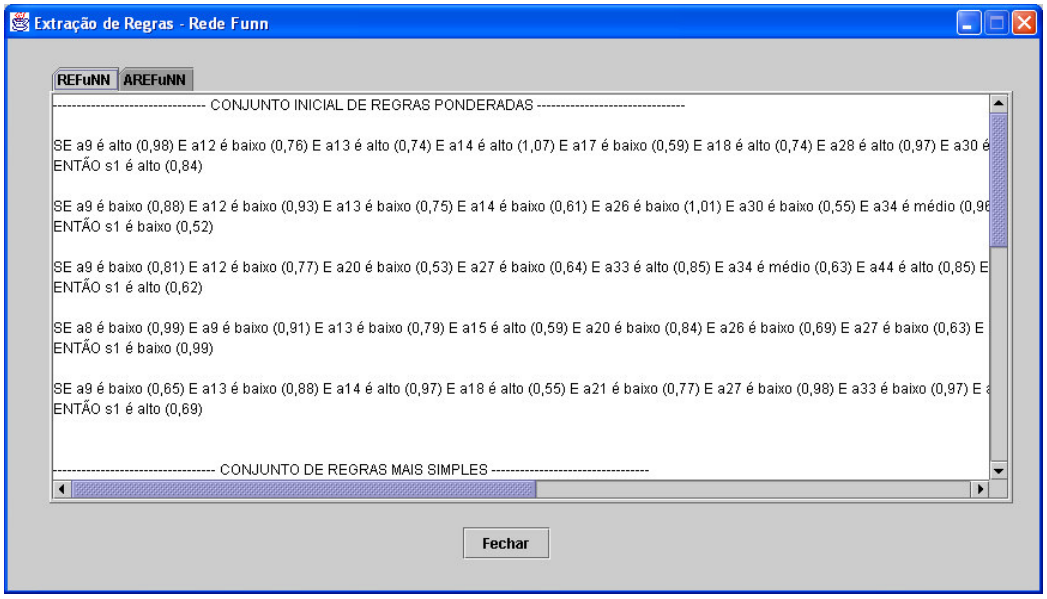

Figura 6.16 - Tela dos resultados da extração de regras do modelo FuNN

Além de exibir a configuração e os pesos de uma rede FuNN, os resultados de classificação das fases de treinamento e teste, e as regras extraídas, a ferramenta *Neural Mining* fornece a opção de salvar estas informações em arquivos cujos formatos estão descritos no Apêndice B.

6.4.3. FWD

Os dados relacionados ao treinamento do modelo FWD foram divididos em dois grupos: arquitetura da rede e parâmetros de treinamento. A definição da arquitetura é realizada a partir da tela apresentada na Figura 6.17. Esta definição pode ser feita de duas formas: a partir de arquivo texto ou preenchendo os componentes gráficos. No caso da primeira opção, os componentes gráficos são automaticamente preenchidos.

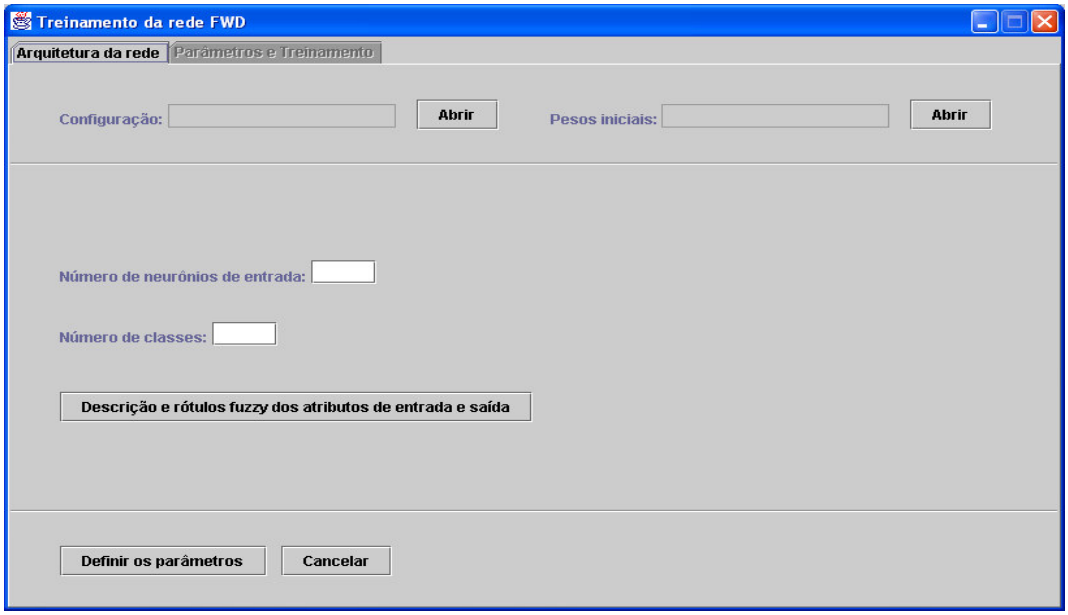

Figura 6.17 - Primeira tela de treinamento do modelo FWD

A partir da tela de definição da arquitetura da rede, o usuário pode especificar (Figura 6.18) a descrição dos atributos de entrada e saída, e os rótulos *fuzzy* das funções de pertinência associadas aos atributos de entrada.

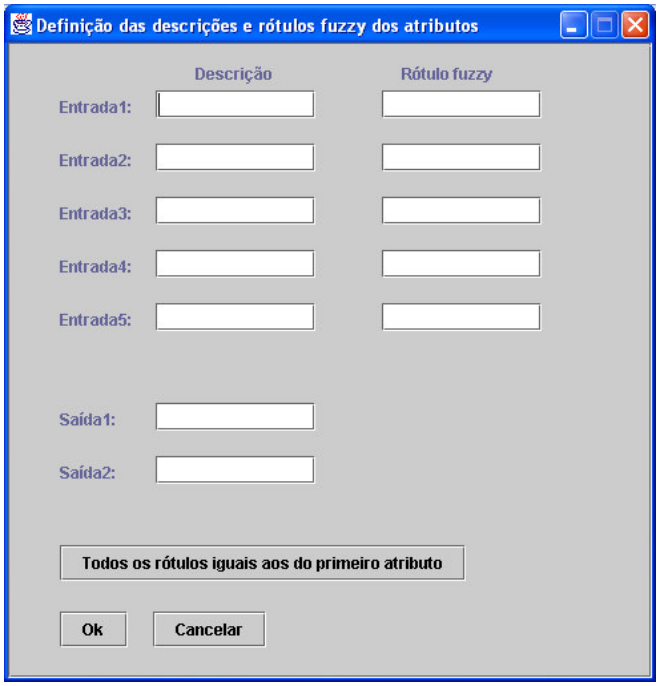

Figura 6.18 - Tela de definição dos rótulos *fuzzy* e descrição dos atributos do modelo FWD

Após a definição da arquitetura da rede, os parâmetros de treinamento podem ser especificados a partir da tela apresentada na Figura 6.19.

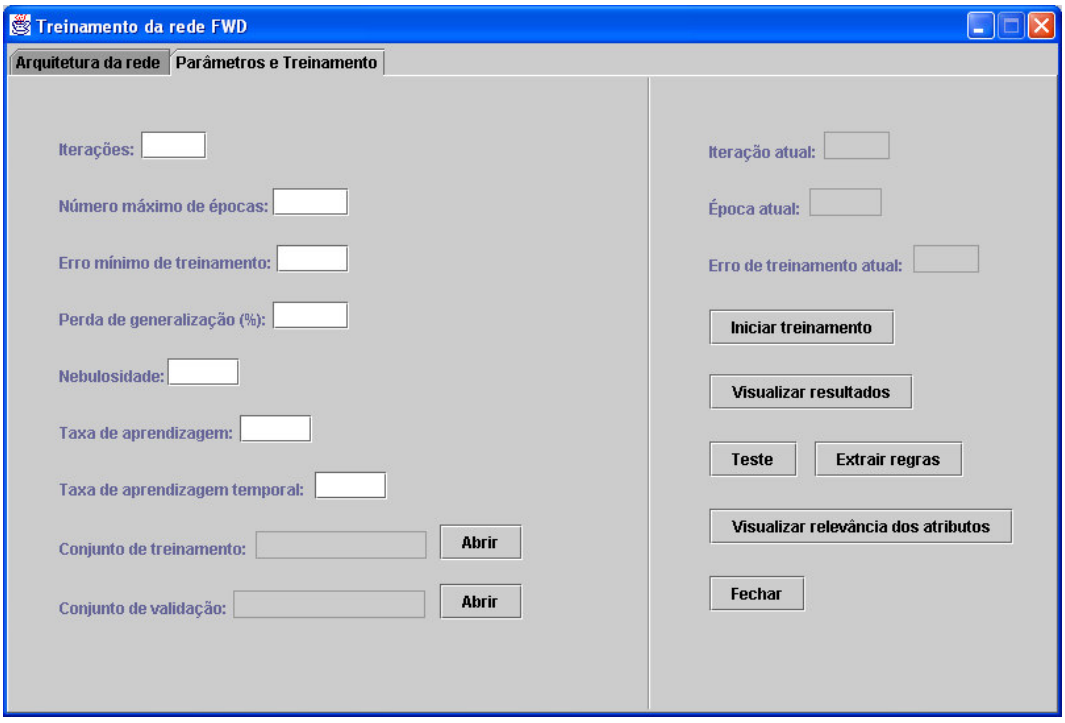

Figura 6.19 - Segunda tela de treinamento do modelo FWD

Quando a fase de treinamento é finalizada, o usuário pode visualizar o resultado da classificação para os conjuntos de treinamento e validação (Figura 6.20), aplicar o modelo treinado no conjunto de teste (Figura 6.11), visualizar as regras extraídas (Figura 6.21) do modelo resultante e observar a relevância dos atributos de entrada com relação às classes (Figura 6.22).

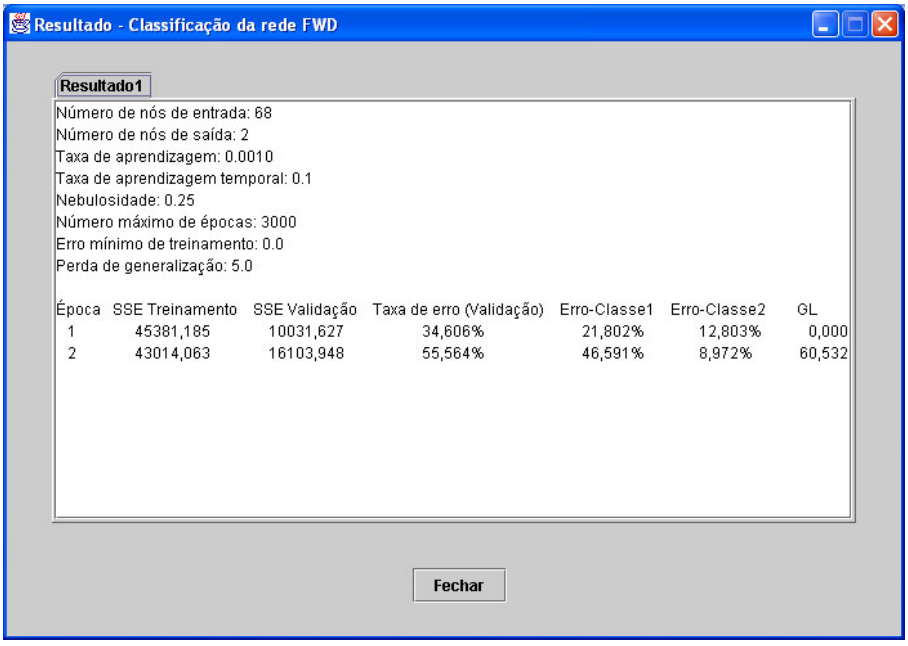

Figura 6.20 - Tela dos resultados da fase de treinamento do modelo FWD

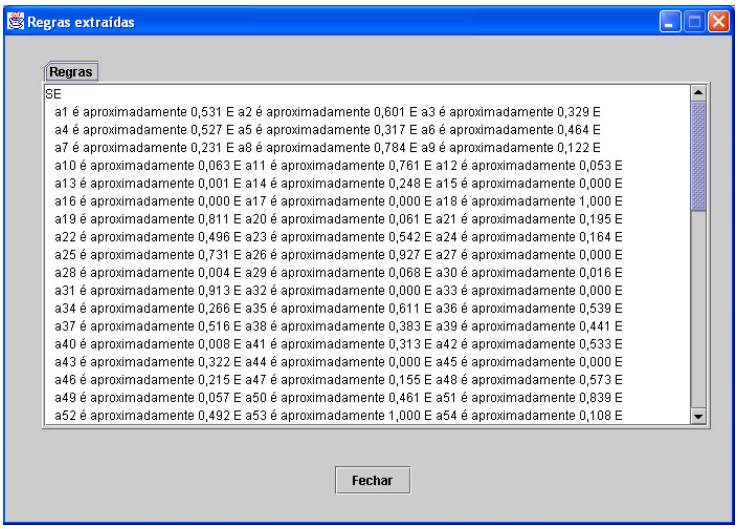

Figura 6.21 - Tela do resultado da extração de regras do modelo FWD

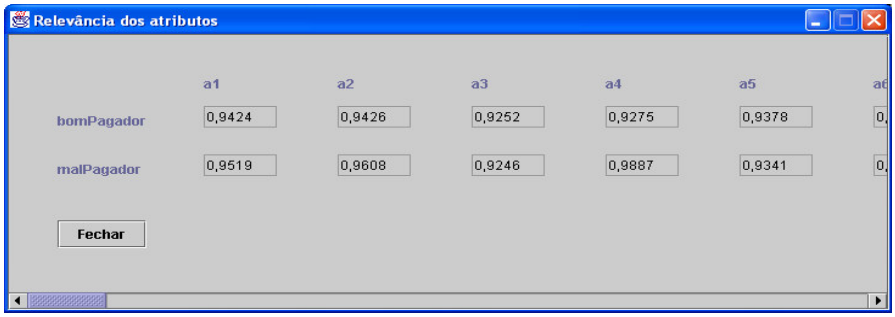

Figura 6.22 - Tela dos resultados da relevância dos atributos

O teste do modelo gerado (Figura 6.13), a extração de regras (Figura 6.23) e a análise da relevância dos atributos de entrada com relação às classes (Figura 6.24) também podem ser realizados a partir das opções da barra de menu referentes ao modelo FWD. Nestes casos, o usuário deve primeiramente selecionar os arquivos de definição e pesos da rede. A extração de regras também pode ser solicitada a partir da tela de teste do modelo (Figura 6.13).

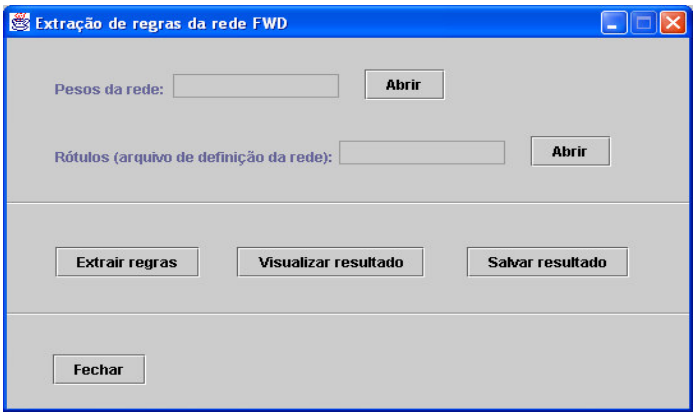

Figura 6.23 - Tela de extração de regras do modelo FWD

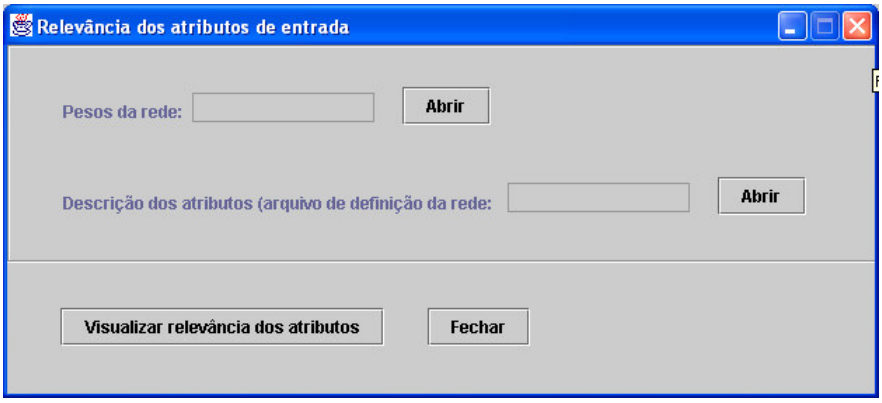

Figura 6.24 - Tela da relevância dos atributos do modelo FWD

As regras extraídas (Figura 6.21) e os resultados do teste (Figura 6.25) podem ser visualizados pelo usuário.

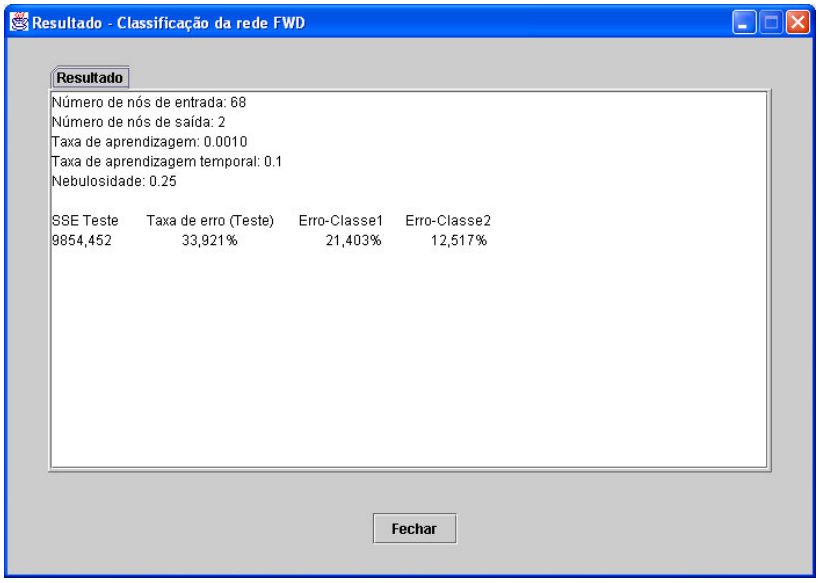

Figura 6.25 - Tela dos resultados do teste do modelo FWD

Além de exibir a configuração e os pesos de uma rede FWD, os resultados de classificação das fases de treinamento e teste, as regras extraídas e a relevância dos atributos, *Neural Mining* oferece a opção de salvar estas informações em arquivos cujos formatos estão descritos no Apêndice B.

6.4.4. MLP

Os dados relacionados ao treinamento do modelo MLP foram divididos em dois grupos: arquitetura da rede e parâmetros de treinamento. A definição da arquitetura é realizada a partir da tela apresentada na Figura 6.26. Esta definição pode ser feita de duas formas: a partir de arquivo texto ou preenchendo os componentes gráficos. No caso da primeira opção, os componentes gráficos são automaticamente preenchidos.

Figura 6.26 - Primeira tela de treinamento do modelo MLP

Após a definição da arquitetura da rede, os parâmetros de treinamento podem ser especificados a partir da tela apresentada na Figura 6.27.

Figura 6.27 - Segunda tela de treinamento do modelo MLP

Os coeficientes de ganho, taxas de aprendizagem e termos *momentum* dos nós intermediários e de saída são definidos separadamente (Figura 6.28). Se o usuário definiu a arquitetura da rede a partir de um arquivo texto, os componentes gráficos desta tela são automaticamente preenchidos.

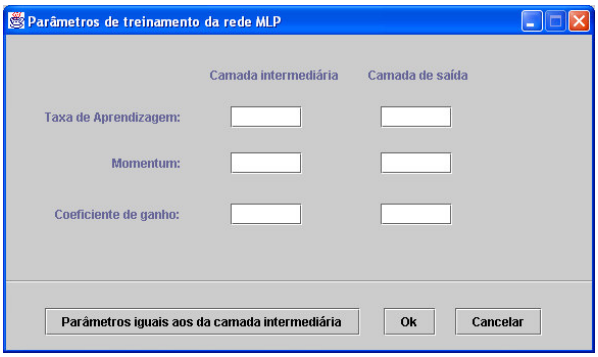

Figura 6.28 - Tela de definição dos parâmetros de treinamento do modelo MLP

Após a fase de treinamento, o usuário pode visualizar o resultado da classificação para os conjuntos de treinamento e validação (Figura 6.29), realizar o teste do modelo gerado (Figura 6.30) e extrair árvores de decisão do modelo (Figura 6.31). Na tela de extração de árvores de decisão, o usuário deve especificar os parâmetros da técnica TREPAN. Os conjuntos de treinamento e teste são novamente selecionados porque os valores dos atributos utilizados por TREPAN diferem dos valores utilizados pela rede MLP. Portanto, as bases de dados são diferentes.

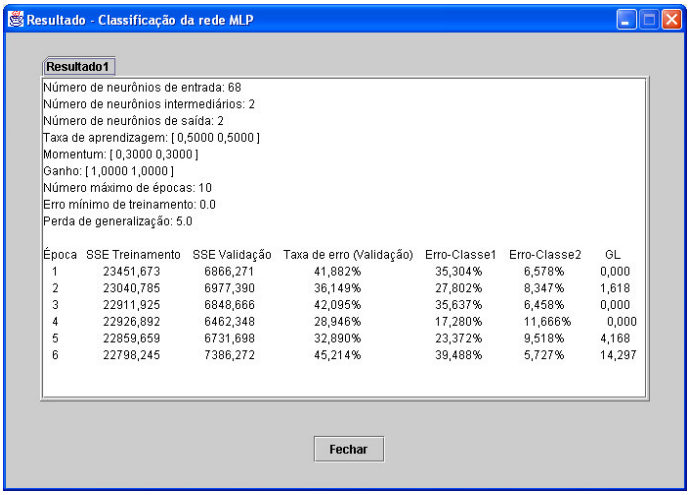

Figura 6.29 - Tela dos resultados da fase de treinamento do modelo MLP

Figura 6.30 - Tela de teste direto do modelo MLP

Figura 6.31 - Tela de extração direta de árvores de decisão do modelo MLP

O teste do modelo (Figura 6.32) e a extração de árvores de decisão (Figuras 6.36 e 6.37) também podem ser realizados a partir das opções da barra de menu referentes ao modelo MLP e à técnica TREPAN. Nestes casos, o usuário deve primeiramente selecionar os arquivos de definição e pesos da rede. A extração da árvore também pode ser solicitada a partir da tela de teste (Figura 6.32).

Figura 6.32 - Tela de teste do modelo MLP

Os resultados do teste realizado pela rede MLP (Figura 6.33), os resultados da classificação realizada pela árvore de decisão extraída pela técnica TREPAN (Figura 6.34) e a própria árvore de decisão (Figura 6.35) podem ser visualizados pelo usuário.

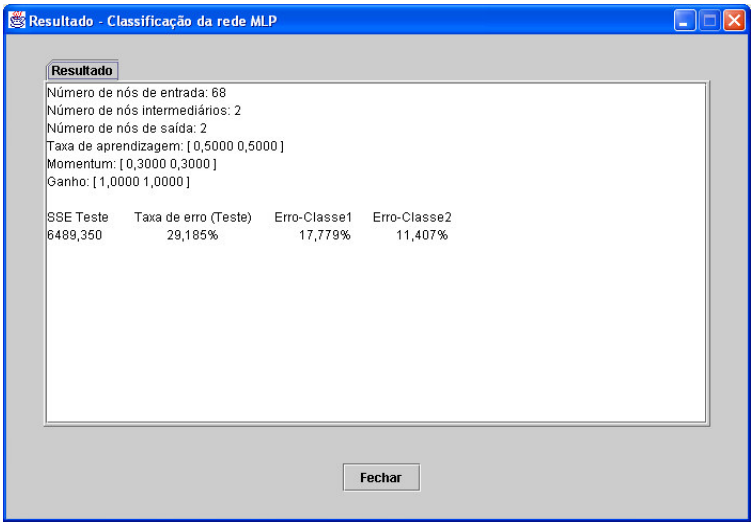

Figura 6.33 - Tela dos resultados do teste do modelo MLP

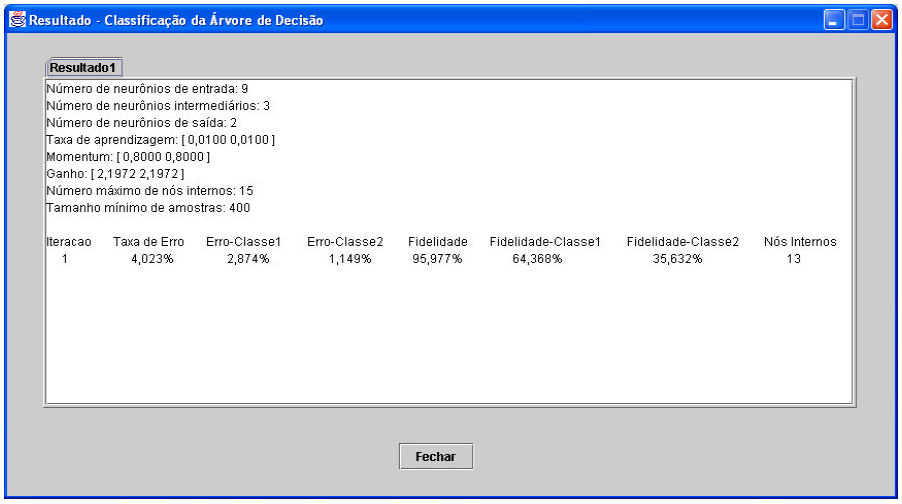

Figura 6.34 - Tela dos resultados da classificação realizada pela árvore de decisão

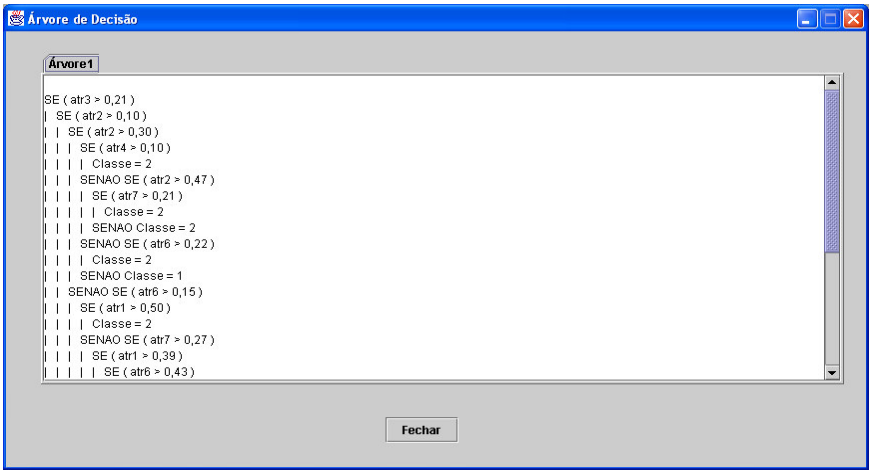

Figura 6.35 - Tela do resultado da extração da árvore de decisão

Além de exibir a configuração e os pesos de uma rede MLP, os resultados de classificação das fases de treinamento e teste da rede, as árvores de decisão extraídas e os resultados da classificação realizada pelas árvores de decisão, a ferramenta *Neural Mining* fornece a opção de salvar estas informações em arquivos cujos formatos estão descritos no Apêndice B.

6.4.5. TREPAN

Os dados relacionados à extração de árvores de decisão a partir de uma rede MLP foram divididos em dois grupos: definição da rede MLP e parâmetros da técnica TREPAN. A definição da rede MLP é realizada a partir da tela apresentada na Figura 6.36. Esta definição pode ser feita de duas formas: a partir de arquivos texto ou preenchendo os componentes gráficos. No caso da primeira opção, os componentes gráficos são automaticamente preenchidos.

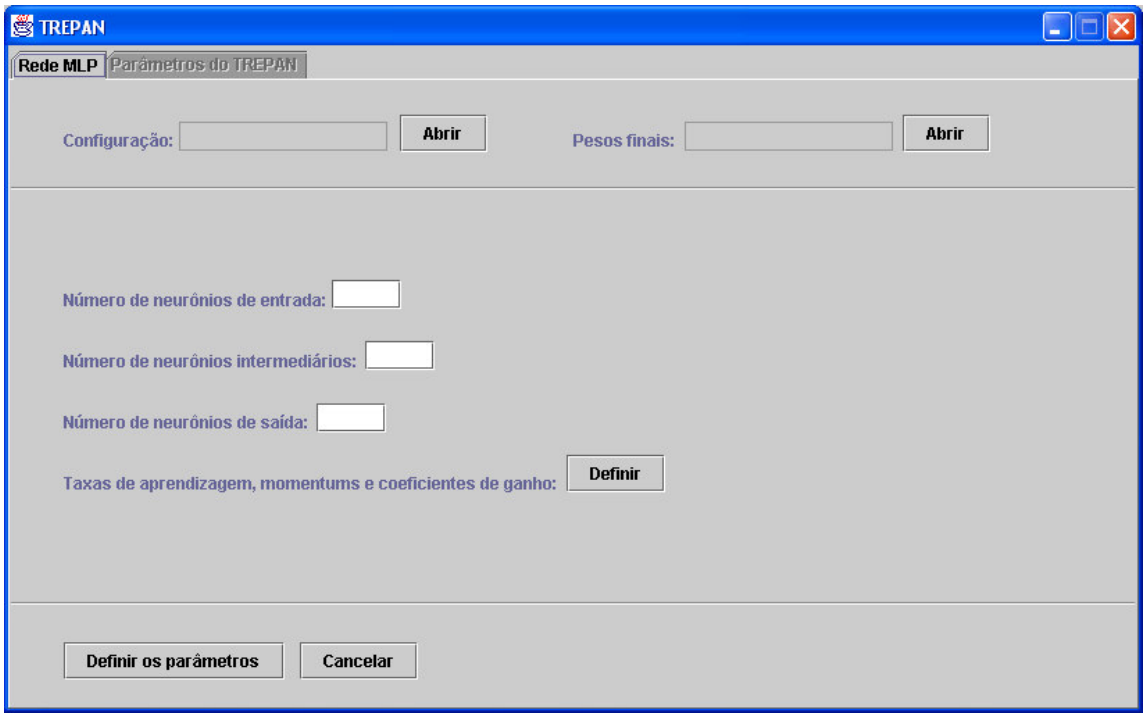

Figura 6.36 - Tela de definição da rede MLP usada pela técnica TREPAN

Os coeficientes de ganho, taxas de aprendizagem e termos *momentum* dos nós intermediários e de saída da rede MLP são definidos separadamente (Figura 6.28). Se o usuário definiu a rede a partir de um arquivo texto, os componentes gráficos desta tela são automaticamente preenchidos.

Após a definição da rede, especificam-se os parâmetros da técnica TREPAN através da tela apresentada na Figura 6.37. A especificação das características (descrição, tipo e valores) dos atributos de entrada e saída é feita nas telas das Figuras 6.38 e 6.39. Na tela da Figura 6.38, definem-se a descrição, o tipo (nominal ou numérico) e, no caso dos atributos nominais, o total de possíveis valores. Em seguida, na tela da Figura 6.39, definem-se os possíveis valores dos atributos nominais e os valores codificados correspondentes que serão utilizados pela rede MLP, que é capaz de manipular apenas dados numéricos.

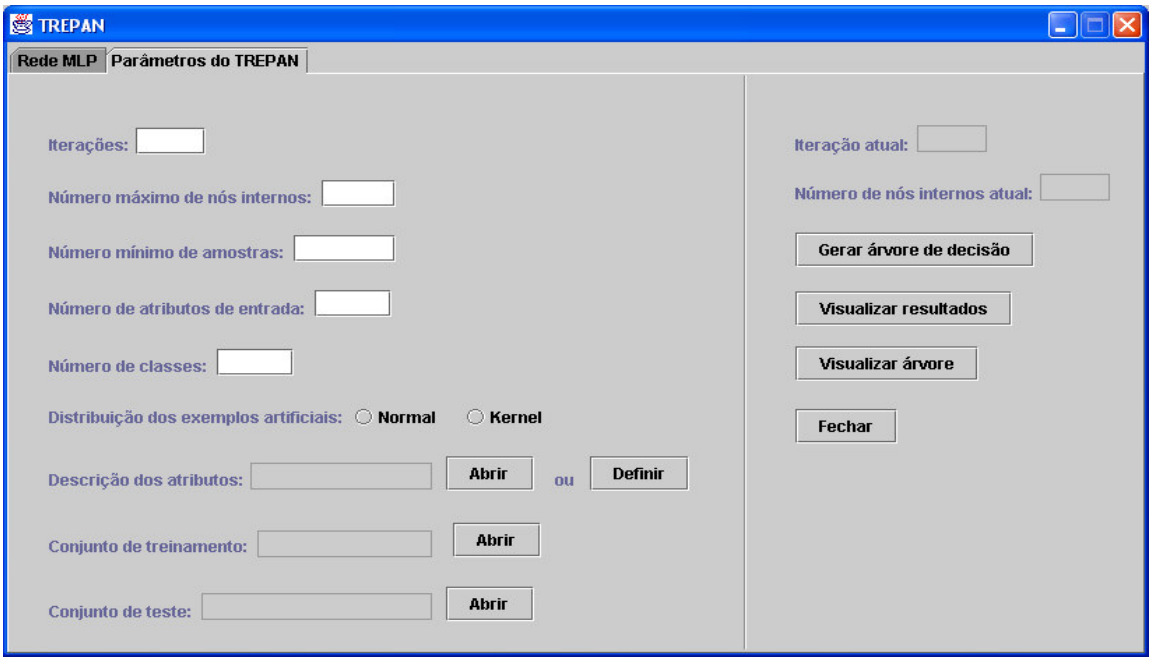

Figura 6.37 - Tela de definição dos parâmetros da técnica TREPAN

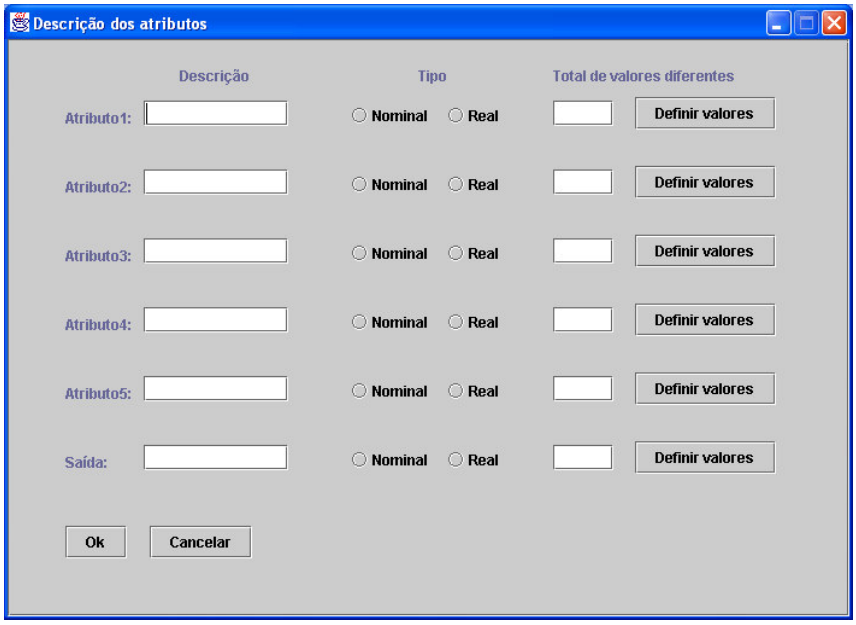

Figura 6.38 - Tela de descrição dos atributos da técnica TREPAN

Ao contrário do que ocorre quando a técnica TREPAN é executada a partir de algum módulo da rede MLP, os conjuntos de treinamento e teste são selecionados apenas uma vez. Neste caso, a diferença entre as bases de dados são implementadas de forma automática pela ferramenta *Neural Mining*, utilizando os valores especificados na tela da Figura 6.39. Desta forma, se um atributo nominal *X* pode assumir os valores *A* (valor codificado *0*) e *B* (valor codificado *1*), antes de um exemplo, cujo atributo *X* assume o valor *A*, ser apresentado à rede MLP, o valor *A* é convertido para *0*.

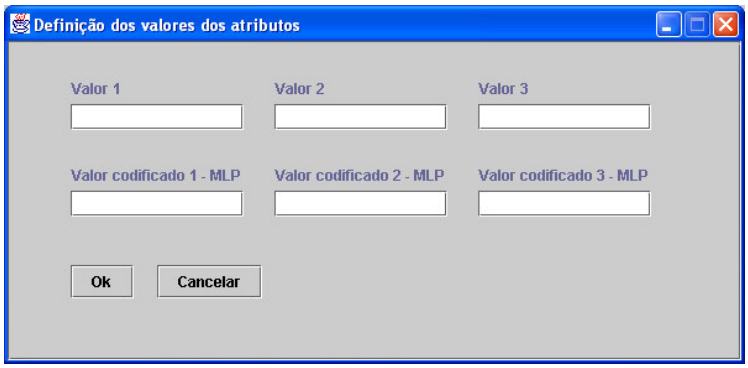

Figura 6.39 - Tela de definição dos valores dos atributos nominais da técnica TREPAN

Após a geração da árvore de decisão, o usuário pode visualizar o resultado da classificação realizada pela árvore no conjunto de teste (Figura 6.34) e a própria árvore (Figura 6.35).

O teste realizado pela árvore de decisão também pode ser solicitado a partir da opção da barra de menu referente à técnica TREPAN (Figura 6.40). Neste caso, o usuário deve primeiramente selecionar os arquivos de definição e pesos da rede, de descrição dos atributos e da estrutura da árvore de decisão gerada anteriormente.

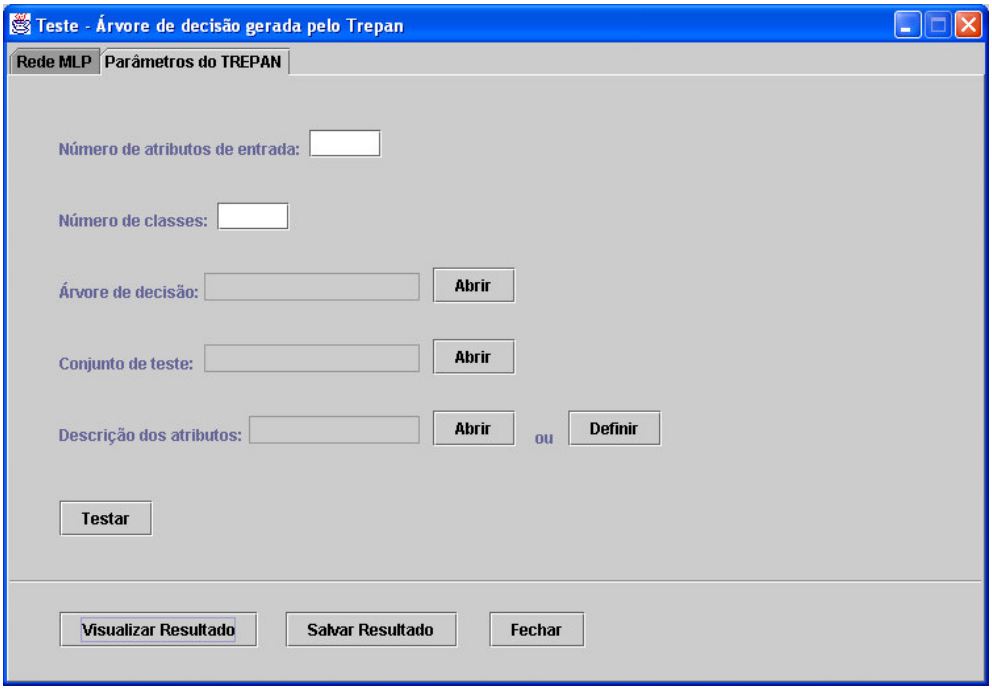

Figura 6.40 - Tela de definição dos parâmetros da fase de teste da técnica TREPAN

Além de exibir as árvores de decisão extraídas, a descrição dos atributos, a definição da rede MLP e os resultados da classificação realizada pelas árvores, a ferramenta *Neural Mining* fornece a opção de salvar estas informações em arquivos cujos formatos estão descritos no Apêndice B.

6.5. Considerações Finais

Embora KDD seja uma área recente com muitos aspectos que ainda precisam ser pesquisados em profundidade, uma grande quantidade de ferramentas de *software* de KDD e mineração de dados já foi desenvolvida [Han & Kamber 2001]. Inicialmente, as ferramentas eram mais utilizadas em ambientes de pesquisa e experimentais. Nos últimos anos, tem-se observado um rápido crescimento de ferramentas sofisticadas voltadas para o meio empresarial [Goebel & Gruenwald 1999]. O domínio de aplicação destas ferramentas está constantemente evoluindo devido ao desenvolvimento de novos sistemas, incorporação de novas funcionalidades aos sistemas existentes e proposição de novos métodos de mineração de dados. Apesar dessa evolução, poucas ferramentas têm adotado a abordagem dos Sistemas Neurais Híbridos [Goebel & Gruenwald 1999] [Data 2003] [Data 2004].

Este capítulo descreveu uma ferramenta baseada na abordagem neural híbrida para descoberta de conhecimento em bases de dados desenvolvida nesta dissertação, a *Neural Mining*. As principais vantagens desta ferramenta são a capacidade de integrar vários métodos de aprendizagem neural híbrida num único ambiente e ter sido projetada para solução de problemas de larga escala. Além disso, *Neural Mining* apresenta uma interface amigável e, por ter sido desenvolvida em Java, possui todas as vantagens de um sistema orientado a objetos, tais como compatibilidade, reusabilidade e fácil manutenção.

Capítulo 7

Estudo de Caso: Análise de Risco de Crédito

7.1. Introdução

Este capítulo apresenta os resultados dos experimentos realizados com a rede MLP; o modelo neuro-*fuzzy* FWD e sua técnica de extração de regras; o modelo neuro-*fuzzy* FuNN e suas técnicas de extração de regras REFuNN e AREFuNN; e a técnica TREPAN, usada em conjunto com a rede MLP da qual uma árvore de decisão é extraída. Os modelos são avaliados e comparados considerando os seguintes aspectos: desempenho em relação à generalização e capacidade de gerar conhecimento compreensível através de suas técnicas de extração de regras. O desempenho dos classificadores em relação à generalização é avaliado observando as taxas de classificação no conjunto de teste (total e por classe), analisando as curvas ROC (*Receiver Operating Characteristics*) [Provost & Fawcett 1997] e o impacto das decisões dos classificadores no contexto específico da aplicação investigada (análise de crédito ao consumidor). A compreensibilidade do conhecimento extraído é avaliada analisando a facilidade de interpretação e aplicação do conhecimento descoberto. Outro aspecto considerado na avaliação do conhecimento extraído é a precisão. Além da análise nas etapas de mineração de dados e apresentação do conhecimento, também são investigadas duas técnicas de seleção de atributos: a técnica da rede FWD e através da árvore de decisão gerada por TREPAN. Ao final da investigação, considerando as vantagens de cada modelo e técnica, são propostas duas soluções neurais híbridas para o processo de KDD.

A investigação experimental abrange todas as etapas do processo de KDD. As etapas de mineração de dados e apresentação do conhecimento foram executadas na ferramenta *Neural Mining*, desenvolvida nesta dissertação. O domínio investigado como estudo de caso foi o de análise de risco de crédito, um problema de classificação que define a aprovação ou não de crédito a um determinado cliente, considerando suas características pessoais e financeiras. Este domínio foi escolhido por se tratar de um problema de larga escala, envolver dados reais com múltiplos atributos relacionados e ser de interesse de diversas instituições e empresas. Tais características tornam o problema mais complexo e permitem a verificação da viabilidade prática dos modelos e técnicas investigados.

7.2. Análise de Risco de Crédito

A crescente automação de atividades comerciais e bancárias tem requerido a utilização de tecnologias cada vez mais sofisticadas para satisfazer os usuários. Como resultado, o mercado financeiro está buscando a utilização de novas técnicas computacionais, entre elas as técnicas inteligentes [Lacerda et al. 2003].

RNA têm sido aplicadas com sucesso no domínio de análise de crédito [Widrow et al. 1994] [Mendes Filho et al. 1997] [Sousa & Carvalho 1999] [Vasconcelos et al. 1999] e em outros problemas na área financeira, tais como detecção de fraude [Abbott et al. 1998] e previsão de séries financeiras [Gately 1996]. O sucesso da aplicação de RNA no setor financeiro é conseqüência da capacidade das mesmas em aprender funções complexas e não-lineares. Tal característica é uma vantagem sobre as técnicas estatísticas convencionais (Ex.: análise discriminante) [Lacerda et al. 2003].

O problema de análise de crédito envolve um conjunto de atividades realizadas por um credor para a concessão de crédito a um solicitante, considerando alguns aspectos como informações econômicas e pessoais do mesmo. Além de considerar as informações relacionadas aos clientes, o analista de crédito também se baseia em sua experiência e na política de crédito adotada pela instituição a que pertence. Por serem realizadas em um cenário de incertezas e constantes mutações, onde as decisões devem ser tomadas rapidamente, o risco associado às concessões de crédito é altíssimo, pois se as análises forem realizadas e gerenciadas erroneamente, podem provocar não apenas prejuízos financeiros à instituição de crédito, mas também prejuízos financeiros e morais aos clientes [Lacerda et al. 2003]. Porém, se bem administradas, podem gerar muito lucro e manter as dívidas dentro de um patamar aceitável.

Na maioria das situações de concessão de crédito existem basicamente três etapas a serem realizadas [Monteiro 1999]:

- Análise retrospectiva: Avalia-se o desempenho histórico do solicitante para identificar fatores na condição atual do solicitante que possam levar ao não cumprimento de seus deveres;
- Análise de tendências: Realiza-se uma projeção da condição financeira do solicitante associada à análise de sua capacidade de suportar certo nível de endividamento;
- Capacidade creditícia: Baseando-se nas análises anteriores, decide-se pela concessão ou não de crédito e, em caso positivo, o valor permitido. O objetivo é obter a máxima proteção da empresa contra eventuais perdas. Após a concessão do crédito, a empresa acompanha as transações financeiras do cliente e, de acordo com um critério definido, o rotula como adimplente ou inadimplente.

O problema de análise de crédito envolve muitos atributos, normalmente relacionados de forma complexa. Por esta razão, durante muito tempo, os especialistas da área achavam que o bom senso ou *feeling* do analista jamais poderia ser substituído ou auxiliado por computador [Schrickel 1995]. No entanto, nos últimos anos, tem-se observado o uso crescente do computador no processo de concessão de crédito [Blatt 1999]. Mais especificamente, diversas empresas têm investido bastante na área de Mineração de Dados para análise do perfil dos clientes. Estas empresas têm obtido um bom retorno dos investimentos e altos índices de lucro devido à aplicação do conhecimento minerado [Ramos 2001].

Nos casos onde o processo de concessão depende primordialmente do elemento humano, os seguintes problemas podem surgir: a experiência de alguns analistas é limitada ou desatualizada, os analistas possuem critérios de decisão diferentes e o processo de decisão exige bastante tempo. Estes problemas podem ser minimizados através da utilização de sistemas automatizados [Monteiro 1999]. Tais sistemas apresentam os seguintes benefícios: rapidez na aprovação de crédito, redução do custo de processamento do empréstimo, maior flexibilidade de adaptação a novas situações, consistência entre decisões de diferentes analistas e maior segurança, pois, confirmam empiricamente a decisão do analista [Carter & Catlett 1987] [Jensen 1992].

Apesar da eficácia comprovada das soluções automatizadas para análise de crédito, estas funcionam como ferramentas de apoio à decisão, ficando, desta forma, a decisão final para o analista. Na verdade, normalmente o analista se preocupa apenas com as propostas mais complexas, nas quais o sistema não chegou a uma decisão dentro de uma margem de confiabilidade [Monteiro 1999].

7.3. Trabalhos Relacionados

Muitos estudos têm destacado o uso de Sistemas Neuro-*Fuzzy* e técnicas de extração de regras de RNA em problemas de análise de crédito.

[Piramuthu 1999] analisou os benefícios dos Sistemas Neuro-*Fuzzy* e Neurais. [Conde 2000] realizou uma análise comparativa entre os modelos neurais MLP e RBF e os modelos neuro-*fuzzy* MLP-*Fuzzy*, RBF-*Fuzzy* e ARTMAP-*Fuzzy*. [Ramos 2001] abordou os algoritmos de *fuzzy clustering* FCM (*Fuzzy C-Means*), GK (*Gustafson-Kessel*), GG (*Gath-Geva*) e suas versões simplificadas, e algumas técnicas de extração de regras. [Malhotra & Malhotra 2002] comparou a performance do modelo ANFIS com a dos modelos de análise discriminante múltipla.

Esses trabalhos concluíram que os Sistemas Neuro-*Fuzzy* e os modelos neurais integrados a técnicas de extração de conhecimento simbólico são bastante atrativos para serem usados em aplicações do mundo real, podendo ser considerados alternativos às RNA e técnicas estatísticas tradicionais.

7.4. Processo de KDD

Esta seção descreve a realização de todas as etapas do processo de KDD, aplicadas em um estudo de caso de análise de crédito, utilizando a ferramenta *Neural Mining* nas etapas de mineração e apresentação do conhecimento.

7.4.1. Identificação do Problema

Quando uma pessoa ou empresa solicita um crédito, existem várias funções objetivo que podem ser consideradas no processo de tomada de decisões. Portanto, as instituições financeiras podem definir diferentes funções de custo a serem otimizadas pelo sistema de suporte à decisão. Estas instituições podem, por exemplo, estar interessadas em maximizar seus lucros, minimizar seus riscos ou os atrasos nos pagamentos. No entanto, a principal meta da instituição é definir quando deverá ou não conceder crédito a um solicitante [Schrickel 1995]. O estudo de caso desta dissertação se concentra neste aspecto do problema.

A base de dados utilizada no estudo de caso possui os dados fornecidos pelos solicitantes no momento da solicitação de crédito a uma operadora de cartões de crédito no Brasil. Essas informações são utilizadas pela empresa para decidir pela concessão ou não de crédito ao solicitante. Os dados de todos os clientes que obtiveram a aprovação do crédito foram armazenados nesta base. Com o passar do tempo, alguns desses clientes, que foram considerados bons pagadores pelo sistema decisório da operadora, se tornaram maus pagadores, como mostra a Figura 7.1 [Ramos 2000]. Portanto, o problema aplicado ao estudo de caso contém informações parciais, pois a base de dados possui apenas informações a respeito dos proponentes aceitos e que vieram a se tornar adimplentes ou inadimplentes na carteira de clientes da empresa. Tal característica torna o problema ainda mais complexo.

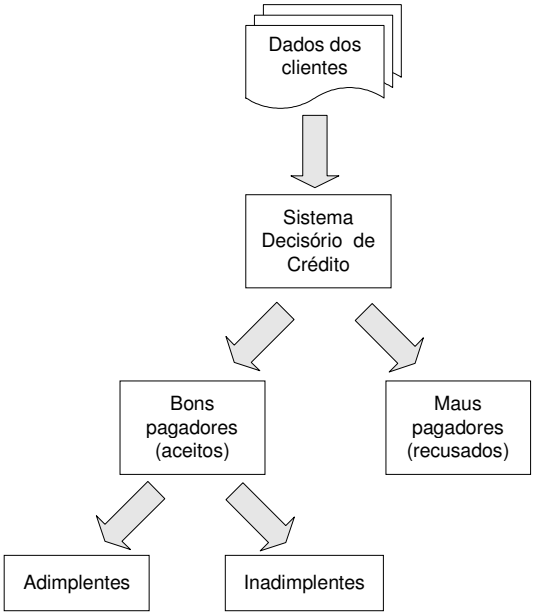

Figura 7.1 - Processo decisório de análise de crédito

Os dados foram obtidos de uma única base e estavam armazenados em um arquivo texto. Cada linha do arquivo representa um exemplo (registro) e cada coluna representa um atributo. Cada exemplo se refere a uma solicitação real para obtenção de crédito e à classificação do cliente (adimplente ou inadimplente) realizada pela empresa.

A base de dados consiste de 60.141 registros, dos quais 11.923 (19,83%) são classificados, pelos critérios da empresa, como inadimplentes e 48.218 (80,17%) como adimplentes. Cada registro é composto de 39 atributos (Tabela 7.1), incluindo o atributo alvo (classificação do cliente). Os atributos são nominais ou numéricos.

Após a seleção dos atributos, a base passou a ser composta de 28 atributos, incluindo o atributo alvo. Os atributos selecionados são especificados na Tabela 7.2.

Tabela 7.1 - Atributos da base de dados original

CODIGO	FLAG_NOME_MAE
NUMERO_LOJA	FLAG_NOME_PAI
SEXO	FLAG_MESMA_CIDADE_RESIDENCIA_COMERCIAL
ESTADO_CIVIL	FLAG_MESMA_UF_RESIDENCIA_COMERCIAL
IDADE	TEMPO_EMPREGO
NUMERO_DE_DEPENDENTES	CODIGO_PROFISSAO
ESCOLARIDADE	RENDA_CONJUGE
UF_RESIDENCIAL	FLAG_ENDERECO_CORRESP_IGUAL_RESIDENCIAL
CIDADE_RESIDENCIAL	FLAG_TEM_OUTRO_CARTAO
BAIRRO_RESIDENCIAL	NUM_CONTAS_BANCO
FLAG_TEL_RESIDENCIAL	REFERENCIA_PESSOAL_1
DDD_RESIDENCIAL	REFERENCIA_PESSOAL_2
CEP_DIGITO1	FLAG_TELEFONE_CELULAR
CEP_DIGITO2	FLAG_TELEFONE_RECADO
CEP_DIGITO3	RENDA_LIQUIDA
CEP_DIGITO4	NUM_SEQ_PONTO_CAPTACAO
DIA_VENCIMENTO1	NUM_CARTOES_ADICIONAIS
TIPO_CLIENTE	FLAG_OPCAO_SEGURO_CADASTRO
TIPO_RESIDENCIA	SITUACAO_MAU
TEMPO_RESIDENCIA	

Tabela 7.2 - Atributos selecionados

NUMERO_LOJA	TIPO_RESIDENCIA
SEXO	TEMPO_RESIDENCIA
ESTADO_CIVIL	FLAG_NOME_PAI
IDADE	FLAG_MESMA_CIDADE_RESIDENCIA_COMERCIAL
UF_RESIDENCIAL	FLAG_MESMA_UF_RESIDENCIA_COMERCIAL
CIDADE_RESIDENCIAL	TEMPO_EMPREGO
FLAG_TEL_RESIDENCIAL	CODIGO_PROFISSAO
DDD_RESIDENCIAL	RENDA_CONJUGE
CEP_DIGITO1	FLAG_ENDERECO_CORRESP_IGUAL_RESIDENCIAL
CEP_DIGITO2	REFERENCIA_PESSOAL_2
CEP_DIGITO3	RENDA_LIQUIDA
CEP_DIGITO4	NUM_SEQ_PONTO_CAPTACAO
DIA_VENCIMENTO1	NUM_CARTOES_ADICIONAIS
TIPO_CLIENTE	SITUACAO_MAU

7.4.2. Integração, Seleção, Limpeza e Transformação dos Dados

A base de dados obtida já se encontrava pré-processada, ou seja, as etapas de integração, seleção, limpeza e transformação dos dados já tinham sido realizadas, considerando que a base seria aplicada em modelos neurais.

Os atributos nominais foram mapeados em vetores binários, aplicando as codificações binárias *1 de N* (Tabela 7.3) ou *M de N* (Tabela 7.4

Tabela 7.4) [Bigus 1996] [Monteiro 1999]. A codificação *1 de N* foi aplicada para os atributos cujo conjunto de valores possíveis é pequeno. Por outro lado, a codificação *M de N* foi aplicada para os atributos cujo conjunto de valores possíveis é extenso. Esta estratégia de codificação permite uma redução da dimensionalidade da base de dados codificada. Após a fase de codificação, os atributos passaram a ser denominados da seguinte forma: *nome_do_atributo + n*, onde *n* varia de 1 até o tamanho do vetor binário resultante da codificação.

Tabela 7.3 - Codificação *1 de N*

ESTADO_CIVIL (N = 5)	TIPO_RESIDENCIA (N = 4)	DDD_RESIDENCIAL (N = 6)
UF_RESIDENCIAL (N = 4)	SITUACAO_MAU (N = 2)	

Tabela 7.4 - Codificação *M de N*

LOJA_NUMERO_LOJA (M = 3 e N = 7)	CODIGO_PROFISSAO (M = 4 e N = 9)
CIDADE_RESIDENCIAL (M = 3 e N = 7)	

Os demais atributos nominais também tiveram seus valores mapeados em valores binários. A Tabela 7.5 mostra a codificação adotada para cada um deles.

Tabela 7.5 - Codificação binária

Atributo	**Valores binários**
SEXO	0, 1
FLAG_TEL_RESIDENCIAL	0, 1
TIPO_CLIENTE	00, 01, 10
FLAG_NOME_PAI	0, 1
FLAG_MESMA_CIDADE_RESIDENCIA_COMERCIAL	0, 1
FLAG_MESMA_UF_RESIDENCIA_COMERCIAL	0, 1
FLAG_ENDERECO_CORRESP_IGUAL_RESIDENCIAL	0, 1
REFERENCIA_PESSOAL_2	0, 1

Os atributos numéricos (Tabela 7.6) foram normalizados de forma a ficarem dentro do intervalo [0, 1], usando a Equação 2.1.

Tabela 7.6 - Atributos numéricos

IDADE	CEP_DIGITO4	RENDA_CONJUGE
CEP_DIGITO1	DIA_VENCIMENTO1	RENDA_LIQUIDA
CEP_DIGITO2	TEMPO_RESIDENCIA	NUM_CARTOES_ADICIONAIS
CEP_DIGITO3	TEMPO_EMPREGO	

Um atributo foi gerado a partir do atributo *DIA_VENCIMENTO1*. O novo atributo, *DIA_VENCIMENTO2*, pode assumir quatorze valores diferentes. A codificação *M de N* (*M = 2, N = 6*) foi usada neste atributo.

Para todos os atributos de entrada, uma análise estatística foi realizada a fim de agrupar todos os casos raros e, portanto, de baixa representatividade, em um único valor codificado.

Após o pré-processamento, a base de dados passou a possuir 68 atributos de entrada e 2 atributos de saída (*10* representando a classe adimplente e *01* representando a classe inadimplente).

Como a base obtida já estava pré-processada para ser aplicada em modelos neurais, antes de executar os experimentos com a técnica TREPAN, foi necessário desfazer a codificação binária. A base de dados resultante passou a ter 27 atributos de entrada e 1 atributo de saída (classe). Como os valores originais dos atributos não são conhecidos, os valores numéricos continuaram normalizados e cada vetor binário foi substituído por um número. Considerando como exemplo o atributo *TIPO_CLIENTE*, que na base codificada está representado pelos atributos *TIPO_CLIENTE1* e *TIPO_CLIENTE2*, seus possíveis valores (*00*, *01* e *10*) foram mapeados para *1*, *2* e *3*, respectivamente. No momento em que os exemplos utilizados pela técnica TREPAN são apresentados à rede MLP, a codificação binária original é refeita automaticamente pela ferramenta *Neural Mining*.

Particionamento dos Dados

Como descrito na Seção 2.2.6, existem vários métodos para estimar uma medida verdadeira de um algoritmo de aprendizagem. A escolha de um método implica na estratégia adotada no particionamento dos dados.

Apesar do método *stratified 10-fold cross-validation* ser bastante recomendado [Witten & Frank 2000], este não foi aplicado ao estudo de caso porque a base de dados é bastante extensa, tornando inviável, em termos de tempo de processamento, a realização de uma investigação experimental detalhada. Devido à extensão da base, optou-se por aplicar o método *Holdout* [Beale & Jackson 1991], que também é sugerido pela comunidade científica para investigação experimental de modelos neurais [Prechelt 1994]. Portanto, os padrões foram divididos em três conjuntos: treinamento, com 50% dos dados; validação, com 25% dos dados; e teste, com os 25% restantes. A Tabela 7.7 mostra o total de registros e a distribuição das classes em cada conjunto.

Tabela 7.7 - Particionamento dos dados

Conjunto de dados	Total de registros	Adimplente	Inadimplente
Treinamento	30.071	24.108	5.963
Validação	15.035	12.055	2.980
Teste	15.035	12.055	2.980

Para evitar que uma classe estivesse ausente em um dos conjuntos de dados, os conjuntos foram particionados para cada classe separadamente e depois foram integrados, mantendo, desta forma, a distribuição das classes da base de dados original. Como resultado, todos os conjuntos de dados possuem 19,83% de clientes da classe inadimplente e 80,17% da classe adimplente.

A diferença acentuada entre a quantidade de exemplos de classes diferentes pode prejudicar o processo de aprendizagem. A repetição de exemplos de treinamento das classes que possuem uma proporção menor e a geração de exemplos ruidosos são algumas técnicas que podem ser usadas para minimizar este tipo de problema [Bigus 1996]. Neste estudo de caso, optou-se por repetir os registros da classe inadimplente várias vezes no conjunto de treinamento até igualar à quantidade de registros da classe adimplente [Conde 2000]. Desta forma, o conjunto de treinamento passou a ter 48.216 registros.

A Figuras 7.2, 7.3 e 7.4 ilustram todas as etapas do particionamento dos dados.

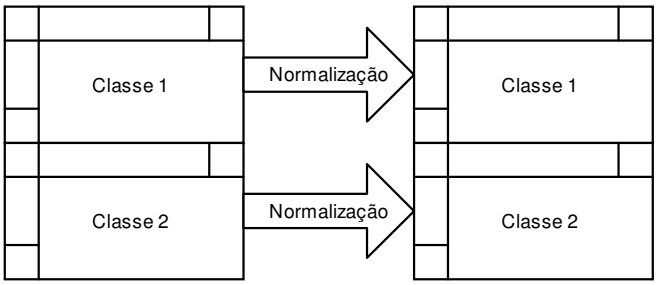

Figura 7.2 - Primeira etapa do particionamento dos dados

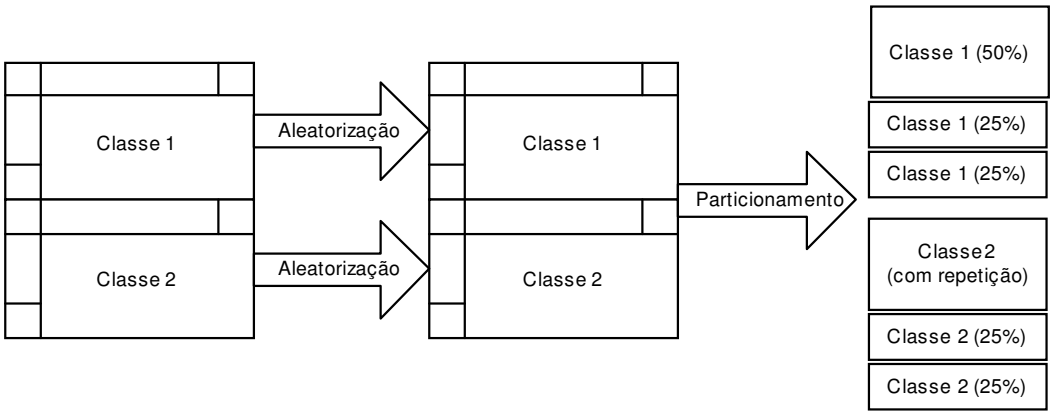

Figura 7.3 - Segunda etapa do particionamento dos dados

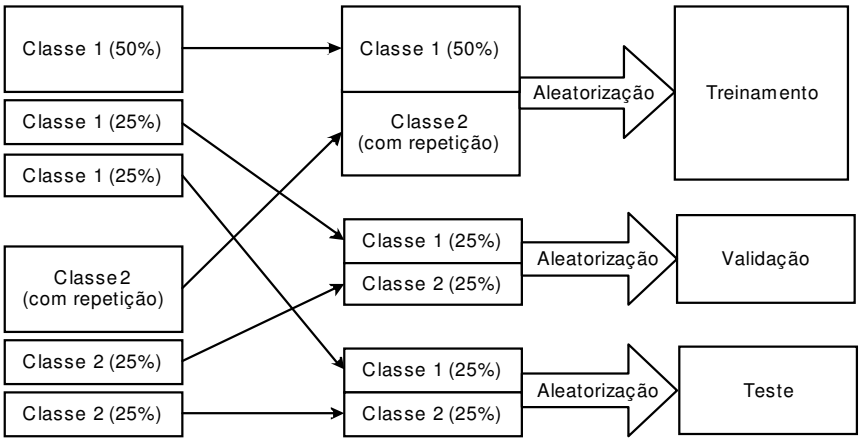

Figura 7.4 - Terceira etapa do particionamento dos dados

7.4.3. Mineração de Dados

O objeto desta etapa é investigar os modelos MLP, FuNN, FWD e a técnica TREPAN para a tarefa de classificação de um cliente como adimplente ou inadimplente.

Modelos Neurais

Metodologia Experimental

Os experimentos realizados com os modelos MLP, FuNN e FWD foram executados obedecendo aos seguintes critérios e metodologia:

- Utilizou-se o modo de treinamento seqüencial;
- Os critérios de parada utilizados foram o número máximo de épocas (3000) e a perda de generalização (5%). O critério de perda de generalização (GL – *Generalization Loss*), definido pela Equação 7.1, encerra o treinamento quando o erro de validação aumenta 5% com relação ao menor erro até a época atual [Prechelt 1994]. Este critério diminui as chances de ocorrer

overfitting e fornece uma visão da capacidade de generalização da rede, pois o seu valor é baseado no conjunto de validação;

$$GL(época\ t) = 100 * \left(\frac{SSE\ atual\ de\ validação}{SSE\ mínimo\ de\ validação} - 1 \right) \quad (7.1)$$

- Foi definido um conjunto de valores para cada parâmetro ajustável dos modelos. Todas as combinações possíveis destes valores resultaram em um conjunto de configurações a serem investigadas;
- Como os valores iniciais dos pesos afetam a precisão dos modelos neurais, a fim de obter uma comparação coerente das diversas configurações, para cada arquitetura, foi utilizado o mesmo conjunto de pesos iniciais;
- O critério utilizado para escolher a melhor configuração de cada modelo foi o SSE (*Sum of Squared Error*) de validação. O custo associado aos tipos de erro não foi considerado como critério de escolha porque as redes escolhidas segundo este critério apresentam taxas de classificação no conjunto de validação inferior a 50%;
- Após a definição da melhor configuração, foram realizadas 30 iterações com pesos iniciais diferentes. O objetivo desta fase é calcular a média e o desvio padrão referentes às taxas de classificação nos conjuntos de validação e teste. Devido à grande diferença entre as distribuições das classes, os resultados obtidos sempre são registrados no total e separadamente para cada classe. Desta forma, é possível obter uma melhor visão do desempenho da rede sobre cada uma das classes;
- Depois desta fase, investigou-se a técnica de seleção de atributos do modelo FWD e a seleção de atributos utilizando a árvore de decisão gerada por TREPAN (apenas os atributos que aparecem na árvore são considerados relevantes). Os atributos considerados irrelevantes foram removidos da base e os modelos foram re-treinados com os mesmos parâmetros de treinamento, a fim de validar a eficácia das técnicas. 30 iterações com pesos iniciais diferentes foram executadas para cada modelo. Em seguida, foram calculados a média e o desvio padrão referentes às taxas de classificação no conjunto de teste;
- As técnicas de extração de regras dos modelos FuNN e FWD foram aplicadas após o treinamento destes modelos com a base de dados resultante do processo de seleção de atributos do modelo FWD.

Parametrização

Os valores selecionados para os parâmetros ajustáveis dos modelos neurais foram bastante diversificados.

Além dos valores usuais aplicados às taxas de aprendizagem, também foram analisados valores muito pequenos. Esta heurística foi sugerida em [Conde 2000]. Os valores dos termos *momentum* e nebulosidade foram escolhidos de tal forma que valores pequenos, medianos e grandes, dentro do intervalo de valores possíveis ([0,1]), fossem analisados. Com relação ao coeficiente de ganho da função de ativação, o valor 1 foi escolhido por resultar na função sigmóide padrão e o valor 2,19722 por ter sido sugerido em [Kasabov 1996].

Nos experimentos com a rede MLP foram utilizados os parâmetros especificados na Tabela 7.8.

Tabela 7.8 - Parâmetros da rede MLP

Parâmetros	Valores
Taxa de aprendizagem	0,1; 0,5; 0,01; 0,05; 0,001; 0,005; 0,0001; 0,0005
Termo *momentum*	0,1; 0,3; 0,5; 0,7; 0,9
Coeficiente de ganho	1; 2,19722
Nós intermediários	2, 4, 8, 16, 32

Na rede MLP foram utilizados 68 nós de entrada, que correspondem aos atributos de entrada da base de dados codificada, e 2 nós de saída, que correspondem às classes. Portanto, o tipo de codificação de saída adotada para as predições da rede MLP foi a codificação *1-de-n* (ou *winner takes all*). Existem duas motivações para escolher esta codificação: fornece maior grau de liberdade para a rede representar a função alvo e a diferença entre a saída de valor mais alto e a saída com segundo maior valor pode ser usada como uma medida de confiança da predição da rede [Mitchell 1997]. Nesta codificação, a saída representada pelo nó cujo valor é o mais alto é considerada a predição da rede. O número de nós intermediários foi escolhido de forma empírica, pois apesar de diversas técnicas terem sido propostas para definição da quantidade destes nós, isto ainda é um problema em aberto [Azevedo et al. 2000].

A função de ativação dos nós intermediários e de saída foi a sigmóide logística, definida pela Equação 3.8. Com relação à inicialização dos pesos, todos os pesos foram inicializados com valores aleatórios dentro do intervalo [-1,1].

Nos experimentos realizados com a rede FWD foram utilizados os parâmetros da Tabela 7.9.

Tabela 7.9 - Parâmetros da rede FWD

Parâmetros	Valores
Taxa de aprendizagem	0,1; 0,5; 0,01; 0,05; 0,001; 0,005; 0,0001; 0,0005
Taxa de aprendizagem temporal	0,1; 0,5; 0,01; 0,05; 0,001; 0,005
Nebulosidade	0,1; 0,25; 0,4; 0,55; 0,7; 0,85

No modelo FWD foram utilizados 68 nós de entrada, que correspondem aos atributos de entrada da base de dados codificada, e 2 nós de saída, que correspondem às classes. Portanto, o tipo de codificação de saída adotada para as predições da rede FWD foi a codificação *winner takes all*.

Com relação à inicialização dos pesos, as conexões de memória foram inicializadas com valores aleatórios (Tabela 4.1), considerando o conjunto de treinamento, e as conexões peso foram inicializadas com 1.

Os experimentos com a rede FuNN foram realizados utilizando os parâmetros da Tabela 7.10.

Tabela 7.10 - Parâmetros da rede FuNN

Parâmetros	Valores
Taxa de aprendizagem	0,1; 0,5; 0,01; 0,05; 0,001; 0,005; 0,0001; 0,0005
Termo *momentum*	0,1; 0,3; 0,5; 0,7; 0,9
Coeficiente de ganho	1; 2,19722
Nós regra	10, 30, 50, 70

No modelo FuNN foram utilizados 68 nós de entrada, que correspondem aos atributos de entrada da base de dados codificada. Os nós de entrada que representam atributos numéricos são associados a três nós condição (*pequeno*, *médio* e *grande*) e os nós que representam atributos booleanos são associados a dois nós condição (*verdadeiro* e *falso*).

Diferente da rede MLP e FWD, a rede FuNN, possui apenas um nó de saída. Neste modelo, as classes adimplente e inadimplente são representadas pelos dois nós da camada de ação. Como resultado, o tipo de codificação de saída adotada para as predições da rede FuNN foi a codificação em que o valor de saída do nó é mapeado para as possíveis saídas do problema. Nos experimentos deste estudo de caso, foi utilizado o seguinte critério: se o valor de saída do nó for menor que 0,5, a predição da rede é inadimplente, caso contrário, é adimplente.

O número de nós regra foi definido de forma empírica, pois, inicialmente, não se sabia quantos nós regras eram necessários para a solução do problema.

Com relação à inicialização dos pesos, os pesos entre as camadas de condição e regra, e as camadas de regra e ação foram inicializados com valores aleatórios dentro do intervalo [-1,1]. Os pesos entre as camadas de entrada e condição, que representam os centros das funções de pertinência dos atributos de entrada, foram inicializados com os valores 0, 0,5 e 1 para os atributos numéricos, e 0 e 1 para os atributos booleanos. Os pesos entre as camadas de ação e saída foram inicializados com os valores 0 e 1.

Resultados Obtidos

Na fase experimental, todas as combinações possíveis dos valores dos parâmetros ajustáveis foram investigadas. Ao final da investigação, definiu-se a melhor configuração de cada modelo (menor SSE de validação). As Tabelas 7.11, 7.12 e 7.13 mostram a melhor configuração dos modelos MLP, FWD e FuNN, respectivamente.

Tabela 7.11 - Melhor configuração da rede MLP

Parâmetros	Valores
Taxa de aprendizagem	0,01
Termo *momentum*	0,9
Coeficiente de ganho	1
Nós intermediários	2

Tabela 7.12 - Melhor configuração da rede FWD

Parâmetros	Valores
Taxa de aprendizagem	0,1
Taxa de aprendizagem temporal	0,5
Nebulosidade	0,7

Tabela 7.13 - Melhor configuração da rede FuNN

Parâmetros	Valores
Taxa de aprendizagem	0,001
Termo *momentum*	0,9
Coeficiente de ganho	1
Nós regra	10

Os resultados da classificação do conjunto de teste (erro quadrático médio, taxa de classificação total e taxas de classificação por classe) obtidos pelas melhores configurações dos modelos neurais são exibidos na Tabela 7.14.

Tabela 7.14 - Resultado da classificação do conjunto de teste dos modelos neurais

Modelo	Erro Quadrático Médio	Total	Adimplente	Inadimplente
MLP	0,202	68,22%	72,97%	49,00%
FWD	0,207	67,32%	71,63%	49,90%
FuNN	0,105	67,73%	71,08%	54,19%

Analisando os resultados da Tabela 7.14, pode-se observar que a performance dos modelos neurais no conjunto de teste foi bastante aproximada. A maior diferença ocorreu na taxa de classificação da classe inadimplente, onde o modelo FuNN apresenta um aumento de 5,19% e 4,29% em relação à taxa de classificação dos modelos MLP e FWD, respectivamente.

Segundo [Adriaans & Zantinge 1996] e [Monard & Baranauskas 2003], o desequilíbrio na distribuição das classes (19,83% de inadimplentes e 80,17% de adimplentes) exige que a performance dos modelos apresente um erro máximo inferior a 19,83% (ou taxa de classificação mínima superior a 80,17%). Tal exigência não foi satisfeita neste estudo de caso devido aos seguintes fatores: a base de dados utilizada contém apenas informações parciais (informações a respeito dos proponentes aceitos e que vieram a se tornar adimplentes ou inadimplentes) e o problema investigado é de larga escala e envolve dados reais com múltiplos atributos relacionados. Tais características tornam o problema bastante complexo, dificultando a aprendizagem.

A diferença existente entre as taxas de classificação das classes adimplente e inadimplente pode ser resultado da discrepância do número de exemplos da classe adimplente com relação ao número de exemplos da classe inadimplente. A alternativa adotada para minimizar este problema foi estratificar o conjunto de treinamento. Esta alternativa se baseia na idéia de que ao repetir os exemplos, os mesmos são apresentados à rede com mais freqüência e, conseqüentemente, o erro associado à classe é reduzido [Bigus 1996]. Entretanto, neste estudo de caso, tal solução não foi suficiente para aproximar o aprendizado das duas classes.

A fim de analisar a capacidade de generalização dos modelos neurais na fase de aprendizagem, foram gerados gráficos da variação da GL durante o treinamento (Figura 7.5). A rede MLP iniciou o treinamento num ponto próximo da SSE de validação mínima para a configuração e pesos iniciais estabelecidos. À medida que o treinamento prosseguia, a GL aumentava. O treinamento foi encerrado na época 589, quando a GL atingiu 5,003%. A rede FWD também iniciou o treinamento num ponto próximo da SSE de validação mínima para a configuração e pesos iniciais estabelecidos. Ao contrário da rede MLP, durante toda a fase de treinamento, a GL permaneceu estável, aumentando apenas na última época para 2,899%. O modelo FuNN se mostrou mais instável que os modelos MLP e FWD. Neste modelo, ocorreram inúmeras oscilações no valor da GL. O treinamento da rede FuNN foi interrompido na época 243, quando a GL atingiu 5,137%.

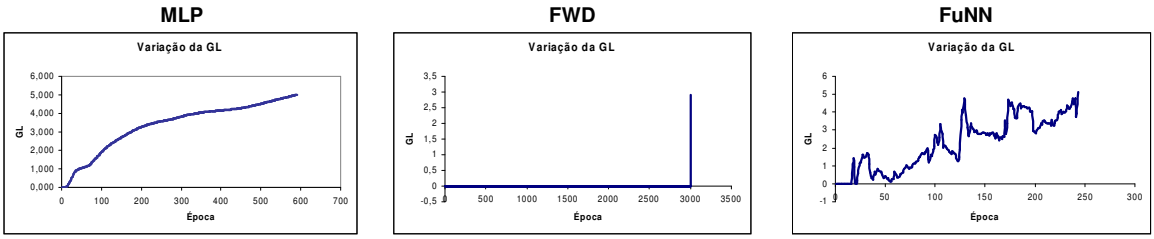

Figura 7.5 - Variação da GL

A Tabela 7.15 mostra os resultados obtidos pelas melhores configurações após a execução de 30 iterações com inicializações de pesos distintas. Esta tabela mostra três tipos de média da taxa de classificação do conjunto de teste (total e por classe) e o desvio padrão para cada tipo de média.

Tabela 7.15 - Resultado da classificação do conjunto de teste dos modelos neurais após 30 iterações

	Médias			Desvio padrão		
Modelo	**Total**	**Adimplente**	**Inadimplente**	**Total**	**Adimplente**	**Inadimplente**
MLP	66,44%	69,41%	54,43%	1,17	2,13	2,79
FWD	67,32%	71,63%	49,90%	0,00	0,00	0,00
FuNN	65,83%	68,36%	55,62%	1,15	2,03	2,83

Como pode ser observado na Tabela 7.15, os modelos neurais apresentaram taxas de classificação no conjunto de teste bastante aproximadas. A maior diferença ocorreu na média da taxa de classificação da classe inadimplente. Neste caso, os modelos MLP e FuNN apresentaram performance superior ao modelo FWD. Outro aspecto interessante é a apresentação de desvio padrão nulo em todas as taxas da rede FWD. Isto ocorreu, pois dependendo da configuração dos parâmetros da rede FWD, esta mantém a taxa de classificação constante em todas as iterações. Esta situação é bastante comum nesta rede, pois as conexões peso são inicializados com 1 e, apesar de existir alguma aleatoriedade na inicialização das conexões de memória, seus valores iniciais consideram os dados do conjunto de treinamento. Desta forma, ao contrário dos modelos MLP e FuNN, existe pouca diferença, de uma iteração para outra, nos conjuntos de pesos iniciais da rede FWD.

Em problemas de classificação é usual medir o desempenho do classificador através da precisão da classificação no conjunto de teste. Conseqüentemente, esta medida é bastante utilizada para avaliar e comparar o desempenho de classificadores. No entanto, comparações baseadas na precisão omitem dois pontos importantes que devem ser considerados, principalmente em problemas reais: a distribuição das classes normalmente não pode ser precisamente especificada, pois a distribuição encontrada no conjunto de treinamento raramente é igual à distribuição em dados novos; e os custos associados aos tipos de erro podem ser diferentes e mudar ao longo do tempo. Portanto, para complementar a análise dos classificadores usados nesta dissertação, faz-se necessária a aplicação de alguma técnica que possa determinar o classificador ótimo, independentemente da distribuição das classes e dos custos associados aos tipos de erro e. A técnica escolhida para esta análise foi as curvas ROC (*Receiver Operating Characteristics*) [Provost & Fawcett 1997].

As curvas ROC mostram a relação das taxas de falsos positivos (FP) e verdadeiros positivos (VP) variando de acordo com um limiar aplicado às saídas dos classificadores. Esta relação prediz o comportamento dos classificadores independentemente da distribuição das classes e dos custos

associados aos tipos de erro. Além da independência com relação ao custo e distribuição das classes, este tipo de gráfico permite uma comparação visual de um conjunto de classificadores.

Numa curva ROC, o eixo das ordenadas (*y*) representa a taxa de VP e o eixo das abscissas (*x*) representa a taxa de FP. As Equações 2.2 e 2.3 mostram como essas taxas são calculadas. O canto inferior esquerdo do gráfico *(0,0)* representa a situação em que todos os casos são reconhecidos como negativos e o canto superior direito *(1,1)* representa a situação em que todos os casos são classificados como positivos. A linha $x = y$ representa a situação em que os casos são aleatoriamente classificados. Um ponto na curva ROC é melhor do que outro quando se encontra mais a noroeste (VP é maior, FP é menor, ou ambos).

A Figura 7.6 mostra as curvas ROC dos modelos neurais investigados nesta dissertação.

Para analisar melhor os resultados das curvas ROC, o gráfico foi dividido em sete regiões (r_1, r_2, ..., r_7). Nas região r_1 e r_7, as curvas dos modelos estão sobrepostas, portanto apresentam desempenho semelhante. Nas regiões r_2, r_4 e r_6, os modelos MLP e FuNN são melhores do que o modelo FWD, pois a posição de suas curvas estão mais a noroeste. Na região r_3, o modelo MLP é melhor do que os modelos FuNN e FWD. Na região r_5, o modelo FuNN apresenta desempenho melhor que os modelos MLP e FuNN. Em nenhuma das regiões, o modelo FWD apresentou desempenho melhor que os modelos MLP e FuNN.

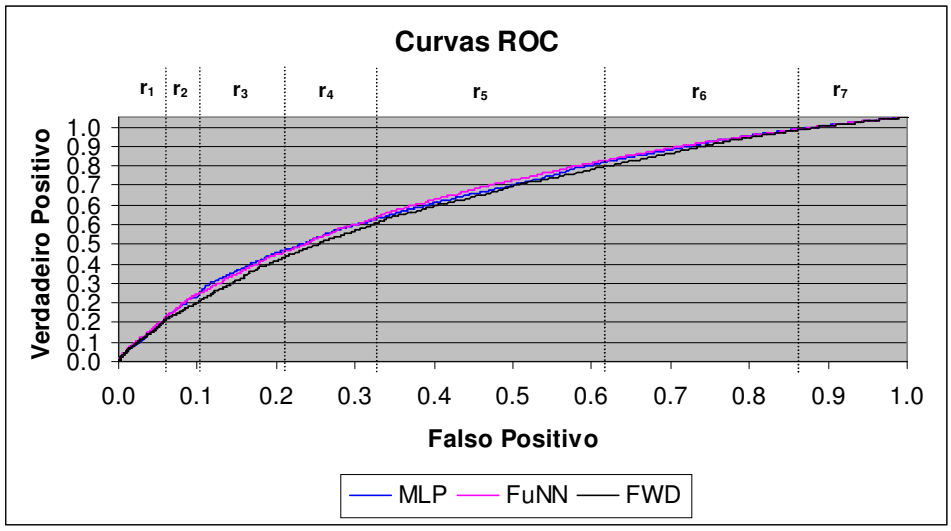

Figura 7.6 - Curvas ROC dos modelos neurais

Além das curvas ROC, também foram gerados gráficos de massa mantida a fim de comparar o desempenho dos modelos neurais entre si e com relação à classificação realizada pela empresa. Este gráfico apresenta duas curvas que representam a massa mantida de clientes e a redução da taxa de inadimplência na carteira de crédito da empresa. A massa mantida indica a quantidade de clientes da massa de dados original que permaneceria na carteira da empresa caso um determinado limiar fosse usado como ponto de corte na saída dos classificadores (pontuação entre 0 e 100 para cada solicitante de crédito). A redução da inadimplência representa o quanto haveria de redução de maus pagadores na carteira da empresa em função do mesmo ponto de corte. Os valores da massa mantida e da redução da taxa de inadimplência são calculados pelas Equações 7.2 e 7.3.

$$massa\ mantida = \frac{total\ da\ base - (total\ de\ bons\ classificados\ como\ mau + total\ de\ maus\ classificados\ como\ mau)}{total\ da\ base} \quad (7.2)$$

$$redução\ da\ inadimplência = \frac{total\ de\ maus - total\ de\ maus\ classificados\ como\ mau}{total\ de\ maus} \quad (7.3)$$

A análise do gráfico de massa mantida é realizada da seguinte forma: se a curva de inadimplência apresentar, em função do ponte de corte, uma redução mais acentuada do que a curva de massa mantida (curva de inadimplência sempre está abaixo da curva de massa mantida), o classificador apresenta um desempenho superior ao critério utilizado pela empresa na concessão de crédito.

As Figuras 7.7, 7.8 e 7.9 apresentam os gráficos de massa mantida dos modelos MLP, FuNN e FWD, respectivamente.

Como pode ser observado, em nenhuma região dos gráficos, a curva de redução da inadimplência apareceu numa posição acima da curva de massa mantida. Portanto, os classificadores apresentam desempenho superior ou equivalente ao critério utilizado pela empresa para classificar os clientes como bons ou maus pagadores. Estes resultados demonstram que a aplicação desses modelos ao processo de decisão da empresa conduziria a um melhor retorno financeiro e maior satisfação para os clientes.

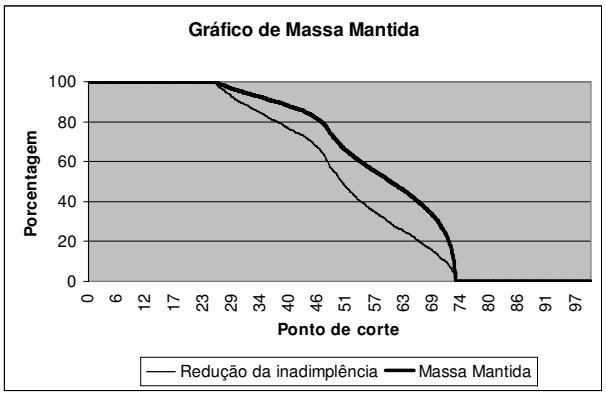

Figura 7.7 - Gráfico de massa mantida da rede MLP

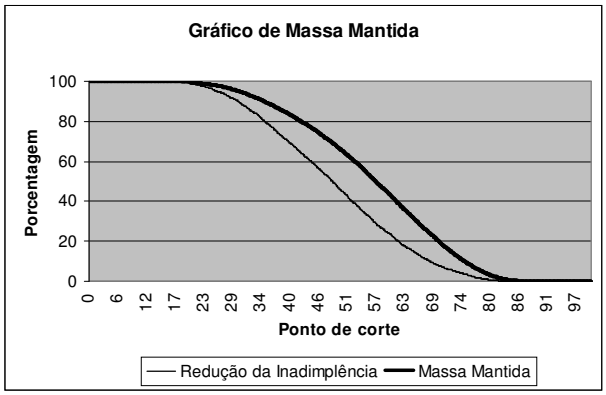

Figura 7.8 - Gráfico de massa mantida da rede FuNN

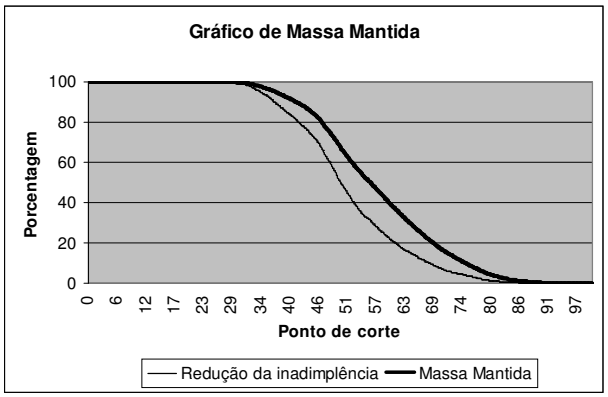

Figura 7.9 - Gráfico de massa mantida da rede FWD

A fim de obter uma visão geral e comparar os modelos neurais com relação à massa mantida e redução da taxa de inadimplência, os gráficos das Figuras 7.7, 7.8 e 7.9 foram combinados em dois gráficos. O primeiro gráfico (Figura 7.10) apresenta as curvas de massa mantida de todos os modelos neurais. O segundo gráfico (Figura 7.11) mostra as curvas de redução da taxa de inadimplência de todos os modelos neurais.

Para analisar melhor os resultados da Figura 7.10, o gráfico foi dividido em dez regiões (r_1, r_2, ..., r_{10}). Nas regiões r_1 e r_{10}, as curvas dos modelos estão sobrepostas, portanto todos os modelos apresentam massa mantida aproximadamente igual. A única região em que o modelo FuNN apresenta massa mantida superior é a r_8. Nesta região, a massa mantida do modelo FWD é aproximadamente igual à do modelo FuNN. As regiões em que o modelo MLP apresenta massa mantida superior são r_2, r_5, r_6 e r_7. Nas regiões r_2 e r_5, a massa mantida do modelo FWD é aproximadamente igual à do modelo MLP. O modelo FWD apresenta massa mantida superior nas regiões r_2, r_3, r_4, r_5, r_8 e r_9.

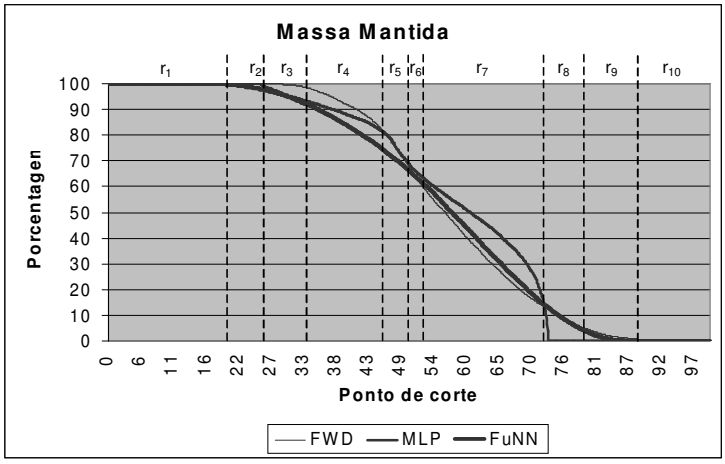

Figura 7.10 - Massa mantida dos modelos neurais

A análise da Figura 7.11 foi realizada da mesma forma que a análise da Figura 7.10. Deste modo, o gráfico foi dividido em dez regiões (r_1, r_2, ..., r_{10}). Nas regiões r_1 e r_{10}, as curvas dos modelos estão

sobrepostas, portanto todos os modelos apresentam redução da taxa de inadimplência aproximadamente igual. A única região em que o modelo FuNN apresenta maior redução na taxa de inadimplência é a r_9. Nesta região, o modelo FWD apresenta uma redução aproximadamente igual à do modelo FuNN. As regiões em que o modelo MLP apresenta maior redução na taxa de inadimplência são r_2, r_6, r_7 e r_8. Nas regiões r_2 e r_6, o modelo FWD apresenta uma redução aproximadamente igual à do modelo MLP. As regiões em que o modelo FWD apresenta maior redução na taxa de inadimplência são r_2, r_3, r_4, r_5, r_6 e r_9.

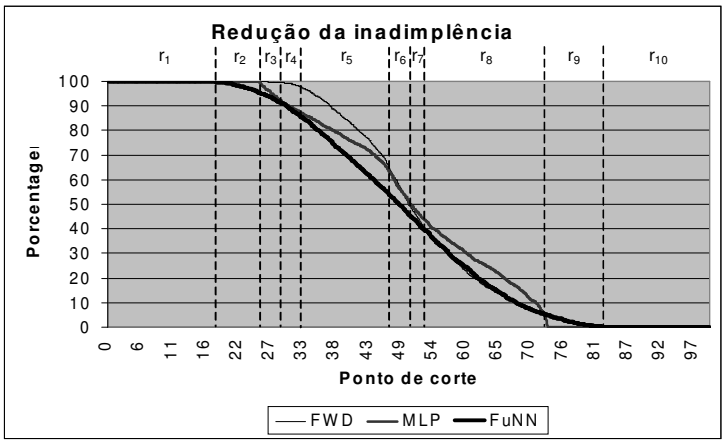

Figura 7.11 - Redução da taxa de inadimplência dos modelos neurais

Aplicação do Problema XOR ao Modelo FWD

Como as limitações das redes *perceptron* com uma camada foram reveladas a partir da aplicação da função lógica XOR (OU exclusivo), este problema tornou-se um padrão por meio do qual a performance dos modelos neurais novos é avaliada [Beale & Jackson 1991]. Por esta razão, o modelo FWD também foi investigado neste problema. Como resultado, verificou-se que tal modelo não é capaz de classificar corretamente todos os casos do problema XOR. Portanto, FWD não é capaz de resolver problemas não linearmente separáveis. Apesar dessa limitação, este modelo apresentou uma performance aproximada à dos modelos FuNN e MLP, e, como será abordado na seção 7.4.4, pode ser aplicado na etapa de seleção de dados para identificar os atributos relevantes.

TREPAN

Metodologia Experimental

Os experimentos realizados com a técnica TREPAN foram executados obedecendo aos seguintes critérios e metodologia:

- A base de dados utilizada contém 27 atributos de entrada e um atributo de saída (classe). O número de atributos utilizados pela técnica TREPAN é bem menor porque esta técnica é capaz de lidar com atributos nominais e numéricos, não sendo, portanto, necessário realizar a codificação binária sobre os atributos nominais;
- A melhor configuração da rede MLP foi utilizada como oráculo;

- ♣ O critério de parada utilizado foi o tamanho máximo da árvore (número de nós internos);
- ♣ Foi definido um conjunto de valores para cada parâmetro ajustável. Todas as combinações possíveis destes valores resultaram em um conjunto de configurações a serem investigadas;
- ♣ Como o objetivo da técnica TREPAN é representar o conhecimento incorporado por uma rede neural, o critério utilizado para escolher a melhor configuração foi a fidelidade da classificação realizada pela árvore de decisão com relação à classificação realizada pela rede.

Parametrização

A Tabela 7.16 mostra os parâmetros utilizados nos experimentos com a técnica TREPAN.

Tabela 7.16 - Parâmetros da técnica TREPAN

Parâmetros	Valores
Tipo de distribuição/estimativa de densidade	Normal, Kernel
Tamanho mínimo das amostras	0, 100, 500, 1000, 2000, 3000
Tamanho máximo da árvore (nós internos)	10, 15, 20, 30, 35

Além de utilizar estimativas de densidade de Kernel para modelar atributos contínuos durante a geração de exemplos artificiais [Craven 1996], também foi investigada a geração de exemplos a partir de uma distribuição Normal [ISEL 2004]. A distribuição Normal também foi investigada, pois a aplicação deste tipo de distribuição vem sendo bastante utilizada em diversos problemas reais e já é uma técnica bastante consolidada na Estatística. Neste caso, a média e o desvio padrão da distribuição Normal são obtidos a partir dos exemplos que alcançam o nó.

Como [Milaré & Carvalho 2001] demonstraram que a geração de exemplos artificiais na técnica TREPAN pode não ser uma boa estratégia, foram realizados experimentos sem a geração de exemplos artificiais (*tamanho mínimo das amostras = 0*) e com a geração de exemplos artificiais (*tamanho mínimo das amostras* $\neq$ *0*), a fim de verificar se para o problema investigado a geração de exemplos artificiais resulta em melhor precisão e fidelidade da árvore de decisão.

Os valores do parâmetro *tamanho máximo da árvore* foram escolhidos de forma empírica, pois não se sabia, inicialmente, quantos nós internos eram necessários para resolver o problema.

Com relação ao tipo de teste presente nos nós da árvore, optou-se por utilizar testes booleanos simples, pois apesar da utilização de testes *m-de-n* gerar árvores de decisão mais concisas, a compreensibilidade pode ser comprometida [Faifer & Janikow 1999].

Resultados Obtidos

Na fase experimental, todas as combinações possíveis dos valores dos parâmetros ajustáveis foram investigadas. Ao final da investigação, definiu-se a melhor configuração (maior a taxa de fidelidade. A Tabela 7.17 mostra a melhor configuração de TREPAN.

Tabela 7.17 - Melhor configuração da técnica TREPAN

Parâmetros	Valores
Tipo de distribuição/estimativa de densidade	-
Tamanho mínimo das amostras	0
Tamanho máximo da árvore (nós internos)	35

O tipo de distribuição ou estimativa de densidade não foi especificado porque na melhor configuração não foram gerados exemplos artificiais (*tamanho mínimo das amostras = 0*).

A Tabela 7.18 mostra os resultados obtidos pela melhor configuração da técnica TREPAN. Esta tabela mostra três tipos de taxa de classificação do conjunto de teste (total e por classe). Também são especificados o tamanho da árvore (número de nós internos) e a fidelidade da classificação realizada pela árvore de decisão com relação à classificação realizada pela rede MLP da qual a árvore foi extraída. A fidelidade mede a porcentagem dos exemplos do conjunto de teste cuja classificação da árvore é igual à classificação fornecida pela rede. Similarmente à taxa de classificação, são especificados três tipos de fidelidade (total e por classe). Como não foram gerados exemplos artificiais na melhor configuração (não existe aleatoriedade associada à geração da árvore de decisão), não foram executadas 30 iterações para obter a média e o desvio padrão dessas taxas.

Tabela 7.18 - Resultados do conjunto de teste da técnica TREPAN

Taxa de classificação			**Fidelidade**			**Nós internos**
Total	**Adimplente**	**Inadimplente**	**Total**	**Adimplente**	**Inadimplente**	12
66,51%	71,20%	47,52%	85,78%	88,81%	79,14%	

A diferença acentuada entre a quantidade de exemplos das classes adimplente e inadimplente pode ter afetado a fidelidade, ou seja, a fidelidade para a classe adimplente foi maior do que a fidelidade para a classe inadimplente.

A Tabela 7.19 mostra os resultados da classificação do conjunto de teste obtidos pela árvore de decisão e pela rede MLP da qual a árvore foi extraída.

Tabela 7.19 - Comparação dos resultados obtidos pela técnica TREPAN e pela rede MLP

	Taxa de Classificação		
Técnica/Modelo	**Total**	**Adimplente**	**Inadimplente**
TREPAN	66,51%	71,20%	47,52%
MLP	68,22%	72,97%	48,99%

As taxas de classificação (total e por classe) da árvore de decisão e da rede MLP foram bastante aproximadas.

7.4.4. Análise das Técnicas de Seleção de Atributos

Apesar da maioria dos algoritmos de aprendizagem serem projetados para aprender quais atributos são mais apropriados para distinguir as classes de padrões, na prática, a adição de atributos irrelevantes ou redundantes e a remoção de atributos relevantes podem resultar em soluções de qualidade inferior e retardar o processo de aprendizagem. Embora o especialista do domínio possa escolher os atributos mais relevantes, essa tarefa pode ser difícil e demandar tempo. Outro benefício resultante da seleção de atributos é a obtenção de uma representação mais compacta do conceito alvo, direcionando a atenção do usuário para os atributos mais importantes. Por estas razões, a seleção de atributos é um aspecto de extrema importância no processo de KDD [Witten & Frank 2000]. Portanto, além da análise nas etapas de mineração de dados e apresentação do conhecimento, também foram investigadas duas técnicas de seleção de atributos: a técnica da rede FWD e através da

árvore de decisão gerada por TREPAN. Os resultados dessa investigação foram obtidos utilizando apenas os atributos considerados relevantes pelas técnicas e os mesmos parâmetros de treinamento dos modelos neurais.

O processo de seleção de atributos da rede FWD é realizado a partir de uma análise dos valores das conexões peso, que representam o grau de importância (de 0 a 1) dos atributos para cada classe. Desta forma, os atributos foram organizados e removidos em grupos (grupo 1 – atributos cujos graus de importância são menores que 0,1 para as duas classes; grupo 2 – atributos cujos graus de importância são menores que 0,2 para as duas classes; e assim por diante) até que a taxa de classificação do conjunto de teste ficasse menor do que a taxa obtida com todos os atributos. A remoção procedeu até o grupo 3 (atributos cujos graus de importância são menores que 0,3 para as duas classes). Do total de 68 atributos, permaneceram apenas 26 atributos (Tabela 7.20).

Tabela 7.20 - Atributos selecionados pela técnica da rede FWD

SEXO	CIDADE_RESIDENCIAL3	TIPO_CLIENTE1
ESTADO_CIVIL1	FLAG_TEL_RESIDENCIAL	TIPO_CLIENTE2
ESTADO_CIVIL4	DDD_RESIDENCIAL1	TEMPO_RESIDENCIA
ESTADO_CIVIL5	DDD_RESIDENCIAL2	FLAG_NOME_PAI
IDADE	DDD_RESIDENCIAL4	TEMPO_EMPREGO
UF_RESIDENCIAL1	CEP_DIGITO1	RENDA_CONJUGE
UF_RESIDENCIAL3	CEP_DIGITO2	RENDA_LIQUIDA
UF_RESIDENCIAL4	CEP_DIGITO3	NUM_CARTOES_ADICIONAIS
CIDADE_RESIDENCIAL2	DIA_VENCIMENTO	

A Tabela 7.21 mostra os resultados obtidos pelos modelos neurais após a remoção dos atributos considerados irrelevantes pela técnica da rede FWD. Os resultados são provenientes da execução de 30 iterações com pesos iniciais diferentes. Esta tabela mostra três tipos de média da taxa de classificação do conjunto de teste (total e por classe) e o desvio padrão para cada tipo de média.

Tabela 7.21 - Resultado da classificação do conjunto de teste após a seleção de atributos pela rede FWD

	Médias			**Desvio padrão**		
Modelo	**Total**	**Adimplente**	**Inadimplente**	**Total**	**Adimplente**	**Inadimplente**
MLP	66,54%	69,72%	53,68%	0,37	0,70	1,04
FWD	67,32%	72,10%	47,99%	0,00	0,00	0,01
FuNN	66,04%	68,84%	54,72%	1,16	2,16	3,02

A seleção de atributos através da árvore de decisão gerada pela técnica TREPAN é realizada da seguinte forma: os atributos que não estiverem presentes na árvore são considerados irrelevantes e são removidos da base [Han & Kamber 2001]. Considerando a base de dados original (antes da codificação para os modelos neurais), do total de 27 atributos, apenas 8 foram considerados relevantes (Tabela 7.22), o que corresponde a 29 atributos na base de dados codificada (Tabela 7.23). A árvore de decisão gerada pela técnica TREPAN é apresentada na próxima seção.

Tabela 7.22 - Atributos selecionados através da árvore de decisão

NUMERO_LOJA	CIDADE_RESIDENCIAL
SEXO	FLAG_TEL_RESIDENCIAL
ESTADO_CIVIL	DDD_RESIDENCIAL
IDADE	FLAG_MESMA_CIDADE_RESIDENCIA_COMERCIAL

Tabela 7.23 - Atributos da base de dados codificada selecionados através da árvore de decisão

NUMERO_LOJA1	ESTADO_CIVIL3	CIDADE_RESIDENCIAL7
NUMERO_LOJA2	ESTADO_CIVIL4	FLAG_TEL_RESIDENCIAL
NUMERO_LOJA3	ESTADO_CIVIL5	DDD_RESIDENCIAL1
NUMERO_LOJA4	IDADE	DDD_RESIDENCIAL2
NUMERO_LOJA5	CIDADE_RESIDENCIAL1	DDD_RESIDENCIAL3
NUMERO_LOJA6	CIDADE_RESIDENCIAL2	DDD_RESIDENCIAL4
NUMERO_LOJA7	CIDADE_RESIDENCIAL3	DDD_RESIDENCIAL5
SEXO	CIDADE_RESIDENCIAL4	DDD_RESIDENCIAL6
ESTADO_CIVIL1	CIDADE_RESIDENCIAL5	FLAG_MESMA_CIDADE_RESIDENCIA_COMERCIAL
ESTADO_CIVIL2	CIDADE_RESIDENCIAL6	

A Tabela 7.24 mostra os resultados obtidos pelos modelos neurais após a remoção dos atributos considerados irrelevantes pela árvore de decisão. Os resultados são provenientes da execução de 30 iterações com pesos iniciais diferentes. Esta tabela mostra três tipos de média da taxa de classificação do conjunto de teste (total e por classe) e o desvio padrão para cada tipo de média.

Tabela 7.24 - Resultado da classificação do conjunto de teste após a seleção de atributos pela árvore

	Médias			**Desvio padrão**		
Modelo	**Total**	**Adimplente**	**Inadimplente**	**Total**	**Adimplente**	**Inadimplente**
MLP	65,13%	67,58%	55,19%	0,57	0,89	1,07
FWD	65,00%	67,75%	53,89%	0,00	0,00	0,00
FuNN	65,15%	67,70%	54,82%	0,66	1,26	2,08

As Figuras 7.12, 7.13 e 7.14 mostram uma comparação das médias das taxas de classificação (total, da classe adimplente e da classe inadimplente) dos modelos neurais no conjunto de teste antes e depois da aplicação das técnicas de seleção de atributos.

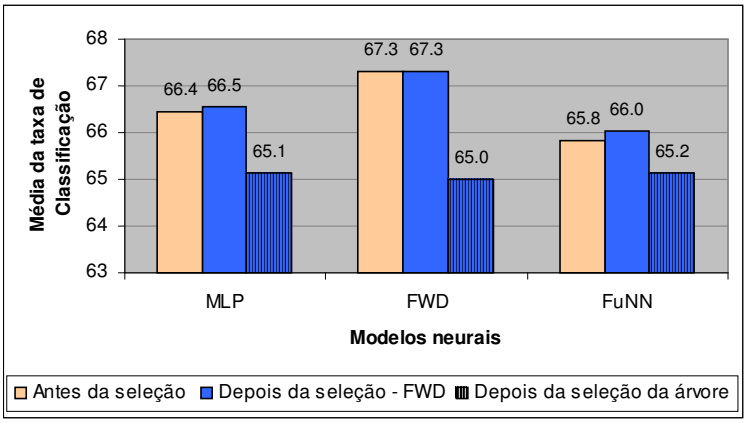

Figura 7.12 - Comparação das médias da taxa de classificação total no conjunto de teste

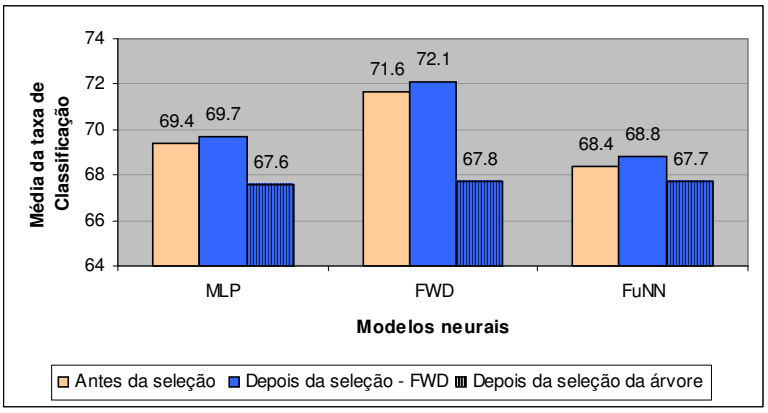

Figura 7.13 - Comparação das médias da taxa de classificação da classe adimplente no conjunto de teste

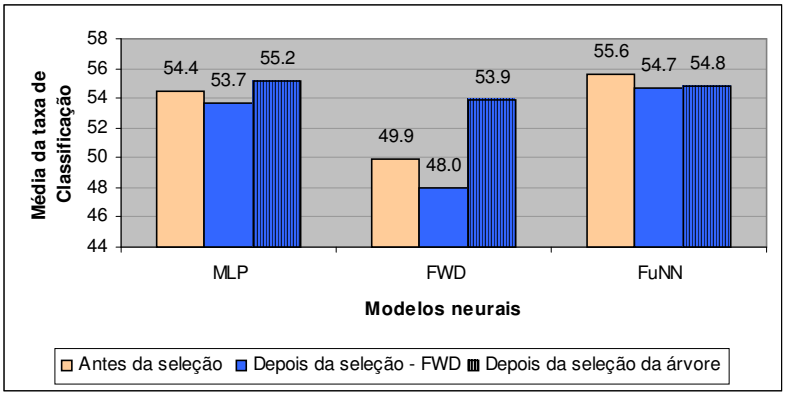

Figura 7.14 - Comparação das médias da taxa de classificação da classe inadimplente no conjunto de teste

Analisando as Figuras 7.12, 7.13 e 7.14, as seguintes observações podem ser realizadas:

- ♣ A remoção de atributos considerados irrelevantes pela técnica de seleção de atributos da rede FWD aumentou a média da taxa de classificação total e da classe adimplente, e reduziu a média da taxa de classificação da classe inadimplente. A maior diferença ocorreu na média da taxa de classificação da classe inadimplente do modelo FWD, que apresentou um decréscimo de 1,91%;
- ♣ A remoção de atributos que não estavam presentes na árvore aumentou a média da taxa de classificação da classe inadimplente dos modelos MLP e FWD, e diminuiu nos demais casos. As maiores diferenças ocorreram nos modelos MLP (decréscimo de 1,31% na média da taxa de classificação total e decréscimo de 1,83% na média da taxa de classificação da classe adimplente) e FWD (decréscimo de 2,32% na média da taxa de classificação total, decréscimo de 3,88% na média da taxa de classificação da classe adimplente e acréscimo de 3,99% na média da taxa de classificação da classe inadimplente);
- ♣ A técnica de seleção de atributos da rede FWD se mostrou mais eficiente do que a realizada com a árvore de decisão. No entanto, a técnica de seleção de atributos utilizando a árvore de decisão também mostrou resultados satisfatórios quando aplicada aos modelos FuNN e MLP.

7.4.5. Apresentação do Conhecimento Minerado

Esta seção mostra a análise da etapa de apresentação do conhecimento minerado. A execução desta etapa foi realizada utilizando dois tipos de base: a base de dados resultante do processo de seleção de atributos da rede FWD e a base de dados original sem codificação. A base de dados resultante do processo de seleção de atributos foi aplicada aos modelos FuNN e FWD. A base de dados original sem codificação foi aplicada à rede MLP, pois não é possível realizar o mapeamento da base de dados reduzida para a base de dados utilizada pela técnica TREPAN, que é capaz de lidar com dados nominais e numéricos. A conversão de um registro original em um registro codificado para a rede MLP é realizada de forma automática pela ferramenta *Neural Mining*.

O conhecimento minerado a partir da rede FWD é representado na forma de regras *fuzzy* Se-Então. Neste modelo, apenas as condições (atributos de entrada) das regras são expressas através de um termo lingüístico. Cada atributo de entrada está associado a duas funções de pertinência (Equação 5.5), uma para cada classe. Para exemplificar, a Tabela 7.25 mostra as funções de pertinência do atributo *DIA_VENCIMENTO1* (indicado pelo índice *18*).

Tabela 7.25 - Funções de pertinência do atributo *DIA_VENCIMENTO1*

Classe Adimplente
$U_{A_{1,18}}(x_{18}) = exp[-\frac{1}{2\sigma^2} \cdot w_{1,18}^2 \cdot (x_{18} - m_{18,1})^2] = exp[-0{,}278 \cdot (x_{18} - 0{,}923)^2]$
Classe Inadimplente
$U_{A_{2,18}}(x_{18}) = exp[-\frac{1}{2\sigma^2} \cdot w_{2,18}^2 \cdot (x_{18} - m_{18,2})^2] = exp[-0{,}376 \cdot (x_{18} - 0{,}391)^2]$

No modelo FWD, a extração das regras é realizada a partir das conexões de memória obtidas durante a fase de treinamento. Para cada classe é gerada uma regra, que contém todos os atributos de entrada no antecedente. [Li et al. 2002] sugerem que o termo lingüístico associado a cada atributo seja expresso da seguinte forma: *aproximadamente conexão_memória_associada_ao_atributo_e_classe*. Deste modo, todos os atributos de entrada são tratados como sendo numéricos. Como resultado, a representação semântica dos atributos booleanos é prejudicada, pois normalmente este tipo de atributo é representado por duas funções de pertinência, uma representando o valor *verdadeiro* e outra representando o valor *falso*.

As regras extraídas da rede FWD são especificadas na Tabela 7.26.

A precisão associada às regras das classes adimplente e inadimplente são iguais às taxas de classificação da rede FWD para estas classes, ou seja, 72,10% e 47,99%, respectivamente. Esta equivalência ocorre porque a saída da rede (Equações 4.16 e 4.17) é obtida pelo produto dos graus de pertinência associados a cada condição. Portanto, se o produto dos graus de pertinência das condições da regra associada à classe adimplente (saída do nó que representa a classe adimplente) for maior do que o produto dos graus de pertinência das condições da regra associada à classe inadimplente (saída do nó que representa a classe inadimplente), o cliente é classificado como adimplente. Caso contrário, o cliente é classificado como inadimplente.

Tabela 7.26 - Regras extraídas da rede FWD

SE sexo *é aproximadamente 0,587* E estado_civil1 *é aproximadamente 0,285* E estado_civil4 *é aproximadamente 0,368* E estado_civil5 *é aproximadamente 0,504* E idade *é aproximadamente 0,555* E ref_residencial1 *é aproximadamente 0,775* E ref_residencial3 *é aproximadamente 0,893* E ref_residencial4 *é aproximadamente 0,617* E cidade_residencial2 *é aproximadamente 0,193* E cidade_residencial3 *é aproximadamente 0,550* E flag_tel_residencial *é aproximadamente 0,788* E ddd_residencia1 *é aproximadamente 0,916* E ddd_residencia2 *é aproximadamente 0,502* E ddd_residencia4 *é aproximadamente 0,386* E cep_digito1 *é aproximadamente 0,858* E cep_digito2 *é aproximadamente 0,250* E cep_digito3 *é aproximadamente 0,807* E dia_vencimento *é aproximadamente 0,963* E tipo_cliente1 *é aproximadamente 0,923* E tipo_cliente2 *é aproximadamente 0,578* E tempo_residencia *é aproximadamente 0,570* E flag_nome_pai *é aproximadamente 0,638* E tempo_emprego *é aproximadamente 0,428* E renda_conjuge *é aproximadamente 0,411* E renda_liquida *é aproximadamente 0,599* E num_cartoes_adicionais *é aproximadamente 0,537* ENTÃO cliente é adimplente	SE sexo *é aproximadamente 0,135* E estado_civil1 *é aproximadamente 0,731* E estado_civil4 *é aproximadamente 0,719* E estado_civil5 *é aproximadamente 0,639* E idade *é aproximadamente 0,924* E ref_residencial1 *é aproximadamente 0,939* E ref_residencial3 *é aproximadamente 0,983* E ref_residencial4 *é aproximadamente 0,938* E cidade_residencial2 *é aproximadamente 0,577* E cidade_residencial3 *é aproximadamente 0,200* E flag_tel_residencial *é aproximadamente 0,251* E ddd_residencia1 *é aproximadamente 0,859* E ddd_residencia2 *é aproximadamente 0,810* E ddd_residencia4 *é aproximadamente 0,750* E cep_digito1 *é aproximadamente 0,865* E cep_digito2 *é aproximadamente 0,965* E cep_digito3 *é aproximadamente 0,803* E dia_vencimento *é aproximadamente 0,391* E tipo_cliente1 *é aproximadamente 0,708* E tipo_cliente2 *é aproximadamente 0,997* E tempo_residencia *é aproximadamente 0,771* E flag_nome_pai *é aproximadamente 0,602* E tempo_emprego *é aproximadamente 0,896* E renda_conjuge *é aproximadamente 0,964* E renda_liquida *é aproximadamente 0,935* E num_cartoes_adicionais *é aproximadamente 0,763* ENTÃO cliente é inadimplente

As regras obtidas da rede FuNN foram extraídas pelas técnicas REFuNN e AREFuNN.

Para este estudo de caso, os atributos de entrada (condições das regras) booleanos da rede FuNN são representados por duas funções de pertinência cujos termos lingüísticos são *falso* e *verdadeiro*. Por outro lado, os atributos numéricos são representados por três funções de pertinência cujos termos lingüísticos são *baixo*, *médio* e *alto*. Como exemplo, a Figura 7.15 mostra as funções de pertinência dos atributos *DDD_RESIDENCIAL4* e *TEMPO_RESIDENCIA* obtidas após a fase de treinamento da rede.

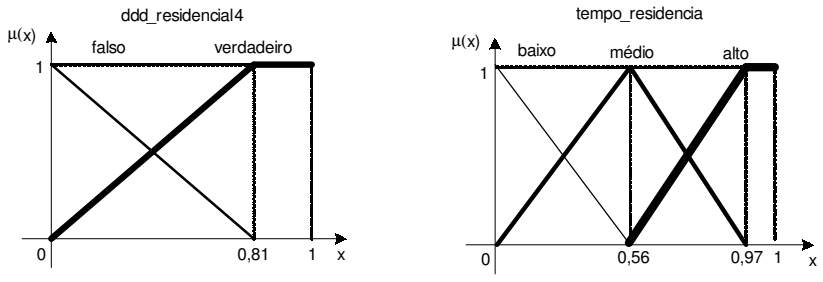

Figura 7.15 - Exemplos de funções de pertinência da rede FuNN

A Tabela 7.27 mostra as regras extraídas pela técnica REFuNN, utilizando $Th_c = 0,5$ e $Th_a = 0,5$.

Tabela 7.27 - Conjunto inicial de regras ponderadas extraído pela técnica REFuNN

Regra$_1$
SE estado_civil1 *é verdadeiro* (0,98) E estado_civil4 *é falso* (0,76) E estado_civil5 *é verdadeiro* (0,74) E
idade *é alta* (1,07) E uf_residencial3 *é falso* (0,59) E uf_residencial4 *é verdadeiro* (0,74) E
ddd_residencial2 *é verdadeiro* (0,97) E ddd_residencial4 *é verdadeiro* (0,90) E cep_digito2 *é baixo* (0,56) E
tempo_emprego *é médio* (0,83) E renda_conjuge *é média* (0,70) E renda_liquida *é baixa* (0,74) E
num_cartoes_adicionais *é baixo* (0,63)
ENTÃO cliente *é adimplente* (0,84)

Regra$_2$
SE estado_civil1 *é falso* (0,88) E estado_civil4 *é falso* (0,93) E estado_civil5 *é falso* (0,75) E
idade *é baixa* (0,61) E flag_tel_residencial *é falso* (1,01) E ddd_residencial4 *é falso* (0,55) E
cep_digito2 *é médio* (0,96) E cep_digito3 *é alto* (0,75) E dia_vencimento *é baixo* (0,57) E
tipo_cliente1 *é falso* (0,66)
ENTÃO cliente *é inadimplente* (0,52)

Regra$_3$
SE estado_civil1 *é falso* (0,81) E estado_civil4 *é falso* (0,77) E cidade_residencial2 *é falso* (0,53) E
ddd_residencial1 *é falso* (0,64) E cep_digito1 *é alto* (0,85) E cep_digito2 *é médio* (0,63) E
tipo_cliente1 *é verdadeiro* (0,85) E renda_conjuge *é alta* (0,66) E renda_liquida *é baixa* (0,82)
ENTÃO cliente *é adimplente* (0,62)

Regra$_4$
SE sexo *é falso* (0,99) E estado_civil1 *é falso* (0,91) E estado_civil5 *é falso* (0,79) E
uf_residencial1 *é verdadeiro* (0,59) E cidade_residencial2 *é falso* (0,84) E flag_tel_residencial *é falso* (0,69) E
ddd_residencial1 *é falso* (0,63) E cep_digito2 *é médio* (0,93) E tempo_emprego *é alto* (0,68) E
num_cartoes_adicionais *é baixo* (0,76)
ENTÃO cliente *é inadimplente* (0,99)

Regra$_5$
SE estado_civil1 *é falso* (0,65) E estado_civil5 *é falso* (0,88) E idade *é alta* (0,97) E
uf_residencial4 *é verdadeiro* (0,55) E cidade_residencial3 *é falso* (0,77) E ddd_residencial1 *é falso* (0,98) E
cep_digito1 *é baixo* (0,97) E cep_digito2 *é alto* (0,60) E tipo_cliente2 *é falso* (0,65) E
tempo_residencia *é médio* (0,58) E flag_nome_pai *é verdadeiro* (0,75) E tempo_emprego *é alto* (1,07) E
renda_liquida *é alta* (0,63) E num_cartoes_adicionais *é alto* (0,74)
ENTÃO cliente *é adimplente* (0,69)

Do total de cinco regras extraídas pela técnica REFuNN, três estão associadas à classe adimplente e duas à classe inadimplente.

Como foi descrito na Seção 5.7.2, a partir do conjunto inicial de regras ponderadas é possível gerar dois tipos de conjunto de regras: simples e agregadas. O conjunto de regras simples é gerado removendo os graus de importância do antecedente e o grau de certeza do conseqüente ou criando regras com uma única condição cujo grau de importância é maior que o limiar Th_{OU}. No conjunto de regras agregadas, as regras iniciais que possuem condições e conclusões iguais, diferindo apenas nos graus de importância, são agregadas em uma regra.

No problema de análise de crédito investigado, foi possível derivar apenas o conjunto de regras simples, pois as condições e conclusões das regras ponderadas são diferentes. A Tabela 7.28 mostra o conjunto de regras simples extraídas pela técnica REFuNN, considerando o limiar $Th_{OU} = 1$.

Tabela 7.28 - Conjunto de regras simples extraído pela técnica REFuNN

Regras simples geradas a partir da Regra$_1$

SE estado_civil1 é *verdadeiro* E estado_civil4 *é falso* E estado_civil5 *é verdadeiro* E idade *é alta* E
uf_residencial3 *é falso* E uf_residencial4 *é verdadeiro* E ddd_residencial2 *é verdadeiro* E
ddd_residencial4 *é verdadeiro* E cep_digito2 *é baixo* E tempo_emprego *é médio* E
renda_conjuge *é média* E renda_liquida *é baixa* E num_cartoes_adicionais *é baixo*
ENTÃO cliente *é adimplente*

SE idade *é alta*
ENTÃO cliente *é adimplente*

Regras simples geradas a partir da Regra$_2$

SE estado_civil1 *é falso* E estado_civil4 *é falso* E estado_civil5 *é falso* E idade *é baixa* E
flag_tel_residencial *é falso* E ddd_residencial4 *é falso* E cep_digito2 *é médio* E cep_digito3 *é alto* E
dia_vencimento *é baixo* E tipo_cliente1 *é falso*
ENTÃO *cliente é inadimplente*

SE flag_tel_residencial *é falso*
ENTÃO cliente *é inadimplente*

Regras simples geradas a partir da Regra$_3$

SE estado_civil1 *é falso* E estado_civil4 *é falso* E cidade_residencial2 *é falso* E ddd_residencial1 *é falso* E
cep_digito1 *é alto* E cep_digito2 *é médio* E tipo_cliente1 *é verdadeiro* E renda_conjuge *é alta* E
renda_liquida *é baixa*
ENTÃO cliente *é adimplente*

Regras simples geradas a partir da Regra$_4$

SE sexo *é falso* E estado_civil1 *é falso* E estado_civil5 *é falso* E uf_residencial1 *é verdadeiro* E
cidade_residencial2 *é falso* E flag_tel_residencial *é falso* E ddd_residencial1 *é falso* E
cep_digito2 *é médio* E tempo_emprego *é alto* E num_cartoes_adicionais *é baixo*
ENTÃO cliente *é inadimplente*

Regras simples geradas a partir da Regra$_5$

SE estado_civil1 *é falso* E estado_civil5 *é falso* E idade *é alta* E uf_residencial4 *é verdadeiro* E
cidade_residencial3 *é falso* E ddd_residencial1 *é falso* E cep_digito1 *é baixo* E cep_digito2 *é alto* E
tipo_cliente2 *é falso* E tempo_residencia *é médio* E flag_nome_pai *é verdadeiro* E tempo_emprego *é alto* E
renda_liquida *é alta* E num_cartoes_adicionais *é alto*
ENTÃO cliente *é adimplente*

SE tempo_emprego *é alto*
ENTÃO cliente *é adimplente*

A Tabela 7.29 mostra as regras extraídas pela técnica AREFuNN. Nesta técnica também não foi possível derivar o conjunto de regras agregadas, pois as condições e conclusões das regras ponderadas são diferentes.

Do total de nove regras extraídas pela técnica AREFuNN, cinco estão associadas à classe adimplente e quatro à classe inadimplente.

Tabela 7.29 - Regras extraídas pela técnica AREFuNN

Regra$_1$
SE estado_civil1 *é verdadeiro* E idade *é alta* E ddd_residencial2 *é verdadeiro* E
ddd_residencial4 *é verdadeiro* E tempo_emprego *é médio*
ENTÃO cliente *é adimplente*

Regra$_2$
SE uf_residencial1 *é verdadeiro* E uf_residencial3 *é falso* E uf_residencial4 *é verdadeiro* E
flag_tel_residencial *é falso*
ENTÃO cliente *é inadimplente*

Regra$_3$
SE cep_digito3 *é médio*
ENTÃO cliente *é adimplente*

Regra$_4$
SE estado_civil1 *é falso* E estado_civil4 *é falso* E flag_tel_residencial *é falso* E cep_digito2 *é médio*
ENTÃO cliente *é inadimplente*

Regra$_5$
SE estado_civil5 *é verdadeiro* E ddd_residencial4 *é falso* E cep_digito2 *é médio* E tempo_emprego *é baixo* E
renda_liquida *é baixa*
ENTÃO cliente *é inadimplente*

Regra$_6$
SE cep_digito1 *é alto* E tipo_cliente1 *é verdadeiro* E renda_liquida *é baixa*
ENTÃO cliente *é adimplente*

Regra$_7$
SE sexo *é falso* E estado_civil1 *é falso* E cidade_residencial2 *é falso* E cep_digito2 *é médio*
ENTÃO cliente *é inadimplente*

Regra$_8$
SE estado_civil1 *é falso* E ddd_residencial1 *é verdadeiro* E ddd_residencial4 *é falso* E cep_digito3 *é baixo* E
tempo_residencia *é baixo* E renda_conjuge *é média*
ENTÃO cliente *é adimplente*

Regra$_9$
SE estado_civil5 *é falso* E idade *é alta* E ddd_residencial1 *é falso* E cep_digito1 *é baixo* E
tempo_emprego *é alto*
ENTÃO cliente *é adimplente*

A precisão das bases de regras geradas pelas técnicas REFuNN e AREFuNN foi obtida usando o método de inferência *fuzzy max-min* [Zadeh 1965], que é realizado da seguinte forma:

- As regras são divididas em dois grupos de acordo com a conclusão, ou seja, grupo das regras associadas à classe adimplente e grupo das regras associadas à classe inadimplente;
- A relevância ou grau de pertinência das condições das regras é calculado aplicando os valores dos atributos de entrada às funções de pertinência das condições das regras;

- O grau de verdade (*rule strength*) de cada regra é calculado como sendo o menor valor dos graus de pertinência das condições;
- As saídas *fuzzy* são determinadas pelo grau de verdade máximo de todas as regras que possuem a mesma conclusão. Portanto, existe uma saída *fuzzy* para cada termo lingüístico (classe) associado ao atributo de saída;
- A classe cuja saída *fuzzy* é maior é considerada a classificação da base de regras.

A Tabela 7.30 mostra a precisão da base de regras extraídas pelas técnicas REFuNN e AREFuNN. Também é considerado o caso em que as regras extraídas pelas duas técnicas são agrupadas em um único conjunto. Esta tabela mostra as seguintes porcentagens: exemplos do conjunto de teste que foram classificados corretamente, exemplos da classe adimplente que foram classificados corretamente, exemplos da classe inadimplente que foram classificados corretamente, exemplos que não ativaram nenhuma regra, exemplos que foram classificados de forma ambígua (as saídas *fuzzy* para as duas classes foram iguais) e exemplos que foram classificados erroneamente.

Tabela 7.30 - Precisão das regras extraídas pelas técnicas REFuNN e AREFuNN

Regras	Acerto	Adimplente	Inadimplente	Não ativaram	Classificação ambígua	Erro
REFuNN	25,10%	24,90%	25,91%	59,45%	0,97%	14,48%
AREFuNN	45,10%	50,04%	25,13%	34,24%	0,00%	20,66%
Completa	49,60%	53,07%	35,60%	23,51%	0,97%	25,92%

Analisando os resultados da Tabela 7.30, as seguintes observações podem ser realizadas:

- O conjunto de regras extraído pela técnica REFuNN apresentou a menor taxa de acerto (25,10%) e o maior número de exemplos que não ativaram nenhuma regra (59,45%);
- Como a técnica AREFuNN foi proposta como uma melhoria da técnica REFuNN, o conjunto de regras extraído por AREFuNN apresentou uma taxa de acerto (45,10%) bastante superior ao conjunto de regras extraído por REFuNN. Além disso, o número de exemplos que não ativaram nenhuma regra foi bastante reduzido;
- O conjunto de regras resultante da integração das regras extraídas pelas duas técnicas apresentou a melhor precisão (49,60%). Para este conjunto, o número de exemplos que não ativaram nenhuma regra foi bastante reduzido (23,51%) com relação aos casos em que o conjunto de cada técnica foi usado separadamente;
- A taxa de classificação do conjunto de teste obtida diretamente pela rede FuNN foi superior (média igual a 66,04%) às taxas de classificação obtidas pelos conjuntos de regras considerados;
- Dependendo dos valores estabelecidos para os limiares, as regras podem ser apresentadas em diferentes níveis de abstração. Quanto maior o limiar, mais genéricas são as regras e melhor a qualidade de explicação (nível de compreensibilidade). No entanto, a qualidade de interpretação (a precisão obtida quando as regras são usadas para raciocínio e inferência em novos dados) é afetada. Portanto, regras genéricas podem ser usadas para finalidades de explicação e entendimento do conhecimento contido nos dados de treinamento, enquanto que regras precisas

e específicas podem ser usadas para inferência sobre novos dados. Neste estudo de caso, os limiares resultaram em regras mais simples e, portanto, com precisão inferior à da rede FuNN.

A técnica TREPAN trata a extração de conhecimento como uma tarefa de aprendizagem indutiva, em que o conceito alvo é a função representada por um oráculo e a hipótese produzida é uma árvore de decisão que aproxima seu comportamento. Uma vez que a função alvo nesta dissertação é o conhecimento adquirido por uma RNA, a própria rede é usada como oráculo. A Tabela 7.31 mostra a árvore de decisão extraída pela técnica TREPAN da rede MLP cuja configuração é especificada na Tabela 7.11.

Tabela 7.31 - Árvore de decisão extraída pela técnica TREPAN

```
SE ( idade > 0,26 )
|  SE ( flag_tel_residencial = 0 )
|  |  SE ( idade > 0,43 )
|  |  |  Classe = Adimplente
|  |  SENÃO SE ( flag_mesma_cidade_residencia_comercial = 0 )
|  |  |  Classe = Inadimplente
|  |  SENÃO Classe = Adimplente
|  SENÃO Classe = Adimplente
SENÃO SE ( sexo = 0 )
|  Classe = Inadimplente
SENÃO SE ( flag_tel_residencial = 0 )
|  Classe = Inadimplente
SENÃO SE ( flag_mesma_cidade_residencia_comercial = 0 )
|  SE ( idade > 0,09 )
|  |  SE ( estado_civil = 5 )
|  |  |  Classe = Adimplente
|  |  SENÃO SE ( numero_loja = 28 )
|  |  |  Classe = Adimplente
|  |  SENÃO SE ( cidade_residencial = 16 )
|  |  |  Classe = Inadimplente
|  |  SENÃO SE ( ddd_residencial = 3 )
|  |  |  Classe = Adimplente
|  |  SENÃO Classe = Inadimplente
|  SENÃO Classe = Inadimplente
SENÃO Classe = Adimplente
```

Os valores dos limiares presentes nas condições associadas aos atributos numéricos estão dentro do intervalo [0,1], pois os dados utilizados neste estudo de caso já estavam normalizados e os valores originais não são conhecidos.

A representação gráfica da árvore de decisão gerada pela técnica TREPAN é mostrada na Figura 7.16.

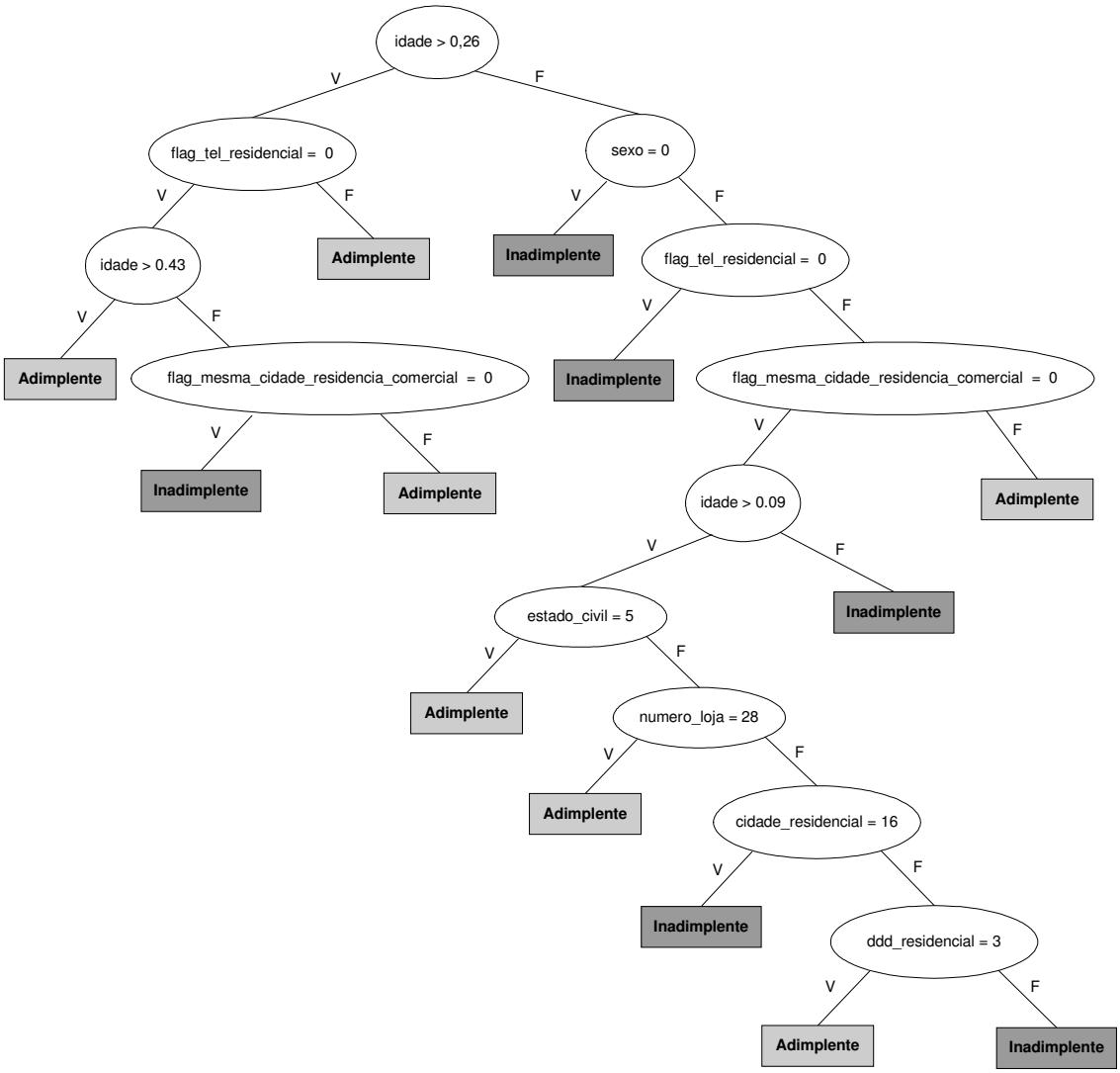

Figura 7.16 - Forma gráfica da árvore de decisão extraída por TREPAN

A Tabela 7.32 mostra as regras obtidas a partir da árvore de decisão gerada pela técnica TREPAN. Cada regra está associada a dois valores: suporte(*s*) e confiança(*c*) [Liu et al. 1998]. O suporte (Equação 7.4) se refere ao número de registros da base de dados que satisfazem as condições da regra e pertencem à mesma conclusão da regra. A confiança (Equação 7.5) representa o número de registros cobertos pelas condições da regra que pertencem à mesma conclusão da regra. Os valores de suporte e confiança foram obtidos considerando a base de dados usada na geração da árvore de decisão, ou seja, o conjunto de treinamento.

$$suporte = \frac{número\ de\ registros\ cobertos\ pelas\ condições\ e\ conclusão}{número\ de\ registros\ da\ base\ de\ dados} \tag{7.4}$$

$$confiança = \frac{número\ de\ registros\ cobertos\ pelas\ condições\ e\ conclusão}{número\ de\ registros\ cobertos\ pelas\ condições} \tag{7.5}$$

Tabela 7.32 - Regras geradas da árvore de decisão

Regra$_1$ (s = 4,28%; c = 79,67%)
SE flag_tel_residencial = 0 E idade > 0,43
ENTÃO cliente é adimplente

Regra$_2$ (s = 39,99%; c = 87,16%)
SE idade > 0,26 E flag_tel_residencial ≠ 0
ENTÃO cliente é adimplente

Regra$_3$ (s = 0,64%; c = 28,59%)
SE 0,26 < idade <= 0,43 E flag_tel_residencial = 0 E flag_mesma_cidade_residencia_comercial = 0
ENTÃO cliente é inadimplente

Regra$_4$ (s = 1,32%; c = 76,89%)
SE 0,26 < idade <= 0,43 E flag_tel_residencial = 0 E flag_mesma_cidade_residencia_comercial ≠ 0
ENTÃO cliente é adimplente

Regra$_5$ (s = 4,32%; c = 27,09%)
SE idade <= 0,26 E sexo = 0
ENTÃO cliente é inadimplente

Regra$_6$ (s = 2,12%; c = 38,09%)
SE idade <= 0,26 E sexo ≠ 0 E flag_tel_residencial = 0
ENTÃO cliente é inadimplente

Regra$_7$ (s = 9,77%; c = 79,79%)
SE idade <= 0,26 E sexo ≠ 0 E flag_tel_residencial ≠ 0 E flag_mesma_cidade_residencia_comercial ≠ 0
ENTÃO cliente é adimplente

Regra$_8$ (s = 1,58%; c = 81,79%)
SE 0,09 < idade <= 0,26 E sexo ≠ 0 E flag_tel_residencial ≠ 0 E
flag_mesma_cidade_residencia_comercial = 0 E estado_civil = 5
ENTÃO cliente é adimplente

Regra$_9$ (s = 1,61%; c = 30,65%)
SE idade <= 0,09 E sexo ≠ 0 E flag_tel_residencial ≠ 0 E flag_mesma_cidade_residencia_comercial = 0
ENTÃO cliente é inadimplente

Regra$_{10}$ (s = 0,44%; c = 81,99%)
SE 0,09 < idade <= 0,26 E sexo ≠ 0 E flag_tel_residencial ≠ 0 E
flag_mesma_cidade_residencia_comercial = 0 E estado_civil ≠ 5 E numero_loja = 28
ENTÃO cliente é adimplente

Regra$_{11}$ (s = 0,03%; c = 12,31%)
SE 0,09 < idade <= 0,26 E sexo ≠ 0 E flag_tel_residencial ≠ 0 E
flag_mesma_cidade_residencia_comercial = 0 E estado_civil ≠ 5 E numero_loja ≠ 28 E
cidade_residencia = 16
ENTÃO cliente é inadimplente

Regra$_{12}$ (s = 0,11%; c = 77,27%)
SE 0,09 < idade <= 0,26 E sexo ≠ 0 E flag_tel_residencial ≠ 0 E
flag_mesma_cidade_residencia_comercial = 0 E estado_civil ≠ 5 E numero_loja ≠ 28 E
cidade_residencia ≠ 16 E ddd_residencial = 3
ENTÃO cliente é adimplente

Regra$_{13}$ (s = 0,78%; c = 26,26%)
SE 0,09 < idade <= 0,26 E sexo ≠ 0 E flag_tel_residencial ≠ 0 E
flag_mesma_cidade_residencia_comercial = 0 E estado_civil ≠ 5 E numero_loja ≠ 28 E
cidade_residencia ≠ 16 E ddd_residencial ≠ 3
ENTÃO cliente é inadimplente

Do total de treze regras, sete estão associadas à classe adimplente e seis à classe inadimplente. Como a árvore de decisão apresentou uma taxa de classificação superior para a classe adimplente, as regras associadas à esta classe apresentaram confiança elevada, variando de 76,89% a 87,16%. As regras associadas à classe inadimplente apresentaram suporte baixo porque o número de registros da base que pertencem a esta classe é inferior (19,83%). A *Regra$_2$* apresentou maior suporte (39,99%) e confiança (87,16%). Esta regra indica, com o nível de confiança de 87,16%, que os clientes com idade maior que 0,26 que possuem telefone residencial são adimplentes. Outras descobertas interessantes podem ser observadas analisando o conjunto de regras especificado na Tabela 7.32.

Análise das Técnicas de Extração de Conhecimento Simbólico

A compreensibilidade de um conjunto de regras está relacionada com a facilidade de interpretação das regras. Portanto, a compreensibilidade pode ser estimada, por exemplo, pelo número de regras e condições por regra. Neste caso, quanto menor a quantidade de regras e condições por regra, maior será a compreensibilidade das regras descobertas [Rezende et al. 2003]. Baseando-se neste critério, na precisão e facilidade de aplicação das regras, esta seção faz uma análise das técnicas de extração de conhecimento simbólico de RNA investigadas nesta dissertação.

A extração de regras do modelo FWD produz um conjunto de regras bastante pequeno, fornecendo uma regra por classe. Através da técnica de seleção de atributos, a parte antecedente da regra torna-se menor, produzindo uma representação mais compacta do problema. Apesar da remoção dos atributos irrelevantes contribuir para geração de regras mais simples, a técnica de extração de regras da rede FWD pode produzir regras extensas e ininteligíveis quando aplicada em problemas de alta dimensionalidade, como é o caso do problema de análise de crédito investigado nesta dissertação, pois todos os atributos relevantes devem estar presentes em todas as regras. Como este modelo não apresenta graus de importância e certeza associados às condições e conclusões, não é possível, após a seleção de atributos, reduzir ainda mais o número de condições por regra. Outro problema observado neste modelo é a representação lingüística de atributos booleanos. [Li et al. 2002] sugerem que o termo lingüístico associado a cada atributo seja expresso da seguinte forma: *aproximadamente conexão_de_memória_associada_ao_atributo_e_classe*. Deste modo, todos os atributos de entrada são tratados como sendo numéricos. Como resultado, a representação semântica dos atributos booleanos é prejudicada, pois normalmente este tipo de atributo é representado por duas funções de pertinência, uma representando o valor *verdadeiro* e outra representando o valor *falso*. Com relação à precisão, as regras apresentam performance equivalente à da rede FWD.

Com relação à extração de regras do modelo FuNN, a técnica REFuNN extraiu cinco regras e a técnica AREFuNN extraiu nove. Apesar do conjunto de regras extraído pela técnica AREFuNN ter sido maior do que o conjunto extraído pela técnica REFuNN, as regras obtidas pela técnica AREFuNN são mais simples e fáceis de compreender, pois o número de condições por regra é menor. Embora estas técnicas utilizem limiares para simplificar as regras, os conjuntos de regras extraídos da rede FuNN são maiores do que o conjunto extraído da rede FWD. Esta característica não é uma desvantagem para as técnicas do modelo FuNN, pois não é suficiente que o conjunto de regras seja pequeno. Para facilitar a compreensão do conhecimento extraído, também é necessário que o número de condições por regra seja pequeno, o que não ocorre nas regras extraídas da rede FWD. Com relação à precisão, os conjuntos de regras extraídos pelas técnicas REFuNN e AREFuNN apresentaram precisão inferior à classificação realizada diretamente pela rede FuNN da qual as regras foram extraídas. No entanto, a precisão das regras pode ser melhorada através da utilização de limiares menores. Deste modo, as regras passam a ter maior precisão, mas se tornam mais complexas. Comparando as técnicas REFuNN e AREFuNN, o conjunto de regras extraído por AREFuNN apresentou melhor precisão. O conjunto de regras resultante da união das regras extraídas pelas duas técnicas também foi investigado e apresentou melhor precisão.

O conhecimento extraído pela técnica TREPAN pode ser representado de duas formas: árvore de decisão e regras *Se-Então*. Inicialmente, TREPAN extrai uma árvore de decisão. A principal vantagem desta forma de representação é a capacidade de apresentar o conhecimento graficamente, facilitando a compreensão e aplicação do mesmo. A partir da árvore de decisão é possível extrair um conjunto de regras que representam os possíveis caminhos da árvore. No problema investigado, foi obtida uma árvore pequena (12 nós internos) de fácil interpretação. A partir desta árvore foram geradas 13 regras. Apesar do conjunto de regras extraído pela técnica TREPAN ter sido maior do que o conjunto extraído pelas demais técnicas, o número de condições por regra não é grande e a aplicação das regras é bastante direta. Com relação à precisão, TREPAN apresentou um desempenho aproximadamente igual ao da rede MLP da qual a árvore de decisão foi extraída.

Embora a Lógica *Fuzzy* ofereça vantagens lingüísticas, as regras *fuzzy* não são tão facilmente interpretadas e diretamente aplicadas quanto às regras *Se-Então* clássicas. A interpretação das regras *fuzzy* pode ser facilitada com o auxílio de técnicas visuais que representem as funções de pertinência das condições e conclusões das regras e o mecanismo de inferência *fuzzy* decorrente da aplicação de um exemplo ao sistema.

7.5. Soluções Neurais Híbridas para o Processo de KDD

Analisando os resultados obtidos, é possível identificar as vantagens e desvantagens dos modelos e técnicas investigados quando aplicados ao problema de análise de crédito.

Apesar de ter apresentado um desempenho (taxa de classificação) aproximadamente igual ao dos modelos MLP e FuNN, o modelo FWD possui uma grande limitação: capacidade de resolver apenas problemas linearmente separáveis. Além desta limitação, as regras extraídas de uma rede FWD são bastante extensas, o que dificulta o entendimento do conhecimento minerado. A funcionalidade do modelo FWD que tem se destacado bastante é a seleção de atributos. Em [Li et al. 2002] 50% dos atributos foram removidos. Em [Amorim et al. 2003], a rede foi validada com duas bases de dados

médicas. Na primeira base, 11% dos atributos foram removidos. Na segunda base, 50%. Nesta dissertação, que lida com uma base de dados extensa e de alta dimensionalidade, 61% dos atributos foram removidos. Estes resultados experimentais demonstram a efetividade e a viabilidade prática da técnica de seleção de atributos do modelo FWD. Portanto, este modelo é bastante promissor para ser aplicado na etapa de seleção de dados do processo de KDD.

Ao contrário da rede FWD, os modelos MLP e FuNN são capazes de resolver problemas não linearmente separáveis. Por esta razão, estes modelos são mais adequados para serem aplicados na etapa de mineração de dados.

Com relação à extração de regras do modelo FuNN, a técnica AREFuNN se destacou mais do que a técnica REFuNN, apresentando regras menores e mais precisas. No entanto, quando os conjuntos de regras extraídos pelas duas técnicas são agrupados, a precisão é melhorada. A interpretação dessas regras pode ser facilitada a partir da aplicação de técnicas visuais que representem as funções de pertinência das condições e conclusões das regras e o mecanismo de inferência *fuzzy* decorrente da aplicação de um exemplo ao sistema.

A técnica TREPAN apresentou desempenho (taxa de classificação) aproximadamente igual ao da rede MLP e a fidelidade da árvore de decisão extraída foi bastante representativa (85,78%). Com relação à seleção de atributos realizada com o auxílio da árvore de decisão, apesar de ter ocorrido um decréscimo na média da taxa de classificação na maioria dos modelos, tal técnica mostrou resultados satisfatórios quando aplicada aos modelos FuNN e MLP. Considerando o aspecto de extração de conhecimento, TREPAN gerou uma árvore de decisão bastante simples. Como resultado, a interpretação e aplicação direta do conhecimento extraído foram facilitadas. Portanto, TREPAN pode ser usada nas etapas de apresentação do conhecimento e seleção de dados.

A partir da constatação das vantagens e desvantagens de cada modelo e técnica, foi possível propor duas soluções neurais híbridas para o processo de KDD utilizando os modelos e técnicas investigados. As Figuras 7.17 e 7.18 mostram as soluções propostas.

Na primeira solução (Figura 7.17), a técnica de seleção de atributos da rede FWD é usada na etapa de seleção de dados. Após a identificação dos atributos relevantes, aplica-se a base reduzida (sem os atributos irrelevantes) à rede FuNN. Quando o treinamento da rede FuNN é finalizado, as técnicas de extração de regras REFuNN e AREFuNN podem ser utilizadas de duas formas: aplicar a técnica AREFuNN isoladamente ou em conjunto com a técnica REFuNN. O objetivo desta fase é apresentar o conhecimento incorporado pela rede na forma de regras *fuzzy* Se-Então.

Como a base de dados resultante do processo de seleção de atributos da rede FWD não pode ser mapeada para a base de dados utilizada pela técnica TREPAN, única técnica investigada nesta dissertação que extrai conhecimento de uma rede MLP, o modelo MLP não é aplicado na etapa de mineração de dados desta solução.

Na segunda solução (Figura 7.18), a técnica de seleção de atributos utilizando a árvore de decisão extraída por TREPAN é usada na etapa de seleção de dados. Como esta técnica mostrou resultados melhores para os modelos MLP e FuNN, estes modelos são utilizados na etapa de mineração de dados. Outro motivo que levou à aplicação dos modelos MLP e FuNN na etapa de mineração de dados é a capacidade destes modelos resolver problemas não linearmente separáveis. Na etapa de apresentação do conhecimento, quatro alternativas podem ser aplicadas. Como TREPAN não faz

nenhuma restrição com relação à arquitetura da rede ou ao algoritmo de treinamento aplicado, esta técnica pode ser usada em conjunto com a rede FuNN ou MLP. Outras possibilidades para a rede FuNN é a aplicação das técnicas de extração de regras REFuNN e AREFuNN. Estas técnicas podem ser utilizadas de duas formas: aplicar a técnica AREFuNN isoladamente ou em conjunto com a técnica REFuNN.

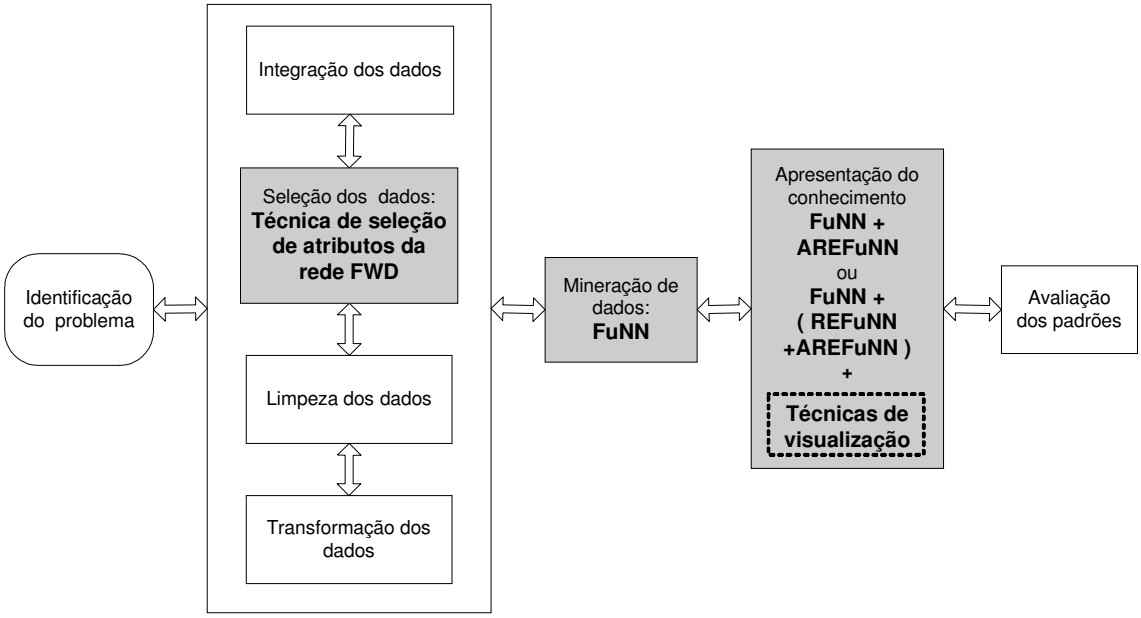

Figura 7.17 - Primeira solução neural híbrida para o processo de KDD

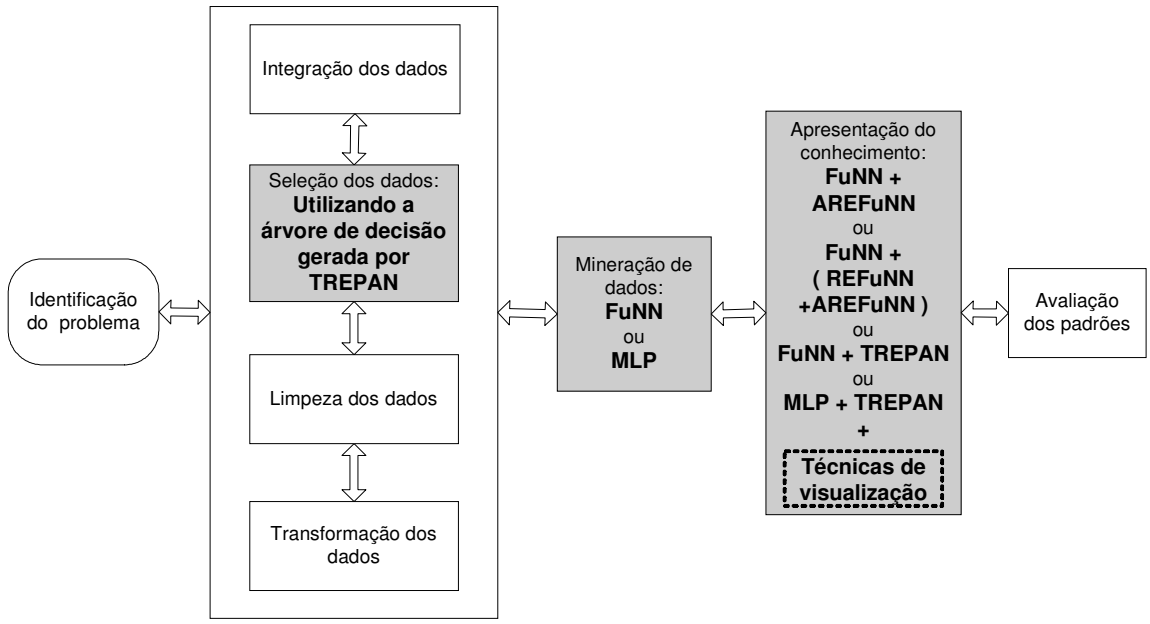

Figura 7.18 - Segunda solução neural híbrida para o processo de KDD

Nas duas soluções propostas, técnicas de visualização podem ser aplicadas na etapa de apresentação do conhecimento com o objetivo de representar as funções de pertinência das condições e conclusões das regras *fuzzy* e o mecanismo de inferência *fuzzy* decorrente da aplicação de um exemplo ao sistema. Desta forma, a interpretação das regras *fuzzy* seria facilitada. As técnicas de visualização [Fayyad et al. 2001] também podem ser aplicadas para representar as árvores de decisão e as regras *fuzzy* e clássicas sobre diversas perspectivas (Ex.: confiança, suporte e outras medidas de interesse).

As etapas das soluções neurais híbridas propostas nesta dissertação para o processo de KDD diferem das etapas descritas por [Han & Kamber 2001] em dois aspectos: adição de uma nova etapa (identificação do problema) e realização da etapa de avaliação de padrões depois da etapa de apresentação do conhecimento. A primeira alteração foi realizada porque é muito importante, nas aplicações de KDD, estabelecer os objetivos que devem ser alcançados e compreender o domínio do problema que será investigado. A segunda alteração foi realizada porque as técnicas de extração de conhecimento simbólico de RNA e as técnicas de visualização são aplicadas apenas na etapa de apresentação do conhecimento, pois a capacidade de geração de conhecimento compreensível não é inerente aos modelos neurais aplicados na etapa de mineração de dados. Portanto, não é possível realizar a avaliação de padrões antes do conhecimento minerado ter sido apresentado numa forma compreensível.

7.6. Considerações Finais

Este capítulo apresentou os resultados da investigação experimental realizada com a rede MLP; o modelo neuro-*fuzzy* FWD e sua técnica de extração de regras; o modelo neuro-*fuzzy* FuNN e suas técnicas de extração de regras REFuNN e AREFuNN; e a técnica TREPAN, usada em conjunto com a rede MLP da qual uma árvore de decisão é extraída. Os modelos foram avaliados e comparados considerando os seguintes aspectos: desempenho em relação à generalização e capacidade de gerar conhecimento compreensível através de suas técnicas de extração de regras. O desempenho dos classificadores em relação à generalização foi avaliado observando as taxas de classificação no conjunto de teste (total e por classe), analisando as curvas ROC e o impacto das decisões dos classificadores no contexto específico da aplicação investigada (análise de crédito ao consumidor). A compreensibilidade do conhecimento extraído foi avaliada analisando a facilidade de interpretação e aplicação do conhecimento descoberto. Outro aspecto considerado na avaliação do conhecimento extraído foi a precisão. Além da análise nas etapas de mineração de dados e apresentação do conhecimento, também foram investigadas duas técnicas de seleção de atributos: a técnica da rede FWD e através da árvore de decisão gerada pela técnica TREPAN. Ao final da investigação, considerando as vantagens de cada modelo e técnica, foram propostas duas soluções neurais híbridas para o processo de KDD.

Os resultados obtidos comprovaram que várias etapas do processo de KDD podem ser realizadas através da aplicação de Sistemas Neurais Híbridos e a integração de RNA com outras técnicas que utilizam representação simbólica, como Sistemas *Fuzzy* e algoritmos simbólicos convencionais, contribui para uma maior aceitação das RNA como uma alternativa bastante viável para ser aplicada no processo de KDD.

A maior dificuldade encontrada na execução do estudo de caso foi o tempo requerido para realizar os experimentos. Além de utilizar uma base de dados extensa e de alta dimensionalidade, foram investigados três modelos neurais (MLP, FWD e FuNN), quatro técnicas de extração de conhecimento simbólico (TREPAN, da rede FWD, REFuNN e AREFuNN) e duas técnicas de seleção de atributos (da rede FWD e através da árvore de decisão gerada por TREPAN). Entre os modelos neurais, o modelo que em geral apresentou menor tempo de treinamento foi a rede MLP. Isto pode ser resultado dos demais modelos apresentarem mais camadas intermediárias, o que requer mais processamento. Na maioria das configurações investigadas, o treinamento da rede FWD foi finalizado apenas no número máximo de épocas (3000). Por outro lado, no geral, o treinamento da rede FuNN foi interrompido antes da época 1000. A técnica TREPAN apresentou dois problemas. Quando o número máximo de nós internos ou o tamanho mínimo das amostras eram grandes, o tempo necessário para geração da árvore era estendido. Além disso, em algumas configurações a geração da árvore não foi concluída. Como TREPAN armazena os exemplos de treinamento e artificiais que alcançam cada nó, a memória requerida é bastante extensa, por isso, em alguns casos, foi gerada exceção de memória indisponível.

Capítulo 8
Conclusões

Sistemas Híbridos Inteligentes (SHI) é uma das abordagens de Inteligência Artificial que vem sendo bastante utilizada na resolução de problemas onde o emprego de uma única técnica não é suficiente para obter resultados satisfatórios. Tais sistemas se inspiram na integração de duas ou mais técnicas inteligentes com o intuito de suprir as limitações de cada técnica.

Como resultado da rápida disseminação dos SHI, no contexto de RNA, um dos temas de pesquisa mais investigados atualmente é o desenvolvimento de Sistemas Neurais Híbridos (SNH). O principal foco de pesquisa em SNH tem sido o de combinar RNA, técnica fortemente baseada em dados, com técnicas que utilizam representação simbólica, como Sistemas *Fuzzy*, algoritmos simbólicos convencionais, Raciocínio Baseado em Casos e Sistemas Tutores.

A proposta desta dissertação foi inspirada no crescente interesse nos SNH que combinam RNA com Sistemas *Fuzzy* e algoritmos simbólicos convencionais. Esta integração permite que o conhecimento incorporado pela rede seja representado numa forma compreensível, tornando as RNA mais adequadas para o processo de KDD, principalmente em aplicações onde a apresentação do conhecimento minerado em um formato de fácil interpretação é um aspecto crucial.

Este capítulo apresenta uma análise do trabalho realizado nesta dissertação de acordo com os objetivos propostos no Capítulo 1, as contribuições, as perspectivas de trabalhos futuros e as considerações finais sobre o trabalho desenvolvido.

8.1. Objetivos Propostos e Resultados Alcançados

Os objetivos propostos inicialmente foram:

- Investigar o paradigma dos Sistemas Neuro-*Fuzzy* e as técnicas de extração de conhecimento simbólico de RNA como uma alternativa para tornar as RNA mais adequadas ao processo de KDD;
- Como resultado da investigação, modelar e implementar uma ferramenta de *software*, a *Neural Mining*, que integre, em um único ambiente, os modelos neuro-*fuzzy* e as técnicas de extração de conhecimento simbólico de RNA mais promissores;
- Aplicar um problema de larga escala no domínio de análise de crédito como ambiente de teste para a ferramenta desenvolvida e os algoritmos investigados;

- ♣ Comparar os resultados alcançados por cada método e técnica, e identificar suas vantagens e deficiências quando aplicados ao processo de KDD em problemas de larga escala;
- ♣ Propor soluções neurais híbridas para o processo de KDD.

As próximas seções analisam os resultados alcançados em relação a cada um dos objetivos propostos.

8.1.1. Investigação dos Sistemas Neurais Híbridos

Uma parte considerável deste trabalho foi dedicada ao estudo de KDD, RNA, Lógica *Fuzzy*, Sistemas *Fuzzy*, Sistemas Neuro-*Fuzzy* e técnicas de extração de conhecimento simbólico de RNA. Durante a execução desta fase, foi realizada uma investigação teórica das principais características, vantagens e deficiências dos modelos neuro-*fuzzy* e das técnicas de extração de conhecimento simbólico de RNA já desenvolvidos. Esta investigação foi realizada com o objetivo de identificar os modelos e técnicas mais promissores para serem implementados na ferramenta *Neural Mining* e investigados de forma mais aprofundada nesta dissertação.

O resultado deste estudo está relatado nos capítulos iniciais da dissertação. O Capítulo 1 apresenta uma visão geral de KDD, dos Sistemas Neuro-*Fuzzy* e das técnicas de extração de conhecimento simbólico de RNA, e as motivações, objetivos e justificativas desta dissertação. O Capítulo 2 descreve detalhadamente todas as etapas do processo de KDD e destaca algumas das principais áreas de aplicação de KDD. O Capítulo 3 aborda os principais aspectos referentes às RNA e a viabilidade de aplicá-las no processo de KDD. O Capítulo 4 aborda os Sistemas *Fuzzy* e Neuro-*Fuzzy*. Este capítulo também apresenta o estado da arte e uma breve descrição da perspectiva histórica dos Sistemas Neuro-*Fuzzy*, enfatizando os modelos FWD e FuNN. O Capítulo 5 aborda os principais aspectos sobre as técnicas de extração de conhecimento simbólico de RNA e descreve detalhadamente o algoritmo TREPAN e as técnicas específicas dos modelos FuNN e FWD.

8.1.2. Ferramenta *Neural Mining*

A proposta desta dissertação vai além do estudo do processo de KDD, dos Sistemas Neuro-*Fuzzy* e das técnicas de extração de conhecimento simbólico de RNA. Este trabalho também se propôs a desenvolver uma ferramenta, *Neural Mining*, responsável por capturar os dados já consolidados, aplicar algoritmos de mineração baseados na abordagem neural híbrida e apresentar o conhecimento descoberto em forma de regras. O desenvolvimento desta ferramenta foi uma das etapas que consumiu mais tempo durante a execução deste trabalho.

Os modelos abordados pela ferramenta são: a rede MLP com o algoritmo de aprendizagem *Backpropagation* e as redes neuro-*fuzzy* FWD e FuNN. Com relação às técnicas de extração de conhecimento simbólico, foram implementadas as técnicas AREFuNN e REFuNN, que extraem regras *fuzzy* de uma rede FuNN; a técnica do modelo FWD, que extrai regras *fuzzy*; e a técnica TREPAN, que extrai uma árvore de decisão de qualquer modelo.

O modelo MLP foi escolhido para ser aplicado em dois contextos: em conjunto com a técnica TREPAN e, separadamente, por ser um modelo tradicional, para comparar sua performance com os modelos neuro-*fuzzy* FWD e FuNN.

A motivação para investigar os modelos FWD e FuNN se baseou no fato destes métodos serem bastante promissores para serem aplicados ao processo de KDD. A rede FWD apresenta uma característica inovadora de englobar, dentro de uma mesma estrutura, três etapas do processo de KDD: seleção de dados, mineração de dados e apresentação do conhecimento. A investigação da rede FWD também foi motivada pelo fato de que existem poucos resultados experimentais demonstrando sua viabilidade em aplicações reais. O modelo FuNN foi escolhido por apresentar as seguintes características: permite o uso de limiares no processo de extração de regras, possibilitando a redução do número de regras e condições nos antecedentes das regras; permite a inserção e o refinamento de regras usadas para definir a arquitetura inicial da rede, possibilitando o uso de duas fontes de dados (base de dados e regras iniciais); e tem apresentado resultados satisfatórios em diversos domínios de aplicação.

As técnicas de extração de regras do modelo FuNN (AREFuNN e REFuNN) e do modelo FWD foram implementadas por possibilitarem a representação do conhecimento incorporado por estes modelos na forma de regras *fuzzy* Se-Então.

A técnica TREPAN foi escolhida por apresentar as seguintes vantagens: permite representação do conhecimento numa forma compreensível; pode ser utilizada para aplicações envolvendo tanto atributos discretos como contínuos; possui a capacidade de produzir árvores de decisão sucintas de redes grandes aplicadas em domínios de larga escala; é capaz de produzir árvores de decisão precisas, compreensíveis e com alto nível de fidelidade; é escalável em relação ao tamanho da base de dados, complexidade dos modelos aprendidos e tempo de execução (o usuário tem controle sobre a complexidade da árvore retornada); e é genérico em sua aplicabilidade (não requer a utilização de um procedimento especial de treinamento e não impõe restrições sobre o modelo utilizado como oráculo).

A ferramenta *Neural Mining* foi desenvolvida em Java e a interface foi implementada utilizando o pacote *Swing*. Devido às facilidades oferecidas pela linguagem Java, foram aplicados fundamentos da programação orientada a objetos, o que possibilitou a utilização dos conceitos de herança, encapsulamento e reusabilidade de código.

O Capítulo 6 descreve a ferramenta *Neural Mining*, apresentando sua estrutura modular, os modelos e técnicas abordados e sua interface gráfica. O formato dos arquivos utilizados pela ferramenta para armazenar as bases de dados, as configurações dos modelos e técnicas, e os resultados da classificação e extração de conhecimento são descritos no Apêndice B.

8.1.3. Estudo de Caso Investigado

O estudo de caso desta dissertação, descrito no Capítulo 7, compreendeu uma investigação experimental bastante detalhada dos modelos e técnicas implementados na ferramenta *Neural Mining*. O objetivo desta etapa foi aplicar um problema de larga escala como ambiente de teste para a ferramenta desenvolvida e os algoritmos investigados (os modelos neurais MLP, FWD e FuNN; as técnicas de extração de conhecimento simbólico TREPAN, REFuNN, AREFuNN e da rede FWD; e as técnicas de seleção de atributos da rede FWD e através da árvore de decisão gerada por TREPAN).

O estudo de caso abrangeu todas as etapas do processo de KDD. As etapas de mineração de dados e apresentação do conhecimento foram executadas na ferramenta *Neural Mining*.

O domínio aplicado como estudo de caso foi o de análise de crédito ao consumidor, um problema de classificação que define a aprovação ou não de crédito a um determinado solicitante, considerando suas características pessoais e financeiras. Este domínio foi escolhido por se tratar de um problema de larga escala, envolver dados reais com múltiplos atributos relacionados e ser de interesse de diversas instituições e empresas. Tais características permitiram a verificação da viabilidade prática dos modelos e técnicas implementados na ferramenta.

A base de dados utilizada no estudo de caso era composta de 60.141 registros, dos quais 11.923 (19,83%) são classificados como inadimplentes e 48.218 (80,17%) como adimplentes. Esta base de dados possui os dados fornecidos pelos solicitantes no momento da solicitação de crédito a uma operadora de cartões de crédito no Brasil. Essas informações são utilizadas pela empresa para decidir pela concessão ou não de crédito ao solicitante. Todos os clientes que obtiveram a aprovação do crédito foram armazenados na base. Com o passar do tempo, alguns desses clientes, que foram considerados bons pagadores pelo sistema decisório da operadora, se tornaram maus pagadores. Portanto, o problema aplicado ao estudo de caso contém informações parciais, pois a base de dados possui apenas informações a respeito dos proponentes aceitos e que vieram a se tornar adimplentes ou inadimplentes na carteira de clientes da empresa. Tal característica tornou o problema ainda mais complexo.

A maior dificuldade encontrada na execução do estudo de caso foi o tempo requerido para realizar os experimentos. Além de utilizar uma base de dados extensa e de alta dimensionalidade, foram investigados três modelos neurais (MLP, FWD e FuNN), quatro técnicas de extração de conhecimento simbólico (TREPAN, da rede FWD, REFuNN e AREFuNN) e duas técnicas de seleção de atributos (da rede FWD e através da árvore de decisão gerada por TREPAN). Entre os modelos neurais, o modelo que em geral apresentou menor tempo de treinamento foi a rede MLP. Isto pode ser resultado dos demais modelos apresentarem mais camadas intermediárias, o que requer mais processamento. Na maioria das configurações investigadas, o treinamento da rede FWD foi finalizado apenas no número máximo de épocas (3000). Por outro lado, no geral, o treinamento da rede FuNN foi interrompido antes da época 1000. A técnica TREPAN apresentou dois problemas. Quando o número mínimo de amostras ou o número máximo de nós internos eram grandes, o tempo necessário para geração da árvore era estendido. Além disso, em determinadas configurações, a geração da árvore de decisão não foi finalizada. Como TREPAN armazena os exemplos de treinamento e artificiais que alcançam cada nó, a memória requerida é bastante extensa, por isso, em alguns casos, foi gerada exceção de memória indisponível.

8.1.4. Análise dos Resultados Experimentais

Os modelos foram avaliados e comparados considerando os seguintes aspectos: desempenho em relação à generalização e capacidade de gerar conhecimento compreensível através de suas técnicas de extração de regras.

O desempenho dos classificadores em relação à generalização foi avaliado observando as taxas de classificação no conjunto de teste (total e por classe), analisando as curvas ROC e o impacto das decisões dos classificadores no contexto específico da aplicação investigada (análise de crédito ao consumidor), utilizando os gráficos de massa mantida. As curvas ROC permitiram uma comparação do

comportamento dos modelos neurais independentemente dos custos associados aos tipos de erro e da distribuição das classes associadas ao problema. Os gráficos de massa mantida possibilitaram uma comparação do desempenho dos modelos neurais entre si e com relação à classificação realizada pela operada de crédito.

A compreensibilidade do conhecimento extraído foi avaliada analisando a facilidade de interpretação e aplicação do conhecimento descoberto. Outro aspecto considerado na avaliação do conhecimento extraído foi a precisão.

Além da análise nas etapas de mineração de dados e apresentação do conhecimento, também foram investigadas duas técnicas de seleção de atributos: a técnica da rede FWD e através da árvore de decisão gerada por TREPAN.

Capacidade de Generalização

Os modelos neurais apresentaram resultados bastante aproximados quando considerando as taxas de classificação (total e por classe) no conjunto de teste. A maior diferença ocorreu na taxa de classificação da classe inadimplente. Neste caso, os modelos MLP e FuNN apresentaram performance superior ao modelo FWD.

Analisando as curvas ROC geradas a partir dos resultados dos modelos neurais, foi possível observar que em nenhuma das regiões o modelo FWD apresentou desempenho melhor que os modelos MLP e FuNN.

Considerando os gráficos de massa mantida, observou-se que em nenhuma das regiões dos gráficos, a curva de redução da inadimplência apareceu numa posição acima da curva de massa mantida. Portanto, os classificadores apresentam desempenho superior ou equivalente ao critério utilizado pela empresa para classificar os clientes como bons ou maus pagadores. Estes resultados demonstraram que a aplicação desses modelos ao processo de decisão da empresa conduziria a um melhor retorno financeiro e maior satisfação para os clientes. Comparando os modelos entre si, foi possível observar que em nenhuma das regiões do gráfico o modelo FuNN apresentou massa mantida e redução da taxa de inadimplência superior aos modelos MLP e FWD.

A árvore de decisão gerada pela técnica TREPAN apresentou taxas de classificação (total e por classe) no conjunto de teste bastante aproximadas das taxas da rede MLP da qual a árvore foi extraída. Além disso, a taxa de fidelidade foi bastante representativa (85,78%).

Conhecimento Extraído

A extração de regras do modelo FWD produziu um conjunto de regras bastante pequeno, fornecendo uma regra por classe. Através da técnica de seleção de atributos, a parte antecedente da regra tornou-se menor, produzindo uma representação mais compacta do problema. Apesar da remoção dos atributos irrelevantes contribuir para geração de regras mais simples, a técnica de extração de regras da rede FWD pode produzir regras extensas quando aplicada em problemas de alta dimensionalidade, como é o caso do problema de análise de crédito investigado nesta dissertação, pois todos os atributos relevantes devem estar presentes em todas as regras. Como este modelo não apresenta graus de importância e certeza associados às condições e conclusões, não é possível, após a seleção de atributos, reduzir ainda mais o número de condições por regra. Outro problema observado neste

modelo é a representação lingüística de atributos booleanos. [Li et al. 2002] sugerem que o termo lingüístico associado a cada atributo seja expresso da seguinte forma: *aproximadamente conexão_de_memória_associada_ao_atributo_e_classe*. Deste modo, todos os atributos de entrada são tratados como sendo numéricos. Como resultado, a representação semântica dos atributos booleanos é prejudicada, pois normalmente este tipo de atributo é representado por duas funções de pertinência, uma representando o valor *verdadeiro* e outra representando o valor *falso*. Com relação à precisão, as regras apresentaram performance equivalente à da rede FWD (67,32%).

Com relação à extração de regras do modelo FuNN, a técnica REFuNN extraiu cinco regras e a técnica AREFuNN extraiu nove. Apesar do conjunto de regras extraído pela técnica AREFuNN ter sido maior do que o extraído pela técnica REFuNN, as regras obtidas pela técnica AREFuNN são mais simples e fáceis de compreender, pois o número de condições por regra é menor. Embora estas técnicas utilizem limiares para simplificar as regras, os conjuntos de regras extraídos da rede FuNN são maiores do que o conjunto extraído da rede FWD. Esta característica não é uma desvantagem para as técnicas do modelo FuNN, pois não é suficiente que o conjunto de regras seja pequeno. Para facilitar a compreensão do conhecimento extraído, também é necessário que o número de condições por regra seja pequeno, o que não ocorre nas regras extraídas da rede FWD. Com relação à precisão, os conjuntos de regras extraídos pelas técnicas REFuNN e AREFuNN apresentaram precisão inferior à classificação realizada diretamente pela rede FuNN da qual as regras foram extraídas. No entanto, a precisão das regras pode ser melhorada através da utilização de limiares menores. Deste modo, as regras passam a ter maior precisão, mas se tornam mais complexas. Comparando as técnicas REFuNN e AREFuNN, o conjunto de regras extraído por AREFuNN apresentou melhor precisão (45,10% para a técnica AREFuNN e 25,10% para a técnica REFuNN). O conjunto de regras resultante da união das regras extraídas pelas duas técnicas também foi investigado e apresentou melhor precisão (49,60%).

O conhecimento extraído pela técnica TREPAN pode ser representado de duas formas: árvore de decisão e regras *Se-Então*. Inicialmente, TREPAN extrai uma árvore de decisão. A principal vantagem desta forma de representação é a capacidade de apresentar o conhecimento graficamente, facilitando a compreensão e aplicação do mesmo. A partir da árvore de decisão é possível extrair um conjunto de regras que representam os possíveis caminhos da árvore. No problema investigado, foi obtida uma árvore pequena (12 nós internos) de fácil interpretação. A partir desta árvore foram geradas 13 regras. Apesar do conjunto de regras extraído pela técnica TREPAN ter sido maior do que o conjunto extraído pelas demais técnicas, o número de condições por regra não é grande e a aplicação das regras é bastante direta. Com relação à precisão, TREPAN apresentou um desempenho aproximadamente igual ao da rede MLP da qual a árvore de decisão foi extraída (66,51% para a técnica TREPAN e 68,22% para a rede MLP).

Embora a Lógica *Fuzzy* ofereça vantagens lingüísticas, as regras *fuzzy* não são tão facilmente interpretadas e diretamente aplicadas quanto às regras *Se-Então* clássicas. A interpretação das regras *fuzzy* pode ser facilitada com o auxílio de técnicas visuais que representem as funções de pertinência das condições e conclusões das regras e o mecanismo de inferência *fuzzy* decorrente da aplicação de um exemplo ao sistema.

8.1.5. Soluções Neurais Híbridas para o Processo de KDD

Ao final da investigação experimental, considerando as vantagens de cada modelo e técnica quando aplicados a um problema de larga escala, foram propostas duas soluções neurais híbridas para o processo de KDD.

Na primeira solução, a técnica de seleção de atributos da rede FWD é usada na etapa de seleção de dados. Após a identificação dos atributos relevantes, aplica-se a base reduzida (sem os atributos irrelevantes) à rede FuNN. Quando o treinamento da rede FuNN é finalizado, as técnicas de extração de regras REFuNN e AREFuNN podem ser utilizadas de duas formas: aplicar a técnica AREFuNN isoladamente ou em conjunto com a técnica REFuNN. O objetivo desta fase é apresentar o conhecimento incorporado pela rede na forma de regras *fuzzy* Se-Então. Como a base de dados resultante do processo de seleção de atributos da rede FWD não pode ser mapeada para a base de dados utilizada por TREPAN, única técnica investigada nesta dissertação que extrai conhecimento de uma rede MLP, o modelo MLP não é aplicado na etapa de mineração de dados desta solução.

Na segunda solução, a técnica de seleção de atributos utilizando a árvore de decisão gerada por TREPAN é usada na etapa de seleção de dados. Como esta técnica mostrou resultados melhores para os modelos MLP e FuNN, estes modelos são utilizados na etapa de mineração de dados. Outro motivo que levou à aplicação dos modelos MLP e FuNN na etapa de mineração de dados é a capacidade destes modelos resolver problemas não linearmente separáveis. Na etapa de apresentação do conhecimento, quatro alternativas podem ser aplicadas. Como TREPAN não faz nenhuma restrição com relação à arquitetura da rede ou ao algoritmo de treinamento aplicado, esta técnica pode ser usada em conjunto com a rede FuNN ou MLP. Outras possibilidades para a rede FuNN é a aplicação das técnicas de extração de regras REFuNN e AREFuNN. Estas técnicas podem ser utilizadas duas formas: aplicar a técnica AREFuNN isoladamente ou em conjunto com a técnica REFuNN.

Nas duas soluções propostas, técnicas de visualização podem ser aplicadas na etapa de apresentação do conhecimento com o objetivo de representar as funções de pertinência das condições e conclusões das regras *fuzzy* e o mecanismo de inferência *fuzzy* decorrente da aplicação de um exemplo ao sistema. Desta forma, a interpretação das regras *fuzzy* seria facilitada. As técnicas de visualização também podem ser aplicadas para representar as árvores de decisão e as regras *fuzzy* e clássicas sobre diversas perspectivas (Ex.: confiança, suporte e outras medidas de interesse).

8.2. Contribuições

As principais contribuições desta dissertação são:

- Pesquisa extensiva sobre KDD, RNA, Lógica *Fuzzy*, Sistemas *Fuzzy*, Sistemas Neuro-*Fuzzy* e técnicas de extração de conhecimento simbólico de RNA.

  Durante a execução desta fase, foram realizadas uma investigação teórica das principais características, vantagens e deficiências dos modelos neuro-*fuzzy* e das técnicas de extração de conhecimento simbólico de RNA já desenvolvidos, e a identificação dos modelos e técnicas mais promissores para serem investigados de forma mais aprofundada nesta dissertação.

  O resultado deste estudo está relatado nos capítulos iniciais da dissertação e pode ser consultado como uma referência nas áreas de KDD e Sistemas Neurais Híbridos;

- Ferramenta *Neural Mining*.

  Embora KDD seja uma área recente com muitos aspectos que ainda precisam ser pesquisados em profundidade, uma grande quantidade de ferramentas de *software* de KDD e mineração de dados já foi desenvolvida. Inicialmente, as ferramentas eram mais utilizadas em ambientes de pesquisa e experimentais. Nos últimos anos, tem-se observado um rápido crescimento de ferramentas sofisticadas voltadas para o meio empresarial. O domínio de aplicação destas ferramentas está constantemente evoluindo devido ao desenvolvimento de novos sistemas, incorporação de novas funcionalidades aos sistemas existentes e proposição de novos métodos de mineração de dados. Apesar dessa evolução, poucas ferramentas têm adotado a abordagem dos Sistemas Neurais Híbridos.

  Uma das principais contribuições desta dissertação é a ferramenta *Neural Mining*, uma ferramenta que abrange as etapas de mineração de dados e apresentação do conhecimento. As principais vantagens da ferramenta *Neural Mining* são a capacidade de integrar vários métodos de aprendizagem neural híbrida num único ambiente e a característica de ter sido projetada para solução de problemas de larga escala. Além disso, *Neural Mining* apresenta uma interface amigável e, por ter sido desenvolvida em Java, possui todas as vantagens de um sistema orientado a objetos, tais como compatibilidade, reusabilidade e fácil manutenção;

- Investigação experimental detalhada de três modelos neurais (MLP, FWD e FuNN).

  Os modelos foram avaliados e comparados considerando o desempenho em relação à capacidade de generalização. A análise foi realizada observando as taxas de classificação no conjunto de teste e avaliando as curvas ROC e o impacto das decisões dos classificadores no contexto específico da aplicação investigada (análise de crédito ao consumidor), utilizando os gráficos de massa mantida. As curvas ROC permitiram uma comparação do comportamento dos modelos neurais independentemente dos custos associados a diferentes tipos de erro e da distribuição das classes associadas ao problema. Os gráficos de massa mantida possibilitaram uma comparação do desempenho dos modelos neurais entre si e com relação à classificação realizada pela operadora de crédito. Portanto, diferente da maioria das avaliações experimentais que avaliam os modelos apenas com relação à precisão da classificação no conjunto de teste, a utilização das curvas ROC e gráficos de massa mantida permitiu a realização de uma análise comparativa mais robusta;

- Investigação experimental detalhada de quatro técnicas de extração de conhecimento simbólico de RNA (TREPAN, da rede FWD, REFuNN e AREFuNN).

  As técnicas de extração de conhecimento simbólico de RNA foram avaliadas considerando os seguintes aspectos: precisão e facilidade de interpretação e aplicação do conhecimento minerado. Esta fase foi importante para identificar as vantagens e deficiências de cada técnica e comparar as regras extraídas por técnicas baseadas na Lógica Booleana (TREPAN) e na Lógica *Fuzzy* (REFuNN, AREFuNN e da rede FWD);

- Investigação experimental detalhada de duas técnicas de seleção de atributos (da rede FWD e através da árvore de decisão gerada por TREPAN).

  Apesar da maioria dos algoritmos de aprendizagem serem projetados para aprender quais atributos são mais apropriados para distinguir as classes de padrões, na prática, a adição de

atributos irrelevantes ou redundantes e a remoção de atributos relevantes podem resultar em soluções de qualidade inferior e retardar o processo de aprendizagem. Embora o especialista do domínio possa escolher os atributos mais relevantes, essa tarefa pode ser difícil e demandar tempo. Outro benefício resultante da seleção de atributos é a obtenção de uma representação mais compacta do conceito alvo, direcionando a atenção do usuário para os atributos mais importantes. Por estas razões, os resultados obtidos a partir da investigação das técnicas de seleção de atributos (a técnica da rede FWD e através da árvore de decisão gerada por TREPAN) são extremamente úteis para solução de problemas de KDD;

- Demonstração da aplicabilidade da ferramenta *Neural Mining* e validação dos modelos e técnicas investigados em um problema real e de larga escala.

  Diferente de muitos trabalhos que utilizam dados artificiais ou problemas simples para validar novos modelos ou comparar modelos já desenvolvidos, a demonstração da aplicabilidade da ferramenta *Neural Mining* e a validação dos modelos e técnicas investigados foram realizadas usando uma base de dados extensa e de alta dimensionalidade. Além disso, o domínio escolhido, o de análise de crédito, representa um problema real que possui múltiplos atributos relacionados. Tais características permitiram a verificação da viabilidade prática da ferramenta *Neural Mining* e dos algoritmos investigados;

- Proposição de soluções neurais híbridas para o processo de KDD.

  A proposição das soluções neurais híbridas para o processo de KDD é uma importante contribuição desta dissertação, pois demonstra que várias etapas do processo de KDD podem ser realizadas através da aplicação de Sistemas Neurais Híbridos e a integração de RNA com outras técnicas que utilizam representação simbólica, como Sistemas *Fuzzy* e algoritmos simbólicos convencionais, faz das RNA uma alternativa bastante viável para ser aplicada no processo de KDD.

8.3. Trabalhos Futuros

Considerando os resultados alcançados nesta dissertação, os seguintes trabalhos futuros podem ser especificados:

- Investigar os modelos e técnicas abordados na ferramenta *Neural Mining* em outros domínios de aplicação;
- Realizar estudos comparativos com outros modelos neuro-*fuzzy* e técnicas de extração de conhecimento simbólico de RNA, tais como os abordados nos Capítulos 4 e 5;
- Analisar mais detalhadamente o mecanismo de geração de exemplos artificiais realizado pela técnica TREPAN, considerando outras formas alternativas para este fim, como por exemplo, imputação de dados [Salinas 1998] e distribuição gaussiana em conjunto com o algoritmo de Box-Müller [Faifer & Janikow 1999];
- Comparar a técnica TREPAN com os algoritmos simbólicos convencionais de geração de regras e árvores de decisão, tais como CART [Breiman et al. 1984], C4.5 [Quilan 1993], C4.5*rules* [Quilan 1993] e CN2 [Clark e Nibeltt 1989];

- Analisar o comportamento dos modelos FWD e FuNN com outros tipos de funções de pertinência. Nos experimentos realizados neste trabalho, foram aplicadas funções de pertinência gaussianas para a rede FWD e funções triangulares para a rede FuNN;
- Propor uma extensão do modelo FWD, a fim de que ele se torne capaz de resolver problemas não linearmente separáveis, gerar mais de uma regra por classe e acomodar mais de uma função de pertinência por atributo de entrada;
- Investigar métodos que permitam a determinação automática do número de regras e funções de pertinência por entrada no modelo FuNN. O algoritmo EDDA (*Extended Dynamic Decay Adjustment*), Algoritmos Genéticos e as técnicas CART (*Classification and Regression Trees*) e *clustering* são exemplos de métodos que podem ser usados para este fim [Jang et al. 1997]. A determinação do número de regras e funções de pertinência por entrada também pode ser auxiliada por um especialista do domínio. Estes aspectos de modelagem são extremamente importantes na concepção de um sistema baseado na Lógica *Fuzzy*, pois o desempenho do sistema é diretamente influenciado pelo número de regras e funções de pertinência por entrada. Uma análise mais detalhada desses aspectos contribuiria para obtenção de conclusões mais precisas sobre o modelo FuNN;
- Investigar e comparar outros métodos de seleção de atributos, tais como os descritos em [Piramuthu 1998] e [Brunzell & Eriksson 2000], com o método da rede FWD e o realizado através da árvore de decisão gerada por TREPAN;
- Atualmente, o processo de seleção de atributos é realizado de forma manual, ou seja, o usuário tem que identificar os atributos relevantes, remover os atributos irrelevantes, re-treinar o modelo e verificar se a remoção dos atributos não resultou em perda de performance. Uma melhoria que poderia ser incorporada à ferramenta *Neural Mining* é a implementação de um processo automático para seleção de atributos utilizando a técnica da rede FWD ou a árvore de decisão gerada por TREPAN. Desta forma, os atributos irrelevantes seriam removidos e o treinamento seria reinicializado sem a intervenção do usuário. Técnicas de visualização que permitissem uma melhor análise da relevância dos atributos também poderiam ser adicionadas à ferramenta;
- Incorporar à ferramenta *Neural Mining* o mecanismo de inferência *fuzzy max-min*, permitindo que a fase de avaliação das regras *fuzzy* seja realizada na própria ferramenta;
- Incorporar técnicas de visualização [Fayyad et al. 2001] à ferramenta *Neural Mining* a fim de facilitar o entendimento do conhecimento minerado. Estas técnicas poderiam ser usadas para mostrar a árvore de decisão extraída por TREPAN numa forma gráfica, apresentar as regras sobre diversas perspectivas (Ex.: confiança, suporte e outras medidas de interesse) ou representar as funções de pertinência das condições e conclusões das regras *fuzzy* e o mecanismo de inferência *fuzzy* decorrente da aplicação de um exemplo ao sistema. Esta necessidade é mais evidente nas regras *fuzzy*, pois é importante que o usuário visualize como cada termo lingüístico é representado e como as saídas *fuzzy* são geradas;
- Implementar outros modelos neuro-*fuzzy* e técnicas de extração de conhecimento simbólico de RNA na ferramenta *Neural Mining*. Outra opção seria incorporar à ferramenta algoritmos simbólicos convencionais de geração de regras de associação (Ex.: Apriori e AprioriTid [Srikant

et al. 1997]), regras de classificação (Ex.: [Liu et al. 1998]) e de árvores de decisão (Ex.: CART [Breiman et al. 1984] e C4.5 [Quilan 1993]), oferecendo mais funcionalidades para o usuário;

- Adicionar à ferramenta *Neural Mining* a funcionalidade de acessar dados armazenados em um SGBD. Deste modo, o usuário não precisaria converter os dados armazenados numa tabela em arquivo texto, evitando problemas decorrentes da conversão, tais como perda de precisão e dados corrompidos;
- Incorporar técnicas de pré-processamento na ferramenta *Neural Mining* a fim de torná-la uma plataforma que englobe todas as etapas do processo de KDD.

8.4. Considerações Finais

Como pode ser observado nas seções abordadas por este capítulo, os objetivos propostos inicialmente foram alcançados.

Visando à investigação do paradigma dos Sistemas Neuro-Fuzzy e das técnicas de extração de conhecimento simbólico de RNA como uma alternativa para tornar as RNA mais adequadas ao processo de KDD, este trabalho apresentou uma investigação experimental bastante detalhada de três modelos neurais (MLP, FWD e FuNN), quatro técnicas de extração de conhecimento simbólico (TREPAN, da rede FWD, REFuNN e AREFuNN) e duas técnicas de seleção de atributos (da rede FWD e através da árvore de decisão gerada por TREPAN).

Os experimentos foram realizados com a ferramenta *Neural Mining*, uma das principais contribuições deste trabalho, usando uma base de dados de um problema real e de larga escala no domínio de análise de crédito.

Com base nos resultados experimentais, não se pode afirmar, em linhas gerais, o melhor modelo ou técnica. Cada modelo e técnica apresentou vantagens e deficiências que os tornaram mais apropriados para serem aplicados em uma ou mais etapas do processo de KDD.

Apesar de ter apresentado um desempenho (taxa de classificação) aproximadamente igual ao dos modelos MLP e FuNN, o modelo FWD possui uma grande limitação: capacidade de resolver apenas problemas linearmente separáveis. Além desta limitação, as regras extraídas de uma rede FWD são bastante extensas, o que dificulta o entendimento do conhecimento minerado. A funcionalidade do modelo FWD que mais se destacou foi a seleção de atributos. Portanto, este modelo é bastante promissor para ser aplicado na etapa de seleção de dados do processo de KDD.

Ao contrário da rede FWD, os modelos MLP e FuNN são capazes de resolver problemas não linearmente separáveis. Por esta razão, estes modelos são mais adequados para serem aplicados na etapa de mineração de dados.

Com relação à extração de regras do modelo FuNN, a técnica AREFuNN se destacou mais do que a técnica REFuNN, apresentando regras menores e mais precisas (45,10% para a técnica AREFuNN e 25,10% para a técnica REFuNN). No entanto, quando os conjuntos de regras extraídos pelas duas técnicas são agrupados, a precisão é melhorada (49,60%). A precisão das regras extraídas pelas técnicas AREFuNN e REFuNN pode ser melhorada através da utilização de limiares menores. Deste modo, as regras passam a ter maior precisão, mas se tornam mais complexas.

A técnica TREPAN apresentou desempenho (taxa de classificação) aproximadamente igual ao da rede MLP (66,51% para a técnica TREPAN e 68,22% para a rede MLP) e a fidelidade da árvore de decisão extraída foi bastante representativa (85,78%). Com relação à seleção de atributos realizada com o auxílio da árvore de decisão gerada por TREPAN, apesar de ter ocorrido um decréscimo nas taxas de classificação na maioria dos modelos, tal técnica mostrou resultados satisfatórios quando aplicada aos modelos FuNN e MLP. No modelo MLP, o decréscimo foi de 1,31% na taxa de classificação total e 1,83% na taxa de classificação da classe adimplente. No modelo FuNN, o decréscimo foi de 0,6% na taxa de classificação total, 0,7% na taxa de classificação da classe adimplente e 0,8% na taxa de classificação da classe inadimplente. Considerando o aspecto de extração de conhecimento, TREPAN gerou uma árvore de decisão bastante simples. Como resultado, a interpretação e aplicação direta do conhecimento extraído foram facilitadas. Portanto, TREPAN pode ser usada nas etapas de apresentação do conhecimento e seleção de dados.

A partir da constatação das vantagens e desvantagens de cada modelo e técnica, foi possível propor duas soluções neurais híbridas para o processo de KDD utilizando os modelos e técnicas investigados nesta dissertação.

Baseando-se nos resultados alcançados, pode-se concluir que várias etapas do processo de KDD podem ser realizadas através da aplicação de Sistemas Neurais Híbridos e a integração de RNA com técnicas que utilizam representação simbólica, como Sistemas *Fuzzy* e algoritmos simbólicos convencionais, é bastante viável para ser aplicada em problemas de KDD onde a apresentação do conhecimento minerado em um formato de fácil interpretação é um fator crucial. Esta viabilidade é resultante da capacidade das técnicas de extração de conhecimento simbólico de RNA de representar o conhecimento incorporado pela rede numa forma compreensível. Os resultados também demonstraram que os Sistemas Neurais Híbridos são bastante atrativos para serem usados em aplicações reais, podendo ser considerados alternativos aos modelos neurais tradicionais, sem perda de desempenho e com a funcionalidade adicional de representar o conhecimento numa forma compreensível.

Apêndice A
Lógica *Fuzzy*

Este apêndice apresenta uma comparação entre a Teoria dos Conjuntos *Fuzzy* e a Teoria dos Conjuntos Clássicos, e os principais conceitos e operações associados à Lógica *Fuzzy*.

A.1. Teoria dos Conjuntos *Fuzzy* x Teoria dos Conjuntos Clássicos

A Teoria dos Conjuntos *Fuzzy* foi inicialmente construída a partir de conceitos estabelecidos pela Teoria dos Conjuntos Clássicos. Seus operadores foram definidos à semelhança dos tradicionalmente utilizados e, posteriormente, outros foram sendo introduzidos [Zimmermann 1991].

Na Teoria dos Conjuntos Clássicos, um conjunto S é definido por uma função f_S, denominada função característica de S. f_S é especificada da seguinte forma [Marsh et al. 1992]:

$$f_S : S \rightarrow \{0, 1\}$$

$$\text{Para um elemento } x \text{ de } S,\ f_S(x) = \begin{cases} 1, \text{ se } x \in S \\ 0, \text{ se } x \notin S \end{cases}$$

Portanto, na Teoria dos Conjuntos Clássicos, um elemento pertence ou não a um conjunto S.

Ao invés de ter um limite abrupto que define os elementos que pertencem ou não a um conjunto, [Zadeh 1965] propôs uma caracterização mais ampla e expressiva, generalizando a função característica, de modo que ela pudesse assumir valores no intervalo [0,1].

Na Teoria dos Conjuntos *Fuzzy*, um conjunto S é definido por uma função μ_S, denominada função de pertinência de S. Para um elemento x de S, $\mu_S(x)$ representa o grau para o qual x é um elemento de S. μ_S é especificada da seguinte forma [Marsh et al. 1992]:

$$\mu_S: S \rightarrow [0, 1]$$
$$\mu_S(x) = 1, \text{ se } x \text{ está completamente compatível com } S$$
$$\mu_S(x) = 0, \text{ se } x \text{ está completamente incompatível com } S$$
$$0 < \mu_S(x) < 1, \text{ se } x \text{ está parcialmente compatível com } S$$

Como pode ser observado, diferentemente da Lógica Booleana, a Lógica *Fuzzy* é multivalorada. Ao invés de uma proposição ser completamente verdadeira ou completamente falsa, através do uso de graus de pertinência, a Lógica *Fuzzy* permite que a proposição seja parcialmente verdadeira e parcialmente falsa, aumentando, desta forma, o poder de expressividade. A Figura A.1 [Azevedo et al. 2000] ilustra esta diferença. Neste caso, se a *altura* de uma pessoa é 1,69, ela é considerada *baixa* na Lógica Booleana, apesar de estar próxima do limiar estabelecido. Na Lógica *Fuzzy*, a pessoa é considerada *alta* e *baixa*, apresentando um grau de pertinência para cada conjunto *fuzzy*.

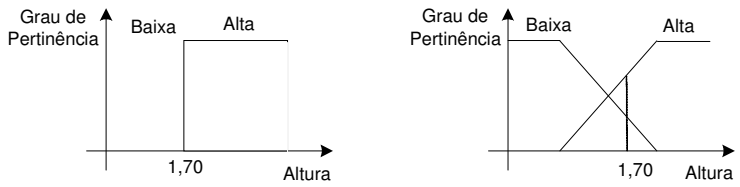

Figura A.1 - Lógica Booleana x Lógica *Fuzzy*

Com relação às regras, na Lógica Booleana, uma regra é ativada apenas quando o fato for exatamente igual ao antecedente da regra e o resultado é exatamente igual ao conseqüente da regra ativada. Na Lógica *Fuzzy*, uma regra é ativada se houver algum grau de similaridade diferente de 0 entre o fato e o antecedente da regra, e o resultado é uma conclusão com grau de similaridade não nulo em relação ao conseqüente da regra [Tanscheit 2003].

Diante das diferenças apresentadas, pode-se concluir que a Teoria dos Conjuntos *Fuzzy* é mais genérica e tem um escopo de aplicação mais amplo do que a dos Conjuntos Clássicos [Zadeh 1965].

A.2. Conceitos Fundamentais

Os principais conceitos da Lógica *Fuzzy*, representados na Figura A.2, são [Zadeh 1965] [Cox 1994] [Almeida & Evsukoff 2003]:

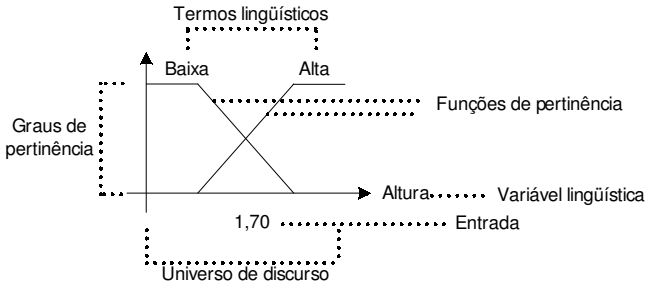

Figura A.2 - Conceitos da Lógica *Fuzzy*

Conjunto *Fuzzy* – Conjunto com fronteiras *fuzzy*, onde cada elemento está associado a um grau de pertinência, que assume valores entre 0 (pertinência nula) e 1 (pertinência total).

Um atributo pode estar associado a vários conjuntos *fuzzy*. Em geral, quanto maior o número de conjuntos *fuzzy* associados aos atributos, melhor a resolução do sistema. No entanto, um número grande de conjuntos requer mais tempo computacional e pode resultar em um sistema instável. Portanto, normalmente são utilizados de 3 a 9 conjuntos *fuzzy* por atributo.

Para garantir suavidade e estabilidade, deve existir uma sobreposição parcial entre conjuntos *fuzzy* adjacentes.

Função de Pertinência – Função matemática que fornece significado numérico para um conjunto *fuzzy*, mapeando entradas *x* em graus de pertinência $\mu(x)$.

Normalmente as funções de pertinência são determinadas pelos especialistas do domínio. No entanto, tais funções podem não ser precisas o suficiente. Portanto, é aconselhável a aplicação de técnicas de otimização para ajustar os parâmetros das funções.

As funções de pertinência podem apresentar diferentes formas. A Figura A.3 [Marsh et al. 1992] mostra alguns exemplos.

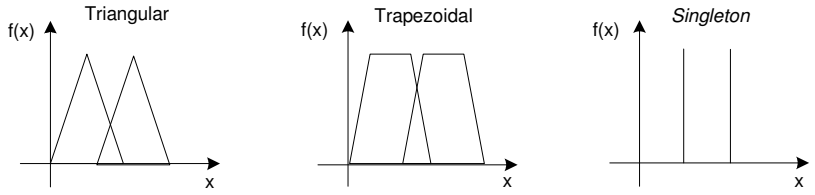

Figura A.3 - Exemplos de funções de pertinência

Por apresentarem fórmulas simples e eficiência computacional, as funções triangulares e trapezoidais têm sido bastante utilizadas, especialmente em aplicações de tempo real. No entanto, como estas funções são compostas de segmentos de linhas retas, elas não são suaves nas suas extremidades.

Singletons são facilmente representados e permitem a aplicação de algoritmos de defuzzificação mais simples. Conseqüentemente, este tipo de função é bastante utilizado para descrever saídas *fuzzy*. No entanto, as demais funções podem fornecer uma saída mais consistente.

As funções de pertinência gaussianas, *bell-shaped* e sigmóides são definidas por funções não-lineares e suaves, descritas pelas Equações A.1, A.2 e A.3, respectivamente.

$$gaussiana(x;\, c,\, \sigma) = e^{-\frac{1}{2}\left(\frac{x-c}{\sigma}\right)^2} \tag{A.1}$$

$$bell\text{-}shapped(x;\, a,\, b,\, c) = \frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}} \tag{A.2}$$

$$sigmóide(x;\, a,\, c) = \frac{1}{1+exp[-a(x-c)]} \tag{A.3}$$

Na Equação A.1, *c* e σ representam, respectivamente, o centro e a largura da função gaussiana. Na Equação A.2, *a* e *c* representam, respectivamente, a largura e o centro da função, e *b* controla a inclinação da função nos pontos de *crossover* (pontos cuja pertinência é igual a 0,5). Na Equação A.3, *a* controla a inclinação nos pontos de *crossover*.

Devido à suavidade e notação concisa, as funções gaussianas e *bell-shapped* estão sendo cada vez mais utilizadas. As funções *bell-shapped* têm um parâmetro a mais que as gaussianas. Como resultado, apresentam um grau de liberdade maior em seu ajuste. Embora as funções gaussianas e *bell-shapped* forneçam suavidade, elas são incapazes de especificar funções de pertinência assimétricas, que são úteis em algumas aplicações. Assimetria pode ser alcançada através da diferença absoluta ou do produto de duas funções sigmóides.

Uma função de pertinência é simétrica quando $\mu(c+x) = \mu(c-x)$, para todo $x \in X$.

Grau de pertinência – Também denominado valor verdade, denota o grau de compatibilidade de um valor com o conceito representado pelo conjunto *fuzzy*. Assume valores no intervalo [0,1].

Na Lógica *Fuzzy*, a soma dos graus de pertinência associados a uma variável lingüística nem sempre é unitária.

Termo lingüístico – Descrição ou nome lingüístico associado a um conjunto *fuzzy*. Os termos lingüísticos podem ser sentenças construídas a partir de termos primários (rótulos de conjuntos *fuzzy*, como por exemplo, *pequeno* e *médio*), conectivos lógicos (*E*, *OU* e *NÃO*), conectivos mascarados (por exemplo, *mas* e *porém*), modificadores (como *muito* e *pouco*) e delimitadores (como parênteses).

Variável lingüística – Entidade utilizada para representar de modo impreciso e, portanto, lingüístico, um conceito ou uma variável de um determinado problema. As variáveis lingüísticas admitem como valores apenas termos lingüísticos, reduzindo, desta forma, a complexidade durante a modelagem do sistema.

Universo de discurso – Todos os valores aplicáveis a uma variável. Vários conjuntos *fuzzy* podem ser definidos dentro do mesmo universo de discurso, cada um com seu próprio domínio, podendo sobrepor os domínios dos conjuntos *fuzzy* adjacentes.

Na prática, quando o universo de discurso é um espaço contínuo, normalmente ele é particionado em vários conjuntos *fuzzy* cujas funções de pertinência cobrem-no de uma maneira uniforme.

A.3. Operações *Fuzzy*

As operações fundamentais na Teoria dos Conjuntos *Fuzzy*, assim como na Teoria dos Conjuntos Clássicos, são interseção (a base do operador *E*), união (a base do operador *OU*) e complemento (a base do operador de *NÃO*). Tais operações foram propostas por [Zadeh 1965] como uma extensão das operações clássicas, preservando, desta forma, propriedades como involução, idempotência, comutatividade, associatividade, absorção, distributividade, Leis de De Morgan e identidade. Ao contrário do observado em conjuntos clássicos, para conjuntos *fuzzy* tem-se [Tanscheit 2003]:

$$A \cap \hat{A} \neq \varnothing \text{ e } A \cup \hat{A} \neq U$$

Na Teoria dos Conjuntos *Fuzzy*, a interseção e a união são implementadas pelas famílias de operadores denominadas *t-normas* e *t-conormas*, respectivamente. Estes operadores se reduzem aos operadores clássicos de união e interseção quando aplicados a conjuntos booleanos [Zimmermann 1991]. A Tabela A.1 mostra os *t-normas* e *t-conormas* mais utilizados [Sandri & Correa 1999].

Tabela A.1 - Principais operadores *t-normas* e *t-conormas*

t-norma	**Nome**	***t-conorma***	**Nome**	
min(a, b)	Mínimo	max(a, b)	Máximo	Zadeh
a . b	Produto algébrico	a + b – ab	Soma probabilística	Probabilista
max(a+b–1, 0)	Produto limitado	min(a+b, 1)	Soma limitada	Lukasiewicz
a, se b = 1 b, se a = 1 0, se a,b < 1	Produto drástico	a, se b = 0 b, se a = 0 1, se a,b > 0	Soma drástica	Weber

A união de dois conjuntos *fuzzy A* e *B*, efetuada através do operador *max* de Zadeh, é ilustrada na Figura A.4 [Sandri & Correa 1999] e determinada por:

$$\mu_{AUB}(x) = max[\mu_A(x), \mu_B(x)] = \mu_A(x) \vee \mu_B(x) \quad (A.4)$$

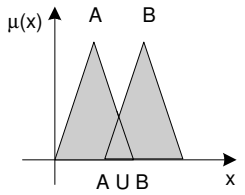

Figura A.4 - União de conjuntos *fuzzy*

A interseção de dois conjuntos *fuzzy A* e *B*, efetuada através do operador *min* de Zadeh, é ilustrada na Figura A.5 [Sandri & Correa 1999] e determinada por:

$$\mu_{A\cap B}(x) = min[\mu_A(x), \mu_B(x)] = \mu_A(x) \wedge \mu_B(x) \quad (A.5)$$

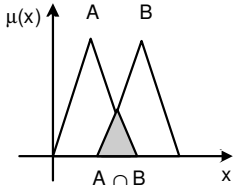

Figura A.5 - Interseção de conjuntos *fuzzy*

O complemento de um conjunto *fuzzy* representa o grau para o qual um elemento não é membro do conjunto. O complemento de um conjunto *fuzzy A*, representado por $\neg A$ ou $\hat{A}$, é ilustrado na Figura A.6 (adaptada de [Almeida e Evsukoff 2003]) e seu principal operador é determinado por:

$$\mu_A(x) = 1 - \mu_A(x) \quad (A.6)$$

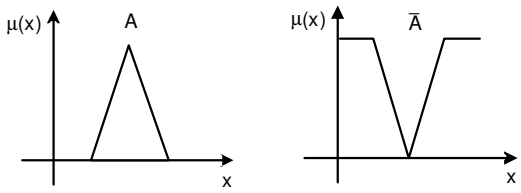

Figura A.6 - Complemento de um conjunto *fuzzy*

A Tabela A.2 mostra outros operadores de complemento [Jang et al. 1997].

Tabela A.2 - Operadores de complemento

Operador	
$N_s(a) = \frac{1-a}{1+sa}$, com s > -1	Sugeno
$N_w(a) = (1-a^w)^{1/w}$, com w > 0	Yager

Os operadores de implicação I: $[0,1]^n \rightarrow [0,1]$ são usados para modelar regras *fuzzy* Se-Então. Considerando os conjuntos *fuzzy* A e B definidos por $\mu_A: X \rightarrow [0,1]$ e $\mu_B: Y \rightarrow [0,1]$, respectivamente, a relação $A \rightarrow B$ é expressa como [Sandri & Correa 1999]:

$$\mu_{A \rightarrow B}(x, y) = I(\mu_A(x), \mu_B(y)) \quad \text{(A.7)}$$

Os principais operadores de implicação são mostrados na Tabela A.3.

Tabela A.3 - Operadores de implicação

Operador	Nome
max(1-a, b)	Kleene – Diemes
min(1-a+b, 1)	Lukasiewicz
1, se a ≤ b 0, senão	Rescher – Gaines "Sharp"
1, se a ≤ b b, senão	Brower – Gödel
min(b/a), se a ≠ b 1, senão	Goguen
1-a+ab	Reichenbach "Estocástica"
max(1-a, min(a,b))	Zadeh – Wilmott
min(a,b)	Mamdani
a.b	Larsen

t-normas não são implicações propriamente ditas, mas são muito empregadas na prática como implicações, principalmente em aplicações de controle [Sandri & Correa 1999].

Além das operações descritas, existem diversas formas de combinar conjuntos *fuzzy*. Produto algébrico, soma algébrica e diferença absoluta são alguns exemplos [Zadeh 1965].

A.4. Proposições *Fuzzy*

Uma sentença da forma "*x é A*", onde *x* é uma variável lingüística e *A* é um conjunto *fuzzy* definido no universo de discurso *U* de *x*, é chamada de proposição *fuzzy* [Tanscheit 2003].

Proposições *fuzzy* podem ser combinadas por meio de diferentes operadores, como por exemplo, os conectivos lógicos *E* e *OU*, e o operador de implicação *Se-Então*. As proposições resultantes podem ser descritas em termos de relações *fuzzy*.

A.4.1. Relações *Fuzzy*

Sejam A_1, A_2, ..., A_n conjuntos *fuzzy* em U_1, U_2, ..., U_n, respectivamente. Uma relação *fuzzy* n-ária é um conjunto *fuzzy* em $U_1 \times U_2 \times ... \times U_n$, expresso da seguinte forma [Jang et al. 1997]:

$$R = \{ \mu_R(x_1, x_2, ..., x_n) / (x_1, x_2, ..., x_n) \}$$

Considerando as variáveis lingüísticas *x* e *y* com universos de discurso *X* e *Y*, os conjuntos *fuzzy A* e *B* definidos em *X* e *Y*, e as proposições *fuzzy* "*x é A*" e "*y é B*", quando essas proposições são conectadas pelo operador *OU*, obtém-se uma relação *fuzzy* em *X x Y*, determinada por uma função f_{OU} (usualmente um *t-conorma*) [Zimmermann 1991]:

$$R_{A\ OU\ B} = \{\mu_R(x,y) / (x,y)\} \text{ tal que } \mu_R(x,y) = f_{OU}(\mu_A(x), \mu_B(y))$$

O mesmo ocorre quando as proposições são conectadas pelo operador *E* e pelo operador de implicação. Neste caso, um *t-norma* é normalmente usado como a f_E e um operador de implicação é usado como a f_I.

A.4.2. Regra de Inferência Composicional e Raciocínio *Fuzzy*

Na Lógica *Fuzzy*, uma regra de inferência importante é o *Modus Ponens* Generalizado (MPG), que constitui uma extensão do *Modus Ponens* da Lógica Clássica. O MPG estabelece a seguinte derivação:

premissa 1: x é A'
premissa 2: x é A → y é B
conseqüência: y é B'

onde *A*, *A'*, *B* e *B'* são conjuntos *fuzzy*.

Normalmente a regra MPG é baseada na lei de inferência composicional [Zadeh 1965]. Nesta lei, a premissa "*x é A → y é B*" é transcrita como uma relação *fuzzy R*, construída de acordo com a definição do operador de implicação:

$$\forall\, x \in X,\ y \in Y; \quad \mu_R(x, y) = \mu_{A \rightarrow B}(x, y) \tag{A.8}$$

onde *X* e *Y* são os universos de discurso associados a *A* e *B*, respectivamente.

Dados o fato "*x é A'* " e a implicação *A → B* representada pela relação *R*, a lei de inferência composicional estabelece que:

$$B' = A' \circ R = A' \circ (A \rightarrow B) \tag{A.9}$$

onde a regra de composição é definida por:

$$\forall\, y \in Y, \quad \mu_{B'}(y) = \max_x \{ \min [\mu_{A'}(x), \mu_R(x,y)] \} \tag{A.10}$$

Esta regra de inferência é conhecida como *max-min*. Genericamente, a regra de composição pode ser expressa como:

$$\forall\, y \in Y, \quad \mu_{B'}(y) = \max_x \{ \tau [\mu_{A'}(x), \mu_R(x,y)] \} \tag{A.11}$$

onde τ é um operador *t-norma*. Esta regra de inferência mais geral é referida por max-τ, sendo suas instâncias as leis *max-min*, *max-produto*, *max-produto limitado* e *max-produto drástico*. As leis de inferência composicional *max-min* e *max-produto* são as mais aplicadas [Medeiros 1996].

Uma simplificação da Equação A.9 resulta em:

$$\mu_{B'}(y) = \{ \max_x [\mu_{A'}(x) \wedge \mu_A(x)] \} \wedge \mu_B(y) = w \wedge \mu_B(y) \tag{A.12}$$

onde *w* é o máximo da função de pertinência de $A \cap A'$.

A função de pertinência de *B'* é igual a função de pertinência de *B* interceptada por *w*. Em geral, *w* denota o grau de compatibilidade entre *A* e *A'*.

No caso em que o antecedente da regra é composto de *n* premissas conectados pelo operador *E*, $w = w_1 \wedge w_2 \wedge ... \wedge w_n$. Quando as premissas da regra estão conectadas pelo operador *OU*, $w = w_1 \, V \, w_2 \, V \, ... \, V \, w_n$.

Quando existem várias regras *fuzzy*, a interpretação normalmente é feita como a união das relações *fuzzy* correspondentes às regras, ou seja, a função de pertinência da conclusão final é igual à união das funções de pertinência dos conseqüentes das regras interceptadas pelos seus *firing strength*s correspondentes.

A regra de inferência descrita nesta seção é denominada regra de inferência composicional. A partir desta regra é possível formalizar um mecanismo de inferência que deriva conclusões a partir de um conjunto de regras *fuzzy* Se-Então e fatos conhecidos. Este mecanismo é denominado raciocínio aproximado ou raciocínio *fuzzy* [Cox 1994].

Apêndice B

Formato dos Arquivos da Ferramenta *Neural Mining*

Os dados de entrada e saída da ferramenta *Neural Mining* são armazenados em arquivos texto formatados (*flat files* ou arquivos *txt*). Para cada tipo de dado, a ferramenta adota um formato específico. Este apêndice aborda os formatos dos arquivos que são usados como entrada ou gerados como saída pela ferramenta *Neural Mining*.

B.1. Arquivos de Dados

Os arquivos que armazenam os dados de treinamento, validação e teste são estruturados da seguinte forma (Tabela B.1): cada linha representa um registro e cada coluna um atributo de entrada ou saída. A ferramenta *Neural Mining* supõe que os atributos de saída estão localizados nas últimas colunas.

Tabela B.1 - Formato dos arquivos de dados

$registro_1$	$\Rightarrow$	$atributo_1$	$atributo_2$	...	$atributo_N$
$registro_2$	$\Rightarrow$	$atributo_1$	$atributo_2$	...	$atributo_N$
...	$\Rightarrow$	...	...	...	...
$registro_M$	$\Rightarrow$	$atributo_1$	$atributo_2$	...	$atributo_N$

B.2. Arquivos de Pesos dos Modelos Neurais

Os arquivos que armazenam os pesos dos modelos neurais são estruturados de acordo com as particularidades de cada modelo. No modelo MLP (Tabela B.2), o total de linhas é igual ao somatório do número de nós intermediários (*I*) com o número de nós de saída (*S*). Nas linhas referentes ao número de nós intermediários, o total de colunas é igual ao número de nós de entrada (*E*), e nas linhas referentes ao número de nós de saída, o total de colunas é igual ao número de nós intermediários.

Tabela B.2 - Formato do arquivo de pesos do modelo MLP

$intermediário_1$	$\Rightarrow$	$entrada_1$	$entrada_2$	...	$entrada_E$
$Intermediário_2$	$\Rightarrow$	$entrada_1$	$entrada_2$	...	$entrada_E$
...	$\Rightarrow$	…	…	…	…
$intermediário_I$	$\Rightarrow$	$entrada_1$	$entrada_2$	...	$entrada_E$
$saída_1$	$\Rightarrow$	$intermediário_1$	$intermediário_2$	...	$intermediário_I$
$saída_2$	$\Rightarrow$	$intermediário_1$	$intermediário_2$	...	$intermediário_I$
...	$\Rightarrow$	...	...	...	...
$saída_S$	$\Rightarrow$	$intermediário_1$	$intermediário_2$	...	$intermediário_I$

No modelo FuNN (Tabela B.3), o total de linhas é igual ao somatório do dobro do número de nós de entrada (*2.E*) com o número de nós regra (*R*) e o número de nós de saída (*S*). As linhas referentes ao número de nós de entrada se dividem em dois grupos. No primeiro grupo, o número de colunas é igual ao número de nós condição associados ao nó de entrada correspondente à linha. No segundo grupo, o número de colunas é igual ao número de nós condição associados ao nó de entrada correspondente à linha vezes o número de nós regra. Nas linhas referentes ao número de nós regra, o total de colunas é igual ao somatório (*T*) do número de nós ação de todos os nós de saída. Nas linhas referentes ao número de nós de saída, o total de colunas é igual ao número de nós ação associados ao nó de saída correspondente à linha.

Tabela B.3 - Formato do arquivo de pesos do modelo FuNN

$entrada_1 \Rightarrow condição_1 \; condição_2 \; ... \; condição_A$ (A = número de nós condição para a $entrada_1$)
$entrada_2 \Rightarrow condição_1 \; condição_2 \; ... \; condição_B$ (B = número de nós condição para a $entrada_2$)
... ⇒ … … ... ...
$entrada_E \Rightarrow condição_1 \; condição_2 \; ... \; condição_C$ (C = número de nós condição para a $entrada_E$)
$entrada_1 \Rightarrow condição_1Regra_1 \; ... \; condição_ARegra_1 \; ... \; condição_1Regra_R \; ... \; condição_ARegra_R$
$entrada_2 \Rightarrow condição_1Regra_1 \; ... \; condição_BRegra_1 \; ... \; condição_1Regra_R \; ... \; condição_BRegra_R$
... ⇒ … … ... ...
$entrada_E \Rightarrow condição_1Regra_1 \; ... \; condição_CRegra_1 \; ... \; condição_1Regra_R \; ... \; condição_CRegra_R$
$regra_1 \Rightarrow ação_1 \; ação_2 \; ... \; ação_T$ (T = total de nós ação = X + Y + ... + Z)
$regra_2 \Rightarrow ação_1 \; ação_2 \; ... \; ação_T$
... ⇒ … ... ...
$regra_R \Rightarrow ação_1 \; ação_2 \; ... \; ação_T$
$saída_1 \Rightarrow ação_1 \; ação_2 \; ... \; ação_X$ (X = número de nós ação para a $saida_1$)
$saída_2 \Rightarrow ação_1 \; ação_2 \; ... \; ação_Y$ (Y = número de nós ação para a $saida_2$)
... ⇒ ... ...
$saída_S \Rightarrow ação_1 \; ação_2 \; ... \; ação_Z$ (Z = número de nós ação para a $saida_S$)

No modelo FWD (Tabela B.4), o total de linhas é igual ao dobro do número de nós de saída (*2.S*). As *S* primeiras linhas representam os valores das conexões peso e as demais representam os valores das conexões de memória. O total de colunas em todas as linhas é igual ao número de nós de entrada (*E*).

Tabela B.4 - Formato do arquivo de pesos do modelo FWD

$pesoSaída_1Entrada_1$	$pesoSaída_1Entrada_2$	...	$pesoSaída_1Entrada_E$
$pesoSaída_2Entrada_1$	$pesoSaída_2Entrada_2$	...	$pesoSaída_2Entrada_E$
...	...	...	...
$pesoSaída_SEntrada_1$	$pesoSaída_SEntrada_2$	...	$pesoSaída_SEntrada_E$
$memóriaSaída_1Entrada_1$	$memóriaSaída_1Entrada_2$	...	$memóriaSaída_1Entrada_E$
$memóriaSaída_2Entrada_1$	$memóriaSaída_2Entrada_2$	...	$memóriaSaída_2Entrada_E$
...	...	...	...
$memóriaSaída_SEntrada_1$	$memóriaSaída_SEntrada_2$	...	$memóriaSaída_SEntrada_E$

B.3. Arquivos de Configuração dos Modelos Neurais

Similarmente aos arquivos de pesos, os arquivos de configuração são específicos para cada modelo neural, pois os mesmos apresentam parâmetros de treinamento, características arquiteturais e requisitos de representação do conhecimento diferentes.

O formato do arquivo de configuração do modelo MLP (Tabela B.5), é estruturado da seguinte forma: na primeira, segunda e terceira linhas, são especificados, respectivamente, as taxas de aprendizagem, os termos *momentum* e os coeficientes de ganho dos nós intermediários e de saída; na quarta, quinta e sexta linhas, são definidos o número de nós de entrada, intermediários e de saída, respectivamente.

Tabela B.5 - Formato do arquivo de configuração do modelo MLP

taxaAprendizagemNósIntermediários taxaAprendizagemNósSaída momentumNósIntermediários momentumNósSaída coeficienteGanhoNósIntermediários coeficienteGanhoNósSaída
númeroNósEntrada númeroNósIntermediários númeroNósSaida

No arquivo de configuração do modelo FuNN (Tabela B.6), as taxas de aprendizagem e os termos *momentum* dos nós condição, regra, ação e saída são definidos na primeira e segunda linhas, respectivamente. Os coeficientes de ganho dos nós regra e ação são especificados na terceira linha. A quarta, quinta e sexta linhas são reservadas para definir o número de nós de entrada, regra e saída, respectivamente. As demais linhas do arquivo definem os atributos de entrada e saída. Nestas linhas, o primeiro valor representa a descrição do atributo e os valores posteriores correspondem aos termos lingüísticos associados às funções de pertinência dos atributos.

Tabela B.6 - Formato do arquivo de configuração do modelo FuNN

taxaAprendizagemCondição taxaAprendizagemRegra taxaAprendizagemAção taxaAprendizagemSaída momentumCondição momentumRegra momentumAção momentumSaída coeficienteGanhoRegra coeficienteGanhoAção
númeroNósEntrada númeroNósRegra númeroNósSaída
$descriçãoEntrada_1$ $termoLingüístico_1$ $termoLingüístico_2$ ... $termoLingüístico_A$ $descriçãoEntrada_2$ $termoLingüístico_1$ $termoLingüístico_2$ ... $termoLingüístico_B$ ... ... $descriçãoEntrada_E$ $termoLingüístico_1$ $termoLingüístico_2$ ... $termoLingüístico_C$
$descriçãoSaída_1$ $termoLingüístico_1$ $termoLingüístico_2$ ... $termoLingüístico_X$ $descriçãoSaída_2$ $termoLingüístico_1$ $termoLingüístico_2$ ... $termoLingüístico_Y$ ... ... $descriçãoSaída_S$ $termoLingüístico_1$ $termoLingüístico_2$ ... $termoLingüístico_Z$

No arquivo de configuração do modelo FWD (Tabela B.7), os parâmetros de treinamento (taxa de aprendizagem, taxa de aprendizagem temporal e nebulosidade) são definidos na primeira, segunda e

terceira linhas. As duas linhas seguintes são utilizadas para especificar o número de nós de entrada e de saída. As demais linhas são reservadas para definição dos atributos de entrada e saída. No caso do modelo FWD, apenas os atributos de entrada são associados a um termo lingüístico, ou seja, são representados por uma função de pertinência.

Tabela B.7 - Formato do arquivo de configuração do modelo FWD

taxaAprendizagem taxaAprendizagemTemporal nebulosidade
númeroNósEntrada númeroNósSaída
descriçãoEntrada$_1$ termoLingüístico$_1$ descriçãoEntrada$_2$ termoLingüístico$_2$ ... descriçãoEntrada$_E$ termoLingüístico$_E$
descriçãoSaida$_1$ descriçãoSaida$_2$ ... descriçãoSaída$_S$

B.4. Arquivo de Descrição dos Atributos

O arquivo de descrição dos atributos (Tabela B.8) é utilizado pela técnica TREPAN. Neste arquivo, cada linha é associada a um atributo. A ferramenta *Neural Mining* supõe que as últimas linhas são utilizadas para descrever os atributos de saída. As linhas deste arquivo devem conter as seguintes informações sobre os atributos: descrição, tipo (*N*, para atributos nominais, ou *R*, para atributos numéricos), valores possíveis (especificados apenas para atributos nominais) e valores codificados (especificados apenas para atributos nominais). Os valores codificados representam a codificação que deve ser aplicada aos exemplos quando estes forem apresentados à RNA. Tal codificação deve ser especificada, pois as RNA são capazes de lidar apenas com valores numéricos.

Tabela B.8 - Formato do arquivo de descrição dos atributos da técnica TREPAN

descAtr$_1$	tipoAtr$_1$	valor$_1$Atr$_1$	valor$_2$Atr$_1$	...	valor$_N$Atr$_1$	valorRna$_1$Atr$_1$	valorRna$_2$Atr$_1$	...	valorRna$_N$Atr$_1$
descAtr$_2$	tipoAtr$_2$	valor$_1$Atr$_2$	valor$_2$Atr$_2$	...	valor$_X$Atr$_2$	valorRna$_1$Atr$_2$	valorRna$_2$Atr$_2$	...	valorRna$_X$Atr$_2$
...	...	...	...	...	...	...	...	...	...
descAtr$_A$	tipoAtr$_A$	valor$_1$Atr$_A$	valor$_2$Atr$_A$	...	valor$_Y$Atr$_A$	valorRna$_1$Atr$_A$	valorRna$_2$Atr$_A$	...	valorRna$_Y$Atr$_A$

B.5. Arquivos do Conhecimento Simbólico Extraído

Na ferramenta *Neural Mining*, a técnica TREPAN é utilizada em conjunto com a rede MLP com o objetivo de extrair uma árvore de decisão que represente o conhecimento incorporado pela rede. A estrutura da árvore de decisão é armazenada em um arquivo cujo formato é especificado na Tabela B.9. Cada linha do arquivo está associada a um nó da árvore e contém as seguintes informações: identificador, teste, identificador do primeiro filho (proveniente do ramo que satisfaz o teste do nó), identificador do segundo filho (proveniente do ramo que não satisfaz o teste do nó) e rótulo da classe.

Nos nós que não possuem filhos, os campos correspondentes aos identificadores dos filhos são preenchidos com *null*. Os testes dos nós são especificados da seguinte forma: *identificadorAtributo operadorLógico valor*.

Tabela B.9 - Formato do arquivo do conhecimento extraído pela técnica TREPAN

$identificadorNó_1$	$testeNó_1$	$identificadorFilho_1Nó_1$	$identificadorFilho_2Nó_1$	$rótuloClasseNó_1$
$identificadorNó_2$	$testeNó_2$	$identificadorFilho_1Nó_2$	$identificadorFilho_2Nó_2$	$rótuloClasseNó_2$
...	...	...	...	...
$identificadorNó_N$	$testeNó_N$	$identificadorFilho_1Nó_N$	$identificadorFilho_2Nó_N$	$rótuloClasseNó_N$

A Tabela B.10 mostra um exemplo de árvore de decisão gerada a partir de um arquivo no formato da Tabela B.9.

Tabela B.10 - Árvore gerada a partir de um arquivo no formato da Tabela B.9

Arquivo no formato da Tabela B.9	**Árvore gerada a partir do arquivo especificado na primeira coluna**
1 2>0,2071 2 3 1 2 1>0,1015 4 5 2 5 null null null 1 4 1>0,3000 6 7 2 3 0>0,7346 null 9 1 9 null null null 1 7 5>0,1454 10 11 2 11 8>0,2617 12 13 1 12 null null null 2 13 null null null 1 10 0>0,5031 14 15 2 14 null null null 2 15 6>0,2734 16 17 2 17 null null null 1 6 3>0,1000 18 19 2 18 null null null 2 16 0>0,3851 20 21 2 21 null null null 2 19 1>0,4691 22 23 2 23 5>0,2216 24 25 2 24 null null null 2 25 null null null 1 20 5>0,4308 null 27 2 27 3>0,2681 null null 2 22 6>0,2099 null null 2	SE (atributo2 > 0,21) | SE (atributo1 > 0,10) | | SE (atributo1 > 0,30) | | | SE (atributo3 > 0,10) | | | | Classe = 2 | | | SENAO SE (atributo1 > 0,47) | | | | Classe = 2 | | | SENAO SE (atributo5 > 0,22) | | | | Classe = 2 | | | SENAO Classe = 1 | | SENAO SE (atributo5 > 0,15) | | | SE (atributo0 > 0,50) | | | | Classe = 2 | | | SENAO SE (atributo6 > 0,27) | | | | SE (atributo0 > 0,39) | | | | | SE (atributo5 > 0,43) | | | | | SENAO Classe = 2 | | | | SENAO Classe = 2 | | | SENAO Classe = 1 | | SENAO SE (atributo8 > 0,26) | | | Classe = 2 | | SENAO Classe = 1 | SENAO Classe = 1 SENAO SE (atributo0 > 0,73) SENAO Classe = 1

As regras extraídas pelas técnicas REFuNN e AREFuNN a partir de uma rede FuNN são armazenadas em um único arquivo cujo formato é descrito na Tabela B.11. As regras extraídas pela técnica REFuNN são agrupadas em três conjuntos: conjunto inicial de regras ponderadas, conjunto de regras simples e conjuntos de regras agregadas. As regras extraídas pela técnica AREFuNN são agrupadas em dois conjuntos: conjunto inicial de regras ponderadas e conjunto de regras agregadas. O conjunto inicial de regras ponderadas contém as regras cujos antecedentes e conseqüentes apresentam, respectivamente, grau de importância e grau de certeza maiores que os limiares

especificados. O conjunto de regras simples é gerado a partir do conjunto de regras ponderadas, removendo os graus de importância e certeza ou criando regras com um único antecedente cujo grau de importância é maior que o limiar Th_{OU}. O conjunto de regras agregadas também é derivado do conjunto de regras ponderadas. Neste caso, as regras que possuem condições e conclusões iguais, diferindo apenas nos graus de importância, são agregadas em uma regra.

Tabela B.11 - Formato do arquivo do conhecimento extraído pelas técnicas REFuNN e AREFuNN

Conjunto inicial de regras ponderadas extraído pela técnica REFuNN
Conjunto de regras simples extraído pela técnica REFuNN
Conjunto de regras agregadas extraído pela técnica REFuNN
Conjunto inicial de regras ponderadas extraído pela técnica AREFuNN
Conjunto de regras agregadas extraído pela técnica AREFuNN

As regras ponderadas e agregadas possuem o seguinte formato:

SE descriçãoAtributo$_X$ é termoLingüístico$_X$ (grauImportância$_X$) E ... E
descriçãoAtributo$_Y$ é termoLingüístico$_Y$ (grauImportância$_Y$)
ENTÃO descriçãoSaída é termoLingüístico$_S$ (grauCerteza$_S$)

A única diferença no formato das regras simples com relação ao formato das regras agregadas e ponderadas é que as regras simples não apresentam graus de importância nos antecedentes e graus de certeza nos conseqüentes.

A Tabela B.12 mostra o formato do arquivo que armazena as regras extraídas de uma rede FWD. Como descrito no Capítulo 5, a técnica do modelo FWD extrai uma regra para cada classe e todos os atributos de entrada fazem parte do antecedente das regras.

Tabela B.12 - Formato do arquivo do conhecimento extraído pela técnica do modelo FWD

RegraClasse$_1$: SE descriçãoAtributo$_1$ é termoLingüístico$_1$ E descriçãoAtributo$_2$ é termoLingüístico$_2$ E ... E descriçãoAtributo$_N$ é termoLingüístico$_N$ ENTÃO classe$_1$
RegraClasse$_2$: SE descriçãoAtributo$_1$ é termoLingüístico$_1$ E descriçãoAtributo$_2$ é termoLingüístico$_2$ E ... E descriçãoAtributo$_N$ é termoLingüístico$_N$ ENTÃO classe$_2$
...
RegraClasse$_C$: SE descriçãoAtributo$_1$ é termoLingüístico$_1$ E descriçãoAtributo$_2$ é termoLingüístico$_2$ E ... E descriçãoAtributo$_N$ é termoLingüístico$_N$ ENTÃO classe$_C$

B.6. Arquivos dos Resultados da Classificação

Os resultados da classificação também são armazenados em arquivos. Estes resultados são armazenados em dois tipos de arquivo: resultado do treinamento e resultado do teste. No arquivo que contém os resultados do treinamento do modelo MLP (Tabela B.13), são armazenados informações arquiteturais (número de nós por camada), parâmetros de treinamento (taxas de aprendizagem, termos *momentum* e coeficientes de ganho), critérios de parada (número máximo de épocas, erro mínimo de

treinamento e perda de generalização) e resultados da classificação nos conjuntos de treinamento e validação (época atual, SSE de treinamento, SSE de validação, taxa de erro total do conjunto de validação, taxa de erro por classe e perda de generalização atual). Nos resultados de teste do modelo MLP, são registrados informações arquiteturais (número de nós por camada), parâmetros de treinamento (taxas de aprendizagem, termos *momentum* e coeficientes de ganho) e resultados da classificação no conjunto de teste (SSE, taxa de erro total e taxa de erro por classe).

Tabela B.13 - Formato do arquivo dos resultados de classificação do modelo MLP

Resultado do Treinamento
Número de nós de entrada
Número de nós intermediários
Número de nós de saída
Taxas de aprendizagem dos nós intermediários e de saída
Termos *momentum* dos nós intermediários e de saída
Coeficientes de ganho dos nós intermediários e de saída
Número máximo de épocas
Erro mínimo de treinamento
Perda de generalização
Época SSE Treinamento SSE Validação Taxa de Erro (Validação) Erro-Classe$_1$ ... Erro-Classe$_C$ GL
... ...
Resultado do Teste
Número de nós de entrada
Número de nós intermediários
Número de nós de saída
Taxas de aprendizagem dos nós intermediários e de saída
Termos *momentum* dos nós intermediários e de saída
Coeficientes de ganho dos nós intermediários e de saída
SSE Teste Taxa de Erro (Teste) Erro-Classe$_1$ Erro-Classe$_2$ ... Erro-Classe$_C$
... ...

No arquivo que contém os resultados do treinamento do modelo FuNN (Tabela B.14), são armazenados informações arquiteturais (número de nós de entrada, regra e de saída), parâmetros de treinamento (taxas de aprendizagem, termos *momentum* e coeficientes de ganho), critérios de parada (número máximo de épocas, erro mínimo de treinamento e perda de generalização) e resultados da classificação nos conjuntos de treinamento e validação (época atual, SSE de treinamento, SSE de validação, taxa de erro total do conjunto de validação, taxa de erro por classe e perda de generalização atual). Nos resultados de teste do modelo FuNN, são registrados informações arquiteturais (número de nós de entrada, regra e de saída), parâmetros de treinamento (taxas de aprendizagem, termos *momentum* e coeficientes de ganho) e resultados da classificação no conjunto de teste (SSE, taxa de erro total e taxa de erro por classe).

No arquivo que contém os resultados do treinamento do modelo FWD (Tabela B.15), são armazenados informações arquiteturais (número de nós de entrada e saída), parâmetros de treinamento (taxa de aprendizagem, taxa de aprendizagem temporal e nebulosidade), critérios de parada (número máximo de épocas, erro mínimo de treinamento e perda de generalização) e resultados da classificação nos conjuntos de treinamento e validação (época atual, SSE de treinamento, SSE de validação, taxa de erro total no conjunto de validação, taxa de erro por classe e perda de generalização atual). Nos resultados de teste do modelo FWD, são registrados informações

arquiteturais (número de nós de entrada e saída), parâmetros de treinamento (taxa de aprendizagem, taxa de aprendizagem temporal e nebulosidade) e resultados da classificação do conjunto de teste (SSE, taxa de erro total e taxa de erro por classe).

Tabela B.14 - Formato do arquivo dos resultados de classificação do modelo FuNN

Resultado do Treinamento
Número de nós de entrada Número de nós regra Número de nós de saída Taxas de aprendizagem dos nós condição, regra, ação e saída Termos *momentum* dos nós condição, regra, ação e saída Coeficientes de ganho dos nós regra e ação Número máximo de épocas Erro mínimo de treinamento Perda de generalização Época SSE Treinamento SSE Validação Taxa de Erro (Validação) Erro-Classe$_1$ ... Erro-Classe$_C$ GL ... ...
Resultado do Teste
Número de nós de entrada Número de nós regra Número de nós de saída Taxas de aprendizagem dos nós condição, regra, ação e saída Termos *momentum* dos nós condição, regra, ação e saída Coeficientes de ganho dos nós regra e ação SSE Teste Taxa de Erro (Teste) Erro-Classe$_1$ Erro-Classe$_2$ ... Erro-Classe$_C$ ... ...

Tabela B.15 - Formato do arquivo dos resultados de classificação do modelo FWD

Resultado do Treinamento
Número de nós de entrada Número de nós de saída Taxas de aprendizagem Taxa de aprendizagem temporal Nebulosidade Número máximo de épocas Erro mínimo de treinamento Perda de generalização Época SSE Treinamento SSE Validação Taxa de Erro (Validação) Erro-Classe$_1$ ... Erro-Classe$_C$ GL ... ...
Resultado do Teste
Número de nós de entrada Número de nós de saída Taxas de aprendizagem Taxa de aprendizagem temporal Nebulosidade SSE Teste Taxa de Erro (Teste) Erro-Classe$_1$ Erro-Classe$_2$ ... Erro-Classe$_C$ ... ...

No arquivo que contém os resultados da técnica TREPAN (Tabela B.16), são armazenados informações sobre a rede neural (número de nós de entrada, intermediários e de saída; taxas de aprendizagem, termos *momentum* e coeficientes de ganho), os parâmetros utilizados durante a

geração da árvore de decisão (número máximo de nós internos e tamanho mínimo das amostras) e os resultados da classificação no conjunto de teste (iteração atual, taxa de erro total, taxa de erro por classe, fidelidade total, fidelidade por classe e número de nós internos).

Tabela B.16 - Formato do arquivo dos resultados da técnica TREPAN

Número de nós de entrada
Número de nós intermediários
Número de nós de saída
Taxas de aprendizagem dos nós intermediários e de saída
Termos *momentum* dos nós intermediários e de saída
Coeficientes de ganho dos nós intermediários e de saída
Número máximo de nós internos
Tamanho mínimo das amostras
Iteração Taxa de Erro(Teste) $Classe_1$ ... $Classe_C$ Fidelidade $Classe_1$ ... $Classe_C$ Nós internos
... ...

Referências Bibliográficas

[Abbott et al. 1998] ABBOTT, D.; MATKOVSKY, I.; ELDER, I. V. An Evaluation of High-end Data Mining Tools for Fraud Detection. In: IEEE INT. CONF. ON SYSTEMS, MAN, AND CYBERNETICS, 1998, San Diego. **Proceedings of the IEEE Int. Conf. on Systems, Man, and Cybernetics**. San Diego: oct. 1998. v.3, p.2836-2841.

[Adriaans & Zantinge 1996] ADRIAANS, P.; ZANTINGE, D. **Data Mining**. Addison-Wesley, 1996.

[Almeida & Evsukoff 2003] ALMEIDA, P. E. M.; EVSUKOFF, A. G. Sistemas Fuzzy. In: REZENDE, S. O. (Coord.) **Sistemas Inteligentes**: Fundamentos e Aplicações. Barueri: Manole, 2003, p.169-201.

[Amorim et al. 2001] AMORIM, B. P. et al. Extraction of fuzzy rules for and/or fuzzy artificial neural networks. In: VIETNAM-JAPAN BILATERAL SYMPOSIUM ON BIOMEDICAL IMAGING/MEDICAL INFORMITICS AND APPLICATIONS (VJMEDIMAG), n.1, 2001, Hanoi. **Proceedings of the First Vietnam-Japan Bilateral Symposium on Biomedical Imaging/Medical Informitics and Applications (VJMEDIMAG)**. Hanoi: 2001. p.108-114.

[Amorim et al. 2003] AMORIM, B. P. et al. Avaliação de um Modelo Neuro-difuso para Classificação de Padrões, Seleção de Atributos e Extração de Regras. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO (SBC'2003), n.23, 2003, Campinas. **Anais do V Encontro Nacional de Inteligência Artificial (ENIA 2003)**. Campinas: 2003. p.365-374.

[Andrews & Geva 1995] ANDREWS, R.; GEVA, S. Inserting and Extracting Knowledge from Constrained Error Backpropagation Networks. In: AUSTRALIAN CONF. NEURAL NETWROKS, n.6, 1995, Sydney. **Proceedings of the Sixth Australian Conf. Neural Networks**. Sydney: 1995.

[Andrews et al. 1996] ANDREWS, R. et al. **An evaluation and comparison of techniques for extracting and refining rules from artificial neural networks**. Technical Report, Queensland University of Technology, 1996.

[Azevedo et al. 2000] AZEVEDO, F. M.; BRASIL, L. M.; OLIVEIRA, R. C. L. **Redes Neurais com Aplicações em Controle e em Sistemas Especialistas**. Florianópolis: Bookstore, 2000.

[Beale & Jackson 1991] BEAL, R.; JACKSON, T. **Neural Computing**: An Introduction. A. Hilger, 1991.

[Berenji 1991] BERENJI, H, R. Refinement of approximate reasoning-based controllers by reinforcement learning. In: INT. MACHINE LEARNING WORKSHOP, n.8, 1991, Evanston. **Proceedings of the Eighth Int. Machine Learning Workshop**. Evanston: 1991. p.475-479.

[Berthold & Diamond 1995] BERTHOLD, M. R.; DIAMOND, J. Boosting the performance of RBF networks with dynamic decay adjustment. **Advances in Neural Information Processing System**, v.7, 1995.

[Bezdek 1981] BEZDEK, J. C. **Pattern Recognition with Fuzzy Objective Function Algorithms**. New York: Plenum, 1981.

[Bigus 1996] BIGUS, J. P. **Data Mining with Neural Networks**: Solving Business Problems from Application Development to Decision Support. McGraw-Hill, 1996.

[Blatt 1999] BLATT, A. **Avaliação de Risco e Decisão de Crédito**: Um Enfoque Prático. Nobel, 1999.

[Boz 2002] BOZ, O. Extracting Decision Trees from Trained Neural Networks. In: ACM SIGKDD INT. CONF. ON KNOWLEDGE DISCOVERY AND DATA MINING, n.8, 2002, Edmonton. **Proceedings of the Eighth ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining**. Edmonton: jul. 2002. p.456-461.

[Braga et al. 2000] BRAGA, A. P.; LUDERMIR, T. B.; CARVALHO, A. C. P. L. F. **Redes Neurais Artificiais***:* Teoria e Aplicações. Rio de Janeiro: LTC, 2000.

[Braga et al. 2003] BRAGA, A. P.; CARVALHO, A. C. P. de L. F. de; LUDERMIR, T. B. Redes Neurais Artificiais. In: REZENDE, S. O. (Coord.) **Sistemas Inteligentes**: Fundamentos e Aplicações. Barueri: Manole, 2003, p.141-168.

[Brasil 1999] BRASIL, L. M. **Proposta de Arquitetura para Sistema Especialista Híbrido e a Correspondente Metodologia de Aquisição do Conhecimento**. 1999. Tese (Doutorado em Engenharia Elétrica) - Programa de Pós-Graduação em Engenharia Elétrica, Departamento de Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis, 1999.

[Breiman et al. 1984] BREIMAN, L. et al. **Classification and Regression Trees**. New York: Chapman & Hall, 1984.

[Breiman 1996] BREIMAN, L. Bagging predictors. **Machine Learning**, v.24, p.123-140, 1996.

[Brunzell & Eriksson 2000] BRUNZELL, H.; ERIKSSON, J. Feature reduction for classification of multidimensional data. **Pattern Recognition**, v.33, p.1741-1748, 2000.

[Carpenter et al. 1992] CARPENTER, G. A. et al. Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. **IEEE Trans. on Neural Networks**, v.2, n.3, p.698-713, 1992.

[Carter & Catlett 1987] CARTER, C.; CATLETT, J. Assessing credit card applications using machine learning. **IEEE Expert**, v.3, p.71-79, 1987.

[Conde 2000] CONDE, G. A. B. **Análise comparativa de Redes Neuro-difusas para Classificação de Padrões e Extração de Regras**. 2000. Dissertação (Mestrado em Ciência da Computação) - Programa de Pós-Graduação em Ciência da Computação, Centro de Informática, Universidade Federal de Pernambuco, Recife, 2000.

[Cox 1994] COX, E. **The Fuzzy Systems Handbook**. Academy Press, 1994.

[Craven 1996] CRAVEN, M. W. **Extracting Comprehensible Models from Trained Neural Networks**. 1996. Ph.D. Dissertation, Dept. Comput. Sci., Univ. Wisconsin, Madison, 1996.

[Craven & Shavlik 1996] CRAVEN, M. W.; SHAVLIK, J. W. Extracting tree-structured representations of trained networks. In: TOURETZKY, D. S.; MOZER, M. C.; HASSELMO, M. E. (Eds.) **Advances in Neural Information Processing Systems**. Denver: MIT Press, 1996. p.37-45.

[Craven & Shavlik 1998] CRAVEN, M. W.; SHAVLIK, J. W. Using neural networks for data mining. **Future Generation Computer Systems** (Special Issue on Data Mining), v.13, p.211-229, 1998.

[Craven & Shavlik 1999] CRAVEN, M. W.; SHAVLIK, J. W. **Rule Extraction: Where Do We Go from Here?**. Machine Learning Research Group Working Paper, Department of Computer Sciences, University of Wisconsin, p.99-104, 1999.

[Cybenko 1988] CYBENKO, G. **Continuous valued neural networks with two hidden layers are sufficient**. Technical Report, Department of Computer Science, Tufts University, 1988.

[Cybenko 1989] CYBENKO, G. Approximation by superpositions of a sigmoid function. **Mathematics of Control, Signals and Systems**, v.2, p.303-314, 1989.

[Data 2003] DATA Mining Softwares. Disponível em: http://www.andypryke.com/university/software.html. Acessado em: nov. 2003.

[Data 2004] DATA Mining and Knowledge Discovery. Disponível em: http://www.kdnuggets.com. Acessado em: fev. 2004.

[Deitel & Deitel 2001] DEITEL, H. M.; DEITEL, P. J. **Java Como Programar**. 3. ed. Porto Alegre: Bookman, 2001.

[Dietterich & Bakiri 1995] DIETTERICH, T. G.; BAKIRI, G. Solving multiclass learning problems via error-correcting output codes. **Journal of Artificial Intelligence Research**, v.2, p.263-286, 1995.

[Duda et al. 2000] DUDA, R. O.; HART, P. E.; STORK, D. G. **Patter Classification**. 2nd ed. Wiley-Interscience, 2000.

[Evsukoff & Almeida 2003] EVSUKOFF, A. G.; ALMEIDA, P. E. M. Sistemas Neuro Fuzzy. In: REZENDE, S. O. (Coord.) **Sistemas Inteligentes**: Fundamentos e Aplicações. Barueri: Manole, 2003, p.203-224.

[Fahlman 1988] FAHLMAN, S. E. Faster-learning variations on back-propagation: an empirical study. In: CONNECTIONIST MODELS SUMMER SCHOOL, 1998, Pittsburg. TOURETZKY, D.; HINTON, G.; SEJNOWSKI T. (Eds.) **Proceedings of the 1988 Connectionist Models Summer School**. Pittsburg: Morgan Kaufmann, 1998, p.38-51.

[Fahlman & Lebiere 1998] FAHLMAN, S. E.; LEBIERE, C. The Cascade-correlation learning architecture. In: LIPPMANN, E, R. P.; MOODY, J. E.; TOURETZKY, D. S. (Eds.) **Advances in Neural Information Processing Systems 2**. Morgan Kaufmann, 1988. p.524-532.

[Faifer & Janikow 1999] FAIFER, M.; JANIKOW, C. Extracting Fuzzy Symbolic Representation from Artificial Neural Networks. In: INT. CONF. OF THE NORTH AMERICAN FUZZY INFORMATION PROCESSING SOCIETY (NAFIPS), n.18, 1999, New York. **Proceedings of the 18th Int. Conf. of the North American Fuzzy Information Processing Society (NAFIPS)**, New York: 1999. p.600-604.

[Fayyad et al. 1996a] FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From Data Mining to Knowledge Discovery in Databases. **AI Magazine**, v.17, p.37-54, 1996a.

[Fayyad et al. 1996b] FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. The KDD process fo extracting useful knowledge from volumes of data. **Communications of the ACM**, v.39, n.11, p.27-34, 1996b.

[Fayyad et al. 2001] FAYYAD, U.; GRINSTEIN, G. G.; WIERSE, A. **Information Visualization in Data Mining and Knowledge Discovery**. Morgan Kaufmann, 2001.

[Freund & Schapire 1997] FREUND, Y.; SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and its application to boosting. **Journal of Computer and System Sciences**, v.55, n.1, p.119-139, 1997.

[Gately 1996] GATELY, E. **Neural Networks for Financial Forecasting**. J. Wiley, 1996.

[Goebel & Gruenwald 1999] GOEBEL, M.; GRUENWALD, L. A survey of data mining and knowledge discovery software tools. In: ACM SIGKDD INT. CONF. ON KNOWLEDGE DISCOVERY AND DATA MINING, n.5, 2002, Edmonton. **Proceedings of the Eighth ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining**. Edmonton: 1999. v.1, n.1, p.20-33.

[Goonatilake & Khebbal 1995] GOONATILAKE, S.; KHEBBAL, S. **Intelligent Hybrid Systems**. J. Wiley, 1995. 1v.

[Goulden 1956] GOULDEN, C. H. **Methods of Statistical Analysis**. 2nd ed. New York: J. Wiley, 1956.

[Hagan & Menhaj 1994] HAGAN, M.; MENHAJ, M. Training feedforward networks with the Marquardt algorithm. **IEEE Trans. on Neural Networks**, v.5, n.6, p.989-993, 1994.

[Han & Kamber 2001] HAN, J.; KAMBER, M. **Data Mining**: concepts and techniques. Morgan Kaufmann, 2001.

[Hayashi 1989] HAYASHI, Y. A neural expert systems using fuzzy teaching input. In: IEEE INT. CONF. ON FUZZY SYSTEMS, 1989, San Diego. **Proceedings of the IEEE Int. Conf. on Fuzzy Systems**. San Diego: 1989. p.485-491.

[Haykin 1999] HAYKIN, S. **Neural Networks**, A Comprehensive Foundation. 2nd ed. Prentice Hall, 1999.

[Hilderman & Hamilton 1999] HILDERMAN, R. J.; HAMILTON, H. J. **Knowledge discovery and interestingness measures**: A survey. Technical Report, Department of Computer Science, University of Regina, Saskatchewan, CS 99-04, oct. 1999.

[Horikawa et al. 1992] HORIKAWA, S.; FORUHASHI, T.; UCHIKAWA, Y. On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm. **IEEE Trans. on Neural Networks**, v.3, n.5, p.801-806, 1992.

[Huang & Xing 2002] HUANG, S. H.; XING, H. Extract intelligible and concise fuzzy rules from neural networks. **Fuzzy Sets and Systems**, v.132, p.233-243, 2002.

[ISEL 2004] INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA. Depto. de Engenharia da Electrónica e das Comunicações. Secção de Sistemas de Telecomunicações. **Geração de Variáveis Aleatórias**. Disponível em: http://www.deetc.isel.ipl.pt/sistemastele/MSST/arquivo/gervaral.pdf. 2003. Acessado em: fev. 2004.

[Jang 1993] JANG, J. R. ANFIS: Adaptive-Network-Based Fuzzy Inference System. **IEEE Trans. on Systems, Man, and Cybernetics**, v.23, n.3, p.665-685, 1993.

[Jang et al. 1997] JANG, J. R.; SUN, C.; MIZUTANI, E. **Neuro-Fuzzy and soft computing**: a computational approach to learning and machine intelligence. Prentice Hall, 1997.

[Jensen 1992] JENSEN, H. L. Using neural networks for credit scoring. **Managerial Finance**, v.18, p.15-26, 1992.

[Kasabov 1996] KASABOV, N. Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems. **Fuzzy Sets and Systems**, v.82, p.135-149, 1996.

[Kasabov et al. 1997] KASABOV, N. et al. FuNN/2 – A Fuzzy Neural Network Architecture for Adaptative Learning and Knowledge Acquisition. **Information Sciences - Applications**, v.101, n.3-4, p.155-175, 1997.

[kasabov et al. 2001] KASABOV, N. et al. Rule Extraction from Fuzzy Neural Networks FuNN: A Method and a Real-World Application. **Journal of Advanced Computational Intelligence**, v.5, n.4, p.193-200, 2001.

[Kimball et al. 1998] KIMBALL, R. et al. **The Data Warehouse**. J. Wiley, 1998.

[Kolmogorov 1957] KOLMOGOROV, A. On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition. **Doklady Akademiia Nauk SSR**, v.114, n.5, p.953-956, 1957.

[Kronszynski & Zhou 1998] KRONSZYNSKI, U.; ZHOU, J. **Fuzzy Clustering***:* Principles, Methods and Examples. IKS, 1998.

[Kuncheva 1992] KUNCHEVA, L. I. Fuzzy rough sets*:* Application to feature selection. **Fuzzy sets and Systems**, v.51, p.147-153, 1992.

[Kuo et al. 2002] KUO, R. J.; WU, P.; WANG, C. P. An intelligent sales forecasting system through integration of artificial neural networks and fuzzy neural networks with fuzzy weight elimination. **Neural Networks**, v.15, p.909–925, 2002.

[Lacerda et al. 2003] LACERDA, E. G.; CARVALHO, A. C. P. de L. F. de; LUDERMIR, T. B. Análise de Crédito Utilizando Rede Neurais Artificiais. In: Rezende, S. O. (Coord.) **Sistemas Inteligentes**: Fundamentos e Aplicações. Barueri: Manole, 2003, p.473-476.

(Lavraè 1999) LAVRAÈ, N. Selected techniques for data mining in medicine. **Artificial Intelligence in Medicine**, v.16, p.3-23, 1999.

[LeCun et al. 1990] LECUN, Y.; DENKER, J.; SOLLA, S. Optimal brain damage. In: TOURETZKY, D. (Ed.) **Advances in Neural Information Processing Systems 2**. San Mateo: Morgan Kaufmann, 1990. p.598-605.

[Li et al. 2002] LI, R.; MUKAIDONO, M.; TURKSEN, I. B. A fuzzy neural network for pattern classification and feature selection. **Fuzzy Sets and Systems**, v.130, p.101-108, 2002.

[Liu et al. 1998] LIU, B.; HSU, W.; MA, Y. Integrating Classification and Association Rule Mining. In: INT. CONF. ON KNOWLEDGE DISCOVERY AND DATA MINING, n.4, 1998, Menlo Park. **Proceedings of the Fourth Int. Conf. on Knowledge Discovery and Data Mining**. Menlo Park: AAAI Press, 1998. p.80-86.

[Lu et al. 1995] LU, H.; SETIONO, R.; LIU, H. Neurorule: A connectionist approach to data mining. In: INT. CONF. VERY LARGE DATA BASES, n.21, 1995, Zurich. **Proceedings of the 21st Int. Conf. Very Large Data Bases**. Zurich: 1995. p.478-489.

[Ludermir et al. 2003] LUDERMIR, T. B. et al. Sistemas Inteligentes Híbridos. In: REZENDE, S. O. (Coord.) **Sistemas Inteligentes**: Fundamentos e Aplicações. Barueri: Manole, 2003, p.249-268.

[Malhotra & Malhotra 2002] MALHOTRA, R.; MALHOTRA, D. K. Differentiating between good credits and bad credits using neuro-fuzzy system. **European Journal of Operational Research**, v.136, iss.1, p.190-211. 2002.

[Marsh et al. 1992] MARSH, S. et al. **Fuzzy Logic Education Program**. Center for Emerging Computer Technologies, Motorola, 1992.

[Masuoka et al. 1991] MASUOKA, R. et al. Neurofuzzy systems - fuzzy inference using a structure neural network. In: INT. CONF. ON FUZZY LOGIC AND NEURAL NETWORKS, 1991, Lisuka. **Proceedings of the Int. Conf. on Fuzzy Logic and Neural Networks**. Lisuka: 1991. p.173-177.

[McCulloch & Pitts 1943] MCCULLOCH, W.; PITTS, W. A logical calculus of the ideas immanentin nervous activity. **Bulletin of Mathematical Biophysics**, v.5, p.115-133, 1943.

[McMillan et al. 1991] MCMILLAN, C.; MOZER, M. C.; SOMLENSKY, P. The Connectionist Scientist Game: Rule Extraction and Refinement in a Neural Network. In: ANN. CONF. CONGNITVE SCIENCE SOC., n.13, 1991, Hillsdale. **Proceedings of 13th Ann. Conf. Cognitive Science Soc.** Hillsdale: Lawrence Erbaum, 1991. p.424-430.

[Medeiros 1996] MEDEIROS, A. G. M. **Modelos Neuro-Difusos**: Um Enfoque para Integração de Redes Neurais Artificiais e Sistemas Difusos. 1996. Dissertação (Mestrado em Ciência da Computação) - Programa de Pós-Graduação, Centro de Informática, Universidade Federal de Pernambuco, Recife, 1996.

[Mendes Filho et al. 1997] MENDES FILHO, E.; CARVALHO, A.; MATIAS, A. Credit assessment using evolutionary MLP networks. In: INT. CONF. COMPUTATIONAL FINANCE, n.5, 1997, London. REFENES, P. (Ed.) **Proceedings of the V Int. Conf. Computational Finance**. London: Kluwer Academic Publishers, 1997. p.365-371.

[Michie et al. 1994] MICHIE, D.; SPIEGELHALTER, D.; TAYLOR, C. **Machine Learning, Neural and Statistical Classification**. The StatLog Project, 1994.

[Milaré & Carvalho 2001] MILARÉ, C. R.; CARVALHO, A. C. P. L. F. Um Estudo Comparativo de Extração de Conhecimento Simbólico de Redes Neurais. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO (SBC'2001), n.21, 2001, Forteleza. **Anais do III Encontro Nacional de Inteligência Artificial (ENIA 2001)**. Fortaleza: 2001. 1 CD-ROM. 10p.

[Mitchell 1997] MITCHELL, T. M. **Machine Learning**. McGraw-Hill, 1997.

[Mitra et al. 1994] MITRAL, S.; DE, R. K.; PAL, S. K. Fuzzy multi-layer perceptron, inferencing and rule generation. **IEEE Trans. on Neural Networks**, v.8, n.6, p.1338-1350, 1997.

[Monard & Baranauskas 2003] MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre Aprendizagem de Máquina. In: REZENDE, S. O. (Coord.) **Sistemas Inteligentes**: Fundamentos e Aplicações. Barueri: Manole, 2003, p.84-114.

[Monteiro 1999] MONTEIRO, D. S. M. P. **Discovery** – Um Ambiente para Descoberta de Conhecimento e Mineração de Dados. 1999. Dissertação (Mestrado em Ciência da Computação) - Programa de Pós-Graduação em Ciência da Computação, Centro de Informática, Universidade Federal de Pernambuco, Recife, 1999.

[Mozer & Smolensky 1989] MOZER, M. C.; SMOLENSKY, P. Skeletonization*:* A technique for trimming the fat from a network via relabance assessment. In: TOURETZKY, D. S. (Ed.) **Advances in Neural Information Processing Systems**. San Mateo: Morgan Kaufmann, 1989. v.1, p.107-115.

[Nauck 1994] NAUCK, D. A fuzzy perceptron as a generic model for neuro-fuzzy approaches. In: GERMAN GI-WORKSHOP FUZZY-SYSTEME'94, n.2, 1994, Munich. **Proceedings of the 2nd German GI-Workshop Fuzzy-Systeme'94**, Munich: 1994.

[Nauck 1997] NAUCK, D. Neuro-Fuzzy Systems*:* Review and Prospects. In: EUROPEAN CONGRESS ON INTELLIGENT TECHNIQUES AND SOFT COMPUTING (EUFIT´97), n.5, 1997, Aachen. **Proceedings of the Fifth European Congress on Intelligent Techniques and Soft Computing (EUFIT´97)**. Aachen: 1997. p.1044-1053.

[Nauck & kruse 1993] NAUCK, D.; KRUSE, R. A fuzzy neural network learning fuzzy control rules and membership functions by fuzzy error backpropagation. In: IEEE INT. CONF. ON NEURAL NETWORKS, 1993, San Francisco. **Proceedings of the IEEE Int. Conf. on Neural Networks**. San Francisco: 1993. p.1022-1027.

[Nauck & Kruse 1995] NAUCK, D.; KRUSE, R. NEFCLASS – A neuro-fuzzy approach for the classification of data. In: ACM Symposium on Applied Computing, 1995, Nashville. GEORGE, K.M. et al. (Eds.) **Proceedings of the ACM Symposium on Applied Computing**. New York: ACM Press, 1995. p.461-465.

[Nauck & Kruse 1999] NAUCK, D.; KRUSE, R. Neuro-fuzzy systems for function approximation. **Fuzzy Sets and Systems**, v.101, n.2, p.261-271, jan. 1999.

[Nobre et al. 1998] NOBRE, C. N. et al. Extração de conhecimento: uma comparação entre os métodos clássico e conexionista. In: SIMPÓSIO BRASILEIRO DE REDES NEURAIS, 1998, Belo Horizonte. **Anais do Simpósio Brasileiro de Redes Neurais**. Belo Horizonte: 1998. v.2, p.126-131.

[Parma et al. 1998] PARMA, G.; MENEZES, B. R.; BRAGA, A. P. Sliding mode algorithm for training multilayer neural networks. **IEEE Electronics Letters**, v.34, n.1, p.97-98, 1998.

[Pau & Gotzche 1992] PAU, L.; GOTZCHE, T. Explanation facility for neural networks. **Journal of Intelligent and Robotic System**, v.5, p.193-206, 1992.

[Phansalkar & Sastry 1994] PHANSALKAR, V. V.; SASTRY, P. S. Analysis of the back-propagation algorithm with momentum. **IEEE Trans. on Neural Networks**, v.5, n.3, p.505-506, 1994.

[Piramuthu 1998] PIRAMUTHU, S. Evaluating feature selection methods for learning in data mining applications. In: HAWAII INT. CONF. ON SYSTEMS SCIENCE, n.31, 1998, Hawaii. **Proceedings of the 31st Hawaii Int. Conf. on Systems Science**. Hawaii: IEEE Computer Society, 1998. v.5, p.294-301.

[Piramuthu 1999] PIRAMUTHU, S. Financial credit-risk evaluation with neural and neurofuzzy systems. **European Journal of Operational Research**, v.112, p.310-321, 1999.

[Pop et al. 1994] POP, E.; HAYWARD, R.; DIEDERICH, J. RULENEG: extracting rules from a trained ann by stepwise negation. In: CONF. OF THE AUSTRALASIAN COGNITIVE SCIENCE SOCIETY, n.3, 1994, Brisbane. **Proceedings of the Third Conf. of the Australasian Cognitive Science Society**. Brisbane: University of Queensland, 1994. p. 62.

[Prechelt 1994] PRECHELT, L. **Proben1** – A set of benchmarks and benchmarking rules for neural network training algorithms. Relatório Técnico 21/94, Faculdade de Informática, Universidade de Karlsruhe, Alemanha, 1994.

[Provost & Fawcett 1997] PROVOST, F.; FAWCETT, T. F. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In: INT. CONF. ON KNOWLEDGE DISCOVERY AND DATA MINING (KDD-97), n.3, 1997, Menlo Park. **Proceedings of the Third Int. Conf. on Knowledge Discovery and Data Mining (KDD-97)**. Menlo Park: AAAI Press, 1997. p.43-48.

[Quek & Zhou 2001] QUEK, C.; ZHOU, R. W. The POP learning algorithms: reducing work in identifying fuzzy rules. **Neural Networks**, v.14, p.1431-1445, 2001.

[Quinlan 1993] QUINLAN, J. R. **C4.5 Programs for machine learning**. Morgan Kaufmann, 1993.

[Ramos 2001] RAMOS, P. G. **Rulexpert**: Uma ferramenta para descoberta de regras em bases de dados. 2001. Dissertação (Mestrado em Ciência da Computação) - Programa de Pós-Graduação em Ciência da Computação, Centro de Informática, Universidade Federal de Pernambuco, Recife, 2001.

[Rezende et al. 1998] REZENDE, S. O. et al. Visualization for knowledge discovery in database. In: EBECKEN, N. F. F. (Ed.) **Data Mining**. England: WIT, 1998. p.81-95.

[Rezende et al. 2003] REZENDE, S. O. et al. Mineração de Dados. In: REZENDE, S. O. (Coord.) **Sistemas Inteligentes**: Fundamentos e Aplicações. Barueri: Manole, 2003. p.307-335.

[Riedmiller & Braun 1993] RIEDMILLER, M.; BRAUN, H. A direct adaptive method for faster backpropagation learning*:* The RPROP algorithm. In: IEEE INT. CONF. ON NEURAL NETWORKS, 1993, San Francisco. **Proceedings of the IEEE Int. Conf. on Neural Networks**, San Francisco: 1993. p.586-591.

[Rosenblatt 1962] ROSENBLATT, F. **Principles of Neurodinamics**: Perceptrons and the theory of brain mechanisms. Washington DC: Spartan, 1962.

[Rumelhart et al. 1986] RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning Representations by Backpropagation Errors. **Nature**, v.323, p.533-536, 1986.

[Rumelhart & McClelland 1986] RUMELHART, D. E.; MCCLELLAND, J. L. **Parallel Distributed Processing**. Cambridge: MIT Press, 1986. 1v. p.318-362.

[Salinas 1998] SALINAS, D. T. P. **Bootstrap não paramétrico aplicado a dados incompletos**. 1998. Dissertação (Mestrado), Inst. de Matemática e Estatística, Univ. de São Paulo, São Paulo, 1998.

[Sandri & Correa 1999] SANDRI, S.; CORREA, C. Lógica Nebulosa. In: Escola de Redes Neurais, n.5, 1999, São José dos Campos. **Anais da V Escola de Redes Neurais**. São José dos Campos: 1999. p.73-90.

[Schellharmmer et al. 1997] SCHELLHARMMER, I. et al. **Knowledge Extraction and Recurrent Neural Networks:** an Analysis of an Elman Network Trained on a Natural Language Learning Task. Technical Report 97-IS1, Queensland University of Technology, Australia, 1997.

[Schrickel 1995] SCHRICKEL, W. **Análise de Crédito Concessão e Gerência de Empréstimos**. Atlas, 1995.

[Sestito & Dillon 1995] SESTITO, S.; DILLON, T. Automated Knowledge Acquisition of Rules with Continuously Valued Attributes. In: INT. CONF. EXPERT SYSTEMS AND THEIR APPLICATIONS (AVIGNON), n.12, 1995. **Proceedings of the 12th Int. Conf. Expert Systems and Their Applications (AVIGNON)**. 1995. p.645-656.

[Shapiro 2002] SHAPIRO, A. F. The merging of neural networks, fuzzy logic, and genetic algorithms. **Insurance: Mathematics and Economics**, v.31, p.115-131, 2002.

[Silva et al. 2002] SILVA, I. G. L. et al. Integration of Data Mining and Hybrid Expert System. In: INT. FLORIDA ARTIFICIAL INTELLIGENCE RESEARCH SOCIETY CONF. (FLAIRS), n.15, 2002, Florida. **Proceedings of the Fifteenth Int. Florida Artificial Intelligence Research Society Conf. (FLAIRS)**. Florida, 2002. p.267-271.

[Silverman 1986] SILVERMAN, B. W. **Density Estimation for Statistics and Data Analysis**. New York: Chapman and Hall, 1986.

[Sousa & Carvalho 1999] SOUSA, H.; CARVALHO, A. Credit assessment using constructive neural networks. In: INT. CONF. ON COMPUTATIONAL INTELIGENCE AND MULTIMIDIA APPLICATIONS, n.3, 1999, India. **Proceedings of the 3rd Int. Conf. on Computational Intelligence and Multimedia Applications**. India: IEEE, 1999. p.40-44.

[Srikant et al. 1997] SRIKANT, R.; VU, Q.; AGRAWAL, R. Mining Association Rules with Item Constraints. In: INT. CONF. KNOWLEDGE DISCOVERY AND DATA MINING, n.3, aug. 1997, California. **Proceedings of the 3rd Int. Conf. Knowledge Discovery and Data Mining**. California: aug. 1997. p.67-73.

[Sun & Jang 1992] SUN, C.; JANG, J. R. Adaptative network based fuzzy classification. In: JAPAN-USA SYMPOSION ON FLEXIBLE AUTOMATION, 1992. **Proceedings of the Japan-USA Symposium on Flexible Automation**. 1992.

[Taha & Ghosh 1999] TAHA, I. A.; GHOSH, J. Symbolic Interpretation of Artificial Neural Networks. **IEEE Trans. on Knowledge and Data Engineering**, v.11, n.3, p.448-461, 1999.

[Tanscheit 2003] TANSCHEIT, R. **Lógica Fuzzy, Raciocínio Aproximado e Mecanismos de Inferência**. Pontifícia Universidade Católica do Rio de Janeiro, 2003.

[Tickle et al. 1996] TICKLE, A. B.; ORLOWSKI, M.; DIEDERICH, J. DEDEC: A Methodology for Extracting Rules from Trained Artificial Neural Networks. In: RULE EXTRACTION FROM TRAINED ARTIFICIAL NEURAL NETWORKS WORKSHOP, 1996, Queensland Univ. of Technology. Andrews, R.; Diederich, M. (Eds.) **Proceedings of the Rule Extraction from Trained Artificial Neural Networks Workshop**. Queensland Univ. of Technology: 1996. p.90-102.

[Towell & Shavlik 1993] TOWELL, G. G.; SHAVLIK, J. W. The Extraction of Refined Rules from Knowledge-Based Networks. **Machine Learning**, v.13, n.1, p.71-101, 1993.

[Towell & Shavlik 1994] TOWELL, G. G.; SHAVLIK, J. W. Knowledge-based artificial neural networks. **Artificial Intelligence**, v.70, n.4, p.119-166, 1994.

[Turban & Aronson 2000] TURBAN, E.; ARONSON, J. E. **Decision Support Systems and Intelligent Systems**. 6th ed. Pearson Education, 2000.

[Two Crows 1999] TWO CROWS CORPORATION. **Introduction to Data Mining and Knowledge Discovery**. 3rd ed. 1999.

[Vasconcelos et al. 1999] VASCONCELOS, G. C.; AEDODATO, P. J.; MONTEIRO, D. S. M. A neural Network based solution for the credit risk assessment problem. In: CONGRESSO BRASILEIRO DE REDES NEURAIS, n.4, 1999, São José dos Campos. **Anais do Congresso Brasileiro de Redes Neurais**. São José dos Campos: 1999. p.269-274.

[Weiss & Indurkhya 1998] WEISS, S. M.; INDURKHYA, N. **Predictive Data Mining**: A Practical Guide. San Francisco: Morgan Kaufmann, 1998.

[Widrow et al. 1994] WIDROW, B.; RUMELHART, D. E.; LEHR, M. A. Neural networks: Applications in industry, business and science. **Communications of the ACM**, v.37, n.3, p.93-105, 1994.

[Witten & Frank 2000] WITTEN, I. H.; FRANK, E. **Data Mining**: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, 2000.

[Zadeh 1965] ZADEH, L. Fuzzy Sets. **Information Control**, v.8, p.338-353, 1965.

[Zimmermann 1991] ZIMMERMANN, H. J. **Fuzzy Set Theory** - and Its Applications. Massachusetts: Kluwer Academic Publishers, 1991.

Dissertação de Mestrado apresentada por **Bruno Pereira de Amorim** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título **"Desenvolvimento de uma Plataforma Híbrida para Descoberta de Conhecimento em Bases de Dados"**, orientada pelo Prof. Germano Crispim Vasconcelos e aprovada pela Banca Examinadora formada pelos professores:

Paulo Jorge Leitão Adeodato
Centro de Informática / UFPE

Edson Nascimento
Departamento de Engenharia de Eletricidade / UFMA

Germano Crispim Vasconcelos
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 29 de março de 2004.

Prof. JAELSON FREIRE BRELAZ DE CASTRO
Coordenador da Pós-Graduação em Ciência da Computação
Centro de Informática da Universidade Federal de Pernambuco